Johannes Fürnkranz
Tobias Scheffer
Myra Spiliopoulou (Eds.)

LNAI 4212

Machine Learning: ECML 2006

17th European Conference on Machine Learning
Berlin, Germany, September 2006
Proceedings

Springer

Johannes Fürnkranz Tobias Scheffer
Myra Spiliopoulou (Eds.)

Machine Learning: ECML 2006

17th European Conference on Machine Learning
Berlin, Germany, September 18-22, 2006
Proceedings

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Johannes Fürnkranz
Technische Universität Darmstadt
Fachbereich Informatik
Hochschulstraße 10, 64289 Darmstadt, Germany
E-mail: juffi@ke.informatik.tu-darmstadt.de

Tobias Scheffer
Humboldt-Universität zu Berlin
Institut für Informatik
Unter den Linden 6, 10099 Berlin, Germany
E-mail: scheffer@informatik.hu-berlin.de

Myra Spiliopoulou
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Universitätsplatz 2, 39016 Magdeburg, Germany
E-mail: myra@iti.cs.uni-magdeburg.de

Library of Congress Control Number: 2006932512

CR Subject Classification (1998): I.2, F.2.2, F.4.1, H.2.8

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-45375-X Springer Berlin Heidelberg New York
ISBN-13 978-3-540-45375-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11871842 06/3142 5 4 3 2 1 0

Preface

The two premier annual European conferences in the areas of Machine Learning and Data Mining have been collocated ever since the joint conference in Freiburg, Germany, 2001. The European Conference on Machine Learning was established 20 years ago, when the first European Working Session on Learning was held in Orsay, France, in 1986. The conference is growing, and is more lively than ever. The European Conference on Principles and Practice of Knowledge Discovery in Databases celebrated its tenth anniversary; the first PKDD took place in 1997 in Trondheim, Norway. Over the years, the ECML/PKDD series has evolved into one of the largest and most selective international conferences in these areas, the only one that provides a common forum for the two closely related fields. In 2006, the 6th collocated ECML/PKDD took place during September 18-22, when the Humboldt-Universität zu Berlin hosted the 17th European Conference on Machine Learning (ECML) and the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD).

The successful model of a hierarchical reviewing process that was introduced last year for the ECML/PKDD 2005 in Porto was taken over in 2006. We have nominated 32 Area Chairs, each of them responsible for several closely related research topics. Suitable areas were selected on the basis of the submission statistics for ECML/PKDD 2005 to ensure a proper load balance among the area chairs. For the first time, a joint Program Committee was nominated for the two conferences, consisting of 280 renowned researchers, mostly proposed by the Area Chairs. This joint PC, the largest of the series to date, allowed us to exploit synergies and deal competently with topic overlaps between ECML and PKDD.

ECML/PKDD 2006 received 564 full paper submissions that entered the reviewing process. The submissions were manually assigned to the Area Chairs, who coordinated the reviewers thereafter. Reviewer assignment was based on bidding with CyberChairPRO, as in the previous years. With very few exceptions, every submission was reviewed by three PC members. Based on these reviews, on feedback from the authors, and on discussions among the reviewers, the Area Chairs provided a recommendation for each paper. Continuing the tradition of previous events in the series, we accepted full papers for oral presentation and short papers for poster presentation. The final decision was made by us based on the recommendations of the area chairs. We selected 46 full papers and 36 short papers for ECML, and 36 full papers and 26 short papers for PKDD. The acceptance rate for full papers is 14.5% and the overall acceptance rate is 25.5%, in accordance with the high-quality standards of the conference series. Next to the paper and poster sessions, ECML/PKDD 2006 also featured five invited talks, ten workshops, seven tutorials and the ECML/PKDD discovery challenge.

We distinguished eight outstanding contributions; the awards were generously sponsored by the *Machine Learning Journal* and the *KD-Ubiq network.*

ECML Best Paper: Quoc Le, Alex Smola, Thomas Gärtner, Yasemin Altun: *Transductive Gaussian Process Regression with Automatic Model Selection.*

PKDD Best Paper: Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, Heikki Mannila: *The Discrete Basis Problem.*

ECML Best Student Paper: Bernd Gutmann and Kristian Kersting: *TildeCRF. Conditional Random Fields for Logical Sequences.*

PKDD Best Student Paper: Arik Friedmann, Assaf Schuster, Ran Wolff: *k-Anonymous Decision Tree Induction.*

ECML Innovative Contribution: Alexander Clark, Christophe Costa Florencio, Chris Watkins: *Languages as Hyperplanes: Grammatical Inference with String Kernels.*

PKDD Innovative Application: Herna Viktor, Eric Paquet, Hongyu Guo: *Measuring to Fit: Virtual Tailoring Through Cluster Analysis and Classification.*

The *ECML/PKDD Best Presentation* and the *ECML/PKDD Best Poster Presentation* awards were elected by participants of the conference.

This year's *Discovery Challenge* focused on personalized spam filtering and generalization across related learning tasks. Steffen Bickel organized the challenge; 26 teams participated. For task A, three teams achieved a first rank: Khurram Nazir Junejo, Mirza Muhammad Yousaf, and Asim Karim; Bernhard Pfahringer; and Kushagra Gupta, Vikrant Chaudhary, Nikhil Marwah, and Chirag Taneja. Task B was won by Gordon Cormack. The solution of Bernhard Pfahringer was distinguished with the Creativity Award.

We are indebted to the Area Chairs, Program Committee members and external reviewers for their effort and engagement in making a rich but selective scientific program for ECML/PKDD. Special thanks go to those reviewers who helped with additional reviews at very short notice to assist us at difficult decisions. We further thank our two workshop and tutorial chairs Tapio Elomaa and Bart Goethals for selecting and coordinating the ten workshop and seven tutorial events that accompany the conference; the workshop organizers, tutorial presenters, and the organizers of the discovery challenge; Richard van de Stadt and CyberChairPRO for competent and flexible support; the local Organizing Committee; and all other people that contributed to the organization of this event. Finally, we are grateful to the the Steering Committee and the ECML/PKDD community that entrusted us with the organization of ECML/PKDD 2006.

Most of all, however, we would like to thank all the authors who honored us by submitting their work to this conference, thereby facilitating the success of this event.

Berlin, September 2006

Johannes Fürnkranz
Tobias Scheffer
Myra Spiliopoulou

Sponsors

We wish to express our gratitude to our sponsors for their great contributions to the conference. We wish to thank Google for featuring the Google ECML Poster Reception and providing ten Student Travel Awards; the Humboldt-Universität zu Berlin for providing the conference venue; the German Science Foundation DFG for supporting all invited speakers; KD-Ubiq for supporting the PKDD Poster Reception and European Projects Poster Reception, four Student Travel Awards, and the Best Paper Awards; the European Office of Aerospace Research and Development (EOARD), Air Force Office of Scientific Research, United States Air Force Research Laboratory for generous financial support; Strato AG for providing the awards to the winners of the Discovery Challenge; the Pascal Network of Excellence and IBM for financial support; the *Machine Learning Journal* for supporting the Student Best Paper Awards.

AFOSR/EOARD support is not intended to express or imply endorsement by the U.S. Federal Government.

Organization

Program Chairs

Johannes Fürnkranz (Technische Universität Darmstadt)
Tobias Scheffer (Humboldt-Universität zu Berlin)
Myra Spiliopoulou (Otto-von-Guericke-Universität Magdeburg)

Local Chairs

Andreas Nürnberger (Otto-von-Guericke-Universität Magdeburg)
Tobias Scheffer (Humboldt-Universität zu Berlin)

Workshop and Tutorial Chairs

Tapio Elomaa (Tampere University of Technology)
Bart Goethals (University of Antwerp)

Discovery Challenge Chair

Steffen Bickel (Humboldt-Universität zu Berlin)

Poster and Fashion Designer, Webmaster

Michael Brückner (Humboldt-Universität zu Berlin)

Local Arrangements Team

Steffen Bickel	Ulf Brefeld
Michael Brückner	Isabel Drost
Thoralf Klein	

Steering Committee

Hendrik Blockeel	Jean-François Boulicaut
Pavel Brazdil	Rui Camacho
Floriana Esposito	João Gama
Dragan Gamberger	Fosca Gianotti
Alípio Jorge	Nada Lavrač
Dino Pedreschi	Ljupčo Todorovski
Luís Torgo	

Area Chairs

Hendrik Blockeel
Henrik Boström
Soumen Chakrabarti
Olivier Chapelle
Pádraig Cunningham
James Cussens
Kurt Driessens
Ronen Feldman
Peter Flach
Eibe Frank
Thomas Gärtner
João Gama
Bart Goethals
Thomas Hofmann
George Karypis
Kristian Kersting
Stefan Kramer
Stan Matwin
Katharina Morik
Klaus Obermayer
Bernhard Pfahringer
Luc De Raedt
Carl Rasmussen
Lorenza Saitta
Dale Schuurmans
Jaideep Srivastava
Csaba Szepesvári
Hannu Toivonen
Luís Torgo
Volker Tresp
Vishy Vishwanathan
Mohammed Zaki

Program Committee

James Abello
Douglas Aberdeen
Gediminas Adomavicius
Charu Aggarwal
Eugene Agichtein
Jesus S. Aguilar-Ruiz
Erick Alphonse
Yasemin Altun
Massih Amini
Aijun An
Josep-Lluis Arcos
Yonatan Aumann
Paolo Avesani
Antonio Bahamonde
David Barber
Roberto Bayardo
Bettina Berendt
Michael R. Berthold
Elisa Bertino
Fernando Berzal
Steffen Bickel
Francesco Bonchi
Gianluca Bontempi
Remco Bouckaert
Jean-François Boulicaut
Thorsten Brants
Mikio Braun
Pavel Brazdil
Ulf Brefeld
Derek Bridge
Björn Bringmann
Klaus Brinker
Wray Buntine
Christian Böhm
Tiberio Caetano
Toon Calders
Rui Camacho
Barbara Catania
Nicolò Cesa-Bianchi
Deepayan Chakrabarti
Sharma Chakravarthy
LiWu Chang
David Cheung
Amanda Clare
Ian Cloete
Antoine Cornuéjols
Koby Crammer
Susan Craw

Daniel Cremers
Bruno Cremilleux
Walter Daelemans
Ernst Damien
Hal Daume III
Ian Davidson
Luc Dehaspe
Olivier Delalleau
Janez Demšar
François Denis
Chris Ding
Chabane Djeraba
Sašo Džeroski
Tina Eliassi-Rad
Tapio Elomaa
Floriana Esposito
Roberto Esposito
Martin Ester
Wei Fan
Ad Feelders
Arthur Flexer
George Forman
Matthias Franz
Paolo Frasconi
Glenn Fung
Linda C. van der Gaag
Dragan Gamberger
Jean-Gabriel Ganascia
Minos Garofalakis
Floris Geerts
Pierre Geurts
Ali Ghodsi
Joydeep Ghosh
Fosca Giannotti
Aristides Gionis
Attilio Giordana
Christophe Giraud-Carrier
Mark Girolami
David Gondek
Gunter Grieser
Marko Grobelnik
Valerie Guralnik
Mark Hall
Howard Hamilton
Eui-Hong Han
Markus Hegland
Jose Hernandez-Orallo
Gerhard Heyer
Colin de la Higuera
Melanie Hilario
Alexander Hinneburg
Susanne Hoche
Achim Hoffmann
Jaakko Hollmén
John H. Holmes
Geoffrey Holmes
Tamas Horvath
Andreas Hotho
San-Yih Hwang
Frank Höppner
Eyke Hüllermeier
Nitin Indurkhya
Iñaki Inza
Manfred Jaeger
Szymon Jaroszewicz
Edwin de Jong
Alípio Jorge
Alexandros Kalousis
Jaz Kandola
Hillol Kargupta
Andreas Karwath
Samuel Kaski
Motoaki Kawanabe
Eamonn Keogh
Latifur Khan
Ross King
Mika Klemettinen
Ralf Klinkenberg
Arno Knobbe
Jens Kohlmorgen
Igor Kononenko
Adam Kowalczyk
Miroslav Kubat
Nicolas Lachiche
Michail Lagoudakis
Pedro Larrañaga
Pavel Laskov
Mark Last
Dominique Laurent
Nada Lavrač

Alessandro Sperduti
Eduardo Spinosa
Nicolas Spyratos
Michael Steinbach
Jan Struyf
Gerd Stumme
Einoshin Suzuki
Vojtech Svatek
Marcin Szczuka
Prasad Tadepalli
Pang-Ning Tan
Tao Tao
Takao Terano
Henry Tirri
Ljupco Todorovski
Ioannis Tsochantaridis
Karl Tuyls
Michalis Vazirgiannis
Katja Verbeeck
Jean-Philippe Vert
Rene Vidal
Jianyong Wang
Wei Wang
Ke Wang
Jason Wang
Haixun Wang
Takashi Washio
Louis Wehenkel
Sholom Weiss
Gerhard Widmer
Marco Wiering
Eric Wiewiora
Ole Winther
Nirmalie Wiratunga
Stefan Wrobel
Hui Xiong
Jiong Yang
Ying Yang
Philip Yu
Kai Yu
Bianca Zadrozny
Gerson Zaverucha
Filip Zelezny
Thomas Zeugmann
ChengXiang Zhai
Byoung-Tak Zhang
Ying Zhao
Dengyong Zhou
Xiaojin Zhu
Alexander Zien
Jean-Daniel Zucker
Blaž Zupan

Additional Reviewers

Sreangsu Acharyya
Elke Achtert
Raman Adaikkalavan
Fabio Aiolli
Florence d'Alché-Buc
Claudia d'Amato
Bill Andreopoulos
Annalisa Appice
Eva Armengol
Stella Asiimwe
Anneleen Van Assche
Maurizio Atzori
Korinna Bade
Gökhan H. Bakır
Pierpaolo Basile
Matthew Beal
Gurkan Bebek
Aditya Belapukar
Margherita Berardi
Jürgen Beringer
Marc Bernard
Kanishka Bhaduri
Marenglen Biba
Hanczar Blaise
Gilles Blanchard
Yann-ael Le Borgne
Karsten Borgwardt
Marco Botta
Dominique Bouthinon
Janez Brank
Michael Brückner
Robert D. Burbidge

Emma Byrne
Nicolas Cebron
Michelangelo Ceci
Eugenio Cesario
Hasan E. Cetingul
Sutanu Chakraborti
Hari Chelladurai
Min Chen
Haibin Cheng
Hong Cheng
Li Cheng
Yun Chi
Silvia Chiappa
Thiago Cordeiro
Gianni Costa
Marco Cristo
Tom Croonenborghs
Vassilis Cutsuridis
Wisam Dakka
Souptik Datta
Sandra De Amo
Fabien De Marchi
Marco Degemmis
Evangelos Dellis
Nicola Di Mauro
Isabel Drost
Norberto Díaz Díaz
Lauri Eronen
Vicent Estruch
Nicola Fanizzi
Francisco J. Ferrer-Troyano
Stefano Ferilli
Cèsar Ferri
Frédéric Flouvat
Francesco Folino
Nuno Fonseca
Blaz Fortuna
Murat Ganiz
Jing Gao
Dimitrios Gavrilis
Peter Vincent Gehler
Liqiang Geng
Pierre Gerard
Atiyeh Ghoreyshi
Amol Ghoting
Arnaud Giacometti
Chris Giannella
Aris Gkoulalas-Divanis
Robby Goetschalckx
Alvina Goh
Ricardo Grau
Steve Greenwald
Gunjan Gupta
Simon Günter
Benjamin Haibe-Kains
Maria Halkidi
Hongxing He
Matthias Hein
Christoph Heinz
Corneliu Henegar
Katherine G. Herbert
Kirsten Hildrum
Petteri Hintsanen
Wai-Shing Ho
Meng Hu
Ming Hua
Jiayuan Huang
Yi Huang
Alex Ihler
Jaikrishna
Christopher D. Janneck
Robert Jäschke
Tao-Yuan Jen
Antonio Jimeno
Frederic Jurie
Dmitry Kamenetsky
Murat Kantarcioglu
Ioannis Karydis
Balakumar Kendai
Wolf Kienzle
David H. Kim
Seyoung Kim
Jee-Hyub Kim
Kwang In Kim
Jiří Kléma
Ioannis Klafaptis
Christine Körner
Ville Kononen
Stasinos Konstantopoulos
Despina Kontos

Arne Koopman
Petra Kralj
Chase Krumpelman
Jussi Kujala
Matjaž Kukar
Peer Kröger
Peter Kunath
Niels Landwehr
Maggie Man Ki Lau
Anne Laurent
Quoc V. Le
Steven Lemm
Shenzhi Li
Ting Li
Oriana Licchelli
Marina Litvak
Alexander Liu
Kun Liu
Yang Liu
George Loizou
Pasquale Lops
Alicia Troncoso Lora
Marc Q. Ma
Patrick Marcel
Keith Marsolo
Eric Martin
Elio Masciari
Stewart Massie
Giuseppe M. Mazzeo
Oscar Meruvia Pastor
Kapila Moonesinghe
Kawanabe Motoaki
Fernando Mourão
Rahman Mukras
Mirco Nanni
Jan Nemrava
Isabel A. Nepomuceno-Chamorro
David Newman
Radu Stefan Niculescu
Blaž Novak
Cheng Soon Ong
Amandine Orecchioni
Matthew Eric Otey
Aritz Pérez
Ignazio Palmisano
Elias Pampalk
Maarten Peeters
Jaakko Peltonen
Michael Perrow
Sergios Petridis
Viet Phan-Luong
Tadek Pietraszek
Aloisio Carlos de Pina
Conrad Plake
Claudia Plant
Mannes Poel
Subhesh Pradhan
Pavel Praks
Kai Puolamäki
Gunnar Rätsch
M. Jose Ramirez-Quintana
S. S. Ravi
Avinash Ravichandran
Simon Rawles
Matthias Renz
Chiara Renso
Konrad Rieck
Stefan Riezler
Carsten Riggelsen
Salvatore Rinzivillo
Pedro Rodrigues
Domingo S. Rodríguez-Baena
Thierson Rosa
Michel de Rougemont
Vishesh Sahu
Luka Šajn
Sagar Savla
Leander Schietgat
Christoph Schmitz
Timon Schroeter
Matthias Schubert
Jerry Scripps
Tomi Silander
Milan Simunek
Dheeraj Singaraju
Ksenia Shalonova
Kazumi Slott
Diego Sona
Bin Song
Le Song

Xiaodan Song
Soeren Sonnenburg
Arnaud Soulet
Junilda Spirollari
Harald Steck
Sebastian Stober
Petr Strossa
Jimeng Sun
Peter Sunehag
Aditya Telang
Choon Hui Teo
Ivan Titov
Igor Trajkovski
George Tsatsaronis
Mihalis Tsoukalos
Miroslav Vacura
Hamed Valizadegan
Antonio Varlaro
Sriharsha Veeramachaneni
Arvind Venkatachalam
Jarkko Venna
Jakob Verbeek
Vassilis Verykios
Akrivi Vlachou
Peter Vrancx
Christian James Walder
Qian Wan
Chao Wang
Lei Wang
Qiang Wang
Xuanhui Wang
Li Wei
Joost van de Weijer
Dietrich Wettschereck
Adam Woznica
Junjie Wu
Mingrui Wu
Xiaopeng Xi
Keisuke Yamazaki
Hui Yang
Xiaoning Yang
Dragomir Yankov
Jin Yu
Shipeng Yu
Monika Zakova
Shijie Zhang
Xinhua Zhang
Bin Zhou
Xingquan Zhu
Arthur Zimek

Table of Contents

Short Papers

On Temporal Evolution in Data Streams

Charu C. Aggarwal

IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA
charu@us.ibm.com

Abstract. In recent years, the progress in hardware technology has made it possible for organizations to store and record large streams of transactional data. This results in databases which grow without limit at a rapid rate. This data can often show important changes in trends over time. In such cases, it is useful to understand, visualize, and diagnose the evolution of these trends. In this talk, we discuss a method to diagnose the changes in the underlying data stream and other related methods for change detection in streams. We also discuss the problem of data stream evolution in the context of mining algorithms such as clustering and classification. In many cases, mining algorithms may not function as effectively because of the change in the underlying data. We discuss the effects of evolution on mining and synopsis construction algorithms and a number of opportunities which may be available for further research on the topic.

The Future of CiteSeer: CiteSeer^x

C. Lee Giles

David Reese Professor, School of Information Sciences and Technology
Professor, Computer Science and Engineering
Professor, Supply Chain and Information Systems
The Pennsylvania State University
University Park, PA, USA
giles@ist.psu.edu

Abstract. CiteSeer, a public online computer and information science search engine and digital library, was introduced in 1997 and was a radical departure from the traditional methods of academic and scientific document access and analysis. Computer and information scientists quickly became used to and expected immediate access to their literature and CiteSeer provided a popular partial solution. CiteSeer was based on these features: actively acquiring new documents, automatic citation indexing, and automatic linking of citations and documents. CiteSeer, now hosted at the Pennsylvania State University with several mirrors, has over 750,000 documents. The current CiteSeer model is to a limited extent portable and was recently extended to academic business documents (SMEALSearch).

Why has CiteSeer been so popular and how should it progress? What is its role with regards to other similar systems such as the Google Scholar and DBLP? What role should CiteSeer play in the open access movement? We discuss this and the Next Generation CiteSeer project, CiteSeerx, which will emphasize CiteSeer as a research tool, research web service, and researcher facilitator and testbed. In contrast to the current tightly integrated CiteSeer architecture, CiteSeerx will be modular, scalable and self managed. We will discuss how new intelligent data mining and information extraction algorithms will provide improved and new indexes, enhanced document access, expanded and automatic document gathering, collaboratories, new data and metadata resources, active mirroring, and web services. As an example of new features, we point out our new API based acknowledgement index and search. This new feature not only provides insight into the impact of acknowledged individuals, funding agencies and others, but also presents an architectural model for integration and expansion of our legacy system.

Learning to Have Fun

Jonathan Schaeffer

University of Alberta
Department of Computing Science
jonathan@cs.ualberta.ca

Abstract. Games have played a major role in the history of artificial intelligence research. The goal of this research largely has been to build programs capable of defeating strong human players. Most of the literature has been devoted to two-player, perfect information games—games where the research results have little wide-spread applicability. However, over the past few years the need for improved AI techniques have become apparent in commercial computer games, a $25 billion industry. Laird and van Lent call the new generation of commercial games "AI's killer application". The buying public wants to see realistic artificial intelligence in these products. Here the the metric is a "fun" experience, not winning. Hence, the outcomes from research using these applications will be of much wider applicability. This talk will discuss the challenges of using machine learning in commercial computer games to create "fun".

Winning the DARPA Grand Challenge

Sebastian Thrun

Stanford University, Robotics Lab
thrun@stanford.edu

Abstract. The DARPA Grand Challenge has been the most significant challenge to the mobile robotics community in more than a decade. The challenge was to build an autonomous robot capable of traversing 132 miles of unrehearsed desert terrain in less than 10 hours. In 2004, the best robot only made 7.3 miles. In 2005, Stanford won the challenge and the $2M prize money by successfully traversing the course in less than 7 hours. This talk, delivered by the leader of the Stanford Racing Team, will provide insights in the software architecture of Stanford's winning robot. The robot massively relied on machine learning and probabilistic modeling for sensor interpretation, and robot motion planning algorithms for vehicle guidance and control. The speaker will explain some of the basic algorithms and share some of the excitement characterizing this historic event. He will also discuss the implications of this work for the future of the transportation.

Challenges of Urban Sensing

Henry Tirri

Nokia Research Center
Henry.Tirri@nokia.com

Abstract. Wireless sensor networks are emerging as a critical information technology, and they are continuing the trend originating in mainframe computing currently at the stage of mobile computing. This trend shows several aspects consistent in the evolution of computing including the increasing hardware miniaturization of the computing units and an increasing emphasis of the role of communication between the computing units – "networking". In addition from the software side there is an increasing need to software solutions that are robust, exhibit distributed control, collaborative interfaces resulting in adaptive capabilities also at the system level. Like the present Internet, wireless sensor networks are large-scale distributed systems, but composed of smart sensors and actuators. They will eventually infuse the physical world and provide "grounding" for the Internet thus creating the Internet of Things. Research on wireless sensor networks has been taking place at several levels, from the lowest physical level to the highest information level – the latter is much less developed than the research at the physical levels. In addition, much of the research in wireless sensor networks has been focusing on military or science applications. However, wireless sensor networks can also play an important role in the realization of ubiquitous computing for everyday life - creating what we call "Urban sensing environment". In urban sensing many natural gateways exist to collect and process the sensor information – static ones such as media devices, or mobile devices such as smart phones that can collect sensor information when entering the communication range of an active sensor. Some of the applications of wireless sensor network technology at home include, in addition to the surveillance functions, adding "intelligence" to utility consumption, electronic tagging, contamination control and disaster monitoring. Similarly at the community level "traffic monitoring" including people allows a development of totally unseen services from micro weather forecasts to new ways for "sensing the environment" for entertainment. In this talk we will outline some of the research challenges for urban sensing, and the role of learning and data analysis techniques for solving those challenges.

Learning in One-Shot Strategic Form Games[⋆]

Alon Altman, Avivit Bercovici-Boden, and Moshe Tennenholtz

Faculty of Industrial Engineering and Management
Technion — Israel Institute of Technology
Haifa, 32000 Israel
alon@vipe.technion.ac.il, avivitb@tx.technion.ac.il
moshet@ie.technion.ac.il

Abstract. We propose a machine learning approach to action prediction in *one-shot* games. In contrast to the huge literature on learning in games where an agent's model is deduced from its previous actions in a multi-stage game, we propose the idea of inferring correlations between agents' actions in different one-shot games in order to predict an agent's action in a game which she did not play yet. We define the approach and show, using real data obtained in experiments with human subjects, the feasibility of this approach. Furthermore, we demonstrate that this method can be used to increase payoffs of an adequately informed agent. This is, to the best of our knowledge, the first proposed and tested approach for learning in one-shot games, which is the most basic form of multi-agent interaction.

1 Introduction

Learning in the context of multi-agent interaction has attracted the attention of researchers in cognitive psychology, experimental economics, machine learning, artificial intelligence, and related fields for quite some time (see e.g. [1,2,3,4,5,6,7,8]). Most of this work uses repeated games and stochastic games as models of such interactions. Roughly speaking, one can distinguish between two types of multi-agent learning: reinforcement learning and model-based/belief-based learning. In reinforcement learning an agent adapts its behavior based on feedback obtained in previous interactions, while model-based/belief-based learning is mostly concerned with inferring an "opponent model" from past observations. The aim of this paper is to tackle the problem of learning/predicting opponent actions, and thus our study can be viewed as part of model-based/belief-based learning. However, our study introduces the first general machine learning approach for tackling the prediction of an agent's action in general **one-shot** games, i.e. without having access to any information on how this agent has played this particular game in the past. This is in difference to the extensive literature on learning in repeated and extensive form games [9,10]. Although some work in experimental economics has been devoted to modeling agent behavior in one-shot games [11,12], that work did not present an effective general learning approach to action prediction/selection in such games. Recent work in AI [13] deals with the idea of learning population statistics in order to predict agent behavior in a specific game, namely the

⋆ We thank Ido Erev for helpful comments and illuminating discussions regarding this paper.

Nash Bargaining game. In difference to these lines of research we are after a general machine learning approach for action prediction in general one-shot games.

In order to tackle the above challenge we suggest to consider agents' interactions in a population of one-shot games. Given information on how different agents (including our "opponent") played in different games, our aim is to learn connections and correlations between agents' behaviors in different games, that will allow us to predict the opponent's behavior in one game (which he did not play yet) based on his or her action in another game. In a sense, what we offer is to try and learn association rules among games, ones that will allow to improve upon prediction of an agent's action in a given game. Our main contribution is by suggesting this approach, and proving its (perhaps surprising) feasibility using real data obtained in experiments involving human subjects.

The economics literature refers to population learning[14]. In population learning we aim at predicting an agent's action based on statistics on how the population played a game in the past. Population learning is considered a good predictor for an agent's behavior in a game, and therefore will be used as a benchmark. What we suggest is a mixed approach: given information about how the population played different one-shot games we aim at predicting an agent's action in a game she did not play based on the population statistics and rules which we try to infer about correlations between games. Our benchmark for success will be highly competitive, and coincide with the population statistics mentioned above: We aim at showing cases where association rules between games can be learned and lead to better predictions than the ones obtained by only using the statistics on how the population played in the game.

As we mentioned, to the best of our knowledge, there was no attempt to provide a machine learning approach for the general task of predicting agents' actions in one-shot games. Here we exploit a simple machine learning technique in order to offer an approach for addressing this issue. In a sense, our work is related to the relatively recent work on case-based decision making[15]. In case-based decision making, the idea of case-based reasoning, which is a classical topic in AI, is exploited to introduce an alternative approach to decision making in strategic contexts. This approach is based on similarity measure between different decision problems. Our work suggests to learn such similarity/association between decision problems, in order to improve upon opponent prediction in games, ultimately improving payoffs. We show that this machine learning approach is highly useful for that context.We have also experimented with other machine learning techniques (such as ID3 and KNN), but they were found to be less efficient for our objectives.

The paper is self contained in introducing the approach, the experimental setup, and the way in which real data has been obtained. Unfortunately, relying on simulations and on reasoning under abstract assumptions is known to be somewhat problematic in game-theoretic settings, and therefore we appealed to experimental design in this work. Our experiments use two standard sources from which the games we have experimented with have been selected: the first experiment uses games found in [16] , a standard game-theory book, and the second experiment uses games from the GAMUT site[17], a standard source for games used by CS researchers.

This paper is organized as follows: In section 2 we present the experimental setting. Sections 3 and 4 introduce and discuss the use of a simple machine learning technique (learning simple association rules) for successful action prediction in one-shot games. Section 5 extends upon the above technique (by considering aggregation of association rules), and shows the applicability of the machine learning approach to improving strategy selection in one-shot games.

2 The Experimental Setting

In this section we will describe the two experiments we performed. In the first experiment we have tested the idea and observed its (surprising) success, against a highly competitive benchmark. The second experiment has been carried out to further validate our observations, and test that our findings are robust for a variety of cases.

2.1 The First Experiment

For the purpose of the first experiment, 16 strategic form games with payoffs between 0 and 10 were constructed, based on [16]. 11 of the games are symmetric, and the remaining 5 asymmetric games were duplicated with the roles of the players interchanged. These games were printed on forms in random order, with a basic explanation on how the form is graded, which consisted of a brief explanation of the concept of a strategic form game. All games tested can be found in Figure 1.

Twenty-six students with basic knowledge of game theory were given these forms. The students were told to select a strategy for the row player in each of the games, and were told that they will be given bonus points to their exam grade based on the result of running one of the games against some other student. It was emphasized that there is no "correct" answer, and the students were encouraged to use any idea they may have in order to try and maximize their revenue.

Additional data were gathered by asking 36 friends and faculty members with a knowledge of game theory to participate in the experiment. These participants were similarly informed.

2.2 The Second Experiment

This experiment was performed on two subject populations. The main population (55 subjects) were undergraduate students who got paid according to their performance. Additional data were gathered from 38 faculty members and graduate students with a knowledge of game theory that were asked to play as if they were being similarly paid.

The subjects were initially allocated 150 points, each point worth about 2.5 cents (US). In the experiment there were three sections. In each of the sections the subjects could gain or lose points. The three sections were as follows:

– **Auctions section:** In this section we study the participants' behavior in first, second, and third price sealed bid auctions. In a k-price auction all agents submit bids simultaneously, the highest bidder wins, and pays the value of the kth highest bid (see [18]). Here, the subjects were presented with three identical tables, one table for each auction, See Table 1 for an example. For each subject, the valuations in the

First Experiment

1		Opponent	
Your	1.	**9**\9	**0**\10
Action	2.	**10**\0	**5**\5

2		Opponent	
Your	1.	**6**\6	**4**\9
Action	2.	**9**\4	**5**\5

3		Opponent	
Your	1.	**9**\9	**0**\10
Action	2.	**10**\0	**1**\1

4/5		Opponent		
Your	1.	**10**\6	**5**\0	**6**\7
Action	2.	**8**\8	**6**\9	**7**\7

6		Opponent	
Your	1.	**10**\10	**0**\3
Action	2.	**3**\0	**2**\2

7		Opponent	
Your	1.	**10**\10	**0**\9
Action	2.	**9**\0	**8**\8

8		Opponent		
Your	1.	**10**\10	**0**\0	**0**\0
Action	2.	**0**\0	**10**\10	**0**\0
	3.	**0**\0	**0**\0	**9**\9

9		Opponent	
Your	1.	**10**\10	**0**\0
Action	2.	**0**\0	**10**\10

10		Opponent	
Your	1.	**0**\0	**7**\6
Action	2.	**6**\7	**0**\0

11		Opponent	
Your	1.	**0**\0	**7**\6
Action	2.	**6**\7	**1**\1

12/13		Opponent	
Your	1.	**6**\8	**9**\6
Action	2.	**5**\7	**8**\9

14/15		Opponent	
Your	1.	**4**\8	**7**\6
Action	2.	**5**\7	**8**\9

16/17		Opponent	
Your	1.	**7**\7	**7**\7
Action	2.	**8**\6	**5**\5

18		Opponent		
Your	1.	**0**\0	**3**\4	**6**\0
Action	2.	**4**\3	**0**\0	**0**\0
	3.	**0**\6	**0**\0	**5**\5

19		Opponent	
Your	1.	**7**\7	**10**\8
Action	2.	**8**\10	**7**\7

20/21		Opponent		
Your	1.	**10**\2	**0**\0	**0**\0
Action	2.	**0**\0	**1**\4	**0**\0
	3.	**0**\0	**0**\0	**3**\3

Second Experiment

1		Opponent		
Your	1.	**100**\100	**0**\200	**0**\150
Action	2.	**200**\0	**150**\150	**0**\100
	3.	**150**\0	**100**\0	**50**\50

2/3		Opponent	
Your	1.	**30**\170	**30**\170
Action	2.	**50**\100	**0**\200

4/5		Opponent		
Your	1.	**70**\120	**150**\30	**80**\90
Action	2.	**90**\20	**200**\30	**90**\90
	3.	**150**\120	**90**\150	**0**\120

6/7		Opponent		
Your	1.	**20**\0	**70**\70	**100**\100
Action	2.	**70**\70	**0**\20	**70**\70
	3.	**100**\100	**70**\70	**20**\20

8		Opponent	
Your	1.	**40**\40	**0**\ − 70
Action	2.	−**70**\0	**100**\100

9		Opponent	
Your	1.	**20**\20	**100**\50
Action	2.	**50**\100	**40**\40

10/11		Opponent	
Your	1.	**70**\70	−**10**\ − 10
Action	2.	−**100**\ − 10	**100**\100

12		Opponent		
Your	1.	**0**\0	**100**\50	**100**\50
Action	2.	**50**\100	**0**\0	**100**\50
	3.	**50**\100	**50**\100	**0**\0

13/14		Opponent	
Your	1.	**0**\ − 50	**100**\ − 100
Action	2.	**50**\50	**50**\70

Fig. 1. List of games used in both experiments

Table 1. Example of an auction table

The product's value for you	Your bid
239	
419	
668	
947	

Table 2. Discretization of Auctions

Range	1st Price	2nd Price	3rd Price
1	$\alpha < 0.91$	$\alpha < 1$	$\alpha < 1$
2	$\alpha \geq 0.91$	$\alpha = 1$	$\alpha = 1$
3		$\alpha > 1$	$1 < \alpha < 1.125$
4			$\alpha \geq 1.125$

tables were uniformly distributed in four ranges between 1 and 1000. The subjects were told the rules of the various auctions, and were asked to provide a bid for each auction and for each of the valuations in the table. The subjects were told they will be partitioned into groups of 10 subjects each, and that for each of the auctions we will choose randomly one of their values and perform the auction. Detailed discussions with the subjects were carried out in order to verify they understand the auction rules and parameters.

– **The Centipede Game section:** This section has been devoted to the Centipede game following the description in [17]. The subjects were partitioned randomly into pairs, where one of each pair is randomly selected to play the role of Player 1 (who moves first), and the other plays the role of Player 2. The game is described as follows. There are many piles of points on the table, where each pile contains 60 points. The players alternate in their moves. At each move the player needs to decide whether to take one pile or two piles of points from the table. If a player decides to take two piles of points then the game ends. Otherwise, the game is over after 6 turns (3 for each subject). The subjects were given a detailed explanation of the game, and were asked to mark their selected moves (i.e. take one pile or take two piles) if they were playing the odd turns of the game. Similarly, given this game the participants were asked to mark their selected moves if they were playing the even turns.
– **Strategic form games section:** For the purpose of this section, 9 strategic form games with payoffs between -100 and 200 were constructed, based on games found in the Gamut web site. Four of the games are symmetric, and the remaining 5 asymmetric games were duplicated with the roles of the players interchanged. These games were printed on forms in random order. The subjects were explained the notion of a strategic form game, and their understanding was verified by solving a sample game with an obvious dominant strategy. The subjects were told to select a strategy for the row player in each of the games, and were told that they will be paid based on the result of running one of the games against some other subject. Two control games with obvious dominant strategies were also mixed in with the other games, and were used to verify the subjects' understanding. All games tested can be found in Figure 1.

3 Learning Algorithm and Evaluation

The experiments on the data were conducted using leave-one-out cross validation. In each iteration the play of one game g by one player x was hidden, and was to be predicted by the remaining data. We used the prediction based on most frequent play in g by the other players as a baseline for comparison. Using play data of the remaining players, rules were learned that map a strategy choice in games $g' \neq g$ where the play is known to a strategy choice in g. The association rule with the highest confidence level which was satisfied was applied to select a strategy choice for x.

Our learning system is based on the identification of association rules with high confidence and support. Association rules specify that given that a certain set of strategies is played by a player in a specific set of games, then the player will play a specific strategy in the game we are trying to predict with a given probability. That probability is called the confidence level of the association rule. The support of an association rule is the percentage of the population who confirm the rule. That is, satisfy both the preconditions and postcondition of the rule.

Formally, let $N = \{1, \ldots, n\}$ be the set of participants and let $G = \{1, \ldots, m\}$ be the set of games. Let $S_g = \{s_g^1, \ldots, s_g^{n_g}\}$ be the set of strategies for the row player in game g, and let $p_g^x \in S_g$ be the action selected by player x in game g. The baseline prediction for p_g^x is defined as

$$\hat{p}_g^x \in \text{argmax}_{s \in S_g} |\{i | i \neq x, p_g^i = s\}|.$$

If the maximum is not unique, the prediction is chosen arbitrarily among the maxima.

Association rules can be written in the form $S \Rightarrow s_g^k$, where

$$S \subseteq \{s_{g'}^i | g' \in G \setminus \{g\}, s_g^i \in S_g\}$$

and $\forall s_{g_1}^i \neq s_{g_2}^j \in S : g_1 \neq g_2$. Such a rule means that a player who has played the strategies in S is likely to play the strategy s_g^k. In this paper, we only consider rules in which $|S| \leq 1$.

For each player x, association rules of the form $\{s_g^i\} \Rightarrow s_{g'}^j$ were learned. These rules mean that if a player has played strategy s_g^i in game g, she is likely to play strategy $s_{g'}^j$ in game g'. For each rule, the support level was calculated as follows:

$$S(\{s_g^i\} \Rightarrow s_{g'}^j) = \frac{|\{k | k \neq x, p_g^k = s_g^i, p_{g'}^k = s_{g'}^j\}|}{n}.$$

Rules with $S(r) < 0.2$ were discarded. The support threshold of 0.2 has been chosen to fit the size of data considered in such experiments (similar results are obtained with slightly different support). The confidence level was calculated as follows:

$$C(\{s_g^i\} \Rightarrow s_{g'}^j) = \frac{|\{k | k \neq x, p_g^k = s_g^i, p_{g'}^k = s_{g'}^j\}|}{|\{k | k \neq x, p_g^k = s_g^i\}| + 1}.$$

One was added to the denominator to account for the uncertainty on the action selection of the agent being predicted.

Furthermore, for every game g, a baseline rule $\emptyset \Rightarrow s_g^i$ was added to the rule set with support 1 and confidence

$$C(\emptyset \Rightarrow s_g^i) = \frac{|\{k | k \neq x, p_g^k = s_g^i\}|}{n}.$$

Let R_x be the set of rules generated for player x.

For every game g, and for every player x, the association rules are tested in declining order of confidence. Thus, the predicted strategy $\tilde{p}_g^x$ is the strategy s for which the following maximum is obtained:

$$\max\{C(S \Rightarrow s) | (S \Rightarrow s) \in R_x \wedge \forall s_{g'}^i \in S : p_{g'}^x = s_{g'}^i\}$$

The auctions were evaluated as follows: For each auction we assumed a linear correlation between valuations and bids and calculated the slope α of the linear regression line for the 4 observations. We then split α into ranges as depicted in Table 2. These range designations were interpreted as strategies in the auction games.

4 Results

Our results show that the use of association rules has significantly improved our prediction of the strategies selected by the agents. This has been found in both experiments. We now illustrate some of the most effective association rules which have been learned, which have led to most significant improvements in prediction.

4.1 First Experiment

It is interesting to note that the improvements of results have been independently obtained for both sub-populations, as well as for the entire group.

The results of the experiment can be seen in Fig. 2. The average increase in prediction was 1% with a standard deviation of 0.05. Specifically, we have found a significant improvement in prediction in games 7, 20 and 21. The declines in prediction are an artifact of the leave-one-out cross validation technique, where removing one agent changes the learned association rules for the worst. The low average increase is expected, and less relevant, as we expect to improve prediciton in only a few of the games. The improvements in prediction are a result of learning and application of several meaningful and nontrivial association rules discussed below[1]:

- $s_3^1 \Rightarrow s_7^1 \begin{array}{|c|c|}\hline \mathbf{9}\backslash 9 & \mathbf{0}\backslash 10 \\ \hline \mathbf{10}\backslash 0 & \mathbf{1}\backslash 1 \\ \hline \end{array} \Rightarrow \begin{array}{|c|c|}\hline \mathbf{10}\backslash 10 & \mathbf{0}\backslash 9 \\ \hline \mathbf{9}\backslash 0 & \mathbf{8}\backslash 8 \\ \hline \end{array}$

  This rule is responsible for a 15% increase in prediction accuracy, up to 75% accuracy in game 7.

  This rule means that players who played *cooperate* in the extreme version of the prisoner's dilemma (game 3), tended to play the *trust* strategy in the extreme version of the trust game (game 7).

  We see that players who hope for the 9\9 result in the prisoner's dilemma also try for the risky 10\10 in the trust game. We capture a tendency to improve the total welfare of both players, even over the benefit of playing a dominant strategy and the risk of playing the trust strategy.

  This rule also captures players who strongly believe in symmetry, and try to grab the best payoff assuming the other player does the same.

[1] Selected strategies are shaded.

The inverse rule was not learned, as many more players have played *trust* in the trust game than *cooperate* in prisoner's dilemma. This is predictable, due to the fact that *defect* is a dominant strategy in prisoner's dilemma, while no such dominant strategy exists in the trust game.

– $s^1_{20} \Rightarrow s^2_{21}$

10\2	**0**\0	**0**\0
0\0	**1**\4	**0**\0
0\0	**0**\0	**3**\3

$\Rightarrow$

2\10	**0**\0	**0**\0
0\0	**4**\1	**0**\0
0\0	**0**\0	**3**\3

This rule is responsible for a 10% increase in prediction accuracy, up to 55% accuracy in the inverse of game 20.

This rule means that in game 20, players that select the row with the highest payoff for themselves when playing rows, tend to do the same when playing columns.

This rule captures players who play a "bully" strategy. That is, try to maximize their own payoffs, under the assumption that the other player will play their best response.

In game 20, this bully strategy is the most common strategy, played by 57% of the participants. In its inverse only 31% of the population do so, many of them have "bullied" in game 20 as well.

In this case, the inverse rule is trivial, as s^1_{20} is already the most common strategy in game 20.

– $s^3_{21} \Rightarrow s^3_{20}$

2\10	**0**\0	**0**\0
0\0	**4**\1	**0**\0
0\0	**0**\0	**3**\3

$\Rightarrow$

10\2	**0**\0	**0**\0
0\0	**1**\4	**0**\0
0\0	**0**\0	**3**\3

This rule is responsible for a 15% increase in prediction accuracy, up to 70% accuracy in game 20.

This rule captures the fact that players who try to reach a middle ground in game 21 tend to do the same in the original game 20.

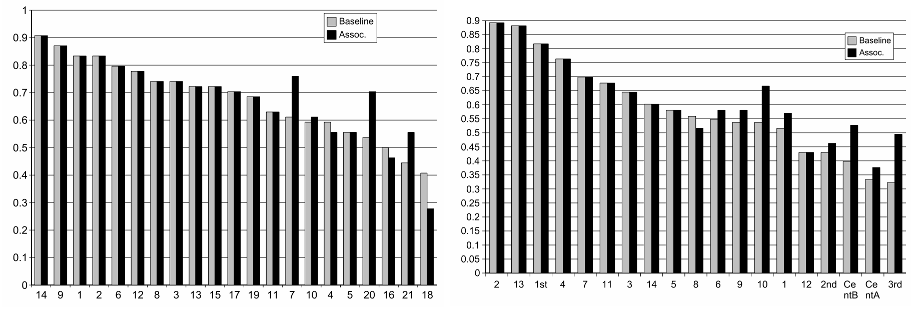
This rule captures players who believe in symmetry in the sense that both players should get the same payoffs, and consistency in the sense that strategy selection in the rows and columns should be in some kind of equilibrium when possible.

Again, the inverse rule is trivial, since s^3_{21} is the most common strategy in the inverse of game 20.

4.2 Second Experiment

The results of running the second experiment on the full corpus of 93 players are in Figure 2. For every game (see Figure 1) there are two bars representing the portion of correct predictions for the baseline(grey) and association rule(black). The results for each of the sub-populations were similar.

We see a significant improvement in prediction for the Centipede, game 10 and the Third price auction. In a few other games, there were less significant differences between the quality of predictions when using the learning rule compared to the baseline predictions. In all games, excluding one, the learning rule predictions were equal or



(a) First Experiment (b) Second Experiment

Fig. 2. Prediction results for the first and second experiments

superior to the baseline. In general the average increase in prediction was 3% with standard deviation of 0.07, which is expected as in the first experiment. The significant improvement in prediction is a result of learning and application of several meaningful and nontrivial association rules as illustrated below.

The following association rules learned by our algorithm resulted in improved predictions:

- $s^1_{CentipedeA} \Rightarrow s^1_{CentipedeB}$
  This rule means that players who took 2 piles of 60 points in the first turn of the Centipede game, when playing the odd turns, tended to take 2 piles in their first turn when they were playing the even turns. We expose here a tendency of the players to be self coherent and take the money in the first opportunity they can, rather than take the chance of highly improving both players' payoffs.
- $s^4_{CentipedeA} \Rightarrow s^3_{CentipedeB}$
  This rule means that players who took only one pile of 60 points in all of their turns when playing the odd turns, tended to take two piles of 60 points only on the 6th turn when playing the even turns. We see that players who hope that the other player will "cooperate" with them in order to accumulate more points, take two piles of points only in their last (6th) turn. We capture here a tendency to improve the total welfare of both players, even over the benefit of playing a dominant strategy and the risk of trusting the other player.
- $s^3_4 \Rightarrow s^2_{10}$

70\120	**150**\30	**80**\90
90\20	**200**\30	**90**\90
150\120	**90**\150	**0**\120

$\Rightarrow$

70\70	**-10**\-10
-100\-10	**100**\100

This rule means that in game 4, players that select the row with the highest risk, tend to do the same when playing game 10.

This rule captures players who are risk seeking, and try to increase social welfare by selecting rows which may lead to significant loss (0 in the former case, and -100 in the latter), but contain the best sum for both players.

- $s_8^2 \Rightarrow s_{10}^2$

40\40	**0**\-70
-70\0	**100**\100

  $\Rightarrow$

70\70	**-10**\-10
-100\-10	**100**\100

  This rule means that in game 8, players that select the row with the highest risk, tend to do the same when playing game 10.

  This rule further captures players who are risk seeking .
- $s_{Second}^1 \Rightarrow s_{Third}^1$
  The meaning of the above rule is that the subjects who underbid in the second price auction also underbid in the third price auction. This rule was highly useful: from among 39 subjects that underbid in the second price auction, 26 underbid in the third price auction.

  This rule captures players who play "safe", and did not notice the dominant strategy, and were fearful of winning the auction but not incurring profit (or incurring loss).

5 Improving Strategy Selection

We have seen that learning of association rules may improve prediction of strategy selection by players in one-shot strategic games. The question arises whether or not this prediction can be used to improve a player's expected payoff when playing against a player whose actions in other games are known. That is, we wish to see whether by learning association rules an agent may obtain higher payoffs in games. Our results indicate that the answer is yes.

In order to do so, we consider an extension of the simple technique discussed in previous sections. Indeed, in order to select an appropriate action one may wish to have a rather accurate estimation of the probability distribution of his opponent strategies. In order to do so we need to exploit the information hidden in the set of association rules.

We use the population data and the 10 association rules with the highest confidence, combined using linear regression to estimate the probability distribution over our opponent's strategies. This enables us to calculate the best response against this particular opponent[2].

The above method differs from choosing a best response to the predicted (pure) strategy by the fact that we compensate for the scale of our lost payoffs due to our mistakes. For example, in game 4 (in the first experiment) our data suggests that the best response to the mixed strategy defined by the population is for the column player to select the leftmost column, even though this strategy is not a best response to any pure strategy of the row player.

Assume we are trying to respond a player p's behavior in game g. The probability distribution over this player's strategy is estimated as follows: For every player x and for every strategy s_g^i in g we compute the following three parameters on the training set with x removed:

1. F_g^i – the frequency of the strategy s_g^i in the population without x:

$$F_g^i = \frac{|\{k|k \neq x, p_g^k = s_g^i\}|}{n}.$$

[2] A similar technique (of using the linear regression to combine a new model and experimental data, to better predict probabilities) has also been considered in economic literature, see [19].

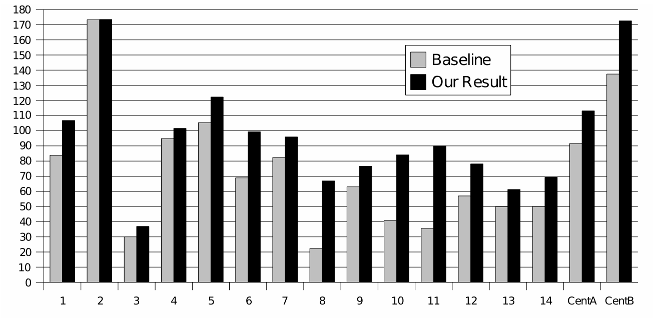


Fig. 3. Results of best response from the second experiment

2. C^i_g – the confidence of the association rule for g with the greatest confidence among the rules whose pre-condition is satisfied by player x if this rule predicts s^i_g, and 0 otherwise.
3. A^i_g – the average confidence among the 10 association rules for g with the greatest confidence, where a rule that does not predict s^i_g is assumed to have confidence 0. That is, if the 10 association rules for g with the greatest confidence predict $s_1, s_2, \ldots, s_{10}$ with confidence levels $c_1, c_2, \ldots, c_{10}$ respectively, then the average confidence will be

$$A^i_g = \frac{1}{10}\sum_{t=1}^{10}\begin{cases} c_t & s_t = s^i_g \\ 0 & \text{Otherwise} \end{cases}$$

Let R^i_g be 1 if player x played strategy s^i_g in game g, and 0 otherwise. For every strategy in g we take these three parameters and develop a linear regression line of the form $B_0 + B_1 * F^i_g + B_2 * C^i_g + B_3 * A^i_g$, where the B_i are selected to minimize

$$(B_0 + B_1 * F^i_g + B_2 * C^i_g + B_3 * A^i_g - R^i_g)^2$$

over all players. Given these B_i values, we calculate these parameters for the game g and player p over the entire training set, and calculate the probability for each move. Then we calculate a best response to the mixed strategy defined by this probability distribution.

We have tested this best response algorithm using leave-one-out cross validation on the data of the second experiment. The results are shown in Fig. 3. The average increase in payoff is 48%. We see a significant increase in payoffs across almost all the games, compared to the very competitive baseline of best-response to the empirical distribution of the players' actions.

6 Concluding Remarks

We have introduced a novel approach to machine learning in one-shot games by utilizing information about actions of the player in other games. Our experimental results are encouraging and confirm that this approach is in fact viable.

We have seen evidence that learning of association rules can improve by some level both prediction and play in one-shot strategic form games.

When larger data-sets are available, one may try to re-visit learning algorithms such as ID3, k-nearest neighbor, Bayesian inference, and Support Vector Machines, which require larger data sets in order to produce meaningful results. As we mentioned, with the current available data, these algorithms have been found less efficient than the technique previously discussed.

In conclusion, we have opened room for work in ML on action prediction and strategy selection in one-shot games, exploiting correlations between games which need not be explicitly specified.

References

1. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of AI Research **4** (1996) 237–285
2. Erev, I., Roth, A.: Predicting how people play games: Reinforcement learning in games with unique strategy equilibrium. American Economic Review **88** (1998) 848–881
3. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multi-agent systems. In: Proc. Workshop on Multi-Agent Learning. (1997) 602–608
4. Fudenberg, D., Levine, D.: The theory of learning in games. MIT Press (1998)
5. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Proc. 11th ICML. (1994) 157–163
6. Hu, J., Wellman, M.: Multi-agent reinforcement learning: Theoretical framework and an algorithms. In: Proc. 15th ICML. (1998)
7. Brafman, R.I., Tennenholtz, M.: R-max – a general polynomial time algorithm for near-optimal reinforcement learning. In: IJCAI'01. (2001)
8. Carmel, D., Markovitch, S.: Exploration strategies for model-based learning in multiagent systems. Autonomous Agents and Multi-agent Systems **2**(2) (1999) 141–172
9. Kagel, J., Roth, A.: The Handbook of Experimental Economics. Princeton University Press (1995)
10. Roth, A., Erev, I.: Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. Games and Economic Behavior **8** (1995) 164–212
11. Camerer, C.F., Ho, T.H., Chong, J.K.: A cognitive hierarchy model of games. The Quarterly Journal of Economics **119**(3) (2004) 861–898
12. Costa-Gomes, M., Crawford, V.P., Broseta, B.: Cognition and behavior in normal-form games: An experimental study. Econometrica **69**(5) (2001) 1193–1235
13. Gal, Y., Pfeffer, A., Marzo, F., Grosz, B.J.: Learning social preferences in games. In: Proc. of AAAI-04. (2004) 226–231
14. Stahl, D.O.: Population rule learning in symmetric normal-form games: theory and evidence. Journal of Economic Behavior and Organization **1304** (2001) 1–14
15. Gilboa, I., Schmeidler, D.: Case-based decision theory. Quarterly Journal of Economics **110** (1995) 605–639
16. Fudenberg, D., Tirole, J.: Game Theory. MIT Press (1991)
17. Nudelman, E., Wortman, J., Shoham, Y., Leyton-Brown, K.: Run the gamut: A comprehensive approach to evaluating game-theoretic algorithms. In: AAMAS '04. (2004) 880–887
18. Wolfstetter, E.: Auctions: An introduction. Journal of Economic Surveys **10**(4) (1996) 367–420
19. Erev, I., Roth, A., Slonim, R., Barron, G.: Learning and equilibrium as useful approximations: accuracy of prediction on randomly selected constant sum games. (2006)

A Selective Sampling Strategy for Label Ranking

Massih Amini[1], Nicolas Usunier[1], François Laviolette[2], Alexandre Lacasse[2], and Patrick Gallinari[1]

[1] Laboratoire d'Informatique de Paris 6
Université Pierre et Marie Curie, Paris, France
{FirstName.LastName}@lip6.fr
[2] Département IFT-GLO
Université Laval, Sainte-Foy (QC), Canada
{FirstName.LastName}@ift.ulaval.ca

Abstract. We propose a novel active learning strategy based on the compression framework of [9] for label ranking functions which, given an input instance, predict a total order over a predefined set of alternatives. Our approach is theoretically motivated by an extension to ranking and active learning of Kääriäinen's generalization bounds using unlabeled data [7], initially developed in the context of classification. The bounds we obtain suggest a selective sampling strategy provided that a sufficiently, yet reasonably large initial labeled dataset is provided. Experiments on Information Retrieval corpora from automatic text summarization and question/answering show that the proposed approach allows to substantially reduce the labeling effort in comparison to random and heuristic-based sampling strategies.

1 Introduction

This paper presents an active learning strategy for label ranking functions - mappings from instances to rankings over a finite set $\mathcal{A}$ of *alternatives*. The supervised learning of label-ranking functions has attracted considerable attention from the Machine Learning (ML) community (see e.g.[3,5]) since it encompasses tasks ranging from multi-class(-label) classification to ranking for Information Retrieval (IR) applications.

In this study we are interested on IR-like applications, such as Document Retrieval (DR). In this case, an instance x is a query and the label ℓ of x is a partial order over a given document collection $\mathcal{A}$. In a supervised setting, the aim is to learn a mapping (or a ranking function) from a predefined set of queries for which there exists a set of *relevance judgments* that indicates which documents in $\mathcal{A}$ are relevant to each query. In such a case labeling an instance often requires an expert to carefully examine the set of alternatives. The human effort to create labeled datasets may in general be unrealistic. It is thus necessary to design accurate methods for reducing the size of the required labeled set.

Different strategies have been proposed in the classification framework to cope with this kind of problem. One approach is selective sampling: given an input pool or stream of unlabeled examples, the algorithm selects a few of them and queries an oracle to obtain their labels. These new labeled examples are then added to the training set. Such strategies have been developed around two main ideas. (a) The shrinking of the version space, which in the case of linear discriminant functions, consists in the selection of the unlabeled instance with the smallest margin [11], and (b) the selection of examples which will reduce an approximation of the generalization error [4].

These theoretical motivations, unfortunately, do not extend to label ranking functions. Indeed, there is no equivalent notion of the version space, and approximations of the generalization error are mostly unknown. For the specific case where all the labels of instances are total orders and when the ranking is predicted by a real-valued scoring function, the notion of margin may be extended. Hence, [2] showed that by taking the minimum difference of scores between two alternatives, and selecting the unlabeled examples with the smallest extended margin is a performing heuristic. However, although the "extended margin" heuristic can be also applied in the general case, we cannot expect it to perform well: taking the example of DR, a real-valued scoring function may assign very similar scores to two relevant or two irrelevant documents. Hence, for a given instance, the extended margin may be close to zero independently from the fact that relevant documents have higher scores than irrelevant ones.

In this paper, we propose a new selective sampling strategy for label ranking. Our starting point is the generalization error bounds using unlabeled data proposed by [7] for classification: the generalization error of a classifier can be bounded by the error of another classifier plus the probability of disagreement between both classifiers. In the specific framework of label ranking described in section 2, we show in section 3 that the bounds proposed by [7] can be extended for label ranking in the following way: given a fixed, but arbitrary, cost function and given some prior knowledge about the labels, a cost-specific notion of disagreement between two ranking functions can be constructed. Then the generalization error of a ranking function $\bar{f}$ we want to learn can be bounded by two terms: the generalization error of a specific ranking function $\bar{f}^{cv}$ built using cross-validation (CV) sets, and the probability of disagreement between this ranking function and $\bar{f}$. We then consider the problem of selective sampling as choosing the unlabeled examples to reduce the generalization error bound of $\bar{f}$, which can be done in a greedy fashion by selecting instances for which the disagreement between $\bar{f}$ and $\bar{f}^{cv}$ is the highest. Finally, in section 4, we show experimental results on two IR corpora from automatic text summarization and question/answering systems comparing our approach to the extended margin heuristic and the random sampling strategy.

2 Notation

Let us define the following notations in addition to those given in the introduction. For simplicity, we identify the set of alternatives $\mathcal{A}$ by $\{1, ..., A\}$. The instance and the label spaces are denoted respectively by $\mathcal{X}$ and $\mathcal{L}$. A ranking function is defined as $\bar{f} : \mathcal{X} \to \sigma_A$, where σ_A is the set of permutations of $\{1, ..., A\}$. Hence for an instance $x \in \mathcal{X}$, an alternative $i \in \mathcal{A}$ is preferred over an alternative $j \in \mathcal{A}$ iff $\bar{f}(x)(i) < \bar{f}(x)(j)$. We furthermore suppose that the training set is composed of a *labeled* set $Z_\ell = ((x_i, \ell_i))_{i=1}^n \in \mathcal{Z}^n$ and an *unlabeled* set $X_\mathcal{U} = (x'_j)_{j=n+1}^{n+m} \in \mathcal{X}^m$, where $\mathcal{Z}$ represents the set of $\mathcal{X} \times \mathcal{L}$. We suppose that each pair $(x, l) \in Z_\ell$ is drawn i.i.d with respect to a fixed but unknown distribution $\mathcal{D}$ and we denote the marginal distribution over $\mathcal{X}$ by $D_\mathcal{X}$.

We will furthermore assume that for each instance x, only a subset $\mathcal{A}_x$ of $\mathcal{A}$ is considered, and that $\mathcal{A}_x$ is known even if the label of x is unknown. The set of possible labels for x, denoted $\mathcal{L}_x$, contains thus only preference relations over $\mathcal{A}_x$. When the labels are induced from binary relevance judgements, any label of $\mathcal{L}_x$ can be represented by two sets of indices Y_x^+ and Y_x^- of relevant and irrelevant alternatives in $\mathcal{A}_x$.

These notations allow us to formulate naturally costs functions in IR. For example, precision at k which counts the proportion of relevant alternatives in the first k positions can be defined by:

$$c_{p@k}(\bar{f}(x), \ell) = \frac{1}{k} \sum_{i \in Y_x^+} [\![\bar{f}(x)(i) \leq k]\!] \tag{1}$$

where $[\![pr]\!]$ is one if predicate pr holds and 0 otherwise. Another example is the rank loss function which measures the mean number of irrelevant elements better ranked (the lower the better) by $\bar{f}$ than relevant ones:

$$c_{Rloss}(\bar{f}(x), \ell) = \frac{1}{|Y_x^+||Y_x^-|} \sum_{j \in Y_x^-} \sum_{i \in Y_x^+} [\![\bar{f}(x)(j) < \bar{f}(x)(i)]\!] \tag{2}$$

Finally we will denote by $\hat{\epsilon}_Z(\bar{f}) = \frac{1}{n}\sum_{i=1}^n c(\bar{f}(x_i), \ell_i)$ the empirical risk of a ranking function $\bar{f}$ and by $\epsilon(\bar{f}) = \mathbb{E}_{(x,\ell)\sim\mathcal{D}} c(\bar{f}(x), \ell)$ its true risk. When $\bar{f}$ predicts outputs based on a real-valued (i.e. scoring) function over the set $\mathcal{A}_x$, we denote by f the associated scoring function.

3 A New Query Selection Strategy

In this section, we present a divergence measure for ranking functions, and present our ranking bounds based on unlabeled data. From that we propose the ranking active learning algorithm which is the central point of the paper.

3.1 Generalization Error Bound for Ranking Functions

We define a randomized ranking function as a σ_A-valued random variable such that for each instance x, a randomized ranking function $\bar{f}_\theta$ is chosen according to a probability distribution Θ over a finite set of ranking functions $\{\bar{f}_1, ..., \bar{f}_K\}$ and an ordered list $\bar{f}_\theta(x)$ over $\mathcal{A}$ is returned. If Θ is a uniform distribution we denote the randomized ranking function by $\bar{f}_K$. The generalization error bound we propose in the following is based on a divergence function $d_c : \sigma_A \times \sigma_A \to [0, 1]$

associated with a risk function c measuring for any query $x \in \mathcal{X}$ the disagreement between two ranking functions $\bar{f}$ and $\bar{f}'$. We define d_c as

$$d_c(\bar{f}(x), \bar{f}'(x)) = \max_{\ell \in \mathcal{L}_x} \left[c(\bar{f}(x), \ell) - c(\bar{f}'(x), \ell) \right]$$

Clearly, d_c is a divergence upper bounded by 1 (i.e., $1 \geq d_c(y, y') \geq 0$ for any y, y' and $d_c(y, y') = 0$ iff $y = y'$). Moreover, we have:

$$\forall (x, \ell) \in \mathcal{Z},\ c(\bar{f}(x), \ell) \leq c(\bar{f}'(x), \ell) + d_c(\bar{f}(x), \bar{f}'(x))$$

For two randomized ranking functions $\bar{f}_\Theta$ and $\bar{f'}_\Lambda$ the notion of risk can be extended by denoting $c(\bar{f}_\Theta(x), \ell) = \mathbb{E}_{\theta \sim \Theta} c(\bar{f}_\theta(x), \ell)$ and $d_c(\bar{f}_\Lambda(x), \bar{f'}_\Theta(x)) = \mathbb{E}_{\lambda \sim \Lambda, \theta \sim \Theta} d_c(\bar{f}_\lambda(x), \bar{f'}_\theta(x))$.

From these definitions we still have:

$$\forall (x, \ell) \in \mathcal{Z},\ c(\bar{f}_\Lambda(x), \ell) \leq c(\bar{f}'_\Theta(x), \ell) + d_c(\bar{f}_\Lambda(x), \bar{f}'_\Theta(x)) \tag{3}$$

Equation 3 shows a link between the values of the risk function c on $\bar{f}_\Lambda$ and on $\bar{f}_\Theta$ and therefore indicates a possible link between the risk of those two (possibly random) ranking functions. In the stochastic case, the true and empirical risks of a randomized ranking function $\bar{f}_\Theta$ are defined as

$$\hat{\epsilon}_Z(\bar{f}_\Theta) = \frac{1}{n} \sum_{i=1}^{n} c(\bar{f}_\Theta(x_i), \ell_i) = \mathbb{E}_{\theta \sim \Theta} \frac{1}{n} \sum_{i=1}^{n} c(\bar{f}_\theta(x_i), \ell_i)$$
$$\epsilon(\bar{f}_\Theta) = \mathbb{E}_{(x,\ell) \sim \mathcal{D}} c(\bar{f}_\Theta(x), \ell) = \mathbb{E}_{\theta \sim \Theta} \mathbb{E}_{(x,\ell) \sim \mathcal{D}} c(\bar{f}_\theta(x), \ell)$$

Notice that if Z is drawn i.i.d. according to $\mathcal{D}$, then $\hat{\epsilon}_Z(\bar{f}_\Theta)$ is an unbiased estimator of $\epsilon(\bar{f}_\Theta)$. We can also notice that, if $X_\mathcal{U}$ is drawn i.i.d. according to $\mathcal{D}_\mathcal{X}$, the mean of $d_c(\bar{f}_\Lambda(x'), \bar{f'}_\Theta(x'))$ for $x' \in X_\mathcal{U}$ is an unbiased estimator of $\mathbb{E}_{x' \sim \mathcal{D}_\mathcal{X}} d_c(\bar{f}_\Lambda(x'), \bar{f'}_\Theta(x'))$. The following theorem is an extension to ranking functions of a classifier risk bound based on unlabeled data introduced in [7].

Theorem 1. *For any two (possibly randomized) ranking functions $\bar{f}_\Lambda$ and $\bar{f'}_\Theta$, we have:*

$$\epsilon(\bar{f}_\Lambda) \leq \epsilon(\bar{f}'_\Theta) + \mathbb{E}_{x' \sim \mathcal{D}_\mathcal{X}} d_c(\bar{f}_\Lambda(x'), \bar{f}'_\Theta(x'))$$

Proof: Using inequality 3, we get the result by taking the expectation over $(x, \ell) \sim \mathcal{D}$.

Thus, using unlabeled data, one can obtain an upper bound on the risk of $\bar{f}_\Lambda$, if such a bound exists for $\bar{f}'_\Theta$. From now, we suppose that one of the ranking function $\bar{f}'_\Theta$ is obtained by cross-validation on a training labeled data set which is defined as follows.

Definition 2 (Cross-validation sets). *Given a labeled dataset Z drawn i.i.d. according to $\mathcal{D}$, a cross-validation (CV) set of size K is any partition of Z into K disjoint subsets $Z_1, ..., Z_K$ of equal size*[1]*. Moreover, for any CV set $Z_1, ..., Z_K$, we associate the sets, for $i = 1, ..., K$, $Z_i^{train} = \bigcup_{j \neq i} Z_j$ and $Z_i^{test} = Z_i$.*

[1] The results contained in this paper remain valid if the CV set size do not divide $|Z|$, but to simplify the notation, we have restricted ourselves to the case where it does.

Hence, given a ranking learning algorithm $\mathcal{R}$, and a CV set of size K, $\{Z_1, ..., Z_K\}$ for Z, a *randomized ranking function obtained by cross-validation* is the randomized function defined by the uniform probability distribution on the set $\{\mathcal{R}(Z_1^{train}), \ldots, \mathcal{R}(Z_K^{train})\}$. We will denote by $\bar{f}_K^{cv}$, the obtained randomized ranking function, and by $\bar{f}_j$, the function $\mathcal{R}(Z_j^{train})$. The results of the rest of this section show how the risk bound of such randomized ranking function can be estimated, and then how the bound of Theorem 1 can be computed in practice. Those results are based on the following version of the Hoeffding's bound:

Theorem 3 (Hoeffding bound). *Let $X_1, ..., X_n$ be n copies of a $[0,1]$-valued random variable X, then, for all $\delta > 0$:*

$$\mathbb{P}\Big(\mathbb{E}X \leq \tfrac{1}{n}\textstyle\sum_{i=1}^{n} X_i + \sqrt{\tfrac{\ln(1/\delta)}{2n}}\Big) > 1 - \delta$$

Combining Theorem 1 and the Hoeffding bound we obtain the following bound for the true risk of $\bar{f}_K^{cv}$

Lemma 4. *Let Z be drawn i.i.d. according to $\mathcal{D}$ and let $\{Z_1, ..., Z_K\}$ be a CV set of size K such that K divides n. Then, with probability at least $1 - \delta/2$ over samples Z, the risk of $\bar{f}_K^{cv}$ is given by:*

$$\epsilon(\bar{f}_K^{cv}) \leq \tfrac{1}{K}\textstyle\sum_{j=1}^{K} \hat{\epsilon}_{Z_j}(\bar{f}_j^{cv}) + \sqrt{\tfrac{K}{2n}\ln\tfrac{2K}{\delta}}$$

Proof: Hoeffding bound implies that for all $j \in \{1, \ldots, K\}$ we have $\mathbb{P}\left(\epsilon(\bar{f}_j^{cv}) > \hat{\epsilon}_{Z_j}(\bar{f}_j^{cv}) + \sqrt{\frac{K}{2n}\ln\frac{2K}{\delta}}\right) \leq \frac{\delta}{2K}$. The result of the lemma then follows from the union bound theorem: $\mathbb{P}(\cup A_i) \leq \sum \mathbb{P}(A_i)$.

The following lemma bounds $\mathbb{E}d_c(\bar{f}(x'), \bar{f}_K^{cv}(x'))$ by its expected value computed over a training unlabeled dataset.

Lemma 5. *Let $X_{\mathcal{U}}$ be an unlabeled dataset drawn independently from a labeled dataset Z, and let $\bar{f}$ be a ranking function that has been learned independently of a subset $X_{\mathcal{U}}^{(k)} = \{x'_{j_1}, ..., x'_{j_k}\}$ of size k of $X_{\mathcal{U}}$. Then we have:*

$$\mathbb{P}\left(\mathop{\mathbb{E}}_{x'\sim\mathcal{D}_{\mathcal{X}}} d_c(\bar{f}(x'), \bar{f}_K^{cv}(x')) \;\leq\; \frac{1}{k}\sum_{l=1}^{k} d_c\big(\bar{f}(x'_{j_l}), \bar{f}_K^{cv}(x'_{j_l})\big) + \sqrt{\frac{\ln(\frac{2}{\delta})}{2k}}\right) > 1 - \frac{\delta}{2}$$

where the probability is taken over the choices of $X_{\mathcal{U}}$.

Proof: Since d_c is a $[0,1]$-random variable which is function of the $x' \in X_{\mathcal{U}}^{(k)}$, and since the x' are i.i.d., the result follows from Theorem 3, [with $\delta := \delta/2$, $n := k$, $X := d_c(\bar{f}(x'), \bar{f}_K^{cv}(x'))$, $X_i := d_c(\bar{f}(x'_{j_i})\bar{f}_K^{cv}(x'_{j_i}))$].

Theorem 1, together with the last two lemmas give an upper bound of the risk of any *a priori* chosen ranking function. This bound can be accurately estimated from the labeled data and a subset of the unlabeled data, provided that the size of the latter and n/K are large enough, and provided that the divergence d_c can be easily estimated. The next proposition shows that this is the case for the risk function $c := c_{Rloss}$ that we consider in our experiments.

Proposition 6. *Let $\bar{f}$ and $\bar{f}'$ be two ranking functions, and x an unlabeled instance in the case where labels are generated based on binary relevance judgements of the alternatives. Then, using the risk functions of Equation 2, we have:*

$$d_{c_{Rloss}}(\bar{f}(x), \bar{f}'(x)) \leq \max_{p,q:p+q=A_x} \frac{1}{pq} \sum_{k=1}^{p} \delta(\bar{f}(x), \bar{f}'(x))_k$$

where $\delta(\bar{f}(x), \bar{f}'(x))$ is the list of length A_x containing all the values of $\bar{f}(x)(i) - \bar{f}'(x)(i)$ for $1 \leq i \leq A_x$ ordered in decreasing value.

Proof: Assume that the true label ℓ of x is $Y_x = (Y_1, ..., Y_{A_x})$. We denote $rg(i) = \bar{f}(x)(i) - 1$, and for each $i \in Y_x^+$, $rg_+(i) = \sum_{j \in Y_x^+} [\![\bar{f}(x)(i) > \bar{f}(x)(j)]\!]$ (the number of relevant alternatives ranked before relevant alternative i). Then, using equation 2, we have $c_{Rloss}(\bar{f}(x), \ell) = \frac{1}{|Y_x^+||Y_x^-|} \sum_{i \in Y_x^+} (rg(i) - rg_+(i))$. Since rg_+ and rg'_+ do only consider relevant alternatives, we have $\sum_{i \in Y_x^+} rg_+(x) = \sum_{i \in Y_x^+} rg'_+(i)$, and therefore

$$c_{Rloss}(\bar{f}(x), \ell) - c_{Rloss}(\bar{f}'(x), \ell) \leq \frac{1}{|Y_x^+||Y_x^-|} \sum_{i=1}^{|Y_x^+|} \delta(\bar{f}(x), \bar{f}'(x))_i$$

The results yields by taking the maximum value of the right-hand side of the last equation over all possible values of these numbers.

For a given instance x, the complexity of $d_{c_{Rloss}}$ is $O(|\mathcal{A}_x| ln |\mathcal{A}_x|)$, since the most expensive computation is sorting a list of size $|\mathcal{A}_x|$.

3.2 A Uniform Risk Bound for Active Learning

To minimize $\epsilon(\bar{f}_\Theta)$, one can try to minimize $\mathbb{E}_{x' \sim \mathcal{D}_\mathcal{X}} d_c(\bar{f}(x'), \bar{f}_K^{cv}(x'))$. However, in order to use Theorem 1 in an active learning algorithm, we will need a bound that is uniformly valid for all ranking functions $\bar{f}$, and all "possible" sequence of queries. This can be done in the same way as for the sample compression scheme [6] in supervised learning via the union bound.

In the sample compression scheme, given a (labeled) training set S of an *a priori* defined size m, any classifier returned by a learning algorithm is described by a compression set. A *compression set* is a subset of the training set S and therefore, when S is given, can be described as a vector of indices $\boldsymbol{i} = (i_1, i_2, \ldots, i_k)$ with $i_j \in \{1, \ldots, m\}$ $\forall j$ and $i_1 < i_2 < \ldots < i_k$. This implies that there exists a deterministic *reconstruction function*, associated with the algorithm, that outputs a classifier when given a training set and a vector of indices. The perceptron and the SVM are such an example. In that setting, given an *a priori* defined vector $\boldsymbol{i}$ of indices, one can use the examples of the training set that do not correspond to any index of $\boldsymbol{i}$ to bound the risk of the classifier defined by $\boldsymbol{i}$ (and the training set S). Moreover, provided a prior distribution is given on the set of all possible vector of indices, one can extend such a bound to a bound which is valid simultaneously for all classifiers that can be reconstructed [8].

Since any active learning ranking algorithm $\mathcal{R}$ considered in this paper is deterministic, the set of all possible ranking functions that can be output by $\mathcal{R}$ depends only on the set of all examples of $X_{\mathcal{U}}$ that have been queried during the execution together with all the corresponding labels that have been given in response to the queries. Moreover, if we make the following assumption:

Assumption 7. *There exists a deterministic function $\phi : \mathcal{X} \rightarrow \mathcal{L}$ such that for all (x, ℓ) drawn according to $\mathcal{D}$, we have $\ell = \phi(x)$.*

The set of all possible ranking functions that can be output by $\mathcal{R}$ will then depend only on the labeled set Z together with the final set of all the activated examples. Thus, as for the sample compression scheme, we have a reconstruction function associated with $\mathcal{R}$. We can therefore apply the same techniques as for the sample compression scheme to deduce risk bounds that will be valid for all ranker functions that can be reconstructed. The next results formalize this idea.

Starting from the whole set $X_{\mathcal{U}}$ of unlabeled, minimizing the generalization error of $\bar{f}$ can then be done by considering a subset of $X_{\mathcal{U}}^{(k)}$ of k elements of $X_{\mathcal{U}}$ for which the value of $d_c(\bar{f}(x'), \bar{f}_K^{cv}(x'))$ are maximal (Algorithm 1). Then, we can ask for the labels of $x' \in X_{\mathcal{U}}^{(k)}$ and learn $\bar{f}$ on $Z_\ell \cup Z_{\mathcal{U}}^{(k)}$, where $Z_{\mathcal{U}}^{(k)}$ denotes the labeled dataset, together with examples $x' \in X_{\mathcal{U}}$ that have been activated.

Algorithm 1. Active Learning strategy for ranking

```
Input :
  – A set of labeled $Z_\ell$ and unlabeled examples $X_{\mathcal{U}}$,
  – $k$ the number of examples to be activated, $T$ the maximum number of rounds.
Initialize:
  – $\forall j \in \{1, ..., K\}$ learn $\bar{f}_j^{cv}$ on $Z_j$, set $Z_{\mathcal{U}}^{(k)} \leftarrow \emptyset$ and $t \leftarrow 1$.
repeat
    – Learn $\bar{f}$ on $Z_\ell \cup Z_{\mathcal{U}}^{(k)}$,
    – Select a subset $X_{\mathcal{U}}^{(k)} \subset X_{\mathcal{U}} | \forall x' \in X_{\mathcal{U}}^{(k)}$ the value $d_c(\bar{f}(x'), \bar{f}_K^{cv}(x'))$ is maximal,
    – Ask for the labels of $x'$ for $x' \in X_{\mathcal{U}}^{(k)}$,
    – Remove $X_{\mathcal{U}}^{(k)}$ from $X_{\mathcal{U}}$ and reaffecte $Z_{\mathcal{U}}^{(k)}$, $Z_{\mathcal{U}}^{(k)} \leftarrow Z_{\mathcal{U}}^{(k)} \cup X_{\mathcal{U}}^{(k)}$, $t \leftarrow t + 1$
until convergence of $\sum_{x' \in X_{\mathcal{U}}} d_c(\bar{f}(x'), \bar{f}_K^{cv}(x')) \vee t > T$ ;
Output : $\bar{f}$
```

The interested reader may refer to [5] to have descriptions of existing supervised algorithms for learning ranking functions in step 1 of the algorithm.

In the following, we suppose that $Z = \{(x_1, \ell_1), (x_2, \ell_2), \ldots (x_n, \ell_n)\}$ and $X_{\mathcal{U}} = \{x'_{n+1}, x'_{n+2}, \cdots x'_{n+m}\}$. Moreover, we will also suppose that any single query of the ranking active learning algorithm corresponds to an activation of exactly k unlabeled data (for a parameter k fixed *a priori*), the total number of activated examples will then always be of the form $k \cdot t$ for some $t \in \mathbb{N}$. Thus, the compression set of any ranking function related to the algorithm $\mathcal{R}$ is the union of the set Z and a subset of size $k \cdot t$ of $X_{\mathcal{U}}$. The set of the labeled data is always in the compression set because the algorithm always consider Z. Now, one can define a prior distribution on the set of all outputs of $\mathcal{R}$ by defining a prior $P_{\mathbb{N}}$ on $\mathbb{N}$ together with, for each t that has weight in $P_{\mathbb{N}}$, a prior P_t on the set of all possible vector of indices of the forms $\boldsymbol{i} = (1, 2, \ldots, n, i_1, i_2, \ldots, i_{kt})$ with $i_j \in \{n+1, \ldots, n+m\}\ \forall j$ and $i_1 < i_2 < \ldots < i_{kt}$.

Most of the time, the prior $P_{\mathbb{N}}$ will have all its weights on the set $\{1, 2, \ldots T\}$ for some parameter T defined *a priori*. Moreover, unless m is too big, since the examples of $X_{\mathcal{U}}$ are supposed i.i.d., we will choose P_t as the uniform distribution under the constraint that the n first indices must always be chosen, that is $P_t(\boldsymbol{i}, t) = \binom{m}{kt}^{-1}$ for any $(\boldsymbol{i}, t)$. We will denote by $\mathcal{R}_{(\boldsymbol{i},t)}$ the corresponding ranking function. Under those assumptions, we have the following result.

Theorem 8. *Let $\mathcal{R}$ be any ranking active learning algorithm whose queries are all of size k. Let $P_{\mathbb{N}}$ and $\{P_t\}_{t\in\mathbb{N}}$ be the priors defined above. Finally, let $\bar{f}_K^{cv}$ be any stochastic ranking function (possibly defined by cross validation on a labeled dataset). Then $\forall t \in \mathbb{N}$ and $\forall\, \boldsymbol{i} = (1,2,\ldots,n,i_1,i_2,\ldots,i_{kt})$ we have:*

$$\mathbb{P}\left(\mathop{\mathbb{E}}_{x'\sim\mathcal{D}_{\mathcal{X}}} d_c\left(\mathcal{R}_{(\boldsymbol{i},t)}(x'),\ \bar{f}_K^{cv}(x')\right) \le \hat{\epsilon}_{Z\cup X_{\mathcal{U}}^{(kt)}} + \sqrt{\frac{\ln\binom{m}{kt} - \ln P_{\mathbb{N}}(t) + \ln(2/\delta)}{2(m-kt)}} \right) > 1 - \frac{\delta}{2}$$

where

$$\hat{\epsilon}_{Z\cup X_{\mathcal{U}}^{(kt)}} \overset{\text{def}}{=} \frac{1}{m-kt} \sum_{x'\in X_{\mathcal{U}}\setminus X_{\mathcal{U}}^{(kt)}} d_c\Big(\mathcal{R}_{(\boldsymbol{i},t)}(x'),\ \bar{f}_K^{cv}(x')\Big).$$

Proof: Similarly as in the proof of Lemma 4, for each $(\boldsymbol{i}, t)$, we use Hoeffding inequality [with $\delta := \frac{\delta . P_{\mathbb{N}}(t)\cdot P_t(\boldsymbol{i},t)}{2}$] and then apply the union bound.

Theorem 8, together with Theorem 1 and Lemma 4 gives us a generalization error bound (with level of confidence $1-\delta$) on any ranking function learned using this type of active learning procedure. Also it shows that any such $\mathcal{R}$ will converge to a ranking function that is at least as good as the cv-ranking function $\bar{f}_K^{cv}$ even if $\mathcal{R}$ is constructing *deterministic* ranking function only. Moreover, it is clear from the definition of the divergence d_c that, for any already constructed ranking function $\bar{f}$, the corresponding label of any unlabeled data for which the value of d_c is maximal will gives rise to one of those three situations: (1)–the c-value of $\bar{f}_K^{cv}$ is "good" and the one of $\bar{f}$ is "bad",(2)–the c-value of $\bar{f}_K^{cv}$ is "bad" and the one of $\bar{f}$ is "good", or (3)–both are "bad". Clearly a query in the situation (1) or (3) points out a weakness of $\bar{f}$. From an active learning point of view, this is something that is suitable. Note also that, if $\bar{f}_K^{cv}$ has a low risk, then situations (1) and (3) would be more likely to occur. This is the central idea that underlies our proposed ranking active learning algorithm[2].

[2] As in [7], we choose to base our approach on the generally accepted assumption that CV methods give good results.

4 Experiments

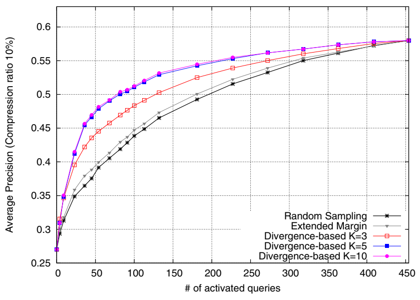
We compared the proposed selective sampling scheme (denoted by *divergence-based* in the following) with the random sampling and the extended margin heuristic of [2] adapted to partial orderings. The reference supervised learning algorithm we used to train the randomized ranking functions is the same as in [1]. For a ranking function $\bar{f}$, we used the c_{Rloss} risk function to learn a linear combination of weights for its associated scoring function f. We employed the divergence measure $d_{c_{Rloss}}$ introduced in proposition 6 to activate queries from $X_{\mathcal{U}}$ and conducted experiments on the Information Retrieval tasks of text-summarization (TS) and question/answering (QA). Performances for TS and QA are resectively averaged over 10 and 25 random splits[3] of training/unlabeled pool/test sets. For the text summarization, the queries we aim to activate are documents for which the list of sentences appearing in its summary is demanded. For QA, queries are questions and for each activated question we ask for passages containing its answer.

4.1 Real Life Applications

Automatic Text Summarization. Automatic Text Summarization (ATS) systems are mostly designed to help users to quickly find a needed information. Most studies consider the task of text summarization as the extraction of text spans (typically sentences) from the original document. Extractive approaches transform the problem of abstracting a text into a simpler problem of *ranking* sentences according to their relevance of being part of the document summary. These approaches have proven to be effective in producing summaries. To rank text spans from a document, most previous studies combine statistical or linguistic features characterizing each sentence in a text. A combination of these features is finally used to order the spans. In this work we considered 4 statistical features borrowed from [1]. For ATS, we compared performance on the `WIPO` collection[4] used in [1]. In our experiments, we have chosen 1000 documents at random from this corpus and removed documents having less than 2 words in their title, and those composed of 1 sentence arguing that a sentence is not sufficient to summarize a scientific document. In total we gathered 854 documents and their associated summaries and Train/Unlabeled/Test splits are respectively 60/394/400 and 30/394/400 in each experiment. For the evaluation we followed the state-of-the-art by comparing the extract of the system for each document in the test collection with the desired summary obtained from its abstract by an alignment technique [10]. We used the *average precision* measure by fixing a 10% compression ratio, that is for each document in the collection, we computed the average number of sentences appearing in its summary in a high ranked sentence list of a length equal to 10% of the document's size.

[3] We conducted the Wilcoxon rank sum tests to decide of the significance of results for Q/A and thus ran more experiments in this case.

[4] http://www.wipo.int/ibis/datasets/index.html



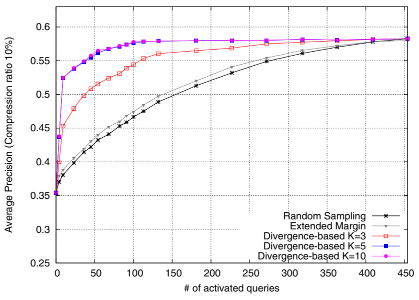


Fig. 1. Average Precision at 10% compression ratio versus the number of activated queries for random, extended margin and divergence-based strategies. Results are averaged over 10 randomly splits of training /unlabeled pool/test sets. For the same number of documents in the unlabeled pool (394) and test set (400), performance are plotted for 30 (left) and 60 (right) documents in the training set.

Passages retrieval for Question/Answering. QA systems address the problem of finding exact answers to natural language (NL) questions. In order to reduce the amount of information, QA systems apply successively two different modules. A *document retrieval* module first identifies spans (documents or paragraphs) that are likely to contain an answer to the asked question. Then an *answer extraction* module extracts the desired answer by performing a deeper NL analysis of the retrieved spans. Here we consider the *document retrieval* module of a QA system. For Q/A, we compared performance on the `TREC-11` question/answering track and the `Aquaint` collection[5] by evaluating the $a@n$ measure which is the proportion of questions in the test set, for which the answer is contained in the first n retrieved passages. Among the 500 questions in the collection, we discarded 193 having no answer in the top 100 passages retrieved by the search engine. For each question, we followed the methodology developed in [12] to convert the retrieved passages into 117 dimensional score features by applying a conventional search engine which assigns a series of scores to each paragraph in the collection. In this setting, Train/Unlabeled/Test splits are 30/121/156.

4.2 Empirical Results

Figure 1, plots the performance of the divergence-based, extended margin and random strategies for the ATS task for different numbers of randomized ranking functions and for different splits of the training set (training sets of size 30 documents in figure 1. left and 60 documents in figure 1. right - the size of unlabeled data and test sets are kept fixed). We see in both cases that divergence-based strategy has a real advantage over random sampling and the extended margin heuristic. The low performance of the extended margin can be explained by the fact that an accurate scoring function should be able to rank relevant

[5] http://trec.nist.gov/data/qa/t2002_qadata.html

Table 1. $a@n$ in % for the divergence-based, random and the extended margin strategies. Results are averaged over 25 randomly splits of training/unlabeled/test sets.

n	Strategy	# of activated queries ($K = 5$)					
		0	4	8	24	60	121
		$a@n$					
	Divergence-based		**39.7**	**41.6**	**44.8**	**45.5**	
5	Extended Margin	35.2	38.6$^{\rightharpoonup}$	40.1$^{\rightleftharpoons}$	41.7$^{\rightleftharpoons}$	44.2$^{\rightharpoonup}$	46.1
	Random		38.0$^{\rightleftharpoons}$	39.6$^{\rightleftharpoons}$	41.3$^{\rightleftharpoons}$	43.8$^{\rightleftharpoons}$	
	Divergence-based		**51.2**	**52.5**	**56.4**	**57.6**	
10	Extended Margin	46.1	49.9$^{\rightharpoonup}$	51.7$^{\rightharpoonup}$	54.2$^{\rightleftharpoons}$	56.9$^{\rightharpoonup}$	57.7
	Random		49.5$^{\rightharpoonup}$	51.2$^{\rightleftharpoons}$	53.7$^{\rightleftharpoons}$	56.2$^{\rightleftharpoons}$	
	Divergence-based		**57.7**	**62.8**	**67.3**	**68.6**	
20	Extended Margin	53.8	56.8$^{\rightharpoonup}$	60.1$^{\rightleftharpoons}$	64.9$^{\rightleftharpoons}$	67.3$^{\rightleftharpoons}$	69.2
	Random		56.2$^{\rightleftharpoons}$	59.5$^{\rightleftharpoons}$	64.4$^{\rightleftharpoons}$	66.9$^{\rightleftharpoons}$	

sentences above irrelevant ones, but we should not expect this scoring function to be confident about the relative ranks of two relevant (or two irrelevant) sentences.

In the case where the randomized ranking functions (RRF) have sufficiently been trained (figure 1. right) we note that after querying of about 50 instances with 5 or 10 RRF, the final ranking function has approximately the same level of performance as when the ranking function is learnt on all the labeled data, together with all the unlabeled data and their corresponding labels.

The convergence rate of the performance of deterministic ranking functions is however lower with a smaller number of RRF. This might be due to the fact that the split of the training set on different cross-validation sets (on which each RRF is trained) is larger with a higher number of RRF. Thus the divergence-based strategy appears to be most effective if there is a reasonable training size for learning and not a too small number of RRFs.

Table 2, shows our second investigation for the Q/A task. We notice the same effect of the divergence-based strategy compared to random sampling and extended margin heuristic than for ATS. Indeed, with only 30 questions in the training set, the divergence-based strategy outperforms the random and extended margin strategies for different values of n. We conducted Wilcoxon rank sum tests with a p-value threshold of 0.05 to decide if the results in Table 2 are significant. The symbols $\rightharpoonup$ and $\rightleftharpoons$ indicate the cases where Extended-Margin and Random strategies are significantly worse than the Divergence-based strategy respectively as a one and two-tailed tests.

5 Conclusion

We proposed an active learning strategy for learning ranking functions. The theoretical analysis and the definition of the notion of disagreement between two ranking functions lead to a novel active learning strategy that shows good empirical performance on real world applications. To the best of our knowledge, this strategy is the first one that can be applied to general cases of ranking.

Moreover, experimental results show that, in practice, the derived active learning strategy is highly effective.

The major theoretical weakness of our work is that we consider the randomized ranking function built with CV sets as the reference, while it is certainly not a perfect ranker. Indeed, the generalization error bound can never be better than the generalization error on the CV-ranker. On the other hand, our analysis comes with several advantages: (1) we actually tend to minimize a generalization error bound, which is a challenging issue in label ranking. (2) The bound remains valid during the active learning process. (3) Our proposed sample compression approach gives a general framework on which one can base other ranking active learning algorithms. Finally, it appears empirically that the divergence-based strategy suggested by the bound significantly reduces the required number of labeled examples. Therefore, while the proposed strategy is, indeed, mainly heuristic, it needed the bound for both the definition of the notion of disagreement, and for the idea of using another ranking function as reference. Possible improvements of our theory include the study of how the ranking function used as reference could evolve as we query for more labels, which would enable the generalization error bound of the learned function become better than the initial error of the randomized ranking function obtained with CV-sets.

References

1. M. Amini, N. Usunier, and P. Gallinari. Automatic text summarization based on word clusters and ranking algorithms. In *Proc. of the* 27th *ECIR*, 2005.
2. Klaus Brinker. Active learning of label ranking functions. In *Proc. of* 21st *International Conference on Machine learning*, 2004.
3. Klaus Brinker, Johannes Fürnkranz, and Eyke Hüllermeier. Label ranking by learning pairwise preferences. In *Journal of Machine learning Research*, 2005.
4. Olivier Chapelle. Active learning for parzen window classifier. In *AI STATS*, 2005.
5. Koby Crammer and Yoram Singer. A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, 3(6):1025 – 1058, 2003.
6. Sally Floyd and Manfred Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):269–304, 1995.
7. Matti Kääriäinen. Generalization error bounds using unlabeled data. In *Proceedings of the* 18th *Annual Conference on Learning Theory*, pages 127–142, 2005.
8. François Laviolette, Mario Marchand, and Mohak Shah. Margin-sparsity trade-off for the set covering machine. *Proc. of the* 16th *ECML*, pages 206–217, 2005.
9. N. Littlestone and Manfred Warmuth. Relating data compression and learnability. *Technical Report, University of California*, 1986.
10. Daniel Marcu. The automatic construction of large-scale corpora for summarization research. In *Proceedings of the 22nd ACM SIGIR*, pages 137–144, 1999.
11. Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, 2002.
12. N. Usunier, M. Amini, and P. Gallinari. Boosting weak ranking functions to enhance passage retrie- val for question answering. In *IR4QA-workshop, SIGIR*, 2004.

Combinatorial Markov Random Fields

Ron Bekkerman[1], Mehran Sahami[2], and Erik Learned-Miller[1]

[1] Department of Computer Science, University of Massachusetts, Amherst MA 01002
{ronb, elm}@cs.umass.edu
[2] Google Inc., 1600 Amphitheatre Parkway, Mountain View CA 94043
sahami@google.com

Abstract. A *combinatorial random variable* is a discrete random variable defined over a combinatorial set (e.g., a power set of a given set). In this paper we introduce *combinatorial Markov random fields (Comrafs)*, which are Markov random fields where some of the nodes are combinatorial random variables. We argue that Comrafs are powerful models for unsupervised and semi-supervised learning. We put Comrafs in perspective by showing their relationship with several existing models. Since it can be problematic to apply existing inference techniques for graphical models to Comrafs, we design two simple and efficient inference algorithms specific for Comrafs, which are based on combinatorial optimization. We show that even such simple algorithms consistently and significantly outperform Latent Dirichlet Allocation (LDA) on a document clustering task. We then present Comraf models for semi-supervised clustering and transfer learning that demonstrate superior results in comparison to an existing semi-supervised scheme (constrained optimization).

1 Introduction

Three decades have passed since McGurk and MacDonald published their work [1] revealing the multi-modal nature of speech perception: sound and moving lips compose one system, so to better process audio signals, an audio/video interaction should be modeled. Since then, machine learning researchers have widely exploited data multi-modality, using many approaches, such as multi-modal neural networks [2], multivariate information bottleneck [3], and more recently multi-view expectation maximization [4] and multi-way distributional clustering [5].

Multi-modality plays an important role in unsupervised learning; given no class labels, learning results mostly depend on data representation. For example, one cannot expect a system to cluster documents by topic if only their *lengths* are given. However, when documents are represented as bags of *words*, meaningful clustering can be built. Moreover, if in addition to bags of words, another representation based on documents' *authorship* is obtained, the two modalities show different angles of documents' topicality and thus provide useful structure to documents' representation that can be leveraged during learning.

In many real-world situations multiple modalities of data can be easily observed. Indeed, consider an email inbox, where in addition to message bodies, one can observe subject lines, names of senders and recipients, markup items, attachments etc. Nevertheless, early multi-modal systems rarely went beyond two modalities (documents/words, audio/video, genes/samples, etc.). Currently, with the availability of massive computational power, using more than two modalities is a feasible and attractive research opportunity.

In many cases, each modality interacts differently with the others, with some interactions being negligibly weak. Hence, when many modalities are available (each of which having its own interaction pattern with the others), we can construct a graph representation of the modalities and their interactions. In previous work, Friedman et al. [3] use a Bayesian network to define *input* and *output* spaces in the multivariate Information Bottleneck; Bekkerman et al. [5] use a *pairwise interaction graph* to describe dependencies between the modalities. In both those studies, the graph is an auxiliary, descriptive component of the model.

Our approach uses the *Markov random field (MRF)* formalism (see, e.g., [6]). In Section 2, we propose a *combinatorial Markov random field (Comraf)*, which allows us to model each modality of the data as a single *combinatorial random variable* in the MRF graph, with edges representing probabilistic interactions between the modalities. Comraf models are (a) compact – the number of nodes in a Comraf is the order of the number of modalities, which allows for easier *model learning*; and (b) data-driven – no generative assumptions are made, which minimizes the model's bias. The main contribution of this work is to present a general framework for multi-modal learning, which is based on the *most probable explanation (MPE) inference* in a Comraf. For unsupervised learning, we show that Comrafs are a general framework that subsumes a number of existing models as special cases (Section 3) and allows us to also explore new modeling possibilities for other learning tasks, such as semi-supervised clustering and transfer learning (Section 4). We show that Comrafs lend themselves to naturally modeling multi-model data, obtaining strong empirical results (Section 5).

2 Combinatorial MRFs

Definition 1. *A* combinatorial random variable *(or* combinatorial r.v.*) X^c is a discrete random variable defined over a combinatorial set.*

A *combinatorial set* in mathematical parlance means a set of all subsets, partitionings, permutations etc. of a given finite set. To capture this intuition, we define a finite set A as *combinatorial* if its size is exponential with respect to another finite set B, i.e. $\log|A| = O(|B|)$. As an example, a combinatorial r.v. X^c can be defined over all the outcomes of *lotto 6 of 49*, in which 6 balls are selected from 49 enumerated balls to produce an outcome of the lottery. In this case, set B consists of 49 balls, while set A consists of $\binom{49}{6}$ possible choices of 6 balls from B. In a *fair* lottery, the distribution of X^c is uniform: each outcome is drawn with probability $1/\binom{49}{6}$. However, in an unfair lottery, some outcomes are more probable than others.

From the theoretical perspective, a combinatorial r.v. behaves exactly as an ordinary discrete random variable with a finite domain. However, from the practical point of view, a combinatorial r.v. is different: in most real-world cases, the event space of X^c is so large that the distribution $P(X^c)$ cannot be explicitly specified. Moreover, the MPE task for combinatorial r.v.'s can be computationally hard. Considering an unfair lottery example, in which the distribution of X^c is flat (close to uniform), say, the probability of value $\{7, 23, 29, 35, 48, 49\}$ is 0 and the probability of value $\{4, 18, 28, 37, 39, 43\}$ is $2/\binom{49}{6}$, while the rest of the values still have the probability $1/\binom{49}{6}$. An exponentially long sampling process is required to detect the most probable value.

It is easy to come up with other examples of combinatorial r.v.'s: all the possible translations of a sentence, orderings in a ranked list of retrieved documents, etc. In this paper, we consider combinatorial r.v.'s over all *partitionings* of a given set. In most complex systems random variables interact with each other. Such interactions are usually represented in a directed or undirected graphical model. In multi-modal systems, which are the focus of our paper, interactions between modalities are symmetric, so the undirected case is more appealing.

A *Markov random field (MRF)* is a model (G, P), where G is an undirected graph whose nodes $\mathbf{X} = \{X_1, \ldots, X_m\}$ represent random variables and whose edges $\mathbf{E}$ denote interactions between these variables. P is a joint probability distribution defined over $\mathbf{X}$. The *Markov property* holds in this model.

Definition 2. *A* combinatorial Markov random field (Comraf) *is an MRF, at least one node of which is a combinatorial random variable.*

2.1 MPE Inference in Comrafs

An *inference* procedure in MRFs answers questions about the model, such as what is the most likely assignment $\mathbf{x}^* = \{x_1^*, \ldots, x_m^*\}$ to variables $\{X_1, \ldots, X_m\}$ (i.e. MPE). Naturally, answering most of such questions is NP-hard since it potentially requires considering every possible assignment. Thus, most inference techniques fall into the category of approximation methods.

The Hammersley-Clifford theorem [7] states that the joint distribution over nodes of an MRF is a Gibbs distribution: $P(\mathbf{x}) = \frac{1}{Z_{\mathbf{f}}} \exp \sum_i f_i(\mathbf{x})$, where $f_i(\mathbf{x})$ are arbitrary potential functions defined over cliques in G, and $Z_{\mathbf{f}}$ is a normalization factor called a partition function. Unsupervised learning problems are usually solved using the *maximum likelihood (ML)* framework (see, e.g. [8]), where model parameters that best explain the data are estimated. Most ML methods deal with approximating $Z_{\mathbf{f}}$, which is generally a hard task, because $Z_{\mathbf{f}}$ depends on the particular choice of f_i's and is a sum over all the possible configurations. However, in our setting the potentials f_i are fixed for each clique, the partition function $Z_{\mathbf{f}}$ becomes a constant, so $\log P(\mathbf{x}) \propto \sum_i f_i(\mathbf{x})$. Thus, for MPE, it is sufficient to directly optimize:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} P(\mathbf{x}) = \arg\max_{\mathbf{x}} \sum_i f_i(\mathbf{x}). \tag{1}$$

This relatively simple formulation is still quite powerful, as it allows us to use a wide variety of potential functions that might be too complicated to use in the general setting where the partition function still needs to be approximated.

3 Unsupervised Learning with Comrafs

To illustrate the power of the Comraf framework, we initially focus on unsupervised learning (e.g., data clustering) and show how several existing clustering schemes are specific instances on Comrafs. Let $s_1, s_2, ..., s_N$ be a dataset of N i.i.d. samples drawn from some discrete distribution. Let $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ be the set of n unique values comprising the event space from which samples s_i are drawn. We now define a random variable X such that $P(X = x_i)$ is given by the empirical frequencies of samples with value x_i in the dataset (i.e., X has a multinomial distribution estimated using maximum likelihood).

Define a *hard clustering* $\tilde{x}^c$ to be a partitioning of $\mathcal{X}$. Let $\mathcal{X}^c = \{\tilde{x}_1^c, \tilde{x}_2^c, ..., \tilde{x}_K^c\}$ be the combinatorial set of all K partitionings of $\mathcal{X}$, where K is exponential in the size of $\mathcal{X}$. We will refer to the subsets of the j-th partitioning $\tilde{x}_j^c$ as $\{\tilde{x}_{j,1}, \tilde{x}_{j,2}, ..., \tilde{x}_{j,k_j}\}$. That is, the first subscript is the index of the particular partitioning, and the second subscript is the subset within that partitioning.

Define $\tilde{X}_j$ to be a random variable over the subsets (*clusters*) in a partitioning $\tilde{x}_j^c$, with the probability of a selected cluster being the probability of choosing any one of its elements, that is, $P(\tilde{X} = \tilde{x}_{j,i}) = \sum_{x \in \tilde{x}_{j,i}} P(x)$. Finally, define $\tilde{X}^c$ to be a combinatorial r.v. with the event space $\mathcal{X}^c$. In this work, we shall use parallel notation for different modalities of data, replacing the "x's" in the above notation with variables appropriate for the data source. For example, w_i would represent a specific word in a dataset, $\tilde{w}^c$ a partitioning of words, and so on.

Interactions between combinatorial r.v.'s (possibly, with ordinary r.v.'s) are represented by edges in a Comraf graph. To use the objective from Equation (1), we should choose relevant cliques in the Comraf graph and define potential functions over these cliques. To make the inference feasible, we consider only the smallest cliques, i.e. adjacent pairs. Since our inference objective allows using complicated potential functions (see Section 2.1), we use the *mutual information (MI)* between r.v.'s defined over values of adjacent nodes. Let $\tilde{x}_i^c$ and $\tilde{y}_j^c$ be such values (particular partitionings of two modalities). A potential is then defined:

$$f(\tilde{x}_i^c, \tilde{y}_j^c) = I(\tilde{X}_i; \tilde{Y}_j) = \sum_{i',j'} P(\tilde{x}_{i,i'}, \tilde{y}_{j,j'}) \log \frac{P(\tilde{x}_{i,i'}, \tilde{y}_{j,j'})}{P(\tilde{x}_{i,i'})P(\tilde{y}_{j,j'})}.$$

Our motivation for choosing MI as a potential function is as follows: a linear combination of MI terms has traditionally been used as a clustering criterion, both in 1-way clustering methods, such as Information Bottleneck (IB) [9], and in 2-way methods [10]. Friedman et al. [3] generalize the IB clustering criterion to a multivariate case: in place of MI, they use Multi-Information, which naturally factors over a directed graphical model. With little effort, we can show that Multi-Information also factors over a tree-structured undirected graphical model, reducing to a sum of pairwise MI terms defined over edges of the tree. However, in the case of an arbitrary Comraf graph, the Multi-Information can only be *approximated* by a sum of pairwise MI terms. Estimating the quality of such an approximation remains an open question that we will address in future work. Presently, we show how existing models can be cast as Comrafs:

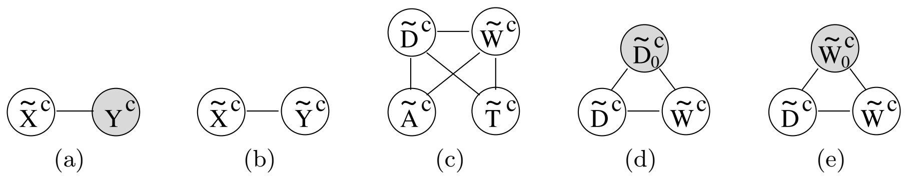


Fig. 1. Comraf graphs for: (a) hard version of Information Bottleneck; (b) information-theoretic co-clustering; (c) 4-way MDC; (d) semi-supervised clustering; (e) clustering with transfer learning.

A hard version of Information Bottleneck [9] is a special case of a Comraf. In IB, a clustering $\tilde{x}^{c*}$ is constructed that maximizes information about a variable Y (and minimizes information about X), i.e. $\tilde{x}^{c*} = \arg\max_{\tilde{x}^c_j}(I(\tilde{X}_j;Y) - \beta I(\tilde{X}_j;X))$, where β is a Lagrange multiplier. The compression constraint $I(\tilde{X}_j;X)$ can be omitted if the number of clusters is fixed: $|\tilde{x}^c_j| = k$. Consider graph G in Figure 1(a), where a shaded Y^c represents an *observed* variable.[1] On the only clique in G we define one potential which is the mutual information $I(\tilde{X}_j;Y)$. The MPE objective is then: $\tilde{x}^{c*} = \arg\max_{\tilde{x}^c_j} P(\tilde{x}^c_j, y^c) = \arg\max_{\tilde{x}^c_j} I(\tilde{X}_j;Y)$.

Information-theoretic co-clustering [10] is a task of simultaneously clustering documents $\mathcal{X}$ and their words $\mathcal{Y}$, while minimizing the information loss $I(X;Y) - I(\tilde{X}_j, \tilde{Y}_j)$ under the constraint $|\tilde{x}^c_j| = k_1$ and $|\tilde{y}^c_j| = k_2$. Note that $I(X;Y)$ is a constant for a given dataset. This scheme is a special case of a Comraf as well: given graph G in Figure 1(b), in analogy to the Comraf model of IB, we define the only potential $I(\tilde{X}_j;\tilde{Y}_j)$. Then the information-theoretic co-clustering can be represented as an MPE procedure in the Comraf: $(\tilde{x}^{c*}, \tilde{y}^{c*}) = \arg\max_{\tilde{x}^c_j,\tilde{y}^c_j} P(\tilde{x}^c_j, \tilde{y}^c_j) = \arg\max_{\tilde{x}^c_j,\tilde{y}^c_j} I(\tilde{X}_j;\tilde{Y}_j)$.

Multi-way distributional clustering (MDC) [5] is a generalization of [10], where the data has a number of interdependent modalities (such as documents, words, authors, titles, etc.). Interactions between the modalities are represented using a *pairwise interaction graph* that has no probabilistic interpretation. Actually, these interactions can be represented in a Comraf, where the modalities are combinatorial r.v.'s $\tilde{\mathbf{X}}^{\mathbf{c}} = \{\tilde{X}^c_1, \dots, \tilde{X}^c_m\}$ that are nodes in a graph G with edges $\mathbf{E}$. The MPE scheme is then:

$$\tilde{\mathbf{x}}^{\mathbf{c}*} = \arg\max_{\tilde{\mathbf{x}}^{\mathbf{c}}_j} P(\tilde{\mathbf{x}}^{\mathbf{c}}_j) = \arg\max_{\tilde{\mathbf{x}}^{\mathbf{c}}_j} \sum_{(\tilde{X}^c_i, \tilde{X}^c_{i'}) \in \mathbf{E}} I(\tilde{X}_{i,j}; \tilde{X}_{i',j}). \tag{2}$$

Here the first subscript is the index of a combinatorial r.v., while the second subscript is the index of this r.v.'s particular value (a partitioning). Equation (2) is equivalent to the MDC objective proposed in [5]. An example Comraf graph for a 4-way MDC (that corresponds to simultaneously clustering documents, words, authors and titles) is shown in Figure 1(c).

[1] For discussion on observed variables see Section 4.

We note that by casting IB and Information-theoretic co-clustering as Comrafs, we not only show the generality of the framework, but also demonstrate that the generalization of these methods to additional modalities of data is naturally accomplished via Comrafs. In the case of MDC, viewing this model as a Comraf allows us to consider generalizations to other tasks, such as semi-supervised learning via the introduction of observed variables in the model.

3.1 Clustering as Inference in a Comraf

Due to unique characteristics of combinatorial r.v.'s, it is problematic to apply existing inference algorithms to Comrafs. Here we propose an inference method specific for Comrafs, which is based on combinatorial optimization. We then craft two simple and efficient inference algorithms based on the proposed method.

Given that a variable X has n values that are clustered into k clusters, the combinatorial r.v. $\tilde{X}^c$ has k^n values, all of which can be represented as points in an n-dimensional lattice L: a point $\tilde{x}^c = (i_1, i_2, \ldots, i_n)$ corresponds to the fact that value x_1 of X belongs to cluster i_1, value x_2 belongs to cluster i_2, ..., value x_n belongs to cluster i_n.[2] In the lattice L there is a (possibly non-unique) point $\tilde{x}^{c*} = (i_1^*, i_2^*, \ldots, i_n^*)$ which is most likely. Since the lattice consists of an exponential number of points, the task of finding the most likely point can be computationally hard. We will attempt to approximate the solution using a quasi-random walk in the lattice. Let us start with two definitions.

Definition 3. *A* transaction $(\ldots, i_j, \ldots) \rightarrow (\ldots, i'_j, \ldots)$, *where* $i_j \neq i'_j$, *is an elementary operation in traversing the lattice* L *of possible clusterings, in which* x_j *is moved from cluster* i_j *to cluster* i'_j.

Definition 4. *A* path *in* L *is a sequence of transactions. A path is called* advantageous *if it leads to a more likely clustering, otherwise it is* disadvantageous.

Note that we can view both splits and mergers of clusters as transactions. A split of a cluster $i_{j'}$ is a transaction $(\ldots, i_{j'}, \ldots) \rightarrow (\ldots, i'_{j'}, \ldots)$, where $\exists j \neq j' : i_{j'} = i_j$ and $\forall j \neq j' : i'_{j'} \neq i_j$. That is, cluster $i_{j'}$ contained at least two elements (x_j and $x_{j'}$), one of which ($x_{j'}$) has been transferred into a newly created cluster $i'_{j'}$. A merger of clusters $i_{j'}$ and $i'_{j'}$ is a transaction $(\ldots, i_{j'}, \ldots) \rightarrow (\ldots, i'_{j'}, \ldots)$, where $\exists j \neq j' : i'_{j'} = i_j$ and $\forall j \neq j' : i_{j'} \neq i_j$, i.e. cluster $i_{j'}$ contained only one element that has been added to the existing cluster $i'_{j'}$ so that the cluster $i_{j'}$ does not exist anymore. These operations will help us to represent both agglomerative (bottom-up) and divisive (top-down) clustering schema as inference in Comrafs.

By applying splits, mergers and other transactions, we construct paths in the lattice of possible clusterings. Thus, to approximate the MPE of a combinatorial r.v. $\tilde{X}^c$, we apply the simplest, greedy combinatorial optimization algorithm—*hill climbing*:[3] we attempt to construct a path in L which is as advantageous as possible on each step, given the available computational resources.

[2] For now, we consider only *hard* clustering, where $P(i_j|x_j) = 1$ for a value x_j assigned to cluster i_j. Generalization of the Comraf model to *soft* clustering is our future task.

[3] More complex algorithms, such as Branch and Bound, while applicable, may be infeasible to use because of their high computational complexity.

Algorithm 1. A template of an MPE procedure in Comrafs.

```
Input:
  $G$ – Comraf graph of nodes $\{\tilde{X}_1^c, \dots, \tilde{X}_m^c\}$ and edges $\mathbf{E}$
  $P(X_i, \dots, X_m)$ – joint probability distribution of data, factorized over $G$
  $l$ – number of optimization iterations
Output:
  Most likely $\tilde{x}_{1,l}^c, \dots, \tilde{x}_{m,l}^c$

  Initialization:
  for $i = 1, \dots, m$ do
    Select a point in $L_i$ to be an initial value $\tilde{x}_{i,0}^c$ of $\tilde{X}_i^c$
  Compute the initial joint $P(\tilde{X}_{1,0}, \dots, \tilde{X}_{m,0})$, factorized over $G$
  Main loop:
  for $j = 1, \dots, l$ do
    Select variable $\tilde{X}_{i'}^c$ for optimization
    Construct advantageous path $\left(\tilde{x}_{i',j-1}^c \rightarrow \tilde{x}_{i',j}^c\right)$ in $L_{i'}$
    For all $i \neq i'$ do $\tilde{x}_{i,j}^c = \tilde{x}_{i,j-1}^c$
```

In a Comraf that has more than one combinatorial r.v., the Comraf inference algorithm becomes a variation of the *Iterative Conditional Mode (ICM)* method [11]. ICM optimizes each node of an MRF iteratively (in a round-robin fashion), given its Markov blanket. At an ICM iteration applied to a node $\tilde{X}_i^c$, the MPE objective from Equation (2) with $O(|\mathbf{X}|^2)$ terms is reduced to:

$$\tilde{x}_i^{c*} = \arg\max_{\tilde{x}_{i,j}^c} \sum_{i':\ (\tilde{X}_i^c, \tilde{X}_{i'}^c) \in \mathbf{E}} I(\tilde{X}_{i,j}; \tilde{X}_{i',j}) \tag{3}$$

that sums over only $O(|\mathbf{X}|)$ neighbors of $\tilde{X}_i^c$.

A template pseudo-code for the MPE approximation in a Comraf is given in Algorithm 1. For each combinatorial r.v. $\tilde{X}_i^c$ in the Comraf, we first select and fix its initial value as a point in the lattice L_i. We then round-robin over each $\tilde{X}_i^c$, for which we search for an advantageous path in L_i. When this path is constructed, we fix its destination point to be a new value of $\tilde{X}_i^c$ and move to another node. We repeat this procedure l times. To transform this template into an actual algorithm, we need to make the following choices:

– Selecting initial values for each combinatorial r.v. in the Comraf. Either random assignment of data points into k clusters or an assignment of all data points into one cluster are two simple choices, while other methods (such as those incorporating prior knowledge) are possible.
– Determining an ordering for variables in the optimization procedure. One obvious approach is a plain or weighted round-robin, but more sophisticated choices can also be made.
– Constructing an advantageous path in L. A greedy method would increase the likelihood with each transaction, leading to a local maximum of the objective. However, we could also consider a stochastic approach in which

some disadvantageous transactions are tolerated assuming that they may lead closer to the global maximum.

The latter point is of especial importance. We propose two algorithms for constructing advantageous paths. In both, we first split or merge clusters in order to meet the traditional requirement on the number of clusters. Then, in the *sequential* algorithm, we iterate over each data point in some ordering, and assign it into its best cluster (the one for which the objective is maximized). In the *randomized* algorithm, we repeat the following step a predefined number of times:[4] we uniformly at random select a data point x_i and a cluster $\tilde{x}_j$, and assign x_i into $\tilde{x}_j$ if this transaction improves the objective.

4 Semi-supervised and Transfer Learning with Comrafs

The Comraf model is a convenient framework for performing semi-supervised and transfer learning. Prior to presenting details of particular Comrafs, let us define the concepts of hidden and observed states in the Comraf model. A combinatorial r.v. is *hidden* if it can take any value from its event space. A combinatorial r.v. is *observed* if its value is preset and fixed.

4.1 Semi-supervised Clustering with Comrafs

Semi-supervised clustering is a clustering task that takes advantage of labeled examples. Usually, semi-supervised clustering is performed when the number of available labeled examples is not sufficient to construct a good classifier (e.g., the constructed classifier would overfit), or when the the labeled data is noisy or skewed to a few classes. Assuming that *most* of the labeled data is accurate, our goal is to incorporate it into the (unsupervised) Comraf model.

In this paper, we consider a uni-labeled case: each labeled data point $x_i|_{i=1}^n$ belongs to one ground truth category $t_j|_{j=1}^k$. We propose an *intrinsic* Comraf approach for incorporating labeled data into clustering (by introducing observed nodes to a Comraf graph), and compare it with an existing *constrained optimization* scheme.

Intrinsic Approach. Comrafs offer an elegant method for incorporating labeled data, which does not require any significant changes in the model. First, note that labels define a natural partitioning of the labeled data: for each label t_j let $\tilde{x}_{0j}$ be a subset of $\mathcal{X}$ labeled with t_j, i.e. $\tilde{x}_{0j} = \{x_i | t_i = t_j\}$. We now define a r.v. $\tilde{X}_0$ over the partitioning $\tilde{x}_0^c = \{\tilde{x}_{0j} | j = 1, \ldots, k\}$, and we also define a combinatorial r.v. $\tilde{X}_0^c$ over all the possible partitionings of the set $\mathcal{X}$. Since the partitioning $\tilde{x}_0^c$ is *given* to us, the variable $\tilde{X}_0^c$ is *observed*, with $\tilde{x}_0^c$ being its fixed value. Observed combinatorial random variables appear shaded on a Comraf graph – see, e.g., Figure 1(c). The objective function from Equation (3) and the MPE inference procedure remain unchanged (with the only difference being

[4] Equal (for fair comparison) to the number of iterations in the sequential algorithm.

that there is no need in optimizing the observed nodes): at each ICM iteration the current node is optimized with respect to the *fixed* values of its neighbors, whereas the values of the observed nodes are fixed by definition.

Constrained optimization. Wagstaff and Cardie [12] perform semi-supervised clustering with two types of boolean constraints. The *must-link* constraint ml equals 1 if two equally labeled data points are assigned into different clusters; the *cannot-link* constraint cl equals 1 if two differently labeled data points are assigned into the same cluster. A clustering objective function incorporates the constraints, e.g. in Comrafs (Equation (3)) for each combinatorial r.v. $\tilde{X}_i^c$ it is:

$$\tilde{x}_i^{c*} = \arg\max_{\tilde{x}_{i,j}^c} \sum_{i':\ (\tilde{X}_i^c, \tilde{X}_{i'}^c) \in \mathbf{E}} I(\tilde{X}_{i,j}; \tilde{X}_{i',j}) - \sum_{i'} w_{i,i'}\ ml_{i,i'} - \sum_{i'} w_{i,i'}\ cl_{i,i'},$$

where $w_{i,i'}$ are weights that we set at $+\infty$, which means that all constraints must be satisfied. Note that in a general case we are free to choose any non-negative weights. In order to fairly compare two semi-supervised methods, for both of them we must use the same underlying clustering algorithm. We use the sequential MPE inference algorithm (see Section 3.1) in both cases.

4.2 Transfer Learning with Comrafs

Transfer learning is the problem of applying the knowledge learned in one task to effectively solve another learning task. In this paper, we represent the acquired knowledge as a partitioning $\tilde{y}_0^c$ pre-built for data $\mathcal{Y}$ that can be used for constructing a partitioning $\tilde{x}^c$ of data $\mathcal{X}$. We note that the intrinsic scheme for semi-supervised clustering presented above allows us to directly use labeled data not from $\mathcal{X}$ but rather from *another* collection $\mathcal{Y}$. Thus, in analogy to the semi-supervised case, we introduce an observed combinatorial r.v. $\tilde{Y}_0^c$ with a fixed value $\tilde{y}_0^c$. During the inference process, we construct $\tilde{x}^{c*}$ that maximizes information about $\tilde{y}_0^c$, while applying the same objective function as in Equation (3).

5 Experimentation

Following [10], we use *micro-averaged accuracy* for evaluation of our clustering methods. Let $\tilde{x}^c$ be a clustering of the data $\mathcal{X}$. Let T be the set of ground truth categories. We fix the number of clusters to match the number of categories $|\tilde{x}^c| = |T| = k$. For each cluster $\tilde{x}_j$, let $\gamma_T(\tilde{x}_j)$ be the maximal number of $\tilde{x}_j$'s elements that belong to one category. Then, accuracy $Acc(\tilde{x}_j, T)$ of a cluster $\tilde{x}_j$ with respect to T is defined as $Acc(\tilde{x}_j, T) = \gamma_T(\tilde{x}_j)/|\tilde{x}_j|$. The micro-averaged accuracy of a clustering $\tilde{x}^c$ is:

$$Acc(\tilde{x}^c, T) = \frac{\sum_{j=1}^k \gamma_T(\tilde{x}_j)}{\sum_{j=1}^k |\tilde{x}_j|} = \frac{\sum_{j=1}^k \gamma_T(\tilde{x}_j)}{|\mathcal{X}|}. \quad (4)$$

We evaluate the Comraf models on six text datasets. In addition to the standard benchmark 20 Newsgroups dataset (20NG) we use five real-world email

Table 1. Left 3 columns: statistics on the datasets. Right 3 columns: clustering accuracies (with standard error of the mean) for LDA and two Comraf algorithms. We report on only one of the two lengthy 20NG experiments with Comrafs.

Dataset	Size (num of docs)	Num of distinct words	Num of classes	LDA	Comraf (sequent)	Comraf (random)
acheyer	664	2863	38	44.3±0.4	47.8±0.4	47.1±0.4
mgervasio	777	3207	15	38.5±0.4	42.4±0.4	44.0±1.0
mgondek	297	1287	14	68.0±0.8	75.9±0.6	75.5±0.5
kitchen-l	4015	15579	47	36.7±0.3	42.4±0.6	41.6±0.8
sanders-r	1188	5966	30	63.8±0.4	67.4±0.3	67.6±0.3
20NG	19997	39764	20	56.7±0.6	69.5±0.7	

directories. Three of them belong to participants in the CALO project[5] and the other two belong to former Enron employees.[6] We preprocess the data following Bekkerman et al. [5]. Table 1 provides basic statistics on the six datasets.

We report on the clustering accuracy averaged over ten independent runs on the email datasets and five runs on 20NG. For the (unsupervised) clustering task we use the Comraf graph from Figure 1(b), with $\tilde{X}^c$ for document clusterings and $\tilde{Y}^c$ for word clusterings. We apply agglomerative clustering to documents and divisive clustering to words. We compare two Comraf algorithms proposed in Section 2.1 with Latent Dirichlet Allocation [8], a popular generative clustering model. We use Xuerui Wang's LDA implementation [13] that applies Gibbs sampling with 10000 sampling iterations.[7] As shown in Table 1, both Comraf algorithms outperform LDA on all five email datasets and by more than 12% on an absolute scale on 20NG. Interestingly, both Comraf algorithms show almost identical results which suggests that the method of constructing advantageous paths does not matter a lot, as soon as the number of iterations is the same.

Figure 1(d) shows a Comraf graph for the intrinsic scheme of semi-supervised clustering (see Section 4). Together with a node $\tilde{D}^c$ over document clusterings and a node $\tilde{W}^c$ over word clusterings, we introduce an observed node $\tilde{D}^c_0$, whose value $\tilde{d}^c_0$ is a given partitioning of labeled documents. Our objective derived from Equation (2) is: $(\tilde{d}^{c*}, \tilde{w}^{c*}) = \arg\max_{\tilde{d}^c_j, \tilde{w}^c_j} I(\tilde{D}_j; \tilde{W}_j) + I(\tilde{D}_j; \tilde{D}_0) + I(\tilde{W}_j; \tilde{D}_0)$.

We conduct the following experiment: for each email dataset, we uniformly at random select 10%, 20%, or 30% of the data and refer to it as labeled examples while the rest of the data is considered unlabeled. We apply both intrinsic and constrained methods on the three setups and plot the accuracy (calculated on unlabeled data only) vs. the percentage of labeled data used. The results are shown in Figure 2. As we can see from the figure, both methods unsurprisingly improve the unsupervised results, while the intrinsic Comraf method usually outperforms the constrained method. On 20NG, we select 10% of data to be

[5] `http://www.ai.sri.com/project/CALO`

[6] The preprocessed Enron email datasets can be obtained from `http://www.cs.umass.edu/ ronb/enron_dataset.html`.

[7] We also tried David Blei's LDA-C [8] that implements variational approximation and obtained significantly inferior accuracy.

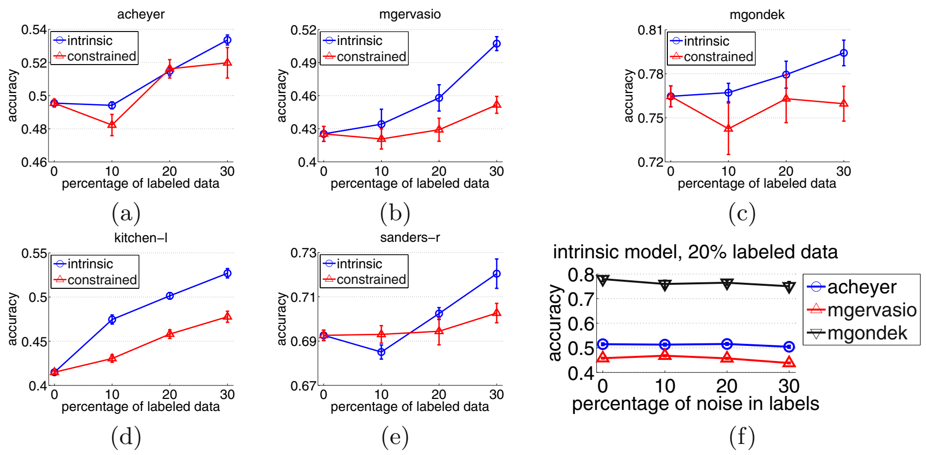


Fig. 2. Plots (a)-(e): comparing accuracies of the semi-supervised Comraf and the constrained optimization method on five email datasets. Plot (f): the semi-supervised Comraf's resistance to noise in labeled data.

labeled. The constrained method obtains 74.8±0.6% accuracy, while the intrinsic method obtains 78.9 ± 0.8% accuracy (over 5% and 9% absolute improvement to the unsupervised result, respectively). For another experiment with a semi-supervised Comraf, see [14].

The intrinsic scheme is resistant to noise. To show this, we conduct the following experiment: on CALO datasets with 20%/80% labeled/unlabeled split, we arbitrarily corrupt labels of 10%, 20% and 30% of the labeled data. Figure 2(f) shows that clustering accuracy remains almost unchanged for all three datasets.

Our *transfer learning* experiments are set up as follows. We notice that in two of the CALO datasets (ACHEYER and MGERVASIO) similar topics are discussed. Our hypothesis is that *known* categories of one dataset can improve the clustering results on another dataset. To test this hypothesis, we first consider one dataset to be labeled, while the other one is unlabeled, and then vice versa. However, since the two datasets do not consist of the *same* documents, we decide to use *word* clusters of the labeled dataset. We first cluster words distributed over categories of the labeled dataset, as described in [15]. Then we introduce the constructed clustering as an observed node $\tilde{W}_0^c$ into the Comraf graph (see Figure 1(e)) and perform the inference. Using this scheme we improve the clustering accuracy on MGERVASIO by 3% absolute over unsupervised clustering. However, we do not see any change in the results on the ACHEYER dataset.

6 Conclusion and Future Work

In this paper, we have presented combinatorial MRFs and empirically shown their utility on fundamental problems of unsupervised clustering, semi-supervised clustering, and transfer learning. In our future work, we aim at applying Comrafs to non-textual domains, such as computer vision. The use of Comrafs is not limited to clustering problems only. We plan to apply Comrafs to ranking, filtering and other tasks. Another interesting research problem is *model learning* in Comrafs. While model learning is often infeasibly expensive in graphical models with thousands or millions of nodes, we have shown that useful Comraf models can still be extremely compact, which makes model learning feasible.

Acknowledgements

We are grateful to Polina Golland, Leslie Pack-Kaelbling, Victor Lavrenko, Uri Lerner, Chris Pal and Yoram Singer for fruitful discussions. Portions of this research were conducted while Ron Bekkerman was an intern at Google Inc. Also, this work was supported in part by the Center for Intelligent Information Retrieval, in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR0011-06 C-0023, and in part by NSF under a CAREER Award #0546666. Ron thanks his wife Anna for her constant support.

References

1. McGurk, H., MacDonald, J.: Hearing lips and seeing voices. Nature **264**(5588) (1976) 746–748
2. de Sa, V.: Unsupervised Classification Learning from Cross-Modal Environmental Structure. PhD thesis, University of Rochester (1994)
3. Friedman, N., Mosenzon, O., Slonim, N., Tishby, N.: Multivariate information bottleneck. In: Proceedings of UAI-17. (2001)
4. Bickel, S., Scheffer, T.: Multi-view clustering. In: Proceedings of ICDM-4. (2004)
5. Bekkerman, R., El-Yaniv, R., McCallum, A.: Multi-way distributional clustering via pairwise interactions. In: Proceedings of ICML-22. (2005) 41–48
6. Li, S.: Markov random field modeling in computer vision. Springer Verlag (1995)
7. Besag, J.: Spatial interaction and statistical analysis of lattice systems. Journal of the Royal Statistical Society **36**(2) (1974) 192–236
8. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet allocation. JMLR **3** (2003) 993–1022
9. Tishby, N., Pereira, F., Bialek, W.: The information bottleneck method (1999) Invited paper to the 37th Annual Allerton Conference.
10. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: Proceedings of SIGKDD-9. (2003) 89–98
11. Besag, J.: On the statistical analysis of dirty pictures. Journal of the Royal Statistical Society **48**(3) (1986)
12. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: Proceedings of ICML-17. (2000)
13. McCallum, A., Corrada-Emmanuel, A., Wang, X.: Topic and role discovery in social networks. In: Proceedings of IJCAI-19. (2005) 786–791
14. Bekkerman, R., Sahami, M.: Semi-supervised clustering using combinatorial MRFs. In: Proceedings of ICML-23 Workshop on Learning in Structured Output Spaces. (2006)
15. Bekkerman, R., El-Yaniv, R., Tishby, N., Winter, Y.: Distributional word clusters vs. words for text categorization. JMLR **3** (2003) 1183–1208

Learning Stochastic Tree Edit Distance*

Marc Bernard[1], Amaury Habrard[2], and Marc Sebban[1]

[1] EURISE – Université Jean Monnet de Saint-Etienne
23, rue Paul Michelon – 42023 Saint-Etienne cedex 2 – France
{marc.bernard, marc.sebban}@univ-st-etienne.fr
[2] LIF – Université de Provence
39, rue Frédéric Joliot Curie – 13453 Marseille cedex 13 – France
{amaury.habrard}@lif.univ-mrs.fr

Abstract. Trees provide a suited structural representation to deal with complex tasks such as web information extraction, RNA secondary structure prediction, or conversion of tree structured documents. In this context, many applications require the calculation of similarities between tree pairs. The most studied distance is likely the tree edit distance (ED) for which improvements in terms of complexity have been achieved during the last decade. However, this classic ED usually uses *a priori* fixed edit costs which are often difficult to tune, that leaves little room for tackling complex problems. In this paper, we focus on the learning of a stochastic tree ED. We use an adaptation of the Expectation-Maximization algorithm for learning the primitive edit costs. We carried out series of experiments that confirm the interest to learn a tree ED rather than a priori imposing edit costs.

Keywords: Stochastic tree edit distance, EM algorithm, generative models, discriminative models.

1 Introduction

Nowadays, there is a growing interest for tree-structured data due to the potential applications in information extraction from the web, computational biology or phylogeny. Indeed, the hierarchical structure of trees is more suited for modeling web pages (XML, HTML), the RNA secondary structure of a molecule or phylogenetic trees than a flat representation such as strings. In applications, one often needs similarity measures to compare two different instances. This is, for example, useful for defining conversion models for dealing with heterogeneous XML data. In this context, many approaches have extended the well known string edit distance (ED) to trees [1].

The tree ED is usually defined as the less costly set of basic operations to change one tree to another. These primitive operations are constituted of the *substitution*, the *deletion* and the *insertion* of a node. The tree ED-based methods use, in general, *a priori* fixed costs for these so-called primitive edit operations.

* This work is part of the ongoing ARA Marmota research project.

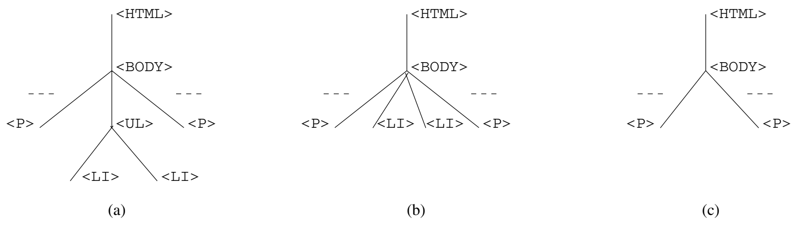


Fig. 1. Strategies to delete of a node within a tree

However, in many domains, an edit cost can highly depend on the nature of the symbols handled in a given operation. For example, the probability of changing a given symbol in a RNA structure depends on the probability that a genetic mutation occurs on this symbol. Thus, the similarity of two trees can strongly vary according to the specific domain in consideration. A solution could consist in assigning costs according to an expert valuation. However, this strategy may not be efficiently done in domains where the expertise is low. Moreover, even if the expertise level is sufficient, assigning a relevant cost to each edit operation can become a tricky task. Another way to overcome this drawback is to learn the edit costs from a sample of tree pairs. This can be achieved by modeling an ED as a stochastic process and using probabilistic methods to learn the model.

Note that in the context of strings, several approaches have been proposed during the last decade to learn a stochastic ED in the form of stochastic transducers [2,3], conditional random fields [4], or pair-Hidden-Markov-Models (pair-HMM) [5]. A parametric approach has been presented in [6] in the context of graph ED, where each edit operation is modeled by a Gaussian Mixture Density. Nevertheless, as far as we know, no method was proposed to directly learn edit costs for a stochastic tree edit distance. The aim of this paper is to fill this gap by a stochastic method specifically adapted to trees.

As we said before, the primitive edit operations for the standard tree ED are the substitution, insertion and deletion of a node. The most efficient procedures proposed notably by Shasha *et al.* [7] and Klein [8] have a polynomial complexity of order 4. In these approaches, when a node r is deleted within a tree, all its children are then connected to the father of r. This may be not relevant in some cases, for example in an HTML document: considering a set of items in an unordered list (see Fig. 1.a), it seems clearly irrelevant to delete the `<UL>` node without deleting the `<LI>` items (Fig. 1.b). Thus, to overcome this drawback and to also reduce the algorithmic complexity, we decided to use the less costly (with a quadratic complexity) tree ED, initially proposed by Selkow [9], as a base of our stochastic approach[1]. In this case, only a deletion of an entire (sub)tree can occur, and its removal implies the deletion of all its nodes from the leaves (Fig. 1.c). Note that the insertion of a (sub)tree follows the same principle, *i.e.* requires the iterative insertion of its nodes.

[1] Note that our learning method can be adapted to any other tree ED.

We propose in this paper two approaches for learning, from a sample of (*input,output*) pairs of trees, the costs used for computing a stochastic tree ED. First, we learn a *generative* model in the form of a joint distribution over tree pairs inspired by [2] in the case of strings. The advantage of such generative models is to provide an estimate of the unknown joint density with a small variance. However, it has an important drawback: the estimate is biased because it depends on the distribution of the *input* trees. In other words, this generative model will work if the distribution over the learning input trees follows the unknown underlying density of the input trees. This constraint justifies our second approach based on the learning of a *discriminative* model in the form of a conditional distribution. This type of models is known [10] to provide an unbiased estimate (despite a higher variance). We will show that such a strategy will work whatever the input distribution we use.

The rest of the paper is organized as follows: After some notations and definitions about the classic tree ED in Section 2, our two learning methods are presented in Section 3. They are based on an adaptation of the well-known *Expectation-Maximization* algorithm (EM) [11]. In Section 4, we carry out several series of experiments before concluding.

2 Tree ED

After some notations and definitions about trees, we present the main edit operations allowing us to change a tree into another one. Then, we describe a usual breadth-first-scanning-based approach for computing the ED.

2.1 Notations and Definitions

We assume we handle ordered labeled trees of arbitrary arity. There is a left-to-right order among siblings of a tree and trees are labeled with elements of a set $\mathcal{L}$ of labels. We denote $\mathcal{T}(\mathcal{L})$ the set of all labeled trees buildable from $\mathcal{L}$.

Definition 1. *Let V be a set of nodes. We inductively define trees as follows: a node is a tree, and given T trees $a_1, \ldots, a_T$ and a node $v \in V$, $v(a_1, \ldots, a_T)$ is a tree. v is the root of $v(a_1, \ldots, a_T)$, and $a_1, \ldots, a_T$ are subtrees.*

Definition 2. *Let $\mathcal{L}$ be a set of labels, and let $\lambda \notin \mathcal{L}$ be the empty label. Let $\phi : V \rightarrow \mathcal{L}$ be a labeling function. $v(a_1, \ldots, a_T)$ is a labeled tree if its nodes are labeled according to ϕ. Assuming that $\phi(v)$ is equal to a given label $l \in \mathcal{L}$, for convenience, we will also denote the labeled tree $v(a_1, \ldots, a_T)$ by $l(a_1, \ldots, a_T)$.*

2.2 Edit Operations and Edit Cost Functions

We are only concerned by three possible edit operations on trees: deletion of a subtree a_i (denoted (a_i, λ)), insertion of a subtree a_j (denoted (λ, a_j)), and

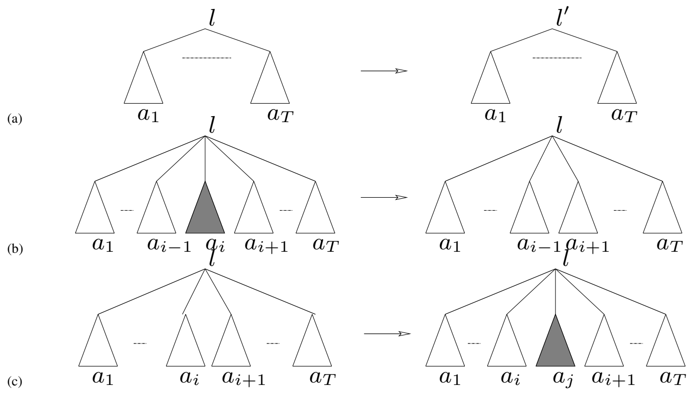


Fig. 2. (a) Substitution of l by l' (b) Deletion of a_i (c) Insertion of a_j

substitution of the label l of a tree root by l' (denoted (l, l')) (see Fig. 2). Let us define a cost function δ_t over these previous edit operations. Since a deletion or an insertion of a tree are respectively achieved by iteratively removing or inserting a set of nodes, δ_t can be directly defined from a cost function δ of edit operations on labels of the nodes. More formally, δ is a function defined from $(\mathcal{L} \cup \{\lambda\}) \times (\mathcal{L} \cup \{\lambda\}) \backslash \{(\lambda, \lambda)\}$ to $[0, 1]$.

The cost of the deletion of a tree can then be recursively computed as follows: $\delta_t(l(a_1, \dots, a_T), \lambda) = \delta(l, \lambda) + \sum_{i=1}^{T} \delta_t(a_i, \lambda)$. As we said in introduction, the cost matrix δ is usually *a priori* fixed. For example, consider the cost matrix δ of Fig. 3 and a given tree $b(c, d)$, then $\delta_t(b(c, d), \lambda) = \delta(b, \lambda) + \delta_t(c, \lambda) + \delta_t(d, \lambda) = \delta(b, \lambda) + \delta(c, \lambda) + \delta(d, \lambda) = 1.5$. Based on the same principle, the insertion of a tree requires successive insertions of its nodes: $\delta_t(\lambda, l'(b_1, \dots, b_V)) = \delta(\lambda, l') + \sum_{j=1}^{V} \delta_t(\lambda, b_j)$. Finally, the substitution of two labels is defined as follows: $\delta_t(l, l') = \delta(l, l')$.

2.3 Classic Tree ED Algorithms

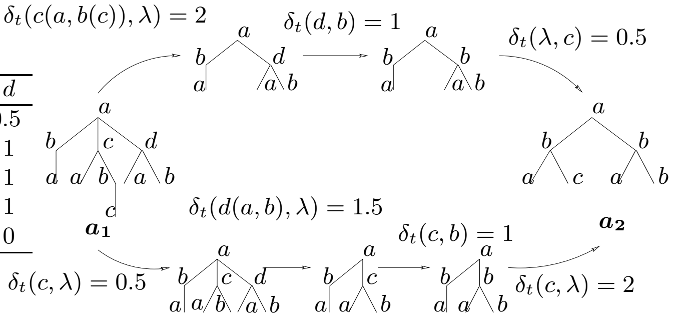
Once the cost function δ_t is established, it is possible to define a tree ED based on the following notion of edit script.

Definition 3. *Let a_1 and a_2 be two trees, an edit script on a_1 and a_2 is a sequence of edit operations changing a_1 into a_2. The cost of an edit script is the sum of the costs of its edit operations.*

Note that several scripts can exist (as shown in Fig. 3).

Definition 4. *The tree ED between two trees is the cost of the minimum cost edit script.*

δ	λ	a	b	c	d
λ	–	0.5	0.5	0.5	0.5
a	0.5	0	1	1	1
b	0.5	1	0	1	1
c	0.5	1	1	0	1
d	0.5	1	1	1	0



$\delta_t(c(a,b(c)),\lambda) = 2$
$\delta_t(d,b) = 1$
$\delta_t(\lambda,c) = 0.5$
$\delta_t(d(a,b),\lambda) = 1.5$
$\delta_t(c,b) = 1$
$\delta_t(c,\lambda) = 0.5$
$\delta_t(c,\lambda) = 2$
a_1
a_2

Fig. 3. A matrix δ and two possible edit scripts on two given trees a_1 and a_2

The tree ED $d(l(a_1,\ldots,a_T), l'(b_1,\ldots,b_V))$ between two trees $l(a_1,\ldots,a_T)$ and $l'(b_1,\ldots,b_V)$ as described in [9] can be recursively computed as follows:

$$\begin{aligned}
d(\lambda,\lambda) &= 0\\
d(l(a_1,\ldots,a_T),\lambda) &= \delta_t(l(a_1,\ldots,a_T),\lambda)\\
d(\lambda,l'(b_1,\ldots,b_V)) &= \delta_t(\lambda,l'(b_1,\ldots,b_V))\\
d(l(a_1,\ldots,a_T),l'(b_1,\ldots,b_V)) &= \delta(l,l') + d'(a_1,\ldots,a_T : b_1,\ldots,b_V)
\end{aligned}$$

where d' is defined as follows:

$$\begin{aligned}
d'(\lambda:\lambda) &= 0\\
d'(a_1,\ldots,a_T:\lambda) &= d'(a_1,\ldots,a_{T-1}:\lambda) + \delta_t(a_T,\lambda)\\
d'(\lambda:b_1,\ldots,b_V) &= d'(\lambda:b_1,\ldots,b_{V-1}) + \delta_t(\lambda,b_V)\\
d'(a_1,\ldots,a_T:b_1,\ldots,b_V) &= \min\begin{cases} d'(a_1,\ldots,a_{T-1}:b_1,\ldots,b_V) + \delta_t(a_T,\lambda)\\ d'(a_1,\ldots,a_T:b_1,\ldots,b_{V-1}) + \delta_t(\lambda,b_V)\\ d'(a_1,\ldots,a_{T-1}:b_1,\ldots,b_{V-1}) + d(a_T,b_V)\end{cases}
\end{aligned}$$

This distance can be efficiently computed using dynamic programming. In the next section, we show how it is possible to automatically learn the matrix δ from a corpus of tree pairs. Our stochastic approach is based on an adaptation of the well known EM algorithm [11]. EM aims at estimating the hidden parameters of a probabilistic model from a learning sample. In our case, these parameters are the costs of the matrix δ. In the following, the densities (joint or conditional) will be denoted with a subscript δ when they will be estimated from δ.

3 Learning Tree ED

We propose in the following two ways of learning a *stochastic edit distance* between two trees $l(a_1,\ldots,a_T)$ and $l'(b_1,\ldots,b_V)$. The first one concerns a *generative* model based on the estimation $p_\delta(l(a_1,\ldots,a_T),l'(b_1,\ldots,b_V))$ of the unknown joint probability $p(l(a_1,\ldots,a_T),l'(b_1,\ldots,b_V))$. The second proposition, a so-called *discriminative* approach, aims at learning a stochastic ED from the

```
**Input:** Two trees $l(a_1,\ldots,a_i)$ and $l'(b_1,\ldots,b_j)$, $1 \le i \le T$ and $1 \le j \le V$
**Output:** Probability of pair $(l(a_1,\ldots,a_i),\ l'(b_1,\ldots,b_j))$
$\alpha[0..T, 0..V]$ a $(T+1) \times (V+1)$ matrix; $\alpha[0,0] \leftarrow \delta(l,l')$
for $t = 0$ to $i$ do
    for $v = 0$ to $j$ do
        if $(t > 0)$ or $(v > 0)$ then $\alpha[t,v] \leftarrow 0$
        if $(t > 0)$ then $\alpha[t,v] \leftarrow \alpha[t,v] + \boldsymbol{\alpha}(a_t,\lambda) \times \alpha[t-1,v]$
        if $(v > 0)$ then $\alpha[t,v] \leftarrow \alpha[t,v] + \boldsymbol{\alpha}(\lambda,b_v) \times \alpha[t,v-1]$
        if $(t > 0)$ and $(v > 0)$ then $\alpha[t,v] \leftarrow \alpha[t,v] + \boldsymbol{\alpha}(a_t,b_v) \times \alpha[t-1,v-1]$
return $\alpha[i][j]$
```

Algorithm 1. $\boldsymbol{\alpha}(l(a_1,\ldots,a_i), l'(b_1,\ldots,b_j))$

estimated conditional distribution $p_\delta(l'(b_1,\ldots,b_V)|l(a_1,\ldots,a_T))$. The main difference between the two approaches occurs during the maximization step of EM.

3.1 Joint Tree ED

A *stochastic ED* supposes that edit operations occur according to an unknown random process. We aim at learning the underlying probability distribution $\delta(l,l')$ of these edit operations in order to estimate a joint probability distribution $p_\delta(l(a_1,\ldots,a_T), l'(b_1,\ldots,b_V))$ over tree pairs. We can show that this joint density will be valid if the following condition is fulfilled over the edit costs:

$$\sum_{(l,l')\in(\mathcal{L}\cup\{\lambda\})^2} \delta(l,l') = 1 \;\; and \;\; \delta(l,l') \ge 0 \qquad (1)$$

The joint probability represents the contribution of all ways to generate the two trees. $p_\delta(l(a_1,\ldots,a_T), l'(b_1,\ldots,b_V))$ is sufficient to model the *stochastic ED* defined as $d_s(l(a_1,\ldots,a_T), l'(b_1,\ldots,b_V)) = -\log p_\delta(l(a_1,\ldots,a_T), l'(b_1,\ldots,b_V))$.

To learn the matrix δ and then compute this joint probability $p_\delta(l(a_1,\ldots,a_T), l'(b_1,\ldots,b_V))$, we use an adaptation of the EM algorithm. Let us recall that EM achieves an expectation step followed by a maximization stage. During the first step, EM accumulates the expectation of each hidden event (edit operation) on the training corpus. In the maximization step, EM sets the parameter values (edit costs) to their relative expectations on the learning sample. To compute the joint probability, EM uses two auxiliary functions, so-called *forward* ($\boldsymbol{\alpha}$) and *backward* ($\boldsymbol{\beta}$).

To learn a stochastic tree ED, we adapted EM in the context of trees. The new forward function $\boldsymbol{\alpha}$ is described in Algorithm 1, whereas the new backward function $\boldsymbol{\beta}$ is presented in Algorithm 2.

These two functions are composed of two recursions: a breadth-first recursion on the children of the considered node and a depth-first one on the subtrees of this node. These two functions are symmetric. Although they process differently, they provide the same estimate $p_\delta(l(a_1,\ldots,a_T), l'(b_1,\ldots,b_V))$. Actually, the *forward*

```
**Input:** Two trees $l(a_i,\ldots,a_T)$ and $l'(b_j,\ldots,b_V)$, $1 \le i \le T$ and $1 \le j \le V$
**Output:** Probability of pair $(l(a_i,\ldots,a_T)$,$l'(b_j,\ldots,b_V))$
$\beta[0..T,0..V]$ a $(T+1)\times(V+1)$ matrix; $\beta[T,V] \leftarrow 1$
for $t = T$ down to $i-1$ do
    for $v = V$ down to $j-1$ do
        if $(t<T)$ or $(v<V)$ then $\beta[t,v] \leftarrow 0$
        if $(t<T)$ then $\beta[t,v] \leftarrow \beta[t,v] + \boldsymbol{\beta}(a_{t+1},\lambda) \times \beta[t+1,v]$
        if $(v<V)$ then $\beta[t,v] \leftarrow \beta[t,v] + \boldsymbol{\beta}(\lambda,b_{v+1}) \times \beta[t,v+1]$
        if $(t<T)$ and $(v<)V$ then
        $\beta[t,v] \leftarrow \beta[t,v] + \boldsymbol{\beta}(a_{t+1},b_{v+1}) \times \beta[t+1,v+1]$
if $i=1$ and $j=1$ then return $\beta[0][0] \times \delta(l,l')$ else return $\beta[i-1][j-1]$
```

Algorithm 2. $\boldsymbol{\beta}(l(a_1,\ldots,a_i),l'(b_1,\ldots,b_j))$

function visits the roots first and then scans the children from left to right, while the backward function processes from right to left and finally visits the roots of the tree pair. Fig. 4 illustrates these two algorithms.

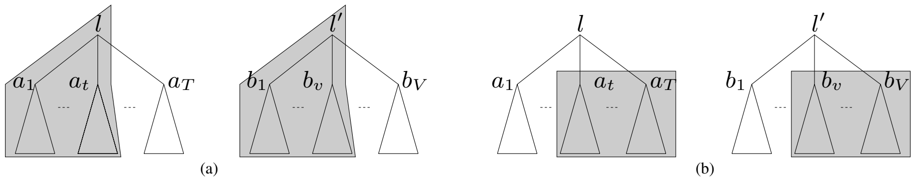


Fig. 4. (a) Evaluation of $\boldsymbol{\alpha}(l(a_1,\ldots,a_t),l'(b_1,\ldots,b_v))$ by the forward algorithm. (b) Evaluation of $\boldsymbol{\beta}(l(a_t,\ldots,a_T),l'(b_v,\ldots,b_V))$ by the backward algorithm.

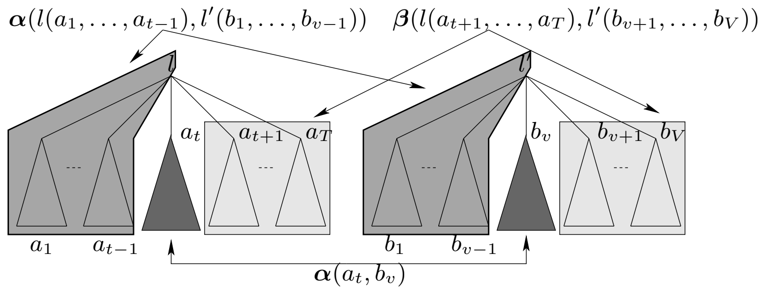
Both functions can be computed with a quadratic complexity using dynamic programming techniques and allow us to define a probability distribution over pairs of trees:

$$\sum_{(a_i,b_j)\in(\mathcal{T}(\mathcal{L}))^2} p_\delta(a_i,b_j) = \sum_{(a_i,b_j)\in(\mathcal{T}(\mathcal{L}))^2} \boldsymbol{\alpha}(a_i,b_j) = \sum_{(a_i,b_j)\in(\mathcal{T}(\mathcal{L}))^2} \boldsymbol{\beta}(a_i,b_j) = 1$$

Let us present now the expectation and maximization steps for learning the edit costs. During the expectation step, we store in an auxiliary matrix γ $(|\mathcal{L}|+1)\times(|\mathcal{L}|+1)$ the expected number of times each edit operation was used to transform a tree in another one from a learning tree pairs LS. We apply for each tree pair $(l(a_1,\ldots,a_T),l'(b_1,\ldots,b_V)) \in LS$ the procedure *expectation*$(l(a_1,\ldots,a_T)$, $l'(b_1,\ldots,b_V))$ described in Algorithm 3 (where $l_r(a_i)$ denotes the label of the root of a_i). Note that this function uses the previously mentioned *backward* and *forward* functions. Fig. 5 gives an illustration for evaluating a substitution.

```
Input: Two trees $l(a_1,\ldots,a_T)$ and $l'(b_1,\ldots,b_V)$
for $t$ from $0$ to $T$ do
    for $v$ from $0$ to $V$ do
        if $(t>0)$ then
            $\gamma(l_r(a_t),\lambda) \leftarrow \gamma(l_r(a_t),\lambda) + \frac{\boldsymbol{\alpha}(l(a_1,\ldots,a_{t-1}),l'(b_1,\ldots,b_v))\boldsymbol{\alpha}(a_t,\lambda)\boldsymbol{\beta}(l(a_{t+1},\ldots,a_T),l'(b_{v+1},\ldots,b_V))}{\boldsymbol{\alpha}(l(a_1,\ldots,a_T),l'(b_1,\ldots,b_V))}$
            $expectation(a_t,\lambda)$
        if $(v>0)$ then
            $\gamma(\lambda,l_r(b_v)) \leftarrow \gamma(\lambda,l_r(b_v)) + \frac{\boldsymbol{\alpha}(l(a_1,\ldots,a_t),l'(b_1,\ldots,b_{v-1}))\boldsymbol{\alpha}(\lambda,b_v)\boldsymbol{\beta}(l(a_{t+1},\ldots,a_T),l'(b_{v+1},\ldots,b_V))}{\boldsymbol{\alpha}(l(a_1,\ldots,a_T),l'(b_1,\ldots,b_V))}$
            $expectation(\lambda,b_v)$
        if $(t>0)$ and $(v>0)$ then
            $\gamma(l_r(a_t),l_r(b_v)) \leftarrow \gamma(l_r(a_t),l_r(b_v)) + \frac{\boldsymbol{\alpha}(l(a_1,\ldots,a_{t-1}),l'(b_1,\ldots,b_{v-1}))\boldsymbol{\alpha}(a_t,b_v)\boldsymbol{\beta}(l(a_{t+1},\ldots,a_T),l'(b_{v+1},b_V))}{\boldsymbol{\alpha}(l(a_1,\ldots,a_T),l'(b_1,\ldots,b_V))}$
            $expectation\ (a_t,b_v)$
```

Algorithm 3. ***expectation***$(l(a_1,\ldots,a_T),l'(b_1,\ldots,b_V))$



$\boldsymbol{\alpha}(l(a_1,\ldots,a_{t-1}),l'(b_1,\ldots,b_{v-1}))$
$\boldsymbol{\beta}(l(a_{t+1},\ldots,a_T),l'(b_{v+1},\ldots,b_V))$
l
l'
a_t
a_{t+1}
a_T
b_v
b_{v+1}
b_V
a_1
a_{t-1}
b_1
b_{v-1}
$\boldsymbol{\alpha}(a_t,b_v)$

Fig. 5. Use of the forward and backward functions to evaluate a substitution cost

The maximization step is crucial in the EM algorithm because it describes the normalization of the expectations ensuring a convergence of the process under constraints. The constraint to fulfill for learning a joint tree ED has been described in Eq.1. Thus, the normalization step is here very simple and only consists in dividing each expectation $\gamma(l,l')$ by the total accumulator $TA = \sum_{l\in\mathcal{L}\cup\{\lambda\}}\sum_{l'\in\mathcal{L}\cup\{\lambda\}}\gamma(l,l')$. The resulting maximization algorithm is described in Algorithm 4. By combining Algorithms 1,2,3,4, we can now draw the general learning algorithm of a joint stochastic tree ED (see Algorithm 5). Note that the process is repeated until convergence. This is reached when the probability of each edit operation does not significantly change between two iterations.

Note that it is the normalization achieved in the maximization step that allows us to learn a joint distribution $p_\delta(l(a_1,\ldots,a_T),l'(b_1,\ldots,b_V))$. However, in order to use such a model in a classification task (for example for converting a structured document a_i into another one b_j), we would need a conditional

```
Input: A matrix of accumulators $\gamma$
Output: A matrix of joint stochastic edit costs $\delta$
$TA \leftarrow 0$
foreach $(l, l') \in (\mathcal{L} \cup \{\lambda\})^2$ do $TA \leftarrow TA + \gamma(l, l')$
foreach $(l, l') \in (\mathcal{L} \cup \{\lambda\})^2$ do $\delta(l, l') \leftarrow \frac{\gamma(l,l')}{TA}$
```

Algorithm 4. ***maximization*** (for joint distribution)

```
Input: $LS$ a learning set of tree pairs
repeat
    foreach $(l, l') \in (\mathcal{L} \cup \{\lambda\})^2$ do $\gamma(l, l') \leftarrow 0$
    foreach $(l(a_1, \dots, a_T), l'(b_1, \dots, b_V)) \in LS$ do
        expectation$(l(a_1, \dots, a_T), l'(b_1, \dots, b_V))$
    maximization$(\gamma)$
until convergence
```

Algorithm 5. ***expectation − maximization***

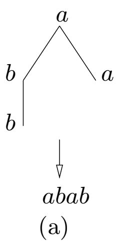
distribution $p_\delta(b_j|a_i)$ rather than a joint one. Actually, in such a context, the input tree is known and we are looking for the optimal corresponding output. A simple solution would consist in computing $p_\delta(b_j|a_i)$ from the joint distribution such that $p_\delta(b_j|a_i) = \frac{p_\delta(a_i,b_j)}{p(a_i)}$. However, this implies a dependence on the input distribution $p(a_i)$, and thus can generate a bias.

One solution to overcome this drawback consists in directly learning a conditional distribution, usually called a discriminative model. The advantage of this approach is to remove the statistical bias of generative models. This is the goal of the next section. We propose a new maximization step aiming at normalizing the accumulators obtained after the expectation step such as to directly obtain a conditional distribution $p_\delta(b_j|a_i)$ at each stage of EM.

3.2 Learning Conditional Tree ED

To achieve this task, we have to draw the new constraints corresponding to this conditional context $p_\delta(b_j|a_i)$. In fact, it is possible to model the output distribution conditionally to an input tree a_i in the form of a non deterministic probabilistic finite state automaton. Let us take a simple example to explain the principle. We assume that the input tree $a(b(b), a)$ is the one described in Fig. 6(a). Since we use a breadth-first scanning for computing the ED, $a(b(b), a)$ can be rewritten in the form of the string "abab". Thus, it is possible to model the output distribution conditionally to the input tree in the form of the probabilistic automaton of Fig. 6(b).

The cycles of each state correspond to the possible insertions before and after the reading of an input symbol. The state with a double circle is a final state and corresponds to the end of the reading of the input tree (which will be character-



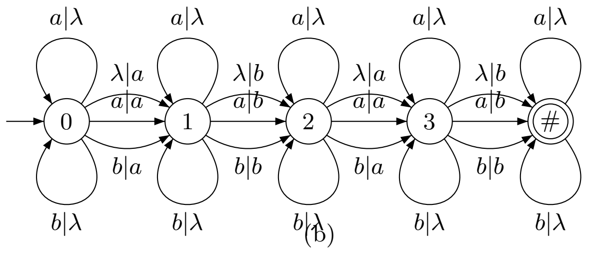


Fig. 6. Output distribution conditionally to an input tree

ized by the termination symbol #). In order to learn a statistical distribution over the pairs of trees, it is easy to show that this automaton must satisfy the following two conditions:

1. First, probabilities of the outgoing transitions of each state must sum to 1:
$$\forall l \in \mathcal{L}, \sum_{l' \in \mathcal{L} \cup \{\lambda\}} \delta(l'|l) + \sum_{l' \in \mathcal{L} \cup \{\lambda\}} \delta(l'|\lambda) = 1 \qquad (2)$$
where $\delta(l'|l)$ is now the probability to generate the output symbol l' conditionally to the input symbol l.
2. Second, probabilities from the final state must also describe a distribution:
$$\sum_{l' \in \mathcal{L}} \delta(l'|\lambda) + \delta(\#) = 1. \qquad (3)$$

The optimal normalization under these new constraints is the solution of an optimization problem as that of presented in Dempster et al. [11]. In the following, we only provide in Algorithm 6 the normalization that fulfills these constraints 2 and 3. Due to the lack of space, we do not provide here the proof justifying this optimal solution, but the interested reader can find in [3] the principle of this proof in the case of *string pairs.*

```
Input: A matrix of accumulators $\gamma$
Output: A matrix of conditional stochastic edit costs $\delta$
$N \leftarrow \sum_{l \in \mathcal{L} \cup \{\lambda\}} \sum_{l' \in \mathcal{L} \cup \{\lambda\}} \gamma(l, l')$ ; $N(\lambda) \leftarrow \sum_{l' \in \mathcal{L}} \gamma(\lambda, l')$
foreach $l \in \mathcal{L}$ do $N(l) \leftarrow \sum_{l \in \mathcal{L} \cup \{\lambda\}} \gamma(l, l')$
$\delta(\lambda|\lambda) \leftarrow \frac{N - N(\lambda)}{N}$
foreach $(l, l') \in (\mathcal{L} \cup \{\lambda\})^2$ do $\delta(l'|l) \leftarrow \frac{\gamma(l,l')}{N(l)} \frac{N-N(\lambda)}{N}$
foreach $l \in \mathcal{L}$ do $\delta(\lambda|l) \leftarrow \frac{\gamma(l,\lambda)}{N(l)} \frac{N-N(\lambda)}{N}$
foreach $l' \in \mathcal{L}$ do $\delta(l'|\lambda) \leftarrow \frac{\gamma(\lambda,l')}{N}$
```

Algorithm 6. ***maximization*** (for conditional distribution)

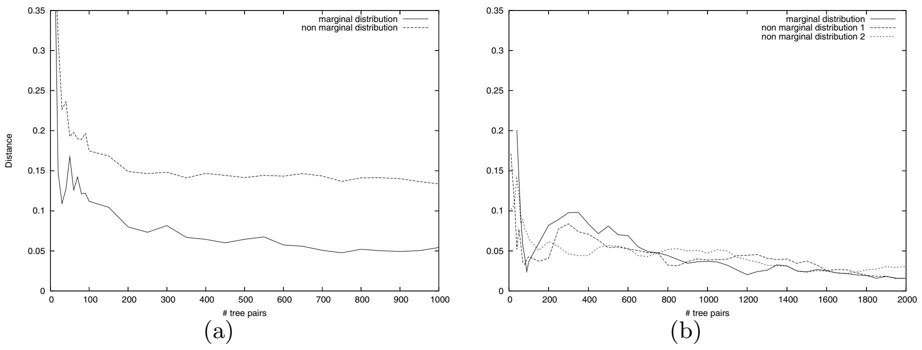


Fig. 7. Results of our experiments

4 Experiments

We carried out experiments to assess the relevance of our two models of stochastic ED to correctly estimate the parameters of a target model. If we are able to learn this target, this will mean that a learned tree ED will always outperform a classic tree ED with *a priori* hand-tuned costs. Actually, in the best case, the latters will be those of the learned matrix δ. In other words, this means that our learning algorithm will be efficient to deal with real-world applications.

The experimental setup is the following: First, we generate a target distribution defined by a theoretical matrix δ^* (describing either a joint or a conditional distribution). Then, we generate a sample of input trees according to a given input distribution. To build a learning set LS of tree pairs, we assign to each input instance an output tree. This one is generated using the input tree and the edit operations described by the target distribution δ^*. Note that in real world applications, such pairs would represent couples of similar instances (for example, pairs of (noisy, unnoisy) trees).

The aim is to learn δ^* from LS (constituted of a growing number of tree pairs) using both of our generative and discriminative models. To assess the effect of the input distribution on the learned model, we use different densities to generate the input trees. The performance criterion we use is the normalized distance between the target and the learned distributions.

In a first series of experiments, we focus on the generative model (*i.e.* a joint one). In this case, we build two sets of input trees. The first one is obtained using the marginal distribution of δ^* which is defined as follows: $\forall l \in \mathcal{L} \cup \{\lambda\}, \delta^*(l) = \sum_{l' \in \mathcal{L} \cup \{\lambda\}} \delta^*(l, l')$. The second one is generated using a random distribution. The chart of Fig.7(a) shows the results. As expected, the only one way to learn the target requires to use its marginal distribution to generate the input trees. The use of another (random) density leads to a bias, *i.e.* a large distance between the target and the learned model.

We use the same experimental setup during the second series of experiments aiming at learning a conditional target model. In this case, we tested three different input distributions (among them one is the marginal one). The chart of Fig.7(b) confirms that whatever the input distribution we use, our discriminative model is able to learn the target model.

5 Conclusion

In this paper, we proposed two original approaches for learning a *stochastic tree ED*. This is, as far as we know, the first attempt to learn such a distance specifically adapted to trees. In the first method, we modeled this distance as a joint distribution on tree pairs. This model has the advantage of having a small variance but is biased. Thus, it is suited for dealing with real applications where the instances are not numerous but describe well the underlying distribution. We also proposed to learn a stochastic edit distance from a conditional distribution that allows us to remove this bias. Such a way to proceed is interesting overall when the size of the learning set is sufficiently large, reducing then the variance of such a model. The experimental results confirm the interest of both approaches.

We plan to extend our work to stochastic models able to take into account edit costs varying according to the tree context. Actually, the cost of an edit operation can depend on the location where it occurs in the tree, that is not taken into account with our current structures. This implies to learn more complex models, such as stochastic tree transducers.

References

1. Bille, P.: A survey on tree edit distance and related problem. Theoretical Computer Science **337**(1-3) (2005) 217–239
2. Ristad, S., Yianilos, P.: Learning string-edit distance. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(5) (1998) 522–532
3. Oncina, J., Sebban, M.: Learning stochastic edit distance: application in handwritten character recognition. Journal of Pattern Recognition (2006) to appear.
4. McCallum, A., Bellare, K., Pereira, P.: A conditional random field for disciminatively-trained finite-state sting edit distance. In: UAI2005. (2005)
5. Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: Biological sequence analysis. Cambridge University Press (1998)
6. Neuhaus, M., Bunke, H.: A probabilistic approach to learning costs for graph edit distance. In: 17th Int. Conf. on Pattern Recognition, IEEE (2004) 389–393
7. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal of Computing (1989) 1245–1262
8. Klein, P.: Computing the edit-distance between unrooted ordered trees. In: Proc. of the 6th European Symposium on Algorithms (ESA), Springer (1998) 91–102
9. Selkow, S.: The tree-to-tree editing problem. Information Processing Letters **6**(6) (1977) 184–186
10. Bouchard, G., Triggs, B.: The trade-off between generative and discrminative classifiers. In: COMPSTAT'2004, Springer (2004)
11. Dempster, A., Laird, M., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc **B**(39) (1977) 1–38

Pertinent Background Knowledge for Learning Protein Grammars

Christopher H. Bryant[1,⋆], Daniel C. Fredouille[1], Alex Wilson[2], Channa K. Jayawickreme[3], Steven Jupe[4], and Simon Topp[5]

[1] School of Computing, The Robert Gordon University, Aberdeen UK
chb@comp.rgu.ac.uk
[2] School of Computing, Division of Mathematics and Statistics, The Robert Gordon University, Aberdeen UK
[3] Discovery Research Biology, GlaxoSmithKline, Durham, USA
[4] Department of Bioinformatics, GlaxoSmithKline, Stevenage, UK
[5] Department of Bioinformatics, GlaxoSmithKline, Harlow, UK

Abstract. We are interested in using Inductive Logic Programming (ILP) to infer grammars representing sets of protein sequences. ILP takes as input both examples and background knowledge predicates. This work is a first step in optimising the choice of background knowledge predicates for predicting the function of proteins. We propose methods to obtain different sets of background knowledge. We then study the impact of these sets on inference results through a hard protein function inference task: the prediction of the coupling preference of GPCR proteins. All but one of the proposed sets of background knowledge are statistically shown to have positive impacts on the predictive power of inferred rules, either directly or through interactions with other sets. In addition, this work provides further confirmation, after the work of *Muggleton et al., 2001* that ILP can help to predict protein functions.

1 Introduction

Inductive Logic Programming (ILP) has tackled many molecular biological applications such as: secondary structure prediction [1, 2], Mutagenic activity of small molecules [3], prediction of genes' functions [4, 5] and prediction of functions of proteins [6]. We are interested in this last application, where grammars inferred from protein sequences have been shown, through a case study, to help to predict the function of proteins. This paper provides a comparison of sources of background knowledge that can be used in such tasks. It is also a confirmation, with stronger statistical evidence than [6], of the utility of ILP inferred rules in predicting protein functions.

The next section introduces protein grammar inference via ILP. Section 2 presents different sets of background knowledge predicates that can be used for inference of grammars over proteins. Section 3 evaluates the main effects and interactions of the different sets using a reliability engineering method known as *Taguchi design* [7], 10-folds cross-validations are used to draw the final conclusions.

⋆ Contact author.

Protein Grammar Inference with ILP

Patterns in the form of grammars have been used with success to model protein families. The use of such grammars is twofold: (1) they can be used to annotate sequences of unknown function, providing molecular biologists with a likely function for such sequences; (2) they can help biologists to understand how biological functions are realised because the grammar structure represents common points between sequences of similar functions. Many grammar formalisms have been used, including String Variable Grammars (SVG) [8], Patscan patterns [9], Prosite patterns [10, 11], Basic Gene Grammars (BGG) [12] and Probabilistic Regular or Context-Free Grammars [13, 14]. The hand development of grammars, using for example SVG or BGG formalisms, is difficult and requires expensive human expertise. Moreover, some patterns might be too subtle to be recognised by a human expert. Thus, given the enormous volume of data arising from genome projects, the acquisition of grammars from sets of biological sequences needs to be automated.

ILP has two advantages in this application domain: first ILP infers logic programs, and logic programs have been shown useful to represent hand designed protein grammars (*e.g.*, with SVG [8]); second, unlike most machine learning techniques, ILP is able to bias inference to take expert knowledge into account. This is certainly an advantage in this application domain since, as protein sequences are not just sequences but represent molecules with physical and chemical properties, expert knowledge is often available. However, as providing more background knowledge predicates enlarges the search space, a compromise between the space size and the amount of knowledge introduced has to be found.

Different approaches to grammar learning with ILP have been considered, mainly by Cussens and Pulman [15] and Muggleton *et al.* [6]. These papers differ in two main points. First the application in [15] is natural language while the one in [6] is molecular biology. Second, the logic representation in [15] uses chart parsing tables, while Definite Clause Grammars (DCG) [16] are used in [6]. Our inference approach takes its roots in the work of Muggleton *et al.* [6] and can be summarised as follows. The inference process takes as inputs: (1) examples (and counter-examples if available) of the form `target(L,[]).` where `L` is a list representing the primary structure of the example protein, *i.e.*, the sequence of its amino-acids (*e.g.*, `L=[n,n,e,v,...]`) and `[]` is a list which is empty *i.e.*, has no elements; and (2) background knowledge predicates of the form `pred`$_i$`(+IL,-OL).` where `IL` is the input list of amino-acids, and `OL` is the output list, which is a suffix of `IL`, obtained by removing the amino-acids matched by the predicate from `IL`. From these, the inference process infers rules of the form `target(A,B):- pred`$_1$`(A,C), pred`$_2$`(C,D),...pred`$_n$`(X,B).` which maximises the score function of the ILP system. For further details see [6].

This study proposes sets of background knowledge predicates for protein grammar learning (*i.e.*, the `pred`$_i$), and a statistical study of the influence of these sets on the predictive power of inferred rules.

BKS	Example predicate	Rules
$\mathcal{L}et$	`a/2`	`a([a|B],B).`
$\mathcal{P}ro$	`tiny/2`	`tiny([a|B],B). tiny([g|B],B). tiny([s|B],B)`
$\mathcal{G}_u$	`gap/2`	`gap(A,A). gap([_|A],B):-gap(A,B).`
$\mathcal{G}_s$	`x0_1/2`	`x0_1(A,A). x0_1([_|A],A).`
$\mathcal{S}_p$ or $\mathcal{S}_n$	`dry/2`	`dry([d,r,y|B],B).`
$\mathcal{P}_a$	`pratt1/2`	`pratt1(A,B):- h(A,C), t_or_i(C,D), x0_1(D,E), tiny(E,F), t(F,B).` `t_or_i([t|A],A). t_or_i([i|A],A).`
$\mathcal{P}_s$	`pratt_sub1/2`	`pratt_sub1(A,B):- h(A,C), x0_1(C,D), t_or_i(D,E).`

Fig. 1. Examples for the different BKSs studied in this paper

2 Protein Sequence Background Knowledge

We can split the background knowledge into two main categories: general molecular biology knowledge (Subsection 2.1), and knowledge specific to each particular data-set (Subsection 2.2). In the following, a set of background knowledge predicates obtained by a common procedure is denoted by BKS (Background Knowledge Set).

2.1 General Molecular Biology Knowledge

This subsection considers expert knowledge which can *a-priori* be considered relevant for any protein grammar inference process. We can split such knowledge in two parts: (1) amino-acid letters and their physico-chemical properties, (2) gaps. Except for some gaps predicates, these predicates have already been used to infer biological grammars in [6].

Amino-Acid Letters and Properties. The two first BKSs we consider are predicates matching exactly one amino-acid letter (denoted by $\mathcal{L}et$), and predicates matching sets of amino-acid with common physico-chemical properties (denoted by $\mathcal{P}ro$). The use of these BKSs is motivated by the knowledge that the conservation – of amino-acids for $\mathcal{L}et$, or of physico-chemical properties for $\mathcal{P}ro$ – at some positions in the proteins can often help predicting the protein function. Different physico-chemical properties can be considered; for this work, we used those proposed by [1] and also used in [6]. Examples of predicates for the $\mathcal{L}et$ and $\mathcal{P}ro$ BKSs are given in Figure 1.

Gaps. Protein sequences contain parts participating to the overall structure of the molecule but which are either not directly relevant to the function or which cannot be characterised by the provided background predicates. To match such parts of the protein, we can use predicates called *gaps*; we consider two types of gaps: *unlimited* and *short* gaps. An unlimited gap is a predicate which can match any sequence. We will denote this BKS by $\mathcal{G}_u$. There is just one predicate for $\mathcal{G}_u$, namely `gap/2` (see Figure 1). The second BKS, short gaps, denoted by

$\mathcal{G}_s$, contains predicates matching sequences with small length (we considered predicates matching sequences with lengths from 0 to 1, 0 to 2, 1 to 1, 1 to 2, and 2 to 2). As an example the predicate matching sequences with lengths from 0 to 1 is given in Figure 1. While unlimited gaps can cover large uncharacterised parts of proteins, short gaps can help the discovery of well conserved groups of amino-acids separated by a few, less conserved, amino-acids. Some biological grammars contains gaps matching a range of large lengths. In this work, we considered that they can be approximated by the `gap/2` predicate.

2.2 Sequence Family Knowledge

In addition to the generic BKSs discussed above, BKSs on the particular protein family under study are available. These BKSs can be obtained from two sources: from experts on that protein family, or by automatically processing the examples. Since the availability and quality of background knowledge provided by experts can vary, it is not taken into account in this study. We therefore focus on knowledge that can be automatically extracted from the training examples, before inference.

Subsequences. We consider providing *exceptionally frequent* subsequences of the positive examples to the ILP system. We proceed in four steps. During step (1), we extract subsequences that are present in at least 10% of the positive training set. This enables inferred rules using the subsequences to cover a reasonable amount of examples. Let $Obs(s)$ be the number of positive examples containing subsequence s. During step (2), we define a distribution over subsequences and compute for each subsequence s the number of times, denoted $Exp(s)$, s is expected to appear in the examples. We consider two distributions detailed in the paragraphs below. In step (3), we score each subsequence using the value $\frac{(Obs(s)-Exp(s))^2}{Exp(s)}$. This score function was proposed in [17] for the extraction of exceptional subsequences in biological sequences. In step (4) the subsequences are ranked using the score and only the best 40 are kept. The next two paragraphs detail the two distributions used for step (2).

Distribution over the positive examples. Using a distribution based on the positive examples enables to detect subsequences describing the positive examples. To obtain such a distribution, we use Verbumculus [17]. Verbumculus trains a Markov Model (MM) on the provided sequences. The expected frequency of the subsequences with respect to the MM distribution can be extracted from Verbumculus output. The order of the MM is an important parameter: when an order of O is taken, only subsequences longer than $O + 1$ have frequency which can be different from the MM expected frequency. To use the maximum amount of information available in the positive examples, we therefore trained a MM of order $L - 2$ to obtain the expected frequencies for subsequences of length L. The subsequences obtained using this distribution are denoted by $\mathcal{S}_p$.

Distribution over the negative examples. Subsequences generated from the above distribution may be present in the negatives as well as in the positives. Subsequences discriminating between positive and negative examples can be obtained by using a distribution based on the set of negative examples. In this case, the distribution and the extracted subsequences are not obtained from the same set of sequences. In consequence we can use a simpler method than the one proposed for the previous distribution. Instead of using a MM, the expected frequency of a subsequence is estimated by $Exp = count(sub) * \frac{P}{N}$, where $count(sub)$ is the number of negative examples containing the subsequence, and P and N are respectively the positive and negative training set size. The subsequences obtained using this distribution are denoted by $\mathcal{S}_n$.

Pratt. In addition to subsequence extraction, software already exist to extract common points in protein sequences and represent them as patterns. The most popular is certainly PRATT [10]. The patterns inferred by PRATT are obtained using only positive examples. PRATT patterns can be seen as simplified regular expressions, an example of such a pattern is: `H-[TI]-x(0,1)-[KRH]-T`. An equivalent DCG is provided in Figure 1, line $\mathcal{P}_a$. Such patterns are widely used by molecular biologists, as shown by their availability in the Prosite database [11]. We propose to use these patterns as they stand (BKS denoted by $\mathcal{P}_a$) or to extract smaller patterns from them (denoted by $\mathcal{P}_s$). For $\mathcal{P}_s$, we extracted all sub-patterns containing two non gap elements. For example, sub-patterns `H-[TI]`, `[TI]-x(0,1)-[KHR]` and `[KHR]-T` can be extracted from the above pattern (see also Figure 1, line $\mathcal{P}_s$). This second usage aims at compensating for the fact that PRATT cannot take into account counter-examples: it potentially returns patterns frequent in both the positive and negative examples sets. Refinements of $\mathcal{P}_s$ by the ILP system could help the creation of patterns rejecting the negatives.

3 Evaluation of Background Knowledge Effects

Subsection 3.1 presents the inference task. Subsection 3.2 explains the experiments that evaluate the influence of the different BKSs on inference and subsection 3.3 discusses the experimental results. Experimental materials are available at: `http://www.comp.rgu.ac.uk/staff/chb/research/data_sets/ecml06/bk` .

3.1 Description of the Inference Data and Task

Data-set. G-protein coupled receptors (GPCRs) are the biggest single class of receptors in biology. An understanding of how they couple with specific classes of G-proteins is vital for further comprehending the function of the receptor within a cell. The data set consists of two sets of sequences representing two qualitatively distinct classes, Gi/o and Gs/q, of GPCRs [18]. Gi/o and Gs/q are the *coupling specificity* of the GPCRs proteins. Data allowing the classification of these proteins into the two sets is proprietary to GlaxoSmithKline (GSK), the industrial collaborator of this project. The Gi/o and Gs/q data-sets contain 64 and 126 sequences respectively. The task we consider is to infer rules which classify GPCRs as either Gi/o or Gs/q. It is possible that some GPCRs have both the Gi/o and Gs/q properties, however the sequences in our data-set are known to belong only to one of these classes.

Different papers tackle the prediction of GPCR coupling using machine learning. These include the use of regular expressions, Naïve Bayes and Hidden Markov Models (see [19] for a good overview). The state of the art methods providing the best classifications are very specialised to the GPCR coupling prediction task [19], showing the difficulty of the task. Our aim in this paper is not to provide a better classifier than the existing ones, but to evaluate the effect of the BKSs on this hard inference task, using generic ILP methods.

GPCRs have a characteristic 7 membrane-spanning regions and thus have regions outside the cell, within the cell membrane and inside the cell. It is believed that the G-protein binding property depends only on the subsequences of GPCRs situated inside the cell. We therefore only considered these four intra-cellular subparts during the inference processes. This means that the original data-set can be separated into eight sets, four containing Gi/o sequences and four containing Gs/q sequences; the four data-sets associated to each class corresponding to the four intra-cellular subparts. The length of these subsequences varies from 11 to 23 in the first subpart, from 16 to 42 in the second, 21 to 245 in the third, and finally 21 to 172 in the fourth. Because the limits of subsequences in each subpart are not always well defined, we decided to infer patterns conserved *inside* the subparts, therefore all bodies of inferred rules for this work start and end with the `gap/2` predicate. Providing (or not) $\mathcal{G}_u$ to the inference process therefore means that we allow (or not) the `gap/2` predicate to be present somewhere other than at the beginning or end of inferred rules' body.

We created cross-validation sets from this data. Our method of partitioning the data ensured training and test sets never contained homologous sequences. To ensure this: (1) we concatenated the four intra-cellular subparts of each GPCR; (2) we created clusters over these sequences with BlastClust [20] (these clusters are based on homology between the sequences); and finally (3), clusters (instead of sequences) are randomly put in n-disjoint sets which are then used to create a n-folds cross-validation set (we used n=5 and n=10, see Subsection 3.2).

Predictions. To be able to make predictions, we have to combine the 8 inferred sets of rules which are obtained by: (1) inferring on the four different intra-cellular subparts, (2) using either Gi/o or Gs/q as positive examples (the examples of the other class being used as negatives). For a given rule r, let $pr(r)$ be its precision over the Gi/o training examples, *i.e.*, $pr(r) = \frac{p}{p+n}$ where p (resp. n) is the number of training Gi/o examples (resp. Gs/q examples) accepted by r. For each sequence to classify, we parse each of its intra-cellular subparts with the associated inferred rules[1]. Let R be the set of rules matching the sequence (on the respective subparts they have been inferred on). The sequence is then associated with the average obtained precisions over the matching rules, *i.e.*, the value $\sum_{r \in R} \frac{pr(r)}{|R|}$. The larger the obtained value, the more likely the sequence is Gi/o, the smaller, the more likely the sequence is Gs/q. This strategy has been used mainly because it is a simple way to weight rules using information from their training set performance.

ILP System and Parameters. We used the Aleph ILP inference platform (`http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph`) to run our experiments. our previous work which provides large speed-ups of Aleph for biological grammar inference.

The Gi/o and Gs/q sets contain a very different number of sequences while having the same importance to the biologists. Therefore, to avoid biasing the inference toward one class, we decided to weight, in the ILP system evaluation function, the examples of each class by the inverse of the number of instances of the class available. The evaluation function used is the accuracy over the weighted examples, *i.e.*, acc $= \frac{1}{2} * (\frac{p}{P} + \frac{n}{N})$, where P (resp. N) is the size of the positive (resp. negative) training set size, and p (resp. n) is the number of positive (resp. negative) training examples covered (resp. rejected) by the rule.

To prevent over-fitting, we consider that a rule is valid only if it covers at least 10% of the positives training examples. To prevent the inference of over-general rules, we constrain the inferred rules to accept a proportion of the positives larger than 1.5 times the proportion of accepted negative (*e.g.*, rules like `target(A,B):-gap(A,B).` are rejected thanks to this condition). Finally, based on results of preliminary experiments, we limited the explored part of the space by setting the parameters `nodes` to 50000 and `depth` to 7.

3.2 Design of Experiments

We designed experiments to answer the following questions: (Q_1) Which combinations of BKSs improve the results of the ILP inference processes, and which make it worse? (Q_2) Does the expected best combination of BKSs actually result in a predictor with significantly high predictive power?

To be able to answer (Q1), we have to sample the space of combinations of BKSs. We have eight different background knowledge types that we want to test ($\mathcal{G}_s$, $\mathcal{G}_u$, $\mathcal{L}et$, $\mathcal{P}_a$, $\mathcal{P}_s$, $\mathcal{P}ro$, $\mathcal{S}_n$, $\mathcal{S}_p$), which can be combined in $2^8 = 256$ different ways. We could not try all combinations because the running times are too long. We therefore had to sample the space of combinations. This was done using the technique known as *Taguchi design* [7]. The Taguchi method takes care to select a set of samples balanced with respect to the use of the different factors (here the BKSs), and of selected sets of interactions between

[1] To ensure there is a score for each of the eight sets of rules, each set is completed with a "default" rule, used when no other matches. The precision of the default rule is set as the number of rejected (by all other rules) Gi/o training examples over the number of rejected Gi/o and Gs/q examples.

the factors (*i.e.*, the effects of combining different BKSs). The effects of these factors and interactions can then be studied independently without sampling biases. We selected our Taguchi design among those available in the statistical software MINITAB (`http://www.minitab.com/`), taking the one allowing for the study of the largest number of interactions. This design requires 32 samples, (*i.e.*, inference over 32 different combinations of BKSs) and allows for the study of 20 interactions between two BKSs. Because gaps by themselves cannot produce interesting rules and are expected to interact, we chose to study interactions between $\mathcal{G}_u$ and all other BKSs, and $\mathcal{G}_s$ and all other BKSs. Other available interactions were fixed by MINITAB and are between $\mathcal{L}et$ and all other BKSs, between $\mathcal{P}_a$ and $\mathcal{S}_p$, and between $\mathcal{P}_a$ and $\mathcal{S}_n$.

To augment the statistical significance of our results, we used a 5-folds cross-validation: the number of the fold being considered as a noise parameter in the Taguchi design. We limited ourselves to a 5-folds due to execution time constraints: one combination of BKSs in a 5-folds experiment takes approximately 60 hours to run on a *SunBlade 2500* processor under *SunOS 5.8.* Hence a total of $60 * 32 = 1920$ hours of cpu time. ROC area on the cross-validation test sets has been chosen as a predictive performance measure. One of its main advantages is that it is independent of the proportions of classes in the test sets.

The analysis of the Taguchi experiments was used to answer question (Q_1). This analysis was conducted by examining Taguchi graphs (available on the web), and fitting the responses (*i.e.*, ROC areas) to a linear model of the different factors (the BKSs), and available interactions. The coefficients of the linear model provide indications of the amount of ROC area each BKS (or interaction) brings or remove to the total area. These coefficients are associated with p-values which represent their significance, *i.e.*, the estimated probability that the hypothesis "the effect is nul" is true. A value of 0.05 (*i.e.*, 5%) is a usual significance threshold. Finally, a R^2 value is provided, representing the percentage of the ROC area variation explained by the linear model.

Using the linear model, we can predict which combinations of BKSs will improve the ROC area. The best combinations were tested by using 10-folds cross-validation, enabling a final selection of the BKSs. (Q_2) was then answered by comparing the ROC area for this selection with random classification.

3.3 Experimental Results and Analysis

This section presents the statistical analysis of the results; full tables of results are available on the web. For the analysis, we first constructed a linear model with all available terms (*i.e.*, main effects and interactions). Then, assuming that effects with large p-values are random, as per Taguchi strategy in small design, interaction terms with large p-values (over 0.4) have been removed and a new model was constructed. The obtained model is given in Table 1. The lower bound estimation for the R^2 value of this model is 82.9%, *i.e.*, approximately 17% of the variation has to be explained by parameters not included in the model (*e.g.*, other unavailable second order interactions, or higher order interactions).

Table 1. Linear model of the ROC areas. P-values are expressed as percentages.

Main effects	Coef.	p-value	Interactions	Coef.	p-value
Constant	54.75	0.0	$\mathcal{L}et$-$\mathcal{P}ro$	-0.81	4.1
$\mathcal{L}et$	0.27	46.4	$\mathcal{L}et$-$\mathcal{S}_p$	-0.89	2.7
$\mathcal{P}ro$	2.49	0.0	$\mathcal{G}_s$-$\mathcal{L}et$	1.16	0.6
$\mathcal{G}_s$	2.59	0.0	$\mathcal{G}_s$-$\mathcal{P}_s$	0.92	2.2
$\mathcal{G}_u$	1.51	1.0	$\mathcal{G}_u$-$\mathcal{S}_n$	0.78	4.7
$\mathcal{P}_a$	0.02	95.2	$\mathcal{G}_u$-$\mathcal{S}_p$	0.70	7.3
$\mathcal{P}_s$	0.59	12.3	$\mathcal{P}_a$-$\mathcal{S}_n$	1.04	1.1
$\mathcal{S}_p$	0.39	30.2			
$\mathcal{S}_n$	0.55	14.8			

Table 2. Results for combinations of BKSs suggested by the linear model (4 first lines), and some complementary experiments (last 3 lines). The "Pred." column corresponds to predictions of mean ROC areas by the linear model on 5-folds experiments.

		5-folds			10-folds	
Combination		Mean	Med.	Pred.	Mean	Med.
BKS_1	$\mathcal{G}_s$-$\mathcal{G}_u$-$\boldsymbol{\mathcal{L}et}$-$\mathcal{P}_a$-$\mathcal{P}_s$-$\mathcal{P}ro$-$\boldsymbol{\mathcal{S}_p}$-$\mathcal{S}_n$	60.5	63.4	66.1	71.6	75.3
BKS_2	$\mathcal{G}_s$-$\mathcal{G}_u$-$\mathcal{P}_a$-$\mathcal{P}_s$-$\mathcal{P}ro$-$\boldsymbol{\mathcal{S}_p}$-$\mathcal{S}_n$	61.7	62.5	66.6	61.3	64.6
BKS_3	$\mathcal{G}_s$-$\mathcal{G}_u$-$\boldsymbol{\mathcal{L}et}$-$\mathcal{P}_a$-$\mathcal{P}_s$-$\mathcal{P}ro$-$\mathcal{S}_n$	60.6	61.8	65.7	71.4	75.0
BKS_4	$\mathcal{G}_s$-$\mathcal{G}_u$-$\mathcal{P}_a$-$\mathcal{P}_s$-$\mathcal{P}ro$-$\mathcal{S}_n$	61.2	61.3	62.7	57.9	66.7
BKS_5	$\mathcal{G}_s$-$\mathcal{G}_u$-$\mathcal{L}et$-$\mathcal{P}_s$-$\mathcal{P}ro$-$\mathcal{S}_p$-$\mathcal{S}_n$	61.0	67.1	63.9	69.8	74.6
BKS_6	$\mathcal{G}_s$-$\mathcal{G}_u$-$\mathcal{L}et$-$\mathcal{P}_a$-$\mathcal{P}_s$-$\mathcal{P}ro$-$\mathcal{S}_p$	60.7	61.8	61.3	70.3	73.1
BKS_7	$\mathcal{G}_s$-$\mathcal{G}_u$-$\mathcal{L}et$-$\mathcal{P}_a$-$\mathcal{P}ro$-$\mathcal{S}_p$-$\mathcal{S}_n$	65.3	65.9	63.0	59.1	60.4

Usefulness of $\boldsymbol{\mathcal{G}_u}$, $\boldsymbol{\mathcal{G}_s}$, $\boldsymbol{\mathcal{P}ro}$, $\boldsymbol{\mathcal{P}_a}$, $\boldsymbol{\mathcal{P}_s}$ and $\boldsymbol{\mathcal{S}_n}$. In the linear model $\mathcal{G}_u$, $\mathcal{G}_s$ and $\mathcal{P}ro$ have the largest main effects coefficients, significant at the 5% level. In addition, $\mathcal{G}_s$, $\mathcal{G}_u$, $\mathcal{P}_a$, $\mathcal{P}_s$, and $\mathcal{S}_n$ are shown to have positive interactions at the 5% significance level. This is strong evidence that these BKSs have to be used.

Interactions with gaps ($\mathcal{G}_s$ and $\mathcal{G}_u$) were expected: gaps cannot describe the sequences by themselves, and in fact the importance of their main effects can be seen as an indication that they positively interact with others background predicates most of the time.

The effects of $\mathcal{L}et$ and $\mathcal{S}_p$ are less clear than for the others BKSs. For $\mathcal{L}et$, two negative interactions are observed (with $\mathcal{P}ro$ and $\mathcal{S}_p$), and one positive with $\mathcal{G}_s$. For $\mathcal{S}_p$, in addition to the negative interaction with $\mathcal{L}et$, a positive interaction is observed with $\mathcal{G}_u$; this interaction is observed at a 10% significance level instead of a stronger 5% level for the negative interaction. Further experiments are therefore needed to prove utility of the $\mathcal{L}et$ and $\mathcal{S}_p$ BKSs.

Usefulness of $\boldsymbol{\mathcal{L}et}$ and $\boldsymbol{\mathcal{S}_p}$. The linear model suggests different combinations to test: always using $\mathcal{G}_s$, $\mathcal{G}_u$, $\mathcal{P}_a$, $\mathcal{P}_s$, $\mathcal{P}ro$ and $\mathcal{S}_n$, but adding or not $\mathcal{L}et$ and $\mathcal{S}_p$ (or both). Results for these 4 combinations both with 5-folds experiments (used for the Taguchi design) and 10-folds ones are in lines BKS_1 to BKS_4 in Table 2.

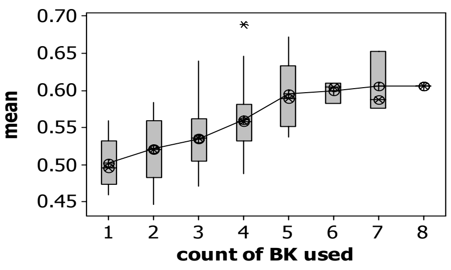


Fig. 2. Mean ROC areas as a function of the number of BKSs used. The plot contains points obtained for the Taguchi design, but also extra-points obtained during preliminary experiments.

The $\mathcal{L}et$ BKS: Wilcoxon Signed Rank tests have been used to test differences between the results of BKS_1 and BKS_2, and of BKS_3 and BKS_4[2] (*i.e.*, comparing inferences with and without $\mathcal{L}et$). On the 5-folds data, no evidence of differences in the results medians is available (at the 10% significance level). However, on the 10-folds data, their is evidence at the 5% significance level that BKS_2 has lower median than BKS_1; and at the 1% significance level that BKS_4 has lower median than BKS_3. The $\mathcal{L}et$ BKS can therefore be considered useful with strong evidence.

The $\mathcal{S}_p$ BKS: Wilcoxon Signed Rank tests between BKS_1 and BKS_3, and BKS_2 and BKS_4 (*i.e.*, comparing inferences with and without $\mathcal{S}_p$) do not detect differences, at the 10% significance level, between these results medians. Therefore, we do not have any statistical evidence that providing $\mathcal{S}_p$ changes inference results.

Is Classification Better Than Random? From the previous results, the best obtained combinations are BKS_1, followed very closely by BKS_3. Using a 1-Sample Wilcoxon test, it can be shown that, for both these combinations, the median is above random (*i.e.*, a value of 50.0), at the 10% significance level on the 5-folds, and at a strong 1% significance level on the 10-folds data. This confirms, after the work of [6], that protein grammars inferred by ILP can be useful for predicting protein functions: a stronger statistical evidence is provided in this work thanks to 10-folds cross-validation (holdout was used in [6]).

High order interactions. Each provided background predicate enlarges the search space, we therefore could expect performance to decrease when adding many BKSs. Two facts tend to show the effect of the search space size can be observed. First, the predictions from the linear model (column *Pred.* in Table 2) are most of the times lower than the results of the practical experiments. Second, Figure 2 shows a plot of mean ROC areas with respect to the number of BKSs used; on this figure, improvements in ROC areas are smaller when more than 5

[2] The Wilcoxon Signed Rank test is used instead of a more classical paired t-test since the differences of distributions cannot be assumed to be normal.

BKSs are provided. Both observations could be explained by the presence of a negative high order interaction like the search space size effect.

If such an interaction takes place, and that BKSs not studied in this work are considered for inference, using them in addition to proposed ones could lower the results. If this is observed, a solution would be to replace BKSs of this study having low contribution by the new one(s). The first BKS suggested for replacement, both by the linear model and by the 10-folds experiments, is $\mathcal{S}_p$. If this does not prove enough, Table 2 suggests removing $\mathcal{P}_a$ or $\mathcal{S}_n$.

Sensitivity to the Examples Count. Different ROC areas are often observed between the 5-folds and 10-folds results (*e.g.*, for BKS_1 in Table 2, but more can be seen on the data table available on the web). This may be due to a sensitivity of the ILP system to the size of the training set available to inference. It may also be due to higher quality BKSs being generated when more sequences are available.

Processing the BKSs. To obtain more insight on generated BKSs of Section 2.2, we ran 10-folds experiments using all BKSs except either $\mathcal{P}_a$, $\mathcal{P}_s$, $\mathcal{S}_n$ or $\mathcal{S}_p$ (Table 2). When using a Wilcoxon Signed Rank test to compare these results with BKS_1, the medians of the results are not shown different at the 10% significance level. However, the Wilcoxon Signed Rank test with the lowest p-value is with BKS_7, *i.e.*, when the $\mathcal{P}_s$ BKS is removed (p-value of 11.8%)[3]. $\mathcal{P}_s$ is also the generated BKS with the lowest p-value (12.3%) on the linear model. This makes us believe that $\mathcal{P}_s$ is likely to be the best generated BKS.

$\mathcal{P}_s$ is the only generated BKS which was not obtained directly from the examples, but obtained by processing another BKS. This encourages us to think that re-working the BKSs obtained from the examples is a possible way to improve further the inference results.

4 Conclusion

This work provides statistical evidence that all but one of the proposed BKSs are useful to inference, sometimes directly, sometimes through interactions with each other. It also provides further confirmation, after the work of Muggleton *et al.* [6] that ILP can help to predict protein functions.

Other sources of background knowledge have still to be studied, these include known regular expressions (*e.g.*, from the Prosite database [11]), but also probabilistic grammars (*e.g.*, weight matrices or Markov models).

Acknowledgements. This work is funded by EPSRC grant GR/S68682. We would like to acknowledge the contributions made by the Systems Research, Transgenics & Gene Cloning, and Gene Expression & Protein Biochemistry groups at GlaxoSmithKline to the proprietary GPCR classification data.

[3] This p-value is not smaller despite the large difference in median of BKS_1 and BKS_7 because these combinations do not perform well on the same folds.

References

1. Muggleton, S., *et al.*: Protein secondary structure prediction using logic-based machine learning. Protein Eng. **5** (1992) 647–657
2. Mozetic, I.: Secondary structure prediction by inductive logic programming. In: Proc. 3rd Meeting on the Critical Assessment of Techniques for Protein Structure Prediction, CASP3. (1998) pp. A–26
3. Srinivasan, A., *et al.*: Theories for mutagenicity: A study in first-order and feature-based induction. Artificial Intelligence **85**(1-2) (1996) 277–299
4. King, R.D.: Applying inductive logic programming to predicting gene function. AI Mag. **25**(1) (2004) 57–68
5. Clare, A., *et al.*: The ILP'05 challenge. `http://www.protein-logic.com/index.html` (2005)
6. Muggleton, S.H., *et al.*: Are grammatical representations useful for learning from biological sequence data? – a case study. Jour. Comp. Biol. **5**(8) (2001) 493–522
7. Taguchi, G.: Introduction to quality engineering. Asian Productivity Organization, Tokyo (distributed by American Supplier Institute, Inc., Dearborn,MI.) (1986)
8. Searls, D.B.: String variable grammar: A logic grammar formalism for the biological language of DNA. Jour. of Log. Prog. **12** (1993)
9. Dsouza, M., *et al.*: Searching for patterns in genomic data. Trends in Genetics **13**(12) (1997) 497–498
10. Brazma, A., *et al.*: Discovering patterns and subfamilies in biosequences. In: Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology, AAAI Press (1996) 34–43
11. Falquet, L., *et al.*: Protein data bank. Nucleic Acid Research **30** (2002) 235–238
12. Leung, S.W., *et al.*: Basic Gene Grammars and DNA-ChartParser for language processing of *Escherichia coli* promoter DNA sequences. Bioinformatics **17**(3) (2001) 226–236
13. Bateman, A., *et al.*: The Pfam protein families database. Nucleic Acids Research **32** (2004) D138–D141
14. Sakakibara, Y., *et al.*: Stochastic context-free grammars for tRNA modeling. Nucleic Acids Research **22** (1994) 5112–5120
15. Cussens, J., Pulman, S.: Experiments in inductive chart parsing. In Cussens, J., ed.: LLL'99, Bled, Slovenia (1999) 72–83
16. Pereira, F., Warren, D.H.D.: Definite clause grammars for language analysis – a survey of the formalism and a comparison with augmented transition networks. Artificial Intelligence **13**(3) (1980) 231–278
17. Apostolico, A., *et al.*: Verbumculus and the discovery of unusual words. Jour. Comp. Sci. and Tech. **19**(1) (2003) 22–41
18. Pierce, K., *et al.*: Seven-transmembrane receptors. Nat Rev Mol Cell Biol **3(9)**(6) (2002) 39–50
19. Sgourakis, N., *et al.*: Prediction of the coupling specificity of GPCRs to four families of G-proteins using hidden markov models and artificial neural networks. Bioinformatics **21(22)** (2005) 4101–4106
20. Altschul, S., *et al.*: Gapped blast and psi-blast: a new generation of protein database search programs. Nucleic Acids Res. **25**(17) (1997) 389–402

Improving Bayesian Network Structure Search with Random Variable Aggregation Hierarchies

John Burge and Terran Lane

University of New Mexico, Department of Computer Science
{lawnguy, terran}@cs.unm.edu

Abstract. Bayesian network structure identification is known to be NP-Hard in the general case. We demonstrate a heuristic search for structure identification based on *aggregation hierarchies*. The basic idea is to perform initial exhaustive searches on composite "high-level" random variables (RVs) that are created via aggregations of atomic RVs. The results of the high-level searches then constrain a refined search on the atomic RVs. We demonstrate our methods on a challenging real-world neuroimaging domain and show that they consistently yield higher scoring networks when compared to traditional searches, provided sufficient topological complexity is permitted. On simulated data, where ground truth is known and controllable, our methods yield improved classification accuracy and structural precision, but can also result in reduced structural recall on particularly noisy datasets.

Keywords: Bayesian network structure search hierarchy fMRI.

1 Introduction

Bayesian networks (BNs) [17] are a widely employed graphical modeling framework used to reason under uncertainty. Their topological structures describe correlational (or possibly causal) relationships among random variables (RVs). This topology may not be known a priori and must be searched for—a process known to be NP-hard in the general case [4]. Instead of directly learning the structure for a BN with a large number of RVs, we propose that searches may first be performed on simpler domains whose RVs are constructed as the aggregation of the original domain's RVs. The results of these searches can then influence searches on the original domain via structural priors or as modifications to search heuristics which allow exhaustive searches on constrained structure spaces.

Our approach is analogous to the statistical problem of blood pooling. Assume that a blood test for some disease must be performed on many patients but is expensive and cannot be applied exhaustively. Instead, blood samples are divided into a small number of groups and pooled into aggregate group samples. Results from the pooled samples can then be used to constrain the application of the test to the individual samples by only testing the individual constituents of a positive group sample.

Just as results on the pooled group samples indicate which individual samples to test, elicited correlations among composite RVs can guide elicitation of correlations among atomic RVs. To illustrate this, consider an example from a neurological domain. At a fine level, some neuroanatomical databases break up the human brain into approximately 70 regions of interest (ROIs). The search space for a BN with 70 RVs contains on the order of 10^{23} structures and cannot be searched exhaustively. However, the neuroanatomical databases can aggregate these ROIs into roughly 50 ROIs, which can then be further aggregated into 12 and then 7 ROIs. A BN with only seven nodes requires roughly 1,000 structures to be searched and could be performed exhaustively. Results from this search could then be used to constrain searches among finer RVs under the assumption that correlations among those RVs will be observable as correlations among the gross ROIs they compose.

We demonstrate our methods on such a neuroimaging domain, but there are many other domains where RVs may be sensibly aggregated together. E.g., other image analyses where pixel neighborhoods of varying size can be grouped together; geographic data such as cities, states and countries; genetic regulatory network reconstruction where genes can be grouped into families and super-families; document topic hierarchies (e.g., newsgroups); word types in grammar trees; medical diagnoses where diseases and symptoms are grouped into sub-categories; and Fourier and wavelet analyses where coefficients are spatially and temporally related.

Typically, the RV aggregations can be arranged into a hierarchy. To form this hierarchy, two domain-specific questions must be answered. First, which RVs should be aggregated together and second, what function should perform the aggregation? In the neuroimaging domain, we group ROIs together based on spatial and functional locality and aggregate them as a weighted linear combination.

Of course, the assumption that correlations will persist across the aggregation hierarchy will be violated to some degree in most domains. Further, while constraining subsequent structure searches based on previous structure results is intuitive and appealing, straightforward implementations can yield unfavorable results. We empirically demonstrate this and propose a constraint mechanism which performs well. For both generative and class-discriminative scores, our methods consistently yield higher scoring structures than traditional searches on four neuroimaging datasets collected under widely differing paradigms, provided that the search is allowed to produce BNs with sufficient structural complexity—typically two to three parents per node. On a simulated domain, in which ground truth is known and controllable, we demonstrate higher classification accuracy and structural precision, but also lowered structural recall on particularly noisy datasets.

2 Background

Bayesian Networks (BNs) [17] are graphical models that explicitly represent dependencies among RVs. A BN's topological structure, represented as a directed acyclic graph, contains nodes for RVs and directed links between correlated *parent* and *child* nodes. A *family* is composed of a single child and its parents. We assume fully observable discrete RVs so that a family's conditional probability, $P(child \mid parents)$, can be represented with a conditional probability table (CPT).

Searching for a BN's topology is accomplished by proposing as many hypothesis structures as possible, guided by a search heuristic, while measuring the goodness of fit between the structures and the data via a structure scoring function. Iterative hill climbing heuristics are commonly employed. For example, starting with a topology with no links, score all legal modifications to the topology where a legal modification is the addition, removal or reversal of a link not resulting in a cycle. Choose the modification that results in the highest score and iterate until no modifications yield improvements. We refer to this as a flat structure search.

Structure scoring functions typically come in two varieties: generative and class-discriminative. Generative scores select structures that increase the posterior likelihood of the data given the structure. Common examples include MDL [13], BIC [18], BDe [10], etc. Discriminative scores select structures that increase the class discriminative ability of learned BNs. Examples include the class-conditional likelihood (CCL) [8] and the approximate conditional likelihood (ACL) [2]. With the notable exception of CCL, most scores are decomposable, i.e., a family's contribution to the score is independent of all other families' topologies.

We use the following notation. Let $\boldsymbol{X}$ represent a set of n RVs, $\{X_1, X_2, \ldots, X_n\}$ with arities $r_1, r_2, \ldots, r_n$. A data point is a fully observable assignment of values to $\boldsymbol{X}$. A BN, B, over $\boldsymbol{X}$ is described by the pair $\langle B_S, B_\Theta \rangle$. B_S is the DAG representing the BN's structural topology. X_i's parent set is denoted $Pa(X_i)$. q_i is the number of configurations for the RVs in $Pa(X_i)$. $B_\Theta = \{\Theta^B_{i,j,k} : 1 \le i \le n, 1 \le j \le r_i, 1 \le k \le q_i\}$ is the set of CPT parameters where $\Theta^B_{i,j,k} = P(X_i = j \mid Pa(X_i) = k)$. $I_{X',Y'}$ is an indicator function which equals one iff there exists a link between RVs in the sets $\boldsymbol{X'}$ and $\boldsymbol{Y'}$. Finally, $I_{\varnothing,\varnothing} = I_{X',\varnothing} = I_{\varnothing,X'} = 1$ and $\overline{I}_{X',Y'} = 1 - I_{X',Y'}$.

2.1 Aggregation Hierarchies

Decomposing a complex model into a series of hierarchically related components has been shown to be helpful in many domains. For example, Fine, Singer and Tishby [7] introduce a hierarchical abstraction of hidden Markov models; Gyftodimos and Flatch [9] introduce a hierarchical abstraction of BNs in general and Sutton, Precup & Singh [19] incorporate hierarchies within reinforcement learning.

As in this previous work, we hierarchically decompose a domain into multiple models of varying complexity. Setting our work apart from much of the prior work, we use structural results learned in one model to guide learning in subsequent models. To our knowledge, we are the first to do this with BN structure search, though similar methods for BN parameter learning have been proposed. E.g., Anderson, Domingos and Weld [1] and McCallum et al. [15] use shrinkage to improve parameter learning by combining varying levels of bias and variance in hierarchically related models.

To form our hierarchies, we create *composite RVs* as aggregations of a domain's original *atomic RVs*. Let $\hat{X} = \{X_1, \ldots, X_\tau\} \in \boldsymbol{X}$. A scalar function of $\hat{X}$, $Y = \xi(\hat{X})$, is an *aggregation function* where Y is a RV whose distribution reflects some aspect of the joint distribution of $\hat{X}$. Common examples of aggregations include *max*, *count*, *variance*, etc. For our neuroimaging domain, we employ the *weighted mean aggregate*, $\xi(\hat{X}) = \sum_{i=1}^{\tau} \alpha_i X_i$, where the α_i's are set by a neuroanatomical database.

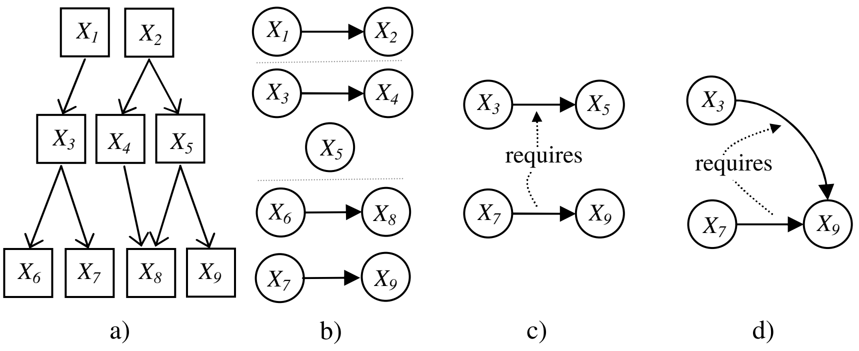


Fig. 1. a) An example hierarchy detailing the hierarchical trellis for the RVs in $X = \{X_1, \ldots, X_9\}$. Boxes are used to emphasize this is not a BN. b) An example BN defined for X. The dotted lines indicate a division between RVs in different hierarchical levels. The link between X_7 and X_9 does not satisfy either hierarchical assumption. c) For the family-wise assumption, the $X_7 \rightarrow X_9$ link requires the existence of the $X_3 \rightarrow X_5$ link. d) For the parent-wise assumption, the $X_7 \rightarrow X_9$ link requires the existence of the $X_3 \rightarrow X_9$ link.

X includes both the atomic RVs as well as the composite RVs. An aggregation hierarchy over ***X*** (Figure 1a) can be graphically represented as a *trellis*. A trellis is a relaxation of a forest such that each node may have multiple parents. Let Λ_X be a trellis over ***X*** whose leaves are atomic RVs and whose internal nodes are aggregations of their children. Let Λ_{X_i} denote the children of X_i, Λ^{X_i} denote the parents of X_i, Λ_i denote the integer-valued level at which X_i is located in and $\Lambda_{X_i}^{X_i} = \Lambda_{X_i} \cup \Lambda^{X_i}$. $\Lambda_{X_i} = \varnothing$ for leaves and $\Lambda^{X_i} = \varnothing$ for the root(s). If X_i is not a leaf node then $X_i = \xi(\Lambda_{X_i})$. $\Lambda(v)$ is the set of RVs at the v^{th} level in the hierarchy and is referred to as an *h-level*. *levels*(Λ_X) returns the number of h-levels in Λ_X. Relationships among RVs, such as *parent* and *child*, are prefixed with *h-* when used in the context of a hierarchy.

2.2 Hierarchical Bayesian Network Structure Search

Exhaustive structure searches can be employed at the highest h-levels where few RVs reside. Searches at lower h-levels cannot normally be performed exhaustively, but can be constrained by the previous h-level's search results. One possible constraint mechanism is based on the assumption that links among high-level RVs will be manifested as links among the low-level RVs they were constructed from. Exhaustive search strategies can then be employed for nodes in the lower h-levels on the space of structures that only contain links obeying this assumption.

Take the BN and the hierarchy in Figure 1 for example. There are two nodes at the highest h-level and structure search is trivial. Assume that a structure search found the $X_1 \rightarrow X_2$ link. According to the hierarchy, $\{X_3\}$ and $\{X_4, X_5\}$ are the h-children of X_1 and X_2. As a correlation between X_1 and X_2 was observed, correlations among their constituents should be searched for. That search could yield, for instance, the $X_3 \rightarrow X_4$ link and then, at the next h-level, the $X_6 \rightarrow X_8$ link. Of course, limiting searches based on this assumption results in links that will not be searched. E.g., the $X_7 \rightarrow X_9$ link will not be searched since there is no link between X_7 and X_9's h-parents, $\{X_3\}$ and $\{X_5\}$. We call this assumption the *family-wise assumption* as the relationships detailed in a family at one level are manifested as families at lower h-levels.

Definition. The *family-wise assumption.* $\forall$ X_i and X_j, if $I_{\Lambda^{X_i},\Lambda^{X_j}} = 0$ or $\Lambda_i \neq \Lambda_j$, then $X_i \notin Pa\left(X_j\right)$,

This assumption does not allow a node's parent set to include its h-parents and h-children, which, given that a node is constructed from its h-children, are likely to be significant. We relax this assumption to allow this.

Definition. *Relaxed family-wise assumption.* $\forall$ X_i and X_j, if $I_{\Lambda^{X_i},\Lambda^{X_j}} = 0$ and $X_i \notin \Lambda_{X_j}^{X_j}$, then $X_i \notin Pa\left(X_j\right)$

The relaxed assumption does not limit candidate parent sets as effectively as the unrelaxed assumption (particularly in dense trellises) and is less likely to allow for exhaustive searches. Ultimately, this will lead to poor search performance. Hence, we introduce the *parent-wise assumption* which only requires a correlation between two RVs to manifest as a correlation between the child and one of the parent's h-parents.

Definition. *Parent-wise assumption.* $\forall$ X_i, X_j, if $I_{\Lambda^{X_i},X_j} = 0$, then $X_i \notin Pa\left(X_j\right)$.

This requirement is not as strict as the family-wise assumption and has the distinct advantage of easily incorporating a node's h-relatives as candidate parents while still effectively restricting structure spaces. Figures 1c and 1d illustrate the different link requirements for the family-wise and parent-wise assumptions.

These assumptions may be incorporated directly into the BN scoring function. Scoring functions include terms (or can generally be modified to include terms) that probabilistically weight structures based on prior knowledge. Formulating domain knowledge as a structural prior is advantageous as it can be easily incorporated into many structure scores. $P_{rf}(B_S)$ and $P_p(B_S)$ give the relaxed family-wise and parent-wise assumptions as structural priors, respectively:

$$P_{rf}(B_S) = \frac{Z}{1+\alpha v_{rf}}, v_{rf} = \sum_{i \times j} I_{\{X_i\},\{X_j\}} \overline{I}_{\Lambda^{X_i},\Lambda^{X_j}} \overline{I}_{X_i,\Lambda_{X_i}^{X_i}}, \quad P_p(B_S) = \frac{Z}{1+\alpha v_p}, v_p = \sum_{i \times j} I_{\{X_i\},\{X_j\}} \overline{I}_{\Lambda^{X_i},\{X_j\}},$$

where Z is a normalization constant, α is a penalty scale factor, and v_{rf} and v_p are the number of links that violate the relaxed family-wise and parent-wise assumptions. When α is sufficiently large, the prior probability of a structure with any violating links can be treated as zero. Incorporation of the hierarchical assumptions can then be equivalently realized as modifications to structure search heuristics by limiting candidate parent sets. It is this case we investigate in this paper, though, future work investigating the case where α is relatively small is also promising.

For the relaxed family-wise assumption, structure search begins by exhaustively searching for the optimal parent sets for each $X_i \in \Lambda(1)$. Structure searches for the remaining h-levels are then iteratively performed with the structural results of prior h-levels constraining candidate parent sets (CPSs) at subsequent h-levels. The runtime of hierarchical structure searches will typically be longer than that of flat searches but ultimately depends on the CPS limits where exhaustive searches are allowed. We have found that it is reasonable to exhaustively search for a node's optimal parent set when its CPS has less than 20 parents, to use a simulated annealing search when its CPS has less than 40 parents and to resort to a hill climbing search otherwise. We refer to this search as the *RFW-Hier* search.

Unlike searches based on the family-wise assumption, searches based on the parent-wise assumption would require RVs to have many simultaneous parents—far more than would be allowed due to overfitting and computational limitations. This can be addressed by searching for the optimal candidate parents for a node one h-level at a time using the following heuristic. For each RV X_i, exhaustively search for the highest scoring set of n legal parents from $\Lambda(1)$. Record and remove these parents. Then, find the best set of n legal parents for each X_i from each subsequent h-level where the recorded results from the prior h-level constrain the parent sets. This results in a final set of (at most) $n \times \mathit{levels}(\Lambda)$ recorded parents. Perform one last search through this set for the final parent set. As in the RFW-Hier search, we use a combination of exhaustive, simulated annealing and greedy searches. We refer to this search as the *PW-Hier* search.

When searching through CPSs one node at a time, cycles could be introduced into the topology. We address this by placing constraints that ensure no cycles can exist (see Section 3). Other methods for dealing with the introduction of cycles exist, e.g., a *repair* operator that removes cycles that have been introduced [14].

3 Experiments

We test on both simulated and real-world neuroimaging domains. The neuroimaging data is temporal and BNs that explicitly represent time are referred to as *dynamic Bayesian Networks* (DBNs). The simulated data is generated from DBNs.

In the most general case, DBNs include one column of RVs for every time step and one node in each column for every RV. For most real world problems, such DBNs are intractably large. We make the *stationary* and *Markov order 1* assumptions, resulting in a topology of two columns: one for time t and one for time $t+1$. The nodes do not represent absolute time points but instead represent RV correlations averaged across time. Links originate in the left column and terminate in the right. DBNs may also include isochronal links, which we omit as temporal links are of primary interest. Thus, all link additions are guaranteed to be acyclic.

Notation for DBNs is slightly modified from BNs in general. X_i^t and X_i^{t+1} represent the ith RV in columns t and $t+1$ and $\boldsymbol{X} = \{X_i^t, X_i^{t+1}: 1 \le i \le n\}$. The parameters for a node's CPT, $P_B(X_i^{t+e} \mid Pa(X_i^{t+e}) = j)$, are denoted $\Theta^B_{e,i,j,k}$, $e \in \{0,1\}$.

We gauge the efficacy of our heuristics using both generative and discriminative scores. For a generative score, we use the BDe metric [10], a commonly employed metric with a strong mathematical underpinning. Its parameter priors are themselves parameterized by the *equivalent sample size* (ESS), which has the effect of controlling for structural complexity. For a discriminative score, we use the *approximate conditional likelihood* (ACL) score [2], a decomposable alternative to CCL.

3.1 Simulated Domain

Simulated data is created from a pair of DBNs whose topologies are selected at random but comply with the parent-wise hierarchical assumption (structures consistent with the relaxed family-wise assumption were omitted due to space constraints, but results were qualitatively very similar). We test the ability of both flat and hierarchical searches to find the underlying generative structure. Three experimental paradigms are used: an IID case in which data is generated from DBNs with varying magnitudes of differences, a noisy case in which the IID assumptions are violated and a case where hierarchical assumptions are violated.

In all cases, a single hierarchy, Λ_X, over RVs $\boldsymbol{X} = \{X_1, \ldots, X_{57}\}$, is created with 3 h-levels containing 3, 9 and 45 nodes. The hierarchy is a perfectly balanced tree with each node in $\Lambda(1)$ linking to three unique node in $\Lambda(2)$, each of which, in turn, links to five unique nodes in $\Lambda(3)$. The two generating DBNs, G_1 and G_2, are constructed with nodes $\{X_1^t, X_1^{t+1}, \ldots, X_{57}^t, X_{57}^{t+1}\}$. Fifteen links—one between nodes in $\Lambda(1)$, four between nodes in $\Lambda(2)$ and ten between nodes in $\Lambda(3)$—are created between 15 parents in the t column and 15 unique children in the $t+1$ column.

The *correlational strength* for a link, measured via a normalized mutual information score (NMIS), is determined by the CPT generated for the child node. At an NMIS of zero, a parent is completely uncorrelated with its child and at an NMIS of one, it is completely correlated. A node with no parents is parameterized by a normalized information score (NIS). At an NIS of zero, the CPT is completely non-uniform and at one its uniform.

The method for generating a CPT for a node X_i^{t+e} that conforms to a NIS or a set of NMISs is outside the scope of this paper. We will refer to it as the distribution $P(\Theta_{e,i}^B \mid S_{e,i}^B)$, where $e \in \{0,1\}$ and $S_{e,i}^B$ is a set containing a single NIS if X_i has no parents, or is a list of NMIS's, with an NMIS for each of the p parents. $\Theta_{e,i}^B$ can be modified to produce a new CPT, $\Theta_{e,i}'^B$, compliant with a different NIS or list of NMIS's, $S_{e,i}'^B$. This generator is the distribution $P(\Theta_{e,i}'^B \mid \Theta_{e,i}^B, S_{e,i}'^B)$. The closer $S_{e,i}'^B$ is to $S_{e,i}^B$, the smaller the KL divergence between $\Theta_{e,i}'^B$ and $\Theta_{e,i}^B$ will be. If $S_{e,i}'^B = S_{e,i}^B$, then $\Theta_{e,i}'^B = \Theta_{e,i}^B$.

The CPTs in G_1 for nodes with no parents are generated from the $P(\Theta_{e,i}^B \mid \{0.9\})$ distribution, yielding fairly uniform CPTs. Nodes with parents are generated from the $P(\Theta_{e,i}^B \mid \{0.1\})$ distribution so that a child's value is only loosely correlated with the parent's value. G_1 and G_2 share an identical structure and all the CPTs in G_2 are copies of those in G_1. Thus, initially G_1 and G_2 represent the same distribution.

The overall process for an experiment is as follows. First, the CPT parameters in G_1 and/or G_2 are modified in accordance to a particular experimental paradigm. Twenty training and twenty testing data points are generated for each class. A DBN is then learned for each class with the BDe score. Classification is performed on a testing data point by selecting the DBN with the largest posterior probability. *Structural precision*, the fraction of links in the learned DBNs present in the generating DBNs, and *structural recall*, the fraction of links in the generating DBNs also found in the learned DBNs, are measured. Each point listed in the resulting graphs in Figures 2 and 3 are calculated as the average of 120 runs of the experiment. Significance tests were computed via the t-test for dependent samples.

3.2 Neuroscience Domain

Functional magnetic resonance imaging (fMRI) is widely used in the study and diagnosis of mental illness. It is a non-invasive technique measuring the activity of small cubic regions of brain tissue (voxels). Psychologists frequently use fMRI data to test hypotheses about the changing neural activity underlying mental illness.

There are too many voxels in each 3D fMRI image to model directly, so voxels are marginalized to *regions of interest* (ROIs) via the widely employed Talairach database [12]. Thus, each image is represented as the activation of 147 ROIs. Then, the time series for each ROI is modeled with a temporal RV. Data for each class of patient, healthy vs. diseased, is grouped together and each class is modeled with a DBN containing the nodes $\boldsymbol{X} = \{X_i^t,\ X_i^{t+1} : 1 \leq i \leq 147\}$. The 147 ROIs are hierarchically related via the *mean aggregate function* given in Section 2.1.

We analyze four fMRI datasets collected under widely differing experimental paradigms on different patient populations suffering from different illnesses. The first was collected by Buckner et al. [3] for analysis of senile dementia, the second and third datasets were collected by the Clark et al. [5] and The Mind Institute [16] for schizophrenia and the fourth dataset was collected by Kiehl [11] and also focused on schizophrenic patients. We will refer to these datasets as the *demented*, *schizoM1, schizoM2* and *schizoK* datasets, respectively.

4 Results

The first set of simulated experiments measures how each search performs under IID conditions (Figure 2, top left). The CPTs in G_2 for the nodes with parents are redrawn from the $P(\Theta_{1,i}^{G_2} | \Theta_{1,i}^{G_2}, \{0.1 \pm c\})$ distribution where c determines the magnitude of difference between G_2's and G_1's CPTs. Addition versus subtraction is chosen at random. As c increases, the difference between classes increases. When $c = 0$, classification is impossible and when $c = 0.02$, classification is trivial.

PW-Hier's accuracy is significantly higher than the flat search's over a wide range. This is due to the increased structural precision of the PW-Hier search. Since PW-Hier decreases the candidate parent space for each node, many candidate parents are omitted which would have only contributed noise. Thus, the flat search is much more likely to add a superfluous node as a parent. Approximately one parent was added per child on average in the flat search compared to only 0.3 in the PW-Hier search.

The magnitude of the structural precision increase is due to the BDe equivalent sample size (ESS), which was set to 500. Figure 2 (top right) gives the results of experiments with c fixed at 0.005 and the ESS varying from 50 to 1,000. As the ESS increases, the search is more likely to add noisy parents. This decreases precision for both classifiers, however, PW-Hier's additional constraints counteract this tendency and its structural precision drops less quickly than the flat search's does.

For most domains, assuming data points are drawn from a noiseless process is unrealistic. The second set of experiments measures a score's tolerance to intra-class noise (Figure 2, bottom left). G_1 and G_2 are treated as *base-line* models, but each data point is generated from a modified version of them. Both G_1 and G_2 are generated as in the first experiment with $c = 0.005$ and the ESS = 500. G_α^g, the generator for the

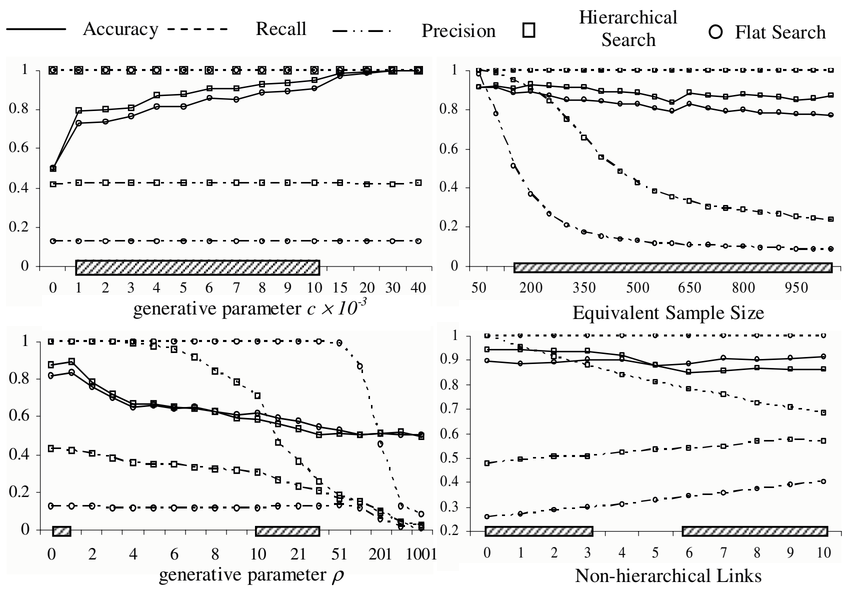


Fig. 2. Classification accuracy, structural recall and structural precision for simulated data experiments. Hierarchical results shown are for the PW-Hier search. RFW-Hier results on simulated data are omitted due to space constraints, but are qualitatively very similar. Shaded boxes on the axis indicate ranges where classification accuracy differences are statistically significant, as measured by the standard t test for dependent samples.

g^{th} generated data point for class α, starts as a copy of G_α. For each X_i, ρ random $\langle j,k \rangle$ tuples are chosen, $1 \le j \le r_i$, $1 \le k \le q_i$, and 0.1 is added to $\Theta^{G^g_\alpha}_{1,i,j,k}$. As ρ increases, intra-class differences increase and class discrimination and the base models' true RV correlations becomes more difficult to elicit.

Initially, when ρ equals *1* or *2,* PW-Hier's accuracy is significantly higher than the flat classifier's accuracy. As more randomizations occur, the flat classifier's accuracy catches up and eventually surpasses PW-Hier's. While the structural precision of the PW-Hier search always dominates the flat search, its structural recall begins to diminish significantly before that of the flat classifier. This is because losing the ability to identify a single link can cause a cascade of failures to identify other links. Not recognizing a link at high levels in the hierarchy automatically results in missing all links that depend on it. In the flat classifier, losing any particular link does not increase the risk of losing further links. So in particularly noisy datasets, PW-Hier's structural precision advantage may be overwhelmed by a decrease in structural recall.

Further, in real-world data, it is possible that the candidate parents that PW-Hier omits would be useful. The final set of simulated experiments (Figure 2, lower right) demonstrate what occurs as the number of links in the generative DBNs that *do not* conform to the parent-wise assumption are added. As expected, as the number of violating links increase, the accuracy of the flat classifier catches up and surpasses that of the PW-Hier classifier. At roughly five violating links, corresponding to 20%

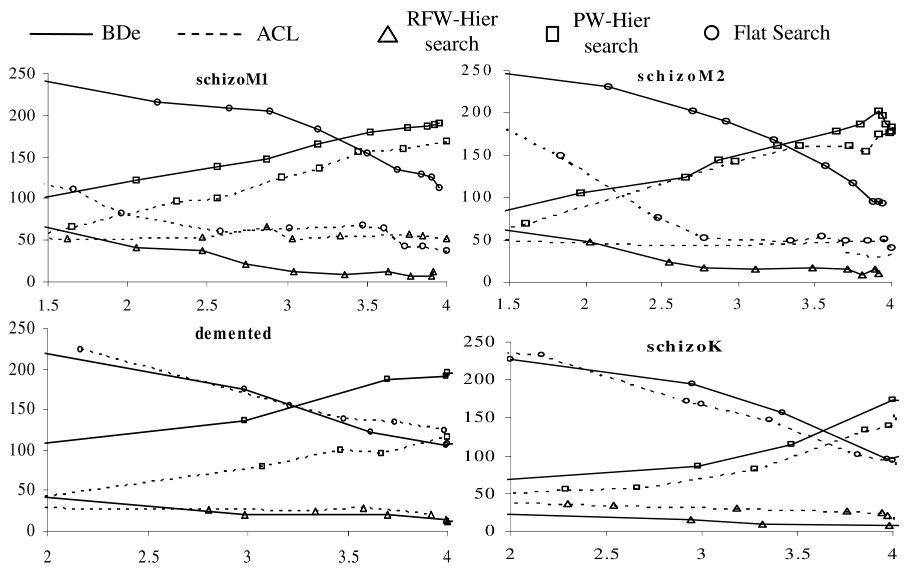


Fig. 3. The number of families for which a search returned the highest scoring parent set. The x-axis gives the average number of parents per family. The PW-Hier search outperformed the flat search in all but the demented dataset with the ACL score provided a certain threshold of complexity was achieved (ranging from 2 to 3.5 average parents).

of the total generative links, the flat classifier and the hierarchical classifier's accuracy are identical. Importantly, as the number of violations increase, PW-Hier's performance degrades gradually, indicating robustness to violations.

4.1 Neuroscience Domain Results

DBNs learned from fMRI data can be employed for several tasks, including the elicitation of correlations among ROIs, classification, creation of simulated surrogate datasets, specific hypothesis testing, etc. For all these tasks, learned DBNs are found by maximizing a scoring metric. In this section, we focus on high scoring networks as a proxy for the myriad of tasks that those networks may eventually be used for.

Figure 3 shows the results for DBNs learned with varying levels of structural complexity (where complexity is measured as the average number of parents per node). For networks learned with BDe, complexity is controlled by the ESS. For networks learned with ACL, complexity is controlled by a minimum description length (MDL) penalty term. Each point in the graph represents the number of families for which the corresponding search returned the highest score. For example, if the highest scoring set of parents for a child node X_i found with a flat search resulted in a ACL score of 15.6, but the *PW-Hier* search found a set of parents for X_i that resulted in a ACL score of 21.7, one point would be added on the y-axis for the *PW-Hier* search results. (Graphs that directly plot structure scores are not shown as they are complicated by complexity penalty trends, but such graphs do not qualitatively differ from those given in Figure 3.)

The results are consistent across each of the datasets. Initially, when the structural complexity is low, the flat search yields family structures with higher scores than the hierarchical search. This is because the hierarchical assumptions are restricting candidate parent sets too dramatically. However, after a certain critical threshold of complexity is reached, around 3.4 parents for BDe and anywhere from 2 to 4 parents for ACL, PW-Hier searches find higher scoring structures than flat searches.

The RFW-Hier search is almost always outperformed. The RFW-Hier search was simply incapable of restricting candidate parents to small enough sets where exhaustive strategies could be used, a key advantage in limiting parent sets to begin with. On the other hand, the PW-Hier search was capable of restricting candidate parents to smaller sets, benefited from exhaustive searches and was capable of outperforming typical flat structure searches on both generative and class discriminative scoring functions.

5 Conclusions

Employing hierarchically related models of varying complexity has proven to be useful in many machine learning applications. We have applied this concept to Bayesian network structure search by aggregating *atomic* random variables (RVs) into a hierarchy of *composite* RVs. Structural results of searches on high-level composite RVs are used to constrain searches on lower-level atomic RVs, allowing exhaustive searches for many of the BN's families.

We introduced two constraint heuristics for restricting searches at one h-level based on the search results at the previous h-level. On both a generative score, BDe [10], and a class-discriminative score, ACL [2], we demonstrated use of these heuristics on multiple datasets in a challenging real-world neuroimaging domain. We empirically showed that the intuitively reasonable *family-wise* search performed poorly while the *parent-wise* search significantly and consistently outperformed traditional, flat structure searches in finding high-scoring families. Results from a simulated domain, in which ground truth was known and controllable, indicated that hierarchical searches increased structural precision and yielded significant improvements to classification. Though, on particularly noisy datasets, a decrease in structural recall was observed which led to decreased classification accuracy.

Our empirical results primarily focused on domains where links between atomic and composite RVs were desirable. This may not be the case in all domains. Unfortunately, the parent-wise search is not useful in such domains, and the family-wise search may not yield desirable results due to its inability to adequately constrain candidate parent sets given dense trellises (such as those used in our neuroimaging domain). Additional work to determine if the family-wise search benefits domains with sparser trellises is warranted, however, as experiments on simulated data indicated similar benefits to the parent-wise search. Another avenue for future work lies in applying our methods to structure searches in relational learning paradigms, whose models contain hierarchies of RVs related with *is-a* and *has-a* relationships.

Acknowledgements. We would like to thank the anonymous reviewers for their valuable feedback as well as the Dartmouth fMRI data center [6], Buckner et al. [3], The Mind Institute [16], Clark et al. [5] and Kiehl [11] for providing us access to their fMRI datasets. This project was partially funded through grant DA012852 from the NIDA, NIH, from NIMH grant number 1R01MH076282-01 as part of the NSF/NIH Collaborative Research in Computational Neuroscience Program and The MIND Institute, DOE Grant DE-FG02-99ER62764.

References

[1] Anderson, C., Domingos, P. and Weld, D. Relational Markov models and their application to adaptive web navigation. *KDD*, 143-152, 2002.

[2] Burge, J., Lane, T. Learning Class-Discriminative Dynamic Bayesian Networks. *ICML*, 22:97-104, 2005.

[3] Buckner, R. L., Snyder, A., Sanders, A., Marcus, R., Morris, J. Functional Brain Imaging of Young, Nondemented, and Demented Older Adults. *Journal of Cognitive Neuroscience*, 12, 2. 24-34, 2000.

[4] Chickering, D., Geiger, D., Heckerman, D. Learning Bayesian Networks is NP-Hard. Technical Report MSR-TR-94-17, Microsoft, 1994.

[5] Clark, V.P., Friedman, L., Manoach, D., Ho, B.C., Lim, K., Andreasen, N. A collaborative fMRI study of the novelty oddball task in schizophrenia: Effects of illness duration. *Society for Neuroscience Abstracts*, 474.9, 2005.

[6] Dartmouth fMRI Data Center, The. http://www.fmridc.org/f/fmridc , 2006.

[7] Fine, S., Singer, Y., Tishby, N. The Hierarchical Hidden Markov Model: Analysis and Applications. *Machine Learning*. vol. 32, 41-62, 1998.

[8] Grossman, D., Domingos, P. Learning Bayesian Network Classifiers by Maximizing Conditional Likelihood. *ICML*, 21, 361-368, 2004.

[9] Gyftodimos, E., Flach, P. Hierarchical Bayesian Networks: An Approach to Classification and Learning for Structured Data. *Proceedings of Methods and Applications of Artificial Intelligence, Third Hellenic Conference in AI*. 291-300, 2004.

[10] Heckerman, D., Geiger, D., Chickering, D.M. Learning Bayesian Networks: the Combination of Knowledge and Statistical Data. *Machine Learning*, 20, 197-243, 1995.

[11] Kiehl, K. An event-related functional magnetic resonance imaging study of an auditory oddball task in schizophrenia. *Schizophrenia Research*, 48:159-171, 2001.

[12] Lancaster J.L., Woldorff M.G., Parsons L.M., Liotti M., Freitas C.S., Rainey L., Kochunov PV, Nickerson D., Mikiten S.A., Fox P.T. Automated Talairach Atlas labels for functional brain mapping. *Human Brain Mapping* 10,120-131, 2000.

[13] Lam, W., Bacchus, F. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269-293, 1994.

[14] Margaritis, D., and Thrun S. Bayesian Network Induction via Local Neighborhoods. *Advances in Neural Information Processing Systems 12*, Denver, CO, 1999.

[15] McCallum, A., Rosenfeld, R., Mitchell, T. and Ng, A. Y. Improving Text Classification by Shrinkage in a Hierarchy of Classes. *ICML*, 1998.

[16] MIND Institute, The. http://www.themindinstitute.org/ , 2006.

[17] Pearl, J. Fusion, Propagation and Structuring in Belief Networks. *AI*, 29, 3, 241-288, 1986.

[18] Schwarz, G. Estimating the dimension of a model. *Annals of Stats.*, 6, 461-464, 1978.

[19] Sutton, R. S., Precup, D., and Singh, S. Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *AI*, v. 112. pg. 181-211, 1999.

Sequence Discrimination Using Phase-Type Distributions

Jérôme Callut[1,2] and Pierre Dupont[1,2]

[1] Department of Computing Science and Engineering, INGI
Université catholique de Louvain,
Place Sainte-Barbe 2,
B-1348 Louvain-la-Neuve, Belgium
{Jerome.Callut, Pierre.Dupont}@uclouvain.be
[2] UCL Machine Learning Group
http://www.ucl.ac.be/mlg/

Abstract. We propose in this paper a novel approach to the classification of discrete sequences. This approach builds a model fitting some dynamical features deduced from the learning sample. These features are discrete *phase-type* (PH) distributions. They model the first passage times (FPT) between occurrences of pairs of substrings. The PHit algorithm, an adapted version of the Expectation-Maximization algorithm, is proposed to estimate PH distributions. The most informative pairs of substrings are selected according to the Jensen-Shannon divergence between their class conditional empirical FPT distributions. The selected features are then used in two classification schemes: a maximum a posteriori (MAP) classifier and support vector machines (SVM) with marginalized kernels. Experiments on DNA splicing region detection and on protein sublocalization illustrate that the proposed techniques offer competitive results with smoothed Markov chains or SVM with a spectrum string kernel.

Keywords: Supervised sequence classification, Markov chains, First passage times, Expectation-Maximization, Jensen-Shannon divergence.

1 Introduction

This paper is concerned with a supervised classification problem in which the instances are sequences defined over a discrete alphabet. Practical applications of this task range from the recognition of boundaries between introns and exons in DNA sequences to musical pieces classification.

The approach proposed in this paper relies on building a model to fit some dynamical features in the sample. In this context, a former technique was based on the *mean first passage times* between individual symbols [3]. In this paper, we focus not only on the mean but on the complete distribution of the times between occurrences of substrings in the sample. More precisely, given a pair of substrings (v, w) called here a *feature* of the sequence to be classified, we are looking at the number of steps taken to observe the next occurrence of w after having observed v. The distribution of these measures forms the First Passage Time (FPT) dynamics of a sequential process with respect to the feature (v, w). The purpose of this paper is to exploit the different FPT dynamics between the classes to perform sequence classification. Since the number of features can be potentially large, only a restricted number of the observed features are considered. The selection of the features (v, w) is performed using the Jensen-Shannon (JS) divergence [8]. Given an observed feature (v, w), the empirical FPT distribution is estimated for each class. The JS divergence is then applied to rank the considered features. The features offering the largest JS divergence between their class conditional distributions are kept for the classification process. Once the features have been selected, the associated FPT dynamics are modeled with discrete phase-type distributions.

Discrete phase-type distributions (PH) form a broad class of distributions that generalize the family of negative binomial distributions and have applications in various stochastic models such as queuing systems [7]. A PH distribution can be defined as the distribution of the time to absorption in an absorbing Markov chain (MC). Our first contribution is an EM algorithm for fitting discrete PH distributions. Our algorithm can be considered as an adaptation to discrete distributions of the work of Asmussen and Olsson [1], which handles continuous PH distributions. Modeling the FPT dynamics with PH distributions allows one to control the generalization, that is, the probability mass given to unseen events, by tuning the number of phases (see section 2). In this sense, the PH modeling can be thought of as a smoothing technique of the empirical FPT distributions observed in the sequences. The estimated PH distributions are used to solve the sequence classification problem. Two classification schemes are considered: a maximum a posteriori (MAP) classifier and support vector machines (SVM).

The rest of this paper is organized as follows. Section 2 reviews standard Markov chains and PH distributions. Section 3 introduces the PHit algorithm for fitting discrete PH distributions. Section 4 describes the feature selection procedure and presents a MAP classifier as well as a kernel based on PH distributions. Finally, section 5 shows experimental results obtained with the proposed techniques applied to DNA splicing junction detection and protein sublocalization.

2 Discrete PH Distributions

The methods proposed to solve the sequence classification problem in section 4 rely on the first passage times (FPT) between events in sequences (see definition 2). These times can be conveniently modeled by PH distributions introduced hereafter. For a detailed introduction to the MC theory and to PH distributions, the reader is respectively referred to the classical text books [6] and [7]. A MC can be represented by a 3-tuple $M = \langle Q, A, \boldsymbol{\iota} \rangle$ where Q is a finite set of states of size m, A is an $m \times m$ stochastic transition matrix and $\boldsymbol{\iota}$ is a $1 \times m$ vector representing the initial probability distribution. In a MC, a state q is said to be *absorbing* if there is a probability 1 to go from q to itself. In other words, once an absorbing state has been reached, the process will stay on this state forever. A MC for which there is a probability 1 to end up in an absorbing state is called an *absorbing MC*. In such a model, the state set can be divided into the absorbing state set Q_A and its complementary set, the transient state set Q_T. An absorbing MC with a single absorbing state will be called a *reduced* absorbing MC (or a *reduced* MC for short). A fundamental characteristic of absorbing MC is the *time to absorption*, *i.e.* the number of steps the process takes to reach an absorbing state. The distribution of the time to absorption is of *phase-type*.

Definition 1 (Discrete Phase-type Distribution). *A probability distribution $\varphi(.)$ on $\mathbb{N}^0$ is a* distribution of phase-type *if and only if it is the distribution of the time to absorption in an absorbing MC.*

It should be pointed out that it is always possible to transform an absorbing MC with several absorbing states into a reduced one with the same distribution of the time to absorption. Hence, without loss of generality, we will only consider reduced MC in normal form, the last state being absorbing :

$$\boldsymbol{\iota} = (\boldsymbol{u} \quad 0), \quad A = \begin{pmatrix} T & \boldsymbol{e} \\ \boldsymbol{0} & 1 \end{pmatrix}$$

where $\boldsymbol{u}$ is a $1 \times (m-1)$ *initial vector* for transient states, T is an $(m-1) \times (m-1)$ matrix called the *phase generator*, $\boldsymbol{e}$ is an $(m-1) \times 1$ vector called the *absorption vector* and $\boldsymbol{0}$ is an $1 \times (m-1)$ vector of zeros. It will be assumed that the process always starts in a non-absorbing state: $\iota_m = 0$. A PH distribution φ is completely determined[1] by a pair $(\boldsymbol{u}, T)$ which is called the *representation* of the distribution. The probability distribution of $\varphi(.)$ is given by $\varphi(k) = \boldsymbol{u}T^{k-1}\boldsymbol{e}$ for all $k \geq 1$ which means that the probability of being absorbed in k steps is the probability of starting in any transient state, then to move over transient states during $k-1$ steps, and finally to get absorbed. Each transient state of the representing MC is called a *phase*. This technique is powerful since it allows one to decompose complex distributions as a combination of phases. For instance, the class of PH distributions contains the negative binomial, the hyper-geometric and the discrete Coxian distributions, to name a few. These distributions can be instantiated using particular absorbing MC structures. This point is illustrated in figure 1. A distribution with an initial vector and a transition matrix with no particular structure is called here a *general PH distribution*.

The next section presents a tool for estimating a PH distribution from a data sample. In this context, tuning the number of phases allows one to deal with the *bias-variance* trade-off. Indeed, fitting a distribution using a few phases gives an important probability mass to unseen events, which are unseen first passage times between two substrings in our context, while the overfitting is likely to happen when using a large number of phases.

[1] Since the matrix A is stochastic, the vector $\boldsymbol{e}$ can be obtained from the matrix T.

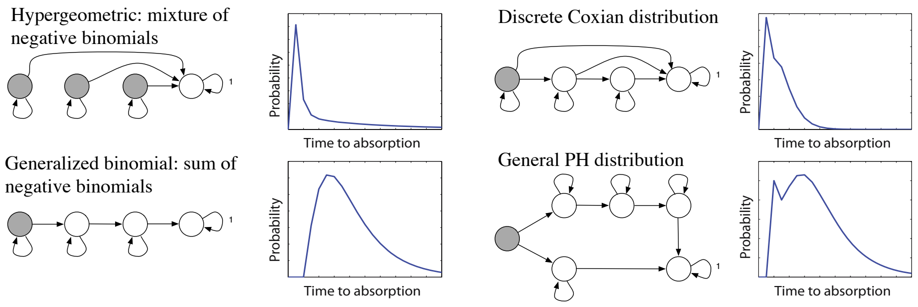


Fig. 1. Different kinds of PH distributions and associate absorbing MC structures. The process has a strictly positive probability to start in states filled with gray and a null probability to start in the other states.

3 Fitting Discrete PH Distributions: PHit

In this section, we introduce the PHit algorithm for fitting discrete PH distributions. This algorithm is used to compute the features in the classification methods presented in section 4. Here, the samples do not directly correspond to the learning sequences of the classification task but to discrete times between two particular substrings in these sequences.

The EMpht algorithm, developed by Asmussen and Olsson [1], fits continuous PH distributions. In contrast, we focus here on discrete PH distributions. In particular PHit deals with negative binomial state duration distribution in an absorbing MC while these durations are modeled in EMpht by a negative exponential density, typical of continuous Markov processes. The re-estimation formula in both algorithms are thus distinct. Bobbio *et al.* [2] also proposed a technique for fitting discrete PH distributions, however it is restricted to a particular class of PH distributions (acyclic PH distributions) while PHit can deal with general PH distributions.

Given a set of l observations (times between two events) $Z = \{z_1, \ldots, z_l\}$ with $z_i \in \mathbb{N}^0$ and a number m of states, PHit estimates the parameters $(\boldsymbol{u}, T)$ of a PH distribution φ with $m-1$ phases that maximize the likelihood with respect to Z. To do so, the iterative Expectation-Maximization (EM) algorithm is used [4]. The basic idea is to consider each z_i as an incomplete observation of the underlying process $\{X_t\}$ (an absorbing MC). More precisely, we observe the time to absorption of a process realization but not the sequence of states reached by the process during this realization. Let $H = \{h_1, \ldots, h_l\}$ be the set of hidden sequences relative to Z, where h_i is a sequence of z_i (transient) states. The likelihood of φ with respect to one complete observation (z, h) is given by $P[(z,h) \mid \varphi] = u_{h_1} \left(\prod_{i=1}^{z-1} T_{h_i, h_{i+1}} \right) e_{h_z}$. We introduce below three kinds of auxiliary variables which are useful in the estimation process:

- $S(i)$: the number of observations in H starting in state i.
- $F(i)$: the number of observations in H ending in state i.

– $N(i,j)$: the number of times state j immediately follows state i in H.

The likelihood of φ with respect to the set of all observations is defined as:

$$P((Z,H) \mid \varphi) = \prod_{i=1}^{m-1} u_i^{S(i)} e_i^{F(i)} \prod_{j=1}^{m-1} T_{ij}^{N(i,j)}$$

The EM algorithm finds the maximum likelihood estimates of the parameters λ of a joint distribution $P(Z,H|\lambda)$ when the variable H is unobserved (H is a *latent* or *hidden* variable). This is achieved by computing iteratively the parameters λ that maximize $E[\ln(P(Z,H \mid \lambda)) \mid Z]$. Each EM iteration involves two steps: (i) the computation of the conditional expectation of the latent variables given the last parameter values λ^{t-1} and the partial observations Z, (ii) the maximization of the parameters λ^t given the conditional expectation of the latent variables. The computations of these two steps in the PHit algorithm are detailed hereafter.

Expectation step
In PHit, the latent variables are the auxiliary variables $S(i)$, $F(i)$ and $N(i,j)$. Their conditional expectations are conveniently computed using *forward* variables defined as $\alpha_{ij}(t) = P[X_t = j \mid X_0 = i]$ for all $1 \le i < m, 1 \le j \le m$ and $t \ge 0$. That is, $\alpha_{ij}(t)$ is the probability of being in state j after t steps while starting from state i. The forward variables can be computed using the following recurrence:

$$\alpha_{ij}(0) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \alpha_{ij}(t) = \sum_{k=1}^{m-1} \alpha_{ik}(t-1).A_{k,j}$$

In the sequel, the shorthand $\alpha_j(t)$ stands for $\sum_{i=1}^{m-1} u_i \alpha_{ij}(t)$, which is the probability of being in state j after t steps when starting according to the initial distribution, $\alpha_m(t) = \varphi(t)$ being a special case. The forward variables $\alpha_j(t)$ can be thought of as a simplification of the forward variables used in the Baum-Welch algorithm which estimates the parameters of Hidden Markov Models (HMMs) [9]. Indeed, in the Baum-Welch algorithm, a forward variable computes the probability of being in a state after having accepted a given string. In PHit, a forward variable computes the probability of being in a given state after t steps no matter how this state is reached.

Conditional expectation of $S(i)$: The variables $S(i)$ can be decomposed as a sum of indicator variables[2], $S(i) = \sum_{k=1}^{l} \mathbb{1}\{h_{k,1} = i\}$ where the notation $h_{k,j}$ stands for the j-th state in the k-th observation. The conditional expectation $E[S(i) \mid z_1, \ldots, z_l]$ is given by

$$\begin{aligned} E[S(i) \mid z_1, \ldots, z_l] &= E[\sum_{k=1}^{l} \mathbb{1}\{h_{k,1} = i\} \mid z_1, \ldots, z_l] = \sum_{k=1}^{l} E[\mathbb{1}\{h_{k,1} = i\} \mid z_k] \\ &= \sum_{k=1}^{l} P[h_{k,1} = i \mid z_k] \end{aligned}$$

[2] The value of an indicator variable $\mathbb{1}\{X = x\}$ is 1 when the random variable X equals x and 0 otherwise. The expectation of an indicator variable is simply $E[\mathbb{1}\{X = x\}] = P[X = x]$.

The conditional probability $P[h_{k,1} = i \mid z_k] = \frac{P[h_{k,1}=i,z_k]}{P[z_k]}$ with $P[h_{k,1} = i, z_k] = u_i \alpha_{im}(z_k)$ and $P[z_k] = \alpha_m(z_k)$. Finally, the conditional expectation is defined as

$$E[S(i) \mid z_1, \ldots, z_l] = \sum_{k=1}^{l} \frac{u_i \alpha_{im}(z_k)}{\alpha_m(z_k)}$$

Conditional expectation of $N(i,j)$: The variables $N(i,j)$ can be decomposed as $\sum_{k=1}^{l} \sum_{d=1}^{z_k-1} \mathbb{1}\{h_{k,d} = i \wedge h_{k,d+1} = j\}$, that is, counting the presence of the transition $i \rightarrow j$ in each position of each hidden sequence. By the same reasoning as above, one obtains

$$E[N(i,j) \mid z_1, \ldots, z_l] = \sum_{k=1}^{l} \sum_{d=1}^{z_k-1} P[h_{k,d} = i \wedge h_{k,d+1} = j \mid z_k]$$

The joint probability $P[h_{k,d} = i \wedge h_{k,d+1} = j, z_k] = \alpha_i(d-1) T_{ij} \alpha_{jm}(z_k - d)$, that is the probability of starting in any transient state, to reach state i after $d-1$ steps, then to perform the transition $i \rightarrow j$, and finally to get absorbed $z_k - d$ steps later. It follows that

$$E[N(i,j) \mid z_1, \ldots, z_l] = \sum_{k=1}^{l} \sum_{d=1}^{z_k-1} \frac{\alpha_i(d-1) T_{ij} \alpha_{jm}(z_k - d)}{\alpha_m(z_k)}$$

Conditional expectation of $F(i)$: The variables $F(i)$ can be decomposed as $F(i) = \sum_{k=1}^{l} \mathbb{1}\{h_{k,z_k} = i\}$. By the same reasoning as above, the conditional expectation is $E[F(i) \mid z_1, \ldots, z_l] = \sum_{k=1}^{l} P[h_{k,z_k} = i \mid z_k]$. The joint probability $P[h_{k,z_k} = i, z_k] = \alpha_i(z_k - 1) e_i$, that is the probability of starting in any transient state, to reach state i after $z_k - 1$ steps then to get absorbed. The conditional expectation is given by

$$E[F(i) \mid z_1, \ldots, z_l] = \sum_{k=1}^{l} \frac{\alpha_i(z_k - 1) e_i}{\alpha_m(z_k)}$$

Maximization step

Once the expected values of the latent variables have been computed, the maximum likelihood estimators of the PH distribution parameters are

$$u_i = \frac{\overline{S(i)}}{\sum_{k=1}^{m-1} \overline{S(k)}}, e_i = \frac{\overline{F(i)}}{\sum_{k=1}^{m-1} \overline{F(k)}} \text{ and } T_{ij} = \frac{\overline{N(i,j)}}{\sum_{k=1}^{m-1} \overline{N(i,k)}} \quad \forall i,j, 1 \leq i,j < m$$

where the bar notation stands for the conditional expectation of the variables.

The parameters $(\boldsymbol{u}, T)$ are initialized at random at the beginning of the algorithm. PHit iterates until the relative likelihood improvement falls below a user-defined threshold. In our experiments, PHit is rather insensitive to the initialization. In other words, the value of the likelihood obtained after convergence is rarely affected by different initializations. However, the parameters found by PHit can be different using different initializations since the representation of a PH distribution is not unique. As in the Baum-Welch algorithm, the forward variables can efficiently be computed by use of a lattice data structure. More precisely, for each transient state i a lattice is built to compute the forward variables $\alpha_{ij}(t)$ with $1 \leq j \leq m$. The time complexity of building such a lattice is $\mathcal{O}(L.m^2)$ where L is the longest absorption time in Z. The time complexity is $\mathcal{O}(L.m^3 + L.l.m^2)$ for the expectation step and $\mathcal{O}(m^2)$ for the maximization step. PHit has been implemented in the ANSI C language and it can estimate PH distributions with 100 phases over 2,000 events in about one minute on a standard PC.

4 Classification Using PH Distributions

In section 2, we argue that the time dynamics between events in a sequential process are conveniently modeled by PH distributions. In practice, these distributions can be estimated from a data sample by use of the PHit algorithm presented in section 3. In the present section we describe how PH distributions are used to solve a supervised classification problem in which the instances are sequences defined over a discrete alphabet Σ and the labels belong to a set $\mathcal{Y}$. Given a set of n examples $\{(s_1, y_1), \ldots, (s_n, y_n)\}$ where $s_i \in \Sigma^*$ is a sequence and $y_i \in \mathcal{Y}$ its label (or its class), one wants to estimate a function $f : \Sigma^* \rightarrow \mathcal{Y}$ for predicting the label of new sequences. Our methods run into 3 steps: (i) features selection, (ii) choosing the number of phases and modeling the selected features with PH distributions, (iii) classifier training. The features extracted from the learning sequences are the first passage times between pairs of events (occurrences of substrings).

Definition 2 (First Passage Times between a pair of events in a sequence). *Given a sequence s defined on an alphabet Σ and two substrings $v, w \in \Sigma^+$. For each occurrence of v in s, the first passage time to w is defined as the finite number of steps taken before observing the next occurrence of w. The* first passage times *from v to w in s is a multiset defined as the first passage times to w for all occurrences of v in s.*

For instance, let us consider the alphabet $\Sigma = \{a, b\}$, the sequence $s = aababba$ and the events $v = ab$ and $w = ba$. The value of the feature (ab, ba) is $\phi_{(ab,ba)}(s) = \{3, 1\}$. Let us note that the step count starts after the last character of v and it does not take the length of w into account. The choice of the features (pairs of events) is an important step in the classification process. The potential number of features is bounded by $(\sum_{i=1}^{N} |\Sigma|^i)^2 \in \mathcal{O}(|\Sigma|^{2N})$, where $|\Sigma|$ denotes the alphabet size. However the number of features observed in practice is often below this bound. It can be shown that an alternative upper bound is $n.L^2.K^2$, where L is the length of the longest sequence in the training set containing n examples and K is the maximal length of the considered substrings (in our experiments, $K = 3$).

Since our classifications methods rely on the difference between the class conditional FPT dynamics, the proposed features selection procedure extracts the features for which the empirical FPT distribution differs the most among the classes. To do so, the generalized Jensen-Shannon (JS) divergence is used [8]. A larger JS divergence between the empirical distributions of various samples indicates that they are more likely to have been drawn from different source distributions (i.e. from different classes). This measure is defined as follows.

$$JS(P_1, \ldots, P_r) = H\left(\sum_{i=1}^{r} \pi_i H(P_i)\right) - \sum_{i=1}^{r} \pi_i H(P_i)$$

where $P_1, \ldots, P_r$ are distributions, $\pi_1, \ldots, \pi_r$ are strictly positive weights summing up to one and $H(P) = -\sum_{x \in \Omega} P[x] \log P[x]$ is the Shannon entropy. The JS divergence has several advantages over the Kullback-Leibler (KL) divergence [8], a classical measure to compare two distributions: (i) it can be applied to more than two distributions, (ii) the relative importance of the distributions can be parametrized with weights (in our experiments, uniform weights are used), (iii) it can be thought as a symmetrized and smoothed (it is relative to the mean of the distributions) variant of the KL divergence, (iv) when applied to two distributions, the square root of the JS divergence enjoys the properties of a true distance metric. For each class, the empirical FPT distributions between every observed event pairs are computed. The score of a feature is defined as the JS divergence between the empirical FPT distributions of each class, weighted by the prior probability of the feature. The prior probability of a feature (v, w) is defined as $P[(v, w)] = \frac{\mathrm{C}(v,w)}{\sum_{v',w' \in \Sigma^{\leq K}} \mathrm{C}(v',w')}$, where $\mathrm{C}(v, w)$ is the number of times a string v is followed by a string w in the training set and $\Sigma^{\leq K}$ is the set of non-empty strings up to length K defined on the alphabet Σ. The features are ranked with respect to their score and the highest ranked features are kept for the classification process. In the sequel, the set of selected features will be denoted by $\mathcal{F}$. In practice, a set of 10^5 features can be ranked in about 30 seconds on a standard PC.

4.1 Maximum a Posteriori (MAP) Classifier

In this section, we introduce a maximum a posteriori (MAP) classifier based on PH distributions. Once the features have been selected, the related FPT dynamics are modeled with PH distributions for each class. The notation $\varphi^{y}_{(v,w)}(.)$ stands for the PH distribution relative to (v, w) estimated from the sequences of the class y. Our classifier makes the assumption that the features are independent. As usual for models making this naive assumption, the independence is not always satisfied but good results are obtained in practice. Consequently, the likelihood of a class y with respect to a sequence s is computed as $P[s \,|\, y] = \prod_{(v,w) \in \mathcal{F}} P[\phi_{(v,w)}(s) \,|\, y]$, where $P[\phi_{(v,w)}(s) \,|\, y] = \prod_{z \in \phi_{(v,w)}(s)} \varphi^{y}_{(v,w)}(z)$. Predicting the label of a sequence s is made by selecting the class that maximizes the posterior probability $\hat{y} = \operatorname{argmax}_y P[s \,|\, y]P[y]$, where $P[y]$ denotes the prior probability of the class y.

4.2 SVM in the PH Feature Space

We introduce here the PH kernel which maps the sequences in a feature space based on PH distributions. For each feature (v, w), a *marginalization kernel* [10] $k_{(v,w)}(.,.)$ computing the probability that two sequences have been generated together is introduced as follows.

$$k_{(v,w)}(s, s') = P_{(v,w)}[s, s'] = \sum_{y \in \mathcal{Y}} P[\phi_{(v,w)}(s) \mid y] P[\phi_{(v,w)}(s') \mid y] P[y]$$

The PH kernel, relative to the complete feature set $\mathcal{F}$, is defined as

$$k(s,s') = \sum_{(v,w)\in\mathcal{F}} P_{(v,w)}[s,s'] = \sum_{(v,w)\in\mathcal{F}} k_{(v,w)}(s,s')$$

The PH kernel amounts to compute a dot product in the space where a sequence s is mapped to $\left(\sqrt{P[y]}.P[\phi_{(v,w)}(s) \mid y]\right)_{y\in\mathcal{Y},(v,w)\in\mathcal{F}}$. The PH distributions used to compute the probabilities $P[\phi_{(v,w)}(s)|y]$ are estimated from a part of the training data (80 % in our experiments) and the rest of the data is used to train the SVM. The rationale is that the training data are no longer independent if they are used to build the kernel mapping. Once the SVM has been trained, new sequences are classified by looking at which side of the hyperplane they lie in the PH feature space.

5 Experiments

This section presents the experimental results obtained for two classification tasks: (i) DNA splicing region detection (`Splice` dataset) and (ii) protein sublocalization (`DBSubloc` database). The `Splice` dataset[3] is made of windows of 60 symbols from DNA sequences containing intro-exon (IE) or exon-intron (EI) boundaries or neither of them. We restrict here our attention to binary classification by considering sequences labeled either EI or IE. The class priors are equal and the training, validation and test sets contain respectively 975, 253 and 253 sequences. The `DBSubloc` database[4] contains protein sequences (primary structures) with their subcellular localization for various organisms. The classification tasks considered here consists in finding if a protein from a plant organism is located in the membrane or in the mitochondria of the cell. The average length of the sequences is 406. The class priors are respectively 0.45 and 0.55 for the membrane and the mitochondria classes and the training, validation and test sets contain respectively 151, 51 and 50 sequences.

The influence of the parameters (the number of phases and the number of features) is evaluated with both classification methods using the `Splice` validation data. Figure 2 presents learning curves using training data of growing sizes. For each size (except for 100%), 10 samples have been randomly extracted from the training set in order to produce averaged results with standard deviations. The left side of Figure 2 shows the accuracy obtained on validation data using the MAP classifier with increasing number of phases and 100 features. Interestingly, one can observe that for very small training set sizes (2% and 5%), using 2 phases leads to significantly better results as for 2% of the training data, the classification accuracy is about 75% while it is round 56% when using 5 or 10 phases. The reason is that the estimation of PH distributions with a larger number of parameters[5] becomes unreliable when there are too few observations. When the

[3] `Splice` is available from the UCI repository.

[4] `DBSubloc` is available at `http://www.bioinfo.tsinghua.edu.cn/dbsubloc.html`.

[5] A PH distribution with p phases has $p^2 + p$ parameters.

training set size becomes greater than 10%, the benefit of additional phases is noticed. The experiments were made using up to 20 phases but it appears that using more than 5 phases does not significantly improve the accuracy for this problem. The right side of Figure 2 shows the accuracy obtained on validation

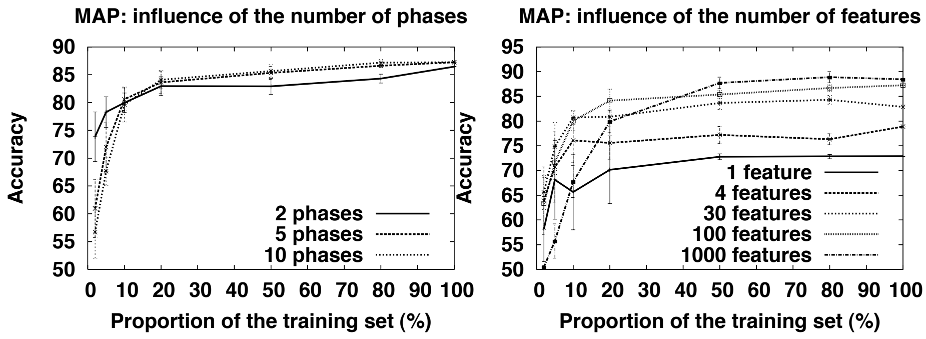


Fig. 2. Influence of the number of phases and of the number of features on the MAP classifier using validation data. Left: influence of the number of phases using 100 features. Right: influence of the number of features using 5 phases.

data using the MAP classifier with an increasing number of features (1, 4, 30, 100 and 1000) and 5 phases. The features were selected according to their JS ranking (see section 4). It can be observed that, in general, the classification accuracy increases with the number of selected features. An accuracy around 73 % is already obtained using a single feature (the highest ranked feature is the substring pair (T, GG)). Using 1000 features only improves the results when 50% of the training data are used as for smaller training set sizes, the lack of observations (for rare features) decreases the accuracy of the fitting. The best accuracy on validation data, 90%, is obtained using the complete training set, 1000 features and 10 phases.

Practical evaluations of SVM with the PH kernel illustrate robustness to overfitting of PH estimations. Indeed, the number of phases has no influence on the results and the method performs well even with small training set sizes and 1000 features. The SVM classifier thus seems able to compensate unreliable estimations by adapting its decision function.

Figure 3 shows comparative results on both dataset using several classifiers: (i) smoothed N-grams [5], (ii) MAP, (iii) SVM with the PH kernel and (iv) SVM with the *blended*[6] *spectrum* string kernel [10] (p. 350). While the PH kernel relies on passage times between substrings, the blended spectrum string kernel is based on the frequencies of all common substrings up to a fixed length. The SVM regularization constant C was tuned according to the heuristic[7] of SVM^{light}

[6] Experiments with the standard p-spectrum kernel [10] (p. 347) offer worse results than the blended version reported here.

[7] $C = \frac{1}{\frac{1}{n}\sum_{i=1}^{n}\sqrt{k(s_i,s_i)}}$ where n is the number of training sequences.

and the kernel parameters were selected using the validation set. The left side of Figure 3 presents results obtained on the `Splice dataset`. The best parameters obtained on validation data are a 4-gram, a blended spectrum string kernel length of 7 with a length weight of 3.5, and a 10 phases PH modeling of the 1000 best features for our methods. When a sufficient amount of data is used (at least 50% of the training data), the best performances are obtained with the MAP classifier. Both kernels have comparable performance on this task. The test classification accuracy for the 4-grams, the PH kernel, the blended spectrum string kernel, and the MAP classifiers are respectively 84.1%, 88.5%, 88.8% and 89.7% when the whole training set is used. The right side of Figure 3 presents results obtained

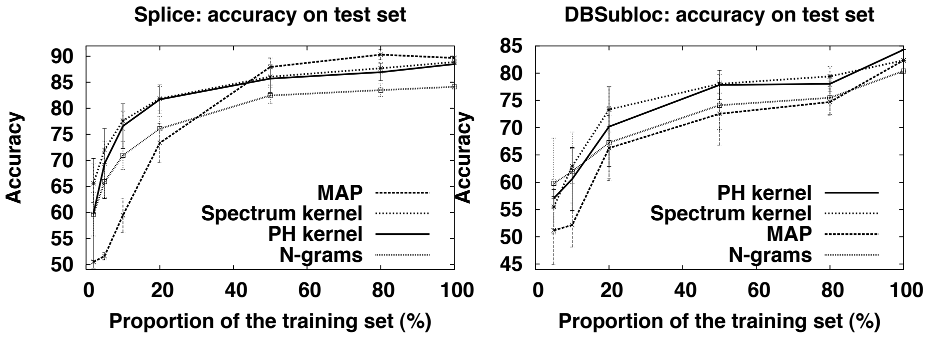


Fig. 3. Comparative classification results for N-grams, MAP and SVM with the PH kernel and a spectrum string kernel on `Splice` (left) and `DBSubloc` (right) test data.

on the `DBSubloc` database. The best parameters obtained on validation data are a 2-gram, a blended spectrum string kernel length of 4 with a length weight of 2.5 and a 5 phases PH modeling of the 100 best features for our methods. The test classification accuracy for 2-grams, the blended spectrum string kernel, the MAP classifier and the PH kernel are respectively 80.4%, 82.35%, 82.4% and 84.3% when the whole training set is used.

6 Conclusion

We propose in this paper a novel approach to the classification of discrete sequences. This approach builds a model fitting some dynamical features deduced from the learning sample. More precisely, the distribution of the times between occurrences of substrings observed in the sample are modeled with discrete phase-type (PH) distributions. Phase-type distributions are defined as the distribution of the time to absorption in finite absorbing Markov chains. This kind of modeling is powerful as it allows one to decompose complex distributions as a combination of phases. The PHit algorithm, an adapted version of the Expectation-Maximization algorithm, is proposed to estimate PH distributions. In this context, tuning the number of phases allows one to deal with the *bias-variance* trade-off. The most informative features (pairs of substrings) are selected according to the Jensen-Shannon divergence between their class conditional empirical FPT distributions. The selected features are used in two classification schemes: a maximum a posteriori (MAP) classifier and support vector machines (SVM) with marginalized kernels. Experiments on DNA splicing region detection and on protein sublocalization illustrate that the proposed techniques offer better results than smoothed Markov chains and competitive results with SVM and a blended spectrum string kernel.

Our future work includes the evaluation of the proposed methods when noisy training data are considered. The HMM learning technique proposed in [3] could also be extended in order to fit first passage time distributions between every pair of symbols, rather than simply the expectations of these times.

Acknowledgment

The authors wish to thank Philippe Delsarte and Guy Latouche for many fruitful discussions about this work. This work is partially supported by the *Fonds pour la formation à la Recherche dans l'Industrie et dans l'Agriculture (F.R.I.A.)* under grant reference F3/5/5-MCF/FC-19271.

References

1. Søren Asmussen, Olle Nerman, and Marita Olsson. Fitting phase-type distributions via the em algorithm. *Scandinavian Journal of Statistics*, 23(4):419–441, 1996.
2. A. Bobbio, A. Horváth, M. Scarpa, and M. Telek. Acyclic discrete phase type distributions: properties and a parameter estimation algorithm. *Perform. Eval.*, 54(1):1–32, 2003.
3. Jérôme Callut and Pierre Dupont. Inducing hidden markov models to model long-term dependencies. In *16th European Conference on Machine Learning (ECML)*, number 3720 in Lecture Notes in Artificial Intelligence, pages 513–521, Porto, Portugal, 2005. Springer Verlag.
4. A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society Ser. B (methodological)*, 39:1–38, 1977.
5. P. Dupont. Noisy sequence classification with smoothed markov chains. In *Conférence francophone sur l'apprentissage automatique 2006, (CAp 2006)*, pages 187–201, Trégastel, France, 2006.
6. J.G. Kemeny and J.L. Snell. *Finite Markov Chains.* Springer-Verlag, 1983.
7. G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling.* Society for Industrial & Applied Mathematics,U.S., 1999.
8. J. Lin. Divergence measures based on the shannon entropy. *IEEE Trans. Information Theory*, 37:145–151, 1991.
9. L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
10. John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, June 2004.

Languages as Hyperplanes: Grammatical Inference with String Kernels

Alexander Clark, Christophe Costa Florêncio, and Chris Watkins

Department of Computer Science, Royal Holloway, University of London, Egham TW20 0EX

Abstract. Using string kernels, languages can be represented as hyperplanes in a high dimensional feature space. We present a new family of grammatical inference algorithms based on this idea. We demonstrate that some mildly context sensitive languages can be represented in this way and it is possible to efficiently learn these using kernel PCA. We present some experiments demonstrating the effectiveness of this approach on some standard examples of context sensitive languages using small synthetic data sets.

1 Introduction

Much data consists of strings of symbols. A set of symbol strings is known as a *language*: a natural machine learning problem is to infer a definition of a language from a set of positive examples of strings in the language. This problem has been much studied as *grammatical inference*, and this is the term we will use. This type of problem occurs in many fields. In data mining, for example, there may be a need to learn to recognise strings in particular flexible formats. Simple cases are email addresses or page formatting commands; more complex cases might be price-lists, postal addresses, or stock upgrades in free-text broker reports. Sequence annotation in bioinformatics is another application.

Grammatical inference algorithms for regular languages are now well understood, either using state merging algorithms for deterministic finite state automata, or using Hidden Markov Models (HMMs), the non-deterministic equivalent, with the EM algorithm. For context free languages, some recent approaches have had some limited success (Starkie et al., 2004). However, it is well known that certain features of natural language cannot be described by context free grammars, and require the power of mildly context sensitive grammars, which cannot be learned with current techniques.

In this paper we consider a new approach to grammatical inference based on the identification of hyperplanes in high-dimensional feature spaces induced by string kernels. This is entirely different from techniques previously studied in computational linguistics. We present and discuss computational experiments on a range of synthetic examples of languages chosen to be at different levels of the Chomsky hierarchy. We compare the performance of the new techniques with HMMs.

Our experiments may appear non-standard to machine learning researchers. We learn from positive examples only, for two reasons: first, this is a standard approach in language-learning theory; and second, because informative negative examples — "near misses" — may be difficult to generate, and they are rare in practice in real data. We do generate negative examples as part of the test data.

A second way in which we depart from the conventions of kernel learning experiments is that we seek language descriptions as hyperplanes in the feature space, rather than as the more conventional half-spaces or clusters. Equality constraints are easy to interpret in simple cases, and can represent many of the particular languages we are interested in, though in other cases inequality constraints are necessary.

Thirdly, we use synthetic data generated from languages of known structure, rather than naturally occurring data. Our experiments are designed to explore the sorts of languages that this technique can learn rather than to demonstrate the utility of these methods on practical problems; an issue that we will address in future work. Already, these results are highly relevant to some issues in language learnability (Gentner et al., 2006).

We seek learnable representations that are sufficiently expressive to represent these mildly context sensitive languages. A key example, which we shall return to is from Swiss German (Shieber, 1985), though similar phenomena occur in Dutch. Abstracting away from some details, Swiss German has some subordinate clauses where a sequence of nounphrases can be followed by a sequence of verbs, of the same length, but where there are agreement constraints between the nouns and the verbs. Particular verbs require their corresponding nouns to be marked as a particular case, accusative or dative. Thus if we represent verbs by V_{acc}, V_{dat} where the subscript indicates the required case of the argument, and the verbs by N_{acc}, N_{dat}, the grammatical sentences are of the form $N_{acc}N_{dat}N_{dat}V_{acc}V_{dat}V_{dat}$: the sequence of nouns must agree with the sequence of verbs in the same order. No other orders are allowed, and there is no strict upper bound on the length of this construction. It is easy to see that this is not a context free language through the application of a pumping lemma. It currently appears that all currently observed non context free phenomena in language[1] lie within the class of mildly context sensitive languages, a class of languages defined by a number of weakly equivalent formalisms such as linear indexed grammars, tree adjoining grammars etc. These languages also include other non CF languages such as $\{a^n b^n c^n \mid n > 0\}$.

Modelling the acquisition of natural languages by children, or acquiring representations of natural language for NLP tasks will eventually require representations that can represent these structures, together with learning algorithms capable of acquiring them from observable data.

Formally we situate our work in the context of classical grammatical inference from positive data: given an unknown language, and a finite sample of strings drawn from that language, and *without* any negative data, i.e. strings not in the

[1] We note a few exceptions whose status is questionable such as Old Georgian, and a fraction of the Chinese number system.

language and marked as such, we wish to have an algorithm that can acquire a representation of the language that will enable us to determine whether a new string is in the language or not. The desiderata for such an algorithm include: reasonable observed sample complexity under natural distributions, polynomial computational complexity, robustness to small amounts of noise in the strings, and convergence over a sufficiently large class of languages.

1.1 Techniques

The familiar representations of languages are rewriting systems and automata of various types. These two families of representations converge at various points to give the well known Chomsky hierarchy. Unfortunately even low levels of the hierarchy are sufficiently powerful to represent cryptographically hard problems when considered as learning problems (Kearns & Valiant, 1989).

A completely different approach is to represent languages through linear constraints on the substrings (Salomaa, 2005). As a trivial example, consider the language over the alphabet $\{a, b\}$ consisting of equal numbers of as and bs in any order: example strings are *bbaa*, *ab*, *ababba* etc. This is a context free language, and can be defined either as a pushdown automaton, or a surprisingly complicated context free grammar. But we can clearly directly represent this as the set of all strings that satisfy a certain linear equation on the occurrences of the symbols a and b, $L = \{u \in \{a, b\}^* \mid |u|_a = |u|_b\}$ where we write $|u|_a$ for the number of times a occurs in u.

Looked at in this representation, a grammatical inference algorithm instantly suggests itself: map the strings into a certain vector space, and look for a low dimensional subspace that the data lie in. In this case the Parikh map (Parikh, 1966) is sufficient. Other languages will require the use of counts of substrings of length greater than one, and in this case we can use the implicit feature map defined by a string kernel, and where appropriate, work in the dual representation. Our technique combines two well understood techniques: kernel PCA (Schölkopf et al., 1998) together with string kernels (Watkins, 2000; Lodhi et al., 2002).

1.2 Preliminaries

An alphabet Σ is a non-empty finite set of symbols, often called letters. The set of all strings over Σ, written Σ^* is defined as the free monoid over Σ with null, the empty string, written as ϵ. A language L is a subset of Σ^*.

If u is of length n we can refer to the individual symbols as $u = u_1 \dots u_n$. If $u, v \in \Sigma^*$, u is a subsequence of v if there are indices $\mathbf{i} = (i_1, \dots i_{|\mathbf{i}|})$ with $1 \leq i_1 < \cdots < i_{|u|} \leq |v|$, such that $u_j = v_{i_j}$ for $j = 1 \dots |u|$. We write $u^R = u_n \dots u_1$ for the reversal or mirror image of u.

For $u, v \in \Sigma^*$ we will write $|u|_v$ for the number of times that v occurs in u as a non-contiguous substring. For example, if $\Sigma = \{a, b, c\}$, and $u = caab$, then $|u|_{ab} = 2, |u|_{cb} = 1$.

The choice of kernel defines the mapping to the feature space. We used a number of different kernels in our experiments. We will use the terminology and notation of (Shawe-Taylor & Christianini, 2004). Here $\mathbf{i}$ refers to a strictly ordered list of indices.

Fixed length subsequences kernel. All non contiguous subsequences of length k. The features are restricted to $|u| = k$.

Parikh kernel. This is the special case of the fixed length subsequences kernel with $k = 1$. The feature space thus has dimension $|\Sigma|$.

Gap weighted subsequences kernel. All non contiguous subsequences of length k where the gaps are weighted by λ. $\phi_u(s) = \sum_{\mathbf{i}:u=s(\mathbf{i})} \lambda^{l(\mathbf{i})}$ where $|u| = k$. $l(i)$ is defined as $1 + i_{|i|} - i_1$.

In the experiments reported here, we use $k = 2$ or $k = 1$.

2 Representational Power

Before discussing the learnability of this class, we can look at the representational power of the formalism. For any given string kernel κ we can define the class of languages which are the pre-images of finite dimensional hyperplanes in the induced feature space. We call these the κ-planar languages. A language is κ-planar, if there exist strings $w_1, \ldots, w_n$ such that

$$L = \{w \in \Sigma^* \mid \exists \alpha_1 \ldots \alpha_n, \sum_i \alpha_i = 1 : \phi(w) = \sum_i \alpha_i \phi(w_i)\}$$

Different kernels will enable different languages to be defined. An important distinction is between kernels where the implicit feature map is injective, and those where it is not. The k-subsequence kernel is not injective, for any k. When $k = 1$, the two strings ab and ba are equivalent, when $k = 2$, the two strings $abba$ and $baab$ are equivalent, and such examples can be generated for any k. In practice, for sufficiently large values of k, the proportion of strings that are mapped to the same point in feature space is small. Other kernels however are normally injective. The gap-weighted kernel weights features by polynomials in a parameter λ, corresponding to the numbers of gaps. Ignoring numerical issues, we can ensure that it is injective by setting the value of λ to be a suitable transcendental number, say $1/e$, which since it will not be the solution of any polynomial, means that the feature values will coincide only when the strings are identical.

We will start with a trivial language over the two letter alphabet $\Sigma = \{a, b\}$; $L_{ab} = \{u \in \Sigma^* \mid |u|_a = |u|_b\}$. This is an infinite language, which consists of all strings with equal numbers of as and bs. Example strings in the language are $ab, ba, aaaabbbabb, bbaa, \ldots$. It is easy to show that this is not a regular language, by an application of a pumping lemma, or to define a push-down automaton or context free grammar that generates this. But the way we have written it is explicitly as a linear relationship between two substring counts. If we consider the feature mapping defined by the Parikh kernel, which has exactly two dimensions, we can see that $\phi(L_{ab}) = \{(x, x) | x \geq 0\}$. Clearly these points lie in a hyperplane (a line in this case) in the feature space $\mathbb{R}^2$. Moreover, the preimage of the minimal hyperplane containing all the points of the language is exactly L_{ab}. More formally we can define for any language L and feature map $\phi : \Sigma^* \to H$, where H is some Hilbert space, the hyperplane defined by (all affine combinations of) L as

$$H(L) = \left\{ \sum_i \alpha_i \phi(u_i) \in H \mid \exists u_i \in L\,,\, \alpha_i \in \mathbb{R},\ \text{s.t.} \sum_i \alpha_i = 1 \right\}$$

and the language $\hat{L}$ as the preimage of this hyperplane

$$\hat{L} = \{ w \in \Sigma^* \mid \phi(w) \in H(L) \}$$

In this case it is easy to see that $\hat{L} = L$.

A slight modification of this approach would be to consider all linear combinations: i.e. removing the constraint that the coefficients sum to one. This would make the dimension of the subspace 1 higher, and for some kernels would also change the representational power. For the Parikh kernel, all such languages would have to include the empty string.

We are interested in learning from finite positive data sets. A natural algorithm suggests itself. Given a finite subset of L, say S, we can clearly define as our hypothesis $\hat{S}$: the preimage of all affine combinations of the sample points. In this particularly trivial case we can see that for all $S \subset L$ with at least two distinct elements, $\hat{S} = L$. If we take a slightly less trivial language

$$L_{a^n b^n} = \{ a^n b^n \mid n > 0 \} \tag{1}$$

it is easy to see that if we use the same Parikh kernel, $\hat{L}_{a^n b^n} = L_{ab}$. Since the Parikh kernel is not injective and indeed any two permutations of the same string are mapped to the same point, the representational power of this is very limited since it cannot represent any order constraints.

Consider the kernel κ_2, the subsequence kernel of length 2. If $w \in L_{a^n b^n}$, then clearly $|w|_{ba} = 0$. Thus using these features this language can be represented as $|s|_a = |s|_b$ and $|s|_{ba} = 0$, without any recursive structure or center-embedding (Gentner et al., 2006). Since κ_2 has such features we will now be able to represent languages like $L_{a^n b^n}$ as hyperplanes in this richer feature space. The use of features corresponding to substrings of length greater than 2 increases the expressive power. For example, while κ_2 can express ordering constraints, κ_3 can express that a certain string must appear *between* two other strings, and so on. This increased expressivity comes at a price; the dimensionality of the feature space is $\mathcal{O}(|\Sigma|^k)$ for κ_k, and thus the amount of data required to learn a simple language can increase radically.

The relationship between k-testable languages and planar languages defined by the k-spectrum kernel is a useful illustration of the power of our technique. k-testable languages are those that are defined by a set of admissible k-length substrings. Clearly any k-testable language defined by n strings $u_1, \ldots u_n$, $|u_i|$

$= k$, is also a planar language defined by the n-dimensional subspace spanned by these n substrings (we neglect here the problems of boundary symbols and prefixes and suffixes). But the class of planar languages contains not just the axis-aligned hyperplanes defined by each of these basis vectors, but also non-axis aligned hyperplanes.

It is worth noting that the class of planar languages does not have nice closure properties. It is closed under reversal and intersection, but not in general under union, concatenation, or other standard operations (Clark et al., 2006).

3 Algorithm

We have described the algorithm informally above in terms of the primal representation in the feature space. In practice, it is more convenient to perform the computations in a dual representation using only kernel operations. For some of the kernels, the number of dimensions in the feature space is less than the number of data points; nonetheless we work throughout with the kernel representation, for ease of use. The training phase of the algorithm follows a standard kernel PCA method (Shawe-Taylor & Christianini, 2004).

1. Inputs: a kernel, a set of training data, a set of test data.
2. Compute Gram matrix of the training data.
3. Compute translated Gram matrix, with center at origin in feature space.
4. Compute k, the rank of the translated Gram matrix.
5. Compute the k eigenvectors and eigenvalues.
6. Compute the translated matrix of training-test products.
7. Project the test strings onto the hyperplane defined by the training data.
8. Compute perpendicular distance from test strings to hyperplane.
9. If this distance exceeds a threshold, label the data as negative, otherwise label it as positive.

It is easy to establish that for a kernel that can be evaluated in polynomial time, and where the size of the representation is taken to be the rank of the plane, the class of all planar languages can be polynomially identified in the limit from positive data alone. Similarly, it can be proved that this class can be PAC-learned, with sample complexity polynomial in the rank of the language. A description of the theoretical aspects of this can be found in (Clark et al., 2006).

4 Experiments

This algorithm was implemented using MATLAB. On all of the experiments reported here, the running times were only a few minutes. We found that the threshold was easy to set: generally the squared residuals were either very close to zero, 10^{-10}, or greater than 0.1.

We generated some synthetic data sets to evaluate the potential of this approach. We selected a number of languages that have been proposed in the literature, generally from small alphabets. For comparison, we also evaluated it on one of the more complex grammars from the Omphalos context free grammatical inference competition (Starkie et al., 2004); this is at the state of the art for context free grammatical inference.

For each of the languages we generated some positive data, by sampling from a natural distribution. For example, for the copy languages, we first generated a random length, and then created a random string by sampling from the uniform distribution over all strings of that length. We then duplicated it to create the sample string. All of the positive data was generated IID. The lengths of the strings were generally less than 20, with a few exceptions: the strings from the Omphalos data set are much longer than this.

For evaluation we need both positive and negative data. Negative data is more of a problem. Simply generating random strings in a similar way does not produce a test set sufficiently difficult to distinguish the true hypothesis from a similar but incorrect one, without using astronomical amounts of data – the same problem was encountered by the organisers of the Omphalos competition (Starkie et al., 2004). We thus generated the negative data sets 50% from random strings from a uniform distribution over strings, and 50% drawn from languages that are close to the true one. Thus for example, when testing languages like **An** − **Dn**, we generated samples from $\{a^+b^+c^+d^+\}$ as well as from $\{a, b, c, d\}^+$.

For comparison, we also implemented two baseline systems, based on Hidden Markov Models, and Probabilistic Context Free Grammars. For the HMM system, we randomly initialised a HMM with a fully connected transition matrix, and with an explicit end of string transition for each state, and for the PCFG system we used a CNF grammar. In both cases they were trained to convergence with respectively the Baum-Welch algorithm and the inside-outside algorithm. We then evaluated the test strings and labelled them as positive or negative according to whether the probability of the string was above a simple length-based threshold. Though slightly *ad hoc*, empirically we observed that this was sufficient to distinguish the language when the model structure was correct.

4.1 Languages

We tried a number of well-studied languages from computational and mathematical linguistics, as well as some variations, see Table 1. **Bracket** is the bracket (Dyck) language (i.e., as and bs are balanced), which is known to be context free. The corresponding phenomenon in natural language is center embedding, which seems to exist only in a very restricted form. **Even** is the set of all strings from $\{a, b\}^*$ that are of even length, which is obviously a regular language. **ChinNr** is an abstract representation of Chinese number words (Radzinski, 1991), **GermScramb** of German verb scrambling (according to (Becker et al., 1992)). **AnBmCnDm** is known to be mildly context sensitive but not expressible by Linear Indexed Grammars (LIG). The same holds for **MultCop** with $k \geq 0$. **DepBranch**, the dependent branches language, is mentioned in (Vijay-Shanker et al., 1987) as an example of a language that cannot be generated by LIG. Note that **An** − **En** is also known to be beyond Tree Adjoining Grammars

Table 1. Definitions of target languages used. The column labels class states whether the language is regular (REG), context free (CF), mildly context sensitive (MCS) or context sensitive (CS). We use the map z, defined as $z(a) = e, z(b) = f, z(c) = g, z(d) = h$. $\pi(u)$ allows any permutation of the string u. Languages with an asterisk have had additional hard negative examples generated.

Name	Class	Definition	Example strings
Bracket	CF	$\{ab, avb, vw \mid v, w \in \textbf{Bracket}\}$	*aaabbb*, *ab*, *aabbab*
PalinDisj	CF	$\{vw \mid v \in \{a,b,c,d\}^*, w^R = z(v)\}$	*abcgfe*, *dh*, *bdhf*
Palin	CF	$\{vv^R \mid v \in \{a,b,c,d\}^*\}$	*aa*, *bdaadb*
Even	REG	$\{\{a,b\}^{2n} \mid n \in \mathbb{N}\}$	*cbbabc*, *acab*
ChinNr $*$	CS	$\{ab^{k_1} \dots ab^{k_r} \mid k_1 > \dots > k_r > 0\}$	*abbbbabbb*, *abbbabbab*
Mix	MCS	$\{s \in \{a,b,c\}^* : \lvert s\rvert_a = \lvert s\rvert_b = \lvert s\rvert_c\}$	*bac*, *babcac*
GermScramb	MCS	$\{\pi(w)v \mid w = z(v), v \in \{a,b,c,d\}^+\}$	*dbhf*, *bdacfghe*
AnBnCn $*$	MCS	$\{a^n b^n c^n \mid n > 0\}$	*bde*, *abdcfe*
An − Dn $*$	MCS	$\{a^n b^n c^n d^n \mid n > 0\}$	*adfh*, *bbcdefhg*
An − En $*$	CS	$\{a^n b^n c^n d^n e^n \mid n > 0\}$	*acegi*, *aadcfeghij*
DepBranch	CS	$\{a^n b^m c^m d^l e^l f^n \mid n = m + l \geq 1\}$	*aaaabcdddeeeffff*,
MultCop	CS	$\{w^k \mid k > 0, w \in \{a,b\}^*\}$	*babbab*, *abaabaaba*
AnBmCnDm $*$	MCS	$\{a^n b^m c^n d^m \mid n, m > 0\}$	*acfh*, *abaabccdeffeehgh*
CrossDepDA	MCS	$\{vzf(v) \mid v \in \{a,b,c,d\}^*\}$	*dbhf*, *cbdcgfhg*
CrossDepCS	MCS	$\{wxw \mid w \in \{a,b,c,d\}^*\}$	*cxc*, *bdaxbda*
CrossDepND	MCS	$\{ww \mid w \in \{a,\dots,h\}^*\}$	*cgcg*, *fcgefcge*
Omp4	CF	Omphalos problem 4	Omphalos website[2]

(TAG), in contrast to **An − Dn** and **AnBnCn**. The languages used in our training data are actually variants, where a^n is replaced by $\{a,b\}^n$, b^n by $\{c,d\}^n$ and so on. This was done to ensure that there were an exponentially large number of distinct strings in the language of bounded length. If not, simple memorization algorithms could perform well. **Mix** (Bach, 1981) is another well-known example of a mildly context sensitive language, and has been shown not to be expressible by TAG.

CrossDepDA is a copy language with just one copy w of v, where v and w have a disjunct alphabet. **CrossDepCS** is a copy language with just one copy, and a center symbol that marks the boundary between the two subwords, and **CrossDepND** is a copy language with one copy, and no center marker.

4.2 Results

Table 2 displays the results of training the baseline models and kernel PCA with two different kernels on these languages.[3] We can see that in all but four cases the string kernel method performs very well, converging to a hypothesis with very small error, whereas the baseline methods overgeneralize. In particular for

[2] `www.irisa.fr/Omphalos/data-sets.html`

[3] Since the data sets are synthetic, it is not appropriate to compare the figures from different rows, since the negative data has been generated to highlight the weaknesses of the various approaches.

Table 2. Test results on various data sets; training sets all of size 100. For each data set we report the percentage error rate separately for positive and negative data (lower is better). FP is the number of false positives as a percentage of the negative data, and FN is similarly the false negative rate. The kernels used are the 2-subsequence kernel and the 2-gap-weighted kernel with $\lambda = 0.5$. In both cases we added the Parikh kernel. The R column reports the dimension of the subspace, equivalently the rank of the data set in the feature space. We could not complete the PCFG experiments for the Omphalos data set because of the length of the strings.

Language	$\|\Sigma\|$	+/- Test	PCFG		HMM		1+2-subseq			GapWeighted		
			FP	FN	FP	FN	FP	FN	R	FP	FN	R
Bracket	2	537/463	0	0	3.4	1.3	10.8	0	3	10.8	0	5
PalinDisj	8	505/495	0.8	0	4	8.1	0	0	20	0	0	30
Palin	4	510/490	6.1	0	83.5	2.9	16.1	0	14	16.1	0	14
Even	3	739/261	0	0	0	0	100	0	12	100	0	12
ChinNr	2	500/500	94.4	0	36.0	0	100	0	6	100	0	6
Mix	3	500/500	100	0	94.6	0	0	0	5	0	0	10
GermScramb	8	500/500	100	0	97.8	0.6	0	0	26	0	0	51
AnBnCn	6	509/491	8.8	0	20.4	0	0	0	17	0	0	25
An – Dn	8	501/499	6.4	0	46.5	0	0	0	24	0	0	38
An – En	10	500/500	38	0	37.5	0	0	0	32	0	0	54
DepBranch	6	500/500	6.4	0	6.4	0	0	0	5	0	0	14
MultCop	2	500/500	100	0	99.2	1.2	100	0	6	100	0	6
AnBmCnDm	8	507/493	50.4	0	49.5	0	0.8	0	31	0.8	0	42
CrossDepDA	8	505/495	4.2	0	5.7	4.3	0	0	20	0	0	36
CrossDepCS	5	515/485	3.3	2.1	8.7	2.1	0	0	20	8.5	0	27
CrossDepND	8	506/494	64.2	5.0	76.3	6.3	70.0	6.7	71	100	0	72
Omphalos	25	277/305	-	-	17.4	0.4	0	30	344	0	6	325

our motivating example from Swiss German, **CrossDepDA**, we see a zero error rate. The four cases in question are: **ChinNr** where the HMM model performs well by learning a simple regular approximation; **Even** where both of the baseline models correctly learn the hypothesis; **MultCop** which is a very hard language to learn, in fact one of the authors was unable to determine what language it was from the generated positive data alone; and **CrossDepND** where in the absence of a midpoint symbol, there are no features that can define the language. In these cases, where the string kernel method fails to produce an accurate hypothesis, it overgenerates significantly. In the case of **AnBmCnDm** the string kernel method overgeneralises slightly, but very plausibly by allowing empty strings (generalising > 0 to ≥ 0). The string kernel method when applied to **Bracket** learns merely the hypothesis that there are equal numbers of as and bs, but is incapable of learning that no prefix must violate the constraint that there are more bs than as. HMM does well on **Bracket**, contrary to expectations, but merely by modelling some local features, even though it is CF.

The kernel approach performs well on **CrossDepDA** and **CrossDepCS**. Either the disjunct alphabet or the inclusion of a center symbol are sufficient for the kernel method to perform well. Note that though both kernels score perfectly on **CrossDepDA**, the 1+2-subsequence kernel will give false positives with certain strings such as *abbafeef*: strings of this type are so rare that they don't show up in test sets of this size. The Omphalos data is from a much more complex grammar, and consists of much longer strings. As a result, we could not use the PCFG algorithm, because the time complexity of the inside outside algorithm is cubic in the length. Note that though neither of the kernels can induce an accurate representation, some structure has been learned, even though as the high rank of the induced representations indicates, it overgenerated substantially.

In general, if the subspaces are of high rank, with respect to the feature space, then this is a clue that the algorithm has failed to capture significant structure. Indeed **CrossDepND** illustrates this perfectly: the dimension of the feature space with an alphabet size of 8 is 72 for both kernels, and the rank is 71 or 72. We do not report results here for kernels with longer features; on the same data sets, with $k = 3$, we have a very large number of false negatives, because the rank of the languages becomes very much higher. For example, on the **GermScramb** data set, with $k = 3$ the false positive rate goes up to 94% with a rank of 94. With the longer features the rank of this language has increased to 914, so the span of the training data, which has size 100, is clearly insufficient.

5 Discussion

When the target language is a planar language for the kernel being used, the algorithm converges rapidly and exactly. Clearly, the dimension of the hyperplane (equivalently, the rank of the data in the feature space) is the key factor. Denoting this by r, it is clear that any exact representation of the hypothesis requires at least r points that are independent in the feature space. Empirically we observed that the hypothesis converged rapidly after the first r points. Conversely, when the language being learned is not exactly expressible as a hyperplane, the hypothesis converged to a superset of the target language. Thus in general, for sufficiently large amounts of data, we observe false positives but no false negatives. In some cases this superset was the whole monoid, Σ^*; for example the language **Even**. This is a good example of a comparatively simple, regular language that cannot be represented as a hyperplane by any of the kernels that we use here. Of course, it would be easy to rectify this by considering a kernel that also had features corresponding to $\phi_n(w) = 1$ iff $|w|$ is divisible by n. This language is easily learnable by the HMM baseline, surprising as it may seem. Overall, the string kernel method performs very well on these languages, and outperforms the baselines in general, especially in the context sensitive languages.

The main computational bottleneck with this algorithm is the eigendecomposition of the *Gram* matrix, which means that the algorithm is cubic in the number of strings in the sample. However there are more efficient algorithms which exploit the generally low rank of the *Gram* matrix in these applications (incomplete Cholesky factorisation) which allow algorithms that are linear in the amount of data. In our experiments we found that the learning was reasonably rapid on standard workstations for data sizes up to about 1000 strings, without any optimisation.

5.1 Related Work

To the best of our knowledge, string kernels have not been used in this way before. The idea of using linear equations to define languages was discussed in (Salomaa, 2005), but the connection with string kernels has not been noted.

In terms of the results, there have been very few grammatical inference algorithms that have worked with representations capable of learning context sensitive languages, ignoring purely theoretical results that allow unbounded computation. The only relevant results that we are familiar with is a body of work using neural networks (Chalup & Blair, 1999). These papers show that under a suitable, carefully tuned training regime, various types of neural network are capable of learning some of these examples. However, these approaches do not generalise well, and are hard to train.

The choice of kernel is clearly very important here: there are a number of other kernels that can be devised that might be able to learn other classes of languages. One of the surprising aspects of this approach is that even when the induced feature space is of quite small dimension, the representational power of the formalism is quite high.

Hyperplanes are in some sense the easiest sets of points to learn in a Hilbert space. While they are effective for some languages, there are other languages, such as $\{a^n b^m \mid n > m > 0\}$, which do not form hyperplanes but rather half-spaces. These of course can be learned using, for example, the generalised portrait algorithm. Similarly other structures such as manifolds, or clusters on hyperplanes, would be learnable using other techniques, and would define other classes of languages.

6 Conclusion

We have put forward a new representation for languages, as hyperplanes in an induced feature space, and shown that these languages can be efficiently learned from positive data. We have demonstrated that this class of languages includes linguistically interesting context sensitive languages that are not learnable with current grammatical inference techniques.

Acknowledgments

This work has been partially supported by the EU funded PASCAL Network of Excellence on Pattern Analysis, Statistical Modelling and Computational Learning.

Bibliography

Bach, E. (1981). Discontinuous constituents in generalized categorial grammars. *North East Linguistics Society (NELS 11)* (pp. 1–12).

Becker, T., Rambow, O., & Niv, M. (1992). *The Derivational Generative Power of Formal Systems or Scrambling is Beyond LCFRS* (Technical Report 92–38). Institute For Research in Cognitive Science, University of Pennsylvania.

Chalup, S., & Blair, A. D. (1999). Hill climbing in recurrent neural networks for learning the $a^n b^n c^n$ language. *Proceedings of the Sixth International Conference on Neural Information Processing.* (pp. 508–513).

Clark, A., Costa Florêncio, C., Watkins, C., & Serayet, M. (2006). Planar languages and learnability. *International Colloquium on Grammatical Inference (ICGI).* Tokyo. to appear.

Gentner, T. Q., Fenn, K. M., Margoliash, D., & Nusbaum, H. C. (2006). Recursive syntactic pattern learning by songbirds. *Nature*, *440*, 1204–1207.

Kearns, M., & Valiant, G. (1989). Cryptographic limitations on learning boolean formulae and finite automata. *21st annual ACM symposium on Theory of computation* (pp. 433–444). New York: ACM.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *JMLR*, *2*, 419–444.

Parikh, R. J. (1966). On context-free languages. *Journal of the ACM*, *13*, 570–581.

Radzinski, D. (1991). Chinese number-names, tree adjoining languages, and mild context-sensitivity. *Comput. Linguist.*, *17*, 277–299.

Salomaa, A. (2005). *On languages defined by numerical parameters* (Technical Report 663). Turku Centre for Computer Science.

Schölkopf, B., Smola, A., & K., M. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*.

Shawe-Taylor, J., & Christianini, N. (2004). *Kernel methods for pattern analysis.* Cambridge University Press.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, *8*, 333–343.

Starkie, B., Coste, F., & van Zaanen, M. (2004). The Omphalos context-free grammar learning competition. *International Colloquium on Grammatical Inference* (pp. 16–27). Athens, Greece.

Vijay-Shanker, K., Weir, D. J., & Joshi, A. K. (1987). Characterizing structural descriptions produced by various grammatical formalisms. *Proceedings of the 25th annual meeting on Association for Computational Linguistics* (pp. 104–111). Morristown, NJ, USA: Association for Computational Linguistics.

Watkins, C. (2000). Dynamic alignment kernels. In A. J. Smola, P. L. Bartlette, B. Schölkopf and D. Schuurmans (Eds.), *Advances in large margin classifiers*, 39–50. MIT Press.

Toward Robust Real-World Inference: A New Perspective on Explanation-Based Learning

Gerald DeJong

Computer Science Department,
University of Illinois at Urbana
mrebl@uiuc.edu

Abstract. Over the last twenty years AI has undergone a sea change. The once-dominant paradigm of logical inference over symbolic knowledge representations has largely been supplanted by statistical methods. The statistical paradigm affords a robustness in the real-world that has eluded symbolic logic. But statistics sacrifices much in expressiveness and inferential richness, which is achieved by first-order logic through the nonlinear interaction and combinatorial interplay among quantified component sentences. We present a new form of Explanation Based Learning in which inference results from two forms of evidence: analytic (support via sound derivation from first-order representations of an expert's conceptualization of a domain) and empirical (corroboration derived from consistency with real-world observations). A simple algorithm provides a first illustration of the approach. Some important properties are proven including tractability and robustness with respect to the real world.

1 Introduction

The classification problem has become so emblematic of supervised machine learning that the two terms are sometimes used synonymously. Yet one can certainly imagine object classification as an inference rather than a learning task.

For example, suppose we know that birds fly, parakeets are birds, grackles are birds, etc. and that a particular object, Tweety, possesses the properties of being yellow, talkative, a parakeet, feathered, and so on. We can then infer that Tweety is to be classified into the set of flying things.

More abstractly we can realize classification using automated reasoning as follows. Let x refer to a world object whose representation X specifies a conjunction of input features. Let $C(\cdot)$ be the predicate denoting membership in some class L of interest, Π be the axioms capturing our prior knowledge, and $\vdash$ our inference relationship of choice. Then

$$X \cup \Pi \vdash C(x) \text{ implements } x \in L \tag{1}$$

Unfortunately, the automated reasoning of conventional logic has proven too brittle to be effective in the real world. Notoriously, the above would equally conclude that the neighbor's parakeet who was just killed by their cat can fly, as can Bugsy the Mafioso grackle whose feet are cast in a block of cement.

Consider an expression ϕ that is derivable through our inference relation $\vdash$ from a set of statements Δ:

$$\Delta \vdash \phi \tag{2}$$

We would like our domain theories to exhibit real-world robustness; inferring a sentence should provide some guarantee that it holds in the real world. This is expressly not the case in conventional logic where formal guarantees about ϕ's apply only to the microworld defined by Δ. If a conventional inferential system is robust in this sense, it is due exclusively to the human implementor's care and cleverness in crafting Δ and is beyond the scope of the formal inferential system.

Explanation-Based Learning (EBL) can be viewed as a path around this brittleness. We suggest a kind of paradigm shift to the interpretation of EBL. Instead of seeing EBL as bringing prior knowledge to the learning task, we explore EBL as bringing learning to the inference task; it formalizes robustness in deductive inference.

Suppose we could achieve real-world robustness within the scope of some formalism. What robustness properties would be desirable and what would be possible? We might at first hope that the ϕ's of (2) be guaranteed to hold in the real world (i.e., that derivability from some appropriate set of axioms suffices to trust that a statement is true in the real world). We believe this to be impossible. Instead, we propose a slightly weaker guarantee in which the inferred ϕ's hold in the real world only with high confidence in a sense that follows from statistical learning.

This new EBL is based on four tenets.

1. The human expert's domain representations reflect a deep appreciation of world's subtleties that the computer learner is unlikely to equal and should not question. In EBL, training examples guide the *interpretation* of the human-supplied domain theory with respect to a particular task, and not its refinement or improvement.
2. Robustness cannot be achieved generally but only with respect to a particular domain task. A formal connection to this real world task is a requisite for robust inference. Statistics supports robustness guarantees but conventional logic does not.
3. The absolute inferential confidence afforded by conventional logic, while seductive, is often unrealistic in the real world and can be a major source of brittleness. Human inference is generally weaker than this as is statistical inference.
4. Complex domains require an appropriately expressive language for the expert to articulate his/her understanding. Propositional models, relational models, description logics, etc. are sufficient for some domains, but first order representation with the combinatorial interaction that unification affords can provide a greater conceptual richness. This is realized in logic but is not easily incorporated into statistics.

Explanation-Based Learning requires an expert-supplied domain theory, Δ. We take this to be a set of first order expressions. But the intent is to allow the expert's unencumbered conceptualization of the domain. As such, we expressly do not require the theory to possess the difficult-to-achieve global properties of consistency or robustness. A second input is a set of training observations, $\mathbf{Z}$. These provide a direct connection to the real world. The result of EBL is a specially tailored logic domain theory, Δ' which is likely to be robust in the task illustrated by the training examples. Thus,

$$\Delta \cup \mathbf{Z} \underset{EBL}{\mapsto} \Delta' \tag{3}$$

In EBL the prior domain theory Δ is deemed to be correct but *not* believed. Unlike ILP or theory revision, EBL does not attempt to improve or augment the expert-supplied domain theory. Rather, Δ is viewed as the most comprehensive and maximally useful general domain description that the human expert can provide. Like all first-order theories, it is subject to the qualification problem:

> Most universally quantified sentences will have to include an infinite number of qualifications if they are to be interpreted as accurate statements about the world. [9]

Flaws cannot be avoided, so encountering them cannot be taken as evidence against the worth of the human's statements. Revising Δ may simply compromise the human's expression of his/her expertise resulting in a worse theory. But by the same token, neither can Δ be believed. Due to the myriad unavoidable flaws, a particular statement cannot be accorded *any* degree of belief simply by virtue of its derivation via sound inference.

In EBL, inference over Δ is permitted only to tie together actual world observations (e.g., to *explain* labeled training examples). This constitutes *analytic* evidence, and Δ may support many incompatible explanations for the same training instance. All are causally well formed. Choosing which (if any) explanations to accept relies largely on statistics.

An explanation is syntactically identical to a theorem. The mechanism for building explanations is just theorem proving. Each explanation "derives" the teacher-assigned classification label from the object's observable features using the statements of Δ. In doing so, the explanation ascribes a causally well-formed set of hidden or latent features to the object. These additional (unobservable) properties are introduced by inference via the domain theory. They are compositions of distinctions that the expert has found useful in expressing his or her causal conceptualization of the domain.

Each explanation hypothesizes that a particular hidden causal structure is sufficient to determine an object's class label accurately in the context of the classification problem. Thus, while syntactically the process looks like theorem proving, semantically it amounts to conjecturing a statistical hypothesis about how to estimate the classification label from a (potentially complex) pattern of observable features. Hypotheses that are statistically confirmed by independent real-world examples (and therefore possess the desired robustness properties) are re-packaged into a conventional domain theory Δ'. Thus, the elements of Δ' are believed, but the elements of Δ are not the kinds of things that merit belief nor for which statistical evidence is even relevant.

It is instructive briefly to consider the classical view of EBL/EBG [14,6,15,25]. Here one would logically deduce that a goal relation holds from the example's input features. Δ, assumed to be complete and correct, might include that x can be *safely-stacked* on y if x is lighter than y or y is not fragile. Other statements provide various methods for computing weight. Observing that a particular vase can be safely stacked atop a particular table, the learner constructs a general sufficient rule for concluding *safely-stacked* from the constituent volumes and densities and not depending on the identity of the objects, their colors, owners, etc. This classical EBL, sometimes referred to as speed-up learning, inherits the brittleness of conventional first-order logic upon which it is built.

2 Brittleness and Robustness

The brittleness of conventional logic can be largely traced to properties (2) and (3) above. In the conventional logical paradigm, an axiom set, Δ, represents a model of world interactions. The set of expressions that can be inferred about the [micro]world is precisely the set of expressions Φ entailed by the axioms:

$$\Delta \models \Phi \tag{4}$$

The brittleness follows from the afore-stated qualification problem and the unforgiving nature of logical semantics. Logical inference ascribes equal absolute belief in all logical consequences. Thus, the qualification problem assures us that there will be some anomalous consequences from the logical formalization of any reasonably interesting subset of the real world, while the semantics of logic assures us that these anomalies, if encountered in practice, may be devastating to our reasoner's robustness.

But statistical inference does not suffer from such brittleness. In the conventional statistical paradigm one adopts some parameterized family of (statistical) models, $\mathfrak{M}$. Often in the statistical literature, the term *model* is used to refer to the family, but we reserve that term for a specific candidate or stand-in for the world. In the simplest version, a member of the family, $\mathcal{M} \in \mathfrak{M}$, is chosen according to a set of world observations $\mathbf{Z}$. Then the set of expressions Φ judged to hold in the world are those that $\mathcal{M}$ accepts as worth believing. As a specific illustration, $\mathfrak{M}$ might be the graphical structure of a Bayesian net. The observations $\mathbf{Z}$ are used to estimate conditional probabilities which in turn individuate a specific explicit probability distribution. Perhaps we believe those things whose ascribed probability is greater than 0.5. Equally, we might choose a discriminative model. For example, $\mathfrak{M}$ might be a family of linear separators. Here $\mathbf{Z}$ is a training set used to select a specific linear separator, $\mathcal{M}$. The collection of believed expressions Φ might be the ones that fall above the linear discriminant $\mathcal{M}$. To paraphrase (4) we might denote a weaker form of "entailment" of the set of new sanctioned beliefs Φ as:

$$\mathfrak{M} \cup \mathbf{Z} \models\!\!\approx \Phi \tag{5}$$

Here the model family $\mathfrak{M}$ augmented with a set of world observations $\mathbf{Z}$ provides sufficient justification to believe each $\phi \in \Phi$. The nonstandard symbol $\models\!\!\approx$ is used (rather than $\models$) to denote that this inference provides less than the absolute confidence of logical entailment.

In the statistical paradigm exceptions are embraced within the formalism. Inferring ϕ but observing $\neg\phi$ in the real world results in a new augmented set of world observations $\mathbf{Z}' = \mathbf{Z} \cup \neg\phi$ which together with $\mathfrak{M}$ may eventually result in the selection of a different specific model $\mathcal{M}'$. Even after observing $\neg\phi$ it is quite possible that $\mathfrak{M} \cup \mathbf{Z}' \models\!\!\approx \phi$, at least until *sufficient* contradictory evidence accrues. In this view, statistical inference naturally embodies a kind of nonmonotonicity; it requires no additional mechanisms.

EBL borrows the two statistical robustness characteristics above, letting Δ (with a sound inference procedure) play the role of the statistical prior commitments, $\mathfrak{M}$. Thus, we examine inference systems that implement something like the following:

$$\Delta \cup \mathbf{Z} \mathrel{|\!\!\approx} \Phi \tag{6}$$

Δ, as before, is a set of first-order sentences. But this Δ is not required to be consistent or robust. EBL implements (6) via (3) so that Δ' is a compact approximate representation of Φ thus: $\Delta' \models \Phi'$ and Φ' approximates Φ. Inference is performed conventionally over Δ'.

3 A Simple Classification EBL Algorithm

To illustrate, we employ English sentences rather than first-order ones. The translations are straightforward. For example, sentence 2 is the notorious $\forall x Bird(x) \Rightarrow Flies(x)$.

As a domain, we are interested in which animals can fly. Each specific animal is defined by a conjunction of ground observable features (name=Tweety, species=parakeet, color=yellow, etc.). An expert supplies us with a (non-robust) domain theory, Δ:

1. flying is kind of locomotion
2. birds fly
3. locomotion is a volitional act
4. dead things do not act volitionally
5. flying requires wings
6. wings need a particular geometry
7. flying requires a favorable power to weight ratio
8. cooking causes animals to be dead
9. sick animals are weak
10. cement blocks are heavy
11. birds are animals
12. penguins, ostriches... do not fly
13. penguins are birds
14. robins are birds

... many other similar sentences

This is not an acceptable conventional axiom set. There are contradictions (e.g., 2, 12, and 13). More importantly, there are many derivable expressions that incorrectly portray the world. For example, from 1-4 we deduce that birds cannot be dead.

But an EBL system will never conclude that any particular bird both flies and does not fly because it will never observe a bird that requires an explanation of how it can simultaneously fly and not fly. Likewise, only after seeing an immortal bird would we entertain the explanation from 1-4 for why this animal might never die.

Robustness is defined with respect to a domain task. We conceive a **task** as an unlimited sequence of related questions posed to the reasoner. The questions are drawn randomly form a space of well-formed questions according to some fixed but unknown distribution. A theory is **robust** for a task if the answers produced by some sound inference procedure are usually the answers that the real world would give. An EBL system is robust if it usually produces robust theories. The user supplies an error tolerance parameter $0 < \epsilon \ll 1$ (bounding the probability of disagreement with the real world) and a confidence parameter $0 < \delta \ll 1$ (bounding the probability that the constructed theory is not robust).

Thus, the five inputs to an EBL system are Δ, **Z**, a task, ϵ, and δ. The output is a new theory Δ' such that with probability of at least $1 - \delta$, the real-world accuracy of Δ' on the task is at least $1 - \epsilon$. Consider Δ:

$$\begin{aligned}
&\Delta_1 : \forall x\ Flies(x) \Rightarrow Locomotes(x)\\
&\Delta_2 : \forall x\ Bird(x) \Rightarrow Flies(x)\\
&\Delta_3 : \forall x\ Locomotes(x) \Rightarrow Volition(x)\\
&\Delta_4 : \forall x\ Dead(x) \Rightarrow \neg Volition(x)\\
&\Delta_5 : ...
\end{aligned}$$

We posit the following procedures:

EXPLAIN(Δ,S,L): a sound theorem prover implementing the inference relation $\vdash$. This serves as an explanation generator. We require that explanations with shorter derivations are constructed before longer ones. Given a non-robust domain theory Δ, a set of world observations S, and a literal of interest L, EXPLAIN returns a succession of proof trees. Each derives the assigned literal truth value from one or more observations in S.

RULEGEN(E): a simple rule generation procedure such as that of Mooney and Bennett [15] which essentially lifts and flattens the proof tree. The result is a new first order statement that concludes L and tests only observable predicates.

WORLD(L,N): a protocol for monitoring the real world. We specify a literal of interest, L, and a positive integer N. It notifies us of new occurrences and succeeds after seeing N.

For each literal L of interest the following algorithm is invoked:

```
1. Set B to the singleton observation WORLD(L,1)
2. Set E (an explanation) to EXPLAIN($\Delta$,B,L)
3. Set R (a hypothesized rule) to RULEGEN(E)
4. Evaluate R on WORLD(L, $\ln(2/\delta)/2\epsilon^2$)
5. If R is correct on all of these, END returning R as the robust rule for L
6. Else reject R and add the new observations to B
7. Go to Step 2
```

We assume that the domain theory was created by a true expert and is *adequate* in a sense we will make formal in section 5. Basically, mixed in with all of the specious causal analyses there must be at least one that satisfies our robustness requirements, and the expert cannot intentionally make those ones more difficult to find.

Now suppose we see Rob, a flying robin. He is explained by an instantiation of Δ_2. Lifting and flattening results in a statement identical to Δ_2 hypothesized to be included in Δ':

$$\forall x\ Bird(x) \Rightarrow Flies(x) \tag{7}$$

Next we see Tom, the non-flying turkey. Encountering Tom initiates two computations. First, he serves to refute 7. Second, a derivation is initiated to explain Tom's non-flight whose simplest explanation is:

$$\begin{array}{rcll}
\neg Flies(Tom) & & & \text{Given} \\
\| & & & \\
\neg Flies(Tom) & \Leftarrow & \neg Locomotes(Tom) & \text{CP } \varDelta_1 \\
& & \| & \\
\neg Volition(Tom) & \Rightarrow & \neg Locomotes(Tom) & \text{CP } \varDelta_3 \\
\| & & & \\
\neg Volition(Tom) & \Leftarrow & Dead(Tom) & \varDelta_4 \\
& & \| & \\
& & Dead(Tom) & \text{Given}
\end{array}$$

(CP means contrapositive of while || shows unifications) This explanation when lifted and flattened yields the conjectured rule:

$$\forall x\ Dead(x) \Rightarrow \neg Flies(x) \tag{8}$$

Next we see more world observations from our bird flying task. Some, the sparrows and blue jays eating from our backyard feeder, fly. Others, the roasted chicken we have for dinner later in the week, the cooked Cornish game hen the next day, the neighbor's parakeet killed by their cat, do not fly. Rule 7 is refuted. This does not change $\varDelta$ (which still contains $\varDelta_2$). But 7 is dropped from consideration from $\varDelta'$. This reinvokes EXPLAIN on Rob, Tom, and the other evidence observations that participated in the evaluation of 7. Two additional inference rules are required by EBL. We will see these in the next section. The second one, (13), is used here to conjecture the rule:

$$\forall x\ Bird(x) \wedge \neg Dead(x) \Rightarrow Flies(x)$$

After a significant number of similar observations this and (8) are statistically confirmed and $\varDelta'$ becomes:

$$\begin{aligned}
&\forall x\ Dead(x) \Rightarrow \neg Flies(x) \\
&\forall x\ Bird(x) \wedge \neg Dead(x) \Rightarrow Flies(x)
\end{aligned}$$

If we had seen a significant number of airplanes, penguins, emus, Mafioso birds, etc. these rules would be different. But in this task context, these rules encounter many confirming and no disconfirming examples.

4 Some Semantic Properties of Explanations

The expert provides us with a set of first-order sentences that capture his or her understanding of the domain:

$$\varDelta = \{\theta_i\}\ \ i = 1, r \tag{9}$$

Each first-order θ only approximates some underlying constraint of the real world. Thus, we interpret the meaning of each statement θ_i as $\alpha_i \Rightarrow \theta_i$ where the α's denote the (possibly infinite) missing qualifiers from the qualification problem. The veracity of a deduction over $\varDelta$ depends on the unmodeled α's and the unifications performed.

Without loss of generality (by renaming variables as needed) assume that we have a single global unifier, $\varGamma$, constructed as a side effect of the explanation process. A necessary and sufficient condition for the deduction to hold in the real world is the conjunction of the implicit qualifications:

$$\left(\bigwedge_j \alpha_j\right) \circ \varGamma \;\; j = 1, s \tag{10}$$

where $\circ$ denotes the specialization of a first-order expression by the application of a unifier. The index j ranges over sentences in the deduction.

To achieve robustness, EBL insures that the corresponding expression (10) of each sentence in $\varDelta'$ holds with high probability. But note that this needs to be met only *when the sentence is applied*. Thus, in EBL we are interested in

$$Pr\left((\textstyle\bigwedge_j \alpha_j)\circ\varGamma \;\; {\scriptstyle j=1,s}\right) \tag{11}$$

where the probability distribution is taken over just those situations in which the inference mechanism chooses to construct and employ the sentence. The EXPLAIN procedure draws inferences according to this very distribution when constructing explanations for real world observations. As a derivation grows, its (10) incurs an additional independent chances to fail so (11) tends to diminish. Let λ be the expected erosion of confidence from a single additional inference step.

We assume the inference mechanism is paraconsistent or "inconsistency-tolerant" (e.g., relevance logic), and we disallow hyper-inference. We also require additional inference rules that define the inference relation $\vdash$ (implemented here by EXPLAIN). The new inference rules are sound but unnecessary in conventional logic. For intuitive clarity we state them using implications. Of course, as with first-order inference rules, the identity match (of ϕ in each case) can be brought about by specializing first-order expressions through unification.

The first rule is a kind of AND introduction:

$$\frac{\begin{array}{c}\psi \Rightarrow \phi \\ \varphi \Rightarrow \phi\end{array}}{(\psi \wedge \varphi) \Rightarrow \phi} \tag{12}$$

While sound, this inference rule is logically unnecessary. Statistically it is quite useful. If alone, explanations from each of two pieces of evidence are insufficient statistically, then the desired accuracy might be achieved by insisting on both evidentiary sources together.

A second important evidentiary rule that is logically unnecessary is:

$$\frac{\begin{array}{c}\psi \Rightarrow \phi \\ \varphi \Rightarrow \neg\phi\end{array}}{(\psi \wedge \neg\varphi) \Rightarrow \phi} \tag{13}$$

The first statement subsumes the inferential conclusion, making it logically redundant. But statistically it resembles conditioning. In the Rob / Tom illustration, given only one of the observations, we can conjecture that "birds fly" or that "dead things do not fly." In the presence of both kinds of world observations, the first statement cannot be confirmed. This rule allows conjecturing the composite sentence that "birds that are not dead can fly."

5 Analysis of Simple EBL

To prove that the simple EBL algorithm works, we will use λ, introduced in the previous section, β and γ, introduced below, and the parameters ϵ and δ, introduced in section 3.

Proposition 1: If simple EBL produces a rule whose actual real-world task accuracy is a, then

$$Pr(a < (1 - \epsilon)) \leq \delta/2 \tag{14}$$

That is, its true accuracy cannot be far from its measured accuracy (which is 100% or else it would not have been produced).

Proof: The additive Chernoff bound requires the true mean of a distribution μ and an estimated mean $\hat{\mu}$ based on m samples from the distribution respect $Pr(\mu < \hat{\mu} - \epsilon) \leq e^{-2m\epsilon^2}$
Let μ be the true real world accuracy of the rule. The rule is accepted only if it makes no errors so $\hat{\mu} = 0$. Letting m be $\ln(2/\delta)/2\epsilon^2$ (the algorithm's number of world observations) yields (14).

Proposition 2: The set of explanations from Δ can grow no faster than exponentially as inference depth increases. This follows from the assumptions of a sound paraconsistent logic without hyper-inference. This is proved in [20].

Definition 1: The *Domain Adequacy Measure* γ of a theory is $\lambda \cdot \beta$ where λ is the expected inferential erosion of confidence from the previous section and β is the base of the exponential from Proposition 2.

Definition 2: An input domain theory Δ is *adequate* iff some robust rule can be derived for any real world questions of interest and $\gamma < 1$. The domain expert must include in Δ a sufficient set of conceptual distinctions and causally simple explanations cannot be hidden in a plethora of easy-to-derive Rube-Goldbergish ones.

Proposition 3: If given an adequate theory, a question of interest, and a set of world observations, then a robust explanation (one giving rise to a rule robustly answering the question of interest in the real world) can be found with probability at least 1-δ/2 in no more than $n + k$ inference steps where n is the number of inference steps needed to find the simplest explanation for the observations, and k grows no faster than logarithmically in δ and is independent of the complexity parameter ϵ.

Proof: We are given an adequate domain theory, an untested explanation derivable with n inference steps, and the assurance that no explanation is possible with fewer than n steps (as this one is stipulated to be the simplest). The probability of finding the first robust explanation R at inference level j is $Pr(R|j) = \gamma^j$. Thus, the distribution is a decaying exponential with base γ. We conditionalize on the known information that there is zero probability of finding an R before inference level n. This shifts the origin of the distribution (which must still sum to 1) from 1 to n. The discrete distribution from n to ∞ is bounded by the continuous function $\int_{n+k}^{\infty} \gamma^{x-n} dx$. Bounding this in turn by

$\delta/2$ yields $k > (ln(\delta/2) + ln(ln(\gamma)))/ln(\gamma)$. Which by inspection has the specified dependence on ϵ and δ.

Theorem: With probability at least δ Simple EBL produces rules with real world accuracy of at least $1 - \epsilon$ and requires no more than a number of relevant world observations polynomial in $1/\epsilon$ and $1/\delta$.

Proof: We split the δ resource into two halves. Proposition 1 limits the probability to $\delta/2$ that a rule passes the robustness test but fails to actually achieve a real world accuracy of $1 - \epsilon$. Each robustness test consumes a polynomial number of relevant real-world observations (Proposition 1). With k additional inference levels only a polynomially growing number of additional explanations can be encountered: The set of explanations may grow exponentially in k (Proposition 2), but k grows only logarithmically in the complexity parameters (Proposition 3). The first adequate explanation might not be found in k additional inference steps but this is unlikely. By Proposition 3 this occurs only with probability $\delta/2$. Thus, the algorithm fails at most δ of the time (half the time by poorly testing a rule, and half the time by failing to search far enough to find the first good rule).

6 Domain Adequacy, Scaling, and Algorithmic Complexity

Will our new form of EBL scale to non-toy problems? Our first rather tentative step does not answer this important question. There is reason for concern: using full first-order explanations, EBL's worst case time (NB: not example) complexity is at least exponential for derivation membership and undecidable for non-membership. Of course, a less expressive explanation engine would force more favorable algorithmic complexity guarantees.

But we believe that the key to scaling may lie in the notion of *domain adequacy*. Domain adequacy provides a new quality measure on domain theories upon which algorithmic complexity can depend. Here we employed a simple adequacy measure $\gamma = \lambda \cdot \beta$ sufficient for example complexity. But this is just a first cut at a deep and important contrast to logical adequacy. The rules of chess and Peano's axioms may appear at first to be perfectly adequate for their respective domains. But an expert chess tutor prefers to describe games using less-precise invented domain terms such as "center control," "weak pawn structure," and "underdeveloped queen side." From the viewpoint of EBL, these expert-introduced terms are both more informative and more flexible. As latent variables or sub-concepts, they must be learned by the student. Their "correct" definition can (and will) depend on the student's own emerging idiosyncratic style of play. We believe this new EBL approach opens the door to a richer notion of "domain theory" and "training example." In this new EBL, the role of prior knowledge is to linearize the learning problem by conjecturing alternative sufficient sets of manageable sub-problems. Conventional notions of accuracy, satisfiability, or sets of possible worlds cannot express these important characteristics of a domain theory. Perhaps domain theory adequacy can asymptotically bound the number of wasted inference steps by employing appropriate sub-concepts.

7 Related Work and Conclusions

The approach owes much to earlier work on learning with domain theories and declarative bias [24,3,4,2,18]. Inductive Logic Programming is also relevant [13,17], as is the work on theory revision [26,19,10,1]. These also acquire expressive representations but theirs is a much more ambitious goal of improving the expert-supplied domain theory rather than constructing a new specialized theory for a particular narrow task. The work combining first-order knowledge with statistics is also relevant (e.g., [16,5]), as is learning in probabilistic logics (see [7]). However, in our Explanation-Based Logic no statistical or numerical manipulations take place during inference; there are no probabilities attached to the sentences in the robust output domain theory Δ'. This avoids a computational pitfall [23] without constraining the expressiveness of the result to a (propositional) Bayesian net as [11,12]. The Knowledge-Based Neural Networks approach [27] is similar, utilizing a propositional neural net rather than Bayesian net. The burgeoning area of relational learning ([22,21]) is also relevant, although link learning, relational learning, learning with description logics, etc. all employ knowledge representations that fall short of first-order expressiveness.

For some application domains, this new form of EBL may allow complex concepts to be learned from small more human-proportioned training sets. Possible applications include intelligent interfaces in which the system can learn to fit its user rather than forcing the human user to learn about it, and context adaptive computing in which a computer system specializes itself to its perceived deployment context. A preliminary less declarative illustration of this direction can be found in [8].

The main contribution is tolerating semantic approximation in the expert-supplied logic-like statements, and the use of world observations as evidence to interpret this domain knowledge.

Acknowledgements. The Illinois EBL group, particularly Arkady Epshteyn, Shiau Hong Lim, Geoff Levine, and Li-Lun Wang, contributed immensely to the work. It also benefited greatly from comments from Ron Brachman that helped to clarify a number of issues. The research was supported by the National Science Foundation under Award NSF IIS 04-13161 and by the Information Processing Technology Office of the Defense Advanced Research Projects Agency under award HR0011-05-1-0040. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF or DARPA.

References

1. S. D. Bay, D. G. Shapiro, and P. Langley. Revising engineering models: Combining computational discovery with knowledge. In *Thirteenth European Conference on Machine Learning*, pages 10–22, 2002.
2. Clifford Brunk and Michael J. Pazzani. A lexical based semantic bias for theory revision. In *ICML*, pages 81–89, 1995.
3. Peter Clark and Stan Matwin. Using qualitative models to guide inductive learning. In *ICML*, pages 49–56, 1993.
4. William W. Cohen. Incremental abductive ebl. *Machine Learning*, 15(1):5–24, 1994.

5. James Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271, 2001.
6. Gerald DeJong and Raymond Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176, 1986.
7. Luc DeRaedt and Kristian Kersting. Probabilistic logic learning. *SIGKDD Explorations*, 5(1):31–48, 2003.
8. A. Epshteyn, M. Garzaran, G. DeJong, D. Padua, G Ren, X. Li, K. Yotov, , and K. Pingali. Analytic models and empirical search. In *The Eighteenth International Workshop on Languages and Compilers for Parallel Computing*, 2005.
9. Michael Genesereth and Nils Nilsson. *Logical Foundations of Artificial Intelligence*. Kaufmann, Los Altos, CA, 1987.
10. Russell Greinter. The complexity of theory revision. *Artificial Intelligence*, 107(2):175–217, 1999.
11. Peter Haddawy. Generating bayesian networks from probablity logic knowledge bases. In *UAI*, pages 262–269, 1994.
12. Niels Landwehr, Kristian Kersting, and Luc De Raedt. nfoil: Integrating naïve bayes and foil. In *AAAI*, pages 795–800, 2005.
13. Nada Lavrac and Saso Dzeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, , New York, 1994.
14. Tom Mitchell, Richard Keller, and Smadar Kedar-Cabelli. Explanation-based learning: A unifying view. *Machine Learning*, 1(1):47–80, 1986.
15. Raymond Mooney and Scott Bennett. A domain independent explanation-based generalizer. In *AAAI*, pages 551–555, 1986.
16. Katharina Morik, Peter Brockhausen, and Thorsten Joachims. Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In *ICML*, pages 268–277, 1999.
17. Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *J. Log. Program.*, 19/20:629–679, 1994.
18. C. Nedellec, C. Rouveirol, H. Ade, F. Bergadano, and B. Tausend. Declarative bias in ILP. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 82–103. IOS Press, Amsterdam, 1996.
19. Dirk Ourston and Raymond Mooney. Theory refinement combining analytical and empirical methods. *Artif. Intell.*, 66(2):273–309, 1994.
20. David Plaisted and Yunshan Zhu. *The Efficiency of Theorem Proving Strategies*. Vieweg & Sohn, Wiesbaden, 1999.
21. Luc De Raedt, Thomas Dietterich, Lise Getoor, and Stephen H. Muggleton, editors. *Probabilistic, Logical and Relational Learning*, Dagstuhl Seminar Proceedings, 2005.
22. Luc De Raedt and Stefan Kramer, editors. *Workshop on Attribute-Value and Relational Learning: Crossing the Boundaries at ICML 2000*, 2000.
23. Dan Roth. On the hardness of approximate reasoning. In *IJCAI*, pages 613–619, 1993.
24. S. J. Russell and B. N. Grosof. A declarative approach to bias in concept learning. In *Proc. of AAAI-87*, pages 505–510, Seattle, WA, 1987.
25. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.
26. Lorenza Saitta, Marco Botta, and Filippo Neri. Multistrategy learning and theory revision. *Machine Learning*, 11(2-3):153–172, 1993.
27. Geoffrey G. Towell and Jude W. Shavlik. Knowledge-based artificial neural networks. *Artif. Intell.*, 70(1-2):119–165, 1994.

Fisher Kernels for Relational Data

Uwe Dick and Kristian Kersting

University of Freiburg, Institute for Computer Science, Machine Learning Lab,
Georges-Koehler-Allee, Building 079, 79110 Freiburg, Germany
{dick, kersting}@informatik.uni-freiburg.de

Abstract. Combining statistical and relational learning receives currently a lot of attention. The majority of statistical relational learning approaches focus on density estimation. For classification, however, it is well-known that the performance of such generative models is often lower than that of discriminative classifiers. One approach to improve the performance of generative models is to combine them with discriminative algorithms. Fisher kernels were developed to combine them with kernel methods, and have shown promising results for the combinations of support vector machines with (logical) hidden Markov models and Bayesian networks. So far, however, Fisher kernels have not been considered for relational data, i.e., data consisting of a collection of objects and relational among these objects. In this paper, we develop Fisher kernels for relational data and empirically show that they can significantly improve over the results achieved without Fisher kernels.

1 Introduction

From a machine learning perspective, many real world applications can be regarded as classification problems: One tries to estimate the dependence of a target variable Y on some observation $\mathbf{X}$, based on a finite set of observations $\mathbf{x}$ for which the value y of the target variable is known. More formally:

Definition 1 (Classification Problem). Given *a finite set of training examples* $\{(\mathbf{x}_i, y_i)\}_{i=1}^m \subseteq \mathcal{X} \times \mathcal{Y}$ *, where* $\mathcal{X}$ *is the feature space and* $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ *is the set of possible classes,* **find** *a function* $f : \mathcal{X} \to \mathcal{Y}$ *with low approximation error on the training data as well as on unseen examples.*

The classification problem has traditionally been considered for data in attribute-value form, i.e., $\mathbf{x}_i$ is a vector of fixed length. Many — if not most — real-world data sets, however, are not in attribute-value form. Applications are characterized by the presence of uncertainty and complex relations. Consider the KDD Cup 2001 localization of genes/proteins task [1].

Example 1 (Genes/Proteins Localization). The KDD Cup 2001 focused on data from life science. One data set, which we also used in our experiments, is from genomics. The data consists of 1243 genes of one particular but unknown type of organism. Each gene encodes a protein, which occupies a particular position in some part of a cell. For each gene, information on the class, the phenotype, i.e., its characteristics, the complex it belongs to etc. are given. Furthermore, the graph of interactions among the genes is provided. Using relational logic, this can elegantly be represented as set of ground atoms `gene(g1). gene(g2). ... phenotype(g1, 1). ... complex(g2, 13). ... interaction(g1, g2). interaction(g3, g245) ...` Here, `gene/1`, `phenotyp/2`, `gene/1` `interaction/2` are *predicates* that identify relations, numbers and lower-case strings like `g1` and `1` are *constants* that identify objects. *Ground atoms* are predicates together with their arguments, for example `interaction(g3, g245)` denotes that genes `g3` and `g245` interact. 381 of the 1243 genes are withheld as test set. The task is to predict the localization of a protein/gene based on the features of the protein/gene and of proteins/genes interacting with the protein/gene and is characterized by the presence of uncertainty, a varying number of objects (genes), and relations (interactions) among the objects.

Inductive logic programming [17] (ILP) and relational learning have been developed for coping with this type of data. They aim at inducing hypotheses consisting of *clauses* c (abstract rules) such as

$$\texttt{localization(A)} \ :- \ \texttt{neighbour(A,B)}, \texttt{localization(B)}$$

which consist of a head(c) $\equiv$ `localization(A)` and a body(c) $\equiv$ {`neighbour(A, B)`, `localization(B)`}. Upper-case strings denote *variables*, i.e., placeholders for objects. *Atoms* are predicates together with their arguments, for example `localization(A, B)`. A clause or atom is called *ground* if it does not contain any variables. Relational abstraction has two advantages: (1) variables such as `A`, i.e., placeholders for objects allow one to make abstraction of specific objects such as `g1`; (2) unification {`A/g3, B/g245`}, i.e., the matching of variables allows one to evaluate abstract knowledge. Thus, relational learning allows to induce general regularities in terms of clauses but it does not handle uncertainty in a principled way. It is therefore not surprising that there has been a significant interest in integrating statistical learning with relational representations. This newly emerging research field is known under the name of *statistical relational learning* (SRL) and aims in principle at estimating a probability distribution $\mathbf{P}(\mathbf{X}, Y)$ over relational $\mathcal{X} \times \mathcal{Y}$. The key idea of SRL is to employ relational abstraction within statistical learning and therefore learning general (abstract) statistical regularities among groups of entities.

For classification, most SRL approaches (in particular the ILP motivated ones) are *generative*, i.e., they aim at estimating the joint distribution $\mathbf{P}(\mathbf{X}, Y)$ by learning the class prior distribution $\mathbf{P}(Y)$ and the class-conditional feature distribution $\mathbf{P}(\mathbf{X}|Y)$. The required posterior distribution $P(Y = y|\mathbf{X} = \mathbf{x})$ is then obtained using Bayes' rule yielding $f(\mathbf{x}) = \arg\max_{y_i \in \mathcal{Y}} P(\mathbf{X} = \mathbf{x}|Y = y_i, \boldsymbol{\lambda}^*) \cdot P(Y = y_i|\boldsymbol{\lambda}^*)$ as solution to the classification problem 1. Here, $\boldsymbol{\lambda}^*$ are the maximum likelihood parameters of the given generative model, which are typically estimated using the EM algorithm.

The classification performance of a generative approach, however, is often lower than that of a discriminative classifier, which estimates $f : \mathcal{X} \rightarrow \mathcal{Y}$ directly without representing the full joint probability distribution $\mathbf{P}(\mathbf{X}, Y)$. To improve the classification accuracy of generative models, different kernel functions have been proposed to make use of the good predictive performance of kernel methods such as support vector machine (SVM) [20]. A prominent representative of these kernel functions is the *Fisher kernel* [9]. The key idea there is to use the gradient of the log likelihood of the generative model with respect to its parameters as features. The motivation to use this feature space is that the gradient captures the generative process rather than just the posterior probabilities.

Fisher kernels have successfully been applied in many learning problems where the instances are described in terms of attribute-value vectors and for sequences of logical atoms [12]. So far, however, they have not been applied to relational data. Our main contribution is the definition of *relational Fisher kernels*, i.e., Fisher kernels derived from SRL models.

Definition 2 (Relational Fisher Kernel). *Relational Fisher kernels are the family of kernel functions k obtained by using the gradient $U_{\mathbf{x}} = \nabla_{\boldsymbol{\lambda}} \log P(\mathbf{X} = \mathbf{x} \mid \boldsymbol{\lambda}^*, M)$ of the log likelihood of a statistical relational model with respect to the model's parameters as features.*

We will experimentally show that the predictive accuracy of a SRL model can considerably be improved using Fisher kernels and SVMs. As showcase, we will focus on Bayesian logic programs [10] as SRL model but the idea applies naturally to any other SRL.

The outline of the paper is as follows. After discussing related work, we review Fisher Kernels in Section 3. In Section 4, we devise relational Fisher kernels based on Bayesian logic programs. Before concluding, we experimentally evaluate relational Fisher kernels in Section 5.

2 Related Work

Discriminative learning and kernels have only recently started to receive attention within SRL. To the best of our knowledge, [22,24,23,21] are the only ones who aim at discriminative (probabilistic) models for structured data. In contrast to relational Fisher kernels, however, [22] and [21] do not explore kernel functions but gradient-based optimization of the conditional likelihood $\mathbf{P}(y|\mathbf{x})$. Taskar *et al.* [24] present a max-margin algorithm, where the structure in the input/output is modeled by a (relational) Markov network and not by a (relational) Bayesian network. In contrast to all these SRL approaches, relational Fisher kernels are easier to implement because gradient-based optimization techniques are typically already implemented for parameter estimation of SRL models. Recently, Landwehr *et al.* [13] (and related approaches) tightly integrated Naïve Bayes with ILP techniques focusing on discriminative objective functions such as conditional likelihood. The idea has been recently even generalized to learning simple relational kernels [14]. They do not consider fully generative models and no recursive dependencies.

Indeed, there has been a lot of interest in kernels for structured input/output spaces data in the kernel community, see e.g. [6,25] and references in their. For structure input, there are in principle two ways to apply support vector machines to structured data: Using *syntax-driven* and *model-driven* kernel functions.

Syntax-driven kernels *decompose* the input into a set of its parts and the *intersection* of two sets of parts. The kernel on two objects is then defined as a measure of the intersection of the two corresponding sets of parts. In the case that the sets are finite or countable sets of vectors it is often beneficial to sum over all pairwise kernels on the elements. This idea of intersection and cross-product kernels is reflected in most work on kernels for structured data, from the early and influential technical reports [8,28], through work on string kernels, kernels for higher order terms, and tree kernels, to more recent work on graph and relational kernels such as [3,18]. They are not generative models.

An alternative to syntax-driven kernels are *model-driven kernels* like Fisher kernels. For instance [26] introduced the TOP kernel function, which is the scalar product between the posterior log-odds of the model and the gradient thereof. The posterior log-odds is the difference in the logarithm of the probability of each class given the instance. Marginalized kernels [27] have later been introduces as a generalization of Fisher kernels. Here, a kernel over both the hidden and the observed data is assumed. The marginalized kernel for the observed data is obtained by taking the expectation over the hidden variables. One advantage of *model-driven kernels* is their ability to explain the data using the underlying generative models. This is generally not the case for the recently proposed generalizations of the classical maximum-margin formulations to structured input/ouput spaces have been proposed, see e.g. [25] and references in their.

3 Kernel Methods and Probabilistic Models

Support vector machines [20] are one kernel method that can be applied to binary supervised classification problems. Being on one hand theoretically well founded in statistical learning theory, they have on the other hand shown good empirical results in many applications. The characteristic aspect of this class of learning algorithms is the formation of hypotheses by linear combination of positive-definite kernel functions 'centered' at individual training examples. It is known that such functions can be interpreted as the inner product in a Hilbert Space. The solution of the support vector machine is then the hyperplane in this Hilbert space that separates positive and negative labeled examples, and is at the same time maximally distant from the convex hulls of the positive and the negative examples. Conversely, every inner product in a linear space is a positive-definite kernel function.

Fisher kernels are derived from a generative probability model of the domain. More precisely, every learning example is mapped to the gradient of the log likelihood of the generative model with respect to its parameters. The kernel is then the inner product of the examples' images under this map. More precisely, given a parametric probability model M with parameters $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)^\top$, maximum likelihood parameters $\boldsymbol{\lambda}^*$, and output probability $P(\mathbf{X} = \mathbf{x} \mid \boldsymbol{\lambda}, M)$, the Fisher score mapping $U_{\mathbf{x}}$ is defined as $U_{\mathbf{x}} = \nabla_{\boldsymbol{\lambda}} \log P(\mathbf{X} = \mathbf{x} \mid$

$\boldsymbol{\lambda}^*, M) = (\{\partial \log P(\mathbf{X} = \mathbf{x} \mid \boldsymbol{\lambda}^*, M)\}/\partial\lambda_1, \ldots, \{\partial \log P(\mathbf{X} = \mathbf{x} \mid \boldsymbol{\lambda}^*, M)\}/\partial\lambda_n)^\top$ The Fisher information matrix is the expectation of the outer product of the Fisher scores over $P(\mathbf{X} = \mathbf{x} \mid \boldsymbol{\lambda}, M)$, more precisely, $J_{\boldsymbol{\lambda}} = E_{\mathbf{x}} \left[\nabla_{\boldsymbol{\lambda}} \log P(\mathbf{x} \mid \boldsymbol{\lambda}, M)\right] \left[\nabla_{\boldsymbol{\lambda}} \log P(\mathbf{x} \mid \boldsymbol{\lambda}, M)\right]^\top$. Given these definitions, the Fisher kernel is defined as $k(\mathbf{x}, \mathbf{x}') = U_{\mathbf{x}}^\top J_{\boldsymbol{\lambda}^*}^{-1} U_{\mathbf{x}'} =$

$$= \left[\nabla_{\boldsymbol{\lambda}} \log P(\mathbf{X} = \mathbf{x} \mid \boldsymbol{\lambda}^*, M)\right]^\top J_{\boldsymbol{\lambda}^*}^{-1} \left[\nabla_{\boldsymbol{\lambda}} \log P(\mathbf{X} = \mathbf{x}' \mid \boldsymbol{\lambda}^*, M)\right] . \qquad (1)$$

In practice often the role of the Fisher information matrix $J_{\boldsymbol{\lambda}}$ is ignored, yielding the kernel $k(\mathbf{x}, \mathbf{x}') = U_{\mathbf{x}}^\top U_{\mathbf{x}'}$. In the remainder of the paper, we will follow this habit mainly to reduce the computational complexity.

4 Relational Fisher Kernels

To devise Fisher kernels for relational data, Equation (1) tells us that it is sufficient to compute the gradient of the log likelihood of a data case with respect to the parameters $\boldsymbol{\lambda}$ of any SRL model for relational data. This also explains the schematic nature of Definition 2 of relational Fisher kernels: *Any SRL model appropriate for the type of data at hand can be used to implement relational Fisher kernels.* Here, we will focus on Bayesian logic programs as SRL model, which we will briefly review now. For more information we refer to [11].

Bayesian Logic Programs [10,11] (BLPs) integrate definite logic programs with Bayesian networks [19] and specify probability distributions over sets of ground atoms. The key idea is to view ground atoms as random variables, thus atoms describe groups of random variables. As an example, consider the KDD Cup BLP shown in Figure 1. The rule graph gives an overview of all probabilistic dependencies (black boxes) among abstract random variables (ovals). For instance, `interaction/1` is specified in terms of `neighbours/1` and `localization/1`. Each dependency gives rise to a local probabilistic model which is composed of a qualitative and a quantitative part. For instance, clause $C2$ `neighbours(GeneX)` | `neighbour(GeneX, GeneY)`, `localization(GeneY)` in Figure 1 encodes that *"the neighbouring information depends on the localization of a neighbouring gene."* Gradient gray ovals represent abstract random variables such as `localization(GeneY)`, which take values from some domain D(`localization/2`). Smaller white circles on boundaries denote arguments, e.g., some genes `GeneY`. Larger white ovals together with undirected edges indicate that arguments refer to the same gene as for `localization(GeneX)` and `neighbour(GeneX, GeneY)`. To quantify the structural knowledge, *conditional probability distributions* $\mathrm{cpd}(c_i)$ are associated with clauses c_i. They encode the distribution of each possible value of the random variable in the head, given the values of the atoms in the body, i.e., $\mathrm{cpd}(c_i)_{jk} = P(u_j \mid \mathbf{u}_k)$, where $u_j \in \mathrm{D}(\mathrm{head}(c))$ and $\mathbf{u}_j \in \mathrm{D}(\mathrm{body}(c))$. Some information might be of qualitative nature only, such as `neighour(GeneX, GeneY)`. It does not affect the distribution but ensures the variable bindings among `neighbours(GeneX)` and `localization(GeneY)`. Such 'logical' atoms are solid gray ovals. Furthermore,

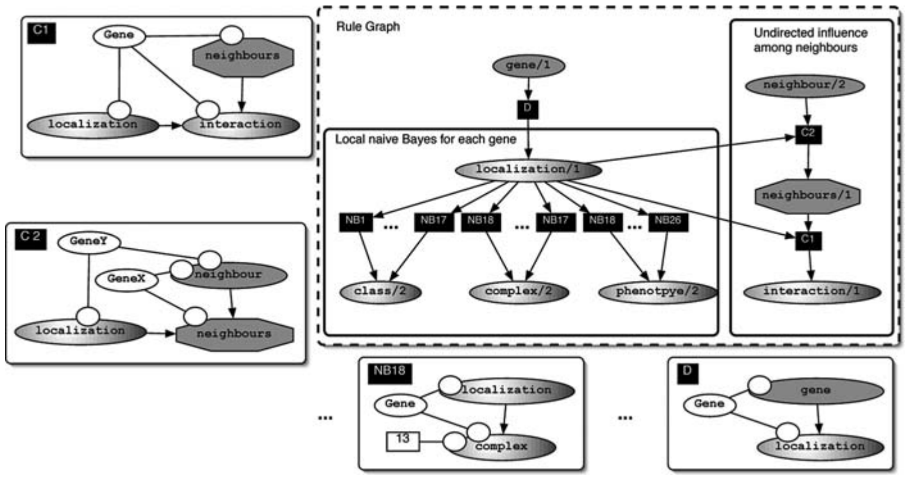


Fig. 1. *Localization* Bayesian logic program. The Bayesian clauses *NB1*, ..., *NB17* encode a Naïve Bayes over local features of `Gene`. Clause D encodes the prior distribution over `localization` for each `Gene`. Clause *C2* aggregates the localizations of all neighbouring genes of `Gene` in `neighbours`(`Gene`). The *mode* of the localizations is used as aggregate function. Clause *C1* implements a mutual influence among `localization`(`Gene`) and the aggregated localization of interacting genes, `neighbours`(`Gene`).

octagonal such as `neighbours`(`GeneX`) denote *aggregation*, i.e., deterministic random variables that summarize the joint state of their parents into a singleton.

The semantics of a BLP M is defined in terms of a Bayesian network. Each ground instance $c\theta$ of a clause c in M, which is entailed by M ($M \models c$), constitute a node head($c\theta$) and its parent body($c\theta$) in the network. The distribution cpd(c) is associated with head($c\theta$) as conditional probability distribution. In case of multiple ground clauses with the same atom in the head, a *combining rule* such as *noisy or* or *mode* is used to combine the cpds associated with the node.

Kersting and De Raedt [10] have shown how to compute the gradient of a BLP w.r.t. a data cases. A data case D is set of (ground atom, state) pairs. The parameter vector $\boldsymbol{\lambda}$ of M consists of all $\mathrm{cpd}(c_i)_{jk}$. Assuming decomposable combining rules, i.e., combining rules, which can be expressed in the structure of the induced Bayesian network (see [10]), the partial derivative of the log-likelihood with respect to a parameter λ of $\boldsymbol{\lambda}$ is

$$\frac{\partial \log P(D|\boldsymbol{\lambda}, M)}{\partial \lambda} = \sum_{\substack{\text{subst. } \theta \text{ with} \\ \text{support}(c_i\theta)}} \frac{P_N(\mathrm{head}(c_i\theta) = u_j, \mathrm{body}(c_i\theta) = \mathbf{u}_k \mid D)}{\mathrm{cpd}(c_i\theta)_{jk}} \tag{2}$$

where P_N denotes the probability distribution of the Bayesian network induced by M for data case D. Note that, in contrast to parameter estimation, we do not reparameterize the Bayesian logic program.

In many cases, it is difficult — if not impossible — to devise a generative Bayesian logic program specifying a probability distribution, which sums up to one over all possible instances, say proteins. For example in our experiments, examples are partly specified within the logical background knowledge. Consequently, their probabilities do not sum up to one and Equation (2) is sensitive to the number of contributing ground clauses. Normalizing (2) with respect to the number of contributing ground clauses, i.e., to compute $\{|\{\theta|\,\mathrm{support}(c_i\theta)\}|\}^{-1} \cdot (\partial \log P(D|\boldsymbol{\lambda}, M)/\partial\lambda)$ worked well in our experiments.

5 Experimental Evaluation

The normalized version of Equation (2) is all we need to devise Fisher kernels for relational data such as the KDD cup 2001 data. In this section, we will experimentally evaluate them. Our intention here is to investigate to which extent relational Fisher kernels are competitive with the generative approach:

Q Do relational Fisher kernels considerably improve the predictive accuracies of their probabilistic baselines with plug-in estimates?

To investigate **Q**, we compare results achieved by Bayesian logic programs alone with results achieved by relational Fisher kernels based on Bayesian logic programs combined with SVMs. The experiments took place in two different domains: protein localization and web page classification. Both data sets are *collective* respectively *networked* data sets (see [16] and references in their), i.e., relational data where individual examples are interconnected, such as web pages (connected through hyperlinks) or gene (connected through interactions). This contrasts with traditional relational domains such as molecules where each individual example is a graph of connected parts. Traditionally, machine learning methods treat examples as independent, i.e., the classification task is treated as a local optimization problem. In contrast, within collective classification tasks, the class membership of one individual may have an influence on the class membership of a related individual. Thus, collective classification induces a global optimization problem.

There is a wide range of possible models that one can apply to the two tasks. We selected a set of models that we felt represented the main idea underlying a variety of collective learners [7,15,16] who globally combine local, propositional Naïve Bayes classifiers. Relational Fisher kernels based on Bayesian logic programs, however, are not designed for collective classification [1]. They assume each individual example as a graph of connected parts. Therefore, we apply the following trick. While learning in a collective way, we consider only individuals together with their direct neighbours at classification time, cf. Figure 2. For any individual without any neighbours, we used a copy of the individual as neighbour. Note that the direct neighbors can come from either the training or test set.

[1] Taking the whole graph at classification time would essentially yield the same feature vector for each individual because the data does not change.

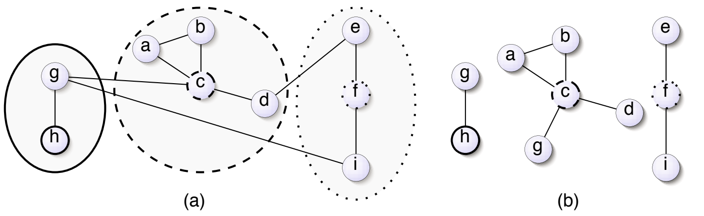


Fig. 2. (a) A collective data set, i.e., a graph of connected individuals each described by a set of local features. **(b)** Data set broken into subgraphs centered around individuals. Each subgraph consists of an individual and all its direct neighbours. Individuals can appear in multiple fragments such as g.

Therefore, their labels are either known or unknown, respectively. This is akin to *iterative* classifiers (see e.g. [16]), which also treat each individual together with all its direct neighbours as a single data case.

We investigated collective Naïve Bayes models and relational Fisher kernels derived from them as described above together with SVMs. We used Weka's [29] using polynomial kernels. To reduce the number of features of the local Naïve Bayes models, we performed WEKA's greedy subset evaluation with default parameters on the training set. That is, we start with an empty feature set for the Naïve Bayes and add one feature on each iteration. If we have added all features or there is no improvement in score of the Naïve Bayes from adding any further features, the search stops and returns the current set of features. To score feature subsets, we used 10-fold cross-validated classification accuracy of the Naïve Bayes on the training set. Finally, both classification tasks are multiclass problems. In order to tackle multiclass problems with SVMs, we followed a round robin approach [5]. That is, each pair of classes is treated as a separate classification problem. The overall classification of an example instance is the majority vote among all pairwise classification problems.

Protein Localization. Reconsider the KDD Cup 2001 localization task of example 1. Figure 1 shows the Bayesian logic program used in the experiments. We listed the genes as ground atoms over `gene`/1 in the logical background knowledge. They were used to encode the prior localization, cf. Bayesian clause D. The feature selection yielded 26 features for the local Naïve Bayes describing the genes, which we encoded as Bayesian clauses *NB1*, …, *NB26*. So far, the Bayesian logic program encodes the simple, non-collective Naïve Bayes model we used in the experiments. To model the collective nature of the data set, we enriched the Naïve Bayes model as follows. We encoded each interaction as a logical ground atom over `d_neighbour`/2, i.e., we omitted the originally given quantification of the interactions. Because interactions are bidirectional, i.e., undirected, we additionally defined the symmetric closure `neighbour(A, B) :- d_neighbour(A, B);d_neighbour(B, A)` (where ';' denotes a logical or) as logical background. The localizations of the direct neighbours of a `Gene` are aggregated in clause *C2* into a single value `neighbours(Gene)` using the *mode* of the interactions. To establish a mutual influence among the localizations

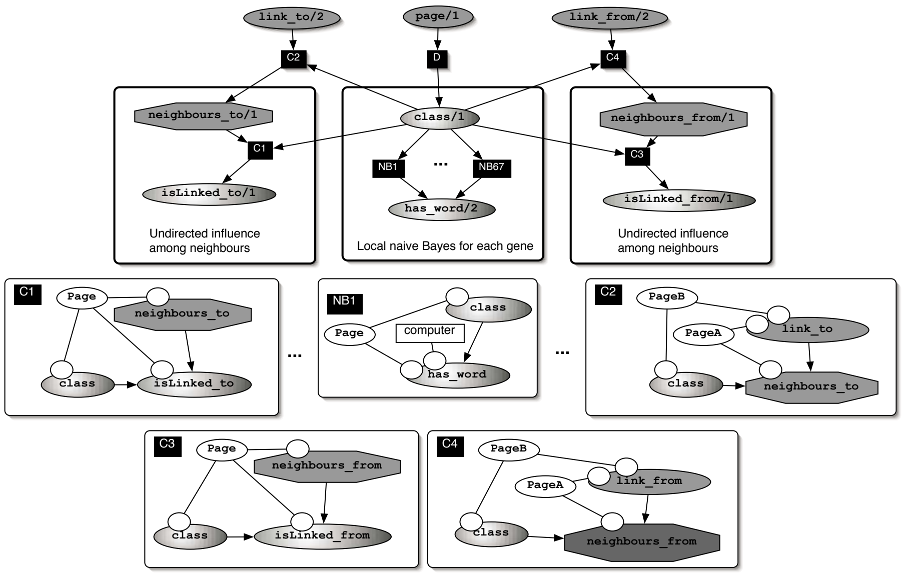


Fig. 3. *WebKB* Bayesian logic program. The Bayesian clauses *NB1*, ..., *NB67* encode a Naïve Bayes over local features of genes web pages, page(Page). Clause *D* encodes the prior distribution over class for each Page. Clause *C2* aggregates the class memberships of all web pages to which Page provides a link. Clause *C4* aggregates the class memberships of all web pages, which link to Page. In both cases, the *mode* of the class memberships is used as aggregate function. Clauses *C1* and *C3* implement a mutual influence among class(Page) and the aggregated class memberships of linked pages.

of a gene and its neighbours, we introduced a boolean random variable interaction(Gene), which has neighbours(Gene) and localization(Gene) as parents, cf. clause *C1*. Setting the evidence for interaction(Gene) always to be true guarantees that both parents are never d-separated, hence, they are probabilistic dependent.

On the test set, the relational Fisher kernel achieved an accuracy of 72.89%, whereas the collective Naïve Bayes only achieved 61.66%, and outperformed Hayashi et al.'s KDD Cup 2001 winning nearest-neighbour approach [1] that achieved a test set accuracy of 72.18%. This affirmatively answers **Q**.

Web Page Classification. This dataset is based on the WebKB Project [2]. It consists of sets of web pages from four CS departments, with each page manually labeled into 7 categories: course, department, faculty, project, staff, student or others. We excluded pages in the 'other' category from consideration and put them into the background knowledge. This yielded a multiclass problem with 6 different classes, 877 web pages, and 1516 links among the web pages.

Figure 3 shows the Bayesian logic program used in the experiments. It essentially follows the idea underlying the Bayesian logic program for the localization task, cf. Figure 1. The feature selection yielded 67 local for the local Naïve Bayes

Table 1. Leave-one-university-out accuracies on the WebKB data. Both collective classifiers used the same Bayesian logic program. The mean difference between collective Naïve Bayes and relational Fisher kernel in test accuracy was 12.94%.

	Cornell	Texas	Washington	Wisconsin	**Mean**
Collective Naïve Bayes	63, 44	59, 20	58, 65	68, 07	62, 34
Relational Fisher Kernel	71, 08	73, 53	71, 93	84, 59	75, 28

model (clauses *NB1*, . . . , *NB67*. Whereas gene interaction is undirected, links among web pages are directed. There are *incoming* and *outcoming* links on a web page. We modeled their influences on the class membership of a web pages separately. The atom `neighbours_from(Page)` (respectively `neighbours_to(Page)`) aggregates the class memberships of all pages that have a link to `Page` (respectively that `Page` links to) using *mode* as aggregate function. Again, we took care that `class(Page)` and the aggregated class memberships of linked pages mutually influence each other, i.e., we introduced `isLinked_from(Page)` and `isLinked_to(page)`, whose evidence is always `yes`.

We performed a leave-one-university-out cross-validation. The experimental results are summarized in Table 1. The Fisher kernels achieved an accuracy of 75.28%, which is significantly higher (two-tailed t-test, $p = 0.05$) than the collective Naïve Bayes' accuracy of 62.34%. For comparison, the performance of the collective Naïve Bayes is in the range of Getoor *et al.*'s [7] probabilistic relational model with link anchor words. The Fisher kernel outperforms the probabilistic relational model with the best predictive accuracy Getoor *et al.* report on. It takes structural uncertainty over the link relationship of web pages into account and achieved with 68% its highest accuracy on the Washington hold-out set. Thus, **Q** is again affirmatively answered.

6 Conclusions and Future Work

In this paper, Fisher kernels for relational data have been introduced and experimentally investigated. They are 'off-the-shelf' kernels and are easy to implement for any SRL model. The experimental results show that Fisher kernels can handle relational data and can indeed significantly improve the predictive performance of their underlying probabilistic model: the WebKB model is outperformed by an advanced probabilistic relational model, which in turn was outperformed by our Fisher kernel; the probabilistic KDD Cup model ranks only around the top 50% level of submitted models (61% accuracy) whereas the corresponding Fisher kernel performs better than the KDD Cup 2001 winning approach.

The research on the intersection of kernel, discriminative, and relational learning has just started, and relational Fisher kernels are only a further step into this direction. There is a lot of space for future research: other learning tasks such as regression, clustering, and ranking should investigated. In general, choosing the appropriate kernel is the major step for the application of kernel method and should take as much domain knowledge into account as possible. To this aim, knowledge-based SVMs [4] have been for instance proposed, which find in addition to a large margin solution an estimate statisfying constraints encoding prior knoweldge in terms of polyhedral sets. As [3] point out, real-valued functions are inappropriate as a general knowledge representation language; they suffer from a non-declarative nature. Statistical relational languages are a natural alternative and an attractive way to embed knowledge into statistical learning algorithms in a principled and flexible way.

Acknowledgments. The authors thank Luc De Raedt for his support. The research was supported by the European Union IST programme, contract no. FP6-508861, *Application of Probabilistic ILP II.*

References

1. J. Cheng, C. Hatzis, M.–A. Krogel, S. Morishita, D. Page, and J. Sese. KDD Cup 2001 Report. *SIGKDD Explorations*, 3(2):47 – 64, 2002.
2. M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. M. Mitchell, K. Nigam, and S. Slattery. Learning to Construct Knowledge Bases from the World Wide Web. *Artificial Intelligence Journal*, 118(1–2):69–113, 2000.
3. P. Frasconi, A. Passerini, S. H. Muggleton, and H. Lodhi. Declarative kernels. Submitted, 2005.
4. G. Fung, O. Mangasaruan, and J. Shavlik. Knowledge-based Support Vector Machine Classifier. In *Advances in Neural Information Processing Systems 15*, 2002.
5. J. Fürnkranz. Round Robin Classification. *Journal of Machine Learning Research (JMLR)*, 2:721–747, 2002.
6. T. Gärtner. Kernel-based Learning in Multi-Relational Data Mining. *ACM-SIGKDD Explorations*, 5(1):49–58, 2003.
7. L. Getoor, N.Friedman, D. Koller, and B Taskar. Learning Probabilistic Models of Link Structure. *Journal of Machine Leaning Research (JMLR)*, 3:679 – 707, 2002.
8. D. Haussler. Convolution kernels on discrete structures. Technical report, Department of Computer Science, University of California at Santa Cruz, 1999.
9. T. Jaakkola and D. Haussler. Exploiting Generative Models in Discriminative Classifiers. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 487–493, 1999.
10. K. Kersting and L. De Raedt. Adaptive Bayesian Logic Programs. In C. Rouveirol and M. Sebag, editors, *Proceedings of the 11th International Conference on Inductive Logic Programming (ILP-01)*, volume 2157 of *LNAI*, pages 118–131, Strasbourg, France, September 2001. Springer.
11. K. Kersting and L. De Raedt. Bayesian Logic Programming: Theory and Tool. In L. Getoor and B. Taskar, editors, *An Introduction to Statistical Relational Learning*. MIT Press, 2006. (to appear).
12. K. Kersting and T. Gärtner. Fisher Kernels for Logical Sequences. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Proceedings of the 15th European Conference on Machine Learning (ECML-04)*, volume 3201 of *LNCS*, pages 205 – 216, Pisa, Italy, 2004.
13. N. Landwehr, K. Kersting, and L. De Raedt. nFOIL: Integrating Naïve Bayes and Foil. In M. Veloso and S. Kambhampati, editors, *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 795–800, Pittsburgh, Pennsylvania, USA, July 9–13 2005. AAAI.

14. N. Landwehr, A. Passerini, L. De Raedt, and P. Frasconi. kFOIL: Learning Simple Relational Kernels. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*. AAAI, 2006. (To appear).
15. Q. Lu and L. Getoor. Link-based Classification. In T. Fawcett and N. Mishra, editors, *Proceedings of the International Conference on Machine Learning (ICML-03)*, pages 496–503, Washington, DC USA, August 21-24 2003.
16. S. A. Macskassy and F. Provost. Classification in Networked Data: A toolkit and a univariate case study. Technical Report CeDER-04-08, CeDER Working Paper, Stern School of Business, New York University, New York, USA, 2004.
17. S. H Muggleton and L. De Raedt. Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming*, 19(20):629–679, 1994.
18. A. Passerini, P. Frasconi, and L. De Raedt. Kernels on Prolog Proof Trees: Statistical Learning in the ILP Setting. *JMLR*, 7:307–342, 2006.
19. J. Pearl. *Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 2. edition, 1991.
20. B. Schölkopf and A.J. Smola. *Learning with Kernels.* MIT Press, 2002.
21. P. Singla and P. Domingos. Discriminative training of markov logic networks. In M. Veloso and S. Kambhampati, editors, *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 868–873, Pittsburgh, Pennsylvania, USA, July 9–13 2005. AAAI Press.
22. B. Taskar, P. Abbeel, and D. Koller. Discriminative Probabilistic Models for Relational Data. In A. Darwiche and N. Friedman, editors, *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 485–492, Edmonton, Alberta, Canada, August 1-4 2002.
23. B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning Structured Prediction Models: A Large Margin Approach. In L. De Raedt and S. Wrobel, editors, *Proceedings of the Twenty Second International Conference on Machine Learning (ICML-05)*, pages 897–902, Bonn, Germany, August 7-10 2005.
24. B. Taskar, C. Guestrin, and D. Koller. Max-Margin Networks. In *Advances in Neural Information Processing Systems 16*, 2004.
25. I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, 2005.
26. K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.-R. Müller. A new discriminative kernel from probabilistic models. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 977–984. The MIT Press, 2002.
27. K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 2002.
28. C. Watkins. Kernels from matching operations. Technical report, Department of Computer Science, Royal Holloway, University of London, 1999.
29. I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, San Francisco, 2nd edition, 2005.

Evaluating Misclassifications in Imbalanced Data

William Elazmeh[1], Nathalie Japkowicz[1], and Stan Matwin[1,2]

[1] School of Information Technology and Engineering
University of Ottawa, K1N 6N5 Canada
{welazmeh, nat, stan}@site.uottawa.ca
[2] The Institute of Computer Science, Polish Academy of Sciences, Poland

Abstract. Evaluating classifier performance with ROC curves is popular in the machine learning community. To date, the only method to assess confidence of ROC curves is to construct ROC bands. In the case of severe class imbalance with few instances of the minority class, ROC bands become unreliable. We propose a generic framework for classifier evaluation to identify a segment of an ROC curve in which misclassifications are balanced. Confidence is measured by Tango's 95%-confidence interval for the difference in misclassification in both classes. We test our method with severe class imbalance in a two-class problem. Our evaluation favors classifiers with low numbers of misclassifications in both classes. Our results show that the proposed evaluation method is more confident than ROC bands.

1 Motivation

Recently, the machine learning community has increased the focus on classifier evaluation. Evaluation schemes that compute accuracy, precision, recall, or F-score have been shown to be insufficient or inappropriate [1,2]. Furthermore, the usefulness of advanced evaluation measures, like ROC curves [3,2,4] and cost curves [5,6], deteriorates in the presence of a limited number of positive examples. The need for confidence in classifier evaluation in machine learning has lead to the construction of ROC confidence bands. Methods in [7,8] construct ROC bands by computing confidence intervals for points along the ROC curve. These methods are either parametric (making assumptions of data distributions), or non-parametric and rely on carefully crafted sampling methods. When faced with severe class imbalance and with a limited number of instances in the minority class, sampling methods become unreliable, especially when the data distribution is unknown [8]. In fact, with severe imbalance, the entire issue of evaluation becomes a serious challenge even when making assumptions of data distributions [9]. In contrast, biostatistical and medical domains impose strong emphasis on error estimates, interpretability of prediction schemes, scientific significance, and confidence [10] whilst machine learning evaluation measures fail to provide such guarantees. Consequently, the usefulness of some machine learning algorithms remains inadequately documented and unconvincingly demonstrated. Thus, despite their interest in using learning algorithms, biostatisticians remain

Table 1. The statistical proportions in a confusion matrix

	Predicted +	Predicted -	total
Class +	a (q_{11})	b (q_{12})	a+b
Class -	c (q_{21})	d (q_{22})	c+d
total	a+c	b+d	n

skeptical of their evaluation methods and continue to develop customized statistical tests to measure characteristics of interest. Our work adopts Tango's test [11] from biostatistics in an attempt to provide confidence in classifier evaluation. Tango's test is a non-parametric confidence test designed to measure the difference in binomial proportions in paired data. This test is shown in [12] to be reliable and robust with power and coverage probability to produce confidence and significance.

Computing the confidence based on the positive or negative rates (using a or d of table 1) can be influenced by class imbalance in favor of the majority class. Alternatively, applying a statistical significance test to those entries (b or c) that resist such influence may provide a solution. Hence, to counter the class imbalance, particularly when the number of instances in the minority class is very small, we use Tango to favor classifiers with similar normalized number of errors in both classes, rather than similar error rates. This solution assumes that the classifier performs reasonably well, in the sense that, it can at least classify the majority class with high accuracy. Therefore, a large portion of instances in the majority class is correctly classified, and the imbalance has no influence on the error values (b or c). Consequently, any evaluation measure that employs rates (false positive or false negative), such as ROC curves, is influenced by data imbalance, while the error analysis we propose is not. Our approach is based on measuring just the error of classification, and therefore, to capture a fuller evaluation of the classifier, we need to combine Tango's analysis together with another evaluation measure that measures how well the classifier performs. As we measure negatively, we need to use our approach along with another measure (eg. ROC) to evaluate positively.

In this paper; (1) we propose a framework for classifier evaluation that identifies confident points along an ROC curve using a statistical confidence test. These points form a balanced misclassification segment on the ROC curve to which we recommend restricting the evaluation. (2) Although our framework can be applied to any data, this work focuses on the presence of severe imbalance (with a very small number of instances in the minority class) where ROC bands, ROC curves and AUC struggle to produce meaningful assessments. (3) We produce a representation of classifier performance based on the average difference in misclassifications and the area under the balanced misclassification segment of the ROC curve. We present experimental results that show the effectiveness of our approach compared to ROC bands, ROC curves, and AUC.

Having motivated this work, subsequent sections present discussions of classification error proportions in both classes, our evaluation framework, and our

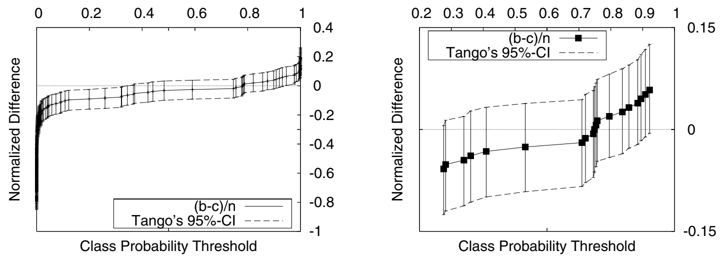


Fig. 1. $\frac{b-c}{n}$ and Tango's 95%-confidence intervals for classification points. Left: all classification points. Right: only points whose Tango's intervals contain 0 difference.

experimental results followed by conclusions and future work. In the appendix, we briefly describe Tango's statistical test of confidence.

2 The Difference in Classification Errors

Common classifier performance measures in machine learning estimate classification accuracy and/or errors. ROC curves provide a visualization of a possible trade-off between accuracy and error rates for a particular class. For the confusion matrix presented in table 1, the ROC curve for the class + plots the true positive rate $\frac{a}{a+b}$ against the false positive rate $\frac{c}{c+d}$. When the number of positive examples is small and is significantly lower than the number of negative examples, the row totals $a + b << c + d$. When changing the class probability threshold, the rate of change in the true positive rate climbs faster with each example than that of the false positives (due to using c and d). This inconsistent rate of change gives the majority class $(-)$ a clear advantage in the rates calculated for the ROC curve. Ideally, a classifier classifies both classes proportionally, but due to the severe imbalance along with a small number of instances of the minority class, comparing the rates of accuracy and/or errors on both classes does not evaluate proportionally. We propose to favor the classifier that performs with similar number of errors in both classes to eliminate the use of the number of correctly classified examples (a and d) in the evaluation to avoid a large portion of examples in the majority class. In fact, our approach favors classifiers that have lower difference in misclassifications in both classes, $\frac{b-c}{n}$. Furthermore, we normalize entries in the confusion matrix by dividing by the number of examples n so the difference $\frac{b-c}{n}$ remains within $[-1, +1]$.

ROC curves are generated by classifying examples while increasing class probability threshold T. When $T = 0$, all data examples are classified as +, thus, $a = |+|$ (the number of positives), $b = 0$, $c = |-|$, $d = 0$, and $\frac{b-c}{n} \in [-1, 0]$. Similarly, for $T = 1$, all examples are classified as $-$, then, $a = 0$, $b = |+|$, $c = 0$, $d = |-|$, and $\frac{b-c}{n} \in [0, +1]$. In fact, these two extreme negative and positive values of $\frac{b-c}{n}$ depend on class distributions in the data. Within these two extremes, $\frac{b-c}{n}$ exhibits a monotone behavior as the threshold varies from 0

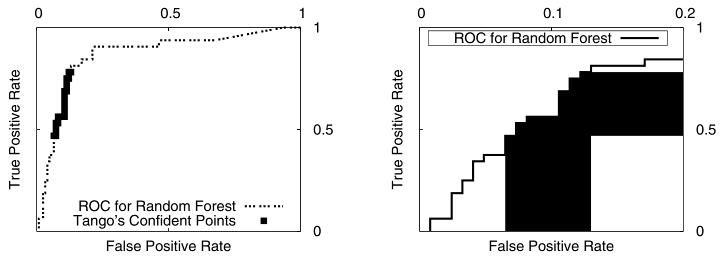


Fig. 2. On the left is a sample balanced misclassification segment and on the right is the area under this segment

to 1. This is illustrated in figure 1. For each threshold value $T := 0$ to 1, the classification produces a confusion matrix a, b, c, d. Initially, a and c are at their maximum values, while b and d are 0. As T increases, examples are classified in any combination of three possibilities; (1) c decreases when false positives become correctly classified, (2) b increases when true positives become misclassified, (3) or, b and c remain unchanged because examples are correctly classified. Since c never increases, b never decreases, and n is constant, then $\frac{b-c}{n}$ exhibits a monotone non-decreasing behavior for a classifier on a set of data. Our evaluation method computes Tango's 95%-confidence intervals for $\frac{b-c}{n}$ for ROC points. Those points whose confidence intervals include the value zero, show no evidence of statistically significant $\frac{b-c}{n}$ and are considered confident. This is explained in more details in the next section. In addition, Tango's test is presented in [11] and is reviewed in the appendix of this paper.

3 The Proposed Method of Evaluation

The proposed evaluation method consists of four steps: **(1)** Generate an *ROC* curve for a classifier K applied on test examples D with increasing class probability thresholds t_i (0 to 1). **(2)** For each resulting point (a confusion matrix along the ROC curve), apply Tango's test to compute the 95%-confidence interval $[u_i, l_i]$, within which lies the point of the observed normalized error difference $\frac{b_i-c_i}{n}$. If $0 \in [u_i, l_i]$, then this point is identified as a confident point and is added into the set of points S. Points in S form the ROC segment illustrated in the left plot of figure 2: we call it the balanced misclassification segment. This framework is generic and accommodates a test of choice provided that it produces a meaningful interpretation of results. **(3)** Compute *SAUC* the area under the segment S as shown in the right plot of figure 2. **(4)** Compute *AveD* the average normalized difference $(\frac{b-c}{n})$ for all points in S. In our experiments, we plot the area under the balanced misclassification segment (*SAUC*) against the average observed misclassification difference (*AveD*). Lower *AveD* values suggest lower misclassification difference and higher *SAUC* values indicate larger balanced misclassification segment. An effective classifier shows low *AveD* and high *SAUC*.

Table 2. UCI data sets [13] and their class distributions $|+|/|-|$

	`dis`	`hypothyroid`	`sick`	`sick-euthyroid`	`SPECT`	`SPECTF`
training	45/2755	151/3012	171/2755	293/2870	40/40	40/40
testing	13/959	–	13/959	–	15/172	55/214

4 Experiments

Having presented our evaluation framework, we now present an overview of our experiments and their data sets followed by an assessment of results to motivate conclusions. The data sets, listed in table 2, are selected from the UCI-Machine Learning repository [13] and consist of examples of two-class problems. They are severely imbalanced with the number of positive examples reaching as low as 1.4% (`dis`) and not exceeding 26% (`spectf`). Only (`spect`) and (`spectf`) data sets have a balanced training set and imbalanced testing set. On these data sets, we train four classifiers and compare their performances as reported by the ROC, by the AUC, and by our method. If testing data sets are unavailable, we use cross-validation of 10 folds. Using `Weka 3.4.6` [14], we build a decision stump classifier without boosting (`S`), a decision tree (`T`), a random forest (`F`), and a Naive Bayes (`B`) classifier. The rationale is to build classifiers for which we can expect a ranking of performance. A decision stump built without boosting is a decision tree with one test at the root (only 2 leave nodes) and is expected to perform significantly worse than a decision tree. Relatively, a decision tree is a stronger classifier since it is more developed and has more leave nodes that cover the training examples. The random forest classifier is a reliable classifier and is expected to outperform a single decision tree. Finally, the naive Bayes classifier tends to minimize classification error and is expected to perform reasonably well when trained on a balanced training set.

We first investigate the usefulness of ROC confidence bands on data with imbalance. Figure 3 shows the ROC confidence bands for our four classifiers on the most imbalanced `dis` data set. These bands are generated using the empirical fixed-width method [8] at the 95% level of confidence. We claim that with severe imbalance and a very small minority class, sampling-based techniques do not work. Clearly, the generated bands are very wide and contain more than 50% of the ROC space proving that they are not very useful. This result is also consistent on the other data sets. Given this failure of the ROC bands, we propose to use Tango's test which is designed to accommodate a small size of proportions (the minority class) while resisting the influence of class imbalance. As shown in the Appendix, Tango test is related to McNemar test which has been used to rank performance of classifiers in [15]. In addition, Newcombe in [12] (which compares 10 different statistical methods that Tango is based on) shows that Tango's confidence intervals are robust (they do not collapse in boundary conditions and do not produce tethering points) and reliable with good probability coverage.

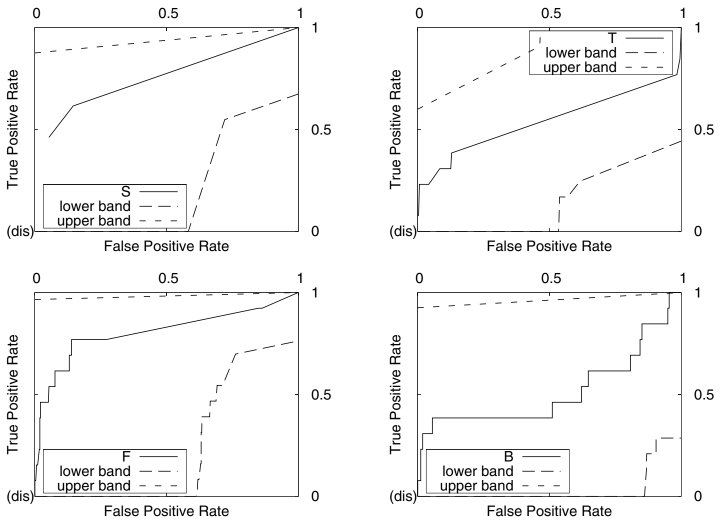


Fig. 3. ROC confidence bands for decision stump (S), decision tree (T), random forest (F), and naive Bayes (B) on (`dis`) data set. The bands are wide and are not very useful.

Next, we consider the ROC curves of our four classifiers on all data sets shown in figure 4. Recall, ROC curves are compared by being more dominantly placed towards the north-west of the plot (higher true positive rate and lower false positive rate). We observe that the decision stump (S) performs the same or better than the decision tree (T) on all data sets. In addition, the random forest (F), consistently, outperforms the naive Bayes (B). In fact, (F) shows the best performance on most data sets. When we consider the AUC values of these classifiers, shown in table 3, (S) has similar or higher AUC values than (T). Furthermore, the AUC of (F) is, clearly, higher than that of the others on most data sets (the bold numbers in table 3). When trained on a balanced data set (`SPECT`), (F) and (B) classifiers perform significantly better than the others.

In contrast, the results obtained by our proposed evaluation measure are presented in figure 5. Each of the six plots in the figure reports our evaluation of the four classifiers on each data set. The x-axis represents the average normalized misclassification difference $\frac{b-c}{n}$ for those points on the balanced misclassification segment of the ROC curve. The y-axis represents the area under this segment. Classifiers placed towards the top-left corner perform better (bigger area under the balanced misclassification segment and less difference in classification error) than those placed closer to the bottom right corner (smaller area and higher difference in misclassifications). Classifiers that fail to produce confident points on their ROC curves are excluded from the plots. The decision stump (S) fails to produce confident points along its ROC, therefore, it does not appear in any of the plots in the top row of figure 5. This is consistent with our expectation of it being less effective. In fact, plots in the bottom row of the same figure show that (S) also performs poorly producing higher misclassification difference.

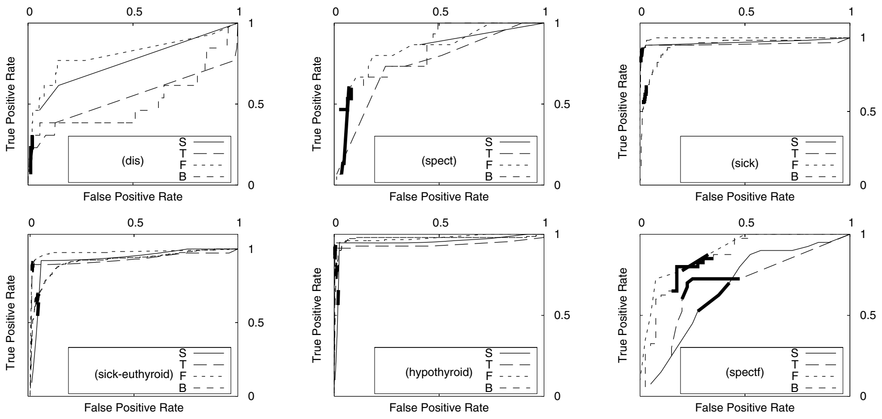


Fig. 4. ROC curves for decision stump (S), decision tree (T), random forest (F), and naive Bayes (B) on all data set. The dark segments are Tango's confident points.

Table 3. AUC values for classifiers (S), (T), (F), and (B) on our data sets

Classifier	dis	hypothyroid	sick	sick-euthyroid	spect	spectf
S	0.7517	0.9491	0.9523	0.9312	0.7298	0.6744
T	0.5408	0.9360	0.9559	0.9296	0.7453	0.6900
F	**0.8051**	**0.9784**	**0.9966**	**0.9777**	0.8326	**0.8925**
B	0.5164	0.9720	0.9460	0.9215	**0.8347**	0.8575

In fact, even when (S) has slightly higher SAUC than (T), in the bottom left and middle plots of figure 5, (S) still shows a significantly higher difference in misclassification than that of (T). The tree (T), on the other hand, performs well in most cases particularly in the top and bottom right plots of figure 5. (T) certainly outperforms the (S) which contradicts observations based on the ROCs and AUCs. Furthermore, (T) fails to produce confident points on the (`spect`) data set (top middle plot of the same figure). Perhaps, since (`spect`) is a binary data set extracted from the continuous (`spectf`) set, this may suggest that the extraction process hinders the decision tree learning. (F) and (B) classifiers appear reasonably consistent on all data sets with (B) being particularly strong on the (`dis`) data set. However, the surprise is (B) showing significantly higher SAUC than (F) in the top and bottom right plots of figure 5.

Our results, clearly, contradict conclusions based on the ROC and AUC evaluations. Therefore, we investigate those points along the balanced misclassification segment for two situations. First, when the four classifiers are trained and tested on imbalanced `dis` data sets, and second, when the four classifiers are trained on a balanced training set and are tested on an imbalanced testing `SPECTF` data set. For the first situation (`dis` data sets), the ROC curves reveal that three

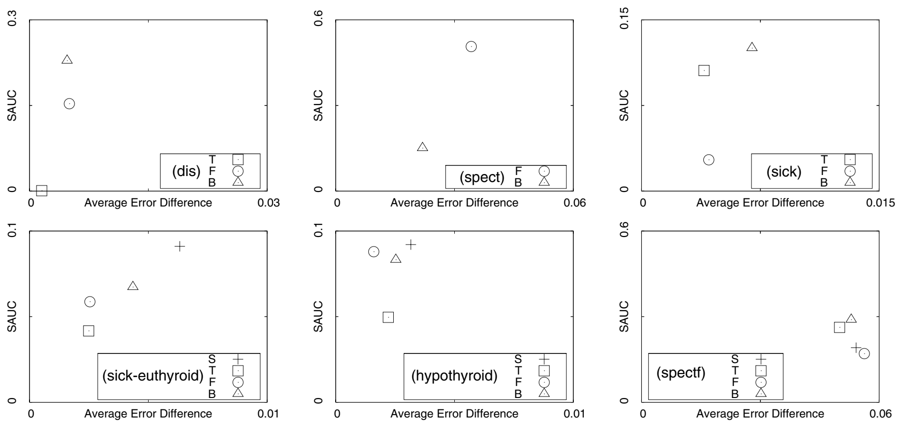


Fig. 5. Our evaluation for decision stump (S), decision tree (T), random forest (F), and naive Bayes (B) on our data sets. The y-axis shows the area under the balanced misclassification segment (SAUC) and the x-axis shows the average observed normalized misclassification difference $\frac{b-c}{n}$.

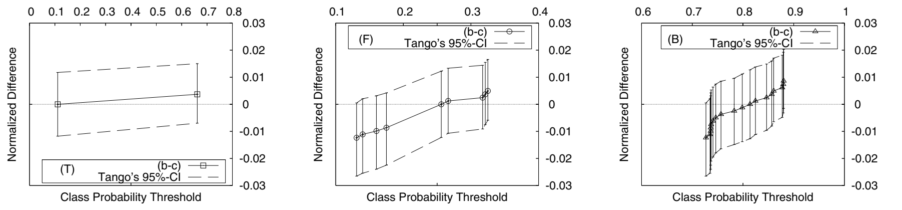


Fig. 6. Tango's 95%-confidence intervals for classification points for decision tree (T), random forest (F), and naive Bayes (B) on (`dis`) set. The center points are ($\frac{b-c}{n}$). (B) produces the most points that satisfy the Tango test.

of the classifiers produce balanced misclassification points in the bottom left section of the ROC space (see the bold segments in the top left plot of figure 4). These points are detected by our method at the 95% level of confidence and are consistent with having severely imbalanced data sets with very few positive examples. When we consider the corresponding Tangos 95%-confidence intervals for these classifier (see figure 6), we see that (T) produces few points (only two) which cover a wider range of probability threshold (0.1 to 0.65 on the x-axis of the left plot). (T) produces only two points which may be due to the very low number of positive examples. Alternatively, despite generating many more confident points, (F) and (B) classifiers show higher variations of misclassification difference for a much narrower range of thresholds values. The top left plot of figure 5 shows that (B) and (F) have a higher SAUC values than (T) which has a significantly lower misclassification difference. At the least, this indicates a distinction between these classifiers.

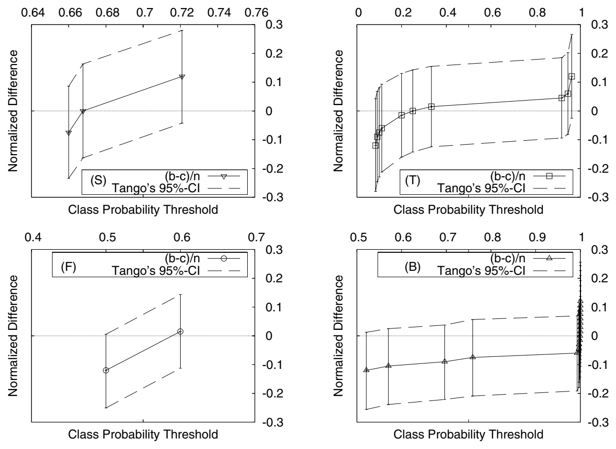


Fig. 7. Tango's 95%-confidence intervals for classification points for decision stump (S), decision tree (T), random forest (F), and naive Bayes (B) on (`spectf`) set. The center points are ($\frac{b-c}{n}$). (T) and (B) produce more points that satisfy Tango's test.

For the second situation (`SPECTF` data sets), the ROC curves in the bottom right plot of figure 4 show that both (F) and (B) dominate (T) and (S). Our method in the bottom right plot of figure 5 suggest that both (T) and (B) outperform (F) and (S). Tango's 95%-confidence intervals of their classification points (shown in figure 7) show that (T) and (B) produce the most number of points on their balanced misclassification segment with low misclassification difference. Also in the same figure, (T) and (B) produce classification points that have exactly zero misclassification difference while the other two come close to the zero misclassification difference.

5 Conclusions and Future Work

We propose a method to address classifier evaluation in the presence of severe class imbalance with significantly fewer positive examples. In this case, our experiments show that ROC confidence bands fail to provide meaningful results. We propose a notion of statistical confidence by using a statistical tests, borrowed from biostatistics, to compute the 95%-confidence intervals on the difference in misclassification. This work presents error-based analysis (using Tango's test) which aims to balance misclassifications. To capture a fuller evaluation of the classifier, we need to combine Tango's analysis together with another evaluation measure (eg. ROC) that measures how well the classifier performs. Our method plots of the trade-off between misclassification difference and area under the balanced misclassification segment of the ROC curve. Our experiments show that our method is more reliable than general ROC and AUC measures.

In the future, it can be useful to compute confidence bands or intervals for these proposed confident ROC segments. This remains a difficult task because the confidence in our method is computed on the misclassification difference which may not map easily to the ROC space. We plan to investigate the feasibility of mapping the confidence intervals from this work into the ROC space. This may be interesting particularly when there is no danger of imbalance. Although this work addresses the case of severe imbalance in the data, Tango's test of confidence can still be applied to balanced data sets. We plan to explore our framework in balanced situations with the aim to drive useful and meaningful evaluation metrics to provide confidence and reliability.

Acknowledgments

The authors thank James D. Malley at the National Institute of Health for his suggestions and communications. In addition, we acknowledge the support of The Natural Sciences and Engineering Research Council of Canada (NSERC) and of the Ontario Centers of Excellence (OCE).

References

1. Ling, C.X., Huang, J., Zang, H.: Auc: a better measure than accuracy in comparing learning algorithms. Canadian Conference on AI (2003) 329–341
2. Provost, F., Fawcett, T.: Analysis and visualization f classifier performance: Comparison under imprecise class and cost distributions. the Third International Conference on Knowledge Discovery and Data Mining (1997) 34–48
3. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. Journal of Artificial Intelligence Research (10) (1999) 243–270
4. Swets, J.: Measuring the accuracy of diagnostic systems. Science (240) (1988) 1285–1293
5. Drummond, C., Holte, R.C.: Explicitly representing expected cost: An alternative to roc representation. the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2000) 198–207
6. Drummond, C., Holte, R.C.: What roc curves can't do (and cost curves can). ECAI'2004 Workshop on ROC Analysis in AI (2004)
7. Macskassy, S.A., Provost, F., Rosset, S.: Roc confidence bands: An empirical evaluation. in Proceedings of the 22nd International Conference on Machine Learning (ICML 2005) (2005) 537 – 544
8. Macskassy, S.A., Provost, F.: Confidence bands for roc curves: Methods and empirical study. in Proceedings of the 1st Workshop on ROC Analasis in AI (ROCAI-2004) at ECAI-2004 (2004)
9. Drummond, C., Holte, R.C.: Severe class imbalance: Why better algorithms aren't the answer. Proceedings of the 16th European Conference of Machine Learning (2005) 539–546
10. Motulsky, H.: Intuitive Biostatistics. Oxford University Press, New York (1995)
11. Tango, T.: Equivalence test and confidence interval for the difference in proportions for the paired-sample design. Statistics in Medicine **17** (1998) 891–908
12. Newcombe, R.G.: Improved confidence intervals for the difference between binomial proportions based on paired data. Statistics in Medicine **17** (1998) 2635–2650

13. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html (1998) University of California, Irvine, Dept. of Information and Computer Sciences.
14. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. 2nd Edition, Morgan Kaufmann (2005)
15. Dietterich, T.G.: Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation **10**(7) (1998) 1895–1923
16. Newcombe, R.G.: Two-sided confidence intervals for the single proportion: comparison of seven methods. Statistics in Medicine **17** (1998) 857–872
17. Everitt, B.S.: The analysis of contingency tables. Chapman-Hall (1992)

Appendix A: Tango's Confidence Intervals

Clinical trials, case-control studies, and sensitivity comparisons of two laboratory tests are examples of medical studies that deal with the difference of two proportions in a paired design. Tango's test [11] builds a model to derive a one-sided test for equivalence of two proportions. Medical equivalence is defined as no more than 100Δ percent inferior, where $\Delta (> 0)$ is a pre-specified acceptable difference. Tango's test also derives a score-based confidence interval for the difference of binomial proportions in paired data. Statisticians have long been concerned with the limitations of hypothesis testing used to summarize data [16]. Medical statisticians prefer the use of confidence intervals rather than p-values to present results. Confidence intervals have the advantage of being close to the data and on the same scale of measurement, whereas p-values are a probabilistic abstraction. Confidence intervals are usually interpreted as margin of errors because they provide magnitude and precision. A method deriving confidence intervals must be a priori reasonable (justified derivation and coverage probability) with respect to the data [16].

The McNemar test is introduced in [17] and has been used to rank the performance of classifiers in [15]. Although inconclusive, the study showed that the McNemar test has low Type I error with high power (the ability to detect algorithm differences when they do exist). For algorithms that can be executed only once, the McNemar test is the only test that produced an acceptable Type I error [15]. Despite Tango's test being an equivalence test, setting the minimum acceptable difference Δ to zero produces an identical test to the McNemar test with strong power and coverage probability [11]. In this work, we use Tango's test to compute confidence intervals on the difference in misclassifications in both classes with a minimum acceptable difference $\Delta = 0$ at the (1-α) confidence level. Tango makes few assumptions; (1) the data points are representative of the class. (2) The predictions are reasonably correlated with class labels. This means that the misclassified positives and negatives are relatively smaller than the correctly classified positives and negatives respectively. In other words, the classifier does reasonable well on both classes, rather than performing a random classification. We consider classifier predictions and class labels as paired machines that fit the matched paired design. As shown in table 1 on page 127, entries a and d are the informative or the discordant pairs indicating the agreement portion $(q_{11} + q_{22})$, while b and c are the uninformative or concordant pairs representing the proportion of disagreement $(q_{12} + q_{21})$ [12]. The magnitude of the difference δ in misclassifications can be measured by testing the null hypothesis $H_0 : \delta = q_{12} - q_{21} = 0$. This magnitude is conditional on the observed split of b and c [12]. The null hypothesis H_0 is tested against the alternative $H_1 : \delta \neq 0$. Tango's test derives a simple asymptotic (1-α)-confidence interval for the difference δ and is shown to have good power and coverage probability. Tango's confidence intervals can be computed by:

$$\frac{b - c - n\delta}{\sqrt{n(2\hat{q_{21}} + \delta(1 - \delta))}} = \pm Z_{\frac{\alpha}{2}} \tag{1}$$

where $Z_{\frac{\alpha}{2}}$ denotes the upper $\frac{\alpha}{2}$-quantile of the normal distribution. In addition, $\hat{q_{21}}$ can be estimated by the maximum likelihood estimator for q_{21}:

$$\hat{q_{21}} = \frac{\sqrt{W^2 - 8n(-c\delta(1 - \delta))} - W}{4n} \tag{2}$$

where $W = -b - c + (2n - b + c)\delta$. Statistical hypothesis testing begins with a null hypothesis and searches for sufficient evidence to reject that null hypothesis. In this case, the null hypothesis states that there is no difference, or $\delta = 0$. By definition, a confidence interval includes plausible values for the null hypothesis. Therefore, if the zero is not included in the computed interval, then the null hypothesis $\delta = 0$ is rejected. On the other hand, if the zero value is included in the interval, then we do not have sufficient evidence to reject the difference being zero, and the conclusion is that the difference can be of any value within the confidence interval at the specified level of confidence (1-α).

Tango's test of equivalence can reach its limits in two cases; (1) when the values of b and c are both equal to zero where the Z statistic does not produce a value. This case occurs when we build a perfect classifier and is consistent with the test not using the number of correctly classified examples a and d. (2) The values b and c differ greatly. This is consistent with the assumption that the classifier is somewhat reasonably good, i.e. the classifier is capable of detecting a reasonable portion of the correct classifications in the domain. In both cases of limitations, the confidence intervals are still produced and are reliable [11] but may be wider in range. Tango's confidence intervals are shown not to collapse nor they exceed the boundaries of the normalized difference of $[-1, 1]$ even for small values of b and c.

Improving Control-Knowledge Acquisition for Planning by Active Learning

Raquel Fuentetaja and Daniel Borrajo

Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30. Leganés (Madrid). Spain
rfuentet@inf.uc3m.es, dborrajo@ia.uc3m.es

Abstract. Automatically acquiring control-knowledge for planning, as it is the case for Machine Learning in general, strongly depends on the training examples. In the case of planning, examples are usually extracted from the search tree generated when solving problems. Therefore, examples depend on the problems used for training. Traditionally, these problems are randomly generated by selecting some difficulty parameters. In this paper, we discuss several active learning schemes that improve the relationship between the number of problems generated and planning results in another test set of problems. Results show that these schemes are quite useful for increasing the number of solved problems.[1]

1 Introduction

The field of Active learning (AL) has been largely studied in the literature of inductive propositional learners [1,2,3,4]. As defined by [5], its aim is to generate an ordered set of instances (trials, experiments, experiences), such that we minimise the expected cost of eliminating all but one of the learning hypotheses. The task is NP-hard as Fedorov [6] showed. Therefore, AL requires a balance between the cost of selecting the ordered set and the cost of exploring all instances.

Several ML techniques have been implemented to acquire knowledge for planning. They go from ML of control knowledge, ML of quality-based knowledge, to ML of domain knowledge. In [7], the reader can find a good overview. In planning, the standard way of generating training examples has been providing a set of problems to planning-learning systems. Planners solve those problems one at a time, and learning techniques generate knowledge taking into account the search tree or the solution to the problems. One important issue to consider is that, opposite to what is common in AL for inductive learning, instances are not the problems themselves, but they are extracted from the process of solving them (like EBL techniques [8]).

Training problems are usually created by a random generator that has a set of parameters. These parameters theoretically define the problems difficulty.

[1] This work has been partially supported by the Spanish MCyT project TIC2002-04146-C05-05, MEC project TIN2005-08945-C06-05 and regional CAM-UC3M project UC3M-INF-05-016.

Usually, these parameters can be domain-independent (number of goals), or domain-dependent (number of objects, trucks, cities, robots, . . .). The advantage of this scheme for generating training problems is its simplicity. As disadvantages, the user needs to adjust the parameters in such a way that the learning extracts as much as possible from problems solved. Also, since problems are generated randomly, most of them will not be useful for learning. On one extreme, they are so difficult that the base planner cannot solve them. If the planner cannot solve them, no training examples will be generated. On the other extreme, if problems are so easy that the planner obtains the best solution without any wrong decisions, it is hard (for some learning techniques) to learn anything. This could be no problem for macro-operators acquisition [9] or CBR, since they learn from solutions. However, learning techniques that are based on decisions made in the search tree can have difficulties on generating training examples when the search trees do not include failure decisions. This is the type of learning techniques that we use in this paper: they learn control knowledge from search tree decisions [10].

In order to solve those disadvantages, we propose in this paper several AL methods to generate new training problems for Machine Learning (ML) techniques in the context of problem solving. The first approach generates problems with an increasing difficulty. The second approach generates new training problems with an increasing difficulty, based on the previous generated problem. Finally, the third approach generates new problems, also with increasing difficulty, based on the previous problem and the learned knowledge. These AL approaches are independent of the planner and the ML technique used. In our case, we have performed some experiments using IPSS as the planner [11] and HAMLET as the ML technique [10].

Section 2 provides a brief overview of the planner and learning technique we use. Section 3 describes the AL methods. Section 4 relates to previous work. Section 5 presents some experimental results. Section 6 makes some analysis, draws some conclusions and proposes future work.

2 The Planner and the Learning Technique

The AL schemes presented in this paper follow from the experience using HAMLET as a ML technique for the IPSSplanner [11]. IPSS is an integrated tool for planning and scheduling that uses QPRODIGY as the planner component. This is a version of the PRODIGY planner that is able to handle different quality metrics. It is a nonlinear planning system that follows a means-ends analysis. It performs a kind of bidirectional depth-first search (subgoaling from the goals, and executing operators from the initial state), combined with a branch-and-bound technique when dealing with quality metrics. The inputs to IPSS are the standard ones for planners (except for the heuristics, which are not usually given as explicit input in most planners): a domain theory, a problem, a set of planning-related parameters and, optionally, some domain-dependent heuristics expressed as control-rules. These control-rules can be manually provided or automatically learned by HAMLET. Given that the AL schemes proposed here are, somehow independent of the planner, we will not devote more space to explain how it works (see [12] for details on the planning algorithm).

HAMLET is an incremental learning method based on EBL and inductive refinement of control-rules [10]. The inputs to HAMLET are a task domain ($\mathcal{D}$), a set of training problems ($\mathcal{P}$), a quality measure (Q) and other learning-related parameters. For each training problem, HAMLET calls IPSS and receives in return the expanded search tree. Then, HAMLET extracts, by a kind of EBL, one control rule for each decision taken by IPSS in the best solution path. Thus, HAMLET output is a set of control-rules ($\mathcal{C}$) that can potentially guide the planner towards *good* quality solutions in future problems. Since these rules might be overly general or specific, HAMLET can generalize from two rules (merging their common parts) or specialize a rule when it failed (by adding more conditions). See [10] for more details. Figure 1 shows an example of a rule automatically learned by HAMLET in the logistics domain for selecting the `unload-airplane` operator. As it is, the control-rule says that if the goal is to have an object in a given location, `<location1>`, and the object is currently inside an airplane, then IPSS should use the `unload-airplane` instead of the `unload-truck` that also achieves the same goal (having an object in a given location).

```
(control-rule induced-select-operators-unload-airplane
  (if (and (current-goal (at <object> <location1>))
           (true-in-state (inside <object> <airplane>))
           (different-vars-p)
           (type-of-object <object> object)
           (type-of-object <airplane> airplane)
           (type-of-object <location1> airport)))
  (then select operator unload-airplane))
```

Fig. 1. Example of a control-rule learned by HAMLET for selecting the `unload-airplane` operator in the logistics domain

3 Active Learning for Planning

As any ML inductive tool, HAMLET needs a set of training problems to be given as input. The standard solution from the ML perspective is that the user provides as input a set of relevant planning problems. As any other ML technique, learning behaviour will depend on how similar those problems are to the ones that IPSS would need to solve in the future. However, in most real world domains, these problems are not available. So, an alternative solution consists on defining a domain-dependent generator of random problems. Then, one can assume (ML theory assumes) that if HAMLET sees enough random problems covering reasonably well the space of problems, and it learns from them, the learned knowledge will be reasonably adapted to the future. This solution requires to build one such generator for each domain, though there is some work on automating this task [13]. However, a question remains: what type of problems are the most adequate ones to train a ML technique for problem solving. Here, we propose the use of active learning schemes. In this approach, HAMLET would need to select at each step what are the characteristics that the next learning problem should have in order to improve the learning process. Then, HAMLET can call a problem generator with these characteristics as input, so that the problem generator returns a problem that is expected to improve learning. Examples of these characteristics in the logistics domain might be number of objects, number of goals, or number of cities. We have implemented four ways of generating training problems for HAMLET. Only versions two to four are really AL schemes. They are: one-step random generation; increasing difficulty generation; generation based on the last solved problem; and generation based on the rules learned. They are described in more detail in the next sections.

3.1 One-Step Random Generation

This is the standard way used in the literature of ML applied to planning. Before learning, a set of random training problems is generated at once, whose difficulty is defined in terms of some domain-independent characteristics (as number of goals) or domain-dependent characteristics (as number of objects to move). Usually, training is performed with simple problems, and generalization is tested by generating more complex test problems, also randomly.

3.2 Increasing Difficulty Generation

In this first AL scheme, at each cycle, a random generator is called to obtain a new training problem. Then, if the problem is suitable for learning, HAMLET uses it. Otherwise, the problem is discarded, and a new problem is randomly generated. The first difference of this second scheme with the previous version is that problems are incrementally generated and tested. Only the ones that pass the filter will go to the learner. As an example of a filter, we can define it as: select those problems such that they are solved, and the first path that the planner explores does not contain the solution. This means at least one backtracking had to be performed by the planner, and, therefore, at least one failure node will appear in the search tree. This simple test for validity will be valid if we try to learn knowledge that will lead the planner away from failure. However, if we want to learn knowledge on how to obtain "good" solutions, we have to use a more strict filter. Then, we can use IPSS ability to generate the whole search tree, to define another filter as: select those problems such that they are solved, the complete search tree was expanded, and the first path that the planner explores does not contain the solution. This filter allows HAMLET to learn from decisions (nodes) in which there were two children that lead to solutions: one with a better solution than the other. So, it can generate training instances from those decisions.

The second difference with respect to the random generation, is that the user can specify some parameters that express the initial difficulty of the problems, and some simple rules to incrementally increase the difficulty (difficulty increasing rules, *DIRs*). An example of the definition of an initial difficulty level, and some rules to increase the difficulty is shown in Figure 2. The level of difficulty (Figure 2(a)) is expressed as a list of domain-dependent parameters (number of objects, initially 1, number of cities, initially 1, ...) and domain-independent parameters (number of goals, initially 1). The rules for increasing the level of difficulty (Figure 2(b)) are defined as a list of rules, each rule is a list formed by the parameters to be increased by that rule and in what quantity. The first DIR, `((object 1) (no-goals 1))`, says that in order to increase the difficulty with it, the AL method should add 1 to the current number of objects, and add 1 to the current number of goals. Therefore, the next problem to be generated will be forced to have 2 objects and 2 goals.

```
((object 1) (city 1) (plane 1)
 (truck 1) (goals 1))
```
(a)

```
(((object 1) (no-goals 1)) ((city 1) (truck 1))
 ((object 1) (no-goals 1)) ((plane 1)))
```
(b)

Fig. 2. Example of (a) difficulty level and (b) rules for increasing the difficulty

The AL method generates problems incrementally, and filters them. If the learning system has spent n (experimentally set as 3) problems without learning anything, the system increases the difficulty using the DIRs. We explored with other values for n, though they lead to similar behaviour. The effect of increasing n is to spend more time on problems of the same difficulty, where we could have potentially explored all types of different problems to learn from. If we decrease n, then it forces the AL method to increase the difficulty more often, leading potentially to not exploring all types of problems within the same difficulty level.

3.3 Generation Based on the Last Problem

This AL method works on top of the previous one. In each level of difficulty, the first problem is generated randomly. Then, each following problem is generated by modifying the previous one. In order to modify it, the AL method performs the following steps:

– it selects a type t from the domain (e.g. in the logistics: truck, object, airplane) randomly from the set of domain types;
– it randomly selects an instance i of that type t from the set of problem-defined instances. For instance, if the previous step has selected `object`, then it will randomly choose from the defined objects in the problem (e.g. $\text{object}_1, \ldots \text{object}_n$);
– it randomly selects whether to change the initial state or the goal of the problem;
– it retrieves the literal l in which the chosen instance appear in the state/goal. For instance, suppose that it has selected $\texttt{object}_i$, such that $\texttt{(in object}_i\ \texttt{airplane}_j\texttt{)}$ is true in the initial state;

- it randomly selects a predicate p from the predicates in which the instances of the chosen type can appear as arguments. For instance, objects can appear as arguments in the predicates: `at` and `in`;
- it changes in the state/goal of the problem the selected literal l by a new literal. This new literal is formed by selecting randomly the arguments (instances) of the chosen predicate p, except for the argument that corresponds to the chosen instance i. Those random arguments are selected according to their corresponding types. For instance, suppose that it has chosen $\texttt{at}_{\texttt{O}bject}$. $\texttt{at}_{\texttt{O}bject}$ definition as predicate is ($\texttt{at}_{\texttt{O}bject}$ `?o - object ?p - place`), where `place` can be either `post-office` or `airport`. Then, it will change (`in` $\texttt{object}_i$ $\texttt{airplane}_j$) for (`at` $\texttt{object}_i$ $\texttt{airport}_k$), where $\texttt{airport}_k$ is chosen randomly.

This AL method assumes that it has knowledge on:

- types whose instances can change "easily" of state. For instance, in the logistics domain, objects can be easily changed of state by randomly choosing to be `at` another place or `in` another vehicle. However, in that domain, trucks cannot move so easily, given that they "belong" to cities. They can only move within a city. Therefore, in order to change the state of trucks, we would have to select a place of the same city where they are initially. This information could potentially be learned with domain analysis techniques as in TIM [14]. Now, we are not currently considering these types.
- predicates where a given type can appear as argument. This is specified indirectly in PDDL (current standard for specifying domain models) [15], and can be easily extracted (predicates are explicitly defined in terms of types of their arguments).
- arguments types of predicates. As described before, this is explicitly defined in PDDL.

As an example of this AL technique, if we had the problem defined in Figure 3(a), this technique can automatically generate the new similar problem in Figure 3(b). The advantage of this approach is that if rules were learned from the previous problem, the new problem forces HAMLET to learn from a similar problem, potentially generalizing (or specializing) the generated rules. And this can eventually lead to generating less problems (more valid problems for learning will be generated) and better learning convergence, by using gradual similar training problems.

3.4 Generation Based on the Rules Learned

In this last scheme of AL, the next training problem is generated taking into account the learned control rules of the previous problem. The idea is that the random decisions made by the previous AL scheme should also take into account the literals, predicates, instances and types that appear in the left-hand side of the rules learned from the last problem. Therefore, we compute the frequency of appearance of each predicate, type and instance before performing

```
(create-problem (name prob-0)
  (objects (ob0 object) (c0 city))
           (po0 post-office (a0 airport)
           (tr0 truck) (pl0 airplane))
  (state (and (same-city a0 po0)
              (same-city po0 a0)
              (at tr0 a0) (in ob0 pl0)
              (at pl0 a0)))
  (goal (at ob0 po0)))
                    (a)
```

```
(create-problem (name prob-1)
  (objects (ob0 object) (c0 city))
           (po0 post-office (a0 airport)
           (tr0 truck) (pl0 airplane))
  (state (and (same-city a0 po0)
              (same-city po0 a0)
              (at tr0 a0) (at ob0 a0)
              (at pl0 a0)))
              (goal (at ob0 po0)))
               (b)
```

Fig. 3. Example of (a) problem solved and (b) problem automatically generated by AL. The difference appears in bold face

the parametrization step when generating each precondition of each control rule learned. Then, instead of randomly choosing a type, instance, and new predicate for the next problem, we do it using a roulette mechanism based on the relative frequency of them.

Suppose that HAMLET has only learned the two rules in Figure 4(a) from a given problem (they are shown before converting the instances to variables).

```
(control-rule select-operators-unload-airplane
  (if (and (current-goal (at ob0 a0))
           (true-in-state (inside ob0 pl0))
           (different-vars-p)
           (type-of-object ob0 object)
           (type-of-object pl0 airplane)
           (type-of-object a0 airport)))
  (then select operator unload-airplane))

(control-rule select-bindings-unload-airplane
  (if (and (current-goal (at ob0 a0))
           (current-operator unload-airplane)
           (true-in-state (at ob0 a0))
           (true-in-state (at pl0 a0))
           (true-in-state (at pl1 a0))
           (different-vars-p)
           (type-of-object ob0 object)
           (type-of-object pl0 airplane)
           (type-of-object pl1 airplane)
           (type-of-object a0 airport)))
  (then select bindings ((<object> . ob0)
                         (<airplane> . pl0)
                         (<airport> . a0))))
```

(a)

Instances	Frequency	Percentage (over the same type)
ob0	2	100%
pl0	2	66%
pl1	1	33%
a0	3	100%

Predicates	Frequency	Percentage
at	3	75%
inside	1	25%

Types	Frequency	Percentage
object	2	25%
airport	3	37%
airplane	3	37%

(b)

Fig. 4. Example of (a) learned rules by HAMLET and (b) the correspondent frecuency table in true-in-state

Then, Figure4(b) shows the computed frequencies. There are some predicates (at), types (object and airport), or instances (ob0 and a0) that have more probability of being changed than others. This is based on the observation that they appear in the preconditions of the learned control rules. Since, the goal is to improve the convergence process, new problems are randomly generated from previous ones, but in such a way that they will more probably generate new training instances of control rules that will force old control rules to be generalized (from new positive examples), or specialized (from new negative examples of the application of the control rules). The difference with the previous approach is that if what is changed from one problem to the next one did not appear on the LHS of rules (it did not matter for making the decission), it would not help on generalizing or specializing control rules.

4 Related Work

One of the first approaches for problem solving was the LEX architecture [4]. It consisted of four interrelated steps of problem solving, AL, and learning: generation of a new suitable learning episode (e.g. a new integral to be solved), problem solving (generation of the solution to the integral), critic (extraction of training examples from search tree), and learning (generation from examples of version space of when to apply a given operator to a particular type of integral). In this case, AL was guided by the version space of each operator. Our approach is based on that work. However, since we do not have a version space for each type of control rule (target concept, equivalent to the operators in LEX), we have defined other approaches for guiding the AL process.

Other AL approaches have also focused on classification tasks using different inductive propositional and relational techniques [5,1,2,3]. Usually, AL selects instances based on their similarity to other previous instances, or to their distance to frontiers among classes. In our case, instances are extracted from a search tree, while AL generates problems to be solved, and training examples are extracted from their search trees. Therefore, for us it is not quite obvious how to use those techniques to generate valid instances (problem solving decisions) from previous ones. Finally, work on reinforcement learning also has dealt with the problem of how to obtain new training instances. This is related to the exploitation vs. exploration balance, in that, at each decision-action step, these techniques have to decide whether to re-use past experience by selecting the best option from previous experience (exploitation) or select an unknown state-action pair and allow learning (exploration).

5 Experiments and Results

This section presents some results comparing the four methods of generating problems described previously. We have used the original *Logistics* and the *Zenotravel* domains. In both domains, using the *One-Step Random Generation* approach, we generated a set of 100 random training problems. In this process, the problem difficulty is defined as parameters of the random generator. In the *Logistics*, we generated problems with a random number of goals (between 1 and 5), objects (between 1 and 2), cities (between 1 and 3) and planes (between 1 and 2). In the *Zenotravel*, we generated problems with a random number of goals (between 1 and 5), persons (between 1 and 3), cities (between 1 and 3) and planes (between 1 and 4).

In the AL approaches, the “a priori” generation of random problems is not needed. We only specify the maximum number of valid problems to generate (set as 100), the initial difficulty level and the rules for increasing the difficulty. The number of problems without learning anything before increasing the difficulty was fixed to 3. The definition of the parameters used in these experiments is shown in Figures 2 and 5.

```
((person 1 1) (city 2 2) (aircraft 1 1))
```
(a)

```
(((person 1)) ((aircraft 1)) ((no-goals 1))
 ((aircraft 1)) ((city1) (person 1))
 ((no-goals 1)) ((aircraft 1) (city 1)))
```
(b)

Fig. 5. (a) Difficulty level and (b) rules for increasing the difficulty in the *Zenotravel* domain

In the learning phase, we let HAMLET generate 100 valid problems, and learn control rules from them using the following schemes: *One-Step Random Generation* (OSRG), *Increasing Difficulty Generation* (IDG), *Generation based on the last Problem* (GBP), and *Generation based on the Rules learned* (GBR). In each case, we saved the control rules learned when 5 (only for the *Logistics*), 25, 50, 75 and 100 training problems were used.

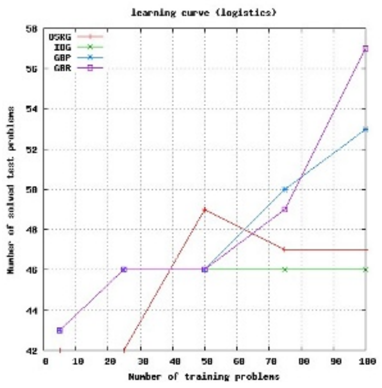
Then, we randomly generated a set of 120 test problems, of varying difficulty from 1 to 15 goals, and from 1 to 5 objects in the *Logistics* domain and from 1 to 10 goals, 2 to 10 persons and cities and 2 to 15 planes in the *Zenotravel* domain.

The IPSS planner was run on the test set for all the sets of control rules saved and for all the schemes. Figure 6 shows the obtained learning curves for the *Logistics* (a) and *Zenotravel* domains (b). In this figure, the y-axis represents the number of test problems (over 120) solved by the planner in each case, while the x-axis represents the number of training problems used for learning.

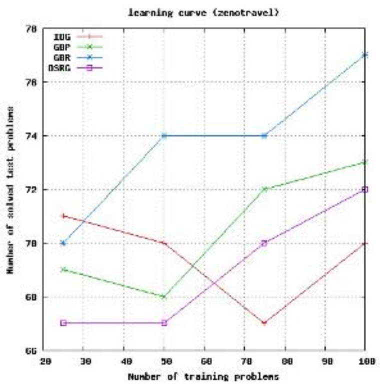
As one would expect, the scheme with the best convergence is GBR, followed by GBP. These results confirm that the more informative the generation of each next training problem is, the better the convergence of the learning process. The improvement in the convergence of these schemes begins in training problem 75 (*Logistics*) and 50 and 75 for GBR and GBP respectively (*Zenotravel*).

These results can be well explained analyzing when the difficulty was increased (in the same level or using the next difficulty rule). The changes in the difficulty are shown in Figure 7.

In the *Logistics* for instance, the first DIR was applied once around the problem 8 in all the AL schemes. However, the second DIR was applied first by GBR in training problem 70, second by GBP in training problem 88, while it was never applied by IDG. On one hand, the fact that GBR applies the second DIR before than GBP means that HAMLET learns from more training problems in this level using GBR than using GBP. For this reason, the final number of solved test problems is bigger using GBR. When the second difficulty rule was applied also explains why the improvement in the convergence appears in problem 75 for these two schemes. On the other hand, the low performance of IDG is due to the



(a)



(b)

Fig. 6. Learning curves in the *Logistics* (a) and *Zenotravel* (b) domains

scheme	DIR 1	DIR 2	DIR 3	DIR 4
IDG	7	-	-	-
GBP	9	88	-	-
GBR	9	70	-	-

(a)

scheme	DIR 1	DIR 2	DIR 3	DIR 4	DIR 5	DIR 6	DIR 7
IDG	9	15	37	42	51	54	91
GBP	9	22	29	-	-	-	-
GBP	9	22	30	-	-	-	-

(b)

Fig. 7. Difficulty changes in the *Logistics* domain (a) and *Zenotravel* domain (b)

fact that using this scheme the learning engine does not have the opportunity of learning from training problems with more than one city and one truck, because the second DIR was never applied. A similar analysis can be done in the case of *Zenotravel*, though, in this case, the IDG scheme changes more often of difficulty level.

6 Discussion and Future Work

One of the main constrains of ML within problem solving is that it needs that the training set contains solvable problems. If the problems are not solvable, nothing will be learned, since learning occurs when in at least one node of the search tree, there is a success child (this is true for learning from success, which is the learning scheme we are using). Without the use of an AL scheme, one must program a domain-dependent random generator for each domain, so that it generates solvable problems. This is not quite obvious in the case of planning problem-solving (there is no guarantee that from any initial state, we can arrive to a state in which goals are true). Formally proving, independently of the domain, that a problem generator generates only solvable problems is a very hard task. One of the main advantages of using AL schemes in the learning process is the fact that they avoid the need of programming a random problem generator for each domain. Thus, we transform a procedural programming task into a task of defining the right declarative knowledge that expresses: the domain types whose instances can "easily" change of state, together with the different possible states of the objects of this type; the initial difficulty level; and the rules to increase the difficulty. Thus, each training problem is generated using the previous one, and if the domain has some features, explained below, each generated problem from a previous solvable problem is solvable also. Furthermore, the AL schemes we presented here only need just one solvable problem for each difficulty level; the first one of each level. Although the learning process has the bias impossed by this first problem, it is compensated with a random exploration of problems in each level.

The main disadvantage of the approaches presented in this paper is that they are limited to some types of domains. These domains should have the following two features: (1) They must include types whose instances can change "easily" of state (though not all types are required to have that feature). This means that the application of operators which have as parameters objects of these types should produce one-literal changes in the state. The reason is that it is not possible to control complex changes (more than one literal involved) in the states just using separately the information of the different possible states of each type of object. This limitation could be solved using a forward planner to generate the next problem by applying just one applicable operator in the initial state of the problem at hand. But, in this case, in order to obtain a solvable problem, the domain has to be symmetrical (next feature). (2) It is preferable to use symmetrical domains (for each instantiated operator o_i, there must be a sequence of instantiated operators, such that, if applied, they return to the planning state before applying o_i). Otherwise, the AL mechanism could generate a big number of unsolvable problems, decreasing thus the number of useful problems to learn. Useful problems to learn are defined by features of their planning process. For example, it can be considered useful problems whose planning process suppossed at least a backtrack, because the first path followed by the planner did not end in the best solution and this produces opportunities to learn. Therefore, not all solvable problems are useful for learning, but a problem only can be useful for learning if it is solvable.

Examples of domains with both of previous features are the *Logistics*, the *Zenotravel* and the *Satellite* domains used in the International Planning Competitions. The *Blocksworld* domain, does not include instances that change "easily" of state. It has only the type *object*. When an object o_1 is changed from a state where it is *on* another object o_2, to a state where it is holding, the change implies changes in the state of o_2 too (it should be *clear*). Changes like this are considered complex. The same happens with the objects in the *Depots* domain. Regarding the second feature, the *Rockets* domain is an example of non-symmetrical domain, because the rockets only have enough fuel to just one change of location.

Currently, we are already performing experiments in other domains with the goal of understanding in what type of domains, it is experimentally useful (apart from the discussion we presented previously on symmetrical domains and "easy" changes).

References

1. Cohn, D., Ghabhramani, Z., Jordan, M.: Active learning with statistical models. Journal of Artificial Intelligence Research **4** (1996) 129–145
2. Liere, R., Tadepalli, P., Hall, D.: Active learning with committees for text categorization. In: Proceedings of the Fourteenth Conference on Artificial Intelligence, Providence, RI (USA) (1997) 591–596
3. Lin, F.R., Shaw, M.: Active training of backpropagation neural networks using the learning by experimentation methodology. Annals of Operations Research **75** (1997) 129–145
4. Mitchell, T.M., Utgoff, P.E., Banerji, R.B.: Learning by experimentation: Acquiring and refining problem-solving heuristics. In: Machine Learning, An Artificial Intelligence Approach. Tioga Press, Palo Alto, CA (1983)
5. Bryant, C., Muggleton, S., Oliver, S., Kell, D., Reiser, P., King, R.: Combining inductive logic programming, active learning and robotics to discover the function of genes. Linkoping Electronic Articles in Computer and Information Science **6**(12) (2001)
6. Fedorov, V.: Theory of Optimal Experiments. Academic Press, London (1972)
7. Zimmerman, T., Kambhampati, S.: Learning-assisted automated planning: Looking back, taking stock, going forward. AI Magazine **24**(2) (2003) 73–96
8. Minton, S.: Learning Effective Search Control Knowledge: An Explanation-Based Approach. Kluwer Academic Publishers, Boston, MA (1988)
9. Fikes, R.E., Hart, P.E., Nilsson, N.J.: Learning and executing generalized robot plans. Artificial Intelligence **3** (1972) 251–288
10. Borrajo, D., Veloso, M.: Lazy incremental learning of control knowledge for efficiently obtaining quality plans. AI Review Journal. Special Issue on Lazy Learning **11**(1-5) (1997) 371–405 Also in the book "Lazy Learning", David Aha (ed.), Kluwer Academic Publishers, May 1997, ISBN 0-7923-4584-3.
11. Rodríguez-Moreno, M.D., Borrajo, D., Cesta, A., Oddi, A.: Integrating planning and scheduling in workflow domains. Expert System with Applications **33**(2) (2007)
12. Veloso, M., Carbonell, J., Pérez, A., Borrajo, D., Fink, E., Blythe, J.: Integrating planning and learning: The PRODIGY architecture. Journal of Experimental and Theoretical AI **7** (1995) 81–120
13. McCluskey, T.L., Porteous, J.M.: Engineering and compiling planning domain models to promote validity and efficiency. Artificial Intelligence **95**(1) (1997) 1–65
14. Fox, M., Long, D.: The automatic inference of state invariants in TIM. Journal of Artificial Intelligence Research **9** (1998) 317–371
15. Fox, M., Long, D.: PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. University of Durham, Durham (UK). (2002)

PAC-Learning of Markov Models with Hidden State

Ricard Gavaldà[1], Philipp W. Keller[2], Joelle Pineau[2], and Doina Precup[2]

[1] Universitat Politècnica de Catalunya, Barcelona, Spain
[2] McGill University, School of Computer Science, Montreal, QC, Canada

Abstract. The standard approach for learning Markov Models with Hidden State uses the Expectation-Maximization framework. While this approach had a significant impact on several practical applications (e.g. speech recognition, biological sequence alignment) it has two major limitations: it requires a known model topology, and learning is only locally optimal. We propose a new PAC framework for learning both the topology and the parameters in partially observable Markov models. Our algorithm learns a Probabilistic Deterministic Finite Automata (PDFA) which approximates a Hidden Markov Model (HMM) up to some desired degree of accuracy. We discuss theoretical conditions under which the algorithm produces an optimal solution (in the PAC-sense) and demonstrate promising performance on simple dynamical systems.

1 Introduction

Hidden Markov Models (HMMs) are widely used tools for prediction under uncertainty. Successful applications of these technologies include speech recognition (Rabiner, 1989) and DNA sequence alignment (Durbin et al, 1998). In this paper, we address the issue of learning such models from data.

The standard approach at the moment is to estimate model parameters directly from trajectories of observations (or action-observation pairs) using Expectation-Maximization (EM) (Rabiner, 1989). This approach has proved successful in many applications, but it also has some significant drawbacks. First, it assumes a known set of "real" hidden states S. In many domains, in particular in physical systems, there is a natural state representation. For example, in speech recognition, the set of phonemes is the standard choice of state representation, and in computational biology, the type of the subsequence (e.g., gene or promoter) is a natural choice. However, there are many domains where the choice of states is not at all obvious. For example in dialogue modelling, the state representation must somehow capture the user's communication goals. Similarly, in medical diagnostic and adaptive treatment design, the state must capture complex information about the patient, his/her disease and treatment history. In these and similar cases, the state is best represented by summary statistics over the set of past observations. Some recent research has focused on modeling just the observed data (Jaeger et al, 2006, Rosencrantz et al, 2004, Singh et al, 2003). In this case, knowing or defining hidden states ahead of time is not necessary. The algorithm we propose in this paper has a similar goal, although the methodology is different. We build a learning algorithm for probabilistic models which can simultaneously estimate a good state topology and a corresponding set of parameters.

The second drawback of EM is that it converges to a locally optimal solution, and there are no guarantees on the quality of the final solution. In some domains, this is very problematic. The algorithm we propose has PAC-style guarantees on the model learned, using a polynomial amount of data.

We use Probabilisitc Deterministic Finite Automata (PDFA), a standard tool in computational learning theory, as the basic representation to be learned. We show how PDFAs can approximate HMMs. Our algorithm is based on a state-splitting and merging technique, and is designed to be able to provide PAC guarantees. We illustrate the algorithm on some example problems, and show promising empirical results.

Some proofs and discussions are omitted in this version. More details can be found in the technical report version, available from the first author's homepage.

2 Background

We address the problem of learning the structure and parameters of a dynamical system, directly from observational data generated by the system. The data typically consists of a set of trajectories, $D = \{d^1, d^2, ..., d^n\}$, each containing a finite sequence of observations $d = \sigma_0\sigma_1...\sigma_k$. Different models have been used to capture this type of data; in this paper, we focus on Hidden Markov Models and Probabilistic Finite Automata.

A *probabilistic deterministic finite automaton (PDFA)* is a tuple $\langle S, \Sigma, T, O, s_0 \rangle$, where S is a finite set of states, Σ is a finite set of observations, $T : S \times \Sigma \rightarrow S$ is the transition function $O : S \times \Sigma \rightarrow [0,1]$ defines the probability of emitting each observation from each state, $O(s,\sigma) = P(\sigma_t = \sigma | s_t = s)$, and $s_0 \in S$ is the initial state. Note that the transition to a new state is deterministic once an observation has been selected: $T(s,\sigma)$ gives the next state s'. A special symbol is reserved to mark the end of a string; alternatively, one can interpret this as a stop state with no outgoing edges. A probabilistic nondeterministic finite automaton (PNFA) is defined similarly except the transition function is stochastic: $T : S \times \Sigma \times S \rightarrow [0,1]$, and $T(s,\sigma,s') = P(s_{t+1} = s' | s_t = s, \sigma_t = \sigma)$.

Given an observation trajectory $d = \sigma_0\sigma_1, ..., \sigma_k$ emitted by a known PDFA, the state at each time step can be tracked by starting from the initial state s_0 and following the labelled transitions according to d. Also, the probability of generating a given trajectory $d = \sigma_0\sigma_1, ..., \sigma_k$ from a state s can be calculated recursively as follows: $O(s, \sigma_0\sigma_1...\sigma_k) = O(s,\sigma_0)O(T(s,\sigma_0), \sigma_1...\sigma_k)$.

A *Hidden Markov Model* is a tuple $\langle S, \Sigma, T, O, b_0 \rangle$, where S is a finite set of states, Σ is a finite set of observations, $T(s,s') = P(s_{t+1} = s' | s_t = s)$ defines the probability of transitioning between states, $O(s,\sigma) = P(\sigma_t = \sigma | s_t = s)$ defines the emission probability of each observation on each state, and $b_0(s) = P(s_0 = s)$ is the initial state distribution. Given an observation trajectory d emitted by a known HMM, the probability distribution over states at any time b_{t+1}, can be estimated recursively by Bayesian updating:

$$b_{t+1}(s) \propto \Sigma_{s' \in S} b_t(s') O(s', \sigma_t) T(s', s) \tag{1}$$

Several links have been established between HMMs and probabilistic automata; a comprehensive review is in (Dupont et al., 2005). From the point of view of this paper, it is most important to note that an HMM can be transformed into an equivalent PNFA with the same number of states. A PNFA can also be transformed into an HMM, but not necessarily with the same number of states. Any PDFA $M = \langle S, \Sigma, T, O, s_0 \rangle$ can be converted to an equivalent HMM $M' = \langle S', \Sigma, T', O', b_0 \rangle$. The states in S' correspond to pairs of states in S among which a transition is possible: $S' = \{(s_1, s_2) \in S \times S | \exists \sigma \in \Sigma \text{ s.t. } T(s, \sigma) = s'\}$. The probability distributions of the HMM are then built as follows:

$$b_0((s_0, s')) = 1/|S| \qquad O'((s,s'),\sigma) = \frac{O(s,\sigma)}{\sum_{\sigma'=\Sigma} O(s,\sigma')}$$

$$T'((s,s'),(s',s'')) = \sum_{\sigma \in \Sigma} O(s',\sigma)\delta(T(s',\sigma),s'')$$

where δ is an indicator function. All other parameters are 0. It is easy to show that M' defines a proper HMM, and that M and M' generate the same probability distribution over observation trajectories. Unfortunately, the reverse is not true: there are finite HMMs that can only be converted into PDFAs of infinite size. However, we will now show that any HMM can be *approximated* with a finite PDFA up to any desired degree of precision.

3 Approximating HMMS with PDFAs

Recalling that every HMM is equivalent to a PNFA (Dupont et al, 2005), we show that every finite-size PNFA can be approximated by a finite-size PDFA.

Theorem 1. *Let N be a PNFA and L be the expected length of strings generated by N. Then there exists a PDFA of size at most L/ε^2 that generates a distribution over trajectories that is ε-close to the distribution generated by N, under the L_∞ distance.*

Proof. Recall that L_∞ measures the maximum difference between the corresponding components of two vectors. Here, we will use it to measure the maximum difference between the probability assigned to the same string by two different distributions. Let S be the set of strings having probability at least ε in N. Note that there are at most $1/\varepsilon$ such strings, i.e., finitely many. It is easy to build a finite, tree-like PDFA M with $|S|$ leaves that generates exactly the strings in S, each with the same probability as N, and no other string. Hence, the distributions of M and N are ε-close.

To explicitly bound the size of the tree, we observe that if $u \in S$, then necessarily $|u| \leq L/\varepsilon$. Let S_N be the random variable describing the string output by PNFA N. Then by Markov's inequality we have
$\varepsilon \leq \Pr[S_N = u] \leq \Pr[|S_N| \geq |u|] \leq E[|S_N|]/|u| \leq L/|u|$
which completes the proof.

This is a generic construction whose value is only to show that finite-size approximation of PNFA (and HMM) by PDFA is always possible. However, the machine we construct for the proof does not capture the internal structure of the PNFA/HMM. But the fact that PDFAs can be used to approximate HMMs suggests a new class of algorithms that could be used to learn HMMs. More precisely, one can think of trying to learn a PDFA that approximates an HMM. The size of the PDFA would depend on factors such as the desired degree of accuracy, and the amount of data available.

PDFAs and PNFAs have been studied extensively in computational learning theory, especially in the context of PAC-learning. In this context, the goal of learning is to find a model that approximates the true probability distribution over observation trajectories, $\mathcal{P}$. A learning algorithm will produce a model which generates a distribution over observation trajectories $\hat{\mathcal{P}}$. A model, or hypothesis, is called ε-good, if the distance between $m(\mathcal{P},\hat{\mathcal{P}}) < \varepsilon$, where m is a reasonable distance measure (e.g. L_∞ or the Kullback-Leibler divergence) and $\varepsilon > 0$ is the desired precision. Given observation trajectories that are drawn i.i.d. from the system, an error parameter $\varepsilon > 0$ and a confidence parameter $\delta \in (0,1)$, a PAC-learning algorithm must output an ε-good model with probability at least $1-\delta$. A class of machines is called efficiently PAC-learnable if there exists a PAC-learning algorithm whose time complexity is polynomial in $1/\varepsilon$, $1/\delta$ and the number of parameters of the target machine. A class of machines is polynomially PAC-learnable if the training sample (i.e. the number of trajectories needed) is polynomial in the same quantities.

Several PAC-style results have been established over the years on the topic of learning PDFAs and HMMs. (see Dupont et al, 2005 for a comprehensive discussion). Of particular relevance to our work is the result by Kearns et al. (1994) establishing that the class of all PDFAs is in fact *not* efficiently PAC-learnable. However Ron et al. (1995) argued that by restricting attention to the class of PDFAs that are acyclic and have a *distinguishability criterion* between states, PAC-learning is possible.

Definition 1. *Let m be a measure of the difference between two probability distributions. A PDFA has* ***distinguishability*** *μ if for any two states s and s', the probability distributions over observation trajectories starting at s and s', $\mathcal{P}_s$ and $\mathcal{P}_{s'}$, differ by at least μ: $m(\mathcal{P}_s,\mathcal{P}_{s'}) \geq \mu, \forall s,s' \in S$.*

Intuitively, this class of machines does not have states that are “too similar” in terms of the probability distribution of trajectories following them. More recently, Clark and Thollard (2004) provided an efficient PAC-learning algorithm for this subclass of PDFAs which requires an amount of data polynomial in the number of states in the target, the “distinguishability” of states and the expected length of strings generated from any state. In the next section, we build on their work to provide a learning algorithm for PDFAs/ HMMs with PAC-style guarantees, then analyze this algorithm.

4 A PAC-Learning Algorithm

The algorithm builds a graph whose nodes intuitively represent postulated states of the target machine. We call these nodes “safe states”. The algorithm also maintains a list of “candidate states” that will eventually be merged with existing safe states or promoted to be new safe states.

The algorithm uses both state splitting and state merging operations. We begin by assuming that the initial model is a trivial graph with a single safe state representing the initial state of the target machine. In the induction step, we refine the graph by adding a new safe state $s\sigma_i$, whenever the training data suggests that there is a sufficient difference between the probability distribution over the trajectories observed from $s\sigma_i$ and the distribution observed from any safe state s'. Similarly. if the distribution of trajectories observed from $s\sigma_i$ and an existing safe state s' are sufficiently similar, we merge (or identify) these states. The remainder of this section formalizes these basic ideas, including the precise criteria for creating new safe states and merging candidate states into existing safe states.

We assume that the set of possible observations Σ is known, and that we have a set of training trajectories D, with each trajectory being sampled i.i.d. from the correct target. The algorithm assumes the following input parameters: δ, n, μ where δ is the desired confidence (as in standard PAC-learning (Valiant, 1984)), n is an upper bound on the number of states desired in the model, and μ is a lower bound on the distinguishability between any two states. We assume the L_∞ norm as the measure m (see Definition 1).

We begin by defining a single safe state $S = \{s_0\}$, labeled with a *null* observation. Then we consider a set of candidate states $s_0\sigma$ for every observation $\sigma \in \Sigma$. With each safe and candidate state, we associate a multiset, D_s and $D_{s\sigma}$ respectively, storing the suffixes of all training trajectories that pass through this state (or a sufficient statistic thereof).

For each given training trajectory $d = \sigma_0 \ldots \sigma_{i-1}\sigma_i\sigma_{i+1} \ldots \sigma_k$, we traverse the graph matching each observation σ_i to a state until either (1) all observations in d have been exhausted (in which case we discard d and proceed to the next training trajectory), or (2) a transition to a candidate state is reached. This occurs when all transitions up to σ_{i-1} are defined and lead to a safe state s, but there is no transition out of s with observation σ_i. In this case, we add the sub-trajectory $\{\sigma_{i+1} \ldots \sigma_k\}$ to the multiset $D_{s\sigma_i}$.

The next step is to decide what to do about candidate state $s\sigma$. There are three possibilities: (1) **retain** it as a candidate state; (2) **merge** it with an existing state $s' \in S$; (3) **promote** it to be a new state $S = S \cup \{s\sigma\}$. This step is the core of the algorithm.

The decision of whether to merge, promote, or retain a candidate state depends on the content of its multiset $D_{s\sigma}$. To better explain this step, we introduce some notation, which applies both for safe and candidate states. We denote by $|D_s|$ the cardinality of D_s and by $D_s(d)$ the number of times trajectory d occurs in D_s. We denote by $|D_s(\sigma)|$ the number of trajectories starting with observation σ in multiset D_s. Note that $|D_s(d)|/|D_s|$ can be regarded as an empirical approximation of the probability that trajectory d will be observed starting from state s.

The decision of whether to retain a candidate is taken first. If the state is not retained, we then consider whether to promote it or merge it with an existing state. A candidate state $s\sigma$ is declared **large** when:

$$\text{(largeness condition) } |D_{s\sigma}| \geq \frac{3(1+\mu/4)}{(\mu/4)^2} \cdot \ln \frac{2}{\delta'} \tag{2}$$

where $\delta' = \frac{\delta\mu}{2(n|\Sigma|+2)}$. When a candidate state is declared large, it will not be retained. Intuitively, in this case there is enough information to promote or merge it correctly.

Suppose state $s\sigma$ has been declared large. If there exists some safe state s' such that for every trajectory d we have

$$\left| \frac{|D_{s\sigma}(d)|}{|D_{s\sigma}|} - \frac{|D_{s'}(d)|}{|D_{s'}|} \right| \leq \mu/2, \tag{3}$$

then we merge $s\sigma$ and s': we therefore remove $s\sigma$ as a candiate state, we create a transition from s to s' labelled with σ, and increase the counts of $|D_{s'}(d)|$ by those of $|D_{s\sigma}(d)|$.

If, on the contrary, for every s' there is a d such that

$$\left|\frac{|D_{s\sigma}(d)|}{|D_{s\sigma}|} - \frac{|D_{s'}(d)|}{|D_{s'}|}\right| > \mu/2$$

then we promote $s\sigma$ to be a new safe state; we add a transition from s to $s\sigma$ labelled with σ and add candidate states $s\sigma\sigma'$ for every observation $\sigma' \in \Sigma$. All trajectories in $D_{s\sigma}$ are moved appropriately to these new candidate states, as if they had been observed from $s\sigma$.

The graph built as described above can easily be transformed into a PDFA. Every safe state becomes a state of the automaton. The set of observations Σ is the same. The observation probability function $O(s,\sigma)$ is calculated using the multiset statistics:

$$O(s,\sigma) = \frac{|D_s(\sigma)|}{\sum_{\sigma' \in \Sigma} |D_s(\sigma')|}(1-(|\Sigma|+1)\gamma)+\gamma, \tag{4}$$

where $\gamma < \frac{1}{|\Sigma|+1}$ is a small smoothing probability (which can be set to 0 if smoothing is not desired).

The only real question left is what to do about the candidate states. Given a candidate state $s\sigma$, we look for the safe state s' that is most similar to it according to a chosen distance metric. E.g., assuming L_∞, we have $s' = \text{argmax}_{s' \in S}\left(\frac{|D_{s\sigma}(d)|}{|D_{s\sigma}|} - \frac{|D_{s'}(d)|}{|D_{s'}|}\right)$. We then add an edge from s to s' with label σ to the automaton M and calculate the observation probability as in Equation 4. Finally, the transition function is $T(s,\sigma) = s\sigma$.

Table 4 summarizes the algorithm presented in this section. Note that, as presented here, the algorithm works in batch mode. As such, there will necessarily be a point at

Table 1. Learning Algorithm

```
M = PDFA-Learn ($\Sigma, D, \delta, n, \mu$)
  Initialize safe states $S = \{s_0\}$                          INITIALIZING
  $D_{s_0} = D$
  Initialize candidates $\bar{S} = \{s_0\sigma | \forall\sigma \in \Sigma\}$
  $D_{s_0\sigma} = \{\sigma_2 \ldots \sigma_k | \exists d \in D_{s_0}, d = \sigma\sigma_2 \ldots \sigma_k\}$
  While $\exists s\sigma \in \bar{S}$ which is large, as given by (2)
    Remove $s\sigma$ from $\bar{S}$
    If $\exists s' \in S$ such that $\forall d$ (3) is satisfied          MERGING
      Add transition from $s$ to $s'$ labelled by $\sigma$
      $D_{s'} = D_{s'} \cup D_{s\sigma}$
    Else                                                        PROMOTING
      $s' = s\sigma$
      $S = S \cup \{s'\}$
      $D_{s'} = D_{s\sigma}$
      $\bar{S} = \bar{S} \cup \{s'\sigma' | \forall\sigma' \in \Sigma\}$
      $D_{s'\sigma} = \{\sigma_2 \ldots \sigma_k | \exists d \in D_{s'}, d = \sigma\sigma_2 \ldots \sigma_k\}$
    End if
  End while
  Construct the output graph representing the learned PDFA.
```

which no candidate state meets the largeness condition, and the algorithm terminates. However, it is easy to imagine implementing this as an incremental algorithm, in which the graph is restructured after each trajectory is received. In this case, the largeness condition will be checked every time a new trajectory is added to the multiset of a state. It is important to note that if the algorithm runs on-line, states can continue to become large as more data is gathered, and the machine will continue to grow. One possibility to stop this is to limit the number of acceptable states, using the parameter n. In Appendix A, we discuss a different, sufficient termination condition for this case. It is based on using the precision ε desired in the approximation of the trajectory distribution, and provides a strong improvement over the bounds of Clark & Thollard (2004) .

It is in general not necessary to recover a true HMM from the learned PDFA; we will consider the learned PDFA to be an approximation of the HMM, which can be used to compute (approximately) the probabilities of different trajectories. Not that an HMM can be recoverred followinng the steps outlines in Sec. 2. It should be noted that this output HMM may be of larger size than the target machine.

5 Analysis

A full analysis of the algorithm should show that 1) after seeing a certain number of examples, the graph under construction becomes isomorphic to that of the target machine, except for low-probability states, and that 2) in addition, after some more examples, the edge probabilities are close enough to the target ones that the distance in the probability distribution over trajectories is small. In this section we present a sketch of these proofs, highlighting the differences with results by Clark and Thollard (2004).

We first state how long it takes for a candidate state to become large. Observe that the more frequent a state is, the sooner it will be identified. In contrast, typical PAC approaches require a lower bound on the desired frequency, P, and run in time polynomial in $1/P$ even if most states have frequency much larger than P. No such parameter is required by our algorithm. This adaptive behavior shows good potential for the practicality of our approach.

Let $|\hat{D}_s|$ denote $E[|D_s|]$ and $|\hat{D}_s(d)|$ denote $E[|D_s(d)|]$.

Theorem 2. *(1) Let s be a candidate or safe node. At the time when s is declared large we have $||D_s| - |\hat{D}_s|| \leq |\hat{D}_s| \cdot (\mu/4)$ with probability $1 - \delta'$. That is, $|D_s|$ is an approximation to $|\hat{D}_s|$ up to a multiplicative factor of $\mu/4$.*

(2) Let $s\sigma$ be a candidate node, and $p \cdot t$ be the expected value of $|\hat{D}_{s\sigma}|$ at time t Then $s\sigma$ is declared large at most

$$T = \frac{3(1+\mu/4)}{(1-\mu/4)(\mu/4)^2 p} \cdot \ln \frac{2}{\delta'}$$

steps after it was created, with probability at least $1 - \delta'$.

The proof is technically similar to some used in (Lipton and Naughton, 1995) in the context of databases. The details are omitted here, but are presented in the associated technical report.

Theorem 3. *The largeness condition in Equation (2) guarantees that, for any large state s,*

$$\forall d \; \left| \frac{|D_s(d)|}{|D_s|} - \frac{|\hat{D}_s(d)|}{|\hat{D}_s|} \right| < \frac{\mu}{4} \tag{5}$$

with probability $1-\delta$.

The proof is essentially given in Section 6.1 of (Clark and Thollard, 2004).

From this claim, one can argue that the decisions to merge and promote candidate states are correct with high probability. Indeed, suppose that at any point we decide to merge $s\sigma$ with s'. This is because $s\sigma$ has become large and

$$\forall d, \left| \frac{|D_{s\sigma}(d)|}{|D_{s\sigma}|} - \frac{|D_{s'}(d)|}{|D_{s'}|} \right| \le \mu/2.$$

Then by the claim and the triangle inequality we have

$$\left| \frac{\hat{D}_{s\sigma}(d)}{|\hat{D}_{s\sigma}|} - \frac{\hat{D}_{s'}(d)}{|\hat{D}_{s'}|} \right| < \mu.$$

Under the assumption that any two states in the target machine are μ-distinguishable, we conclude that $s\sigma$ and s' indeed reach the same state in the target machine.

Similarly, suppose that we decide to promote $s\sigma$ to be a new safe state. This is because for every s' there is some d such that

$$\left| \frac{|D_{s\sigma}(d)|}{|D_{s\sigma}|} - \frac{|D_{s'}(d)|}{|D_{s'}|} \right| > \mu/2.$$

Then by the claim and the triangle inequality we have

$$\left| \frac{\hat{D}_{s\sigma}(d)}{|\hat{D}_{s\sigma}|} - \frac{\hat{D}_{s'}(d)}{|\hat{D}_{s'}|} \right| > 0.$$

So, assuming μ-distinguishability, we know that $s\sigma$ reaches a state not reached by any safe s' in the target machine.

Finally with these claims one can make the following argument: suppose that every state in the target machine can be reached by some path containing only transitions of probability $\ge p$. Then, every candidate state will be either promoted or merged correctly in time T, where T is given by Theorem 2. Therefore, by time at most $n \cdot T$, no candidate state is left and the graph constructed is isomorphic to the graph of the target machine.

In other words, if any candidate states remain after time $n \cdot T$, they should have probability less than p. Thus we can show that for sufficiently low p, these nonfrequent states can be ignored without introducing large errors.

Finally, putting all these steps together, we obtain the following result:

Theorem 4. *For every PDFA M with n states, with distinguishabilty* $\mu > 0$, *such that the expected length of the string generated from every state is less than L, for any* $\delta > 0$ *and* $\varepsilon > 0$, *the PDFA-Learn algorithm will output a hypothesis PDFA M′ such that, with probability greater than* $1-\delta$, *the maximum difference in the probability assigned by the PDFA to any string is at most* ε.

Using the previous result on approximating PNFAs with PDFAs, and the fact that HMMs can be mapped to PNFAs, we now have a PAC-learning algorithm which will enable us to learn a good approximation of an HMM.

6 Illustration

We consider a few examples to illustrate the empirical behaviour of the algorithm. Consider first a synthetic text generator with a simple alphabet $\Sigma = \{a, b, \#\}$, which is designed to generate only three words $d = \{abb, aaa, bba\}$ and where # indicates word termination. We can make a generative model for this text generator using an HMM as shown in Figure 1. All observations are deterministic, transitions are also deterministic, except from $s10$, and the initial state distribution is the same as transitions from $s10$.

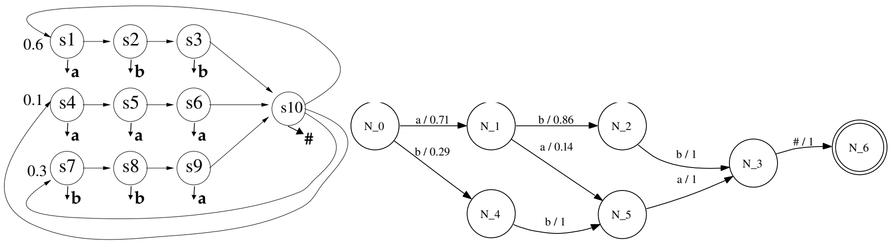


Fig. 1. A simple text-generation HMM (left) and the learned model (right)

We generate a number of trajectories from this HMM and apply the algorithm presented in Section 4 (using $\delta = 0.05$, $n = 8$, $\mu = 0.1$). The right panel in Figure 1 shows the model that is learned. Nodes represent safe states. Edges are annotated by an observation and its probability (zero probability observations are not shown).

We now modify the HMM to produce noisy observations and repeat the experiment. We assume each state in Figure 1 generates the character shown with $P = 0.9$, and generates the other character (i.e. "b" instead of "a" and vice-versa) with $P = 0.1$. In this case, as shown in Figure 2, our algorithm learns a slightly more complex model to account for the greater number of possible trajectories. It is easy to verify that the models shown in Figures 1 and 2 generate the observation strings with the same probability as the corresponding HMM.

The right panel in Figure 2 shows the bound on the number of samples required as a function of the desired model precision. The increased data requirement with low ε values are natural, since the size of the model must grow to achieve this increased precision. As expected, greater amounts of data are required when learning with noisy observations.

Next, we learn a model for a maze navigation domain called Cheese, illustrated in the left panel of Figure 3. We modify the original problem slightly as follows. We assume the agent randomly starts in states $s5$ or $s7$. We assume a single action *float* which moves the agent to any adjacent cell with equal probability. The task resets whenever the agent gets to $s10$. Observations are generated deterministically and correspond to the number

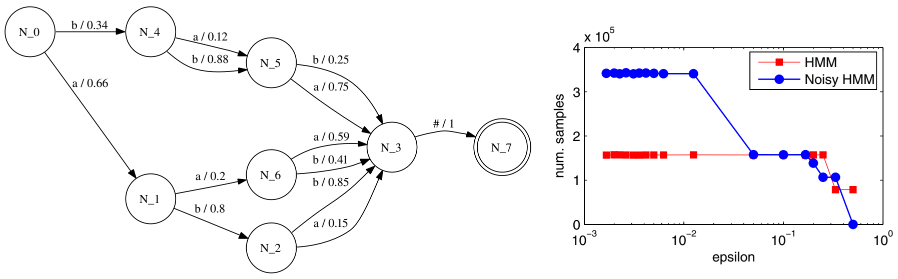


Fig. 2. Learned model with noisy observations (left) and the number of samples predicted by the PAC bounds for achieving the desired model precision (right). The noisy observation case is in blue.

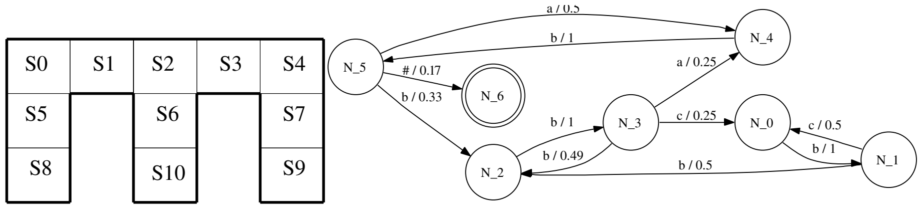


Fig. 3. Cheese maze (left) and the corresponding learned model (right)

of walls in each state ("a"=1 wall, "b"=2 walls, "c"=3 walls), with the exception of $s10$ which produces a distinct terminal observation ("#").

The right panel of Figure 3 shows the results of applying our learning algorithm. It is interesting to note the particular structure learned by our model. The states can be seen to represent core beliefs of the HMM after each *float* action (and before the observation is seen). For example, the states of the learned model in the figure represent the following:

N_0: $Pr(s5) = Pr(s7) = 0.5$;
N_1: $Pr(s8) = Pr(s0) = Pr(s4) = Pr(s9) = 0.25$;
N_2: $Pr(s5) = Pr(s1) = Pr(s3) = Pr(s7) = 0.25$;
N_3: $Pr(s8) = Pr(s9) = 0.125$, $Pr(s0) = Pr(s4) = Pr(s2) = 0.25$;
N_4: $Pr(s1) = Pr(s3) = Pr(s6) = 0.333$;
N_5: $Pr(s0) = Pr(s4) = Pr(s10) = 0.0833$, $Pr(s2) = 0.5$;
N_6: end of trajectory.

This confirms that the graph learned is not arbitrary and has a nice structural interpretation.

7 Discussion and Future Work

There is considerable literature devoted to learning for all of the models introduced above. In HMMs, most of the existing work uses expectation maximization, like the ones described in (Rabiner, 1989) . In these algorithms, the number of hidden states is assumed known. The algorithm starts with a guess about the parameters of the model and modifies this guess in such a way as to improve the likelihood of the observed data.

Several papers have looked at removing assumptions about the model topology. State splitting/merging approaches exist for learning PDFAs (Carrasco and Oncina, 1994; Ron et al, 2005; Thollard et al, 2000) and HMMs (Stolcke et al, 1992, Ostendorf et al, 1997). However the criterion for splitting/merging is typically heuristic or Bayesian in nature and does not provide correcteness guarantees.

More recent procedures rely on finding a minimal linear basis for the space of possible trajectories, by using techniques similar to singular value decomposition or principal component analysis (Jaeger et al, 2006,Singh et al, 2003, Rosencrantx et al, 2004). These procedures aim to find a globally or locally optimal solution in the sense of the L_2 norm. Usually, very large amounts of data are required for a good solution, and no PAC-style guarantees exist yet. A procedure very similar to the one we propose has been devised very recently by Holmes and Isbell (2006), but only for deterministic systems. In the future, we will explore more the connections with their work.

To summarize, we developed an algorithm that learns a PDFA that approximates an HMM. The algorithm addresses the problem of joint topology and parameter inference in Markov models with hidden state. We provided improved theoretical guarantees for PAC-learning of PDFAs from data, and described a natural extension to learning HMMs and POMDPs. This paper highlights important connections between the literature on learning automata and the problem of HMM and POMDP learning. Preliminary empirical results suggest that the algorithm learns correct models for simple HMMs. Further experiments will be conducted to better investigate generality and scalability of the approach.

Acknowledgements

Ricard Gavaldà was supported in part the EU PASCAL Network of Excellence, IST-2002-506778, and by MOISES-TA, TIN2005-08832-C03. Philipp Keller, Joelle Pineau, and Doina Precup were supported in part by grants from NSERC and CFI.

References

Carrasco, R. and Oncina, J. "Learning stochastic regular grammars by means of a state merging method". LNAI 862. 1994.

Clark, A. and Thollard, F. "PAC-learnability of Probabilistic Deterministic Finite State Automata". Journal of Machine Learning Research, 5. 2004.

Dupont, P., Denis, F. and Esposito, Y. "Links between Probabilistic Automata and Hidden Markov Models". Pattern Recognition, 38 (9). 2005.

Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.J. "Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids". Cambridge University Press. 1998.

Holmes, M. and Isbell, C. "Looping Suffix Tree-Based Inference of Partially Observable Hidden State ". In *Proceedings of ICML*. 2006.

Jaeger, H., Zhao, M. and Kolling, A. "Efficient estimation of OOMs". In *Proceedings of NIPS*. 2005.

Kearns, M., Mansour, Y., Ron, D., Rubinfeld, R., Schapire, R. E. and Sellie, L. "On the learnability of discrete distributions". ACM Symposium on the Theory of Computing. 1995.
Lipton, R.J. and Naughton, J.F. "Query size estimation by adaptive sampling", J. Computer and System Sciences 51, 1995, 18–25.
Ostendorf, M. and Singer, H. "HMM topology design using maximum likelihood successive state splitting". Computer Speech and Language, 11. 1997.
Rabiner, L. R. "A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". Proceedings of the IEEE, 77(2). 1989.
Ron, D., Singer, Y. and Tishby, N. "On the learnability and usage of acyclic probabilistic finite automata". In *Proceedings of COLT*,1995.
Rosencrantz, M., Gordon, G. and Thrun, S. "Learning Low Dimensional Predictive Representations". In *Proceedings of ICML*, 2004.
Singh, S., Littman, M. L., Jong, N. K., Pardoe, D. and Stone, P. "Learning Predictive State Representations". In *Proceedings of ICML*, 2003.
Stolcke, A. and Omohundro, S. M. "Hidden Markov Model Induction by Bayesian Model Merging". In *Proceedings of NIPS*, 1993.
Thollard, F., Dupont, P. and Higuera, C. de la. "Probabilistic DFA Inference using Kullback-Leibler Divergence and Minimality". In *Proceedings of ICML*, 2000.
Valiant, L. "A theory of the learnable" Communications of the ACM, 27(11). 1984.

Appendix A: A Termination Condition for On-Line Learning

Suppose that the target model M^* would not only let us sample trajectories d, but also provide their true probability of occurring, $p_{M^*}(d)$. We can let the PDFA construction algorithm proceed until the distance between the target model M^* and the current model M is estimated to be less than a desired error parameter ε. Clearly, this step, and hence the running time of the algorithm, depend on the chosen notion of distance.

We propose the following test, based on the L_∞ distance. For a suitably defined B, draw B trajectories from M^*, and obtain their probabilities, $p_{M^*}(d)$. For every trajectory d, compute its probability using the learned model so far, $p_M(d)$. If there is some d such that $|p_M(d) - p_{M^*}(d)| \geq \varepsilon$, consider that $L_\infty(M,M^*) \geq \varepsilon$ and let the learning continue. Otherwise, consider that $L_\infty(M,M^*) \leq \varepsilon$ and terminate.

We set $B = \frac{3}{(\varepsilon/4)^2} \cdot \ln \frac{8}{\delta\varepsilon}$. We will now show that this test gives the correct answer whenever $L_\infty(M,M^*) \leq \varepsilon/2$ or $L_\infty(M,M^*) \geq 3\varepsilon/2$, i.e., when the L_∞ is at a certain distance from ε either way.

Claim. Let D_1 and D_2 be the two probability distributions to which the test is applied. With probability $1-\delta$, if $L_\infty(D_1,D_2) \geq 3\varepsilon/2$ then the test above says "distance greater than ε", and if if $L_\infty(D_1,D_2) \leq \varepsilon/2$ it says 'distance less than ε"

The proof is easy and omitted in this version. It can be furthermore shown as in Clark and Thollard (2004) that the distance between hypothesis and target machines will be below ε in a number of steps polynomial in the parameters: $1/\varepsilon$, $1/\mu$, $\ln(1/\delta)$, n, as well as the expected length of strings generated at any step, L.

A Discriminative Approach for the Retrieval of Images from Text Queries

David Grangier[1,2], Florent Monay[1,2], and Samy Bengio[1]

[1] IDIAP Research Institute, Martigny, Switzerland,
firstname.lastname@idiap.ch,
[2] Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Abstract. This work proposes a new approach to the retrieval of images from text queries. Contrasting with previous work, this method relies on a discriminative model: the parameters are selected in order to minimize a loss related to the ranking performance of the model, i.e. its ability to rank the relevant pictures above the non-relevant ones when given a text query. In order to minimize this loss, we introduce an adaptation of the recently proposed *Passive-Aggressive* algorithm. The generalization performance of this approach is then compared with alternative models over the *Corel* dataset. These experiments show that our method outperforms the current state-of-the-art approaches, e.g. the average precision over *Corel* test data is 21.6% for our model versus 16.7% for the best alternative, Probabilistic Latent Semantic Analysis.

1 Introduction

Several organizations, such as advertising companies or publishers, need tools to efficiently access and organize large collections of pictures. For instance, Getty Images proposes to its customers to browse and search more than 30 million pictures. This paper focuses on one of the tools needed by such organizations: a system that retrieves pictures from text queries. Given a picture collection P and a text query q, the goal of such a system is to rank the pictures of P such that the pictures relevant to q appear above the others. In order to perform such a ranking, a scoring function F which assigns a real value $F(q, p)$ to any picture/query pair (p, q) is used: given a query q, the pictures of P are ranked by decreasing scores.

In the ideal case, such a function F would always rank relevant pictures above non-relevant ones, i.e. F would satisfy,

$$\forall q, \forall p^+ \in R(q), \forall p^- \notin R(q), F(q, p^+) - F(q, p^-) > 0, \tag{1}$$

where $R(q)$ is the set of pictures relevant to query q.

In the following, we propose a discriminative approach to identify a scoring function close to this ideal property, relying on a set of training data D_{train}. For that purpose, we first introduce a parameterized function F_w and a loss $L(F_w, D_{train})$ related to (1). The *Passive-Aggressive* algorithm [1] is then adopted to identify the parameter vector w^* which minimizes $w \rightarrow L(F_w, D_{train})$.

This model is referred to as Passive-Aggressive Model for Image Retrieval (PAMIR) in the following.

The proposed model contrasts with previous approaches that mostly rely on generative models and likelihood maximization [2,3,4], see Section 4. In fact, the optimization of a loss related to the final retrieval performance is a key aspect of PAMIR and our experiments over the *Corel* data show the advantage of this discrimative approach (see Section 5). PAMIR is reported to outperform several models, such as Cross Media Relevance Model, CMRM [3], Cross Media Translation Table, CMTT [5], or Probabilistic Latent Semantic Analysis, PLSA [4] for various feature extraction setups. For instance, when the *SIFT* features are employed (see Section 3), PAMIR yields 16.0% average precision which should be compared to 12.3% for PLSA, the best alternative (see Section 5).

The remainder of this paper is organized as follows: Section 2 introduces PAMIR, Section 3 presents the features extracted to represent texts and pictures, Section 4 briefly describes the related work and Section 5 reports the experiments and results. Finally, Section 6 draws some conclusions.

2 The PAMIR Model

In this section, we first introduce the notation used, we then describe the parameterization of F_w and the loss $L(\cdot,\cdot)$, we finally explain how the *Passive-Aggressive* learning algorithm is applied.

2.1 Notation

In this problem, we face two types of data: pictures and texts. Both of them are represented as vectors. The picture vector space is referred to as $\mathcal{P}$ while the text vector space is referred to as $\mathcal{T}$. It should further be added that $\mathcal{T}$ is a subset of $\mathbb{R}^T$, where T is the vocabulary size. The i^{th} component of a vector $t \in \mathcal{T}$ is referred to as the weight of term i in text t. A detailed description of both text and picture representations is given in Section 3.

2.2 Model Parameterization

The parameterization of PAMIR is inspired by approaches developed for text retrieval, i.e. the task of retrieving *text* documents from *text* queries. In this case, documents are generally ranked with respect to their inner product with the submitted query [6]. In other words, the scoring function is

$$F^{text} : \mathcal{T} \times \mathcal{T} \rightarrow \mathbb{R}, \text{ where } F^{text}(q,d) = \sum_{i=1}^{T} q_i \cdot d_i.$$

We would like to adopt a similar approach to assign a score $F(q,p)$ to any pair (q,p) consisting of a text query $q \in \mathcal{T}$ and a picture $p \in \mathcal{P}$. For that purpose, we first introduce a mapping $f_w : \mathcal{P} \rightarrow \mathcal{T}$ that assigns a text vector $f_w(p) \in \mathcal{T}$ to any picture $p \in \mathcal{P}$ and we then compute the score of any query/picture pair (q,p) as,

$$F_w(q,p) = F^{text}(q, f_w(p)).$$

In the following, we restrict ourselves to mappings f_w of the form,

$$f_w : \mathcal{P} \rightarrow \mathbb{R}^T \text{ where } f_w(p) = (w_1 \cdot p, \ldots, w_T \cdot p)$$

and $w = (w_1, \ldots, w_T) \in \mathcal{P}^T$.

2.3 Ranking Loss

As mentioned in the introduction, we would ideally like to identify the parameters w such that F_w verifies all constraints in (1). However, we are only given a finite training set,

$$D_{train} = ((q_1, p_1^+, p_1^-), \ldots, (q_n, p_n^+, p_n^-)),$$

where for all k, q_k is a text query (i.e. $q_k \in \mathcal{T}$), p_k^+ is a picture relevant to q_k (i.e. $p_k^+ \in R(q_k)$) and p_k^- is a picture non-relevant to q_k (i.e. $p_k^- \notin R(q_k)$). Hence, we would like to select w relying on D_{train} data such that F_w ensures good generalization performance. In other words, w should be chosen such that F_w is likely to satisfy the constraints (1) for unseen data. For that purpose, a first approach would be to identify F_w such that all training constraints are satisfied, i.e.

$$\forall k, \quad F_w(q_k, p_k^+) - F_w(q_k, p_k^-) > 0. \tag{2}$$

However, to ensure better generalization, we propose to select w such that,

$$\forall k, \quad F_w(q_k, p_k^+) - F_w(q_k, p_k^-) \geq \epsilon$$

where $\epsilon > 0$. This equation can then be rewritten as,

$$\forall k, \quad l(w; (q_k, p_k^+, p_k^-)) = 0,$$
$$\text{where } \; l(w; (q_k, p_k^+, p_k^-)) = \max\left\{0, \epsilon - F_w(q_k, p_k^+) + F_w(q_k, p_k^-)\right\}.$$

This means that for all k, we would like the score $F_w(q_k, p_k^+)$ to be greater than $F_w(q_k, p_k^-)$ by at least a *margin* of ϵ (in the following, we arbitrarily set $\epsilon = 1$ since any positive value would lead to the same optimization problem). This *margin* criterion is inspired from the ranking SVM approach, which has successfully been applied to text retrieval [7]. Our model is however different from ranking SVM in both its parameterization and its optimization procedure [1]. In fact, we use the online Passive-Aggressive minimization algorithm which does not rely on quadratic optimization like ranking SVM, allowing PAMIR to scale to large constraint sets (e.g. there are $\sim 10^8$ constraint triplets in the training data presented in Section 5).

2.4 Training Procedure

Our goal is to minimize the loss

$$L(w; D_{train}) = \sum_{k=1}^{n} l(w; (q_k, p_k^+, p_k^-)). \tag{3}$$

For that purpose, we adapt the *Passive-Aggressive* (PA) algorithm, originally introduced for classification and regression problems [1], to minimize this retrieval loss. For this minimization, the algorithm constructs a sequence of weight vectors $(w^0, \ldots, w^m)$ according to the following iterative procedure: the first vector is set to be zero, $w^0 = 0$ and, at the i^{th} iteration, the weight w^i is selected according to the i^{th} training example and the previous weight w^{i-1},

$$w^i = \operatorname*{argmin}_w \frac{1}{2}\|w - w^{i-1}\|^2 + C \cdot l(w; (q_i, p_i^+, p_i^-)). \tag{4}$$

This means that, at each iteration, we select the weight w^i as a trade-off between minimizing the loss on the current example $l(w; (q_i, p_i^+, p_i^-))$ and remaining close to the previous weight vector w^{i-1}. The *aggressiveness* parameter C controls this trade-off. Adopting an approach similar to [1], it can be shown that the solution of problem (4) is

$$\begin{aligned} w^i &= w^{i-1} + \tau_i v_i, \\ \text{where} \quad \tau_i &= \min\left\{C, \frac{l(w^{i-1}; (q_i, p_i^+, p_i^-))}{\|v_i\|^2}\right\} \\ \text{and} \quad v_i &= -(q_1(p_k^+ - p_k^-), \ldots, q_T(p_k^+ - p_k^-)). \end{aligned}$$

At the end of the iterative process, the best weight among $\{w^0, \ldots, w^m\}$ is selected according to some validation data D_{valid}, i.e.

$$w = \operatorname*{argmin}_{w \in \{w^0, \ldots, w^m\}} L(w; D_{valid}).$$

The hyperparameter C has also been selected to maximize the performance over D_{valid}. The proof that the above procedure actually minimizes the loss (3) is not included here due to space constraint but can easily be inferred from the proof given in [1].

3 Text and Picture Representations

This section describes the representations used for text and pictures.

3.1 Text Representation

As mentioned before, textual data are represented with vocabulary-sized vectors, e.g. a query q will be assigned the vector

$$q = (q_1, \ldots, q_T),$$

where q_i is the weight of term i in the query q and T is the vocabulary size. This type of vector is often referred to as *bag-of-words* vector since this representation does not take word ordering into account. In our case, the term weights correspond to the popular $tf \cdot idf$ representation with Euclidean normalization [6], i.e. given $t \in \mathcal{T}$,

$$t_i = \frac{tf_{i,t} \cdot idf_i}{\sqrt{\sum_{j=1}^{T} (tf_{j,t} \cdot idf_i)^2}}$$

where the term frequency $tf_{i,t}$ corresponds to the number of occurrences of term i in t and the inverse document frequency idf_i is defined as $idf_i = -log(r_i)$, r_i being the fraction of training picture captions containing term i. It should be noted that the definition of *idf* assumes that the training pictures are labeled with a caption. This is the case for the *Corel* data used in our experiments (see Section 5). However, were such captions to be unavailable, it would still be possible to compute *idf* relying on another textual corpus, such as an encyclopedia.

3.2 Picture Representation

Similarly to previous work (See Section 4), the *visterm* approach has been used for picture representation. The main idea of this approach is to define different classes of image regions, referred to as the *visual vocabulary*, which then allows the representation of each picture p as a histogram over this vocabulary. In practice, vocabulary definition is performed automatically through the following 3-step process: first, regions of interests are detected from each training picture; second, each extracted region is assigned a vector describing its visual properties; third, the vocabulary is built through k-means clustering of the training region descriptors. Finally, any picture p (either from train or test set) is assigned the histogram,

$$p = (vtf_{p,1}, \ldots, vtf_{p,V}), \tag{5}$$

where V is the visual vocabulary size and $vtf_{p,i}$ is the number of regions of p that belongs to the i^{th} visual vocabulary cluster. In our case, we used two types of visterms, either individually or jointly.

Blobs describes the visual properties of large, color-homogeneous regions. In this case, region detection is performed with a normalized cut algorithm and the region descriptors are 36-dimensional vectors summarizing color (18), texture (12) and shape (6) information of the region, see [8].

SIFTs describes edge properties of areas around salient points of the picture. In this case, region detection is performed with a difference-of-Gaussian detector and region descriptors consist of edge histograms, see [9].

Blob+SIFT visterms have also been combined through the concatenation of their histograms.

Like for the text features, we also applied the normalized $tf \cdot idf$ weighting to visterm histograms, i.e. each picture p is represented with:

$$p = (p_1, \ldots, p_V), \text{ where } p_i = \frac{vtf_{p,i} \cdot vidf_i}{\sqrt{\sum_{j=1}^{V} vtf_{p,i} \cdot vidf_j}} \tag{6}$$

where $vidf_i = -log(vr_i)$ with vr_i referring to the fraction of training pictures containing at least one region mapped to the i^{th} cluster. Space limitation prevents us from reporting the results of the experiments over validation data that concluded on the superiority of this weighting compared to (5).

4 Related Work

The previous work in image retrieval from text queries mainly focused on an intermediate step, *image auto-captioning.* This task consists in estimating the likelihood of a textual annotation, or caption, given an unannotated picture. Given a query q, such a model then allows the user to retrieve the pictures for which q is the most likely. In this context, several models such as Cross-Media Relevance Models (CMRM) [3], Probabilistic Latent Semantic Analysis (PLSA) [4] or Latent Dirichlet Allocation (LDA) [2] have been proposed. These model hence learn a captioning model from a set of training picture that have been manually annotated. Even if such approaches are leading to state-of-the-art performance, it could seem questionable to focus on an intermediate annotation problem when the final goal is to solve a retrieval problem. It would be more appropriate to adopt a discriminative approach and directly optimize a loss related to the retrieval performance of the model. However, to the best of our knowledge, no discriminative approaches have been proposed in the context of image retrieval prior to this work. Previous discriminative approaches have only focussed on categorization ranking problems (e.g. [10,11]), i.e. the task of ranking unseen pictures with respect to queries or categories *known* at training time. This task is hence different from a true retrieval task in which a *new* query (i.e. any set of vocabulary words) can be submitted.

In absence of discriminative alternatives, this section will therefore focus on the non-discriminative approaches that have shown to be the most effective over the benchmark *Corel* dataset: Cross-Media Relevance Model (CMRM) [3], Cross-Media Translation Table (CMTT) [5] and Probabilistic Latent Semantic Analysis (PLSA) [4]. The proposed PAMIR approach will then be compared to these models in Section 5.

4.1 Cross-Media Relevance Model

In order to estimate the probability of a term t given a picture p^{test}, $P(t|p^{test})$, CMRM [3] estimates the joint probability $P(t, p^{test})$ and then relies on Bayes rule. The joint probability $P(t, p^{test})$ is estimated as its expectation over the training pictures,

$$P(t, p^{test}) = \sum_{p^{train} \in D_{train}} P(p^{train}) \cdot P(t, p^{test}|p^{train}).$$

The picture p^{test} is considered as a set of discrete features or visterms (see Section 3), i.e. $p^{test} = \{v_1, \ldots, v_m\}$, which means that:

$$P(t, p^{test}) = \sum_{p^{train} \in D_{train}} P(p^{train}) \cdot P(t, v_1, \ldots, v_m|p^{train}).$$

Terms and visterms are then assumed to be independent given a training picture, leading to:

$$P(t, p^{test}) = \sum_{p^{train} \in D_{train}} P(p^{train}) \cdot P(t|p^{train}) \prod_{i=1}^{m} P(v_i|p^{train})$$

The probabilities $P(t|p^{train})$ and $P(v_i|p^{train})$ are then estimated through maximum likelihood estimates, smoothed with the *Jelinek-Mercer* method. Although simple, this approach has shown to yield good performance over the standard *Corel* dataset [3].

4.2 Cross-Media Translation Table

The CMTT model borrows its parameterization from cross-lingual retrieval techniques [5]. In this case, textual terms and visterms are considered as words originating from two different languages and CMTT constructs a translation table containing the similarities $sim(t, v)$ between any pair of term/visterm (t, v). This translation table is then used to estimates $p(t|p^{test})$ for any term t and any picture:

$$P(t|p^{test}) = \frac{w_{t,p_{test}}}{\sum_{i=1}^{T} w_{i,p_{test}}}, \text{ where } w_{t,p_{test}} = \sum_{i=1}^{m} sim(t, v_i),$$

$v_1, \ldots, v_m$ being the visterms of p^{test}. The translation table is computed from the training data D_{train} according to the following process: in a first step, each term i and each visterm j is represented by a $|D_{train}|$ dimensional vector, t_i or v_j, in which each component k is the weight of term i (or visterm j) in the k^{th} training example (the weighting scheme used here is $tf \cdot idf$, as defined in Section 3). As a noise removal step, the matrix $M = [t_1, \ldots, t_T, v_1, \ldots, v_V]$ containing all term and visterm vectors is approximated with a lower rank matrix, $M' = [t'_1, \ldots, t'_T, v'_1, \ldots, v'_K]$, through Singular Value Decomposition (SVD). The similarity $sim(i, j)$ between a term i and a visterm j is then defined as

$$sim(i, j) = \frac{\cos(t'_i, v'_j)}{\sum_{k=1}^{V} \cos(t'_i, v'_k)}.$$

Like CMRM, this method has also been evaluated over the *Corel* corpus [5], where it has shown to be effective. The use of SVD has notably shown to improve noise robustness. However, CMTT has also some limitations, the main one being that cosine similarity only allows to model simple relationships between terms and visual features. In order to circumvent this problem, approaches allowing to model more complex relationships, such as Probabilistic Latent Semantic Analysis [4], have been applied.

4.3 Probabilistic Latent Semantic Analysis

PLSA, introduced for text retrieval [12], has recently been applied to image retrieval [4]. This model assumes that the observation of a picture p and a term t in its caption are independent conditionally to a discrete latent variable $z_k = \{z_1, \ldots, z_K\}$,

$$P(p, t) = P(p) \sum_{k=1}^{K} P(z_k|p) P(t|z_k), \tag{7}$$

where K is a hyperparameter of the model. A similar conditional independence assumption is also made for visterms,

$$P(p, v) = P(p) \sum_{k=1}^{K} P(z_k|p) P(v|z_k).$$

model, i.e. $P(z_k|p)$, $P(t|z_k)$, $P(v|z_k)$ are trained through the Expectation Maximization (EM) algorithm. In fact, a modified version of EM is applied such that the latent space is constrained toward the text modality. This yields a latent space that better models the semantic relationships between pictures. Once parameter fitting over the set of training pictures is performed, it is still needed to infer $P(z_k|p), \forall k$, for any unnatotated test picture p. This estimation is performed to maximize the test picture likelihood, keeping $P(v|z_k), \forall(v, k)$ to the values estimated during training. After this step, (7) can then be used to infer $P(p, t)$ for any test picture/term pair (p, t). Similarly to CMRM, Bayes rule is applied to compute $P(t|p)$ from $P(p, t)$. This PLSA model has shown to be effective empirically, especially when the latent space is constraint toward the text modality as explained in [4].

5 Experiments and Results

This section presents the experiments performed. The experimental setup is first described and the results are then discussed.

5.1 Experimental Setup

The Corel Dataset[1] consists of photographs of various scenes such as bears in the wilderness, sunsets, air-shows, etc. Each picture is annotated with several keywords describing the main objects depicted. In this work, we used a $5,000$-picture subset of *Corel*. This subset has been defined in [8]: it contains 4500 development pictures (P_{dev}) and 500 test pictures (P_{test}). This split has been widely used in the literature, e.g. [5,4], and has hence become a kind of benchmark to compare image retrieval algorithm. In our case, we further split the development set into a $4,000$-picture train set (P_{train}) and a 500-picture validation set (P_{valid}), which allows us to perform model training and hyperparameter selection on different subsets.

Relevance data has been defined relying on picture captions, as explained in [3]: a picture p is considered as relevant to a query q if and only if its caption contains all the terms of q. The query sets Q_{train}, Q_{valid} and Q_{test} are then defined as the set of all queries which have at least one relevant picture among P_{train}, P_{valid} and P_{test} respectively. The statistics for the three picture/query sets, i.e. $D_{train} = (P_{train}, Q_{train})$, $D_{valid} = (P_{valid}, Q_{valid})$ and $D_{test} = (P_{test}, Q_{test})$ are summarized in Table 1 and Table 2. The PAMIR model has then been trained and evaluated relying on these data with the following setup: parameter fitting has been first performed over D_{train} (i.e. the training criterion is optimized over this set) and the hyperparameters (i.e. the aggressiveness C and the number of iterations m) have been selected relying on D_{valid}. Finally, D_{train} and D_{valid} have been used jointly to re-train the model with its selected hyperparameters. Model evaluation has then been performed over D_{test}, as explained in the next section. The alternative models CMRM, CMTT and

[1] *Corel* data are available at `http://www.emsps.com/photocd/corelcds.htm`.

Table 1. Picture Set Statistics

	P_{train}	P_{valid}	P_{test}
Number of pictures	4,000	500	500
Number of Blob clusters		500	
Avg. # of Blobs per pic.	9.43	9.33	9.37
Number of SIFT clusters		1,000	
Avg. # of SIFTs per pic.	232.8	226.3	229.5

Table 2. Query Set Statistics

	Q_{train}	Q_{valid}	Q_{test}
Number of queries	7,221	1,962	2,241
Avg. # of rel. pic. per q.	5.33	2.44	2.37
Vocabulary size		179	
Avg. # of words per query	2.78	2.51	2.51

PLSA have also been trained and evaluated according to the same setup for the sake of comparison.

Evaluation Methodology. The performance of PAMIR over the test data has been assessed according to standard IR measures [6]. For each test query $q \in Q_{test}$, the pictures of P_{test} have been ranked with respect to $\{F_w(q,p), \forall p \in P_{test}\}$. This ranking is then compared to the ideal case, i.e. the pictures relevant to q appear above the others, according to the following measures:

P10. Precision at top 10 pictures is defined as the percentage $Pr(10)$ of relevant pictures within the top 10 positions of the ranking. This measure hence corresponds to the percentage of relevant material that would appear in the first 10–result page of a search engine. Although it is easy to interpret, this measure tends to overweight queries with a large number of relevant pictures when averaging over a query set. In the case of such queries, it is easier to rank some relevant pictures within the top 10, simply because the relevance set is larger and not because of any property of the ranking approach.

BEP. Break-Even Point evaluates the precision at the top $|R(q)|$ pictures, $|R(q)|$ being the number of relevant pictures for the evaluated query q. This hence corresponds to the percentage $Pr(|R(q)|)$ of relevant documents within top $|R(q)|$. It is also often called R-precision. Contrary to $P10$, this measure does not overweight queries with many relevant pictures.

AvgP. Average Precision is the standard measure used for IR benchmark [6], and it corresponds to the average of the precision at each position where a relevant document appears, i.e. $AvgP = \frac{1}{|R(q)|} \sum_{d \in R(q)} Pr(rk_{d,q})$, where $rk_{d,q}$ is the rank of document d for query q.

The results of PAMIR are then reported according to the average of these measures over the set of test queries Q_{test}. The alternative models (i.e. CMRM, CMTT and PLSA) have also been evaluated according to this methodology. The next section summarizes these results.

Table 3. Average precision (%) for test queries

	CMRM	CMTT	PLSA	PAMIR
Blobs	10.4	11.8	9.7	11.9
SIFTs	10.8	9.1	12.3	**16.0**
Blobs + SIFTs	14.7	11.5	16.7	**21.6**

Table 4. Model hyperparameters

	C	m
Blobs	0.01	$1.75 \cdot 10^6$
SIFTs	0.001	$94.6 \cdot 10^6$
Blobs + SIFTs	0.01	$19.0 \cdot 10^6$

5.2 Experimental Results

Table 3 reports the AvgP results for all visual feature setups (see Section 3) while Table 4 reports the hyperparameters selected for these experiments. In all feature configurations, PAMIR is reported to outperform the other models, e.g. for the combination of *Blob* and *SIFT* features, PAMIR yields 21.6% AvgP which corresponds to a relative improvement of 29% over the second best model (PLSA with 16.7% AvgP). In order to determine whether the PAMIR advantage observed on the average could be due to a few queries, we further compared PAMIR results with those of the alternative approaches for each of the 2,241 queries and performed the Wilcoxon signed rank test [13] over these data. The test rejected this hypothesis with 95% confidence for both *SIFT* and *Blob+SIFT* features (such a test outcome is indicated by bold numbers in the tables). In the case of *Blob* features, the test concluded that PAMIR performance is similar to CMTT but better than the other models. The low number of visterms per picture ($\sim$ 9.5 on average, see Table 1) may explain the relatively good results of CMTT in the case of *Blobs*: we hypothesize that such a concise representation may only provide sufficient statistics to train highly constraint models, such as CMTT. On the contrary, the SIFT representation, where richer statistics are available ($\sim$ 230 visterms per picture on average, see Table 1), allows less constraint models, such as PAMIR or PLSA, to reach higher performance than CMTT.

As an alternative to AvgP, we also looked at the performance in terms of P10 and BEP, as explained in the previous section. Table 5 reports these results for the *Blob+SIFT* features[2]. These measurements confirm the superiority of PAMIR: for all measures, PAMIR yields significantly better results when compared to any alternative model among CMRM, CMTT and PLSA. Looking closely at Table 5, one could remark that the P10 values reported are quite low, e.g. only 0.88 relevant picture within top 10 for PAMIR. These low values should however not be regarded as a failure of the models since the very low number of relevant pictures per query should also be considered (see Table 2). In fact, P10 cannot be higher than 20.2% for our Q_{test} set.

[2] We do not report the measurements for Blobs and SIFTs individually due to space limitation.

Table 5. Average precision, break even point and precision at top 10 (%) over test queries (Q_{test}) for *Blob + SIFT* features

	CMRM	CMTT	PLSA	PAMIR
AvgP	14.7	11.5	16.7	**21.6**
BEP	10.5	5.9	10.5	**13.4**
P10	5.8	5.5	7.1	**8.8**

Table 6. Average precision, break even point and precision at top 10 (%) over *single-word* test queries for *Blob + SIFT* features

	CMRM	CMTT	PLSA	PAMIR
AvgP	19.2	19.1	24.5	**30.7**
BEP	19.7	17.4	22.2	**27.2**
P10	17.8	17.9	21.3	**25.3**

Since several previous papers only reported results over single word queries (e.g. [5,4]), we also performed a set of experiments over this type of query. For that purpose, PAMIR has been trained and evaluated relying on the subsets of Q_{train}, Q_{valid} and Q_{test} containing only single word queries. These queries correspond to a more restrictive scenario, i.e. the users are not given the possibility to submit multiple-word queries. Moreover, single-word queries generally have more relevant pictures than multiple-word queries, which makes the retrieval task easier (in our test data, each single-word query has 9.3 relevant pictures on average, compared to 2.4 for the whole query set). Table 6 reports the results of the experiments over single-word queries for the best feature configuration, i.e. *Blobs+SIFTs*. In this case, PAMIR outperforms the alternative approaches for all measures, this improvement being significant according to the Wilcoxon test at the 95% confidence level. The use of PAMIR is hence advantageous over the alternative models in both the case where the users focus on the first ranking positions (as shown by P10 results) and the case where the users are interested in the whole ranking (as shown by AvgP results).

The overall outcome of these experiments is hence positive, underscoring the benefit of using a discriminative approach to the problem of image retrieval from text queries.

6 Conclusions

In this paper, we proposed a discriminative approach to the retrieval of images from text queries. After introducing the model parameterization, we presented a margin loss adapted to this retrieval task. We then proposed an adaptation of the *Passive-Aggressive* algorithm [1] to identify the model parameters which minimize this loss.

Our model, PAMIR, has then been evaluated over the *Corel* dataset. These experiments have been performed relying on different visual features that describe color-homogeneous regions or salient points of the images. The results have then been compared to those of state-of-the-art approaches, which rely on non-discriminantive models. It has been observed that PAMIR outperforms the alternative approaches for most queries, e.g. for the most effective visual features, *Blobs+SIFTs*, the reported AvgP for PAMIR is 21.6% which should be compared to 16.7% for PLSA, the second best model.

The results of PAMIR are hence promising and need to be confirmed over other datasets. Furthermore, it would also be of a great interest to investigate on the use of non-linear kernels in PAMIR. In this work, we relied on the linear kernel over feature histograms to compare images. However, like any *Passive-Agressive* model [1], PAMIR could benefit from other Mercer kernels. In particular, recently proposed image kernels, such as [14], could be effective for our task.

Acknowledgments. This work has been performed with the support of the Swiss NSF through the MULTI project. It was also supported by the PASCAL European Network of Excellence, funded by the Swiss SER.

References

1. Crammer, K., Dekel, O., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. In: Neural Information Processing Systems (NIPS). (2003)
2. Barnard, K., Duygulu, P., Forsyth, D., de Freitas, N., Blei, D.M., Jordan, M.I.: Matching words and pictures. Journal of Machine Learning Research (JMLR) **3** (2003) 1107–1135
3. Jeon, J., Lavrenko, V., Manmatha, R.: Automatic image annotation and retrieval using cross-media relevance models. In: ACM Special Interest Group on Information Retrieval (SIGIR). (2003)
4. Monay, F., Gatica-Perez, D.: PLSA-based image auto-annotation: constraining the latent space. In: ACM Multimedia. (2004) 348–351
5. Pan, J.Y., Yang, H.J., Duygulu, P., Faloutsos, C.: Automatic image captioning. In: International Conference on Multimedia and Expo (ICME). (2004) 1987–1990
6. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley, Harlow, England (1999)
7. Joachims, T.: Optimizing search engines using clickthrough data. In: International Conference on Knowledge Discovery and Data Mining (KDD). (2002)
8. Duygulu, P., Barnard, K., de Freitas, N., Forsyth, D.: Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In: European Conference on Computer Vision (ECCV). (2002) 97–112
9. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision (IJCV) **60**(2) (2004) 91–110
10. Tieu, K., Viola, P.: Boosting image retrieval. International Journal of Computer Vision (IJCV) **56**(1) (2004) 17 – 36
11. Wu, H., LuE, H., Ma, S.: A practical SVM-based algorithm for ordinal regression in image retrieval. In: ACM Multimedia. (2003)
12. Hofmann, T.: Unsupervised learning by probabilistic latent semantic analysis. Machine Learning **42** (2001) 177–196
13. Rice, J.: Rice, Mathematical Statistics and Data Analysis. Duxbury Press (1995)
14. Wallraven, C., Caputo, B.: Recognition with local features: the kernel recipe. In: International Conference on Computer Vision (ICCV). (2003)

TildeCRF: Conditional Random Fields for Logical Sequences

Bernd Gutmann and Kristian Kersting

University of Freiburg, Institute for Computer Science, Machine Learning Lab,
Georges-Koehler-Allee, Building 079, 79110 Freiburg, Germany
{bgutmann, kersting}@informatik.uni-freiburg.de

Abstract. Conditional Random Fields (CRFs) provide a powerful instrument for labeling sequences. So far, however, CRFs have only been considered for labeling sequences over flat alphabets. In this paper, we describe TildeCRF, the first method for training CRFs on logical sequences, i.e., sequences over an alphabet of logical atoms. TildeCRF's key idea is to use relational regression trees in Dietterich et al.'s gradient tree boosting approach. Thus, the CRF potential functions are represented as weighted sums of relational regression trees. Experiments show a significant improvement over established results achieved with hidden Markov models and Fisher kernels for logical sequences.

1 Introduction

Sequential data are ubiquitous and are of interest to many communities. Such data can be found in virtually all application areas of machine learning including computational biology, user modeling, speech recognition, empirical natural language processing, activity recognition, information extractions, etc. Therefore, it is not surprising that sequential data has been the subject of active research for decades. One of the many problems investigated concerns assigning labels to sequences of objects. For example, in protein secondary structure prediction, the task is to assign a secondary structure class to each amino acid residue in the protein sequence [13]. Dietterich *et al.* [4] have called this general class of problems *sequential supervised learning* (SSL), which can be formalized as follows.

Definition 1 (Sequential Supervised Learning). Given *a finite set of training examples of the form* $\{(X_i, Y_i)\}_{i=1}^m$*, where each* X_i *is a sequence* $\langle x_{i,j}\rangle_{j=1}^{T_i} \in \bigotimes \mathcal{X}$ *of elements in the input space* $\mathcal{X}$ *and each* Y_i *is the corresponding sequence* $\langle y_{i,j}\rangle_{j=1}^{T_i} \in \bigotimes \mathcal{Y}$ *of elements in the output space* $\mathcal{Y}$*,* ***find*** *a function* $H : \otimes\mathcal{X} \to \otimes\mathcal{Y}$ *with low approximation error on the training data as well as on unseen examples.*

One appealing approach to SSL are probabilistic sequence models as they take uncertainty into account explicitly. A probabilistic sequence model assumes the X_i's and Y_i's to be sampled from some random variables X and Y and attempt to learn the statistical dependency $P(X, Y)$ between them. Hidden Markov models (HMMs) [14] are among the most popular probabilistic sequence models. An

HMM models a sequence X_i by assuming that there is an underlying sequence of states Y_i drawn from a finite set of states S. To model the joint distribution $P(X, Y)$ tractably, HMMs make two independency assumptions: each state depends only on its immediate predecessor and each observation sequence $x_{i,j}$ depends only on the current state $y_{i,j}$. Given this, it is relatively straightforward to estimate their parameters. Furthermore, HMMs are relatively easy to understand by humans. Despite of their success, HMMs have two major weaknesses:

(A) they are able to only handle sequences over flat alphabets, and
(B) it is cumbersome to model arbitrary dependencies in the input space.

To overcome **(A)**, *logical hidden Markov models* (LoHMMs) [6] have recently been introduced as an extension of HMMs. They allow for *logical sequences*, i.e., sequences of atoms in a first order logic. In [6], LoHMMs have been applied to the problem of discovering structural signatures of protein folds and led to more compact models. The trained LoHMM consisted of 120 parameters corresponding to an HMM with more than 62000 parameters. However, LoHMMs still suffer from limitation **(B)**, i.e., the difficulty to model arbitrary dependencies in the input space. One way to address this problem is to explicitly model these dependencies by using complex LoHMM structures. Selecting a structure of a LoHMM, however, is a significant problem [8]. Whereas HMMs are commonly learned by estimating the ML parameters of a fixed, fully connected model, this is not feasible for LoHMMs: different abstraction levels have to be explored.

To overcome **(B)**, i.e., to easily model arbitrary dependencies in the input space, conditional random fields [9] (CRFs) have become popular in language processing, computer vision, and information extraction. They have outperformed HMMs on language processing tasks such as information extraction and shallow parsing. CRFs are undirected graphical models that represent the conditional probability distribution $P(Y|X)$. Instead of the generatively trained (Lo)HMM, the discriminatively trained CRF is designed to handle non-independent input features, which can be beneficial in complex domains. For example, we would like to exploit other features of an amino acid, such as its molecular weight or its neighboring words.

Many sequences occurring in real-world problems such as in computational biology, planning, and user modeling, however, exhibit internal structure. The elements of such sequences can be seen as atoms in a relational logic (see e.g. [10] for an introduction to logic). For example, the secondary structure of the Ribosomal protein L4 can be represented as

$$\mathtt{st(null,2),he(h(right,alpha),6),st(plus,2),he(h(right,alpha),4),\ldots}$$

Here, helices of a certain type and length `he`(*HelixType*,*Length*) and strands of a certain orientation and length `st`(*Orientation*,*Length*) are essentially structured symbols, i.e., atoms over logical predicates. The application of traditional CRFs to such sequences requires one to either ignore the structure of helices and strands, which results in a loss of information, or to take all possible combinations (of arguments such as orientation and length) into account, which leads to a combinatorial explosion in the number of parameters.

The main contribution of this paper is TildeCRF, the first method for training CRFs for logical sequences, i.e., sequences over an alphabet of logical atoms. The key idea of TildeCRF is to use relational regression trees in Dietterich *et al.*'s gradient tree boosting approach [4] to make relational abstraction through logical variables and unification. Thus, the TildeCRF potential functions are represented as weighted sums of relational regression trees. Experiments show a significant improvement over previous results achieved with hidden Markov models and Fisher kernels for logical sequences.

The outline of the paper is as follows. After discussing related work, we will briefly review CRFs in Section 3. In Section 4, we devise TildeCRFs for logical sequences. Before concluding, we experimentally evaluate TildeCRF in Section 5.

2 Related Work

CRFs for logical sequences combine two different research directions. On the one hand, they are related to several extensions of HMMs and CRFs. On the other hand, they are related to the recent interest in combining relational learning with probabilistic models such as Markov random fields [3].

In the first type of approaches, the underlying idea is to upgrade HMMs and CRFs to represent more structured state spaces. Sutton and McCallum's Factorial CRFs [17], Quattoni *et al.*'s and Dietterich *et al.*'s tree-shaped CRFs [12,4], and Sutton *et al.*'s dynamic CRFs [18] decompose the state variables into smaller units. The key differences with TildeCRF is that these approaches do not consider learning the features and that they do not employ the logical concepts of variables and unification. Both are essential because variables allows one to group states together and unification allows one to share knowledge between abstract states via abstract transitions. LoHMMs [6] and relational Markov models [1] extend (H)MMs to handle sequences of logical atoms. Consequently, both suffer from the same difficulty to model arbitrary dependencies in the input space.

In the second type of approaches, most attention has been devoted to developing highly expressive formalisms. Taskar *et al.* 's relational Markov networks (RMN) [19] extend Markov random fields by providing a relational language for describing clique structures and enforcing parameter sharing at the template level. RMNs have been applied to computer vision and natural language problems. Domingos and Richardson [15] introduced Markov logic networks (MLNs). MLNs also upgrade Markov random fields to the relational case. In contrast to RMNs, MLNs view logical formulas as soft constraints on the set of possible worlds: if a world violates one formula, it is less probable but not necessarily impossible as in classical logic. This is essentially realized by representing potentials as weighted sets of logical formulas; the weights reflect how strong the constraints are. Both approaches are not specifically designed for analyzing logical sequences. Recently, Shanghei *et al.* [16] introduced dynamic probabilistic relational models. In contrast to TildeCRF, they extend directed dynamic models.

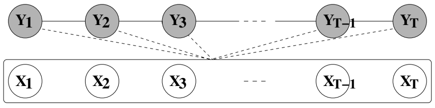


Fig. 1. Graphical representation of linear-chain CRF

TildeCRF can be seen as an attempt towards downgrading such highly expressive frameworks for handling logical sequences.

3 Conditional Random Fields

In recent years, conditional random fields [9] (CRFs) turned out to be a suitable representation for SSL. CRFs are undirected graphical models that encode a conditional probability distribution using a given set of features. CRFs are defined as follows. Let G be an undirected graphical model over sets of random variables X and Y. As a special case, consider a *linear-chain* CRF, that is $X = \langle x_{i,j} \rangle_{j=1}^{T_i}$ and $Y = \langle Y_{i,j} \rangle_{j=1}^{T_i}$, so that Y is a labeling of an observed sequence X. Then, CRFs define the conditional probability of a state sequence given the observed sequence as

$$P(Y|X) = Z(X)^{-1} \exp \sum\nolimits_{t=1}^{T} \Psi_t(y_t, X) + \Psi_{t-1,t}(y_{t-1}, y_t, X).$$

where $\Psi_t(y_t, X)$ and $\Psi_{t-1,t}(y_{t-1}, y_t, X)$ are potential functions and $Z(X)$ is a normalization factor over all state sequences X. A *potential function* is a real-valued function that captures the degree to which the assignment y_t to the output variable fits the transition from y_{t-1} and X. Due to the global normalization by $Z(X)$, each potential has an influence on the overall probability.

To apply CRFs to SSL problems, one must choose a representation for the potentials. Typically, it is assumed that the potentials factorize according to a set of features $\{f_k\}$, which are given and fixed, so that $\Psi(y_t, X) = \sum \alpha_k g_k(y_t, X)$ and $\Psi(y_{t-1}, y_t, X) = \sum \beta_k f_k(y_{t-1}, y_t, X)$ respectively. The model parameters are now a set of real-valued weights α_k, β_k, one weight for each feature. In linear-chain CRF, a first-order Markov assumption is made on the hidden variables. A graphical model for this is shown in Figure 1. In this case, there are features for each label transition. Feature functions can be arbitrary such as a binary test that has value 1 if and only if y_{t-1} has the label a.

4 TildeCRF: CRFs for Logical Sequences

Originally, Lafferty *et al.* introduced CRFs as an essentially propositional representation: symbols used to represent states and outputs are flat. So far, CRFs have not been considered for sequences of logical (ground) atoms. Here, we will describe how to lift CRFs to the relational case. More precisely, we consider the following variant of the SSL problem in Definition 1.

Definition 2 (Relational-propositional SSL (RP-SSL)). Given *a set of training examples* (X_i, Y_i)*, where each* X_i *is a sequences of logical atoms and each* Y_i *is a corresponding sequence of class labels* $y_{i,j} \in \{c_1, \ldots, c_n\}$*,* **find** *a classifier* H *with low approximation error on the training data as well as on unseen examples.*

The idea underlying TildeCRF, i.e., a CRF for solving RP-SSL, is now to pick up the idea of MLNs and to represent potentials as sets of weighted logical formulas.

4.1 Relational Logic and Relational Potentials

Based on the representation of the Ribosomal protein $L4$ given in the introduction, we describe the necessary concepts of relational logic. The symbols `st`, `null`, 2, `he`, `h`, ... are distinguished into predicate and function symbols. Associated with every symbol is the *arity*, i.e., number of arguments. In the example, `st/2` and `he/2` are predicates of arity 2, `h/2` is a function of arity 2, and `plus/0`, `1/0`, ... are functions of arity 0, i.e., constants. The *alphabet* Σ consists of predicates, functions, and variables (e.g., `X`). A *term* is a variable or a function symbol followed by its arguments in brackets such as `h(right, X)` or 4; an *atom* is a predicate symbol followed by its arguments in brackets such as `he(h(right, X), 4)`. Valid arguments of functions and predicates are terms. A *ground term* or *atom* is one that does not contain any variables. In the protein example `st(null, 2)`, `he(h(right, alpha), 6)`, ... are ground atoms and `null`, 2, `h(right, alpha)`, `right`, `alpha`, ... are ground terms. A substitution $\sigma = \{\texttt{X}/\texttt{plus}\}$ is an assignment of terms `plus` to variables `X`. Applying a substitution σ to a term or an atom e yields the instantiated term or atom $e\sigma$ where all occurrences of the variables `X` are simultaneously replaced by the term `plus`, e.g., `(st(X, 12), st(X, 10))` yields `st(plus, 12), st(plus, 10)`. A substitution σ is a *unifier* of a set of atoms S if $S\sigma$ is singleton; if furthermore for every unifier σ' of S there is a substitution σ'' such that $\sigma = \sigma'\sigma''$ then σ is the *most general unifier* (MGU) of S. A conjunction A is θ-subsumed by a conjunction B, denoted by $A \preceq_\theta B$, if there exists a substitution θ such that $B\theta \subseteq A$.

Relational abstraction within potentials offers a great compactness. Consider

$$0.938 \ : \ \texttt{outPrevIs(city(c)), containsAt(1, a(X, f)), containsAt(4, a(X, a))}$$

taken from the regression tree shown in Figure 2. It groups all ground instances, where `X` is substituted by some term such as $\{\texttt{X}/1\}, \{\texttt{X}/2\}, \ldots$ Therefore, relational abstraction makes useful prediction possible in very large state spaces, where many of the states are never observed in the training data.

The compactness and even comprehensibility, however, comes at the expense of a more complex parameter estimation problem: they are non-parametric functional representations. Therefore, gradient-based optimization techniques such as McCallum's MALLET [11], which assume a parameterized representation, cannot be applied. Instead, we follow Dietterich *et al.*'s gradient tree boosting technique [4], called TreeCRF. In TreeCRF, the potential functions are represented by sums of traditional regression trees, which are grown stage-wise in

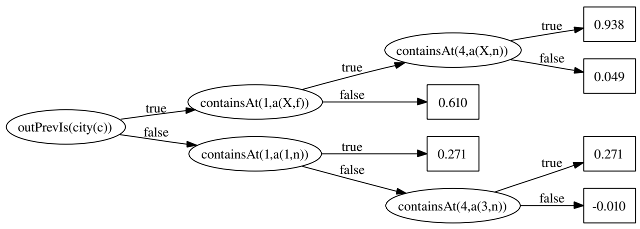


Fig. 2. A relational regression tree (taken from the *job scheduling* experiment of Section 5.2). An inner node represents a literal, a path constitutes a conjunction, and a leave represents the regression value (mean) of all examples sorted in this leave. As explained in Section 4.2, not the complete input X but only windows $w_d(X)$ at time steps d of fixed size s are used. `outPrevIs(Y)` denotes the output `Y` at time step $d-1$ and `containsAt(P, X)` the input `X` at position `P` in the current window $w_d(X)$.

the manner. Each regression tree can be viewed as defining several new feature combinations one corresponding to each path in the tree from the root to a leaf. The resulting potential functions still have the form of a linear combination of features, but the features can be quite complex.

4.2 Model Selection Via Functional Gradient Ascent

(Conditional) maximum likelihood parameter estimation is a common framework to determine the parameter Θ of a CRF. The likelihood of the training data given the current parameter Θ_{m-1} is used to improve the parameter. Normally, one uses some sort of gradient search for doing this. The parameter in the next iteration are the current plus the gradient of the likelihood function: $\Theta_m = \Theta_0 + \delta_1 + \ldots + \delta_m$ where $\delta_m = \eta_m \cdot \partial/\partial\Theta_{m-1} \sum_i \log P(y_i|x_i;\Theta_{m-1})$ is the gradient multiplied by a constant η_m, which is obtained by doing a line search along the gradient. In our non-parametric case, the potential can be arbitrarily chosen. One starts with some initial potential Ψ_0, e.g. the zero function, and adds iteratively corrections $\Psi_m = \Psi_0 + \Delta_1 + \ldots + \Delta_m$, cf. TreeBoost in Alg. 1. In contrast to the standard gradient approach, Δ_i here denotes the so-called functional gradient, i.e., $\Delta_m = \eta_m \cdot E_{x,y}\left[\partial/\partial\Psi_{m-1} \log P(y|x;\Psi_{m-1})\right]$. Since the joint distribution $P(x, y)$ is unknown, one cannot evaluate the expectation $E_{x,y}$. Dietterich *et al.* suggested to evaluate the gradient function at every position in every training example and fit a regression tree to these derived examples, cf. GenExamples in Alg. 1. In our case, these regression trees are relational.

Relational Regression Trees. Relational regression trees upgrade the attribute value representation used within classical regression trees: every test is a relational conjunction of atoms; a variable introduced in some node cannot appear in its right subtree, i.e., variables are bounded along left-tree paths. Consider the relational regression tree shown in Figure 2. The set of ground atoms $\{$`outPrevIs(city(c))`, `containsAt(1, a(2, f))`, `containsAt(4, a(2, n))`$\}$ is sorted into the left most leaf, i.e., the value 0.938 is assigned. In contrast, changing the last atom to `containsAt(4, a(4, n))` yields 0.049 as value.

Now, to induce a relational regression tree, we essentially employ Blockeel and De Raedt's TILDE [2], which also explains the name of our approach: TildeCRF. TILDE learns relational trees in the *learning from interpretations* setting, i.e., examples are sets of ground atoms. It basically follows Quinlan's well-known C4.5 algorithm. The only point where TILDE differs from C4.5 is in the computation of the tests to be placed in a node. To this aim, it employs a classical *refinement operator* under θ-subsumption. The operator basically adds a literal, unify variables, and grounds variables. When a node is to be splitted, the set of all refinements are computed and evaluated. That is one starts with the empty tree and repeatedly searches for the best test for a node according to some splitting criterion. Next, the examples D in the node are split into D_1 (success) and D_2 (failure) according to the test. For each split, the procedure is recursively applied, obtaining subtrees for the respective splits. As splitting criterion, we use the weighted variance on D_1 and D_2. The procedure stops if the variance in one node is small enough or the depth limit was reached. In leaves, the average regression value is predicted.

Using relational trees, Dietterich *et al.*'s TreeCRF can be adapted as follows.

Relational Functional Gradients. Following Dietterich *et al.*'s notation, we define $F^{y_t}(y_{t-1}, X) = \Psi(y_t, X) + \Psi(y_{t-1}, y_t, X)$. Then, the gradient $\frac{\partial \log P(Y|X)}{\partial F^v(u, w_d(X))}$ can be evaluated quite easily as Corollary 1 (see below) shows. By evaluating the gradient at every known position in our training data and fitting a regression model to this values, we get an approximation of the expectation of the gradient. In order to simplify the derivation of the gradient and afterwards the evaluation, we do not use the complete input X but a window $w_d(X) = x_{d-s}, \ldots, x_d, \ldots, x_{d+s}$, where s is a fixed window size. This is exactly the learning setting of TILDE: each window, i.e., each regression example is a (weighted) set of ground atoms.

Corolla 1. *The functional gradient with respect to* $F^v(u, w_d(X))$ *is*

$$\frac{\partial \log P(Y|X)}{\partial F^v(u, w_d(X))} = I(y_{d-1} \subseteq_\Theta u, y_d \subseteq_\Theta v) - P(y_{d-1} \subseteq_\Theta u, y_d \subseteq_\Theta v | w_d(X))$$

where I is the identity function, $\subseteq_\Theta$ denotes that u θ-subsumes y, and $P(y_{d-1} \subseteq_\Theta u, y_d \subseteq_\Theta v | w_d(X))$ is the probability that class labels u, v fit the class labels at positions $d, d-1$. It is calculated as shown in GENEXAMPLES *in Alg. 1.*

Proof. This is a straightforward adaption of the proof of proposition 1 in [4].

All the rest of TreeCRF remains unchanged. That is, we can use the forward-backward algorithm as proposed by [4] to compute $Z(X)$. The forward recursion is defined as $\alpha(k, 1) = \exp F^k(\bot, w_1(X))$ and $\alpha(k, t) = \sum_{k' \in K} \left[\exp F^k(k', w_t(X))\right] \cdot \alpha(k', t-1)$. The backward recursion is defined as $\beta(k, T) = 1$ and $\beta(k, t) = \sum_{k' \in K} \left[\exp F^{k'}(k, W_{t+1}(X))\right] \cdot \beta(k', t+1)$.

We will now turn over to how to use CRFs for making predictions.

Algorithm 1. Gradient Tree Boosting for SSL as introduced by [4]

```
 1: function TreeBoost($Data, L$)
 2:     for $1 \leq m \leq M$ do                                  ▷ Iterate Functional Gradient
 3:         for $1 \leq k \leq K$ do                              ▷ Iterate through the class labels
 4:             $Sk$ :=GenExamples($k, Data, Pot_{m-1}$)         ▷ Generate examples
 5:             $\Delta_m(k)$ :=FitRelRegressTree($S(k), L$)      ▷ Functional gradient
 6:             $F_m^k := F_{m-1}^k + \Delta_m(k)$                ▷ Update Models
 7:     return $Pot_M$                                            ▷ Return Relational Potential
 8: function GenExamples($k, Data, Pot_m$)
 9:     $S := \emptyset$                                          ▷ Initialize relational regression examples
10:     for all $(X_i, Y_i) \in Data$ do                          ▷ Iterate over all training examples
11:         $\big(\alpha, \beta, Z(X_i)\big)$ = ForwardBackward($X_i, T, K$)   ▷ Compute forward and
                backward probabilities
12:         for $1 \leq t \leq T_i$ do                            ▷ Iterate over all positions
13:             for $1 \leq k' \leq K$ do                         ▷ Iterate over all class labels
                                          ▷ Compute value of gradient at position $t$ for class label $k$
14:                 $P(y_{t-1} = k', y_t = k|X_i) := \frac{\alpha(k', t-1) \cdot \exp(F_m^k(k', w_t(X)) \cdot \beta(k, t)}{Z(X_i)}$
15:                 $\Delta(k, k', t) := I(y_{t-1} \subseteq_\Theta k', y_t \subseteq_\Theta k) - P(y_{t-1} \subseteq_\Theta k', y_t \subseteq_\Theta k|X_i)$
16:                 $S := S \cup \{((w_t(X_i), k'), \Delta(k, k', t))\}$   ▷ Update set of relational
                        regression examples
17:     return $S$
```

4.3 Making Predictions

There are several ways for getting a classifier from a trained CRF. We can predict the output sequence Y with the highest probability: $H(X) = \arg\max_Y P(Y|X)$. The Viterbi algorithm [14] can be used for this. Another option is to predict every atom y_t in the output sequence individually. This makes sense when we want to maximize the number of correctly tagged input atoms: $H_t(X) = \arg\max_{k \in K} P(y_t = k|X)$. Finally, one can also use a CRF for sequence classification, i.e., to predict a single label for the entire sequence. To do so, we can simply make a kind of majority vote. That is, we first predict $H(X)$. Next, we count the number of times each class atom was predicted, i.e., $\mathit{count}(c, Y) := |\{i \in \{1, \dots, T\} \mid y_i = c\}|$. Then, the sequence X is assigned to class c with probability $P(c|X) = T^{-1} \cdot \mathit{count}(c, H(X))$.

5 Experiments

Our intention here is to investigate to which extent TildeCRF for logical sequences is competitive with related approaches. To this aim, we implemented our system in Yap 5.1.0 prolog and investigate the following questions:

(Q1) Does TildeCRF perform equally well as traditional CRFs?
(Q2) If so, are there cases where TildeCRF leads to better results?
(Q3) If so, are there real-world datasets on which TildeCRFs performs better than established methods?

In the following, we will describe the experiments carried out to investigate **Q1**–**Q3** and the results.

5.1 (Q1) Protein Secondary Structure Prediction

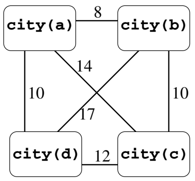
To show that CRFs for logical sequences perform equally well as traditional CRFs, we evaluated our gradient relational tree boosting algorithm on the protein secondary structure predication benchmark considered by Dietterich et. al [4]. The protein secondary structure benchmark was originally published by Qian and Sejnowski [13]. A protein consists of a sequence of amino acid residues. Each residue is represented by a single feature with 20 possible values (corresponding to the 20 standard amino acids). There are three classes: alpha helix, beta sheet, and coil (everything else). There is a training set of 111 sequences and a test set of 17 sequences.

The input features consisted of an 11-residue sliding window and we allowed the regression trees of up to 32 leafs. Dietterich et. al 's TreeCRF attained a test set performance of 64.7%. Our TildeCRF achieved a 64.2% test set accuracy. Qian and Sejnowski's method attained 64.5%, whereas McCallum's Mallet (a gradient based optimization approach for traditional CRFs) reached 62.9%. Thus, TildeCRF is in the range of TreeCRFs. Completely reproducing the TreeCRF results was difficult because we did not know the tree size used by Dietterich et al.. Overall, the results affirmatively answers **Q1**.

5.2 (Q2) Job Scheduling

To see whether there are cases where TildeCRF leads to better results than propositional approaches such as TreeCRF, we considered the task of job scheduling. Many jobs in industry and elsewhere require completing a collection of jobs while satisfying resource constraints. Thus, the goal is to arrange a total order among the jobs satisfies all the constraints while taking as little overall time as possible. Here, we will consider a version of the classical *travel salesman problem.*

There are a number of cities $\mathcal{C}$ given and different types of activities $\mathcal{A}$, which can have some parameters. E.g. an activity can be done with normal speed or fast. There is a cost function for traveling from city to city $c_{\text{travel}} : \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{R}_+$, and there is a cost function $c_{\text{act}} : \mathcal{A} \times \mathcal{C} \rightarrow \mathbb{R}_+$ which gives for every city and every activity the costs of doing this activity in that city. It might be the case, that a special activity isn't possible in some cities, therefore c_{act} can be a partial function. The task within this domain looks now as follows: given is a sequence of activities $a_1, \ldots, a_T$ goal is to find a sequence of cities $c_1, \ldots, c_T$ such that the overall costs are minimized: $costs(c_1, \ldots, c_T) = \sum_{t=1}^{T} c_{\text{travel}}(a_i, a_{i+1}) + c_{\text{act}}(c_i, a_i)$, where $a_{T+1} = a_1$. In the experiments, we considered the instance with 4 cities and 8 possible activities. Each activity `act(Type,Speed)` can be executed with normal speed and fast, therefore $\texttt{Speed} \in \{\texttt{normal}, \texttt{fast}\}$ and $\texttt{Type} \in \{\texttt{1}, \ldots, \texttt{8}\}$. The travel cost are listed in Figure 3 and the activity costs consists of two parts, namely $c_{\text{act}} = c_{\text{act'}} + c_{\text{speedcosts}}$ as listed in the same figure. To generate a data set, we randomly generated 100 independent activity sequences of length 15 and



act	1	2	3	4	5	6	7	8
city(a)	-	7	2	1	10	-	-	2
city(b)	11	-	3	-	5	-	-	-
city(c)	12	8	-	-	5	8	10	3
city(d)	13	9	-	-	5	8	-	10

	Extra
city(a)	22
city(b)	50
city(c)	12
city(d)	10

Fig. 3. Job scheduling: Instance with 4 cities and 8 activities, that was used in the experiment. (left) The map for the job scheduling domain. Nodes represent cities and edges transitions with associated costs. (middle) Costs of activities (right) Costs of doing an activity fast in one city.

searched brute force for an optimal travel sequence. This yield sequences such as $X = \langle \mathtt{act(4, normal)}, \mathtt{act(1, fast)}, \mathtt{act(8, normal)}, \mathtt{act(7, normal)}, \ldots \rangle$ with $Y = \langle \mathtt{city(a)}, \mathtt{city(d)}, \mathtt{city(c)}, \mathtt{city(c)}, \ldots \rangle$ We ran two experiments. At first we allowed just ground atoms as tests in the regression trees. This equals to the propositional approach of TreeCRF. In the second experiment we allowed atoms with variables as tests. Figure 4(b) shows the 10-fold cross-validated accuracy of predicted output symbols after each training iteration. One can readily see that TildeCRF outperforms TreeCRF. This affirmatively answers **Q2**.

5.3 (Q3) Protein Fold Classification

This experiment is concerned with how proteins fold up in nature. This is an important problem, as the biological functions of proteins depend on the way they fold up. A common approach to protein fold recognition is to start from a protein with unknown structure and search for the most similar protein (fold) with known structure in the database. This approach has been followed by Kersting *et al.* [6] where LoHMMs with the plug-in estimate were able to achieve a cross-validate predictive accuracy of 75%. Notice that the number of parameters of the LoHMMs used were by an order of magnitude smaller than the number of an equivalent HMM (120 vs. approx. 62000). Based on these results, Kersting and Gärtner [7] devised Fisher kernels for logical sequences and achieved a cross-validated accuracy of about 84%.

The data consists of logical sequences of the secondary structure of protein domains. The task is to predict one of the five most populated SCOP folds of alpha and beta proteins (a/b): TIM beta/alpha-barrel (c1), NAD(P)-binding Rossmann-fold domains (c2), Ribosomal protein L4 (c23), Cysteine hydrolase (c37), and Phosphotyrosine protein phosphatases I-like (c55). The class of a/b proteins consists of proteins with mainly parallel beta sheets (beta-alpha-beta units). Overall, the class distribution is as follows (class,#sequences): (c1, 721), (c2,360), (c23,274), (c37,441), (c55,290). Thus, this is a multiclass problems with 5 different classes. Although, CRFs are indeed able to treat multiclass problems, a round robin approach [5] worked better in our experiments. That is, each pair of classes is treated as a separate classification problem. The overall classification

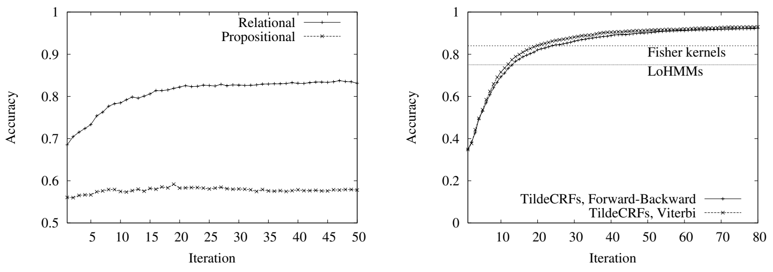


Fig. 4. Cross-validated classification accuracy (y axis) vs. number of iterations (x axis). (Left) Job scheduling: TildeCRF achieved 83.13 whereas TreeCRF achieves 57.8%. The difference is significant (one-tailored t-test, $p = 0.05$). (Right) Protein Fold Classification: TildeCRF achieved 92.96%, HMMs (resp. Fisher kernels) for logical sequences achieved 75% (resp. 84%) as indicated by the vertical lines. The differences are significant (one-tailored t-test, $p = 0.05$).

of an example instance is the majority vote among all pairwise classification problems.

Figure 4(b) summarizes the experimental results. It shows the 10-fold cross-validated accuracy learning curves. The accuracy converges around 92.96% (using Viterbi labeling). Thus, compared to LoHMMs, the error rate of CRFs is about 3 times smaller. The CRF also performed better than Fisher kernels; the error rate dropped about half. The differences are significant (one-tailored t-test, $p = 0.05$). This finally affirmatively answers **Q3**.

6 Conclusions

So far, Conditional Random Fields (CRFs) have only been considered for sequences of flat symbols. In this paper, CRFs for logical sequences, i.e., sequences over an alphabet of logical atoms have been introduced and experimentally investigated. Experiments have demonstrated that CRFs can handle logical sequences, the learning algorithm presented performs well in practice, and CRFs for logical sequences can indeed lead to significantly better results than flat CRFs, and HMMs respectively Fisher kernels for logical sequences.

The approach presented suggest a very interesting line of future research, namely to address a more general labeling problem: labeling of sequences of sets of ground atoms with ground atoms. Many problems in learning relational actions and within relational reinforcement learning are of this type.

Acknowledgments. The authors thank Alan Fern for useful comments, Tom Dietterich for providing his version of Qian and Sejnowski's benchmark, and Luc De Raedt for his support. The research was supported by the European Union IST programme, contract no. FP6-508861, *Application of Probabilistic ILP II.*

References

1. C. R. Anderson, P. Domingos, and D. S. Weld. Relational Markov Models and their Application to Adaptive Web Navigation. In *Proc. of the 8th Int. Conf. on Knowledge Discovery and Data Mining (KDD-02)*, pages 143–152, 2002.
2. H. Blockeel and L. De Raedt. Top-down Induction of First-order Logical Decision Trees. *Artificial Intelligence*, 101(1–2):285–297, 1998.
3. L. De Raedt and K. Kersting. Probabilistic Inductive Logic Programming. In *Proc. 15th Int. Conf. on Algorithmic Learning Theory (ALT-04)*, pages 19–36, 2004.
4. T. Dietterich, A. Ashenfelter, and Y. Bulatov. Training conditional random fields via gradient tree boosting. In *Proc. 21st International Conf. on Machine Learning*, pages 217–224. ACM, 2004.
5. J. Fürnkranz. Round Robin Classification. *Journal of Machine Learning Research (JMLR)*, 2:721–747, 2002.
6. K. Kersting, L. De Raedt, and T. Raiko. Logial Hidden Markov Models. *Journal of Artificial Intelligence Research (JAIR)*, 25:425–456, 2006.
7. K. Kersting and T. Gärtner. Fisher Kernels for Logical Sequences. In *Proc. of 15th European Conference on Machine Learning (ECML-04)*, pages 205 – 216, 2004.
8. K. Kersting and T. Raiko. 'Say EM' for Selecting Probabilistic Models for Logical Sequences. In *Proc. of the 21st Conf. on Uncertainty in Artificial Intelligence (UAI-05)*, pages 300–307, 2005.
9. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th Int. Conf. on Machine Learning (ICML-01)*, pages 282–289, 2001.
10. J. W. Lloyd. *Foundations of Logic Programming.* Springer, Berlin, 2. edition, 1989.
11. A. McCallum. Effciently inducing features of conditional random fields. In *Proc. of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-03)*, 2003.
12. A. Quanttoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems 17*, pages 1097–1104, 2005.
13. N. Quian and T. J. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *JMB*, 202:865–884, 1988.
14. L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–285, 1989.
15. M. Richardson and P. Domingos. Markov Logic Networks. *Machine Learning*, 62:107–136, 2006.
16. S. Sanghai, P. Domingos, and D. Weld. Dynamic probabilistic relational models. In *Proc. of the 8th Int. Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 992–997, 2003.
17. C. Sutton and A. McCallum. Piecewise training of undirected models. In *Proc. of the 21. Conference on Uncertainty in Artificial Intelligence (UAI-05)*, 2005.
18. C. Sutton, K. Rohanimanesh, and A. McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proc. 21st International Conf. on Machine Learning.* ACM, 2004.
19. B. Taskar, P. Abbeel, and D. Koller. Discriminative Probabilistic Models for Relational Data. In *Proc. of the 8th Conf. on Uncertainty in Artificial Intelligence (UAI-02)*, pages 485–492, 2002.

Unsupervised Multiple-Instance Learning for Functional Profiling of Genomic Data

Corneliu Henegar[1], Karine Clément[1,2,3], and Jean-Daniel Zucker[1,4]

[1] INSERM, UMR U-755 Nutriomique, Hôtel-Dieu, Paris, France
corneliu@henegar.info
[2] Université Paris VI, Faculté de Médecine Les Cordeliers, Paris, France
[3] AP-HP, Pitié-Salpêtrière, Service de Nutrition, Paris, France
[4] LIM&BIO EA3969, Université Paris Nord, Bobigny, France

Abstract. Multiple-instance learning (MIL) is a popular concept among the AI community to support supervised learning applications in situations where only incomplete knowledge is available. We propose an original reformulation of the MIL concept for the unsupervised context (UMIL), which can serve as a broader framework for clustering data objects adequately described by the multiple-instance representation. Three algorithmic solutions are suggested by derivation from available conventional methods: agglomerative or partition clustering and MIL's citation-kNN approach. Based on standard clustering quality measures, we evaluated these algorithms within a bioinformatic framework to perform a functional profiling of two genomic data sets, after relating expression data to biological annotations into an UMIL representation. Our analysis spotlighted meaningful interaction patterns relating biological processes and regulatory pathways into coherent functional modules, uncovering profound features of the biological model. These results indicate UMIL's usefulness in exploring hidden behavioral patterns from complex data.

1 Introduction

The conceptual frame of the multiple-instance learning (MIL) was proposed in 1997 by Dietterich [1], together with a first meaningful application to drug activity prediction. Since then, an important amount of research has dealt with the development of specific learning algorithms, adapted to MIL's particular context, and to comparative performance assessment in relation with different types of applications, as well as with various other conventional supervised learning approaches [2,3,4,5,6,7,8,9,10]. As a result, MIL's applicability has been tested in numerous domains, ranging from content-based image retrieval and classification [11], text categorization [6] and web mining [12], to protein sequence analysis, robot vision and stock market prediction [13,14]. Conventional MIL is a variation on supervised learning, fitting those situations in which the knowledge about the labels of training examples is incomplete. Under such circumstances MIL allows for modeling weaker assumptions about the labeling information by assigning labels to sets of instances (bags), instead of assigning them to each individual instance. Bags labels can be positive or negative in the Boolean case, or have a continuous real value in the real data MIL [15]. A bag is labeled as positive if *at least one* of its instances is positive (linearity constraint), and negative if *all* of its instances are negative. In generalized MIL, a variant of the conventional model, bags labels are determined by a non-disjunctive function over their instances, thus eliminating the linearity constraint in order to reduce noise level [9].

In this paper we propose an abstract reformulation of the conventional MIL paradigm, which preserves the general multiple-instance representation, while further weakening the supervised learning constraints into a fully unsupervised multiple-instance learning (UMIL) framework. The main motivation behind this reformulation resides in the usefulness of the multiple-instance schema, which allows to describe some difficult unsupervised learning problems through simple and yet robust representations. Such representations can provide a basis for solving intricate clustering problems, aiming at discovering hidden behavioral patterns from complex data objects described by multiple types of attributes (e.g. numerical, symbolic, etc.). Among other possible examples, such complex objects are found in genomic data sets in which RNA transcripts are sharing numerous descriptive features in relation to their various biological roles. Therefore, we relied on the functional genomics framework to illustrate the UMIL concept by relating RNA expression data to functional annotations to build multiple-instance representations. These representations were further used to perform a functional analysis of two genomic data sets, aiming at identifying context related biological interaction patterns involving cellular processes and regulatory pathways. Section two outlines the main characteristics of the UMIL paradigm. The third section suggests three algorithmic solutions, derived from existent unsupervised learning or conventional MIL approaches, adapted to the UMIL context. The fourth section details the experimental framework and results. Finally we indicate some potential directions for future work.

2 The Unsupervised Multiple-Instance Model

2.1 UMIL Definition

In order to allow for a maximum flexibility in building multiple-instance representations, we imagined the UMIL paradigm as an abstract generalization of the conventional multiple-instance schema. Let us consider a data set $\mathcal{D}$ composed of n objects $o_j \in \mathcal{D}$, sharing similar data structures, each of them being characterized by an ensemble of feature values $o_j = \{f_1 = v_{1j}, f_2 = v_{2j}, ..., f_i = v_{ij}, ...\}$, be it numerical, Boolean or set-valued attributes. Among the ensemble $\mathcal{F}$ of all features describing objects $o_j \in \mathcal{D}$, let $f_i \in \mathcal{F}$ be a *feature* whose domain contains m distinct values, $f_i = \{v_1, v_2, ..., v_k, ..., v_m\}$, each object $o_j \in \mathcal{D}$ being characterized by one or more values of f_i. Based on the feature f_i we derive the ensemble $\mathcal{B}$ of bags $b_k \in \mathcal{B}$, $(k \leq m)$, defining an UMIL model, where each bag b_k corresponds to the ensemble of objects $o_j \in \mathcal{D}$ sharing (at least) one common feature value $f_i = v_k$, which defines the bag b_k. As each of the objects $o_j \in \mathcal{D}$ can be characterized by one or more values of $f_i \in \mathcal{F}$, it follows that UMIL bags are non disjoint (e.g. overlapping) sub-ensembles of $\mathcal{D}$, their distinctiveness being guaranteed by the common feature value $f_i = v_k$ of their instances. We propose that this multiple-instance abstraction may constitute a relevant framework for exploring complex relationships between multiple-instance objects in an unsupervised learning context. Under these circumstances, the UMIL problem can be stated formally as to find an optimum partition of $\mathcal{B}$ into $l < m$ disjoint classes of interrelated bags $C_1 \cup C_2 \cup ... \cup C_l$.

2.2 Multiple-Instance Representations of Genomic Data

In genomic data sets RNA transcripts are represented through complex data structures, which are regrouping heterogeneous information related to expression measurements (real value data), molecular structure, functional roles, regulatory mechanisms, etc. Biological roles of RNA transcripts are formally represented through functional annotations established in relation with available biological evidence. These representations are built through an annotation process which relates RNA transcripts to a taxonomic hierarchy of functional categories (set-valued attributes), allowing to represent biological knowledge about transcripts roles with various degrees of precision. In the most general case, the relations among transcripts and functional categories are of the many-to-many type, in which a transcript may be related to one or more biological processes, each of these processes involving one or more transcripts. Considered as a major challenge, the functional analysis, which aims at translating RNA expression data into relevant biological mechanisms, is an indispensable step for the comprehension of the underlying biological phenomena defining an experimental model. Besides assessing the individual dynamics of various biological processes, based upon the expression patterns of the transcripts known to be involved in those processes, the functional profiling aims also at characterizing intricate biological interactions involving cellular processes and regulatory pathways. These considerations suggest the relevance of the UMIL paradigm as a formal framework for assessing interactions between functional categories, represented as multiple-instance objects (e.g. *bags*) which regroup annotated transcripts (e.g. *instances*).

2.3 Similarity and Relationship Measures for UMIL Objects

As a consequence of definition (2.1) two types of measures seem relevant for comparing objects belonging to an UMIL representation. The first one will evaluate the similarity between individual instances, thus conditioning the second one which will assess the relationship between bags. In our context we selected the pairwise mutual information (MI) as the similarity metric for transcripts expression, based on its ability to recognize as proximal positively, negatively and nonlinearly correlated transcript profiles [16, 17]. MI computation is based on the notion of entropy of a random variable suggested by Shannon's theory of information. Thus for a discrete random variable X, whose probability distribution is $P(X = x_i)$, $i = 1, ..., N_x$, where N_x is the number of possible values of X, the entropy $H(X)$ is defined as:

$$H(X) = -\sum_{i=1}^{N_x} P(X = x_i) \log_2 P(X = x_i) \ . \tag{1}$$

For the case of continuous random variables (e.g. expression profiles) a preliminary discretization, through a histogram technique, is necessary in order to compute their probability distribution. Based on (1) the pairwise mutual information of two random variables X, Y is defined as:

$$MI(X,Y) = H(X) + H(Y) - H(X,Y) \tag{2}$$

where $H(X,Y)$ is their joint entropy. The normalized $\overline{MI}(X,Y)$ is a relative measure [17] which reduces the influence of the magnitudes of individual entropies:

$$\overline{MI}(X,Y) = \frac{MI(X,Y)}{\max\{H(X),\, H(Y)\}} \ . \tag{3}$$

From (3) it follows that $0 \leq \overline{MI}(X,Y) \leq 2$. Moreover, it is possible to estimate a threshold of significance T_{MI} for the pairwise mutual information through iterative random permutations over the matrix of expression measurements [16]. Given two possibly overlapping bags A and B, the strength of their relationship can be quantified separately, from each bag's perspective, through a non-disjunctive function over all instances belonging to that bag for which there is at least one similar (or identical) instance in the other bag, and vice versa. Let n_{ab} be the sub-ensemble of instances $a_i \in A$ for which there is at least one instance $b_j \in B$ satisfying the similarity constraint T_{MI}:

$$n_{ab} = \{a_i \in A \mid \exists\, b_j \in B,\, \overline{MI}(a_i, b_j) \geq T_{\mathrm{MI}}\} \ . \tag{4}$$

Consider $\bar{n}_{ab}$ the cardinality of n_{ab} and $\bar{n}_{ba}$ its equivalent for bag B. From (4) it follows that in the most general case $\bar{n}_{ab} \neq \bar{n}_{ba}$. Under these circumstances, the ratio $S_{A\to B} = \frac{\bar{n}_{ab}}{\bar{n}_A}$, where $\bar{n}_A$ is the cardinality of bag A, can be considered as an *asymmetrical measure* of the relationship between the two bags from bag A perspective, satisfying $0 \leq S_{A\to B} \leq 1$. In order to give a better account of the qualitative value of instances similarity we can further refine $S_{A\to B}$ by weighting it with the average of the maximal similarities of individual instances $a_i \in A$ satisfying (4) in relation to $b_j \in B$ and define an *asymmetrical measure* of the relationship of A with B as:

$$D_{A\to B} = 1 - S_{A\to B} \left[\frac{1}{2\bar{n}_{ab}} \sum_{i=1}^{\bar{n}_{ab}} \max_{j=1}^{\bar{n}_B} \overline{MI}(a_i, b_j) \right] \tag{5}$$

From (5) it follows that $0 \leq D_{A\to B} \leq 1$ and also that $D_{A\to B} \neq D_{B\to A}$ in the most general case. Based on (5) a *symmetrical measure* of the relationship between two bags A and B can be defined as:

$$D_{AB} = \frac{1}{2} \left(D_{A\to B} + D_{B\to A} \right) \ . \tag{6}$$

3 Algorithmic Solutions

Two directions were explored in search for algorithmic solutions adapted to the UMIL context. The first one was to examine possible adaptations of existing unsupervised learning approaches. The second was to consider adaptations of supervised MIL approaches to the unsupervised context. Our analysis shows that some of the difficulties which need to be addressed are different in each of these two cases, while others are common.

3.1 Unsupervised Clustering Approaches for the UMIL Context

The proposed definition (2.1) of the UMIL paradigm suggests the idea of adapting conventional unsupervised clustering approaches for the UMIL context. For instance, one simple solution could be to initiate a conventional hierarchical agglomerative clustering algorithm with the partition of the instances in their corresponding bags (considered as "clusters" of instances). In these circumstances, the hierarchical clustering algorithm could presumably be used to identify classes of related bags by relying only on the similarity of their instances. However, some of the characteristics of the UMIL representation, like the possible overlapping between bags in the most general case, cannot be handled correctly by a conventional unsupervised clustering approach. A possible solution to this obstacle could be to reduce the multiple instance model to a simple instance one, by relying on the symmetrical measure of the relationship between bags (6) defined previously. This reductive approach allowed us to test two conventional unsupervised clustering techniques for the UMIL context: an hierarchical agglomerative algorithm [18] and a k-means partitioning algorithm [19], each of them combined with a standard quality measure for cluster partitions which allows to identify an optimal partition of bags into classes. The prediction of the correct number of clusters is a fundamental question in unsupervised classification problems [20]. Although there is no best approach to fit all situations, the computation of the Silhouette index [19] was shown to be a simple and yet robust strategy for the prediction of optimal clustering partitions from transcript expression data [21].

3.2 A Citation Approach for the UMIL Context

A conventional MIL solution that may be easily adapted for the unsupervised context is that proposed originally by Wang and Zucker [3], which combines k-nearest neighbor (kNN) lazy learning with the citation concept (citation-kNN) inspired from library and information science. In our context the concept of bibliographic citations is suggested by the asymmetrical aspect of the relationship between bags (5). This results in the fact that two bags can "refer" to each other with a different degree of confidence strength. Based on this observation we imagined an *unsupervised citation-kNN* (UC-kNN) solution whose main steps are illustrated by Algorithm 1. Let m be the number of individual bags $b_i \in \mathcal{B}$ contained in the UMIL representation $\mathcal{B}$. Considering (5) as the measure of relationship between bags, a bag $b_j \in \mathcal{B}$ can be presumed to be a good "reference" for another bag $b_i \in \mathcal{B} \backslash b_j$ if bag b_j is ranked among the $k < m$ most closely related bags to bag b_i (considered therefore as its k nearest neighbors or kNN).

Algorithm 1. A sketch of the UC-kNN algorithm

```
Input: an UMIL representation $\mathcal{B} = \{b_1, ..., b_m\}$, containing $m$ bags with their instances, and the similarity matrix for instances computed with (3)
Output: the optimal partition of the bags
Compute bags relationship matrix with (5)
For each $k$, $1 \leq k \leq m-1$ (e.g. the number of nearest neighbors) do:
    Compute a ranked vector $\mathcal{R}$ of the bags reference scores $R_b = \sum_i rank(b, b_i)$, for each $b \in \mathcal{B}$, in relation to the rest of the bags $b_i \in \mathcal{B} \backslash b$ which satisfy $rank(b, b_i) \leq k$
    For each $p$, $2 \leq p < m$, select the first $p$ bags from $\mathcal{R}$ as cluster seeds, then do:
        For each $m - p$ bags $b_i$, distinct from the $p$ selected cluster seeds, do:
            Find the $k$ best references $b_j$ for $b_i$ then compute for each of the $p$ cluster seeds $s$ the value $V_{sb_i} = rank(s, b_i) + \frac{1}{k} \sum_{j=1}^{k} rank(s, b_j)$ and cluster $b_i$ to the closest seed
        Compute the Silhouette index for the resulting partition of bags and store results
Select the optimal partition of bags, among those computed for each possible combination of the values of $k$ and $p$, which maximizes the Silhouette index
```

On this base a reference score R_b can be computed for each value of $k < m$ and for each bag $b \in \mathcal{B}$, in relation to the rest of bags $b_i \in \mathcal{B} \backslash b$, as the sum of b's ranking positions for all the situations where $rank(b, b_i) \leq k$ (see Algorithm 1). This suggests that, for a given value of k, it is possible to initiate an agglomerative clustering procedure by considering as seeds of the future classes (or clusters) the first p bags, $2 \leq p < m$, having the best reference scores (e.g. the most "cited" ones). Under these circumstances, a kNN clustering approach can group each of the rest of the bags to their most closest seed, by relying not only on the individual similarity between the bags and the seeds, but by considering also the similarity of their k nearest neighbors to these seeds, integrated into a *weighted voting procedure*. This is to say that for each bag b_i, distinct from the considered p seeds, we search the closest seed s minimizing the value of:

$$V_{sb_i} = rank(s, b_i) + \frac{1}{k} \sum_{j=1}^{k} rank(s, b_j) \tag{7}$$

where b_j, $1 \leq j \leq k$, belongs to the k nearest neighbors of bag b_i. Thus, for each couple of values (k, p), with $k, p < m$, the UC-kNN approach will build a partition $P_{(k,p)} = \{C_1 \cup ... \cup C_p\}$ of the ensemble of bags $\mathcal{B}$ into p distinct classes. As for the adaptation of the conventional unsupervised clustering approaches, an optimal partition of bags can be selected from the ensemble of computed partitions by using a standard quality evaluation measure. For coherence and simplicity reasons we combined UC-kNN with the Silhouette technique [19].

4 Experimental Frame

The experimental context, which served to build multiple-instance representations and to test UMIL algorithmic solutions, belongs to functional genomics.

4.1 Adipose Tissue Data Sets

The potential benefit of the UMIL concept for the genomic functional analysis was assessed on two interrelated RNA expression measurements data sets. Both of them resulted from pangenomic cDNA microarray expression profiling of white adipose tissue in morbidly obese human subjects, and were extensively described in [22]. The first data set resulted from differential expression profiling of the two cellular fractions of human white adipose tissue: mature adipocytes and stroma-vascular fraction cells (SVF). The second one resulted from microarray expression profiling of whole white adipose tissue in morbidly obese human subjects, before/after undergoing a form of bariatric surgery. These two data sets were combined in order to constitute a coherent experimental model, designed to characterize the functional profiles of each of the two cellular fractions of the adipose tissue in obese human subjects, as well as their evolution after a significant weight loss induced by bariatric surgery.

4.2 Experimental Setup

The three proposed algorithmic solutions were implemented in the R environment for statistical computation (available at `http://www.r-project.org/`). As originally indicated [22], transcripts with significant expression changes were identified by using the significance analysis of microarrays (SAM) procedure (available at `http://www-stat.stanford.edu/tibs/SAM/`). Significant differential expression was assessed by imposing a 5% false discovery rate (FDR) threshold in the SAM selection procedure. Automated functional annotation of the differentially expressed transcripts, identified in the two data sets, relied on Gene Ontology Consortium (GO [available at `http://www.geneontology.org`]) and Kyoto Encyclopedia of Genes and Genomes (KEGG [available at `http://www.genome.ad.jp/kegg/`]) annotations. EntrezGene numbers (available at `http://www.ncbi.nlm.nih.gov/entrez`) were used as a standard transcript accession system to ensure a correct over-representation analysis, as they allow to map transcript identifiers to GO or KEGG categories in an unequivocal way. In order to minimize the false over-representation resulting from redundant annotation, the automated GO annotation procedure was restricted to directly annotated transcripts by each GO category. As originally indicated [22], the significance of the over-representation of each GO and KEGG category was assessed by using a Fisher's exact test. Afterwards, significantly over-represented GO and KEGG categories were related to their annotated transcripts into an UMIL model, in which each category (GO or KEGG) was considered as a bag of individual instances represented by its annotated transcripts. A threshold T_{MI} for the normalized pairwise mutual information of transcripts expression was computed previously to applying unsupervised agglomerative or partitioning clustering and UC-kNN algorithms to the UMIL representation of genomic data. As previously suggested [16], T_{MI} estimation was based on the average $\overline{MI}$ distribution computed from 30 randomly permuted repetitions of RNA expression measurements. The significance threshold for the pairwise mutual information among transcripts was chosen to be $T_{\mathrm{MI}} = mean(\overline{MI}) + 2SD(\overline{MI})$, where $mean(\overline{MI})$ is the average of $\overline{MI}$ and $SD(\overline{MI})$ the standard deviation of the mean.

Table 1. Characteristics of the optimal partitions obtained by applying the agglomerative hierarchical clustering (HC), k-means partition clustering (K-means) and the unsupervised citation kNN (UC-kNN) algorithms to the two adipose tissue data sets

HC	Min	Max	Average
Clusters number	2	29	6.81 ± 6.64
Clusters length	1	35	4.18 ± 7.05
Clusters Silhouette	0	0.83	0.14 ± 0.17
Partitions Silhouette	0.05	0.52	0.14 ± 0.11
K-means	**Min**	**Max**	**Average**
Clusters number	2	53	16.31 ± 13.23
Clusters length	1	12	1.75 ± 1.80
Clusters Silhouette	0	1	0.06 ± 0.16
Partitions Silhouette	0.05	0.20	0.11 ± 0.04
UC-kNN	**Min**	**Max**	**Average**
Clusters number	2	6	3.44 ± 1.21
Clusters length	1	64	8.29 ± 13.1
Clusters Silhouette	0	1	0.35 ± 0.34
Partitions Silhouette	0.04	0.68	0.37 ± 0.15

4.3 Results

A few characteristics of the results produced by the three algorithmic approaches are summarized in Table 1. As it can be seen the Silhouette indexes of the partitions produced by the UC-kNN approach are much higher than those resulting from the two adaptations of unsupervised clustering approaches. Moreover, unsupervised clustering partitions were on average more sparse than those produced by the UC-kNN solution. For all these reasons, and also because of space restrictions, only a fraction of the UC-kNN clustering results are detailed hereafter and discussed in terms of biological relevance.

Table 2. Main UC-kNN clusters of KEGG categories specifically expressed in each of the two adipose tissue fractions: adipocytes and stroma-vascular fraction (SVF)

KEGG Category	Nb. Transcr.*	P-value**
Cluster 1 - Adipocytes	109	$2.84\ 10^{-12}$
Tryptophan metabolism	26	$9.58\ 10^{-3}$
Fatty acid metabolism	23	$1.35\ 10^{-5}$
Pyruvate metabolism	22	$2.05\ 10^{-6}$
Valine, leucine & isoleucine degrad.	22	$1.57\ 10^{-4}$
Basal transcription factors	10	$4.87\ 10^{-2}$
Other metabolic processes (9 terms)	64	—
Cluster 1 - SVF	186	$3.93\ 10^{-22}$
Cytokine-cytokine recept. interact.	65	$5.61\ 10^{-8}$
Hematopoietic cell lineage	37	$5.10\ 10^{-9}$
Ribosome	33	$2.93\ 10^{-9}$
Natural killer cell med. cytotox.	32	$5.15\ 10^{-4}$
Complement & coagulation cascades	23	$7.47\ 10^{-4}$
TGF-beta signaling pathway	22	$2.65\ 10^{-2}$

* number of annotated transcripts
** transcript enrichment p-value computed with Fisher's exact test

Table 2 shows one cluster (from a total of 4, with individual Silhouettes of 0.50 and 0.48 respectively, and a partition Silhouette of 0.31) grouping KEGG categories annotating adipocytes transcripts, and one cluster (from a total of 3, with an individual Silhouette of 0.33, and a partition Silhouette of 0.31) characterizing the stroma-vascular fraction (SVF) transcripts. Cluster 1 - Adipocytes (Table 2) is grouping 13 metabolic processes known to be highly interrelated and specific of mature adipocytes. It thus depicts the functional profile of mature adipocytes involving various metabolic processes (energetic, lipidic or protidic) [22]. An interesting aspect is that these metabolic processes were grouped together with a set of 10 transcription factors, which suggests a specific regulating role over these processes. Indeed, at least four of them (TAF6, TAF7, TAF10 and TAF12) are known to be pro-adipogenic factors, enhancing the action of C/EBPα and TBP/TFIIB which are key regulators of the adipogenesis [23, 24]. Cluster 1 - SVF (Table 2) illustrates the preponderant role of the SVF in the pathogenesis of local and systemic inflammatory processes accompanying the inflation of the adipose tissue in humans. The presence of the TGF-beta signaling pathway in this cluster has strong biological significance, since TGF-beta is known to stimulate the proliferation of pre-adipocytes while inhibiting adipogenesis [23]. These findings may corroborate with available evidence, indicating the conversion of pre-adipocytes into macrophages under particular circumstances [25], thus supporting the paradigm of a major role of local adipose tissue macrophages in the pathogenesis of inflammatory processes characterizing human obesity [22]. For

Table 3. Main UC-kNN cluster of Gene Ontology (Biological Process) categories significantly down-regulated in human adipose tissue after bariatric surgery.

Gene Ontology Category	Nb. Transcr.*	P-value**
Cluster 1	86	$2.18\ 10^{-3}$
Apoptosis	61	$3.14\ 10^{-2}$
Anti-apoptosis	25	$8.31\ 10^{-3}$
Acute phase response	8	$3.18\ 10^{-2}$
Induction of apoptosis / intracel. sign.	5	$2.03\ 10^{-2}$

* number of annotated transcripts
** transcript enrichment p-value computed with Fisher's exact test

all these reasons the two analyzed clusters can be considered as a convincing illustration of the complex dynamics of the adipogenic regulatory mechanisms, in which pro-adipogenic factors act concomitantly with anti-adipogenic ones, thus resulting into an ever changing network of complex interactions [23].

Table 3 present one Gene Ontology Biological Process cluster (from a total of 4, with an individual Silhouette of 0.36, and a partition Silhouette of 0.40), characterizing adipose tissue transcripts down-regulated after bariatric surgery. This cluster indicate a coherent deflation of inflammatory phenomena accompanying weight loss. Indeed, the reduction in local synthesis of the acute phase response molecules, together with a consecutive reduction of apoptotic processes corroborate with previously reported results [22].

4.4 Discussion

Except for some particular situations in which supplementary knowledge is available, the validation of the unsupervised clustering results remains a difficult issue. In spite of their relative value, cluster quality measures were shown to be useful indicators of the relevance of transcript data partitions [21]. In our experimental context, the UC-kNN solution yielded much higher Silhouette indexes than the hierarchical clustering approach. These findings seem coherent with previous observations suggesting a good adequacy of the local approaches for the multiple-instance context [3]. Subsequently, the results of the functional profiling of the adipose tissue expression data were discussed in terms of biological significance, in accord with available biological knowledge. Our assessment pointed out the biological relevance of the UMIL functional analysis which spotlighted significant biological regulatory mechanisms, thus illustrating the underlying modular structure of the transcriptional regulatory networks.

5 Conclusion and Future Work

This paper proposes a new framework for the unsupervised clustering of complex data objects adequately described by an abstract multiple-instance representation.

Three algorithmic solutions, adapted to the new framework, are suggested. The application of the UMIL concept to the functional analysis of genomic data illustrates its usefulness in exploring hidden behavioral patterns from complex data. The UMIL model shares common features with other unsupervised learning models. Among them, the concept of a variable size transaction, used in market basket data analysis, may be the closest one from that of an UMIL bag. Defined as a finite set of items from a common item universe, the transaction concept can be considered as a particularization of the bag concept for the case in which instances are all categorical data structures. Therefore investigating the possibility of adapting existent categorical data algorithms to the UMIL context might prove interesting, as this could result in useful solutions for sparse and high dimensional data, known to be less adapted to local approaches. Another research direction will be to examine the possibility of a Bayesian solution for the UMIL frame. Besides this, other potential applications of the UMIL framework need to be considered, especially in those domains in which conventional multiple-instance framework proved useful. One such domain could be the content-based image retrieval and classification problem. An obvious advantage of considering this problem, besides the evident interest of this application, lies in a presumably simpler and more objective assessment of clustering results.

Acknowledgments

This work was supported by the Institut National de la Santé et de la Recherche Médicale (INSERM) and the Assistance Publique - Hôpitaux de Paris.

References

1. Dietterich, T.G., Lathrop, R.H., Lozano-Pérez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artif. Intell. **89**(1-2) (1997) 31–71
2. Maron, O., Lozano-Pérez, T.: A framework for multiple-instance learning. In: NIPS. (1997)
3. Wang, J., Zucker, J.D.: Solving the multiple-instance problem: A lazy learning approach. In: ICML. (2000) 1119–1126
4. Chevaleyre, Y., Zucker, J.D.: Solving multiple-instance and multiple-part learning problems with decision trees and rule sets. application to the mutagenesis problem. In: Canadian Conference on AI. (2001) 204–214
5. Zhang, Q., Goldman, S.A.: Em-dd: An improved multiple-instance learning technique. In: NIPS. (2001) 1073–1080
6. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: NIPS. (2002) 561–568
7. Goldman, S.A., Scott, S.D.: Multiple-instance learning of real-valued geometric patterns. Ann. Math. Artif. Intell. **39**(3) (2003) 259–290
8. Tao, Q., Scott, S., Vinodchandran, N.V., Osugi, T.T.: SVM-based generalized multiple-instance learning via approximate box counting. In: ICML. (2004)
9. Tao, Q., Scott, S.D.: A faster algorithm for generalized multiple-instance learning. In: FLAIRS Conference. (2004)

10. Ray, S., Craven, M.: Supervised versus multiple instance learning: an empirical comparison. In: ICML 2005 Conference. (2005)
11. Zhang, Q., Goldman, S.A., Yu, W., Fritts, J.: Content-based image retrieval using multiple-instance learning. In: ICML. (2002) 682–689
12. Zhou, Z.H., Jiang, K., Li, M.: Multi-instance learning based web mining. Appl. Intell **22**(2) (2005) 135–147
13. Brown, J., Zhang, J., Scott, S.: On generalized multiple-instance learning. Technical report, University of Nebraska (2003)
14. Yang, J.: Review of multi-instance learning and its applications. Technical report, School of Computer Science Carnegie Mellon University (2005)
15. Dooly, D.R., Zhang, Q., Goldman, S.A., Amar, R.A.: Multiple-instance learning of real-valued data. Journal of Machine Learning Research **3** (2002) 651–678
16. Butte, A., Kohane, I.: Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. Pac Symp Biocomput (2000) 418–29
17. Zhou, X., Wang, X., Dougherty, E., Russ, D., Suh, E.: Gene clustering based on clusterwide mutual information. J Comput Biol **11**(1) (2004) 147–61
18. Murtagh, F.: Multidimensional clustering algorithms. In Physica-Verlag, V., ed.: COMPSTAT Lectures 4. (1985)
19. Kaufman, L., Rousseuw, P.: Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley & Sons, Inc. (1990)
20. Berkhin, P.: Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA (2002)
21. Azuaje, F., Bolshakova, N.: Cluster validation techniques for genome expression data. Signal Processing **83**(4) (2003) 825–833
22. Cancello, R., Henegar, C., Viguerie, N., Taleb, S., Poitou, C., Rouault, C., Coupaye, M., Pelloux, V., Hugol, D., Bouillot, J., Bouloumie, A., Barbatelli, G., Cinti, S., Svensson, P., Barsh, G., Zucker, J., Basdevant, A., Langin, D., Clement, K.: Reduction of macrophage infiltration and chemoattractant gene expression changes in white adipose tissue of morbidly obese subjects after surgery-induced weight loss. Diabetes **54**(8) (2005) 2277–86
23. Feve, B.: Adipogenesis: cellular and molecular aspects. Best Pract Res Clin Endocrinol Metab **19**(4) (2005) 483–99
24. Pedersen, T., Kowenz-Leutz, E., Leutz, A., Nerlov, C.: Cooperation between C/EBPalpha TBP/TFIIB and SWI/SNF recruiting domains is required for adipocyte differentiation. Genes Dev **15**(23) (2001) 3208–16
25. Charriere, G., Cousin, B., Arnaud, E., Andre, M., Bacou, F., Penicaud, L., Casteilla, L.: Preadipocyte conversion to macrophage. Evidence of plasticity. J Biol Chem **278**(11) (2003) 9850–5

Bayesian Learning of Markov Network Structure

Aleks Jakulin[1] and Irina Rish[2]

[1] Department of Statistics, Columbia University, 1255 Amsterdam Ave, New York, NY 10027, USA
jakulin@acm.org

[2] IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532, USA
rish@us.ibm.com

Abstract. We propose a simple and efficient approach to building undirected probabilistic classification models (Markov networks) that extend naïve Bayes classifiers and outperform existing directed probabilistic classifiers (Bayesian networks) of similar complexity. Our Markov network model is represented as a set of consistent probability distributions on subsets of variables. Inference with such a model can be done efficiently in closed form for problems like class probability estimation. We also propose a highly efficient Bayesian structure learning algorithm for conditional prediction problems, based on integrating along a hill-climb in the structure space. Our prior based on the degrees of freedom effectively prevents overfitting.

1 Introduction

Learning probabilistic models from data has been an area of active and fruitful research in machine learning due to several reasons. First, despite its simplicity, the naïve Bayes (NB) classifier demonstrated surprisingly high accuracy in many domains, and became a popular choice in practice. Its success also led to multiple extensions that attempted to further improve the performance of naïve Bayes by incorporating higher-order dependencies (e.g., tree-augmented naive Bayes and Bayesian networks [1]). Second, in practical applications we are often interested not just in accurate classification, but also in accurate estimation of class probability for solving ranking and cost-based decision problems. Moreover, we may need to learn joint distribution models that allow answering various probabilistic queries besides computing the conditional class probability. A popular choice are graphical probabilistic models such as Markov and Bayesian networks, which also have an advantage of interpretability as they explicitly represent interactions among features.

In this paper, we propose a simple and efficient Bayesian approach that learns undirected probabilistic models (Markov networks). We evaluate our approach on the tasks of class probability estimation and classification. We have chosen undirected models over directed ones since computing the conditional class probability is an easy inference problem that does not require an explicit model of a joint distribution provided by a Bayesian network; it suffices to have an unnormalized representation given by a set of potentials in a Markov network. We also adopt a discriminative structure learning approach [2,3,4], using a conditional likelihood function to score model structures. Being Bayesian about the structure, we integrate it out, rather than search for a single optimal structure. Our empirical results demonstrate that such Bayesian approach frequently outperforms existing directed probabilistic classifiers of similar complexity (e.g., Bayesian networks with same maximal clique size), while also being extremely fast, sometimes order of magnitudes faster than some competing approaches.

2 Related Work

Most of previous work on probabilistic classifiers focused on directed models, or Bayesian networks. However, we decided to focus on undirected graphical models (Markov networks) since learning explicit (normalized) joint probability distribution $P(\mathbf{X}, Y)$, as in case of Bayesian networks, is unnecessary if our goal is just computing the conditional class probability $P(Y|\mathbf{X})$. This is an easy inference problem even with an unnormalized distribution represented by a Markov network. Undirected models permit the inclusion of a larger number of connections between variables, as we are no longer restricted by the decomposability requirements imposed by the chain rule.

Previous approaches to learning Markov networks often focused on bounded-treewidth models [5,6,7,8], in order to bound the inference complexity; again, this restriction is unnecessary if we are only concerned with the queries described above. In our approach, we only have to bound the *original* hyperedge cardinality in a Markov network, for the sake of representation efficiency. Note that removing the bounded-treewidth constraint allows to account for important k-way interactions between the variables that the corresponding bounded-treewidth model would ignore.

Note that despite being related, our approach is also different from the conditional random fields (CRFs) [9]. We focus on “standard” i.i.d. rather than sequential non-i.i.d. classification problem, and learn a Markov network over the features and class, rather than (conditional) Markov network (random field) over a sequence of dependent class labels. Extending our approach to CRFs would be an interesting direction for future work. Our Bayesian prior which depends on the complexity of the structure can be seen as an approach to penalization of complex structure, just as the maximum-margin criterion penalizes unusually oriented decision boundaries.

3 Markov Network Models

3.1 Notation and Overview

Let $\mathbf{X} = \{X_1, \ldots, X_n\}$ be a set of observed random variables, called *attributes*, and let $\mathbf{x} = (x_1, \ldots, x_n)$ be a vector of values assigned to variables in $\mathbf{X}$. Herein, we assume discrete-valued attributes, i.e. $\mathbf{x} \in \mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ where each range $\mathcal{X}_i$ is a set of possible values of X_i. Let Y denote an unobserved random variable called the *class*, where $y \in \mathcal{Y}, |\mathcal{Y}| = m$. The set of attributes together with the class (i.e., all variables) is denoted $\mathbf{V} = \mathbf{X} \cup \{Y\}$. An assignment $\mathbf{v}^{(i)} = (\mathbf{x}^{(i)}, y^{(i)})$ of values to the attributes and the class is called an *instance*, or *example* with index (i). We will use a short notation $P(\mathbf{v}) = P(\mathbf{x}, y) = P(x_1, \ldots, x_n, y)$ to describe the joint probability distribution $P(X_1 = x_1, \ldots, X_n = x_n, Y = y)$.

Our models will have the undirected structure of Markov networks. We will define a *Markov network*, or *Markov random field* on random variables $\mathbf{V}$ as $\langle \mathcal{M}, \mathcal{T} \rangle$ where $\mathcal{M}$ is an (undirected) hypergraph $\mathcal{M} = \{\mathcal{S}_1, S_2, \ldots, S_\ell\}$ and $\mathcal{T} = (\Phi_1, \ldots, \Phi_\ell)$ is a set of positive functions, called *potentials* for each of the ℓ hyperedges[1] in $\mathcal{M}$, such that the joint distribution $\hat{P}(\mathbf{v})$ factorizes over them: $\hat{P}(\mathbf{v}) = (1/Z) \prod_{i=1}^{\ell} \Phi(\mathbf{v}_i)$ where Z is a normalization constant. This latter form is referred to as the Gibbs distribution. We use $\hat{P}(\cdot)$ as a shorthand for $P(\cdot|\langle \mathcal{M}, \mathcal{T} \rangle)$. Each hyperedge S_R contains the variables linked to it. These variables form a vector $\mathbf{V}_R$. The potential $\Phi(\mathbf{v}_R)$ corresponding to each hyperedge then maps any combination of values of $\mathbf{v}_R$ into a positive real number.

We now outline our algorithm for class probability estimation. The outline contains many terms that will be defined later, in the section referenced for each step.

1. Given $\mathbf{V} = \mathbf{X} \cup \{Y\}$, and a bound k on hyperedge cardinality, select a set of hyperedges $\mathcal{M} = \{M | M \subseteq \mathbf{Y}\}$ using the approach described in Sect. 4.2.
2. Given $\mathcal{M}$, compute the region graph $\mathcal{R}$ using the *cluster variation method (CVM)*[10] where each hyperedge corresponds to an initial region (Sect. 3.2). The region graph captures the overlap between hyperedges.
3. For each region R estimate the submodel $P(\mathbf{V}_R)$ from data (Sect. 4.1). Each submodel is an ordinary probabilistic model, but for a subset of variables.
4. Approximate $P(\mathbf{V})$ by the product $\Phi(\mathbf{v}) = \prod_{\langle R, c_R \rangle \in \mathcal{R}} P(\mathbf{v}_R)^{c_R}$ where c_R is the counting number for region R in the region graph (Sect. 3.2).
5. Normalize $\hat{P}(y|\mathbf{x}) = \Phi(\mathbf{x}, y) / \sum_{y'} \Phi(\mathbf{x}, y')$; classify $y^*(\mathbf{x}) = \arg\max_y P(y|\mathbf{x})$.

3.2 Computing the Potentials

The general problem with learning Markov networks from data once the structure is known is how to obtain potentials from the data. Specifically, we tractably express the potentials in terms of *submodels*, where a submodel $P(\mathbf{v}_R)$ is a probability distribution or mass function on the subset of variables corresponding to each hyperedge. Each submodel is estimated from the data. We then make use of the following recursive definition of potentials Φ_R [7]:

$$\Phi_R(\mathbf{v}_R) \triangleq \frac{P(\mathbf{v}_R)}{\prod_{R' \subset R} \Phi_{R'}(\mathbf{v}_{R'})}. \tag{1}$$

A particular $P(\mathbf{v}_S)$, $S \subset R_1$ is computed by marginalizing $P(\mathbf{v}_{R_1})$, which in turn is modeled directly from data. As S may be a part of another hyperedge

[1] Usually referred to as 'cliques', but with hypergraphs the notion of a clique could be confusing.

$S \subset R_2$, there could be several versions of $P(\mathbf{v}_S)$, depending on what submodel is marginalized (R_1 or R_2). To assure *consistency* we require that there exists some hypothetical $P(\mathbf{V})$ so that each $P(\mathbf{v}_R)$ is its marginalization.

It is of practical convenience to construct an intermediate data structure called a *region graph* $\mathcal{R}$ [10]. Table 1 shows a variant of the *cluster variation method* algorithm [10] for constructing a region graph from the set of hyperedges. The region graph is defined as $\mathcal{R} = \{\langle R, c_R \rangle, R \subseteq \mathbf{V}\}$, where for each region R, there is a corresponding *counting number* c_R, that accounts for the overlaps between regions, and helps avoid the double-counting of evidence.

Given the region graph, we can compute the joint probability distributions in terms of the Kikuchi approximation to probability [11,12]:

$$\hat{P}(\mathbf{v}) = \frac{1}{Z} \prod_{\langle R, c_R \rangle \in \mathcal{R}} P(\mathbf{v}_R)^{c_R}. \tag{2}$$

It is well-known [13] that when the Markov network is triangulated and thus yields a clique tree, the Gibbs distribution can be represented exactly through (2) and no normalization is needed, as $P(\mathbf{v}) = \prod_{R \in \mathcal{R}} \Phi_R(\mathbf{v}_R)$, where the potentials $\Phi_R(\mathbf{v}_R)$ are defined by (1). In general, when the counting numbers are greater than zero only for the initial regions, the recursive definition of potentials is exact [10].

Table 1. Cluster variation method for constructing the region graph given a set of hyperedges $\mathcal{M} = \{S_1, S_2, \ldots, S_\ell\}$

```
$\mathcal{R}_0 \leftarrow \{\emptyset\}$ {Redundancy-free set of hyperedges.}
**for all** $S \in \mathcal{M}$ **do** {for each hyperedge}
  **if** $\forall S' \in \mathcal{R}_0 : S \nsubseteq S'$ **then**
    $\mathcal{R}_0 \leftarrow \mathcal{R}_0 \cup \{S\}$ {$S$ is not redundant}
  **end if**
**end for**
$\mathcal{R}_0 \leftarrow \{\langle S, 1 \rangle;\ S \in \mathcal{R}_0\}$
$k \leftarrow 1$
**while** $|\mathcal{R}_{k-1}| > 2$ **do** {there are feasible subsets}
  $\mathcal{R}_k \leftarrow \{\emptyset\}$
  **for all** $\mathcal{I} = S^\dagger \cap S^\ddagger : S^\dagger, S^\ddagger \in \mathcal{R}_{k-1}, \mathcal{I} \notin \mathcal{R}_k$ **do** {feasible intersections}
    $c \leftarrow 1$ {the counting number}
    **for all** $\langle S', c' \rangle \in \mathcal{R}, \mathcal{I} \subseteq S'$ **do**
      $c \leftarrow c - c'$ {consider the counting numbers of all regions containing the
      intersection}
    **end for**
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{\langle \mathcal{I}, c \rangle\}$
    $\mathcal{R}_k \leftarrow \mathcal{R}_k \cup \{\mathcal{I}\}$
  **end for**
**end while**
**return** $\{\langle R, c \rangle \in \mathcal{R}; c \neq 0\}$ {Region graph.}
```

3.3 Performing Inference

While in general it is NP-hard to compute $P(\mathbf{Q}|\mathbf{E})$, where $\mathbf{Q} \subseteq \mathbf{V}$, $\mathbf{E} \subseteq \mathbf{V}$, in a Markov network representing a joint distribution $P(\mathbf{V})$, the problem becomes easy when the number of unobserved variables $\mathbf{V} \setminus \mathbf{E}$ is *small*, or when the *treewidth* of the network is small. Treewidth, also known as induced width, is a graph parameter that controls the complexity of some commonly used probabilistic inference algorithms (the complexity is exponential in the treewidth). The treewidth of a network, given a particular variable ordering, equals to largest clique size of the *triangulated* network, where the triangulation is performed along the given ordering and reflects the process of creating new probabilistic functions by the inference algorithm.

Given a set of random variables $\mathbf{V} = \mathbf{X} \cup \{Y\}$, a set $\mathcal{R} = \{R | R \subseteq \mathbf{V}\}$ of subsets (regions) of $\mathbf{V}$, where Y belongs to at least one region, and a product $\Phi(\mathbf{v}) = \Phi(\mathbf{x}, y) = \prod_{R \in \mathcal{R}} \Phi_R(\mathbf{v}_R)$ of non-negative functions (potentials) defined on these regions, let $\hat{P}(\mathbf{v}) = (1/Z)\Phi(\mathbf{v})$ be the corresponding joint probability distribution over $\mathbf{V}$, where Z is a normalization constant. It is very easy to see that:

1. Computing $\hat{P}(Y|\mathbf{x})$ does not require global normalization, i.e. $\hat{P}(Y|\mathbf{x}) = \Phi(\mathbf{x}, Y) / \sum_{y'} \Phi(\mathbf{x}, y')$;[2]
2. The classifier can be computed using a product of only those potentials that contain Y, i.e. $h^*(\mathbf{x}) = \arg\max_y \prod_{\{R \in \mathcal{R} | Y \in R\}} \Phi_R(\mathbf{v}_R)$.[3]

Of course, this holds also when we have several query variables $\mathbf{Y}$, but only the vector is short. More complex queries (e.g. with missing data) might require several iterations, where each individual iteration can take the simple form as for inferring the class probability.

4 Bayesian Structure Learning

The above formulation of the Markov network model allows efficient inference. The task for learning is to determine the parameters of the model: the *structure* and the *submodels.* We will adopt the Bayesian framework, based on an explicit description of the model in terms of its parameters $\phi = \langle \mathcal{M}, \boldsymbol{\Theta}, \vartheta \rangle$, where $\mathcal{M}$ is the model structure (hypergraph), while ϑ and $\boldsymbol{\Theta}$ are the submodel prior and the submodel parameters, respectively. Each submodel $\mathbf{V}_R$ is specified in terms of a parameter vector $\boldsymbol{\theta}_R$, so that $P(\mathbf{V}_R | \boldsymbol{\theta}_R)$.

[2] Indeed, the first claim follows from $\hat{P}(y|\mathbf{x}) = \hat{P}(\mathbf{x}, y)/\hat{P}(\mathbf{x}) = (1/Z)\Phi(\mathbf{x}, y)/\sum_{y'}(1/Z)\Phi(\mathbf{x}, y')$, since by definition $\Phi(\mathbf{v}) = \Phi(\mathbf{x}, y)$.

[3] The second claim is easily obtained from the definition of Bayesian classifier, $h^*(\mathbf{x}) = \arg\max_y \hat{P}(y|\mathbf{x})$, and the following observation: $\hat{P}(y|\mathbf{x}) = \frac{\Phi(\mathbf{x},y)}{\sum_{y'} \Phi(\mathbf{x},y')} = \frac{\prod_{\{Q \in \mathcal{R} | Y \notin Q\}} \Phi(\mathbf{v}_Q)}{\sum_{y'} \Phi(\mathbf{x},y)} \prod_{\{R \in \mathcal{R} | Y \in R\}} \Phi_R(\mathbf{v}_R)$, where $(\prod_{\{Q \in \mathcal{R} | Y \notin Q\}} \Phi(\mathbf{v}_Q))/\sum_{y'} \Phi(\mathbf{x}, y)$ is independent of Y.

We will assume a prior distribution over structures $P(\mathcal{M})$, and a prior distribution over the submodel parameters $P(\boldsymbol{\Theta}|\vartheta)$. The prior for the whole model is then $P(\phi) = P(\mathcal{M})P(\vartheta)P(\boldsymbol{\Theta}|\vartheta) = P(\mathcal{M})P(\vartheta)\prod_R P(\boldsymbol{\theta}_R|\vartheta)$. Because we assume independence of $\boldsymbol{\Theta}$ and $\mathcal{M}$, the submodels remain the same irrespectively of the structure: this results in a major speed-up.

The Bayesian paradigm (to be distinguished from the Bayes rule) is that one should be uncertain about what the exact model is. Instead of finding the 'best' model parameters, we assign probabilities to each setting of ϕ, 'averaging' together a weighted ensemble of models (both structures and parameters). For prediction we make use of all plausible structures instead of arbitrarily picking just the best one [14]. This has also been shown to improve results in practice [15]. In a class probability estimation setting, the final result of our inference based on data $\mathcal{D}$ will be the following class predictive distribution:

$$P(y|\mathbf{x}) \propto \int P(\phi|\mathcal{D})P(y|\mathbf{x},\phi)d\phi \tag{3}$$

Here, $P(y|\mathbf{x},\phi)$ is based on (2). For efficiency purposes, we employ the formulation of Bayesian model averaging [16], where only those parameter values with a sufficiently high posterior probability are remembered and used.

4.1 Parameters for Consistent Submodels

Our Markov network model is based on partially overlapping submodels. Although technically not necessary, it is desirable for the submodels to be consistent in the sense that all of them are marginalizations of some joint model. We model the submodels on discrete variables as multinomials with a symmetric Dirichlet prior:

$$P(\boldsymbol{\theta}_R|\vartheta) = \text{Dirichlet}(\alpha_R, \ldots, \alpha_R), \quad \alpha_R = \frac{\vartheta}{\prod_{V \in R} |\mathcal{V}|}$$

Here, $|\mathcal{V}|$ denotes the number of values that variable V can take. It is easy to prove that this prior assures that all the posterior mean submodels are consistent if the same value of ϑ was used for each of them. This prior is best understood as the expected number of outliers: to any data set, we add ϑ instances uniformly distributed across the space of variables. We have set the parameter $P(\vartheta = 1) = 1$, which means that one outlier per dataset was assumed: we see this to be a reasonable prior assumption that speeds up the learning. Due to conjugacy of the Dirichlet prior, the desired posterior mean probability given data $\mathcal{D}$ within region R is:

$$P(\mathbf{v}_{\boldsymbol{R}}|\mathcal{D},\vartheta) = \frac{\vartheta/|\mathcal{V}_R| + \sum_i^{|\mathcal{D}|} \mathbb{I}\{\mathbf{v}_R^{(i)} = \mathbf{v}_R\}}{|\mathcal{D}| + \vartheta}.$$

4.2 Structure Learning

Parsimonious Structures. The structure in the context of our Markov network model is simply a selection of the submodels. $P(\mathcal{M})$ models our prior expectations about the structure of the model. We will now introduce a parsimonious prior that asserts a higher prior probability to simpler selections of submodels, and a lower prior probability to complex selections of submodels as to prevent overfitting. A quantification of complexity based on degrees of freedom is given by [17]. In many practical applications we are not interested in the joint model. Instead, we want to predict labels Y from attributes $\mathbf{X}$. In such cases, a considerable part of uncertainty about the value of $\mathbf{X}$ gets canceled out, and the effective degrees of freedom are fewer ("Conditional density estimation is easier than joint density estimation.").

Let us assume a set of overlapping submodels of the vector $\mathbf{V}$, and the resulting region graph $\mathcal{R}$ obtained using the CVM. The number of *degrees of freedom* of the model $\mathcal{M}$ with a corresponding region graph $\mathcal{R}$ intended for predicting Y from $\mathbf{X}$ is:

$$df_{\mathcal{M}_Y} \triangleq \sum_{\langle S,c\rangle \in \mathcal{R}} c\left(\prod_{V \in S} |\mathcal{V}| - \prod_{\substack{V \in \mathcal{S} \\ V \neq Y}} |\mathcal{V}| \right) \tag{4}$$

V is either Y or a part of $\mathbf{X}$, and $\mathcal{V}$ is the number of values V can take. This quantification accounts for overlap between submodels in the same fashion as cluster variation method does for probabilities. Of course, conditional modeling corresponds to joint modeling when $Y = \emptyset$.

The following prior corresponds to the assumption of exponentially decreasing prior probability of a structure with an increasing number of degrees of freedom (or effective parameters):

$$P(\mathcal{M}) \propto e^{-df_{\mathcal{M}}}. \tag{5}$$

The likelihood function for conditional modeling can also be adjusted to account for the fact that we will be using the model for predicting Y from $\mathbf{X}$. The non-Bayesian approach searches for the structure that yields the maximum conditional likelihood [4]. A Bayesian approach instead scores structures by the means of a conditional likelihood function, as is customary in Bayesian regression. We hereby use the following conditional likelihood function that assumes i.i.d.:

$$P(\mathbf{v}^{(1)\dots(m)}|\phi) \triangleq \prod_{i=1}^{m} P(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \phi) \tag{6}$$

Because $\mathcal{M}$ was assumed to be independent of ϑ and $\boldsymbol{\Theta}$, we prepare $\boldsymbol{\Theta}$ in advance, before assessing $\mathcal{M}$. The $P(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \mathcal{M})$ is obtained using (2).

Sampling the Structure Space. In the process of structure learning, we perform a walk in the space of structures. For all practical purposes, we are not interested in the 'best' structure, but the walk should nevertheless attempt to visit more structures with high posterior probability than structures with low posterior probability, as the latter do not affect the predictive distribution (3) much. While similar MCMC approaches have been proposed in the past [14], we apply a simple hill-climbing approach that does not faithfully model the posterior distribution over structures, but does improve the predictive performance for a considerably lower computational cost.

During the hill climb, we seek to greedily maximize the posterior probability of a structure. Let us assume that we are performing conditional modeling, with the intention of predicting Y. Our initial structure will have a single initial hyperedge of cardinality 1, $\{Y\}$. In the successive step, we will consider all possible attributes X_i creating hyperedges $\{X_i\} \cup \{Y\}$, and pick the one that yields the highest posterior probability: this corresponds to step-wise forward selection algorithm with one-step look-ahead. This approach is very efficient: including a new hyperedge corresponds to just multiplying the predictions for an individual instance with another term and renormalizing. With the considerable increase in performance that ensues, we can afford to find the best hyperedge at every step of the forward selection. All the models that were evaluated are included in the model average: even if they were not selected, they might still have a relatively high posterior probability.

With the above algorithm we can discover very interesting structures in a very short amount of time. An example of a maximum posterior probability structure for the tic-tac-toe dataset is shown in Fig. 1: the structure was obtained in 0.03 seconds on an ordinary laptop computer. The hyperedges correspond to meaningful notions of corner and center points, to connections between them, and finally to the diagonals and edges: indeed these structures are what humans examine when playing the game. Another example of structures obtained with our algorithm appears in Fig. 2.

In the past we evaluated interactions one by one and formed the structure from such marginal evaluations [11]. However, the results are considerably better when performing evaluations of complete structures. This effectively implies that inclusions of individual interactions for the model structure are not independent decisions.

5 Empirical Evaluation

To validate our modeling approach from Sections 3 and 4, we have applied the methodology to the problem of class-probability estimation. Numerous techniques exist for this purpose, and they can be roughly divided into those that pursue a discriminative structure, yet employ the generative chain rule (such as the naïve Bayes, tree-augmented naïve Bayes [1] and general Bayesian network classifiers [4]) and those that employ both discriminative structure and discriminative parameter values [3,19,20]. It is widely recognized that it is generally too hard to perform both general structure search and optimization of discriminative parameter values. Still, a limited amount of structure selection is performed even with discriminative parameter values, such as step-wise model selection [21] or TAN-like structures [3,20], but rarely one can afford an exhaustive search for interactions.

We will evaluate the benefit gained by a) allowing hyperedges that result in cyclic dependencies, b) the benefits of Bayesian model averaging, and c) verifying if our prior protects against overfitting. Furthermore, we compare our approach to other related approaches.

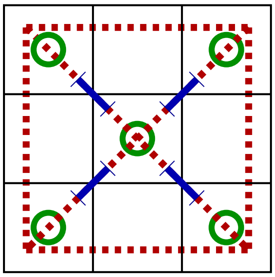


Fig. 1. Hyperedges of cardinality 4 are not merely a theoretical curiosity. In this illustration we show the tic-tac-toe game board, which comprises 9 squares, each corresponding to a 3-valued variable with the range $\{\times, \circ, \_\}$. The goal is to develop a predictive model that will indicate if a board position is winning for $\times$ or not: this is the 2-valued class variable. The illustration shows the hyperedges in the MAP model identified by our algorithm: 2-way hyperedges (5 green circles), 3-way hyperedges (4 blue serif lines), and 4-way hyperedges (6 red dashed lines). Each hyperedge includes the class (not shown).

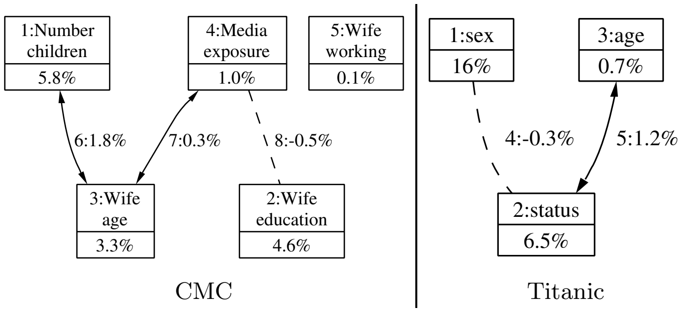


Fig. 2. This figure shows the Bayesian model average for two real-life datasets: CMC (contraception use in Indonesia) and Titanic (survival of Titanic passengers). Each node and each connection is numbered with the step of the hill-climb when it was selected. We can observe the order in which the edges entered the model: in the case of Titanic, the ordering was [sex,survival], [status,survival], [age,survival], followed by the two 3-variable hyperedges, [status, sex, survival] and [status of the passenger, age, survival]. The percentages indicate the interaction information [18] expressed as a proportion of class entropy: which helps understand the nature of the hyperedge. For example, bi-directed arrows indicate synergies between variables (such as between age and number of children: it helps to distinguish young women with children from young women without children for predicting the contraception method used), and dashed links indicate partial redundancies between variables (such as between media exposure and education: the more education, the greater the media exposure). Synergies and redundancies explain the reason for including complex hyperedges, and allow interpreting the model.

To evaluate a class-probability estimate, we will use the expected negative log-likelihood (log-loss) of class assignment $-E[\log P(y|\mathbf{x})]$. For each UCI data set, we performed 5 replications of 5-fold cross-validation. The data sets were all

Table 2. A comparison of different undirected and directed probability models on 46 datasets. NB is naïve Bayes, TN is tree-augmented naïve Bayes, BC is the discriminative search for Bayesian network classifiers [4], M2-M4 is the maximum a posteriori Markov network with structure search with maximum hyperedge cardinality of 2 through 4, B2-B4 are corresponding Bayesian model averaged Markov networks, and BT is the Bayesian model averaging (BMA) on cycle-free Markov hypertrees with hyperedges of cardinality less than 4. The best result is typeset in bold, the results of those methods that outperformed the best method in at least 2 of the 25 experiments on each dataset are underlined (because they are not significantly worse). The worst result is marked with (·). At the bottom we list the average rank of a method across all the datasets, both for log-loss (LL) and error rate (ER).

	log-loss / instance									
domain	NB	TN	BC	M2	M3	M4	B2	BT	B3	B4
adult	·*0.42*	0.33	0.39	0.31	0.30	0.30	0.31	0.30	0.30	**0.30**
audiology	3.55	·*5.56*	2.95	1.69	1.69	1.69	1.65	1.65	**1.65**	**1.65**
glass	1.25	·*1.76*	1.21	1.19	1.19	1.19	1.10	1.10	**1.10**	**1.10**
horse-colic	1.67	·*5.97*	3.36	0.84	0.85	0.85	0.82	0.82	**0.82**	**0.82**
krkp	·*0.29*	0.19	0.12	0.26	0.08	0.05	0.26	0.11	0.08	**0.05**
lung	5.41	·*6.92*	3.05	3.12	3.12	3.12	2.15	2.15	**2.15**	**2.15**
lymph	1.10	1.25	1.23	1.04	1.04	1.04	0.90	0.90	**0.90**	**0.90**
monk2	0.65	0.63	0.61	·*0.65*	0.53	0.43	0.65	0.60	0.53	**0.43**
p-tumor	3.17	·*4.76*	2.84	2.58	2.58	2.58	2.51	2.51	**2.51**	**2.51**
promoters	0.60	·*3.14*	2.56	0.67	0.67	0.67	0.55	0.55	**0.55**	**0.55**
soy-large	0.57	0.47	0.71	0.47	0.47	0.47	0.44	0.44	**0.44**	**0.44**
spam	·*0.53*	0.32	0.32	0.21	0.19	0.19	0.21	0.19	0.19	**0.19**
tic-tac-toe	·*0.55*	0.49	0.52	0.53	0.40	0.03	0.53	0.53	0.40	**0.03**
titanic	0.52	0.48	0.48	·*0.52*	0.48	0.48	0.52	0.48	0.48	**0.48**
zoo	0.38	0.46	0.51	0.28	0.28	0.28	0.24	0.24	**0.24**	**0.24**
segment	0.38	1.06	·*1.29*	**0.17**	**0.17**	**0.17**	**0.17**	**0.17**	**0.17**	0.17
cmc	1.00	·*1.03*	1.00	0.93	0.94	0.94	0.93	**0.92**	0.92	0.92
heart	1.25	·*1.53*	1.38	1.10	1.12	1.12	1.10	**1.09**	1.09	1.09
ionosphere	0.64	0.74	·*1.70*	0.38	0.40	0.40	0.34	**0.32**	0.32	0.32
vehicle	·*1.78*	1.14	1.29	0.81	0.72	0.72	0.80	**0.69**	0.69	0.69
wdbc	0.26	0.29	0.39	0.14	0.15	0.15	0.13	**0.13**	0.13	0.13
australian	0.46	·*0.94*	0.78	0.37	0.40	0.41	**0.36**	0.38	0.38	0.39
balance	**0.51**	·*1.13*	0.74	0.51	0.57	0.57	**0.51**	0.52	0.52	0.52
breast-LJ	0.62	0.89	0.80	0.57	0.69	0.69	**0.56**	0.59	0.59	0.59
breast-wisc	0.21	0.23	0.25	0.17	0.21	0.21	**0.17**	0.18	0.18	0.18
crx	0.49	·*0.93*	0.91	0.37	0.38	0.38	**0.35**	0.36	0.36	0.36
german	0.54	·*1.04*	1.00	0.53	0.70	0.70	**0.53**	0.64	0.64	0.64
hepatitis	0.78	·*1.31*	1.11	0.48	0.58	0.58	**0.44**	0.44	0.44	0.44
lenses	2.44	·*2.99*	1.15	0.69	0.69	0.69	**0.37**	0.37	0.37	0.37
post-op	0.93	1.78	1.25	0.80	0.80	0.80	**0.68**	0.68	0.68	0.68
voting	·*0.60*	0.53	0.48	0.16	0.23	0.23	**0.15**	0.15	0.15	0.15
hayes-roth	0.46	·*1.18*	0.76	**0.45**	**0.45**	**0.45**	0.45	0.45	0.45	0.45
monk1	·*0.50*	0.09	0.09	0.49	0.08	**0.00**	0.49	0.08	0.08	0.00
pima	0.50	0.49	0.50	**0.48**	0.49	0.51	0.48	0.48	0.48	0.48
ecoli	0.89	0.94	**0.67**	0.91	0.91	0.91	0.85	0.85	0.85	0.85
iris	0.27	0.32	**0.20**	0.28	0.28	0.28	0.23	0.23	0.22	0.22
monk3	0.20	0.11	**0.08**	·*0.20*	0.11	0.11	0.20	0.11	0.11	0.11
o-ring	0.83	0.76	**0.59**	1.14	1.14	1.14	0.68	0.68	0.67	0.67
bupa	0.62	**0.60**	0.61	0.62	0.62	0.63	0.62	0.61	0.61	0.61
car	0.32	**0.18**	0.18	0.32	0.19	0.19	·*0.32*	0.19	0.19	0.19
mushroom	·*0.01*	**0.00**	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00
shuttle	0.16	**0.06**	0.06	0.16	0.06	0.06	·*0.17*	0.06	0.06	0.06
soy-small	0.00	**0.00**	·*0.39*	0.05	0.05	0.05	0.03	0.03	0.03	0.03
anneal	**0.07**	0.17	·*0.24*	0.09	0.09	0.09	0.10	0.09	0.09	0.09
wine	**0.06**	0.29	·*0.46*	0.17	0.17	0.17	0.13	0.13	0.13	0.13
yeast-class	**0.01**	0.03	·*1.96*	0.23	0.23	0.23	0.21	0.21	0.21	0.21
avg rank (LL)	7.45	7.74	7.41	6.03	6.00	6.00	4.67	3.66	3.17	**2.86**
avg rank (ER)	5.84	6.24	5.85	6.58	5.47	5.28	6.04	4.82	4.49	**4.40**

discretized with the Fayyad-Irani method [22] beforehand. The missing values were interpreted as special values. Structure learning with our procedure for all the 46 datasets using our method implemented in Python and C++ took less than 9 minutes, in comparison to over 686 minutes consumed by a C++ implementation of Bayesian network classifiers which also yielded worse performance.

Judging from the rankings in Table 2, we can conclude that the single best-performing feature is Bayesian model averaging: it has consistently outperformed the maximum a posteriori structures. The second important conclusion is that our Bayesian prior successfully prevents overfitting in a systematic way: as we increase the depth of structure search, the results improve (although B2 does win by performing essentially just feature selection in a number of cases when there seem to be no higher-order hyperedges). The third conclusion is that Markov networks perform well regardless of whether the task is classification (as assessed via error rate), or class probability estimation (log-loss). The fourth conclusion is that allowing cycles does help, but not in a radical way (of course this may be simply due to our simplified way of computing potentials).

6 Conclusion

In summary, we feel that undirected models have many advantages over directed models: it is not possible or at least controversial to establish causal direction from observational data. Our definition of potentials avoids problems with sparse conditional probability tables. Our priors and Bayesian model averaging work surprisingly well and effectively prevent overfitting. Our heuristic structure search is also much faster than most alternatives, and we hope that it would inspire others not to rigidly follow the posterior sampling approach in complex parameter spaces, but instead to seek combining search and averaging. Because methods such as Bayesian logistic regression continue to outperform our approach on datasets without interactions, it would be highly desirable to combine the handling of higher-order interactions in Markov networks with effective discriminative parameter learning in regression models. This could perhaps be achieved by finding discriminative parameters for well-performing discriminative structures, or by finding an equally efficient way of performing inference on Markov networks but for the specific purpose of conditional prediction.

References

1. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning **29** (1997) 131–163
2. Salojärvi, J., Puolamäki, K., Kaski, S.: On discriminative joint density modeling. In: Proceedings of 16th European Conference on Machine Learning. Volume 3270 of Lecture Notes in Artificial Intelligence., Berlin, Germany, Springer-Verlag (2005) 341–352
3. Pernkopf, F., Bilmes, J.: Discriminative versus generative parameter and structure learning of Bayesian network classifiers. In: Proc. 22nd ICML, Bonn, Germany, ACM Press (2005) 657–664

4. Grossman, D., Domingos, P.: Learning Bayesian network classifiers by maximizing conditional likelihood. In: Proc. 21st ICML, Banff, Canada, ACM Press (2004) 361–368
5. Chow, C.K., Liu, C.N.: Approximating discrete probability distributions with dependence trees. IEEE Trans. on Information Theory **14**(3) (1968) 462–467
6. Meilă, M., Jordan, M.I.: Learning with mixtures of trees. Journal of Machine Learning Research **1** (2000) 1–48
7. Srebro, N.: Maximum likelihood bounded tree-width Markov networks. In: Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI). (2001) 504–511
8. Bach, F., Jordan, M.: Thin junction trees. In: Advances in Neural Information Processing Systems 14. (2002) 569–576
9. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. of the International Conference on Machine Learning (ICML). (2001) 282–289
10. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Constructing free-energy approximations and generalized belief propagation algorithms. IEEE Transactions on Information Theory **51**(7) (2005) 2282–2312
11. Jakulin, A., Rish, I., Bratko, I.: Kikuchi-Bayes: Factorized models for approximate classification in closed form. Technical Report RC23314, IBM (2004)
12. Santana, R.: Estimation of distribution algorithms with Kikuchi approximations. Evolutionary Computation **13**(1) (2005) 67–97
13. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, San Francisco, CA, USA (1988)
14. Friedman, N., Koller, D.: Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. Machine Learning **50** (2003) 95–126
15. Cerquides, J., López de Màntaras, R.: Tractable Bayesian learning of tree augmented naive Bayes classifiers. In: Proc. 20th ICML. (2003) 75–82
16. Hoeting, J.A., Madigan, D., Raftery, A.E., Volinsky, C.T.: Bayesian model averaging: A tutorial. Statistical Science **14**(4) (1999) 382–417
17. Krippendorff, K.: Information Theory: Structural Models for Qualitative Data. Volume 07–062. Sage Publications, Inc., Beverly Hills, CA (1986)
18. Jakulin, A., Bratko, I.: Analyzing attribute dependencies. In: Proc. of Principles of Knowledge Discovery in Data (PKDD), Springer-Verlag (2003) 229–240
19. Greiner, R., Su, X., Shen, B., Zhou, W.: Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. Machine Learning **59** (2005) 297–322
20. Jing, Y., Pavlovic, V., Rehg, J.M.: Efficient discriminative learning of Bayesian network classifiers via boosted augmented naive Bayes. In: Proc. 22nd ICML, Bonn, Germany, ACM Press (2005) 369–376
21. Roos, T., Wettig, H., Grünwald, P., Myllymäki, P., Tirri, H.: On discriminative Bayesian network classifiers and logistic regression. Machine Learning **59** (2005) 267–296
22. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI 1993, AAAI Press (1993) 1022–1027

Approximate Policy Iteration for Closed-Loop Learning of Visual Tasks

Sébastien Jodogne, Cyril Briquet, and Justus H. Piater

University of Liège — Montefiore Institute (B28)
B-4000 Liège, Belgium
{S.Jodogne, C.Briquet, Justus.Piater}@ULg.ac.be

Abstract. *Approximate Policy Iteration* (API) is a reinforcement learning paradigm that is able to solve high-dimensional, continuous control problems. We propose to exploit API for the closed-loop learning of mappings from images to actions. This approach requires a family of function approximators that maps visual percepts to a real-valued function. For this purpose, we use Regression Extra-Trees, a fast, yet accurate and versatile machine learning algorithm. The inputs of the Extra-Trees consist of a set of visual features that digest the informative patterns in the visual signal. We also show how to parallelize the Extra-Tree learning process to further reduce the computational expense, which is often essential in visual tasks. Experimental results on real-world images are given that indicate that the combination of API with Extra-Trees is a promising framework for the interactive learning of visual tasks.

1 Introduction

Since the rise of embedded CCD sensors, many robots are nowadays equipped with cameras and, as such, face input spaces that are extremely high-dimensional and potentially very noisy. Therefore, though real-world visual tasks can often be solved by directly connecting the visual space to the action space (i.e. by learning a direct image-to-action mapping), such mappings are especially hard to derive by hand and should be *learned* by the robotic agent. The latter class of problems is commonly referred to as *Vision-for-Action*. A breakthrough in modern AI would be to design an artificial system that would acquire object or scene recognition skills using only its interactions with the environment.

In this paper, an algorithm is introduced for closed-loop learning of image-to-action mappings. Our algorithm is defined within the biologically-inspired framework of *reinforcement learning* (RL) [1]. RL models an agent that learns a percept-to-action mapping through its interactions with the environment: It is only implicitly guided through a *reinforcement signal*, which is generally delayed.

RL is an attractive framework for Vision-for-Action problems. Unfortunately, basic RL algorithms are highly sensitive to the noise and to the dimensionality of the percepts, which forbids their direct use when solving visual tasks. We have previously proposed an algorithm that adaptively discretizes the visual space into a small number of visual classes [2]. Schematically, a decision tree is progressively built that tests, at each of its nodes, the presence of one highly informative image pattern (a *visual feature*). The tree is incrementally refined in a sequence of attempts to remove perceptual aliasing. This dramatically reduces the size of the input space, so that standard RL algorithms become usable.

One might wonder, however, whether the feature selection process is actually desirable, as it might introduce a high variance in the computed image classifiers, just as in the case of incremental learning of decision trees [3]. Furthermore, selecting visual features requires the introduction of an equivalence relation between the features. This relation is difficult to define. Often, a fixed threshold on a metric in feature space is used. However, modifying this threshold can lead to significant changes in the learned image-to-action mapping.

We describe a method that uses the whole set of visual features, without taking decisions involving a subset of informative features, and without relying on a similarity measure between them. In other words, we exploit the *raw* visual features. As visual data can be sampled only sparsely, we resort to the embedding of function approximators inside the RL process. Among the RL algorithms that use function approximators, we use *Approximate Policy Iteration* (API) [1]. API together with a linear approximation architecture has already been proposed in the context of continuous state spaces, giving rise to the *Least-Squares Policy Iteration* algorithm [4].

The *Visual Approximate Policy Iteration* (V-API) is defined, which is an instance of API designed to work in visual spaces. V-API uses Regression Extra-Trees, a family of nonparametric function approximators. This choice is motivated by the low bias and variance, as well as the good performance in generalization of the Extra-Trees [3]. Furthermore, Classification Extra-Trees are successful for solving image classification tasks [5]. To the best of our knowledge, this makes of V-API the first application of API to high-dimensional *discrete* spaces. Thus, V-API is potentially of major interest, as it proves that fully automatic, nonparametric RL methods can succeed in visual tasks. Likewise, the embedding of Extra-Trees in API is novel, and should also be useful in continuous state spaces.

Even if the computational expense of Extra-Trees is small with respect to other machine learning algorithms, the complexity of visual spaces still prevents their direct use in V-API. An additional contribution is to parallelize the Extra-Trees learning algorithm, which greatly reduces the execution time of V-API.

The paper is organized as follows. Firstly, we discuss the three important tools that are used inside V-API, namely the Modified Policy Iteration algorithm, the extraction of visual features and the Regression Extra-Trees. Then, we formally describe V-API and the distributed learning of Extra-Trees. Finally, we conclude with experimental results on a complex, visual navigation task.

2 Theoretical Background

2.1 Modified Policy Iteration

V-API is defined in the framework of *Reinforcement Learning* (RL). In RL, the environment is traditionally modeled as a *Markov Decision Process* (MDP). An

MDP is a quadruple $\langle S, A, \mathcal{T}, \mathcal{R} \rangle$, where S is a finite set of *states* or *percepts*[1], A is a finite set of *actions*, $\mathcal{T}$ is a probabilistic *transition function* from $S \times A$ to S, and $\mathcal{R}$ is a *reinforcement signal* from $S \times A$ to $\mathbb{R}$. An MDP obeys the following discrete-time dynamics: If at time t, the agent takes the action a_t while the environment lies in a state s_t, the agent perceives a numerical reinforcement $r_{t+1} = \mathcal{R}(s_t, a_t)$, then reaches some state s_{t+1} with probability $\mathcal{T}(s_t, a_t, s_{t+1})$. Therefore, from the point of view of the agent, an *interaction* with the environment is summarized as a quadruple $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$.

A (deterministic) *percept-to-action mapping* (or, equivalently, a *control policy*) is a fixed function $\pi : S \mapsto A$ from percepts to actions. Each control policy π is associated with a *state-action value function* $Q^\pi(s, a)$ that gives, for each state $s \in S$ and each action $a \in A$, the expected discounted return obtained by starting from state s, taking action a, and thereafter following π:

$$Q^\pi(s, a) = \mathrm{E}^\pi \left\{ \sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s, a_0 = a \right\}, \tag{1}$$

where $\gamma \in [0, 1[$ is the *discount factor* that gives the current value of the future reinforcements, and where E^π denotes the expected value if the agent follows the mapping π. Theory shows that all the optimal policies for a given MDP share the same Q function, denoted Q^* and called the *optimal state-action value function*, that always exists. Once the optimal state-action value function Q^* is known, an optimal percept-to-action mapping π^* is easily derived by choosing:

$$\pi^*(s) = \operatorname*{argmax}_{a \in A} Q^*(s, a), \text{ for each } s \in S. \tag{2}$$

Dynamic Programming (DP) is a set of algorithmic methods for solving MDPs. DP algorithms assume the knowledge of the transition function $\mathcal{T}$ and of the reinforcement signal $\mathcal{R}$. The well-known *Modified Policy Iteration* (MPI) [6] is an important DP algorithm that will be useful in the sequel. Starting with an initial, arbitrary percept-to-action mapping π_0, MPI builds a sequence of increasingly better policies $\pi_1, \pi_2, \ldots$ by relying on two interleaved learning processes: (1) *policy estimation* (the *critic* component), which computes the Q^{π_k} state-action value function of the current policy π_k; and (2) *policy improvement* (the *actor* component), which uses Q^{π_k} to generate an improved policy π_{k+1}. The algorithm stops when there is no change between successive policies. Here is a brief description of the two processes:

Policy estimation: Q^{π_k} is computed by building a sequence of state-action value functions $Q_0, Q_1, \ldots$ until convergence using the update rule:

$$Q_{i+1}(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in S} \mathcal{T}(s, a, s') Q_i(s', \pi(s')). \tag{3}$$

After convergence, Bellman's theorem [1] shows that $Q_{i_k} = Q^{\pi_k}$. Q_0 can be chosen freely (generally, $Q_0(s, a) = 0$ for each $s \in S$ and $a \in A$).

[1] MDP assumes the full observability of the environment, which allows us to talk indifferently about states and percepts. In visual tasks, S is a set of images.

Policy improvement: At each state $s \in S$, $\pi_k(s)$ is replaced by the action with the best state-action value (as computed by the policy estimation process):

$$\pi_{k+1}(s) = \operatorname*{argmax}_{a \in A} Q^{\pi_k}(s, a). \tag{4}$$

Finally, reinforcement learning is defined as the counterpart of DP when the transition function $\mathcal{T}$ and the reinforcement signal $\mathcal{R}$ are unknown. The input of RL algorithms is basically a database of interactions $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$.

2.2 Extraction of Visual Features

Because standard RL algorithms rely on a tabular representation of the value functions, they quickly become impractical as the number of possible percepts increases. This is evidently a problem in visual tasks, as images are high-dimensional and noisy. Similar problems often arise in many fields of Computer Vision.

For this purpose, the popular, highly successful *local-appearance methods* have been introduced [7,8]. They postulate that, to take the right decision in a visual problem, it is often sufficient to focus one's attention only on a few interesting patterns occurring in the images. They summarize the images as a set of *visual features*, that are vectors of real numbers. Formally, they introduce a *feature transform* $F : S \mapsto \mathcal{P}(\mathbb{R}^n)$, where S is the set of images and $\mathcal{P}$ denotes the power set. For an image $s \in S$, $F(s)$ typically contains between 10 and 1000 visual features. Most feature transforms have in common that (1) they select *interest points* in the image (e.g. by detecting discontinuities in the visual signal), and (2) they compute a *local description* of the neighborhood of the interest points. As an illustration, here are two possible choices of feature transforms:

1. "Traditional" feature transforms, that use a standard interest point detector (Harris, Harris-affine, Hessian-Laplace,...) [7] in conjunction with a standard local descriptor (steerable filters, local jet, SIFT,...) [8].
2. Randomized feature transforms. They randomly select a fixed number of subwindows in the image (a subwindow is a rectangle that has an arbitrary position, scale and orientation). Then, the subwindows are downscaled to a patch of fixed size (typically 11×11), in a fixed colorspace (graylevel, RGB or HSV). This simple approach is very fast, as well as highly successful [5].

V-API uses such methods to digest an image into a set of raw visual features.

2.3 Regression Extra-Trees

As discussed in the Introduction, V-API rely on Extra-Trees [3]. We restrict our study of Extra-Trees to the case where all attributes (both the inputs and the output) are numerical. An Extra-Tree model is constituted by a forest of M independent decision trees. Each of their internal nodes is labeled by a threshold on one of the input attributes, that is to be tested in that node. The leaves are labeled by a regression output. The regression response for a sample is obtained by computing the response of each subtree. This is achieved by starting at the root node, then progressing down the tree according to the result of the thresholding tests found during the descent, until a leaf is reached. By doing so, each subtree votes for a regression output. Finally, the mean of these outputs is assigned to the sample.

The subtrees are built in a top-down fashion, by successively splitting the leaf nodes where the output variable varies. For each input variable, the algorithm computes its variation bounds and uniformly chooses one random threshold between those bounds. Once a threshold has been chosen for every input variable, the split that gives the best score on the regression output is kept. The tree construction is stopped when the output is constant in the leaf. This will guarantee that learning bias is small, as well as learning variance thanks to the aggregation of a sufficient number of randomized trees.

Algorithms 1 and 2 describe how to build an Extra-Tree model. In this pseudocode, $\boldsymbol{x}_i \in \mathbb{R}^n$ contains the input attributes of the ith sample in the learning set and $y_i \in \mathbb{R}$ is the observed regression output for this sample. We assume the existence of a function $\texttt{score}(\{\langle \boldsymbol{x}_i, y_i \rangle\}, v, t)$ that returns the score of the threshold t on the variable v in the database $\{\langle \boldsymbol{x}_i, y_i \rangle\}$. In our implementation, variance reduction was used as the `score` function. These algorithms are identical to those presented by Geurts et al. [3] and are restated here to complement the description of our distributed algorithms (cf. Section 3.5). We refer the reader to Geurts et al. [3] for a complete and thorough treatment.

3 Visual Approximate Policy Iteration

3.1 Nonparametric Approximate Policy Iteration

In the next sections, a general, nonparametric RL version of *Approximate Policy Iteration* (API) is described. Nonparametric API is a generalization of MPI that can use any kind of nonparametric function approximators. This is in contrast to *Least-Squares Policy Iteration* [4] that explicitly targets continuous state spaces and uses linear approximation. The two components of MPI (policy estimation and improvement) are adapted so as to compute the state-action value function of a policy without relying on any knowledge of the underlying MDP.

The existence of an oracle called `learn` is assumed, as we are concerned with nonparametric function approximators. Given a database of samples $\langle s_t, a_t, v_t \rangle$, where s_t is a state, a_t is an action and v_t is a real number, `learn` builds a function approximator that represents a state-action value function $Q : S \times A \mapsto \mathbb{R}$ that is the closest possible to the given sample distribution. For instance, Algorithm 1 constitutes one possible oracle that is suitable for problems with continuous state space. An oracle for visual spaces will be introduced in Section 3.4.

3.2 Representation of the Generated Policies

Nonparametric API relies on the state-of-the-art principle that was proposed in Least-Squares Policy Iteration [4]: Any state-action value function $Q(s, a)$ induces a *greedy policy* $\widetilde{\pi}[Q]$ that always selects the action maximizing Q:

Algorithm 1. — General structure for Regression Extra-Tree learning

```
1: extra-trees($\{\langle \boldsymbol{x}_i, y_i \rangle\}, M$) :–
2:    $\mathcal{T} := \phi$
3:    for $i := 1$ to $M$ do
4:       $\mathcal{T} := \mathcal{T} \cup \{$subtree($\{\langle \boldsymbol{x}_i, y_i \rangle\}$)$\}$
5:    end for
6:    return $\mathcal{T}$
```

Algorithm 2. — Recursive induction of one single subtree

```
1: subtree($\{\langle \boldsymbol{x}_i, y_i \rangle\}$) :–
2:    if too few samples or each input $\boldsymbol{x}_i$ is constant or $y_i$ is constant then
3:       $o := \text{mean}(\{y_i\})$
4:       return a leaf labeled with output $o$
5:    else
6:       for $v := 1$ to $n$ do
7:          $a, b := \min_i\{x_{i,v}\}, \max_i\{x_{i,v}\}$
8:          $t[v]$ := random value in $[a, b]$
9:          $s[v]$ := score($\{\langle \boldsymbol{x}_i, y_i \rangle\}, v, t[v]$)
10:      end for
11:      $v^* := \text{argmax}_v\{s[v]\}$
12:      $i_\ominus := \{i \mid x_{i,v*} < t[v^*]\}$
13:      $i_\oplus := \{i \mid x_{i,v*} \geq t[v^*]\}$
14:      $T_\ominus$ := subtree($\{\langle \boldsymbol{x}_i, y_i \rangle\} \mid i \in i_\ominus$)
15:      $T_\oplus$ := subtree($\{\langle \boldsymbol{x}_i, y_i \rangle\} \mid i \in i_\oplus$)
16:      return a binary decision node $\langle t[v^*], T_\ominus, T_\oplus \rangle$
17:   end if
```

Algorithm 3. — Nonparametric Approximate Policy Iteration

```
1: approximate-policy-iteration ($\{\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle\}$) :–
2:    $k := 0$
3:    $Q^{\pi_0}$ := learn ($\{\langle s_t, a_t, r_{t+1} \rangle\}$)
4:    loop
5:       $Q^{\pi_{k+1}}$ := estimation ($\widetilde{\pi}\,[Q^{\pi_k}]$)
6:       if $||Q^{\pi_{k+1}} - Q^{\pi_k}||_\infty < \varepsilon$ then
7:          return $\widetilde{\pi}\,[Q^{\pi_k}]$
8:       end if
9:       $k := k + 1$
10:   end loop
```

$$\widetilde{\pi}\,[Q]\,(s) = \underset{a \in A}{\text{argmax}}\, Q(s, a), \text{ for each } s \in S.^2 \tag{5}$$

This property enables us to deal only with state-action value functions, and never directly with policies. So, there is no need of a separate representation system for policies. In terms of the notation of Section 2.1, nonparametric API

[2] This maximization can easily be achieved, as the action space is assumed to be finite.

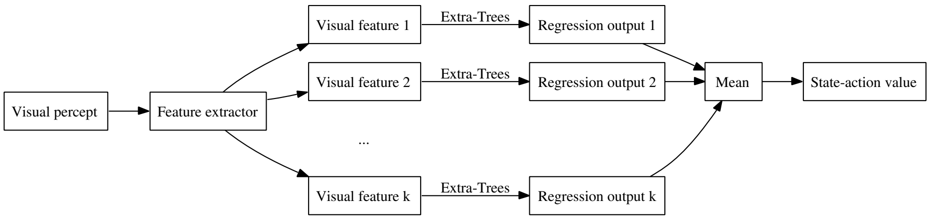


Fig. 1. Computing the state-action value of a visual percept for a given action

does not keep track of $\pi_k = \widetilde{\pi}\left[Q^{\pi_{k-1}}\right]$, but only of Q^{π_k}. The policy π_k is evaluated on demand from $Q^{\pi_{k-1}}$ through Equation 5. Thanks to this implicit representation of the policies, the policy improvement step becomes trivial: It is sufficient to define $Q^{\pi_{k+1}}$ as the state-action value function of the greedy policy with respect to Q^{π_k}, that is computed by the policy estimation component.

Algorithm 3 summarizes the backbone of nonparametric API. The input of the algorithm is a database of interactions $\langle s_t, a_t, r_{t+1}, s_{t+1}\rangle$, which makes of nonparametric API an off-policy, model-free RL algorithm. The third line builds an initial policy π_0 that maximizes immediate reinforcements. The algorithm stops when the difference between two successive Q^{π_k} drops below a threshold.

3.3 Nonparametric Approximate Policy Estimation

The `estimation` component in Algorithm 3 computes the state-action value function $Q^{\pi_{k+1}}$ of a policy $\widetilde{\pi}\left[Q^{\pi_k}\right]$ given the input database of interactions $\langle s_t, a_t, r_{t+1}, s_{t+1}\rangle$. To this end, a sequence of state-action value functions $Q_i(s,a)$ is generated. This is done according to the principle of Modified Policy Iteration, but the functions Q_i are now function approximators that are built through the `learn` oracle. As the underlying MDP is unknown, the update rule of Equation 3 cannot be used. Instead, we use the stochastic version of this equation:

$$Q_{i+1}(s,a) = \mathcal{R}(s,a) + \gamma\ Q_i\left(\delta(s,a), \widetilde{\pi}\left[Q^{\pi_k}\right](\delta(s,a))\right). \tag{6}$$

Hence, the update rule that is induced by the database of interactions is:

$$Q_{i+1} := \texttt{learn}\left(\{\langle s_t,\ a_t,\ r_{t+1} + \gamma\ Q_i\left(s_{t+1}, \widetilde{\pi}\left[Q^{\pi_k}\right](s_{t+1})\right)\rangle\}\right). \tag{7}$$

A new Extra-tree model is learned through each application of this update rule. This learning process stops when the difference between two successive Q_i drops below a threshold. Q_0 can be chosen freely. In practice, Q_0 is set to Q^{π_k}. This is a starting point that reduces the number of iterations before convergence, as the policy $\widetilde{\pi}\left[Q^{\pi_{k+1}}\right]$ generally shares common decisions with $\widetilde{\pi}\left[Q^{\pi_k}\right]$. This algorithm can be motivated similarly than *Fitted Q Iteration* [9]: The stochastic aspect of the environment will eventually be captured by the function approximators.

3.4 Visual State-Action Value Function Approximators

So far, we have not defined a family of function approximators that is suitable for visual tasks. As motivated in the Introduction, we propose to take advantage of the raw visual features. Therefore, Regression Extra-Trees cannot be used directly, for two reasons: (1) The action input is discrete, and cannot be fed into a Regression Extra-Tree model as defined in Section 2.3; and (2) feature transforms map *one* visual percept to *many* visual features (cf. Section 2.2).

The solution to the first problem is straightforward: The single Extra-Tree model $Q(s, a)$ is replaced by $|A|$ Extra-Tree models $Q_a(s)$, one for each possible action $a \in A$. The latter problem is more fundamental, and is solved by applying the Extra-Trees model independently on each visual feature in the input image. This process generates one regression output per visual feature. Then the value of the function approximator is defined as the mean of these regression outputs. This approach is depicted in Figure 1, and is directly inspired by recent, successful results about image classification through Extra-Trees [5].

The `learn` oracle for this type of function approximators is now defined formally. Given a database of samples $\langle s_t, a_t, v_t\rangle$ (where s_t are images), the Extra-Trees model Q_a that corresponds to the action $a \in A$ is defined as:

$$Q_a(s) := \texttt{extra-trees}\left(\{\langle \boldsymbol{x}_i, y_i\rangle \mid (\exists t \in T_a)(\boldsymbol{x}_i \in F(s_t) \wedge y_i = v_t)\};\ M\right), \quad (8)$$

where F is the used feature transform, and $T_a = \{t \mid a_t = a\}$ is the set of time stamps for the interactions that are labeled with the action a. Intuitively, for each sample $\langle s_t, a_t, v_t\rangle$, all the visual features in the image s_t are associated with the value v_t in the database that is used to train the Extra-Tree model Q_{a_t}.

Nonparametric API along with such a family of function approximators will be referred to as the *Visual Approximate Policy Iteration* (V-API) algorithm.

3.5 Parallelizing Extra-Trees

V-API applies Regression Extra-Trees on large databases. This induces a high computational cost for the learning of Extra-Trees. We propose to reduce this computational expense by taking advantage of the extremely parallelizable nature of Algorithm 1: Each execution of Algorithm 2 is totally independent of other instances of the same algorithm, and each subtree can be computed in a separate computational task. In other words, the learning of Extra-Tree models can be formulated as a so-called *bag of tasks*, where tasks are independent and can be processed on separate computer nodes. Once all the tasks are completed, it is straightforward to merge all the subtrees to generate the Extra-Tree model.

Our implementation of Regression Extra-Trees follows this principle. The resulting speedup is huge: If N homogeneous hosts are used, the computation time roughly equals $\lceil M/N\rceil T + U$, where M is the parameter of Algorithm 1, T is the mean amount of time for building one single subtree, and U corresponds to the distribution time of the database among all the hosts.

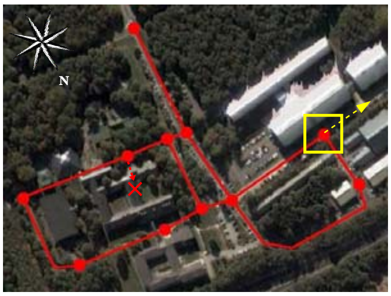
Note that if no attention is paid, the transmission overhead U can quickly become a bottleneck. Indeed, the same large database has to be sent to N hosts by the central task manager, which causes both reduction of bandwidth and augmentation of network congestion effects. One possible solution is to rely on the use of UDP multicast. Unfortunately, due to the lack of flow control in UDP, slow hosts will be overwhelmed by the massive amount of data they receive.

We have therefore used the peer-to-peer BitTorrent protocol [10] to distribute the databases. Schematically, in BitTorrent, each host becomes part of a swarm that grabs all the pieces of a file. Whenever a host acquires a piece, this piece is made available for download to the other hosts. A distinguished host, the *tracker*, is used to keep track of the hosts that belong to the swarm. This approach for file distribution is elegant and scalable, and indeed highly reduces the transmission overhead U, making it roughly independent of the number of hosts in the cluster. Our solution should be useful in many other distributed computing applications, and especially in the context of distributed data mining and Grid computing.

4 Experimental Results

We have applied the V-API to a simulated navigation task. In this task, the agent moves between 11 distinct locations of our campus (cf. Figure 2 (a)). Every time the agent is at one of the 11 locations, its body can aim at 4 possible orientations: North, South, West, East. It can take 3 different actions: Turn left, turn right, go forward. Its goal is to enter the Montefiore Institute, where it gets a reward of 100. Turning left or right induces a penalty of -5, and moving forward, a penalty of -10. The discount factor γ was set to 0.8.

The agent does not have access to its position and its orientation. Rather, it only perceives a picture of the area that is in front of it. So, the agent has to connect images directly to the appropriate reactions without knowing the underlying physical structure of the task. For each possible location and each possible viewing direction, a database of 24 images of size 1024×768 with viewpoint changes was collected. Those 44 databases were divided into a learning set of 18 images and a test set of 6 images (cf. Figure 2 (b)). V-API has been applied on a static database of 10,000 interactions that has been collected using a fully randomized exploration policy. The same database is used throughout the entire V-API algorithm, and only contains images that belong to the learning set.

(a)


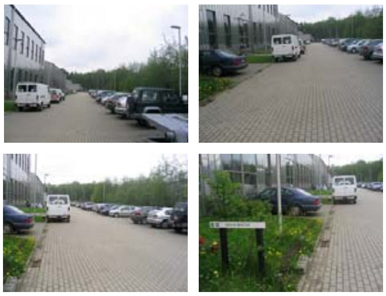
(b)


Fig. 2. (a) Navigation around Montefiore Institute. Red spots corresponds to the places between which the agent moves. The agent can only follow the links between the different spots. On this map, Montefiore Institute is labeled by a red cross. (b) The percepts of the agent. Four different percepts are shown that correspond to the location and viewing direction marked in yellow on the map on the left.

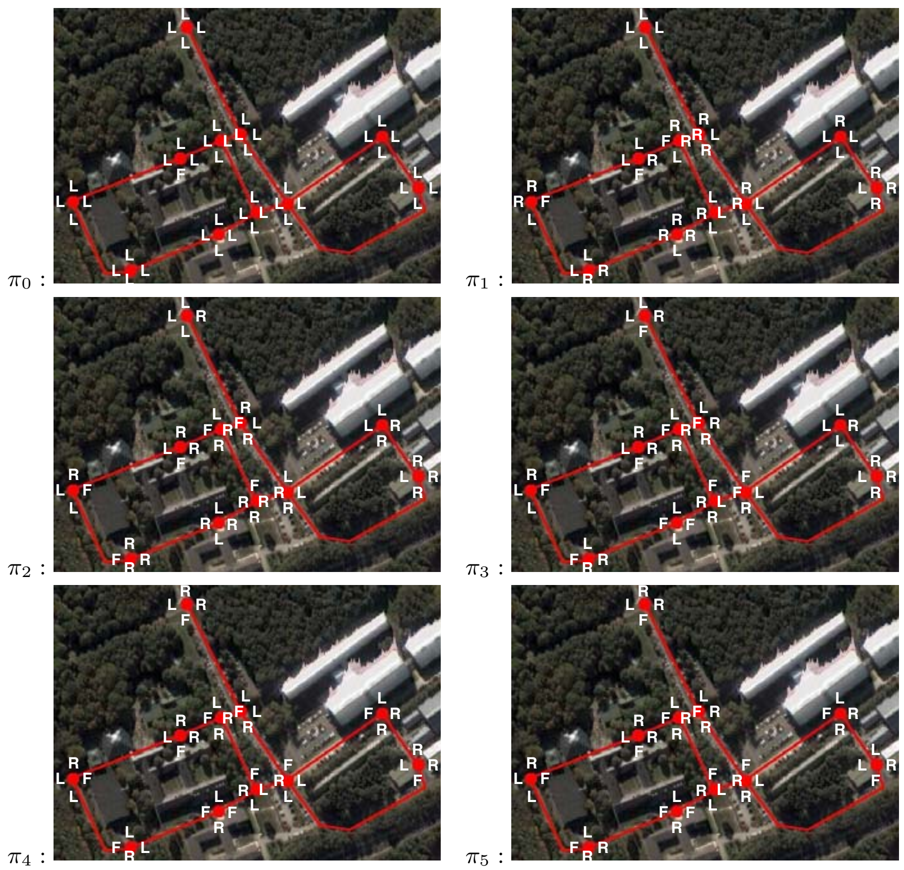


Fig. 3. The sequence of policies π_k generated by V-API. At each location, 4 letters from the set {F, L, R} are written, one for each viewing direction. Each letter represents the action (go Forward, turn Left, turn Right) that receives the majority of votes in the learning set, for the corresponding pair *location / viewing direction*.

In our experiments, we have used traditional feature transforms with the SIFT descriptors (whose dimension is $n = 128$) [11]. This choice is mostly arbitrary. Randomized feature transforms are promising, and will be investigated in future work. The parallel implementation of Regression Extra-Trees runs on a testbed cluster of $N = 67$ heterogeneous ix86 machines. It consists of a set of 27 AMD Athlon 1800+, 27 Intel Celeron 2.4Ghz, and 13 Intel Pentium IV 2.8Ghz CPUs. They are interconnected via a switched 100Mbps Ethernet network.

Figure 3 shows the sequence of policies that is generated by V-API. The algorithm stops after 6 iterations, which is a surprisingly small number. A total of 147 Extra-Tree models were generated that correspond to $49 = 147/|A|$ visual state-action value functions (as defined in Section 3.4). The overall running time was about 98 hours. This shows the interest of taking advantage of the intrinsic parallelism of Extra-Trees, as this has divided the running time by about fifty.

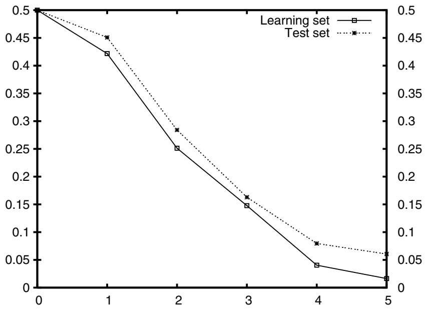


Fig. 4. Policy error as a function of the step counter k. The solid (resp. dashed) plot corresponds to the error rate of the policy π_k on the learning (resp. test) set.

Statistics about the generated policies are given in Figure 4. The error of the last policy in the generated sequence was 1.6% on the learning set and 6.6% on the test set, with respect to the optimal policy when the agent has direct access to its position and viewing direction. These error rates are better than those obtained through RLVC (2% on the learning and 14% on the test set) [12].

The advantage of using BitTorrent is clear: In optimal conditions, if m Extra-Tree models (as discussed above, $m = 147$) are to be built from databases of typical size $S = 80$MB, the transmission overhead corresponding to FTP would roughly equal $U \approx m \times N \times S[\text{MB}]/100[\text{Mbps}] = 17$ hours. Using BitTorrent, this time is approximately reduced to $U \approx m \times S[\text{MB}]/100[\text{Mbps}] = 16$ minutes.

5 Conclusions

We have introduced the Visual Approximate Policy Iteration (V-API) algorithm. V-API is designed for the closed-loop solution of visual control problems. It extensively relies on the use of Regression Extra-Trees as function approximators. The embedding of Extra-Trees inside the framework of API is a first important contribution of this paper. Experiments indicate that the algorithm is sound and applicable to non-trivial visual tasks. V-API outperforms RLVC in terms of performance in generalization over the test set. Future work will demonstrate that the proposed combination of nonparametric API with Extra-Trees is a convenient choice for RL in high-dimensional, continuous state spaces.

We have also shown how to take advantage of the highly parallelizable nature of the induction of Extra-Trees by distributing the construction of the subtrees over a cluster of computers. This allows us to greatly reduce the computational time, which is often an important issue with visual spaces. Of course, other fields of application of Extra-Trees, such as supervised learning [3], image classification [5] and the *Fitted Q Iteration* algorithm [9], will directly benefit from this new economical advantage. Finally, the peer-to-peer BitTorrent protocol was shown to be an effective tool for reducing the database distribution expense.

Unfortunately, even when taking advantage of a cluster of computers, the running time of the algorithm is still relatively long. A compromise seems to exist between the requirement of an equivalence relation among visual features, as in the algorithm RLVC [2], and the use of raw visual features, as in V-API. RLVC runs faster, does not require a cluster of computers, and can be used to generate higher-level visual features [13]. On the other hand, V-API benefits from the full discriminative power of the visual features, exhibits the low variance and bias of Extra-Trees, and requires less parameter tuning. Therefore, V-API and RLVC constitute two complementary techniques. An interesting open question is whether the advantages of these two techniques can be combined.

Acknowledgments. We kindly acknowledge P. Geurts and the PEPITe S.A. team (`http://www.pepite.be/`) for providing us with an implementation of the Extra-Trees, which has inspired our parallel implementation. We also wish to thank Prof. L. Wehenkel for helpful comments on a preliminary version of this paper, and S. Martin for his valuable help regarding network programming and BitTorrent's protocol.

References

1. Bertsekas, D., Tsitsiklis, J.: Neuro-Dynamic Programming. Athena Scientific (1996)
2. Jodogne, S., Piater, J.: Interactive learning of mappings from visual percepts to actions. In De Raedt, L., Wrobel, S., eds.: Proc. of the 22nd International Conference on Machine Learning (ICML), Bonn (Germany), ACM (2005) 393–400
3. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning **36**(1) (2006) 3–42
4. Lagoudakis, M., Parr, R.: Least-squares policy iteration. Journal of Machine Learning Research **4** (2003) 1107–1149
5. Marée, R., Geurts, P., Piater, J., Wehenkel, L.: Random subwindows for robust image classification. In: IEEE Conference on Computer Vision and Pattern Recognition. Volume 1., San Diego (CA, USA) (2005) 34–40
6. Puterman, M., Shin, M.: Modified policy iteration algorithms for discounted Markov decision problems. Management Science **24** (1978) 1127–1137
7. Schmid, C., Mohr, R., Bauckhage, C.: Evaluation of interest point detectors. International Journal of Computer Vision **37**(2) (2000) 151–172
8. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. Volume 2., Madison (WI, USA) (2003) 257–263
9. Ernst, D., Geurts, P., Wehenkel, L.: Tree-based batch mode reinforcement learning. Journal of Machine Learning Research **6** (2005) 503–556
10. Cohen, B.: Incentives build robustness in BitTorrent. In: Proc. of the Workshop on Economics of Peer-to-Peer Systems. (2003)
11. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2) (2004) 91–110
12. Jodogne, S., Piater, J.: Learning, then compacting visual policies (extended abstract). In: Proc. of the 7th European Workshop on Reinforcement Learning (EWRL), Napoli (Italy)(2005) 8–10
13. Jodogne, S., Scalzo, F., Piater, J.: Task-driven learning of spatial combinations of visual features. In: Proc. of the IEEE Workshop on Learning in Computer Vision and Pattern Recognition, San Diego (CA, USA), IEEE (2005)

Task-Driven Discretization of the Joint Space of Visual Percepts and Continuous Actions

Sébastien Jodogne and Justus H. Piater

University of Liège — Montefiore Institute (B28)
B-4000 Liège, Belgium
{S.Jodogne, Justus.Piater}@ULg.ac.be

Abstract. We target the problem of closed-loop learning of control policies that map *visual* percepts to *continuous* actions. Our algorithm, called *Reinforcement Learning of Joint Classes* (RLJC), adaptively discretizes the joint space of visual percepts and continuous actions. In a sequence of attempts to remove perceptual aliasing, it incrementally builds a decision tree that applies tests either in the input perceptual space or in the output action space. The leaves of such a decision tree induce a piecewise constant, optimal state-action value function, which is computed through a reinforcement learning algorithm that uses the tree as a function approximator. The optimal policy is then derived by selecting the action that, given a percept, leads to the leaf that maximizes the value function. Our approach is quite general and applies also to learning mappings from *continuous* percepts to continuous actions. A simulated visual navigation problem illustrates the applicability of RLJC.

1 Introduction

Reinforcement Learning (RL) [1,2] is an attractive framework for the automatic design of robotic controllers. RL algorithms are indeed able to learn direct mappings from percepts to actions given a set of interactions of the robotic agent with its environment. These algorithms build on a careful analysis of a so-called *reinforcement signal* that implicitly defines the task to be solved. Using RL potentially simplifies the design process, as real-world robotic applications are in general difficult to model and to solve directly in a programming language.

Unfortunately, although robotic controllers often interact with their environment through a set of continuously-valued actions (position, velocity, torque,...), relatively little consideration has been given to the development of RL algorithms that learn direct mappings from percepts to *continuous* actions. This is in contrast to continuous perceptual spaces, for which many solutions exist. The challenge of continuous actions spaces arises from the fact that standard update rules based upon Bellman's optimality equations are only applicable on finite sets of actions, as they rely on a maximization over the action space. Furthermore, an *a priori* discretization of the action space generally suffers from an explosion of the representational size of the domains known as the *curse of dimensionality*, and may introduce artificial noise.

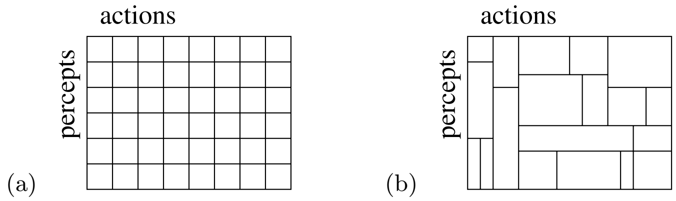


Fig. 1. Illustration of the discretization process of (a) RLVC, and (b) RLJC

Previously-investigated solutions for handling continuous actions without *a priori* discretization generally use function approximators such as neural networks [3], tile coding [4], or wire fitting [5]. However, to the best of our knowledge, none of these methods can cope simultaneously with high-dimensional, *discrete* perceptual spaces. As a consequence, vision-based robotic tasks with continuous output such as visual servoing cannot currently be solved through RL. This paper presents the *Reinforcement Learning of Joint Classes* (RLJC) algorithm, which enables such closed-loop learning of direct mappings from images to continuous actions. RLJC is a generalization of the *Reinforcement Learning of Visual Classes* (RLVC) algorithm [6] to continuous actions.

RLJC discretizes the problem space by applying tests in the input perceptual space *and* in the output action space, i.e. by testing the presence of *perceptual features* and of *action features*. For example, when the input of the agent is a binary number, suitable perceptual features could be tests on a single bit of the input. Similarly, if uni-dimensional continuous actions are considered, an action feature could be a real number that would serve as a threshold. RLJC progressively subdivides the combined percept-action (or *joint*) space, in a sequence of attempts to remove perceptual aliasing. In each region that is induced by the discretization process, the state-action value functions are constant. This way, the uncountable joint space is mapped to a finite number of regions, and specific RL algorithms are then used to extract the optimal control policies. Very importantly, the discretization process is *adaptive*: A new split occurs only when it succeeds at distinguishing between two regions of the problem space that have dissimilar properties with respect to the optimal value function. Therefore, the discretization of the action space can be inhomogeneous with respect to the perceptual space, and the action space can possibly be discretized differently at each percept. This difference is illustrated in Figure 1.

The idea of discretizing the joint space is also present in the *JoSTLe* algorithm [7], an extension of *Variable Resolution Grids* [8]. However, JoSTLe is specifically designed for *continuous* perceptual spaces, as it heavily relies on Kuhn triangulations of the joint space. Conversely, RLJC is not limited to discrete perceptual spaces, and it can also be applied to continuous perceptual spaces. Indeed, RLJC only requires that features can be defined on the perceptual and action spaces. Therefore, one key advantage of RLJC lies in its generality. Experimental results on a simulated navigation task indicate that RLJC is a promising framework for the interactive learning of visual tasks.

2 Reinforcement Learning of Visual Classes

2.1 Theoretical Background

The *Reinforcement Learning of Visual Classes* (RLVC) [6] algorithm is first described and will serve as a basis for the *Reinforcement Learning of Joint Classes* (RLJC) algorithm[1]. RLVC is a *Reinforcement Learning* (RL) algorithm [1,2].

In RL, the environment is modeled as a set S of *states* or *percepts*[2], and the agent interacts with it through a set A of *actions*. The environment obeys a stationary discrete-time dynamics: If at time t, the agent takes the action a_t while the environment lies in a state s_t, the state s_{t+1} is reached with probability $\mathcal{T}(s_t, a_t, s_{t+1})$. A stationary *reinforcement signal* $\mathcal{R} : S \times A \mapsto \mathbb{R}$ gives a quantitative evaluation of taking an action in the presence of a percept. This signal is possibly delayed, meaning that a good (resp. bad) reaction is not required to be rewarded (resp. penalized) immediately. Therefore, an *interaction* with the environment is summarized as a quadruple $\langle s_t, a_t, r_{t+1}, s_{t+1}\rangle$. If S and A are finite, the quadruple $\langle S, A, \mathcal{T}, \mathcal{R}\rangle$ is known as a *Markov Decision Process* (MDP).

A stationary *percept-to-action mapping* (or *control policy*) is a function $\pi : S \mapsto A$ that links the percepts to the actions. Any control policy π induces a *value function* $V^\pi : S \mapsto \mathbb{R}$ that corresponds to the expected discounted return over time if that policy is followed from a given percept $s \in S$:

$$V^\pi(s) = \mathrm{E}^\pi \left\{ \sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s \right\}, \text{ for each } s \in S, \tag{1}$$

where $\gamma \in [0, 1[$ is the *discount factor* that gives the current value of the future reinforcements. The goal of RL is to learn an *optimal policy* π^* that maximizes the value function for all the percepts $s \in S$. The value function V^* of an optimal policy π^* is unique and is called the *optimal value function*. RL algorithms are able to extract an optimal policy π^* from a database of interactions $\langle s_t, a_t, r_{t+1}, s_{t+1}\rangle$ without relying on any knowledge of $\mathcal{T}$ or $\mathcal{R}$.

Another useful concept is that of the *state-action value function* $Q^\pi : S \times A \mapsto \mathbb{R}$ of a policy π. Such a function provides a convenient way to embed, in a single framework, the dynamics of the environment and the value function V^π. For each state $s \in S$ and each action $a \in A$, $Q^\pi(s, a)$ is the expected discounted return obtained by starting from state s, taking action a, and thereafter following π:

$$Q^\pi(s, a) = \mathrm{E}^\pi \left\{ \sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s, a_0 = a \right\}, \text{ for each } s \in S \text{ and } a \in A. \tag{2}$$

The (unique) *optimal state-action value function* Q^* is defined as the state-action value function of an optimal policy π^*. Once Q^* is known, it is possible to extract the optimal value function V^*, as well as an optimal policy π^* by choosing for each $s \in S$: $V^*(s) = \sup_{a \in A} Q^*(s, a)$ and $\pi^*(s) = \operatorname{argsup}_{a \in A} Q^*(s, a)$.

[1] The formalism is different from that originally used to describe RLVC [6]. This allows us to unify RLVC and RLJC within a single theoretical framework.

[2] More explicitly, we assume that the perceptual space is fully observable.

2.2 Incremental Discretization of the Perceptual Space

Because standard RL algorithms rely on a tabular representation of the value functions, they quickly become impractical as the number of possible percepts increases. This is evidently a problem in visual tasks. In RLVC, we have proposed to constrain the allowed structure of the state-action value functions $Q(s, a)$ by resorting to a *percept classifier* $\mathcal{C}$ that discretizes the perceptual space S into a finite set of *perceptual classes* $\{c_1, \dots, c_k\}$ by testing the presence of features in the percepts. RLVC assumes the finiteness of the action space: $A = \{a_1, \dots, a_m\}$.

Formally, let F_S be a (possibly infinite) set of *perceptual features* that can be defined on the perceptual space. Perceptual features suitable for visual tasks are discussed in Section 2.5. These features are required to be binary: Given a percept and a perceptual feature, the feature is either present in the percept or not. Therefore, the existence of a *perceptual feature detector* is assumed, which is a Boolean function $\mathcal{D}_S : S \times F_S \mapsto \mathcal{B}$ testing whether a given percept exhibits a given perceptual feature. Furthermore, we assume the presence of a *perceptual feature generator* $\mathcal{G}_S$ that, given a percept, computes the set of all the perceptual features that are present in this percept:

$$\mathcal{G}_S : S \mapsto \mathcal{P}(F_S) : s \mapsto \{f \in F_S \mid \mathcal{D}_S(s, f)\}, \tag{3}$$

where $\mathcal{P}$ denotes the power set.

Now, the percept classifier $\mathcal{C}$ is a binary decision tree. Each of its internal nodes is labeled by the perceptual feature, the presence of which is to be tested in that node. The n leaves of the tree define the set of perceptual classes $\{c_1, \dots, c_n\}$. To classify a percept, the system starts at the root node, then progresses down the tree according to the result of the perceptual feature detector $\mathcal{D}_S$ for each perceptual feature found during the descent, until it reaches a leaf.

Once a percept classifier $\mathcal{C}$ is fixed, all the percepts $s, s' \in c_i$ that lie in the same perceptual class c_i are required to share the same value for any state-action value function: $Q(s, a_j) = Q(s', a_j)$, for any action $a_j \in A$. Therefore, for a percept classifier $\mathcal{C}$ that induces n perceptual classes, any state-action value function $Q(s, a)$ is approximated as a function

$$\widetilde{Q}_{\mathcal{C}}(s, a, \boldsymbol{r}) = \boldsymbol{r}[i, j], \text{ if } \mathcal{C}(s) = c_i \text{ and } a = a_j, \tag{4}$$

where $\boldsymbol{r} \in \mathbb{R}^{n \times m}$ is a matrix of free parameters whose dimension is equal to the number of perceptual classes in $\mathcal{C}$ times the number of possible actions.

RLVC starts with a binary decision tree $\mathcal{C}_0$ that consists of a single leaf. Such a percept classifier maps all the percepts to the same perceptual class. Then, RLVC computes a matrix of parameters $\boldsymbol{r}_0^*$ that defines the optimal state-action value function $Q_0^*(s, a) = \widetilde{Q}_{\mathcal{C}_0}(s, a, \boldsymbol{r}_0^*)$ that is induced by $\mathcal{C}_0$. As the perceptual space is discretized, this can be done using standard RL algorithms [6]. Of course, the optimal decisions cannot always be made using Q_0^*, as percepts requiring different reactions are associated with the same class: $\mathcal{C}_0$ introduces *perceptual aliasing* [9], and the agent must refine the aliased class. So, the agent dynamically selects a new *distinctive* perceptual feature, i.e. one that best disambiguates the aliased percepts with respect to Q_0^*. This selection process is described in Sections 2.3 and 2.4. Then, the selected perceptual feature is used to refine the percept classifier $\mathcal{C}_0$, leading to a new classifier $\mathcal{C}_1$, and the process iterates.

To summarize, RLVC builds a sequence $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \ldots$ of growing decision trees, in a sequence of attempts to remove perceptual aliasing. An optimal state-action value function Q_k^* is computed for each percept classifier $\mathcal{C}_k$ in the sequence. At each step k, some leaves are replaced by tests on highly informative features, and the number of perceptual classes in the classifier grows.

2.3 Detecting Perceptual Aliasing

RLVC uses Bellman residuals to detect the perceptual classes that are aliased in a percept classifier $\mathcal{C}_k$. Bellman's optimality equation states that, if Q^* is the optimal state-action value function of the controlled system, then:

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma \cdot \sum_{s' \in S} \mathcal{T}(s,a,s') \sup_{a' \in A} Q^*(s',a'), \tag{5}$$

for all $s \in S$ and $a \in A$. If A is finite, if the transition relation $\mathcal{T}$ is assumed deterministic, and if one interaction $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$ is given, we deduce that:

$$Q^*(s_t, a_t) = r_{t+1} + \gamma \cdot \max_{a' \in A} Q^*(s_{t+1}, a'). \tag{6}$$

Let now consider Q_k^*, the optimal state-action value function induced by $\mathcal{C}_k$. As a consequence of Equations 4 and 6, for any time stamp t in the database of interactions, if a_t corresponds to the jth action in the finite set A, the scalar

$$\begin{aligned} \varDelta_t &= Q_k^*(s_t, a_t) - r_{t+1} - \gamma \cdot \max_{a' \in A} Q_k^*(s_{t+1}, a') & (7) \\ &= \boldsymbol{r}_k^* \left[\mathcal{C}_k(s_t), j\right] - r_{t+1} - \gamma \cdot \max_{a' \in \{1,\ldots,m\}} \boldsymbol{r}_k^* \left[\mathcal{C}_k(s_{t+1}), a'\right] & (8) \end{aligned}$$

is called the *Bellman residual* at time t, and is a measure of the perceptual aliasing occurring in the perceptual class $\mathcal{C}_k(s_t)$. If the environment is deterministic and if the percept classifier $\mathcal{C}_k$ is free of aliasing, then $\varDelta_t$ should always be zero.

Let c_i be a perceptual class that belongs to the percept classifier $\mathcal{C}_k$, and let $a \in A$ be an action. The set $T_k(c_i, a)$ is defined as the time stamps of all the interactions that are simultaneously related to the class c_i and to the action a:

$$T_k(c_i, a) = \{t \mid \mathcal{C}_k(s_t) = c_i \text{ and } a_t = a\}. \tag{9}$$

Following the reasoning above, the perceptual class c_i is considered aliased with respect to an action $a \in A$ if the set of Bellman residuals $\{\varDelta_t \mid t \in T_k(c_i, a)\}$ has a variance that exceeds a given threshold $\tau \in \mathbb{R}_0^+$.

2.4 Selecting Distinctive Perceptual Features

We now turn to the problem of selecting a distinctive feature that best disambiguates an aliased perceptual class c_i in the percept classifier $\mathcal{C}_k$ with respect to an action $a \in A$. To this end, we extract all the perceptual features that can be generated from the interactions that are simultaneously related to c_i and a:

$$F_k(c_i, a) = \{f \mid (\exists t \in T_k(c_i, a))\ f \in \mathcal{G}_S(s_t)\}. \qquad (10)$$

Among this set of candidate perceptual features, we select the feature that best explains the variations in the set of Bellman residuals. This is a regression problem, for which we apply the popular splitting rule that is used in the CART algorithm for building regression trees [10].

Each feature $f \in F_k(c_i, a)$ splits the Bellman residuals into two parts: $\{\Delta_t \mid t \in T_k(c_i, a) \wedge \mathcal{D}_S(s_t, f)\}$ and $\{\Delta_t \mid t \in T_k(c_i, a) \wedge \neg\mathcal{D}_S(s_t, f)\}$. We select the feature $f \in F(c_i, a)$ leading to the greatest reduction in the variance of these two sub-distributions. For each candidate feature, a Student's t-test decides whether the two sub-distributions of Bellman residuals are significantly different. This is important, as the transition relation $\mathcal{T}$ is in general non-deterministic, which generates variations in Bellman residuals that are not a consequence of aliasing.

2.5 Application to Visual Tasks

We now introduce perceptual features that enable RLVC to solve visual tasks [6]. Evidently, the high dimensionality and the noise of images cause problems in many fields of Computer Vision. For this purpose, the popular, highly successful *local-appearance methods* have been introduced. They postulate that, to take the right decision in a visual problem, it is often sufficient to focus one's attention only on a few interesting patterns occurring in the images.

They introduce a *visual feature transform* $\mathcal{F} : S \mapsto \mathcal{P}(\mathbb{R}^v)$, where S is the set of images, which summarizes an image as a set of *visual features* that are vectors of reals. For an image $s \in S$, $\mathcal{F}(s)$ typically contains between 10 and 1000 visual features. Most visual feature transforms have in common that: (1) they identify *interest points* in the images through specialized algorithms (Harris, Harris-affine,...) [11]; and (2) they compute a *local description* (local jets, SIFT,...) of the neighborhood of these interest points [12].

RLVC uses local descriptors as perceptual features. The set F_S of perceptual features corresponds to $\mathbb{R}^v$, and the perceptual feature detector $\mathcal{D}_S$ tests whether an image $s \in S$ exhibits some local descriptor at one of its interest points:

$$\mathcal{D}_S(s, \boldsymbol{f}) = \textbf{true} \text{ if and only if } (\exists \boldsymbol{f}' \in \mathcal{F}(s))\ ||\boldsymbol{f} - \boldsymbol{f}'|| < \varepsilon, \qquad (11)$$

where $\boldsymbol{f} \in F_S$ is a visual feature, and $\varepsilon \in \mathbb{R}_0^+$ is a fixed threshold. Any suitable metric $||\cdot||$ can be used to test the similarity of two visual features, e.g. the Mahalanobis distance. The corresponding perceptual feature generator $\mathcal{G}_S$ returns the local description of all the interest points in the input image: $\mathcal{G}_S(s) = \mathcal{F}(s)$.

3 Reinforcement Learning of Joint Classes

The aliasing criterion defined in Section 2.3 cannot be used anymore when the action space is continuous, for at least two reasons: The Bellman residuals (as defined by Equation 7) are unavailable, as the sup operator cannot be replaced by a max; and the set of time stamps of Equation 9 is useless, because the action space can only be sparsely sampled, so that any $T(c_i, a)$ essentially collapses to a set containing at most one element. A natural idea is therefore to also discretize the action space. RLJC follows this principle, and discretizes the *joint space* $S \times A$ instead of simply S. Whereas RLVC learns a sequence of *perceptual classifiers* $\mathcal{C}_k$ that discretize the percept space by testing perceptual features, RLJC learns a sequence of *joint classifiers* $\mathcal{J}_k$ that discretize the joint state-action space by testing features on the perceptual *and* on the action space.

Formally, a (possibly infinite) set F_A of *action features* is introduced in addition to the set F_S of perceptual features. Just as perceptual features, the action features are required to be binary, and the presence of an *action feature detector* $\mathcal{D}_A : A \times F_A \mapsto \mathcal{B}$ that tests the presence of an action feature in an action is assumed, as well as the presence of an *action feature generator* $\mathcal{G}_A : A \mapsto \mathcal{P}(F_A)$ that computes the action features that a given action exhibits.

3.1 Features for Continuous Action Spaces

We are interested in closed-loop learning of mappings from images to continuous actions. Thus, $A = \mathbb{R}^a$ for some positive number a. We now introduce action features that are suitable for such a continuous space. They simply consist in testing a threshold on a particular component of the action space. Precisely, the set of action features is defined as $F_A = \mathbb{R} \times \{1, \ldots, a\}$. The corresponding action feature detector $\mathcal{D}_A$ checks whether the considered component is below the threshold or not: $\mathcal{D}_A(\boldsymbol{a}, (t, i))$ is **true** if and only if $a_i < t$. On the other hand, the action feature generator $\mathcal{G}_A$ converts an action to a action features, one for each component of the input action $\boldsymbol{a} \in \mathbb{R}^a$: $\mathcal{G}_A(\boldsymbol{a}) = \{(a_i, i) \mid i \in \{1, \ldots, a\}\}$.

3.2 Joint Features on the Percept-Action Space

Let us call $F = F_S \cup F_A$ the set of *features*, that is the union of the perceptual and of the action features. Given a state-action pair and a feature, either the feature is present in the pair or not. As a consequence, the perceptual feature detector $\mathcal{D}_S$ along with the action feature detector $\mathcal{D}_A$ trivially induces a *joint feature detector* $\mathcal{D}$ that works on the joint space, and that is defined as:

$$\mathcal{D} : (S \times A) \times (F_S \cup F_A) \mapsto \mathcal{B} : ((s, a), f) \mapsto \begin{cases} \mathcal{D}_S(s, f) & \text{if } f \in F_S, \\ \mathcal{D}_A(a, f) & \text{otherwise.} \end{cases} \tag{12}$$

Similarly, $\mathcal{G}_S$ and $\mathcal{G}_A$ can be extended to a *joint feature generator* $\mathcal{G} : (S \times A) \mapsto \mathcal{P}(F_S \cup F_A)$, by choosing $\mathcal{G}(s, a) = \mathcal{G}_S(s) \cup \mathcal{G}_A(a)$ for each $s \in S$ and $a \in A$.

In terms of this notation, a *joint classifier* $\mathcal{J}$ is a binary decision tree whose internal nodes are labeled by a feature. A joint classifier maps the (possibly infinite) joint space $S \times A$ to a finite number of *joint classes* $\{c_1, \ldots, c_n\}$ using the joint feature detector $\mathcal{D}$. Identically to the case of RLVC, such joint classifiers are thereafter used to constrain the allowed structure of the state-action value functions $Q(s, a)$. For a joint classifier $\mathcal{J}$ that induces n perceptual classes, any state-action value function $Q(s, a)$ is now approximated as:

$$\widetilde{Q}_{\mathcal{J}}(s, a, \boldsymbol{r}) = \boldsymbol{r}[\mathcal{J}(s, a)], \tag{13}$$

where $\boldsymbol{r} \in \mathbb{R}^n$ is a vector of free parameters. Note that this relation treats percepts and actions symmetrically, contrarily to Equation 4.

Very importantly, since the state-action value function $\widetilde{Q}_{\mathcal{J}}(s, a, \boldsymbol{r})$ is constrained by a joint classifier $\mathcal{J}$, the maximization step that is required by the RL algorithms is now feasible. To compute $\sup_{a' \in A} \widetilde{Q}_{\mathcal{J}}(s, a', \boldsymbol{r})$ for a percept $s \in S$, we first evaluate the set of joint classes that are *compatible* with this percept:

$$C_{\mathcal{J}}(s) = \{c_i \mid (\exists a \in A)\ \mathcal{J}(s, a) = c_i\}. \tag{14}$$

This set $C_{\mathcal{J}}(s)$ can easily be computed by a depth-first search in the binary decision tree $\mathcal{J}$: For each path from the root node to a leaf, the corresponding leaf is added if and only if the percept s violates none of the tests on perceptual features that label this path. Finally, as $C_{\mathcal{J}}(s)$ is obviously finite, we obtain:

$$\sup_{a' \in A} \widetilde{Q}_{\mathcal{J}}(s, a', \boldsymbol{r}) = \sup_{a' \in A} \boldsymbol{r}[\mathcal{J}(s, a')] = \max_{c_i \in C_{\mathcal{J}}(s)} \boldsymbol{r}[i]. \tag{15}$$

A similar reasoning allows the derivation of a policy from a function $\widetilde{Q}_{\mathcal{J}}(s, a, \boldsymbol{r})$.

The general scheme of RLJC is then identical to that of RLVC. A sequence of joint classifiers $\mathcal{J}_0, \mathcal{J}_1, \mathcal{J}_2, \ldots$ is generated, starting with a joint classifier $\mathcal{J}_0$ that contains one single leaf. For each $\mathcal{J}_k$ in the sequence, the optimal state-action value function Q_k^* constrained by $\mathcal{J}_k$ is computed, thanks to an algorithm that is described in Section 3.3. Then, some informative features are selected by relying on an analysis of the Bellman residuals that are induced by Q_k^*. The corresponding process is described in Section 3.4. The selected features are used to refine $\mathcal{J}_k$, leading to the joint classifier $\mathcal{J}_{k+1}$. New joint classifiers are generated until perceptual aliasing vanishes.

3.3 Optimal State-Action Value Functions in the Joint Space

At each step k, the function Q_k^* is to be computed given the database of interactions $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$ that is the input of RLJC. As the structure of Q_k^* is constrained by Equation 13, this amounts to computing a vector $\boldsymbol{r}_k^* \in \mathbb{R}^{n_k}$, where n_k is the number of joint classes in $\mathcal{J}_k$ [1].

For this purpose, we use the *Fitted Q Iteration* algorithm [13], that generalizes the *Value Iteration* algorithm [2]. It uses an arbitrary family of nonparametric function approximators. Therefore, the existence of an oracle called `learn` is assumed. Given a database of samples $\langle s_t, a_t, v_t \rangle$, where s_t is a state, a_t is an action and v_t is a real number, `learn` builds a function approximator that represents a state-action value function $Q : S \times A \mapsto \mathbb{R}$ that is the closest possible to the given sample distribution. The algorithm computes a sequence $Q_0, Q_1, \ldots, Q_i$ of state-action value functions, starting with $Q_0 = \texttt{learn}(\{\langle s_t,\ a_t,\ r_{t+1} \rangle\})$. Equation 6 is then turned into an update rule that makes calls to the oracle:

$$Q_{i+1} = \texttt{learn}\left(\left\{\langle s_t,\ a_t,\ r_{t+1} + \gamma \cdot \sup_{a' \in A} Q_i(s_{t+1}, a')\rangle\right\}\right). \tag{16}$$

The algorithm stops when $Q_i \approx Q_{i+1}$. By virtue of Bellman's equations, it is possible to show that $Q_i \approx Q^*$ after convergence [13].

In our framework, RLJC directly uses the nonparametric function approximators $\widetilde{Q}_{J_k}(s, a, \boldsymbol{r})$ that are defined by Equation 13. Given a set of samples $\langle s_t, a_t, v_t\rangle$, the corresponding `learn` oracle computes a vector $\boldsymbol{r}$ that simply averages the values of the samples over the joint classes that are defined by $\mathcal{J}_k$: $\boldsymbol{r}[j] = \mu(\{v_t \mid \mathcal{J}_k(s_t, a_t) = c_j\})$, for each $j \in \{1, \ldots, n_k\}$, where $\mu(\cdot)$ denotes the mean of a set of reals. The maximization over the action space that is present in the update rule is achieved through Equation 15. When Fitted Q Iteration has completed the generation of the sequence $\boldsymbol{r}^{(0)}, \boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(i)}$, the parameter $\boldsymbol{r}_k^*$ is set to $\boldsymbol{r}^{(i)}$, which defines the optimal state-action value function Q_k^*.

3.4 Detecting and Removing Aliasing in the Joint Space

The algorithms that were presented for selecting new features are now adapted to continuous action spaces (cf. Sections 2.3 and 2.4). Thanks to Equation 15, the definition of Bellman residuals of Equation 7 can be further expanded:

$$\Delta_t = \boldsymbol{r}[\mathcal{J}(s_t, a_t)] - r_{t+1} - \gamma \cdot \max_{c_i \in C_{\mathcal{J}}(s_{t+1})} \boldsymbol{r}[i]. \tag{17}$$

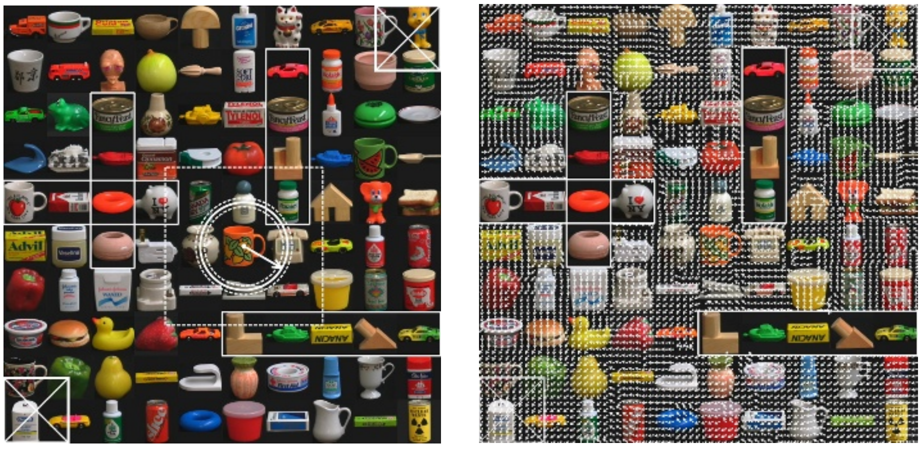
Once again, if the environment is deterministic and if the percept classifier $\mathcal{J}_k$ is free of aliasing, these residuals should be zero. Let c_i be a joint class of $\mathcal{J}_k$. Just as in RLVC, we define the set $T_k(c_i)$ of time stamps of interactions that are related to the class c_i, and the set $F_k(c_i)$ of candidate features for this class:

$$T_k(c_i) = \{t \mid \mathcal{J}_k(s_t, a_t) = c_i\}, \tag{18}$$
$$F_k(c_i) = \{f \mid (\exists t \in T_k(c_i))\ f \in \mathcal{G}(s_t, a_t)\}. \tag{19}$$

In terms of these definitions, the aliasing criterion and the feature selection process can be adapted to the joint space. A joint class c_i of the joint classifier $\mathcal{J}_k$ is considered aliased if the set of residuals $\{\Delta_t \mid t \in T_k(c_i)\}$ has a variance that exceeds a threshold $\tau \in \mathbb{R}_0^+$. If c_i is considered aliased, RLJC then selects the candidate feature inside $F_k(c_i)$ that most reduces the variance in the two subdistributions of Bellman residuals that are induced by the feature. A Student's t-test is also applied, to make RLJC robust to non-deterministic environments.

4 Experimental Results

RLJC has been evaluated on an abstract task that parallels a real-world scenario while avoiding any unnecessary complexity. This task is depicted in Figure 2 (a). An agent moves inside a maze in which walls are present. The agent is reduced to a single point, so it is always free to move between any two walls. Its goal is



(a) (b)

Fig. 2. (a) A continuous, noisy navigation task. The exits of the maze are indicated by boxes with a cross. Walls of glass are identified by solid lines. The agent is depicted at the center of the figure. The continuum of possible actions is represented by a solid circle. The two dashed circles indicate the standard deviation due to the noise. The sensors return a picture that corresponds to the dashed rectangular portion of the image. (b) The resulting image-to-action mapping $\pi^* = \text{argsup}_{a \in A} Q_k^*(s, a)$, sampled at regularly-spaced points. RLJC manages to choose the correct action at most locations[4].

to reach as fast as possible one of the two exits of the maze. At each location, the agent can make one step forward in any direction: The set A of actions is the continuous interval $[0°, 360°[$. Every move is altered by a Gaussian noise, the standard deviation of which is 1% the size of the maze. Whenever a move would take the agent into a wall or outside the maze, its location is not changed.

The agent earns a reward of 100 when an exit is reached. Any other move, including the forbidden ones, generates zero reinforcement. In this task, γ was set to 0.9. When the agent succeeds at escaping the maze, it reaches a terminal state. Note that the agent is faced with the delayed-reward problem, and that it must take the distance to the two exits into consideration for choosing the most attractive one. The maze has a ground carpeted with a color image of 1280×1280 pixels, that is a montage of pictures from the COIL-100 database[5]. The agent does not have direct access to its (x, y) position in the maze. Rather, its sensors take a picture of a surrounding portion of the ground. This portion is larger than the blank areas, which makes the input space fully observable, as long as too small displacements are not considered. Importantly, the walls are transparent, so that the sensors also return the portions of the tapestry that are behind them. Therefore, the agent cannot directly locate the walls.

[4] A full-sized version of this image is available for download at: `http://www.montefiore.ulg.ac.be/~jodogne/papers/rljc-policy.pdf`

[5] `http://www.cs.columbia.edu/CAVE/coil-100.html`

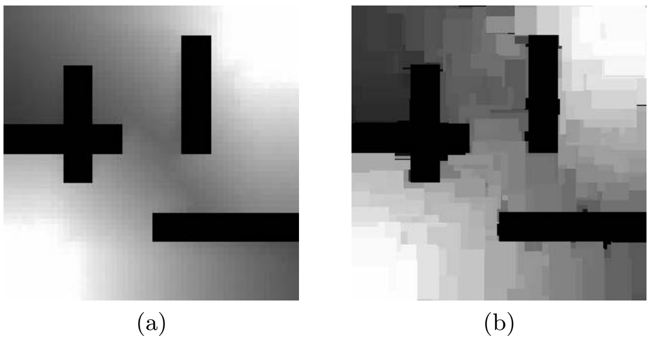


Fig. 3. (a) The optimal value function, if the agent has direct access to its (x, y) position, if the set of possible locations is discretized into a 50×50 grid, and if the set of actions is discrete and contains 4 actions (go up, down, left or right). The brighter the location, the greater its value. (b) The final value function obtained by RLJC.

In this experiment, SIFT visual features were used [14]. The entire tapestry includes 5520 interest points, leading to a subset of 2467 distinct visual features. The computation stopped when k reached 183, which took about 6 hours on a 3.0GHz Pentium IV using a database of 10,000 interactions that were collected by a fully randomized exploration policy. The final joint classifier $\mathcal{J}_k$ induces 896 joint classes, and tests the presence of 586 visual features and 309 action features. The optimal policy that results from this classifier is shown in Figure 2 (b). Figure 3 compares the optimal value function of a discretized version of the problem with the one obtained through RLJC. The similarity between the two pictures indicates the soundness of our approach.

Interestingly enough, when applied to a similar task with only four *discrete* actions, RLVC generates a perceptual classifier $\mathcal{C}_k$ that contains 205 perceptual classes [6]. In that case, $\mathcal{C}_k$ induces an optimal state-action value function that is characterized by a vector $\boldsymbol{r}_k^*$ of dimension $205 \times 4 = 820$ (cf. Equation 4). This latter number is very close to the number of joint classes that is produced by RLJC (i.e. 896). Therefore, discretizing the joint space produces a number of joint classes that corresponds to the underlying physical structure of the task.

5 Conclusions

This paper introduces *Reinforcement Learning of Joint Classes* (RLJC). RLJC is designed for closed-loop learning of mappings that directly connect visual stimuli to continuous actions that are optimal for the surrounding environment. RLJC adaptively discretizes the joint space of states and actions into a finite set of joint classes, by testing the presence of highly distinctive features. The homogeneous treatment of states and actions is at the same time elegant and powerful, and is conceptually similar to that of JoSTLe [7]. However, RLJC is more general, in the sense that it can be applied to any perceptual space and to any action space upon which it is possible to define binary features. This notably includes visual input spaces, and continuous input/output spaces. Therefore, RLJC could learn mappings from *continuous* perceptual spaces to continuous action spaces as well.

Future research includes the demonstration of the applicability of our algorithms in a reactive robotic application, such as grasping objects by combining visual and haptic feedback [15]. Our current work considers tasks with complete perception and stationary environments. Of course, applying our algorithms directly on a real-world environment would raise practical problems, including partial observability, which would require the combination of our techniques with POMDP-based approaches. Another interesting open question is to test how well RLJC scales with respect to the dimensionality of the output action vector.

References

1. Bertsekas, D., Tsitsiklis, J.: Neuro-Dynamic Programming. Athena Scient. (1996)
2. Sutton, R., Barto, A.: Reinforcement Learning, an Introduction. MIT Press (1998)
3. Gross, H.M., Stephan, V., Krabbes, M.: A neural field approach to topological reinforcement learning in continuous action spaces. In: Proc. of the IEEE World Congress on Computational Intelligence. Volume 3. (1998) 1992–1997
4. Santamaria, J., Sutton, R., Ram, A.: Experiments with reinforcement learning in problems with continuous state and action spaces. Adaptive Behavior **6**(2) (1998) 163–218
5. Gaskett, C., Wettergreen, D., Zelinsky, A.: Q-learning in continuous state and action spaces. In: Australian Joint Conf. on Artificial Intelligence. (1999) 417–428
6. Jodogne, S., Piater, J.: Interactive learning of mappings from visual percepts to actions. In De Raedt, L., Wrobel, S., eds.: Proc. of the 22nd Intern. Conf. on Machine Learning (ICML), Bonn (Germany), ACM (2005) 393–400
7. Monson, C., Wingate, D., Seppi, K., Peterson, T.: Variable resolution discretization in the joint space. In: Intern. Conf. on Machine Learning and Applications. (2004)
8. Munos, R., Moore, A.: Variable resolution discretization in optimal control. Machine Learning **49** (2002) 291–323
9. Whitehead, S., Ballard, D.: Learning to perceive and act by trial and error. Machine Learning **7** (1991) 45–83
10. Breiman, L., Friedman, J., Stone, C.: Classification and Regression Trees. Wadsworth Intern. Group (1984)
11. Schmid, C., Mohr, R., Bauckhage, C.: Evaluation of interest point detectors. Intern. Journal of Computer Vision **37**(2) (2000) 151–172
12. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. Volume 2., Madison (WI, USA) (2003) 257–263
13. Ernst, D., Geurts, P., Wehenkel, L.: Tree-based batch mode reinforcement learning. Journal of Machine Learning Research **6** (2005) 503–556
14. Lowe, D.: Distinctive image features from scale-invariant keypoints. Intern. Journal of Computer Vision **60**(2) (2004) 91–110
15. Coelho, J., Piater, J., Grupen, R.: Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. Robotics and Autonomous Systems, special issue on Humanoid Robots **37**(2–3) (2001) 195–218

EM Algorithm for Symmetric Causal Independence Models

Rasa Jurgelenaite and Tom Heskes

Institute for Computing and Information Sciences, Radboud University Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands
{rasa, tomh}@cs.ru.nl

Abstract. Causal independence modelling is a well-known method both for reducing the size of probability tables and for explaining the underlying mechanisms in Bayesian networks. In this paper, we present the EM algorithm to learn the parameters in causal independence models based on the symmetric Boolean function. The developed algorithm enables us to assess the practical usefulness of the symmetric causal independence models, which has not been done previously. We evaluate the classification performance of the symmetric causal independence models learned with the presented EM algorithm. The results show the competitive performance of these models in comparison to noisy OR and noisy AND models as well as other state-of-the-art classifiers.

1 Introduction

Bayesian networks [1] are well-established as a sound formalism for representing and reasoning with probabilistic knowledge. However, because the number of conditional probabilities for the node grows exponentially with the number of its parents, it is usually unreliable if not infeasible to specify the conditional probabilities for the node that has a large number number of parents. The task of assessing conditional probability distributions becomes even more complex if the model has to integrate expert knowledge. While learning algorithms can be forced to take into account an expert's view, for the best possible results the experts must be willing to reconsider their ideas in light of the model's 'discovered' structure. This requires a clear understanding of the model by the domain expert. *Causal independence models* [2], [3], [4] can both limit the number of conditional probabilities to be assessed and provide the ability for models to be understood by domain experts in the field. The main idea of causal independence models is that causes influence a given common effect through intermediate variables and interaction function.

Causal independence assumptions are often used in practical Bayesian network models [5], [6]. However, most researchers restrict themselves to using only the logical OR and logical AND operators to define the interaction among causes. The resulting probabilistic submodels are called *noisy OR* and *noisy AND*; their underlying assumption is that the presence of either at least one cause or all causes at the same time give rise to the effect. Several authors proposed to expand the space of interaction functions by other symmetric Boolean functions: the idea was already mentioned but not developed further in [7], analysis of the qualitative patterns was presented in [8], and assessment of conditional probabilities was studied in [9].

Even though for some real-world problems the intermediate variables are observable (see [10]), in many problems these variables are latent. Therefore, conditional probability distributions depend on unknown parameters which must be estimated from data, using *maximum likelihood* (ML) or *maximum a posteriori* (MAP). One of the most widespread techniques for finding ML or MAP estimates is the *expectation-maximization* (EM) algorithm. Meek and Heckerman [7] provided a general scheme how to use the EM algorithm to compute the maximum likelihood estimates of the parameters in causal independence models assumed that each local distribution function is collection of multinomial distributions. Vomlel [11] described the application of the EM algorithm to learn the parameters in the noisy OR model. However, the proposed schemes of the EM algorithm are not elaborated, specific to a given causal independence model, and hence not directly applicable to the general case of parameter learning in causal independence models.

Learning the parameters in causal independence models with a symmetric Boolean function as an interaction function (further referred to as the *symmetric causal independence models*) is the main topic of this paper. We develop an EM algorithm to learn the parameters in symmetric causal independence models. The presented algorithm enables us to assess the practical usefulness of this expanded class of causal independence models, which has not been done by other authors. The evaluation is done by using the symmetric causal independence models learned with the developed EM algorithm as classifiers. Experimental results show the competitive classification performance of these models in comparison with the noisy OR classifier as well as other widely-used classifiers.

The remainder of this paper is organised as follows. In the following section, we review Bayesian networks and discuss the semantics of symmetric causal independence models. In Section 3, we first describe the general scheme of the EM algorithm and then develop the EM algorithm for finding the parameters in symmetric causal independence models. Section 4 presents the experimental results, and conclusions are drawn in Section 5.

2 Symmetric Boolean Functions for Modelling Causal Independence

2.1 Bayesian Networks

A *Bayesian network* $\mathcal{B} = (G, \Pr)$ represents a factorised joint probability distribution on a set of random variables $\mathbf{V}$. It consists of two parts: (1) a qualitative part, represented as an acyclic directed graph (ADG) $G = (\mathbf{V}(G), \mathbf{A}(G))$, where there is a 1–1 correspondence between the vertices $\mathbf{V}(G)$ and the random variables in $\mathbf{V}$, and arcs $\mathbf{A}(G)$ represent the conditional (in)dependencies between the variables; (2) a quantitative part Pr consisting of local probability distributions $\Pr(V \mid \pi(V))$, for each variable $V \in \mathbf{V}$ given the parents $\pi(V)$ of the corresponding vertex (interpreted as variables). The joint probability distribution Pr is factorised according to the structure of the graph, as follows:

$$\Pr(\mathbf{V}) = \prod_{V \in \mathbf{V}} \Pr(V \mid \pi(V)) .$$

Each variable $V \in \mathbf{V}$ has a finite set of mutually exclusive states. In this paper, we assume all variables to be binary; as an abbreviation, we will often use v^+ to denote $V = \top$ (true) and v^- to denote $V = \bot$ (false). We interpret $\top$ as 1 and $\bot$ as 0 in an arithmetic context. An expression such as

$$\sum_{\psi(H_1,\ldots,H_n)=\top} g(H_1, \ldots, H_n)$$

stands for summing $g(H_1, \ldots, H_n)$ over all possible values of the variables H_k for which the constraint $\psi(H_1, \ldots, H_n) = \top$ holds.

2.2 Semantics of Symmetric Causal Independence Models

Causal independence (also known as independence of causal influence) is a popular way to specify interactions among cause variables. The global structure of a causal independence model is shown in Figure 1; it expresses the idea that causes $C_1, \ldots, C_n$ influence a given common effect E through hidden variables $H_1, \ldots, H_n$ and a deterministic function f, called the *interaction function.* The impact of each cause C_i on the common effect E is independent of each other cause $C_j, j \neq i$. The hidden variable H_i is considered to be a contribution of the cause variable C_i to the common effect E. The function f represents in which way the hidden effects H_i, and indirectly also the causes C_i, interact to yield the final effect E. Hence, the function f is defined in such a way that when a relationship, as modelled by the function f, between $H_i, i = 1, \ldots, n$, and $E = \top$ is satisfied, then it holds that $f(H_1, \ldots, H_n) = \top$. It is assumed that $\Pr(e^+ \mid H_1, \ldots, H_n) = 1$ if $f(H_1, \ldots, H_n) = \top$, and $\Pr(e^+ \mid H_1, \ldots, H_n) = 0$ if $f(H_1, \ldots, H_n) = \bot$.

A causal independence model is defined in terms of the causal parameters $\Pr(H_i \mid C_i)$, for $i = 1, \ldots, n$ and the function $f(H_1, \ldots, H_n)$. Most papers on

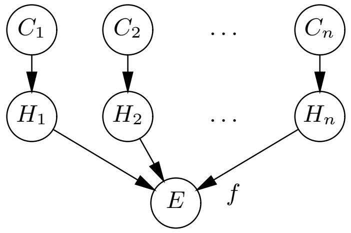


Fig. 1. Causal independence model

causal independence models assume that absent causes do not contribute to the effect [1]. In terms of probability theory this implies that it holds that $\Pr(h_i^+ \mid c_i^-) = 0$; as a consequence, it holds that $\Pr(h_i^- \mid c_i^-) = 1$. In this paper we make the same assumption.

In situations in which the model does not capture all possible causes, it is useful to introduce a *leaky cause* which summarizes the unidentified causes contributing to the effect and is assumed to be always present [12]. We model this leak term by adding an additional input $C_{n+1} = 1$ to the data; in an arithmetic context the leaky cause is treated in the same way as identified causes.

The conditional probability of the occurrence of the effect E given the causes $C_1, \dots, C_n$, i.e., $\Pr(e^+ \mid C_1, \dots, C_n)$, can be obtained from the causal parameters $\Pr(H_l \mid C_l)$ as follows [4]:

$$\Pr(e^+ \mid C_1, \dots, C_n) = \sum_{f(H_1,\dots,H_n)=\top} \prod_{i=1}^{n} \Pr(H_i \mid C_i) . \tag{1}$$

In this paper we assume that the function f in Equation (1) is a Boolean function. However, there are 2^{2^n} different n-ary Boolean functions [13], [14]; thus, the potential number of causal interaction models is huge. However, if we assume that the order of the cause variables does not matter, the Boolean functions become *symmetric* [14] and the number reduces to 2^{n+1}.

An important symmetric Boolean function is the *exact* Boolean function ϵ_l, which has function value true, i.e. $\epsilon_l(H_1, \dots, H_n) = \top$, if $\sum_{i=1}^n \nu(H_i) = l$ with $\nu(H_i)$ equal to 1, if H_i is equal to true and 0 otherwise. A symmetric Boolean function can be decomposed in terms of the exact functions ϵ_l as [14]:

$$f(H_1, \dots, H_n) = \bigvee_{i=0}^{n} \epsilon_i(H_1, \dots, H_n) \wedge \gamma_i \tag{2}$$

where γ_i are Boolean constants depending only on the function f. For example, for the Boolean function defined in terms of the OR operator we have $\gamma_0 = \bot$ and $\gamma_1 = \dots = \gamma_n = \top$.

Another useful symmetric Boolean function is the *threshold* function τ_k, which simply checks whether there are at least k trues among the arguments, i.e. $\tau_k(I_1, \dots, I_n) = \top$, if $\sum_{j=1}^n \nu(I_j) \geq k$ with $\nu(I_j)$ equal to 1, if I_j is equal to true and 0 otherwise. To express it in the Boolean constants we have: $\gamma_0 = \cdots = \gamma_{k-1} = \bot$ and $\gamma_k = \cdots = \gamma_n = \top$. Causal independence model based on the Boolean threshold function further will be referred to as the *noisy threshold models.*

2.3 The Poisson Binomial Distribution

Using the property of Equation (2) of the symmetric Boolean functions, the conditional probability of the occurrence of the effect E given the causes $C_1, \dots, C_n$ can be decomposed in terms of probabilities that exactly l hidden variables $H_1, \dots, H_n$ are true, as follows:

$$\Pr(e^+ \mid C_1, \ldots, C_n) = \sum_{\substack{0 \le l \le n \\ \gamma_l}} \sum_{\epsilon_l(H_1,\ldots,H_n)} \prod_{i=1}^{n} \Pr(H_i \mid C_i) \ . \tag{3}$$

Let l denote the number of successes in n independent trials, where p_i is a probability of success in the ith trial, $i = 1, \ldots, n$; let $\mathbf{p} = (p_1, \ldots, p_n)$, then $\mathrm{B}(l; \mathbf{p})$ denotes the *Poisson binomial distribution* [15]:

$$\mathrm{B}(l; \mathbf{p}) = \left\{ \prod_{i=1}^{n} (1 - p_i) \right\} \sum_{1 \le j_1 < \ldots < j_l \le n} \prod_{z=1}^{l} \frac{p_{j_z}}{1 - p_{j_z}} \ . \tag{4}$$

Let us define a vector of probabilistic parameters $\mathbf{p}(C_1, \ldots, C_n) = (p_1, \ldots, p_n)$ with $p_i = \Pr(h_i^+ \mid C_i)$. Then the connection between the Poisson binomial distribution and the class of symmetric causal independence models is as follows.

Proposition 1. *It holds that:*

$$\Pr(e^+ \mid C_1, \ldots, C_n) = \sum_{i=0}^{n} \mathrm{B}(i; \mathbf{p}(C_1, \ldots, C_n)) \gamma_i \ .$$

3 EM Algorithm

In this section, we first describe the general scheme of the EM algorithm. Then we develop the EM algorithm that finds the unknown parameters $\theta = (\theta_1, \ldots, \theta_n)$ of a symmetric causal independence model where $\theta_i = \Pr(h_i^+ \mid c_i^+)$.

3.1 Basic EM

Let $\mathbf{D} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ be a data set of independent and identically distributed settings of the observed variables in a symmetric causal independence model, where

$$\mathbf{x}^j = (\mathbf{c}^j, e^j) = (c_1^j, \ldots, c_n^j, e^j) \ .$$

We assume that no additional information about the model is available. Therefore, to learn the parameters of the model we maximize the conditional log-likelihood

$$CLL(\theta) = \ln(CL(\theta)) = \sum_{j=1}^{N} \ln \Pr(e^j \mid \mathbf{c}^j, \theta) \ .$$

where $\theta = (\theta_1, \ldots, \theta_n)$ are unknown parameters of the model.

The expectation-maximization (EM) algorithm [16] is a general method to find the maximum likelihood estimate of the parameters in probabilistic models, where the data is incomplete or the model has hidden variables.

We start from the following simple identity:

$$\ln \Pr(e^j \mid \mathbf{c}^j, \theta) = \ln \Pr(\mathbf{H}, e^j \mid \mathbf{c}^j, \theta) - \ln \Pr(\mathbf{H} \mid e^j, \mathbf{c}^j, \theta) \tag{5}$$

and take expectations of both sides, treating $\mathbf{H}$ as a random variable with the distribution $\Pr(\mathbf{H} \mid e^j, \mathbf{c}^j, \theta^{(old)})$, where $\theta^{(old)}$ is the current (old) guess. The left hand side of Equation (5) does not depend on $\mathbf{H}$, so averaging over $\mathbf{H}$ yields

$$\begin{aligned}\ln \Pr(e^j \mid \mathbf{c}^j, \theta) = &\sum_{\mathbf{H}} \Pr(\mathbf{H} \mid e^j, \mathbf{c}^j, \theta^{(old)}) \ln \Pr(\mathbf{H}, e^j \mid \mathbf{c}^j, \theta) \\ &- \sum_{\mathbf{H}} \Pr(\mathbf{H} \mid e^j, \mathbf{c}^j, \theta^{(old)}) \ln \Pr(\mathbf{H} \mid e^j, \mathbf{c}^j, \theta) \; . \end{aligned} \tag{6}$$

The key result for the EM algorithm is that the last term in the above equation is maximized at $\theta = \theta^{(old)}$, thus any increase of the first term on the right side of Equation (6) is guaranteed to increase the expected complete (conditional) log-likelihood.

Let us denote

$$Q(\theta; \theta^{(z)}) = \sum_{j=1}^{N} \sum_{\mathbf{H}} \Pr(\mathbf{H} \mid e^j, \mathbf{c}^j, \theta^{(z)}) \ln \Pr(\mathbf{H}, e^j \mid \mathbf{c}^j, \theta) \; . \tag{7}$$

The EM algorithm at each iteration maximizes this functional:

$$\theta^{(z+1)} = \operatorname*{argmax}_{\boldsymbol{\theta}} Q(\theta; \theta^{(z)}) \; .$$

In the next subsection, we find the values of the parameters $\theta = (\theta_1, \ldots, \theta_n)$ that maximize the function $Q(\theta; \theta^{(z)})$ for the symmetric causal independence model.

3.2 Maximization Step

We start by transforming $\ln \Pr(\mathbf{H}, e^j \mid \mathbf{c}^j, \theta)$ so that it becomes a sum of logarithms:

$$\ln \Pr(\mathbf{H}, e^j \mid \mathbf{c}^j, \theta) = \ln \Pr(e^j \mid \mathbf{H}) + \sum_{i=1}^{n} \ln \Pr(H_i \mid c_i^j, \theta_i) \; . \tag{8}$$

The conditional probability $\Pr(H_i \mid c_i^j, \theta_i)$ can be written in the form

$$\Pr(H_i \mid c_i^j, \theta_i) = c_i^j H_i \theta_i + c_i^j (1 - H_i)(1 - \theta_i) + (1 - c_i^j)(1 - H_i) \; . \tag{9}$$

Combining (7), (8) and (9), we obtain

$$\begin{aligned} Q(\theta; \theta^{(z)}) = &\sum_{j=1}^{N} \sum_{\mathbf{H}} \Pr(\mathbf{H} \mid e^j, \mathbf{c}^j, \theta^{(z)}) \cdot \\ &\left(\ln \Pr(e^j \mid \mathbf{H}) + \sum_{i=1}^{n} \ln \left(\theta_i c_i^j (2H_i - 1) + 1 - H_i \right) \right) \; . \end{aligned}$$

We can maximize this result by computing the partial derivatives of $Q(\theta; \theta^{(z)})$ with respect to $\theta_k : k = 1, \ldots, n$ and setting them to zero:

$$\frac{\partial Q(\theta; \theta^{(z)})}{\partial \theta_k} = \sum_{j=1}^{N} \sum_{\mathbf{H}} \Pr(\mathbf{H} \mid e^j, \mathbf{c}^j, \theta^{(z)}) \frac{c_k^j (2H_k - 1)}{\theta_k c_k^j (2H_k - 1) + 1 - H_k} = 0 \,. \quad (10)$$

Now let us define $\mathbf{H}_{\setminus k} = \{H_1, \ldots, H_{k-1}, H_{k+1}, \ldots, H_n\}$. Then Equation (10) can be simplified writing it as a sum over the states of the hidden variable H_k:

$$\sum_{1 \leq j \leq N} c_k^j \sum_{\mathbf{H}_{\setminus k}} \left(\frac{\Pr(\mathbf{H}_{\setminus k}, h_k^+ \mid e^j, \mathbf{c}^j, \theta^{(z)})}{\theta_k} - \frac{\Pr(\mathbf{H}_{\setminus k}, h_k^- \mid e^j, \mathbf{c}^j, \theta^{(z)})}{1 - \theta_k} \right) = 0 \,.$$

It can be shown that Equation (10) is solved by

$$\theta_k = \frac{\sum_{1 \leq j \leq N} c_k^j \Pr(h_k^+ \mid e^j, \mathbf{c}^j, \theta^{(z)})}{\sum_{1 \leq j \leq N} c_k^j} \,. \quad (11)$$

It is easy to check whether this extremum is a maximum by computing the second partial derivatives of $Q(\theta; \theta^{(z)})$ with respect to $\theta_k, k = 1, \ldots, n$. The matrix formed from these second partial derivatives is negative semidefinite, and hence this stationary point is indeed always a maximum of the function $Q(\theta; \theta^{(z)})$.

In the next subsection, we derive the expectation step which corresponds to computing the conditional probabilities $\Pr(h_k^+ \mid e^j, \mathbf{c}^j, \theta^{(z)})$ for all $k = 1, \ldots, n$, $j = 1, \ldots, N$ where $c_k^j = 1$.

3.3 Expectation Step

Using Bayes rule, we can write the probability of $\mathbf{H}$ given a data sample $\mathbf{x}^j$ and the parameters $\theta^{(z)}$ as follows:

$$\Pr(\mathbf{H} \mid e^j, \mathbf{c}^j, \theta^{(z)}) = \frac{\Pr(e^j \mid \mathbf{H}) \prod_{i=1}^{n} \Pr(H_i \mid c_i^j, \theta^{(z)})}{\Pr(e^j \mid \mathbf{c}^j, \theta^{(z)})} \,.$$

By marginalizing $\mathbf{H}_{\setminus k}$ out we obtain the conditional probability of the hidden variable H_k being true:

$$\Pr(h_k^+ \mid e^j, \mathbf{c}^j, \theta^{(z)}) = \frac{\Pr(h_k^+ \mid c_k^j, \theta_k^{(z)})}{\Pr(e^j \mid \mathbf{c}^j, \theta^{(z)})} \cdot \sum_{\mathbf{H}_{\setminus k}} \Pr(e^j \mid \mathbf{H}_{\setminus k}, h_k^+) \prod_{\substack{1 \leq i \leq n \\ i \neq k}} \Pr(H_i \mid c_i^j, \theta_i^{(z)}) \,. \quad (12)$$

Let us define $\hat{\theta}_{(k=1)}^{(z)} = (\hat{\theta}_1, \ldots, \hat{\theta}_n)$ where $\hat{\theta}_k^{(z)} = 1$ and $\hat{\theta}_i^{(z)} = \theta_i^{(z)}$, $\forall_{i \neq k}$. Using the defined vector $\hat{\theta}_{(k=1)}^{(z)}$ and $\Pr(h_k^+ \mid c_k^j, \theta_k^{(z)}) = c_k^j \theta_k^{(z)}$, Equation (12) takes the form

$$\Pr(h_k^+ \mid e^j, \mathbf{c}^j, \theta^{(z)}) = \frac{c_k^j \theta_k^{(z)} \Pr(e^j \mid \mathbf{c}^j, \hat{\theta}^{(z)}_{(k=1)})}{\Pr(e^j \mid \mathbf{c}^j, \theta^{(z)})} \, . \tag{13}$$

Now we can express the obtained result in terms of the Poisson binomial probabilities. First, let us define

$$\begin{aligned} \mathbf{p}^{(z,j)} &= (p_1^{(z,j)}, \ldots, p_n^{(z,j)}) \quad \text{where} \quad p_i^{(z,j)} = \theta_i^{(z)} c_i^j \, , \\ \hat{\mathbf{p}}^{(z,j)}_{(k=1)} &= (\hat{p}_1^{(z,j)}, \ldots, \hat{p}_n^{(z,j)}) \quad \text{where} \quad \hat{p}_k = 1 \text{ and } \hat{p}_i^{(z,j)} = \theta_i^{(z)} c_i^j, \forall_{i \neq k} \, . \end{aligned}$$

From the following property of the Poisson binomial distribution [17]:

$$\mathrm{B}(i; \mathbf{p}) = \mathrm{B}(i; \mathbf{p}_{\setminus k})(1 - p_k) + \mathrm{B}(i-1; \mathbf{p}_{\setminus k}) p_k \tag{14}$$

it follows that

$$\mathrm{B}(i; \hat{\mathbf{p}}^{(z,j)}_{(k=1)}) = \mathrm{B}(i-1; \mathbf{p}^{(z,j)}_{\setminus k}) \, .$$

Using the last identity and Proposition 1 the left hand side of (13) can be expressed in terms of the Poisson binomial probabilities as follows:

$$\Pr(h_k^+ \mid e^j, \mathbf{c}^j, \theta^{(z)}) = \begin{cases} \dfrac{p_k^{(z,j)} \sum_{i=0}^{n-1} \mathrm{B}\left(i; \mathbf{p}^{(z,j)}_{\setminus k}\right) \gamma_{i+1}}{\sum_{i=0}^{n} \mathrm{B}\left(i; \mathbf{p}^{(z,j)}\right) \gamma_i} & \text{if } e^j = 1 \, , \\[2ex] \dfrac{p_k^{(z,j)} \left(1 - \sum_{i=0}^{n-1} \mathrm{B}\left(i; \mathbf{p}^{(z,j)}_{\setminus k}\right) \gamma_{i+1}\right)}{1 - \sum_{i=0}^{n} \mathrm{B}\left(i; \mathbf{p}^{(z,j)}\right) \gamma_i} & \text{if } e^j = 0 \, . \end{cases} \tag{15}$$

Summarizing, the EM algorithm for symmetric causal independence models is given by:

Expectation step: For every instance $\mathbf{x}^j = (\mathbf{c}^j, e^j)$ with $j = 1, \ldots, N$, we form

$$\mathbf{p}^{(z,j)} = (p_1^{(z,j)}, \ldots, p_n^{(z,j)}) \quad \text{where} \quad p_i^{(z,j)} = \theta_i^{(z)} c_i^j \, .$$

Subsequently, the probability $\mathrm{P}(h_k^+ \mid \mathbf{c}^j, e^j, \theta^{(z)})$ is computed from (15) for all hidden variables H_k with $k = 1, \ldots, n$.

Maximization step: Update the parameter estimates for all $k = 1, \ldots, n$ using Equation (11).

4 Experimental Results

The introduced EM algorithm enables us to evaluate the practical significance of the symmetric causal independence models. As it is difficult to provide an interpretation of the learned parameters, we evaluate the learned symmetric causal independence models based on their classification performance.

4.1 Evaluation Scheme

Since we do not have an efficient algorithm to perform a search in the space of symmetric Boolean functions, we chose to model the interaction among cause and effect variables by means of Boolean threshold functions, which seem to be the most probable interaction functions for the given domains.

Given the model parameters θ, the training data $\mathbf{D}_{train}$ and the classification threshold $\frac{1}{2}$, the classifications and misclassifications for both classes are computed. Let *tp (true positives)* stand for the number of data samples $(\mathbf{c}^j, e^{j+}) \in \mathbf{D}_{train}$ for which $\Pr(e^+ \mid \mathbf{c}^j, \theta) \geq \frac{1}{2}$ and *fp (false positives)* stand for the number of data samples $(\mathbf{c}^j, e^{j+}) \in \mathbf{D}_{train}$ for which $\Pr(e^+ \mid \mathbf{c}^j, \theta) < \frac{1}{2}$. Likewise, *tn (true negatives)* is the number of data samples $(\mathbf{c}^j, e^{j-}) \in \mathbf{D}_{train}$ for which $\Pr(e^+ \mid \mathbf{c}^j, \theta) < \frac{1}{2}$ and *fp (false positives)* is the number of data samples $(\mathbf{c}^j, e^{j-}) \in \mathbf{D}_{train}$ for which $\Pr(e^+ \mid \mathbf{c}^j, \theta) \geq \frac{1}{2}$. To evaluate the classification performance we use *accuracy*, which is a measure of correctly classified cases,

$$\eta = \frac{tp + tn}{tp + tn + fn + fp} \ ,$$

and *F-measure*, which combines *precision* $\pi = \frac{tp}{tp+fp}$ and *recall* $\rho = \frac{tp}{tp+fn}$,

$$F = \frac{2\pi\rho}{\pi + \rho} \ .$$

4.2 Non-Hodgkin Lymphoma Data Set

For our experiments we use a database with data from the patients with gastric non-Hodgkin lymphoma (NHL) collected by the clinical experts from the Netherlands Cancer Institute (NKI). The data set consists of the factors that influence the result of treatment, and hence the learned models can be argued to follow the causal interpretation. We will cover only the basic facts; a thorough description of the disease and collected data can be found in [18].

Gastric non-Hodgkin lymphoma is a type of cancer of the lymphatic system, the disease-fighting network spread throughout the body, which originates in the stomach. Response to treatment is one of the most important prognostic indicators of a long-term disease-free survival, particularly in patients with aggressive NHL [19]. We learn a causal independence model that models the interaction between the early outcome of the treatment and the pretreatment prognostic factors. The early outcome of the treatment, i.e. the effect in the model, stands for endoscopically verified result of the treatment, six to eight weeks after treatment with complete remission defining a situation in which all clinical signs of disease disappear with the treatment. The following pretreatment information, i.e. the causes in the model, is available: (1) age; (2) general health status; (3) bulky disease; (4) histological classification; (5) stage of the cancer; (6) clinical signs (hemmorhage, perforation, obstruction) due to the disease.

Based on the medical literature we converted the data to binary form and chose the state of every variable that corresponds to the presence of the cause/effect. The resulting model is shown in Figure 2 where the name of the variable indicates

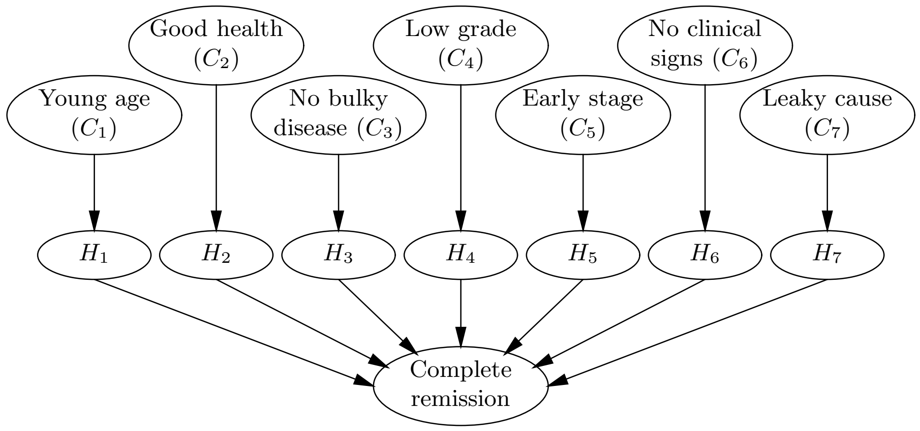


Fig. 2. Causal independence model modelling complete remission following treatment of non-Hodgkin lymphoma. The variable 'Young age' represents a patient younger than 60 years, the variable 'Early stage' stands for the first clinical stage of NHL, and the variable 'No clinical signs' represents a patient who has no hemorrhage, no perforation and no obstruction.

its positive state. To learn the parameters of the model we used 125 patient cases with no missing data. 95 of the patients had complete remission six to eight weeks after the treatment and for the other 30 patients the disease did not disappear. As the data set is small, a leave-one-out cross-validation scheme was employed both to evaluate the performance of the model and to avoid data overfitting. Classification performance measures for symmetric causal independence models with the interaction function $\tau_k,\ k = 1, \ldots, 7$ are listed in Table 1. The results show that the interaction between the pretreatment variables and the outcome of the treatment is best modelled by the interaction function τ_2. Note that noisy threshold model with the threshold $k = 2$ outperforms the noisy OR model, while the noisy AND model is a poor choice to model the given problem.

In order to see how well the causal independence models classify compared with other classification algorithms, we evaluated the classification performance

Table 1. Classification performance measures for noisy threshold models with the threshold $k = 1, \ldots, 7$ for Non-Hodgkin Lymphoma data set

Causal independence model	Accuracy (%)	F-measure
noisy OR	75.2	0.854
noisy threshold $k = 2$	83.2	0.896
noisy threshold $k = 3$	82.4	0.891
noisy threshold $k = 4$	78.4	0.857
noisy threshold $k = 5$	71.2	0.795
noisy threshold $k = 6$	56.8	0.625
noisy AND	36.8	0.288

Table 2. Classification performance measures for different classifiers for Non-Hodgkin Lymphoma data set. Weka's default parameter settings were used.

Classifier	Accuracy (%)	F-measure
noisy threshold $k = 2$	83.2	0.896
naive Bayes	84.0	0.899
logistic regression	82.4	0.885
multilayer perceptron	82.4	0.885
decision tree (C4.5)	73.6	0.832
support vector machine	77.6	0.861

of a few widely-used classifiers on NHL data set. The experiments were performed using the Weka system [20]. The results reported in Table 2 show that noisy threshold model provides very similar results to those of naive Bayes, logistic regression and multilayer perceptron and outperforms decision tree and support vector machine classifiers.

5 Discussion

In this paper, we developed the EM algorithm to learn the parameters in symmetric causal independence models and studied its computational complexity and convergence. The presented algorithm enabled us to evaluate the utility of symmetric causal independence models. The reported experimental results indicate that it is unnecessary to restrict causal independence models to only two interaction functions, logical OR and logical AND. Additionally, competitive performance of symmetric causal independence models present them as a potentially useful additional tool to the set of classifiers.

The current study has only examined the problem of learning conditional probabilities of hidden variables. The problem of learning an optimal interaction function has not been addressed. Efficient search in symmetric Boolean function space is a possible direction for future research.

Acknowledgments. This research, carried out in the TimeBayes project, was supported by the Netherlands Organization for Scientific Research (NWO) under project number FN4556. The authors are grateful to Henk Boot and Babs Taal for the provided non-Hodgkin's lymphoma data. We would also like to thank Jiří Vomlel for sharing his code and insights.

References

1. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kauffman Publishers (1988)
2. Díez, F.J.: Parameter Adjustment in Bayes Networks. The generalized noisy OR-gate. In: Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence (1993) 99–105

3. Heckerman D., Breese, J.S.: A New Look at Causal Independence. In: Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (1994) 286–292
4. Zhang, N.L., Poole, D.: Exploiting Causal Independence in Bayesian Networks Inference. Journal of Artificial Intelligence Research, Vol. 5 (1996) 301–328
5. Kappen, H.J., Neijt, J.P.: Promedas, a Probabilistic Decision Support System for Medical Diagnosis. Technical report, SNN - UMCU (2002)
6. Shwe, M.A., Middleton, B., Heckerman, D.E., Henrion, M., Horvitz, E.J., Lehmann, H.P., Cooper, G.F.: Probabilistic Diagnosis using a Reformulation of the INTERNIST-1/QMR Knowledge Base, I – The Probabilistic Model and Inference Algorithms. Methods of Information in Medicine, Vol. 30 (1991) 241–255
7. Meek, C., Heckerman, D.: Structure and Parameter Learning for Causal Independence and Causal Interaction Models. In: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (1997) 366–375
8. Lucas, P.J.F.: Bayesian Network Modelling Through Qualitative Patterns. Artificial Intelligence, Vol. 163 (2005) 233–263
9. Jurgelenaite, R., Lucas, P.J.F., Heskes, T.: Noisy Threshold Functions for Modelling Causal Independence in Bayesian Networks. Technical report ICIS–R06014, Radboud University Nijmegen (2006)
10. Visscher, S., Lucas, P.J.F., Bonten, M., Schurink, K.: Improving the Therapeutic Performance of a Medical Bayesian Network using Noisy Threshold Models. In: Proceedings of ISBMDA 2005, the 6th International Symposium on Biological and Medical Data Analysis (2005) 161–172
11. Vomlel, J.: Noisy-or Classifier. International Journal of Intelligent Systems, Vol. 21 (2006) 381–398
12. Henrion, M.: Some Practical Issues in Constructing Belief Networks. Uncertainty in Artificial Intelligence, Vol. 3 (1989) 161–173
13. Enderton, H.B.: A Mathematical Introduction to Logic. Academic Press, San Diego (1972)
14. Wegener, I.: The Complexity of Boolean Functions. John Wiley & Sons, New York (1987)
15. Le Cam, L.: An Approximation Theorem for the Poisson Binomial Distribution. Pacific Journal of Mathematics, Vol. 10 (1960) 1181–1197
16. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society, Vol. 39 (1977) 1–38
17. Darroch, J.: On the Distribution of the Number of Successes in Independent Trials. The Annals of Mathematical Statistics, Vol. 35 (1964) 1317–1321
18. Lucas, P.J.F., Boot, H., Taal, B.: Computer-based Decision Support in Management of Primary Gastric non-Hodgkin Lymphoma. Methods of Information in Medicine, Vol. 37 (1998) 206–219
19. Bast, R.C., Kufe, D.W., Pollock, R.E., Weichselbaum, R.R., Holland, J.F., Frei, E.: Cancer Medicine - 5 Review. B C Decker Inc., Ontario (2000)
20. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco (2005)

Deconvolutive Clustering of Markov States

Ata Kabán and Xin Wang

School of Computer Science, The University of Birmingham,
Birmingham, B15 2TT, UK
{axk, xcw}@cs.bham.ac.uk

Abstract. In this paper we formulate the problem of grouping the states of a discrete Markov chain of arbitrary order simultaneously with deconvolving its transition probabilities. As the name indicates, this problem is related to deconvolutive blind signal separation. However, whilst the latter has been studied in the context of continuous signal processing, e.g. as a model of a real-room mixing of sound signals, our technique tries to model computer-mediated group-discussion participation from a discrete event-log sequence. In this context, convolution occurs due to various time-delay factors, such as the network transmission bandwidth or simply the typing speed of the participants. We derive a computationally efficient maximum likelihood estimation algorithm associated with our model, which exploits the sparsity of state transitions and scales linearly with the number of observed higher order transition patterns. Results obtained on a full day worth dynamic real-world Internet Relay Chat participation sequence demonstrate the advantages of our approach over state grouping alone, both in terms of penalised data likelihood and cluster clarity. Other potential applications of our model, viewed as a novel compact approximation of large Markov chains, are also discussed.

1 Introduction

The classical scenario that illustrates the problem of deconvolutive separation of mixed signals (deconvolutive blind source separation, DBSS) is as follows. Consider K simultaneous speakers (sound signals) in a real room and M microphone sensors. Each sensor receives a different convolutive mixture of the source signals. The mixing is convolutive due to the echo in a real room setting, i.e. several delayed and attenuated versions of the signals arrive at the sensor. The task is then to recover the individual speakers from the recorded mixture signals.

Due to its practical relevance in a number of problems, such as interference and echo cancellation, a large body of research on solving DBSS problems has been devised over the last decade. These include algebraic methods [12], predictability-based heuristics [18], information-theoretic methods [3] probabilistic and Bayesian methods [2] and various combinations thereof. A common technique is to work in the frequency domain, where the convolution becomes multiplication.

Our problem is conceptually related to DBSS, however it differs in a number of respects, which makes existing models and methods not directly applicable:

(1) The observed signal is a symbolic sequence rather than a continuous real-valued stream. (2) The 'original source signals' that we assume to exist and try to recover are the cluster-membership probabilities of the symbols that make up the observed sequence. (3) There is a single observation channel. (4) The mixing proportions change over time. Further specificities and model assumptions will be discussed in more detail later.

An illustrative real-world example application, which is also in close analogy with the discussed classical DBSS problem setting is the modelling of a computer-mediated discussion session such as the Internet Relay Chat (IRC), as a dynamic social network. The observed signal in an IRC channel consists of typed contributions and the session is automatically logged in a sequential manner, as a single symbolic sequence. We make abstraction of the textual content here, instead, we focus on the participation sequence. Thus we have a symbolic sequence of user ID-s and what we are looking for is their cluster memberships, in order to identify user communities in this dynamic social network.

In this context, convolution occurs due to various time-delay factors, such as the network transmission bandwidth or simply the number and typing speed of the participants. In other words, we posit that it would be unrealistic to assume that consecutiveness of the contributions, as logged, is an accurate indicator of user interaction. Instead, our convolutive model allows for contributions situated at earlier time lags in the logged sequence to represent user links with a certain probability. The task is then to recover the most likely allocation of symbols to clusters, the mixing (interleaving) parameters and the convolution filter, i.e. the probability with which earlier time lags contribute towards the cluster structure.

There are many studies of community identification in static settings and with given network connections. [10,13]. However, dynamic analyses have been less researched and represent increasing interest recently [6,15]. To our best of knowledge, our deconvolutive clustering approach is new. However the method that we present is not restricted to this application but more generally applicable, in principle, to approximating large Markovian chains.

2 The Model

Let $X = \{x_1, x_2, \cdots, x_N\}$ denote a symbolic sequence with each symbol $x_n \in \{1, 2, \cdots, T\}$ in a T-symbol state space. We assume that X is a stationary L-th order Markov chain, i.e. $P(x_n|x_{n-1}, \cdots, x_1) = P(x_n|x_{n-1}, \cdots, x_{n-L})$. We then define a compact approximation to the full L-th order transition probability at any discrete time n as the following convolutive mixture.

$$P(x_n|x_{n-1}, \cdots, x_{n-L}) = \sum_{l=1}^{L} e_l \sum_{k=1}^{K} s_{x_n,k} a_{k,x_{n-l}} \tag{1}$$

The parameters of this model will be sought in the form of probabilities, as follows. The quantities of primary interest are sought as cluster membership probabilities $a_{k,s} \equiv P(k|s)$ so for all symbols s, we impose $\sum_k a_{k,s} = 1, \forall s$.

Likewise, the convolution filters $e_l \equiv P(l)$ will signify the probability with which each time lag, up to a memory depth of L, contributes towards the state clustering. Further, in order to obtain the required transition probability from this decomposition, it is convenient to also seek for the remaining mixing parameter in a probabilistic form, i.e. $s_{x_n,k} \equiv P(x_n|k)$.

2.1 Interpretations

Since all parameters of our model (1) are probabilities, we can interpret k as the outcome of a latent class variable $\Xi \in \{1, 2, ..., k, ..., K\}$. Likewise, we can also interpret l as the outcome of another discrete latent variable $\Lambda \in \{1, 2, ..., l, ..., L\}$ – the time delay variable. Then the complete data likelihood can be written as the following.

$$\mathcal{L}^C = \prod_n \prod_l \prod_k P(l)^{\delta(\Lambda_n = l)} \left\{ P(x_n|k) P(k|x_{n-l}) \right\}^{\delta(\Xi_n = k)} \tag{2}$$

where δ is the Kronecker delta function. So at each discrete event we have a latent time delay and a latent class. The observed L-th order transition probability is assumed by this model to be generated by instantiating these latent variables.

The expectation of the log of (2) is the following.

$$E[\log \mathcal{L}^C] = \sum_{n,l,k} P(\Xi = k, \Lambda = l | x_n, x_{n-1}, ..., x_{n-L}) \log \left\{ P(l) P(x_n|k) P(k|x_{n-l}) \right\}$$

This may then be maximised employing the Expectation Maximisation [8] methodology in order to estimate the model parameters. Doing so is straightforward, however tedious for this model, since the posterior probabilities of the discrete latent variables would need to be computed and stored for every possible L-gram. Therefore for deriving an efficient estimation algorithm, the form (1) will be manipulated so that the posterior probabilities need not be computed explicitly during the parameter estimation. The latent discrete variable view is however useful for the interpretation of the model and the posteriors may be computed for inference purposes after the parameter estimation is complete.

2.2 Relation to Existing Discrete Sequence Models

State aggregation in first order Markov chains through a discrete 'bottleneck' latent variable has been studied before and it is known as the aggregate Markov model [17]. This was previously used e.g. in language modelling [17,11] and bibliometric analysis [7]. The aggregate Markov model seeks for the first order transition probabilities in the following form.

$$P(x_n|x_{n-1}) = \sum_{k=1}^{K} P(x_n|k) P(k|x_{n-1}) \tag{3}$$

However, extensions to higher order models have not been considered previously.

A model that could be related to the problem of deconvolving transition probabilities, in the manner we employ it, was studied in statistics and it is known as the mixed transition model [4,1,14]. This has the following form:

$$P(x_n|x_{n-1},\cdots,x_{n-L}) = \sum_{l=1}^{L} P(l)P(x_n|x_{n-l}) \tag{4}$$

A slightly different version, having separate transition matrices for all lags appears as the mixed memory Markov model [16]. In the mentioned works the goal has been to approximate a higher order Markov chain by a model that uses fewer parameters. The goodness of approximation has been studied [14,4]; moreover, the steady-state distribution was shown to be given by the principal eigenvector of the stochastic matrix in (4), $P(s|s')$ [1] where s, s' are the symbols in the dictionary.

We can view our model (1) as a combination of an aggregate Markov model and a mixed transition Markov model. Although our primary motivation in this paper has been to model the convolutive mixing process, owing to the results of [14,4,1], it also follows that our model (1) provides an approximate solution to clustering the states of a higher order Markov chain.

Finally, it should be pointed out that despite we make use of hidden variables (as discussed in Sec.2.1), our model is different from the hidden Markov model (HMM) [19]. This is evident from the fact that we are using an approximation of an L-th order Markov model [4] of the observed sequence, whereas the HMM makes no Markov assumption of any order on the observed sequence. Instead, it makes the (first order) Markov assumption on a hidden sequence. The difference has several implications. From the point of view of parameter interpretation, we can regard $P(l)$ as the integrated saliency of past information at progressive temporal lags. These provide explicit credit-assignments for each time lag up to the depth L of the Markov process, which may serve e.g. as a compact characterisation of the sequence[1] There is no such credit-assignment indicator in HMMs. Finally, from the computation complexity point of view, the estimation of a HMM requires time quadratic in the number of hidden states, whereas, as we will see in the next section, our method can be estimated in time that is linear with the number of non-zero observed higher-order transition patterns. In consequence we view our model as being complementary to HMM both technically and in terms of functionality.

2.3 Maximum Likelihood Estimation for Deconvolutive State Clustering

In this section we derive an efficient iterative estimation algorithm for the deconvolutive clustering of Markov states, based on maximum likelihood (ML). Simple

[1] Although outside the scope of this paper, we experimentally found that the distribution $P(l)$ obtained from sequences that represent computer-mediated discussions tends to have high entropy, while those that represent individual activity have low entropy.

manipulations of (1) yield the log likelihood of a sequence $X = \{x_1, \cdots, x_N\}$ under the model as follows:

$$\log P(X|\Theta) = \sum_{t_0,t_1,\cdots,t_L=1}^{T} N_{t_0,t_1,\cdots,t_L} \log \sum_{l=1}^{L} e_l \sum_{k=1}^{K} a_{k,t_l} s_{t_0,k} \tag{5}$$

where $(t_0, t_1, \cdots, t_l, \cdots, t_L)$ is used to denote an $(L+1)$-gram, such that $(x_n = t_0, x_{n-1} = t_1, \cdots, x_{n-L} = t_L)$, for some n, where $t_0, t_1, \cdots, t_l, \cdots, t_L$ are symbols $\in \{1, 2, \cdots, T\}$ in a dictionary of size T. Further, $N_{t_0,t_1,\cdots,t_L}$ denotes the frequency of the $(L+1)$-gram specified in the index, i.e. the frequency of the subsequence $t_L \rightarrow t_{L-1} \rightarrow \cdots \rightarrow t_0$. Finally, as earlier, $l \in \{1, \cdots, L\}$ stands for time-lags.

Maximising (5) with respect to all the parameters e_l, a_{k,t_l} and $s_{t_0,k}$ under the requirement of being probabilities, is carried out iteratively, by alternating optimisation. This yields the following multiplicative fixed point updates:

$$e_l^{i+1} \propto e_l^i \sum_{t_0,t_1,\cdots,t_L=1}^{T} \sum_{k=1}^{K} \frac{N_{t_0,t_1,\cdots,t_L} a_{k,t_l}^i s_{t_0,k}^i}{\sum_{l'} e_{l'}^i \sum_{k'} a_{k',t_{l'}}^i s_{t_0,k'}^i}$$

$$a_{k,t_l}^{i+1} \propto a_{k,t_l}^i \sum_{t_0,\cdots,t_{l-1}}^{T} \cdots \sum_{t_{l+1},\cdots,t_L=1}^{T} \frac{N_{t_0,t_1,\cdots,t_L} e_l^i s_{t_0,k}^i}{\sum_{l'} e_{l'}^i \sum_{k'} a_{k',t_{l'}}^i s_{t_0,k'}^i}$$

$$s_{t_0,k}^{i+1} \propto s_{t_0,k}^i \sum_{t_1,\cdots,t_L=1}^{T} \sum_{l=1}^{L} \frac{N_{t_0,t_1,\cdots,t_L} e_l^i a_{k,t_l}^i}{\sum_{l'} e_{l'}^i \sum_{k'} a_{k',t_{l'}}^i s_{t_0,k'}^i}$$

where $\propto$ stands for proportionality and i is the iteration index. The iterated application of the above updates is guaranteed to converge to a local optimum of the likelihood. This is because each of the above updates can also be derived as the consecutive application of a complete E-step and an M-step for one parameter set only, while keeping the remaining parameters fixed at their current values (cf. the interpretation discussed in Sec. 2.1.).

2.4 Scaling

The worst-case time complexity per iteration of our algorithm is $O(T^{L+2} \times L \times K)$. However, typically, real data tend to be sparse and this is what our algorithm exploits. Indeed, it can easily be seen from the updates, that whenever a count $N_{t_0,t_1,\cdots,t_L}$ is zero, the fraction is also zero and therefore needs not be evaluated. Let S denote the number of non-zero observed (L+1)-gram counts. Expressed in terms of S, our algorithm scales as $O(S \times L \times K)$. Typically $L \times K \ll S$, so the scaling is linear with the number of observed non-zero $(L+1)$-gram counts.

2.5 Posterior Computations

As already pointed out, computing the posterior probabilities of the discrete variables is not required for estimating the parameters. Instead, once the parameter estimation is complete, we may wish compute the discrete posteriors once, in order to make further inferences about the data. This is in contrast with existing solutions of related models such as the aggregate Markov [17,11] and the mixed transition Markov model [14,17], and could be applied to those as well.

Given the estimated parameter values, the joint posterior can be written as the following.

$$P(k,l|t_0,t_1,\cdots,t_L) = P(l|t_0,t_1,\cdots,t_L)P(k|l,t_0,t_1,\cdots,t_L) \tag{6}$$

Then, for each time-window of size (L+1), we obtain the posterior class $P(k|t_0,t_1,\cdots,t_L) = \sum_l P(k,l|t_0,t_1,\cdots,t_L)$ by marginalisation.

The two terms in (6) are computed using the model parameters and applying Bayes' theorem:

$$P(k|l,t_0,t_1,\cdots,t_L) \propto a_{k,t_l} s_{t_0,k} \tag{7}$$

$$P(l|t_0,t_1,\cdots,t_L) \propto N_{t_0,t_1,\cdots,t_L} e_l \sum_k a_{k,t_l} s_{t_0,k} \tag{8}$$

The latter, $P(l|t_0,t_1,\cdots,t_L)$, gives the posterior distribution of time delays in each (L+1)-window, and naturally, its maximum argument provides the most probable posterior delay in a particular window. So it can be used to reconstruct the a-posteriori transition graph, or, in other words, the *inferred* links between the states.

2.6 Clustering the States

In order to obtain cluster labels for the states (symbols), based on the model, we may proceed in several alternative ways.

One option is to use the *parameters* a_{k,t_l}, which, by the model design (cf. Sec.2), sum to one w.r.t. k, for each symbol t_l from the dictionary and are meant to be interpretable as cluster memberships. It is useful to observe that during the iterative maximum likelihood estimation algorithm, each step minimises a weighted sum of entropies of all parameters [21]. Therefore if there are clusters in the sense defined by the model, the entropy of $a_{.,t_l} \equiv P(.|t_l)$ may reach a reasonably low value at convergence, so that it is sensible indeed to interpret $\text{argmax}_k\ a_{k,t_l}$ as the cluster label of symbol t_l. This can then be used to group similar states together. This procedure is convenient, since the parameters $a_{.,t_l}$ are readily available and so only the inferred links need to be computed after completing the parameter estimation.

A second procedure for labelling the states falls out naturally from the posterior computations. This is to consider the *context-conditional* cluster-label $\underset{k}{\text{argmax}}\, P(k|t_0,t_1,\cdots,t_L)$ as the label of t_0 conditional on the previous L-gram.

Thirdly, we may obtain *unconditional* labels for the states by applying Bayes' rule and integrating over the contexts in the prior:

$$P(k|t_0) \propto P(k)P(t_0|k);\ \text{where}\ P(k) \propto \sum_{t_0,t_1,\cdots,t_L=1}^{T} N_{t_0,t_1,\cdots,t_L} P(k|t_0,t_1,\cdots,t_L)$$

and where $P(t_0|k) \equiv s_{t_0,k}$ is available.

3 Finding Communities in a Dynamical Social Network

Here we apply our deconvolutive state clustering method to the problem of community identification from dynamic social networks.

Social network analysis [10,13] is an interdisciplinary area dealing with the study of the structure of human relationships and associations which take the form of a network. Although most of these relations are dynamic in their nature, existing approaches to social network analysis, with very few exceptions [15], typically neglect the time component. Social networks are typically represented in the form of a graph with observable connections, and the analysis proceeds using various graph partitioning techniques.

However, we argue that the time component should be taken more seriously in many cases, and our proposed method provides a viable approach for doing so. In order to demonstrate this, we carry out (1) a data-driven evaluation of whether or not our proposed temporal deconvolution helps to better explain/model the dynamic social network, and (2) whether or not it helps obtaining clearer community structures.

The data that we use here is a sequence formed by real-world dynamic chat participation, collected from Internet Relay Chat (IRC) lines [5]. It consists of a stream of about one day worth discussions, totalling $N = 25,355$ contributions from $T = 844$ different chat participants. We study the symbolic sequence of user ID-s, making abstraction of the textual content of the contributions. The latter has been the focus of previous study [5], while here we are looking for communities (as state clusters) based on user interactions rather than based on topical similarities. The temporal connections in such data are quite sparse — the number of observed $(L+1)$-grams ($L \in \{1 \cdots 10\}$) is 14030, 23808, 24790, 24968, 25040, 25085, 25128, 25163, 25193, 25216 respectively, far less than 844^{L+1}. Owing to the ability of our algorithm to take advantage of data sparseness to reduce computations, as discussed in Section 2.4, running our algorithm on this data takes less than a minute on a standard computer.

3.1 Penalised Data Likelihood and Selecting the Best Model Order

For each combination of K and L, the models are trained 20 times to avoid local optima. The Akaike Information Criterion (AIC) is used to determine the optimal model order. Since we estimate our models by Maximum Likelihood, and our primary aim is data explanation, this criterion is appropriate [20]. The AIC has the simple form of a penalised data likelihood:

$$AIC = -2 \log P(X|\Theta) + 2P \tag{9}$$

where P is the number of free parameters in the model, which in our case is $(L-1)+(T-1)K+(K-1)T$. The optimal model order is then given by the L and K pair that maximises AIC.

Fig. 1 shows the AIC curves of our models. As we can see, all deconvolutive models (i.e. $L > 1$) achieve higher scores than the Aggregate Markov (i.e.

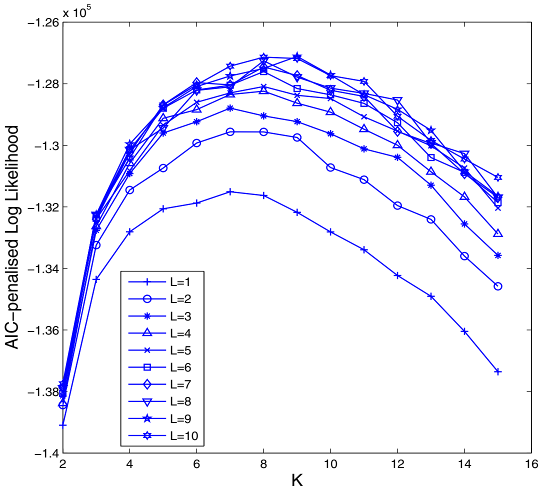


Fig. 1. The AIC curves (AIC values against number of clusters) for various models with differing order L. The model at $L = 1$ reduces to the aggregate Markov (AM) model. Clearly, all models with $L > 1$ outperform the AM and the best model order is found to be at $L = 9, K = 9$. Thus, the optimum number of clusters in this data is 9, with a memory depth of 9 past symbols.

L=1), which tries to group the states without a deconvolution model, based on consecutive (first-order) relationships. This demonstrates the convolutive models contribute to increasing the data likelihood under the model sufficiently to justify the extra parameters. In consequence, we have a data-driven evidence that our deconvolutive clustering model is more appropriate for modelling the dynamic social network analysed, in comparison to state aggregation alone.

3.2 Cluster Clarity

Next, with the optimal model order selected above ($L = 9, K = 9$), we investigate how does our deconvolutive model compare with state aggregation alone, in terms of the clarity of the clusters found. To this end, we make use of the posterior computations as described in Section 2.5. We take the maximum argument of the posterior time delay in each consecutive window, $l^* = \underset{l}{\text{argmax}}\ P(l|t_0, t_1, \cdots, t_l, \cdots, t_L)$, and use that to infer the de-convolved posterior transition matrix. Denoting this matrix by $\boldsymbol{M}$, each $M(i, j)$ will contain the frequency of symbol j having triggered symbol i, after some time delay. $\boldsymbol{M}$ is easily constructed by scanning the sequence and increasing $M(t_0, t_{l^*})$ by 1 for each symbol. Further, we use any of the state cluster labelling procedures, as described in Section 2.6, to reorder M by grouping the states that are assigned the same label.

Fig.2. shows the raw data in the form of a first-order adjacency graph, having the states in alphabetical order of the user IDs (upper left plot), the same graph with states reordered according to cluster labels obtained from the best Aggregate

Markov model (upper right plot), and the results of the optimal deconvolutive clustering model (i.e. L=9, K=9, cf. Fig.1.): The inferred posterior de-convolved adjacency matrix ($\boldsymbol{M}$), reordered using labels obtained from parameters of the model, i.e. $\underset{k}{\operatorname{argmax}}\, a_{k,t_l}$ is shown on the lower left plot and $\boldsymbol{M}$ reordered using the context-conditional labels, i.e. $\underset{k}{\operatorname{argmax}}\ P(k|t_0, t_1, \cdots, t_l, \cdots, t_L)$ is on the lower right plot. It is visually most apparent from these plots, that even though state aggregation alone does display some structure, a much clearer clustering has been obtained from our deconvolutive approach. Another observation is that the two lower plots are qualitatively similar. Using the marginal cluster labels, $\underset{k}{\operatorname{argmax}}\ P(k|t_0)$ (omitted) has been qualitatively similar as well.

In order to provide a quantitative comparison in terms of cluster clarity, we use classical numeric measures to express the goodness of clustering: the strength of intra-connectivity vs. inter-connectivity in the identified clusters of states. We define the intra-connectivity $\mathcal{C}_{ki}$ of a cluster k to be the density of links between

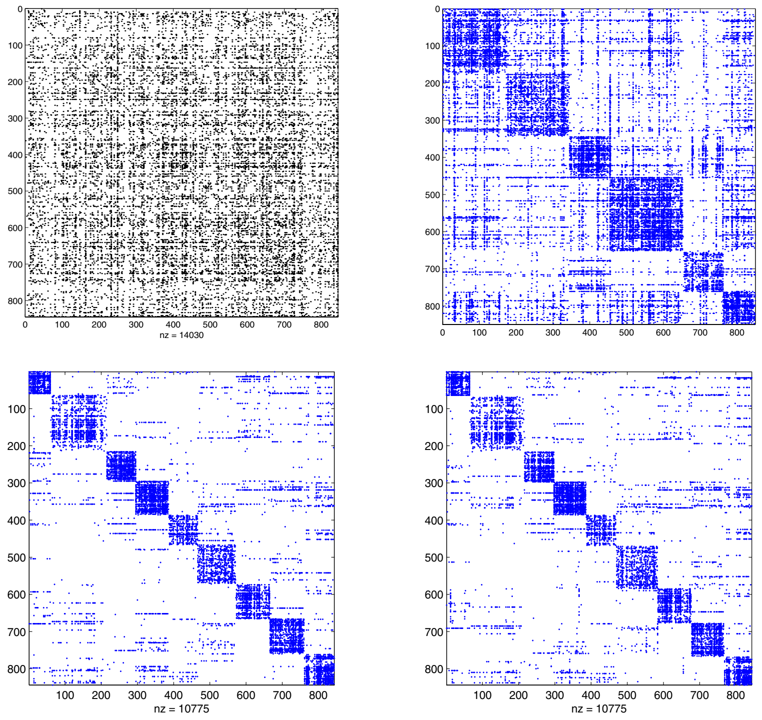


Fig. 2. The raw data shown as the first-order adjacency graph (upper left), the same graph with reordered states by using the best Aggregate Markov model (upper right), the inferred posterior de-convolved adjacency matrix $\boldsymbol{M}$, reordered using $\operatorname{argmax}_k a_{k,t_l}$ as obtained from the best (K=9, L=9) deconvolutive clustering model (lower left) and $\boldsymbol{M}$ reordered using the context-conditional labels $\operatorname{argmax}_k P(k|t_0, t_1, \cdots, t_l, \cdots, t_L)$. The cluster clarity is superior for the deconvolutive models.

Table 1. Quantitative comparison of cluster clarity. 'DC' = the optimal deconvolutive clustering model (L=9,K=9); AM: Aggregate Markov; HMM: Hidden Markov Model. $\mathcal{C}_i$: intra-connectivity, averaged over all clusters (higher is better); $\mathcal{C}_o$: intra-cluster connectivity (lower is better). The three variants of DC refer to the three different state labelling schemes (see text). All three labelling schemes of DC produce clearer clusters than AM, and even HMM, the overall winner being the labelling based on the class posteriors conditioned on the previous L-gram.

Model	$\mathcal{C}_i$	$\mathcal{C}_0$	$\mathcal{C}_i/\mathcal{C}_0$
DC-context	**0.3187**	**0.0068**	**47.054**
DC-marginal	0.3173	**0.0068**	46.6618
DC-parameter	0.3171	0.0073	43.677
AM	0.13	0.0165	7.8788
HMM(K=12)	0.2616	0.0257	10.1962

members of that cluster, while inter-connectivity $\mathcal{C}_{ko}$ will refer to the density of links between members of different clusters. More formally,

$$\mathcal{C}_{ki} = \frac{1}{|T_k| \times |T_k|} \sum_{i,j \in T_k} n_{ij}; \qquad \mathcal{C}_{ko} = \sum_{l=1, l \neq k}^{K} \frac{1}{|T_k| \times |T_l|} \sum_{i \in T_k} \sum_{j \in T_l} (n_{ij} + n_{ji})$$

where, T_k, T_l denotes the state space of cluster k and l; n_{ij} and n_{ji} are transition counts in a reordered transition matrix. Table 1 summarises the results comparatively, for the optimal deconvolutive clustering (DC) model (K=9, L=9), using the three labelling schemes (cf. Sec. 2.6.) and the optimal Aggregate Markov Model. We also included HMM (with 12 states, selected using AIC) in this comparison, for completeness. All three labelling schemes of DC produce clearer clusters than AM and HMM, the overall winner being the labelling based on the class posteriors conditioned on the previous L-gram.

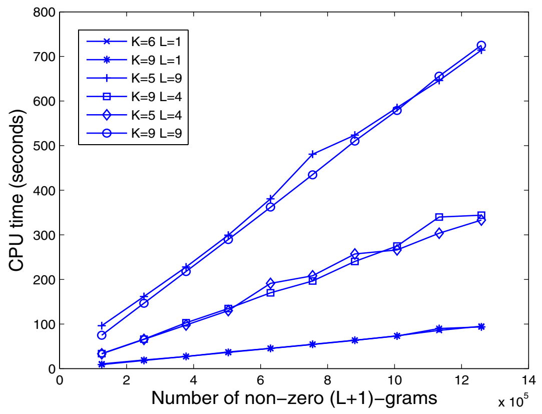


Fig. 3. CPU time versus the number of non-zero entries in the data, when the size of data is progressively increased

3.3 Scalability

Since the application demonstrated in the previous subsections is focused on a particular data set, now we assess the computation time requirements of our parameter estimation algorithm on data sets of increasing size. Figure 3 shows the required CPU time per iteration (on a 2GHz Intel processor), as a function of the number of nonzero entries. As expected, the scaling is indeed linear and this backs up the theoretical complexity given earlier in Section 2.4.

4 Conclusions

We formulated the problem of deconvolutive clustering and developed a probabilistic model for this problem. We discussed analogies with deconvolutive source separation – which is fairly well-studied in signal processing – as well as similarities and differences with existing discrete sequence models. We derived a computationally efficient maximum likelihood estimation algorithm associated with our model, which exploits the sparsity of state transitions and scales linearly with the number of observed higher order transition patterns. Results obtained on a real-world dynamic social network demonstrated the advantages our approach over state grouping alone, both in terms of penalised data likelihood and cluster clarity.

Since our model can be viewed as a mixed-transition generalisation of the aggregate Markov model, and due to earlier results in statistics regarding the mixed transition model with shared transition probabilities as an approximation of a full higher order Markov chain [14], it follows that our approach is interpretable as providing an approximate solution to the problem of partitioning the states of a higher order Markov chain. Due to this, our algorithm may have further applications that are worthy of investigation, for example, in computing the steady state distribution of very large Markov models, such as those of interest in performance engineering and distributed systems [9].

References

1. S.R Adke and S.R Deshmukh. Limit Distribution of a Higher Order Markov Chain. Journal of the Royal Statistical Society B, 1988, 105–108.
2. H Attias and C.E Schreiner. Blind Source Separation and Deconvolution: The Dynamic Component Analysis. Neural Computation, 10(6): 1373–1424, 1998.
3. A.J Bell and T Sejnowski. An information Maximisation Approach to Blind Separation and Blind Deconvolution. Neural Computation, 7:1129–1159, 1995.
4. A Berchtold and A.E Raftery. The Mixture Transition Distribution Model for High-Order Markov Chains and Non-Gaussian Time Series. Statistical Science, 17(3):328-356, 2002.
5. E Bingham, A Kaban and M Girolami. Topic Identification in Dynamical Text by Complexity Pursuit, Neural Processing Letters 17: 1-15, 2003, pp. 69–83.
6. C Cortes, D Pregibon and C Volinsky. Communities of interest. Lecture Notes in Computer Science 2189. 2001.

7. D. Cohn and H. Chang. Learning to Probabilistically Identify Authoritative Documents. Proc. of 17th Int. Conf. on Machine Learning, pp. 167-174, 2000.
8. A.P Dempster, N.M Laird and D.B Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society. Series B (Methodological), Vol. 39, No. 1. (1977), pp. 1-38.
9. N.J Dingle, P.G Harrison and W.J Knottenbelt. Uniformization and Hypergraph Partitioning for the Distributed Computation of Response Time Densities in Very Large Markov Models. Journal of Parallel and Distributed Computing, 64, 2004, pp. 908–920.
10. L Danon, A Diaz-Guilera, J Duch and A Arenas. Comparing Community Structure Identifications. Journal of Statistical Mechanics, 2005. P09008.
11. D Gildea, T Hofmann. Topic-Based Language Models Using EM. Proc. of the 6th European Conference on Speech Communication and Technology, pp 2167-2170, 1999.
12. R.H Lambert. Multi-channel blind deconvolution: FIR matrix algebra and separation of multipath mixtures, Ph.D. Thesis, Univ. of Southern California, 1996.
13. M.E.J Newman. Detecting community structure in networks. Euro. Phys. Journal B, 38: 321–330, 2004.
14. A.E Raftery. A Model for High-order Markov Chains. *Journal of the Royal Statistical Society*, series B, **47**:528-539, 1985.
15. P Sarkar and A Moore. Dynamic Social Network Analysis using Latent Space Models. Proc. Neural Information Processing Systems, 2005.
16. L.K Saul and M.I Jordan. Mixed Memory Markov Models: Decomposing Complex Stochastic Processes as Mixtures of Simpler Ones. Machine Learning, 37(1):75-87, 1999.
17. L. Saul and F. Pereira. Aggregate and Mixed-Order Markov Models for Statistical Language Processing. Proc. of the Second Conference on Empirical Methods in Natural Language Processing, pp. 81-89, 1997.
18. J.V. Stone: Blind deconvolution using temporal predictability. Neurocomputing 49(1-4): 79-86 (2002)
19. L.R Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proc. of the IEEE, 77(2): 257–286, 1989.
20. B. Ripley. Pattern Recognition in Neural Networks. Cambridge Univ. Press, 1996.
21. K Rose, E. Gurewitz and G.C Fox. Statistical Mechanics and Phase Transitions in Clustering. Phys. Rev. Letters, vol. 65, no. 8, pp.945–948.

Patching Approximate Solutions in Reinforcement Learning

Min Sub Kim[1] and William Uther[2]

[1] ARC Centre of Excellence for Autonomous Systems, School of Computer Science and Engineering, University of New South Wales, Sydney NSW 2052, Australia
[2] National ICT Australia, Sydney NSW 2052, Australia
msk@cse.unsw.edu.au, william.uther@nicta.com.au

Abstract. This paper introduces an approach to improving an approximate solution in reinforcement learning by augmenting it with a small overriding patch. Many approximate solutions are smaller and easier to produce than a flat solution, but the best solution within the constraints of the approximation may fall well short of global optimality. We present a technique for efficiently learning a small patch to reduce this gap. Empirical evaluation demonstrates the effectiveness of patching, producing combined solutions that are much closer to global optimality.

1 Introduction

Approximations are widely used in reinforcement learning to cope with large state spaces. The potential advantages offered by approximations include reduced storage requirements and faster learning than a flat solution. The main drawback is that it may be impossible to represent the globally optimal solution, and the best solution within the constraints of the approximation may be arbitrarily worse than global optimality.

In this paper we discuss a technique for learning a small patch, which, when combined with an approximate solution to a reinforcement learning problem, produces performance much closer to the global optimal. This is motivated by the observation that the sub-optimality of many approximate solutions may be attributed to sub-optimal behaviour in small but important regions of the state space. Augmenting the approximate solution with an overriding patch can overcome the sub-optimality in these regions while retaining the benefits of approximation elsewhere.

2 Background

We adopt the usual reinforcement learning setting of finite Markov Decision Problems with discrete time steps [1], with the following notation. A Markov Decision Problem M is a 5-tuple $< S, A, P, R, S_0 >$ where S is a finite set of states, A is a finite set of actions, $P(s, a, s')$ is the probability of reaching state s' after executing action a in state s, $R(s, a)$ is the immediate reward received for executing action a in state s, and $S_0(s)$ is the probability that M starts in state s.

The objective is to learn a policy $\pi : S \rightarrow A$ that optimises some measure of future reward. In this paper, we will use expected undiscounted reward: the expected sum of reward from following the policy until reaching a terminal state. However, the methods discussed are also applicable with discounting.

We use the action-value or Q function [2] to represent the expected value of a policy. Specifically, for state-action (s, a) and policy π, $Q^\pi(s, a)$ is defined using the Bellman equation:

$$Q^\pi(s, a) = R(s, a) + \sum_{s' \in S} P(s, a, s')Q^\pi(s', \pi(s')) \tag{1}$$

For learning the patch, we adapt prioritised sweeping [3], a model-based reinforcement learning algorithm that efficiently orders backups using a priority queue. At each step, the most recent state-action is promoted to the top of the backup queue. Then, before the next action is taken, a certain number of state-actions are removed from the top of the queue and processed one at a time. Processing a state-action consists of updating its Q value, and adding its predecessors to the backup queue, with priority equal to the expected change in Q value. This has the effect of concentrating computation where the Q function is changing most rapidly.

3 Related Work

Patching starts with an approximate solution and incrementally learns over it, an approach shared by many other methods. One of the earliest on-line algorithms to use this approach was Learning real-time A* [4], a real-time search algorithm. A heuristic cost function serves as the initial approximation, and as the agent searches the problem, it is incrementally overridden by a revised cost function. Real-time dynamic programming [5] generalises Learning real-time A* to Markov Decision Problems. However, both algorithms assume that storage is allocated for all states visited in practice, which becomes intractable over time for large problems.

An alternative approach is to initialise the solution with the approximation and then learn over it directly, instead of incrementally building a partial override. Naively seeding the value function in this manner may cause learning to be slower than starting from scratch [6]. However, careful application has been shown to be capable of accelerating learning [7].

A related method from multi-agent learning is Sparse cooperative Q-learning [8]. In this approach, the value function is approximated by agent-wise decomposition for some states, but depends on the entire joint state for others. This kind of partially abstract, partially flat value function is similar to that produced by patching, although patching is not limited to this particular type of decomposition. Coordination dependencies are specified by the user in this algorithm; Utile coordination [9] is an extension of Sparse cooperative Q-learning that detects coordination dependencies automatically.

4 Patching in Reinforcement Learning

4.1 A Small Example

To illustrate the basic concepts and motivation for patching, consider the problem shown in Fig. 1. Two actors, A and B, are initially placed randomly on two separate paths with the goal of reaching Home.

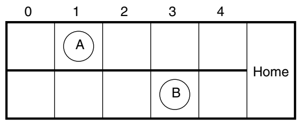


Fig. 1. A small coordination problem.

At each time step, each actor may either Move one cell to the right if it is not at Home, or Wait in its current position. These actions are deterministic. Each actor contributes a reward of 10 if it Moves to Home on that time step, or 0 if it is already at Home, or -1 otherwise. The total reward at each time step is the sum of the individual actor rewards, except if both actors Move to Home simultaneously, in which case the reward is 100. The task is undiscounted and terminates when both actors have reached Home.

An intuitive approximation for this problem is to assume that the actors are entirely independent. This divides the problem into two separate and identical sub-problems of one actor reaching Home individually. This type of decomposition is referred to as a *parallel decomposition* [10], because the problem is divided into sub-problems that "run in parallel". These sub-problems are much smaller and easier to solve separately than the whole problem. In this example, the optimal policy for one actor (defined over its individual state-action space only) is to Move on each step to reach Home and quickly as possible. Therefore, by combining the sub-problem solutions, the approximate solution suggests that both actors Move at every step until reaching Home.

The price paid for the reduction in solution complexity is that the solution is not optimal: the approximation is unable to represent that it is more profitable for the actors to cooperate on the last step before Home to collect the large combined reward. In terms of the policy value, it is easy to see that the approximate policy may have arbitrarily worse expected reward than optimal, depending on the value of the combined reward for reaching Home together. However, from the perspective of the policy, the approximate solution only requires modification at that last step. Ideally, we would like to bridge the gap to optimality, but without reverting to a flat Q table over the entire problem. This is exactly the aim of patching.

4.2 Specifying the Approximate Solution

We assume that the approximate solution is specified by:

– an approximate Q function, $\hat{Q}$;
– an approximate model of the transitions, $\hat{P}$, and rewards $\hat{R}$.

There are no strict requirements on the underlying representation of these functions, but in practice, it is expected that they will be compactly represented, *e.g.* $\hat{Q}$ may be a hierarchically decomposed Q function. We assume that these functions are pre-computed and fixed throughout patch learning.

For our example problem, $\hat{Q}$ is the sum of the corresponding single actor Q values: the expected reward for both actors reaching Home is estimated as the expected reward for the two actors reaching Home separately. This can be calculated on demand by looking up the corresponding entries in the single actor Q table. $\hat{P}$ and $\hat{R}$ are calculated similarly using their single actor counterparts.

4.3 Patching the Q Function

Patching aims to improve on the policy defined by $\hat{Q}$ by overriding some values in $\hat{Q}$. This override combines with $\hat{Q}$ to form the Q function representation used by patching.

Definition 1. *The* Q function patch*,* $Q_{\text{patch}} : S \times A \rightarrow \Re$ *is a partial function that overrides some values of* $\hat{Q}$*. Then, for any state-action pair* (s, a)*:*

$$Q(s,a) = \begin{cases} Q_{\text{patch}}(s,a) & \textit{if } Q_{\text{patch}}(s,a) \textit{ defined} \\ \hat{Q}(s,a) & \textit{otherwise} \end{cases} \tag{2}$$

The “default” choice of representation for Q_{patch} is a dynamically sized hashtable over flat state-action pairs, holding only as many entries as are added to it. Q_{patch} may also employ abstractions, but this requires careful design: it needs to have sufficient representational power to cover sub-optimalities caused by approximations used in $\hat{Q}$, but over-generalising may make the solution worse. In this representation, updates to the Q function are made by adding or updating entries in Q_{patch}, overriding the value in $\hat{Q}$.

Patches to cover inaccuracies in $\hat{P}$ and $\hat{R}$ are defined analogously, and have similar conventions for partial override of $\hat{P}$ and $\hat{R}$. We omit details due to lack of space; full details are presented in an accompanying technical report [11].

4.4 Seeding the Patch

Having decided how the patch is represented, the next problem is to decide which action values should be added to the patch.

Definition 2. *The* patch seed predicate*, defined over state-actions, indicates the starting points for* Q_{patch}*, from which it will grow.*

We require the user to supply the patch seed predicate. A reasonable strategy for seeding the patch is to focus on the parts of the problem where $\hat{Q}$ may lack sufficient representational power, or where structural assumptions and abstractions used in $\hat{Q}$ may over-generalise. In general, patch seeding is not intended to be an exact listing of sub-optimalities in the approximation, but allows the user to suggest regions of the problem that deserve attention, instead of growing the patch blindly over the entire state-action space. An automatic method for seeding the patch that we use in our experiments is to detect inaccuracies in the model, discussed in Sect. 5.4. This is appropriate when $\hat{Q}$, $\hat{P}$, and $\hat{R}$ all depend on the same structural assumptions for approximation.

For our example, the assumption made when constructing the approximation was that the actors are entirely independent. However, when both actors Move to Home simultaneously, they will receive a combined reward of 100 instead of the sum of individual rewards predicted by $\hat{R}$ (+10 for each actor, for a total of 20). Therefore, this state-action is a patch seed, indicating that $\hat{Q}$ may be inaccurate around this state-action, and therefore the policy may be improved by patching around this state-action.

5 Learning the Patch

5.1 Unbounded Patching

With the initial approximation and patch seed predicate set, we now need an algorithm to learn Q_{patch}. Intuitively, we want to improve the policy defined by $\hat{Q}$ by adding override values to Q_{patch}, starting from the areas of interest suggested by the seed predicate.

Unbounded patching directly adapts prioritised sweeping for this purpose as follows. We follow the policy according to the current Q values as per usual. Then, if the most recent state-action is a patch seed according to the seed predicate, it is added to the backup queue. It then proceeds as per prioritised sweeping, by processing entries from the backup queue and making model-based Q function updates, with the difference that updates are made by adding or updating values in Q_{patch}. This has the effect of quickly growing Q_{patch} from the patch seeds through predecessors, adjusting the policy as it proceeds.

Eventually, unbounded patching will add all ancestors of all patch seeds to Q_{patch}, effectively reverting to prioritised sweeping over the flat problem. This is not unexpected, since unbounded patching is just prioritised sweeping with the first entries to the backup queue determined by patch seeds, combined with partial override by Q_{patch}. We need heuristics to bound patch growth while still repairing the policy where required.

5.2 Policy Bounding

Unbounded patching propagates changes in value from the seed points through predecessors. In our example problem, a lot of these changes in value do not affect the policy. An example is both actors Waiting in their current position: this is equivalent to a null action and is not optimal in any state, but nevertheless, unbounded patching will patch it as long as it is a patch seed ancestor. Consequently, values are added to Q_{patch} without actually improving the resulting behaviour.

Policy bounding adds the restriction that patch values are added only when they immediately affect the current greedy policy (including the current set of Q_{patch} values). This can be seen as using immediate change in the policy as a heuristic to decide whether growing the patch is still effective. Under policy bounding, a proposed update for state-action (s, a) that is not currently in Q_{patch} is accepted if either the greedy action at s would change, or if a is the greedy action at s and has a non-zero probability of self-transition[1]. This condition is checked both when entries are added to and removed from the backup queue.

Policy bounding tends to constrain patching to only local adjustments around patch seeds. This usually keeps patch sizes smaller, but also limits patching to local policy repair only. In general, if the approximate solution requires correction on a more global scale, then policy bounding will either ignore some corrections or be ineffective in reducing Q_{patch} growth.

5.3 Utility Bounding

If there are hard limits on storage, policy bounding alone may not be sufficient. In this case, we would like to obtain the greatest improvement possible from the limited storage. One way to do this is to rank entries in Q_{patch} according to some measure of usefulness, so that the least useful values can be discarded if necessary.

Utility bounding implements Q_{patch} as a priority queue with fixed capacity specified by the user. Entries are prioritised by the absolute difference between the patched value and the corresponding value in $\hat{Q}$, *i.e.*, priority will be highest where $\hat{Q}$ is least accurate. This can be seen as using estimated Q function error as a heuristic measure of usefulness of entries in Q_{patch}. If Q_{patch} exceeds capacity, the state-action with lowest priority is dropped, reverting the Q function to $\hat{Q}$ for that state-action.

5.4 Patching the Model

Patching relies on the model to calculate the adjusted values for Q_{patch}. As with patching for the Q function, we augment the provided estimates $\hat{P}$ and $\hat{R}$ with partial patches, and avoid building the entire flat model by patching only the inaccuracies in the estimates. We omit details due to lack of space, but sketch the procedures briefly. Full details are presented in an accompanying technical report [11].

Inaccuracies in the approximate transition and reward models are detected with the χ^2 and Kolmogorov-Smirnov statistical tests. Transition and reward

[1] This is required because greedy actions re-use their own value to calculate their updated value.

samples are collected during learning, and compared to $\hat{P}$ or $\hat{R}$ once enough samples have been observed. If the test shows a significant difference between the distributions, then that state-action is patched with the observed samples, and future references to that state-action refer to the patched distribution rather than $\hat{P}$ or $\hat{R}$. Storage for sampling is kept limited by selective sampling, directing storage to those transitions experienced most frequently in practice.

6 Experiments

We evaluate patching on two domains, to examine the effectiveness of patching in bridging the gap to global optimality. We compare patching against two instances of prioritised sweeping:

- *Prioritised sweeping from scratch*: All Q values are initialised to 0, and the agent's estimates of P and R are built from scratch. This provides a lower baseline to determine whether the initial approximate solution is helpful.
- *Initialised prioritised sweeping*: All Q values are initialised to $\hat{Q}$'s values, and the approximate model is provided and updated by patching, as discussed in Sect. 5.4. The only differences between this algorithm and patching that affect the policy are patch seeding and the bounding heuristics, thus providing a measure of effectiveness of those aspects.

For all experiments, we use ε-greedy exploration, with $\varepsilon = 0.1$. A maximum of 2 state-actions were processed from the backup queue per step. All plots in this section show the average and standard deviation over 10 runs for each experiment.

6.1 Modified Taxi

The first set of experiments uses a modified version of Dietterich's taxi problem [12]. In this problem, a taxi agent in a 5-by-5 grid world (shown in Fig. 2) has the objective of delivering a passenger from a specially marked taxi stand to a destination taxi stand.

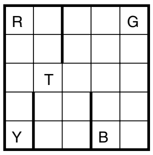


Fig. 2. The taxi problem. R, G, B, Y indicate the taxi stands, T indicates the taxi.

States are described by three variables: the taxi location, the passenger location, and the passenger destination. The taxi has stochastic navigation actions in the four compass point directions that move one cell in the intended direction with probability 0.8 and to the left or right of the intended direction with probability 0.1

each, subject to barriers that block movement (marked by thicker lines in Fig. 2). Two special actions are also available for picking up and putting down the passenger, effective only when the taxi is at the correct stand. The reward is -10 for failed pick-up or put-down actions, +19 on successful delivery, or -1 otherwise. The task is undiscounted and terminates on successful delivery of the passenger.

For the original taxi problem, MAXQ can be used to efficiently learn a compact hierarchical solution. Importantly, the task hierarchy includes navigation sub-tasks for each taxi stand. These sub-tasks are context independent with respect to the passenger location and destination – the optimal policy to navigate to a particular stand is the same regardless of the passenger.

We use the MAXQ solution as the initial approximation on a modified version of the taxi problem, and patch over it to handle the modifications. In the modified problem, 40 navigation state-actions in the middle row of the grid that are optimal in the original problem are modified. These modified actions have an irregular outcome in either P or R, such that the navigation sub-tasks are now *not* entirely context independent with respect to the passenger. Modified state-actions in P move in the intended direction with probability 0.1 and to the left or right of the intended direction with probability 0.45 each. Modified state-actions in R incur an unexpectedly costly reward of -6 with probability 0.8, and the usual -1 otherwise. These changes are deliberately conceived to be costly – an optimal policy for the original problem falls well short of global optimality when directly applied to the modified problem.

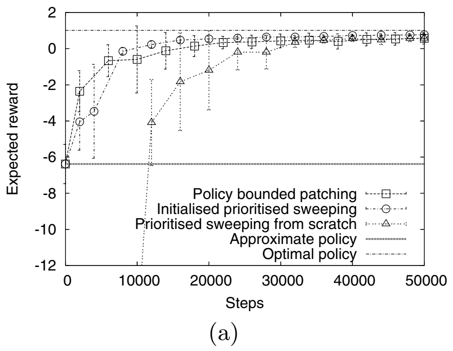
We apply patching in this domain as follows. $\hat{Q}(s, a)$ is calculated on demand from the MAXQ value function by finding the highest value path in the task hierarchy from the root node to the leaf node for a [2]. $\hat{Q}$ requires 632 values. $\hat{P}$ and $\hat{R}$ are initialised to the transition and reward models of the original taxi domain. Since $\hat{Q}$, $\hat{P}$, and $\hat{R}$ are all based on the original domain, it is reasonable to seed the patch at the transitions where $\hat{P}$ and $\hat{R}$ are found to be inconsistent with the modified domain. These inconsistencies in the model are detected using the procedures discussed in Sect. 5.4.

Figure 3(a) compares the expected reward for policy bounded patching and the two instances of prioritised sweeping. The solutions using the initial approximation have a clear head-start on prioritised sweeping from scratch, but the difference is reduced fairly quickly, and all algorithms reach policies of similar quality. In terms of storage, policy bounded patching settles with Q_{patch} coverage of approximately one third of the state-action space, requiring less total storage than the other algorithms.

If capacity for Q_{patch} is undersized, we can expect that the policy will consequently be worse. Figure 3(b) shows the expected reward for patching with policy and utility bounding, with Q_{patch} capacities of 800, 900, and 1,000 (26.7%, 30%, and 33.3% of the state-action space). As Q_{patch} capacity is reduced, the policy deteriorates, both in terms of expected reward and consistency.

Table 1 summarises the results for this domain.

[2] Subject to the termination predicates in the hierarchy – each sub-task in the path must be valid at s.



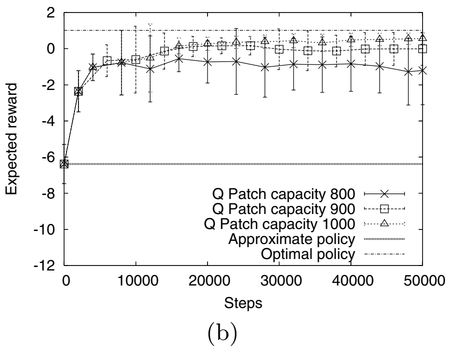


Fig. 3. Results for the modified taxi domain. Expected reward was determined by calculating the policy value, and averaging over the initial state distribution.

Table 1. Summary of results for the modified taxi domain. Statistics were taken at the end of 50,000 steps for all algorithms. For patched solutions, the total size is listed as the size of $\hat{Q}$ plus the size of Q_{patch}.

Solution	Expected reward	# Q values
Initial approximation ($\hat{Q}$)	-6.38 ± 1.09	632
Optimal flat	1.02	3000
Initialised prioritised sweeping	0.78 ± 0.07	3000
Prioritised sweeping from scratch	0.58 ± 0.19	3000
Policy bounded patching	0.58 ± 0.32	632 + 1035.60 ± 54.62
Policy and utility bounded patching		
– with Q_{patch} capacity 800	-1.21 ± 1.90	632 + 800
– with Q_{patch} capacity 900	-0.01 ± 0.91	632 + 900
– with Q_{patch} capacity 1000	0.56 ± 0.31	632 + 989.70 ± 16.64

6.2 Multi-taxi

The second set of experiments will examine patching on the multi-taxi problem: the grid remains the same as in the original taxi problem, but there are now two taxis and two passengers.

The multi-taxi problem is approximately equal to two instances of the original taxi problem running in parallel, but with some differences. Most importantly, the taxis are subject to collisions with each other, in which case neither taxi location is changed. A taxi may pick-up either passenger, but only one at a time. The reward is decomposed by passengers: at each time step, a reward of -1 is received for each undelivered passenger, plus -10 for each failed pick-up or put-down action. In addition to the action set from the original taxi problem, each taxi also has a null action for staying in place. The task is undiscounted and terminates when both passengers have been successfully delivered.

We apply patching in this domain as follows. $\hat{Q}$ is calculated by using a solution for the task of delivering one passenger with one taxi, requiring 4,200 values.

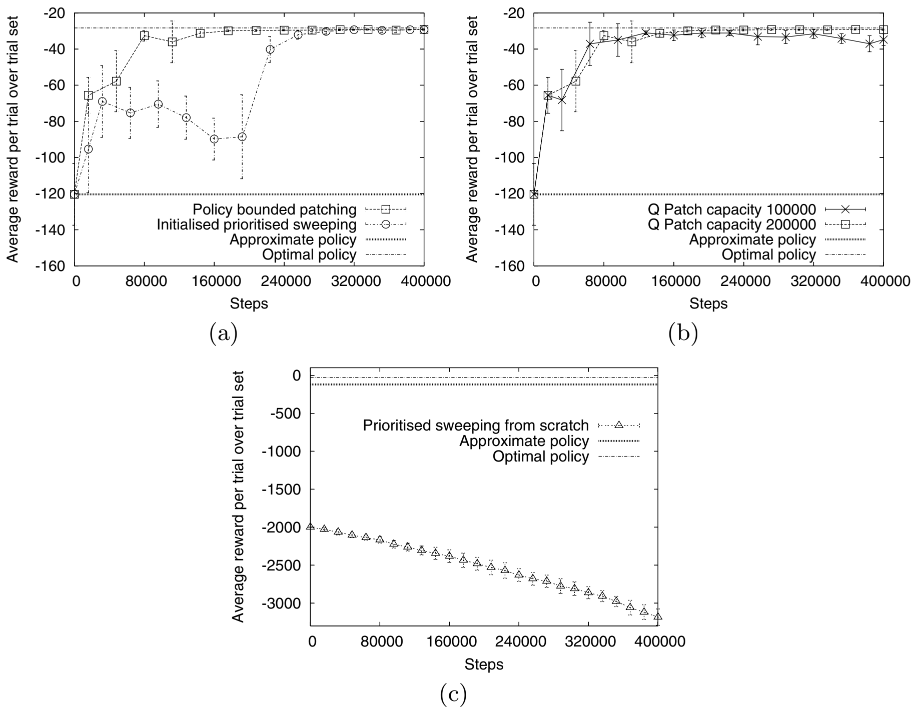


Fig. 4. Results for the multi-taxi domain. Average reward per trial was determined by evaluating the policy on a random test set of 1,000 initial states, fixed for each experiment run but different for separate runs. A maximum trial length of 1,000 steps was imposed for evaluation.

A hand-crafted allocation function determines allocation of taxis to passengers. Given an allocation, the expected reward for the two deliveries is estimated as the sum of the expected reward for the individual deliveries, *i.e.* assuming that the two taxis are entirely independent. $\hat{P}$ and $\hat{R}$ calculated similarly under the same assumption. Patch seeds are found by detecting inaccuracies in $\hat{P}$, which occur when the taxis collide. Both patching and prioritised sweeping make use of symmetry between the taxis to accelerate learning.

Figure 4(a) compares policy bounded patching and initialised prioritised sweeping. In this domain, patch seeding and bounding results in a noticeable difference in early performance – policy bounded patching reduces the gap to global optimality much faster than initialised prioritised sweeping by focusing updates to where the policy immediately requires correction.

In terms of storage, policy bounding alone does not appear sufficient in this domain to limit Q_{patch} growth. One reason for this is that most of the sub-optimality in $\hat{Q}$ can be resolved by handling collisions, but further small improvements to the policy are possible, *e.g.* cooperative strategies that make positive use of collisions. Figure 4(b) plots the expected reward with both policy bounding and utility bounding, for Q_{patch} capacities of 100,000 and 200,000 (0.6%

and 1.1% of the state-action space). Combining both bounding heuristics makes efficient use of the limited storage, with little loss in policy value compared to patching without utility bounding.

Lastly, Fig. 4(c) shows the expected reward for prioritised sweeping from scratch. While all algorithms initialised with $\hat{Q}$ had learning curves between $\hat{Q}$ and the optimal solution, prioritised sweeping from scratch starts far below $\hat{Q}$, and proceeds to blindly explore the problem. Clearly, while the initial approximation is not perfect, it is a much more preferable starting point to nothing.

Table 2 summarises the results for this domain.

Table 2. Summary of results for the multi-taxi domain. Statistics were taken at the end of 400,000 steps for all algorithms. For patched solutions, the total size is listed as the size of $\hat{Q}$ plus the size of Q_{patch}.

Solution	Reward per trial	# Q values
Initial approximation ($\hat{Q}$)	-120.37 ± 17.12	4200
Optimal flat	-28.29 ± 0.28	17434200
Initialised prioritised sweeping	-29.26 ± 0.32	17434200
Prioritised sweeping from scratch	-3184.23 ± 107.44	17434200
Policy bounded patching	-29.10 ± 0.33	4200 + 379263.10 ± 3444.89
Policy and utility bounded patching		
– with Q_{patch} capacity 100000	-34.75 ± 3.21	4200 + 100000
– with Q_{patch} capacity 200000	-29.26 ± 0.37	4200 + 200000

7 Conclusions and Future Work

In this paper, we introduced an approach to reinforcement learning in which an approximate solution is taken as the starting point, and patched to improve performance beyond the constraints imposed by the approximation. We started with unbounded patching as a direct adaptation of prioritised sweeping to patching, and proposed policy bounding and utility bounding as two heuristics for bounding patch growth. Empirical results demonstrated the effectiveness of patching, producing near optimal solutions with limited storage, using two different types of underlying Q function approximations.

Future work will aim to apply patching to larger problems, with more sophisticated approximations and patch functions. We used patching in the scope of entire tasks, but it may be possible to apply patching in separate components of a decomposed solution, such as at various levels of a task hierarchy.

Acknowledgements

We thank Claude Sammut, Bernhard Hengst, Robert Fitch, and Malcolm Ryan for helpful feedback that assisted in developing the ideas in this paper. We also thank the anonymous reviewers for their helpful comments.

This research is supported by the Australian Research Council Centre of Excellence for Autonomous Systems. National ICT Australia is funded through the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT Press (1998)
2. Watkins, C.J.C.H.: Learning from delayed rewards. PhD thesis, King's College, Oxford (1989)
3. Moore, A.W., Atkeson, C.G.: Prioritized sweeping: Reinforcement learning with less data and less real time. Machine Learning **13** (1993) 103–130
4. Korf, R.E.: Real-time heuristic search. Artificial Intelligence **42** (1990) 189–211
5. Barto, A.G., Bradtke, S.J., Singh, S.P.: Learning to act using real-time dynamic programming. Artificial Intelligence **72** (1995) 81–138
6. Bowling, M., Veloso, M.: Reusing learned policies between similar problems. In: Proceedings of the AI*IA-98 Workshop on New Trends in Robotics, Padua, Italy (1998)
7. Taylor, M.E., Stone, P.: Behavior transfer for value-function-based reinforcement learning. In: The 4th International Joint Conference on Autonomous Agents and Multiagent Systems, ACM Press (2005) 53–59
8. Kok, J.R., Vlassis, N.: Sparse cooperative Q-learning. In: Proceedings of the 21st International Conference on Machine Learning, ACM (2004) 481–488
9. Kok, J.R., Hoen, P.J., Bakker, B., Vlassis, N.: Utile coordination: Learning interdependencies among cooperative agents. In: IEEE Symposium on Computational Intelligence and Games. (2005)
10. Boutilier, C., Dean, T., Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. Journal of Artificial Intelligence Research **11** (1999) 1–94
11. Kim, M.S., Uther, W.: Patching approximate solutions in reinforcement learning. Technical Report 0610, School of Computer Science and Engineering, University of New South Wales (2006)
12. Dietterich, T.: Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research **13** (2000) 227–303

Fast Variational Inference for Gaussian Process Models Through KL-Correction

Nathaniel J. King and Neil D. Lawrence

Department of Computer Science,
University of Sheffield, Regent Court,
211 Portobello Street, Sheffield,
S1 4DP. United Kingdom
`{nat, neil}@dcs.shef.ac.uk`

Abstract. Variational inference is a flexible approach to solving problems of intractability in Bayesian models. Unfortunately the convergence of variational methods is often slow. We review a recently suggested variational approach for approximate inference in Gaussian process (GP) models and show how convergence may be dramatically improved through the use of a positive correction term to the standard variational bound. We refer to the modified bound as a KL-corrected bound. The KL-corrected bound is a lower bound on the true likelihood, but an upper bound on the original variational bound. Timing comparisons between optimisation of the two bounds show that optimisation of the new bound consistently improves the speed of convergence.

1 Introduction

A key problem with many variational approximations is the slow speed of convergence. In this paper we will show how the speed of convergence for variational approximations can be radically improved by 'KL-correction' of the variational bound. Empirically we find that our approach dramatically improves convergence speed for a range of benchmark data sets.

We consider the variational approximation proposed independently by [1] and [2]. This approximation allows us to consider the process of inference in the Gaussian process independently of the noise model [2]. We follow [2] in referring to this formulation of the variational approach as probabilistic point assimilation (PPA).

The paper is laid out as follows, in Sections 2 and 3, we introduce notation and describe the underlying probabilistic model, as well as the PPA variational approximation and the KL-corrected bound. In Section 4 we demonstrate the performance of the approach on some benchmark data sets, including timing comparisons, and we conclude in Section 5 with a short discussion.

2 Gaussian Processes

Consider a data set consisting of input data, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^{\mathrm{T}}$, and labels, $\mathbf{y} = [y_1, \ldots, y_N]^{\mathrm{T}}$. We will assume that the labels are dependent on an $N \times 1$

vector, $\mathbf{f} = [f_1, \ldots, f_N]^{\mathrm{T}}$ through a 'noise model' $p(y_n|f_n)$. The label y_n relates to $\mathbf{x}_n$ through the latent variable f_n. In the case of our simple classification noise model the relationship to f_n is given by,

$$p(y_n|f_n) = \phi(y_n f_n),$$

where $\phi(z) = \int_{-\infty}^{z} N(t|0,1)\,dt$ is the cumulative Gaussian distribution function and $N(\mathbf{z}|\boldsymbol{\mu}, \Sigma)$ denotes a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance Σ.

The latent variable is normally then related to the input data through a Gaussian process prior [3,4] over $\mathbf{f}$. For the moment we depart from this approach and define an additional spherical distribution over $\mathbf{f}$,

$$p(\mathbf{f}|\bar{\mathbf{f}}, \beta) = \prod_{n=1}^{N} p(f_n|\bar{f}_n, \beta) = \prod_{n=1}^{N} N(f_n|\bar{f}_n, \beta^{-1}),$$

where the β is a precision (inverse variance), and $\bar{\mathbf{f}}$ is a vector of means, the nth element being $\bar{f}_n$. Clearly under this definition $\mathbf{y}$ is independent of $\mathbf{X}$, to rectify this we now introduce a prior distribution over $\bar{\mathbf{f}}$,

$$p(\bar{\mathbf{f}}|\mathbf{X}, \boldsymbol{\theta}) = N(\bar{\mathbf{f}}|\mathbf{0}, \mathbf{K}),$$

which is a Gaussian process prior over $\bar{\mathbf{f}}$ with a mean of zero and a covariance function $\mathbf{K}$. This matrix is a function of $\mathbf{X}$ and its form is controlled by a set of parameters, $\boldsymbol{\theta}$. Note that this prior distribution can be combined with our distribution over $\mathbf{f}$ to obtain

$$p(\mathbf{f}|\mathbf{0}, \mathbf{K}) = \int \prod_{n=1}^{N} p(f_n|\bar{f}_n, \beta)\, N(\bar{\mathbf{f}}|\mathbf{0}, \mathbf{K})\, d\bar{\mathbf{f}} = N(\mathbf{f}|\mathbf{0}, \mathbf{K} + \beta^{-1}\mathbf{I}),$$

which, since a diagonal term is often added to the kernel matrix, does not in practice lead to a richer model. However, as we shall see, augmentation of the basic model with the vector of means $\bar{\mathbf{f}}$ renders the application of variational approaches to the model more convenient.

The marginal likelihood of a data set can be obtained through marginalisation of the latent variables $\mathbf{f}$ and $\bar{\mathbf{f}}$,

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \beta) = \int N(\bar{\mathbf{f}}|\mathbf{0}, \mathbf{K}) \prod_{n=1}^{N} \int p(y_n|f_n)\, p(f_n|\bar{f}_n, \beta)\, df_n d\bar{\mathbf{f}}. \tag{1}$$

In practise we will find that for non-Gaussian noise models this marginal likelihood will not be tractable, forcing us to turn to approximate methods.

2.1 Variational Inference

Variational inference is a popular choice for approximate inference in Bayesian models. In [2] we showed how to implement variational inference in Gaussian processes in a generic manner, we refered to this approach as probabilistic point assimilation (PPA). The same approach was also independently suggested by [1] in the context of multi-class classification in Gaussian processes.

The first step in PPA is to introduce an approximating distribution, $q\left(\bar{\mathbf{f}}\right)$, for the mean parameters giving

$$\log p\left(\mathbf{y}|\mathbf{X},\boldsymbol{\theta},\beta\right) \geq \sum_{n=1}^{N}\left\langle \log p\left(y_n|\bar{f}_n,\beta\right)\right\rangle_{q\left(\bar{\mathbf{f}}\right)} + \left\langle \log p\left(\bar{\mathbf{f}}|\mathbf{X},\boldsymbol{\theta}\right)\right\rangle_{q\left(\bar{\mathbf{f}}\right)} - \left\langle \log q\left(\bar{\mathbf{f}}\right)\right\rangle_{q\left(\bar{\mathbf{f}}\right)}. \tag{2}$$

This is in effect the standard variational formalism for Gaussian processes. Ideally we would now seek to maximise the bound through free-form optimisation with respect to $q\left(\bar{\mathbf{f}}\right)$ [5]. Unfortunately, for most noise models, such a free form optimisation of the bound is not possible. The next step is, therefore, to assume a form for $q\left(\bar{\mathbf{f}}\right)$ which renders the bound tractable. Seeger [6] made the natural assumption that $q\left(\bar{\mathbf{f}}\right)$ is a Gaussian process and sought its mean and covariance by maximising the resulting bound. Unfortunately this approach greatly complicates the process of inference as it demands gradient based optimisation of the variational bound, which for practicality often requires further constraints on the posterior covariance matrix. In PPA we depart from the standard approach through introduction of a further approximating distribution, $q\left(\mathbf{f}\right)$, to lower bound the first term of (2),

$$\begin{aligned}\log p\left(\mathbf{y}|\mathbf{X},\boldsymbol{\theta},\beta\right) \geq & \sum_{n=1}^{N}\left\langle \log p\left(y_n|f_n\right)\right\rangle_{q(f_n)} + \sum_{n=1}^{N}\left\langle \log p\left(f_n|\bar{f}_n,\beta\right)\right\rangle_{q\left(\bar{f}_n\right)q(f_n)} \\ & + \left\langle \log p\left(\bar{\mathbf{f}}|\mathbf{X},\boldsymbol{\theta}\right)\right\rangle_{q\left(\bar{\mathbf{f}}\right)} - \sum_{n=1}^{N}\left\langle \log q\left(f_n\right)\right\rangle_{q(f_n)} \\ & - \left\langle \log q\left(\bar{\mathbf{f}}\right)\right\rangle_{q\left(\bar{\mathbf{f}}\right)} = \mathcal{L}. \end{aligned} \tag{3}$$

Each of the two lower bounds we have made use of can independently be made to be equalities if their variational distributions are optimised, however when combined they will only reach equality if the true posterior distribution factorises. For later convenience we shall refer to this bound (3) as the standard variational approach. The key advantage associated with introduction of the second lower bound is that we can now perform free-form optimisation of the posterior approximations [5] in the manner of standard variational inference. Under free-form optimisation it turns out that the approximating distribution over $\mathbf{f}$ factorises, $q\left(\mathbf{f}\right) = \prod_{n=1}^{N} q\left(f_n\right)$, with each factor being given by

$$q\left(f_n\right) \propto p\left(y_n|f_n\right)\exp\left\langle \log p\left(f_n|\bar{f}_n,\beta\right)\right\rangle_{q\left(\bar{f}_n\right)}.$$

Recalling that $p\left(f_n|\bar{f}_n,\beta\right)$ is a Gaussian distribution, we can re-write this formula as

$$q\left(f_n\right) = \frac{1}{Z_n}p\left(y_n|f_n\right)N\left(f_n|\left\langle \bar{f}_n\right\rangle,\beta^{-1}\right), \tag{4}$$

where the normalisation constant is given by Z_n. The tractability of the normalisation constant is dependent on the form of the noise model. However, even when Z_n is analytically intractable, the integral can be solved numerically through quadrature.

Different Noise Models. A key advantage of the PPA approach is that we can make use of many different noise models in (4) without significantly changing our algorithm. This is achieved in the following manner. It is well known (see *e.g.* [7]) that expectations under distributions of the form given in (4) can be computed through differentiation of $\log Z_n$. The mean of (4) can be shown to be

$$\langle f_n \rangle = \left\langle \bar{f}_n \right\rangle + \beta^{-1} g_n$$

where $g_n = \nabla_{\left\langle \bar{f}_n \right\rangle} \log Z_n$ and the second moment can be shown to be

$$\left\langle f_n^2 \right\rangle = 2\beta^{-2} \varGamma_n + \beta^{-1} + 2 \left\langle \bar{f}_n \right\rangle \langle f_n \rangle - \left\langle \bar{f}_n \right\rangle^2$$

with $\varGamma_n = \nabla_{\beta_n^{-1}} \log Z_n$. For a given noise model of interest, it is therefore only necessary to compute $\log Z_n = \log \int p\left(y_n | f_n\right) N\left(f_n | \left\langle \bar{f}_n \right\rangle, \beta^{-1}\right) df_n$ for it to be used in the inference process. This was our main motivation in describing this model within [2].

Approximating Distribution for $\bar{\mathbf{f}}$. The moments under $q\left(f_n\right)$ can be used to find the form of the approximating component associated with the mean vector $\bar{\mathbf{f}}$. Free-form optimisation of the variational bound with respect to $q\left(\bar{\mathbf{f}}\right)$ recovers

$$q\left(\bar{\mathbf{f}}\right) \propto p\left(\bar{\mathbf{f}} | \mathbf{X}, \boldsymbol{\theta}\right) \prod_{n=1}^{N} \exp \left\langle \log p\left(f_n | \bar{f}_n, \beta_n\right) \right\rangle . \tag{5}$$

This implies that $q\left(\bar{\mathbf{f}}\right)$ has the form of a Gaussian process,

$$q\left(\bar{\mathbf{f}}\right) = N\left(\bar{\mathbf{f}} | \mu, \mathbf{C}\right)$$

whose posterior covariance function is given by $\mathbf{C} = \left(\mathbf{K}^{-1} + \beta \mathbf{I}\right)^{-1}$, while the posterior mean function is given by $\mu = \beta \mathbf{C} \langle \mathbf{f} \rangle$. Computation of the required moments under this process posterior is straightforward, the first moment is given by $\left\langle \bar{\mathbf{f}} \right\rangle = \mu$ and the second moment by $\left\langle \bar{\mathbf{f}} \bar{\mathbf{f}}^{\mathrm{T}} \right\rangle = \mathbf{C} + \mu \mu^T$. Note that the first and second moment of our posterior approximation can be computed by inspection; contrast this with the situation in [6] where these moments must be found through gradient based methods.

We also see that the form of $q\left(\bar{\mathbf{f}}\right)$ is not directly dependent on the form of the noise model. This dependence occurs through the latent variables $\mathbf{f}$.

3 Updating Parameters

One of the advantages of the Gaussian process framework is that we can seek to optimise kernel parameters through optimisation of the model's log-likelihood. In approximate variational inference direct optimisation of the marginal likelihood is not possible; instead we seek to maximise the variational lower bound. For our model the relevant terms of the bound are

$$L(\beta, \boldsymbol{\theta}) = \left\langle \log p\left(\bar{\mathbf{f}}|\mathbf{X}, \boldsymbol{\theta}\right)\right\rangle_{q\left(\bar{\mathbf{f}}\right)} + \sum_{n=1}^{N} \left\langle \log p\left(f_n|\bar{f}_n, \beta\right)\right\rangle_{q\left(\bar{f}_n\right)q(f_n)} = \mathcal{L}(\theta) \quad (6)$$

The bound is normally optimised with respect to $\boldsymbol{\theta}$ by gradient based methods.

3.1 KL-Corrected Inference

A common problem with variational methods is slow convergence to a maximum. This can occur if the quality of the bound as a function of the parameters, $\mathcal{L}(\boldsymbol{\theta})$, falls away rapidly as $\boldsymbol{\theta}$ changes. In other words convergence will be slow if the quality of the bound is very sensitive to changes in the parameters. The effect is shown in Figure 1(a). The motivation behind this paper was to discover whether we could obtain an upper bound, $\mathcal{L}'(\boldsymbol{\theta})$, on (3) which is also a lower bound on the true likelihood, then we are also likely to achieve faster convergence. The intuition behind this idea is shown schematically in Figure 1(b). If $\mathcal{L}'(\boldsymbol{\theta})$ is an upper bound on $\mathcal{L}(\boldsymbol{\theta})$ and a lower bound on the true likelihood $L(\boldsymbol{\theta})$ then its maxima is likely to be closer to the maxima of $L(\boldsymbol{\theta})$ than the maxima of $\mathcal{L}(\boldsymbol{\theta})$ is.

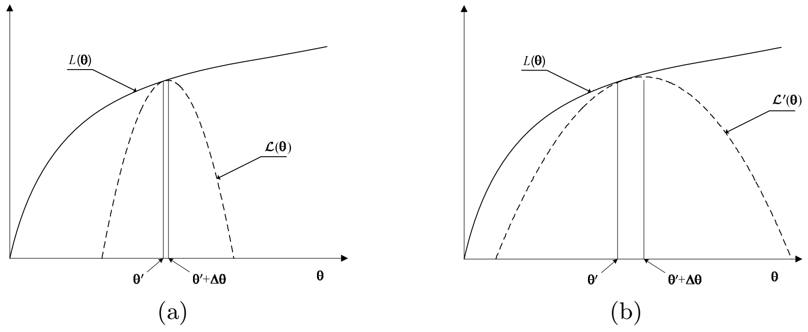


Fig. 1. Variational optimisation. (a) The schematic shows the log likelihood, $L(\boldsymbol{\theta})$ as a function of the parameters and the variational lower bound, $\mathcal{L}(\boldsymbol{\theta})$. The lower bound is shown as being quadratic in the parameters. The bound shown has been maximised with respect to the q-distributions for a set of parameters $\boldsymbol{\theta}'$, however the bound falls away sharply for quite small changes in $\boldsymbol{\theta}$. As a result optimisation of the lower bound with respect to $\boldsymbol{\theta}$ leads to only a small change $\Delta\boldsymbol{\theta}$. Many iterations are required for convergence. (b) Here we show a schematic of the effect of KL-correction of the bound. The bound is less sensitive to the variational distributions and it falls away from the likelihood less quickly, as a result larger steps are taken when $\boldsymbol{\theta}$ is optimised.

An Improved Bound. Ideally we would like to optimise the marginal likelihood,

$$L(\boldsymbol{\theta}) = \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \beta) = \log \int \prod_{n=1}^{N} p\left(y_n|\bar{f}_n, \beta\right) p\left(\bar{\mathbf{f}}|\mathbf{X}, \boldsymbol{\theta}\right) d\bar{\mathbf{f}}, \quad (7)$$

with respect to $\boldsymbol{\theta}$; unfortunately the integral is, in general, intractable. We previously discussed the fact that the log of the noise model can be lower bounded variationally. This lower bound is maintained when taking the exponential of both sides (as the exponential is a monotonic function). Thus, the noise model is lower bounded by

$$p\left(y_n|\bar{f}_n,\beta_n\right) \geq \exp\left(\left\langle \log p\left(y_n|f_n\right)\right\rangle_{q(f_n)} + \left\langle \log p\left(f_n|\bar{f}_n,\beta\right)\right\rangle_{q(f_n)} \right. \\ \left. - \left\langle \log q\left(f_n\right)\right\rangle_{q(f_n)}\right). \tag{8}$$

Substituting this expression into the marginal log likelihood (7) gives the following lower bound

$$\mathcal{L}'\left(\boldsymbol{\theta}\right) = \log \int \prod_{n=1}^{N} \exp\left\langle \log p\left(f_n|\bar{f}_n,\beta\right)\right\rangle_{q(f_n)} p\left(\bar{\mathbf{f}}|\mathbf{X},\boldsymbol{\theta}\right) d\bar{\mathbf{f}} - \sum_{n=1}^{N} \left\langle \log q\left(f_n\right)\right\rangle_{q(f_n)} \\ + \sum_{n=1}^{N} \left\langle \log p\left(y_n|f_n\right)\right\rangle_{q(f_n)} \leq L\left(\boldsymbol{\theta}\right). \tag{9}$$

Note that the only term in this bound which is now dependent on $\boldsymbol{\theta}$ is the first term. The integral in this term can be computed analytically. To see this we first rewrite it as a Gaussian integral,

$$\mathcal{L}'\left(\boldsymbol{\theta}\right) = \log \int \prod_{n=1}^{N} \exp\left\langle \log p\left(f_n|\bar{f}_n,\beta\right)\right\rangle_{q(f_n)} p\left(\bar{\mathbf{f}}|\mathbf{X},\boldsymbol{\theta}\right) d\bar{\mathbf{f}} + \text{const} \\ = \log \int \prod_{n=1}^{N} N\left(\left\langle f_n\right\rangle|\bar{f}_n,\beta^{-1}\right) p\left(\bar{\mathbf{f}}|\mathbf{X},\boldsymbol{\theta}\right) d\bar{\mathbf{f}} + \text{const}, \tag{10}$$

leading to a tractable objective function for $\boldsymbol{\theta}$ that does not directly depend on $q\left(\bar{\mathbf{f}}\right)$. The result is a new bound that is actually an upper bound on the original variational lower bound. It thus has the characteristics suggested in Section 3.1 which are conducive to faster convergence.

Positive Correction Term. The KL-corrected bound is still a lower bound on the log-likelihood, however it is typically a tighter bound than the standard variational bound: it contains a correction term which is always positive or zero. The KL-corrected bound (9) can be rewritten using (3) as

$$L\left(\boldsymbol{\theta}\right) \geq \mathcal{L}\left(\boldsymbol{\theta}\right) + \text{KL}\left(q\left(\bar{\mathbf{f}}\right)||p\left(\bar{\mathbf{f}}|\left\langle\mathbf{f}\right\rangle,\mathbf{X},\boldsymbol{\theta}\right)\right)$$

where $\text{KL}\left(q\left(\bar{\mathbf{f}}\right)||p\left(\bar{\mathbf{f}}|\left\langle\mathbf{f}\right\rangle\mathbf{X},\boldsymbol{\theta}\right)\right)$ is the Kullback-Leibler divergence[1] between the distribution $q\left(\bar{\mathbf{f}}\right)$ and

$$p\left(\bar{\mathbf{f}}|\left\langle\mathbf{f}\right\rangle,\mathbf{X},\boldsymbol{\theta}\right) \propto \prod_{n=1}^{N} N\left(\left\langle f_n\right\rangle|\bar{f}_n,\beta^{-1}\right) p\left(\bar{\mathbf{f}}|\mathbf{X},\boldsymbol{\theta}\right).$$

[1] The Kullback-Leibler divergence between two distributions is defined as $\text{KL}\left(q\left(x\right)||p\left(x\right)\right) = \int q\left(x\right)\log\frac{q(x)}{p(x)}dx$.

This implies that the difference between the KL-corrected bound and the traditional variational bound is the Kullback-Leibler divergence between $q\left(\bar{\mathbf{f}}\right)$ and $p\left(\bar{\mathbf{f}}|\left\langle \mathbf{f}\right\rangle, \mathbf{X}, \boldsymbol{\theta}\right)$. Inspection of (5) shows that this divergence is zero after updates of $q\left(\bar{\mathbf{f}}\right)$. However, as $\boldsymbol{\theta}$ changes the divergence will become non-zero and provide a positive correction to the standard variational bound in the manner depicted in Figure 1(b). The KL-corrected objective is therefore a lower bound on the marginal likelihood and an upper bound on the traditional variational objective. Optimisations of the KL-corrected objective are therefore guaranteed to converge. In Section 4 we will show that empirically this convergence is much faster than that of the standard variational optimisation. Before that we will consider the KL-corrected bound in more detail.

3.2 Inference on the Corrected Bound

So far we have discussed optimising the KL-corrected bound primarily with respect to the parameters of the kernel function. Optimisation with respect to β is also straightforward. However, we have implicitly assumed that we will find $q\left(f_n\right)$ by optimising the original variational bound (which also entails optimisation of $q\left(\bar{\mathbf{f}}\right)$). In this section we consider the possibility of updating $q\left(f_n\right)$ through optimisation of the KL-corrected bound. To do this we first consider the dependence of the KL-corrected bound (9) on $q\left(f_n\right)$. First we make use of the fact that $p\left(f_n|\bar{f}_n, \beta\right) = N\left(f_n|\bar{f}_n, \beta^{-1}\right)$ to rewrite

$$\left\langle \log p\left(f_n|\bar{f}_n, \beta\right)\right\rangle = \log N\left(\left\langle f_n\right\rangle|\bar{f}_n, \beta^{-1}\right) - c_n$$

where c_n has the form $c_n = \frac{\beta}{2}\left(\left\langle f_n^2\right\rangle - \left\langle f_n\right\rangle^2\right)$. The integral in the first term of (9) can now be computed analytically by making use of the fact that $p\left(\bar{\mathbf{f}}|\mathbf{X}, \theta\right) = N\left(\bar{\mathbf{f}}|\mathbf{0}, \mathbf{K}\right)$ and

$$\log \int \prod_{n=1}^{N} N\left(\left\langle f_n\right\rangle|\bar{f}_n, \beta^{-1}\right) N\left(\bar{\mathbf{f}}|\mathbf{0}, \mathbf{K}\right) d\bar{\mathbf{f}} = \log N\left(\left\langle \mathbf{f}\right\rangle|\mathbf{0}, \left(\mathbf{K} + \beta^{-1}\mathbf{I}\right)\right).$$

We are interested in the dependence of this term on a particular $q\left(f_n\right)$. This can be obtained by factorising the distibution,

$$p\left(\left\langle \mathbf{f}\right\rangle\right) = p\left(\left\langle f_n\right\rangle|\left\langle \mathbf{f}_{\backslash n}\right\rangle\right) p\left(\left\langle \mathbf{f}_{\backslash n}\right\rangle\right),$$

where only the first term of this factorisation is dependent on $q\left(f_n\right)$. This conditional distribution has the form of a Gaussian,

$$p\left(\left\langle f_n\right\rangle|\left\langle \mathbf{f}_{\backslash n}\right\rangle\right) = N\left(\left\langle f_n\right\rangle|\mu_n, \sigma_n^2\right),$$

with mean $\mu_n = \mathbf{k}_{\backslash n}^{\mathrm{T}}\left(\mathbf{K}_{\backslash n} + \beta^{-1}\mathbf{I}\right)^{-1}\left\langle \mathbf{f}_{\backslash n}\right\rangle$ and variance

$$\sigma_n^2 = \beta^{-1} + k_{nn} - \mathbf{k}_{\backslash n}^{\mathrm{T}}\left(\mathbf{K}_{\backslash n} + \beta^{-1}\mathbf{I}\right)^{-1}\mathbf{k}_{\backslash n},$$

where $\mathbf{k}_{\setminus n}$ is the nth column of the covariance matrix with the nth element removed, $\mathbf{K}_{\setminus n}$ is the covariance matrix with the nth row and column removed and $\mathbf{f}_{\setminus n}$ is the vector $\mathbf{f}$ with the nth element removed. The terms of (9) which are dependent on $q\left(f_n\right)$ are then given by

$$\begin{aligned}\mathcal{L}'_n\left(\boldsymbol{\theta}\right) = &-\left\langle \log N\left(f_n|\mu_n,\sigma_n^2\right)\right\rangle + \left\langle \log p\left(y_n|f_n\right)\right\rangle + \left\langle \log q\left(f_n\right)\right\rangle_{q(f_n)} \\ &+\delta_n - \frac{1}{2}\log 2\pi\sigma_n^2\end{aligned}$$

where

$$\delta_n = \frac{1}{2}\left(\beta - \frac{1}{\sigma_n^2}\right)\left(\left\langle f_n^2\right\rangle - \left\langle f_n\right\rangle^2\right). \tag{11}$$

Now if we assume that δ_n is relatively insensitive to changes in $q\left(f_n\right)$ then we can optimise the KL-corrected bound with respect to $q\left(f_n\right)$ to obtain

$$q\left(f_n\right) \propto p\left(y_n|f_n\right) N\left(f_n|\mu_n,\sigma_n^2\right) \tag{12}$$

where we recall that from our definitions $N\left(f_n|\mu_n,\sigma_n^2\right)$ is the prediction at the nth point having removed the nth point from our data set. In the statistical physics literature this is known as a *cavity* distribution. Such cavity distributions are reminiscent of TAP approximations and the expectation propagation algorithm [8,7]. Of course, there is in general no guarantee that δ_n will be insensitive to changes in $q\left(f_n\right)$, however it is still possible to make use of this update in place of the variational updates (of $q\left(\bar{\mathbf{f}}\right)$ and $q\left(f_n\right)$) but it may be prudent to check that the KL-corrected bound is higher than that generated by the standard variational updates after updating $q\left(f_n\right)$. In the experiments in Section 4 we chose to always make use of the standard update so that we knew that any resulting increase in convergence speed was entirely due to optimisation of (9) with respect to the parameters $\boldsymbol{\theta}$.

3.3 Monitoring Convergence

Convergence of the algorithm can be monitored through evaluation of (9). However, this bound contains an expectation of $\log p\left(y_n|f_n\right)$ under the noise model which will typically require quadrature to compute. However, if we only compute the bound after updating $q\left(f_n\right)$ then we find (9) may be replaced by

$$\mathcal{L}'_c\left(\boldsymbol{\theta}\right) = \log N\left(\left\langle \mathbf{f}\right\rangle|\mathbf{0},\mathbf{K}+\beta^{-1}\mathbf{I}\right) - \sum_{n=1}^{N}\log N\left(\left\langle f_n\right\rangle|\left\langle \bar{f}_n\right\rangle,\beta^{-1}\right) + \sum_{n=1}^{N}\log Z_n$$

which will only require quadrature if Z_n requires quadrature (see Section 2.1).

4 Results

We performed a series of classification experiments with benchmark data sets to evaluate the performance of the PPA algorithm. For comparison we also include published results from the support vector machine on these data sets. In all our experiments we ordered updates as specified in Algorithm 1. Code for recreating our results is available on-line, for details see Appendix A.

Algorithm 1. Optimisation of the Gaussian Process with PPA. Note that algorithmically it is still necessary to update $q\left(\bar{\mathbf{f}}\right)$ for both variational and KL-corrected approaches as it is a pre-requisite for computation of each $q\left(f_n\right)$.

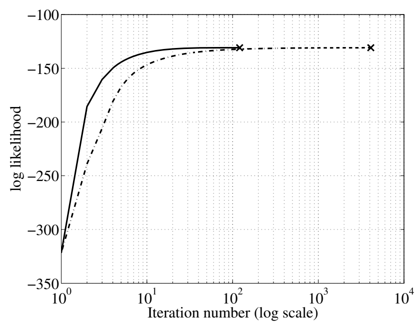
```
Inputs $\mathbf{X} = [\mathbf{x}_1, \ldots \mathbf{x}_N]^{\mathrm{T}}$ and outputs $\mathbf{y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^{\mathrm{T}}$, a convergence tolerance,
initial values for $\beta$ and $\boldsymbol{\theta}$.

E-Step — Iterate over the $q$-distributions
 Update $\langle\bar{\mathbf{f}}\rangle$ and $\langle\bar{\mathbf{f}}\bar{\mathbf{f}}^{\mathrm{T}}\rangle$.
 Calculate $\mathbf{g}$ and $\Gamma$ based on the given noise model.
 $n = 1 : N$
 Update $\langle f_n\rangle$ and $\langle f_n^2\rangle$
 *
M-Step — Update the parameters
 Use gradient based optimisation for updating $\boldsymbol{\theta}$.
 For standard variational approach optimise (3), for KL-corrected approach optimise
(9).
 Update $\beta$
 bound on likelihood changes by less than the convergence tolerance.
```

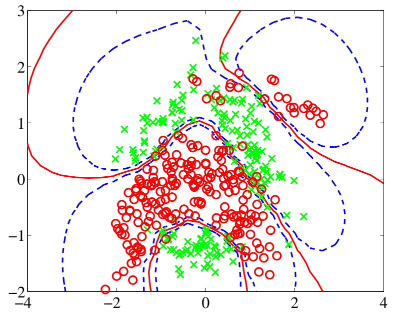
4.1 Convergence Speed

We first considered a synthetic data set **banana** [9]. This data set consists of two dimensional inputs sampled from Gaussian distributions. One hundred training/test partitions of the data are provided. We used the first partition to illustrate the improvements in training speed gained by using the KL-corrected objective function instead of the standard variational lower bound. The results are shown in Figure 2 (a). They show almost two orders of magnitude improvement in convergence in terms of iterations. These figures carry over into improvement in terms of timing as well. The final learnt decision boundary is shown in Figure 2 (b).

As well as the synthetic set, **banana**, we tested the algorithm using seven other data sets from the **UCI**, **DELVE** and **STATLOG** benchmark repositories with partitions provided by [9]. To allow the classification error comparisons to be accurate, we mimicked the experimental setup found in [9] as far as possible. Each data set is presented as a binary classification problem and partitioned into 100 different training and test data sets. In [9] kernel parameters were chosen through running 5-fold cross validation on the first five realisations of each data set. The median of the parameters was then chosen. In PPA the marginal likelihood can be maximised to obtain the kernel parameters. Therefore, for these methods, no cross validation was used. We simply maximised the lower bound on the marginal likelihood for the first five data sets. The kernel parameters associated with the



(a)

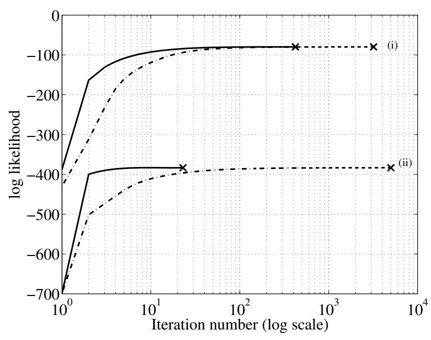


(b)

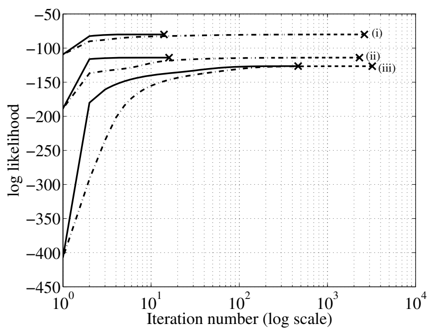
Fig. 2. (a) Plot of log-likelihood vs iteration number (log-scale) for the KL-corrected objective function (solid line) and the standard variational bound (dashed line). KL-corrected requires 120 iterations for convergence while the standard variational approach requires 4102 iterations. The point of convergence for each line is marked on the plot with a cross. Note that both approaches converge to the same likelihood. (b) The resulting classification of the **banana** data set. Decision boundaries are given by solid lines, the dashed lines indicate contours at 0.25 and 0.75 probabilities.

median RBF kernel width were then used for all data sets to compute the final results.

In Figure 3 we provide convergence plots for several of the data sets. Convergence plots and CPU timings were generated using the first partition of each data set. The final classification error is provided in Table 1 (a). Also included in this table for interest are the classification results reported by [9] for the SVM. The total time for convergence is given in Table 1 (b).



(a)



(b)

Fig. 3. (a) Convergence plot for (i) **twonorm** data set and (ii) **German** data set. (b) convergence plot for (i) **titanic** data set, (ii) **breast-cancer** data set and (iii) **waveform** data set. Each plot shows the bound on the likelihood vs iteration number for the KL-corrected objective function (solid line) and the standard variational bound (dashed line).

Table 1. (a) shows the classification error results of experiments with benchmark data sets compared to published results. The SVM results are taken from [9]. (b) displays CPU time comparisons for the experiments with benchmark data sets. Timings are given for the standard variational approach (Std) and the KL-corrected approach (Klc). The increase in speed is summarised by the speed up factor. Average speed up was 25.6.

Dataset	SVM	GP-PPA
Banana	11.5 ± 0.7	10.9 ± 0.5
B. Cancer	26.0 ± 4.7	29.4 ± 5.0
Diabetes	23.5 ± 1.7	23.0 ± 2.0
German	23.6 ± 2.1	23.9 ± 2.0
Heart	16.0 ± 3.3	17 ± 3.0
Titanic	22.4 ± 1.0	23.2 ± 0.3
Twonorm	3.0 ± 0.2	2.8 ± 0.3
Waveform	9.9 ± 0.4	11.9 ± 0.4

(a)

Dataset	Time Std $/10^3$s	Time Klc $/10^3$s	Speed Up Factor
Banana	22.5	1.13	19.9
B. Cancer	4.10	0.187	21.9
Diabetes	34.2	3.92	8.75
German	111	1.08	103
Heart	2.77	0.153	18.1
Titanic	1.94	0.0919	21.1
Twonorm	30.0	4.67	6.42
Wavenorm	36.3	6.16	5.89

(b)

The experimental results show that a Gaussian Process with variational inference through PPA has broadly similar performance to the support vector machine (as we might expect).

5 Discussion

We have presented a correction to the standard variational bound in the context of Gaussian process models. The KL-corrected bound leads to an much improved speed up for variational learning, without losing the guarantee of convergence. In experiments on benchmark data, the bound lead to a speed increase for all our experiments. The lowest speed up was 5.89 times faster whilst the highest was 103 times faster.

There is potential for KL-correction to be applied in other models and not just when Gaussian likelihoods and priors are used. We discussed how updates with respect to the marginal variational approximations, $q(f_n)$, could also be done to optimise the KL-corrected bound, but we leave exploration of these updates and a study of the general conditions for which KL-correction can be applied to further work.

References

1. Girolami, M., Rogers, S.: Variational bayesian multinomial probit regression with gaussian process priors. Neural Computation **18**(8) (2006) 1790–1817
2. King, N.J., Lawrence, N.D.: Variational inference in Gaussian processes via probabilistic point assimilation. Technical Report CS-05-06, The University of Sheffield, Department of Computer Science (2005)

3. O'Hagan, A.: Some Bayesian numerical analysis. In Bernardo, J.M., Berger, J.O., Dawid, A.P., Smith, A.F.M., eds.: Bayesian Statistics 4, Valencia, Oxford University Press (1992) 345–363
4. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press, Cambridge, MA (2006)
5. Waterhouse, S., MacKay, D.J.C., Robinson, T.: Bayesian methods for mixtures of experts. In Touretzky, D., Mozer, M., Hasselmo, M., eds.: Advances in Neural Information Processing Systems. Volume 8., Cambridge, MA, MIT Press (1996) 351–357
6. Seeger, M.: Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers. In Solla, S.A., Leen, T.K., Müller, K.R., eds.: Advances in Neural Information Processing Systems. Volume 12., Cambridge, MA, MIT Press (2000) 603–609
7. Minka, T.P.: A family of algorithms for approximate Bayesian inference. PhD thesis, Massachusetts Institute of Technology (2001)
8. Opper, M., Winther, O.: Gaussian processes for classification: Mean field algorithms. Neural Computation **12** (2000) 2655–2684
9. Rätsch, G., Onoda, T., Müller, K.R.: Soft margins for adaboost. Machine Learning **42**(3) (2001) 287–320

A On-Line Source Code

The source code for re-running all the experiments detailed here is available online from `http://www.dcs.shef.ac.uk/~neil/ppa/`.

Bandit Based Monte-Carlo Planning

Levente Kocsis and Csaba Szepesvári

Computer and Automation Research Institute of the
Hungarian Academy of Sciences, Kende u. 13-17, 1111 Budapest, Hungary
kocsis@sztaki.hu

Abstract. For large state-space Markovian Decision Problems Monte-Carlo planning is one of the few viable approaches to find near-optimal solutions. In this paper we introduce a new algorithm, UCT, that applies bandit ideas to guide Monte-Carlo planning. In finite-horizon or discounted MDPs the algorithm is shown to be consistent and finite sample bounds are derived on the estimation error due to sampling. Experimental results show that in several domains, UCT is significantly more efficient than its alternatives.

1 Introduction

Consider the problem of finding a near optimal action in large state-space Markovian Decision Problems (MDPs) under the assumption a generative model of the MDP is available. One of the few viable approaches is to carry out sampling based lookahead search, as proposed by Kearns et al. [8], whose sparse lookahead search procedure builds a tree with its nodes labelled by either states or state-action pairs in an alternating manner, and the root corresponding to the initial state from where planning is initiated. Each node labelled by a state is followed in the tree by a fixed number of nodes associated with the actions available at that state, whilst each corresponding state-action labelled node is followed by a fixed number of state-labelled nodes sampled using the generative model of the MDP. During sampling, the sampled rewards are stored with the edges connecting state-action nodes and state nodes. The tree is built in a stage-wise manner, from the root to the leafs. Its depth is fixed. The computation of the values of the actions at the initial state happens from the leafs by propagating the values up in the tree: The value of a state-action labelled node is computed based on the average of the sum of the rewards along the edges originating at the node and the values at the corresponding successor nodes, whilst the value of a state node is computed by taking the maximum of the values of its children. Kearns et al. showed that in order to find an action at the initial state whose value is within the ϵ-vicinity of that of the best, for discounted MPDs with discount factor $0 < \gamma < 1$, K actions and uniformly bounded rewards, regardless of the size of the state-space fixed size trees suffice [8]. In particular, the depth of the tree is proportional to $1/(1-\gamma)\log(1/(\epsilon(1-\gamma)))$, whilst its width is proportional to $K/(\epsilon(1-\gamma))$.

Although this result looks promising,[1] in practice, the amount of work needed to compute just a single almost-optimal action at a given state can be overwhelmingly large. In this paper we are interested in improving the performance of this vanilla Monte-Carlo planning algorithm. In particular, we are interested in Monte-Carlo planning algorithms with two important characteristics: (1) small error probability if the algorithm is stopped prematurely, and (2) convergence to the best action if enough time is given.

Besides MPDs, we are also interested in game-tree search. Over the years, Monte-Carlo simulation based search algorithms have been used successfully in many non-deterministic and imperfect information games, including backgammon [13], poker [4] and Scrabble [11]. Recently, Monte-Carlo search proved to be competitive in deterministic games with large branching factors, viz. in Go [5]. For real-time strategy games, due to their enormous branching factors and stochasticity, Monte-Carlo simulations seems to be one of the few feasible approaches for planning [7]. Intriguingly, Monte-Carlo search algorithms used by today's games programs use either uniform sampling of actions or some heuristic biasing of the action selection probabilities that come with no guarantees.

The main idea of the algorithm proposed in this paper is to sample actions selectively. In order to motivate our approach let us consider problems with a large number of actions and assume that the lookahead is carried out at a fixed depth D. If sampling can be restricted to say half of the actions at all stages then the overall work reduction is $(1/2)^D$. Hence, if one is able to identify a large subset of the suboptimal actions early in the sampling procedure then huge performance improvements can be expected.

By definition, an action is suboptimal for a given state, if its value is less than the best of the action-values for the same state. Since action-values depend on the values of successor states, the problem boils down to getting the estimation error of the state-values for such states decay fast. In order to achieve this, an efficient algorithm must balance between testing alternatives that look currently the best so as to obtain precise estimates, and the exploration of currently suboptimal-looking alternatives, so as to ensure that no good alternatives are missed because of early estimation errors. Obviously, these criteria are contradictory and the problem of finding the right balance is known as the the exploration-exploitation dilemma. The most basic form of this dilemma shows up in *multi-armed bandit problems* [1].

The main idea in this paper it to apply a particular bandit algorithm, UCB1 (UCB stands for Upper Confidence Bounds), for rollout-based Monte-Carlo planning. The new algorithm, called UCT (UCB applied to trees) described in Section 2 is called UCT. Theoretical results show that the new algorithm is consistent, whilst experimental results (Section 3) for artificial game domains (P-games) and the sailing domain (a specific MDP) studied earlier in a similar context by others [10] indicate that UCT has a significant performance advantage over its closest competitors.

[1] In fact, as also noted by [8] the bound might be unimprovable, though this still remains an open problem.

2 The UCT Algorithm

2.1 Rollout-Based Planning

In this paper we consider Monte-Carlo planning algorithms that we call *rollout-based*. As opposed to the algorithm described in the introduction (stage-wise tree building), a rollout-based algorithm builds its lookahead tree by repeatedly sampling episodes from the initial state. An episode is a sequence of state-action-reward triplets that are obtained using the domains generative model. The tree is built by adding the information gathered during an episode to it in an incremental manner.

The reason that we consider rollout-based algorithms is that they allow us to keep track of estimates of the actions' values at the sampled states encountered in earlier episodes. Hence, if some state is reencountered then the estimated action-values can be used to bias the choice of what action to follow, potentially speeding up the convergence of the value estimates. If the portion of states that are encountered multiple times in the procedure is small then the performance of rollout-based sampling degenerates to that of vanilla (non-selective) Monte-Carlo planning. On the other hand, for domains where the set of successor states concentrates to a few states only, rollout-based algorithms implementing selective sampling might have an advantage over other methods.

The generic scheme of rollout-based Monte-Carlo planning is given in Figure 1. The algorithm iteratively generates episodes (line 3), and returns the action with the highest average observed long-term reward (line 5).[2] In procedure UpdateValue the total reward q is used to adjust the estimated value for the given state-action pair at the given depth, completed by increasing the counter that stores the number of visits of the state-action pair at the given depth. Episodes are generated by the *search* function that selects and effectuates actions recursively until some terminal condition is satisfied. This can be the reach of a terminal state, or episodes can be cut at a certain depth (line 8). Alternatively, as suggested by Peret and Garcia [10] and motivated by iterative deepening, the search can be implemented in phases where in each phase the depth of search is increased. An approximate way to implement iterative deepening, that we also follow in our experiments, is to stop the episodes with probability that is inversely proportional to the number of visits to the state.

The effectiveness of the whole algorithm will crucially depend on how the actions are selected in line 9. In vanilla Monte-Carlo planning (referred by MC in the following) the actions are sampled uniformly. The main contribution of the present paper is the introduction of a bandit-algorithm for the implementation of the selective sampling of actions.

2.2 Stochastic Bandit Problems and UCB1

A bandit problem with K arms (actions) is defined by the sequence of random payoffs X_{it}, $i = 1, \ldots, K$, $t \geq 1$, where each i is the index of a gambling machine

[2] The function *bestMove* is trivial, and is omitted due to the lack of space.

```
function MonteCarloPlanning($state$)
repeat
    search($state$, 0)
until Timeout
return bestAction($state$,0)

function search($state$, $depth$)
if Terminal($state$) then return 0
if Leaf($state, d$) then return Evaluate($state$)
$action$ := selectAction($state$, $depth$)
($nextstate$, $reward$) := simulateAction($state$, $action$)
$q$ := $reward$ + $\gamma$ search($nextstate$, $depth$ + 1)
UpdateValue($state, action, q, depth$)
return $q$
```

Fig. 1. The pseudocode of a generic Monte-Carlo planning algorithm

(the "arm" of a bandit). Successive plays of machine i yield the payoffs X_{i1}, X_{i2}, ... For simplicity, we shall assume that X_{it} lies in the interval $[0,1]$. An allocation policy is a mapping that selects the next arm to be played based on the sequence of past selections and payoffs obtained. The expected regret of an allocation policy A after n plays is defined by $R_n = \max_i \mathbb{E}\left[\sum_{t=1}^{n} X_{it}\right] - \mathbb{E}\left[\sum_{j=1}^{K}\sum_{t=1}^{T_j(n)} X_{j,t}\right]$, where $I_t \in \{1,\ldots,K\}$ is the index of the arm selected at time t by policy A, and where $T_i(t) = \sum_{s=1}^{t} \mathbb{I}(I_s = i)$ is the number of times arm i was played up to time t (including t). Thus, the regret is the loss caused by the policy not always playing the best machine. For a large class of payoff distributions, there is no policy whose regret would grow slower than $O(\ln n)$ [9]. For such payoff distributions, a policy is said to resolve the exploration-exploitation tradeoff if its regret growth rate is within a constant factor of the best possible regret rate.

Algorithm UCB1, whose finite-time regret is studied in details by [1] is a simple, yet attractive algorithm that succeeds in resolving the exploration-exploitation tradeoff. It keeps track the average rewards $\overline{X}_{i,T_i(t-1)}$ for all the arms and chooses the arm with the best upper confidence bound:

$$I_t = \underset{i\in\{1,\ldots,K\}}{\operatorname{argmax}} \left\{\overline{X}_{i,T_i(t-1)} + c_{t-1,T_i(t-1)}\right\}, \tag{1}$$

where $c_{t,s}$ is a bias sequence chosen to be

$$c_{t,s} = \sqrt{\frac{2\ln t}{s}}. \tag{2}$$

The bias sequence is such that if X_{it} were independently and identically distributed then the inequalities

$$\mathbb{P}\left(\overline{X}_{is} \geq \mu_i + c_{t,s}\right) \leq t^{-4}, \tag{3}$$

$$\mathbb{P}\left(\overline{X}_{is} \leq \mu_i - c_{t,s}\right) \leq t^{-4} \tag{4}$$

were satisfied. This follows from Hoeffding's inequality. In our case, UCB1 is used in the internal nodes to select the actions to be sampled next. Since for any given node, the sampling probability of the actions at nodes below the node (in the tree) is changing, the payoff sequences experienced will drift in time. Hence, in UCT, the above expression for the bias terms $c_{t,s}$ needs to replaced by a term that takes into account this drift of payoffs. One of our main results will show despite this drift, bias terms of the form $c_{t,s} = 2C_p\sqrt{\frac{\ln t}{s}}$ with appropriate constants C_p can still be constructed for the payoff sequences experienced at any of the internal nodes such that the above tail inequalities are still satisfied.

2.3 The Proposed Algorithm

In UCT the action selection problem as treated as a separate multi-armed bandit for every (explored) internal node. The arms correspond to actions and the payoffs to the cumulated (discounted) rewards of the paths originating at the node.s In particular, in state s, at depth d, the action that maximises $Q_t(s, a, d) + c_{N_{s,d}(t),N_{s,a,d}(t)}$ is selected, where $Q_t(s, a, d)$ is the estimated value of action a in state s at depth d and time t, $N_{s,d}(t)$ is the number of times state s has been visited up to time t at depth d and $N_{s,a,d}(t)$ is the number of times action a was selected when state s has been visited, up to time t at depth d.[3]

2.4 Theoretical Analysis

The analysis is broken down to first analysing UCB1 for non-stationary bandit problems where the payoff sequences might drift, and then showing that the payoff sequences experienced at the internal nodes of the tree satisfy the drift-conditions (see below) and finally proving the consistency of the whole procedure.

The so-called drift-conditions that we make on the non-stationary payoff sequences are as follows: For simplicity, we assume that $0 \le X_{it} \le 1$. We assume that the expected values of the averages $\overline{X}_{in} = \frac{1}{n}\sum_{t=1}^{n} X_{it}$ converge. We let $\mu_{in} = \mathbb{E}\left[\overline{X}_{in}\right]$ and $\mu_i = \lim_{n\to\infty} \mu_{in}$. Further, we define δ_{in} by $\mu_{in} = \mu_i + \delta_{in}$and assume that the tail inequalities (3),(4) are satisfied for $c_{t,s} = 2C_p\sqrt{\frac{\ln t}{s}}$ with an appropriate constant $C_p > 0$. Throughout the analysis of non-stationary bandit problems we shall always assume without explicitly stating it that these drift-conditions are satisfied for the payoff sequences.

For the sake of simplicity we assume that there exist a single optimal action.[4] Quantities related to this optimal arm shall be upper indexed by a star, e.g., μ^*, $T^*(t)$, $\overline{X}_t^*$, etc. Due to the lack of space the proofs of most of the results are omitted.

We let $\Delta_i = \mu^* - \mu_i$. We assume that C_p is such that there exist an integer N_0 such that for $s \ge N_0$, $c_{ss} \ge 2|\delta_{is}|$ for any suboptimal arm i. Clearly, when UCT is applied in a tree then at the leafs $\delta_{is} = 0$ and this condition is automatically satisfied with $N_0 = 1$. For upper levels, we will argue by induction by showing an upper bound δ_{ts} for the lower levels that C_p can be selected to make $N_0 < +\infty$.

Our first result is a generalisation of Theorem 1 due to Auer et al. [1]. The proof closely follows this earlier proof.

[3] The algorithm has to be implemented such that division by zero is avoided.

[4] The generalisation of the results to the case of multiple optimal arms follow easily.

Theorem 1. *Consider UCB1 applied to a non-stationary problem. Let $T_i(n)$ denote the number of plays of arm i. Then if i the index of a suboptimal arm, $n > K$, then $\mathbb{E}\left[T_i(n)\right] \le \frac{16C_p^2 \ln n}{(\Delta_i/2)^2} + 2N_0 + \frac{\pi^2}{3}$.*

At those internal nodes of the lookahead tree that are labelled by some state, the state values are estimated by averaging all the (cumulative) payoffs for the episodes starting from that node. These values are then used in calculating the value of the action leading to the given state. Hence, the rate of convergence of the bias of the estimated state-values will influence the rate of convergence of values further up in the tree. The next result, building on Theorem 1, gives a bound on this bias.

Theorem 2. *Let $\overline{X}_n = \sum_{i=1}^{K} \frac{T_i(n)}{n} \overline{X}_{i,T_i(n)}$. Then*

$$\left|\mathbb{E}\left[\overline{X}_n\right] - \mu^*\right| \le |\delta_n^*| + O\left(\frac{K(C_p^2 \ln n + N_0)}{n}\right), \tag{5}$$

UCB1 never stops exploring. This allows us to derive that the average rewards at internal nodes concentrate quickly around their means. The following theorem shows that the number of times an arm is pulled can actually be lower bounded by the logarithm of the total number of trials:

Theorem 3 (Lower Bound). *There exists some positive constant ρ such that for all arms i and n, $T_i(n) \ge \lceil \rho \log(n) \rceil$.*

Among the the drift-conditions that we made on the payoff process was that the average payoffs concentrate around their mean quickly. The following result shows that this property is kept intact and in particular, this result completes the proof that if payoff processes further down in the tree satisfy the drift conditions then payoff processes higher in the tree will satisfy the drift conditions, too:

Theorem 4. *Fix $\delta > 0$ and let $\Delta_n = 9\sqrt{2n \ln(2/\delta)}$. The following bounds hold true provided that n is sufficiently large: $\mathbb{P}\left(n\overline{X}_n \ge n\mathbb{E}\left[\overline{X}_n\right] + \Delta_n\right) \le \delta$, $\mathbb{P}\left(n\overline{X}_n \le n\mathbb{E}\left[\overline{X}_n\right] - \Delta_n\right) \le \delta$.*

Finally, as we will be interested in the failure probability of the algorithm at the root, we prove the following result:

Theorem 5 (Convergence of Failure Probability). *Let $\hat{I}_t = \operatorname{argmax}_i \overline{X}_{i,T_i(t)}$. Then $P(\hat{I}_t \ne i^*) \le C\left(\frac{1}{t}\right)^{\frac{\rho}{2}\left(\frac{\min_{i \ne i^*} \Delta_i}{36}\right)^2}$ with some constant C. In particular, it holds that $\lim_{t\to\infty} P(\hat{I}_t \ne i^*) = 0$.*

Now follows our main result:

Theorem 6. *Consider a finite-horizon MDP with rewards scaled to lie in the $[0, 1]$ interval. Let the horizon of the MDP be D, and the number of actions per state be K. Consider algorithm UCT such that the bias terms of UCB1 are multiplied by D. Then the bias of the estimated expected payoff, $\overline{X}_n$, is $O(\log(n)/n)$. Further, the failure probability at the root converges to zero at a polynomial rate as the number of episodes grows to infinity.*

Proof. (Sketch) The proof is done by induction on D. For $D = 1$ UCT just corresponds to UCB1. Since the tail conditions are satisfied with $C_p = 1/\sqrt{2}$ by Hoeffding's inequality, the result follows from Theorems 2 and 5.

Now, assume that the result holds for all trees of up to depth $D-1$ and consider a tree of depth D. First, divide all rewards by D, hence all the cumulative rewards are kept in the interval $[0, 1]$. Consider the root node. The result follows by Theorems 2 and 5 provided that we show that UCT generates a non-stationary payoff sequence at the root satisfying the drift-conditions. Since by our induction hypothesis this holds for all nodes at distance one from the root, the proof is finished by observing that Theorem 2 and 4 do indeed ensure that the drift conditions are satisfied. The particular rate of convergence of the bias is obtained by some straightforward algebra.

By a simple argument, this result can be extended to discounted MDPs. Instead of giving the formal result, we note that if some desired accuracy, ϵ_0 is fixed, similarly to [8] we may cut the search at the effective ϵ_0-horizon to derive the convergence of the action values at the initial state to the ϵ_0-vicinity of their true values. Then, similarly to [8], given some $\epsilon > 0$, by choosing ϵ_0 small enough, we may actually let the procedure select an ϵ-optimal action by sampling a sufficiently large number of episodes (the actual bound is similar to that obtained in [8]).

3 Experiments

3.1 Experiments with Random Game Trees

A P-game tree [12] is a minimax tree that is meant to model games where at the end of the game the winner is decided by a global evaluation of the board position where some counting method is employed (examples of such games include Go, Amazons and Clobber). Accordingly, rewards are only associated with transitions to terminal states. These rewards are computed by first assigning values to moves (the moves are deterministic) and summing up the values along the path to the terminal state.[5] If the sum is positive, the result is a win for MAX, if it is negative the result is a win for MIN, whilst it is draw if the sum is 0. In the experiments, for the moves of MAX the move value was chosen uniformly from the interval $[0, 127]$ and for MIN from the interval $[-127, 0]$.[6] We have performed experiments for measuring the convergence rate of the algorithm.[7]

First, we compared the performance of four search algorithms: alpha-beta (AB), plain Monte-Carlo planning (MC), Monte-Carlo planning with minimax value update (MMMC), and the UCT algorithm. The failure rates of the four algorithms are plotted as function of iterations in Figure 2. Figure 2, left corresponds to trees with branching factor (B) two and depth (D) twenty, and

[5] Note that the move values are not available to the player during the game.

[6] This is different from [12], where 1 and -1 was used only.

[7] Note that for P-games UCT is modified to a negamax-style: In MIN nodes the negative of estimated action-values is used in the action selection procedures.

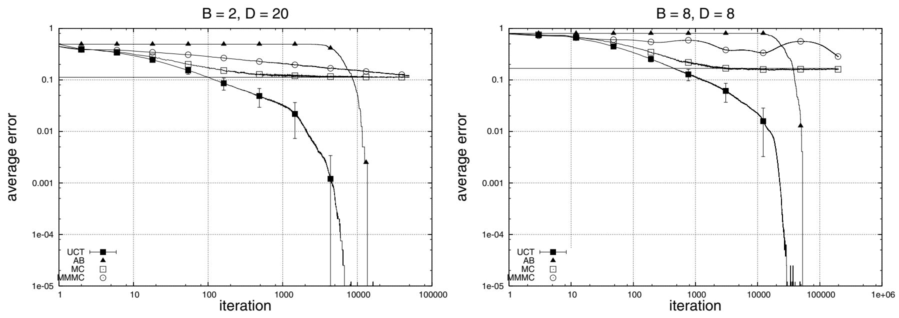


Fig. 2. Failure rate in P-games. The 95% confidence intervals are also shown for UCT.

Figure 2, right to trees with branching factor eight and depth eight. The failure rate represents the frequency of choosing the incorrect move if stopped after a number of iterations. For alpha-beta it is assumed that it would pick a move randomly, if the search has not been completed within a number of leaf nodes.[8] Each data point is averaged over 200 trees, and 200 runs for each tree. We observe that for both tree shapes UCT is converging to the correct move (i.e. zero failure rate) within a similar number of leaf nodes as alpha-beta does. Moreover, if we accept some small failure rate, UCT may even be faster. As expected, MC is converging to failure rate levels that are significant, and it is outperformed by UCT even for smaller searches. We remark that failure rate for MMCS is higher than for MC, although MMMC would eventually converge to the correct move if run for enough iterations.

Second, we measured the convergence rate of UCT as a function of search depth and branching factor. The required number of iterations to obtain failure rate smaller than some fixed value is plotted in Figure 3. We observe that for P-game trees UCT is converging to the correct move in order of $B^{D/2}$ number of iterations (the curve is roughly parallel to $B^{D/2}$ on log-log scale), similarly to alpha-beta. For higher failure rates, UCT seems to converge faster than $o(B^{D/2})$.

Note that, as remarked in the discussion at the end of Section 2.4, due to the faster convergence of values for deterministic problems, it is natural to decay the bias sequence with distance from the root (depth). Accordingly, in the experiments presented in this section the bias $c_{t,s}$ used in UCB was modified to $c_{t,s} = (\ln t/s)^{(D+d)/(2D+d)}$, where D is the estimated game length starting from the node, and d is the depth of the node in the tree.

3.2 Experiments with MDPs

The sailing domain [10,14] is a finite state- and action-space stochastic shortest path problem (SSP), where a sailboat has to find the shortest path between two

[8] We also tested choosing the best looking action based on the incomplete searches. It turns out, not unexpectedly, that this choice does not influence the results.

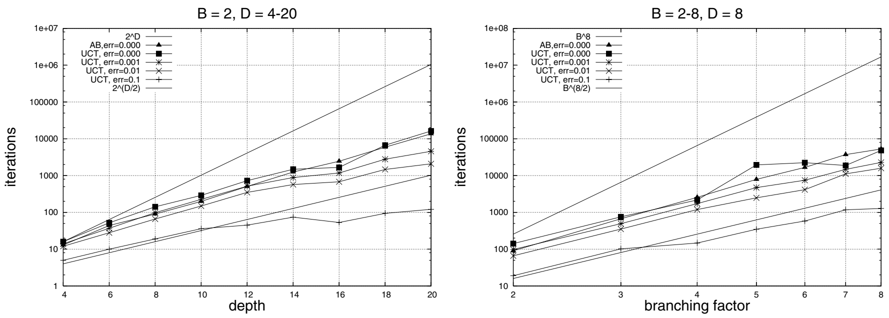


Fig. 3. P-game experiment: Number of episodes as a function of failure probability

points of a grid under fluctuating wind conditions. This domain is particularly interesting as at present SSPs lie outside of the scope of our theoretical results.

The details of the problem are as follows: the sailboat's position is represented as a pair of coordinates on a grid of finite size. The controller has 7 actions available in each state, giving the direction to a neighbouring grid position. Each action has a cost in the range of 1 and 8.6, depending on the direction of the action and the wind: The action whose direction is just the opposite of the direction of the wind is forbidden. Following [10], in order to avoid issues related to the choice of the evaluation function, we construct an evaluation function by randomly perturbing the optimal evaluation function that is computed off-line by value-iteration. The form of the perturbation is $\hat{V}(x) = (1 + \epsilon(x))V^*(x)$, where x is a state, $\epsilon(x)$ is a uniform random variable drawn in $[-0.1; 0.1]$ and $V^*(x)$ is the optimal value function. The assignment of specific evaluation values to states is fixed for a particular run. The performance of a stochastic policy is evaluated by the error term $Q^*(s, a) - V^*(s)$, where a is the action suggested by the policy in state s and Q^* gives the optimal value of action a in state s. The error is averaged over a set of 1000 randomly chosen states.

Three planning algorithms are tested: UCT, ARTDP [3], and PG-ID [10]. For UCT, the algorithm described in Section 2.1 is used. The episodes are stopped with probability $1/N_s(t)$, and the bias is multiplied (heuristically) by 10 (this multiplier should be an upper bound on the total reward).[9] For ARTDP the evaluation function is used for initialising the state-values. Since these values are expected to be closer to the true optimal values, this can be expected to speed up convergence. This was indeed observed in our experiments (not shown here). Moreover, we found that Boltzmann-exploration gives the best performance with ARTDP and thus it is used in this experiment (the 'temperature' parameter is kept at a fixed value, tuned on small-size problems). For PG-ID the same parameter setting is used as [10].

[9] We have experimented with alternative stopping schemes. No major differences were found in the performance of the algorithm for the different schemes. Hence these results are not presented here.

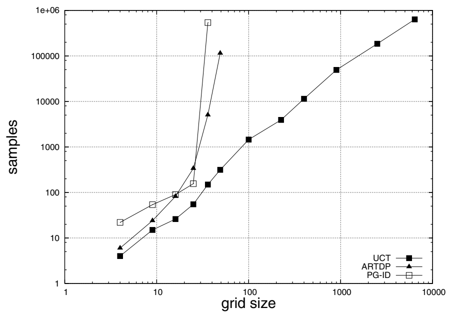


Fig. 4. Number of samples needed to achieve an error of size 0.1 in the sailing domain. 'Problem size' means the size of the grid underlying the state-space. The size of the state-space is thus 24×'problem size', since the wind may blow from 8 directions, and 3 additional values (per state) give the 'tack'.

Since, the investigated algorithms are building rather non-uniform search trees, we compare them by the total number of samples used (this is equal to the number of calls to the simulator). The required number of samples to obtain error smaller than 0.1 for grid sizes varying from 2×2 to 40×40 is plotted in Figure 4. We observe that UCT requires significantly less samples to achieve the same error than ARTDP and PG-ID. At least on this domain, we conclude that UCT scales better with the problem size than the other algorithms.

3.3 Related Research

Besides the research already mentioned on Monte-Carlo search in games and the work of [8], we believe that the closest to our work is the work of [10] who also proposed to use rollout-based Monte-Carlo planning in undiscounted MDPs with selective action sampling. They compared three strategies: uniform sampling (uncontrolled search), Boltzmann-exploration based search (the values of actions are transformed into a probability distribution, i.e., samples better looking actions are sampled more often) and a heuristic, interval-estimation based approach. They observed that in the 'sailing' domain lookahead pathologies are present when the search is uncontrolled. Experimentally, both the interval-estimation and the Boltzmann-exploration based strategies were shown to avoid the lookahead pathology and to improve upon the basic procedure by a large margin. We note that Boltzmann-exploration is another widely used bandit strategy, known under the name of "exponentially weighted average forecaster" in the on-line prediction literature (e.g. [2]). Boltzmann-exploration as a bandit strategy is inferior to UCB in stochastic environments (its regret grows with the square root of the number of samples), but is preferable in adversary environments where UCB does not have regret guarantees. We have also experimented with a Boltzmann-exploration based strategy and found that in the case of our domains it performs significantly weaker than the upper-confidence value based algorithm described here.

Recently, Chang et al. also considered the problem of selective sampling in finite horizon undiscounted MDPs [6]. However, since they considered domains where there is little hope that the same states will be encountered multiple times, their algorithm samples the tree in a depth-first, recursive manner: At each node they sample (recursively) a sufficient number of samples to compute a good approximation of the value of the node. The subroutine returns with an approximate evaluation of the value of the node, but the returned values are not stored (so when a node is revisited, no information is present about which actions can be expected to perform better). Similar to our proposal, they suggest to propagate the average values upwards in the tree and sampling is controlled by upper-confidence bounds. They prove results similar to ours, though, due to the independence of samples the analysis of their algorithm is significantly easier. They also experimented with propagating the maximum of the values of the children and a number of combinations. These combinations outperformed propagating the maximum value. When states are not likely to be encountered multiple times, our algorithm degrades to this algorithm. On the other hand, when a significant portion of states (close to the initial state) can be expected to be encountered multiple times then we can expect our algorithm to perform significantly better.

4 Conclusions

In this article we introduced a new Monte-Carlo planning algorithm, UCT, that applies the bandit algorithm UCB1 for guiding selective sampling of actions in rollout-based planning. Theoretical results were presented showing that the new algorithm is consistent in the sense that the probability of selecting the optimal action can be made to converge to 1 as the number of samples grows to infinity.

The performance of UCT was tested experimentally in two synthetic domains, viz. random (P-game) trees and in a stochastic shortest path problem (sailing). In the P-game experiments we have found that the empirically that the convergence rates of UCT is of order $B^{D/2}$, same as for alpha-beta search for the trees investigated. In the sailing domain we observed that UCT requires significantly less samples to achieve the same error level than ARTDP or PG-ID, which in turn allowed UCT to solve much larger problems than what was possible with the other two algorithms.

Future theoretical work should include analysing UCT in stochastic shortest path problems and taking into account the effect of randomised terminating condition in the analysis. We also plan to use UCT as the core search procedure of some real-world game programs.

Acknowledgements

We would like to acknowledge support for this project from the Hungarian National Science Foundation (OTKA), Grant No. T047193 (Cs. Szepesvári) and from the Hungarian Academy of Sciences (Cs. Szepesvári, Bolyai Fellowship).

References

1. P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.
2. P. Auer, N. Cesa-Bianchi, Y. Freund, and R.E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32:48–77, 2002.
3. A.G. Barto, S.J. Bradtke, and S.P. Singh. Real-time learning and control using asynchronous dynamic programming. Technical report 91-57, Computer Science Department, University of Massachusetts, 1991.
4. D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The challenge of poker. *Artificial Intelligence*, 134:201–240, 2002.
5. B. Bouzy and B. Helmstetter. Monte Carlo Go developments. In H.J. van den Herik, H. Iida, and E.A. Heinz, editors, *Advances in Computer Games 10*, pages 159–174, 2004.
6. H.S. Chang, M. Fu, J. Hu, and S.I. Marcus. An adaptive sampling algorithm for solving Markov decision processes. *Operations Research*, 53(1):126–139, 2005.
7. M. Chung, M. Buro, and J. Schaeffer. Monte Carlo planning in RTS games. In *CIG 2005, Colchester, UK*, 2005.
8. M. Kearns, Y. Mansour, and A.Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markovian decisi on processes. In *Proceedings of IJ-CAI'99*, pages 1324–1331, 1999.
9. T.L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.
10. L. Péret and F. Garcia. On-line search for solving Markov decision processes via heuristic sampling. In R.L. de Mántaras and L. Saitta, editors, *ECAI*, pages 530–534, 2004.
11. B. Sheppard. World-championship-caliber Scrabble. *Artificial Intelligence*, 134(1–2):241–275, 2002.
12. S.J.J. Smith and D.S. Nau. An analysis of forward pruning. In *AAAI*, pages 1386–1391, 1994.
13. G. Tesauro and G.R. Galperin. On-line policy improvement using Monte-Carlo search. In M.C. Mozer, M.I. Jordan, and T. Petsche, editors, *NIPS 9*, pages 1068–1074, 1997.
14. R. Vanderbei. Optimal sailing strategies, statistics and operations research program. University of Princeton, http://www.sor.princeton.edu/r̃vdb/sail/sail.html., 1996.

Bayesian Learning with Mixtures of Trees

Jussi Kollin and Mikko Koivisto

HIIT Basic Research Unit, Department of Computer Science
Gustaf Hällströmin katu 2b, FI-00014 University of Helsinki, Finland
jussi.kollin@cs.helsinki.fi, mikko.koivisto@cs.helsinki.fi

Abstract. We present a Bayesian method for learning mixtures of graphical models. In particular, we focus on data clustering with a tree-structured model for each cluster. We use a Markov chain Monte Carlo method to draw a sample of clusterings, while the likelihood of a clustering is computed by exact averaging over the model class, including the dependency structure on the variables. Experiments on synthetic data show that this method usually outperforms the expectation–maximization algorithm by Meilă and Jordan [1] when the number of observations is small (hundreds) and the number of variables is large (dozens). We apply the method to study how much single nucleotide polymorphisms carry information about the structure of human populations.

1 Introduction

Mixture models provide a flexible way to learn regularities from data. By mixing simple component models one can obtain a significantly more complex model. A finite mixture can be interpreted as a semi-parametric model, with applications in density estimation and related activities. Another interpretation treats the labels of the mixture components, one per observation, as unobserved data. This latent variable formulation supports, e.g., unsupervised clustering.

Meilă and Jordan [1] present an expectation–maximization (EM) algorithm to maximum likelihood estimation of mixtures of tree-structured graphical models; remarkably, the method also estimates the tree structure. A shortcoming of this and similar maximum likelihood methods is, however, that only a single estimate of the mixture model parameters is provided, leaving the uncertainty around the estimate as poorly characterized. While bootstrapping [2] can be used for finding approximate confidence intervals, it is computationally demanding and requires a large data set. Moreover, bootstrap proportions can be hard to interpret (see, e.g., [3]) and nuisance parameters hard to handle.

Bayesian inference avoids many of the above shortcomings as it concerns the posterior distribution of the quantities of interest. Thanks to the recent developments in Markov chain Monte Carlo (MCMC) methods, Bayesian inference has become computationally feasible, at least for some important model classes, such as mixtures of multivariate Gaussians [4]. Early MCMC methods relied on Gibbs sampling that, analogously to the EM algorithm, draws samples from the joint space of model parameters and the latent component labels. Later works (e.g., [5]) have turned to the more general Metropolis–Hastings algorithm that avoids sampling of the latent labels and is thus arguably more efficient [6]. Despite this progress, the current techniques are insufficient when the component models are more complex and involve lots of parameters [5]. This is the case, for example, when the number of variables is moderate (dozens) or large (hundreds), and when the dependency structure among the variables is unknown.

In this paper, we describe a full Bayesian method for learning with mixtures of graphical models. Our approach is motivated by two simple observations. First, for some important classes of graphical models, such as trees, there exist efficient algorithms for structure learning. Also, the space of data clusterings is discrete and "relatively small," provided that the number of data points is moderate. Consequently, it is reasonable to run MCMC over the space of clusterings, the prior and the (marginal) likelihood of a clustering being relatively cheap to evaluate. We call this approach *hidden data sampling* (HDS).

Related previous works have focused on simple models of independent or multivariate Gaussians. Neal [7] introduces a Markov chain sampling method for mixtures of Dirichlet processes. Rasmussen [8] describes a similar method for infinite Gaussian mixtures. Dawson and Belkhir [9], followed by Corander et al. [10], adopt the approach to a clustering problem in population genetics under a simple model where all attributes (genetic markers) are conditionally independent given the clustering (partition into subpopulations). We are not aware of any implementations under more complex models, for which the benefits of the approach should be more substantial. We stress that, unlike Gibbs sampling, HDS should not be viewed as a dual of EM; in Gibbs sampling and EM one only needs means for handling the conditional distribution of the hidden data $\mathbf{z}$ given the model parameters θ (either for sampling $\mathbf{z}$ or for computing suitable expectations), whereas HDS requires that one is able to compute the marginal probability of any given $\mathbf{z}$, obtained by integrating θ out.

We apply the HDS method for data clustering via mixtures of trees. We extend the mixture model of Meilă and Jordan [1] to a full Bayesian model and handle each mixture component using the algorithm of Meilă and Jaakkola [11]. We describe the building blocks in Sect. 2–4. In Sect. 5 we report experimental results on synthetic data, with a comparison to an EM algorithm. In Sect. 6 we apply the method to study how much Single Nucleotide Polymorphisms (SNPs) carry information about human subpopulations [12].

2 Bayesian Networks and Trees

A Bayesian network (BN) over a vector of variables $x = (x_1, \ldots, x_n)$ specifies a probability distribution of x. The *network structure* of a BN encodes conditional independence assertions among the variables via a directed acyclic graph. We represent this graph as a vector $G = (G_1, \ldots, G_n)$ where each G_v is a subset of the index set $V = \{1, \ldots, n\}$ and specifies the parents of v in the graph. We may index with subsets: if $S = \{u_1, \ldots, u_s\}$ with $u_1 < \cdots < u_s$, then x_S denotes the vector $(x_{u_1}, \ldots, x_{u_s})$. Along the structure G, a BN factorizes the probability distribution of x into a product of *local conditional distributions*. Usually these conditional distributions belong to some parametric family, parametrized by θ, and it is convenient to write the probability distribution of x as

$$p(x|G,\theta) = \prod_{v=1}^{n} p(x_v|x_{G_v}, G, \theta) \, . \tag{1}$$

Our notation supports the Bayesian treatment of the network structure G and the parameters θ as random variables (whenever their values are unknown).

A *Bayesian tree* (BT) is a Bayesian network where the network structure is a directed tree, i.e., one variable (the root) has no parents and every other variable has exactly one parent. BTs form an attractive subclass of BNs, as many important computational problems can be efficiently solved for trees but not for unconstrained BNs. We will soon come back to this issue.

BNs can be used to model multiple vectors $x[1], \ldots, x[m]$, called *data* and denoted by $\mathbf{x}$. When the vectors are judged to be exchangeable, the probability distribution of the data given the structure G, is expressed as

$$p(\mathbf{x}|G) = \int \Big(\prod_{j=1}^{m} p(x[j]|G,\theta) \Big) p(\theta|G) \mathrm{d}\theta \, ,$$

where $p(\theta|G)$ is a parameter prior, and each $p(x[j]|G,\theta)$ factorizes as in (1). When the network structure is unknown, a prior distribution $p(G)$ is introduced and the marginal distribution of data can be written as

$$p(\mathbf{x}) = \sum_{G} p(G) p(\mathbf{x}|G) \, , \tag{2}$$

where G runs through all possible network structures.

To best exploit the structural nature of BNs, usually the joint prior over structures and parameters is factorized by $p(G) \propto \prod_{v=1}^{n} \rho_v(G_v)$ and $p(\theta|G) = \prod_{v=1}^{n} p(\theta_{v,G_v}|G_v)$, where each ρ_v is a nonnegative function and θ_{v,G_v} is a set of parameters that fully specify the conditional distribution of x_v given x_{G_v}. If a prior satisfies these conditions we call it *modular* or *decomposable* [13,14,15]. Sometimes it also reasonable to force the prior to be *likelihood equivalent* [13,14,15], i.e., if two structures G and G' represent the same assertions of conditional independence, then $p(G|\mathbf{x})$ and $p(G'|\mathbf{x})$ must be equal (for all data sets).

In this paper we consider the common setting of discrete variables with independent local multinomial distributions. We let the multinomial parameters be independent and follow a Dirichlet distribution such that both modularity and likelihood equivalence hold [13,11].

A modular model structure makes computations easier, albeit not always feasible. In general, it is hard to find a network structure that maximizes the posterior $p(G|\mathbf{x})$ (see, e.g., [13]), and we suspect that it is not easier to evaluate the probability of the data, $p(\mathbf{x})$. Recently, Koivisto and Sood [15] present an algorithm that solves these problems in time that scales as $n2^n$. This algorithm is practical for small instances, up to about $n = 25$ variables.

On the contrary, for Bayesian trees several key tasks, such as maximum likelihood estimation and Bayesian inference, are computationally feasible even for large numbers of variables [16,13,1]. Meilă and Jaakkola [11] show the important result that also the marginal probability $p(\mathbf{x})$ for given data $\mathbf{x}$ can be computed in time cubic in n. The result applies whenever the structure prior is modular and symmetric, $\rho_v(u) = \rho_u(v) \geq 0$ for each edge (u, v), and the parameters have a Dirichlet prior with the parameters (pseudo counts) $N'_{uv}(st) > 0$, where s and t refer to the states of the variables x_u and x_v, respectively; the counts $N'_{uv}(st)$ are subject to certain marginal constraints to ensure likelihood equivalence, see [17, Theorem 5] for details. The algorithm arises from an extension to the matrix tree theorem by Kirchhoff (in 1848); it computes the determinant of an $(n-1) \times (n-1)$ matrix that, in essence, represents for each pair of variables (excluding an arbitrarily chosen root) the marginal probability of the data on these variables. These probabilities are fully determined by the pairwise *sufficient statistics*, counts of occurrences for each value combination of two variables.

Unfortunately, the involved determinants tend to be ill-conditioned: to evaluate the determinant accurately one should let the precision of intermediate results grow linearly in the number of data points [18]. Although this costs "only" about a linear factor in the time complexity [19], the algorithm seems to be practical for data sets with at most some thousands of records.

3 Mixture Models and Clusterings

Sometimes an inhomogeneous population can be well modeled by a relatively small number of homogeneous subpopulations. Finite mixture models embody this idea by forming the distribution of x as a convex combination of a fixed number of other distributions, usually members of some parametric family. A mixture model with k components is parametrized by mixture proportions $\alpha = (\alpha_1, \ldots, \alpha_k)$ which sum up to unity, and component-wise parameters $\beta = (\beta_1, \ldots, \beta_k)$, each β_c specifying the cth component distribution. Thus,

$$p(x|\alpha, \beta) = \alpha_1 f_1(x; \beta_1) + \cdots + \alpha_k f_k(x; \beta_k)\,,$$

where each $f_c(x; \beta_c)$ belongs to a family of parametric distributions of x, which may be different for different components c. It is convenient to interpret a component as an unobserved variable z that takes values in $\{1, \ldots, k\}$. Augmenting our probability model p to z we can write $\alpha_c = p(z = c|\alpha)$. Similarly we can replace $f_c(x; \beta_c)$ by $p(x|\beta, z = c)$.

We now focus on mixtures of Bayesian networks and, in particular, of Bayesian trees. Thus, each β_c specifies a network structure, G_c, and the parameters of the local conditional distributions, θ_c. We can read $p(x|\beta, z = c)$ as $p(x|G_c, \theta_c)$, which we expressed in (1).

The intimate relationship of mixture components and data clusters is obvious: for the jth data point, the unobserved label $z[j]$ defines the cluster to which the data point belongs. Accordingly, the sets $\mathcal{C}_c = \{j : z[j] = c\}$, for $c = 1, \ldots, k$, form a partition of the indices of the data points. To illustrate this relationship, we first write the distribution of the data $\mathbf{x}$ as

$$p(\mathbf{x}) = \int\int \Big(\prod_{j=1}^{m} p(x[j]|\alpha, \beta)\Big) p(\alpha)p(\beta)\, \mathrm{d}\alpha\, \mathrm{d}\beta\,,$$

where we assume that α and β are independent a priori.[1] Then we give an alternative expression as a sum over clusterings:

$$p(\mathbf{x}) = \sum_{\mathcal{C}} p(\mathcal{C}_1, \ldots, \mathcal{C}_k)p(\mathbf{x}|\mathcal{C}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})\,,$$

where $\mathbf{z}$ consists of the labels $z[j]$ and is one to one with $\mathcal{C} = (\mathcal{C}_1, \ldots, \mathcal{C}_k)$, and where the two terms in the latter sum have the following expressions. Assuming the prior of α is Dirichlet$(m'_1, \ldots, m'_k)$, we have

$$p(\mathbf{z}) = \frac{\Gamma(m')}{\Gamma(m+m')} \prod_{c=1}^{k} \frac{\Gamma(m_c + m'_c)}{\Gamma(m'_c)}\,, \tag{3}$$

where $m_c = |\mathcal{C}_c|$ and $m' = m'_1 + \cdots + m'_k$. The second term can be written as

$$p(\mathbf{x}|\mathbf{z}) = \prod_{c=1}^{k} \Big[\int \Big(\prod_{j \in \mathcal{C}_c} p(x[j]|\beta_c, z[j] = c) \Big) p(\beta_c)\, \mathrm{d}\beta_c \Big] = \prod_{c=1}^{k} p(x[\mathcal{C}_c])\,, \tag{4}$$

where $p(x[\mathcal{C}_c])$ obeys (2), with $x[\mathcal{C}_c]$ denoting the data points in the cth cluster.

We note that for a given clustering, the former term (3) is easy to evaluate, while computing the latter term (4) is feasible for general BNs on a small number of variables and for BTs on up to some hundreds of variables. This fact motivates our Markov chain sampler for the posterior distribution $p(\mathcal{C}_1, \ldots, \mathcal{C}_k|\mathbf{x}) = p(\mathbf{z}|\mathbf{x})$, which we describe in the next section.[2]

4 MCMC over Clusterings

We apply standard simulation techniques to draw a sample of clusterings along a Markov chain whose stationary distribution is the posterior distribution $p(\mathbf{z}|\mathbf{x})$. Posterior inference is then implemented via Monte Carlo averages.

4.1 Sampling Along a Markov Chain

We use the Metropolis–Hastings algorithm [20] with a simple proposal distribution as given as Algorithm 1. The algorithm is given as input a random initial clustering $\mathbf{z}^{(0)}$; in our experiments we used a very simple procedure that generates k nonempty clusters of sizes $m_1, \ldots, m_k \geq 1$ with a probability proportional to the product $m_1 m_2 \cdots m_k$. The algorithm does not sample any tree structures or parameters. Instead we apply the results of Meilă and Jaakkola [11,17] to calculate the exact average over all BTs as in (4) given a clustering.

[1] We slightly abuse the notation $\mathrm{d}\beta$, as β contains also the network structure.

[2] We also remark that the prior $p(\mathbf{z})$ does not have to take the form (3) derived from the mixture model, but any prior, e.g., the uniform distribution, could be used.

Algorithm 1. Metropolis–Hastings algorithm for sampling clusterings

```
Input: Data set $\mathbf{x}$, cluster count $k$, initial clustering $\mathbf{z}^{(0)}$, and for all nodes $u$ and $v$
    edge priors $\rho_u(v)$ and pseudo counts $N'_{uv}(st)$ (described in Sect. 2 and [11]).
Output: Sample of clusterings.
1: for iteration $i = 1, \ldots, T$ do
2:     Draw uniformly at random a data point $x[j]$ which is in a cluster of size $\geq 2$.
3:     Generate candidate clustering $\mathbf{z}'$ from $\mathbf{z}^{(i-1)}$ by moving $x[j]$ to another cluster,
       drawn uniformly at random.
4:     Evaluate $p(\mathbf{x}|\mathbf{z}') = \prod_{c=1}^{k} p(x\,[\mathcal{C}'_c])$ by computing the factors corresponding to the
       two changed clusters using the algorithm of Meilă and Jaakkola [17, Eq. 31].
5:     Let $\eta(\mathbf{z})$ be the number of singleton clusters in $\mathbf{z}$ and $p(\mathbf{z})$ (see Sect. 3). Let
       $$A \leftarrow \frac{p(\mathbf{z}')p(\mathbf{x}|\mathbf{z}')}{p\left(\mathbf{z}^{(i)}\right)p\left(\mathbf{x}|\mathbf{z}^{(i)}\right)} \times \frac{m - \eta\left(\mathbf{z}^{(i)}\right)}{m - \eta(\mathbf{z}')}$$
6:     With probability $\min\{1, A\}$, let $\mathbf{z}^{(i)} \leftarrow \mathbf{z}'$; otherwise let $\mathbf{z}^{(i)} \leftarrow \mathbf{z}^{(i-1)}$.
7: return $\left(\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(T)}\right)$
```

An advantage of this simple proposal distribution is that at each iteration only two data clusters are changed: a single data point is either removed or added. Thus, the $O(n^2)$ sufficient statistics needed for likelihood evaluation in tree structured models can be updated in $O(n^2)$ time, instead of the $O(mn^2)$ time needed for the initialization. In summary, the time complexity of Algorithm 1 is $O(mn^2 + (k + T)n^3)$ where T is the number of iterations and n^3 is the time required to evaluate the marginal probability of a single data cluster, given the sufficient statistics.

4.2 Inference Using Monte Carlo

The posterior expectation of any function $\phi(\mathbf{z})$ can be estimated by the arithmetic mean of the values $\phi(\mathbf{z}^{(i)})$ at the sampled clusterings. For example, we may estimate the probability that two data points j and j' belong to the same cluster by taking average of the indicator function $\mathrm{I}(z[j] = z[j'])$. While the posterior probabilities $p(z[j] = c|\mathbf{x})$ are rarely interesting,[3] the pairwise counterparts, $p(z[j] = z[j']|\mathbf{x})$, provide useful summaries, especially when the interest is in discovering data clusters; e.g., Dawson and Belkhir [9] apply a hierarchical clustering algorithm using the "co-assignment probabilities" as a measure of similarity.

We emphasize that one may also infer quantities that are indirect functions of clusterings. For example, we can estimate the posterior probability that all the k trees include a particular edge e. To do this, we need to notice two facts. First, the events "e belongs to the cth tree" are mutually independent given

[3] For every j and c, the probability $p(z[j] = c|\mathbf{x})$ must equal $1/k$, assuming that the prior is symmetric over the k components.

Algorithm 2. An algorithm for generating a synthetic data set

```
Input: Number of components $k$, variables $n$ and data points $m$.
Output: A synthetic data set.
1: Draw $k$ mixture proportions from the uniform distribution on the simplex.
2: for cluster $c = 1, \dots, k$ do
3:     Draw a linear order of the variables uniformly at random.
4:     For each of the $n$ variables select a parent from its predecessors uniformly at
       random (except for the root, i.e., the first variable in the order).
5:     For each multinomial distribution, generate the parameters from the uniform
       distribution on the simplex.
6: return Set of $m$ data points over $n$ variables drawn from the mixture of $k$ BTs.
```

the clustering. Second, for each cluster we can efficiently compute the posterior probability that the corresponding tree includes the edge e [11]. Consequently, the probability that e is included in every component is easy to compute for a given clustering. Taking a Monte Carlo average of these probabilities for the sampled clusterings finally yields the desired marginal posterior probability.

5 Experimental Results on Synthetic Data

We demonstrate the presented method in data clustering with mixtures of trees.

5.1 The Clustering Problem

We consider the following clustering problem. Given a set of data points, $x[1], \dots, x[m]$, and a number k, the task is to assign a cluster label $z[j]$ from $\{1, \dots, k\}$ to each data point $x[j]$ such that the global assignment, or clustering, $\mathbf{z}$ is as good as possible. We assume that there exists a unique correct clustering $\mathbf{z}^*$ and that the goodness of $\mathbf{z}$ is defined w.r.t. $\mathbf{z}^*$ via some discrepancy measure between two clusterings. Here we consider the *Jaccard index* [21], the ratio a/b, where a is the number of pairs of data points that appear in the same cluster in *both* clusterings, and b is the number of pairs of data points that appear in the same cluster in *one or both* clusterings. (We have also examined other measures, including the *variation of information* (VI) metric [22]. The results are qualitatively very similar and therefore not reported here.)

5.2 The Tested Methods and Data Sets

We consider three scenarios of (n, m, k) for the number of variables n, data points m, and clusters k: A $= (10, 100, 2)$, B $= (20, 200, 4)$, C $= (50, 500, 10)$. For each scenario we generated 50 random data sets with two-state variables, and another 50 random data sets with four-state variables, as shown in Algorithm 2. Note that the described sampling distribution of trees is far from uniform: the flatter the tree, the larger the probability, chains being the least probable.

In our experiments we used the following configuration of the proposed method. We set the prior over tree structures to the uniform distribution. Note that this distribution differs from the one we used in generating the test data sets. To the multinomial parameters and mixture proportions we assigned the uniform priors over the corresponding simplexes. In the determinant computations we used the standard double precision arithmetic, which is sufficient for our particular setup where every cluster typically contains less than 100 data points. As the posterior guess about the unobserved clustering we used the clustering that was visited by the simulated Markov chain and has the largest posterior probability. [4] We refer to this method as BMT (Bayesian mixture of trees). We assigned uniform prior over tree structures, i.e., equal $\rho_u(v)$ for each pair of nodes (u, v). The parameter prior was set by $N'_{uv}(st) \equiv 1$; see Sect. 2 and Meilă and Jordan [17] for the interpretation of the parameters.

We compared BMT to our implementation of the MIXTREE algorithm [1].[5] This EM algorithm finds k Bayesian trees along the mixture proportions so as to (locally) maximize the likelihood. We examined two initialization methods: either we generate a random mixture model, as done by Meilă and Jordan [1], or we draw a random soft clustering assignment, i.e., independent membership distributions for each data point; in the discussion below we will focus on the latter initialization method, for it produced slightly better results. Based on preliminary experiments we decided to stop each EM restart when the relative increase in the log-likelihood between the last two steps remains below the threshold 10^{-9} for two consecutive iterations; results for different stopping criteria were qualitatively very similar. Finally, a maximum likelihood clustering is found by assigning each data point to the most probable cluster, given the estimated mixture model and the data point. We refer to this method as MT (mixture of trees). It should be noted that we did not implement any of the smoothing tricks proposed by Meilă and Jordan [1], as they seem to lack a principled interpretation. Meilă and Jordan [1] report good results for MIXTREE in density estimation and classification; however, they do not consider the clustering task.

Both methods, BMT and MT, were given roughly the same amount of running time per data set. On each data set from scenarios A, B, and C, we allowed the algorithms run for 1, 2, and 5 hours, respectively. Given this amount of time, BMT typically completed millions of iterations, while MT completed thousands of random restarts of the EM algorithm.

5.3 Analysis of Clustering Results

The clustering results for BMT and MT are summarized in Fig. 1. On the smallest data sets (scenario A), BMT and MT perform about equally well on average.

[4] Alternatively, one could think of returning a clustering that maximizes the expected Jaccard index. However, such a clustering might be suboptimal w.r.t. other measures of clustering accuracy.

[5] We have implemented the algorithms in the C++ language. The experiments were run on several PCs, each having a 3.0 GHz processor.

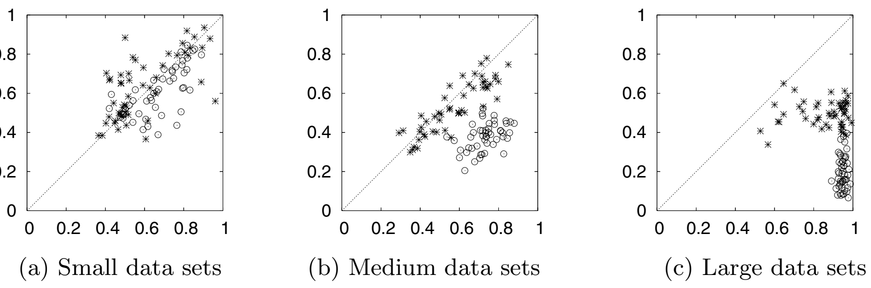


Fig. 1. Clustering error measured by the Jaccard index, for BMT (x-axis) and MT (y-axis). On scenarios A, B, and C (left to right) results are shown for 100 random data sets. The number of states per variable is two ($\circledast$) or four ($\odot$).

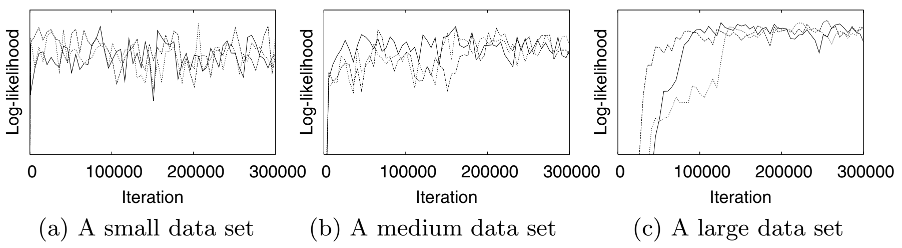


Fig. 2. Traces of the log-likelihood for three independent runs per scenario. Each picture shows every 5,000th sample from the first 300,000 iterations of three runs per case.

However, MT is slightly more accurate when the variables are binary-valued, whereas BMT is superior to MT when each variable can take four values. On the largest data sets (scenario C), BMT clearly outperforms MT. As expected, the difference in the performance grows with the number of states per variable.

We could explain these results theoretically as follows. When there are only 10 binary variables MT does not suffer much from using a point estimate (a mixture of BTs). On the contrary, BMT uses a somewhat biased prior, which renders its performance suboptimal. For larger numbers of variables the point estimates used by MT are no longer reliable—model averaging, as carried out by BMT, starts paying off. In practice, however, we realized that on larger data sets MT usually did not find the global optimum, despite the fair number of EM restarts. This may be the dominating reason for the relatively poor performance of MT. On the contrary, stochastic local search in the space of clusterings, as implemented in BMT, seems to be sufficient for finding plausible clusterings.

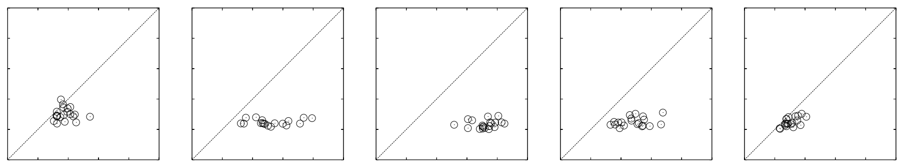
We visually inspected the mixing properties of BMT by running several independent chains from random initial states, for a few data sets selected at random. As shown in Fig. 2, mixing is rapid on the small data set, and, as expected, gets slower on larger data sets. Yet, the needed burn-in period is short compared to the total number of iterations for these data sets.

6 Application to SNP-Haplotype Clustering

Geographically distant human subpopulations also tend to be genetically distant, relative to the variation within subpopulations. This is because after separation from a common founder population (a long time ago), subpopulations have evolved quite independently under population genetic forces, such as mutation, recombination, random drift, and selection. The variation within and between subpopulations can be observed at marker locations, especially, at Single Nucleotide Polymorphisms (SNPs). A SNP is a one-base location of the DNA where two different variants (alleles) appear in the population. Recently, large efforts have been put into typing (reading) millions of SNPs over the whole genome for groups of ethnically diverse individuals [12].

We apply the presented clustering method to study how many and how sparsely spaced SNPs are needed for inferring the known subpopulations solely from the SNP data. This question is interesting, since SNPs spanning a short region of the genome are not expected to be informative about the present subpopulations—rather, they reveal the content of the common, ancient founder population. We use the Perlegen data [12] that contains alleles at over 1.5 million SNPs for 24 European American samples, 23 African American samples, and 24 Han Chinese samples, each sample contributing two haplotypes to constitute a data set of 142 data points in total. Modeling with a mixture of trees makes a compromise with computational convenience and biological plausibility. Roughly speaking, we expect that the dependency of two markers gradually decays with their physical distance in the genome, suggesting a Markov chain model. However, several factors disturb this view, suggesting more complex models [23]. We use the BMT and MT methods with the details described in the previous section.

Figure 3 shows the results obtained at several randomly picked regions with varying number and spacing of the SNPs. We observe that the cluster structure of the data approaches the division into the three subpopulations as the number of SNPs and the average spacing get larger. For 100 SNPs with the average spacing of 500 kb, the subpopulations are fairly accurately discovered by BMT. (The maximum-likelihood method, MT, is inadequate for revealing this trend.)



(a) 100, 20 kb (b) 100, 100 kb (c) 100, 500 kb (d) 50, 500 kb (e) 20, 500 kb

Fig. 3. Deviation of the subpopulation division and inferred clusterings measured by the Jaccard index, for BMT (x-axis) and MT (y-axis). Results shown for 20 random regions per case: (a–c) 100 SNPs, the mean spacing of 20, 100, and 500 kb, and (c–e) 100, 50, and 20 SNPs, the mean spacing of 500 kb. The range of both axes is $[0, 1]$.

7 Concluding Remarks

We have applied the hidden data sampling (HDS) method for Bayesian learning with mixtures of graphical models. The HDS method is applicable whenever the likelihood of a single data cluster can be efficiently evaluated. We used Bayesian trees as an example of such a model class; Bayesian networks with a fixed variable order and bounded indegree is another example not considered here. Previous applications of essentially the same MCMC method have considered only simple Gaussian and product multinomial models [7,8,9,10]. These works also show (rather straightforward) ways to extend the models and methods to an unknown number of clusters; notice, however, that inferring the number of clusters is usually quite sensitive to the priors of the model parameters.

In our experiments on data clustering, the proposed Bayesian method, BMT, outperformed the maximum likelihood method implemented via the EM algorithm, MT (MIXTREE, [1]). The observed poor performance of the EM algorithm is partially due to its sensitivity to the initial model. Another, and perhaps more interesting, reason is that MT fits all model parameters, including the tree structure, to a relatively small number of data points. To overcome this drawback, Meilă and Jordan [1] propose semi-Bayesian smoothing techniques to be used for small data sets. Our results suggest that "smoothing" is good to implemented in its entirety, via full model averaging, like done in BMT.

The results on the genetic data agreed with our hypothesis about the relationship of SNP haplotypes and human subpopulations. We note that, in general, population structure can be more powerfully inferred using data on different kind of genetic markers (microsatellites) [9,10]. However, SNPs provide unique means for mapping disease predisposing genes [12]—it is useful to characterize what information SNPs carry about the genetic variation between individuals. Finally, we admit that trees are not perfect models for weakly correlated SNPs; in fact, independence models and, e.g., the k-means algorithm might perform equally well.

Acknowledgments

The authors would like to thank Heikki Mannila for useful discussions and Marina Meilă for providing some details of the MIXTREE algorithm.

References

1. Meilă, M., Jordan, M.I.: Learning with mixtures of trees. Journal of Machine Learning Research **1** (2000) 1–48
2. Efron, B.: Bootstrap methods: another look at the jackknife. Annals of Statistics **7** (1979) 1–26
3. Alfaro, M.E., Zoller, S., Lutzoni, F.: Bayes or bootstrap? A simulation study comparing the performance of Bayesian Markov chain Monte Carlo sampling and bootstrapping in assessing phylogenetic confidence. Molecular Biology and Evolution **20** (2003) 255–266

4. Diebolt, J., Robert, C.P.: Estimation of finite mixture distributions through Bayesian sampling. J. Royal Statistical Society B **56** (1994) 363–375
5. Jasra, A., Holmes, C.C., Stephens, D.A.: Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modelling. Statistical Science **20** (2005) 50–67
6. Celeux, G., Hurn, M., Robert, C.P.: Computational and inferential difficulties with mixture posterior distributions. J. Amer. Statist. Assoc. **95** (2000) 957–970
7. Neal, R.M.: Markov chain sampling method for Dirichlet process mixture models. Journal of Computational and Graphical Statistics **9** (2000) 249–265
8. Rasmussen, C.E.: The Infinite Gaussian Mixture Model. In Solla, S.A., Leen, T.K., Müller, K.R., eds.: NIPS 12, The MIT Press (2000) 554–560
9. Dawson, K.J., Belkhir, K.: A Bayesian approach to the identification of panmictic population and the assignment of individuals. Genetical Research **78** (2001) 59–77
10. Corander, J., Waldman, P., Sillanpää, M.J.: Bayesian analysis of genetic differentiation between populations. Genetics **163** (2003) 367–374
11. Meila, M., Jaakkola, T.: Tractable Bayesian learning of tree belief networks. In Boutilier, C., Goldszmidt, M., eds.: UAI, Morgan Kaufmann (2000) 380–388
12. Hinds, D.A., et al.: Whole-genome patterns of common DNA variation in three human populations. Science **307** (2005) 1072–1079
13. Heckerman, D., Geiger, D., Chickering, D.M.: Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning **20** (1995) 197–243
14. Friedman, N., Koller, D.: Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. Machine Learning **50** (2003) 95–125
15. Koivisto, M., Sood, K.: Exact Bayesian structure discovery in Bayesian networks. Journal of Machine Learning Research **5** (2004) 549–573
16. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory **14** (1968) 462–467
17. Meilă, M., Jaakkola, T.: Tractable Bayesian learning of tree belief networks. Technical Report CMU-RI-TR-00-15, Carnegie Mellon University Robotics Institute (2000)
18. Cerquides, J., de Mántaras, R.L.: TAN classifiers based on decomposable distributions. Machine Learning **59** (2005) 1–32
19. Kaltofen, E., Villard, G.: On the complexity of computing determinants. Computational Complexity **13** (2004) 91–130
20. Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika **57** (1970) 97–109
21. Jaccard, P.: The distribution of the flora in the Alpine zone. The New Phytologist **XI** (1912) 37–50
22. Meila, M.: Comparing clusterings by the variation of information. In Schölkopf, B., Warmuth, M.K., eds.: COLT. Volume 2777 of Lecture Notes in Computer Science., Springer (2003) 173–187
23. Thomas, A., Camp, N.J.: Graphical modeling of the joint distribution of alleles at associated loci. American Journal of Human Genetics **74** (2004) 1088–1101

Transductive Gaussian Process Regression with Automatic Model Selection

Quoc V. Le[1], Alex J. Smola[1], Thomas Gärtner[2], and Yasemin Altun[3]

[1] RSISE, Australian National University, 0200 ACT, Australia
Statistical Machine Learning Program, National ICT Australia, 0200 ACT, Australia
[2] Fraunhofer IAIS, Schloß Birlinghoven, 53754 Sankt Augustin, Germany
[3] Toyota Technological Institute at Chicago, Chicago IL 60632, USA

Abstract. In contrast to the standard *inductive inference* setting of predictive machine learning, in real world learning problems often the test instances are already available at training time. *Transductive inference* tries to improve the predictive accuracy of learning algorithms by making use of the information contained in these test instances. Although this description of transductive inference applies to predictive learning problems in general, most transductive approaches consider the case of classification only. In this paper we introduce a transductive variant of Gaussian process regression with automatic model selection, based on approximate moment matching between training and test data. Empirical results show the feasibility and competitiveness of this approach.

1 Introduction

Machine learning research mostly concentrates on estimating an underlying unknown conditional or functional dependence of a target property on some other variables. This estimate is based on a set of training instances, respecting this dependency. It is then usually applied to test instances for which the target property has not been observed. This setting is known as *supervised learning* or *inductive inference*. The downside of such algorithms is that they ignore the test data at training time even when such data is available. In this case, *transductive inference* approaches promise improved predictive accuracy as they exploit available knowledge about the test instances at training time. A related class of methods are *semi-supervised learning* algorithms that take advantage of additional unlabeled data which may or may not be used for testing purposes. See, e.g., [1] for an overview of recent results.

This work has led to a number of competitive algorithms mostly making use of the "cluster assumption", i.e., that "the decision boundary should not cross high-density regions", e.g., [2]. Although the transductive as well as the semi-supervised learning settings have no inherent restriction to classification only, there is so far very little work on transductive nor semi-supervised regression or structural prediction. Most work on transductive or semi-supervised regression is primarily concerned with designing kernel matrices such as the inverse graph Laplacian [3,4] or related matrices [5,6] on both labeled and unlabeled data. Similarly, Bayesian Committee Machines [7] can also be considered as a transductive method where the test data is incorporated in the computation of the kernel [8]. In [9] the labels for the test data are chosen to minimise the leave-one-out error of ridge regression on the joint training and test data and are constraint to be close to the inductive solution. In [10] the disagreement on unlablelled data between the hypotheses and an origin function is minimised and in [11] the disagreement on unlablelled data between hypotheses from different views is minimised.

In this paper we introduce a transductive algorithm for Gaussian process regression. The algorithm is based on the idea that the moments on training and test set, i.e., mean and variance, should match approximately. This is a realistic assumption in many real-world datasets and theoretically justified by the assumption of iid data and the observation that scalar quantities are much more concentrated around their mean than, say, the distribution of maximum a posteriori estimators. More precisely, let $\{y_1, \ldots, y_m\}$ denote the labels on the training data. We make sure that the observed mean and variance on the training set match the predicted mean and variance on the test set $\mathbf{E}[y]$ or $\mathbf{E}\left[y^2\right]$. In the present paper we achieve this by directly modifying the prior such that only parameters which are consistent between training and test set are considered in the inference procedure. The algorithm has the potential of being combined with previous approaches based on modifying the kernel or on minimising the disagreement between hypotheses.

In this fashion our setting draws on [12] which studies transductive classification based on a similar principle, namely that the predicted conditional class probabilities should match the observed counts on the training set. While we frame our approach in terms of a homoscedastic Gaussian Process estimator [13] it is readily extensible to heteroscedastic estimation [14], albeit at the expense of additional complication in the notation.

This paper is organized as follows: Section 2 introduces Gaussian process regression and model selection strategies for Gaussian processes. Our general approach to transductive Gaussian processes with automatic model selection is described in Section 3 and the optimization details are laid out in Section 4. Finally, Section 5 contains our empirical findings and Section 6 concludes.

2 Gaussian Process Regression

2.1 Setting

We begin with a very brief overview over Gaussian Process (GP) Regression, as described, e.g., in [15,13]. Denote by $\mathcal{X} \times \mathcal{Y}$ the domain of patterns and labels respectively from which m pairs (x_i, y_i) are drawn independently and identically distributed (iid). For regression assume that $\mathcal{Y} \subseteq \mathbb{R}$. Moreover assume that there exists a Gaussian process on $\mathcal{X}$ with covariance kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and mean $\mu : \mathcal{X} \to \mathbb{R}$. For notational convenience we set $\mu(x) = 0$, i.e., we ignore the offset for the remainder of the paper.

The key assumption of GP regression is that y is given by $y = t + \xi$ where $\xi \sim \mathcal{N}(0, \sigma^2)$ is an iid Gaussian random variable and that t is drawn from the Gaussian process on $\mathcal{X}$ specified by k. That is,

$$Y = (y_1, \ldots, y_m) \sim \mathcal{N}(0, K + \sigma^2 \mathbf{I})$$

where $K_{ij} = k(x_i, x_j)$ and $\mathbf{I}$ is the identity matrix.

2.2 Regression

Since $Y|X \sim \mathcal{N}(0, K + \sigma^2\mathbf{I})$ is normal, so is the conditional distribution of test labels given training and test data $p(Y_{\text{test}}|Y_{\text{train}}, X_{\text{train}}, X_{\text{test}})$. We have $Y_{\text{test}}|Y_{\text{train}}, X_{\text{train}}, X_{\text{test}} \sim \mathcal{N}(\mu, \Sigma)$ where

$$\mu = K_{\text{test,train}}(K_{\text{train,train}} + \sigma^2\mathbf{I})^{-1}Y_{\text{train}} \tag{1}$$

$$\Sigma = K_{\text{test,test}} + \sigma^2\mathbf{I} - K_{\text{test,train}}(K_{\text{train,train}} + \sigma^2\mathbf{I})^{-1}K_{\text{train,test}} \,. \tag{2}$$

Here $K_{\text{train,train}}$ is the covariance matrix computed on the training set $X_{\text{train}} = \{x_1, \ldots, x_m\}$, $K_{\text{test,test}}$ is the corresponding part computed on $X_{\text{test}} = \{x'_1, \ldots, x'_{m'}\}$, $K_{\text{test,train}}, K_{\text{train,test}}$ contain the cross-terms, and Y_{train} is the vector of training labels y_i. Eq. (2) contains the Schur complement arising from conditioning on a subset of random variables.

Note that the distribution of $Y_{\text{test}}|Y_{\text{train}}, X_{\text{train}}, X_{\text{test}}$ may differ significantly from the distribution of observed Y_{train}. In particular, there is no guarantee that any of the moments of the conditional distribution match that of the observed data. This is the key weakness of the model which we will address in Section 3.

2.3 Model Selection

If we knew the correct k and σ^2, Equations (1) and (2) would be all we need for inference. In reality, the kernel k and the degree of noise σ^2 need to be adjusted. By Bayes rule this leads to

$$p(Y_{\text{train}}, \sigma^2, k|X_{\text{train}}) \propto p(Y_{\text{train}}|X_{\text{train}}, \sigma^2, k)p(\sigma^2, k|X_{\text{train}}) \,.$$

Inference is then carried out either by sampling from the posterior or by maximum a posteriori (MAP) estimation with respect to (k, σ^2). For the purpose of this paper we focus on the latter due to its superior computational efficiency. Lacking further knowledge about the prior, one typically assumes that $p(\sigma^2, k|X_{\text{train}}) = p(\sigma^2)p(k)$ factorizes. Typically $p(k)$ is non-zero only for a parameterised family of kernels. For instance, the liberty in choosing k might relate to the width and scaling in a Gaussian RBF kernel, leading to parameter scaling in a fashion similar to Automatic Relevance Determination [16]. We then need the derivatives with respect to these parameters (see Section 4.4).

This leads to the minimization of the negative log-posterior $\mathcal{P}(\sigma^2, k) := - \log p(Y_{\text{train}}, \sigma^2, k|X_{\text{train}})$ which is given (up to constants) by

$$\mathcal{P}(\sigma^2, k) = \frac{1}{2}\log|K + \sigma^2\mathbf{I}| - \log p(\sigma^2) - \log p(k) + \frac{1}{2}y^\top(K + \sigma^2\mathbf{I})^{-1}y \,. \tag{3}$$

Here we skipped the "train" subscripts on K and Y for a more compact notation. Using tr to denote the trace, the derivatives of $\mathcal{P}$ with respect to σ^2 and k are readily obtained via:

$$\begin{aligned}\partial_{\sigma^2}\mathcal{P} &= \frac{1}{2}\operatorname{tr}(K + \sigma^2\mathbf{I})^{-1} - \partial_{\sigma^2}\log p(\sigma^2) - \frac{1}{2}\left\|(K + \sigma^2)^{-1}y\right\|^2 \\ \partial_k\mathcal{P} &= \frac{1}{2}\operatorname{tr}(K + \sigma^2\mathbf{I})^{-1}[\partial_k K] - \partial_k \log p(k) \\ &\quad - \frac{1}{2}y^\top(K + \sigma^2\mathbf{I})^{-1}[\partial_k K](K + \sigma^2\mathbf{I})^{-1}y \,.\end{aligned} \tag{4}$$

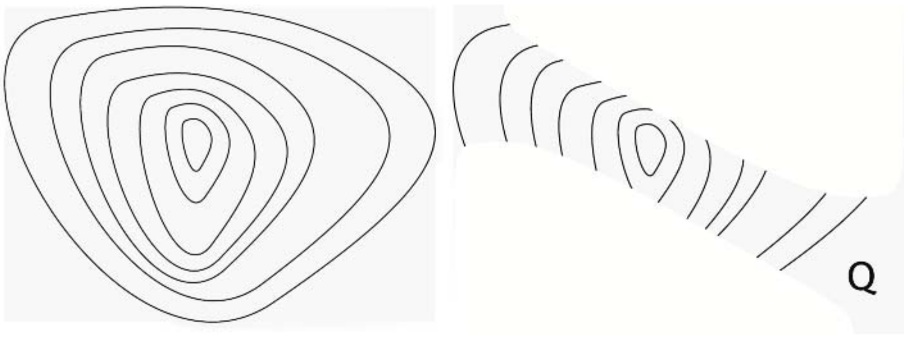


Fig. 1. Left: unrestricted prior, e.g., $p(\sigma^2, k)$, with contour lines indicating equal prior probability; Right: effective prior as restricted by $\mathcal{Q}_\epsilon$ to a sub-domain of (σ^2, k) which satisfy the marginal constraints on the test set. The order of hypotheses within the feasible set remains unchanged by the intersection with $\mathcal{Q}_\epsilon$. However, the normalization changes due to the restriction of the domain.

Minimization is achieved, e.g., by gradient descent. In terms of computation, the key cost is to deal with the inverse of $(K + \sigma^2\mathbf{I})$.

3 Transduction and Empirical Bayes

When viewing the negative log-posterior (3) it is obvious that X_{test} does not enter the discussion. This is perfectly reasonable provided that our prior on k and σ^2 is well specified. In reality, however, we can rarely be sure that the prior is sufficiently accurate. We address this issue in the following by a semi-empirical construction of the prior on σ^2, k.

3.1 Restricting the Prior

We begin with a prior $p(k, \sigma^2)$ which denotes our (so far observation independent) knowledge of the estimation problem. We would like to modify the prior such that it only contains values of k and σ^2 such that $Y_{\text{test}}|Y_{\text{train}}, X$ has a distribution similar to that of Y_{train}. In particular, we consider distributions from the family $\mathcal{Q}_\epsilon$ which on the test set Y_{test} has mean and variance close to the observed values on the training set:

$$\mathcal{Q}_\epsilon := \{q| \, \|\mathbf{E}_{Y_{\text{test}} \sim q} [\phi(y)] - \bar{\mu}\| \leq \epsilon\} \ . \tag{5}$$

Here $\phi(y) := \left(y, -\frac{1}{2}y^2\right)$ are the sufficient statistics of the normal distribution and $\bar{\mu} = m^{-1} \sum_{i=1}^m \phi(y_i)$ is the empirical statistics of y on the training set Y_{train}.

We could now simply perform inference by minimizing $\mathcal{P}(\sigma^2, k)$ subject to the constraint that $p(Y_{\text{test}}|Y_{\text{train}}, X, \sigma^2, k) \in \mathcal{Q}_\epsilon$ (See Figure 1 for an example). However, there is no guarantee that any (σ^2, k) satisfies the constraint on the distribution. Hence we relax the conditions in the following sections to also include distributions close to $\mathcal{Q}_\epsilon$.

3.2 Unconstrained Minimization

Denote by $D(p\|q)$ the Kullback-Leibler (KL) divergence between two distributions

$$D(q\|p) := \int \log \frac{q(x)}{p(x)} dq(x)$$

and denote by $D(\mathcal{Q}\|p) := \inf_{q \in \mathcal{Q}} D(q\|p)$ the KL-divergence between a distribution p and a subset of distributions $\mathcal{Q}$. Since the KL divergence vanishes only for equivalent distributions, $D(\mathcal{Q}\|p) = 0$ is equivalent to $p \in \mathcal{Q}$. Moreover $D(\mathcal{Q}\|p) \geq 0$ for all p.

This provides us with a *barrier function* to ensure that $p \in \mathcal{Q}$ whenever possible and a measure for the distance between $\mathcal{Q}$ and some $p \notin \mathcal{Q}$. Instead of minimizing $\mathcal{P}(\sigma^2, k)$ we modify the negative log-likelihood and minimize now:

$$\mathcal{P}(\sigma^2, k) + \lambda D(\mathcal{Q}\|p(Y_{\text{test}}|Y_{\text{train}}, X, \sigma^2, k)) \; , \tag{6}$$

where $\lambda \geq 0$. For $\lambda \to \infty$ we obtain the optimization problem with hard constraints on (σ^2, k). For $\lambda \to 0$ we recover the unrestricted problem.

Similar to variational methods the objective function can be rewritten in terms of the entropy of the closest distribution in $\mathcal{Q}$ and an effective likelihood term in p. The problem of minimising (6) can then be rewritten as a joint minimization over (σ^2, k, q) as

$$\inf_{q \in \mathcal{Q}, \sigma^2, k} \mathcal{P}(\sigma^2, k) + \lambda D(q(Y_{\text{test}})\|p(Y_{\text{test}}|Y_{\text{train}}, X, \sigma^2, k)) \; .$$

Decomposing the KL-divergence we have

$$\begin{aligned} \inf_{q \in \mathcal{Q}, \sigma^2, k} & - \log p(Y_{\text{train}}|X_{\text{train}}, \sigma^2, k) - \log p(\sigma^2) - \log p(k) \\ & - \lambda \mathbf{E}_{Y_{\text{test}} \sim q} \left[\log p(Y_{\text{test}}|Y_{\text{train}}, X, \sigma^2, k) \right] - \lambda H(q) \end{aligned} \tag{7}$$

where H denotes the entropy ($H(q) = -\int \log q(x) dq(x)$).

This decomposition closely resembles variational methods for estimation, where an intractable model is replaced by a tractable approximation, see e.g., [17].

The joint minimization problem over q and (σ^2, k) can be solved, e.g., by subspace descent. The advantage of this approach is that while the objective function (7) is jointly *nonconvex* in the parameters, the resulting subproblems may be more amenable to minimization. For instance, the problem of finding a minimizer in q for fixed (σ^2, k) can be recast as a convex problem for certain $\mathcal{Q}$. We have the following algorithm:

1. For fixed q minimize

$$- \log p(Y_{\text{train}}, \sigma^2, k|X_{\text{train}}) - \lambda \mathbf{E}_{Y_{\text{test}} \sim q} \left[\log p(Y_{\text{test}}|Y_{\text{train}}, X, \sigma^2, k) \right] \tag{8}$$

   with respect to (σ^2, k).
2. For fixed (σ^2, k) minimize

$$D(q(Y_{\text{test}})\|p(Y_{\text{test}}|Y_{\text{train}}, X, \sigma^2, k))$$

   with respect to q, where $q \in \mathcal{Q}$.

In the following section we discuss both steps in greater detail for the case of regression. We begin with Step 2.

4 Minimizing the Effective Posterior

4.1 A Duality Theorem for q

Recall the definition of $\mathcal{Q}_\epsilon$ as in (5). There we required that q evaluated on Y_{test} has approximate mean $\bar{\mu}$ with regard to the statistic $\phi(y)$. The following theorem, which follows immediately from [18] is a generalization of the well-known duality between maximum likelihood estimation and entropy maximization with moment matching constraints. It states the connection between maximum a posteriori estimation and entropy maximization with *approximate moment matching constraints*,

Theorem 1 (Approximate KL Minimization). *Denote by $\mathcal{X}$ a domain and let p, q be distributions on $\mathcal{X}$. Moreover, let $\phi(x) : \mathcal{X} \to \mathcal{B}$ be a map from $\mathcal{X}$ to a Banach space $\mathcal{B}$. Then for any $\epsilon \geq 0$ the problem*

$$\min_q \; D(q\|p) \text{ subject to } \left\|\mathbf{E}_{x\sim q}\left[\phi(x)\right] - \bar{\mu}\right\| \leq \epsilon$$

has the solution

$$q_\theta(x) = p(x)\exp\left(\langle\phi(x), \theta\rangle - g(\theta)\right)$$

where θ is an element of the dual space of $\mathcal{B}$. Here $g(\theta)$ ensures that $q(x)$ is normalized to 1. Moreover θ is found as solution of the maximum a posteriori estimation problem

$$\min_\theta \; g(\theta) - \langle\bar{\mu}, \theta\rangle + \epsilon\left\|\theta\right\| \; . \tag{9}$$

Equivalently for every feasible ϵ there exists some $\Lambda \geq 0$ such that the minimum of $g(\theta) - \langle\bar{\mu}, \theta\rangle + \frac{\Lambda}{2}\left\|\theta\right\|^2$ minimizes (9).

The quadratic formulation in $\|\theta\|^2$ is preferable in terms of optimization as it is always feasible. In terms of the transductive regression estimation problem this means that

$$q(Y_{\text{test}}) = p(Y_{\text{test}}|Y_{\text{train}}, X, \sigma^2, k)\exp\left(\langle\phi(Y_{\text{test}}), \theta\rangle - g(\theta)\right)$$

where $\phi(Y_{\text{test}}) = \frac{1}{m'}\sum_{i=1}^{m'}\left(y'_i, \frac{1}{2}{y'_i}^2\right)$ for m' test instances. Since p is a normal distribution and $\phi(Y_{\text{test}})$ only contains linear and quadratic terms in Y_{test}, the overall distribution $q(Y_{\text{test}})$ will also be normal. This greatly simplifies the calculation of $g(\theta)$ and its derivatives:

$$\partial_\theta g(\theta) = \mathbf{E}\left[\phi(Y_{\text{test}})\right] \text{ and } \partial_\theta^2 g(\theta) = \operatorname{Cov}\left[\phi(Y_{\text{test}})\right] \; .$$

4.2 Minimizing with Respect to q

Let $\mathbf{1}$ denote the all one vector and $\mathbf{I}$ the identity matrix. The linear and quadratic terms in $-\log q(Y_{\text{test}})$, as a function of λ and the mean and variance (μ and Σ) of $p(Y_{\text{test}}|Y_{\text{train}}, X, \sigma^2, k)$ are then given by

$$\frac{1}{2}(Y_{\text{test}} - \mu)^\top \Sigma^{-1}(Y_{\text{test}} - \mu) - \theta_1\mathbf{1}^\top Y_{\text{test}} + \frac{1}{2}\theta_2\left\|Y_{\text{test}}\right\|^2$$

which corresponds to a normal distribution with variance and mean

$$\begin{aligned} \Sigma_q^{-1} &= \Sigma^{-1} + \theta_2 \mathbf{I} \\ \mu_q &= (\Sigma^{-1} + \theta_2 \mathbf{I})^{-1}(\Sigma^{-1}\mu + \theta_1 \mathbf{1}) \,. \end{aligned}$$

The latter can be seen by some tedious but very straightforward algebra matching up linear and quadratic terms in the expansion in Y_{test}. It also allows us to compute the expected value of $\phi(Y_{\text{test}})$ as follows:

$$\begin{aligned} \mathbf{E}\left[\phi_1(Y_{\text{test}})\right] =& \frac{1}{m'}\mathbf{1}^\top \mu_q = \frac{1}{m'}\mathbf{1}^\top(\Sigma^{-1} + \theta_2\mathbf{I})^{-1}(\Sigma^{-1}\mu + \theta_1\mathbf{1}) \\ \mathbf{E}\left[\phi_2(Y_{\text{test}})\right] =& \frac{1}{m'}\left[\operatorname{tr}\Sigma_q + \|\mu_q\|^2\right] \\ =& \frac{1}{m'}\operatorname{tr}\left(\Sigma^{-1} + \theta_2\mathbf{I}\right)^{-1} + \frac{1}{m'}\left\|(\Sigma^{-1} + \theta_2\mathbf{I})^{-1}(\Sigma^{-1}\mu + \theta_1\mathbf{1})\right\|^2 \,. \end{aligned}$$

Putting everything together we obtain the conditions for finding the optimal value of q in transductive regression:

$$\partial_\theta\left[-\log q(\bar{\mu}) + \tfrac{\Lambda}{2}\|\theta\|^2\right] = 0 \Longleftrightarrow \mathbf{E}\left[\phi(Y_{\text{test}})\right] - \bar{\mu} + \Lambda\theta = 0 \,. \qquad (10)$$

Moreover, the solution is unique and the problem can be solved by the Newton method or conjugate gradient descent as the Jacobian of the LHS of (10) is positive definite.

4.3 Minimizing with Respect to p

We now describe how to perform the optimization in Step 1. With regard to the minimization in p we already accomplished a significant part of the calculations in (4). What remains is to deal with the expected log-likelihood of $p(Y_{\text{test}}|Y_{\text{train}}, X, \sigma^2, k)$ with respect to q. We use the following simple lemma:

Lemma 1. *Let $\Sigma, \Sigma_q \succeq 0$ be covariance matrices in $\mathbb{R}^{n\times n}$ and let $\mu, \mu_q \in \mathbb{R}^n$ be corresponding means. In this case*

$$\mathbf{E}_{x\sim\mathcal{N}(\mu_q,\Sigma_q)}\left[(x-\mu)^\top\Sigma^{-1}(x-\mu)\right] = \operatorname{tr}\Sigma^{-1}\Sigma_q + (\mu_q-\mu)^\top\Sigma^{-1}(\mu_q-\mu) \,.$$

Proof (Sketch only). By the trace formula $\mathbf{E}\left[x^\top\Sigma^{-1}x\right] = \operatorname{tr}\mathbf{E}\left[xx^\top\right]\Sigma^{-1}$. Expanding $(x-\mu) = (x-\mu_q) + (\mu_q-\mu)$ and direct calculation yields the desired result.

Consequently we can expand the expected log-likelihood (up to constants) as

$$\begin{aligned} \mathcal{T}(\sigma^2, k, q) :=& -\mathbf{E}_{Y_{\text{test}}\sim\mathcal{N}(\mu_q,\Sigma_q)}\log p(Y_{\text{test}}|Y_{\text{train}}, X, k, \sigma^2) \\ =& \frac{1}{2}\log|\Sigma| + \frac{1}{2}\operatorname{tr}\Sigma^{-1}\Sigma_q + \frac{1}{2}(\mu_q-\mu)^\top\Sigma^{-1}(\mu_q-\mu) \end{aligned}$$

where μ, Σ are given by (1) and (2) respectively. The last step is to take derivatives with respect to those parameters in analogy to $\mathcal{P}$. By standard matrix calculus [19] we obtain

$$\begin{aligned} \partial_k\mathcal{T}(\sigma^2, k, q) =& \frac{1}{2}\operatorname{tr}\Sigma^{-1}\left[\partial_k\Sigma\right] - \frac{1}{2}\operatorname{tr}\Sigma^{-1}\left[\partial_k\Sigma\right]\Sigma^{-1}\Sigma_q \\ & + (\mu-\mu_q)^\top\Sigma^{-1}\left[\partial_k\mu\right] - \frac{1}{2}(\mu_q-\mu)^\top\Sigma^{-1}\left[\partial_k\Sigma\right]\Sigma^{-1}(\mu_q-\mu) \,. \end{aligned}$$

The terms arising from $\partial_{\sigma^2}\mathcal{T}$ are analogous. Finally, the derivatives of Σ and μ with respect to k and σ^2 are given by

$$\begin{aligned}
\partial_{\sigma^2}\mu =& - K_{\text{test,train}}(K_{\text{train,train}} + \sigma^2\mathbf{I})^{-2}Y_{\text{train}} \\
\partial_k\mu =& - K_{\text{test,train}}(K_{\text{train,train}} + \sigma^2\mathbf{I})^{-1}\partial_k K_{\text{train,train}}(K_{\text{train,train}} + \sigma^2\mathbf{I})^{-1}Y_{\text{train}} \\
\partial_{\sigma^2}\Sigma =& \mathbf{I} - K_{\text{test,train}}(K_{\text{train,train}} + \sigma^2\mathbf{I})^{-2}K_{\text{train,test}} \\
\partial_k\Sigma =& \partial_k K_{\text{test,test}} - \partial_k K_{\text{test,train}}(K_{\text{train,train}} + \sigma^2\mathbf{I})^{-1}K_{\text{train,test}} \\
& - K_{\text{test,train}}(K_{\text{train,train}} + \sigma^2\mathbf{I})^{-1}\partial_k K_{\text{train,test}} \\
& + K_{\text{test,train}}(K_{\text{train,train}} + \sigma^2\mathbf{I})^{-1} \\
& + \partial_k K_{\text{train,train}}(K_{\text{train,train}} + \sigma^2\mathbf{I})^{-1}K_{\text{train,test}} \ .
\end{aligned}$$

Finally, the derivatives of the restricted log-posterior given in (8) are given by summing over the terms $\mathcal{P}(\sigma^2, k) + \lambda\mathcal{T}(\sigma^2, k, q)$. Standard optimization methods for choosing adequate parameters in k and σ^2 can subsequently be applied to the problem.

4.4 Application to Automatic Relevance Determination

ARD [16] is a means of determining the scale of random variables. This gives us a principled method for choosing the appropriate parameters k and σ^2. In the context of Gaussian processes, we can parameterize the kernel k by

$$k_\Theta(x, x') := k(\Theta x, \Theta x')$$

where Θ is a diagonal matrix which ensures proper scaling of x in different coordinates. For Gaussian RBF kernels, we have $k(x, x') = \exp\left(-\|\Theta(x - x')\|^2\right)$, whose derivative is given by

$$\partial_\Theta k_\Theta(x, x') = -2((x_1 - x_1')^2\Theta_1, \ldots, (x_n - x_n')^2\Theta_n)^\top \exp\left(-\|\Theta(x - x')\|^2\right) \ .$$

A suitable choice of a prior on the coefficients $\Theta_i \in \mathbb{R}$ can ensure that many of them will vanish. In particular we choose a factorizing gamma prior, for which

$$-\log p(\Theta) = \sum_{i=1}^{n} -a\log\Theta_i + b\Theta_i + \text{const} \ .$$

Similarly we choose a gamma prior for the additive noise σ^2.

5 Experimental Results

5.1 Regression Datasets

Dataset Facts. For experimental evaluation we decided to use the same datasets and preprocesing as in [20]. There, 23 regression datasets from UCI [21] and the R [22]

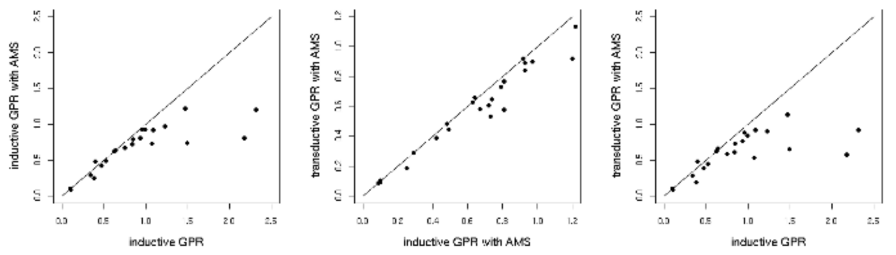


Fig. 2. Mean root mean squared errors of the different approaches on all used datasets. Note the different scaling of the figures

packages `mlbench, quantreg, alr3` and `MASS` were picked[1]. No datasets with missing values were used. In some cases where the target variable was not obvious, it was selected arbitrarily. The sample sizes vary from $m = 43$ to $m = 1375$ and the lengths of input vectors vary from $n = 1$ to $n = 60$. Finally, some datasets were standardized to have zero mean and unit variance (the datasets were also used in this form in [20]).

Overview of Results. We compared transductive and inductive GP regression in 10-fold cross-validations. For inductive GP regression the kernel bandwidth and the additive noise level are chosen via cross validation within the training sample as this is the common practice in many other papers. For transductive GP regression Λ and λ are chosen via cross validation within the training sample. The automatic relevance determination parameters are held constant throughout all experiments ($a = 1, b = 0.5$). To compare the two models we used the root mean squared error over 10-fold cross validations.

The results are illustrated in Figure 2 and full details are given in Table 1. The last three columns are the mean $\pm$ standard deviation of the root mean square errors. In the large majority of the test cases, the transductive GP regression outperforms the inductive GP regression in terms of root mean square error (20 wins/ 3 losses). However, not in all cases the difference is significant.

Statistical Comparison. To verify that transductive Gaussian processes with automatic model selection (AMS) significantly outperform inductive Gaussian processes (with AMS) over all datasets, we need to perform a proper statistical test with the null hypothesis that the algorithms perform equally well. As suggested recently [23] we used the Wilcoxon signed ranks test.

The Wilcoxon signed ranks test is a nonparametric test to detect shifts in populations given a number of paired samples. The underlying idea is that under the null hypothesis the distribution of differences between the two populations is symmetric about 0. It proceeds as follows: (i) compute the differences between the pairs, (ii) determine the ranking of the absolute differences, and (iii) sum over all ranks with positive and negative difference to obtain W_+ and W_-, respectively. The null hypothesis can be rejected

[1] Descriptions are available at http://cran.r-project.org/src/contrib/PACKAGES.html

Table 1. Dataset facts (number of instances, number of attributes, class attribute, dropped attributes, standardized (Yes, No)) and regression results (root mean squared error of inductive, inductive Gaussian processes with AMS, and transductive Gaussian processes with AMS, respectively). Bold numbers denote smaller error.

Data set	#Inst	#Att	Class	Dropped	Std	Inductive	Ind. (AMS)	Transd (AMS)
diabetes	43	3	c_peptide	-	N	$\mathbf{0.64 \pm 0.66}$	$\mathbf{0.64 \pm 0.66}$	0.66 ± 0.45
triazines	186	61	activity	-	N	0.10 ± 0.12	0.10 ± 0.10	$\mathbf{0.09 \pm 0.17}$
pyrimidines	74	28	activity	-	N	0.10 ± 0.09	$\mathbf{0.09 \pm 0.05}$	$\mathbf{0.09 \pm 0.05}$
BigMac2003	69	10	BigMac	City	Y	1.08 ± 0.95	0.73 ± 0.85	$\mathbf{0.53 \pm 0.55}$
UN3	125	7	Purban	Locality	Y	0.84 ± 0.44	0.72 ± 0.42	$\mathbf{0.61 \pm 0.38}$
topo	52	3	z	-	Y	1.49 ± 2.75	0.74 ± 1.05	$\mathbf{0.65 \pm 0.40}$
mcycle	133	2	accel	-	Y	1.23 ± 0.38	0.97 ± 0.20	$\mathbf{0.9 \pm 0.25}$
CobarOre	38	3	z	-	Y	1.47 ± 1.32	1.22 ± 0.92	$\mathbf{1.14 \pm 0.61}$
highway	39	12	Rate	-	Y	1.00 ± 0.84	0.93 ± 0.68	$\mathbf{0.84 \pm 0.66}$
sniffer	125	5	Y	-	Y	0.75 ± 0.37	0.67 ± 0.33	$\mathbf{0.58 \pm 0.31}$
caution	100	3	y	-	Y	1.03 ± 0.86	0.92 ± 0.54	$\mathbf{0.91 \pm 0.55}$
gilgais	365	9	e80	-	Y	0.85 ± 0.62	0.79 ± 0.6	$\mathbf{0.73 \pm 0.55}$
ftcollinssnow	93	2	Late	YR1	Y	2.31 ± 3.45	1.20 ± 0.95	$\mathbf{0.92 \pm 0.51}$
crabs	200	7	CW	index	Y	0.34 ± 0.26	0.29 ± 0.21	$\mathbf{0.29 \pm 0.21}$
BostonHousing	506	14	medv	-	Y	0.47 ± 0.35	0.42 ± 0.29	$\mathbf{0.39 \pm 0.27}$
engel	235	2	y	-	Y	2.18 ± 4.05	0.81 ± 0.75	$\mathbf{0.58 \pm 0.50}$
heights	1375	2	Dheight	-	Y	$\mathbf{0.10 \pm 0.10}$	$\mathbf{0.10 \pm 0.10}$	$\mathbf{0.10 \pm 0.09}$
snowgeese	45	5	photo	-	Y	0.53 ± 0.44	0.49 ± 0.43	$\mathbf{0.44 \pm 0.49}$
ufc	372	5	Height	-	Y	$\mathbf{0.63 \pm 0.39}$	$\mathbf{0.63 \pm 0.31}$	$\mathbf{0.63 \pm 0.31}$
birthwt	189	8	bwt	ftv, low	Y	0.38 ± 0.55	0.25 ± 0.51	$\mathbf{0.19 \pm 0.22}$
GAGurine	314	2	GAG	-	Y	0.94 ± 0.72	0.81 ± 0.79	$\mathbf{0.77 \pm 0.82}$
geyser	299	2	waiting	-	Y	0.96 ± 0.61	0.93 ± 0.65	$\mathbf{0.89 \pm 0.48}$
cpus	209	8	estperf	name	Y	$\mathbf{0.40 \pm 0.46}$	0.48 ± 0.78	0.48 ± 0.78

if W_+ (or $\min(W_+, W_-)$, respectively) is located in the tail of the null distribution which has sufficiently small probability.

The critical value of the one-sided Wilcoxon signed ranks test for 23 samples on a 0.5% significance level is 55. On this significance level we can reject the null hypotheses.

6 Outlook and Future Work

We presented a new transductive GP regression method, where the prior distribution on model selection parameters is modified for approximate moment matching between training and test set. Experimental results show the competitiveness of our approach. Note that significant improvements were achieved in cases where the size of the unlabelled data is only $1/10$-th of the training data. We expect even larger improvements over inductive GP when more unlabelled data is used. We also would like to emphasize that this method is in fact orthogonal to other transductive methods, eg. one can use a semi-supervised kernel function as well as the moment matching constraints.

It is important to note the generality of this method. The approximate moment matching constraints have been applied to classification problems and can easily be extend to structural learning: All we need to do is impose the constraints that the expectations of singleton labels as well as the expectations of the neighboring label clusters over the test set should approximately match the statistics of the training data. That is, we impose moment matching conditions on the class marginals. Note that if we have a large amount of data at our disposition, imposing only moment constraints may be wasteful. That is, we have information not only about the class marginals globally but also *locally*. This leads to an interesting crossover of inductive and transductive estimation, which is subject of current research.

Finally note the similarity of our setup to empirical Bayes estimation insofar as we adjust the prior over the hypothesis space *after* seeing the data. While this clearly runs counter to proper Bayesian procedure, it still produces convincingly better results. It would be interesting to see whether it is possible to obtain statistical confidence bounds for our estimator: From [18] it follows immediately that the expected log-likelihood is well concentrated. This, however, is not our aim — we would like to obtain bounds that are *better* than the conventional uniform convergence bounds taking advantage of the fact that we have additional test data.

The transductive Gaussian process regression with automatic relevance determination can help us to determine which feature is important in regression. This information can be useful in many fields, for example in bio-informatics, where the knowledge of which genes play important roles is valuable.

Acknowledgements

The authors wish to thank the reviewers for valuable comments. The National ICT Australia is partially funded through the Australian Government's *Baking Australia's Ability* initiative and the Australian Research Council. Part of this work is also funded by the German Science Foundation DFG under grant WR40/2-2.

References

1. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison (2005) http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf.
2. Chapelle, O., Zien, A.: Semi-supervised classification by low density separation. In: Tenth International Workshop on Artificial Intelligence and Statistics. (2005)
3. Zhu, X., Lafferty, J., Ghahramani, Z.: Semi-supervised learning using gaussian fields and harmonic functions. In: Proc. Intl. Conf. Machine Learning. (2003)
4. Smola, A.J., Kondor, I.R.: Kernels and regularization on graphs. In Schölkopf, B., Warmuth, M.K., eds.: Proc. Annual Conf. Computational Learning Theory. Lecture Notes in Computer Science, Springer (2003) 144–158
5. Pozdnoukhov, A., Bengio, S.: Semi-supervised kernel methods for regression estimation. In: IEEE International Conference on Acoustic, Speech, and Signal Processing. (2006)
6. Verbeek, J., Vlassis, N.: Gaussian fields for semi-supervised regression and correspondence learning. Pattern Recognition, special issue on similarity based pattern recognition (2006)
7. Tresp, V.: A Bayesian committee machine. Neural Computation **12**(11) (2000) 2719–2741

8. Schwaighofer, A., Tresp, V.: Transductive and inductive methods for approximate guassian process regression. In: Neural Information Processing Systems, MIT Press (2003)
9. Chapelle, O., Vapnik, V., Weston, J.: Transductive inference for estimating values of functions. In: Advances in Neural Information Processing Systems. (1999)
10. Schuurmans, D., Southey, F., Wilkinson, D., Guo, Y.: Metric-based approaches for semi-supervised regression and classification. In: Semi-Supervised Learning. MIT Press (2006)
11. Brefeld, U., Gärtner, T., Scheffer, T., Wrobel, S.: Efficient co-regularised least squares regression. In: 23rd International Conference on Machine Learning. (2006)
12. Gärtner, T., Le, Q., Burton, S., Smola, A.J., Vishwanathan, S.V.N.: Large-scale multiclass transduction. In Weiss, Y., Schölkopf, B., Platt, J., eds.: Advances in Neural Information Processing Systems 18, Cambride, MA, MIT Press (2006) 411 – 418
13. Neal, R.M.: Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report Technical Report 9702, Dept. of Statistics (1997)
14. Le, Q.V., Smola, A.J., Canu, S.: Heteroscedastic gaussian process regression. In: Proc. Intl. Conf. Machine Learning. (2005)
15. Williams, C.K.I.: Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In Jordan, M.I., ed.: Learning and Inference in Graphical Models. Kluwer Academic (1998) 599–621
16. Neal, R.M.: Assessing relevance determination methods using delve. In: Neural Networks and Machine Learning, Springer (1998) 97–129
17. Jordan, M.I., Gharamani, Z., Jaakkola, T.S., Saul, L.K.: An introduction to variational methods for graphical models. In Jordan, M.I., ed.: Learning in Graphical Models. Kluwer Academic (1998) 105–162
18. Altun, Y., Smola, A.: Divergence minimization and convex duality. In: Proc. Annual Conf. Computational Learning Theory. (2006) to appear.
19. Lütkepohl, H.: Handbook of Matrices. John Wiley and Sons, Chichester (1996)
20. Takeuchi, I., Le, Q., Sears, T., Smola, A.: Nonparametric quantile estimation. Journal of Machine Learning Research (2006) To appear and available at http://sml.nicta.com.au/$\sim$quocle.
21. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
22. R Development Core Team: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. (2005) ISBN 3-900051-07-0.
23. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7(1)** (2006)

Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees

Alessandro Moschitti

Department of Computer Science
University of Rome “Tor Vergata”, Italy
moschitti@info.uniroma2.it

Abstract. In this paper, we provide a study on the use of tree kernels to encode syntactic parsing information in natural language learning. In particular, we propose a new convolution kernel, namely the Partial Tree (PT) kernel, to fully exploit dependency trees. We also propose an efficient algorithm for its computation which is futhermore sped-up by applying the selection of tree nodes with non-null kernel. The experiments with Support Vector Machines on the task of semantic role labeling and question classification show that (a) the kernel running time is linear on the average case and (b) the PT kernel improves on the other tree kernels when applied to the appropriate parsing paradigm.

1 Introduction

Literature work shows several attempts (e.g. [1]) to define linking theories between the syntax and semantics of natural languages. As no complete theory has yet been defined the design of syntactic features to learn semantic structures requires a remarkable research effort and intuition. Tree kernels have been applied to reduce such effort for several natural language tasks, e.g. syntactic parsing re-ranking [2], relation extraction [3], named entity recognition [4,5] and Semantic Role Labeling [6].

These studies show that the kernel ability to generate large feature sets is useful to quickly model new and not well understood linguistic phenomena in learning machines. However, it is often possible to manually design features for linear kernels that produce high accuracy and fast computation time whereas the complexity of tree kernels may prevent their application in real scenarios.

In general, the poor tree kernel results depend on the specific application but also on the absence of studies that suggest which tree kernel type should be applied. For example, the *subtree* (ST) kernel defined in [7] is characterized by structures that contain all the descendants of the target root node until the leaves whereas the *subset trees* (SSTs) defined in [2] may contain internal subtrees, with no leaves. How do such different spaces impact on natural language tasks? Does the parsing paradigm (constituent or dependency) affect the accuracy of different kernels?

Regarding the complexity problem, although the SST kernel computation time has been proven to be inherently quadratic in the number of tree nodes [2], we may design algorithms that run fast on the average case.

In this paper, we study the impact of the ST and SST kernels on the modeling of syntactic information in Support Vector Machines. To carry out a comprehensive investigation, we have defined a novel tree kernel based on a general form of substructures, namely, the partial tree (PT) kernel. Moreover, to solve the computation problems, we propose algorithms which, on the average case, evaluate the above kernels in a running time linear in the number of nodes of the two input parse trees.

We experimented with such kernels and Support Vector Machines (SVMs) on (a) the classification of semantic roles defined in PropBank [8] and FrameNet [9] and (b) the classification of questions from Question Answering scenarios. We used both gold standard trees from the Penn Treebank [10] and automatic trees derived with the Collins [11] and Stanford [12] parsers. The results show that: (1) the SST kernel is more appropriate to exploit syntactic information from constituent trees. (2) The new PT kernel is slightly less accurate than the SST one on constituent trees but much more accurate on dependency structures. (3) Our fast algorithms show a linear running time.

In the remainder of this paper, Section 2 introduces the different tree kernel spaces. Section 3 describes the kernel functions and our fast algorithms for their evaluation. Section 4 introduces the Semantic Role Labeling (SRL) and Question Classification (QC) problems and their solution along with the related work. Section 5 shows the comparative kernel performance in terms of execution time and accuracy. Finally, Section 6 summarizes the conclusions.

2 Tree Kernel Spaces

The kernels that we consider represent trees in terms of their substructures (fragments). The kernel function detects if a tree subpart (common to both trees) belongs to the feature space that we intend to generate. For such purpose, the desired fragments need to be described. We consider three important characterizations: the subtrees (STs), the subset trees (SSTs) and a new tree class, i.e. the partial trees (PTs).

As we consider syntactic parse trees, each node with its children is associated with a grammar production rule, where the symbol at the left-hand side corresponds to the parent and the symbols at the right-hand side are associated with the children. The terminal symbols of the grammar are always associated with the tree leaves.

We define as a **subtree** (ST) any node of a tree along with all its descendants. For example, Figure 1 shows the parse tree of the sentence `"Mary brought a cat"` together with its 6 STs. A **subset tree** (SST) is a more general structure since its leaves can be non-terminal symbols.

For example, Figure 2 shows 10 SSTs (out of 17) of the subtree of Figure 1 rooted in `VP`. The SSTs satisfy the constraint that grammatical rules cannot be broken. For example, `[VP [V NP]]` is an SST which has two non-terminal symbols, `V` and `NP`, as leaves whereas `[VP [V]]` is not an SST. If we relax the constraint over the SSTs, we obtain a more general form of substructures that we

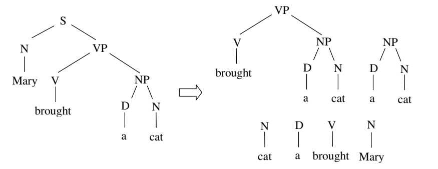


Fig. 1. A syntactic parse tree with its subtrees (STs)

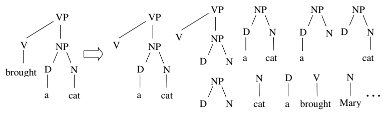


Fig. 2. A tree with some of its subset trees (SSTs)

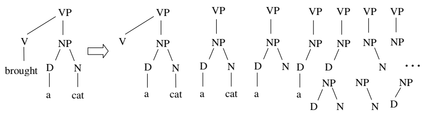


Fig. 3. A tree with some of its partial trees (PTs)

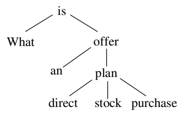


Fig. 4. A dependency tree of a question

call **partial trees** (PTs). These can be generated by the application of partial production rules of the grammar, consequently `[VP [V]]` and `[VP [NP]]` are valid PTs. Figure 3 shows that the number of PTs derived from the same tree as before is still higher (i.e. 30 PTs). These different substructure numbers provide an intuitive quantification of the different information levels among the tree-based representations.

3 Fast Tree Kernel Functions

The main idea of tree kernels is to compute the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. We have designed a general function to compute the ST, SST and PT kernels. Our fast evaluation of the PT kernel is inspired by the efficient evaluation of non-continuous subsequences (described in [13]). To increase the computation speed of the above tree kernels, we also apply the pre-selection of node pairs which have non-null kernel.

3.1 The Partial Tree Kernel

The evaluation of the common PTs rooted in nodes n_1 and n_2 requires the selection of the shared child subsets of the two nodes, e.g. `[S [DT JJ N]]` and `[S [DT N N]]` have `[S [N]]` (2 times) and `[S [DT N]]` in common. As the order of the children is important, we can use subsequence kernels for their generation. More in detail, let $\mathcal{F} = \{f_1, f_2, .., f_{|\mathcal{F}|}\}$ be a tree fragment space of type PTs and let the indicator function $I_i(n)$ be equal to 1 if the target f_i is rooted at node n and 0 otherwise, we define the PT kernel as:

$$K(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2), \tag{1}$$

where N_{T_1} and N_{T_2} are the sets of nodes in T_1 and T_2, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1)I_i(n_2)$, i.e. the number of common fragments rooted at the n_1 and n_2 nodes. We can compute it as follows:

- if the node labels of n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
- else

$$\Delta(n_1, n_2) = 1 + \sum_{\boldsymbol{J}_1, \boldsymbol{J}_2, l(\boldsymbol{J}_1)=l(\boldsymbol{J}_2)} \prod_{i=1}^{l(\boldsymbol{J}_1)} \Delta(c_{n_1}[\boldsymbol{J}_{1i}], c_{n_2}[\boldsymbol{J}_{2i}]) \qquad (2)$$

where $\boldsymbol{J}_1 = \langle J_{11}, J_{12}, J_{13}, ..\rangle$ and $\boldsymbol{J}_2 = \langle J_{21}, J_{22}, J_{23}, ..\rangle$ are index sequences associated with the ordered child sequences c_{n_1} of n_1 and c_{n_2} of n_2, respectively, $\boldsymbol{J}_{1i}$ and $\boldsymbol{J}_{2i}$ point to the i-th children in the two sequences, and $l(\cdot)$ returns the sequence length.

We note that (1) Eq. 2 is a convolution kernel according to the definition and the proof given in [14]. (2) Such kernel generates a richer feature space than those defined in [7, 2, 3, 5, 13]. Additionally, we add two decay factors: μ for the height of the tree and λ for the length of the child sequences. It follows that

$$\Delta(n_1, n_2) = \mu\Big(\lambda^2 + \sum_{\boldsymbol{J}_1, \boldsymbol{J}_2, l(\boldsymbol{J}_1)=l(\boldsymbol{J}_2)} \lambda^{d(\boldsymbol{J}_1)+d(\boldsymbol{J}_2)} \prod_{i=1}^{l(\boldsymbol{J}_1)} \Delta(c_{n_1}[\boldsymbol{J}_{1i}], c_{n_2}[\boldsymbol{J}_{2i}])\Big) \qquad (3)$$

where $d(\boldsymbol{J}_1) = \boldsymbol{J}_{1l(\boldsymbol{J}_1)} - \boldsymbol{J}_{11}$ and $d(\boldsymbol{J}_2) = \boldsymbol{J}_{2l(\boldsymbol{J}_2)} - \boldsymbol{J}_{21}$. In this way, we penalize both larger trees and subtrees built on child subsequences that contain gaps. Moreover, to have a similarity score between 0 and 1, we also apply the normalization in the kernel space, i.e. $K'(T_1, T_2) = \frac{K(T_1,T_2)}{\sqrt{K(T_1,T_1)\times K(T_2,T_2)}}$.

3.2 Efficient Tree Kernel Computation

Clearly, the naïve approach to evaluate Eq. 3 requires exponential time. We can efficiently compute it by considering that the summation in Eq. 3 can be distributed with respect to different types of sequences, e.g. those composed by p children; it follows that $\Delta(n_1, n_2) = \mu\big(\lambda^2 + \sum_{p=1}^{lm} \Delta_p(c_{n_1}, c_{n_2})\big)$, (4)

where Δ_p evaluates the number of common subtrees rooted in subsequences of exactly p children (of n_1 and n_2) and $lm = min\{l(c_{n1}), l(c_{n2})\}$. Also note that if we only consider the contribution of the longest child sequence from node pairs that have the same children, we implement the SST kernel. For the STs computation we also need to remove the λ^2 term from Eq. 4.

Given the two child sequences $s_1a = c_{n_1}$ and $s_2b = c_{n_2}$ (a and b are the last children),

$$\Delta_p(s_1a, s_2b) = \Delta(a, b) \times \sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} \lambda^{|s_1|-i+|s_2|-r} \times \Delta_{p-1}(s_1[1:i], s_2[1:r]),$$

where $s_1[1:i]$ and $s_2[1:r]$ are the child subsequences from 1 to i and from 1 to r of s_1 and s_2. If we name the double summation term as D_p, we can rewrite the relation as:

$$\Delta_p(s_1 a, s_2 b) = \begin{cases} \Delta(a,b) D_p(|s_1|,|s_2|) & \text{if } a = b; \\ 0 & \mathit{otherwise}. \end{cases}$$

Note that D_p satisfies the recursive relation: $D_p(k,l) =$

$$\Delta_{p-1}(s_1[1:k], s_2[1:l]) + \lambda D_p(k, l-1) + \lambda D_p(k-1, l) + \lambda^2 D_p(k-1, l-1) \quad (5)$$

By means of the above relation, we can compute the child subsequences of two sequences s_1 and s_2 in $O(p|s_1||s_2|)$. This means that the worst case complexity of the PT kernel is $O(p\rho^2|N_{T_1}||N_{T_2}|)$, where ρ is the maximum branching factor of the two trees. Note that the average ρ in natural language parse trees is very small and the overall complexity can be reduced by avoiding the computation of node pairs with different labels. The next section shows our fast algorithm to find non-null node pairs.

Table 1. Pseudo-code for fast evaluation of the node pairs with non-null kernel (FTK)

```
function Evaluate_Pair_Set(Tree $T_1$, $T_2$)
LIST $L_1$,$L_2$;
NODE_PAIR_SET $N_p$;
begin
   $L_1$ = $T_1$.ordered_list;
   $L_2$ = $T_2$.ordered_list; // lists sorted at loading time
   $n_1$ = extract($L_1$); // get the head element and remove it from the list
   $n_2$ = extract($L_2$);
   while ($n_1$ and $n_2$ are not NULL)
      if (label($n_1$) > label($n_2$))
         then $n_2$ = extract($L_2$);
         else if (label($n_1$) < label($n_2$))
            then $n_1$ = extract($L_1$);
            else
               while (label($n_1$) == label($n_2$))
                  while (label($n_1$) == label($n_2$))
                     add($\langle n_1, n_2 \rangle$, $N_p$);
                     $n_2$=get_next_elem($L_2$); /*get the head element and
                     move the pointer to the next element*/
                  end
                  $n_1$ = extract($L_1$);
                  reset($L_2$); //set the pointer at the first element
               end
   end
   return $N_p$ ;
end
```

3.3 Fast Non-null Node Pair Computation

To compute the tree kernels, we sum the Δ function for each pair $\langle n_1, n_2 \rangle \in N_{T_1} \times N_{T_2}$ (Eq. 1). When the labels associated with n_1 and n_2 are different, we can avoid evaluating $\Delta(n_1, n_2)$ since it is 0. Thus, we look for a node pair set $N_p = \{\langle n_1, n_2 \rangle \in N_{T_1} \times N_{T_2} : \mathit{label}(n_1) = \mathit{label}(n_2)\}$. N_p can be evaluated by (i) extracting the L_1 and L_2 lists of nodes from T_1 and T_2, (ii) sorting them in alphanumeric order and (iii) scanning them to derive the node intersection. Step (iii) may require only $O(|N_{T_1}| + |N_{T_2}|)$ time, but, if $\mathit{label}(n_1)$=$\mathit{label}(n_2)$ appears r_1 times in T_1 and r_2 times in T_2, the number of pairs will be $r_1 \times r_2$. The formal algorithm (FTK) is shown in Table 1.

Note that the list sorting can be done only once at data preparation time (i.e. before training) in $O(|N_{T_1}| \times log(|N_{T_1}|))$. The worst case occurs when the two parse trees are both generated by only one production rule since the two internal *while* cycles generate $|N_{T_1}| \times |N_{T_2}|$ pairs. Moreover, the probability of two identical production rules is lower than that of two identical nodes, thus, we can furthermore speed up the SST (and ST) kernel by (a) sorting the node list with respect to production rules and (b) replacing the `label(`n`)` function with `production_at(`n`)`.

3.4 Partial Tree Kernel Remarks

In order to model a very fast PT kernel computation, we have defined the algorithm in Section 3.2 to evaluate it efficiently and we apply the selection of non-null node pairs (algorithm in Table 1) which can be also applied to the ST and SST kernels.

Our algorithm in Section 3.2 allows us to evaluate PT kernel in $O(\rho^3|N_{T_1}||N_{T_2}|)$, where ρ is the maximum branching factor of the two trees T_1 and T_2. It should be emphasized that the naïve approach for the evaluation of the PT function is exponential. Therefore, a fairer comparison of our approach should be carried out against the efficient algorithm proposed in [3] for the evaluation of relation extraction kernels (REKs). These are not convolution kernels and produce a much lower number of substructures than the PT kernel. The complexity of REK was $O(\rho^4)$ when applied to only two nodes. If we applied it to all the node pairs of two trees (as we do with the PT kernel), we would obtain a complexity of $O(\rho^4|N_{T_1}||N_{T_2}|)$ which is higher than the one produced by our method. Consequently, our solution is very efficient and produces larger substructure spaces.

Moreover, to further speed up the kernel computation, we apply Eq. 4 to node pairs for which the output is not null. A similar approach was suggested in [2,13] for the computation of the SST kernel. However, its impact on such kernel has not been clearly shown by an extensive experimentation and the effect on the new PT kernel should also be measured. For this purpose, in sections 5.1 and 5.2 we report the running time experiments for the evaluation of the SST and PT kernels and the training time that they generate in SVMs.

4 Semantic Applications of Parse Tree Kernels

Semantic Role Labeling (SRL) and Question Classification (QC) are two interesting natural language tasks in which the impact of tree kernels can be measured. The former relates to the classification of the predicate argument structures defined in PropBank [8] or FrameNet [9]. For example, Figure 5 shows the parse tree of the sentence: `"Mary brought a cat to school"` along with the predicate argument annotation proposed in the PropBank project. Only verbs are considered as predicates whereas arguments are labeled sequentially from Arg0 to Arg5. Additionally, adjuncts are labeled with several ArgM labels, e.g. ArgM-TMP or ArgM-LOC.

In FrameNet predicate/argument information is described by means of rich semantic structures called Frames. These are schematic representations of situations involving various participants, properties and roles in which a word may

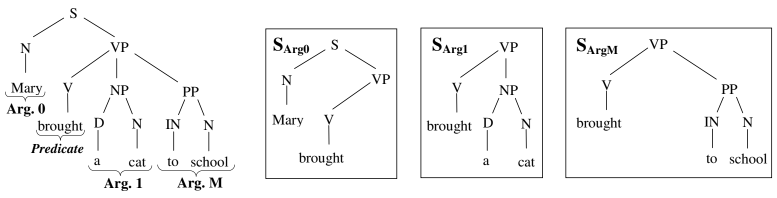


Fig. 5. Tree substructure space for predicate argument classification

typically be used. Frame elements or semantic roles are arguments of target words, i.e. the predicates. For example the following sentence is annotated according to the Arrest Frame:

$[_{Time}$ `One Saturday night`$]$ $[_{Authorities}$ `police in Brooklyn` $]$ $[_{Target}$ `apprehended` $]$ $[_{Suspect}$ `sixteen teenagers`$]$.

The semantic roles *Suspect* and *Authorities* are specific to this Frame.

The common approach to learn the classification of predicate arguments relates to the extraction of features from syntactic parse trees of the training sentences [15]. An alternative representation based on tree kernels selects the minimal partial tree that includes a predicate with only one of its arguments [6]. For example, in Figure 5, the semantic/syntactic substructures associated with the three arguments of the verb *to bring*, i.e. S_{Arg0}, S_{Arg1} and S_{ArgM}, are shown inside the three boxes. Note that such representation is quite intuitive.

Another interesting task is the classification of questions in the context of Question Answering (QA) systems. Detecting the type of a question, e.g. whether it asks for a person or for an organization, is critical to locate and extract the right answer from the available documents. The long tradition of QA in TREC has produced a large question set used in several researches. These are categorized according to different taxonomies of different *grains*. We consider the *coarse grained* classification scheme described in [16, 17]: Abbreviations, Descriptions (e.g. *definition* and *manner*), Entity (e.g. *animal*, *body* and *color*), Human (e.g. *group* and *individual*), Location (e.g. *city* and *country*) and Numeric (e.g. *code* and *date*).

The idea of using tree kernels for Question Classification is to encode questions by means of their whole syntactic parse tree. This is simpler than tailoring the subtree around the semantic information provided by PropBank or FrameNet for the SRL task. Additionally, we can easily experiment with other kind of parsing paradigms, e.g. the dependency parsing. A dependency tree of a sentence is a syntactic representation that denotes grammatical relations between words. For example, Figure 4 shows a dependency tree of the question "`What is an offer of direct stock purchase plan?`".

We note that (1) the father-children node relationship encodes the dependency between the head, e.g. *plan*, and its modifiers, e.g. *direct*, *stock* and *purchase*. In our approximation, we only consider the dependency structure by removing the link labels, i.e. we do not use either "of" between *offer* and *plan* or the other labels like "object" and "subject". (2) It is clear that the SST and ST kernels cannot fully exploit the representational power of a dependency tree since from subtrees like `[plan [direct stock purchase]]`, they cannot generate substructures like `[plan [stock purchase]]` or `[plan [direct purchase]]`. In contrast, the PT kernel can generate all of these subsequences allowing SVMs to better generalize on dependency structures although the strong specialization of the SST kernel may be superior in some tasks. The experiments of Section 5 confirm our observations.

4.1 Related Work

In [2], the SST kernel was experimented with the Voted Perceptron for the parse-tree re-ranking task. The combination with the original PCFG model improved the syntactic parsing. In [18], an interesting algorithm that speeds up the average running time is presented. Such algorithm uses the explicit fragment space to compute the kernel between small trees. The results show an increase of the speed similar to the one produced by our methods. In [3], two kernels over syntactic shallow parser structures were devised for the extraction of linguistic relations, e.g. *person-affiliation*. To measure the similarity between two nodes, the *contiguous string kernel* and the *sparse string kernel* were used. In [5] such kernels were slightly generalized by providing a matching function for the node pairs. The time complexity for their computation limited the experiments on a data set of just 200 news items. In [4], a feature description language was used to extract structural features from the syntactic shallow parse trees associated with named entities. The experiments on named entity categorization showed that too many irrelevant tree fragments may cause overfitting. In [6] the SST kernel was firstly proposed for semantic role classification. The combination between such kernel and a polynomial kernel of standard features improved the state-of-the-art. To complete such work, an analysis of different tree kernel spaces as carried out here was required. In [19], the computational complexity problem is addressed by considering only selected trees and the RankBoost algorithm.

5 The Experiments

In these experiments, we study tree kernels in terms of (a) average running time, (b) accuracy on the classification of predicate argument structures of PropBank (gold trees) and FrameNet (automatic trees) and (c) accuracy of QC on automatic question trees.

The experiments were carried out with the SVM-light-TK software available at `http://ai-nlp.info.uniroma2.it/moschitti/` which encodes ST, SST and PT kernels in the SVM-light software [21]. We adopted the default regularization parameter and we tried a few cost-factor values (i.e., $\{1, 3, 7, 10, 30, 100\}$) to adjust the rate between Precision and Recall on the development set. We modeled the multiclassifiers by training an SVM for each class according to the ONE-vs-ALL scheme and by selecting the class associated with the maximum score.

For the ST, SST and PT kernels, we found that the best λ values (see Section 3) on the development set were 1, 0.4 and 0.8, respectively, whereas the best μ

was 0.4. We measured the performance by using the F_1 measure[1] for the single arguments and the accuracy for the final multiclassifiers.

5.1 Kernel Running Time Experiments

To study the FTK running time, we extracted from the Penn Treebank 2 [10] several samples of 500 trees containing exactly n nodes. Each point of Figure 6 shows the average computation time[2] of the kernel function applied to the 250,000 pairs of trees of size n. It clearly appears that the FTK and FTK-PT (i.e. FTK applied to the PT kernel) average running time has linear behavior whereas, as expected, the algorithm (QTK) which does not use non-null pair selection shows a quadratic curve.

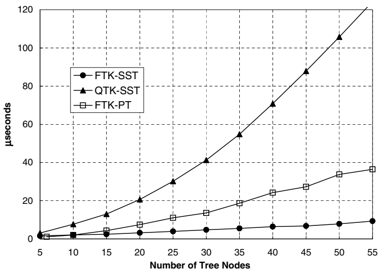


Fig. 6. Average time in μseconds for the QTK, FTK and FTK-PT evaluations

5.2 Experiments on ProbBank

The aim of these experiments is to measure the impact of kernels on the semantic role classification accuracy. We used PropBank (`www.cis.upenn.edu/∼ace`) along with the gold standard parses of the Penn Treebank.

The corpus contains about 53,700 sentences and a fixed split between training and testing used in other researches, e.g. [22]. Sections from 02 to 21 are used for training, Section 23 for testing and Section 22 as development set for a total of 122,774 and 7,359 arguments in training and testing, respectively. We considered arguments from Arg0 to Arg5, ArgA and ArgM. This latter refers to all adjuncts collapsed together, e.g. *adverb*, *manner*, *negation*, *location* and so on (13 different types).

Figure 7 illustrates the learning curves associated with the above kernels for the SVM multiclassifiers. We note that: (a) the SST and linear kernels show the highest accuracy, (b) the richest kernel in terms of substructures, i.e. the one based on PTs, shows lower accuracy than the SST and linear kernels but higher than the ST kernel and (c) the results using all training data are comparable with those obtained in [22], i.e. 87.1% (role classification) but we should take into account the different treatment of ArgMs.

Regarding the convergence complexity, Figure 8 shows the learning time of SVMs using QTK, FTK and FTK-PT for the classification of one large argument (Arg0), according to different sizes of training data. With 70% of the data, FTK is about 10 times faster than QTK. With all the data FTK terminated in 6 hours

[1] F_1 assigns equal importance to Precision P and Recall R, i.e. $f_1 = \frac{2P \times R}{P+R}$.

[2] We run the experiments on a Pentium 4, 2GHz, with 1 Gb ram.

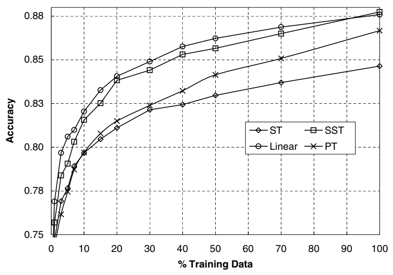


Fig. 7. Multiclassifier accuracy according to different training set percentages

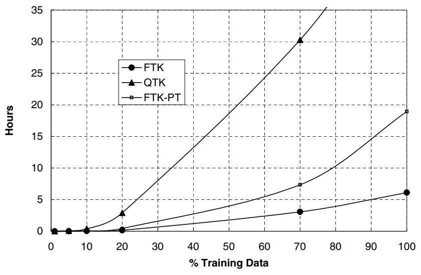


Fig. 8. Arg0 classifier learning time according to different training percentages and kernel algorithms

whereas QTK required more than 1 week. However, the real *complexity burden* relates to working in the dual space. To alleviate such problem interesting and effective approaches have been proposed [23, 24].

5.3 Classification Accuracy with Automatic Trees on FrameNet

As PropBank arguments are defined with respect to syntactic considerations, we should verify that the syntactic information provided by tree kernels is also effective to detect other forms of semantic structures. For this purpose, we experimented with our models and FrameNet data (`www.icsi.berkeley.edu/~framenet`) which is mainly produced based on semantic considerations. We extracted all 24,558 sentences from the 40 Frames selected for the *Automatic Labeling of Semantic Roles* task of Senseval 3 (`www.senseval.org`). We considered the 18 most frequent roles, for a total of 37,948 examples (30% of the sentences for testing and 70% for training/validation). The sentences were processed with the Collins' parser [11] to generate automatic parse trees.

Table 2 reports the F_1 measure of some argument classifiers and the accuracy of the multiclassifier using all available training data for linear, ST, SST and PT kernels. We note that: (1) the F_1 of the single arguments across the different kernels follows a behavior similar to the accuracy of the global multiclassifier. (2) The high F_1 measures of tree kernels on automatic trees of FrameNet show that they are robust with respect to parsing errors.

5.4 Experiments on Question Classification

We used the data set available at `http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/`. This contains 5,500 training and 500 test questions from the TREC 10 QA competition. As we adopted the question taxonomy known as coarse grained introduced in Section 4, we can compare with literature results, e.g. [16, 17].

These experiments show that the PT kernel can be superior to the SST kernel when the source of syntactic information is expressed by dependency rather than constituent trees. For this purpose, we run the Stanford Parser (available

Table 2. Evaluation of kernels on 18 FrameNet semantic roles

Roles	Linear	ST	SST	PT
agent	89.8	86.9	87.8	86.2
theme	82.9	76.1	79.2	79.4
manner	70.8	79.9	82.0	81.7
source	86.5	85.6	87.7	86.6
Acc.	82.3	80.0	81.2	79.9

Table 3. Kernel evaluation on Question Classification according to different parsing approaches

Parsers	Const.		Depend.		BOW
Kernels	SST	PT	SST	PT	Linear
Acc.	88.2	87.2	82.1	90.4	87.3

at `http://www-nlp.stanford.edu/software/lex-parser.shtml`) to generate both parse types. Moreover, we used an SVM with the linear kernel over the bag-of-words (BOW) as baseline. Columns 2 and 3 of Table 3 show the accuracy of the SST and PT kernels over the constituent trees, columns 4 and 5 report the accuracy on the dependency data and Column 6 presents the BOW kernel accuracy.

We note that (1) the SST kernel is again superior to the PT kernel when using constituent trees. If we apply the SST kernel on the dependency trees the resulting accuracy is rather lower than the one of the PT kernel (82.1% vs. 90.4%). This is quite intuitive as the SST kernel cannot generate the features needed to represent all the possible n-ary relations derivable from father-children relations. Overall, the accuracy produced by the dependency trees is higher than the one attainable with the constituent trees. Nevertheless, when the SST kernel applied to the dependency structures is combined with BOW, the SVM accuracy reaches 90% as well [16].

6 Conclusions

In this paper, we have studied the impact of diverse tree kernels for the learning of syntactic/semantic linguistic structures. We used the subtree (ST) and the subset tree (SST) kernels defined in previous work, and we designed a novel general tree kernel, i.e. the partial tree (PT) kernel. Moreover, we improved the kernel usability by designing fast algorithms which process syntactic structures in linear average time.

The experiments with Support Vector Machines on the PropBank and FrameNet predicate argument structures show that richer kernel spaces are more accurate, e.g. SSTs and PTs produce higher accuracy than STs. However, if such structures are not relevant for the representation of the target linguistic objects improvement does not occur, e.g. PTs are not better than SSTs to describe constituent trees. On the contrary, as suggested by the experiments on Question Classification, the richer space provided by PTs produces a much higher accuracy than SSTs when applied to dependency trees. This because the SST kernel seems not adequate to process such data.

Finally, the running time experiments show that our fast tree kernels can be efficiently applied to hundreds of thousands of instances.

References

1. Jackendoff, R.: Semantic Structures, Current Studies in Linguistics series. Cambridge, Massachusetts: The MIT Press (1990)
2. Collins, M., Duffy, N.: New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In: ACL. (2002)
3. Zelenko, D., Aone, C., Richardella, A.: Kernel methods for relation extraction. JMLR (2003)
4. Cumby, C., Roth, D.: Kernel methods for relational learning. In: ICML. (2003)
5. Culotta, A., Sorensen, J.: Dependency tree kernels for relation extraction. In: Proceedings ACL, Barcelona, Spain (2004)
6. Moschitti, A.: A study on convolution kernels for shallow semantic parsing. In: proceedings of ACL, Barcelona, Spain (2004)
7. Vishwanathan, S., Smola, A.: Fast kernels on strings and trees. In: Proceedings of NIPS. (2002)
8. Kingsbury, P., Palmer, M.: From Treebank to PropBank. In: Proceedings of LREC, Las Palmas, Spain (2002)
9. Fillmore, C.J.: Frame semantics. In: Linguistics in the Morning Calm. (1982)
10. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of english: The Penn Treebank. CLJ. (1993).
11. Collins, M.: Three generative, lexicalized models for statistical parsing. In: Proceedings of the ACL, Somerset, New Jersey. (1997).
12. Klein, D., Manning, C.D.: Fast exact inference with a factored model for natural language parsing. In: NIPS. (2002)
13. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
14. Haussler, D.: Convolution kernels on discrete structures. Technical report ucs-crl-99-10, University of California Santa Cruz (1999)
15. Gildea, D., Jurasfky, D.: Automatic labeling of semantic roles. CLJ. (2002)
16. Zhang, D., Lee, W.S.: Question classification using support vector machines. In: Proceedings of SIGIR. (2003).
17. Li, X., Roth, D.: Learning question classifiers: The role of semantic information. JNLE. (2005).
18. Kazama, J., Torisawa, K.: Speeding up training with tree kernels for node relation labeling. In: Proceedings of EMNLP, Toronto, Canada (2005)
19. Kudo, T., Suzuki, J., Isozaki, H.: Boosting-based parse reranking with subtree features. In: Proceedings ACL'05, (2005).
20. Carreras, X., Màrquez, L.: Introduction to the CoNLL-2005 shared task: Semantic role labeling. In: Proceedings of CoNLL-2005. (2005).
21. Joachims, T.: Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods - Support Vector Learning. (1999)
22. Pradhan, S., Hacioglu, K., Krugler, V., Ward, W., Martin, J.H., Jurafsky, D.: Support vector learning for semantic argument classification. MLJ. (2005).
23. Kudo, T., Matsumoto, Y.: Fast methods for kernel-based text analysis. In: Proceedings of ACL. (2003).
24. Suzuki, J., Isozaki, H., Maeda, E.: Convolution kernels with feature selection for natural language processing tasks. In: Proceedings of ACL, Spain (2004).

Why Is Rule Learning Optimistic and How to Correct It

Martin Možina, Janez Demšar, Jure Žabkar, and Ivan Bratko

Faculty of Computer and Information Science, University of Ljubljana, Tržaška cesta 25, SI-1001 Ljubljana

Abstract. In their search through a huge space of possible hypotheses, rule induction algorithms compare estimations of qualities of a large number of rules to find the one that appears to be best. This mechanism can easily find random patterns in the data which will – even though the estimating method itself may be unbiased (such as relative frequency) – have optimistically high quality estimates. It is generally believed that the problem, which eventually leads to overfitting, can be alleviated by using m-estimate of probability. We show that this can only partially mend the problem, and propose a novel solution to making the common rule evaluation functions account for multiple comparisons in the search. Experiments on artificial data sets and data sets from the UCI repository show a large improvement in accuracy of probability predictions and also a decent gain in AUC of the constructed models.

1 Introduction

Most rule learning algorithms [8] induce models by iteratively searching for the best rule and removing the examples covered by it. Rules are usually sought by a beam search, which gradually adds conditions to the rule with aim to decrease the number of covered (so-called) negative examples, while at the same time losing as few positive examples as possible. The search is guided by two measures, one which evaluates the partial rules and the other which selects between the final rule candidates; most often, as in the case of this paper, the same measure is used for both purposes.

A good rule should give accurate class predictions, or, in other words, have a high probability of the positive class among all examples (not only learning examples) covered by rule. Hence relative frequency, an unbiased estimator of probability, seems to be a reasonable choice for the measure of quality of rule:

$$Q(r) = \frac{s}{n} \tag{1}$$

where n is the number of learning examples covered by the rule r and s is the number of positive examples among them.

However, the assumption that the relative frequency indeed estimates the probability of positive class is wrong. Fig. 1(a) shows how searching through a large space of rules, which tries to maximize relative frequencies, can always find rules with 100% positive subsets, though these are usually purely random patterns in the data and their true positive class probabilities are much lower.[1] Class proportions for the rules found by the search process are thus completely uncorrelated with the true class probabilities.

[1] Experimental details are provided in the next section.

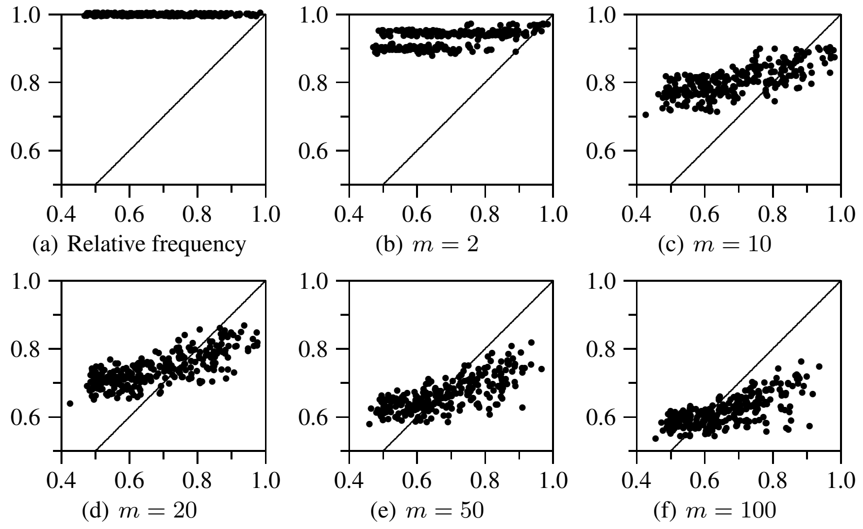


Fig. 1. Relation between the estimated (y-axis) and true (x-axis) class probabilities for rules from artificial data sets

A more general version of this problem has been extensively explored by Jensen and Cohen [12] who blame multiple comparisons during the search to be responsible for plethora of pathologies in induction algorithms. Our paper proposes a method which can fix the relative frequency estimate and other rule evaluation measures by taking multiple comparisons into account through the use of extreme value distributions [7].

Since a review of all proposed improvements of evaluation measures would take the entire paper, the next section only studies the effect of the m-estimate of probability [2] as a good representative of such techniques. We then present our algorithm and, in the following section, validate it on several artificial and UCI data sets. The conclusion gives a list of several open questions and limitations of the methods.

2 Experimental Study of Rule Estimators

The m-estimate [2] computes the class probability (or, in our case, the rule quality) as

$$Q_m(r) = \frac{s + m \times p_a}{n + m} \tag{2}$$

where p_a is the prior probability and m is a parameter of the method. Fuernkranz and Flach [8] showed that the m-estimate presents a trade off between precision (relative frequency) and linear cost metrics (for instance, weighted relative accuracy [13,16]). Different values of parameter m can be used to approximate many currently used evaluation functions. For instance, when $m = 0$, m-estimate equals the relative frequency. Instead of citing various proposals from the extensive related work, we shall thus concentrate on the more general m-estimate.

Table 1. Comparison of rules obtained from artificial data sets with different values for m: the average true class probability, Spearman correlation between the true probability and the estimate, and the mean square error of the estimate

m	avg. accuracy	Spearman	mean error
0	0.68	0.00	0.119
2	0.68	0.54	0.074
10	0.68	0.68	0.027
20	0.68	0.72	0.015
50	0.67	0.70	0.009
100	0.66	0.65	0.010

To observe the correlation between the true and the estimated class probabilities, we constructed a set of artificial data sets with controlled class probabilities for each possible rule. We have prepared 300 data sets with ten binary attributes. Five attributes in each data set were unrelated with the class. For the other five, we prescribed a (random) class probability for each combination of their values. We then generated 2^{10} examples for each data set, one for each combination of attribute values, and assigned the classes randomly according to the prescribed probabilities for the combination of informative attributes. Note that the actual class proportions in the data set do not necessarily match the defined probabilities for a particular combination of attribute values.[2]

For each of 300 data sets we learned a single rule using different values for m (0, 2, 10, 20, 50, 100). Fig. 1 shows the relation between the rule's estimated class probability $Q_m(r)$ and the known true probability, which we shall denote by $\widetilde{Q}(r)$. As we already mentioned in the introduction, at $m = 0$ (relative frequency), the method is extremely optimistic. With increasing values of m, the method is still optimistic for rules with lower true probability, but pessimistic for rules with higher true probability. It seems that m-estimate lowers the estimated quality by the same amount for all rules, which can not adjust the estimates to lie on (or at least close to) the ideal diagonal line representing the perfect correlation.

Table 1 compares the measured evaluation functions by

- the average true prediction accuracy of the induced rules, which reveals the quality of the evaluation function as search heuristics;
- the Spearman correlation coefficient between $Q_m(r)$ and $\widetilde{Q}(r)$, that shows the quality of the rule ordering, which is crucial when rules are used for classification, where we need to distinguish between "stronger" and "weaker" rules;
- the mean square difference between $Q_m(r)$ and $\widetilde{Q}(r)$ which indicates the rule's accuracy when used as probabilistic predictor.

The first column of Table 1 suggests that lower values of m give (marginally) better rules than higher values. However, higher m's score better in terms of the Spearman correlation and give better probability estimates.

In conclusion, using the m-estimate with a suitably tuned m can considerably decrease the error of the estimated probabilities, yet, as seen from the graphs, the major

[2] We obtained similar results in experiments with other ways of constructing artificial data sets.

$$\begin{tikzcd}
s \arrow[d] & & \tilde{s} \\
LRS \arrow[dr, "\text{EVD}"'] & & \widetilde{LRS} \arrow[u] \\
& P_a(r) \arrow[ur, "\chi^2"'] &
\end{tikzcd}$$

Fig. 2. An outline of the proposed procedure

effect comes from reducing the optimism, while the correlation between the true and the estimated probability remains rather poor. m-estimate and the many other similar techniques are thus not a satisfactory solution to the problem of overfitting, wrong rule quality estimates and optimistic probability predictions.

3 Algorithm for Improved Probability Estimate

Relative frequencies, m-estimates and other potential measures of rule quality are computed from the number of examples covered by the rule (n) and the number positive examples among them (s). We have seen that relative frequencies overestimate the true probability because the algorithm searches for the rule with the highest $s : n$ ratio. Since the training data presents only a limited sample from the population, the observed ratio for each rule is subject to random distribution, so the found rule is therefore not necessarily the optimal one, and it almost certainly has an optimistic $s : n$ ratio. One way of preventing these unwanted effects and improving the probability estimate s/n is to try to find the expected value of s, which we shall denote by $\tilde{s}$.

The outline of the proposed procedure is illustrated in Fig. 2. For reasons that will become clear later, we start by computing the log-likelihood ratio statistics (LRS) for 2×2 tables derived by Dunning [6]. It is usually assumed that LRS is distributed according to $\chi^2(1)$. This is, however, true only for randomly chosen rules, or, in our case, for LRS computed from the expected value of s, $\tilde{s}$.

The highest observed LRS (computed from s, not $\tilde{s}$) is distributed according to the Fisher-Tippet extreme value distribution (EVD) [7].[3] Since EVD depends only the number of rules considered in the search (it is more likely to get higher LRS if number of rules considered is higher), which is determined by the rule length, the chosen search algorithm and the properties of the data set (number of examples, number and type of attributes), we will be able to compute corresponding EVDs – for rules of different lengths for the selected algorithm and a particular data set – in advance.

Now consider a specific rule. From n and the observed s, we can compute the LRS and then, knowing its distribution (EVD), find the probability that a rule this good (or better) is found under the null-hypothesis of no relation between the attribute and class values. We shall denote this probability, the "significance of the rule", by $P_a(r)$. Note that $P_a(r)$ takes the multiple comparisons into account, so this estimate is unbiased.

[3] For illustration, although the daily levels of a river are usually distributed normally, the distribution of maximal annual levels is not normal but Fisher-Tippet's.

On the other hand, imagine that we knew the expected $\tilde{s}$ of that rule. We could compute its true log-likelihood ratio $\widetilde{\mathrm{LRS}}$ and, through $\chi^2(1)$ distribution, arrive at the same significance $P_a(r)$ as above.

The trick that we use in this paper is to reverse the second path. So, from the unbiased $P_a(r)$ which we get from the known (but optimistic) s through LRS and EVD, we shall compute the unbiased $\widetilde{\mathrm{LRS}}$ and the corresponding $\tilde{s}$.

The reason for which we need to compute LRS instead of computing the extreme value distribution for $Q(r) = s/n$ directly, and estimate the unbiased $\widetilde{\mathrm{Q}}$ through $P_a(r)$, is that the extreme value of a sampled random variable (such as s, $Q(r)$ or LRS) is distributed by the Fisher-Tippet (or some other) extreme value distribution only if the variable's values are taken from a fixed distribution (independent from s and n). LRS, as we just noted, fulfils this criterion, while s/n is distributed according to $\beta(s, n-s)$ and is thus not the same for all rules.

As a side note, our approach to correcting quality estimators can be generalized to other criteria beside $Q(r) = s/n$. If the density distribution of a criterion depends upon the rule (like is the case with s/n), we need to find a measure which is well-correlated with the criterion (additional explanation is given later), yet drawn from a fixed distribution (like LRS, which is drawn from $\chi^2(1)$ and still reasonably correlated with s/n).[4] If the observed criterion already comes from a fixed distribution (if, for example, LRS would be used as the main evaluation function), finding a correlated measure is not needed and we can immediately proceed to the computation of EVD.

This section will present the details of the algorithm, along with a running example for illustration.

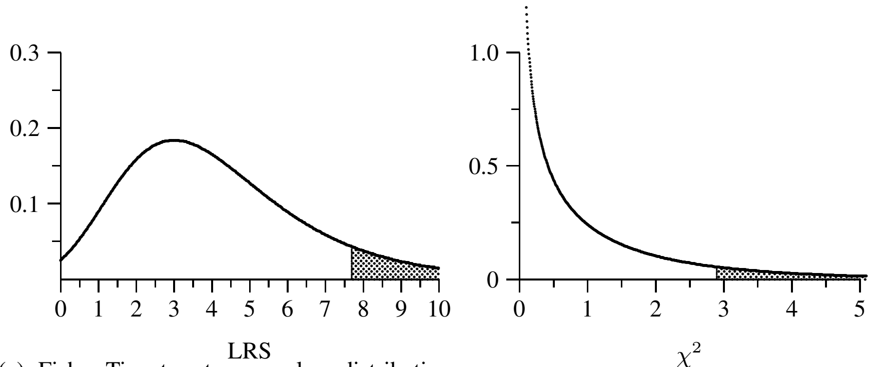
Step 1: From s to LRS. Let s again be the number of positive examples covered by rule and let s^c be the number of positive examples not covered by the rule. Similarly let n be the number of covered examples by the rule and n^c be the number of examples not covered by the rule. LRS is then defined as:

$$\mathrm{LRS} = 2\left[s \log \frac{s}{e_s} + (n-s)\log\frac{n-s}{e_{n-s}} + s^c \log\frac{s^c}{e_{s^c}} + (n^c - s^c)\log\frac{n^c - s^c}{e_{n^c - s^c}}\right] \quad (3)$$

where e_x is the expected value of x. For instance, e_s is computed as $n\frac{s+s^c}{n+n^c}$. When computed on a randomly chosen rule, LRS is distributed according to $\chi^2(1)$ distribution, disregarding properties of the rule (length, s, n...) and the data. Note that a similar formula for LRS, without the last two terms, was used in [4,3] for computing significance of rules. However, as that formula is approximately correct only if n is small enough when compared to n^c, we prefer to use the formula 3 derived by Dunning [6].

Example. We have a data set with 20 examples where the prior probability of the positive class is 0.5. Learning from that data, the rule search algorithm found a rule r with two conditions which covers 10 examples with 8 of them belonging to the positive class. Its LRS is, according to (3), 7.7.

[4] The correlation would be perfect if every rule covered the same number of examples.



(a) Fisher-Tippet extreme value distribution ($\mu = 3, \beta = 2$)

(b) χ^2 with 1 degree of freedom.

Fig. 3. Probability density functions

Step 2: From LRS to $P_a(r)$. $P_a(r)$ measures the probability that, given a random data with no relation between the attribute and class values, the rule found by the chosen search procedure will have the quality of at least $\mathrm{LRS}(r)$ (or another suitable measure of quality). This definition suggests a way of computing $P_a(r)$: like Jensen and Cohen [12], we permute the class values in the data set so that all rules are purely random and their true probability for positive class equals the prior probability. We then induce a rule on the randomized data set and compute its LRS. Repeating it for many times we get a distribution for LRSs.

Gumbel and Lieblein [9,10] (cited in [14]) have shown that the limiting distribution of all χ^2 distributions is the Fisher-Tippet distribution (Fig. 3(a)). Fisher-Tippet distribution is characterized by two parameters, location (μ) and scale (β). For LRS, it can be shown that β always equals 2, and μ equals the median of the above sample to which we add $2 \ln \ln 2$ (see Appendix A for a proof). In general, values of μ and β depend upon the number of rules covered by the search (which does not necessarily equal the number of *explicitly* evaluated rules), which in turn depends upon the rule length and the data set (and, of course, the search algorithm). Due to their independence of the actual rule, we can compute values $\mu(L)$ and $\beta(L)$ for different rule lengths before we begin learning, using the algorithm shown in Fig. 4. The algorithm runs until $\mu(L)$ is smaller than $\mu(L-1)$, which signifies that rules of length $L-1$ can not be improved because they are perfect or they do not cover enough examples.

During learning we use the cumulative Fisher-Tippet distribution function (see the formula in Appendix A) with the pre-computed parameters to estimate $P_a(r)$.

Example (continued). Say that algorithm from Fig. 4 found $\mu(2) = 3$ and $\beta(2) = 2$ (remember that rule r has two conditions). The curve with such parameters is depicted in Fig. 3(a), so the probability $P_a(r)$ for the rule from our example corresponds to the shaded area right of LRS=7.7. $P_a(r)$ equals approximately 0.09.

Step 3: From $P_a(r)$ to $\widetilde{\mathrm{LRS}}(r)$. To compute $\widetilde{\mathrm{LRS}}(r)$ we need to do the opposite from the last step. Looking at the $\chi^2(1)$ distribution (Fig. 3(b)), we need to find such a value $\widetilde{\mathrm{LRS}}(r)$ that the area under the curve to the right of it will equal $P_a(r)$. In other words, the shaded areas under the curves in Fig. 3 should be the same.

1. Let $L = 1$ (L is the maximum rule length).
2. Permute values of class in the data.
3. Learn a rule on this data (using LRS as evaluation measure), where the maximum length of rule is L.[5]
4. Record the LRS of the rule learned.
5. Repeat steps 2-4 to collect a large enough (say 100) sample of LRSs
6. Estimate parameters $\mu(L)$ and $\beta(L)$ of the Fisher-Tippet distribution (see Appendix A).
7. If $\mu(L) > \mu(L-1)$, then $L = L + 1$ and return to step 2.

Fig. 4. The algorithm for computing parameters of the Fisher-Tippet distributions

Example (continued). The corresponding $\widetilde{\mathrm{LRS}}$ for our examples as read from Fig. 3(b) is 2.9. Note that this is much less than LRS $= 7.7$, which we computed directly from the data and which would essentially be used by an unmodified rule induction algorithm.

Step 4: From $\widetilde{\mathrm{LRS}}$ to $\tilde{s}$. The remaining task is trivial: compute $\tilde{s}$ from the formula for $\widetilde{\mathrm{LRS}}$ using an arbitrary root finding algorithm. Similar would be done for statistics other than $\widetilde{\mathrm{LRS}}$ and $\tilde{s}$. In our task we are correcting probability estimates based on relative frequencies, so we shall compute them by dividing the corrected $\tilde{s}$ by n.

Example (conclusion). We used Brent's method [1] to find that $\widetilde{\mathrm{LRS}} = 2.9$ corresponds to $\tilde{s} = 6.95$. The rule covers ten examples, so the corresponding class probability is $6.95/10 = 0.695$. Note that this estimate is quite smaller than the uncorrected 0.8.

4 Experiments

We have tested the algorithm on artificial data described in Section 2 and on a selection of data sets from the UCI repository [15]. In all experiments we used beam search [3,4] with a beam width set to 5. The algorithm was implemented as a component for the rule based learner in machine learning system Orange [5].

The results of using the corrected measure on the artificial data are shown in Fig. 5. The estimated class probabilities are nicely strewn close to the diagonal axis, which is a clear improvement in comparison with the results from Fig. 1. This is also confirmed by the quantitative measure of fit: the average true probability is the same as the highest values in Table 1, the mean quadratic error is a little better than that of m-estimates, while the Spearman coefficient is clearly superior.

We mentioned that LRS is perfectly correlated with class probabilities only if every rule covers the same number of examples. Our data is constructed in that way, while real data sets certainly do not possess that property. To test the practical impact of our correction, we observed its behaviour on a set of UCI data sets. Each data set was split evenly onto learn and test sets. For learning we then generated ten bootstrap samples from the learn set.

[5] Note that using LRS at a given rule length will always order rules the same as would $\widetilde{\mathrm{LRS}}$. However, as we will be using $\tilde{(s)}/n$ in the actual learning phase, in order to correctly estimate parameters of Fisher-Tippet distribution, measures $\tilde{s}/n$ and $\widetilde{\mathrm{LRS}}$ should be well correlated.

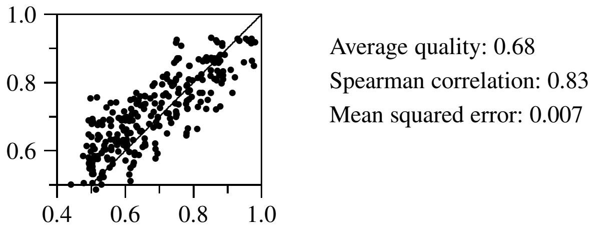


Fig. 5. Relation between the corrected (y-axis) and the true (x-axis) class probability

We ran the algorithm on the bootstrap samples and then used the examples from the test set to count the number of positive and the number of all examples covered by each induced rule. We took this ratio to be the true positive class probability for the rule (although it is, as a matter of fact, still only an estimate, it is at least an unbiased one, since it is computed from the test data). Results in Table 2(a) show that we succeeded in improving the probability estimates: the probability estimates by our method are far more accurate than those by any m in m-estimate measure.

This would, however, be easily achieved and surpassed by a method returning a single rule covering all examples and which would estimate the probability with the prior class probability. To test that our gains are not due to oversimplification we also computed the average AUC over the ten bootstrap samples. To make predictions from lists of rules, we used a simple classifier that takes the first rule that triggers for each class (we get one rule for each class), and normalize the class probabilities of these rules to sum up to 1. Although there exist better classifiers from a set of rules, we believe that using them would not considerably change the ranking of examples and the related AUC. Table 2(b) shows that the performance of our method in terms of AUC is comparable to that of the other methods.

5 Conclusion

We have described a correction for removing the optimism in rule evaluation measures which arises since the rule was selected among many other rules considered during the search based on this same measure. The correction is based on the idea that the optimistic statistics, which is distributed according to extreme-value distribution, and the sought for statistics, which is (in case of LRS, which we used) distributed by $\chi^2(1)$ should predict the same probability that the rule was found by chance.

Tests on artificial data sets show that the correction works well. Experiments on real-world UCI data sets also confirm the gain in terms of probability predictions without decreasing the accuracy of predictive models.

There remain several limitations and unsolved problems. First, extreme value distributions are computed in advance, using entire data set. Common rule learning algorithms use a separate-and-conquer approach in which the covered examples are removed at each step, therefore changing the statistical properties of the data set. As a most obvious consequence, removing examples reduces the effective search space, which makes our correction too strict. We should therefore recompute the parameters of

Table 2. Mean squared errors and AUCs for estimation by relative frequencies (rel.)m, m-estimate and our method (EVD)

(a) Mean squared error over all rules

Data set	rel	m=2	m=10	m=20	m=50	m=100	EVD
adult	0.43	0.13	0.08	0.06	0.06	0.06	**0.04**
australian	0.41	0.12	**0.05**	**0.05**	**0.05**	0.08	**0.05**
balance	0.25	0.12	0.11	0.11	0.08	0.07	**0.05**
breast (lju)	0.39	0.14	0.09	0.09	0.08	**0.06**	**0.06**
breast (wsc)	0.15	0.08	0.13	0.18	0.26	0.30	**0.05**
car	0.07	0.06	0.05	0.04	0.03	**0.02**	0.04
credit	0.41	0.11	0.07	0.07	**0.06**	0.07	**0.06**
german	0.42	0.13	0.06	0.06	0.05	0.05	**0.04**
hayes-roth	0.26	0.10	0.16	0.21	0.26	0.29	**0.08**
hepatitis	0.35	0.12	**0.05**	0.06	0.09	0.09	0.07
ionosphere	0.27	0.05	0.06	0.09	0.13	0.13	**0.03**
iris	0.20	0.07	0.09	0.12	0.17	0.22	**0.04**
lymphography	0.28	0.10	0.17	0.21	0.22	0.24	**0.05**
monks-1	0.07	0.07	0.16	0.13	0.15	0.20	**0.06**
monks-2	0.40	0.13	0.10	0.11	0.07	0.08	**0.05**
monks-3	0.32	0.09	0.08	0.11	0.14	0.13	**0.03**
mushroom	**0.00**	0.01	0.08	0.13	0.18	0.25	0.01
pima	0.48	0.15	0.05	**0.04**	**0.04**	**0.04**	0.05
SAHeart	0.46	0.19	0.08	0.07	**0.05**	0.07	0.07
shuttle	0.26	0.13	0.18	0.14	0.17	0.19	**0.11**
tic-tac-toe	0.19	0.03	0.07	0.14	0.24	0.30	**0.01**
titanic	**0.01**	0.02	0.04	0.04	0.04	0.03	0.02
voting	0.28	0.08	0.10	0.12	0.11	0.10	**0.04**
wine	0.09	0.07	0.14	0.20	0.24	0.31	**0.05**
zoo	0.16	0.09	0.22	0.31	0.42	0.47	**0.04**

(b) AUC

Data set	rel	m=2	m=10	m=20	m=50	m=100	EVD
adult	0.74	0.76	0.76	0.77	0.77	0.78	**0.84**
australian	0.85	0.87	0.88	0.88	0.88	0.88	**0.91**
balance	**0.82**	0.81	0.81	**0.82**	**0.82**	0.81	**0.82**
breast (lju)	0.60	**0.62**	0.60	0.58	0.60	0.60	**0.62**
breast (wsc)	0.97	0.97	0.97	0.97	0.96	0.96	**0.98**
car	0.84	0.84	0.85	0.86	0.89	**0.90**	**0.90**
credit	0.82	0.88	0.88	0.88	0.87	0.88	**0.91**
german	0.69	0.68	0.69	0.68	0.69	0.69	**0.73**
hayes-roth	0.88	0.89	0.87	0.86	0.87	0.86	**0.90**
hepatitis	**0.77**	0.76	0.76	0.73	0.73	0.71	**0.77**
ionosphere	0.90	0.91	0.89	0.89	0.89	0.91	**0.92**
iris	**0.97**	0.95	0.95	0.95	0.95	0.95	0.95
lymphography	0.78	0.81	0.83	**0.85**	0.84	0.83	0.81
monks-1	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
monks-2	0.66	**0.67**	0.66	0.66	0.64	0.65	0.64
monks-3	0.97	0.98	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
mushroom	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
pima	0.68	0.72	0.72	0.72	0.72	0.73	**0.76**
SAHeart	0.59	0.62	0.63	0.63	**0.65**	**0.65**	**0.65**
shuttle	**0.99**	0.98	0.98	**0.99**	**0.99**	**0.99**	0.98
tic-tac-toe	0.96	**1.00**	**1.00**	**1.00**	0.99	0.94	**1.00**
titanic	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**
voting	0.95	0.95	0.96	0.96	**0.97**	**0.97**	0.96
wine	0.97	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	0.94
zoo	**1.00**	0.99	0.99	0.99	0.99	0.99	0.99

EVD distributions after each step, which is not practically feasible. The alternative would be to develop a rule learning algorithm that does not remove learning examples.

Extreme value distributions, as computed in the paper, account for multiple comparisons between the rules of the same length, but not between the rules of different lengths. We have developed and tested a remedy for this, but we omitted it in the paper since the impact of this correction is minimal – the number of comparisons between rules with different length is usually small.

We believe that the proposed method has a lot of potential. Although we here applied it only for correcting the class probability estimates, the same trick could, in principle, be applied to correcting other measures of rule quality that are being optimized by the search process. It may be even adoptable to other machine learning methods that extensively search through the space of possible hypotheses, such as learning decision trees, and which could significantly benefit from such corrections.

Acknowledgements

This work was partly supported by the European Commission's Information Society Technologies (IST) programme, through Project ASPIC (IST-FP6-002307), and Slovene Agency for Research and Development (ARRS).

References

1. Kendall E. Atkinson. *An Introduction to Numerical Analysis*. John Wiley and Sons, New York, 1989.
2. B. Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence*, pages 147–149, 1990.
3. Peter Clark and Robin Boswell. Rule induction with CN2: Some recent improvements. In *Machine Learning - Proceeding of the Fifth Europen Conference (EWSL-91)*, pages 151–163, Berlin, 1991.
4. Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning Journal*, 4(3):261–283, 1989.
5. J. Demšar and B. Zupan. Orange: From experimental machine learning to interactive data mining. White Paper [`http://www.ailab.si/orange`], Faculty of Computer and Information Science, University of Ljubljana, 2004.
6. Ted E. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.
7. R.A. Fisher and L.H.C. Tippett. Limiting forms of the frequency distribution of the largest and smallest member of a sample. *Proc. Camb. Phil. Soc.*, 24:180–190, 1928.
8. Johannes Fuernkranz and Peter A. Flach. Roc 'n' rule learning – towards a better understanding of covering algorithms. *Machine Learning*, 58(1):39–77, January 2005.
9. Emil J. Gumbel. Statistical theory of extreme values and some practical applications. *National Bureau of Standards Applied Mathematics Series (US Government Printing Office)*, 33, 1954.
10. Emil J. Gumbel and J. Lieblein. Some applications of extreme-value models. *American Statistician*, 8(5):14–17, 1954.
11. Shanti S. Gupta. Order statistics from the gamma distribution. *Technometrics*, 2:243–262, 1960.

12. David D. Jensen and Paul R. Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3):309–338, March 2000.
13. Nada Lavrač, Peter Flach, and Blaž Zupan. Rule evaluation measures: A unifying view. In Saša Džeroski and Peter Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, pages 174–185, Bled, Slovenia, 1999.
14. Wentian Li, Fengzhu Sun, and Ivo Grosse. Extreme value distribution based gene selection criteria for discriminant microarray data analysis using logistic regression. *Journal of Computational Biology*, 11(2/3):215–226, 2004.
15. P. M. Murphy and D. W. Aha. UCI Repository of machine learning databases `[http://www.ics.uci.edu/~mlearn/mlrepository.html]`. Irvine, CA: University of California, Department of Information and Computer Science, 1994.
16. Ljupčo Todorovski, Peter Flach, and Nada Lavrač. Predictive performance of weighted relative accuracy. In D. Zighed, J. Komorowski, and J. Zytkow, editors, *Proceedings of the 4th European Conference of Principles of Data Mining and Knowledge Discovery (PKDD-00)*, pages 255–264, Lyon, France, 2000.

A Appendix: Computing Parameters of Extreme-Value Distribution

Section 3 describes an algorithm for computing extreme distributions of rules learned from random data which involves calculating the parameters of extreme value distribution for a vector of maxima of evaluations of rules distributed by χ^2 with 1 degree of freedom. The limiting distribution of all χ^2 distributions is Fisher-Tippet [7,9,10]. The cumulative distribution function of this distribution is

$$P(x < x_0) = e^{-e^{\frac{\mu - x_0}{\beta}}} \tag{4}$$

where μ and β are parameters of the distribution. Distribution's mean, median, and variance are

$$\text{mean} = \mu + \beta\gamma, \quad \text{median} = \mu - \beta * \ln\ln 2, \quad \text{var} = \pi^2\beta^2/6 \tag{5}$$

where γ is Euler-Mascheroni constant 0.57721. The natural way to compute the parameters μ and β from the sample would be to first estimate the variance from the data and use it to compute β, followed by the estimation of μ from the sample's mean or median. However, error of estimation of variance and mean propagates to estimations of parameters μ and β, where variance is a bigger problem than mean, as it is used for estimation of both parameters.

Gupta [11] showed that for p independent and identically distributed values taken from χ^2 with one degree of freedom, where p is large, the following properties holds for their maxima M:

$$E(M) = 2\ln p - \ln\ln p - \ln\pi + 2\gamma \tag{6}$$

$$m(M) = 2\ln p - \ln\ln p - \ln\pi - 2\ln\ln 2 \tag{7}$$

$$\sigma(M) = \sqrt{2/3\pi^2} \tag{8}$$

Since $\sigma(M)$ is independent of the number of values (or the number of considered rules, in our case), combining 5 and 8 gives $\beta = 2$. We thus only need to estimate the remaining parameter μ. In our algorithm we computed the median from the vector of maximum values, so μ equals the median plus $2\ln\ln 2$.

Automatically Evolving Rule Induction Algorithms

Gisele L. Pappa and Alex A. Freitas

Computing Laboratory
University of Kent
Canterbury, Kent, CT2 7NF, UK
{glp6, A.A.Freitas}@kent.ac.uk
http://www.cs.kent.ac.uk

Abstract. Research in the rule induction algorithm field produced many algorithms in the last 30 years. However, these algorithms are usually obtained from a few basic rule induction algorithms that have been often changed to produce better ones. Having these basic algorithms and their components in mind, this work proposes the use of Grammar-based Genetic Programming (GGP) to automatically evolve rule induction algorithms. The proposed GGP is evaluated in extensive computational experiments involving 11 data sets. Overall, the results show that effective rule induction algorithms can be automatically generated using GGP. The automatically evolved rule induction algorithms were shown to be competitive with well-known manually designed ones. The proposed approach of automatically evolving rule induction algorithms can be considered a pioneering one, opening a new kind of research area.

1 Introduction

Research in the rule induction field has being carried out for more than 30 years and certainly produced a large number of algorithms. However, these are usually obtained from the combination of a basic rule induction algorithm (typically following the sequential covering approach) with new evaluation functions, pruning methods and stopping criteria for refining or producing rules, generating many "new" and more sophisticated sequential covering algorithms.

We cannot deny that these attempts to improve the basic sequential covering approach have succeeded. Among the most successful and popular rule induction algorithms are, for example, CN2 [1] and RIPPER [2]. The CN2 algorithm was developed following the concepts of the successful ID3[3] and AQ algorithms. Its current version is a product of small modifications on its first rule evaluation function, together with the added feature of producing ordered or unordered rules. RIPPER is an improvement of IREP [4], which in turn is an improvement of REP, which was created to improve the performance of pFOIL in noisy domains. From IREP to RIPPER, for instance, the metric to evaluate the rules during the pruning phase and the rule stopping criterion were changed. A post-processing phase was also included to optimize the set of learned rules.

If changing these major components of rule induction algorithms can result in new, significantly better ones, why not keep on trying systematically? Our idea is to do this, but not by using the manual, ad-hoc approach of the previous research in the area. Rather, we propose the ambitious idea of automating the process of designing a rule induction algorithm.

Genetic Programming (GP) [5] is a suitable tool for automatically evolving computer programs. The program evolved by a GP can produce the same solution humans use to solve the target problem; but it can also produce something completely new and perhaps better than the "conventional" manually designed solution. Examples of human-competitive GP solutions can be found at [6].

Automatically evolving a rule induction algorithm "from scratch" would certainly be an extremely hard task for a GP. However, we can provide the GP with background knowledge about the basic structure of rule induction algorithms, making the task more feasible.

Grammar-based GP (GGP) [7] is a special type of GP that incorporates in its search mechanism prior knowledge about the problem being solved. Intuitively, GGP is an appropriated tool to automatically evolve rule induction algorithms.

The motivation to design a GP algorithm for automatically evolving a rule induction algorithm is three-fold. First, although there are various rule induction algorithms available, their accuracy in many important, complex domains is still far from 100%, and it is not clear how much, if any, improvement is still possible with current methods [8]. Hence, extensive research has been done to try to improve the results obtained by current rule induction systems. GP provides an automatic way of performing a global search that evaluates, in parallel, many combinations of elements of rule induction algorithms, which can find new, potentially more effective algorithms.

Second, all current rule induction algorithms were manually developed by a human being, and so they inevitably incorporate a human bias. In particular, the majority of rule induction algorithms select one-attribute-value-at-a-time, in a greedy fashion, ignoring attribute interactions. A machine–developed algorithm could completely change this kind of algorithm bias, since "its bias" would be different from the kind of algorithm bias imposed by a human designer.

At last, it has already been shown that no classification algorithm is the best to solve all kinds of tasks [9]. Therefore, a GP algorithm could be used to generate rule induction algorithms targeting specific sets of data, which share similar statistical features, or even generating a rule induction algorithm tailored for a given data set. It would allow us to generate different classifiers for different types of data, just by changing the training data given to the GP.

In [10] we presented the first concepts about automatically evolving a rule induction algorithm at a high level of abstraction. In this paper, we refine those ideas in much greater detail. The remainder of this paper is organized as follows. Section 2 briefly discusses rule induction algorithms. Section 3 gives an overview of GP and GGP. Section 4 introduces the proposed GGP, and Section 5 reports the results of several computational experiments. Section 6 presents the conclusions and describes future research directions.

2 Rule Induction Algorithms

There are three common strategies used to induce rules from data [11]: (1) The separate and conquer strategy [12]; (2) Generate a decision tree–using the divide and conquer strategy–and then extract one rule for each leaf node of the tree [3]; (3) The use of evolutionary algorithms, like genetic algorithms and genetic programming, to extract rules from data [13,14].

Among these three strategies, the separate and conquer is certainly the most explored. The separate and conquer strategy (also known as sequential covering) learns a rule from a training set, remove from it the examples covered by the rule, and recursively learns another rule that covers the remaining examples, until all or almost all examples are covered. It is the most common strategy used for rule induction algorithms, and the methods based on this approach differ from each other in four main points [15,12], although the last one can be absent:

1. The representation of the candidate rules : propositional or first-order logic.
2. The search mechanisms used to explore the space of candidate rules (Usually a bottom-up, top-down or bi-directional strategy combined with a greedy, beam or best-first search).
3. The way the candidate rules are evaluated, using heuristics such as information gain, information content, Laplace accuracy, confidence, etc.
4. The pruning method, which can be used during the production of the rules (pre-pruning) or in a post processing step (post-pruning) to help avoiding over-fitting and handling noisy data.

These 4 points will be the starting point for the definition of the grammar which the proposed GP will use, as described in Section 4.1.

3 Overview of Genetic Programming

Genetic Programming (GP) [5,16] is an area of evolutionary computation which aims to automatically evolve computer programs. Together with other evolutionary algorithms, its application is successful because of its problem independency, global search and associated implicit parallelism [16].

Essentially, a GP algorithm evolves a population of individuals, where each individual represents a candidate solution to the target problem. These individuals are evaluated using a fitness function, and the fittest individuals are usually selected to undergo reproduction, crossover and mutation operations. The new individuals produced during these processes create a new population, which replaces the old one. This evolution process is carried out until an optimum solution is found, or a pre-established number of generations is reached.

In this work we use a Grammar-based GP (GGP). As the name suggests, the major difference between a GP and a GGP is the definition and use of a grammar. The motivation to combine grammars and GP is two-fold [17]. First, it allows the user to incorporate prior knowledge about the problem domain in the GP, to guide its search. Second, it guarantees the closure property through the definition of grammar production rules.

Grammars are simple mechanisms capable of representing very complex structures. Context Free Grammars (CFG), the focus of this work, can be represented as a four-tuple $\{N, T, P, S\}$, where N is a set of non-terminals, T is a set of terminals, P is a set of production rules, and S (a member of N) is the start symbol. The production rules have the form $x ::= y$, where $x \in N$ and $y \in \{T \cup N\}$.

There are three special symbols used in the notation to write production rules: "|","[ ]" and "( )". "|" represents a choice, like in $x ::= y|z$, where x generates the symbol y or z. "[ ]" wraps an optional symbol which may or may not be generated when applying the rule. "( )" is used to group a set of choices together, like in $x ::= k(y|z)$, where x generates k followed by y or z.

A derivation step is the application of a production rule from $p \in P$ to some non-terminal $n \in N$, and it is represented by the symbol $\Longrightarrow$. Consider the production rules $x ::= yz$ and $y ::= 0|1$. A derivation step starting in x would be represented as $x \Longrightarrow yz$ and $yz \Longrightarrow 0z$.

In the GGP algorithm used in this work, each individual of the population is generated by applying a set of derivation steps from the grammar, guaranteeing that only valid programs (individuals) are generated [7], as detailed in Section 4.

4 Grammar-Based Genetic Programming for Rule Induction

This work proposes the use of Grammar-based Genetic Programming (GGP) to automatically evolve rule induction algorithms. In contrast to projects that use GP to discover a set of *rules for a specific data set*, like [14] and [13], this project aims to automatically invent a *generic rule induction algorithm*, that is, a rule induction algorithm that can be applied to data sets in general, regardless of the application domain. Hence, each individual in our population represents a new rule induction algorithm, potentially as complex as well-known algorithms.

To the best of our knowledge, there has been just two attempts in the literature to use a GGP for improving the design of a sequential covering rule induction algorithm. Wong [18] used a GGP to automatically evolve the evaluation function of the FOIL algorithm. Our work goes considerably beyond that work, as follows. In [18] the GGP was used to evolve only the evaluation function of a rule induction algorithm. By contrast, in our work GGP is used to evolve virtually all components of a sequential covering rule induction algorithm. Hence, the search space for our algorithm is the space of sequential covering rule induction algorithms, whilst the search space for [18]'s GGP is just the space of evaluation functions for FOIL. Suyama *et* al. [19] also used a GP to evolve a classification algorithm. However, the ontology used in [19] has coarse-grained building blocks, where a leaf node of the ontology is a full classification algorithm. By contrast, our grammar is much more fine-grained; its building blocks are programming constructs ("while","if", etc), search strategies and evaluation procedures not used in [19]. Finally, in both [18] and [19], the GP was trained with a single data set, like in any other use of GP for discovering classification rules. By contrast, in this work the GGP is trained with 6 data sets in the same run of the GGP, because the goal is to evolve a truly generic rule induction algorithm, and not just a rule induction algorithm for one particular data set.

The GGP method proposed was implemented as follows. In the first generation of the GGP a population of individuals is created using a grammar. The grammar contains background knowledge about the basic structure of rule induction algorithms following the separate and conquer approach.

Each individual in the population is represented by a derivation tree, built from a set of derivation steps produced by using the grammar. The individuals (rule induction algorithms) are evaluated using a set of data sets, named the meta-training set. The classification accuracies obtained from the runs of the rule induction algorithms represented by the individuals in the meta-training set are used to generate a fitness measure, as will be explained later.

After evaluation, a tournament selection scheme is used to select the individuals for the new population. Before being inserted in the new population, the

Table 1. Grammar definition

```
1-<Start>::=(<CreateRuleList>|<CreateRuleSet>)[<PostProcess>].
2-<CreateRuleSet>::=forEachClass <whileLoop> endFor <RuleSetTest>.
3-<CreateRuleList>::=<whileLoop> <RuleListTest>.
4-<whileLoop>::=while <condWhile> <CreateOneRule> endWhile.
5-<condWhile>::=uncoveredNotEmpty|uncoveredGreater(10TrainEx|20TrainEx|
                90%TrainEx|95%TrainEx|97%TrainEx|99%TrainEx).
6-<RuleSetTest>::=lsContent|laplaceAccuracy.
7-<RuleListTest>::=appendRule|prependRule.
8-<CreateOneRule>::=<InitializeRule><innerWhile>[<PrePruneRule>]
                    <RuleStoppingCriterion>.
9-<innerWhile>::=while(candNotEmpty|negNotCovered)<FindRule>endWhile.
10-<InitializeRule>::=emptyRule|randomEx|typicalEx|<MakeFirstRule>.
11-<MakeFirstRule>::=NumCond1|NumCond2|NumCond3|NumCond4.
12-<FindRule>::=(<RefineRule>|<innerIf>)<EvalRule><StoppingCriterion>
                 <SelectCandidateRules>.
13-<RefineRule>::=<AddCond>|<RemoveCond>.
14-<AddCond>::=Add1|Add2.
15-<RemoveCond>::=Remove1|Remove2.
16-<innerIf> ::=if <condIf> then <RefineRule> else <RefineRule>.
17-<condIf>::=<condIfExamples>|<condIfRule>.
18-<condIfExamples>::=(numCovExpSmaller|numCovExpGreater)(90p|95p|99p).
19-<condIfRule> ::=ruleSizeSmaller(2|3|5|7).
20-<EvalRule>::=accuracy|purity|laplace|infoContent|informationGain.
21-<RuleStoppingCriterion>::=noStop|purityStop|accuracyStop|nCoveredStop.
22-<StoppingCriterion>::=noStop|SignifTest90|SignifTest95|SignifTest99|
                  PurityCrit60|PurityCrit70|PurityCrit80|defaultAccuracy.
23-<SelectCandidateRules>::=1CR|2CR|3CR|4CR|5CR|8CR|10CR.
24-<PrePruneRule>::=Prune1Cond|PruneLastCond|PruneFinalSeqCond.
25-<PostProcess> ::=RemoveRule EvaluateModel|<RemoveCondRule>.
26-<RemoveCondRule>::=(Remove1Cond|Remove2Cond|RemoveFinalSeq)<EvalRule>.
```

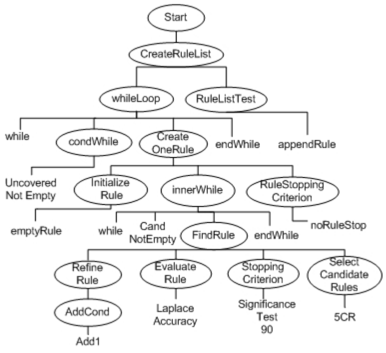


Fig. 1. Example of an GGP Individual (a complete rule induction algorithm)

winners of the tournaments undergo either reproduction, mutation, or crossover operations, depending on user-defined probabilities.

The evolution process is conducted until a maximum number of generations is reached. At the end of the process, the best individual (highest fitness) is returned as the solution for the problem. The chosen rule induction algorithm is then evaluated in a new set of data sets, named the meta-test set, which contains data sets different from the data sets in the meta-training set.

4.1 The Grammar

The grammar is the most important element in a GGP system, since it determines the search space. Table 1 presents the grammar. It uses the terminology introduced in Section 3, and the non-terminal *Start* as its Start symbol. The symbols that appear between "<>" are the grammar non-terminals.

The grammar is made of 26 production rules (PR), each one representing a non-terminal. For simplification purposes, this first version of the grammar does not include all the possible terminals/non-terminals we intend to use, but it is still an elaborate grammar, allowing the generation of many different kinds of rule induction algorithms.

According to PR 1 in Table 1 (*Start*), the grammar can produce either a decision list (where the rules are applied to an unclassified example in the order they were generated) or a rule set (where there is no particular order to apply rules to new examples). The derivation trees which can be obtained applying the production rules of the grammar will create an algorithm following the basic sequential covering approach. However, the non-terminals in the grammar will define how to initialize, refine and evaluate the rules being created. They also specify a condition to stop the refinement of rules and the production of the rule set/list, and define how the search space will be explored.

4.2 The Design of the GGP Components

Individual Representation. In our GGP system, an individual is represented by a derivation tree. This derivation tree is created using a set of production

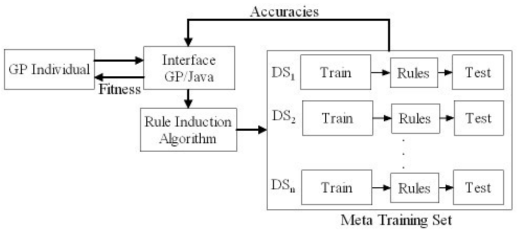


Fig. 2. Fitness evaluation process of a GGP Individual

rules defined by the grammar described in Section 4.1. Recall that an individual represents a complete rule induction algorithm. Figure 1 shows an example of an individual's derivation tree. The root of the tree is the non-terminal *Start*. The tree is then derived by the application of PRs for each non-terminal. For example, PR 1 (*Start*) generates the non-terminal *CreateRuleList*. Then the application of PR 3 produces the non-terminals *whileLoop* and *RuleListTest*. This process is repeated until all the leaf nodes of the tree are terminals.

To extract from the tree the pseudo-code of the corresponding rule induction algorithm, we have to read all the terminals in the tree from left to right. The tree in Figure 1 represents an instance of the CN2 algorithm [1], with the beam-width parameter set to 5 and the significance threshold set to 90%.

The fitness function. Evolution works selecting the fittest individuals of a population to reproduce and generate new offspring. In this work, an individual represents a rule induction algorithm. Therefore, we have to design a fitness function able to evaluate an algorithm RI_A as being better/worse than another algorithm RI_B.

In the rule induction algorithm literature, an algorithm RI_A is usually said to outperform an algorithm RI_B if RI_A has better classification accuracy in a set of classification problems. Hence, in order to evaluate the rule induction algorithms being evolved, we selected a set of classification problems, and created a meta-training set. In the meta-training set, each "meta-instance" represents a complete data set, divided into conventional training and test sets.

As illustrated in Figure 2, each individual in the GP population is decoded into a rule induction algorithm (implemented in Java) using a GP/Java interface. Each terminal in the grammar is associated with a block of Java code. When the evaluation process starts, the terminals in the individual are read, and together they generate a rule induction algorithm.

The Java code is compiled, and the rule induction algorithm is run on all the data sets belonging to the meta-training set. It is a conventional run where, for each data set, a set or list of rules is built using the set of training examples and evaluated using the set of test examples.

After the rule induction algorithm is run on all data sets in the meta-training set, the fitness of the individual is calculated as the average of the values of function f_i for each data set i in the meta training set. The function f_i is defined:

$$f_i = \begin{cases} \frac{Acc_i - DefAcc_i}{1 - DefAcc_i}, \text{ if Acc}_i > \text{DefAcc}_i \\ \frac{Acc_i - DefAcc_i}{DefAcc_i}, \text{ otherwise} \end{cases}$$

In this definition, Acc_i represents the accuracy (on the test set) obtained by the rules discovered by the rule induction algorithm in data set *i*. $DefAcc_i$ represents the default accuracy (the accuracy obtained when using the class of the majority of the examples to classify new examples) in data set *i*. According to the definition of f_i, if the accuracy obtained by the classifier is better than the default accuracy, the improvement over the default accuracy is normalized, by dividing the absolute value of the improvement by the maximum possible improvement. In the case of a drop in the accuracy with respect to the default accuracy, this difference is normalized by dividing the negative value of the difference by the maximum possible drop (the value of $DefAcc_i$).

Hence, f_i returns a value between -1 (when $Acc_i = 0$) and 1 (when $Acc_i = 1$). The motivation for this elaborate fitness function, rather than a simpler fitness function directly based only on Acc_i (ignoring $DefAcc_i$) is that the degree of difficulty of the classification task depends strongly on the value of $DefAcc_i$. The above fitness function recognizes this and returns a positive value of f_i when $Acc_i > DefAcc_i$. For instance, if $DefAcc_i$= 0.95, then Acc_i=0.90 would lead to a negative value of f_i, as it should.

Crossover and Mutation Operators. In GGP, the new individuals produced by crossover and mutation have to be consistent with the grammar. For instance, when performing crossover the system cannot select a subtree *EvaluateRule* to be exchanged with a subtree *SelectCandidateRules*. Therefore, crossover operations have to exchange subtrees whose roots contain the same non-terminal, apart from *Start*. Crossing over two individuals swapping the subtree rooted at *Start* (actually, the entire tree) would generate exactly the same two individuals, and so it would be useless.

Mutation can be applied to a subtree rooted at a non-terminal or applied to a terminal. In the former case, the subtree undergoing mutation is replaced by a new subtree, produced by keeping the same label in the root of the subtree and then generating the rest of the subtree by a new sequence of applications of production rules, so producing a new derivation subtree. When mutating terminals, the terminal undergoing mutation is replaced by another "compatible" symbol, i.e., a (non-)terminal which represents a valid application of the production rule whose antecedent is that terminal's parent in the derivation tree.

5 Results and Discussion

The experimentation phase started with the definition of the meta-training and meta-test sets mentioned in Section 4.2. The current version of the system does not support continuous attributes. Hence, we used 10 public domain data sets having only categorical attributes. Out of the 10 data sets available, we arbitrarily chose 6 for the meta-training set and the other 4 for the meta-test set.

Table 2. Data sets used in the meta-training set

Data set	Examples	Attributes	Classes
Monks-2	169/432	6	2
Monks-3	122/432	6	2
Balance-scale	416/209	4	3
Tic-tac-toe	640/318	9	2
Lymph	98/50	18	4
Zoo	71/28	16	7

Table 3. Data sets used in the meta-test set

Data set	Examples	Attributes	Classes
Monks-1	556	6	2
Mushroom	8124	23	2
Promoters	106	58	2
Wisconsin	683	9	2
Splice	3190	63	3

Tables 2 and 3 show respectively the data sets used in the meta-training and meta-test sets. During the evolution of the rule induction algorithm by the GGP, for each data set in the meta-training set, each candidate rule induction algorithm (individual) was trained with 70% of the examples, and then tested in the remaining 30%. In order to avoid overfiting, these sets of data were merged and then randomly divided in 70-30% for each of the generations of the GGP. In Table 2, the figures in the column *Examples* indicate the number of instances in the training/test sets used by each rule induction algorithm during the GGP run, respectively. In the meta-test set, data sets were processed using a 5-fold cross validation process. Hence, in Table 3, *Examples* indicates the total number of examples in the data set.

Once the meta data sets have been created, the next step was the choice of the GGP parameters. Population size was set to 100 and the number of generations to 30. These two numbers were empirically chosen based on preliminary experiments, but are not optimized. Considering crossover, mutation and reproduction probabilities, GPs usually use a high probability of crossover and low probabilities of mutation and reproduction. However, the balance between these three numbers is an open question, and may be very problem dependent [16]. In our experiments we set the reproduction probability as 0.05, and vary the balance between the crossover and mutation probabilities in order to choose appropriate values for these parameters. The empirically adjusted values were 0.8 crossover probability and 0.15 mutation probability. Sections 5.1 and 5.2 report the results obtained for the meta-training and meta-test sets respectively.

5.1 Results in the Meta-training Set

First, we report results about the accuracy of the evolved rule induction algorithms in the test set of each of the data sets in the meta-training set. It should be stressed that this is not a measure of "predictive accuracy" because each test set in the meta-training set was seen during the evolution of the GGP. Nonetheless, the accuracy on the test sets of the meta-training set is useful to evaluate the success of the training of the GGP, and so it is reported here.

Table 4 shows the default accuracy (accuracy obtained when using the class of the majority of the examples to classify any new example) of the data sets in the meta-training set, followed by the results of runs of CN2-Unordered and CN2-

Table 4. Accuracy rates (%) for the Meta-training set

Data set	Def.	CN2Un	CN2Ord	RIPPER	C4.5R	GGP-RI
Monks-2	67.1	67.1	72.9	62.5	69.4	85.5±0.56
Monks-3	52.7	90.7	93.3	90.28	96.3	98.16±0.38
Balance-scale	45.9	77.5	81.3	77.03	78	80.48±0.68
Tic-tac-toe	65.4	99.7	98.7	98.43	100	96.16±1.14
Lymph	54	80	82	76	88	76.26±1.83
Zoo	43.3	96.7	96.7	90	93.3	99.34±0.66

Table 5. Accuracy rates (%) for the Meta-test set

Data set	Def.	CN2Un	CN2Ord	RIPPER	C4.5R	GGP-RI
Monks-1	50	100±0	100±0	93.84±2.93	100±0	100±0
Mushroom	51.8	100±0	100±0	99.96±0.04	98.8±0.06	99.99±0.01
Promoters	50	74.72±4.86	81.9±4.65	78.18±3.62	83.74±3.46	78.83±2.16
Wisconsin	65	94.16±0.93	94.58±0.68	93.99±0.63	95.9±0.56	94.54±0.56
Splice	51.8	74.82±2.94	90.32±0.74	93.88±0.41	89.66±0.78	89.24±0.32

Ordered (using default parameters), RIPPER and C4.5 Rules. These results are baselines against which we compare the accuracy of the rule induction algorithms evolved by the GGP. Table 4 also reports the results obtained by the GGP-RI (Rule Induction algorithms evolved by the GP).

In Table 4 the numbers after the symbol "±" are standard deviations. Results were compared using a statistical t-test with confidence level 0.05. Cells in dark gray represent winnings of GGP-RI against a baseline method, while light gray cells represent GGP-RI losses. In total, Table 4 contains 24 comparative results between GGP-RI and baseline methods – 6 data sets × 4 classification algorithms. Out of theses 24 cases, the accuracy of GGP-RI was statistically better than the accuracy of the baseline methods in 15 cases, whilst the opposite was true in only 5 cases. In the other 4 cases there was no significant difference.

5.2 Results in the Meta-test Set

The results obtained by the GGP-RIs for the data sets in the meta-training set were expected to be competitive with other algorithms, since the GGP evolved rule induction algorithms based on the data sets in that meta-training set. The challenge for the GGP is to evolve rule induction algorithms that obtain at least a competitive performance for data sets in the meta-test set, which were not used during the evolution of the rule induction algorithm. As in the previous section, Table 5 shows the default accuracy and the accuracies obtained by baseline methods in the data sets in the meta-test set, followed by the results obtained by the GGP-RI. Recall that in the meta-test set every algorithm was run using a 5-fold cross-validation procedure, and the results reported are the average accuracy on the test set over the 5 iterations of the cross-validation procedure.

As shown in Table 5, most of the results obtained in the 5 data sets used in the meta-test set are statistically the same as the ones produced by the baseline methods. The only exceptions are *Mushroom* and *Splice*. In both data sets, GGP-RI gets statistically better results than one of the baseline methods. *Splice* is the only data set in which RIPPER produces a better result than GGP-RI.

One of the main goals of this project was to automatically evolve rule induction algorithms that perform as well or better than human designed rule induction algorithms. Another goal was to automatically produce a rule induction algorithm different from human-designed ones. Out of the 5 GGP-RI discovered (in the 5 runs of the GGP with different random seeds), the one most different from the human designed ones can be summarized as follows.

It searches for rules starting from an example chosen from the training set using the typicality concept [20], and removes 2 conditions at a time from it (bottom-up approach). It evaluates rules using the Laplace accuracy and stops refining them when the rules' accuracy is smaller then 70%.

6 Conclusions and Future Work

This work showed that effective rule induction algorithms can be automatically generated using Genetic Programming. The automatically evolved rule induction algorithms were show to be competitive with well-known manually designed (and refined over decades of research) rule induction algorithms. The proposed approach of automatically evolving rule induction algorithms can be considered a pioneering one, opening a new kind of research area, and so there are still many problems to be solved.

One research direction is to create a more powerful version of the grammar, which could potentially lead to the discovery of more innovative rule induction algorithms. Another possible research direction is to design a fitness function that considers not only the accuracy of the rules discovered by the rule induction algorithms, but also a measure of the size of the discovered rule set. However, this introduces the problem of coping with the trade-off between accuracy and rule set simplicity in the fitness function, an open problem.

Yet another possible research direction is to evolve rule induction algorithms for specific kinds of data sets. Instead of using very different kinds of data sets in the meta-training set, we can assign to the meta-training set several data sets that are similar to each other, according to a pre-specified criterion of similarity. Then, in principle, the GGP algorithm would evolve a rule induction algorithm particularly tailored for that kind of data set, which should maximize the performance of the rule induction algorithm in data sets of the same kind, to be used in the meta-test set. However, this introduces the problem of defining a good measure of similarity between the data sets: an open problem. It is also possible to evolve a rule induction algorithm tailored for one given data set, as in [18,19].

Acknowledgments

The first author is financially supported by CAPES, a Brazilian research-funding agency, process number 165002-5.

References

1. Clark, P., Boswell, R.: Rule induction with cn2: some recent improvements. In: EWSL-91: Proc. of the Working Session on Learning on Machine Learning. (1991)
2. Cohen, W.W.: Fast effective rule induction. In: Proc. of the 12^{th} International Conference on Machine Learning. (1995)
3. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann (1993)
4. Furnkranz, J., Widmer, G.: Incremental reduced error pruning. In: Proc. the 11^{th} Int. Conf. on Machine Learning, New Brunswick, NJ (1994) 70–77
5. Koza, J.R.: Genetic Programming: On the Programming of Computers by the means of natural selection. The MIT Press, Massachusetts (1992)
6. Koza, J.: http://www.genetic-programming.org/. (June, 2006)
7. Whigham, P.A.: Grammatically-based genetic programming. In: Proc. of the Workshop on GP: From Theory to Real-World Applications. (1995)
8. Domingos, P.: Rule induction and instance-based learning: A unified approach. In: Proc. of the 14^{th} International Joint Conference on Artificial Intelligence. (1995) 1226–1232
9. Lim, T., Loh, W., Shih, Y.: A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Machine Learning **40**(3) (2000) 203–228
10. Pappa, G.L., Freitas, A.A.: Towards a genetic programming algorithm for automatically evolving rule induction algorithms. In Furnkranz, J., ed.: Proc. ECML/PKDD-2004 Workshop on Advances in Inductive Learning. (2004) 93–108
11. Mitchell, T.: Machine Learning. Mc Graw Hill (1997)
12. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (1999)
13. Freitas, A.A.: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer-Verlag (2002)
14. Wong, M.L., Leung, K.S.: Data Mining Using Grammar-Based Genetic Programming and Applications. Kluwer (2000)
15. Furnkranz, J.: Separate-and-conquer rule learning. Artificial Intelligence Review **13**(1) (1999) 3–54
16. Banzhaf, W., Nordin, P., Keller, R., Francone, F.: GP – An Introduction; On the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann (1998)
17. O'Neill, M., Ryan, C.: Grammatical Evolution Evolutionary Automatic Programming in an Arbitrary Language. Morgan Kaufmann (2003)
18. Wong, M.L.: An adaptive knowledge-acquisition system using generic genetic programming. Expert Systems with Applications **15**(1) (1998) 47–58
19. Suyama, A., Negishi, N., Yamaguchi, T.: CAMLET: A platform for automatic composition of inductive learning systems using ontologies. In: Pacific Rim International Conference on Artificial Intelligence. (1998) 205–215
20. Zhang, J.: Selecting typical instances in instance-based learning. In: Proc. of the 9^{th} Int. Workshop on Machine Learning. (1992)

Bayesian Active Learning for Sensitivity Analysis

Tobias Pfingsten[1,2]

[1] Robert Bosch GmbH, Stuttgart, Germany
[2] Max Planck Institute for Bioligical Cybernetics
Tobias.Pfingsten@de.bosch.com

Abstract. Designs of micro electro-mechanical devices need to be robust against fluctuations in mass production. Computer experiments with tens of parameters are used to explore the behavior of the system, and to compute sensitivity measures as expectations over the input distribution. Monte Carlo methods are a simple approach to estimate these integrals, but they are infeasible when the models are computationally expensive. Using a Gaussian processes prior, expensive simulation runs can be saved. This Bayesian quadrature allows for an active selection of inputs where the simulation promises to be most valuable, and the number of simulation runs can be reduced further.

We present an active learning scheme for sensitivity analysis which is rigorously derived from the corresponding Bayesian expected loss. On three fully featured, high dimensional physical models of electro-mechanical sensors, we show that the learning rate in the active learning scheme is significantly better than for passive learning.

1 Introduction

Before computational power was widely available, general purpose simulation software hardly existed and computer models were largely tailored to answer specific questions. Today, computer models are often one-to-one emulations of physical systems and describe all their relevant features. They are usually built using powerful simulation tools—which use e.g. finite element methods to model electro-mechanical properties—and do therefore not necessarily lead to a better understanding of the system. They are used in computer experiments to replace experimental specimens.

In industrial engineering these computer experiments are often used to estimate the robustness of a design with respect to unavoidable fluctuations in mass production. Especially in the production of micro electro-mechanical systems (MEMS) such variations can significantly affect the devices' functionality. Sensitivity analysis (SA) is a standard procedure in the designing process of MEMS, and computer experiments are used to determine the influence of input parameters on the resulting fluctuation in the output. A comprehensive discussion of SA is given by [1,2].

When fluctuations are small, SA can be done using a local approximation such as linearization. However, when this assumption does not hold, the response of the software needs to be explored over the whole range of parameter settings, and the sensitivity measures are given as expectations over the input distribution. Realistic models have tens of input parameters and the function cannot be evaluated on a regular grid. Hence, Monte Carlo (MC) methods are the most common approach for computing the expectations, where random samples from the input distribution replace the grid.

The convergence rate for MC methods is independent of the function's smoothness and the input dimension. This is certainly a useful property, but if we can use prior information—e.g. when we know that the output is a smooth function of the input parameters—we can make more efficient use of the data and save valuable simulation runs. O'Hagan [3] proposes what he calls *Bayesian quadrature*, using a Gaussian process to model the output of the simulation software, e.g. for SA [4]. Previous works have shown that, compared to MC, Bayesian quadrature can significantly improve the accuracy using the same number of randomly sampled simulation runs [2,5].

In Bayesian quadrature we are not restricted to using samples from the input distribution and we can thus evaluate the model where the output promises to be most informative. Random sampling corresponds to what is called *passive learning* in machine learning. Actively choosing promising inputs is known as *active learning*, which has been discussed as early as 1956 by Lindley [6]. However, Bayesian active learning—also called Bayesian *experimental design*—is computationally demanding, and naturally depends strongly on what is defined to be "optimal". Therefore it cannot be considered a solved problem.

In this work we present an active learning scheme for nonparametric Gaussian process regression used in Bayesian quadrature. The learning scheme is derived as a greedy approximation to the optimal Bayesian design, where model parameters are updated after each query. We minimize the average predictive variance in the region of interest, which is derived in closed form for uniform and Gaussian input distributions. We show on three fully featured simulations of micro electromechanical sensors that the active learning scheme significantly outperforms passive learning in terms of learning rate.

We outline the Bayesian approach to active learning in section 2, discussing its relation to experimental design. Based on the generic concepts we derive a sampling scheme for sensitivity analysis in section 3. We compare the performance of passive and active learning in several experiments in section 4 and discuss the results in section 5.

2 Bayesian Active Learning

In the following section we discuss Bayesian active learning. Section 2.1 introduces the Bayesian concept of expected utility, which provides the formal framework for experimental design. We show in 2.2 how experimental design corresponds to active learning and define the algorithm which we use for our experiments. For sensitivity analysis, just as for other regression setups, the objective is to minimize the expected generalization error in a region of interest. We define the corresponding utility function in 2.3.

2.1 The Expected Utility

Active learning is a typical example for problems which can be solved using Bayesian decision theory, where the purpose of the experiments is expressed in a utility function. The utility function will usually depend on uncertain quantities, such as model parameters and the outcomes of the experiments which are to be performed. We can therefore not directly optimize the utility function and need to average over these quantities according to our prior belief. After averaging out the unknowns, the utility function is called the *Bayesian expected utility.* Berger [7, Chap. 4] gives a comprehensive discussion of Bayesian decision theory, [8] and [9] review its application to optimal experimental design.

Assume our aim is to collect N samples, where we can choose inputs $\mathbf{x} \in \mathbb{R}^D$ at which we query targets $y \in \mathbb{R}$. We collect the targets in a vector $\mathbf{y} = (y_1, y_2 \dots y_N)^T$ and the inputs in the *design matrix* $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_N)^T$. We collect both in the *dataset* $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$. To refer to the available data at time t of decision making we use the same symbol with a time index, $\mathcal{D}_t$.

Before any optimal sampling scheme can be computed, we need to specify what is to be meant by "optimal", i.e. we need to define some *utility function*

$$U(\mathbf{X}, \mathbf{y}, \boldsymbol{\theta} | \mathcal{D}_o) \ . \tag{1}$$

The utility function usually depends on the design matrix $\mathbf{X}$ which is chosen to maximize U, the unknown outcomes of the experiments $\mathbf{y}$, and the unknown model parameters $\boldsymbol{\theta}$. The objective may also depend on the model assumption and prior information $\mathcal{D}_o$. In contrast to the remaining quantities, $\mathcal{D}_o$ is fixed.

As mentioned above, we can usually quantify the utility of a design matrix only after observing the outcomes of the experiments and for given model parameters. Bayesian decision theory provides the formalism to handle these uncertain quantities: they need to be integrated out, using the prior distribution which corresponds to $\mathcal{D}_o$,

$$U(\mathbf{X}|\mathcal{D}_o) = \int \mathrm{d}\mathbf{y} \int \mathrm{d}\boldsymbol{\theta} \quad U(\mathbf{X}, \mathbf{y}, \boldsymbol{\theta}|\mathcal{D}_o) \quad \underbrace{p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \mathcal{D}_o)}_{\text{model}} \ \underbrace{p(\boldsymbol{\theta}|\mathbf{X}, \mathcal{D}_o)}_{\text{prior}} \ . \tag{2}$$

As for the utility function we use the symbol U for the *expected utility*, simply omitting those arguments over which we have averaged.

Note, that in the expected utility (2) we assume that our prior assumptions are correct: As we average over the predictive distribution of the model $p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \mathcal{D}_o)$ and the prior $p(\boldsymbol{\theta}|\mathbf{X}, \mathcal{D}_o)$, the loss does not account for unexpected parameter settings or measurements which cannot be explained by the model. MacKay [10] argues that this is the "Achilles' heel" of active learning. As we assume that we are completely certain about the model, an active learning scheme might tend to choose extreme designs which automatically confirm the model. O'Hagan discusses this problem in [11], where he introduces Gaussian processes as localized linear models. GPs relax the hard assumptions of parametric models, which can lead to designs with samples only at the limits of the input domain.

2.2 Greedy Scheme for Active Learning

Experimental design has traditionally been used to determine a complete optimal design of N samples before any experiments are performed. The main issue is to find approximately optimal designs for large N, as the exact problem is NP-hard [12]. In the machine learning community the term "active learning" has replaced "experimental design". The focus has moved from planning a whole batch of experiments to actively planning the experiments one after the other, while updating the learning algorithm after each query. Although these approaches are quite different in their goal, both are optimally solved by maximizing the expected utility in (2).

In classical experimental design the queries $\mathbf{X}$ are planned as a batch, maximizing the expected utility $U(\mathbf{X}|\mathcal{D}_o)$. If we assume that we obtain the outcomes of all experiments at once the solution is optimal. However, if the results come one-by-one, we should refine the remaining experimental schedule in each step ℓ, by considering the measured $y_1, y_2 \dots y_\ell$ in the prior belief $\mathcal{D}_\ell$ at that time. In the Bayesian formalism it is clear that this information is correctly considered by maximizing $U(\mathbf{x}_{\ell+1} \dots \mathbf{x}_N|\mathcal{D}_\ell)$ in each query.

Most active learning schemes avoid the computational burden of planning all remaining experiments by greedily planning only one step ahead, optimizing the expected utilities $U(\mathbf{x}_{\ell+1}|\mathcal{D}_\ell)$. We use this query scheme for our experiments:

Algorithm 1. Greedy active learning

```
Require: $N_o$ initial samples $\mathcal{D}_{N_o}$.
1: for $\ell = N_o + 1$ to $N$ do
2:     find $\mathbf{x}_\ell \leftarrow \mathrm{argmax}_{\mathbf{x}} U(\mathbf{x}|\mathcal{D}_{\ell-1})$.
3:     query target $y_\ell$ to obtain new dataset $\mathcal{D}_\ell \leftarrow \mathcal{D}_{\ell-1} \cup \{(\mathbf{x}_\ell, y_\ell)\}$.
4: end for
```

2.3 Predictive Performance in a Region of Interest

The utility function (1) formalizes the goal of the experimenter and may thus vary from problem to problem. Our aim in sensitivity analysis is to explore the output of the computer code in a region of interest, which is given by an input distribution $p(\mathbf{x})$. To measure the generalization error of the model we use its predictive variance, averaged over $p(\mathbf{x})$. Integrating out the unseen targets $\mathbf{y}$, we obtain

$$U(\mathbf{X}, \boldsymbol{\theta}|\mathcal{D}_o) = \underbrace{\int \mathrm{d}\mathbf{y}\, p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \mathcal{D}_o)}_{\substack{\text{average over unseen} \\ \text{training targets}}} \quad \underbrace{\int \mathrm{d}\mathbf{x}\, p(\mathbf{x})}_{\substack{\text{average over} \\ \text{region of interest}}} \quad \underbrace{\Big[- \mathrm{var}[y|\mathbf{x}, \boldsymbol{\theta}, \mathcal{D}, \mathcal{D}_o] \Big]}_{\substack{\text{objective: (negative)} \\ \text{pred. uncertainty}}} . \tag{3}$$

MacKay [10] discusses several utility functions to measure the generalization error in a region of interest. For the linear model information-based measures lead to so-called alphabetical designs [6,13,8]. While (3) is usually approximated e.g. using a sum over a pool of test cases, our setup allows for an exact solution. We derive the expected utility in the following section.

3 Active Learning for Nonlinear Sensitivity Analysis

In the preceding section we have outlined the generic concept of Bayesian active learning. We adapt the concepts to the application of Bayesian quadrature for sensitivity analysis (SA), deriving the resulting optimal sampling scheme in this section: We briefly outline SA and Bayesian quadrature in 3.1 and introduce Gaussian process regression in 3.2. We show in 3.3 how the Bayesian expected utility for SA can be derived in closed form.

3.1 Bayesian Quadrature for Sensitivity Analysis

Global SA. The simulation software itself can be seen as a deterministic mapping from the input parameters $\mathbf{x}$ to an output $f(\mathbf{x})$. In combination with a known input distribution—which resembles fluctuations in mass production—the model can be used to determine the corresponding output distribution and the influence of single parameters.

Whenever the fluctuations are not small enough to use a local approximation of f around the nominal value, we need to use a global analysis which explores the model over the complete range of $p(\mathbf{x})$. Global sensitivity measures are thus based on expectations of the type

$$I[f] = \int \mathrm{d}\mathbf{x}\, p(\mathbf{x})\; F[f(\mathbf{x})] \,, \tag{4}$$

where F is some functional of f [1]. A first step is to compute the mean and variance of the output distribution, which are the basis for most sensitivity measures:

$$\mathrm{E}_{\mathbf{x}}[f] = \int \mathrm{d}\mathbf{x}\, p(\mathbf{x})\; f(\mathbf{x}) \qquad \text{and} \quad \mathrm{var}_{\mathbf{x}}[f] = \int \mathrm{d}\mathbf{x}\, p(\mathbf{x})\; f^2(\mathbf{x}) - \mathrm{E}_{\mathbf{x}}[f]^2 \;. \tag{5}$$

Classical quadrature and Monte Carlo. The integrals (4) can be evaluated using classical quadrature if the input space is low dimensional. The error of the trapezoidal rule, for example, scales as $\mathcal{O}(N^{-2/D})$ for $F[f] \in \mathcal{C}^2$. For higher dimensions Monte Carlo (MC) estimates are to be preferred. The MC approximation to the integrals (4) is the empirical mean,

$$I[f] \approx \tfrac{1}{N} \sum_{\ell=1}^{N} F[f(\mathbf{x}_\ell)] \,, \quad \text{over samples } \mathbf{x}_\ell \text{ from } p(\mathbf{x}). \tag{6}$$

MC methods are characterized by probabilistic error bounds which scale as $\mathcal{O}(N^{-1/2})$. The MC bounds are independent of the dimension D and only require $F[f]$ to be integrable [14]. Hence, MC outperforms classical quadrature for $D \geq 5$.

Bayesian quadrature. As MC hardly makes any assumption about $F[f]$, it guarantees convergence in almost all cases. However, it is clear that the convergence rate could be better if we were able to incorporate prior knowledge into the estimates. Especially in machine learning such a trade off between bias and variance is well known to lead to a drastic improvement of the learning rate. O'Hagan [15] discusses this potential improvement of MC estimates, claiming that "Monte Carlo is fundamentally unsound".

O'Hagan [3] describes what he calls "Bayesian quadrature" to improve classical quadrature using a Gaussian process (GP) prior. Rasmussen and Williams [5] propose the "Bayesian Monte Carlo" method and show that it can outperform classical MC in high dimensions. The Bayesian quadrature scheme is:

Algorithm 2. Baysian quadrature

```
Require: simulation runs $\mathcal{D}$, possibly from an optimal design
 1: train a Gaussian process to estimate $f$.
 2: use the posterior $p(f|\mathcal{D})$ to estimate the integral $I[f]$ (4):
        $$p(I|\mathcal{D}) = \int \mathrm{d}f \; p(f|\mathcal{D}) \; \left[ \int \mathrm{d}\mathbf{x} \, p(\mathbf{x}) \; F[f(\mathbf{x})] \right] .$$
    The posterior distribution for the integral $p(I|\mathcal{D})$ includes the remaining uncertainty. For SA the integrals for mean and variance (5) can be solved analytically.
```

Recall the error bounds of the trapezoidal rule and the MC method: MC hardly assumes any structure in f and its error scales as $\mathcal{O}(N^{-1/2})$ according to the strong law of large numbers. In contrast, the trapezoidal rule assumes that the function is twice differentiable and uses linear interpolation. The error $\mathcal{O}(N^{-2/D})$, guaranteed by the Taylor expansion, is better than for MC in up to four dimensions, as more structure of the function is used.

The GP regression used in Bayesian quadrature can uncover the structure of functions in high dimensional spaces, and we can therefore expect to extend the favorable convergence rate to quadrature in higher dimensions. However, the improvement comes with the cost of restricting the method to functions covered by the GP prior.

3.2 Gaussian Processes Applied to Bayesian Quadrature

GP regression. A comprehensive introduction to GPs can be found in [16]. In the following we briefly outline the basic concepts. GPs are now widely used in machine learning, however, the model is long known for interpolation in computer experiments [17,18].

Assume we model a mapping f from the input parameters $\mathbf{x} \in \mathbb{R}^D$ to an output $f(\mathbf{x}) \in \mathbb{R}$. Gaussian processes are defined by a (parameterized) mean and covariance function

$$\mathrm{E}[f(\mathbf{x})] = \mu(\mathbf{x}) \qquad \text{and} \qquad \mathrm{cov}\left[f(\mathbf{x}), f(\mathbf{x}')\right] = k(\mathbf{x}, \mathbf{x}') , \tag{7}$$

which model a known main contribution (mean) and deviations, whose structure is defined by the covariance function. We set the mean function to zero for notational simplicity, as this does not make any conceptual difference.

The GPs' behavior is governed by the choice of the covariance function $k(\mathbf{x}, \mathbf{x}')$. A common choice is to assume that correlations between the function values decay exponentially, i.e.

$$k(\mathbf{x}, \mathbf{x}') = w_o^2 \exp\left\{ -\tfrac{1}{2} \left[(\mathbf{x} - \mathbf{x}')^T A^{-1} (\mathbf{x} - \mathbf{x}') \right] \right\} \tag{8}$$

with $A = \mathrm{diag}(w_1^2, \ldots, w_d^2)$. The parameters w_o and $w_1 \ldots w_D$ control the strength of the correlations and the typical length scales of the individual input dimensions. We collect the parameters in a vector $\boldsymbol{\theta} = (w_o \ldots w_D)$.

Bayes' rule is used to combine observed data with the GP prior $p(f|\boldsymbol{\theta})$. Let the observed data $\mathcal{D}$ consist of a set of N possibly noisy observations y_ℓ of function values $f(\mathbf{x}_\ell)$. We assume normal noise, i.e. $y_\ell = f(\mathbf{x}_\ell) + \epsilon_\ell$ with $\epsilon \sim \mathcal{N}(\epsilon|0, \sigma_y{}^2)$, and add the unknown variance $\sigma_y{}^2$ to the parameter vector $\boldsymbol{\theta}$. The predictive distribution at unseen inputs $\mathbf{x}^*$ is

$$p(f^*|\mathbf{x}^*, \mathcal{D}, \boldsymbol{\theta}) = \mathcal{N}\left(f^*|m(\mathbf{x}^*), v(\mathbf{x}^*)\right) \tag{9a}$$

$$\text{with mean} \quad m(\mathbf{x}^*) = \mathbf{k}(\mathbf{x}^*)^T Q^{-1} \mathbf{y} \tag{9b}$$

$$\text{and variance} \quad v(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^T Q^{-1} \mathbf{k}(\mathbf{x}^*) \ ,$$

where we have defined $Q = K + \mathrm{diag}[\sigma_y{}^2, \ldots, \sigma_y{}^2]$, and used the abbreviations $\mathbf{k}(\mathbf{x}^*) \in \mathbb{R}^N$ and $K \in \mathbb{R}^{N \times N}$ with $[\mathbf{k}(\mathbf{x}^*)]_\ell = k(\mathbf{x}_\ell, \mathbf{x}^*)$ and $K_{i\ell} = k(\mathbf{x}_i, \mathbf{x}_\ell)$. As described in [7, Chap. 3] we handle the hyper parameters $\boldsymbol{\theta}$ using the maximum likelihood II (ML-II) approach, which replaces the posterior distribution for the parameters by

$$p(\boldsymbol{\theta}|\mathcal{D}) \approx \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \qquad \text{with} \quad \hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\mathrm{argmax}} \left[p(\mathcal{D}|\boldsymbol{\theta}) \right] \ . \tag{10}$$

Bayesian quadrature. Having computed the posterior process $p(f|\mathcal{D}, \hat{\boldsymbol{\theta}})$, we can estimate mean or variance of the output under $p(\mathbf{x})$ (5) using the predictive mean and variance (9b) of the GP. The integral can be reduced to integrating products of the input distribution $p(\mathbf{x})$ and the covariance function. All necessary integrals can be computed in closed form if the covariance function (8) is used and the input distribution is Gaussian, $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{x}_o, B)$, or uniform. A Gaussian input distribution can almost always be assumed for sensitivity analysis, as it describes natural fluctuations in mass production (strong law of large numbers). The uniform distribution is appropriate for plain regression setups. The derivation of the analytic expressions is given in [2,4].

3.3 Active Learning for Bayesian Quadrature

In 3.1 we have argued why Bayesian quadrature uses the available data more efficiently than MC. In the following we discuss the optimal design which improves the convergence of Bayesian quadrature.

There is a large amount of work on the design of computer experiments [19,18,17]. Most work reports on methods of constructing space filling designs such as the Latin Hypercube design [20] for space filling in low dimensional projections, or the *MaxiMin* and *MiniMax* criteria [21]. These designs partly correspond to special cases of Bayesian optimal designs, but they are mostly based on intuitive considerations. The problem of computing uniform designs in high dimensional spaces is well studied. However, how to learn an appropriate distance measure and how to treat the input distribution is often not clear. Space filling designs are used in *quasi Monte Carlo* methods to improve the MC bounds by minimizing the *discrepancy* [14]. However, they are still limited to (dependent) samples from $p(\mathbf{x})$.

For Bayesian sensitivity analysis we can do more than space filling, as we explicitly know the input distribution which is naturally given by the fluctuations in mass-production. Based on the expected predictive variance over $p(\mathbf{x})$ (3) we derive a Bayesian optimal design which is exact other than using the greedy scheme (algorithm 1):

$U(\mathbf{x}_\ell|\mathcal{D}_{\ell-1})$ is given as an integral over the unknown quantities y_ℓ and $\boldsymbol{\theta}$ (2) and the input distribution $p(\mathbf{x})$ (3). The average over the parameters $\boldsymbol{\theta}$ is trivial in the ML-II framework (10) where we integrate over a δ-distribution around $\hat{\boldsymbol{\theta}}$. The integral over $p(y_\ell|\mathbf{x}_\ell, \hat{\boldsymbol{\theta}}, \mathcal{D}_{\ell-1})$ collapses as the predictive variance (9b) is independent of y_ℓ. We are left with the integral

$$U(\mathbf{x}_\ell|\mathcal{D}_{\ell-1}) = \int \mathrm{d}\mathbf{x}\, p(\mathbf{x}) \left[-\operatorname{var}\left[y|\mathbf{x}, \mathcal{D}_\ell, \hat{\boldsymbol{\theta}} \right] \right] , \tag{11}$$

which can be solved analytically. For notational simplicity we compute the utility for adding a sample $\tilde{\mathbf{x}}$ to the dataset $\mathcal{D}$ and use the definitions in (9).
The change in the predictive variance is[1]

$$\operatorname{var}[y|\mathbf{x}, \mathcal{D}, (\tilde{\mathbf{x}}, \tilde{y})] - \operatorname{var}[y|\mathbf{x}, \mathcal{D}] = -\frac{\left[k(\mathbf{x}, \tilde{\mathbf{x}}) - \mathbf{k}(\mathbf{x})^T Q^{-1} \mathbf{k}(\tilde{\mathbf{x}})\right]^2}{\operatorname{var}[\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}]} , \tag{12}$$

which, through integrating over $p(\mathbf{x})$, leads to

$$\begin{aligned} U(\tilde{\mathbf{x}}|\mathcal{D}) &= \text{const} + \int \mathrm{d}\mathbf{x}\, p(\mathbf{x})\, \frac{\left[k(\mathbf{x}, \tilde{\mathbf{x}}) - \mathbf{k}(\mathbf{x})^T Q^{-1} \mathbf{k}(\tilde{\mathbf{x}})\right]^2}{\operatorname{var}[\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}]} \qquad (13) \\ &= \text{const} + \frac{\left[l(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) + \left(Q^{-1}\mathbf{y}\right)^T L \left(Q^{-1}\mathbf{y}\right) - 2\left(Q^{-1}\mathbf{y}\right)^T \mathbf{l} \right]}{\operatorname{var}[\tilde{y}|\tilde{\mathbf{x}}, \mathcal{D}]} . \end{aligned}$$

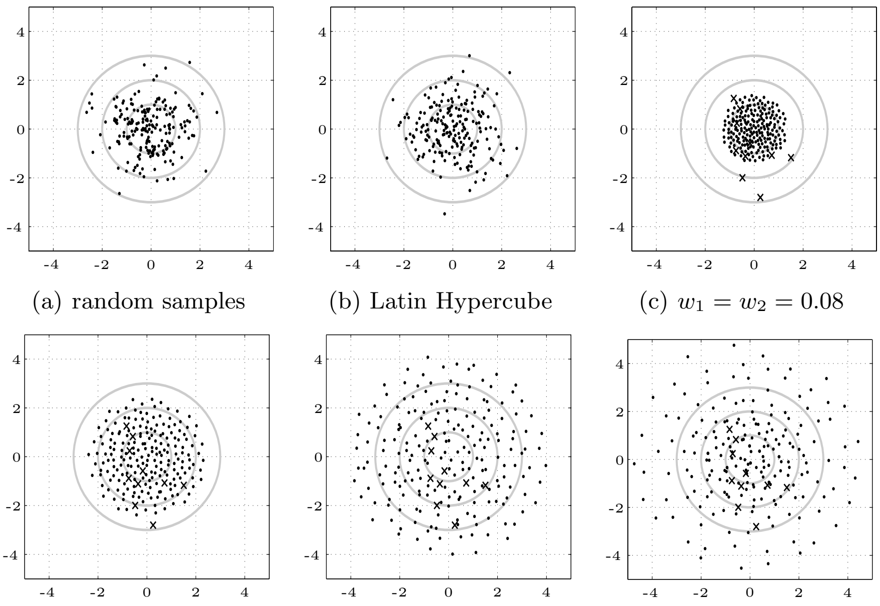
As an integral over a product of Gaussians[2] $l(\mathbf{x}', \mathbf{x}'') = \int \mathrm{d}\mathbf{x}\, p(\mathbf{x})\; k(\mathbf{x}, \mathbf{x}') k(\mathbf{x}, \mathbf{x}'')$ is easily solved analytically.

In our learning scheme we optimize (13) by using the maximum $U(\tilde{\mathbf{x}}|\mathcal{D})$ from a pool of 10 000 samples $\tilde{\mathbf{x}}$ from $p(\mathbf{x})$ and 10 000 from a Gaussian with variance $2B$, which is resampled for each draw[3].

[1] The predictive variance is given by (9b). The change for an additional sample can be derived using a rank-one update of Q^{-1}. The utility is also valid for both, noisy ($\sigma_y \neq 0$) and exact ($\sigma_y = 0$) observations y.

[2] As for k we use: $\mathbf{l}(\tilde{\mathbf{x}}) \in \mathbb{R}^N$, $L \in \mathbb{R}^{N \times N}$ with $[\mathbf{l}(\tilde{\mathbf{x}})]_\ell = l(\mathbf{x}_\ell, \tilde{\mathbf{x}})$ and $L_{i\ell} = l(\mathbf{x}_i, \mathbf{x}_\ell)$.

[3] The number of samples in the pool is somewhat arbitrary. We have made the pool large enough to obtain stable performance. To improve this brute-force optimization of (13) one can use a gradient based method to find the maximum, starting from several points to avoid local extrema.



(a) random samples (b) Latin Hypercube (c) $w_1 = w_2 = 0.08$

(d) $w_1 = w_2 = 0.22$ (e) $w_1 = w_2 = 0.6$ (f) $w_1 = w_2 = 1.7$

Fig. 1. Random samples compared to optimal designs. Plot (a) shows 200 independent samples from the input distribution $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$. Plots (b) shows a Latin Hypercube design with 200 samples. In (c–f) we have plotted optimal designs of 200 points (•), computed using 10 initial samples (×). The noise was set to a small level ($\sigma_y^2 = 10^{-5}$, $w_o = 1$). In contrast to random samples, optimal designs tend to spread the samples well apart from each other, where the length scales control the distances between the points. Latin Hypercube stratifies the design only on one dimensional projections, and uncovered areas can still be found.

Illustrative example. To visualize the difference between random samples and optimal designs, we have plotted two dimensional examples with 200 points each in figure 1. We have chosen a Gaussian input distribution with $\mathbf{x}_o = \mathbf{0}$ and $B = \mathrm{diag}(1, 1)$. Observe, in plot (a), that random samples tend to leave large areas under the input distribution uncovered, while we find some dense clusters. Naturally, most samples are found around $\mathbf{x}_o$.

Latin Hypercube sampling [20], plot (b), stratifies the design on one dimensional projections, but may show poor filling in the full space. Latin Hypercube designs sample from $p(\mathbf{x})$. Therefore they hardly provide points from low-density areas.

By considering prior measurements, the Bayesian scheme can adjust the length scales w_ℓ to reflect the variability of the function in each dimension. Plots (c–f) show the optimal designs for very short and long length scales. When the length scale parameter is very small ($w_1 = w_2 = 0.08$ in plot c), the correlations between function values decay rapidly and measurements need to be placed very close to each other. As the weighting factor $p(\mathbf{x})$ decreases with $|\mathbf{x}|$, the first 200 inputs are chosen close to $\mathbf{0}$. For $w_1 = w_2 = 1.7$ (f), which corresponds to

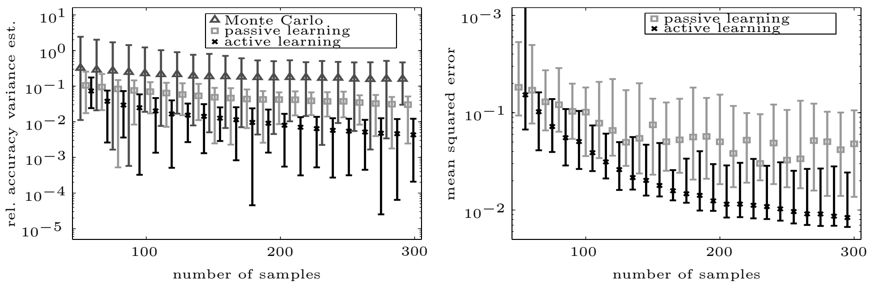


Fig. 2. Learning rates for the pressure sensor model: The estimates for the output variance using the MC method, passive and active Bayesian quadrature are plotted in the left panel. The test error for active and passive learning is shown in the right panel. The error bars indicate the median, minimum and maximum value out of 35 runs.

a smoother function, the inputs are chosen much further apart and the input distribution is explored even where we would hardly draw a random sample from $p(\mathbf{x})$. Note that the length scales are adjusted in the ML-II scheme each time a new measurement has been observed. Hence, the sampling scheme adapts to the characteristics of the output function.

4 Experiments

When the underlying model is correct we can be certain to improve the learning rate in the Bayesian active learning scheme. It is not clear, however, how much improvement the scheme gives in real applications. The Bayesian sensitivity analysis, as presented above, is used for the design analysis of novel micro electro-mechanical sensors at Robert Bosch GmbH. We have tested the active learning scheme on three fully featured models of different devices, which are based on FEM simulations.

We have analyzed the model of a pressure sensor with 28 fluctuating parameters, of an accelerometer with 29 parameters, and a yaw rate sensor with 15 parameters. In all cases we have initialized the active scheme with $N_o = 20$ random samples from $p(\mathbf{x})$. To test the generalization error we have used an independent test set of 22 950, 30 000 and 50 000 samples from $p(\mathbf{x})$.

The learning curves for the pressure sensor model are shown in figure 2. The plot on the left hand side compares the accuracy of the variance estimate using the simple MC method and the Bayesian quadrature with random and actively chosen samples. On 300 samples the Bayesian quadrature is by an order of magnitude more accurate than the MC method, and by using active learning we gain another factor of five. The plot to the right shows the mean squared error on the test set, which reflects this improvement.

For the accelerometer and the yaw rate sensor we show the learning curves in figure 3. As in the other example, the active scheme clearly outperforms passive learning. At 270 samples we gain roughly a factor five in accuracy. Note, that

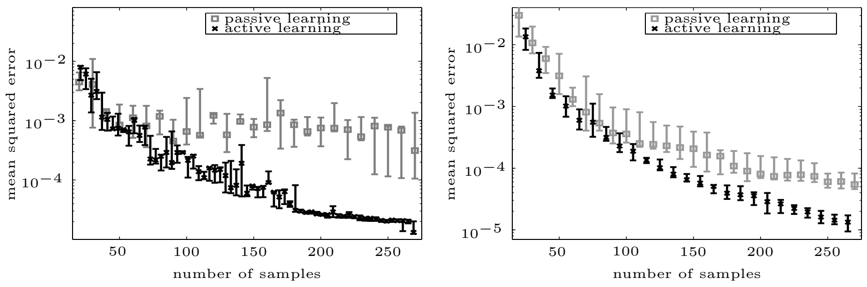


Fig. 3. Learning curves for the model of the accelerometer (left) and the yaw rate sensor (right). The markers indicate the median of 6 (left) and 3 (right) runs, the error bars cover the interval from the minimal to the maximal value.

the performance for random sampling scatters much stronger than that of the active scheme, as the latter is only partly randomized.

5 Discussion

Monte Carlo estimates are commonly used to explore the global behavior of computer models for sensitivity analysis. They have the advantage that they are simple to implement and that they do not make strong assumptions about the structure of the output. However, MC may not be feasible when the function is computationally too complex to be evaluated at a great number of parameter settings.

In industrial engineering computer experiments often model the behavior of complete physical systems, and they are used to analyze the robustness of a design with respect to fluctuations in mass production. For many models a sensitivity analysis is not feasible using the MC approach. Bayesian quadrature can resolve the problem by using the available data more efficiently.

In SA we are given—in contrast to most benchmark problems in machine learning—the region of interest and simulation software which can be called at any input. We can therefore use an active learning scheme which calls the software where the evaluation promises most informative. In contrast to previous work, which mainly uses space filling designs, our approach directly optimizes the Bayesian expected utility and updates the model parameters in each step. As the input distribution in SA is Gaussian, we can compute the expected utility analytically.

To quantify the benefit of the active learning scheme we have used three high dimensional, fully featured models from industrial engineering, which resemble micro electro-mechanical sensors. The models have up to 29 uncertain inputs, where the input distributions reflect fluctuations from mass production.

Bayesian quadrature proves much more efficient than the simple MC method. Compared to passive Bayesian quadrature with random samples from $p(\mathbf{x})$, the active learning scheme leads to a significant improvement of the generalization error. By using a uniform input distribution, the learning scheme can be applied to standard regression setups.

References

1. Saltelli, A., Chan, K., Scott, E.M.: Sensitivity Analysis. Wiley (2000)
2. Pfingsten, T., Herrmann, D.J., Rasmussen, C.E.: Model-based design analysis and optimization. IEEE Trans. on Semiconductor Manufacturing (in revision) (2006)
3. O'Hagan, A.: Bayes-Hermite Quadrature. Journal of Statistical Planning and Inference **29**(3) (1991) 245–260
4. Oakley, J.E., O'Hagan, A.: Probabilistic sensitivity analysis of complex models: a Bayesian approach. Journal of the Royal Statistical Society, Series B **66**(3) (2004) 751–769
5. Rasmussen, C.E., Ghahramani, Z.: Bayesian Monte Carlo. In: NIPS 15. (2003)
6. Lindley, D.V.: On the measure of information provided by an experiment. Ann. math. Statist. **27** (1956) 986–1005
7. Berger, J.O.: Statistical Decision Theory and Bayesian Analysis. Springer New York (1985)
8. Chaloner, K., Verdinelli, I.: Bayesian experimental design: A review. Statistical Science **10**(3) (1995) 273–304
9. Lindley, D.: Bayesian Statistics — A Review. SIAM (1972)
10. Mackay, D.: Information-based objective functions for active data selection. Neural Computation **4**(4) (1992) 589–603
11. O'Hagan, A.: Curve Fitting and Optimal Design for Prediction. J.R. Statist. Soc. B **40**(1) (1978) 1–42
12. Ko, C.W., Lee, J., Queyranne, M.: An exact algorithm for maximum entropy sampling. Operations Research **43**(4) (1995) 684–691
13. Lindley, D.: The choice of variables in multiple regression. Journal of the Royal Statistical Society B **30**(1) (1968) 31–66
14. Niederreiter, H.: Random number generation and quasi-Monte Carlo methods. SIAM (1992)
15. O'Hagan, A.: Monte Carlo is Fundamentally Unsound. The Statistician **36**(2/3) (1987) 247–249
16. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press (2006)
17. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical Science **4**(4) (1989) 409–423
18. Welch, W.J., Buck, R.J., Sacks, J.S., Wynn, H.P., Mitchell, T.J., Morris, M.D.: Screening, prediction, and computer experiments. Technometrics **34**(1) (1992) 15–25
19. Santner, T.J., Williams, B.J., Notz, W.I.: The Design and Analysis of Computer Experiments. Springer New York (2003)
20. Mackay, M.D., Beckmann, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2) (1979) 239–245
21. Johnson, M.E., Ylvisaker, D., Moore, L.: Minimax and maximin distance designs. J. of Statistical Planning and Inference **26** (1990) 131–148

Mixtures of Kikuchi Approximations

Roberto Santana, Pedro Larrañaga, and Jose A. Lozano

Intelligent System Group
Department of Computer Science and Artificial Intelligence
University of the Basque Country, 20080, San Sebastián, Spain
{rsantana, pedro.larranaga, ja.lozano}@ehu.es

Abstract. Mixtures of distributions concern modeling a probability distribution by a weighted sum of other distributions. Kikuchi approximations of probability distributions follow an approach to approximate the free energy of statistical systems. In this paper, we introduce the mixture of Kikuchi approximations as a probability model. We present an algorithm for learning Kikuchi approximations from data based on the expectation-maximization (EM) paradigm. The proposal is tested in the approximation of probability distributions that arise in evolutionary computation.

Keywords: Mixture of distributions, Kikuchi approximations, estimation of distribution algorithms, EM.

1 Introduction

Probabilistic modeling by finite mixture of distributions [8] concerns modeling a statistical distribution by a mixture (or weighted sum) of other distributions. Let $\mathbf{X} = (X_1, \ldots, X_n)$ denote a vector of discrete random variables, and $\mathbf{x} = (x_1, \ldots, x_n)$ denote an assignment to the variables. A mixture $q^m(\mathbf{x})$ of distributions $p_j(\mathbf{x})$ is defined to be a distribution of the form:

$$q^m(\mathbf{x}) = \sum_{j=1}^{m} \lambda_j p_j(\mathbf{x}) \tag{1}$$

with $\lambda_j > 0$, $j = 1, \ldots, m$, $\sum_{j=1}^{m} \lambda_j = 1$.

The $p_j(\mathbf{x})$ are called mixture components, and the λ_j are called mixture coefficients. m is the number of components of the mixture. A mixture of distributions can be viewed as containing an unobserved choice variable Z which takes value $j \in \{1, \ldots, m\}$ with probability $p(Z = j) = \lambda_j$. In some cases the choice variable Z is known.

Probabilistic modeling based on mixtures of distributions has been used in many domains. Two of the most frequent applications are data clustering and approximation of probability distributions [8]. Mixtures are specially suited for modeling problems that exhibit complex interactions between their variables.

Much research has gone into elucidating the properties of mixture distributions as well as into designing efficient algorithms to learn them. One important issue is whether the capacity of mixtures for probabilistic modeling can be enhanced by considering as components of the mixture probability distributions able to represent a higher number of dependencies of the data. Mixtures of mean field distributions [1] are one example of the way this type of models can improve the results achieved with simple factorial models.

Recently, the research on probability distribution approximation has widened its scope by the emergence of approximation methods inspired on region-based decompositions of the free energy [16]. These methods have been applied in the context of probabilistic modeling for classification [5] and evolutionary computation [13], where algorithms for learning Kikuchi approximations from data have been introduced.

One of these methods is the Kikuchi approximation of a probability that uses clique-based decomposition of independence graphs [13,14]. Kikuchi approximations can represent complex interactions between the variables of a problem. This provides our motivation to define a class of mixture of Kikuchi approximations and an algorithm to learn this approximation from data. The main goal in tackling this problem is to combine the capacity of mixtures to exploit asymmetric independence assertions with the power of Kikuchi approximations to represent complex interactions. To evaluate the efficiency of this model, we apply it in the context of function optimization by means of estimation of distribution algorithms (EDAs) [7,10] and for unsupervised learning. EDAs are optimization algorithms that explicitly model probabilistic dependencies between variables of the problem domain to make an efficient search for optimal solutions.

The remainder part of the paper is ordered as follows. In the next section, the Kikuchi approximation defined on clique-based decompositions is presented. Section 3 introduces the mixture of Kikuchi approximations and an algorithm for learning these approximation from data. Section 4 briefly explains EDAs and introduces an EDA that uses mixtures of Kikuchi approximations. Section 5 presents a number of experiments that analyze the dynamics of the introduced learning algorithm, and the effect of using the mixture of Kikuchi approximations as the probability model in EDAs. The conclusions of our paper are presented in Section 6.

2 Kikuchi Approximation: Recapitulation

Kikuchi approximations of the free energy [6] are region-based decompositions of the free energy that satisfy certain constraints. The Kikuchi approximation of a probability distribution from a clique-based decomposition of an independence graph [13] is a particular type of factorization in probability marginals. The marginals in the factorization are completely determined by the independence graph. Given this graph, the clique-based decomposition is formed by the maximal cliques of the graphs and their intersections. All these cliques are called regions. More formally: let S denote a set of indices in $N = \{1, \dots, n\}$, and $\mathbf{X}_S$ (respectively $\mathbf{x}_S$) a subset of the variables of $\mathbf{X}$ (respectively a subset of values of $\mathbf{x}$) determined by the indices in S. We will work with positive probability distributions denoted by $p(\mathbf{x})$. Similarly, $p(\mathbf{x}_S)$ will denote the marginal probability for $\mathbf{X}_S$. We use $p(x_i \mid x_j)$ to denote the conditional probability distribution of X_i given $X_j = x_j$.

Given a probability distribution $p(\mathbf{x})$, its independence graph $G = (V, E)$ associates one vertex with every variable of $\mathbf{X}$, and two vertices are connected if the corresponding variables are conditionally dependent given the rest of the variables. We define a region R of the independence graph $G = (V, E)$ of a probability distribution $p(\mathbf{x})$ as a subset of V. A graph region-based decomposition $(\mathcal{R}, U)$, is a set of regions $\mathcal{R}$ that covers all the V, and an associated set of *overcounting numbers* U which is formed by assigning one overcounting number c_R for each $R \in \mathcal{R}$. c_R will always be an integer, and might be zero or negative for some R.

To find a region-based decomposition, the cluster variation method (CVM) can be used [6,16]. In CVM, $\mathcal{R}$ is formed recursively by an initial set of regions $\mathcal{R}_0$ such that all the nodes are in at least one region of $\mathcal{R}_0$, and any other region in $\mathcal{R}$ is the intersection of one or more of the regions in $\mathcal{R}$. The set of regions $\mathcal{R}$ is closed under the intersection operation, and can be ordered as a partially ordered set.

In a clique-based decomposition the CVM is applied making a particular choice of the initial regions. The set $\mathcal{R}_0$ is formed by taking one region for each maximal clique in G. As a result, all the regions $R \in \mathcal{R}$ will be cliques because they are the intersection of two or more cliques.

We define the Kikuchi approximation of the probability distribution $p(\mathbf{x})$ associated with a clique-based decomposition, $k(\mathbf{x})$ as:

$$k(\mathbf{x}) = \prod_{R \in \mathcal{R}} p(\mathbf{x}_R)^{c_R} \tag{2}$$

where $\mathcal{R}$ comes from a clique-based decomposition and the overcounting numbers c_R are calculated using the following recursive formula:

$$c_R = 1 - \sum_{\substack{S \in \mathcal{R} \\ R \subset S}} c_S \tag{3}$$

where c_S is the overcounting number of any region S in $\mathcal{R}$ such that S is a superset of R. c_R values corresponding to the initial maximal cliques are equal to 1. If c_R is different from zero, the region is included in the clique-based decomposition.

From now on, when we refer to a Kikuchi approximation, we imply a Kikuchi approximation obtained from a clique-based decomposition. The Kikuchi approximation has a number of convenient properties for approximating distributions. If the independence graph is chordal, the Kikuchi approximation calculated from a clique-based decomposition corresponds to an exact factorization of the probability distribution calculated from a junction tree of the independence graph. The Kikuchi approximation also satisfies a number of Markov and decomposability properties [14].

3 Mixture of Kikuchi Approximations

In this section, we define the mixture of Kikuchi approximations and present an algorithm for learning this type of model from data.

Definition 1. *A mixture of Kikuchi approximations is defined to be an approximation of the form:*

$$k^m(\mathbf{x}) = \sum_{j=1}^{m} \lambda_j k_j(\mathbf{x}) \tag{4}$$

with $\lambda_j > 0$, $j = 1, \ldots, m$, $\sum_{j=1}^{m} \lambda_j = 1$.

The components of $k^m(\mathbf{x})$ are Kikuchi approximations. Since Kikuchi approximations are not probability distributions in general, the mixture of Kikuchi approximations is not either. However, notice that whenever the clique-based decompositions correspond to chordal graphs, each component of the mixture will be a junction tree, and $k^m(\mathbf{x})$ will be a probability distribution. In fact, a mixture of Kikuchi approximations opens the possibility of combining components that are probability distributions with other that are not.

Approaches used to learn mixtures of distributions include [8]: graphical methods, minimum-distance methods, maximum likelihood, methods of moments, and Bayesian approaches. If the choice variable is not observed, one of the alternatives that can be used for learning the structure and parameters of the components is the EM algorithm [4], that looks for a mixture that maximizes the likelihood of the data. The iterative EM algorithm is a general, usually reliable numerical method for obtaining maximum likelihood (or Bayesian maximum a posteriori) estimates of parameters in incomplete-data contexts.

To learn a mixture of Kikuchi approximations, we propose to use a version of the EM algorithm. The general scheme is similar to the procedure used to learn mixture of trees [9]. However, fundamental differences arise in the expectation and maximization steps. The learning problem can be established as: Given a set of observations $D = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N\}$, we are required to find the mixture of Kikuchi approximations $k^m(\mathbf{x})$ that satisfies

$$k^m(\mathbf{x}) = \arg \max_{k'^m(\mathbf{x})} \sum_{i=1}^{N} \log k'^m(\mathbf{x}^i) \tag{5}$$

Within the framework of the EM algorithm, expresion (5) is commonly referred as the incomplete log-likelihood of the data given a probability distribution. Since Kikuchi and mixture of Kikuchi approximations are not probability distributions in general, in these cases this expression will not correspond to the incomplete log-likelihood. Nevertheless, we will use the right term in equation (5) as a measure of the accuracy of the approximation given by the mixture of Kikuchi approximations.

In the EM algorithm, the complete likelihood is defined as the log-likelihood of both, the observed and the unobserved data, given the current model estimate $k^m(\mathbf{x})$. A version of the complete log-likelihood for the mixture of Kikuchi approximations is shown in equation (6).

$$\begin{aligned}\mathcal{L}(\mathbf{x}^1,\ldots,\mathbf{x}^N,z^1,\ldots,z^N|k^m(\mathbf{x})) &= \sum_{i=1}^{N} log \prod_{j=1}^{m} \left(\lambda_j k_j(\mathbf{x}^i)\right)^{\delta_{j,z^i}} \\ &= \sum_{i=1}^{N}\sum_{j=1}^{m} \delta_{j,z^i}(\log\lambda_j + \log k_j(\mathbf{x}^i)) \qquad (6)\end{aligned}$$

where δ_{j,z^i} is equal to one if z^i is equal to the j^{th} value of the choice variable, and zero otherwise.

By maximimizing (6), the mixture Kikuchi-EM learning algorithm pursues to indirectly find a solution to the maximization problem defined by equation (5). The idea underlying the Kikuchi-EM algorithm is to compute and optimize the expected value of $\mathcal{L}(\mathbf{x}^1,\ldots,\mathbf{x}^N,z^1,\ldots,z^N|k^m(\mathbf{x}))$. However, the way the expectation and maximization steps are implemented is very particular. In the expectation step of the mixture Kikuchi-EM, we use Kikuchi approximations values for estimating the posterior probability of the hidden variable for each of the observations. In our case, this means estimating the probability of each component of the mixture generating data point $\mathbf{x}^i$.

$$p(Z^i = j|\mathbf{x}^i, k^m(\mathbf{x})) = \gamma_j(\mathbf{x}^i) = \frac{\lambda_j k_j(\mathbf{x}^i)}{\sum_{j'} \lambda_{j'} k^{j'}(\mathbf{x}^i)} \qquad (7)$$

One uses these posterior probabilities to compute the expectation of $\mathcal{L}$, which is a linear function of the $\gamma_j(\mathbf{x})$ values. Let us introduce the following quantities:

$$\Gamma_j = \sum_{i=1}^{N} \gamma_j(\mathbf{x}^i),\ j = 1,\ldots,m \qquad (8)$$

$$q^j(\mathbf{x}^i) = \frac{\gamma_j(\mathbf{x}^i)}{\Gamma_j} \qquad (9)$$

The sums $\Gamma_j \in [0, N]$ can be interpreted as the total number of data points that are generated by the j-th component. By normalizing the posteriors $\gamma_j(\mathbf{x}^i)$ with Γ_j we obtain a probability distribution $q^j(\mathbf{x}^i)$ over the data set. Notice that even if $k_j(\mathbf{x})$ is not a probability distribution, $q^j(\mathbf{x})$ is a probability distribution because it is the result of normalization.

The maximization step of the Kikuchi-EM algorithm looks for estimating the parameters of the model so as to maximize $E[\mathcal{L}(\mathbf{x}^1,\ldots,\mathbf{x}^N|k^m(\mathbf{x}))]$. Consequently, it is necessary to obtain the model that best fits the data in each component. In the case of the mixtures of trees, this problem can be solved using an algorithm that is guaranteed to give the best structure [9]. For Kikuchi approximations, we use a learning algorithm introduced in [13]. This algorithm serves for searching in the space of the Kikuchi approximations, but the optimum is not guaranteed to be found.

The algorithm learns an independence graph from the data and finds its clique-based decomposition. To learn the independence graph, independence tests are used. We use the Chi-square independence test. If for two variables X_i and X_j, we reject the null hypothesis of independence with a specified level of significance α, they are joined by an edge. The pseudocode of the Kikuchi-EM learning algorithm is shown in Algorithm 1. As a method for constructing the initial mixture of Kikuchi approximations, we propose a heuristic approach in which the learning algorithm proposed in [13] is used to learn each initial Kikuchi approximation component from the data using a different value of parameter α for each component. The termination criterion used is that the difference between the likelihood achieved at iterations $t+1$ and t is below a given threshold.

Algorithm 1. Mixture of Kikuchi EM

```
1   $t \leftarrow 1$; Set an initial mixture of Kikuchi approximations
2   do {
3      for $j \Leftarrow 1$ to $m$
4         for $i \Leftarrow 1$ to $N$
5            Compute $\gamma_j(\mathbf{x}^i)$, $\Gamma_j$, $q^j(\mathbf{x}^i)$ using equations (7), (8) and (9) re-
             spectively
6      for $j \Leftarrow 1$ to $m$
7         Compute the Kikuchi approximation $k_j(\mathbf{x})$ of $q^j(\mathbf{x})$
8         $\lambda_j = \frac{\Gamma_j}{N}$
9      $t \leftarrow t+1$
10  } until Termination criteria met
```

4 Application Domain: The Mixture of Kikuchi Approximations EDA

The mixture of Kikuchi approximations can be used in classification and in the approximation of distributions. In this section, we will describe an application of mixtures of Kikuchi approximations to function optimization by means of EDAs [7].

The goal of EDAs is function optimization. One essential assumption of these algorithms is that it is possible to build a probabilistic model of the search space that can be used to guide the search for the optimum. The probabilistic model can be built using available information about the function or learned from samples.

EDAs work with a set (or population) of points. Initially, a random sample of points is generated. These points are evaluated using the function, and a subset of points is selected based on this evaluation. Usually, points with higher evaluation has a higher probability of being selected. A probabilistic model of the selected solutions is built, and the model is sampled to obtain a new set of points. The process iterates until the optimum has been found or another termination criterion is fulfilled. A key characteristic and crucial step of EDAs is the construction of the probabilistic model. These models may differ in the order and number of the probabilistic dependencies that they represent.

Applications of mixtures of distributions in EDAs include the use of mixtures of Gaussian models for the solution of multiobjective continuous problems [2], the application of mixtures of Bayesian models [11] for clustering in continuous optimization, and the use of mixtures of trees [15] in discrete optimization. In Algorithm 2, the pseudo-code of the mixture of Kikuchi approximations EDA (MKA-EDA) is presented.

Algorithm 2. Mixture of Kikuchi approximations EDA

```
1  Set $t \Leftarrow 0$. Generate $N \gg 0$ points randomly
2  **do** {
3      Evaluate the points using the fitness function
4      Select a set $S$ of $M \leq N$ points according to a selection method
5      Calculate a mixture of Kikuchi approximations $Q(\mathbf{x})$ using a learning
       algorithm
6      Generate new points sampling from $Q(\mathbf{x})$
7      $t \Leftarrow t + 1$
8  } **until** Termination criteria are met
```

Algorithm 2 uses Kikuchi-EM algorithm to learn the model. To generate points from the Kikuchi approximations, a Gibbs sampling algorithm introduced in [13] is used. The selection method is truncation selection of parameter T, in which the $M = NT$ points with best function evaluation are selected.

The computational cost of MK-EDA depends on the cost of the algorithms used to learn and sample the Kikuchi approximation plus the cost of the EM algorithm. The complexity of learning the parameters depends n, N, the number of cliques μ, and their size. The order of this steps is $O(N\mu) \approx O(Nn2)$. The total complexity of the learning algorithm is roughly estimated as $O(Nn3)$.

5 Experiments

We start this section by presenting the optimization problem and instances used in our experiments. The following experiments study the dynamics of the Kikuchi-EM learning algorithm and compare it with other probabilistic models. Finally, the section shows the results of the MKA-EDA.

The satisfiability (SAT) problem consists in finding an assignment of values to a set of n boolean variables such that they satisfy a given set of clauses $c_1, c_2, \ldots, c_r$, where c_i is a disjunction of literals, and a literal is a variable or its negation. The restriction of SAT to instances where all clauses have length 3 is denoted 3-SAT. This problem is NP-complete [3]. In our representation, a variable X_i is associated to each boolean variable. As the objective function, we use the sum of clauses satisfied by the solution.

The selected problems benchmark is composed by three sets of difficult instances. These instances are contained in the files *uf20-91*, *aim-50-3-4-yes1-j* and *aim-100-3-4-yes1-j*[1]. The *uf20-91* file contains 1000 instances, all have 20 variables and 91 clauses, all are satisfiable. File *aim-50-3-4-yes1-j* and *aim-100-3-4-yes1-j* contain four instance each. The number of the variables of instances in these two files are, respectively, 50 and 100. The number of clauses are, respectively, 170 and 340.

In a preprocessing step, the *uf20-91* instances where classified in five groups according to the difficulty they pose for a very simple EDA. The criterion for classification was the EDA success rate in 100 runs. The most difficult class comprises to instances for which the simple EDA converged in less than 20 runs. This class includes 38 instances and was used to evaluate the performance of MKA-EDA.

5.1 Dynamics of the Kikuchi-EM Learning Algorithm

In the first experiment, we evaluate the behavior of the Kikuchi-EM learning algorithm in the approximation of the empirical probability distribution of data obtained from the optimization of the *aim-50-3-4-yes1-1* instance. A population size of 500 points and truncation selection with parameter $T = 0.1$ are used. First, all the solutions are evaluated, the first selected set of solutions is used for the experiment. The goal of the experiment is to evaluate the approximation achieved at each step of the Kikuchi-EM learning algorithm.

To evaluate the quality of the approximation we use a quality measure similar to the Kullback-Leibler divergence between the target empirical distribution and the learned mixture of Kikuchi approximations $D(p||k^m) = \sum_{\mathbf{x}} p(\mathbf{x}) log \frac{p(\mathbf{x})}{k^m(\mathbf{x})}$. Kullback-Leibler divergence is only defined between probability distributions. We employ the same measure but warning that as mixture of Kikuchi approximations are not in general probability distributions, the measure used does not fulfill a number of properties satisfied by the Kullback-Leibler divergence. Additionally, we measure the Kullback-Leibler divergence between the empirical distribution and the mixture of Kikuchi approximations normalized in the set of data $\bar{k}^m(\mathbf{x}) = \sum_{j=1}^{m} \lambda_j q^j(\mathbf{x})$. Normalization of the Kikuchi approximation guarantees to obtain a probability distribution on the set of data. Therefore, the Kullback-Leibler divergence will be always non-negative in this case. Additionally, normalization is a required step of the EM method used to learn the Kikuchi approximations. In each step of the learning algorithm, the divergences are calculated from the current approximation learned by the model.

[1] All files, together with a description about the instances, are available from http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/benchm.html

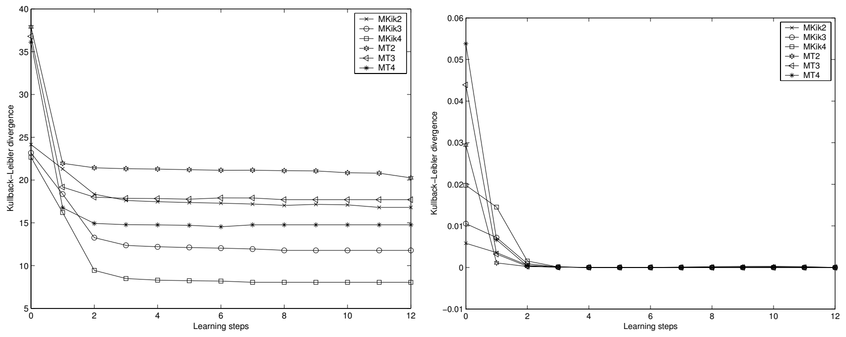


Fig. 1. Amount of decrease in the Kullback-Leibler divergence between the mixture probability model and the target probability during the learning process. Left: Kullback-Leibler divergence has been calculated from the mixture. Right: Kullback-Leibler divergence has been calculated from the normalized mixture.

Figure 1 shows the results of this experiment for mixtures of Kikuchi approximations (MKik) with different number of components. In the figure, we have included the results obtained using the mixture of trees (MT) EM learning algorithm. For every mixture, the results are the average from 1000 runs. As learning may take different number of steps for each selected population, the average of the divergence at step i is calculated from all the experiments that reach step i. It can be seen in Figure 1 that during the first steps of the learning algorithm the divergence to the empirical distribution decreases. The best results are achieved for the mixtures of Kikuchi approximations of 4 components (MKik4). For both types of mixtures, the divergence to the target probability decreases with the increase in the number of components. It can be appreciated that MT4 outperforms the behavior of MKik2.

We use the information collected from these experiments to calculate the gain in the approximation measured as the difference between the Kullback-Leibler obtained in the last and second iterations of the learning algorithms. Results are shown in Figure 2. The value of divergence at the second iteration is taken as a reference because the algorithm for learning mixtures of trees starts for a complete random initialization, while the heuristic method used to initialize the Kikuchi-EM learning algorithm takes profit of the information contained in the data. It can be appreciated in the figure that also the gain achieved by the Kikuchi-EM learning algorithm is higher than for the mixtures of trees.

Table 1. Percentage of success of MN-EDA and MKA-EDA for a set of difficult *uf20-91* instances

Alg.	MN-EDA	MKA-EDA$_2$	MKA-EDA$_3$	MKA-EDA$_4$	MKA-EDA$_5$
	72.78	84.11	78.63	75.84	72.96

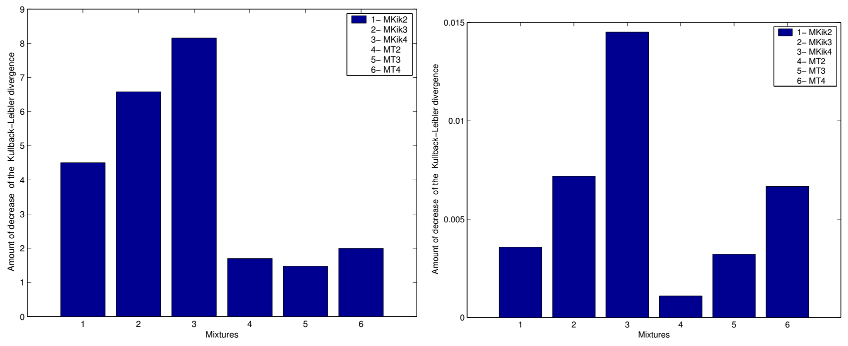


Fig. 2. Amount of decrease in the Kullback-Leibler divergence between the mixture probability model and the target probability after the learning procedure is complete. Left: Kullback-Leibler divergence has been calculated from the mixture. Right: Kullback-Leibler divergence has been calculated from the normalized mixture.

5.2 Results of MKA-EDA

In the following experiments we investigate the ability of MKA-EDA to find the solution of the SAT problem. In an initial experiment, we compare the performance of MKA-EDA with an EDA based on Kikuchi aproximations (MN-EDA)[13]. MN-EDA uses a probabilistic model based on a single Kikuchi approximation. We calculate the average success of MN-EDA and MKA-EDA with different number of components for a set of difficult *uf20-91* instances. MN-EDA and MKA-EDA use a population size $N = 500$, a maximum of 25 generations, and the parameter of truncation was $T = 0.15$.

Table 1 shows the results of experiments. Notice the improvement in the results achieved by MKA-EDA compared to those obtained by MN-EDA. When the number of components of the mixture of Kikuchi approximations is increased results deteriorate. This fact may be due to the overfitting problem. However, it is even better to use MKA-EDA with a mixture of five components than the MN-EDA.

In the next experiment, we evaluate the behavior of MKA-EDA for instances of higher size. We compare its performance to FDA, MN-FDA and MN-EDA which are EDA with an increasing complexity in their probability models [12,13]. We employ a very simple local optimization method to accelerate the convergence of all the EDAs. We apply this algorithm to every solution during the evaluation step. Except in one case, all the experiments were done for a population size $N = 1000$, parameter of truncation $T = 0.15$, and a maximum number of generations 50. The only exception is MKA-EDA$_4$. For this algorithm, $N = 5000$ and the maximum number of generations was 200. The goal of including MKA-EDA with these parameters was to evaluate the improvement in the convergence results when the population size is increased.

Table 2. Comparison among MN-EDAs for the aim instances of the SAT problem

		FDA		MN-FDA		MN-EDA		MKA-EDA$_2$		MKA-EDA$_4$	
n	mc	mb	v	mb	v	mb	v	mb	v	mb	v
50	170	169	10	170	2	170	1	170	2	170	10
50	170	170	6	170	7	170	9	170	9	170	10
50	170	170	7	170	8	170	9	170	10	170	10
50	170	170	9	170	9	170	10	170	10	170	10
100	340	337	1	336	6	337	2	337	1	336	4
100	340	337	1	337	2	337	4	337	3	340	1
100	340	336	7	336	7	336	9	336	6	336	10
100	340	340	2	340	3	340	2	340	2	340	6
$Tot.$			24		29		31		31		47

Table 2 shows the results achieved with the algorithms for instances *aim-50-3-4-yes1-j* and *aim-100-3-4-yes1-j*. In the table, mc is the number of clauses in each instance (optimum of the function), mb is the best value reached by the corresponding algorithm, and v is the number of times that the best found value was reached in 10 runs. The first rows correspond to the four instances of *aim-50-3-4-yes1-j*, and the next four to the four instances of *aim-100-3-4-yes1-j*. The last row shows the number of time the optimum was found in the 80 experiments.

It can be seen in Table 2 that MKA-EDA$_2$ achieves better or equal results than the rest of algorithms. For these intances however, the difference between results achieved by MN-EDA and MKA-EDA$_2$ is not statistically significant. The results of MKA-EDA can be improved by augmenting the number of components in the mixture, the population size, and the number of generations of the algorithm. However, determining the exact number of components for mixture of Kikuchi approximations is an open problem. Overfitting can arise and it is a general problem for Kikuchi and other mixture of distributions. On the other hand, EDAs based on trees and mixture of trees have good behavior for problems with low dependencies between the variables. Kikuchi and mixture of Kikuchi approximations outperform them for problems with more complex interactions.

6 Conclusions

In this paper, we have introduced a new class of probability models based on Kikuchi approximations of probability distributions. An algorithm has been introduced lo learn mixtures of Kikuchi approximations from data. The approximations learned by the algorithm can be more accurate, in terms of the Kullback-Leibler divergence, than other approximations based on mixtures of less complex components. The mixture of Kikuchi approximations combines the capacity of mixtures to exploit asymmetric independence assertions with the power of Kikuchi approximations to handle complex interactions. We recommend its application to problems with very complex interactions that can not be represented with simpler model.

References

1. C. M. Bishop, N. Lawrence, T. Jaakkola, and M. I. Jordan. Approximating posterior distributions in belief networks using mixtures. In *Proc. Conf. Advances in Neural Information Processing Systems 10, NIPS*, pages 416–422. MIT Press, 1998.
2. P. A. Bosman and D. Thierens. Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. *International Journal of Approximate Reasoning*, 31(3):259–289, 2002.
3. S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, Ohio, 1971.
4. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B(39):1–38, 1977.
5. A. Jakulin, I. Rish, and I. Bratko. Kikuchi-Bayes: Factorized models for approximate classification in closed form. Technical Report RC23314 (WO408-175), IBM, August 2004.
6. R. Kikuchi. A theory of cooperative phenomena. *Physical Review*, 81(6):988–1003, 1951.
7. P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation.* Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.
8. G. McLachlan and D. Peel. *Finite Mixture Models.* John Wiley & Sons, 2000.
9. M. Meila and M. I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.
10. H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, pages 178–187, Berlin, 1996. Springer Verlag. LNCS 1141.
11. J. Peña, J. A. Lozano, and P. Larrañaga. Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks. *Evolutionary Computation*, 13(1):43–66, 2005.
12. R. Santana. A Markov network based factorized distribution algorithm for optimization. In *Proceedings of the 14th European Conference on Machine Learning (ECML-PKDD 2003)*, volume 2837 of *Lecture Notes in Artificial Intelligence*, pages 337–348, Dubrovnik, Croatia, 2003. Springer-Verlag.
13. R. Santana. Estimation of distribution algorithms with Kikuchi approximations. *Evolutionary Computation*, 13(1):67–97, 2005.
14. R. Santana, P. Larrañaga, and J. A. Lozano. Properties of Kikuchi approximations constructed from clique based decompositions. Technical Report EHU-KZAA-IK-2/05, Department of Computer Science and Artificial Intelligence, University of the Basque Country, April 2005. Available from http://www.sc.ehu.es/ccwbayes/technical.htm.
15. R. Santana, A. Ochoa, and M. R. Soto. The mixture of trees factorized distribution algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001*, pages 543–550, San Francisco, CA, 2001. Morgan Kaufmann Publishers.
16. J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report TR-2004-040, Mitsubishi Electric Research Laboratories, May 2004.

Boosting in PN Spaces

Martin Scholz

Artificial Intelligence Group, University of Dortmund, Germany
`scholz@ls8.cs.uni-dortmund`

Abstract. This paper analyzes boosting in unscaled versions of ROC spaces, also referred to as PN spaces. A minor revision to AdaBoost's reweighting strategy is analyzed, which allows to reformulate it in terms of stratification, and to visualize the boosting process in nested PN spaces as known from divide-and-conquer rule learning. The analyzed confidence-rated algorithm is proven to take more advantage of its base models in each iteration, although also searching a space of linear discrete base classifier combinations. The algorithm reduces the training error quicker without lacking any of the advantages of original AdaBoost. The PN space interpretation allows to derive a lower-bound for the area under the ROC curve metric (AUC) of resulting ensembles based on the AUC after reweighting. The theoretical findings of this paper are complemented by an empirical evaluation on benchmark datasets.

1 Introduction

Boosting is one of the most popular learning techniques in practice, but not yet fully understood in terms of its selection metrics and convergence behavior. The classical AdaBoost algorithm [1] has been presented more than one decade ago, but is still on the agenda of research.

This paper shows how boosting translates into ROC spaces, more precisely into their unscaled counterparts which are referred to as *PN spaces*. ROC analysis provides a unifying framework for studying the behavior of different evaluation metrics [2], for illustrating how to handle class skews, asymmetric misclassification costs, and how to correctly choose a confidence threshold for soft classifiers in different settings [3]. Moreover, the area under the ROC curve (AUC) is the standard machine learning metric for the ranking performance of soft classifiers.

This paper analyzes a revised version of AdaBoost, which is basically subsumed by the framework of confidence-rated boosting [4]. The adapted algorithm allows for a simplified illustration in PN spaces. As its main advantage it allows AdaBoost to take more advantage of its base models, generally increasing the learning rate and generalization performance of boosting. The resulting algorithm still searches a linear combination of crisp base classifiers and can be reformulated in simpler terms, as to continuously stratify the target attribute.

The aim of the analysis is to foster a better understanding of the implicit AUC optimization property of boosting. Original AdaBoost's excellent ranking behavior has just recently been explained by showing its similarity to RankBoost if equal loss is suffered from positive and negative examples [5]; another technical proof exists for Real AdaBoost [4]. This paper contributes an intuitive and much simpler proof of a tighter ranking error (AUC) bound for Real AdaBoost-like ensemble classifiers, derived from a geometric interpretation in PN spaces.

2 Formal Framework and Basic Properties

ROC analysis has become a popular tool for analyzing classifier performances and to study evaluation metrics [6]. The unscaled counter-part, PN spaces [2], are well-suited to visualize the nested subspaces of divide-and-conquer rule learning. The only difference to ROC spaces is that the axes of PN spaces show the absolute numbers of positives and negatives, while in ROC spaces both axes are scaled to the range of $[0, 1]$. This paper confines itself to boolean classification problems, so models are functions mapping an instance space $\mathcal{X}$ to a boolean target label $\mathcal{Y} = \{+1, -1\}$. The notation used in this paper is chosen to be similar to the one used in [2] for rule learning in PN spaces.

Definition 1. *For a model $h : \mathcal{X} \to \mathcal{Y}$ and a given data set $\mathcal{E}$ with an absolute number of P positive examples and N negative examples, the absolute number of true positives is denoted as p, the number of false positives as n. Analogously, the absolute number of false negatives is denoted as $\overline{p}$, the absolute number of true negatives as $\overline{n}$.*

Fig. 1 shows nested PN spaces from specific to general as obtained when adding a single rule to a decision list in each iteration. The $p + n$ examples for which the rule applies are removed from further consideration, and the remaining examples are represented by nested rectangles of shrinking size. Analogously, refining a rule adding one literal at a time shrinks the covered subsets from general to specific.

ROC spaces can be considered to show *stratified* versions of PN spaces; axes represent classes and are normalized to the same scale. The term stratification will reoccur in this paper. It can be interpreted as the process of altering class proportions in the training set, so that all classes are equally frequent. It is a common preprocessing step in the machine learning literature, e.g. for training classifiers under skewed class distributions or varying misclassification costs [3].

Stratification can be realized by reweighting or by subsampling. The goal in the former case is to obtain the same total example weight for each class. To this end, all examples sharing a class receive a common weight, chosen inverse proportionally to the frequency of the class in the training set. In the latter case one samples with equal probability from each class, hence implicitly samples from another than the i.i.d. distribution underlying the training data. The following definition captures the resulting implicit new target distribution.

Definition 2. *For a distribution $D : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ over an instance space $\mathcal{X}$ and a target label $\mathcal{Y}$ the stratified random sample distribution D' of D is*

$$D'(x, y) := \frac{D(x, y)}{|\mathcal{Y}| \cdot P_{(x', y') \sim D}(y = y')}.$$

A typical application of *ROC* analysis in machine learning is to visualize soft classifiers, that yield continuous confidence scores for examples being positive. Each crisp classifier obtained by applying a threshold to a soft classifier is represented in a ROC diagram as a point depicting the resulting true positive rate p/P and false positive rate n/N [3]. The area under the resulting graph is referred to as the *area under the ROC curve (AUC)*, which equals the probability that a randomly selected positive example is ranked higher (higher confidence) than a randomly selected negative example. Maximizing the AUC is a learning task of its own right and has also been shown to lead to a competitive but more robust selection of models regarding maximization of predictive accuracy [7].

In this paper a quantity closely related to the AUC is analyzed, the area *over* the curve. An asterisk indicates quantities in PN space rather than ROC space.

Definition 3. *For a given soft classifier and example set $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ of positive examples $\mathcal{E}^+$ and negative examples $\mathcal{E}^-$ the* area over the curve in PN spaces (AOC*) *is defined as the number of misranked tuples $(e^+, e^-) \in \mathcal{E}^+ \times \mathcal{E}^-$, that is the number of pairs for which e^- is predicted positive with higher confidence than e^+. Example pairs (e^+, e^-) with associated weights w^+ and w^- are accounted for by $w^+ \cdot w^-$ misranks.*

Ties are considered to be broken randomly, so half of all equally ranked pairs are considered to be misranked. It is easily seen that $AOC^* = (1 - AUC) \cdot P \cdot N$ for the unweighted case, which is a special case of the weighted case with $w^{+/-} := 1$. An example with a weight of w naturally represents an example *set* of size w.

The main diagonal in ROC/PN space represents the performances of default classifiers and random classifiers that do not incorporate any data at all, but predict $y = +1$ with a fixed probability. The AOC* of such uninformed models equals half the area of the corresponding PN space, i.e. $AOC^* = (P \cdot N)/2$. Changing the class proportions clearly does not affect this ranking performance. To preserve fundamental semantics it is hence suggested not to change the AOC* during steps of skewing the data, as it reflects the absolute ranking error. This is further justified at a later point. It translates into the constraint $P' \cdot N' = P \cdot N$ for the new values P' and N' obtained by skewing P and N, respectively.

The learning algorithms used in this paper are assumed to implicitly normalize the training set, so that the weights describe a distribution. Hence, the only quantities of interest for stratification are the class ratios P/N and P'/N'.

Proposition 1. *The reweighting rule for changing the ratio of P/N by a factor c while meeting the constraint $P \cdot N = P' \cdot N'$ is unique:*

$$w'(x,y) := w(x,y) \cdot \begin{cases} \sqrt{c}, \text{ for } & \text{positive } y \\ \frac{1}{\sqrt{c}}, \text{ for } & \text{negative } y \end{cases}$$

Proof. It directly follows that after reweighting we have

$$P' = \sqrt{c}P \ , \ N' = \frac{N}{\sqrt{c}} \ , \ \frac{P'}{N'} = c \ , \ P' \cdot N' = P \cdot N.$$

Multiplying positives with a constant of c' requires to divide negatives by the same constant to satisfy the constraint. Only $c' = \sqrt{c}$ is valid, since $P'/N' = c'^2$.

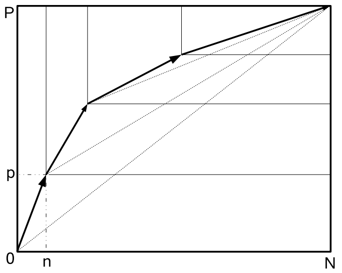


Fig. 1. Nested PN-Spaces

```
Initialize weights $w_1(x_i, y_i) := 1$ for $(x_i, y_i) \in \mathcal{E}$
for $t = 1$ to $k$ do
    $h_t \leftarrow$ base learner$(\mathcal{E}, w_t)$
    Compute $\epsilon_t := \sum_{i=1}^{|\mathcal{E}|} w_t(x_i, y_i) I[h_t(x_i) \neq y_i]$
    Let $\alpha_t := \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
    $w_{t+1}(x_i, y_i) := w_t(x_i, y_i) \cdot \exp(-y_i \alpha_t h_t(x_i))$
end for
Output classifier: Predict sign $\left(\sum_{i=1}^{k} \alpha_t h_t(x)\right)$
```

Fig. 2. AdaBoost for $y \in \{+1, -1\}$

As required, the weighting does not change the AOC* (Def. 3). Stratification is a specific case of skewing the data, leading to equal class proportions. It has a further important property in the context of boosting, as will be shown in Sec. 3.

Proposition 2. *Among all skewing operations preserving $P \cdot N$, stratifying the data by choosing $c = N/P$ leads to the minimal total example weight of* $2\sqrt{PN}$.

Proof. For valid reweightings we have $P' \cdot N' = P \cdot N$. The weight to be minimized is $P' + N' = \sqrt{c}P + N/\sqrt{c} =: f(c)$. Setting the derivative to 0 yields the result.

3 Boosting

3.1 AdaBoost

Combining individual classifiers to weighted ensembles is an effective way to increase predictive accuracy and other metrics like the AUC. The best known, most studied, and probably the most frequently applied ensemble method is AdaBoost [1], depicted in Fig. 2. It fits a sequence of base models $h_t : \mathcal{X} \rightarrow \mathcal{Y}$, each to a reweighted version of the training set, or to an analogously constructed subsample, respectively. The term $I[\cdot]$ used in the algorithm refers to the indicator function. To simplify subsequent analysis the algorithm is formulated without the step of normalizing weights, which is left to the base learner.

The reweighting scheme of AdaBoost gives higher weight to the "hard" examples of the training set, and finally predicts based on a weighted majority vote. A different perspective has been fostered in [8], pointing out AdaBoost's similarity to additive logistic regression. From an optimization point of view AdaBoost fits into the broader AnyBoost framework [9], as it performs gradient descent in function space in order to minimize the exponential loss function $\exp(-y_i \sum_{t=1}^{k} \alpha_t h_t(x_i))$. For a (weighted) error rate of ϵ_t of the base classifier in iteration t and $\beta_t(x) := (1-\epsilon_t)/\epsilon_t = \exp(2\alpha_t)$ the reweighting strategy computes

$$w_{t+1}(x, y) = w_t(x, y) \cdot \exp(-y\alpha_t h_t(x_i)) = \prod_{t=1}^{k} (\sqrt{\beta_t})^{-y \cdot h_t(x)} \quad (1)$$

as the new weight for each example (x, y), starting with uniform weights. All examples with a final weight w_{t+1} of less than 1 are classified correctly, since

$$\sum_{t|h_t(x)=y} \alpha_t > \sum_{t|h_t(x)\neq y} \alpha_t \Leftrightarrow 1/2 \left(\sum_{t|h_t(x)=y} \ln \frac{1-\epsilon_t}{\epsilon_t} - \sum_{t|h_t(x)\neq y} \ln \frac{1-\epsilon_t}{\epsilon_t} \right) > 0$$
$$\Leftrightarrow \prod_{t=1}^{k} \sqrt{\beta_t}^{(y \cdot h_t(x))} > 1 \;\Leftrightarrow\; w_{t+1}(x,y) < 1.$$

In turn, examples with a weight of greater than 1 are misclassified. For this reason one of the most important properties of AdaBoost is that it reduces the total weight quickly if the base learner provides useful classifiers h_t.

3.2 Ada²Boost

One disadvantage of AdaBoost is that it does not take full advantage of its base models. For illustration we consider a classification rule covering significantly less than half of the examples (respecting weights), but having a low error rate for this subset. Such a model is generally useful for ensemble learning. However, it is not necessarily useful for AdaBoost, because the error rate of the large uncovered part might be significantly higher, resulting in a value of $\alpha_t \approx 0$.

Such asymmetric cases can be handled by using separate estimates of the error rate for the *covered* part $\mathcal{C}_t := \{(x,y) \in \mathcal{E} | h_t(x) = +1\}$ and the *uncovered* part $\overline{\mathcal{C}}_t := \{(x,y) \in \mathcal{E} | h_t(x) = -1\}$. Both local error rates, denoted as

$$\epsilon^+ := n/(p+n) \text{ for } \mathcal{C}_t, \text{ and } \epsilon^- := \overline{p}/(\overline{p}+\overline{n}) \text{ for } \overline{\mathcal{C}}_t$$

can easily be computed from the contingency matrix and will usually differ. Please note, that for $\overline{\mathcal{C}}_t$ the *negative* examples are the correctly classified ones. We will replace the static values of β_t by functions $\beta_t(h_t(x))$ that depend only on the prediction of their corresponding base model $h_t(x) \in \{+1, -1\}$. This leads to two separate factors, the odds ratio for $\mathcal{C}_t$, and the inverse odds ratio for $\overline{\mathcal{C}}_t$:

$$\beta(+1) := \frac{1-\epsilon^+}{\epsilon^+} = \frac{p}{n} \quad , \qquad \beta(-1) := \frac{1-\epsilon^-}{\epsilon^-} = \frac{\overline{n}}{\overline{p}} \tag{2}$$

With $\alpha(h(x)) := (\ln \beta(h(x))/2$ the weight update of AdaBoost translates into:

$$w_{t+1}(x_i, y_i) := w_t(x_i, y_i) \cdot \exp\left[-y_i \cdot \alpha_t(h_t(x_i)) \cdot h_t(x_i)\right]$$

The rule for predicting a label $\hat{y} \in \{+1, -1\}$ is changed accordingly:

$$\hat{y} := \text{sign}\left(\sum_{t=1}^{k} \alpha_t(h_t(x)) h_t(x)\right) \tag{3}$$

This adapted version of AdaBoost is referred to as *Ada²Boost* in this paper. It is only analyzed in combination with plain boolean base classifiers, which does not require regression-capabilities of base learners, as e.g. LogitBoost [8] that uses working responses and weights at the same time.

Ada^{2}Boost is similar to the confidence-rated Real AdaBoost [4], which allows for continuous predictions $h_t : \mathcal{X} \to \mathbb{R}$. Real AdaBoost reweights examples using the same rule as shown for AdaBoost, but the more general setting of continuous functions h_t requires to optimize α_t "manually" (not based on ϵ_t) to minimize the total example weight. The prediction rule is identical to the one shown in Fig. 2. If each h_t takes only values from $\{-1, +1\}$ the choice of asymmetric model weights made by Ada^{2}Boost reduces weights optimally. Differences to Real AdaBoost are that Ada^{2}Boost (i) uses boolean *crisp* base classifiers, adding confidence-like scores as part of the boosting procedure, and (ii) that it incorporates confidence ratings only in a very moderate form, which constrains the potential to overfit to the training data; when using the same fixed number of base models, Ada^{2}Boost selects ensemble models from almost the same search space as AdaBoost:

Proposition 3. *If estimated error rates are bounded away from zero the search space of AdaBoost and Ada^{2}Boost for boolean classification tasks are identical up to a constant additive offset.*

Proof. A model of the form given by eqn. (3) can be transformed into another classifier of the form

$$\hat{y} := \text{sign}\left(\alpha'_0 + \sum_{t=1}^{k} \alpha'_t h_i(x)\right), \tag{4}$$

with offset α'_0 and model weights $\alpha'_1, \ldots, \alpha'_k \in \mathbb{R}$ by computing for each model

$$\text{avg}_{0,t} := \frac{\alpha_t(+1) + \alpha_t(-1)}{2}, \qquad \alpha'_t := \alpha_t(+1) - \text{avg}_{0,t} \qquad \alpha'_0 := \sum_{t=1}^{k} \text{avg}_{0,t}.$$

The transformed model (4) is obviously identical to the original model.

Aiming to minimize generalization error it is quite natural to bound the error rates away from 0, e.g. by using Laplace or m-estimates for pure subsets.

Although the difference in expressiveness seems marginal, it allows Ada^{2}Boost to take more advantage of its base models, reflected by quicker weight reduction.

Theorem 1. *If $\epsilon = \epsilon^+ = \epsilon^-$ then the reweighting strategies of AdaBoost and Ada^{2}Boost are identical. Otherwise Ada^{2}Boost reduces the weights more quickly.*

Proof. AdaBoost reweights the $p + \overline{n} = 1 - \epsilon$ correctly classified examples multiplying with $\sqrt{\beta}$, and misclassified examples dividing by the same term. Hence the total weight $W_{t+1} = \sum_{i=1}^{|\mathcal{E}|} w_{t+1}(x_i, y_i)$ in iteration $t + 1$ can be computed as

$$W_{t+1} = \frac{1}{\sqrt{\beta}}((1 - \epsilon)W_t) + \sqrt{\beta}(\epsilon W_t) = 2W_t\sqrt{\epsilon \cdot (1 - \epsilon)} = 2W_t\sqrt{(p + \overline{n})(\overline{p} + n)}.$$

Ada^{2}Boost reweights the $p{+}n$ covered examples ($\mathcal{C}$) multiplying with $\sqrt{\beta(+1)}^{(\pm 1)}$. Since positives are divided by and negatives are multiplied with $\sqrt{p/n}$ the weight of $\mathcal{C}$ reduces from $W_t \cdot (p{+}n)$ to $W_t \left(p/\sqrt{p/n} + n\sqrt{p/n}\right) = 2W_t\sqrt{p \cdot n}$. The weight of $\overline{\mathcal{C}}$ changes analogously, applying the factor $\sqrt{\beta(-1)}^{(\pm 1)} = \sqrt{\overline{n}/\overline{p}}^{(\pm 1)}$ instead, so the new total weight for Ada^{2}Boost is $W_{t+1} = 2W_t \cdot \left(\sqrt{p \cdot n} + \sqrt{\overline{p} \cdot \overline{n}}\right)$.
We hence need to show $\sqrt{(p+\overline{n})(\overline{p}+n)} \geq \sqrt{p \cdot n} + \sqrt{\overline{p} \cdot \overline{n}}$, which follows from

$$\sqrt{(p+\overline{n})(\overline{p}+n)} \geq \sqrt{p \cdot n} + \sqrt{\overline{p} \cdot \overline{n}} \Leftrightarrow (p+\overline{n})(\overline{p}+n) \geq pn + \overline{pn} + 2\sqrt{pn\overline{pn}}$$
$$\Leftrightarrow p\overline{p} + n\overline{n} \geq 2\sqrt{pn\overline{pn}} \Leftrightarrow (p\overline{p})^2 + 2pn\overline{pn} + (n\overline{n})^2 \geq 4pn\overline{pn} \Leftrightarrow (p\overline{p} - n\overline{n})^2 \geq 0.$$

Both strategies yield the same result, iff $p\overline{p} = n\overline{n}$. This is equivalent to

$$\frac{p}{n} = \frac{\overline{n}}{\overline{p}} \Leftrightarrow \frac{p/(p+n)}{n/(p+n)} = \frac{\overline{n}/(\overline{n}+\overline{p})}{\overline{p}/(\overline{n}+\overline{p})} \Leftrightarrow \frac{1-\epsilon^+}{\epsilon^+} = \frac{1-\epsilon^-}{\epsilon^-} \Leftrightarrow \epsilon^+ = \epsilon^-$$

If n or $\overline{p}$ are 0, then either *both* error rates need to be 0, or one of the local error rates is undefined, because the subset contains no examples. The fact that ϵ is a weighted average of ϵ^+ and ϵ^- completes the proof.

The following proposition summarizes some useful properties of Ada^{2}Boost.

Proposition 4. *After Ada^{2}Boost reweights for the first time we have $P' = N'$. After each iteration t the subsets $\mathcal{C}_t$ and $\overline{\mathcal{C}}_t$ corresponding to model h_t are both stratified. The error rates ϵ_t^+ and ϵ_t^- of h_t are exactly $1/2$ with respect to w_{t+1}.*

Ada^{2}Boost performs especially well in in the case of conditionally independent base classifiers. Denoting with $P_t(\cdot)$ probabilities based on weights w_t it uses a product of β_t terms as defined in eqn. (2) to compute odds-ratio estimates

$$\widehat{\beta(x)} = \frac{P(y=+1 \mid h_1(x)...h_k(x))}{P(y=-1 \mid h_1(x)...h_k(x))} = \prod_{t=1}^{k} \frac{P_t(y=+1 \mid h_t(x))}{P_t(y=-1 \mid h_t(x))} \tag{5}$$

This happens to be identical to the estimate of NaïveBayes on top of the base model predictions $h_t(x)$, which yields the Bayes' optimal decision rule in this setting. This simple interpretation requires discrete prediction domains for base models. Even in cases where conditional independence is lacking Ada^{2}Boost continuously fits an additive model to the log-odds, similar to logistic regression, which suggests that it may yield good estimates of conditional class distributions.

3.3 An Analysis in PN Spaces

The reweighting strategy of Ada^{2}Boost is very similar to the stratification proposed in Sec. 2. In each iteration both the covered ($\mathcal{C}$) and the uncovered subsets ($\overline{\mathcal{C}}$) are stratified in the sense of Prop. 1.

These results suggest a reformulation of boosting in terms of stratification. The use of exponential or logarithmic functions (α_t values) seems to be unnecessarily complicated in this setting, because Ada^{2}Boost only requires the more

```
// Init with uniform weights:
Let $w_1(x,y) := 1$ for all $(x,y) \in \mathcal{E}$
// Train k base classifiers:
for $t = 1$ to $k$ do
    $h_t \leftarrow$ base learner$(\mathcal{E}, w_t)$
    // Compute odds ratios:
    $\beta'_t(+1) := p_t/n_t$, $\beta'_t(-1) := \overline{p}_t/\overline{n}_t$
    // Stratification:
    $w_{t+1}(x,y) := \frac{w_t(x,y)}{\sqrt{\beta'_t(h_t(x))}^y}$
end for
Output: $\hat{\beta}(x) = \prod_{t=1}^{k} \beta'_t(h_t(x))$
$\hat{P}(y = +1 \mid x) := \hat{\beta}(x)/(1 + \hat{\beta}(x))$
```

Fig. 3. Ada^{2}Boost for $y \in \{+1, -1\}$

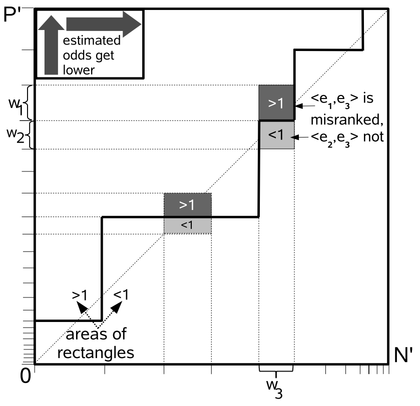


Fig. 4. Illustration for proof of theorem 2

intuitive β_t values. As a further simplification the algorithm uses β'_t, which *always* refers to the odds ratio, while β_t refers to the inverse odds ratio whenever a base classifier predicts negatively. Ada^{2}Boost (Fig. 3) boosts boolean base classifiers in a very simple fashion. In each iteration t another stratified (for $t > 1$, see Prop. 4) example set is presented to the base learner. The learner returns a base classifier $h_t : \mathcal{X} \rightarrow \{+1, -1\}$ that partitions the example set into "unstratified" subsets $\mathcal{C}$ and $\overline{\mathcal{C}}$; this automatically happens when maximizing accuracy [10]. The terms p_t to $\overline{n}_t$ denote the true positives to false negatives of model h_t using example weights w_t. For both partitions, $\mathcal{C}$ and $\overline{\mathcal{C}}$, the odds are computed and stored for later predictions, before stratifying the subsets separately, respecting the constraint stated in Prop. 1. When predicting a label the local odds are combined applying eqn. (5), which easily allows to derive probability estimates.

Fig. 5 shows a step of stratification for two partitions, e.g. for a classification rule in PN space. The two rectangles represent the performances of the two dual rules $(h_t(x) = +1) \rightarrow (y = +1)$ and $(h_t(x) = -1) \rightarrow (y = -1)$, where the slope of the diagonal is $\beta'_t(+1) = p/n$ in the former, and $\beta'_t(-1) = \overline{p}/\overline{n}$ in the latter. Stratification turns each of these rectangles into a square of equal size (hence Ada^2). This can also be thought of as a transformation of the underlying distribution (see Def. 2) that can be inverted precisely when making predictions.

It is interesting to note that the role of the base learner can as well be stated as to divide $\mathcal{E}$ into "unstratified" subsets, which are continuously stratified (conquered) by the meta-algorithm as long as the base learner succeeds.

In divide-and-conquer *rule learning* examples that are covered are removed from subsequent learning iterations. Similarly, boosting can be considered to probabilistically discard examples. Shifting both squares to the upper right, as depicted on the right side of Fig. 5, we reach at a visualization of boosting as nested PN-spaces. The weight lost by this transformation shows as the part of the axes of the original PN space below and left to the embedded PN space. As for AdaBoost, the total weight upper-bounds the number of misclassified examples, because only examples with a weight of greater than 1 are misclassified.

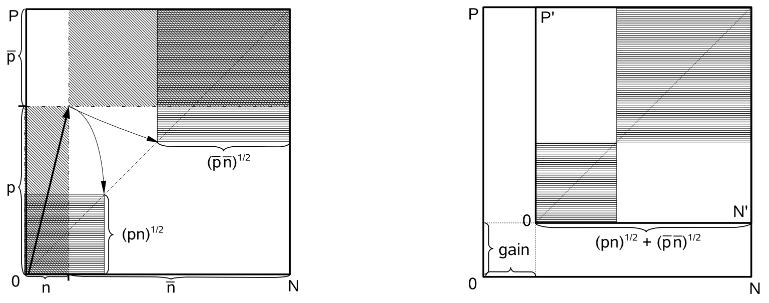


Fig. 5. Ada^{2}Boost transforms the boxes representing $\mathcal{C}$ and $\overline{\mathcal{C}}$ into squares (left) by reweighting. Moving these squares to the upper right (right) yields a PN subspace.

Prop. 2 states, that Ada^{2}Boost reduces the total weight as much as possible, while respecting the constraint to preserve the area of each subset in PN space. The advantage of this constraint is that the areas of the nested PN spaces reflect the progress in minimizing the ranking error at the same time.

Theorem 2. *The absolute ranking error (AOC*) of Ada^{2}Boost ensemble models for the original (unweighted) data is upper-bounded by the AOC* of the model for the inner nested PN space (reweighted example set).*

Proof. The crucial observation is, that final confidences and weights are closely related. A pair of examples (e^+, e^-) with weights w^+ and w^- will be misranked, iff the estimated confidence of being positive is higher for e^- than for e^+. The confidences are monotone in the estimated odds $\hat{\beta}(e^+)$ and $\hat{\beta}(e^-)$. Applying the reweighting scheme of Ada^{2}Boost recursively and computing $\hat{\beta}(x)$ we find that

$$w_{k+1}(x, y) := \prod_{t=1}^{k} \sqrt{\beta_t'(h_t(x))}^{(-y)} \quad \text{and} \quad \hat{\beta}(x) = \prod_{t=1}^{k} \beta_t'(h_t(x))$$

for an ensemble of size k. This implies $w^+ = \sqrt{1/\hat{\beta}(e^+)}$ and $w^- = \sqrt{\hat{\beta}(e^-)}$.

If $\hat{\beta}(e^-) > \hat{\beta}(e^+)$, then we have $(w^-)^2 > 1/(w^+)^2 \Leftrightarrow w^+ \cdot w^- > 1$. This means that each misranked pair (e^+, e^-) "occupies" a rectangle with an area of at least 1 in the inner nested PN space. All rectangles representing different pairs (e^+, e^-) are disjoint. Hence, if the nested PN space has a size of $P' \cdot N'$, then this quantity upper-bounds the AOC* of the ensemble for the original data.

When ordering examples by confidence, as for soft classifier ROC plots, weights of positives ascend along the P' axis, while weights of negatives descend along the N' axis (see Fig. 4). The areas of rectangles representing example pairs grow monotonically towards the upper left corner $(0, P')$. The border where areas become larger than 1 is depicted as a thick line. Example pairs share their estimated odds along the border, since $w^+ \cdot w^- = 1 \Rightarrow \hat{\beta}(e^+) = \hat{\beta}(e^-)$, so a threshold is associated to each point. Apart from the scale, Fig. 4 provides a ROC plot of the ensemble for the *reweighted* example set. By construction we expect the ensemble to perform as good as random guessing after reweighting, having an AOC* of $(P' \cdot N')/2$. In this case only *half* of the nested PN space represents misclassified pairs (areas ≥ 1), which also halves the AOC* upper-bound for the original data. The same argument applies for any other AOC* score.

The proof does not require base classifiers to provide *boolean* partitionings of $\mathcal{X}$, so theorem 2 holds for Real AdaBoost-like ensemble classifiers in general. The derived bounds are tighter than those provided in [4], based only on the worst case AOC* of $P' \cdot N'$. To the best of the author's knowledge theorem 2 provides the first AUC (AOC*) bound based on ranking performances after reweighting.

Corollary 1. *For an example set $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ a reduction of the initial weight of $|\mathcal{E}|$ to W results in an $AUC \geq 1 - W^2/(8 \cdot |\mathcal{E}^+| \cdot |\mathcal{E}^-|)$ if the AUC of the ensemble is at least* 1/2 *(random guessing) for the reweighted example set.*

For AdaBoost nested PN spaces are less intuitive, but also share the semantics of the quantity $P' \cdot N'$. Improved weight reduction strategies imply a more efficient reduction of this quantity, however. This becomes obvious when finally adding a classifier with constant predictions to each AdaBoost ensemble, which just stratifies the example set, so that the weight determines $P' \cdot N'$.

4 Evaluation

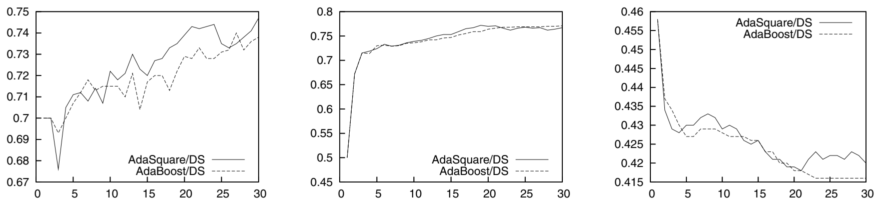
This section empirically evaluates generalization performances of the previously analyzed metrics on 4 benchmark datasets taken from the UCI library [11], and on a 10k sample of the quantum physics datasets from KDD Cup 2004. The evaluated metrics are accuracy (ACC), the area under the ROC curve (AUC), and finally, since evidence has been provided that Ada^{2}Boost may perform well in estimating conditional class probabilities, the root mean squared error (RMSE).

Ada^{2}Boost was implemented in YALE[1] [12] without any optimizations like LaPlace estimates. The Weka library [13] provides AdaBoost and decision stumps as base learners. Decision stumps are popular base classifiers and do not apply a greedy search themselves, which eases the evaluation of greedily operating boosting techniques. In the proposed experimental setting Real AdaBoost with "reasonable" confidence ratings for each decision stump yields the same predictions as Ada^{2}Boost. The focus of the evaluation is on AUC maximization and its relation to ACC optimization; for error rate minimization refer to [4] or [14].

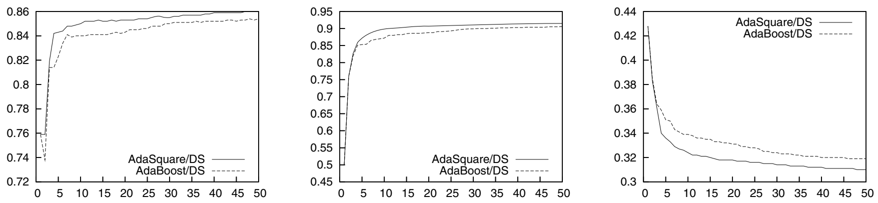
Fig. 6 shows the learning curves for different numbers of base models. Each point in the plots is the result of a ten-fold cross-validation. The results illustrate that boosting in fact maximizes all three considered metrics simultaneously, with AUC and RMSE providing finer-grained indicators of progress than ACC. Moreover, the moderate adaptation of AdaBoost does not only improve the ACC learning rate, as shown earlier, but leads to similar improvements for the metrics AUC and RMSE. The difference between AdaBoost and Ada^{2}Boost for Credit-G containing very few examples *and* attributes (16) are the smallest (if any), lying within half a standard deviation. For adult (32K examples) improvements are

[1] `http://yale.sf.net/`

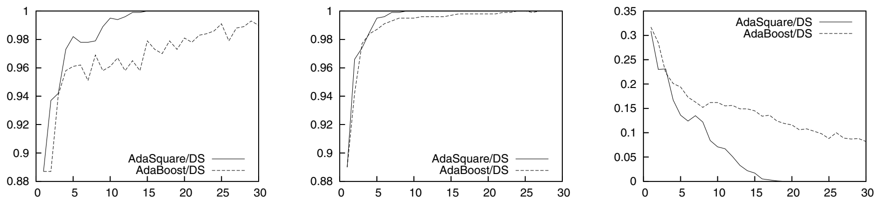
accuracy	**AUC**	**RMSE**



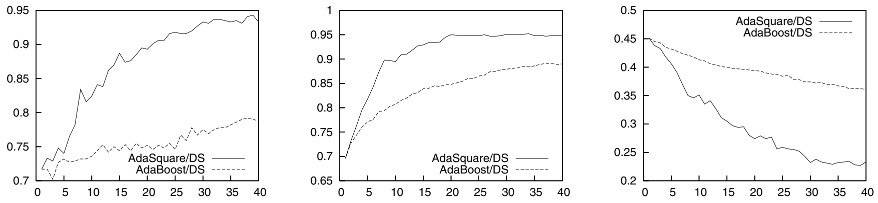
Credit-G data set (UCI), 16 attributes, 690 examples



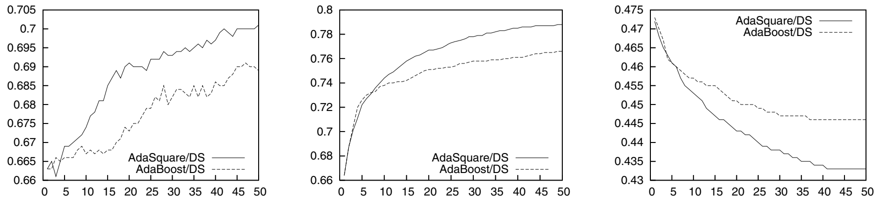
Adult data set (UCI), 15 attributes, 32K examples



Mushrooms data set (UCI), 23 attributes, 8K examples



Musk data set (UCI), 169 attributes, 476 examples



KDD Cup 2004, quantum physics data set, 80 attributes, 10K sample

Fig. 6. Generalization performances: AdaBoost vs. Ada^{2}Boost for decision stumps

small but significant: For all 3 metrics and e.g. 10 or 50 stumps it passes a t-test at a level of 2%. Advantages are much clearer for the remaining 3 data sets. On mushrooms, Ada^{2}Boost produces a perfect ranking with only 9, and perfect *soft* predictions with 19 stumps. In contrast, AdaBoost requires 24 stumps to rank perfectly and 100 stumps to reach an RMS of 2%. The curves differ most drastically for musk, having few examples but 169 attributes. The monotonicity of the AUC plots, well visible e.g. for the KDD Cup data, reflects the high robustness of this metric. AUC and RMSE behave similarly for all data sets.

5 Conclusions

A simplified confidence-rated AdaBoost variant based on stratification was analyzed. Visualizing the boosting process in nested PN spaces allowed to point out similarities between boosting and rule learning. Theoretical results have been provided that ease to understand (i) why boosting with accuracy as the objective function implicitly maximizes the AUC, and (ii) why confidence-rated strategies perform even better. Finally, a tighter than the commonly known bound for the AUC of boosting ensembles has been derived from a PN space analysis. An empirical study confirmed the results on implicit AUC maximization, indicating similar benefits for minimizing the root mean squared error.

References

1. Freund, Y., Schapire, R.R.: A decision–theoretic generalization of on-line learning and an application to boosting. Computer and System Sciences **55**(1) (1997)
2. Fürnkranz, J., Flach, P.: ROC 'n' Rule Learning – Towards a Better Understanding of Covering Algorithms. Machine Learning **58**(1) (2005)
3. Fawcett, T.: ROC Graphs: Notes and Practical Considerations for Researchers Tech report HPL-2003-4. HP Laboratories, Palo Alto, CA, USA (2004)
4. Schapire, R.E., Singer, Y.: Improved Boosting Using Confidence-rated Predictions. Machine Learning **37**(3) (1999)
5. Rudin, C., Cortes, C., Mohri, M., Schapire, R.E.: Margin-Based Ranking Meets Boosting in the Middle. In Proc. of COLT (2005)
6. Flach, P.A.: The Geometry of ROC Space: Understanding Machine Learning Metrics through ROC Isometrics. In Proc. of ICML (2003)
7. Rosset, S.: Model Selection via the AUC. In Proc. of ICML (2004)
8. Friedman, J.H., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting. Annals of Statistics (28) (2000)
9. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Boosting algorithms as gradient descent in function space. Technical report, Australian National University (1999)
10. Scholz, M.: Sampling-Based Sequential Subgroup Mining. In Proc. of KDD (2005)
11. Blake, C., Merz, C.: UCI Repository of machine learning databases (1998)
12. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: YALE: Rapid Prototyping for Complex Data Mining Tasks. In Proc. of KDD (2006)
13. Witten, I., Frank, E.: Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (2000)
14. Freund, Y., Mason, L.: The alternating decision tree learning algorithm. In Proc. of ICML (1999)

Prioritizing Point-Based POMDP Solvers[⋆]

Guy Shani, Ronen I. Brafman, and Solomon E. Shimony

Ben-Gurion University, Beer-Sheva, Israel

Abstract. Recent scaling up of POMDP solvers towards realistic applications is largely due to point-based methods such as PBVI, Perseus, and HSVI, which quickly converge to an approximate solution for medium-sized problems. These algorithms improve a value function by using *backup* operations over a single belief point. In the simpler domain of MDP solvers, prioritizing the order of equivalent backup operations on states is well known to speed up convergence.

We generalize the notion of prioritized backups to the POMDP framework, and show that the ordering of backup operations on belief points is important. We also present a new algorithm, Prioritized Value Iteration (PVI), and show empirically that it outperforms current point-based algorithms. Finally, a new empirical evaluation measure, based on the number of backups and the number of belief points, is proposed, in order to provide more accurate benchmark comparisons.

1 Introduction

Many interesting reinforcement learning (RL) problems can be modeled as partially observable Markov decision problems (POMDPs), yet POMDPs are frequently avoided due to the difficulty of computing an optimal policy. Research has focused on approximate methods for computing a policy (see e.g. [8],[7]). A standard way to define a policy is through a value function that assigns a value to each belief state, thereby also defining a policy over the same belief space. Sondik [9] show that this value function can be represented by a set of vectors and is therefore piecewise linear and convex.

A promising approach for computing value functions is the point-based method, where a value function is computed over a finite set of reachable belief points, in the hope that it would generalize well to the entire belief space. Generalization is possible through the use of the vector representation of a value function.

Improving a value function represented by vectors can be done by performing a *backup* operation over a single belief state, resulting in a new vector that can be added to the value function. Even though a vector is computed for a single belief state, it defines a value over the entire belief space, though this value may not be optimal for many belief states. A single backup operation can therefore, and in many cases does, improve the value function for numerous belief points. Backup operations are relatively expensive, and POMDP approximation algorithms can be improved by reducing the number of backup operations needed to approximate the optimal policy.

[⋆] Partially supported by the Lynn and William Frankel Center for Computer Sciences, and by the Paul Ivanier Center for Robotics and Production Management at BGU. Ronen Brafman is partially supported by NSF grants SES-0527650 and IIS-0534662.

For the simpler domain of Markov decision processes (MDPs), it was previously observed (e.g. [13]) that the order by which states are updated can change the convergence rate of the value function. For example, as the value for a state is influenced by the values of its successors it is more useful to execute a backup operation for a state only after values for its successors are computed. In an MDP it is also easy to find the set of predecessors for a given state making backward state space traversals possible. Such methods can be viewed as ordering backups by decreasing state *priorities*.

This paper aims at taking advantage of prioritized backups in a similar manner for POMDPs. Since a direct implementation of the techniques used for MDPs is not possible, this issue is nontrivial. First, one cannot efficiently find the set of predecessors for a belief state, which may have unbounded size. Second, a backup operation for a belief state potentially improves the value for many other belief states as well, and therefore affecting belief states that are not direct predecessors of the improved state.

Point-based methods tackle these problems by using only a finite subset of the belief space, reachable from the initial belief state. The main contribution of this paper is in showing how priorities can be computed for this finite set of belief points, and clearly demonstrating the resulting improvement in speed of convergence towards an approximate policy. We can hence provide prioritized versions for the PBVI [7] and Perseus [12] algorithms, as well as a new prioritized algorithm, Prioritized Value Iteration (PVI), which outperforms the unprioritized algorithms.

Another, methodological contribution of this paper is related to the schemes used for reporting experimental results evaluating the performance of point-based algorithms. Previous researchers have implemented their own algorithms and compared the results to previous published results, usually reporting Average Discounted Reward (ADR) as a measure of the quality of the computed policy, and convergence time, over well-known benchmarks. Observe that while the ADR of a policy is identical over different implementations, the convergence time is an insufficient measurement. Execution time for an algorithm is highly sensitive to variations in machines (CPU speed, memory capacity), selected platform (OS, programming language) and implementation efficiency. We comply with the commonly used result reporting scheme, but additional measures are also reported, which may provide meaningful comparisons in future publications.

2 Background and Related Work

2.1 MDPs, POMDPs and the Belief-Space MDP

A Markov Decision Process (MDP) is a tuple $\langle S, A, tr, R\rangle$ where S is a set of world states, A is a set of actions, $tr(s, a, s')$ is the probability of transitioning from state s to state s' using action a, and $R(s, a)$ defines the reward for executing action a in state s. An MDP models an agent that can directly observe its state in the environment.

A Partially Observable Markov Decision Process (POMDP) is a tuple $\langle S, A, tr, R, \Omega, O, b_0\rangle$ where S, A, tr, R are the same as in an MDP, Ω is a set of observations and $O(a, s, o)$ is the probability of observing o after executing a and reaching state s. A POMDP is a better model for real agents, such as robots, that do not have direct access to the current state of world but rather observe the world through a set of sensors that provide noisy observations. The agent hence must maintain a *belief* over its current state — a vector b of probabilities such that $b(s)$ is the probability that the agent is at state s. Such a vector is known as a belief state or *belief point*. b_0 defines the initial belief state before the agent has executed an action or received an observation.

Given a POMDP it is possible to define the belief-space MDP — an MDP over the belief states of the POMDP. The transition from belief state b to belief state b' using action a is deterministic given an observation o and defines the τ transition function. That is, we denote $b' = \tau(b, a, o)$ where:

$$b'(s') = \frac{O(a, s', o) \sum_s b(s) tr(s, a, s')}{pr(o|b, a)} \tag{1}$$

$$pr(o|b, a) = \sum_s b(s) \sum_{s'} tr(s, a, s') O(a, s', o) \tag{2}$$

Thus, τ can be computed in time $O(|S|^2)$.

2.2 Value Functions for POMDPs

It is well known that the value function V for the belief-space MDP can be represented as a finite collection of $|S|$-dimensional vectors known as α vectors. Thus, V is both piecewise linear and convex [9]. A policy over the belief space is defined by associating an action a to each vector α, so that $\alpha \cdot b = \sum_s \alpha(s) b(s)$ represents the value of taking a in belief state b and following the policy afterwards. It is therefore standard practice to compute a value function — a set V of α vectors. The policy π_V is immediately derivable using:

$$\pi_V(b) = \operatorname{argmax}_{a:\alpha_a \in V} \alpha_a \cdot b \tag{3}$$

The belief-space value function can be iteratively computed

$$V_{n+1}(b) = \max_a [b \cdot r_a + \gamma \sum_o pr(o|a, b) V_n(\tau(b, a, o))] \tag{4}$$

where $r_a(s) = R(s, a)$ is a vector representation of the reward function. The computation of the next value function $V_{n+1}(b)$ out of the current V_n (Equation 4) is known as a *backup* step, and can be efficiently implemented [2,7] by:

$$g^{\alpha}_{a,o}(s) = \sum_{s'} O(a, s', o) tr(s, a, s') \alpha^i(s') \tag{5}$$

$$g^b_a = r_a + \gamma \sum_o \operatorname{argmax}_{g^{\alpha}_{a,o}:\alpha \in V} b \cdot g^{\alpha}_{a,o} \tag{6}$$

$$backup(b) = \operatorname{argmax}_{g^b_a : a \in A} b \cdot g^b_a \tag{7}$$

Note that the $g^{\alpha}_{a,o}$ computation (Equation 5) does not depend on b and can therefore be cached for future backups. All the algorithms we implemented use caching to speed up backup operations. Without caching the execution time of the backup operation takes $O(|S|^2|V||\Omega||A|)$. In most benchmark POMDPs considered in our experiments, the number of actions and size of observation space are bounded, leading to a complexity of $O(|S|^2|V|)$ per backup step.

While it is possible to update V over the entire belief space, hence computing an optimal policy [2], the operation is computationally hard. Various approximation schemes attempt to decrease the complexity of computation, potentially at the cost of optimality.

The vector representation is suitable only for lower bounds over the optimal value function. When a value function is given using some other representation, such as a direct mapping between belief states and values, one can define the H operator, known as the Bellman update, that computes a value function update as:

$$Q_V(b,a) = b \cdot r_a + \gamma \sum_o pr(o|a,b)V_n(\tau(b,a,o)) \tag{8}$$
$$HV(b) = \max_a Q_V(b,a) \tag{9}$$

The computation time of the H operator is $O(T_v|S|^2|O||A|)$, where T_v is the computation time of a specific belief point value using the value function V.

2.3 Point Based Value Iteration

Computing an optimal value function over the entire belief space does not seem to be a feasible approach. A possible approximation is to compute an optimal value function over a subset of the belief space [5]. Note that an optimal value function for a subset of the belief space is no more than an approximation of a full solution. We hope, however, that the computed value function will generalize well for unobserved belief states.

Point-based algorithms [7,12,11] choose a subset of the belief points that is reachable from the initial belief state through different methods, and compute a value function only over these belief points.

Algorithm 1. PBVI

```
Function PBVI
1: $B \leftarrow \{b_0\}$
2: while true do
3:     $Improve(V,B)$
4:     $B \leftarrow Expand(B)$

Function Improve(V,B)
Input: $V$ — a value function
Input: $B$ — a set of belief points
1: repeat
2:     for each $b \in B$ do
3:         $\alpha \leftarrow backup(b)$
4:         $add(V,\alpha)$
5: until $V$ has converged

Function Expand(B)
Input: $B$ — a set of belief points
1: $B' \leftarrow B$
2: for each $b \in B$ do
3:     $Succ(b) \leftarrow \{b' | \exists a, \exists o\, b' = \tau(b,a,o)\}$
4:     $B' \leftarrow B' \cup argmax_{b' \in Succ(b)} dist(B,b')$
5: return $B'$
```

Point Based Value Iteration (PBVI) [7] (Algorithm 1), begins with b_0, and at each iteration computes an optimal value function for the current belief points set. After convergence the belief points set is expanded by adding the most distant immediate successors of the previous set. Following Pineua et al. we used the L_2 distance metric.

Spaan and Vlassis [12] explore the world randomly, gathering a set B of belief points, and then executing the Perseus[1] algorithm (Algorithm 2). Perseus appears to provide good approximations with small sized value functions rapidly. However, it is very stochastic due to the random selection of belief points and the random selection of backup operations. These random selections cause a high variation in performance and in more complicated problems may cause the algorithm to fail to converge at all.

Algorithm 2. Perseus

```
Input: $B$ — a set of belief points
1: repeat
2:   $\tilde{B} \leftarrow B$
3:   while $\tilde{B} \neq \phi$ do
4:     Choose $b \in \tilde{B}$
5:     $\alpha \leftarrow backup(b)$
6:     if $\alpha \cdot b \geq V(b)$ then
7:       $add(V, \alpha)$
8:     $\tilde{B} \leftarrow \{b \in \tilde{B} : \alpha \cdot b < V(b)\}$
9: until $V$ has converged
```

Algorithm 3. HSVI

```
Function HSVI
1: Initialize $\underline{V}$ and $\bar{V}$
2: while $\underline{V}(b_0) - \bar{V}(b_0) > \epsilon$ do
3:   $Explore(b_0, 0)$
```

```
Function Explore($b, d$)
Input: $b$ — a belief state
Input: $d$ — depth of recursion
1: if $\underline{V}(b) - \bar{V}(b) < \epsilon\gamma^{-d}$ then
2:   $a^* \leftarrow \text{argmax}_a Q_{\bar{V}}(b, a')$ (Equation 8)
3:   $o^* \leftarrow \text{argmax}_o(\bar{V}(\tau(b, a, o)) - \underline{V}(\tau(b, a, o))$
4:   Explore($\tau(b, a^*, o^*), d + 1$)
5:   $add(\underline{V}, backup(b, \underline{V}))$
6:   $\bar{V} \leftarrow HV(b)$
```

Smith and Simmons [10,11] present the Heuristic Search Value Iteration algorithm (HSVI - Algorithm 3) that maintains both an upper bound $\bar{V}$ and lower bound $\underline{V}$ over the value function. HSVI traverses the belief space following the $\bar{V}$ policy, greedily selecting successor belief points where the gap between the bounds is the largest, until some stopping criteria has been reached. Afterwards it executes backup and H operator updates over the observed belief points on the explored path in a reversed order. The policy computed by HSVI is based on $\underline{V}$. $\bar{V}$ is used only for exploration.

[1] We present here a single value function version of Perseus.

3 Prioritized Point Based Value Iteration

Point based algorithms compute a value function using α vectors by iterating over some finite set of belief points and executing a sequence of backup operations over these belief points. For each set of belief points there are many possible sequences of backup executions. As our goal is to approximate the value function as quickly as possible, we say that a backup sequence seq_1 is better than sequence seq_2 if seq_1 is shorter than seq_2, and produces a policy which is no worse than the one produced by seq_2.

We suggest creating (hopefully) better sequences using a heuristic that predicts useful backups. The heuristic computation must be efficient so that the overhead of computing the heuristic does not outweigh any savings achieved by performing fewer backups.

3.1 Prioritizing MDP Solvers

A comparable scheme used for prioritizing in MDP solvers, suggests performing the next backup on the MDP state that maximizes the Bellman error:

$$e(s) = max_a[R(s,a) + \sum_{s'} tr(s,a,s')V(s')] - V(s). \tag{10}$$

$e(s)$ measures the change in $V(s)$ from performing a backup. Wingate and Seppi[13] present a very simple version of prioritized value iteration for MDPs (Algorithm 4).

Algorithm 4. Prioritized Value Iteration for MDPs

```
1: $\forall s \in S, V(s) \leftarrow 0$
2: while $V$ has not converged do
3:     $s \leftarrow \text{argmax}_{s' \in S}\, e(s')$
4:     $backup(s')$
```

A key observation for the efficiency of their algorithm is that after a backup operation for state s, the Bellman error recomputation need be performed only for the predecessors of s $\{s' : \exists a, tr(s',a,s) \neq 0\}$.

3.2 Prioritizing POMDP Solvers

While the Bellman error generalizes well to POMDPs:

$$e(b) = max_a[r_a \cdot b + \sum_o pr(o|a,b)V(\tau(b,a,o))] - V(b) \tag{11}$$

there are two key differences between applying priorities to MDPs and POMDPs; First, a backup update affects more than a single state. A new vector usually improves the local neighborhood of its witness belief point. Second, the set of predecessors of a belief state cannot be efficiently computed, and its size is potentially unbounded.

Moreover, even supposing that some similarity metric for finding the neighborhood of a belief point were defined, and that computation of the predecessor set were only for the finite set of belief points we use, directly applying the approach would still be infeasible. In practice, algorithms such as Perseus, frequently converge to an optimal solution while computing fewer backups than the number of belief points in the finite set. Pre-computations such as similarity matrixes will take more time than the original algorithm they are designed to improve in the first place.

As we cannot find the set of belief states affected by the backup operation directly, we recompute the Bellman error for all belief states after every backup from scratch. When the number of belief points we use is relatively small this computation can be done without seriously damaging the performance. As the size of the problem — states, actions, observations and belief set size — increases, we can no longer afford the overhead of recomputing the Bellman error for all belief states.

We take a stochastic approach, sampling (uniformly, with repetitions) a subset of the belief points set and computing the Bellman error only for the sampled subset. If the subset does not contain a point with positive error, we sample again from the remaining subset until a belief point with positive error is found. If there is no belief point with positive Bellman error then the value function reached a fixed point and therefore converged to an optimal solution over the finite set of belief points, and cannot be improved using point-based backups.

3.3 Prioritizing Existing Algorithms

Prioritizing Perseus is straight forward. The choose step is implemented in Perseus as a uniform selection among any of the current belief points inside $\tilde{B}$. Prioritized Perseus uses $e(b)$ (Equation 11) to choose a belief point whose value can be improved the most.

PBVI improves its value function (Algorithm 1, line 3) by arbitrarily preforming backups overs belief points. We replace this inefficient computation of the Improve operation with our PVI algorithm (see Section 3.4). As the number of points used by PBVI is relatively small, no sampling was used when computing the Bellman error.

3.4 Prioritized Value Iteration

Finally, we present an independent algorithm — Prioritized Value Iteration (PVI). Like Perseus, PVI computes a value function over a pre-collected fixed set of belief points. However, Perseus operates in iterations over the set of belief points, attempting to improve all belief points between considering the same point twice. PVI considers at each step every possible belief state for improvement. It is likely, therefore, that some belief states will be backed up many times, while other belief states will never be used.

Algorithm 5 presents our PVI algorithm. Note however that while the algorithm described here is the clean version of PVI, in practice we implement the $argmax$ operation (line 2) using our sampling technique described above. If the prioritization metric is good, PVI executes a shorter sequence of backup operations. Indeed, experiments show that it uses significantly fewer backup operations than Perseus using our locally greedy Bellman error prioritization metric.

Algorithm 5. Prioritized Value Iteration

```
Input: $B$ — a set of belief points
1: while $V$ has not converged do
2:     $b^* \leftarrow \text{argmax}_{b \in B}\, e(b)$
3:     $\alpha \leftarrow backup(b^*)$
4:     $add(V, \alpha)$
```

4 Empirical Evaluations

4.1 Improved Evaluation Metrics

Previous researchers [1,7,11,12,6] limit their reported results to execution time, ADR and in some cases the number of vectors in the final value function.

Value function evaluation — Average discounted reward (ADR) is computed by simulating the agent interaction with the environment over a number of steps (called a *trial*) and averaging over a number of different trials: $\frac{\sum_{i=0}^{\#trials} \sum_{j=0}^{\#steps} \gamma^j r_j}{\#trials}$.

ADR is widely agreed as a good evaluation of the value of a value function. It can, however, present very noisy results when the number of trials or the number of steps is too small. To remove such noise we used a first order filter with weight 0.5. We stopped the execution once the filtered ADR has converged to a predefined target.

Execution time — As all algorithms discussed in this paper compute a value function using identical operations such as backups, τ function computations, and dot products ($\alpha \cdot b$), it seems that recording the number of executions of those basic building blocks of the algorithm is more informative than just reporting CPU time.

Memory — The size of the computed value function and the total amount of maintained belief points are good estimates for the memory capacity required for the computation of the algorithm.

4.2 Experimental Setup

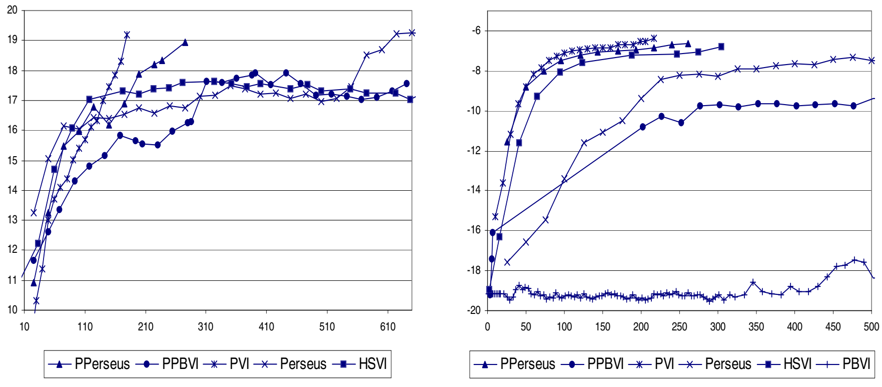
In order to test our prioritized approach, we tested all algorithms on a number of benchmarks from the point-based literature: Hallway, Hallway2 [4] (two maze navigation problems), TagAvoid [7] (a robot must capture another escaping robot) and RockSample [10] (a robot identifies and visits valuable rocks on a map). Table 2 contains the problem measurements for the benchmarks including the size of the state space, action space and observation space, the number of belief points in the set $|B|$ used for Perseus, Prioritized Perseus and PVI, and the ADR measurement error over $10,000$ trials.

We implemented in Java a standard framework that incorporated all the basic operators used by all algorithms such as vector dot products, backup operations, τ function and so forth. All reported results were gathered by executing the algorithms on identical machines — x86 64-bit machines, dual-proc, processor speed 2.6Ghz, 4Gb memory, 2Mb cache, running linux and JRE 1.5.

As previous researchers have already shown the maximal ADR achievable by their methods, we focus our attention on convergence speed of the value function to the reported ADR. We executed all algorithms, interrupting them from time to time in order to compute the efficiency of the current value function using ADR over 5000 trials. Once the filtered ADR has reached the same level as reported in past publications execution was stopped. The reported ADR was then measured over additional $10,000$ trials (error in measurement is reported in Table 2).

We pre-computed 5 different sets of belief points for each problem, by simulating an interaction with the system following the Q_{MDP} policy with an ϵ-greedy exploration factor ($\epsilon = 0.1$). These were then used for algorithms that require a given set of belief states B — Perseus, Prioritized Perseus and PVI. For each such belief points set we ran 5 different executions with different random seeds resulting in 25 different runs for each stochastic method. The belief points set size for each problem is specified in Table 2. Using Q_{MDP} for gathering B allowed us to use a smaller belief set than [12].

Algorithms that are deterministic by nature — PBVI, Prioritized PBVI and HSVI — were executed once per problem.



(a) (b)

Fig. 1. Convergence on the Rock Sample 5,5 problem (a) and the Tag Avoid problem (b). The X axis shows the number of backups and the Y axis shows ADR.

4.3 Results

Table 1 presents our experimental results. For each problem and method we report: the resulting ADR, the size of the final value function ($|V|$), the CPU time until convergence, the number of backups, of $g^{\alpha}_{a,o}$ operations, of computed belief states, of τ function computations, and of dot product operations.

The reported numbers do not include the repeated expensive computation of the ADR, or the initialization time (identical for all algorithms). Results for algorithms that require a pre-computed belief space do not include the effort needed for this pre-computation. We note, however, that it took only a few seconds (less than 3) to compute the belief space over all problems.

To illustrate the convergence of the algorithms we have also plotted the convergence of the ADR vs. the number of backups an algorithm preforms in Figure 1. The graphs

Table 1. Performance measurements. The algorithms that executed fewer backups and converged faster are bolded.

Method	ADR	$\lvert V\rvert$	Time (secs)	# Backups	$\#g^{\alpha}_{a,o}$ x 10^6	# belief states x 10^4	$\#\tau$ x 10^3	# $\alpha \cdot b$ x 10^6
Hallway								
PVI	0.517±0.0027	144±32	**75**±32	**504**±107	3.87±1.75	1.99±0.04	4.8±9.8	13.11±5.19
PPerseus	0.517±0.0025	173±43	126±47	607±166	5.52±2.95	1.99±0.04	4.8±9.8	26.87±9.2
Perseus	0.517±0.0024	466±164	125±110	1456±388	31.56±27.03	0.03±0	0±0	32.07±27.16
PPBVI	0.519	235	95	725	9.09	1.49	13.45	25.72
PBVI	0.517	253	118	3959	31.49	1.49	15.79	31.69
HSVI	0.516	182	314	634	5.85	3.4	34.52	6.67
Hallway2								
PVI	0.344±0.0037	234±32	75±20	262±43	2.59±0.84	2.49±0.11	5.47±9.99	6.96±2.01
Pperseus	0.346±0.0036	273±116	219±155	343±173	4.76±5.48	2.49±0.11	5.25±9.99	18.97±12.79
Perseus	0.344±0.0034	578±95	134±48	703±120	17.03±6.08	0.03±0	0±0	17.31±6.13
PPBVI	0.347	109	**59**	**137**	0.61	2.03	10.77	4.22
PBVI	0.345	128	76	1279	7.96	1.52	5.59	8.02
HSVI	0.341	172	99	217	1.56	2.11	11.07	1.81
Tag Avoid								
PVI	-6.467±0.19	204±38	**40**±12	211±38	0.42±0.19	0.16±0	0.5±1.02	0.95±0.25
PPerseus	-6.387±0.18	260±43	105±26	265±44	5.27±1.8	1.73±0.02	5.68±11.59	12.82±3.01
Perseus	-6.525±0.20	365±69	212±174	11242±10433	28.69±32.09	0.04±0	0±0	30.78±33.96
PPBVI	-6.271	167	50	**168**	2.09	0.41	32.11	2.45
PBVI	-6.6	179	1075	21708	407.04	0.41	56.53	409.5
HSVI	-6.418	100	52	304	0.5	0.29	1.74	0.53
Rock Sample 4,4								
PVI	17.725±0.32	231±41	4±2	232±42	0.36±0.14	0.41±0.01	1.17±2.38	1.74±0.42
PPerseus	17.574±0.35	229±26	5±2	228±27	0.34±0.08	0.41±0.01	1.17±2.38	1.91±0.29
Perseus	16.843±0.18	193±24	158±33	24772±5133	59.96±13.06	0.05±0	0±0	66.52±14.25
PPBVI	18.036	256	229	265	0.62	2.43	55.31	9.46
PBVI	18.036	179	442	52190	113.16	1.24	35.47	119.8
HSVI	18.036	123	**4**	**207**	1.08	0.1	1.17	1.09
Rock Sample 5,5								
PVI	19.238±0.07	357±56	**21**±7	**362**±63	0.99±0.36	0.46±0.01	1.37±2.79	3.39±0.87
PPerseus	19.151±0.33	340±53	20±6	339±53	0.88±0.28	0.46±0.01	1.37±2.79	3.5±0.73
Perseus	19.08±0.36	413±56	228±252	10333±9777	60.34±66.62	0.05±0	0±0	63.17±69.21
PPBVI*	17.97	694	233	710	4.95	0.95	17.97	18.12
PBVI*	17.985	353	427	20616	72.01	0.49	11.28	75.64
HSVI	18.74	348	85	2309	10.39	0.26	2.34	10.5
Rock Sample 5,7								
PVI	22.945±0.41	358±88	**89**±34	359±89	1.28±0.64	0.29±0.01	0.73±1.49	2.98±1.16
Pperseus	22.937±0.70	408±77	118±37	407±77	1.61±0.59	0.29±0.01	0.73±1.49	4.09±0.98
Perseus	23.014±0.77	462±70	116±31	1002±195	5.18±1.9	0.02±0	0±0	5.36±1.93
PPBVI*	21.758	255	117	**254**	0.61	0.23	2.71	1.59
PBVI*	22.038	99	167	2620	3.05	0.15	1.66	3.23
HSVI	23.245	207	156	314	0.83	0.71	4.2	0.88

contain data collected over separate executions with fewer trials (500 instead of 10000) so Table 2 is more accurate.

HSVI is the only method that also maintains an upper bound over the value function ($\bar{V}$). Table 3 contains additional measurements for the computation of the upper bound: the number of points in $\bar{V}$, the number of projections of other points onto the upper bound, and the number of upper bound updates ($HV(b)$ — Equation 9).

PBVI and PPBVI failed in two cases (Rock Sample 5,5 and 5,7 — marked with an asterix) to improve the reported ADR even when allowed more time to converge.

4.4 Discussion

Our results clearly show that an informed choice of the order by which backups are preformed over a predefined set of points improves the convergence speed. The most significant result is that our new algorithm, PVI, is among the quickest to converge in

Table 2. Benchmark problem parameters

Problem	$\|S\|$	$\|A\|$	$\|O\|$	$\|B\|$	ADR Error
Hallway	61	5	21	250	±0.0015
Hallway2	93	5	17	300	±0.004
Tag Avoid	870	5	30	350	±0.045
Rock Sample 4,4	257	9	2	500	±0.075
Rock Sample 5,5	801	10	2	500	±0.3
Rock Sample 5,7	3201	12	2	500	±0.25

Table 3. Upper bound measurements for HSVI

Problem	$\|\bar{V}\|$	**#$\bar{V}$(b)**	**#HV(b)**	$\|B\|$
Hallway	423	106132	1268	523
Hallway2	232	37200	434	171
Tag Avoid	1101	29316	1635	248
Rock Sample 4,4	344	6065	414	176
Rock Sample 5,5	801	101093	6385	1883
Rock Sample 5,7	3426	9532	628	268

all but one test-bed (in Hallway2 it is outperformed only by PPBVI). The efficiency of PVI's backup choices shows up nicely in Figure 1, where we see the steep improvement of PVI. Its improvement seems to be the steepest among the algorithms tested, indicating that it is a good choice for a fast approximation algorithm.

We can also see that, in general, prioritization helps each specific algorithm. We see it clearly in the case of PBVI – its running time is always faster with prioritization – whereas for Perseus there is one domain (Hallway2) in which the prioritized version is significantly slower, and two domains where the performance is virtually the same (Hallway and Rock Sample 5,7).

We can see an even more pronounced effect of prioritization on the number of backups, both between the two version of PBVI and Perseus, and with respect to PVI. In all these cases, there is an order of magnitude reduction in the number of backup operations when the next backup to preform is chosen in an informed manner. However, we also see that there is a penalty we pay for computing the Bellman error, so that the saving in backups does not fully manifest in execution time. Nevertheless, this investment is well worth it, as the overall performance improvement is clear. Note that the ADR to which the different algorithms converge is not identical, but the differences are minor, never exceeding 2%, making all ADRs shown equivalent, for all practical purposes.

In many case, HSVI executes less backups than other algorithms. Indeed, one may consider HSVI's selection of belief space trajectories as a form of prioritization metric. As such, we note that in most cases our form of backup selection exhibits superior runtime to HSVI, even when the number of backups HSVI uses is smaller. This is due to the costly maintenance of the value function upper bound.

5 Conclusions

This paper demonstrates how point-based POMDP solvers such as PBVI and Perseus can greatly benefit from intelligent selection of the order of backup operations, and that such selection can be performed efficiently, so that the algorithms' overall performance improves. It provides an extensive experimental analysis of different aspects of the performance of current point-based algorithms as well as their prioritized versions on popular domains from the literature. It also presents an independent algorithm — Prioritized Value Iteration (PVI) — that outperforms current point-based algorithm on a large set of benchmarks converging faster toward comparable values of ADR.

Given that point-based algorithms are the methods of choice for approximately solving POMDPs, PVI appears to be the fastest current approximation algorithm for POMDPs.

All the prioritized algorithms described in this paper use the same heuristic measure, the Bellman error, to decide on the sequence of backups. The method for selecting the order of backups using the Bellman error is pointwise greedy. While this choice may be optimal in the context of MDPs, in the context of POMDPs it does not take into account the possible improvement of a backup over other belief points as well. It is quite likely that executing a backup that improves a region of the belief space rather than a single belief point may have better influence over the convergence of the value function. Thus, future work should examine other possible heuristic functions that take this issue into account. The Bellman error is also expensive to compute, forcing us to estimate only a sampled subset of the belief points – this was very noticeable in our experiments. This implies that cheaper alternatives that lead to similar quality of backup selection may lead to algorithms that are an order of magnitude faster than current algorithms.

Another possible direction for future research is the choice of belief points [3] different algorithms use. Point-based algorithms use different methods for selecting belief points, and better techniques can probably enhance the performance of these algorithms.

References

1. R. I. Brafman. A heuristic variable grid solution method for pomdps. In *AAAI'97*, 1997.
2. A. R. Cassandra, M. L. Littman, and N.L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In *UAI'97*, pages 54–61, 1997.
3. M. Izadi, D. Precup, and D. Azar. Belief selection in point-based planning algorithms for pomdps. In *AI'06*, 2006.
4. M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *ICML'95*.
5. W. S. Lovejoy. Computationally feasible bounds for partially observable markov decison processes. *Operations Research*, 39:175–192, 1991.
6. S. Paquet, L. Tobin, and B. Chaib-draa. Real-time decision making for large pomdps. In *AI'2005*, 2005.
7. J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, August 2003.
8. P. Poupart and C. Boutilier. VDCBPI: an approximate scalable algorithm for large POMDPs. In *NIPS 17*. MIT Press, 2004.
9. R. Smallwood and E. Sondik. The optimal control of partially observable processes over a finite horizon. *Operations Research*, 21, 1973.
10. T. Smith and R. Simmons. Heuristic search value iteration for pomdps. In *UAI 2004*, 2004.
11. T. Smith and R. Simmons. Point-based pomdp algorithms: Improved analysis and implementation. In *UAI 2005*, 2005.
12. M. T. J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *JAIR*, 24:195–220, 2005.
13. D. Wingate and K. D. Seppi. Prioritization methods for accelerating mdp solvers. *JMLR*, 6:851–881, 2005.

Graph Based Semi-supervised Learning with Sharper Edges

Hyunjung (Helen) Shin[1][2], N. Jeremy Hill[3], and Gunnar Rätsch[1]

[1] Friedrich Miescher Laboratory, Max Planck Society, Spemannstrasse. 37, 72076 Tübingen, Germany
[2] Dept. of Industrial & Information Systems Engineering, Ajou University, San 5, Wonchun-dong, Yeoungtong-gu, 443–749, Suwon, Korea
[3] Max Planck Institute for Biological Cybernetics, Spemannstrasse. 38, 72076 Tübingen, Germany
{shin, jez, raetsch}@tuebingen.mpg.de
http://www.kyb.tuebingen.mpg.de/∼shin

Abstract. In many graph-based semi-supervised learning algorithms, edge weights are assumed to be fixed and determined by the data points' (often symmetric) relationships in input space, without considering directionality. However, relationships may be more informative in one direction (e.g. from labelled to unlabelled) than in the reverse direction, and some relationships (e.g. strong weights between oppositely labelled points) are unhelpful in either direction. Undesirable edges may reduce the amount of influence an informative point can propagate to its neighbours – the point and its outgoing edges have been "blunted." We present an approach to "sharpening" in which weights are adjusted to meet an optimization criterion wherever they are directed towards labelled points. This principle can be applied to a wide variety of algorithms. In the current paper, we present one ad hoc solution satisfying the principle, in order to show that it can improve performance on a number of publicly available benchmark data sets.

1 Introduction

Given sets of labelled and unlabelled data points, the task of predicting the missing labels can under some circumstances be aided by the information from unlabelled data points, for example by using information about the *manifold structure* of the data in input space. Many state-of-the art methods implement a *semi-supervised learning* (SSL) approach in that they incorporate information from unlabelled data points into the learning paradigm—see [3,6,24,23,17,5]. Our focus will be on a graph-based SSL approach. Despite their many differences as regards both guiding philosophy and performance, one thing common to most algorithms is the use of a matrix of values representing the pairwise relationships between data points. In graph-based SSL, the matrix of edge weights often denoted as W reflects the points' *influence* on each other,[1] which is an

[1] Many such systems are equivalent to a form of *spreading activation network* in which information is propagated around the graph.

inherently directional concept. The graph may therefore in principle be asymmetric. It is typically a sparse matrix. By contrast, in kernel-based methods like the TSVM [21,13,10], the kernel K denotes the points' *similarity* to each other, an intrinsically symmetrical property.

When adopting the kernel approach we can utilize the recent approaches of *learning the kernel matrix* [9,15,8,20,18]. In particular, the methods of [22] and [1] are focused on the use of unlabelled as well as labelled data. Using a kernel method requires that the similarity matrix satisfy the conditions of positive definiteness and symmetry to be a valid kernel [16]. It will often be a dense matrix. Most kernel learning methods are computationally demanding because of the operations involved on dense matrices–simply computing the product of two dense matrices already takes $O(n^3)$. It is possible to fit graph-based representations of pairwise relationships into a kernel-learning framework. One can directly calculate K from a graph using the diffusion kernel method [14], but this generally requires fairly expensive computation. Alternatively one can simply define similarity from the outset in terms of the graph, taking a simple formula such as $K = W^\top W$—note that this already entails a decrease in sparseness.

One of the merits of graph-based SSL lies in its computational efficiency: learning can often be done by solving a linear system with a sparse matrix W, which is nearly linear in the number of non-zero elements in W [19]. To preserve this advantage, it will be desirable that learning or manipulating W be achieved directly, without going via the route of learning a graph-based kernel matrix. To the best of our knowledge there have been relatively few approaches to learning the weights W of a graph, *Zhu et al* [24]'s being a notable exception. They address the issue of manipulating the edge weights, by a computationally intensive procedure for learning the scaling parameters of the Gaussian function that best aligns W with the data. The width parameters reflect the importance of input features, which makes their approach useful as a *feature selection* mechanism.

In this paper, we present a method which is immediately applicable to the weight matrix W. The proposed method is based on the following intuition. In an undirected graph, all connections are reciprocated and so the matrix of edge weights W is symmetric. However, when W describes relationships between labelled and unlabelled points, it is not necessarily desirable to regard all such relationships as symmetric. Some edges may convey more useful information in one direction (e.g. from labelled to unlabelled) than in the reverse direction. Propagating activity in the reverse direction, from unlabelled to labelled, may be harmful since it allows points about which information is uncertain to corrupt the very source of information in the system. Since we are already using the language of "points" and "edges", we will say that this causes the point and its outgoing edges to be "blunted", reducing their effectiveness. There are many problem settings (for example protein function prediction and other applications in the field of bio-informatics) in which (a) there is a high degree of certainty about the input-space representation of each labelled point and its label, and (b) the number of labelled points is very low. In such a situation, it seems intuitively desirable to avoid blunting, to preserve the effectiveness of the precious sources of information. Propagation of information *between* unlabelled points is a different issue—while some edges of the graph may be more helpful than others in solving the overall problem, a priori we do not know which these might be. Allowing the unlabelled points to harmonize themselves with their neighbours (implementing the assumption of *smoothness* common to most such learning approaches) is a desirable process.

To confirm this intuition, we begin with the well-known graph-based SSL formulation of [2] using Tikhonov regularization. First, we re-formulate the objective function in terms of W. Blockwise consideration of the weight matrix will allow us to state a condition which solutions W must satisfy if the objective function is to be optimized—there are many such solutions, some of which will be trivial and not lead to learning. Exploring the class of solutions, and developing a basis for comparison of their potential generalization ability, is beyond the scope of this paper and is left as an open problem. However, we propose one very simple specific solution, concordant with the logic already stated. Blockwise analysis of the inverse matrix used to make predictions will show the implications of this solution for the unlabelled points. This in turn makes clear the link between the Tikhonov regularization formulation we started with and the harmonic function solution to the Gaussian random field formulation as presented by [24].

The paper is organized as follows. In section 2, we briefly introduce the graph-based SSL algorithm under consideration. In section 3, we present the proposed idea in detail, and provide an ad hoc solution as a preliminary work, showing the connection to an earlier work based on harmonic function. In section 4, we show experimental results: illustrating the effects before and after the removal of the undesired weights. We summarize and conclude in section 5.

2 Graph-Based Semi-supervised Learning

A data point $\boldsymbol{x}_i$ $(i = 1, \ldots, n)$ is represented as a node i in a graph, and the relationship between data points is represented by an edge where the connection strength from each node j to each other node i is encoded in element w_{ij} of a weight matrix W. Often, a Gaussian function of Euclidean distance between points, with length scale σ, is used to specify connection strength:

$$w_{ij} = \begin{cases} \exp\left(- \frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)}{\sigma^2} \right) & \text{if} \quad i \sim j, \\ 0 & \text{otherwise.} \end{cases}$$

The $i \sim j$ stands for node i and j has an edge between them which can be established either by k nearest neighbors or by Euclidean distance within a certain radius r, $||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 < r$. [2] The labelled nodes have labels $\boldsymbol{y}_l \in \{-1, 1\}$,

[2] We represents scalars as lower case, vectors as boldface lower case, and martrices are uppercase. $\mathbf{0}$ (or $\mathbf{1}$) are a vector or matrix of variable-dependent size containing of all zeros (or ones).

while the unlabeled nodes have zeros $\boldsymbol{y}_u = \mathbf{0}$. Our algorithm will output an n-dimensional real-valued vector $\boldsymbol{f} = [\boldsymbol{f}_l^\top \ \boldsymbol{f}_u^\top]^\top = (f_1, \cdots, f_l, f_{l+1}, \cdots, f_{n=l+u})^\top$. which can be thresholded to make label predictions on $f_{l+1}, \ldots, f_n$ after learning. It is assumed that (a) f_i should be close to the given label y_i in labelled nodes, and (b) overall, f_i should not be too different from f_j of adjacent nodes ($i \sim j$). One can obtain $\boldsymbol{f}$ by minimizing the following quadratic functional [2]:

$$\sum_{i=1}^{l}(f_i - y_i)^2 + \mu \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 + \mu_u \sum_{i=l+1}^{n} f_i^2. \tag{1}$$

The first term corresponds to the *loss function* in terms of condition (a), and the second term represents the *smoothness* of the predicted outputs in terms of condition (b). The parameter μ (and μ_u) trades off loss versus smoothness. The last term is a regularization term to keep the scores of unlabelled nodes in a reasonable range. Alternative choices of smoothness and loss functions can be found in [6]. Hereafter, we focus on the special case of $\mu_u = 1$ [23] so that it is incorporated into the loss function. Then, the three terms degenerate to the following two:

$$\min_{\boldsymbol{f}} \ (\boldsymbol{f} - \boldsymbol{y})^\top(\boldsymbol{f} - \boldsymbol{y}) + \mu \boldsymbol{f}^T L \boldsymbol{f}, \tag{2}$$

where $\boldsymbol{y} = (y_1, \ldots, y_l, 0, \ldots, 0)^\top$, and the matrix L, called the *graph Laplacian matrix* [7], is defined as $L = D - W$ where $D = \text{diag}(d_i)$, $d_i = \sum_j w_{ij}$. Instead of L, the *normalized Laplacian*, $\tilde{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ can be used to get a similar result [7]. The solution of this problem is obtained as

$$\boldsymbol{f} = (I + \mu L)^{-1} \boldsymbol{y} \tag{3}$$

where I is the identity matrix.

The values of $\boldsymbol{f}$ are obtained by solving *a large sparse linear system* $\boldsymbol{y} = (I + \mu L)\boldsymbol{f}$. This numerical problem has been intensively studied, and there exist efficient algorithms, of which computational time is nearly linear in the number of nonzero entries in the coefficient matrix [19]. Therefore, the computation gets faster as the Laplacian matrix gets sparser.

3 Sharpening the Edges

3.1 Optimal Weight Matrix

Equation (3) gives us a closed-form solution that minimizes the objective function with respect to $\boldsymbol{f}$ for a given μ and fixed W. We now pose the question: what if W is not considered fixed? Is it possible to change some or all of the w_{ij} such that our algorithm performs better? We begin by re-formulating our objective function in terms of W. The smoothness term of (2) can also be expressed as

$$\mu \boldsymbol{f}^\top L \boldsymbol{f} = \boldsymbol{f}^\top \boldsymbol{y} - \boldsymbol{f}^\top \boldsymbol{f}, \tag{4}$$

by using $\boldsymbol{f}^\top(I+\mu L)\boldsymbol{f} = \boldsymbol{f}^\top\boldsymbol{y}$ which follows from $(I+\mu L)\boldsymbol{f} = \boldsymbol{y}$ from (3). Plugging (4) into (2) we have

$$\begin{aligned}\min_W \quad & (\boldsymbol{f}-\boldsymbol{y})^\top(\boldsymbol{f}-\boldsymbol{y}) + \mu\boldsymbol{f}^\top L\boldsymbol{f} \\ &= (\boldsymbol{f}-\boldsymbol{y})^\top(\boldsymbol{f}-\boldsymbol{y}) + \boldsymbol{f}^\top\boldsymbol{y} - \boldsymbol{f}^\top\boldsymbol{f} \\ &= \boldsymbol{y}^\top\boldsymbol{y} - \boldsymbol{y}^\top\boldsymbol{f} \\ &= \boldsymbol{y}^\top\boldsymbol{y} - \boldsymbol{y}^\top(I+\mu L)^{-1}\boldsymbol{y}. \end{aligned} \tag{5}$$

The constant term $\boldsymbol{y}^\top\boldsymbol{y}$ does not affect our optimization. Eliminating this constant term and negating, (5) becomes

$$\begin{aligned}\max_W \quad & d(W) = \boldsymbol{y}^\top(I+\mu L)^{-1}\boldsymbol{y}, \\ \text{s.t.} \quad & W \geq 0, \end{aligned} \tag{6}$$

where the non-negativeness constraint of W is introduced from the natural assumption of semi-supervised learning. Given an undirected graph, (6) is a convex problem since W and hence $(I+\mu L)^{-1}$ are positive symmetric—a function $z(A) = A^p$ of a positive symmetric matrix A is convex for $-1 \leq p \leq 0$ or $1 \leq p \leq 2$ [4]. Since we wish to consider asymmetric W, we cannot guarantee convexity. We could optimize W by a gradient descent method, the derivative of (6) with respect to w_{ij} being equal to by $\mu g_i(f_i - f_j)$, where $\boldsymbol{g} = (I+\mu L^\top)^{-1}\boldsymbol{y}$ and $\boldsymbol{f}$ is given as usual by (3). However, without imposing some additional constraint, one can see that the problem has trivial solutions since any diagonal W gives an optimal value by leading $(I+\mu L)^{-1}$ to the identity matrix. Removal of all the weights clearly does not fit the goals of learning since no generalization will be possible if no information is propagated between nodes.

Optimization must proceed under some constraints which reflect our prior assumptions about the problem. Consideration of the block structure of the problem will allow us to implement the intuition expressed in section 1 and indicate parts of the weight matrix W that can be optimized, without running foul of the "no free lunch" limitation. First, note that the most part of (6) that involve $\boldsymbol{y}_u$ simply vanish since $\boldsymbol{y} = \begin{bmatrix}\boldsymbol{y}_l \\ \boldsymbol{y}_u\end{bmatrix} = \begin{bmatrix}\boldsymbol{y}_l \\ \boldsymbol{0}\end{bmatrix}$. Accordingly, (6) is simplified by (3) and becomes

$$\begin{aligned}\max_W \quad & d(W) = \boldsymbol{y}_l^\top\boldsymbol{f}_l, \\ \text{s.t.} \quad & W \geq 0, \end{aligned} \tag{7}$$

which implies that the objective is simply to maximize the dot product of $\boldsymbol{y}_l$ and $\boldsymbol{f}_l$ with respect to weight matrix W. Given that all f_i must satisfy $-1 \leq f_i \leq 1$ [12,24], the solution that maximizes $d(W)$ must clearly satisfy $\boldsymbol{y}_l = \boldsymbol{f}_l$. Next, let us represent the weight matrix as a block matrix,

$$W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}.$$

Remember that, in the interpretation of W as a matrix of edge weights in a directed graph, the row index denotes the destination and the column index the source—so for example W_{lu} should be read as "the weights of the edges *from* unlabelled *to* labelled points, $u \rightarrow l$." For notational simplicity, let us also define $M = (I + \mu L)$, which has similar blockwise structure:

$$\begin{aligned} M &= \begin{bmatrix} M_{ll} & M_{lu} \\ M_{ul} & M_{uu} \end{bmatrix} \\ &= \begin{bmatrix} I + \mu(D_{ll} - W_{ll}) & -\mu W_{lu} \\ -\mu W_{ul} & I + \mu(D_{uu} - W_{uu}) \end{bmatrix}. \end{aligned} \tag{8}$$

Rearranging (3) in terms of $\boldsymbol{y}$ and writing it in a similar blockwise fashion, we obtain :

$$\begin{bmatrix} \boldsymbol{y}_l \\ \boldsymbol{y}_u \end{bmatrix} = \begin{bmatrix} M_{ll} & M_{lu} \\ M_{ul} & M_{uu} \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_l \\ \boldsymbol{f}_u \end{bmatrix} \tag{9}$$

Considering only the top row, we obtain the following relationship between $\boldsymbol{y}_l$ and $\boldsymbol{f}_l$:

$$\boldsymbol{y}_l = [I + \mu(D_{ll} - W_{ll})]\, \boldsymbol{f}_l - \mu W_{lu} \boldsymbol{f}_u. \tag{10}$$

from which we see by substituting the optimal solution $\boldsymbol{f}_l = \boldsymbol{y}_l$, that the condition

$$\boxed{(D_{ll} - W_{ll})\boldsymbol{y}_l = W_{lu} \boldsymbol{f}_u, \qquad (11)}$$

must hold. Equally, any solution that satisfies (11) is also optimal. This begins to show the role of the individual blocks of W in finding a solution. To express (11) solely in terms of block matrices, we use the block inverse of M,

$$\begin{aligned} &M^{-1} = \\ &\begin{bmatrix} M_{ll}^{-1} + M_{ll}^{-1} M_{lu} S^{-1} M_{ul} M_{ll}^{-1} & -M_{ll}^{-1} M_{lu} S^{-1} \\ -S^{-1} M_{ul} M_{ll}^{-1} & S^{-1} \end{bmatrix} \end{aligned} \tag{12}$$

where S is the *Schur complement* (see [4]) of M_{ll} in M,

$$S = M_{uu} - M_{ul} M_{ll}^{-1} M_{lu}. \tag{13}$$

With $\boldsymbol{f}_l = \boldsymbol{y}_l$, this yields

$$\underbrace{\left[M_{ll}^{-1} + M_{ll}^{-1} M_{lu} S^{-1} M_{ul} M_{ll}^{-1} - I\right]}_{(a)} \boldsymbol{y}_l = 0 \tag{14}$$

from which we see that there exist a potentially large class of solutions that satisfying this condition—the right hand side of (11) may be matched to the left through manipulation of any of the four blocks of W, to affect $\boldsymbol{f}_u$. So far, however, it seems intractable to calculate a general solution class for this complex system.

3.2 An Ad-Hoc Solution

In this paper, we present an ad hoc solution for the optimal condition (11) as a preliminary work. This simplest and blockwise form of solution will be used in our experiment to exemplify the effect of the condition. Many solutions can be obtained from non-zero matrix of (a) in (14), however, we focus on a subset of solutions by confining (a) to be $\mathbf{0}$.

As we mentioned earlier, any W as a form of diagonal matrix produces optimal value of (6) or (7). More concisely speaking, both W_{ll} and W_{uu} can be any diagonal matrices, while W_{lu} and W_{ul} should be null matrices. However, no one wants to compensate a null vector of $\boldsymbol{f}_u$ as a return for holding the condition (11). Thus, let us selectively decide which block matrix can be null matrix or diagonal, by examining

$$\boldsymbol{f}_u = -S^{-1} M_{ul} M_{ll}^{-1} \boldsymbol{y}_l. \tag{15}$$

First, note that M_{ul} should not be a null matrix thus $W_{ul} \neq \mathbf{0}$ from (8), $\boldsymbol{f}_u$ will be $\mathbf{0}$ otherwise. Second, M_{ll}^{-1} will not matter unless M_{ll} is singular, which implies we can regard W_{ll} as a diagonal matrix and W_{lu} as a null matrix. Then M_{ll} becomes an identity matrix. Next, let us take S^{-1} into consideration. With W_{ll} as a diagonal matrix, W_{lu} as a null matrix, and W_{ul} as a non-zero matrix, S defined in (13) will not be singular:

$$S = I + \mu(D_{uu} - W_{uu}).$$

This allows W_{uu} to be a diagonal matrix. However, one should be careful of setting W_{uu} be a diagonal matrix which will lead to

$$\boldsymbol{f}_u = \mu W_{ul} \boldsymbol{y}_l. \tag{16}$$

This means we cannot obtain the output prediction for the unlabelled data points unless they are *directly* connected to labelled points. Remembering that W is a sparse matrix in graph-based semi-supervised learning, we hardly expect full connection from labelled to unlabelled points. Therefore, we should not allow W_{uu} to be a diagonal matrix. Note that if W_{ul} is a full matrix, (16) stands for output prediction by *k-nearest neighbor* method. To summarize, by setting W_{ll} to a non-negative diagonal matrix (including null matrix) and W_{lu} to $\mathbf{0}$,

$$\boxed{W_s = \begin{bmatrix} \text{diagonal matrix} & \mathbf{0} \\ W_{ul} & W_{uu} \end{bmatrix}, \tag{17}}$$

we can satisfy the condition (11) but still expect to obtain meaningful output prediction for unlabelled data points

$$\boldsymbol{f}_u = \mu(I + \mu(D_{uu} - W_{uu}))^{-1} W_{ul} \boldsymbol{y}_l. \tag{18}$$

In spreading activation network terms, is equivalent to activity being propagated from labelled to unlabelled data *once* ($W_{ul}\boldsymbol{y}_l$) to set the initial condition for subsequent spreading activation among $u \leftrightarrow u$, analogous to (3) but now *excluding* the labelled points. This also has intuitive appeal. First, for labelled points, it assures $\boldsymbol{f}_l = \boldsymbol{y}_l$— *there is no loss of information on labelled data points*. By disconnecting unnecessary and unhelpful edges, we allow the labelled points and their outgoing edges to stay "sharp" in their influence on the rest of the network. Second, for unlabelled points, it preserves an important principle of SSL, namely *exploitation of the manifold structure inferred from unlabelled data points*, by keeping the edges, $u \leftrightarrow u$ and $l \rightarrow u$, of W.

3.3 Harmonic Functions Revisited

The condition (11) provides a link to the formulation of [24], which characterized semi-supervised learning in terms of a *harmonic function* solution to an energy minimization problem. Particularly, the solution (18) is very similar to their solution

$$\boldsymbol{f}_u = (D_{uu} - W_{uu})^{-1} W_{ul}\boldsymbol{y}_l,$$

to which our (18) converges as μ becomes arbitrarily large. But note that their optimization proceeds from the *a priori assumption* that labels should be reconstructed without loss, $\boldsymbol{f}_l = \boldsymbol{y}_l$. Unfortunately, in the general formulation (2) of semi-supervised learning, it is not natural to hold this assumption due to the smoothness term. In the light of that, (11) plays a role of bridge between two methods, [2] and [24]: we begin with the formulation of [2] and reach at the minimum-energy solution of [24] without the necessity of assuming $\boldsymbol{f}_l = \boldsymbol{y}_l$ a priori. Note that in our formulation hyperparameter μ naturally remains from (3) through to (18) and can be tuned to the needs of each particular learning problem.

4 Experiment

We compare the performance of the original solutions (3) with W and sharpened (18) with W_s on 5 real and artificial data sets that have been used as benchmarks for comparing the performance of semi-supervised learning algorithms by [5]. We used five of the nine data sets made available by the authors of [5] for the purposes of testing semi-supervised learning algorithms. The data sets encompass both artificial and real data in a number of different settings, and are summarized in table 1. More details, and the data sets themselves, are available at: `http://www.kyb.tuebingen.mpg.de/ssl-book/`. Each data set has binary labels, and comes with 24 pre-determined splits, i.e. sets of indices dictating which data points are to be labelled. Of these, 12 splits each contain 10 randomly chosen labelled points (at least one in each class), and the other 12 splits each contain 100 randomly chosen labelled points. For each data set, an initial undirected edge graph W was constructed by making a symmetrical connection between each point and its k nearest neighbours as measured by Euclidean separation in the input space, with k set either to 10 or to 100. Weights were then set

Table 1. Summary of the five benchmark data sets used

index	name	points	dims	comment
1	Digit1	1500	241	artificial images
2	USPS	1500	241	2s and 5s vs rest
3	COIL$_2$	1500	241	images
4	BCI	400	117	small, noisy
5	g241c	1500	241	artificial

for each edge according to the function $w_{ij} = \exp(-s_{ij}^2/\sigma^2)$ of edge length s_{ij}, with σ set either to 10 times or to 1 times the median length of the connected edges (the former setting ensured in practice that all connected weights were roughly equal, and close to 1, and the latter setting ensured some variation in the weights). For each of the available splits, we obtain solutions (3) and (18) for four different smoothing parameter values $\mu \in \{0.1, 1, 10, 100\}$, and record the ROC scores of the unlabelled outputs $\boldsymbol{f}_u$ with respect to the true labels.

The results are summarized in Fig.1. Each pair of subplots corresponds to one of the five data sets, with results from the splits with 10 labelled points on the left and from the splits with 100 labelled points on the right. The grey points show the comparison between the two methods for each of 192 runs (2 settings of k times 2 settings of σ times 4 settings of μ times 12 random splits). In addition, red crosses show the best setting for each method—performance of the sharpened method on the 12 splits under the $\{k, \sigma, \mu\}$ setting for which that method yielded the best mean ROC score across splits, against performance of the original method on the 12 splits under *its* best setting. We can see from Fig.1(a) that the sharpening modification leads to performance that is equal to or better than the original algorithm. In some cases the improvement is small in magnitude, but it is consistent in sign. For data set 2, 3 and 4, in particular, we clearly see that the majority of grey points lie above the diagonal, indicating that, for a randomly chosen hyperparameter setting among those explored, sharpening is very likely to result in easier model selection and improvements in performance. The sharpening modification tends to gain more improvement when more labelled points are given. In the subplots of the right column (of 100 labelled points), consistently across the random splits, the best performance obtained by the sharpened method is better than the best performance obtained by the original method. We illustrate the algorithms' hyperparameter dependence in Fig.1(b). From this representation we see that the sharpened method's performance is generally equal to or better than the original. We also see that, for data sets 2, 3 and 4, one of the sharpened method's advantages lies in its relative insensitivity to the values of smoothness-loss tradeoff parameter μ. This relative insensitivity is a desirable property in situations where correct hyperparameter selection is a hit-and-miss affair. Table 2 shows the best ROC scores and the results of the Wilcoxon signed-ranks test (see [11]). Considering the best averaged ROCs across the splits, the highest scores (the numbers in boldface in the second column) are obtained by the sharpened method in 9 out of the 10

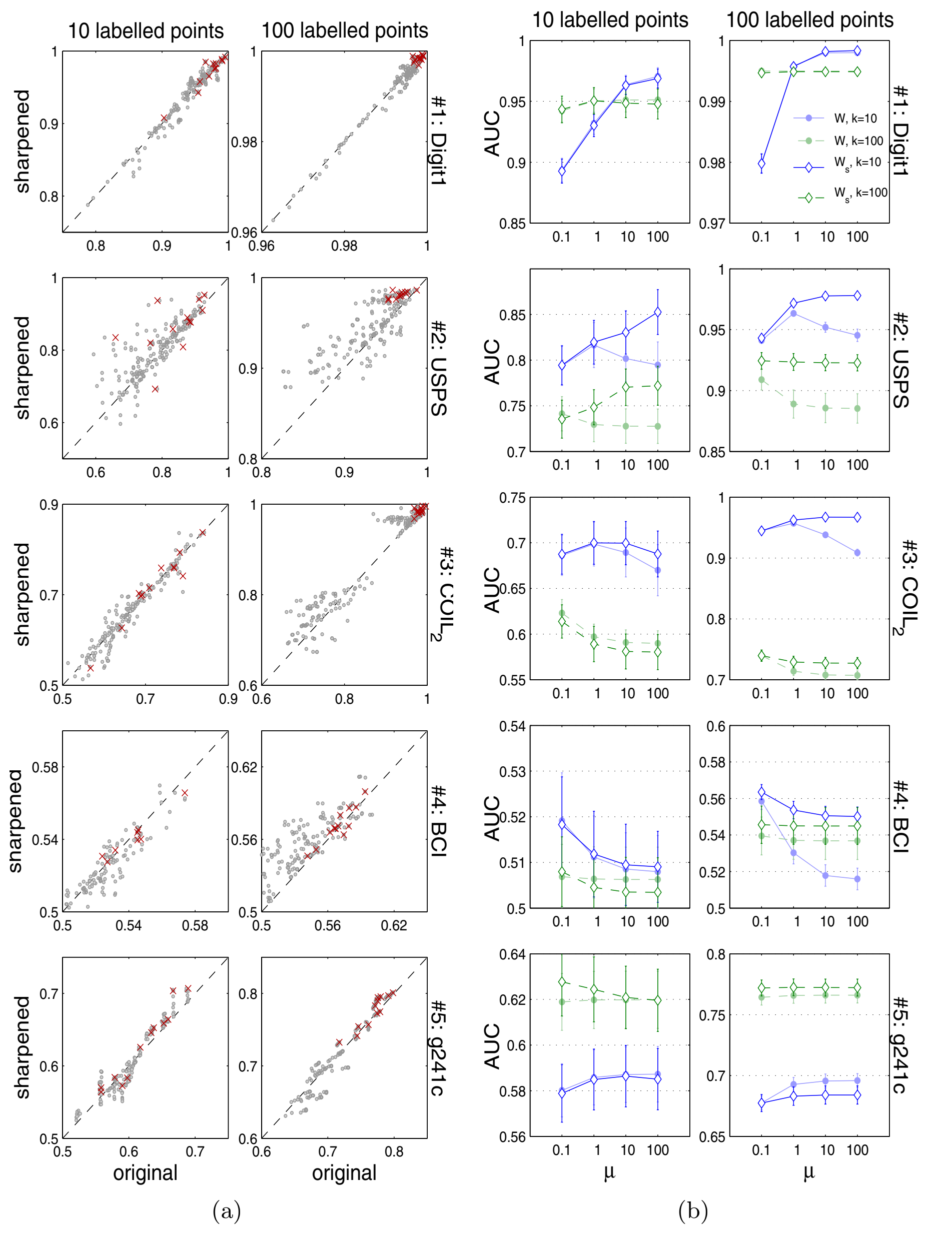


Fig. 1. Results: (a) ROC scores for the sharpened method against those for the original method, across 12 random splits of the data in each panel. Results are shown for all hyperparameter settings (grey points) and for each method's best hyperparameter setting (red crosses). (b) Hyperparameter dependence of the sharpened method using modified weight matrix W_s (open diamonds) and for the original method using W (filled circles). Mean ROC scores across the 12 splits are shown as a function of μ (on the abscissa) and k (blue–solid for $k = 10$, green–dashed for $k = 100$). Results are only shown for $\sigma = 10$ (results for $\sigma = 1$ follow a roughly similar pattern).

Table 2. Summary of the results for the five data sets

Datasets		Best ROC score (%)		Frequency of outperformance (#)		p-value
		original	sharpened	original	sharpened	
(1) `Digit1`	10 labelled	**97.13**	97.10	**200**	184	0.6280
	100 labelled	99.81	**99.84**			
(2) `USPS`	10 labelled	85.44	**87.60**	107	**277**	0.0000
	100 labelled	96.95	**98.83**			
(3) `COIL`$_2$	10 labelled	73.24	**74.49**	172	**212**	0.0006
	100 labelled	98.39	**98.50**			
(4) `BCI`	10 labelled	53.25	**53.39**	136	**248**	0.0000
	100 labelled	57.51	**58.29**			
(5) `g241c`	10 labelled	62.05	**63.09**	173	**211**	0.0564
	100 labelled	75.77	**77.51**			
Total				788	**1132**	0.0000

cases, with almost similar performance being attained by both methods in the remaining one. The third column compares the two methods in frequency of outperformance for the 384 (=2 ×192) paired ROC comparisons per dataset. In 4 out of the 5 datasets, the sharpened method outperformed the original. The p-values in the last column statistically present the significance of outperformance of the sharpened method.

5 Conclusion

In this paper, we present a simple yet efficient method for manipulating weight matrix W based on graph-based semi-supervised learning. By analyzing the objective function in blockwise fashion according to the four combinations of labelled and unlabelled points, we show an optimal condition for W that tells us which block can be manipulated, and how they may be manipulated, in order to enhance the flow of activation. This approach provides two main advantages without high computational cost or resorting to heuristics. For labelled points, it ensures that the predicted output equals its given label: *there is no loss of information on labelled data points.* For unlabelled points, it preserves the principle of semi-supervised learning: *prediction with manifold structure for unlabelled data points.* This allows us to enjoy the best of both worlds: improved performance due to "sharpening" of previously "blunted" labelled points and edges (as is also the case for [24]) and the ability to explore different smoothing settings in search of the best generalization performance (as in [2]).

For the sake of analytical convenience, the current method takes a very simple, conventional semi-supervised framework as its basis. However, incorporated into more sophisticated state-of-the-art algorithms, it has the potential to improve considerably on their original performance.

References

1. F.R. Bach and M.I. Jordan. Learning spectral clustering. *NIPS 16*, 2004.
2. M. Belkin, I. Matveeva, and P. Niyogi. Regularization and regression on large graphs. *Lecture Notes in Computer Science (In COLT)*, pages 624–638, 2004.
3. M. Belkin and P. Niyogi. Using manifold structure for partially labelled classification. *NIPS 15*, 2003.
4. S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.
5. O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning.* MIT Press, Cambridge, MA (in press), 2006.
6. O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. *NIPS 15*, 2003.
7. F.R.K. Chung. *Spectral Graph Theory.* Number 92 in Regional Conference Series in Mathematics. American Mathematical Society, Providence, RI, 1997.
8. K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. *NIPS 15*, 2003.
9. N. Cristianini, J. Shawe-Taylor, and J. Kandola. On kernel target alignment. *NIPS 14*, 2002.
10. T. De Bie and N. Cristianini. Convex method for transduction. *NIPS 16*, 2004.
11. J. Demsăr. Statistical comparisons of claissifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
12. P. Doyle and J. Snell. Random walks and electric networks. *Mathematical Association of America*, 1984.
13. T. Joachims. Transductive inference for text classification using support vector machines. *In Proc. ICML*, 1999.
14. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. *In Proc. ICML*, 2002.
15. G.R.G. Lanckriet, N. Cristianini, L.E. Ghaoui, P. Bartlett, and M.I. Jordan. Learning the kernel matrix with semi-definite programming. *In Proc. ICML*, 2002.
16. B. Schölkopf and A. J. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.
17. V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. *In Proc. ICML*, 2005.
18. S. Sonnenburg, G. Rätsch, S. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 2006. accepted.
19. D.A. Spielman and S.H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proc. of the 26th annual ACM symposium on Theory of computing*, pages 81–90. ACM Press, 2004.
20. K. Tsuda, G. Rätsch, and M.K. Warmuth. Matrix exponentiated gradient updates for on-line learning and bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005.
21. V. N. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, 1998.
22. Z. Zhang, D.Y. Yeung, and J.T. Kwok. Bayesian inference for transductive learning of kernel matrix using the tanner-wong data augmentation algorithm. *In Proc. ICML*, 2004.
23. D. Zhou, O. Bousquet, J. Weston, and B. Schölkopf. Learning with local and global consistency. *NIPS 16*, 2004.
24. X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. *In Proc. ICML*, 2003.

Margin-Based Active Learning for Structured Output Spaces

Dan Roth and Kevin Small

Department of Computer Science
University of Illinois at Urbana-Champaign
201 N. Goodwin Avenue, Urbana, IL 61801, USA
{danr, ksmall}@uiuc.edu

Abstract. In many complex machine learning applications there is a need to learn multiple interdependent output variables, where knowledge of these interdependencies can be exploited to improve the global performance. Typically, these structured output scenarios are also characterized by a high cost associated with obtaining supervised training data, motivating the study of active learning for these situations. Starting with active learning approaches for multiclass classification, we first design querying functions for selecting entire structured instances, exploring the tradeoff between selecting instances based on a global margin or a combination of the margin of local classifiers. We then look at the setting where subcomponents of the structured instance can be queried independently and examine the benefit of incorporating structural information in such scenarios. Empirical results on both synthetic data and the semantic role labeling task demonstrate a significant reduction in the need for supervised training data when using the proposed methods.

1 Introduction

The successful application of machine learning algorithms to many domains is limited by the inability to obtain a sufficient amount of labeled training data due to practical constraints. The *active learning* paradigm offers one promising solution to this predicament by allowing the learning algorithm to incrementally select a subset of the unlabeled data to present for labeling by the domain expert with the goal of maximizing performance while minimizing the labeling effort. One particularly appropriate family of machine learning applications for active learning is the scenario where there are multiple learning problems such that there is a specified relationship between the output variables of the individual classifiers, described as *learning in structured output spaces*. In such situations, the target applications are generally more complex than single classification tasks and the cost for supervised training data is correspondingly higher.

There are many applications of learning in structured output spaces across numerous domains, including the semantic role labeling (SRL) task [1]. The goal for SRL is, given a sentence, to identify for each verb in the sentence which constituents fill a semantic role and determine the type of the specified argument.

For the example sentence, "I left my pearls to my daughter-in-law in my will," the desired output is

$$[_{A0} \text{ I }][_{V} \text{ left }][_{A1} \text{ my pearls }][_{A2} \text{ to my daughter-in-law }][_{AM-LOC} \text{ in my will }],$$

where A0 represents the *leaver*, A1 represents the *item left*, A2 represents the *benefactor*, and AM-LOC is an adjunct indicating the location of the action. Examples of specifying structural relationships include declarative statements such as *every sentence must contain exactly one verb* or *no arguments can overlap.*

This paper describes a margin-based method for active learning in structured output spaces where the interdependencies between output variables are described by a general set of constraints able to represent any structural form. Specifically, we study two querying protocols and propose novel querying functions for active learning in structured output spaces: querying complete labels and querying partial labels. In the SRL example, these two protocols correspond to requiring the learner to request the labels for entire sentences during the instance selection process or single arguments, such as *my pearls*, respectively. We proceed to describe a particular algorithmic implementation of the developed theory based on the Perceptron algorithm and propose a mistake-driven explanation for the relative performance of the querying functions. Finally, we provide empirical evidence on both synthetic data and the semantic role labeling (SRL) task to demonstrate the effectiveness of the proposed methods.

2 Preliminaries

This work builds upon existing work for learning in structured output spaces and margin-based active learning. We first describe a general framework for modeling structured output classifiers, following the approach of incorporating output variable interdependencies directly into a discriminative learning model [2,3]. We then proceed by describing previous margin-based active learning approaches based on the output of linear classifiers [4,5].

2.1 Structured Output Spaces

For our setting, let $\mathbf{x} \in \mathcal{X}^{n_x}$ represent an instance in the space of input variables $\mathbf{X} = (X_1, \ldots, X_{n_x}); X_t \in \mathbb{R}^{d_t}$ and $\mathbf{y} \in \mathcal{C}(\mathcal{Y}^{n_y})$ represent a structured assignment in the space of output variables $\mathbf{Y} = (Y_1, \ldots, Y_{n_y}); Y_t \in \{\omega_1, \ldots, \omega_{k_t}\}$. $\mathcal{C} : 2^{\mathcal{Y}^*} \rightarrow 2^{\mathcal{Y}^*}$ represents a set of constraints that enforces structural consistency on $\mathbf{Y}$ such that $\mathcal{C}(\mathcal{Y}^{n_y}) \subseteq \mathcal{Y}^{n_y}$. A learning algorithm for structured output spaces takes m structured training instances, $\mathcal{S} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$ drawn i.i.d over $\mathcal{X}^{n_x} \times \mathcal{C}(\mathcal{Y}^{n_y})$ and returns a classifier $h : \mathcal{X}^{n_x} \rightarrow \mathcal{Y}^{n_y}$. This assignment generated by h is based on a global scoring function $f : \mathcal{X}^{n_x} \times \mathcal{Y}^{n_y} \rightarrow \mathbb{R}$, which assigns a score to each structured instance/label pair $(\mathbf{x}_i, \mathbf{y}_i)$. Given an instance $\mathbf{x}$, the resulting classification is given by

$$\hat{\mathbf{y}}_{\mathcal{C}} = h(\mathbf{x}) = \underset{\mathbf{y}' \in \mathcal{C}(\mathcal{Y}^{n_y})}{\operatorname{argmax}} f(\mathbf{x}, \mathbf{y}'). \tag{1}$$

The output variable assignments are determined by a global scoring function $f(\mathbf{x},\mathbf{y})$ which can be decomposed into local scoring functions $f_{y_t}(\mathbf{x},t)$ such that $f(\mathbf{x},\mathbf{y}) = \sum_{t=1}^{n_y} f_{y_t}(\mathbf{x},t)$. When structural consistency is not enforced, the global scoring function will output the value $f(\mathbf{x},\hat{\mathbf{y}})$ resulting in assignments given by $\hat{\mathbf{y}} = \mathrm{argmax}_{\mathbf{y}' \in \mathcal{Y}^{n_y}} f(\mathbf{x},\mathbf{y}')$. An inference mechanism takes the scoring function $f(\mathbf{x},\mathbf{y})$, an instance $(\mathbf{x},\mathbf{y})$, and a set of constraints $\mathcal{C}$, returning an optimal assignment $\hat{\mathbf{y}}_{\mathcal{C}}$ based on the global score $f(\mathbf{x},\hat{\mathbf{y}}_{\mathcal{C}})$ consistent with the defined output structure. Specifically, we will use general constraints with the ability to represent any structure and thereby require a general search mechanism for inference to enforce structural consistency [6]. As active learning querying functions are designed to select instances with specific properties, we define the notions of *locally learnable instances* and *globally learnable instances* for exposition purposes.

Definition 1. (Locally Learnable Instance) *Given a classifier,* $f \in \mathcal{H}$*, an instance* $(\mathbf{x},\mathbf{y})$ *is locally learnable if* $f_{y_t}(\mathbf{x},t) > f_{y'}(\mathbf{x},t)$ *for all* $y' \in \mathcal{Y} \backslash y_t$*. In this situation,* $\hat{\mathbf{y}} = \hat{\mathbf{y}}_{\mathcal{C}} = \mathbf{y}$*.*

Definition 2. (Globally Learnable Instance) *Given a classifier,* $f \in \mathcal{H}$*, an instance* $(\mathbf{x},\mathbf{y})$ *is globally learnable if* $f(\mathbf{x},\mathbf{y}) > f(\mathbf{x},\mathbf{y}')$ *for all* $\mathbf{y}' \in \mathcal{Y} \backslash \mathbf{y}$*. We will refer to instances that are globally learnable, but not locally learnable as* **exclusively globally learnable** *in which case* $\hat{\mathbf{y}} \neq \hat{\mathbf{y}}_{\mathcal{C}} = \mathbf{y}$*.*

2.2 Margin-Based Active Learning

The key component that distinguishes active learning from standard supervised learning is a querying function $\mathcal{Q}$ which when given unlabeled data $\mathcal{S}_u$ and the current learned classifier returns a set of unlabeled examples $\mathcal{S}_{select} \subseteq \mathcal{S}_u$. These selected examples are labeled and provided to the learning algorithm to incrementally update its hypothesis. The most widely used active learning schemes utilize querying functions based on heuristics, often assigning a measure of certainty to predictions on $\mathcal{S}_u$ and selecting examples with low certainty.

We denote the margin of an example relative to the hypothesis function as $\rho(\mathbf{x},\mathbf{y},f)$, noting that this value is positive if and only if $\hat{\mathbf{y}}_{\mathcal{C}} = \mathbf{y}$ and the magnitude is associated with the confidence in the prediction. The specific definition of margin for a given setting is generally dependent on the description of the output space. A *margin-based learning algorithm* is a learning algorithm which selects a hypothesis by minimizing a loss function $\mathcal{L} : \mathbb{R} \to [0,\infty)$ using the margin of instances contained in $\mathcal{S}_l$. We correspondingly define an active learning algorithm with a querying function dependent on $\rho(\mathbf{x},\mathbf{y},f)$ as a *margin-based active learning algorithm.*

The standard active learning algorithm for binary classification, $Y \in \{-1,1\}$, with linear functions utilizes the querying function $\mathcal{Q}_{binary}$ [4], which makes direct use of the margin $\rho_{binary}(\mathbf{x},y,f) = y \cdot f(\mathbf{x})$ by assuming the current classifier generally makes correct predictions on the training data and selecting those unlabeled examples with the smallest margin and thereby minimal certainty,

$$\mathcal{Q}_{binary} : x_\star = \underset{x \in \mathcal{S}_u}{\mathrm{argmin}}\, |f(\mathbf{x})|.$$

For multiclass classification, a widely accepted definition for multiclass margin is $\rho_{multiclass}(\mathbf{x}, \mathbf{y}, f) = f_y(\mathbf{x}) - f_{\dot{y}}(\mathbf{x})$ where y represents the true label and $\dot{y} = \text{argmax}_{y' \in \mathcal{Y} \backslash y} f_{y'}(\mathbf{x})$ corresponds to the highest activation value such that $\dot{y} \neq y$ [7]. Previous work on multiclass active learning [5] advocates a querying function closely related to this definition of multiclass margin where $\hat{y} = \text{argmax}_{y' \in \mathcal{Y}} f_{y'}(\mathbf{x})$ represents the predicted label and $\tilde{y} = \text{argmax}_{y' \in \mathcal{Y} \backslash \hat{y}} f_{y'}(\mathbf{x})$ represents the label corresponding to the second highest activation value,

$$\mathcal{Q}_{multiclass} : x_\star = \underset{x \in \mathcal{S}_u}{\text{argmin}}[f_{\hat{y}}(\mathbf{x}) - f_{\tilde{y}}(\mathbf{x})].$$

3 Active Learning for Structured Output

We look to augment the aforementioned work to design querying functions for learning in structured output spaces by exploiting structural knowledge not available for individual classifications. Without loss of generality, we assume that y_t represents a multiclass classification.

3.1 Querying Complete Labels

The task of a querying function for complete labels entails selecting instances $\mathbf{x}$ such that all output labels associated with the specified instance will be provided by the domain expert. Following the margin-based approach for designing querying functions, a reasonable definition of margin for structured output spaces is $\rho_{global}(\mathbf{x}, \mathbf{y}, f) = f(\mathbf{x}, \mathbf{y}) - f(\mathbf{x}, \dot{\mathbf{y}}_{\mathcal{C}})$ where $\dot{\mathbf{y}}_{\mathcal{C}} = \text{argmax}_{\mathbf{y}' \in \mathcal{C}(\mathcal{Y}^{n_y}) \backslash \mathbf{y}} f(\mathbf{x}, \mathbf{y}')$. The corresponding querying function for a structured learner that incorporates the constraints into the learning model is defined by

$$\mathcal{Q}_{global} : x_\star = \underset{x \in \mathcal{S}_u}{\text{argmin}}[f(\mathbf{x}, \hat{\mathbf{y}}_{\mathcal{C}}) - f(\mathbf{x}, \tilde{\mathbf{y}}_{\mathcal{C}})],$$

where $\tilde{\mathbf{y}}_{\mathcal{C}} = \text{argmax}_{\mathbf{y}' \in \mathcal{C}(\mathcal{Y}^{n_y}) \backslash \hat{\mathbf{y}}_{\mathcal{C}}} f(\mathbf{x}, \mathbf{y}')$. It should be noted that $\mathcal{Q}_{global}$ does not require $f(\mathbf{x}, \mathbf{y})$ to be decomposable, thereby allowing usage with arbitrary loss functions. The only requirement is that the inference mechanism is capable of calculating $f(\mathbf{x}, \hat{\mathbf{y}}_{\mathcal{C}})$ and $f(\mathbf{x}, \tilde{\mathbf{y}}_{\mathcal{C}})$ for a given structured instance.

However, for many structured learning settings the scoring function and consequently the loss function is decomposable into local classification problems. Furthermore, it has been observed that when the local classification problems are easy to learn without regard for structural constraints, directly optimizing these local functions often leads to a lower sample complexity [3]. As these findings are predicated on making concurrent local updates during learning, selecting structured examples that make as many local updates as possible may be desirable for such situations. This observation motivates a querying function that selects instances based on local predictions, resulting in the margin-based strategy of selecting examples with a small average local multiclass margin,

$$\mathcal{Q}_{\overline{local(\mathcal{C})}} : x_\star = \underset{x \in \mathcal{S}_u}{\text{argmin}} \frac{\sum_{t=1}^{n_y}[f_{\hat{y}_{\mathcal{C},t}}(\mathbf{x}, t) - f_{\tilde{y}_{\mathcal{C},t}}(\mathbf{x}, t)]}{n_y},$$

where $\hat{y}_{\mathcal{C},t} = \text{argmax}_{y'_t \in \mathcal{C}(\mathcal{Y})} f_{y'_t}(\mathbf{x}, t)$ and $\tilde{y}_{\mathcal{C},t} = \text{argmax}_{y'_t \in \mathcal{C}(\mathcal{Y}) \backslash \hat{y}_t} f_{y'_t}(\mathbf{x}, t)$.

3.2 Querying Partial Labels

We noted that $\mathcal{Q}_{global}$ makes no assumptions regarding decomposability of of the scoring function and $\mathcal{Q}_{\overline{local(\mathcal{C})}}$ requires only that the scoring function be decomposable in accordance with the output variables. We now examine active learning in settings where $f(\mathbf{x}, \mathbf{y})$ is decomposable and the local output variables can be queried independently, defined as querying partial labels. The intuitive advantage of querying partial labels is that we are no longer subject to cases where a structured instance has one output variable with a very informative label, but the other output variables of the same instance are minimally useful and yet add cost to the labeling effort. While this configuration is not immediately usable for applications with a scoring function not easily decomposable into local output variables that can be independently queried, we will see this approach is very beneficial in scenarios where such restrictions are possible.

Observing that querying partial labels requires requesting a single multiclass classification, the naive querying function for this case is to simply ignore the structural information and use $\mathcal{Q}_{multiclass}$, resulting in the querying function

$$\mathcal{Q}_{local} : (\mathbf{x}, t)_\star = \operatorname*{argmin}_{\substack{(\mathbf{x}, y_t) \in \mathcal{S}_u \\ t=1,\ldots,n_y}} [f_{\hat{y}_t}(\mathbf{x}, t) - f_{\tilde{y}_t}(\mathbf{x}, t)].$$

One of the stronger arguments for margin-based active learning is the notion of selecting instances which attempt to halve the version space with each selection [4]. A local classifier which either ignores or is ignorant of the structural constraints maintains a version space described by

$$\mathcal{V}_{local} = \{f \in \mathcal{H} | f_{y_t}(\mathbf{x}, t) > f_{\dot{y}_t}(\mathbf{x}, t); \forall (\mathbf{x}, y) \in \mathcal{S}_l\}.$$

If the learning algorithm has access to an inference mechanism that maintains structural consistency, the version space is only dependent on the subset of possible output variable assignments that are consistent with the global structure,

$$\mathcal{V}_{local(\mathcal{C})} = \{f \in \mathcal{H} | f_{y_t}(\mathbf{x}, t) > f_{\dot{y}_{\mathcal{C},t}}(\mathbf{x}, t); \forall (\mathbf{x}, y) \in \mathcal{S}_l\}$$

where $\dot{y}_{\mathcal{C},t} = \operatorname{argmax}_{y'_t \in \mathcal{C}(\mathcal{Y}) \setminus y_t} f_{y'_t}(\mathbf{x}, t)$. Therefore, if the learning algorithm enforces structural consistency within the learning model, we advocate also utilizing this information to augment $\mathcal{Q}_{local}$, resulting in the querying function

$$\mathcal{Q}_{local(\mathcal{C})} : (\mathbf{x}, t)_\star = \operatorname*{argmin}_{\substack{(\mathbf{x}, y_t) \in \mathcal{S}_u \\ t=1,\ldots,n_y}} [f_{\hat{y}_{\mathcal{C},t}}(\mathbf{x}, t) - f_{\tilde{y}_{\mathcal{C},t}}(\mathbf{x}, t)].$$

4 Active Learning with Perceptron

This work specifically utilizes classifiers of a linear representation with parameters learned using the Perceptron algorithm. In this case, $f(\mathbf{x}, \mathbf{y}) = \boldsymbol{\alpha} \cdot \Phi(\mathbf{x}, \mathbf{y})$

represents the global scoring function such that $\boldsymbol{\alpha} = (\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^{|\mathcal{Y}|})$ is a concatenation of the local $\boldsymbol{\alpha}^y$ vectors and $\Phi(\mathbf{x}, \mathbf{y}) = (\Phi^1(\mathbf{x}, \mathbf{y}), \ldots, \Phi^{|\mathcal{Y}|}(\mathbf{x}, \mathbf{y}))$ is a concatenation of the local feature vectors, $\Phi^y(\mathbf{x}, \mathbf{y})$. Utilizing this notation, $f_y(\mathbf{x}, t) = \boldsymbol{\alpha}^y \cdot \Phi^y(\mathbf{x}, t)$ where $\boldsymbol{\alpha}^y \in \mathbb{R}^{d_y}$ is the learned weight vector and $\Phi^y(\mathbf{x}, t) \in \mathbb{R}^{d_y}$ is the feature vector for local classifications.

Margin-based active learning generally relies upon the use of support vector machines (SVM) [4,5]. While there is existing work on SVM for structured output [8], the incremental nature of active learning over large data sets associated with structured output makes these algorithms impractical for such uses. This work builds upon the *inference based training* (IBT) learning strategy [3,2] shown in Table 1, which incorporates the structural knowledge into the learning procedure. We first modify the IBT algorithm for partial labels by updating only local components which have been labeled. Secondly, we add a notion of large margin IBT heuristically by requiring thick separation between class activations. While this can likely be tuned to improve performance depending on the data, we simply set $\gamma = 1.0$ and require that $\|\Phi^{y_t}(\mathbf{x}, t)\| = 1$ through normalization for our experiments. During learning, we set $T = 7$ for synthetic data and $T = 5$ for experiments with the SRL task. To infer $\hat{\mathbf{y}}_{\mathcal{C}}$, we use an index ordered beam search with beam size of 50 for synthetic data and 100 for SRL. Beam search was used since it performs well, is computationally fast, accommodates general constraints, and returns a global score ranking which is required for $\mathcal{Q}_{global}$.

Table 1. Learning wth Inference Based Feedback (IBT)

```
INPUT: $\mathcal{S} \in \{\mathcal{X}^* \times \mathcal{Y}^*\}^m, \gamma, T$
------------------------------------------------------------
Initialize $\boldsymbol{\alpha} \leftarrow 0$
Repeat for $T$ iterations
  foreach $(\mathbf{x}, \mathbf{y}) \in \mathcal{S}$
      $\hat{\mathbf{y}}_{\mathcal{C}} \leftarrow \mathrm{argmax}_{\mathbf{y} \in \mathcal{C}(\mathcal{Y}^{n_y})} \boldsymbol{\alpha} \cdot \Phi(\mathbf{x}, \mathbf{y})$
      foreach $t = 1, \ldots, n_y$ such that $(\mathbf{x}, y_t) \in \mathcal{S}_l$
          if $f_{y_t}(\mathbf{x}, t) - \gamma < f_{\hat{y}_{\mathcal{C},t}}(\mathbf{x}, t)$
              $\boldsymbol{\alpha}^{y_t} \leftarrow \boldsymbol{\alpha}^{y_t} + \Phi^{y_t}(\mathbf{x}, t)$
              $\boldsymbol{\alpha}^{\hat{y}_t} \leftarrow \alpha^{\hat{y}_t} - \Phi^{\hat{y}_t}(\mathbf{x}, t)$
------------------------------------------------------------
OUTPUT: $\{f_y\}_{y \in \mathcal{Y}} \in \mathcal{H}$
```

4.1 Mistake-Driven Active Learning

A greedy criteria for active learning querying functions makes the most immediate progress towards learning the target function with each requested label. For the mistake-driven Perceptron algorithm, a suitable measurement for progress is to track the number of additive updates for each query. This intuition proposes two metrics to explain the performance results of a given querying function, *average Hamming error per query*, $\mathcal{M}_{Hamming}$, and *average global error per query*, $\mathcal{M}_{global}$. For a specific round of active learning, the current hypothesis is used to select a set of instances $\mathcal{S}_{select}$ for labeling. Once the labels are received, we calculate the Hamming loss $\mathcal{H}(h, \mathbf{x}) = \sum_{t=1;(\mathbf{x},y_t)\in\mathcal{S}_l}^{n_y} I[\![\hat{y}_{\mathcal{C},t} \neq y]\!]$ and the global loss $\mathcal{G}(h, \mathbf{x}) = I[\![\hat{\mathbf{y}}_{\mathcal{C}} \neq \mathbf{y}]\!]$ at the time when the instance is first labeled. $I[\![p]\!]$ is an indicator function such that $I[\![p]\!] = 1$ if p is true and 0 otherwise. We measure the quality of a querying function relative to the average of these values for all queries up to the specific round of active learning.

Noting that only $\mathcal{H}(h, \mathbf{x})$ is useful for partial labels, we hypothesize that for partial label queries or cases of complete label queries where the data sample $\mathcal{S}$ is largely locally separable, the relative magnitude of $\mathcal{M}_{Hamming}$ will determine the relative performance of the querying functions. Alternatively, for complete queries where a significant portion of the data is exclusively globally separable, $\mathcal{M}_{global}$ will be more strongly correlated with querying function performance.

5 Experiments

We demonstrate particular properties of the proposed querying functions by first running active learning simulations on synthetic data. We then verify practicality for actual applications by performing experiments on the SRL task.

5.1 Synthetic Data

Our synthetic structured output problem is comprised of five multiclass classifiers, $h_1, \ldots, h_5$, each having the output space $Y_t = \omega_1, \ldots, \omega_4$. In addition, we define the output structure using the following practical constraints:

1. $\mathcal{C}_1 : [h_2(\mathbf{x}) \neq \omega_3] \wedge [h_5(\mathbf{x}) \neq \omega_1]$
2. $\mathcal{C}_2$: At most one $h_t(\mathbf{x})$ can output ω_2.
3. $\mathcal{C}_3$: For one or more $h_t(\mathbf{x})$ to output ω_3, at least one $h_t(\mathbf{x})$ must output ω_1.
4. $\mathcal{C}_4$: $h_t(\mathbf{x})$ can output ω_4 if and only if $h_{t-1}(\mathbf{x}) = \omega_1$ and $h_{t-2}(\mathbf{x}) = \omega_2$.

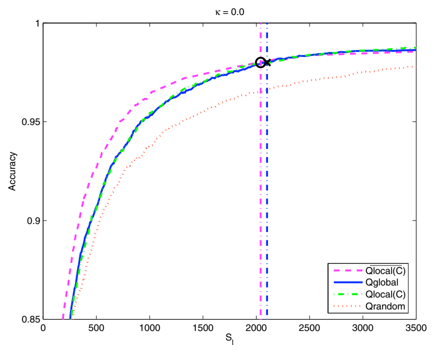
To generate the synthetic data, we first create four linear functions of the form $\mathbf{w}_i \cdot \mathbf{x} + b_i$ such that $\mathbf{w}_i \in [-1, 1]^{100}$ and $b_i \in [-1, 1]$ for each h_t. We then generate five local examples $\mathbf{x}_t \in \{0, 1\}^{100}$ where the normal distribution $\mathcal{N}(20, 5)$ determines the number of features assigned the value 1, distributed uniformly over the feature vector. Each vector is labeled according to the function $\text{argmax}_{i=1,\ldots,k}[\mathbf{w}_i \cdot \mathbf{x} + b_i]$ resulting in the label vector $\mathbf{y}_{local} = (h_1(\mathbf{x}), \ldots, h_5(\mathbf{x}))$. We then run the inference procedure to obtain the final labeling $\mathbf{y}$ of the instance $\mathbf{x}$. If $\mathbf{y} \neq \mathbf{y}_{local}$, then the data is exclusively globally separable. We control the total amount of such data with the parameter κ which represents the fraction of exclusively globally separable data in $\mathcal{S}$. We further filter the difficulty of the data such that all exclusively globally separable instances have a Hamming error drawn from a stated normal distribution $\mathcal{N}(\mu, \sigma)$. We generate 10000 structured examples, or equivalently 50000 local instances, in this fashion for each set of data parameters we use.

Figure 1 shows the experimental results for the described complete querying functions in addition to $\mathcal{Q}_{random}$, which selects arbitrary unlabeled instances at each step, and $\mathcal{Q}_{local(\mathcal{C})}$ where an entire structured instance is based upon the score of a single local classifier to demonstrate that it is prudent to design querying functions specifically for complete labels. The querying schedule starts as $|\mathcal{S}_l| = 2, 4, \ldots, 200$ and slowly increases the step size until $|\mathcal{S}_l| = 6000, 6100, \ldots, 8000$ and 5-fold cross validation is performed. The primary observation for the synthetic data set where $\kappa = 0.0$ is that $\mathcal{Q}_{\overline{local(\mathcal{C})}}$ performs better than $\mathcal{Q}_{global}$ when the data is locally separable. For the data set where $\kappa = 0.3; \mathcal{N}(3, 1)$, we see that as the data becomes less locally separable, $\mathcal{Q}_{global}$ performs better than $\mathcal{Q}_{\overline{local(\mathcal{C})}}$. We also plot $\mathcal{M}_{Hamming}$ and $\mathcal{M}_{global}$ for each respective querying functions. As expected, when the data is locally separable, the querying function performance is closely related to $\mathcal{M}_{Hamming}$ and when the data is less locally separable, the relative querying function performance is more closely related to $\mathcal{M}_{global}$. The vertical lines denote when the specified querying function achieves an accuracy equivalent to the largest accuracy achieved by using $\mathcal{Q}_{random}$. Remembering that there are 8000 training examples, we measure between $25\% - 75\%$ reduction in required training data.

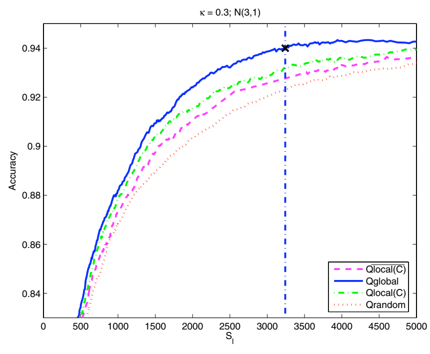
Figure 2 shows our experimental results for partial querying functions on the synthetic data. We completed experiments with the two partial querying functions $\mathcal{Q}_{local}$ and $\mathcal{Q}_{local(\mathcal{C})}$ in addition to $\mathcal{Q}_{random}$ on three sets of data. The querying schedule starts by querying 10 partial labels at a time from $|\mathcal{S}_l| = 10, 20, \ldots, 2000$ and increases until the step size is $|\mathcal{S}_l| = 20000, 21000, \ldots, 40000$ and once again 5-fold cross validation is performed. The first synthetic data set is where $\kappa = 0.0$ and the data is completely locally separable. In this case, active learning for both $\mathcal{Q}_{local}$ and $\mathcal{Q}_{local(\mathcal{C})}$ perform better than $\mathcal{Q}_{random}$. Somewhat more surprising is the result that $\mathcal{Q}_{local(\mathcal{C})}$ performs noticeably better that $\mathcal{Q}_{local}$ even though they should query similar points for $\kappa = 0.0$. The results for the synthetic data set $\kappa = 0.3; \mathcal{N}(3, 1)$ also demonstrate a similar ordering where $\mathcal{Q}_{local(\mathcal{C})}$ outperforms $\mathcal{Q}_{local}$ which in turn outperforms $\mathcal{Q}_{random}$. Finally, we used a synthetic data set where $\kappa = 1.0; \mathcal{N}(5, 1)$, meaning that the data is completely exclusively globally separable and the difference between $\mathcal{Q}_{local(\mathcal{C})}$ and $\mathcal{Q}_{local}$ is most noticable. For this data set, we also plotted $\mathcal{M}_{Hamming}$ noting that this value is always greater for $\mathcal{Q}_{local(\mathcal{C})}$ than $\mathcal{Q}_{local}$, which is consistent with our expectations for $\mathcal{M}_{Hamming}$ relative to querying function performance. As there are 40000 training examples for each fold, we show a decrease in necessary data of between $65\% - 79\%$ depending on the specific experiment.

5.2 Semantic Role Labeling

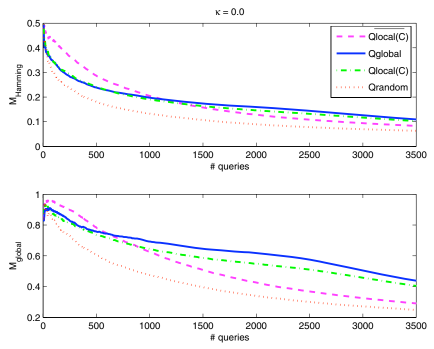
Finally, we also perform experiments on the SRL task as described in the CoNLL-2004 shared task [1]. We essentially follow the model described in [3] where linear classifiers $f_{A0}, f_{A1}, \ldots$ are used to map constituent candidates to one of 45 different classes. For a given argument / predicate pair, the multiclass classifier returns a set of scores which are used to produce the output $\hat{\mathbf{y}}_{\mathcal{C}}$ consistent with the structural constraints associated with other arguments relative to the same predicate. We simplify the task by assuming that the constituent boundaries are given, making this an argument classification task. We use the CoNLL-2004



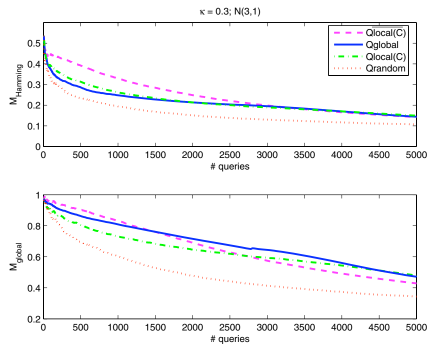
(a)



(b)



(c)



(d)

Fig. 1. Experimental results for the complete label querying problem, noting that the labeling effort is reduced between 25% − 75% depending on the particular situation. (a) Active learning curve for $\kappa = 0.0$ (b) Active learning curve for $\kappa = 0.3; \mathcal{N}(3,1)$ (c) Plot of $\mathcal{M}_{hamming}$ and $\mathcal{M}_{global}$ for $\kappa = 0.0$ (d) Plot of $\mathcal{M}_{hamming}$ and $\mathcal{M}_{global}$ for $\kappa = 0.3; \mathcal{N}(3,1)$.

shared task data, but restrict our experiments to sentences that have greater than five arguments to increase the number of instances with interdependent variables and take a random subset of this to get 1500 structured examples comprised of 9327 local predictions. For our testing data, we also restrict ourself to sentences with greater than five arguments, resulting in 301 structured instances comprised of 1862 local predictions. We use the same features and the applicable subset of families of constraints which do not concern segmentation as described by [9]. Figure 3 shows the emperical results for the SRL experiments. For querying complete labels, we start with a querying schedule of $|\mathcal{S}_l| = 50, 80, \ldots, 150$ and slowly increase the step size until ending with $|\mathcal{S}_l| = 1000, 1100, \ldots, 1500$. For the complete labeling case, $\mathcal{Q}_{\overline{local(\mathcal{C})}}$ performs better than $\mathcal{Q}_{global}$, implying that the data is largely locally separable which is consistent with the findings of [3]. Furthermore, both functions perform better than $\mathcal{Q}_{random}$ with approximately a 35% reduction in labeling effort. For partial labels, we used a querying schedule

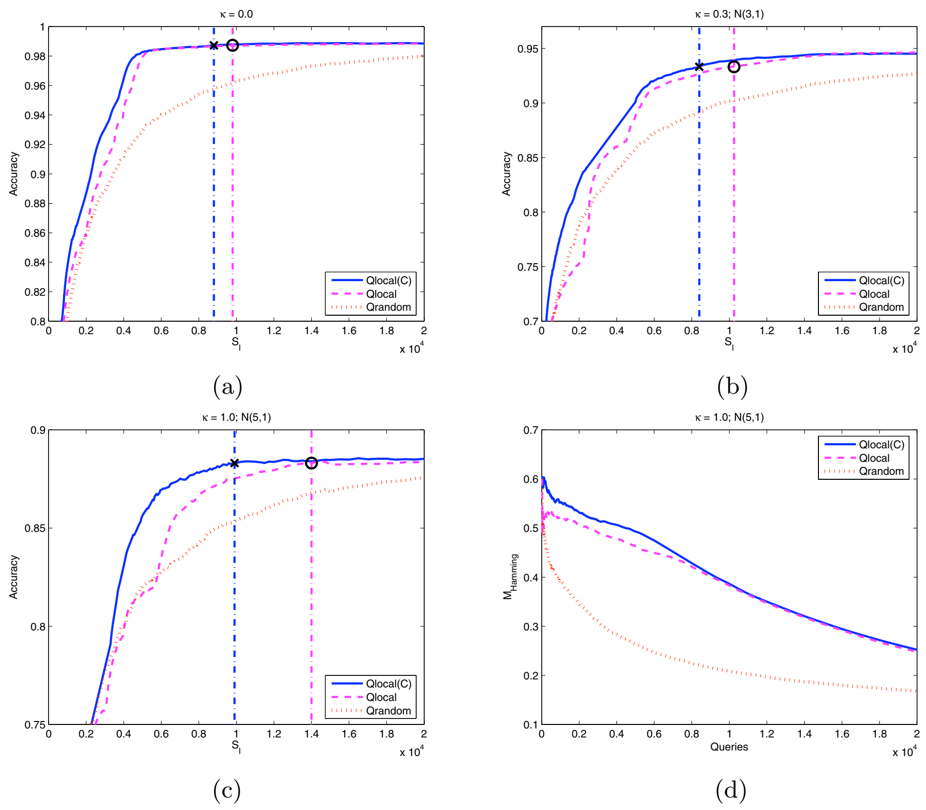


Fig. 2. Experimental results for the partial label querying problem, noting that the labeling effort is reduced between 65% − 79% depending of the particular situation. (a) Active learning curve for $\kappa = 0.0$ (b) Active learning curve for $\kappa = 0.3; \mathcal{N}(3, 1)$ (c) Active learning curve for $\kappa = 1.0; \mathcal{N}(5, 1)$ (d) Plot of $\mathcal{M}_{hamming}$ for $\kappa = 1.0; \mathcal{N}(5, 1)$.

that starts at $|\mathcal{S}_l| = 100, 200, \ldots, 500$ and increases step size until ending at $|\mathcal{S}_l| = 6000, 7000, \ldots, 9327$. In this case, $\mathcal{Q}_{local(\mathcal{C})}$ performs better than $\mathcal{Q}_{local}$ and $\mathcal{Q}_{random}$, requiring only about half of the data to be labeled.

6 Related Work

Some of the earliest works on active learning in a structured setting is the work in language parsing including [10,11,12], which utilize specific properties of the parsing algorithms to assign uncertainty values to unlabeled instances. There has also been work on active learning for hidden markov models (HMM) [13,14], which is a learning algorithm for structured output with a specific set of sequential constraints. More directly related is the active learning work using conditional random fields (CRFs) [15], which can theoretically incorporate general

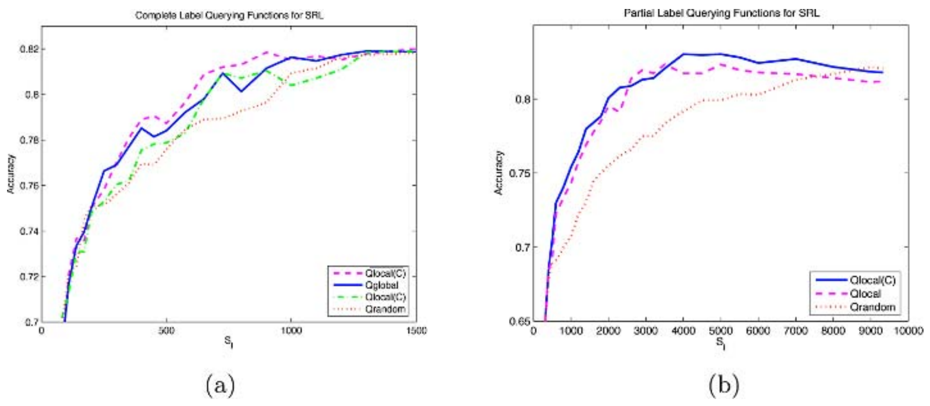


Fig. 3. Experimental results for SRL. (a) Active learning curve for the complete label querying scenario (b) Active learning curve for the partial label querying scenario.

constraints, basing selection on a probabilistic uncertainty metric. In this case, the complete labels are selected and the emphasis is on reducing the actual cost of labeling through a more sophisticated interaction with the expert.

7 Conclusions and Future Work

This work describes a margin-based active learning approach for structured output spaces. We first look at the setting of querying complete labels, defining $\mathcal{Q}_{global}$ to be used in situations where the scoring function $f(\mathbf{x}, \mathbf{y})$ is not decomposable or the data is expected to be exclusively globally learnable and define $\mathcal{Q}_{\overline{local(\mathcal{C})}}$ to be used when the scoring function is decomposable and the data is expected to be locally learnable. We further demonstrate that in cases where the local classifications can be queried independently, the labeling effort is most drastically reduced using partial label queries with the querying function $\mathcal{Q}_{local(\mathcal{C})}$. These propositions are also supported empirically on both synthetic data and the semantic role labeling (SRL) task. There appears to be many dimensions for future work including examining scenarios where subsets of the output variables are queried, providing a continuum between single and complete labels. Furthermore, developing a more realistic model of labeling cost along this continuum and looking at the performance of other margin-based learning algorithms within this framework would likely enable this work to be applied to a wider range of structured output applications.

Acknowledgments

The authors would like to thank Ming-Wei Chang, Vasin Punyakanok, Alex Klementiev, Nick Rizzolo, and the reviewers for helpful comments and/or discussions regarding this research. This work has been partially funded by a grant from Motorola Labs and NSF grant ITR-IIS-0428472.

References

1. Carreras, X., Màrquez, L.: Introduction to the CoNLL-2004 shared tasks: Semantic role labeling. In: Proc. of the Conference on Computational Natural Language Learning (CoNLL). (2004)
2. Collins, M.: Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In: Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP). (2002)
3. Punyakanok, V., Roth, D., Yih, W., Zimak, D.: Learning and inference over constrained output. In: Proc. of the International Joint Conference on Artificial Intelligence (IJCAI). (2005) 1124–1129
4. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. Journal of Machine Learning Research **2** (2001) 45–66
5. Yan, R., Yang, J., Hauptmann, A.: Automatically labeling video data using multiclass active learning. In: Proc. of the International Conference on Computer Vision (ICCV). (2003) 516–523
6. Daumé III, H., Marcu, D.: Learning as search optimization: Approximate large margin methods for structured prediction. In: Proc. of the International Conference on Machine Learning (ICML). (2005)
7. Har-Peled, S., Roth, D., Zimak, D.: Constraint classification for multiclass classification and ranking. In: The Conference on Advances in Neural Information Processing Systems (NIPS). (2003) 785–792
8. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: Proc. of the International Conference on Machine Learning (ICML). (2004) 823–830
9. Punyakanok, V., Roth, D., Yih, W., Zimak, D.: Semantic role labeling via integer linear programming inference. In: Proc. the International Conference on Computational Linguistics (COLING). (2004)
10. Thompson, C.A., Califf, M.E., Mooney, R.J.: Active learning for natural language parsing and information extraction. In: Proc. of the International Conference on Machine Learning (ICML). (1999) 406–414
11. Hwa, R.: Sample selection for statistical grammar induction. In: Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP). (2000)
12. Baldridge, J., Osbourne, M.: Active learning for HPSG parse selection. In: Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL). (2003) 17–24
13. Scheffer, T., Wrobel, S.: Active learning of partially hidden markov models. Lecture Notes in Computer Science **2189** (2001)
14. Anderson, B., Moore, A.: Active learning for hidden markov models: Objective functions and algorithms. In: Proc. of the International Conference on Machine Learning (ICML). (2005)
15. Culotta, A., McCallum, A.: Reducing labeling effort for stuctured prediction tasks. In: Proceedings of the National Conference on Artificial Intelligence (AAAI). (2005)

Skill Acquisition Via Transfer Learning and Advice Taking*

Lisa Torrey[1], Jude Shavlik[1], Trevor Walker[1], and Richard Maclin[2]

[1] University of Wisconsin, Madison WI 53706, USA
[2] University of Minnesota, Duluth, MN 55812, USA

Abstract. We describe a reinforcement learning system that transfers skills from a previously learned source task to a related target task. The system uses inductive logic programming to analyze experience in the source task, and transfers rules for when to take actions. The target task learner accepts these rules through an advice-taking algorithm, which allows learners to benefit from outside guidance that may be imperfect. Our system accepts a human-provided mapping, which specifies the similarities between the source and target tasks and may also include advice about the differences between them. Using three tasks in the RoboCup simulated soccer domain, we demonstrate that this system can speed up reinforcement learning substantially.

1 Introduction

Machine learning tasks are often addressed independently, under the implicit assumption that each new task has no relation to the tasks that came before. In some domains, particularly reinforcement learning (RL) ones, this assumption is often incorrect since tasks in the same domain tend to be related. Even tasks that are quite different in their specifics may have general similarities, such as shared *skills*; that is, conditions under which an agent should take an action. Our goal is to *transfer* general skills from a source task in order to speed up learning in a new but similar target task.

For example, suppose an RL soccer player has learned, in a source task, to keep the ball from its opponents by passing to its teammates. In the target task, suppose it must learn to work with teammates to score goals against opponents. If this player could apply its passing skills from the source task, it might master the target task more quickly.

Even when RL tasks have shared skills, transfer between them is a difficult problem because differences in action sets and reward structures create differences in shared skills. For example, the passing skill in the source task above is incomplete for the target task, where passing needs to cause progress toward the goal. This indicates that RL agents using transferred information must continue to learn, filling in gaps left by transfer. Since transfer might also produce

* This research is partially supported by DARPA grant HR0011-04-1-0007 and US Naval Research Laboratory grant N00173-06-1-G002.

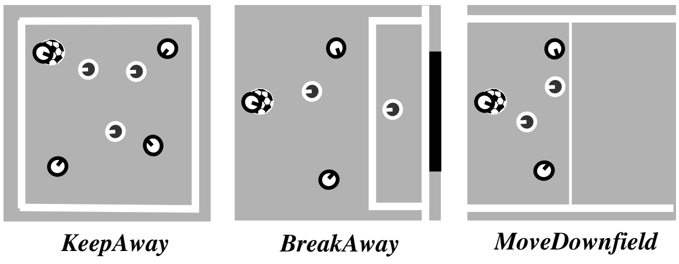


Fig. 1. Snapshots of RoboCup soccer tasks

partially irrelevant or incorrect skills, RL agents must also be able to modify or ignore transferred information that is imperfect.

One way to facilitate transfer is for a human observer with basic domain knowledge to provide a *mapping* between source and target tasks. A mapping describes the structural similarities between the tasks, such as correspondences between player objects in the example above. It might also include simple advice that reflects the differences between the tasks. In our example, tips like "prefer passing toward the goal" and "shoot when close to the goal" would be helpful.

We present a system for transfer learning in RL called AI^2 (Advice via Induction and Instruction). It constructs relational *transfer advice* by using inductive logic programming to analyze experience in the source task and learn skills in first-order logic. The user contributes a mapping between the tasks that may include *user advice*. The target-task learner considers the advice while learning and can follow it, refine it, or ignore it according to its value.

The AI^2 approach performs transfer at a higher level of abstraction than some previous approaches [14,15]. For this reason, it performs well in transfer scenarios involving more distant tasks. We present empirical results in the challenging RoboCup simulated soccer domain, demonstrating significantly faster learning in the target task *BreakAway* [15] after performing user-guided transfer from the source tasks *KeepAway* [9] and *MoveDownfield* (see Figure 1).

2 Reinforcement Learning in RoboCup

In reinforcement learning [13], an agent navigates through an environment trying to earn rewards or avoid penalties. The environment's state is described by a finite number of features, and the agent takes actions to cause the state to change. In Q-learning, the agent learns a Q-function to estimate the value of taking an action from a state. An agent's policy is typically to take the action with the highest Q-value in the current state, except for occasional exploratory actions. After taking the action and receiving some reward, the agent updates its Q-value estimates for the current state. AI^2 uses the $SARSA$ and TD(λ) reinforcement learning algorithms designed by Sutton [11,12].

In the RoboCup learning task of M-on-N KeepAway [9], the objective of the M reinforcement learners called *keepers* is to keep the ball away from N

hand-coded players called *takers*. The game ends when an opponent takes the ball or when the ball goes out of bounds. The learners receive a +1 reward for each time step their team keeps the ball. Keepers without the ball follow a hand-coded strategy to receive passes.

In the original KeepAway task, the keeper who has the ball can choose only to hold it or pass to a teammate. We introduce a new version called *Mobile KeepAway*, in which this keeper can also move (inwards, outwards, clockwise and counterclockwise with respect to the field center). With more realistic movement, this version may transfer better to other games.

Our KeepAway state representation is based on the one designed by Stone and Sutton [9]. The keepers are ordered by their distance to the learner *k0*, as are the takers. The features are listed in Table 1.

Note that our logical variables are capitalized and typed (*Player*, *Keeper*, etc.). For simplicity we indicate types by variable names, leaving out terms like *player(Player)*, *keeper(Keeper)*, etc. Constants are uncapitalized.

A second RoboCup task is M-on-N BreakAway [15], where the objective of the M reinforcement learners called *attackers* is to score a goal against $N - 1$ hand-coded *defenders* and a hand-coded *goalie*. The game ends when they succeed, when an opponent takes the ball, when the ball goes out of bounds, or after a time limit of 10 seconds. The learners receive a +1 reward if they score a goal, and zero reward otherwise. Attackers without the ball follow a hand-coded strategy to receive passes. The attacker who has the ball may choose to move (ahead, away, left, or right with respect to the goal), pass to a teammate, or shoot (at the left, right, or center part of the goal).

Our BreakAway state representation is the one presented in Torrey et al. [15]. The attackers are ordered by their distance to the learner *a0*, as are the defenders. The features are listed in Table 1.

We also introduce a third RoboCup task called M-on-N MoveDownfield, where the objective of the attackers is to move toward the opposing team's goal while maintaining possession of the ball. The game ends when they cross a vertical line on the field, when an opponent takes the ball, when the ball goes out

Table 1. RoboCup task feature spaces

KeepAway features	*BreakAway and MoveDownfield features*
distBetween(k0, Player)	distBetween(a0, Player)
distBetween(Keeper, ClosestTaker)	distBetween(Attacker, ClosestDefender)
angleDefinedBy(Keeper, k0, ClosestTaker)	angleDefinedBy(Attacker, a0, ClosestDefender)
xPosition(Object)	xPosition(Object)
yPosition(Object)	yPosition(Object)
distBetween(Keeper, fieldCenter)	distBetween(Attacker, goalCenter)
	distBetween(a0, GoalPart)
	angleDefinedBy(GoalPart, a0, goalie)
	angleDefinedBy(topRight, goalCenter, a0)
	distBetween(Attacker, goalie)
	angleDefinedBy(Attacker, a0, goalie)
	timeLeft

of bounds, or after a time limit of 25 seconds. The learners receive symmetrical positive and negative rewards for horizontal movement forward and backward. Attackers without the ball follow a hand-coded strategy to receive passes. The action set and feature set are the same as in BreakAway, except without the *shoot* actions and most of the features involving the goal.

Our system discretizes each feature in these tasks into 32 intervals called *tiles*, each of which is associated with a Boolean feature. For example, the tile denoted by *distBetween(a0, a1)*$_{[10,20]}$ takes value 1 when *a1* is between 10 and 20 units away from *a0* and 0 otherwise. This enhancement of the state space is used in RoboCup by Stone and Sutton [9], and we adopt it to give our linear Q-function model the ability to represent more complex functions.

These three RoboCup games have substantial differences in features, actions, and rewards (and therefore Q-values), but they all require the skill of passing the ball among teammates without losing it to the opponents.

3 AI^2: Transferring Skills

Because RL agents learn to take actions, a natural human interpretation of RL is that the agents acquire *skills*. However, what they typically acquire is a Q-function, which is highly task-specific and does not readily translate into discrete skills. Some researchers have therefore proposed methods for transferring an entire Q-function or policy [14,15].

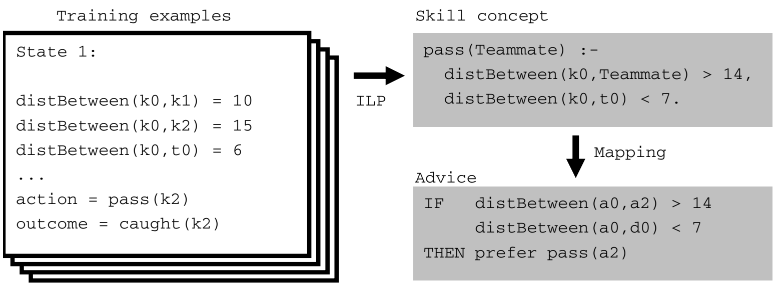
We present an approach that does not use the Q-function to perform transfer. Instead, it analyzes games played in the source task to learn *skills* in first-order logic. By learning high-level concepts, AI^2 favors the transfer of general, behavioral information over the specific, low-level details of the Q-function.

In the AI^2 framework, games are collections of state-action pairs where the action is the classification of the state. It uses these pairs as training examples to learn to classify states. For example, from traces of KeepAway games, AI^2 can learn the concept "states in which passing to a teammate is a good action."

AI^2 can be used when a new task arises in a domain and data from an old task already exists. To use it, the user identifies which skills should be transferred, provides a mapping that relates logical objects in the source task to those in the target task, and optionally gives advice about new or transferred skills.

Table 2. The AI^2 algorithm

GIVEN	DO
Game traces from source task	For each skill to transfer:
List of skills to be transferred	Collect training examples
Object mapping between tasks	Learn rules with Aleph
User advice (optional)	Select rule with highest $F(\beta)$ score
	Translate rule into transfer advice
	Learn target task with all advice



```
State 1:

distBetween(k0,k1) = 10
distBetween(k0,k2) = 15
distBetween(k0,t0) = 6
...
action = pass(k2)
outcome = caught(k2)
```

```
pass(Teammate) :-
  distBetween(k0,Teammate) > 14,
  distBetween(k0,t0) < 7.
```

```
IF   distBetween(a0,a2) > 14
     distBetween(a0,d0) < 7
THEN prefer pass(a2)
```

Fig. 2. Example showing how AI^2 transfers skills

Given this information, AI^2 performs transfer automatically. From existing game traces in the source task, the system learns skill concepts and translates them into advice for the target task. It then applies both the transfer advice and the user advice to learning in the target task.

Table 2 summarizes the AI^2 algorithm in high-level pseudocode. Figure 2 illustrates the transfer part of this algorithm with an example from RoboCup.

Each advice item is a conjunction of conditions and a constraint to be applied if the conditions are met, as shown in Figure 2. Advice need not be followed exactly; it can be refined or even ignored if it disagrees with the learner's experience, using the advice-taking algorithm of Maclin et al. [5], which we explain in Section 4. As we demonstrate with an experiment in Section 5, this provides some protection against imperfect transfer.

3.1 Learning Skills

AI^2 uses inductive logic programming (ILP) to learn skills. ILP is a method for learning first-order conjunctive rules that works with data described by logical relations, such as the RoboCup feature space as presented in Section 2.

A first-order rule, unlike a propositional rule, can contain variables like *Teammate* in Figure 2. The advantage of first-order rules is that they are more general. For example, the rule *pass(Teammate)* is likely to capture the essential elements of the passing skill better than rules for passing to specific teammates. We expect these common skill elements to transfer better to new tasks.

An advantage of ILP in general is that it can accommodate background knowledge for a domain. Our system allows a sophisticated user to define new predicates and add them to the search space. For example, we added predicates to the RoboCup domain to represent aggregate features like "the average distance to an opponent." Defining such mid-level concepts sometimes results in simpler rules.

There are several ILP algorithms for searching the space of possible rules [6]. AI^2 uses the Prolog-based Aleph software package [8], which can conduct both random and heuristic search in the hypothesis space. It selects the rule it finds with the highest F(β) score (a generalization of the more familiar F(1) metric; we use $\beta^2 = 0.1$).

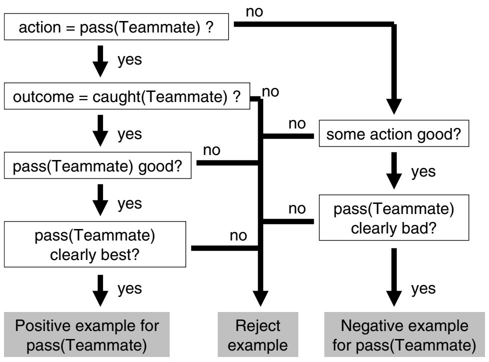

action = pass(Teammate) ?
no
yes
outcome = caught(Teammate) ?
no
yes
pass(Teammate) good?
no
yes
pass(Teammate) clearly best?
no
yes
some action good?
no
yes
pass(Teammate) clearly bad?
no
yes
Positive example for pass(Teammate)
Reject example
Negative example for pass(Teammate)

Fig. 3. Example showing how AI^2 selects training examples

To produce datasets for this search, AI^2 examines states from games in the source task and selects positive and negative examples. In a positive example, several conditions must be met: the skill was performed, the desired outcome occurred, the expected Q-value (using the most recent Q-function) is above a minimum score $minQ_{pos}$ and is at least $ratio_{pos}$ times the predicted Q-values of other actions. In a negative example, some other action was performed, the highest Q-value is above a minimum score $minQ'_{neg}$, and the expected Q-value of the skill being learned is at most $ratio_{neg}$ times the highest Q-value in that state and is below a maximum score $maxQ_{neg}$.

The standard settings in AI^2 are $ratio_{pos} = 1.05$ and $ratio_{neg} = 0.95$, since we have found that Q-values in stochastic domains like RoboCup are often not widely separated. The other parameters are set by the system so that there are at least 100 positive and 100 negative examples, which we have found to be enough to learn reasonable rules. Figure 3 illustrates the sorting process with an example from RoboCup.

3.2 Mapping Skills

To produce advice for the new task, the system translates source-task objects into target-task objects based on the user-provided mapping. For example, a reasonable mapping from 4-on-3 KeepAway to 3-on-2 BreakAway might relate each keeper to an attacker and each taker to the defender.

Not all the objects in the target task need to appear in the mapping. Objects in the source task may be left out too; in this case AI^2 will simply not include those objects in rules. Examples in the mapping above are the BreakAway *goalie* and the KeepAway *fieldCenter*. Similarly, the ILP algorithm will leave out of the search space any predicates in the domain that are not shared by both tasks.

The KBKR advice-taking algorithm ultimately requires advice that is propositionalized for a specific task. Therefore, the final step AI^2 takes in the mapping process is to propositionalize the rules.

First it instantiates skills like *pass(Teammate)* for the target task. For 3-on-2 BreakAway, this would produce two rules, *pass(a1)* and *pass(a2)*. Next it deals with any other conditions in the rule body that contain variables. For example, a rule might have this condition:

$$10 < \text{distBetween(a0, Attacker)} < 20$$

This is effectively a disjunction of conditions: either the distance to *a1* or the distance to *a2* is in the interval $[10, 20]$. Since disjunctions are not part of the advice language, AI^2 uses tile features to represent them. Recall that each feature range is divided into Boolean tiles that take value 1 when the feature value falls into their interval and 0 otherwise. This disjunction is satisfied if at least one of several tiles is active; e.g. for 3-on-2 BreakAway:

$$\text{distBetween(a0, a1)}_{[10,20]} + \text{distBetween(a0, a2)}_{[10,20]} \geq 1$$

If these exact tile boundaries do not exist in the target task, AI^2 adds new tile boundaries to the feature space. Thus transfer advice can be expressed exactly even though the target task feature space is unknown at the time the source task is learned.

It is possible for multiple conditions in a rule to refer to the same variable. For example:

distBetween(a0, Attacker) > 15,
angleDefinedBy(Attacker, a0, ClosestDefender) > 25

Here the variable *Attacker* represents the same object in both clauses, so the system cannot propositionalize the two clauses separately. Instead, it defines a new Boolean background-knowledge predicate:

```
newFeature(Attacker, ClosestDefender) :-
   Dist is distBetween(a0, Attacker),
   Ang is angleDefinedBy(Attacker, a0, ClosestDefender),
   Dist > 15, Ang > 25.
```

It then expresses the required condition using the new feature; e.g. for 3-on-2 BreakAway:

$$\text{newFeature(a1, d0)} + \text{newFeature(a2, d0)} \geq 1$$

AI^2 adds these new Boolean features, which could be considered multi-dimensional tiles, to the target task. Thus transfer advice can actually enhance the feature space of the target task.

3.3 User Advice

Users can optionally include advice in the source-target mapping to further guide transfer by pointing out the differences between the tasks. For example, the passing skills transferred from KeepAway to BreakAway make no distinction between passing toward the goal and away from the goal. Since the new objective is to score goals, players should clearly prefer passing toward the goal. A user could provide this guidance by instructing the system to add a condition like this to the *pass(Teammate)* skill:

$$\text{distBetween(a0, goal) - distBetween(Teammate, goal)} \geq 1$$

Alternatively, an expert user could make use of the system's ability to define new features in the target task. The advantage of this approach is that formally defining the feature allows it to be tiled. To do this, the user would first write the definition in Prolog:

```
diffGoalDistance(Teammate, Value) :-
    DistTeammate is distBetween(Teammate, goal),
    DistA0 is distBetween(a0, goal),
    Value is DistA0 - DistTeammate.
```

Then the user would instruct the system to add to the *pass(Teammate)* rule:

$$\text{diffGoalDistance(Teammate)} \geq 1$$

User advice may also describe new skills that will be needed in the target task. An example is the *shoot* skill in BreakAway, which is an important difference from the KeepAway source task. This type of user advice is not required in AI^2, but it provides a natural and powerful way for users to facilitate transfer.

4 Advice Implementation

The rules produced by transfer or provided by users are likely to be imperfect and may even be incorrect. Therefore, obeying them exactly could prevent effective learning. AI^2 instead treats advice as a soft constraint: an RL agent can selectively refine or ignore advice if it disagrees with the agent's experience in the target task.

AI^2 incorporates advice into Q-learning using a linear optimization method called KBKR. The linear optimizer creates a Q-function by finding, for each action, a weight for each state feature so that the Q-value of each state-action pair in the training set is approximately the weighted sum of the features. It does so by minimizing the following quantity:

$$\text{ModelSize} + C \times \text{DataMisfit} + \mu \times \text{AdviceMisfit}$$

Here *ModelSize* is the sum of the absolute values of the feature weights, *DataMisfit* is the disagreement between the learned function's outputs and the training examples, and *AdviceMisfit* is the disagreement between the learned function's outputs and the advice constraints. The numeric parameters C and μ specify the relative importance of minimizing disagreements versus finding a simple model. AI^2 decays μ over time, so that advice fades as the learner gains experience and no longer requires guidance. When μ becomes essentially zero, AI^2 stops applying advice altogether.

As training progresses, this linear program is resolved after every 25 games. See Maclin et al. [5] for more details.

5 Empirical Results

We present results for AI^2 skill transfer between several RoboCup games. These are challenging transfer scenarios because the games have very different reward structures, as described in Section 2. We also include a study of how the KBKR advice-taking algorithm handles imperfect and incorrect advice. Our player code for these experiments is based on the University of Amsterdam Trilearn players.

5.1 Skill Transfer Experiments

In our first experiment, we use AI^2 to perform transfer from 4-on-3 Mobile Keepaway to 3-on-2 BreakAway. The skill we transfer is *pass(Teammate)*, and we use the mapping described in Section 3.2. We assume the user encourages passing toward the goal by adding the *diffGoalDistance* condition from Section 3.3, and approximates some new skills in BreakAway as follows:

```
IF   distBetween(a0, goalLeft) < 10            AND
     angleDefinedBy(goalLeft, a0, goalie) > 40
THEN prefer shoot(goalLeft) over all actions

IF   distBetween(a0, goalRight) < 10           AND
     angleDefinedBy(goalRight, a0, goalie) > 40
THEN prefer shoot(goalRight) over all actions

IF   distBetween(a0, goalCenter) > 10
THEN prefer moveAhead over moveAway and the 3 shoot actions
```

In our second experiment, we perform transfer from 3-on-2 MoveDownfield to 3-on-2 BreakAway using a similar mapping. The skills we transfer are *pass (Teammate)* and *moveAhead*, and we assume the user advice includes only the *shoot* skills above. That is, we assume that passing forward and moving ahead are learned in MoveDownfield, so the user does not need to provide this guidance.

We now analyze the results for the first experiment in detail. AI^2 learned the following rule from Mobile KeepAway:

```
pass(Teammate) :-
   distBetween(k0, Teammate) > 14,
   angleDefinedBy(Teammate, k0, ClosestTaker) ∈ [30, 150],
   distBetween(k0, Taker) < 7,
   distBetween(k0, Player) < 11.
```

This rule indicates that it is good to pass when an opponent is too close, a teammate is somewhat far away, and no opponent is blocking a pass. AI^2 translates this rule into two items of transfer advice, one per BreakAway teammate, and adds in the user advice.

Figure 4 compares learning curves in BreakAway with and without AI^2 transfer from Mobile KeepAway. It also shows learning curves with the transferred skills and the user advice separately, so that we can analyze their individual

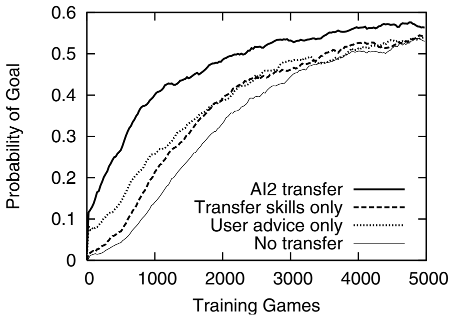


Fig. 4. Learning curves for BreakAway with transfer from Mobile KeepAway

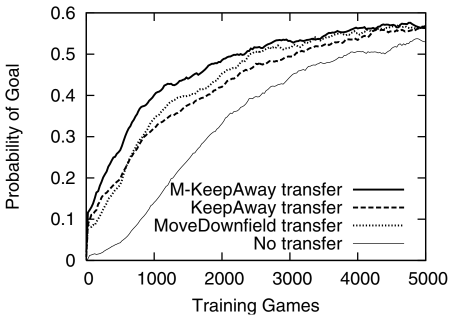


Fig. 5. Learning curves for BreakAway with transfer from several tasks

contributions. Each curve is an average of 10 independent runs with $C = 1500$ and $mu = 10$, and each data point is smoothed over the last 500 games (or all previous games if there are fewer than 500).

Using the transferred skills alone, the scoring probability is higher at the 90% confidence level, based on unpaired t-tests, up to 2500 games. With the full AI^2 system, scoring is more probable at the 95% confidence level at nearly every point. The full system also performs significantly better than either the transferred skills or user advice alone at the 95% confidence level; transferred skills and user hints together perform better than the sum of their parts.

For the second experiment, Figure 5 compares learning curves in BreakAway with and without AI^2 transfer from MoveDownfield. The AI^2 transfer curve for Mobile KeepAway is duplicated here, and we also include results for transfer from the original 3-on-2 KeepAway task [9] to compare the performance of AI^2 transfer in the two KeepAway variants. As expected, Mobile KeepAway transfers better than non-mobile KeepAway. However, both variants as well as MoveDownfield do successfully transfer to BreakAway using AI^2, causing a higher scoring probability at the 95% confidence level at nearly every point.

5.2 Experiments with Imperfect Advice

To demonstrate that AI^2 can cope with imperfect and incorrect advice, we include a third experiment: transfer with intentionally bad user advice. We perform transfer from Mobile KeepAway to BreakAway with the opposite of the user advice above. With its inequalities reversed, this bad advice instructs the learner to pass backwards, shoot when far away from the goal and at a narrow angle, and move when close to the goal.

Figure 6 shows the results of this advice, both in AI^2 transfer and alone. This experiment shows that while bad advice can decrease the positive effect of transfer, it does not cause the AI^2 system to impact learning negatively. On its own, bad advice does have an initial negative effect, but KBKR quickly learns to ignore the advice and the learning curve recovers completely.

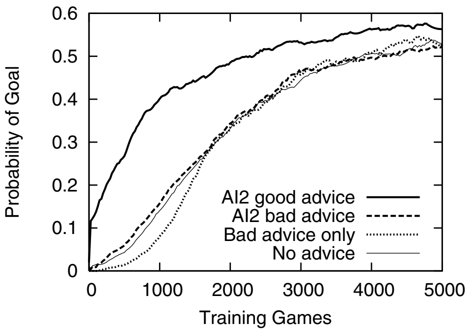


Fig. 6. Learning curve for BreakAway with bad user advice

6 Related Work

Our approach builds on several previous methods for providing advice to reinforcement learners. Maclin and Shavlik [4] develop an IF-THEN advice language to incorporate rules into a neural network for later adjustment. Driessens and Dzeroski [1] use human guidance to create a partial initial Q-function for a relational RL system. Kuhlmann et al. [3] propose a rule-based advice system that increases Q-values by a fixed amount.

Another aspect of our work is extracting explanatory rules from complex functions. Sun [10] studies rule learning from neural-network based reinforcement learners. Fung et al. [2] investigate extracting rules from support vector machines.

We also address knowledge transfer in RL. Singh [7] studies transfer of knowledge between sequential decision tasks. Taylor and Stone [14] copy initial Q-functions to transfer between KeepAway games of different sizes. In Torrey et al. [15] we introduce transfer from KeepAway to BreakAway using the Q-function; we advise each action when its Q-value under the mapped model is highest. In contrast, in this work we learn advice rules in first-order logic to transfer individual skills, working with KeepAway game traces rather than Q-functions. By doing so, we achieve better transfer.

A more detailed study of transfer learning in RoboCup is available online as UW Machine Learning Group Working Paper #06-2.

7 Conclusions and Future Work

Reinforcement learners can benefit significantly from the user-guided transfer of skills from a previous task. We have presented the AI^2 system, which transfers shared skills by learning first-order rules from agent behavior and translating them with a user-designed mapping. This system does not assume a similar reward structure between the source and target tasks and provides robustness to imperfect transfer through advice-taking. Our experimental results demonstrate the effectiveness of this approach in a complex RL domain.

A challenge that we have encountered in RL transfer learning is that differences in action sets and reward structures between the source and target task make it difficult to transfer even shared actions. Changing the game objective or adding a new action changes the meaning of a shared skill. We have addressed this problem with user guidance, using human domain knowledge to help apply transferred skills and encourage the learning of new skills. In the future we hope to reach similar levels of transfer with less user guidance.

We believe that the underlying issue is the separation of *general* from *specific* information in a source task. In RL transfer learning we want to transfer only general aspects of skills in a domain, filtering out task-specific aspects. Our use of ILP to learn general, first-order skill concepts is a step toward this goal. A future step we are considering is learning skills from multiple games in a domain, which we believe may lead to more general rules and therefore better transfer.

References

1. K. Driessens and S. Dzeroski. Integrating experimentation and guidance in relational reinforcement learning. In *Proc. ICML*, 2002.
2. G. Fung, S. Sandilya, and B. Rao. Rule extraction from linear support vector machines. In *Proc. KDD*, 2005.
3. G. Kuhlmann, P. Stone, R. Mooney, and J. Shavlik. Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *AAAI Workshop on Supervisory Control of Learning and Adaptive Systems*, 2004.
4. R. Maclin and J. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22:251–281, 1996.
5. R. Maclin, J. Shavlik, L. Torrey, T. Walker, and E. Wild. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *Proc. AAAI*, 2005.
6. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming 19,20*, pages 629–679, 1994.
7. S. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning 8 (3-4)*, pages 323–339, 1992.
8. A. Srinivasan. The Aleph manual. `http://web.comlab.ox.ac.uk/oucl/research/areas /machlearn/Aleph/aleph.html` , 2001.
9. P. Stone and R. Sutton. Scaling reinforcement learning toward RoboCup soccer. In *Proc. ICML*, 2001.
10. R. Sun. Knowledge extraction from reinforcement learning. *New Learning Paradigms in Soft Computing*, pages 170–180, 2002.
11. R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning 3*, pages 9–44, 1988.
12. R. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Proc. NIPS*, 1996.
13. R. Sutton and A. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.
14. M. Taylor and P. Stone. Behavior transfer for value-function-based reinforcement learning. In *Proc. AAMAS*, 2005.
15. L. Torrey, T. Walker, J. Shavlik, and R. Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Proc. ECML*, 2005.

Constant Rate Approximate Maximum Margin Algorithms

Petroula Tsampouka and John Shawe-Taylor

ECS, University of Southampton, UK

Abstract. We present a new class of Perceptron-like algorithms with margin in which the "effective" learning rate η_{eff}, defined as the ratio of the learning rate to the length of the weight vector, remains constant. We prove that for η_{eff} sufficiently small the new algorithms converge in a finite number of steps and show that there exists a limit of the parameters involved in which convergence leads to classification with maximum margin. A soft margin extension for Perceptron-like large margin classifiers is also discussed.

1 Introduction

It is generally believed that the larger the margin of the solution hyperplane the greater is the generalisation ability of the learning machine [9,1]. The simplest online learning algorithm for binary linear classification, Rosenblatt's Perceptron [6], does not aim at any margin. The problem, instead, of finding the optimal margin hyperplane lies at the core of Support Vector Machines (SVMs) [9,1]. SVMs, however, require solving a quadratic programming problem which makes their efficient implementation difficult and often time consuming.

The difficulty in implementing SVMs has respurred a lot of interest in alternative large margin classifiers many of which are based on the Perceptron algorithm. The most well-known such variants are the standard Perceptron with margin [2,5] and the ALMA [4] algorithms. Here we address the maximum margin classification problem in the context of Perceptron-like algorithms which, however, differ from the above mentioned variants in that the ratio of the learning rate to the length of the weight vector remains constant. This new (class of) algorithm(s), called Constant Rate Approximate Maximum Margin Algorithm(s) (CRAMMA), emerges naturally if one attempts to classify Perceptron-like classifiers with margin in a few very broad categories according to the dependence on time of the misclassification condition or of the effect that an update has on the current weight vector. Under certain conditions CRAMMA converges in a finite number of steps to an approximation of the optimal solution which improves continually as its parameters follow a specific limiting process.

Maximal margin classifiers cannot be used directly in many real-world problems due to the inseparability of the data sets appearing in most applications. To cover the case of inseparable data we discuss a soft margin extension of the hard margin approach which is particularly suited for Perceptron-like large margin classifiers since it does not rely on convex optimisation theory.

A taxonomy of Perceptron-like large margin classifiers can be found in Sect. 2. CRAMMA is described in Sect. 3 together with an analysis regarding its convergence. Section 4 contains our discussion of the soft margin. Section 5 contains some experiments whereas Sect. 6 our conclusions.

2 Taxonomy of Perceptron-Like Large Margin Classifiers

In what follows we make the assumption that we are given a training set which, even if not initially linearly separable can, by an appropriate feature mapping into a space of a higher dimension [9,1], be classified into two categories by a linear classifier. This higher dimensional space in which the patterns are linearly separable will be the considered space. By adding one additional dimension and placing all patterns in the same position at a distance ρ in that dimension we construct an embedding of our data into the so-called augmented space [2]. The advantage of this embedding is that the linear hypothesis in the augmented space becomes homogeneous. Thus, only hyperplanes passing through the origin in the augmented space need to be considered even for tasks requiring bias. Throughout our discussion a reflection with respect to the origin in the augmented space of the negatively labelled patterns is assumed in order to allow for a uniform treatment of both categories of patterns. Also, we use the notation $R = \max_k \|\boldsymbol{y}_k\|$, where $\boldsymbol{y}_k$ is the kth augmented pattern. Obviously, $R \geq \rho$.

The relation characterising optimally correct classification of the training patterns $\boldsymbol{y}_k$ by a weight vector $\boldsymbol{u}$ of unit norm in the augmented space is

$$\boldsymbol{u} \cdot \boldsymbol{y}_k \geq \gamma_{\mathrm{d}} \equiv \max_{\boldsymbol{u}:\|\boldsymbol{u}\|=1} \min_i \{\boldsymbol{u} \cdot \boldsymbol{y}_i\} \qquad \forall k \ . \tag{1}$$

We call the quantity γ_{d} the maximum directional margin. It determines the maximum distance from the origin in the augmented space of the hyperplane normal to $\boldsymbol{u}$ placing all training patterns on the positive side and coincides with the maximum margin in the augmented space with respect to hyperplanes passing through the origin if no reflection is assumed. In the determination of this hyperplane only the direction of $\boldsymbol{u}$ is exploited with no reference to its projection onto the original space. Notice, however, that between γ_{d} and the maximum geometric margin γ in the original space the inequality

$$1 \leq \frac{\gamma}{\gamma_{\mathrm{d}}} \leq \frac{R}{\rho} \tag{2}$$

holds. In the limit $\rho \to \infty$, $R/\rho \to 1$ and from (2) $\gamma_{\mathrm{d}} \to \gamma$ [8]. Thus, with ρ increasing γ_{d} approaches γ.

We concentrate on algorithms that update the augmented weight vector $\boldsymbol{a}_t$ by adding a suitable positive amount in the direction of the misclassified (according to an appropriate condition) training pattern $\boldsymbol{y}_k$. The general form of such an update rule is

$$\boldsymbol{a}_{t+1} = (\boldsymbol{a}_t + \eta_t f_t \boldsymbol{y}_k)\, N_{t+1}^{-1} \ , \tag{3}$$

where η_t is the learning rate which could depend explicitly on the number t of updates that took place so far and f_t an implicit function of the current step (update) t, possibly involving the current weight vector $\boldsymbol{a}_t$ and/or the current misclassified pattern $\boldsymbol{y}_k$, which we require to be bounded by positive constants. We also allow for the possibility of normalising the newly produced weight vector $\boldsymbol{a}_{t+1}$ to a desirable length through a factor N_{t+1}. For the Perceptron $\eta_t = \eta$ is constant, $f_t = 1$ and $N_{t+1} = 1$. Each time the misclassification condition is satisfied by a training pattern the algorithm proceeds to the update of the weight vector. We adopt the convention of initialising t from 1.

A sufficiently general form of the misclassification condition is

$$\boldsymbol{u}_t \cdot \boldsymbol{y}_k \leq C(t) \ , \tag{4}$$

where $\boldsymbol{u}_t$ is the weight vector $\boldsymbol{a}_t$ normalised to unity and $C(t) > 0$ if we require that the algorithm achieves a positive margin. If $\boldsymbol{a}_1 = \mathbf{0}$ we treat the first pattern in the sequence as misclassified. We distinguish two cases depending on whether $C(t)$ is bounded from above by a strictly decreasing function of t which tends to zero or remains bounded from above and below by positive constants. In the first case the minimum directional margin required by such a condition becomes lower than any fixed value provided t is large enough. Algorithms with such a condition have the advantage of achieving some fraction of the unknown existing margin provided they converge. Examples of such algorithms are the well-known standard Perceptron algorithm with margin [2,5], in which the suppression of $C(t) = b/\|\boldsymbol{a}_t\|$ with t increasing is due to the growth of the length of the weight vector, and the ALMA$_2$ algorithm [4] with $C(t) = b/\|\boldsymbol{a}_t\|\sqrt{t}$. In the second case the condition amounts to requiring a directional margin, assumed to exist, which is not lowered arbitrarily with the number t of updates. In particular, if $C(t)$ is equal to a constant β [8] successful termination of the algorithm leads to a solution with margin larger than β. Obviously, convergence is not possible unless $\beta < \gamma_{\mathrm{d}}$. In this case an organised search through the range of possible β values is necessary.

An alternative classification of the algorithms with the perceptron-like update rule (3) is according to the dependence on t of the "effective" learning rate

$$\eta_{\mathrm{eff}\,t} \equiv \frac{\eta_t R}{\|\boldsymbol{a}_t\|} \tag{5}$$

which controls the impact that an update has on the current weight vector. More specifically, $\eta_{\mathrm{eff}\,t}$ determines the update of the direction $\boldsymbol{u}_t$

$$\boldsymbol{u}_{t+1} = \frac{\boldsymbol{u}_t + \eta_{\mathrm{eff}\,t} f_t \boldsymbol{y}_k / R}{\|\boldsymbol{u}_t + \eta_{\mathrm{eff}\,t} f_t \boldsymbol{y}_k / R\|} \ . \tag{6}$$

Again we distinguish two cases depending on whether $\eta_{\mathrm{eff}\,t}$ is bounded from above by a strictly decreasing function of t which tends to zero or remains bounded from above and below by positive constants. We do not consider the case that $\eta_{\mathrm{eff}\,t}$ increases indefinitely with t since we do not expect such algorithms to converge always in a finite number of steps. In the first category belong again the standard Perceptron algorithm, in which $\eta_t = \eta$ remains constant and $\|\boldsymbol{a}_t\|$ is bounded from below by a positive linear function of t, and ALMA$_2$ in which η_t decreases as $1/\sqrt{t}$. In the second category belong algorithms with the fixed directional margin condition $C(t) = \beta$, $\|\boldsymbol{a}_t\|$ normalised to a constant value and fixed learning rate [8].

In summary, the function $C(t)$ entering the misclassification condition and the effective learning rate ${\eta_{\text{eff}}}_t$ of a Perceptron-like algorithm could, roughly speaking, either be suppressed with time or remain practically constant. Thus, we are led to four broad categories of algorithms out of which the one with condition "relaxed" with time and a t-independent η_{eff} has not, to the best of our knowledge, been examined before. This is the subject of the present work.

3 The Constant Rate Approximate Maximum Margin Algorithm CRAMMA$^\epsilon$

We consider algorithms with constant effective learning rate ${\eta_{\text{eff}}}_t = \eta_{\text{eff}}$ in which the misclassification condition takes the form

$$\boldsymbol{u}_t \cdot \boldsymbol{y}_k \leq \frac{\beta}{t^\epsilon} \tag{7}$$

not involving $\|\boldsymbol{a}_t\|$. Here β and ϵ are positive constants. We assume that the initial value $\boldsymbol{u}_1$ of $\boldsymbol{u}_t$ is the unit vector in the direction of the first training pattern. Then,

$$\boldsymbol{u}_t \cdot \boldsymbol{u} > 0 \ . \tag{8}$$

This is true given that, on account of (6), $\boldsymbol{u}_t$ is a linear combination with positive coefficients of the training patterns $\boldsymbol{y}_k$ all of which have positive inner products with the optimal direction $\boldsymbol{u}$ because of (1). Additionally, we set $f_t = 1$. Since the misclassification condition (7) does not depend on $\|\boldsymbol{a}_t\|$ and given that the update (6) of $\boldsymbol{u}_t$ with $f_t = 1$ depends on $\|\boldsymbol{a}_t\|$ only through η_{eff} the algorithm does not depend separately on η_t and $\|\boldsymbol{a}_t\|$ but only on their ratio i.e. on η_{eff}.

```
Require: A linearly separable augmented training set with reflection
assumed $S = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m)$
Define:
For $k = 1, \ldots, m$
$R = \max_k \|\boldsymbol{y}_k\|$ ,  $\bar{\boldsymbol{y}}_k = \boldsymbol{y}_k / R$
Fix: $\eta_{\text{eff}}$,  $\beta_1 \ (= \beta/R)$
Initialisation:
$t = 1$,  $\boldsymbol{u}_1 = \bar{\boldsymbol{y}}_1 / \|\bar{\boldsymbol{y}}_1\|$
repeat until no update made within the for loop
  for $k = 1$ to $m$ do
    if $\boldsymbol{u}_t \cdot \bar{\boldsymbol{y}}_k \leq \beta_t$ then
      $\boldsymbol{u}_{t+1} = \frac{\boldsymbol{u}_t + \eta_{\text{eff}} \bar{\boldsymbol{y}}_k}{\|\boldsymbol{u}_t + \eta_{\text{eff}} \bar{\boldsymbol{y}}_k\|}$
      $t = t + 1$
      $\beta_t = \beta_1 / t^\epsilon$
```

Fig. 1. The Constant Rate Approximate Maximum Margin Algorithm CRAMMA$^\epsilon$

The above (family of) algorithm(s) parametrised in terms of the exponent ϵ and having a constant effective learning rate will be called the Constant Rate Approximate Maximum Margin Algorithm CRAMMA$^\epsilon$ and is summarised in Fig. 1. A justification of the qualification of the algorithm as an "Approximate Maximum Margin" one stems from the following theorem.

Theorem 1. *The* CRAMMA$^\epsilon$ *algorithm of Fig. 1 converges in a finite number of steps provided* $\eta_{\rm eff} < \frac{1}{2}\left(\sqrt{1+8\frac{\gamma_{\rm d}}{R}}-1\right)$*. Moreover, if* $\eta_{\rm eff}$ *is given a dependence on* β *through the relation* $\eta_{\rm eff} = \eta_0 \left(\frac{\beta}{R}\right)^{-\delta}$ *the directional margin* $\gamma'_{\rm d}$ *achieved by the algorithm tends in the limit* $\frac{\beta}{R} \to \infty$ *to the maximum one* $\gamma_{\rm d}$ *provided* $0 < \epsilon\delta < 1$.

Proof. Taking the inner product of (6) with the optimal direction $\boldsymbol{u}$ and expanding $\|\boldsymbol{u}_t + \eta_{\rm eff}\boldsymbol{y}_k/R\|^{-1}$ we have

$$\boldsymbol{u}_{t+1}\cdot\boldsymbol{u} = \left(\boldsymbol{u}_t\cdot\boldsymbol{u} + \eta_{\rm eff}\frac{\boldsymbol{y}_k\cdot\boldsymbol{u}}{R}\right)\left(1 + 2\eta_{\rm eff}\frac{\boldsymbol{y}_k\cdot\boldsymbol{u}_t}{R} + \eta_{\rm eff}^2\frac{\|\boldsymbol{y}_k\|^2}{R^2}\right)^{-\frac{1}{2}}$$

from where, by using the inequality $(1+x)^{-\frac{1}{2}} \geq 1 - \frac{x}{2}$, we get

$$\boldsymbol{u}_{t+1}\cdot\boldsymbol{u} \geq \left(\boldsymbol{u}_t\cdot\boldsymbol{u} + \eta_{\rm eff}\frac{\boldsymbol{y}_k\cdot\boldsymbol{u}}{R}\right)\left(1 - \eta_{\rm eff}\frac{\boldsymbol{y}_k\cdot\boldsymbol{u}_t}{R} - \eta_{\rm eff}^2\frac{\|\boldsymbol{y}_k\|^2}{2R^2}\right) .$$

Thus, we obtain for $\mathcal{D} \equiv \boldsymbol{u}_{t+1}\cdot\boldsymbol{u} - \boldsymbol{u}_t\cdot\boldsymbol{u}$

$$\frac{R}{\eta_{\rm eff}}\mathcal{D} \geq \boldsymbol{y}_k\cdot\boldsymbol{u} - (\boldsymbol{u}_t\cdot\boldsymbol{u})(\boldsymbol{y}_k\cdot\boldsymbol{u}_t) - \frac{\eta_{\rm eff}}{2R}\left(\|\boldsymbol{y}_k\|^2\,\boldsymbol{u}_t\cdot\boldsymbol{u} + 2(\boldsymbol{y}_k\cdot\boldsymbol{u})(\boldsymbol{y}_k\cdot\boldsymbol{u}_t)\right) - \frac{\eta_{\rm eff}^2}{2R^2}\|\boldsymbol{y}_k\|^2\,\boldsymbol{y}_k\cdot\boldsymbol{u} .$$

By employing (1), (7) and (8) we get a lower bound on $\mathcal{D}$

$$\frac{\mathcal{D}}{\eta_{\rm eff}} \geq \left(\frac{\gamma_{\rm d}}{R} - \frac{\eta_{\rm eff}}{2} - \frac{\eta_{\rm eff}^2}{2}\right) - (1+\eta_{\rm eff})\frac{\beta}{R}t^{-\epsilon} . \tag{9}$$

From the misclassification condition it is obvious that convergence of the algorithm is impossible unless $\beta/t^\epsilon < \gamma_{\rm d}$ i.e.

$$t > t_0 \equiv \left(\frac{\beta}{\gamma_{\rm d}}\right)^{\frac{1}{\epsilon}} . \tag{10}$$

A repeated application of (9) $(t-[t_0])$ times yields

$$\frac{\boldsymbol{u}_{t+1}\cdot\boldsymbol{u} - \boldsymbol{u}_{[t_0]+1}\cdot\boldsymbol{u}}{\eta_{\rm eff}} \geq \left(\frac{\gamma_{\rm d}}{R} - \frac{\eta_{\rm eff}}{2} - \frac{\eta_{\rm eff}^2}{2}\right)(t-[t_0]) - (1+\eta_{\rm eff})\frac{\beta}{R}\sum_{m=[t_0]+1}^{t} m^{-\epsilon}$$

with $[t_0]$ denoting the integer part of t_0. By employing the inequality

$$\sum_{m=[t_0]+1}^{t} m^{-\epsilon} \leq \int_{t_0}^{t} m^{-\epsilon} dm + t_0^{-\epsilon} = \frac{t^{1-\epsilon} - t_0^{1-\epsilon}}{1-\epsilon} + t_0^{-\epsilon}$$

and taking into account (8) we finally obtain

$$1 \geq \eta_{\text{eff}} \left(\frac{\gamma_{\text{d}}}{R}\right) \chi \left(t - t_0\right) - \eta_{\text{eff}} \left(1 + \eta_{\text{eff}}\right) \frac{\beta}{R} \frac{\left(t^{1-\epsilon} - t_0^{1-\epsilon}\right)}{1-\epsilon} - \omega \ . \tag{11}$$

Here

$$\chi \equiv \left(1 - \frac{\eta_{\text{eff}}}{2} \left(1 + \eta_{\text{eff}}\right) \frac{R}{\gamma_{\text{d}}}\right) \quad \text{and} \quad \omega \equiv \eta_{\text{eff}} \left(1 + \eta_{\text{eff}}\right) \frac{\gamma_{\text{d}}}{R} \ .$$

Let us define the new variable $\tau \geq 0$ through the relation

$$t = t_0 \left(1 + \tau\right) = \left(\frac{\beta}{\gamma_{\text{d}}}\right)^{\frac{1}{\epsilon}} \left(1 + \tau\right) \ . \tag{12}$$

In terms of τ (11) becomes

$$\frac{1}{\eta_{\text{eff}}} \left(\frac{\beta}{R}\right)^{-\frac{1}{\epsilon}} \left(\frac{\gamma_{\text{d}}}{R}\right)^{\left(\frac{1}{\epsilon}-1\right)} \left(1 + \omega\right) \geq \chi\tau - \left(1 + \eta_{\text{eff}}\right) \frac{\left(1+\tau\right)^{1-\epsilon} - 1}{1-\epsilon} \ . \tag{13}$$

Let $g(\tau)$ be the r.h.s. of the above inequality. Since $\chi > 0$, given that $\eta_{\text{eff}} < \frac{1}{2}\left(\sqrt{1 + 8\frac{\gamma_{\text{d}}}{R}} - 1\right)$, it is not difficult to verify that $g(\tau)$ (with $\tau \geq 0$) is unbounded from above and has a single extremum, actually a minimum, at $\tau_{\min} = \left(1 + \eta_{\text{eff}}\right)^{\frac{1}{\epsilon}} \chi^{-\frac{1}{\epsilon}} - 1 > 0$ with $g(\tau_{\min}) < 0$. Moreover, the l.h.s of (13) is positive. Therefore, there is a single value τ_{b} of τ where (13) holds as an equality which provides an upper bound on τ

$$\tau \leq \tau_{\text{b}} \tag{14}$$

satisfying $\tau_{\text{b}} > \tau_{\min} > 0$. Combining (12) and (14) we obtain the bound on the number of updates

$$t \leq t_{\text{b}} \equiv \left(\frac{\beta}{\gamma_{\text{d}}}\right)^{\frac{1}{\epsilon}} \left(1 + \tau_{\text{b}}\right) \tag{15}$$

proving that the algorithm converges in a finite number of steps. From (15) and taking into account the misclassification condition (7) we obtain a lower bound $\beta/t_{\text{b}}^{\epsilon}$ on the margin γ'_{d} achieved. Thus, the fraction f of γ_{d} that the algorithm achieves satisfies

$$f \equiv \frac{\gamma'_{\text{d}}}{\gamma_{\text{d}}} \geq f_{\text{b}} \equiv \frac{\beta/\gamma_{\text{d}}}{t_{\text{b}}^{\epsilon}} = \left(1 + \tau_{\text{b}}\right)^{-\epsilon} \ . \tag{16}$$

Let us assume that $\frac{\beta}{R} \to \infty$ in which case from $\eta_{\text{eff}} = \eta_0 \left(\frac{\beta}{R}\right)^{-\delta}$ we have that $\eta_{\text{eff}} \to 0$. Consequently $\chi \to 1$, $\omega \to 0$ and (13) becomes

$$\frac{1}{\eta_0} \left(\frac{\beta}{R}\right)^{-\left(\frac{1}{\epsilon}-\delta\right)} \left(\frac{\gamma_{\text{d}}}{R}\right)^{\left(\frac{1}{\epsilon}-1\right)} \geq \tau - \frac{\left(1+\tau\right)^{1-\epsilon} - 1}{1-\epsilon} \ . \tag{17}$$

Provided $\epsilon\delta < 1$ the l.h.s. of the above inequality vanishes in the limit $\frac{\beta}{R} \to \infty$. Then, since $\tau_{\min}$ vanishes as well, the r.h.s. of the inequality becomes a strictly increasing function of τ and (17) obviously holds as an equality only for $\tau = 0$. Therefore,

$$\tau_{\mathrm{b}} \to \tau_{\min} \to 0 \quad \text{as} \quad \frac{\beta}{\mathrm{R}} \to \infty \ . \tag{18}$$

Combining (16) with (18) and taking into account that $f \leq 1$ by definition we conclude that

$$f \to 1 \quad \text{as} \quad \frac{\beta}{\mathrm{R}} \to \infty \ .$$

□

Remark 1. In the case $\epsilon = \frac{1}{2}$ by solving the quadratic equation derived from (13) we obtain explicitly an upper bound t_{b} on the number of updates and a lower bound f_{b} on the fraction f of the margin that the algorithm achieves. They are the ones of (15) and (16), respectively with

$$\tau_{\mathrm{b}} = \left\{ \frac{1+\eta_{\mathrm{eff}}}{\chi} + \sqrt{\left(\frac{1+\eta_{\mathrm{eff}}}{\chi} - 1\right)^2 + \eta_{\mathrm{eff}}^{-1}\left(\frac{\beta}{R}\right)^{-2} \frac{\gamma_{\mathrm{d}}\,(1+\omega)}{\chi R}} \right\}^2 - 1 \ . \tag{19}$$

As $\frac{\beta}{R} \to \infty$, $\eta_{\mathrm{eff}} = \eta_0 \left(\frac{\beta}{R}\right)^{-\delta} \to 0$, $\chi \to 1$ and $\omega \to 0$. Then, $\tau_{\mathrm{b}} \to 0$ given that $\eta_{\mathrm{eff}}^{-1}\left(\frac{\beta}{R}\right)^{-2} = \eta_0^{-1}\left(\frac{\beta}{R}\right)^{\delta-2} \to 0$ if $0 < \delta < 2$. This demonstrates explicitly the statement of Theorem 1. Explicit bounds t_{b} and f_{b} are also obtainable for $\epsilon = 2$.

4 Soft Margin Extension

Maximal margin classifiers, representing the hard margin approach, cannot be employed in many real-world problems since there is in general no linear separation in the feature space and the use of powerful kernels might lead to overfitting. The most widely accepted solution to this problem is the adoption of the so-called soft margin approach. In the SVM formulation [9,1] the soft margin approach is implemented through the introduction of “slack” variables in order to allow for violations of the margin condition by some training patterns.

Freund and Shapire [3] have shown how a function of the margin distribution different from the minimum margin one can be used to bound the number of mistakes of an online Perceptron algorithm. Their technique makes the data set linearly separable by extending the instance space by as many dimensions as the number of instances and placing each instance at a distance $|\Delta|$ from the origin in the corresponding dimension. An interesting result in this connection is the observation that the hard margin optimisation task in the extended space is equivalent to the soft margin optimisation in the original instance space if the 2-norm of the slack variables is employed [7].

In the sequel, following the approach of [3], we show how one moves in the direction of minimising an objective function $\mathcal{J}$ involving the new margin distribution by making use of Perceptron-like algorithms which, however, are seeking a hard margin in the extended space. This may not be surprising in the light of the result just mentioned regarding the equivalence between the hard margin optimisation in the extended space and the soft margin one in the original space. Nevertheless, we hope that our analysis, which does not rely on convex optimisation theory, will contribute to a better understanding of what an algorithm running in the extended space actually achieves with respect to the original space. Although our instance space prior to its extension is the augmented one in the present section the instances $\boldsymbol{y}_k$ are explicitly accompanied by their labels l_k since we found convenient not to assume a reflection with respect to the origin.

Theorem 2. *Let $((\boldsymbol{y}_1, l_1), \ldots, (\boldsymbol{y}_m, l_m))$ be a sequence of m labelled instances, $\boldsymbol{u}$ a unit vector and $\gamma > 0$. Define $d_i = \max\{0, \gamma - l_i \boldsymbol{u} \cdot \boldsymbol{y}_i\}$ and set $D = \sqrt{\sum_i d_i^2}$. In addition define an extended instance space $\boldsymbol{y}_i^{\text{ext}} = (\boldsymbol{y}_i, \Delta\delta_{1i}, \ldots, \Delta\delta_{mi})$ parametrised by Δ, where δ_{ij} is Kronecker's δ.*

1. *Let $\Gamma_{\Delta\text{opt}}$ be the maximum margin in the extended space with respect to hyperplanes passing through the origin. Then, for any $\boldsymbol{u}$ and γ,*

$$\Gamma_{\Delta\text{opt}}^{-2} \leq \mathcal{J}(\boldsymbol{u}, \gamma, \Delta) \equiv \frac{1}{\gamma^2} + \frac{1}{\Delta^2}\left(\frac{D}{\gamma}\right)^2 \quad . \tag{20}$$

2. *Assume that a zero-threshold algorithm converges in the extended space to a solution vector $\boldsymbol{a}^{\text{ext}}$ which describes a hyperplane passing through the origin with margin Γ_Δ. Let $\boldsymbol{u} = \boldsymbol{a}/\|\boldsymbol{a}\|$ and $\gamma = \Gamma_\Delta \|\boldsymbol{a}^{\text{ext}}\| / \|\boldsymbol{a}\|$, where $\boldsymbol{a}$ is the projection of $\boldsymbol{a}^{\text{ext}}$ onto the original instance space. Then, employing such a $\boldsymbol{u}$ and γ provided by the algorithm, we have*

$$\Gamma_{\Delta\text{opt}}^{-2} \leq \mathcal{J}(\boldsymbol{u}, \gamma, \Delta) \leq \Gamma_\Delta^{-2} \quad . \tag{21}$$

Proof. 1. Notice that $\mathcal{J}(\boldsymbol{u}, \gamma, \Delta) = Z^2/\gamma^2$ with $Z \equiv \sqrt{1 + D^2/\Delta^2}$. Then, (20) is equivalent to $\gamma/Z \leq \Gamma_{\Delta\text{opt}}$ which is proved in [3].

2. Let us assume that a zero-threshold algorithm converges in the extended space to a weight vector $\boldsymbol{a}^{\text{ext}}$ in the direction of the unit vector $\boldsymbol{u}^{\text{ext}}$

$$\boldsymbol{u}^{\text{ext}} = \frac{\boldsymbol{a}^{\text{ext}}}{\|\boldsymbol{a}^{\text{ext}}\|} = \frac{1}{Z'}\left(\boldsymbol{u}, l_1\frac{d'_1}{\Delta}, \ldots, l_i\frac{d'_i}{\Delta}, \ldots, l_m\frac{d'_m}{\Delta}\right) \quad , \tag{22}$$

where $Z' = \|\boldsymbol{a}^{\text{ext}}\|/\|\boldsymbol{a}\| = \sqrt{1 + D'^2/\Delta^2}$ with $D' = \sqrt{\sum_i d_i'^2}$. Here $\boldsymbol{a}$ is the projection of $\boldsymbol{a}^{\text{ext}}$ onto the original instance space and $\boldsymbol{u}$ is the unit vector in the direction of $\boldsymbol{a}$. Let Γ_Δ be the margin achieved by $\boldsymbol{u}^{\text{ext}}$ and $\gamma \equiv \Gamma_\Delta Z'$. We have

$$l_i \boldsymbol{u}^{\text{ext}} \cdot \boldsymbol{y}_i^{\text{ext}} = \frac{1}{Z'}\left(l_i \boldsymbol{u} \cdot \boldsymbol{y}_i + d'_i\right) \geq \Gamma_\Delta = \frac{\gamma}{Z'}$$

from where

$$d'_i \geq \gamma - l_i \boldsymbol{u} \cdot \boldsymbol{y}_i \quad . \tag{23}$$

The above inequality, taking into account the definition of d_i, leads to

$$|d_i'| \geq d_i \geq 0 \tag{24}$$

and consequently to $Z' \geq Z$. Therefore, taking into consideration the definition of γ, we obtain

$$\frac{\gamma}{Z} \geq \frac{\gamma}{Z'} = \Gamma_\Delta \tag{25}$$

which leads to

$$\mathcal{J}(\boldsymbol{u}, \gamma, \Delta) \leq \Gamma_\Delta^{-2} \tag{26}$$

given that $Z^2/\gamma^2 = \mathcal{J}(\boldsymbol{u}, \gamma, \Delta)$. The proof is completed by combining (20) and (26). □

Remark 2. Let the zero-threshold algorithm be a Perceptron-like algorithm with initial weight vector $\boldsymbol{a}_1^{\text{ext}} = \sum_k \alpha_k l_k \boldsymbol{y}_k^{\text{ext}}$ and $\alpha_k \geq 0$. From the initialisation, the update rule (3) and the definition of the extended space follows that $d_i' \geq 0$.

Remark 3. If the algorithm converges to the maximal margin hyperplane passing through the origin in the extended space then $\Gamma_\Delta = \Gamma_{\Delta\text{opt}}$. Moreover, (20) is equivalent to $\gamma/Z \leq \Gamma_{\Delta\text{opt}}$ which combined with (25) and given that $\Gamma_\Delta = \Gamma_{\Delta\text{opt}}$ gives $Z' = Z$ or $D' = D$ from where $|d_i'| = d_i$ follows taking into account (24). In addition, $d_i' \geq 0$. Indeed, if $d_i' < 0$ then $d_i = 0$ because of (23) and the definition of d_i. But in this case $d_i' = d_i = 0$ contradicting our assumption that $d_i' < 0$. Thus, for the optimal extended space solution $d_i' = d_i$.

Remark 4. Setting $\boldsymbol{w} = \boldsymbol{u}/\gamma$, $\xi_i = |d_i'|/\gamma \geq d_i/\gamma = \max\{0, 1 - l_i \boldsymbol{w} \cdot \boldsymbol{y}_i\}$ and $C = \Delta^{-2}$ yields

$$\frac{1}{\gamma^2} + \frac{1}{\Delta^2}\left(\frac{D'}{\gamma}\right)^2 = \|\boldsymbol{w}\|^2 + C\sum_i \xi_i^2 \ .$$

We recognise the objective function of the primal form of the 2-norm soft margin optimisation problem in which the role of the constraints is played by (24) but the bias term is missing since it is, at least partially, incorporated in the augmented weight vector $\boldsymbol{w}$. If the optimal solution is found $d_i' = d_i$ and the "slack" variables ξ_i become $\xi_i = \max\{0, 1 - l_i \boldsymbol{w} \cdot \boldsymbol{y}_i\}$.

Theorem 2 shows that minimisation of the objective function $\mathcal{J}$ is equivalent to finding the maximum margin in the extended space. The $\boldsymbol{u}$ and γ for which the minimum $\mathcal{J}_{\min}$ is attained determine uniquely both the maximum margin $\Gamma_{\Delta\text{opt}} = \mathcal{J}_{\min}^{-\frac{1}{2}}$ and the direction $\boldsymbol{u}_{\text{opt}}^{\text{ext}}$ of the optimal weight vector in the extended space which is given by (22) with $d_i' = d_i$. Moreover, (21) provides an estimate of the deviation of the value of $\mathcal{J}$ achieved as a result of an incomplete optimisation from $\mathcal{J}_{\min}$ if we have an estimate of the difference between Γ_Δ and $\Gamma_{\Delta\text{opt}}$.

We conclude this section with a well-known lower bound on the margin $\Gamma_{\Delta\text{opt}}$. Let $\boldsymbol{u}^{\text{ext}} = \text{sgn}(\Delta) m^{-\frac{1}{2}} (\boldsymbol{0}, l_1, l_2, \ldots, l_m)$ be an extended unit vector with vanishing projection onto the original instance space. It is straightforward to see that $l_i \boldsymbol{u}^{\text{ext}} \cdot \boldsymbol{y}_i^{\text{ext}} = |\Delta|/\sqrt{m}$ meaning that $\boldsymbol{u}^{\text{ext}}$ achieves a margin of $|\Delta|/\sqrt{m}$. Thus,

$$\Gamma_{\Delta\text{opt}} \geq |\Delta|/\sqrt{m} \ . \tag{27}$$

Table 1. Results for the sonar data set. The directional margin γ'_{d} achieved and the number of updates (upds) are given for the Perceptron, ALMA$_2$ and CRAMMA$^{\frac{1}{2}}$. For CRAMMA$^{\frac{1}{2}}$ we choose $\eta_{\mathrm{eff}} = 0.001 \left(\frac{\beta}{R}\right)^{-1}$.

Perceptron			ALMA$_2$			CRAMMA$^{\frac{1}{2}}$		
$\frac{b}{\eta R^2}$	$10^3\gamma'_{\mathrm{d}}$	upds	α	$10^3\gamma'_{\mathrm{d}}$	upds	$\frac{\beta}{R}$	$10^3\gamma'_{\mathrm{d}}$	upds
1	5.78	247,140	0.75	5.65	415,119	0.73	5.73	248,267
3.1	7.14	660,698	0.55	7.08	1,584,785	1.52	7.12	669,170
5.4	7.45	1,117,124	0.45	7.45	3,133,968	2	7.46	1,047,757
20	7.80	3,977,612	0.35	7.76	6,657,109	3	7.82	2,143,989
90	7.91	17,647,271	0.3	7.91	10,170,590	3.45	7.92	2,762,005
200	7.92	39,131,402	0.2	8.11	28,339,340	5	8.11	5,531,113
500	7.93	97,717,549	0.1	8.27	137,693,242	10	8.28	21,220,354
1000	7.93	195,358,932	0.03	8.37	1,735,836,937	30.1	8.37	188,073,965

Table 2. Results for the sonar data set with CRAMMA^2 and $\eta_{\mathrm{eff}} = 0.4(\frac{\beta}{R})^{-0.3}$

$\frac{\beta}{R}$	10^6	10^7	10^8	10^9	10^{10}	10^{11}	10^{12}	10^{13}
$10^3\gamma'_{\mathrm{d}}$	1.03	3.66	5.52	6.69	7.37	7.80	8.10	8.27
upds	70,798	106,507	264,396	756,035	2,275,334	6,994,002	21,690,267	67,893,557

5 Experiments

In this section we present the results of experiments performed in order to verify our theoretical analysis and evaluate the performance of the CRAMMA$^{\epsilon}$ algorithm in comparison with the other two well-known similar in spirit algorithms, namely the Perceptron with margin and ALMA$_2$ [1].

First we analyse the training data set of the sonar classification problem as originally selected for the aspect-angle dependent experiment. It consists of 104 patterns each with 60 attributes obtainable from the UCI repository. Here the data are embedded in the augmented space at a distance $\rho = 1$ from the origin in the additional dimension leading to $R \simeq 3.8121$ and $\gamma_{\mathrm{d}} \simeq 0.00841$. The results of our comparative study of the Perceptron, ALMA$_2$ and CRAMMA$^{\frac{1}{2}}$ ($\epsilon = \frac{1}{2}$) algorithms are presented in Table 1. We observe that for values of the margin γ'_{d} near the maximum one CRAMMA$^{\frac{1}{2}}$ is certainly the fastest by far. Moreover, the data suggest that the Perceptron is not able to approach the maximum margin arbitrarily close. We also present in Table 2 results obtained by the CRAMMA^2 ($\epsilon = 2$) algorithm.

We additionally analyse a linearly separable data set, which we call WBC$_{-11}$, consisting of 672 patterns each with 9 attributes. It is constructed from the

[1] The parameters for ALMA$_2$ were chosen to correspond to the ones of the theorem in [4] if the data are normalised such that the longest pattern has unit length. The parameter $\alpha \in (0, 1]$ controls the accuracy to which the maximum margin is approximated.

Table 3. Results for the WBC_{-11} data set for the algorithms Perceptron, ALMA_2 and $\mathrm{CRAMMA}^{\frac{1}{2}}$. For $\mathrm{CRAMMA}^{\frac{1}{2}}$ the choice $\eta_{\mathrm{eff}} = 0.0001 \left(\frac{\beta}{R}\right)^{-1}$ is made.

Perceptron			ALMA_2			$\mathrm{CRAMMA}^{\frac{1}{2}}$		
$\frac{b}{\eta R^2}$	$10^2\gamma'_{\mathrm{d}}$	upds	α	$10^2\gamma'_{\mathrm{d}}$	upds	$\frac{\beta}{R}$	$10^2\gamma'_{\mathrm{d}}$	upds
0.52	1.784	1,718,705	0.8	1.783	2,704,553	0.22	1.794	259,036
0.9	2.008	2,720,447	0.7	2.008	6,254,523	0.32	2.019	431,543
1.4	2.141	3,976,477	0.6	2.141	13,320,425	0.42	2.143	660,486
2.1	2.228	5,734,457	0.5	2.228	27,666,246	0.49	2.238	824,120
4	2.317	10,508,566	0.35	2.315	88,363,792	0.8	2.318	2,044,555

Wisconsin Breast Cancer (WBC) data set obtainable from the UCI repository by first omitting the 16 patterns with missing features and subsequently removing from the data set containing the remaining 683 patterns the 11 patterns having the positions 2, 4, 191, 217, 227, 245, 252, 286, 307, 420 and 475. The value $\rho = 30$ is chosen for the parameter ρ of the augmented space leading to $R = \sqrt{1716}$ and $\gamma_{\mathrm{d}} \simeq 0.0243$. In Table 3 we present the results of a comparative study of the Perceptron, ALMA_2 and $\mathrm{CRAMMA}^{\frac{1}{2}}$ algorithms. The superiority of the performance of the $\mathrm{CRAMMA}^{\frac{1}{2}}$ on this data set is apparent.

Table 4. Results for the (extended) WBC data set (with $\Delta = 1$). The relative deviations $\frac{\delta D}{D}$ and $\frac{\delta \Gamma}{\Gamma}$, the margin Γ_Δ and the number of updates (upds) are given for the Perceptron, ALMA_2 and $\mathrm{CRAMMA}^{\frac{1}{2}}$. For $\mathrm{CRAMMA}^{\frac{1}{2}}$ we choose $\eta_{\mathrm{eff}} = \frac{1.7}{R\sqrt{683}} \left(\frac{\beta}{R}\right)^{-1}$.

Perceptron					ALMA_2					$\mathrm{CRAMMA}^{\frac{1}{2}}$				
$\frac{b}{\eta R^2}$	$10\frac{\delta D}{D}$	$10\frac{\delta \Gamma}{\Gamma}$	$10\Gamma_\Delta$	upds	α	$10\frac{\delta D}{D}$	$10\frac{\delta \Gamma}{\Gamma}$	$10\Gamma_\Delta$	upds	$\frac{\beta}{R}$	$10\frac{\delta D}{D}$	$10\frac{\delta \Gamma}{\Gamma}$	$10\Gamma_\Delta$	upds
1	2.22	2.14	1.0244	67,913	0.75	2.18	2.19	1.0185	79,061	0.95	2.36	2.22	1.0143	80,671
2.4	1.13	1.11	1.1585	144,938	0.6	1.13	1.16	1.1524	248,461	1.64	1.10	1.12	1.1568	185,687
10	0.39	0.38	1.2542	560,591	0.35	0.42	0.42	1.2481	1,625,682	3.1	0.36	0.37	1.2551	560,229
45	0.19	0.19	1.2789	2,474,607	0.2	0.19	0.19	1.2784	7,184,572	5	0.18	0.19	1.2791	1,401,588
700	0.15	0.15	1.2837	38,336,601	0.1	0.08	0.08	1.2933	35,542,412	11.5	0.08	0.08	1.2934	7,252,904

Finally, we test our algorithms on the extended instance space constructed from the linearly inseparable WBC data set comprising 683 patterns each with 9 attributes after ignoring the 16 patterns with missing attributes. We embed the data in the augmented space at a distance $\rho = 10$ from the origin in the additional dimension and we subsequently construct the extended instance space parametrised by $\Delta = 1$. In order to determine the value of η_{eff} we take advantage of the lower bound (27) on the margin $\Gamma_{\Delta\mathrm{opt}}$ of the extended space and set $\eta_{\mathrm{eff}} = 1.7|\Delta|/R\sqrt{m}$ for $\beta = R$ which satisfies the constraint of Theorem 1. Here $m = 683$ and $R = \sqrt{917}$. We also take advantage of another property of the extended space in order to attempt an assessment of the relative deviation of the margin Γ_Δ found from the (unknown) maximum $\Gamma_{\Delta\mathrm{opt}}$: the quantities D and D' defined in Sect. 4 for which $D' \geq D$ holds become equal, according to Remark 3, if the optimal extended solution vector is found. Thus, we may take the relative deviation $\delta D/D \equiv (D' - D)/D$ as a measure of the departure from optimality. In Table 4 we give the results of our comparative study of the Perceptron, ALMA$_2$ and CRAMMA$^{\frac{1}{2}}$ algorithms. We observe that once again CRAMMA$^{\frac{1}{2}}$ is the fastest near the maximum margin $\Gamma_{\Delta\mathrm{opt}} \simeq 0.13033$ where the objective function $\mathcal{J}$ is minimised. Moreover, $\delta D/D$ proves a surprisingly accurate measure of the relative deviation $\delta\Gamma/\Gamma \equiv (\Gamma_{\Delta\mathrm{opt}} - \Gamma_{\Delta})/\Gamma_{\Delta\mathrm{opt}}$ of Γ_{Δ} from $\Gamma_{\Delta\mathrm{opt}}$.

6 Conclusions

We presented a new class of Perceptron-like large margin classifiers characterised by a constant effective learning rate. Our theoretical approach proved sufficiently powerful in establishing asymptotic convergence to the optimal hyperplane for a whole class of such algorithms in which the misclassification condition is relaxed with an arbitrary power of the number of updates. Thus, it becomes obvious that the ability to approach the maximum margin arbitrarily close is not a property of some very special algorithmic constructions but, instead, characterises larger families of algorithms under rather mild assumptions. We additionally discussed a soft margin extension for Perceptron-like large margin classifiers. Finally, we provided experimental evidence in support of our theoretical analysis.

References

1. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines (2000) Cambridge, UK: Cambridge University Press
2. Duda, R.O., Hart, P.E.: Pattern Classsification and Scene Analysis (1973) Wiley
3. Freund, Y., Shapire, R. E.: Large margin classification using the perceptron algorithm. Machine Learning **37**(3) (1999) 277–296
4. Gentile C.: A new approximate maximal margin classification algorithm. Journal of Machine Learning Research **2** (2001) 213–242
5. Krauth, W., Mézard, M.: Learning algorithms with optimal stability in neural networks. Journal of Physics **A 20** (1987) L745–L752
6. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review **65**(6) (1958) 386–408
7. Shawe-Taylor, J., Cristianini, N.: Further results on the margin distribution. In COLT'99 (1999) 278–285
8. Tsampouka, P., Shawe-Taylor, J.: Analysis of generic perceptron-like large margin classifiers. ECML 2005, LNAI **3720** (2005) 750–758, Springer-Verlag
9. Vapnik, V. N.: The Nature of Statistical Learning Theory (1995) Springer Verlag

Batch Classification with Applications in Computer Aided Diagnosis

Volkan Vural[1], Glenn Fung[2], Balaji Krishnapuram[2], Jennifer Dy[1], and Bharat Rao[2]

[1] Department of Electrical and Computer Engineering, Northeastern University
[2] Computer Aided Diagnosis and Therapy, Siemens Medical Solutions, USA

Abstract. Most classification methods assume that the samples are drawn independently and identically from an unknown data generating distribution, yet this assumption is violated in several real life problems. In order to relax this assumption, we consider the case where batches or groups of samples may have internal correlations, whereas the samples from different batches may be considered to be uncorrelated. Two algorithms are developed to classify all the samples in a batch jointly, one based on a probabilistic analysis and another based on a mathematical programming approach. Experiments on three real-life *computer aided diagnosis* (CAD) problems demonstrate that the proposed algorithms are significantly more accurate than a naive SVM which ignores the correlations among the samples.

1 Introduction

Most classification systems assume that the data used to train and test the classifier is independently and identically distributed. For example, samples are classified one at a time in a *support vector machine* (SVM), thus the classification of a particular test sample does not depend on the features from any other test sample. Nevertheless, this assumption is commonly violated in many real-life problems where sub-groups of samples have a high degree of correlation amongst both their features and their labels.

Good examples of the problem described above are *computer aided diagnosis* (CAD) applications where the goal is to detect structures of interest to physicians in medical images: e.g., to identify potentially malignant tumors in computed tomography (CT) scans, X-ray images, etc. In an almost universal paradigm for CAD algorithms, this problem is addressed by a three-stage system: (1) identification of potentially unhealthy candidates *regions of interest* (ROI) from a medical image, (2) computation of descriptive features for each candidate, and (3) classification of each candidate (e.g. normal or diseased) based on its features. CAD applications were the main motivation for the work presented in this paper, although the algorithms presented here can be applied to any problem where the data is provided in batches of samples.

As an illustrative example, consider Figure 1, a CT image of a lung showing circular marks that point to potential diseased candidate regions that are detected by a CAD algorithm. There are five candidates on the left and six candidates on the right (marked by circles) in Figure 1. Descriptive features are extracted for each candidate and each candidate region is classified as healthy or unhealthy.

In this setting, correlations exist among both the features and the labels of candidates belonging to the same (batch) image both in the training data-set and in the unseen

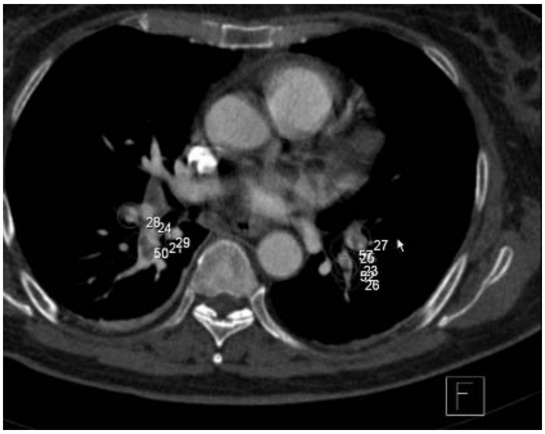


Fig. 1. Two emboli as they are detected by the Candidate Generation algorithm in a CT image. The candidates are shown as five circles for the left embolus & six circles for the right embolus. The disease status of spatially overlapping or proximate candidates is highly correlated.

testing data. Further, the level of correlation is a function of the pairwise-distance between candidates: the disease status (class-label) of a candidate is highly correlated with the status of other spatially proximate candidates, but the correlations decrease as the distance is increased. Most conventional CAD algorithms classify one candidate at a time, ignoring the correlations amongst the candidates in an image. By explicitly accounting for the correlation structure between the labels of the test samples, the algorithms proposed in this paper improve the classification accuracy significantly.

Beyond the domain of CAD applications, our algorithms are quite general and may be used for batch-wise classification problems in many other contexts. In general, the proposed classifiers can be used whenever data samples are presented in independent batches. In the CAD example, the batch corresponds to the candidate ROIs from an image, but in other contexts a batch may correspond to data from the same hospital, the patients treated by the same doctor or nurse, etc.

1.1 Related Work

In natural language processing (NLP), *conditional random fields* (CRF) [4] and recently *maximum margin Markov* (MMM) networks [7] are used to identify part-of-speech information about words by using the context of nearby words. CRF are also used in similar applications in spoken word recognition. We are not aware of previous work on CAD algorithms that exploit internal correlations among the samples. However, while CRF and MMM are also fairly general algorithms, they are both computationally very demanding and it is also not very easy to implement them for problems where the relationship structure between the samples is in any form other than a linear chain (as in the text and speech processing applications). Certainly their application would be difficult in many large-scale medical applications where run time requirements would be quite severe. For example, in the CAD applications shown in our experiments, the run-time of the testing phase usually has to be less than a second in order that the end user's (radiologist's) time would not be wasted.

Our algorithm is also related to the *multiple instance learning* (MIL) problem, where one is given bags (batches) of samples; class labels are provided only for the bags, not for the individual samples. A bag is labeled positive if we know that at least one sample from it is positive, and a negative bag is known to not contain any positive sample. In this manner, the MIL problem also encodes a form of prior knowledge about correlations between the labels of the training instances.

There are two differences between our algorithm and MIL. First, we want to classify each instance (candidate) in our algorithm; unlike MIL, we are not only trying to label a bag of related instances. Second, unlike the MIL problem which treats all the instances in a bag as equally related to each other, we account for more fine grained differences in the level of correlation between samples (via the covariance matrix Σ).

1.2 Organization of the Paper

Section 2 presents the clinical motivation behind our work and describes the training and testing data that are used in these applications. In Section 3, we build a probabilistic model for batch classification of samples. Although dramatically faster than CRFs and their other cousins, the probabilistic algorithm is still too slow to be practical on several CAD problems, hence we propose another faster algorithm in Section 4. Unlike the previous methods such as CRF and MMM, both the proposed algorithms are easy to implement for arbitrary correlation relationships between samples, and further we are able to run these fast enough to be viable in commercial CAD products. In Section 5, we provide experimental evidence from three different CAD problems to show that the proposed algorithm is more accurate in terms of the metrics appropriate to CAD as compared to a naive SVM which is routinely used for these problems as the state-of-the-art in the current literature and commercial products. We conclude with a review of our contributions in Section 6.

Throughout this paper, we will utilize the following notations. The notation $A \in R^{m\times n}$ will signify a real $m \times n$ matrix. For such a matrix, A' will denote the transpose of A and A_i will denote the i-th row of A. All vectors will be column vectors. A vector of ones in a real space of arbitrary dimension will be denoted by e. Thus, for $e \in R^m$ and $y \in R^m$, $e'y$ is the sum of the components of y. A vector of zeros in a real space of arbitrary dimension will be denoted by 0.

2 Data in the Medical Domain

Data collection process for training CAD classifiers. Medical images (such as, CT scans, MRI, X-ray etc.) are collected from the archives of hospitals that routinely screen patients for cancer. Depending upon the disease, ground truth is determined for each patient based either on a more expensive, potentially invasive test (e.g., biopsy of breast lesions, or colonoscopy for colon polyps), or via consensus opinion of a panel of expert radiologists for organs when a definitive test (lung biopsy) is deemed too dangerous. In all cases, expert radiologist's opinion is also required to mark the location, size, and extent of all "positive" regions within the images. A CAD system is then designed from the database of training images. Considerable human intervention and domain knowledge engineering is employed on the first two stages of a CAD system: (a) candidate generation: identify all potentially suspicious regions in a candidate generation stage with very high sensitivity, and (b) feature-extraction: to describe each such region quantitatively using a set of medically relevant features. For example, quantitative measurements based on texture, shape, intensity, contrast and other such characteristics may be used to characterize any region of interest (ROI). Finally, the candidate ROIs are assigned class labels based upon the overlap or spatial proximity to any radiologist-marked (diseased) region.

From the above description it is clear that the samples (candidates) are naturally collected in batches. While there are no correlations between the candidate ROIs in different images, the labels of all the regions identified from the same patient's medical images are likely to be at least somewhat correlated. This is true both because metastasis is an important possibility in cancer, and because the patient's general health and preparation for imaging are important factors in diagnostic classification (e.g., how thoroughly was the cleaning of stool undertaken before a colonoscopy). Further, in order to identify suspicious regions with high sensitivity, most candidate generation algorithms tend to produce several candidates that are spatially close to each other, often referring to the same underlying structure in the image. Since they often refer to regions that are physically adjacent in an image, both features and class labels for these candidates are also highly correlated.

Shortcomings in standard classification algorithms. Most of the classification algorithms such as *neural networks* and *support vector machines* (SVM) assume that the training samples or instances are drawn identically and *independently* from an underlying distribution. However, as mentioned in the introduction and in the previous subsection, due to spatial adjacency of the regions identified by a candidate generator, both the features and the class labels of several adjacent candidates are highly correlated. This is true both in the training and testing data. The proposed batch-classification algorithms account for these correlations explicitly.

3 A Probabilistic Batch Classification Model

Let $x_i^j \in R^n$ represent the n features for the i^{th} candidate in the j^{th} image, and let $w \in R^n$ be the parameters of some hyperplane classifier. Traditionally, linear classifiers label samples one at a time (i.e., independently) based on:

$$z_i^j = w'x_i^j = (x_i^j)'w \ , z \in R^1 \tag{1}$$

For example, in logistic regression, the posterior probability of the sample x_i^j belonging to class $+1$ is obtained using the sigmoid function $P(y_i^j = 1|x_i^j) = \frac{1}{1+\exp(-w'x_i^j)}$.

By contrast, in our model, we claim z_i^j is only a noisy observation of the underlying, unobserved variable $u_i^j \in R^1$ that actually influences classification (as opposed to the traditional classification approach, where classification directly depends on z_i^j).

We have an a-priori guess or intuition about u_i^j even before we observe any x_i^j (therefore before z_i^j), which is purely based on the proximity of the spatial locations of candidates in the j^{th} image. Indeed this spatial adjacency is what induces the correlation in the predictions for the labels; we model this as a Gaussian prior on u_i^j.

$$P(u^j \in R^{n_j}) = N(u^j|0, \Sigma^j) \tag{2}$$

where n_j is the number of the candidates in the j^{th} image, and the covariance matrix Σ^j (which encodes the spatial proximity based correlations) can be defined in terms of S, the matrix of Euclidean distances between candidates inside a medical image (from a patient) as $\Sigma^j = \exp(-\alpha S)$.

Having defined a prior, next we define the likelihood as follows:

$$P(z_i^j|u_i^j) = N(z_i^j|u_i^j, \sigma^2) \tag{3}$$

After observing x_i^j and therefore z_i^j, we can modify our prior intuition about u^j in (2), based on our observations from (3) to obtain the Bayesian posterior:

$$P(u^j|z^j) = N\left(u^j|({\Sigma^j}^{-1}\sigma^2 + I)^{-1}z^j; ({\Sigma^j}^{-1} + \tfrac{1}{\sigma^2}I)^{-1}\right) \tag{4}$$

The class-membership prediction for the i^{th} candidate in the j^{th} image is controlled exclusively by u_i^j. The prediction probability for class labels, y^j is then determined as:

$$P(y^j = 1|B^j, w, \alpha, \sigma^2) = 1/\left(1 + \exp\left(-[{\Sigma^j}^{-1}\sigma^2 + I]^{-1}[B^j w]\right)\right). \tag{5}$$

Where $B^j \in R^{m_j \times n}$ represents the m_j training points that belong to the j^{th} batch. Note however, that this approach to batchwise prediction is potentially slow due to the matrix inversion, if the test data arrives in large batches.

3.1 Learning in This Model

For batch-wise prediction using (5), w, α and σ^2 can be learned from a set of N training images via *maximum-a-posteriori* (MAP) estimation as follows:

$$[\hat{w}, \hat{\alpha}, \hat{\sigma^2}] = \arg\max_{w,\alpha,\sigma^2} P(w) \textstyle\prod_{j=1}^{N} P(y^j|B^j, w, \alpha, \sigma^2) \tag{6}$$

where, $P(y^j|B^j, w, \alpha, \sigma^2)$ is defined as in (5) and $P(w)$ may be assumed to be Gaussian $N(w|0, \lambda)$. The regularization parameter λ is typically chosen by cross-validation.

3.2 Intuition About Batch Classification

Equations (4) and (5) imply that $\mathbb{E}[u^j|z^j] = ({\Sigma^j}^{-1}\sigma^2 + I)^{-1}z^j$. In other words, the class membership prediction for any single sample is a weighted average of the noisy prediction quantity z^j (distance to the hyperplane), where the weighting coefficients depend on the pairwise Euclidean distances between samples. Hence, the intuition presented above is that we predict the classes for all the n_j candidates in the j^{th} image together, as a function of the features for all the candidates in the batch

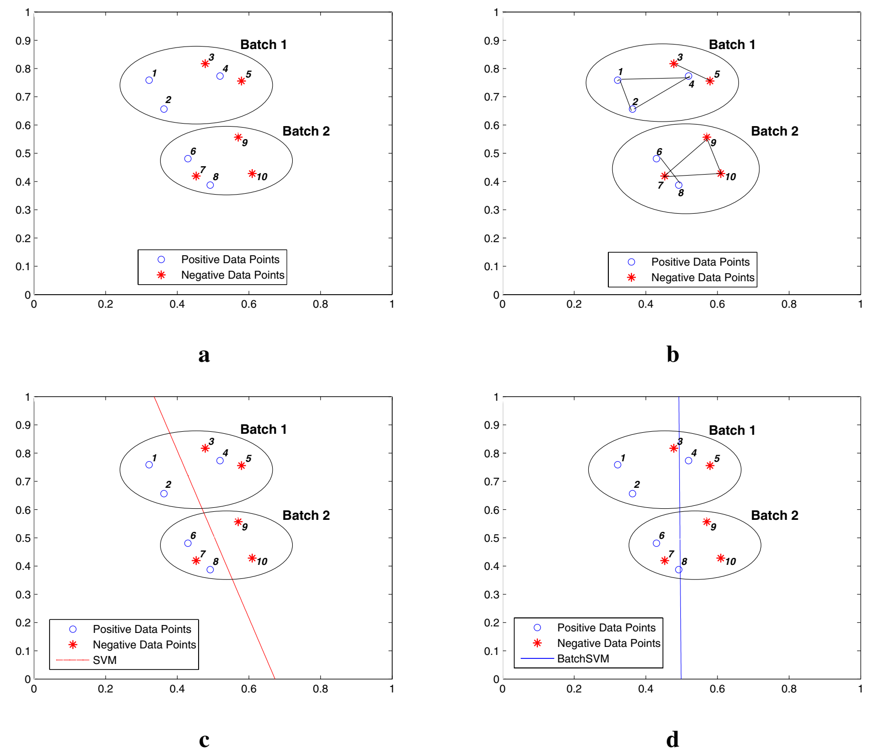


Fig. 2. An illustrative example for batch learning. **a**) Training data points are displayed in batches. **b**) Relations within training points are displayed as a linked graph. **c**) Classifier produced by SVM. **d**) Pre-classifier produced by BatchSVM. Unlike standard SVMs, the hyperplane, $f(x)$, produced by BatchSVM (preclassifier) is not the decision function. Instead, the decision of each test sample x_i, is based on a weighted average of the $f(x)$ values for the points linked to x_i.

(here a batch corresponds to an image from a patient). In every test image, *each* of the candidates is classified using the features from *all* the samples in the image.

4 A Mathematical Programming Approach

Motivated by equations (5) and (6), we now re-formulate the problem of learning for batch-wise prediction as an SVM-like mathematical program.

In a standard SVM a hyperplane classifier, $f(x) = x'w - \gamma$ is learned from the training instances individually, ignoring the correlations among them. Consider the problem of classifying m points in the n-dimensional real space R^n, represented by the $m \times n$ matrix A, according to class membership of each point x_i (ith row of A) in the classes $A+$, $A-$ as specified by a given $m \times m$ diagonal matrix D with $+1$ or -1 along its diagonal, this is, $D = diag(y)$. The standard 1-norm support vector machine with a linear kernel [8,2] is given by the following linear program with parameter $\nu > 0$:

$$\begin{aligned}\min_{(w,\gamma,\xi,v)\in R^{n+1+m+n}} \quad & \nu e'\xi + e'v \\ \text{s.t.} D(Aw - e\gamma) + \xi \geq e & \\ v \geq w \geq -v & \\ \xi \geq 0 &\end{aligned} \tag{7}$$

where, ν is the cost parameter and at a solution, $v = |w|$ is the absolute value of w.

While $A \in R^{m\times n}$ represents the entire traning data, $B^j \in R^{m_j \times n}$ represents the m_j training points that belong to the j^{th} batch and the labels of these training points are represented by the $m_j \times m_j$ diagonal matrix $D^j = diag(y^j)$ with positive or negative ones along its diagonal. Then, the standard SVM set of constraints: $D(Aw-e\gamma)+\xi \geq e$ can be modified in order to take into account the correlations among samples in the same batches, using the idea in equation 5 as:

$$D^j \left[\left({\Sigma^j}^{-1}\sigma^2 + I \right)^{-1} (B^j w - e\gamma) \right] + \xi^j \geq e, \quad \text{for } j = 1, \ldots, k \tag{8}$$

In a naive implementation, for each batch j, the probabilistic method requires calculating two matrix inversions to compute $\left(\Sigma^{-1}\sigma^2 + I\right)^{-1}$. Hence, training and testing using this method can be time consuming for large batch sizes. In order to avoid this problem while retaining the intuition presented in subsection 3.2, we modify equation (8). In particular, we replace the expression $\left(\Sigma^{-1}\sigma^2 + I\right)^{-1}$ by a much simpler expression: $(\Sigma\theta + I)$. As a result, the correlation among samples belonging to the same batch can be enforced by replacing the standard set of SVM constraints by:

$$D^j \left[\left(\theta\Sigma^j + I \right) (B^j w - e\gamma) \right] + \xi^j \geq e, \quad \text{for } j = 1, \ldots, k \tag{9}$$

As in equation (8), the class membership prediction for any single sample in batch j is a weighted average of the batch members prediction vector $B^j w$, and again the weighting coefficients depend on the pairwise Euclidean distances between samples. Using this constraint in the SVM equations (7), we obtain the optimization problem for learning BatchSVM with parameters ν and θ:

$$\begin{aligned}\min_{(w,\gamma,\xi,v)\in R^{n+1+m+n}} \quad & \nu e'\xi + e'v \\ \text{s.t.} D^j \left[\left(\theta\Sigma^j + I \right) (B^j w - e\gamma) \right] + \xi^j \geq e, \quad & \text{for } j = 1, \ldots, k \\ v \geq w \geq -v & \\ \xi \geq 0 &\end{aligned} \tag{10}$$

Unlike standard SVMs, the hyperplane ($f(x) = w^{'}x - \gamma$) produced by BatchSVM is *not* the final decision function. We refer to $f(x)$ as a pre-classifier that will be used in the next stage to make the final decision on a batch of instances. While testing an arbitrary datapoint x_i^j in batch B^j, the BatchSVM algorithm accounts for the pre-classifier prediction $w'x_p^j$ for every member in the batch. The final prediction $\hat{f}(x_i^j)$ is given by:

$$sign(\hat{f}(x_i^j)) = sign(w^{'}x_i^j - \gamma + \theta\Sigma_i^j \left[B^j w - \gamma \right]) \tag{11}$$

Table 1. Outputs of the classifier produced by SVM, pre-classifier and the final classifier produced by BatchSVM. The outputs are calculated for the data points presented in Figure 2. The first column of the table indicates the order of the data points as they are presented in Figure 2a and the second column specifies the corresponding labels. Misclassified points are displayed in bold. Notice that the combination of the pre-classifier outputs at the final stage corrects the mistakes.

Point	Batch	Label	SVM	Pre-classifier	Final classifier
1	1	+	0.2826	0.1723	0.1918
2	1	+	0.2621	0.1315	0.2122
3	1	-	-0.2398	**0.0153**	-0.0781
4	1	+	**-0.3188**	**-0.0259**	0.2909
5	1	-	-0.4787	-0.0857	-0.0276
6	2	+	0.2397	0.0659	0.0372
7	2	-	**0.2329**	**0.0432**	-0.0888
8	2	+	0.1490	0.0042	0.0680
9	2	-	-0.2525	-0.0752	-0.1079
10	2	-	-0.2399	-0.1135	-0.1671

Consider the two dimensional example in Figure 2, showing batches of training points. The data points that belong to the same batch are indicated by the elliptical boundaries in the figure. Figure 2b displays the correlations amongst the training points given in Figure 2a using an edge. In Figure 2c, the hyperplane $f_{svm}(x)$ is the final decision function for standard SVM and gives the results displayed in Table 1, where we observe that the fourth and the seventh instances are misclassified. In Figure 2d, the pre-classifier produced by BatchSVM, $f_{batch}(x)$ gives the results displayed in the fifth column of Table 1 for the training data. If this pre-classifier were to be considered as the decision function, then three training points would be misclassified. However, during batch-testing (eq 11), the predictions of those points are corrected as seen in the sixth column of Table 1.

Kernelized nonlinear algorithm. To obtain a more general nonlinear algorithm, we can "kernelize" equations (10,11) by making a transformation of the variable w as: $w = A'v$, where v can be interpreted as an arbitrary variable in $\Re^m$. This transformation can be motivated by duality theory [5]. Employing this idea will result in a term $B^jA'v$ instead of B^jw in our formulations. If we now replace the linear kernels, B^jA', by more general kernels, $K(B^j, A')$, we obtain a "kernelized" version of equations (10,11).

5 Experiments

5.1 The Similarity Function

As mentioned earlier, the matrix Σ^j represents the level of correlation between all pairs of candidates from a batch (an image in our case) and it is a function of the pairwise-similarity between them. In CAD applications, the covariance matrix Σ^j can be defined in terms of the matrix of Euclidean distances between candidates inside a medical image. Let r_p and r_q represent the coordinates of two candidates, B^j_p and B^j_q on the j^{th}

image. For our experiments, we used the Euclidean distance between r_p and r_q to define the pairwise-similarity, $s(p,q)$, between B_p^j and B_q^j as: $s(p,q) = \exp\left(-\alpha\|r_p - r_q\|^2\right)$.

Experimentally, we found it useful to discretize the continuous similarity function, $s(p,q)$ to the binary similarity function, $s^*(p,q)$ by applying a threshold as following:

$$s^*(p,q) = \begin{cases} 0 \text{ , } s(p,q) < e^{-4} \\ 1 \text{ , } s(p,q) \geq e^{-4} \end{cases} \tag{12}$$

In all experiments, we set the threshold at e^{-4} to provide us with a similarity of one if the neighbor is at a 95% confidence level of belonging to the same density as the candidate assuming that the neighborhood is a Gaussian distribution with mean equal to candidate and variance $\varsigma^2 = \frac{1}{\alpha}$. Each element of Σ is given by: $\Sigma_{pq} = s^*(p,q)$.

5.2 Comparisons

In this section, we compare three techniques: regular SVM, probabilistic batch learning ($BatchProb$), and BatchSVM. Receiver Operating Characteristic (ROC) plots are used to study the classification accuracy of these techniques on three CAD applications for detecting pulmonary embolism, colon cancer, and lung cancer. In clinical practice, CAD systems are evaluated on the basis of a somewhat domain-specific metric: to maximize the fraction of *positives* that are correctly identified by the system while displaying at most a clinically acceptable number of false-marks per image. We report this domain-specific metric in an ROC plot, where the y-axis is a measure of sensitivity and the x-axis is the number of false-marks per patient (in our case, per image is also per patient). Sensitivity is the number of patients diagnosed as having the disease divided by the number of patients that has the disease. High sensitivity and low false-marks are desired. All our parameters in these experiments are tuned by 10-fold patient cross-validation on the training data (i.e., the training data is split into ten folds). During cross-validation, a range of parameters $(\theta, \sigma, \varsigma)$ were evaluated for the proposed methods: for θ in $BatchSVM$ and σ in $BatchProb$, we considered $-1, -0.9, ..., 0.9, 1$ and for ς that is necessary for Σ matrix, we used a logarithmically spaced range from 10^{-3} through 10^{1}. All classification algorithms are trained on the training dataset and evaluated on the sequestered (held-out) test set.

5.3 Data Sources and Domain Description

Example: Pulmonary Embolism. Pulmonary embolism (PE), a potentially life-threatening condition, is a result of underlying venous thromboembolic disease. An early and accurate diagnosis is the key to survival. Computed tomography angiography (CTA) has emerged as an accurate diagnostic tool for PE. There are hundreds of CT slices in each CTA study, thus manual reading is laborious, time consuming and complicated by various PE look-alikes (false positives). Several CAD systems are developed to assist radiologists in this process by helping them detect and characterize emboli in an accurate, efficient and reproducible way [6], [9]. We have collected 72 cases with 242 PEs marked by expert chest radiologists at four different institutions (two North American sites and two European sites). For our experiments, they were

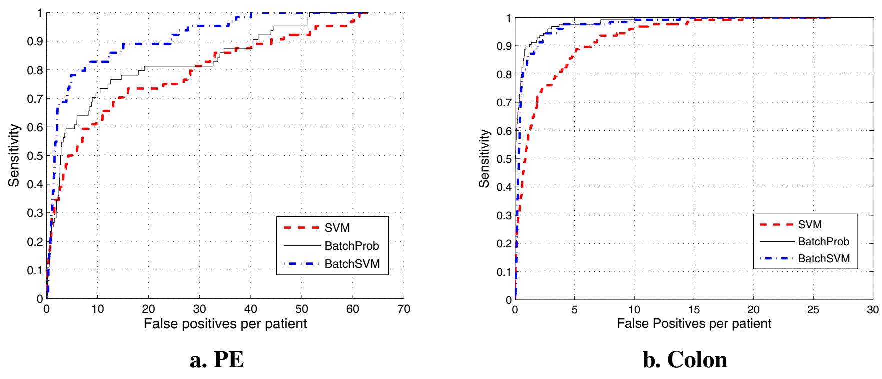


Fig. 3. SVM, BatchProb and BatchSVM ROC curves comparisons for (a) the PE data and (b) the Colon Cancer data

randomly divided into two sets: a training and a testing set. The training set was used to train and validate the classifiers and consists of 48 cases with 173 PEs and a total of 3655 candidates. The testing set consists of 24 cases with 69 true PEs out of a total of 1857 candidates. This set was only used to evaluate the performance of the final system. A combined total of 70 features were extracted for each candidate.

Example: Colon Cancer Detection. Colorectal cancer is the third most common cancer in both men and women. It is estimated that in 2004, nearly 147, 000 cases of colon and rectal cancer will be diagnosed in the US, and more than 56, 730 people would die from colon cancer [3]. In over 90% of the colon cancer cases that progressed rapidly is from local (polyp adenomas) to advanced stages (colorectal cancer), which has very poor survival rates. However, identifying (and removing) lesions (polyp) when still in a local stage of the disease, has very high survival rates, thus illustrating the critical need for early diagnosis. Most polyps in the training data are inherently represented by multiple candidates. The database of high-resolution CT images used in this study were obtained from seven different sites across US, Europe and Asia. The 188 patients were randomly partitioned into a training and a test set. The training set consists of 65 cases containing 127 volumes. Fifty polyps were identified in this set out of a total of 6748 candidates. The testing set consists of 123 cases containing 237 volumes. There are 103 polyps in this set from a total of 12984 candidates. A total of 75 features were extracted for each candidate.

Example: Lung Cancer. LungCAD is a computer aided detection system for detecting potentially cancerous pulmonary nodules from thin slice multi-detector computed tomography (CT) scans. The final output of LungCAD is provided by a classifier that classifies a set of candidates as positive or negative. This is a very hard classification problem: most patient lung CTs contain a few thousand structures (candidates), and only a few ($\leq$ 5 on average) of which are potential nodules that should be identified as positive by LungCAD, all within the run-time requirements of completing the classification on-line during the time the physician completes their manual review. The

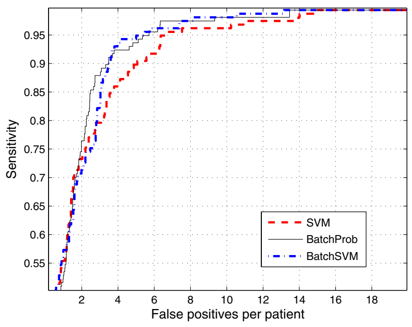


Fig. 4. SVM, BatchProb and BatchSVM ROC curves comparisons for the Lung Cancer data

training set consists of 60 patients. The number of candidates labeled as nodules in the training set are 157 and the total number of candidates is 9987. The testing set consists of 26 patients. In this testing set, there are 79 candidates labeled as nodules out of 6159 generated candidates. The number of features extracted for this dataset were 15.

5.4 Results

Figures 3a, 3b, and 4 show the ROC curves for pulmonary embolism, colon cancer, and lung cancer data respectively. In our medical applications high-sensitivity is critical as early detection of lung and colon cancer is believed to greatly improve the chances of successful treatment [1]. Furthermore, high specificity is also critical, as a large number of false positives will vastly increase physician load and lead (ultimately) to loss of physician confidence.

In Figure 3a, corresponding to the comparison of the ROC curves on the PE dataset, we observe that standard SVM can only achieve 53% sensitivity for six false positives. However, BatchSVM achieves 80% with a remarkable improvement (27%). BatchProb also outperforms SVM with a 64% sensitivity. As seen from the figure, the two proposed methods are substantially more accurate than standard SVMs at any specificity level.

Colon cancer data is a relatively easier data set than pulmonary embolism since standard SVM can achieve 54.5% sensitivity at one false positive level as illustrated in Figure 3b. However, BatchSVM improved SVM's performance to 84% sensitivity for the same number of false positives. Note that BatchProb improved the sensitivity further, giving 89.6% for the same specifity. In one to ten false positives region which constitutes the region of interest in our applications, our proposed methods outperform standard SVM significantly.

Although SVM is very accurate for lung cancer application, Figure 4 shows that BatchProb and BatchSVM could still improve SVM's performance further. BatchProb method is superior to the other methods at two and three false positives per image. Both BatchProb and BatchSVM outperform SVM in the 2-6 false positives per image region, which is the region of interest for commercial clinical lung CAD systems. All three of the methods are comparable at other specificity levels.

6 Conclusions

Two related algorithms have been proposed for classifying batches of correlated data samples. Although primarily motivated by real-life CAD applications, the problem occurs commonly in many situations; our algorithms are sufficiently general to be applied in other contexts. Experimental results indicate that the proposed method can substantially improve the diagnosis of (a) early stage cancer in the Lung & Colon, and (b) pulmonary embolisms (which may result in strokes). With the increasing adoption of these systems in routine clinical practice, these experimental results demonstrate the potential of our methods to impact a large cross-section of the population.

Acknowledgments

This research is supported by the Computer Aided Diagnosis Group, Siemens Medical Solutions, USA and NSF CAREER IIS-0347532.

References

1. L. Bogoni, P. Cathier, M. Dundar, A. Jerebko, S. Lakare, J. Liang, S. Periaswamy, M. Baker, and M. Macari. CAD for colonography: A tool to address a growing need. *British Journal of Radiology*, 78:57–62, 2005.
2. P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *Proc. 15th International Conf. on Machine Learning*, pages 82–90, San Francisco, CA, 1998. Morgan Kaufmann.
3. D. Jemal, R. Tiwari, T. Murray, A. Ghafoor, A. Saumuels, E. Ward, E. Feuer, and M. Thun. Cancer statistics, 2004.
4. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289, San Francisco, CA, 2001. Morgan Kaufmann.
5. O. L. Mangasarian. Generalized support vector machines. In *Advances in Large Margin Classifiers*, pages 135–146, 2000.
6. M. Quist, H. Bouma, C. V. Kuijk, O. V. Delden, and F. Gerritsen. Computer aided detection of pulmonary embolism on multi-detector CT, 2004.
7. B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.
8. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.
9. C. Zhou, L. M. Hadjiiski, B. Sahiner, H.-P. Chan, S. Patel, P. Cascade, E. A. Kazerooni, and J. Wei. Computerized detection of pulmonary embolism in 3D computed tomographic (CT) images: vessel tracking and segmentation techniques. In *Medical Imaging 2003: Image Processing. Proceedings of the SPIE, Volume 5032, pp. 1613-1620 (2003).*, pages 1613–1620, May 2003.

Improving the Ranking Performance of Decision Trees

Bin Wang and Harry Zhang

Faculty of Computer Science, University of New Brunswick, Canada
{bin.wang, hzhang}@unb.ca

Abstract. An accurate ranking of instances based on their class probabilities, which is measured by AUC (area under the Receiver Operating Characteristics curve), is desired in many applications. In a traditional decision tree, two obstacles prevent it from yielding accurate rankings: one is that the sample size on a leaf is small, and the other is that the instances falling into the same leaf are assigned to the same class probability. In this paper, we propose two techniques to address these two issues. First, we use the statistical technique *shrinkage* which estimates the class probability of a test instance by using a linear interpolation of the local class probabilities on each node along the path from leaf to root. An efficient algorithm is also brought forward to learn the interpolating weights. Second, we introduce an instance-based method, the weighted probability estimation (*WPE*), to generate distinct local probability estimates for the test instances falling into the same leaf. The key idea is to assign different weights to training instances based on their similarities to the test instance in probability estimation. Furthermore, we combine *shrinkage* and *WPE* together to compensate for the defects of each. Our experiments show that both *shrinkage* and *WPE* improve the ranking performance of decision trees, and that their combination works even better. The experiments also indicate that various decision tree algorithms with the combination of *shrinkage* and *WPE* significantly outperform the original ones and other state-of-the-art techniques proposed to enhance the ranking performance of decision trees.

Keywords: Decision Tree, Class Probability, Ranking, AUC, *Shrinkage*, *WPE*

1 Introduction

Decision trees have been regarded as one of the most popular models in the fields of machine learning and data mining. Traditionally, accuracy is often used to evaluate the classification performance of decision trees. However, it is not sufficient to merely classify an instance into the most possible class in many applications. A ranking of instances based on the class probability $P(c|e)$, the probability of an instance e in the class c, is more desirable. For example, a credit card company can consider the top X% in a ranking of applicants, who are most likely to belong to the profitable class. In this paper, we use AUC (Area

Under the Receiver Operating Characteristics Curve) to evaluate the ranking performance of decision trees, which has received considerable attention as a measure of ranking [14,8,6].

Accurate probability estimation certainly leads to accurate ranking which is based on the class probabilities. Unfortunately, decision trees, such as C4.5 [15], have been observed to produce poor probability estimates which result in the poor ranking performance [11,3,13]. In a decision tree, the class probability $P(c|e)$ is estimated by the fraction of instances of class c on the leaf which e falls into. It causes two problems [17]. One is the high bias: decision tree growing methods try to make leaves pure, so the probability estimates on leaves are shifted towards zero or one; the other is the high variance: the training instances on a leaf are often not enough to provide reliable probability estimates. Besides, decision tree algorithms often assign the same class probability to the instances falling into the same leaf. The ranks of the instances with equal probability are generated randomly, and thus the AUC score tends to decrease.

In this paper, we introduce *shrinkage* and the weighted probability estimation (*WPE*) to solve the above problems. The probability estimate with *shrinkage* for a test instance is decided by the linear interpolation of the local probability estimates on each node along the path from leaf to root, instead of merely being decided by the leaf. An efficient algorithm is proposed to determine the interpolating weights. *WPE* is an instance-based method, which assigns the distinct class probabilities to the instances falling into the same node, and thus leads to better ranking performance. However, there are still some flaws in *shrinkage* and *WPE*. We combine these two techniques together to compensate for the defects of each. *Shrinkage*, *WPE* and their combination are applicable to any decision tree algorithms without changing the tree-building process and tree structure. We design empirical experiments to verify that both *shrinkage* and *WPE* can improve the ranking performance of traditional decision tree algorithms, such as C4.5 [15] and C4.4 [12], in terms of AUC. We also show that the combination of these two techniques is even stronger than either single one, and according to AUC outperforms other techniques such as m-Branch [5], bagging [12] and Ling's algorithm [9], which aim to improve the probability-based ranking in decision trees.

The rest of the paper is organized as follows. In Section 2, we discuss the related work on improving the probability-based ranking performance of decision trees. In Section 3, we describe *shrinkage* and the algorithm for training the interpolating weights. In Section 4, we illuminate the process of *WPE*. In Section 5, we show how to combine *shrinkage* and *WPE* together. In Section 6, the experiments and results are presented and analyzed. Finally, we summarize our work and bring forward the future research in Section 7.

2 Related Work

As the foregoing analysis has shown, the ranking performance of C4.5 is poor (i.e., low AUC score). Provost and Domingos [12] utilize two techniques to improve the AUC of C4.5. The first is to turn off pruning and the second is to use *Laplace* correction. They call the resulting algorithm C4.4. However, turning off pruning results in a large tree so that the number of training instances on each leaf tends to be small. Then the corresponding probability estimation could be unreliable even when using *Laplace* correction. Moreover, the same probability estimate is assigned on the same leaf in C4.4. Bagging is also used to improve the AUC of decision trees [2,13], but the results produced by bagging are not comprehensible.

Some researchers have noticed that the information used for estimating the probability of an instance should not be limited to the leaf which the instance falls into. Ling and Yan [9] present an algorithm to average probability estimates from all leaves, instead of a single leaf. The contribution of each leaf is determined by the deviation in attribute values from root to leaf. But the deviation they described has only been reflected by a "confusion factor" which can be regarded as the probability of errors that alters the attribute values. Although it produces distinct probability estimates for the instances on the same leaf, setting up good "confusion factors" could still be an issue. Moreover, the algorithm should go through the whole tree to calculate the contribution of each leaf. Consequently, the complexity of this algorithm tends to be fairly high.

Ferri et al. [5] introduce the m-Branch method to smooth the probability estimates on leaves with the history information along the paths. M-Branch is a recursive process in which the probability estimate on the parent node is put into the probability estimate on the child node. The parameters of m-Branch are adjusted by the height of a node and the cardinality (the number of instances associated with a node). Although they notice that the information from other nodes should be utilized, they still assign the same class probability to the instances on the same leaf.

Zadrozny and Elkan [17] suggest a method called *curtailment*, to improve the probability estimation of decision trees. In *curtailment*, if a leaf contains few training instances and can not induce reliable probability estimates, probability estimation can be raised to an ancestor node of this leaf, in which there are enough training instances. *Curtailment* blurs the distinction between internal nodes and leaves, because a node may serve as an internal node which owns child nodes, or serve as a leaf which assigns the same probability estimate to instances. *Curtailment* is reminiscent of the methods proposed by Bahl et al. [1] and Buntine [4] that calculate a weighted average of training frequencies at nodes along the path from root to leaf. However, they do not propose an effective algorithm to learn the weights. Hastie and Pregibon [7] provide a shrinking process called *recursive shrinking* to smooth the pruning in decision trees. The shrinking process is parameterized by a scalar θ which must be specified based on the data.

Shrinkage has been brought into text classification [10], in which the word probability estimates are improved by *shrinkage* in a hierarchical structure. In this method, the final class probabilities are estimated by naive Bayes. Besides, the weights of *shrinkage* are determined by an EM algorithm. The EM algorithm needs an iterative procedure that converges usually after many iterations. Thus, it is quite time-consuming.

3 Shrinkage

Figure 1 shows a sample decision tree, which has five internal nodes $N_1,\ldots,N_5$ and six leaves $N_6,\ldots,N_{11}$, associated with the subsets of training instances D_1, ..., D_{11}.

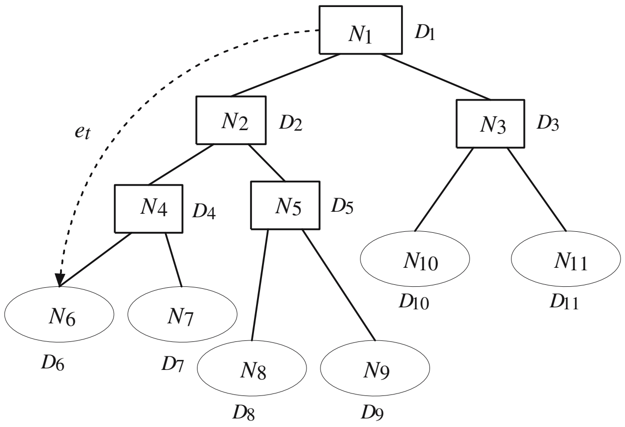


Fig. 1. A sample of decision tree

Assume that the test instance e_t falls into leaf N_6 passing internal nodes N_1, N_2 and N_4. In the traditional decision tree algorithm, the class probability $P(c_j|e_t)$ is estimated by the fraction of training instances in class c_j on leaf N_6. The *shrinkage* estimate of $P(c_j|e_t)$ is a linear combination of the local class probability estimates on the nodes N_1, N_2, N_4 and N_6. Given class c_j, the local class probability of e_t on the *ith* node is estimated as follows.

$$P^i(c_j|e_t) = \frac{n_j + 1/|C|}{|D_i| + 1}, \qquad (1)$$

where n_j is the number of training instances belonging to class c_j, $|C|$ is the number of classes, and $|D_i|$ is the number of training instances on the *ith* node. In Equation 1, Laplace correction is used to smooth the probability estimate towards the uniform distribution of class labels.

Although the root N_1 contains all the training instances, it probably has few instances whose class labels are rare, which may result in unreliable probability estimation. Therefore, we extend the tree by adding a uniform node N_0 beyond the root [10], on which the uniform distribution of instances is adopted. In order to keep the consistency of expression, we define the class probability of e_t given class c_j on the uniform node N_0 as $P^0(c_j|e_t) = \frac{1}{|D_1|}$, where $|D_1|$ is the number of training instance on root N_1. The *shrinkage* estimate of $P(c_j|e_t)$ is shown as follows.

$$P(c_j|e_t) = w_j^0 P^0(c_j|e_t) + w_j^1 P^1(c_j|e_t) + w_j^2 P^2(c_j|e_t) + w_j^3 P^3(c_j|e_t) + w_j^4 P^4(c_j|e_t), \tag{2}$$

where $P^1(c_j|e_t)$, $P^2(c_j|e_t)$, $P^3(c_j|e_t)$, $P^4(c_j|e_t)$ are the local probability estimates on N_1, N_2, N_4, N_6 respectively, w_j^0, w_j^1, w_j^2, w_j^3, w_j^4 are the interpolating weights for class c_j assigned to the corresponding nodes, in which $\sum_{m=0}^{4} w_j^m = 1$. Figure 2 shows the path, probabilities and related weights. Equation 2 can be extended to deal with the general case which has k nodes on the path.

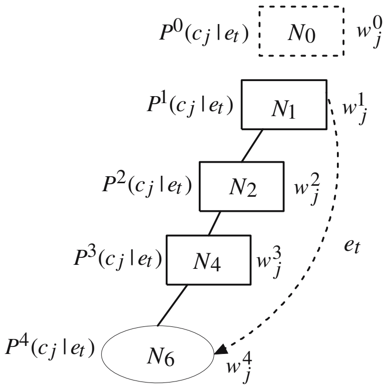


Fig. 2. The path of e_t in the decision tree. Uniform node is added above N_1. The local probabilities are estimated on each node, and the interpolating weights are assigned to each node.

Shrinkage represents a tradeoff between the specificity and generality. At a leaf, since the probability estimates are yielded based on the training instances that come through a series of partitions on the internal nodes, they are more specific but less general than the ones on the ancestors of the leaf. At the root, the estimates are more general because all the training instances are included, but they are less specific than the estimates on the descendants of the root.

A key problem to apply *shrinkage* is to determine the interpolating weights effectively and efficiently. Assume that N_0, N_1, …, N_k is a path, where N_0 is the uniform node, N_1 is the root, and N_k is the leaf. Given class label c_j, let β_j^0, β_j^1, …, β_j^k be the influence degree of node N_i to class c_j, and w_j^0, w_j^1, …, w_j^k be the interpolating weights. We use the following algorithm to determine the interpolating weights.

Algorithm DIW (*path*, D, c_j)
Input: *path* is a sequence of nodes N_0, N_1, …, N_k, D is the training set, and c_j is a class label.
Ouput: A set of interpolating weights for c_j for all nodes N_0, N_1, …, N_k.

Step 1: Initialize each weight w_j^i as $\frac{1}{k+1}$ so that $\sum w_j^i = 1$, and initialize each degree β_j^i to zero. Set each training instance on leaf unmarked.
Step 2: Choose an unmarked training instance x on leaf N_k, remove x from all the training subsets $D_1, \ldots, D_k$ on each node along the path.
Step 3: Set $P^0(c_j|x) = 1/|D_1|$, where $|D_1|$ is the number of training instances on root N_1. From N_1 to N_k, estimate the local class probabilities $P^i(c_j|x)(i = 1, \ldots, k)$ using Equation 1.
Step 4: For each node, update its degree as follows:

$$\beta_j^i = \beta_j^i + \frac{w_j^i P^i(c_j|x)}{\sum_{m=0}^{k} w_j^m P^m(c_j|x)}, i = 0, \ldots, k. \tag{3}$$

Step 5: Mark instance x and put x back to each training subset. If there is an un-marked training instance on N_k, go back to **Step 2**.
Step 6: Compute w_j^i by normalizing the set of degrees $\{\beta_j^0, \beta_j^1, \ldots, \beta_j^k\}$ as follows:

$$w_j^i = \frac{\beta_j^i}{\sum_{m=0}^{k} \beta_j^m}, i = 0, \ldots, k. \tag{4}$$

Return: $\{w_j^0, w_j^1, \ldots, w_j^k\}$.

Note that on a node N_i, the local class probabilities $P^i(c_j|x)(x \in D_k)$ have the same estimate in **Step 3**, since the fraction of training instance in class c_j on N_i is used in Equation 1. Because of this, Algorithm **DIW** is not able to be adapted to a multiple-iteration algorithm. The interpolating weights are returned only after one iteration (go through each instance on the leaf once).

When we apply *shrinkage* to a decision tree algorithm, a decision tree is built by that algorithm first. Then the Algorithm **DIW** is applied to each path to set up the interpolating weights for each node and each class label. Given a test instance e_t, it is sorted down to a leaf, and then its class probability $P(c_j|e_t)$ is computed using Equation 2.

4 Weighted Probability Estimation

As mentioned before, a major issue for decision trees is that all the instances falling into the same leaf will have the same probability estimate, which is an obstacle to yielding accurate ranking. We notice that the instance-based method can generate the distinct and local estimates.

We introduce *WPE*, an instance-based method, to estimate the class probabilities on the leaves. Given a test instance e_t which consists of a set of attribute values and a class label, e_t falls into a leaf L. We define the similarity between e_t and a training instance e_r on L as follows.

$$sim(e_t, e_r) = \sum_{i=1}^{n} equ(A_i(e_t), A_i(e_r)), \tag{5}$$

where n is the number of attributes, $equ(a,b)$ is a boolean function whose value is either 1 ($a=b$) or 0 ($a \neq b$), and $A_i(e)$ is the ith attribute value of e. In *WPE*, we calculate $sim(e_t, e_r)$ for each training instance e_r on L as the weight of e_r and estimate $P(c_j|e_t)$ as follows.

$$P(c_j|e_t) = \frac{\sum_{e_r \in D_L, C(e_r)=c_j}(sim(e_t,e_r)+1)+1/|C|}{\sum_{e_r \in D_L}(sim(e_t,e_r)+1)+1}, \tag{6}$$

where D_L is the training instance subset on L, $C(e_r)$ is the class label of e_r. At the numerator of Equation 6, the total weights of the instances which belong to class c_j in D_L is computed, and the denominator is roughly the total weights of all the instances in D_L. Equation 6 uses the same *Laplace* correction as Equation 1.

Without changing the process of building a decision tree, we return the class probabilities from the leaves, which are estimated by *WPE*. Note that, for different test instances e_{t1} and e_{t2} falling into the same leaf, $P(c_j|e_{t1})$ and $P(c_j|e_{t2})$ could be different based on Equation 6. This means that we could make distinct probability estimates for the instances on the same node, which is the key to obtaining accurate ranking of instances.

5 Combination of *Shrinkage* and Weighted Probability Estimation

Both *shrinkage* and *WPE* are able to make up for the deficiencies of decision tree algorithms. *Shrinkage* breaks the restriction that the probabilities only can be estimated on the leaves, and *WPE* solves the problem that the different test instances falling into the same leaf are assigned to the same class probability estimate. However, they still suffer from some problems. In Algorithm **DIW**, the same probability estimate is assigned to different instances on the same node in **Step 3**. The generated interpolating weights for *shrinkage* are not effective enough. Moreover, when the sizes of training subsets on leaves are small, *WPE* could not generate reliable class probability estimates without the support of other nodes on the paths. The combination of *shrinkage* and *WPE* could compensate for the weakness of each. We illuminate the whole process of decision tree algorithm with the combination as follows.

Training:
First, a decision tree is built by a traditional decision tree algorithm. Then, Algorithm **DIW** is carried out to train the interpolating weights for *shrinkage*. We use *WPE* to estimate the class probabilities $P^i(c_j|x)$ for a training instance x on the leaf in **Step 3** of Algorithm **DIW**. Here x is treated as a test instance, and $P^i(c_j|x)$ is estimated by Equation 6 (instead of Equation 1). When the similarity is calculated using Equation 5, we compare all of the attribute values and the class labels, since the class label is known for x.

Testing:
Given a test instance e_t, it is sorted along a path from the root to a leaf. The local class probabilities $P^i(c_j|e_t)$ are estimated by *WPE* (Equation 6). We do not use

the class label to calculate the similarity in Equation 5, since the class label is unknown for e_t. Finally, the returned class probability $P(c_j|e_t)$ is estimated by *shrinkage* (Equation 2) with the interpolating weights determined by Algorithm **DIW** and the local probabilities estimated by *WPE*.

Using *WPE* in **Step 3** of Algorithm **DIW**, the distinct probability estimates are assigned to the instances on the same node so that the returned interpolating weights are more effective. This also makes it possible that Algorithm **DIW** can be adapted to a multiple-iteration algorithm, which terminates when the interpolating weights are converged. The multiple-iteration algorithm appears like an EM algorithm. However, a typical EM algorithm takes thousands of iterations to converge, which is fairly time-consuming. In the experiments, we still use the single-iteration Algorithm **DIW**. The experimental results show that the outcome of weights after many iterations is not significantly better than the one from just one iteration.

6 Experiments and Results

Our experiments are conducted on 35 data sets from Weka [16], which come from the UCI repository. We download these data sets in the format of $arff$ from the website of Weka. There are some preprocessing stages adopted on each data set. First, we use the filter *ReplaceMissingValues* in Weka to replace the missing values of attributes in each data set. Second, we use the filter *Discretize*, the unsupervised ten-bin discretization in Weka, to discretize numeric attributes. Thus, all the attributes are treated as nominal. Third, we notice that, if the number of values of an attribute is almost equal to the number of instances in a data set, this attribute does not contribute any information to the purpose of prediction. So we use the filter *Remove* in Weka to delete this type of attribute.

In the first experiment, we compare the algorithms, such as C4.5 with combination of *shrinkage* and *WPE* (C45-C), C4.5 with *shrinkage* (C45-S), C4.5 with *WPE* (C45-W), C4.5 (C45), C4.5 with m-Branch (C45-M), C4.5 with LingYan's algorithm (C45-L), C4.5 with bagging (C45-B). In the second experiment, we compare the algorithms, such as C4.4 with the combination of *shrinkage* and *WPE* (C44-C), C4.4 with *shrinkage* (C44-S), C4.4 with *WPE* (C44-W), C4.4 (C44), C4.4 with m-Branch (C44-M), C4.4 with LingYan's algorithm (C44-L), C4.4 with bagging (C44-B). We implement *shrinkage*, *WPE*, the combination[1], C4.4, m-Branch, LingYan's algorithm and AUC evaluation within the Weka framework, and use the implementation of C4.5 and bagging in Weka. In all experiments, the AUC score of an algorithm on a data set is obtained via 5 runs of ten-fold cross validation. Runs with the various algorithms are carried out on the same training data sets and evaluated on the same test data sets. Finally, we conduct two-tailed t-test with a 95% confidence level to compare each pair of algorithms.

[1] Codes for *shrinkag*, *WPE* and their combination are available at `http://www.cs.unb.ca/~hzhang/ShrinkageCode.rar`

Table 3 and Table 4 show the AUC scores of the algorithms on each data set. The two-tailed t-test results are shown in Table 1 and Table 2. Each entry of Table 1 and Table 2 has the format of $w/t/l$. It means that compared with the algorithm in the corresponding column, the algorithm in the corresponding row wins in w data sets, ties in t data sets and loses in l data sets. From our experiments, we have the following observations:

- C45-S and C44-S outperform C45 and C44 respectively. C45-W and C44-W outperform C45 and C44 respectively.
- Either *shrinkage* or *WPE* is not perfect compared to some techniques. C45-S is worse than C4.5 with other techniques. C44-S is not as good as C44-M and C44-B. C44-W is worse than C4.4 with other techniques.
- C45-C outperforms C45 and C4.5 with any other techniques. C44-C outperforms C44 and C4.4 with any other techniques.

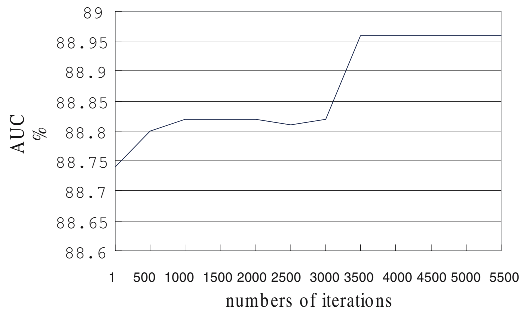


Fig. 3. The AUC scores corresponding to the different numbers of iterations

The experimental results are not surprising. Due to assigning the same local probability estimates on each node, Algorithm **DIW** could not return more effective interpolating weights, so *shrinkage* is worse than other techniques. Although *WPE* is able to estimate the probability distinctly, it suffers from the case when the sizes of training subsets on leaves are small, and that's why C44-W is worse than C4.4 with other techniques. The combination of *shrinkage* and *WPE* solves the above problems: First, applying *WPE* in Algorithm **DIW**

Table 1. Summary of comparisons for the algorithms related with C4.5

	C45-C	C45-S	C45-W	C45	C45-M	C45-L
C45-S	0/5/30					
C45-W	3/18/14	19/16/0				
C45	0/5/30	3/18/14	0/4/31			
C45-M	2/9/24	8/27/0	1/17/17	16/19/0		
C45-L	0/10/25	14/15/6	2/17/16	19/13/3	10/18/7	
C45-B	2/7/26	14/18/3	4/17/14	24/11/0	10/22/3	7/20/8

Table 2. Summary of comparisons for the algorithms related with C4.4

	C44-C	C44-S	C44-W	C44	C44-M	C44-L
C44-S	2/14/19					
C44-W	0/12/23	3/24/8				
C44	1/11/23	0/11/24	4/20/11			
C44-M	2/14/19	7/24/4	7/27/1	16/19/0		
C44-L	0/14/21	4/24/7	10/16/9	14/16/5	7/20/8	
C44-B	2/17/16	9/24/2	7/27/1	19/16/0	6/29/0	9/25/1

Table 3. Experimental results on AUC for the algorithms related with C4.5

Data set	C45-C	C45-S	C45-W	C45	C45-M	C45-L	C45-B
anneal	95.91	90.45	95.24	83.45	89.73	93.70	86.49
anneal.ORIG	95.61	91.42	94.18	86.00	90.04	91.33	86.53
audiology	70.66	62.81	70.08	61.54	62.68	69.98	64.86
autos	95.07	93.11	95.22	73.84	93.95	89.93	77.91
balance-scale	88.00	56.01	63.34	52.72	54.94	67.76	59.20
breast-cancer	92.28	62.89	92.21	60.86	62.73	67.54	64.97
breast-w	99.45	94.62	98.64	96.43	97.36	98.24	98.12
colic	95.01	85.42	91.35	81.17	85.41	85.63	85.32
colic.ORIG	92.42	84.50	90.25	83.56	85.38	80.96	87.68
credit-a	96.46	89.71	94.54	88.17	89.82	91.26	91.31
credit-g	82.49	72.90	75.35	68.48	72.32	75.08	74.16
diabetes	93.68	77.69	85.47	76.26	78.05	79.33	79.74
glass	89.61	81.93	84.60	77.11	79.88	82.46	81.26
heart-c	84.25	83.27	83.89	83.16	83.31	83.85	83.75
heart-h	84.58	81.01	84.40	80.91	81.02	83.76	83.77
heart-statlog	92.98	84.65	88.01	81.65	85.48	88.17	86.39
hepatitis	91.42	70.61	90.78	70.43	70.50	72.90	81.54
hypothyroid	95.12	66.50	98.33	68.56	68.74	82.05	83.31
ionosphere	96.32	84.29	94.25	88.98	90.57	88.32	94.72
iris	99.68	99.12	99.41	98.99	98.67	98.12	99.00
kr-vs-kp	99.73	99.56	99.93	99.81	99.88	98.90	99.91
labor	97.50	82.75	93.50	78.25	82.75	85.42	84.46
lymph	88.70	86.47	85.90	71.16	86.29	87.33	83.02
mushroom	100.00	100.00	100.00	100.00	100.00	99.46	100.00
primary-tumor	75.93	69.73	72.67	63.98	70.67	74.29	69.01
segment	98.60	97.81	99.24	98.47	98.97	96.19	99.33
sick	99.12	93.69	96.67	93.42	94.48	94.90	94.01
sonar	86.84	75.05	76.26	69.49	74.14	75.33	81.92
soybean	99.73	98.95	99.15	98.12	98.96	99.50	98.99
splice	98.70	97.88	98.12	96.84	98.21	98.90	98.52
vehicle	88.74	84.59	86.66	83.13	87.57	81.49	89.03
vote	98.69	96.20	98.60	96.57	97.53	98.23	97.64
vowel	97.20	94.65	95.32	92.83	95.61	92.96	96.40
waveform-5000	91.71	85.79	87.06	84.63	88.54	90.92	91.03
zoo	88.48	80.38	88.00	79.57	80.29	85.76	80.62

can generate the different probability estimates on each node for each training instances on the leaf; Second, *shrinkage* balances the probability estimation towards the nodes with large training subsets; Third, the class probability for a test instance is estimated by the local probabilities from *WPE* and *shrinkage* weights from Algorithm **DIW**.

In another interesting experiment, we adapt Algorithm **DIW** in C45-C to a multiple-iteration algorithm. For data set "Vehicle", the AUC scores from the different numbers of iterations are plotted in Figure 3. We observe that the AUC score is not changed at 88.96% after 3500 iterations. Compared with the AUC score 88.74% from a single iteration, we can see that the result after many iterations is not significantly better than the one from a single iteration.

Table 4. Experimental results on AUC for the algorithms related with C4.4

Data set	C44-C	C44-S	C44-W	C44	C44-M	C44-L	C44-B
anneal	96.02	94.71	94.77	93.95	94.13	93.67	94.83
anneal.ORIG	95.61	93.54	93.43	92.11	93.56	92.25	93.25
audiology	70.28	67.42	70.08	65.91	67.00	70.45	69.69
autos	95.12	93.68	94.92	91.28	94.31	90.43	94.91
balance-scale	79.50	57.76	66.79	59.42	57.90	66.45	65.15
breast-cancer	76.51	61.01	62.60	59.44	62.43	67.78	64.09
breast-w	99.30	96.58	98.59	98.08	98.18	98.27	98.79
colic	90.90	87.15	85.29	84.13	87.22	85.69	88.06
colic.ORIG	89.36	83.80	83.45	82.43	83.60	80.66	86.00
credit-a	94.40	90.99	89.43	88.30	91.31	91.26	90.42
credit-g	79.21	73.06	69.67	68.06	72.70	75.04	73.44
diabetes	89.04	78.91	77.74	74.66	78.65	79.98	78.78
glass	88.77	83.11	80.29	80.57	81.23	82.06	79.52
heart-c	84.03	83.29	83.35	83.19	83.47	83.84	83.65
heart-h	84.13	83.52	83.45	83.21	83.67	83.82	83.65
heart-statlog	90.14	85.24	84.33	82.82	85.59	88.66	86.73
hepatitis	86.62	81.29	81.62	78.64	80.76	77.92	83.03
hypothyroid	85.60	86.31	83.13	82.23	83.74	81.40	82.28
ionosphere	94.28	88.86	93.18	92.17	92.46	91.28	95.17
iris	99.65	98.00	98.93	98.52	98.85	97.33	98.68
kr-vs-kp	99.74	99.65	99.96	99.95	99.91	98.75	99.97
labor	95.42	82.96	86.58	84.63	86.29	84.04	89.42
lymph	88.78	88.12	87.04	86.44	87.22	87.61	88.08
mushroom	100.00	100.00	100.00	100.00	100.00	99.46	100.00
primary-tumor	76.05	72.48	72.10	69.23	72.96	74.44	73.07
segment	99.09	98.79	99.35	99.20	99.29	95.88	99.53
sick	98.92	97.87	99.20	99.11	99.19	94.41	99.24
sonar	83.30	78.44	79.32	77.35	78.65	75.57	82.58
soybean	99.76	99.08	98.93	98.12	99.01	99.54	98.90
splice	98.74	98.10	98.19	97.90	98.50	98.93	98.70
vehicle	88.56	85.32	86.76	86.19	87.84	82.06	88.91
vote	98.72	96.53	98.50	97.62	97.90	98.83	98.45
vowel	97.43	95.48	94.81	91.37	96.14	93.03	96.33
waveform-5000	90.55	87.46	83.50	80.85	86.98	90.77	89.87
zoo	88.48	81.00	88.00	80.57	80.81	87.05	81.19

7 Conclusions and Future Work

In this paper, we present a statistical technique, *shrinkage*, and an instance-based method, *WPE*, to improve the ranking performance of decision trees, which is measured by AUC. The class probability estimate with *shrinkage* is a linear interpolation for the local probability estimates on each node along the path from leaf to root. Algorithm **DIW** is proposed to determine the interpolating weights used in *shrinkage*. *WPE* produces the distinct probability estimates for the instances on the same node. In order to compensate for the deficiencies of *shrinkage* and *WPE*, we combine them together. In this process, we use *WPE* to generate distinct probability estimates on each node in training and testing processes, and also use *shrinkage* to return the final class probability estimate. The experiments show that decision tree algorithms with *shrinkage* and *WPE* outperform the original ones, and that the decision tree algorithms with this combination significantly outperform the original ones and other state-of-the-art techniques proposed to enhance the ranking performance of decision trees.

In our future research, we will study other local probability estimation methods in decision trees. We notice that naive Bayes model also generates distinct probability estimates for different instances falling into the same node. In *shrinkage*, deploying naive Bayes model along a path to estimate class probabilities may produce good ranking performance.

References

1. Bahl, L., Brown, P., deSouza, P., Mercer, R.: A tree-based statistical language model for natural language speech recognition. IEEE Transactions on Acoustics, Speech and Signal Processing. (1989) bf 37, 1001-1008
2. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting and variants. Artificial Intelligence **36** (1989) 105-142
3. Bradley, A. P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition **30** (1997) 1145-1159
4. Buntine, W.: Learning classification trees. Artificial Intelligence frontiers in statistics. Chapman & Hall, London (1993) 182–201
5. Ferri, C., Flach, P. A., Hernandez-Orallo, J.: Improving the AUC of Probabilistic Estimation Trees. Proceedings of the 14th European Conference on Machine Learning (ECML2003). Springer (2003)
6. Hand, D. J., Till, R. J.: A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. Machine Learning **45** (2001) 171-186
7. Hastie, T., Pregibon, L.: Shrinking Trees. AT & T Bell Laboratories (1990)
8. Ling, C. X., Huang, J., Zhang, H.: AUC: a statistically consistent and more discriminating measure than accuracy. Proceedings of 18th International Conference on Artificial Intelligence (IJCAI-2003). Morgan Kaufmann (2003) 329-341
9. Ling, C. X., Yan, R. J.: Decision tree with Better Ranking. Proceedings of the Twentieth International Conference on Machine Learning (ICML2003). AAAI Press (2003)
10. McCallum, A., Rosenfeld, R., Mitchell, T., Ng, A. Y.: Improving text classification by shrinkage in a hierarchy of classes. Proceedings of the 15th International Conference on Machine Learning. Morgan Kaufmann (1998) 359-367
11. Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., Bunk, C.: Reducing misclassification costs. Proceedings of the Eleventh International Conference on Machine Learning. Morgan Kaufmann (1994) 217-225
12. Provost, F., Domingos, P.: Tree Induction for Probability-based Ranking. Machine Learning. Kluwer Academic Publishers (2002)
13. Provost, F., Fawcett, T., Kohavi, R.: The case against accuracy estimation for comparing induction algorithms. Proceedings of the Fifteenth International Conference on Machine Learning. Morgan Kaufmann (1999) 445-453
14. Provost, F., Fawcett, T.: Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97). AAAI Press (1997) 43-48
15. Quinlan, J. R.: C4.5 Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA (1993)
16. Witten, I. H., Frank, E.: Data mining-practical machine learning tools and techniques with java implementation. Morgan Kaufmann, San Mateo, CA (2000)
17. Zadrozny, B., Elkan, C.: Obtaining calibrated probability estimates from decision trees and Naive Bayesian classifiers. Proceedings of the 18th International Conference on Machine Learning. Morgan Kaufmann (2001) 609-616

Multiple-Instance Learning Via Random Walk

Dong Wang, Jianmin Li, and Bo Zhang

State Key Laboratory of Intelligent Technology and System,
Department of Computer Science and Technology,
Tsinghua University, Beijing, 100084, P.R. China
wdong01@mails.tsinghua.edu.cn, {lijianmin, dcszb}@mail.tsinghua.edu.cn

Abstract. This paper presents a decoupled two stage solution to the multiple-instance learning (MIL) problem. With a constructed affinity matrix to reflect the instance relations, a modified Random Walk on a Graph process is applied to infer the positive instances in each positive bag. This process has both a closed form solution and an efficient iterative one. Combined with the Support Vector Machine (SVM) classifier, this algorithm decouples the inferring and training stages and converts MIL into a supervised learning problem. Compared with previous algorithms on several benchmark data sets, the proposed algorithm is quite competitive in both computational efficiency and classification accuracy.

1 Introduction

Multiple-instance learning (MIL) is a generalization of supervised learning. Unlike supervised learning, this model assumes that instances are contained in bags and the instance labels are hidden. The bag label is related to the hidden labels of the instances as follows: the bag is labeled as positive if any single instance in it is positive, otherwise it is labeled negative. Given a training set of labeled bags, an MIL algorithm attempts to find a hypothesis that correctly explains the labels for the bags on the training set and generalizes well to predict the labels for unseen bags. Many real world applications can be formulated in the MIL setting, e.g. drug design, protein identification, hard drive failure prediction, stock prediction, text categorization, content-based image retrieval (CBIR) and more recently content-based video retrieval (CBVR).

After Dietterich *et. al* first formulate MIL in [1], substantial research has been carried out in this area in the last few years, e.g. [2,3,4,5,6]. They can be roughly divided into two categories. A few algorithms operate directly on the bag level while others try to infer the hidden instance labels and then resort to supervised learning techniques. Intuitively, an instance is likely to be positive if it relates to many positive bags and does not relate to any negative bags. However the strict intersection of the positive bags may be empty when features are real valued. Therefore, a notable concept of "Diverse Density" (DD) is defined to measure in the feature space the co-occurrence of instances from different positive bags [2]. Unfortunately, it is computationally intensive to exhaustively search the feature space for the point which maximizes DD [2,3] and overfitting may occur since no regularization term is present. Searching for multiple local maximum points called "instance prototypes" [5] is also possible but it suffers from the same computation problem.

The motivation of the present study is to efficiently infer the underlying positive instances for all positive bags in parallel and let a classifier with proper regularization term do the left work. There are two advantages by doing so: 1) computational efficiency comes from operating only on the instances instead of searching in the whole space; 2) the two stages of inferring and training are decoupled and thus the training stage is open to different classification schemes. Holding this in mind, we take a Random Walk on a Graph approach to infer the underlying positive instances. Sending these inferred positive instances and the ones in negative bags into the state-of-the-art Support Vector Machine (SVM) classifier, we further takes advantage of the generalizing capability of SVM. Tested on several standard benchmark data sets, this approach is quite competitive in both computational efficiency and classification accuracy with previous ones.

The paper is organized as follows: Section 2 briefly presents related work. Section 3 introduces the Random Walk on a Graph and adapts it to the MIL setting. Section 4 gives the implementation details and Section 5 presents the experimental results. Section 6 discusses the relation of the present study and the most related work and Section 7 concludes this paper.

2 Related Work

2.1 Multiple Instance Learning

The MIL problem is first presented in [1], and an algorithm is also proposed with hypothesis classes consisting of Axis Parallel Rectangles (APR). Two methods focused on the DD concept are developed [2,3]. The former (DD algorithm) takes a two step scaling and gradient search approach, and the latter (EMDD) treats the hidden information of the underlying positive instances as a missing value and uses Expectation-Maximum (EM) to estimate it. Both include computationally dense operations. A few research tries to tailor the supervised learning algorithms for the MIL setting, such as [7,8,9]. Recently, an interesting comparison is made between supervised and multiple instance learning methods [6].

Another line of research handles the bag directly. A kernel-based approach is suggested which derives MI-kernels on bags from statistics of instances [10]. [4] propose both the maximum pattern and the maximum bag margin formulation (mi/MI-SVM) of MIL and solve the derived mixed-integer programming problem in heuristic manner due to formidable computation otherwise. DD-SVM is developed to search for multiple local maximum points and define similar bag kernels in an SVM framework [5]. Approximate Box Counting (ABC) [11] is proposed for extended "*r-of-k*" MIL setting. Ensemble learning methods of Boosting [12] and Ensemble-EMDD [13] are also explored.

2.2 Random Walk on a Graph

Random Walk on a Graph has been initially proposed to compute the absolute relevance of pages (vertices) in a hyperlinked environment, like the web, in the form of the PageRank algorithm [14]. A slightly different algorithm is developed for manifold learning [15]. This manifold learning approach has attracted more research attention in the learning community recently. For example, sparsely or densely connected graphs are built to deal with semi-supervised learning problems [16,17,18,19,20] and to cluster data [21,22]. However, though connected by propagating the labels on manifold data structure, MIL is different from semi-supervised learning in that the latter deals with both labeled and unlabeled data while the former concerns itself with the hierarchical label ambiguity and tries to identify the underlying positive instances in positive bags.

3 Random Walk on a Graph for MIL

In this section, after introducing some assumptions and notations, we first adapt the Random Walk on a Graph algorithm to infer the underlying positive instances, then combine an SVM classifier to solve the converted supervised learning problem. We also show that the adapted Random Walk on a Graph algorithm is somewhat similar in spirit to the original DD concept.

3.1 Assumptions and Notations

We make two assumptions here. The first one is that the positive instances form certain clusters in the feature space and the negative ones are distributed in the remaining space. The second one is that the bags and the instances are drawn independently from the actual data. These are two general assumptions which are usually satisfied in real world scenarios.

Let B_i denote the i^{th} bag, B_i^+ a positive bag and B_i^- a negative one. Let $\mathcal{B} = \{B_i\}$, $\mathcal{B}^+ = \{B_i^+\}$ and $\mathcal{B}^- = \{B_i^-\}$. Let J denote the set of all instances and $n = |J|$. An instance $I_j, j \in J$ is denoted I_j^+ when it is positive and is denoted I_j^- when negative. I_j can also be denoted as B_{ij} to emphasis that $I_j \in B_i$ and as B_{ij}^+ if it is positive. Note that j is a global index for instances and does not relate to specific bag index. Let NN_j denote the set of k nearest neighbor (k-NN) instances from *other* bags for each instance I_j. Denote by $p(I_j^+)$ the probability of I_j being positive and $p(I_j^-)$ similarly.

Each instance I_j is represented by a d-dimensional feature vector. Let $A_{n\times n}$ be the affinity matrix defined for the instances. Normalize A by $S = D^{-1}A$, where D is the diagonal matrix with $D_{ii} = \sum_{k=1}^{n} A_{ik}$.

3.2 Random Walk on a Graph

A Random Walk on a given graph $G = \{V, E\}$, where V is the vertex set of size n and E the edge set, describes how a random walker jumps among vertices following the edges with certain probabilities. This can be characterized by a discrete time markov chain which allows us to compute the probability x_p of being located in each vertex p at time t. Suppose that the transition probability matrix is P and the probability distribution over all the vertices is $x(t) = [x_1(t), \dots, x_n(t)]^T$, a unique stationary distribution x^* is readily derived since P is a stochastic matrix having its maximum eigenvalue equal to one and this guarantees the convergence (see e.g. [23], chapter 4).

Initialize $x(0) = r$ at $t = 0$ so that r is a probabilistic distribution for the random walker to start with. A restart coefficient α is introduced to indicate the probability αr that the walker stops following edges and restarts from the vertices again. Suppose that a random walker starts at vertex u, it follows the arc (u, v) to vertex v with probability $(1 - \alpha)p_{vu}$, where $p_{vu} = P(v, u)$ is the transition probability from vertex u to v, or restarts from v with probability αr_v. The probability $x_v(t)$ are updated at each time step by the following equation

$$x_v(t+1) = (1-\alpha) \sum_{u \in V} p_{vu} x_u(t) + \alpha r_v. \tag{1}$$

The compact matrix form is that

$$x(t+1) = (1-\alpha)Px(t) + \alpha r. \tag{2}$$

Inserting $y = x - \alpha(I - (1-\alpha)P)^{-1}r$ into (2), we have that $y(t+1) = (1-\alpha)Py(t)$. The convergence to the stationary distribution x^* comes directly since $y^* = 0$. We can easily show that

$$x^* = \alpha(I - (1-\alpha)P)^{-1}r. \tag{3}$$

This completes the closed form solution.

Although x^* can be expressed in a closed form, the iterative solution provides a more efficient algorithm for large scale problems. It just uses (2) to iterative update x until convergence arrived. The exponential convergence of x^* is easily derived from the Dobrushin convergence theorem (see also [23]).

3.3 Adapting Random Walk to the MIL Setting

The original DD concept does capture the nature of the MIL problem. It tries to find the point c $\arg\max_c p(\mathcal{B}|c) = p(\mathcal{B}^+|c)p(\mathcal{B}^-|c)$. Assuming both bag and instance independence and under a noisy-or model[2], it turns out to be $p(\mathcal{B}|c) = \prod_i (1 - \prod_j (1 - p(c|B_{ij}^+))) \prod_l \prod_k (1 - p(c|B_{lk}^-))$ which measures how *different* positive bags have instances near c and how far negative instances are from c [2]. One well placed negative instance will bring DD to near zero since DD is very sensitive to instances in negative bags. However, the computation of searching the whole feature space for c is formidable.

Adopting an additive model instead of the noisy-or model, we attempt to estimate a likelihood ratio (LR) of $p(\mathcal{B}^+|c)/p(\mathcal{B}^-|c)$ instead of the likelihood for each instance in parallel, and treat the inferred instances in a supervised learning framework. This additive model assumes that each instance casts its probabilistic vote to every instance independently and each instance receives votes additively. So it is more robust to instances in negative bags than the noisy-or model due to its additive nature. We substitute c with I_j so that $p(\mathcal{B}^+|c) = p(\mathcal{B}^+|I_j^+)$ and $p(\mathcal{B}^-|c) = p(\mathcal{B}^-|I_j^-)$ since we are only interested in the instances, not the whole feature space. To be specific, we model that

$$p(\mathcal{B}^+|I_j^+) \propto p(I_j^+|\mathcal{B}^+) = \sum_i p(I_j^+|B_i^+) = \sum_i \sum_k p(I_j^+|B_{ik}^+)\hat{p}(B_{ik}^+|B_i^+), \tag{4}$$

where $\hat{p}(B_{ik}^+|B_i^+)$ is the local probability for I_k being positive given its bag label as positive. The interpretation of (4) is that the collective instance votes for instance I_j sum up to the probability of I_j being positive given $\mathcal{B}^+$. We model that

$$p(\mathcal{B}^-|I_j^-) \propto p(I_j^-|\mathcal{B}^-) = \sum_l p(I_j^-|B_l^-) = \sum_l \sum_k p(I_j^-|B_{lk}^-)\hat{p}(B_{lk}^-|B_l^-), \tag{5}$$

similarly. This results in our LR measure of $p(\mathcal{B}^+|I_j^+)/p(\mathcal{B}^-|I_j^-)$ since $p(\mathcal{B}^+|I_j^+)$ and $p(\mathcal{B}^-|I_j^-)$ are modeled in the same way. In contrast, the likelihood measure is adopted since the noisy-or model treat these two parts asymmetrically.

We can grasp the solution (3) from this probabilistic voting view. Let $\beta = 1 - \alpha$, and omit the constant coefficient α in (3) for the time being, we have

$$\begin{aligned} x^* &= (I - \beta P)^{-1} r \\ &= (I + \beta P + \beta^2 P^2 + \ldots) r \\ &= r + \beta P r + \beta P(\beta P r) + \ldots. \end{aligned} \tag{6}$$

Compare (4) and (6) by plugging in $r_j = \hat{p}(I_j^+|\mathcal{B}^+)$, $x_j^* = p(I_j^+|\mathcal{B}^+)$ for each instance I_j and remember that $B_{ik} = I_k$, we have that

$$p(I_j^+|B_{ik}^+) = \delta_{jk} + \beta p_{jk} + \sum_l \beta^2 p_{jl} p_{lk} + \ldots \tag{7}$$

where δ_{jk} is the indication function. $p(I_j^+|B_{ik}^+)$ is the probability of commuting from I_k to I_j in infinite time with a damping factor β punishing long commuting time. This choice of $p(I_j^+|B_{ik}^+)$ is in accordance with our intuition that closeness makes similar. The second assumption stated in Section 3.1 guarantees that the closeness in the feature space is meaningful for inferring instances. Estimating $p(I_k^-|\mathcal{B}^-)$ follows an identical procedure with similar input r_k as $\hat{p}(B_{lk}^-|B_l^-)$.

The LR calculation is thus completed within the Random Walk framework. However, two issues of how to construct P and choose r remain to be solved. Firstly in the graph construction, the normalized affinity matrix $S = D^{-1}A$ is taken as the transition probability matrix P. The affinity matrix A can typically be defined by a Gaussian weighted k-NN distances in (8). The intra-bag k-NN relations are intentionally exclude by only allowing k-NN from *other* bags in A. So the random walker is forced to move among different bags in every step to avoid *self-reinforcement*. Otherwise two nearby negative instances in the same positive bag will vote heavily on each other and cause false high scores in x^*.

Secondly, the input vector r^+ for estimating $p(I_j^+|\mathcal{B}^+)$ is set so that $r_j = 1$ if $I_j \in B_i^+$ and $r_j = 0$ otherwise. This means that $\hat{p}(B_{ik}^+|B_i^+) = 1$ if $I_k \in B_i^+$ and $\hat{p}(B_{ik}^+|B_i^+) = 0$ if $I_k \in B_l^-$. So according to (4), the instances in negative bags will not contribute to $p(I_j^+|\mathcal{B}^+)$. Similarly, the input vector r^- for estimating $p(I_j^-|\mathcal{B}^-)$ is set so that $r_j = 1$ if $I_j \in B_l^-$ and $r_j = 0$ otherwise. However, noticing that $P\mathbf{1} = \mathbf{1}$, where $\mathbf{1} = [1, \ldots, 1]^T$ and $r^+ + r^- = \mathbf{1}$, it follows that $p(I_j^+|\mathcal{B}^+) + p(I_j^-|\mathcal{B}^-) = 1$. So the LR for I_j is further simplified to $\frac{x_j^*}{1-x_j^*}$ and instances in each positive bag can be ranked as being positive according to this LR measure.

After adapting the Random Walk on a Graph process to the MIL setting, a further instance selection and classifier training stage is added. SVM is chosen for its generalization ability. We omit an introduction to SVM and refer the readers to [24]. However the Random Walk, by itself, is neutral to the choice of classifiers.

4 Algorithm Description and Details

The completed running algorithm, Random Walk with SVM (RW-SVM) is shown in Figure 1. The algorithm implementation and parameter settings are given as follows.

4.1 Implementation Details

Given the bags and the instance features, the first step to construct the affinity matrix A is to calculate the k nearest neighbors (k-NN) from *other* bags for each instance and set the edge with the distance defined in (8). There are several other methods to generate A. For example, A can be dense if all inter-bag distances are incorporated. However the computational cost will increase dramatically. Another

```
Algorithm: Random Walk with SVM
Input: $n$ instances with the corresponding feature vectors, bag indexes and bag labels.
Output: SVM classifier $C$.
1. Construct affinity matrix $A$ by setting $a_{jk} = a_{kj} = d(j,k)$, where $k \in NN_j$, $k$-NN from other bags for $I_j$, and $d(j,k)$ is the distance function in (8) or (9).
2. Normalize $A$ so that $S = D^{-1}A$ in which $D$ is diagonal with $d_{ii} = \sum_{k=1}^{n} a_{ik}$.
3. Set input vector $r$ as $r_j = 1$ if $I_j \in B_i^+$ and $r_j = 0$ otherwise.
4. Let $P = S$, $x(0) = r$, and iteratively compute $x(t)$ with (2) until it converges to $x^*$.
5. Select $I_j$ for each $B_i^+$ that $\arg\max_j \frac{x_j^*}{1-x_j^*}, I_j \in B_i^+$ and select all instances $I_j \in B_l^-$.
6. Train the SVM classifier $C$ using the selected instances.
```

Fig. 1. Algorithm description of Random Walk with SVM

possibility is that the relative importance of each feature dimension may be different regarding the distance calculation. However, the scaling parameter σ_l for every feature dimension l, which is iteratively optimized in [2,3], is set to a constant σ across feature dimensions in (8) since the equal importance of feature dimensions works well in [4].

Notice that in this way, the constructed graph is not necessarily connected and may consist of several separate clusters. Some instances will still have zero scores after the iteration process. However, those are of course negative ones since they are not connected with the instances in positive bags.

In step 2, S can also be symmetrically normalized as $S = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ at the cost of losing the probabilistic interpretation for the Random Walk process. So the current form is preserved.

After S and r are set, they are used to compute the stationary distribution x^* until convergence. Then the instance I_j that maximizes $x_j^*, I_j \in B_i^+$ is chosen as the positive instance for each B_i^+; all instances are chosen as negative instances for each B_i^-. This is a conservative scheme for choosing positive instances since each positive bag contains at least one positive instance. More complicated schemes for choosing the positive instances are also possible, e.g. set a threshold for positive instances across positive bags, or use mi/MI-SVM like iterative scheme.

These selected instances are fed into the SVM classifier. LIBSVM [24] is used as our SVM implementation.

During test phase, the instances are classified by C and the bag is decided to be positive when any of the instances in it is classified as positive.

4.2 Parameter Settings

There are three parameters for graph construction, namely, the distance function, σ and k. The L_2 distance with Gaussian kernel is chosen for defining edge weight in affinity matrix A as follows:

$$a_{pq} = d_G(p,q) = \frac{1}{Z}\prod_{l=1}^{d} \exp\Big(\frac{-(p_l - q_l)^2}{\sigma_l^2}\Big), \tag{8}$$

where p, q are two instances in the feature space and Z the normalization constant. Note that $\sigma_l = \sigma$ for all dimensions. For some features depending on the data set (see 5 for details), the cosine distance with Laplace kernel is adopted as follows:

$$a_{pq} = d_C(p,q) = \frac{1}{Z}\exp\Big(\frac{-(1 - \cos\langle p,q\rangle)}{\sigma_c}\Big). \tag{9}$$

There is one parameter for the iteration process, the restart probability α. These four parameters are so-called hyper-parameters. There are also different parameters for different kinds of SVMs on each data set.

Generally speaking, it is difficult to tune the hyper-parameters. Due to the limited data available and the need of fair comparison with previous approaches, the parameter tuning method adopted here is somewhat complicated. The final classification results are obtained by 10-fold cross-validation runs on each data set.

Inside each cross-validated run, a nested 3-fold cross-validation is carried out to determine these parameters. The hyper-parameters are chosen with fixed SVM parameters and each hyper-parameter is determined independently with other hyper-parameters fixed. After these three parameters are set, the SVM parameters are further determined on the same 3-fold cross validation data by a 5×5 grid search procedure. The distance function are chosen according to the data set used. The SVM kernel type are chosen as the same with the previous approaches for fairness.

5 Experiments

5.1 Experimental Setup

Following [4], we perform experiments on the same data sets[1] to evaluate the proposed algorithm and compare it with other methods. The data sets are from a variety of application domains, including the most frequently used MUSK, a small portion of Corel and TREC9. Bag classification accuracy is taken as the performance measure for comparison. The results are averaged over 10-fold cross-validation runs. This random cross-validation is again repeated 10 times to get the significance of the results at $p > 0.95$. However we do not have the corresponding confidence interval for the comparative methods on these data sets since this kind of measurement is not provided in previous studies.

The testing data are *excluded* from the Random Walk process in each fold. The performances of previous methods are adopted from [4] unless otherwise stated. The actual running time are also reported on the MUSK data set to show the effectiveness and efficiency of the proposed approach. However for the Corel and TREC9 data sets, comparison are made only among EMDD, mi/MI-SVM and RW-SVM due to the lack of performance data of other approaches mentioned.

To simplify the parameter tuning process, we choose the default value of the numerical hyper-parameters as $\sigma = 10$ for (8), $k = 15$ and $\alpha = 0.1$. These values are those frequently chosen by the inner 3-fold cross-validation processes on the MUSK data set. These default parameters works pretty well for the data sets and are used unless otherwise stated.

5.2 Drug Design

The MUSK data sets, which are used for drug design, are the benchmark used in virtually all previous approaches. Both data sets, MUSK1 and MUSK2, consist of descriptions of molecules (bags) using multiple low-energy conformations (instances). Each conformation is represented by a 166-dimensional feature vector derived from surface properties. The L_2 distance function in (8) is used for constructing A. The RBF kernel $K(x; y) = exp(-\kappa\|x - y\|^2)$ is adopted for both data sets and feature vectors are normalized in a per-dimension min-max style [24] for

[1] For detailed data set descriptions , see [4]. The data sets are available from http://www.cs.brown.edu/people/stu/mil/datasets.html

Table 1. Classification accuracy of different methods on the Musk data sets

Data set	APR	DD	EMDD	MI-NN	mi-SVM	MI-SVM	DD-SVM	ABC	Boosting	RW-SVM
Musk1	**92.4**	88.0	84.8	88.9	87.4	77.9	85.8	91.2	92.0	87.6±1.1
Musk2	89.2	84.0	84.9	82.5	83.6	84.3	**91.3**	90.3	87.1	87.1±0.9

Table 2. Classification accuracy of different methods on the Corel image data sets

Data set	EMDD	mi-SVM			MI-SVM			RW-SVM		
Category		linear	poly	rbf	linear	poly	rbf	linear	poly	rbf
Elephant	78.3	82.2	78.1	80.0	81.4	79.0	73.1	82.1±0.8	**83.7±1.1**	83.3±1.7
Fox	56.1	58.2	55.2	57.9	57.8	59.4	58.8	57.0±1.8	58.5±2.0	**60.0±2.0**
Tiger	72.0	78.4	78.1	78.9	**84.0**	81.6	66.6	78.1±1.0	78.7±1.0	79.5±0.7

SVM input. As summarized in Table 1 with the confidence interval, RW-SVM achieves fairly good performance on both MUSK1 and MUSK2[2] among all these algorithms (See Section 2 for proper abbreviations for the algorithms). We try to add more instances as positive by introducing a global threshold of $th = \max x_j^*$ for all instances in negative bags. All instances in positive bags with $x_j^* > th$ are taken as positive ones. However, no significantly better result is produced.

5.3 Automatic Image Annotation

For the image annotation task from the Corel data, the original color images (bags) have been segmented to regions (instances), each characterized by color, texture and shape descriptors of total 230 dimensions. Three different categories ("elephant", "fox", "tiger") are used. In each case, the data set has 100 positive and 100 negative example images. The latter have been randomly drawn from a pool of photos of other animals. The L_2 distance in (8) is chosen for constructing A. The feature vectors are normalized in a per-dimension min-max style [24] for the three kinds of kernels used. As shown in Table 2, RW-SVM outperforms EMDD by a few percent, and performs comparably with mi/MI-SVM.

5.4 Text Categorization

We also test our algorithm on data sets from text categorization. These data sets are extremely sparse and high-dimensional, which makes them more challenging. In short, the data set which comes from TREC9 is randomly subsampled and split into subsets. Each subset contains a few thousands paragraphs (instances) and at lease 100 documents (bags). The L_2 normalized cosine distance in (9) is chosen for constructing A and $\sigma_c = 0.5$ for (9) and $k = 40$ are set for these data sets. The two parameters are again chosen as the values which are frequently selected

[2] The performances of DD-SVM, ABC and Boosting are adopted from [5,11,12] respectively.

Table 3. Classification accuracy of different methods on the TREC9 document categorization sets

Data set	EMDD	mi-SVM			MI-SVM			RW-SVM		
Category		linear	poly	rbf	linear	poly	rbf	linear	poly	rbf
TST1	85.8	93.6	92.5	90.4	93.9	93.8	93.7	**96.1±0.3**	95.3±0.4	96.0±0.3
TST2	84.0	78.2	75.9	74.3	**84.5**	84.4	76.4	81.4±0.6	78.8±1.8	82.3±0.8
TST3	69.0	87.0	83.3	69.0	82.2	85.1	77.4	**88.9±0.4**	71.7±2.6	88.9±0.4
TST4	80.5	82.8	80.0	69.6	82.4	82.9	77.3	84.7±0.5	74.2±3.5	**84.9±0.6**
TST7	75.4	**81.3**	78.7	**81.3**	78.0	78.7	64.5	79.1±0.8	77.9±2.1	79.9±0.9
TST9	65.5	67.5	65.6	55.2	60.2	63.7	57.0	70.9±0.5	62.1±0.9	**71.4±1.1**
TST10	78.5	79.6	78.3	52.6	79.5	81.0	69.1	83.6±0.5	73.4±2.1	**83.7±0.5**

by the inner 3-fold cross-validation processes for hyper-parameters on TS1 data set. The feature vectors are normalized to unit length with L_2 norm for the three kinds of kernels. As shown in Table 3, RW-SVM shows improved performance over mi/MI-SVM (and EM-DD subsequently) in five out of seven subsets.

5.5 Running Time

As shown in Table 4, the total time spent by our algorithm was 56.5s (2 hours) for Musk 1 (Musk 2) on a P4-3.0 GHz PC. This time includes graph construction and 5×5 grid search of 3 fold cross validation for SVM parameters which is carried out in each of the 10 runs. The Random Walk process itself typically requires less than a few tens of iterations which equal to 0.1 second for several thousand instances. Although the time is not directly comparable across algorithms[3], it does give us some hint of the order of the algorithms' computational complexity.

Table 4. Running time of several algorithms on the MUSK data sets. Note that it only provides some hint of the order of computational complexity of each algorithm.

Data set	RW-SVM	ABC	EMDD
MUSK1	56.5 seconds	2 hours	135 hours
MUSK2	2 hours	40 hours	485 hours

6 Discussion

Our algorithm extends the DD concept in the following aspects: the iterative two stage operations are separated and simplified as two cascade stages of inferring and training; the influence of not-so-near instances is accumulated with discounting factor β; the additive probabilistic model, which is more preferable in some cases

[3] The running time for ABC and EMDD are adopted from [11] on a 750 MHz Sun Blade.

as discussed in [5], replaces the noisy-or model; a regularization term is added in both the Random Walk [25] and the SVM classification process afterwards.

Both [4] and [5] consider SVMs for MIL. The main difference between [4] and our algorithm is that their formulation still results in two interleaved stages of inferring and classifying while ours decouples the two stages. In the presentation of [5], multiple local maximum points for DD are searched in the whole feature space. In the current study, the inferred underlying positive instances instead of the local maximum points are sent to SVM for classification and a dramatic speedup is thus produced.

7 Conclusion and Further Work

In this paper, a decoupled two stage solution is presented for the multiple-instance learning (MIL) problem. With an affinity matrix to reflect the data manifold, a modified Random Walk on a Graph process tries to infer the positive instances. This algorithm has both a guaranteed convergence and a fast iterative implementation. It is also open to different classification schemes. Comparative experiments on some benchmark data sets have shown that this algorithm is quite competitive with previous algorithms in both accuracy and speed when combined with SVMs.

Further work includes designing different schemes to choose the positive and negative instances for classifier training, comparing more thoroughly with other supervised and MI learning methods, applying advanced manifold learning algorithms, fusing different kinds of features and classifiers in this framework and applying this algorithm to large scale real-world tasks which are not handled before due to computational limits, such as the large scale CBVR data set of TRECVID 2006.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (60135010, 60321002), by the Chinese National Key Foundation Research and Development Plan (2004CB318108). Special thanks goes to Andrews Stuart for providing the data sets and insightful comments and Chih-Jen Lin for the LIBSVM implementation. Thanks the anonymous reviewers for invaluable advices.

References

1. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artificial Intelligence **89** (1997) 31–71
2. Maron, O.: Learning from Ambiguity. PhD thesis, MIT (1998)
3. Zhang, Q., Goldman, S.: Em-dd: An improved multiple-instance learning technique. In: NIPS (14). (2002) 1073–1080
4. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: NIPS(15). (2003)

5. Chen, Y., Wang, J.Z.: Image categorization by learning and reasoning with regions. Journal of Machine Learning Research **5** (2004) 913–939
6. Ray, S., Craven, M.: Supervised versus multiple instance learning: An empirical comparison. In: Proc. 22th International Conf. on Machine Learning. (2005)
7. Ramon, J., Raedt, L.D.: Multi instance neural networks. In: Proceedings of ICML-2000, Workshop on Attribute-Value and Relational Learning. (2000)
8. Wang, J., Zucker, J.D.: Solving the multiple-instance problem: a lazy learning approach. In: Proc. 17th Int. Conf. on Machine Learning. (2000) 1119–1125
9. Ruffo, G.: Learning single and multiple instance decision trees for computer security applications. PhD thesis, Universitá di Torino, Torino, Italy (2000)
10. Gärtner, T., Flach, P.A., Kowalczyk, A., Smola, A.J.: Multi-instance kernels. In: Proc. 19th International Conf. on Machine Learning. (2002) 179–186
11. Tao, Q., Scott, S.D., Vinodchandran, N.V.: Svm-based generalized multiple-instance learning via approximate box counting. In: Proc. 21th International Conf. on Machine Learning. (2004)
12. Auer, P., Ortner, R.: A boosting approach to multiple instance learning. In: Proc. of the 15th European Conf. on Machine Learning. (2004)
13. Rahmani, R., Goldman, S.A., Zhang, H., Krettek, J., Fritts, J.E.: Localized content based image retrieval. In: Proc. ACM Int. Conf. on Multimedia (ACM MM). (2005)
14. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: Proc. 7th Int. World Wide Web Conf. (1998)
15. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: NIPS(15). (2003) 237–244
16. Szummer, M., Jaakkola, T.: Partially labeled classification with markov random walks. In: NIPS. (2002)
17. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using gaussian fields and harmonic functions. In: ICML. (2003)
18. Belkin, M., Niyogi, P.: Using manifold stucture for partially labeled classification. In: NIPS. (2003)
19. Zhou, D., Huang, J., Schölkopf, B.: Learning from labeled and unlabeled data on a directed graph. In: Proc. 22th International Conf. on Machine Learning. (2005)
20. Zhu, X., Lafferty, J.: Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In: Proc. 22th International Conf. on Machine Learning. (2005)
21. Kulis, B., Basu, S., Dhillon, I.S., Mooney, R.J.: Semi-supervised graph clustering: A kernel approach. In: Proc. 22th International Conf. on Machine Learning. (2005)
22. Breitenbach, M., Grudic, G.Z.: Clustering through ranking on manifolds. In: Proc. 22th International Conf. on Machine Learning. (2005)
23. Seneta, E.: Non-Negative Matrices and Markov Chains. Springer-Verlag (1981)
24. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. (2001)
25. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Ranking on data manifolds. In: NIPS(15). (2003) 169–176

Localized Alternative Cluster Ensembles for Collaborative Structuring

Michael Wurst, Katharina Morik, and Ingo Mierswa

University of Dortmund, Department of Computer Science
Baroperstr. 301, 44221 Dortmund, Germany
{wurst, morik, mierswa}@ls8.cs.uni-dortmund

Abstract. Personal media collections are structured in very different ways by different users. Their support by standard clustering algorithms is not sufficient. First, users have their personal preferences which they hardly can express by a formal objective function. Instead, they might want to select among a set of proposed clusterings. Second, users most often do not want hand-made partial structures be overwritten by an automatic clustering. Third, given clusterings of others should not be ignored but used to enhance the own structure. In contrast to other cluster ensemble methods or distributed clustering, a global model (consensus) is not the aim. Hence, we investigate a new learning task, namely learning localized alternative cluster ensembles, where a set of given clusterings is taken into account and a set of proposed clusterings is delivered. This paper proposes an algorithm for solving the new task together with a method for evaluation.

1 Introduction

Collaborative approaches allow users to share preferences and knowledge without requiring a common semantic or explicit coordination. Data-driven methods as link analysis for web search and collaborative filtering have proven to be successful despite their lack of a clear semantic. Furthermore, not requiring coordination is one of the key factors that led to the fast growth of the Internet, as users can contribute information completely independently of other users.

Recently, new applications emerged under this Web 2.0 paradigm. Systems as flikr or del.icio.us allow users to annotate items with arbritrary chosen tags. Such tags complement global properties, e.g. artist, album, genre, etc. for music collections used by media organizers as iTunes. In contrast to these global properties, many user-assigned tags are *local*, i.e. they represent the personal views of a certain user not aiming at a global structure or semantic.

While users tend to start the organization of their personal collection eagerly, they often end up with a large set of items which are not yet annotated and a structure which is too coarse. A major challenge for machine learning is to exploit such local information in order to enable other users to navigate and structure media collections.

If there are enough annotated items, classification learning can deliver a decision function φ which maps items x of the domain X to a class g in a set of classes G. New items will be classified as soon as they come in and the user has no burden of annotation any more. However, classification does not refine the structure. If there is no structure given yet, clustering is the method to choose. It creates a structure of groups G for the not yet annotated items $S \subseteq X$. Traditional clustering schemes do not take into account the structure which users already have built up. Semi-supervised clustering obeys given groupings [1,2], but it does not refine structures. Non-redundant data clustering creates alternative structures to a set of given ones [3]. Given a structure G for all items in the collection, it creates an alternative structure G' for all items. However, it does not focus on the not yet annotated items S but restructures also the items which were already carefully structured.

Non-redundant clustering is connected to another area that has recently found increasing attention: clustering with background knowledge. In general, the idea of exploiting (user supplied) background knowledge has shown advantages, e.g., in text clustering [4] or lane finding in GPS data [5]. Although must-link constrained clustering reuse existing clustering, the label information will not be preserved. In addition, these approaches use a feature-based clustering instead of given input clusterings and are hence not applicable to our problem.

We may consider the structuring achieved so far a set of partitionings φ_i, each mapping S to a set of groups G_i. Ensemble clustering then produces a consensus φ which combines these input partitionings [6]. This is almost what we need. However, there are three major drawbacks: first, all input clusterings must be defined at least on S. Second, the consensus model does not take the locality of S into account. Finally, merging several heterogenous user clusterings by a global consensus does not preserve valuable label information.

In many current applications it is important to consider structures of several users who interact in a network, each offering a clustering $\varphi_i : S_i \rightarrow G_i$. A user with the problem of structuring her left-over items S might now exploit the cluster models of other users in order to enhance the own structure. Distributed clustering learns a global model integrating the various local ones [7]. However, this global consensus model again destroys the structure already created by the user and does not focus on the set S of not appropriately structured items.

Whether own partial clusterings or those of other peers in a network are given, the situation is the same: current clustering methods deliver a consensus model overwriting the given ones and do not take into account S. In addition, users might want to select among proposed models which the learner delivers. The practical need of the user in organizing her media collection is not yet covered by existing methods. The situation we are facing is actually a new learning task.

Let X denote the set of all possible items. A function $\varphi : S \rightarrow G$ is a function that maps objects $S \subseteq X$ to a (finite) set G of groups. The set Φ contains all possible functions φ. We denote the domain of a function φ with D_φ. In cases where we have to deal with overlapping and hierarchical groups, we denote the set of groups as 2^G.

Definition 1 (Localized Alternative Cluster Ensembles). *Given a set $S \subseteq X$, a set of input functions $I \subseteq \{\varphi_i : S_i \to G_i\}$, and a quality function*

$$q : 2^{\Phi} \times 2^{\Phi} \times 2^{S} \to R \tag{1}$$

with R being partially ordered[1] LOCALIZED ALTERNATIVE CLUSTERING ENSEMBLES *delivers the output functions $O \subseteq \{\varphi_i | \varphi_i : S_i \to G_i\}$ so that $q(I, O, S)$ is maximized and for each $\varphi_i \in O$ it holds that $S \subseteq D_{\varphi_i}$.*

Note that in contrast to cluster ensembles, the input clusterings can be defined on any subset S_i of X. Since for all $\varphi_i \in O$ it must hold that $S \subseteq D_{\varphi_i}$, all output clusterings must at least cover the items in S.

We present a method solving this task in two steps: a base algorithm (Section 2.1) which is enhanced to become a hierarchical clustering in Section 2.2. The method is well suited for distributed clustering (Section 3) and we present the application from which the work originated (Section 3.1). Based on actual structures of music collections we can evaluate our approach in a way similar to that of evaluating supervised learning tasks (Section 4).

2 An Approach to Localized Alternative Cluster Ensembles

In the following, we describe a clustering method, that is based on the idea of bag of clusterings: deriving a new clustering from existing ones by extending the existing clusterings and combining them such, that each of them covers a subset of objects in S. In order to preserve existing label information but allowing the group mapping for new objects we define the extension of functions φ_i:

Definition 2 (Extended function). *Given a function $\varphi_i : S_i \to G_i$, the function $\varphi'_i : S'_i \to G_i$ is an* EXTENDED FUNCTION *for φ_i, if $S_i \subset S'_i$ and $\forall x \in S_i : \varphi_i(x) = \varphi'_i(x)$.*

Extended functions allow us to define a bag of extensions of non-overlapping originally labeled subsets that covers the entire collection:

Definition 3 (Bag of clusterings). *Given a set I of functions. A* BAG OF CLUSTERINGS *is a function*

$$\varphi_i(x) = \begin{cases} \varphi'_{i1}(x), & \textit{if } x \in S'_{i1} \\ \vdots & \vdots \\ \varphi'_{ij}(x), & \textit{if } x \in S'_{ij} \\ \vdots & \vdots \\ \varphi'_{im}(x), & \textit{if } x \in S'_{im} \end{cases} \tag{2}$$

where each φ'_{ij} is an extension of a $\varphi_{ij} \in I$ and $\{S'_{i1}, \ldots, S'_{im}\}$ partitioning S.

[1] For example, $R = \mathbb{R}$ if one is interested in a unique solution.

Since each φ'_{ij} is an extension of an input clustering φ_{ij} on a subset S_{ij}, the label information is preserved. Now, we can define the objective function for our bag of clusterings approach to local alternative clustering ensembles.

Definition 4 (Quality of an output function). *The* QUALITY OF AN INDIVIDUAL OUTPUT FUNCTION *is measured as*

$$q^*(I, \varphi_i, S) = \sum_{x \in S} \max_{x' \in S_{ij}} sim(x, x') \text{ with } j = h_i(x) \tag{3}$$

where sim is a similarity function sim : $X \times X \rightarrow [0,1]$ *and* h_i *assigns each example to the corresponding function in the bag of clusters* $h_i : S \rightarrow \{1, \ldots, m\}$ *with*

$$h_i(x) = j \Leftrightarrow x \in S'_{ij}. \tag{4}$$

The QUALITY OF A SET OF OUTPUT FUNCTIONS *now becomes*

$$q(I, O, S) = \sum_{\varphi_i \in O} q^*(I, \varphi_i, S). \tag{5}$$

Besides optimizing this quality function, we want to cover the set S with a bag of clusterings that contains as few clusterings as possible.

2.1 The Algorithm

In the following, we present a greedy approach to optimizing the bag of clusterings problem. The main task is to cover S by a bag of clusterings φ. The basic idea of this approach is to employ a sequential covering strategy. In a first step, we search for a function φ_i in I that best fits the set of query objects S. For all objects not sufficiently covered by φ_i, we search for another function in I that fits the remaining points. This process continues until either all objects are sufficiently covered, a maximal number of steps is reached, or there are no input functions left covering the remaining objects. All data points that could not be covered are assigned to the input function φ_j containing the object which is closest to the one to be covered. Alternative clusterings are produced by performing this procedure several times using each input function at most once.

We now have to formalize the notion of a function sufficiently covering an object and a function fitting a set of objects such that the quality function is optimized. When is a data point sufficiently covered by an input function so that it can be removed from the query set S? We define a threshold based criterion for this purpose:

Definition 5. *A function* φ SUFFICIENTLY COVERS *a object* $x \in S$ *(written as* $x \sqsubset_\alpha \varphi$ *), iff* $x \sqsubset_\alpha \varphi :\Leftrightarrow \max_{x' \in Z_\varphi} sim(x, x') > \alpha$.

The set Z_{φ_i} of items is delivered by φ. This threshold allows us to balance the quality of the resulting clustering and the number of input clusters. A small value of α allows a single input function to cover many objects in S. This, on average, reduces the number of input functions needed to cover the whole query set. However, it may also reduce the quality of the result, as the algorithm covers many objects in a greedy manner, which could be covered better using an additional input function.

Turning it the other way around: when do we consider an input function to fit the items in S well? First, it must contain at least one similar object for each object in S. This is essentially what is stated in the quality function q^*. Second, it should cover as few additional objects as possible. This condition follows from the locality demand. Using only the first condition, the algorithm would not distinguish between input functions which span a large part of the data space and those which only span a small local part. This distinction, however, is essential for treating local patterns in the data appropriately. The situation we are facing is similar to that in information retrieval. The target concept S – the ideal response – is approximated by φ delivering a set of items – the retrieval result. If all members of the target concept are covered, the retrieval result has the highest recall. If no items in the retrieval result are not members of S, it has the highest precision. We want to apply precision and recall to characterize how well φ covers S. We can define

$$prec(Z_{\varphi_i}, S) = \frac{1}{|Z_{\varphi_i}|} \sum_{z \in Z_{\varphi_i}} max\,\{sim(x,z) | x \in S\} \tag{6}$$

and

$$rec(Z_{\varphi_i}, S) = \frac{1}{|S|} \sum_{x \in S} max\,\{sim(x,z) | z \in Z_{\varphi_i}\}. \tag{7}$$

Please note that using a similarity function which maps identical items to 1 (and 0 otherwise) leads to the usual definition of precision and recall. The fit between an input function and a set of objects now becomes a continuous f-measure:

$$q^*_f(Z_{\varphi_i}, S) = \frac{(\beta^2+1) rec(Z_{\varphi_i}, S) prec(Z_{\varphi_i}, S)}{\beta^2 rec(Z_{\varphi_i}, S) + prec(Z_{\varphi_i}, S)}. \tag{8}$$

Recall directly optimizes the quality function q^*, precision ensures that the result captures local structures adequately. The fitness $q^*_f(Z_{\varphi_i}, S)$ balances the two criteria.

Deciding whether φ_i fits S or whether an object $x \in S$ is sufficiently covered requires to compute the similarity between an object and a cluster. If the cluster is represented by all of its objects ($Z_{\varphi_i} = S_i$, as usual in single-link agglomerative clustering), this central step becomes inefficient. If the cluster is represented by exactly one point ($|Z_{\varphi_i}| = 1$, a centroid in k-means clustering), the similarity calculation is very efficient, but sets of objects with irregular shape, for instance, cannot be captured adequately. Hence, we adopt the representation by "well scattered points" Z_{φ_i} as representation of φ_i [8], where $1 < |Z_{\varphi_i}| < |S_i|$. These points are selected by stratified sampling according to G.

We can now dare to compute the fitness q^*_f of all $Z_{\varphi_i} \in I$ with respect to a query set S in order to select the best φ_i for our bag of clusterings. The

```
$O = \emptyset$
$I' = I$
**while** $(|O| < max_{alt})$ **do**
   $S' = S$
   $B = \emptyset$
   $step = 0$
   **while** $((S' \neq \emptyset) \wedge (I' \neq \emptyset) \wedge (step < max_{steps}))$ **do**
      $\varphi_i = \arg\max_{\varphi \in J} q_f^* (Z_\varphi, S')$
      $I' = I' \setminus \{\varphi_i\}$
      $B = B \cup \{\varphi_i\}$
      $S' = S' \setminus \{x \in S' | x \sqsubset_\alpha \varphi_i\}$
      $step = step + 1$
   **end while**
   $O = O \cup \{bag(B, S)\}$
**end while**
```

Fig. 1. The sequential covering algorithm finds bag of clusterings in a greedy manner. max_{alt} denotes the maximum number of alternatives in the output, max_{steps} denotes the maximum number of steps that are performed during sequential covering. The function *bag* constructs a bag of clusterings by assigning each object $x \in S$ to the function $\varphi_i \in B$ that contains the object most similar to x.

whole algorithm works as depicted in figure 1. We start with the initial set of input functions I and the set S of objects to be clustered. In a first step, we select an input function that maximizes $q_f^*(Z_{\varphi_i}, S)$. φ_i is removed from the set of input functions leading to a set I'. For all objects S' that are not sufficiently covered by φ_i, we select a function from I' with maximal fit to S'. This process is iterated until either all objects are sufficiently covered, a maximal number of steps is reached, or there are no input functions left that could cover the remaining objects. All input functions selected in this process are combined to a bag of clusters, as described above. Each object $x \in S$ is assigned to the input function containing the object being most similar to x. Then, all input functions are extended accordingly, again by nearest-neighbor classification (cf. definition 2). We start this process anew with the complete set S and the reduced set I' of input functions until the maximal number of alternatives is reached.

As each function is represented by a fixed number of representative points, the number of similarity calculations performed by the algorithm is linear in the number of query objects and in the number of input functions, thus $O(|I||S||Z_{\varphi_i}|)$. The same holds for the memory requirements.

2.2 Hierarchical Matching

A severe limitation of the algorithm described so far is, that it can only combine complete input clusterings. In many situations, a combination of partial clusterings or even individual clusters would yield a much better result. This is especially true, if local patterns are to be preserved being captured by maximally specific concepts. Moreover, the algorithm does not yet handle hierarchies. Our motivation for this research was the structuring of media collections. Flat structures are not sufficient with respect to this goal. We cannot use a standard hierarchical clustering algorithm, since we still want to solve the new task of local alternative cluster ensembles. In the following, we extend our approach to the combination of partial hierarchical functions. A hierarchical function maps objects to a hierarchy of groups.

Definition 6 (Group hierarchy). *The set G_i of groups associated with a function φ_i builds a* GROUP HIERARCHY*, iff there is a relation $<$ such that $(g < g') :\Leftrightarrow (\forall x \in S_i : g' \in \varphi_i(x) \Rightarrow g \in \varphi_i(x))$ and $(G_i, <)$ is a tree. The function φ_i is then called a* HIERARCHICAL FUNCTION.

It should be possible to match functions that correspond to only a partial group hierarchy. We formalize this notion by defining a hierarchy on functions, which extends the set of input functions such that it contains all partial functions.

Definition 7 (Function hierarchy). *Two hierarchical functions φ_i and φ_j, are in* DIRECT SUB FUNCTION RELATION *$\varphi_i \prec \varphi_j$, iff $G_i \subset G_j$, $\forall x \in S_i$: $\varphi_i(x) = \varphi_j(x) \cap G_i$, and $\neg\exists\varphi'_i : G_i \subset G'_i \subset G_j$.*

Let the set I^* be the set of all functions which can be achieved following the direct sub function relation starting from I, thus

$$I^* = \{\varphi_i | \exists\varphi_j \in I : \varphi_i \prec^* \varphi_j\} \tag{9}$$

where $\prec^*$ is the transitive hull of $\prec$. While it would be possible to apply the same algorithm as above to the extended set of input functions I^*, this would be rather inefficient, because the size of I^* can be considerably larger than the one of the original set of input functions I. We therefore propose an algorithm which exploits the function hierarchy and avoids multiple similarity computations. Each function $\varphi_i \in I^*$ is again associated with a set of representative objects Z_{φ_i}. We additionally assume the standard taxonomy semantics:

$$\varphi_i \prec \varphi_j \Rightarrow Z_{\varphi_i} \subseteq Z_{\varphi_j}. \tag{10}$$

Now, the precision can be calculated recursively in the following way:

$$prec(Z_{\varphi_i}, S) = \frac{|Z^*_{\varphi_i}|}{|Z_{\varphi_i}|} prec(Z^*_{\varphi_i}, S) + \sum_{\varphi_j \prec \varphi_i} \frac{|Z_{\varphi_j}|}{|Z_{\varphi_i}|} prec(Z_{\varphi_j}, S) \tag{11}$$

where $Z^*_{\varphi_i} = Z_{\varphi_i} \setminus \bigcup_{\varphi_j \prec \varphi_i} Z_{\varphi_j}$. For recall a similar function can be derived. Note, that neither the number of similarity calculations is greater than in the base version of the algorithm nor are the memory requirements increased.

Moreover, the bottom-up procedure also allows for pruning. We can optimistically estimate the best precision and recall, that can be achieved in function hierarchy using all representative objects Z_e for which the precision is already known. The following holds:

$$prec(Z_{\varphi_i}, S) \leq \frac{|Z_e| prec(Z_e, S) + |Z_{\varphi_i} \setminus Z_e|}{|Z_{\varphi_i}|} \tag{12}$$

with $Z_e \subset Z_{\varphi_i}$. An optimistic estimate for the recall is one. If the optimistic f-measure estimate of the hierarchy's root node is worse than the current best score, this hierarchy does not need to be processed further. This is due to the optimistic score increasing with $|Z_{\varphi_i}|$ and $|Z_{\varphi_i}| > |Z_{\varphi_j}|$ for all sub functions $\varphi_j \prec \varphi_i$. No sub-function of the root can be better than the current best score, if the score of the root is equal or worse than the current best score.

This conversion to hierarchical cluster models concludes our algorithm for Local Alternative Cluster Ensembles (LACE).

3 A Distributed Algorithm

The LACE algorithm is well suited for distributed scenarios. We assume a set of nodes connected over an arbitrary communication network. Each node has one or several functions φ_i together with the sets S_i. If a node A has a set of objects S to be clustered, it queries the other nodes and these respond with a set of functions. The answers of the other nodes form the input functions I. A computes the output O for S. The node B being queried uses its own functions φ_i as input and determines the best fitting φ_i for S and sends this output back to A. The algorithm is the same for each node. Each node executes the algorithm independently of the other nodes.

We introduce three optimizations to this distributed approach. First, given a function hierarchy, each nodes returns exactly one optimal function in the hierarchy. This reduces the communication cost, without affecting the result, because any but the optimal function would not be chosen anyway (see pruning in the last section).

Second, input functions returned by other nodes can be represented more efficiently by only containing the items in the query set, that are sufficiently covered by the corresponding function. Together with the f-measure value q_f^* (equation 8) for the function, this information is sufficient for the querying node in order to perform the algorithm.

In many application areas, we can apply a third optimization. If objects are uniquely identified, such as audio files, films, web resources, etc. they can be represented by these IDs only. In this case, the similarity between two objects is 1, if they have the same ID, and 0 otherwise. A distributed version of our algorithm only needs to query other nodes using a set of IDs. This reduces the communication cost and makes matching even more efficient. Furthermore, such queries are already very well supported by current (p2p) search engines.

In a distributed scenario, network latency and communication cost must be taken into account. If objects are represented by IDs, both are restricted to an additional effort of $O(|S|+|I^*|)$. Thus, the algorithm is still linear in the number of query objects.

3.1 Distributed Media Management

The LACE algorithm is applied within Nemoz[2], a distributed media organization system which focuses on the application of data mining in p2p networks. It supports users in structuring their private media collections by exploiting information from other peers. Each user may create arbitrary, personal classification schemes to organize her media, e.g. music. For instance, some users structure their collection according to mood and situations, others according to genres, etc. Some such structures overlap, e.g., the blues genre may cover similar music as does the melancholic mood.

Nemoz supports the users in structuring their media objects while not forcing them to use the same set of concepts or annotations. If an ad hoc network has been established, peers support each other in structuring. A user who needs to structure a set of media objects S (e.g., refining an over-full node in her taxonomy) invokes the distributed algorithm described above. Then, the system offers a set of alternative clusterings, each combined from peers' response and covering S. The user chooses which of the clusterings she wants to incorporate into her collection's structure. Note, that in this scenario, the enhanced functions from definition 2 become particularly meaningful – she receives recommendations for similar music in addition to her own set S!

4 Experiments

The evaluation of LACE is performed on a real world benchmark dataset gathered in a student project on distributed audio classification based on peer-to-peer networks (Nemoz). The data set contains 39 taxonomies (functions $\varphi_1, ..., \varphi_{39}$) and overall 1886 songs [9][3]. All experiments described in this paper were performed with the machine learning environment YALE [10][4].

The evaluation of LACE is performed by subsequently leaving out one function φ_i of the dataset. Then we apply clustering to reconstruct this taxonomy. Hence, we can evaluate cluster models in a way similar to classification learning. We have a "ground truth" available. A user taxonomy φ is compared with a taxonomy φ' created automatically by clustering as follows. We construct the usual tree distance matrix for the two taxonomies and compare these matrices on all pairs of objects in the set S. For the *absolute distance* criterion, the difference between the tree distance in φ and the one in φ' are summed-up and divided by the number of objects (see Table 1 for illustration).

As second criterion we use the *correlation* between these tree distances. Finally, for each cluster in the left-out taxonomy we search for the best corresponding cluster in the learned taxonomy according to f-measure. The average performance over all user-given clusters is then used as the (*FScore*) evaluation measure [11]. Note, that although we report the FScore, it is not normalized with

[2] Available at http://www.sourceforge.net/projects/nemoz
[3] Available at http://www-ai.cs.uni-dortmund.de/audio.html
[4] Available at http://yale.sf.net.

Table 1. Tree distance matrix indicating for all pairs of items in S how many edges they are away from each other, once concerning the hierarchy of φ and once concerning the hierarchy of φ'. For instance, in φ there is only one edge between x_1 and x_2, but in φ, there are three. The last columns sums-up the differences between the distances in φ and φ' for one item with respect to all other items. The last field gives the total of all differences. Total/m gives the absolute distance of φ and φ'.

S	x_1	x_2	...	x_m	sum of differences
x_1	-	φ:1;φ':3			2+
x_2		-		φ:1;φ':2	1+
...			-		
x_m				-	
Total					3+

Table 2. The results for different evaluation measures

Method	Correlation	Absolute distance	FScore
LACE	0.44	0.68	0.63
TD audio	0.19	2.2	0.51
TD ensemble	0.23	2.5	0.55
single-link audio	0.11	9.7	0.52
single-link ensemble	0.17	9.9	0.60
random	0.09	1.8	0.5

respect to the number of created clusters. Finer grained structures therefore always lead to equal or better performance than their coarse grained variants. This, however, does often not reflect the similarity to the user-given taxonomy.

We compare our approach with single-link agglomerative clustering using cosine measure, top down divisive clustering based on recursively applying kernel k-means [12] (TD), and with random clustering. Localized Alternative Cluster Ensembles were applied using cosine similarity as inner similarity measure. TD and random clustering were started five times with different random initializations. We use a set of 20 features which were shown to work well in a wide range of applications [13] as underlying audio features. Since, here, we want to test the new clustering method, we do not investigate different feature sets. The parameter β was set to 1.

Table 2 shows the results. As can be seen, the local alternative cluster ensembles approach LACE performs best. Note however, that absolute distance does not lead to results that are representative for agglomerative clustering as such, because it usually builds-up quite deep hierarchies, while the user constructed hierarchies were rather shallow.

A second experiment inspects the influence of the representation on the accuracy. The results of LACE with different numbers of instances at a node are shown in Table 3. Representing functions by all points performs best. Using a single centroid for representing a subtree leads to inferior results, as we already

Table 3. The influence of concept representation (cardinality of $|Z|$)

Representation	Correlation	Absolute distance	FScore
all points	0.44	0.68	0.63
$\|Z\| = 10$	0.44	0.68	0.63
$\|Z\| = 5$	0.41	0.69	0.63
$\|Z\| = 3$	0.40	0.69	0.62
centroid	0.19	1.1	0.42

Table 4. The influence of response set cardinality $|O|$

Alternatives	Correlation	Absolute distance	FScore
5	0.44	0.68	0.63
3	0.38	0.73	0.60
1	0.34	0.85	0.56

expected. Well scattered points perform well. We obtain good results even for a very small number of representative items at each node of the cluster model.

We also evaluated how the number of output functions influences the quality of the result. The result should be clearly inferior with a decreasing number. Table 4 shows the result. On one hand, we observe that even with just one model, i.e. $|O| = 1$, LACE still outperforms the other methods with respect to tree distance. On the other hand, the results are, indeed, getting worse with less alternatives. Providing alternative solutions seems to be essential for improving the quality of results at least in heterogeneous settings as the one discussed here. Probably, the performance would increase even further for more output clusterings. Although a user still would select the best available clustering from all alternatives – which motivates this form of evaluation – the number of solutions should be rather small and was restricted to 5 in this setting.

5 Conclusion

Structuring media collections is one of the most important tasks for current and future media organization applications. Clustering is a basic technique for this problem. A correct or optimal clustering of items depends strongly on intentions and preferences of the user. An important challenge for new clustering techniques is the question of how to integrate clusterings provided by other users in a way that allows for a certain personalization which reflects the locality of the data and preserves user created clusterings. In contrast to other cluster ensemble methods or distributed clustering, a global model (consensus) is not the aim.

Investigating the practical needs carefully has led to the definition of a new learning task, namely learning localized alternative cluster ensembles, where a set of given clustering is taken into account and a set of proposed clusterings is delivered. We have formalized the learning task and developed a greedy approach solving it. Enhancements for hierarchical structures accomplish the LACE algorithm. It is well suited for distributed settings.

The performance of algorithms solving the localized alternative cluster ensembles task can be measured by a leave-one-structuring-out approach. The proposed algorithm outperforms standard clustering schemes on a real-world data set in the domain of music collections. We also investigated the influence of the number of representative points and the influence of response set cardinality which are important in distributed scenarios.

In our opinion, applications in the Web 2.0 context offer many interesting opportunities for machine learning. LACE is a very promising approach to overcome some of the problems associated with this new kind of applications.

References

1. Cohn, D., Caruana, R., McCallum, A.: Semi-supervised clustering with user feedback. Technical Report TR2003-1892, Cornell University (2000)
2. Finley, T., Joachims, T.: Supervised clustering with support vector machines. In: Proc. of the International Conference on Machine Learning. (2005)
3. Gondek, D., Hofmann, T.: Non-redundant data clustering. In: Proc. of the International Conference on Data Mining. (2004)
4. Hotho, A., Staab, S., Stumme, G.: Ontologies improve text document clustering. In: Proc. of the International Conference on Data Mining. (2003) 541–544
5. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: Proc. of the International Conference on Machine Learning. (2001)
6. Strehl, A., Ghosh, J.: Cluster ensembles – a knowledge reuse framework for combining partitionings. In: Proc. of AAAI 2002, Edmonton, Canada. (2002)
7. Datta, S., Bhaduri, K., Giannella, C., Wolff, R., Kargupta, H.: Distributed data mining in peer-to-peer networks. IEEE Internet Computing, special issue on distributed data mining (2005)
8. Guha, S., Rastogi, R., Shim, K.: CURE: an efficient clustering algorithm for large databases. In: Proc. of ACM SIGMOD International Conference on Management of Data. (1998) 73–84
9. Homburg, H., Mierswa, I., Möller, B., Morik, K., Wurst, M.: A benchmark dataset for audio classification and clustering. In: Proc. of the International Symposium on Music Information Retrieval. (2005)
10. Fischer, S., Klinkenberg, R., Mierswa, I., Ritthoff, O.: Yale: Yet Another Learning Environment – Tutorial. Technical Report CI-136/02, Collaborative Research Center 531, University of Dortmund, Dortmund, Germany (2002)
11. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: Proc. of the KDD Workshop on Text Mining. (2000)
12. Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means: spectral clustering and normalized cuts. In: Proc. of the conference on Knowledge Discovery and Data Mining. (2004)
13. Moerchen, F., Ultsch, A., Thies, M., Loehken, I., Noecker, M.and Stamm, C., Efthymiou, N., Kuemmerer, M.: Musicminer: Visualizing perceptual distances of music as topograpical maps. Technical report, Dept. of Mathematics and Computer Science, University of Marburg, Germany (2004)

Distributional Features for Text Categorization

Xiao-Bing Xue and Zhi-Hua Zhou

National Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210093, China
{xuexb, zhouzh}@lamda.nju.edu.cn

Abstract. In previous research of text categorization, a word is usually described by features which express that whether the word appears in the document or how frequently the word appears. Although these features are useful, they have not fully expressed the information contained in the document. In this paper, the *distributional features* are used to describe a word, which express the distribution of a word in a document. In detail, the *compactness* of the appearances of the word and the *position of the first appearance* of the word are characterized as features. These features are exploited by a TFIDF style equation in this paper. Experiments show that the distributional features are useful for text categorization. In contrast to using the traditional term frequency features solely, including the distributional features requires only a little additional cost, while the categorization performance can be significantly improved.

1 Introduction

In the last ten years, content-based document management tasks have gained a prominent status in the information system field, due to the increased availability of documents in digital form and the ensuring need to access them in flexible ways [14]. Among such tasks, text categorization has attracted more and more attention due to its wide applicability. Many classifiers widely used in Machine Learning community have been applied to this task, such as Naïve Bayes, Decision Tree, Neural Network and k-Nearest Neighbor (kNN). Recently, some excellent results have been obtained by SVM [6] and AdaBoost [13].

While a wide range of classifiers have been used, virtually all of them were based on the same text representation, 'bag of words', where a document is represented as a set of words appearing in this document. Features used to describe a word are usually the ones which express whether the word appears in a document or how frequently this word appears. Are these features enough?

Considering the following example, 'Here you are' and 'You are here' are two sentences corresponding to the same vector using the above features, but their meanings are totally different. Although this is a somewhat extreme example, it clearly illustrates that besides the appearance and the frequency of appearance of a word, the distribution of a word is also important. Therefore, this paper attempts to design some *distributional features* to measure the characteristics of a word's distribution in a document.

The first consideration is the *compactness of the appearances* of a word. Here, the 'compactness' measures the extent that the appearances of a word concentrate. A word is compact if its appearances concentrate in a specific part of a document, and less compact if its appearances spread over the whole document. This consideration is motivated by the following facts. A document usually contains several parts. If the appearances of a word are less compact, the word is more likely to appear in different parts and more likely to be related to the theme of the document. For example, consider Document A (NEWID=2367) and Document B (NEWID=7154) in Reuters-21578. Document A talks about the debate on whether expanding the 0/92 programme or just limiting this programme on wheat. Obviously, this document belongs to the category 'wheat'. Document B talks about the U.S. Agriculture Department's proposal on tighter federal standards about insect infections in grain shipments and this document belongs to the category 'grain' but not to the category 'wheat'. Let's consider the importance of the word 'wheat' in both documents. Since the content of A is more closely related to wheat than B, the importance of the word 'wheat' should be higher in A. However, the normalized frequency of this word is almost the same in both documents. Therefore, the frequency is not enough to distinguish this difference of importance. Here, the compactness of the appearances of a word can provide a different view. In A, since the document mostly discusses the 0/92 programme on wheat, the word 'wheat' appears in different parts of this document. In B, since the document mainly discusses the contents of the new standard on grain shipment and just one part of the new standard refers to wheat, the word 'wheat' only appears in one paragraph of this document. So the compactness of the appearances of the word 'wheat' is lower in A than in B, which well expresses the importance of this word.

The second consideration is the *position of the first appearance* of a word. This consideration is based on an intuition that the author naturally mentions the important contents in the earlier parts of a document. Let's consider Document A (NEWID=3981) and Document B (NEWID=4679) in Reuters-21578. Document A belongs to the category 'grain' and talks about the heavy rain in Argentine grain area. Document B belongs to the category 'cotton' and discusses that China is trying to increase cotton output. Obviously, the word 'grain' should be more important in A than in B. Unfortunately, the frequency of the word 'grain' is even lower in A. Now, let's consider the position of the first appearance of the word 'grain'. In A, it firstly appears in the title. It's not strange, because this document is mostly about Argentine grain area. In B , the word 'grain' firstly appears at the end of the document. It's not strange either. Since the theme of this document is about increasing cotton output, the suggestion that the production of cotton be coordinated with other crops such as grain is indirectly related to this theme, so the author naturally mentioned this suggestion at the end of the document. Obviously, the position of the first appearance of a word could express the importance of this word to some extent.

Above all, while the frequency of a word expresses the intuition that *the more frequent, the more important*, the compactness of the appearances of a word shows that *the less compact, the more important* and the position of the first appearance of a word shows that *the earlier, the more important*. In order to test the effect of these distributional features, kNN and SVM are used. Experiments suggest that the distributional features are useful for text categorization.

This paper designs some distributional features for text categorization, which can help improve the performance while requiring only a little additional cost. The paper also explores how to use these distributional features and discusses that when these features are greatly helpful.

The rest of this paper is organized as follows. Section 2 briefly introduces some related works. Section 3 describes how to extract the distributional features. Section 4 discusses how to utilize these features. Section 5 reports on the experiments. Finally, Section 6 concludes.

2 Related Work

This section focuses on the features of a word used in previous text categorization work. A thorough review of text categorization can be found in [14].

Moschitti and Basili [9] has studied the effect of two kinds of linguistic features, POS-tag and word senses. The POS-tag describes the part of speech of each word, which includes verb, noun, pronoun, adjective and so on. A word's POS-tag is identified through Brill tagger. The word senses describe the meanings of a word. For example, consider the word 'bass', its two senses are: a type of fish; tones of low frequency. For a given word, its most appropriate sense is chosen from the possible senses in WordNet through some Word Sense Disambiguation (WSD) algorithms. In general, the improvement of performance brought by these linguistic features is not significant, especially when the cost of getting such features is considered.

Recently, Sauban and Phahringer [12] proposed a new text representation method. In their work, a discriminative score for every word is firstly calculated. Then, with every word input in sequence, a document is shown as a curve depicting the change of the accumulated scores. This curve is called 'Document Profiling'. Two different methods are used to turn a profile into a constant number of features. One is to sample from the profile with a fixed gap and the other is to get some high-level summary information from the profile. Comparable results with the 'bag of words' representation were achieved with lower computational cost. Although no new features are explicitly extracted for a word in this work, the information about the word sequence in a document is utilized.

3 How to Extract Distributional Features

From Section 2, it is noticed that the effect of the distributional features has not been explored in previous researches on text categorization. In this section, the extraction of the distributional features is discussed.

Firstly, a document is divided into several parts. Then, the distribution of a word could be modelled as an array where each element records the number of appearances of this word in the corresponding part. The length of this array is the total number of the parts.

For the above distributional model, what is a part is a basic problem. As mentioned by Callan [3], there are three types of passages used in information retrieval. Here, the meaning of 'passage' is the same as 'part' which is defined as any sequence of text from a document. *Discourse Passage* is based on logic components of documents such as sentences and paragraphs, *semantic passage* corresponds to a topic or subtopic and *window passage* is simply a sequence of words. Considering efficiency, the semantic passage is not used. Compared with window passage, discourse passage is more accurate. Furthermore, sentence is more consistent in length than paragraph. Thus, a sentence is used as a part in this work. For example, for a document d with 10 sentences, the distribution of the word 'corn' is depicted as Fig. 1, then the distributional array for 'corn' is [2,1,0,0,1,0,0,3,0,1].

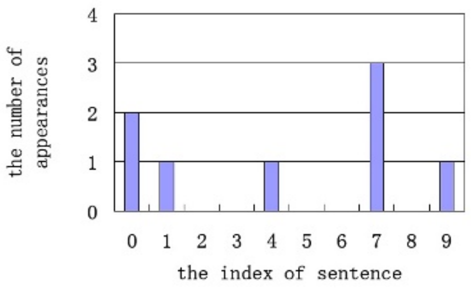


Fig. 1. The distribution of 'corn'

In order to measure the compactness of the appearances of a word, the mean distance between all appearances of this word and the centroid of this word is used. The centroid of a word records the mean position of all appearances of this word. If a word appears in a given sentence, then the position of this appearance is the index of the sentence. The position of the first appearance of a word can be calculated similarly.

Suppose in a document d containing n sentences, the distributional array of the word t is $array(t, d) = [c_0, c_1, ..., c_{n-1}]$. Then, the compactness ($ComPact$) of the appearances of the word t and the position of the first appearance ($FirstApp$) of the word t are defined, respectively, as follows:

$$count(t,d) = \sum_{i=0}^{n-1} c_i \qquad centroid(t,d) = \frac{\sum_{i=0}^{n-1} c_i \times i}{count(t,d)}$$

$$ComPact(t,d) = \frac{\sum_{i=0}^{n-1} c_i \times |i - centroid(t,d)|}{count(t,d)} \tag{1}$$

$$FirstApp(t,d) = \min_{i \in \{0..n-1\}} c_i > 0 ? i : n \tag{2}$$

In Eq. 2, $exp = a?b : c$ means if the condition a is satisfied, then the value of expression exp is b, otherwise the value is c.

The example in Fig. 1 is used again to illustrate how to calculate the distributional features.

$$\begin{aligned}
count(\text{‘corn’}, d) &= 2 + 1 + 1 + 3 + 1 = 8\\
centroid(\text{‘corn’}, d) &= (2 \times 0 + 1 \times 1 + 1 \times 4 + 3 \times 7 + 1 \times 9)/8 = 4.375\\
ComPact(\text{‘corn’}, d) &= (2 \times 4.375 + 1 \times 3.375 + 1 \times 0.375 + 3 \times 2.625\\
&\quad +1 \times 4.625)/8 = 3.125\\
FirstApp(\text{‘corn’}, d) &= min\{0, 1, 10, 10, 4, 10, 10, 7, 10, 9\} = 0
\end{aligned}$$

Then, let's compare the cost of extracting the distributional features and that of extracting only term frequency. Suppose the size of the longest document in corpus is l, the size of the vocabulary is m, the biggest number of sentences a document contains is n and the number of documents in corpus is s. A memory block with size l is required for loading a document and an $m \times 1$ array is required for recording the number of appearances of each word in the vocabulary. When the scan of a document is completed, the term frequency can be directly obtained from the above array. In order to extract the distributional features, an additional $m \times n$ array is needed, since for each word, an $n \times 1$ array is used to record the distribution of this word. When the scan of a document is completed, Eq. 1 and Eq. 2 are used to calculate the distributional features. No other additional cost is needed compared with extracting the term frequency. Overall, the additional computational cost for extracting the distributional features is $s \times m \times$ (Cost of Eq. 1+Cost of Eq. 2) and the additional storage cost is $m \times n$. It is worth noting that the above additional computational cost is the worst case, since practically the calculation is only required for words that appear at least once in a document. Actually, the number of such words of a document is significantly smaller than m. Thus, the additional computational and storage cost for extracting the distributional features is not big. The process of extracting term frequency and the distributional features is illustrated in Fig. 2.

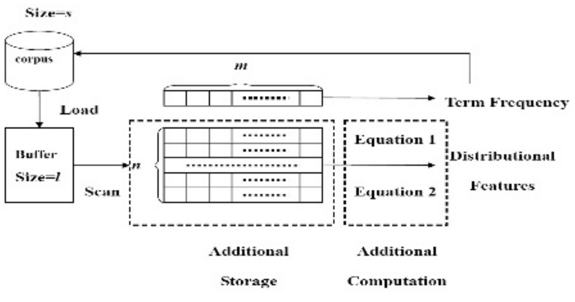


Fig. 2. The process of extracting term frequency and distributional features

4 How to Utilize Distributional Features

Term Frequency in TFIDF could be regarded as a value that measures the importance of a word in a document. As discussed in Section 1, the importance of a word not only can be measured by its term frequency, but also can be measured by the compactness of its appearances and the position of its first appearance. Therefore, the standard TFIDF equation can be generalized as follows:

$$tfidf(t,d) = Importance(t,d) \times IDF(t) \tag{3}$$

When different features are involved, $Importance(t,d)$ corresponds to different values. When the feature is the frequency of a word, TermFrequency (TF) is used. When the feature is the compactness of the appearances of a word, ComPactness (CP) is used. When the feature is the position of the first appearance of a word, FirstAppearance (FA) is used. TF, CP and FA are calculated as follows:

$$TF(t,d) = \frac{count(t,d)}{size(d)} \tag{4}$$

$$CP(t,d) = \frac{ComPact(t,d)+1}{len(d)} \tag{5}$$

$$FA(t,d) = 1 - \frac{FirstApp(t,d)}{len(d)} \tag{6}$$

$size(d)$ in Eq. 4 is the total number of words of Document d. $len(d)$ in Eqs. 5 and 6 is the total number of sentences of Document d. In Eq. 5, $ComPact(t,d)$ is added by 1 in order to ensure $CP(t,d) > 0$. In Eq. 6, the position of the first appearance of word t is subtracted from 1 to reflect the intuition that the earlier a word appears, the more important this word is. Actually, FA value assumes different importance for different positions of a document. Fig. 3 shows the FA value of a word when it firstly appears in different sentences in a document containing 10 sentences. In this paper, this importance is simply assumed to decrease linearly with the index of sentence. Notice that $len(d)$ in Eq. 6 determines the speed of the decreasing of the FA value. The smaller the value of $len(d)$, the faster the speed of decreasing. For short documents, this property contradicts the intuition that the importance of words in short documents differs slightly with the change of position. So a heuristic rule is used here. When the number of sentences in a document is less than 10, $len(d)$ is set to 10, otherwise $len(d)$ is set to the actual number of sentences. Finally, if a word t doesn't appear in Document d, $Importance(t,d)$ is set to 0, no matter which feature is used.

Since TF, CP and FA measure the importance of a word from different views, the combination of them may improve the performance. The strategy used here is to exploit the ensemble learning technique [5]. A group of classifiers are trained based on different features. The label of a new document is decided by the combination of the outputs of these classifiers. Note that the outputs of each classifier are the confidence scores which approximately indicate the probabilities that this new document belongs to each category.

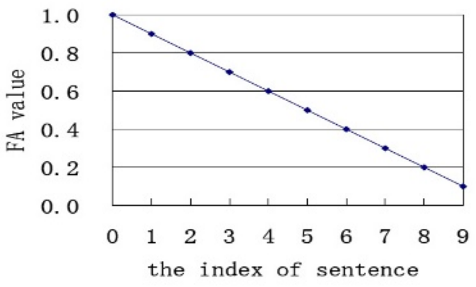


Fig. 3. FA value when a word firstly appears in different sentences

5 Experiments

In this section, the effect of the distributional features is explored for kNN and SVM on three datasets: Reuters-21578, 20 Newsgroup and WebKB. On these three datasets, a lot of work has been published [1][2][6][11][15].

5.1 Datasets

The Reuters-21578 corpus [8] contains 21578 articles taken from Reuters newswire. The 'ModeApte' split is used. Following Yang and Liu [15], 90 categories which have at least one document in both training set and test set are extracted.

The 20 Newsgroup corpus [7] contains 19997 articles taken from the Usenet newsgroup collections. Following Schapire and Singer [13], the duplicate documents are removed and the documents with multiple labels are detected both using the 'Xrefs' header. There are 19465 documents left. Four-fold cross validation is executed.

The WebKB corpus [4] is a collection of 8282 web pages obtained from four academic domains. Following Nigam [10], four categories: *course*, *faculty*, *project*, *student* are used and this part of corpus contains 4199 documents. Four-fold cross validation is executed. Since this corpus consists of web pages, it is difficult to accurately extract sentence from each document as on Reuters-21578 and 20 Newsgroup. Therefore, as mentioned in Section 3, window passage is used on this corpus. Empirically, 20 words are used as the window size.

5.2 Performance Measure

Reuters-21578 and 20 Newsgroup are multi-label datasets. For evaluating the performance on these two corpus, the standard precision, recall and F1 measure is used. Given the contingency table of category C_i (Table 1), the precision(p_i), recall(r_i) and F1 measure($F1_i$) of category C_i is calculated as follows.

$$p_i = \frac{TP_i}{TP_i + FP_i} \qquad r_i = \frac{TP_i}{TP_i + FN_i} \qquad F1_i = \frac{2 \times p_i \times r_i}{(p_i + r_i)}$$

Table 1. The contingency table for category C_i

Category C_i		Expert Judgement	
		Yes	No
Classifier Judgement	Yes	TP_i	FP_i
	No	FN_i	TN_i

These measures can be aggregated over all categories in two ways. One is to average each category's precision, recall and F1 to get the global precision, recall and F1. This method is called *macro-averaging*. The other is based on the global contingency table (Table 2), which is called *micro-averaging*. Macro-averaging is more affected by the classifier's performance on rare categories while micro-averaging is more affected by performance on common categories. In this paper, micro-F1 and macro-F1 are both reported. WebKB is a uni-label dataset, and therefore *accuracy* is used for evaluating performance on this dataset.

Table 2. The global contingency table

Category set $C = C_1, C_2, ..., C_{\|C\|}$		Expert Judgement	
		Yes	No
Classifier Judgement	Yes	$TP = \sum_{i=1}^{\|C\|} TP_i$	$FP = \sum_{i=1}^{\|C\|} FP_i$
	No	$FN = \sum_{i=1}^{\|C\|} FN_i$	$TN = \sum_{i=1}^{\|C\|} TN_i$

In order to explore the effect of the distributional features, the traditional TFIDF is used as baseline. First, CP and FA are respectively used as the importance measure. Second, three combinations of any two features are tested. Finally, the result of the combination of all three features is reported.

5.3 Results

Table 3 shows results of kNN and SVM on three datasets where the best performance is boldfaced. Here, *All* means the combination of three features, i.e. TF+FA+CP. The first question is: are the distributional features useful for text categorization?

On Reuters-21578, distributional features behaves well for micro-F1. The results are similar for SVM and kNN. FA is slightly inferior to TF while CP is slightly better than TF. When different combinations are tried, the combined results are always better than each component. At last, the best result is achieved by TF+FA+CP. For macro-F1, distributional features failed to show any improvement for kNN while CP significantly improves the baseline for SVM.

On 20 Newsgroup, distributional features significantly improve the baseline result for micro-F1. For kNN, FA and CP significantly improve the baseline result. For different combinations, no combination significantly further improves the result of FA except the combination of FA and CP. The best result is achieved

Table 3. Results of *k*NN and SVM on three datasets

	*k*NN					SVM				
	Reu		New		Web	Reu		New		Web
	miF1	maF1	miF1	maF1	acc	miF1	maF1	miF1	maF1	acc
TF	0.844	**0.495**	0.815	0.816	0.808	0.857	0.509	0.887	0.886	0.916
FA	-0.7%	-12.4%	6.7%	6.6%	**5.2**%	-0.4	-2.1%	1.5%	1.6%	**3.1**%
CP	0.5%	-1.9%	3.9%	3.8%	0.9%	0.5%	**2.7**%	0.0%	0.0%	1.4%
TF+FA	1.2%	-4.9%	5.7%	5.6%	3.9%	1.1%	0.0%	**2.0**%	**2.0**%	2.9%
TF+CP	1.0%	-1.5%	3.2%	3.1%	1.4%	0.9%	2.1%	0.7%	0.7%	1.5%
FA+CP	0.7%	-7.6%	**6.9**%	**6.8**%	3.9%	0.8%	1.5%	1.7%	1.7%	3.0%
All	**1.8**%	-6.2%	6.0%	5.9%	3.5%	**1.3**%	1.9%	1.7%	1.7%	2.8%

Table 4. Statistical significance test of *k*NN and SVM

	*k*NN										SVM									
	Reu				New				Web		Reu				New				Web	
vs. TF	s	S	T	T'	s	S	T	T'	s	p	s	S	T	T'	s	S	T	T'	s	p
FA	$<$	$\ll$	$\ll$	$\sim$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\sim$	$\sim$	$\sim$	$>$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$
CP	$\sim$	$\sim$	$\sim$	$\sim$	$\gg$	$\gg$	$\gg$	$\gg$	$\sim$	$\sim$	$\sim$	$>$	$\sim$	$\gg$	$\sim$	$\sim$	$\sim$	$\sim$	$\gg$	$>$
TF+FA	$\gg$	$\sim$	$\sim$	$\sim$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$>$	$\sim$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$
TF+CP	$\gg$	$\sim$	$\sim$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\sim$	$\gg$	$>$	$\sim$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$
FA+CP	$\sim$	$<$	$\sim$	$\sim$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$>$	$>$	$\sim$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$
All	$\gg$	$\sim$	$\sim$	$\sim$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$>$	$\sim$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$

by FA+CP. For SVM, FA performs better than TF while CP shows no improvement for TF. When different combinations are tried, the combined results are better than each component. The best result is achieved by TF+FA. The result of macro-F1 is almost the same as micro-F1, since each category is almost equally distributed on this corpus.

On WebKB, distributional features also significantly improve the baseline. The results of SVM and *k*NN are similar. FA significantly improves the basline on this corpus, and CP slightly improves the baseline. When different combinations are tried, no combination can further improve the result of FA. The best result is achieved by FA.

Statistical significance tests including s-test, p-test, S-test, T-test and T'-test are conducted on the results reported in Table 3. The s-test and p-test were designed to evaluate the performance at a micro level and the S-test, T-test and T'-test were designed to evaluate the performance at a macro level. Further information on these tests can be found in [15]. Note that for each corpus the types of significance tests conducted are determined by the performance measure used on this corpus. The results are summarized in Table 4, where '$A \gg B$' implies that the performance with A is significantly better than B at 0.01 significance level, '$A > B$' implies that the performance with A is significantly better than B at 0.05 significance level, and '$A \sim B$' implies that the performances of A and B are comparable at 0.05 significance level. In general, it is clear that the distributional features are helpful in text categorization.

Table 5. Results of *k*NN on Short and Long datasets

	ReuS		NewS		WebS	ReuL		NewL		WebL
	miF1	maF1	miF1	maF1	acc	miF1	maF1	miF1	maF1	acc
TF	0.880	0.467	0.767	0.774	0.817	0.680	0.291	0.817	0.805	0.752
FA	-0.8%	-8.9%	**7.7**%	**7.0**%	**2.9**%	-1.5%	-9.7%	**7.0**%	**7.2**%	**12.2%**
CP	0.2%	-2.1%	5.4%	4.9%	0.2%	-0.9%	-1.6%	2.6%	2.6%	4.7%
TF+FA	0.6%	**2.5**%	6.1%	5.8%	2.1%	0.3%	-0.8%	4.8%	4.9%	7.6%
TF+CP	0.5%	0.0%	3.7%	3.4%	0.5%	1.0%	3.9%	2.5%	2.8%	4.2%
FA+CP	0.0%	-5.5%	7.6%	**7.0**%	1.4%	**2.2**%	**4.3**%	6.4%	6.4%	10.5%
All	**1.1**%	2.4%	6.6%	6.2%	1.7%	-0.1%	3.2%	5.3%	5.5%	8.3%

Table 6. Results of SVM on Short and Long datasets

	ReuS		NewS		WebS	ReuL		NewL		WebL
	miF1	maF1	miF1	maF1	acc	miF1	maF1	miF1	maF1	acc
TF	0.895	0.498	0.845	0.850	0.916	0.673	0.333	0.904	0.896	0.890
FA	-0.7%	-0.7%	**2.4**%	**2.5**%	**2.8**%	1.2%	-6.8%	1.8%	2.0%	**4.1**%
CP	-0.1%	**1.4**%	1.0%	1.1%	0.9%	3.4%	-0.2%	-0.5%	-0.4%	-1.1%
TF+FA	-0.1%	-0.7%	**2.4**%	2.4%	2.3%	3.1%	-1.2%	**2.0**%	**2.1**%	4.0%
TF+CP	**0.1**%	0.3%	1.0%	1.0%	1.2%	4.2%	**6.0**%	0.7%	0.8%	0.7%
FA+CP	-0.3%	0.2%	**2.4**%	2.4%	2.6%	**4.8**%	-0.8%	1.9%	2.0%	3.8%
All	-0.1%	1.0%	2.0%	2.0%	2.2%	3.3%	-2.2%	1.9%	**2.1**%	3.5%

Furthermore, note that when the distributional features are introduced, there is slight improvement on Reuters-21578 but significant improvement on 20 Newsgroup and WebKB. Therefore, the second question arises: when are the distributional features greatly useful?

As mentioned before, when the compactness of the appearances of a word is introduced, it is assumed that a document contains several parts and the word only appears in a part is not closely related to the theme of the document. When the position of the first appearance of a word is introduced, it is assumed that the word mentioned late by the author is not closely related to the theme of the document. Intuitively, these two assumptions are more likely to be satisfied when a document is long enough. This intuition is based on human's habit of writing. When the length of a document is limited, the author will concentrate on the most related content, such as when writing the abstract section in an academic paper. When there is no limit for the length, the author may write some indirectly related content, such as when writing the body of a paper. In order to verify this intuition, the mean length of documents from these three experimental corpora is reported. Here, the length of a document is measured by its number of sentences. The average length of documents is 6.99, 13.66 and 14.11 respectively for Reuters-21578, WebKB and 20 Newsgroup. It seems that the improvement brought by the distributional features is closely related to the mean length of documents. In order to further verify this idea, each of the three corpora is split into two new corpora, i.e. the Short corpus and the Long corpus, according to the length of documents. For each corpus, the Short corpus contains documents with length no more than 10 and the Long corpus contains documents with length more than 10. Experiments are repeated for these six new generated datasets. The results of kNN on Short and Long datasets are reported in Table 5. The results of SVM on Short and Long datasets are reported in Table 6.

According to Tables 5 and 6, the distributional features brought more significant improvement on the Long dataset than on the Short dataset of Reuters-21578 and WebKB. Comparable improvements are achieved on the Short and Long datasets of 20 Newsgroup. In general, the effect of the distributional features is more obvious on the Long datasets than on the Short ones.

However, the differences of the improvement brought by the distributional features still exist among three corpora in Tables 5 and 6. The improvement is more significant on 20 Newsgroup and WebKB than on Reuters-21578 in most situation. It seems that there are other factors that also contribute to the performance of the distributional features. Note that the sources of three corpora are different, the documents in Reuters-21578 are taken from news reports, the documents in 20 Newsgroup are taken from newsgroup documents and the documents in WebKB are taken from web pages. For news reports, they are written by professional journalists and editors and the writing style is formal and precise, therefore the loosely related content is less likely to appear in this type of articles. In contrast, for newsgroup documents and web pages, they are written by ordinary web users and the writing style is very causal, therefore the loosely related content is more likely to appear in this type of articles. Thus, it seems that the effect of the distributional features is more obvious for informal documents than for formal ones.

Therefore, the answer to the second question, i.e. when the distributional features are greatly useful, is: when the documents are long enough and when the documents are informal.

6 Conclusion

Previous researches on text categorization usually use the features of appearance or the frequency of appearance to characterize a word. These features are not enough for fully capturing the information contained in a document. In this paper, the distributional features of a word are explored. These features encode a word's distribution from some aspects. In detail, the compactness of the appearances of a word and the position of the first appearance of a word are used. A TFIDF style equation is constructed to utilize these distributional features. Experiments show that the distributional features are useful for text categorization, especially when they are combined with term frequency. Further analysis reveals that the effect of the distributional features is obvious when the documents are long enough and when the documents are informal.

It is noticed that the task on WebKB is somewhat genre-based while the tasks on Reuters-21578 and 20 Newsgroup are topic-based. Intuitively, it is convincing that the distributional features may bring more benefits on genre-based corpus than on topic-based corpus. This supposition will be explored in the future.

In this paper, Eqs.1 and 2 are mainly designed for validating the usefulness of distributional features. More careful designs are anticipated to improve the performance, e.g. an alternative to Eq. 1 may work well on more peaks, which is an interesting work in the future. How to design an alternative feature to IDF in Eq. 3 specifically to work with the proposed distributional features is another interesting future issue.

Acknowledgment. Supported by the NSFC (60505013, 60325207) and JiangsuSF (BK2005412).

References

1. L. D. Baker and A. K. McCallum. Distributional clustering of words for text classification. In Proceedings of SIGIR-98, Melbourne, Australia, 1998. 96-103.
2. R. Bekkerman, R. El-Yaniv, N. Tishby and Y. Winter. Distributional word clusters vs. words for text categorization. Journal of Machine Learning Research, 2003, **3** : 1182-1208.
3. J. P. Callan. Passage retrieval evidence in document retrieval. In Proceedings of SIGIR-94, Dublin, Ireland, 1994. 302-310.
4. M. Craven, D. DiPasquo, D. Freitag, A. K. McCallum, T. M. Mitchell, K. Nigam and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In Proceedings of AAAI-98, Madison, WI, 1998. 509-516.
5. T. G. Dietterich. Machine learning research: Four current directions. AI Magazine, 1997, **18** (4): 97-136.
6. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In Proceedings of ECML-98, Chemnitz, Germany, 1998. 137-142.
7. K. Lang. Newsweeder: Learning to filter netnews. In Proceedings of ICML-95, Tahoe City, CA, 1995. 331-339.
8. D. Lewis. Reuters-21578 text categorization test colleciton, Distrib. 1.0, September 26, 1997.
9. A. Moschitti and R. Basili. Complex linguistic features for text classification: A comprehensive study. In Proceedings of ECIR-04, Sunderland, UK, 2004. 181-196.
10. K. Nigam, A. K. McCallum, S. Thrun and T. M. Mitchell. Learning to classify text from labeled and unlabeled documents. In Proceedings of AAAI-98, Madison, WI, 1998. 792-799.
11. J. Rennie, L. Shih, J. Teevan and D. Karger. Tackling the poor assumptions of Naive Bayes text classifiers. In Proceedings of ICML-03, Washington, DC, 2003. 616-623.
12. M. Sauban and B. Pfahringer. Text categorization using document profiling. In Proceedings of PKDD-03, Cavtat-Dubrovnik, Croatia, 2003. 411-422.
13. R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. Machine Learning, 2000, **39** (2-3): 135-168.
14. F. Sebastiani. Machine learning in automated text categorization. ACM Computing Surverys, 2002, **34** (1): 1-47.
15. Y. Yang and X. Liu. A re-examination of text categorization methods. In Proceedings of SIGIR-99, Berkeley, CA, 1999. 42-49.

Subspace Metric Ensembles for Semi-supervised Clustering of High Dimensional Data

Bojun Yan and Carlotta Domeniconi

Department of Information and Software Engineering
George Mason University
Fairfax, Virginia 22030, USA
byan@gmu.edu, carlotta@ise.gmu.edu

Abstract. A critical problem in clustering research is the definition of a proper metric to measure distances between points. Semi-supervised clustering uses the information provided by the user, usually defined in terms of constraints, to guide the search of clusters. Learning effective metrics using constraints in high dimensional spaces remains an open challenge. This is because the number of parameters to be estimated is quadratic in the number of dimensions, and we seldom have enough side-information to achieve accurate estimates. In this paper, we address the high dimensionality problem by learning an ensemble of subspace metrics. This is achieved by projecting the data and the constraints in multiple subspaces, and by learning positive semi-definite similarity matrices therein. This methodology allows leveraging the given side-information while solving lower dimensional problems. We demonstrate experimentally using high dimensional data (e.g., microarray data) the superior accuracy achieved by our method with respect to competitive approaches.

1 Introduction

Clustering is the subject of active research in several fields such as statistics, pattern recognition, and machine learning. The clustering problem concerns the discovery of homogeneous groups of data according to a certain similarity measure, such that data in a cluster are more similar to each other than data assigned to different clusters. The definition of a proper similarity measure is a difficult problem that lies at the core of the field of machine learning. The structure of the groups discovered in the data by a given clustering technique strongly depends on the similarity measure used. Data mining adds to clustering the complication of large data sets with high dimensionality. Large amounts of unlabeled data are available in real-life data mining tasks, e.g., unlabeled messages in an automated email classification system, or genes of unknown functions in microarray data. This imposes unique computational requirements on clustering algorithms. Furthermore, the sparsity of the data in high dimensional spaces can severely compromise the ability of discovering meaningful clustering solutions.

Recently, semi-supervised clustering has become a topic of significant research interest. While labeled data are often limited and expensive to generate, in many cases it is relatively easy for the user to provide pairs of similar or dissimilar examples. Semi-supervised clustering uses a small amount of supervised data, usually under the form of pairwise constraints on some instances, to aid unsupervised learning. The main approaches for semi-supervised clustering can be basically categorized into two general methods: constrained-based [22,23,3,4] and metric-based [24,8,2]. The work in [5,6] combines constraints with a distance metric. However, when facing high dimensional data, learning an effective metric with limited supervision remains an open challenge. The reason is that the number of parameters to be estimated is quadratic in the number of dimensions, and we seldom have enough side-information to achieve accurate estimates. For example, in our experiments we deal with microarray data with 4026 dimensions and less that 100 samples (a typical scenario with these kinds of data). Learning a similarity metric with a limited number of constraints becomes a very hard problem under these conditions due to the large parameter space to be searched.

In this paper, we address the high dimensionality problem by learning an ensemble of subspace metrics. This is achieved by projecting the data and the pairwise constraints in multiple subspaces, and by learning positive semi-definite similarity matrices therein [24]. This methodology allows leveraging the given side-information while solving lower dimensional problems. The diverse clusterings discovered within the subspaces are then combined by means of a graph-based consensus function that leverages the common structure shared by the multiple clustering results [9]. Our experimental results show the superior accuracy achieved by our method with respect to competitive approaches, which learn the metric in the full dimensional space.

2 Background

2.1 Distance Metric Learning

In the context of semi-supervised clustering, limited supervision is provided as input. The supervision can have the form of labeled data or pairwise constraints. In many applications it is more realistic to assume that pairwise constraints are available.

Suppose we have a set of points $X = \{\mathbf{x}_i\}_{i=1}^N \subseteq \Re^D$, and a set of pairwise constraints: must-link $ML = \{(\mathbf{x}_i, \mathbf{x}_j)\}$ and cannot-link $CL = \{(\mathbf{x}_i, \mathbf{x}_j)\}$. The goal is to learn a distance metric that brings the must-link points close to each other and moves the cannot-link points far away from each other. Consider learning a distance metric of the form: $d_A(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T A (\mathbf{x}-\mathbf{y})}$. To ensure that d_A is a metric (i.e., satisfies non-negativity and the triangle inequality), A is required to be positive semi-definite, i.e., $A \succeq 0$. In general terms, A parameterizes a family of Mahalanobis distances over $\Re^D$. When $A = I$, d_A gives the standard Euclidean distance; if A is diagonal, learning A corresponds to assigning different *weights* to features. In general, learning such a distance metric is equivalent to finding a transformation of the data that substitutes each point $\mathbf{x}$ with $\sqrt{A}\mathbf{x}$.

The problem of learning A can be formulated as a convex optimization problem [24]: $\min_A \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in ML} \|\mathbf{x}_i - \mathbf{x}_j\|_A^2$, such that $\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in CL} \|\mathbf{x}_i - \mathbf{x}_j\|_A \geq 1$, and $A \succeq 0$. If we restrict A to be diagonal, i.e., $A = diag\{A_{11}, ..., A_{DD}\}$, the problem can be solved using the Newton-Raphson method and by minimizing the following function: $G(A) = G(A_{11}, \cdots, A_{DD}) = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in ML} \|\mathbf{x}_i - \mathbf{x}_j\|_A^2 - \log\left(\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in CL} \|\mathbf{x}_i - \mathbf{x}_j\|_A\right)$. The Newton-Raphson method computes the Hessian matrix (of size $D \times D$) in each iteration to determine the new search direction. When learning a full matrix A, the Newton-Raphson method requires $O(D^6)$ time to invert the Hessian over D^2 parameters. Clearly, in high dimensional spaces this computation becomes prohibitively expensive.

2.2 Cluster Ensembles

In an effort to achieve improved classifier accuracy, extensive research has been conducted in classifier ensembles. Recently, cluster ensembles have emerged. Cluster ensembles offer a solution to challenges inherent to clustering arising from its ill-posed nature. In fact, it is well known that off-the-shelf clustering methods may discover very different structures in a given set of data. This is because each clustering algorithm has its own bias resulting from the optimization of different criteria. Cluster ensembles can provide robust and stable solutions by leveraging the consensus across multiple clustering results, while averaging out spurious structures due to the various biases to which each participating algorithm is tuned. In the following we formally define the clustering ensemble problem.

Consider a set of data $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$. A clustering ensemble is a collection of S clustering solutions: $C = \{C_1, C_2, \cdots, C_S\}$. Each clustering solution C_l, for $l = 1, \cdots, S$, is a partition of the set X, i.e. $C_l = \{C_l^1, C_l^2, \cdots, C_l^{K_l}\}$, where $\bigcup_K C_l^K = X$. Given a collection of clustering solutions C and the desired number of clusters K, the objective is to combine the different clustering solutions and compute a new partition of X into K disjoint clusters.

Different methods have been introduced in the literature to solve the clustering ensemble problem. The techniques presented in [10] compute a matrix of similarities between pairs of points, and then perform agglomerative clustering to generate a final clustering solution. In [20,21] the authors introduce new features to describe the data, and apply K-means and EM to output the final clustering solutions. Recently, several approaches have modeled the clustering ensemble problem as a graph partitioning problem [17,21]. In the following, we provide the necessary definitions of graph partitioning.

A graph partitioning problem takes in input a weighted graph G and an integer K. A weighted graph G is defined as a pair $G = (V, E)$, where V is a set of vertices and E is a $|V| \times |V|$ similarity matrix. Each element E_{ij} of E captures the similarity between vertices V_i and V_j, with $E_{ij} = E_{ji}$ and $E_{ij} \geq 0 \; \forall \; i, j$. Given G and K, the problem of partitioning G into K subgraphs consists in computing a partition of V into K groups of vertices $V = \{V_1, V_2, \cdots, V_K\}$. The sum of the weights (i.e., similarity values) of the crossed edges is defined as the cut of the partition. In general, we want to find a K-way partition that minimizes the cut. In [9], the authors propose a method (Hybrid-Bipartite-Graph-Formulation, or

HBGF), which considers both the similarity of instances and the similarity of clusters when producing the final clustering solution. Specifically, given a cluster ensemble $C = \{C_1, C_2, \cdots, C_S\}$, HBGF constructs a bipartite graph $G = (V, E)$ as follows. $V = V^C \cup V^I$, where each vertex in V^C represents a cluster of the ensemble C, and V^I contains N vertices each representing an instance of the data set X. If both vertices i and j represent clusters or instances, $E_{ij} = 0$; otherwise, if instance i belongs to cluster j, $E_{ij} = E_{ji} = 1$, and 0 otherwise. The authors in [9] use a multi-way spectral graph partitioning algorithm [15] to find a K-way partition of the resulting bipartite graph.

3 Subspace Metric Cluster Ensemble Algorithm

A limited number of pairwise constraints may not be effective for learning a distance metric in high dimensional spaces due to the large parameter space to be searched. We tackle this issue by reducing the given high dimensional problem with fixed supervision into a number of *smaller* problems, for which the dimensionality is reduced while the amount of supervision is unchanged. To achieve this goal, we utilize and leverage the paradigm of learning with ensembles. It is well known that the effectiveness of an ensemble of learners depends on both the accuracy and diversity of the individual components [12]. A good accuracy-diversity trade-off must be achieved to obtain a consensus solution that is superior to the components. Our method generates accurate learners by assigning each of them a problem of lower dimensionality, and, at the same time, by providing each of them the entire amount of constraints. Furthermore, diversity is guaranteed by providing the learners different *views* (or projections) of the data. Since such views are generated randomly from a (typically) large pool of dimensions, it is highly likely that each learner receives a different perspective of the data, which leads to the discovery of diverse (and complementary) structures within the data. The experimental results presented in this paper corroborate the motivation behind our approach. The details of our subspace metric ensemble algorithm follow.

We are given a set X of data in the D dimensional space, a set of must-link constraints ML, and a set of cannot-link constraints CL. We assume that the desired number of clusters to be discovered in X is fixed to K. We reduce a D dimensional semi-supervised clustering problem into a number (S) of semi-supervised clustering problems of reduced dimensionality F. To this end we draw S random samples of F features from the original D dimensional feature space. Moreover, for each must-link constraint $(\mathbf{x}_i, \mathbf{x}_j) \in ML$, we generate the projected must-link constraints $(\mathbf{x}_i, \mathbf{x}_j)_{F_l}$, for $l = 1, \cdots, S$. This gives new S sets of must-link constraints: $ML_{F_1}, \cdots, ML_{F_S}$. Similarly, for each cannot-link constraint $(\mathbf{x}_i, \mathbf{x}_j) \in CL$, we generate the projected cannot-link constraints $(\mathbf{x}_i, \mathbf{x}_j)_{F_l}$, for $l = 1, \cdots, S$. This results in S new sets of cannot-link constraints: $CL_{F_1}, \cdots, CL_{F_S}$.

We have now reduced the original problem into S smaller problems, each of dimensionality F, where we can assume $F \ll D$. We proceed by learning

```
Algorithm: Subspace metric cluster ensemble
Input: $X$, $ML$, $CL$, $K$, number of features $F$, ensemble size $S$
Output: Partition of $X$ into $K$ clusters
Method:

  1. Generate $S$ subspaces $F_1$, $F_2$, $\cdots$, $F_S$ by random sampling $F$ features without
     replacement from the $D$-dimensional original space.
  2. For each constraint $(\mathbf{x}_i, \mathbf{x}_j) \in ML$, generate the projected constraints
     $(\mathbf{x}_i, \mathbf{x}_j)_{F_l}$, for $l = 1, \cdots, S$; this gives the new $S$ sets $ML_{F_1}, \cdots, ML_{F_S}$. Like-
     wise, generate new $S$ sets of cannot-link constraints $CL_{F_1}, \cdots, CL_{F_S}$.
  3. Learn matrix $A_l$ in subspace $F_l$, using the corresponding sets of constraints
     $ML_{F_l}$ and $CL_{F_l}$, for $l = 1, \cdots, S$, according to the method presented in [24].
  4. Cluster data $X$ in each subspace $F_l$ with $K$-means, using the metric $d_{A_l}$ and
     the number of clusters fixed to $K$. This gives an ensemble of $S$ clusterings.
  5. Use the HBGF algorithm [9] to construct the bipartite graph $G = (V, E)$ from
     the resulting $S$ clusterings.
  6. Use spectral graph partitioning to obtain the final clustering result.
```

Fig. 1. Subspace Metric Cluster Ensemble Algorithm

the matrices A_l in each subspace F_l, using the corresponding sets of constraints ML_{F_l} and CL_{F_l}, with $l = 1, \cdots, S$, according to the method presented in [24] (as described in Section 2.1). Note that the dimensionality of each matrix A_l is now reduced to $F \times F$. We then cluster the data X in each subspace F_l, with K-means, using the corresponding distance metrics d_{A_l}, for $l = 1, \ldots, S$, and with the number of clusters fixed to K. This gives an ensemble of S clusterings. We finally use the HBGF algorithm [9] to construct the bipartite graph $G = (V, E)$ from the resulting S clusterings as described in Section 2.2, and apply spectral graph partitioning to obtain the final clustering result. The algorithm is summarized in Figure 1. In our experiments, we refer to this algorithm as *KMeans-Metric-S*.

The authors in [24] also provide clustering results obtained by performing *constrained* K-means combined with the learned matrix A. During the assignment of points to clusters, each pair of points in the set of must-link constraints is assigned to the same cluster (only must-link constraints are used in [24] in this case). To compare our subspace approach with the constrained K-means clustering in full space, we also perform constrained K-means using the learned matrices A_l in each of the generated subspaces. In our experiments, we refer to the resulting technique as *KMeans-Metric-S-cst*. We take advantage of both cannot-link and must-link constraints, and compare the clustering results (in full and reduced spaces) under the same conditions.

In order to implement constrained K-means, once we have learned the matrix A, in full space and in each of the subspaces, we rescale the data according to $\mathbf{x} \rightarrow \sqrt{A}\mathbf{x}$, and minimize the objective function [4,5,11]:

$$J_{obj} = \sum_{C_k} \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in ML} w_{ij} 1[C_i \neq C_j] + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in CL} \overline{w}_{ij} 1[C_i = C_j]$$

where C_k, $k = 1, \cdots, K$, indexes the clusters, $\boldsymbol{\mu}_k$ is the centroid of cluster C_k, C_i and C_j are the clusters to which points $\mathbf{x}_i$ and $\mathbf{x}_j$ are assigned, w_{ij} and $\overline{w}_{ij}$ are the constraint violation costs between $\mathbf{x}_i$ and $\mathbf{x}_j$, and $1[\cdot]$ is an indicator function with a value of 1 if its argument is true, and 0 otherwise. w_{ij} and $\overline{w}_{ij}$ are parameters whose values can be provided as part of the supervision, or can be chosen by the user according to the degree of confidence in the constraints. Here, our concern is to perform a fair comparison between the subspace metric learning approach and the metric learning in full space under the same conditions. Thus, following [4,5], we set each w equal to the average distance between pairs of points in the data. Such constant provides a scaling for the constraint violation costs which is comparable to the within-cluster scatter measure (first term in the equation above). The same constraint information is provided to K-means performed in full space and in each of the subspaces. We observe that the approaches in [6,5] can easily be extented to incorporate our concept of subspace metric cluster ensemble.

4 Experimental Design

To demonstrate the effectiveness of our subspace metric cluster ensemble, we consider two microarray data sets which reflect the challenges discussed above: the NC160 [16] and Lymphoma [1] data sets. For the NC160 data set, cDNA microarrays were used to examine the variation in 1155 gene expression values among the 61 cell lines from the National Center Institutes anticancer drug screen. The data set contains 8 classes. To deal with missing values, we deleted genes that have more than 50 missing values, and used the K-nearest neighbor method to impute the remaining missing values. For a gene with missing values, the K nearest neighbors are identified from the subset of genes that have complete expression values ($K = 7$ in our experiments). The average of the neighbors' values is used to substitute a missing value. The Lymphoma data set [1] contains 96 samples from patients, each with 4026 gene expression values. The samples are categorized into 9 classes according to the type of mRNA sample studied. Classes that have less than 5 samples are removed from the experiments, and hence 6 classes and 88 samples remained. We also perform experiments on two UCI data sets: Wine (N=178, D=13, K=3) and Breast-Cancer (N=569, D=30, K=2).

We compare two variants of the proposed subspace cluster ensemble approach: the *KMeans-Metric-S* and the *KMeans-Metric-S-cst* algorithms, as described in Section 3. Constrained K-means is performed by minimizing the objective function provided in Section 3, where the weights of the constraint violation costs are set to the average distance between pairs of points in the data set. We also compare the corresponding variants for metric learning in full feature space, which we call *KMeans-Metric-F* and *KMeans-Metric-F-cst*, respectively, as well as constrained K-Means (*KMeans-cst*), and *K-Means* with no supervision.

For K-Means, KMeans-Metric-F and KMeans-Metric-S, we randomly initialize the clusters, and set the number of clusters K to the actual number of classes in the data. For KMeans-cst, KMeans-Metric-F-cst and KMeans-Metric-S-cst, the clusters are initialized using the approach presented in [5,11]: we take the transitive closure of the constraints to form neighborhoods, and then perform a farthest-first traversal on these neighborhoods to get the K initial clusters. We ensure that the same constraint information is given to each competitive algorithm. For all four methods KMeans-Metric-S, KMeans-metric-F, KMeans-metric-F-cst, and KMeans-metric-F-cst we learn a distance metric d_A with diagonal matrix A.

5 Experimental Results

To evaluate clustering results, we use the Rand Statistic index [19,24,22]. Figures 2-3 show the learning curves using 20 runs of 2-fold cross-validation for each data set (30% for training and 70% for testing). These plots show the improvement in clustering quality on the test set as a function of an increasing amount of pairwise constraints. For studying the effect of constraints in clustering, 30% of the data is randomly drawn as the training set at any particular fold, and the constraints are generated only using the training set. We observe that, since folds are randomly generated, there is no guarantee that all classes are represented within the training data. The clustering algorithm was run on the whole data set, but we calculate the Rand Statistic only on the test set. Each point on the learning curve is an average of results over 20 runs.

We learn the metrics in 120 subspaces separately for the NCI60 and Lymphoma data sets, each of which is produced by randomly selecting 60 features (i.e., $S = 120$ and $F = 60$). For the Wine and Breast-Cancer data sets, the metrics are learned separately in 30 subspaces ($S = 30$), each of which is produced by randomly selecting 8 features ($F = 8$) for Wine data and 10 features ($F = 10$) for Breast-Cancer data.

From Figures 2-3, we can clearly appreciate the benefits of using our subspace metric ensemble techniques. For the two high dimensional data, NCI60 and Lymphoma, when a small number of constraints is used, the KMeans-Metric-S and KMeans-Metric-S-cst algorithms show large clustering quality improvements with respect to the competing techniques. In fact, for these two data sets, the algorithms KMeans-Metric-S and KMeans-Metric-S-cst leverage the given side-information while solving much lower dimensional problems (from 1155 dimensions down to 60, and from 4026 down to 60, respectively).

For the Wine and Breast-Cancer data, the improvement of the clustering quality for KMeans-Metric-S and KMeans-Metric-S-cst is more gradual throughout the increasing of the number of constraints. The dimensionalities of these two data sets is much lower (13 and 30 respectively), and the dimensionalities of the subspaces are 8 and 10, respectively. The overall gap in quality improvement between the subspace metric ensemble methods and the remaining techniques on these data clearly shows the positive effect of using ensembles of clusterings: the accuracy and diversity of the individual clustering solutions allow to achieve an improved consensus clustering that is superior to the component ones. In

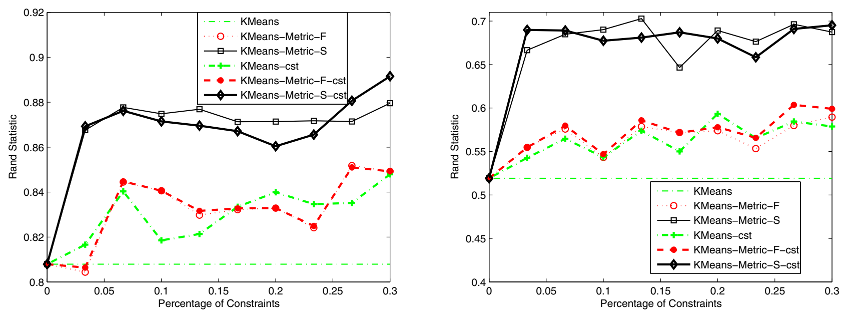


Fig. 2. Clustering results (*left*) on NCI60 data and (*right*) on Lymphoma data

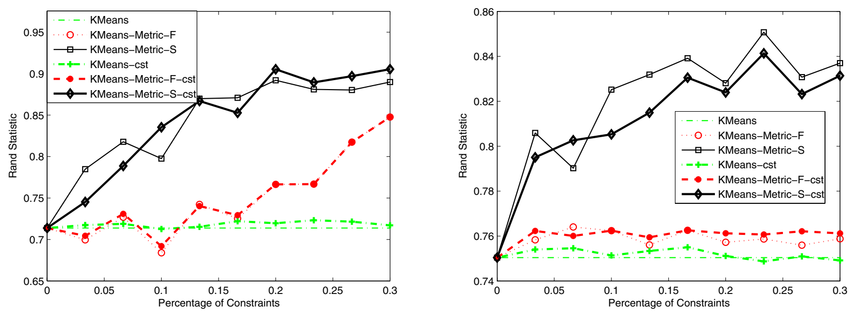


Fig. 3. Clustering results (*left*) on Wine data and (*right*) on Breast-Cancer data

fact, although the dimensionalities of the full space and the subspace do not differ greatly, the ensemble technique is capable of taking good advantage of the increased amount of supervision (this is particularly evident in Figure 3 for the Breast-Cancer data set). We observe that, in general, the trend for the algorithms that operate in full space is rather flat, showing that a limited amount of supervision does not have a high impact in high dimensional spaces.

No significant difference between KMeans-Metric-S and KMeans-Metric-S-cst was observed throughout the tested data sets. The same is true for the corresponding algorithms in full space.

Analysis of Diversity. We investigate here the trade-off between accuracy and diversity achieved by our subspace cluster ensembles. A Kappa-Error diagram [12] allows to visualize the diversity and the accuracy of an ensemble of classifiers. To analyze the diversity-quality trade-off of our subspace metric ensembles, we measure diversity using the *Normalized Mutual Information* (NMI) [18] between each pair of clustering solutions. In addition, we average the two NMI values of the pair, each computed using the ground truth labels. Such average provides a single quality measure for each pair of clustering solutions.

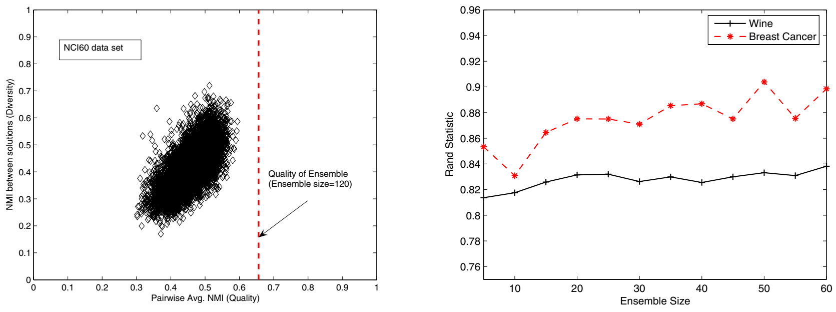


Fig. 4. (*left*) Diversity-Quality on NCI data; (*right*) Quality vs. Ensemble-Size on Wine and Breast-Cancer data

We have plotted the diversity-quality diagrams based on the KMeans-Metric-S algorithm for each of the four data sets considered (the percentage of constraints is fixed to 12%). For lack of space we report here only the diagram for the NCI60 data set (Figure 4 (left)). For every data set, the quality of the ensemble is superior to that of the individual components, proving the effectivess of our subspace cluster ensembles. The vertical dashed line indicates the quality of the final clustering provided by the ensemble (measured in terms of NMI with respect to the underlying class structure). We note that when the NMI between two clustering solutions (shown on the y axis) is zero, the diversity is maximized. On the other hand, when the average NMI of each pair (shown on the x axis) is maximized, their quality is also maximized. Thus, ideally, the points would populate the right-hand bottom corner of the figure. Figure 4 (left) shows that the components of the ensemble have a relatively high diversity and quality, thus they achieve a good trade-off which is reflected in the increased quality of the ensemble with respect to the clustering components. This result emphasizes the fact that, while our technique reduces the curse-of-dimensionality effect, interestingly, it also takes advantage of the high dimensionality of the data to guarantee diversity among the ensemble components. As a consequence, it is most effective with high dimensional data.

Table 1. Running Times (measured in seconds)

Data set	Method	Learn. A	K-Means (1-run)	HBGF	S	T_{total}
NCI60	Full space	8.914	8.527	—	1	17.441
($N = 61, D = 1155, K = 8$)	Subspace	0.086	0.058	0.191	120	17.471
Lymphoma	Full space	277.338	138.586	—	1	415.924
($N = 88, D = 4026, K = 6$)	Subspace	0.136	0.067	0.158	120	24.518
Wine	Full space	0.027	0.045	—	1	0.072
($N = 178, D = 13, K = 3$)	Subspace	0.021	0.033	0.032	30	1.652
Breast-Cancer	Full space	0.038	0.178	—	1	0.216
($N = 569, D = 30, K = 2$)	Subspace	0.025	0.078	0.041	30	3.131

Figure 4 (right) shows the quality of the ensemble as a function of different ensemble sizes for the Wine and Breast-Cancer data sets. Fixing the constraints (12%), each point in the graph is an average of results over 20 runs. While for the NCI60 and Lymphoma data (not included for lack of space) no significant increase in quality is observed within the range of sizes tested, for the Wine and Breast-Cancer data (Figure 4 (right)) we observe an increasing trend in quality as the ensemble size increases. It is reasonable to expect that increasing the ensemble size for the two data sets of lower dimensionality results in a better trade-off between accuracy and diversity of the components.

Time Complexity. The KMeans-Metric-F (or KMeans-Metric-F-cst) technique performs the computation of the distance metric d_A followed by the K-Means clustering. The first step (achieved via the Newton-Raphson method) requires the computation of $O(D^2)$ partial derivatives. The time-complexity of K-Means is $O(NKRD)$ [13], where N is the number of data, K is the number of clusters, R is the number of iterations, and D is the dimensionality of the data. When $D \gg NK$, the most costly step is the computation of the distance metric. The corresponding time complexities for the KMeans-Metric-S (or KMeans-Metric-S-cst) are $O(F^2 \times S)$ for the first step, and $O(N \times K \times r \times F \times S)$ for the second step, where F is the dimensionality of the subspaces (usually $F \ll D$), S is the ensemble size, and r is the number of iterations of K-Means. The subspace ensemble technique includes also the construction of the bipartite graph and the execution of a K-way graph partitioning (using spectral graph partitioning) whose cost is $O((\max\{N, K \times S\})^{3/2}K + rNK^2)$. The first term is due to the computation of K eigenvectors for a $N \times (K(S))$ matrix, and the second term corresponds to the complexity of K-Means in K dimensions. To compare the running times of the two approaches (full space vs. subspace ensemble), we fix the number of constraints and the ensemble size, and record the running times for each phase of the algorithms. We repeat the computation 20 times and report the average running times in Table 1. All the experiments are performed on a Linux machine with 2.8 GHz Pentium IV processor and 1 GB main memory. The total time for the approach in full space is $T_{total} = T_{metriclearning} + T_{KMeans}$; the total time for the subspace ensemble is $T_{total} = (T_{metriclearning} + T_{KMeans}) \times S + T_{HBGF}$.

The proposed subspace ensemble algorithm greatly reduce the running time for the Lymphoma data set. The full space and subspace approaches show comparable running times on the NCI60 data set. The ensemble approach has a larger running time for the Breast-Cancer and Wine data sets, since their dimensionalities are small. We emphasize that the computed running times are based on sequential executions for the ensemble components. Nevertheless, such computations can be easily run in parallel, allowing for further speed-up.

6 Related Work

The authors in [22] proposed the COP-KMeans algorithm, which assigns each point to the closest cluster that minimizes the violation of constraints. If no such cluster exists, it fails to assign the point. In [3], the authors utilized labeled data to initialize the clusters. Constraints could be violated (Seeded-KMeans) in successive iterations, or could be strictly enforced (Constrainted-KMeans) throughout the algorithm. Moreover, [4] proposed the PCKMeans algorithm, which assigns weight parameters to the constraints. The work in [11] applies kernel methods to enable the use of both vector-based and graph-based data for semi-supervised clustering. However, the work in [11] does not learn a distance metric based on pairwise constraints.

In recent work on semi-supervised clustering with pairwise constraints, [8] used gradient descent combined with a weighted Jensen-Shannon divergence in the context of EM clustering. [2] proposed a Redundant Component Analysis (RCA) algorithm that uses must-link constraints to learn a Mahalanobis distance. [24] utilized both must-link and cannot-link constraints to formulate a convex optimization problem which is local-minima-free. [5,6] proposed a method based on Hidden Markov Random Fields (HMRFs) which learns a metric during clustering to minimize an objective function which incorporates the constraints. This is equivalent to the minimization of the posterior energy of the HMRF.

7 Conclusions and Future Work

We have addressed the problem of learning effective metrics for clustering in high dimensional spaces when limited supervision is available. We have proposed an approach based on learning with ensembles that is capable of producing components which are both accurate and diverse. In our future work we will investigate the sensitivity of our approach with respect to the dimensionality of subspaces, and possibly define an heuristic to automatically estimate an “optimal” value for such parameter. Furthermore, we will explore alternative mechanisms to credit weights to features by utilizing the constraints; consequently we will bias the sampling in feature space to favor the estimated most relevant features.

Acknowledgments

This work was in part supported by NSF CAREER Award IIS-0447814. The authors would like to thank Xiaoli Zhang Fern for the code of the HBGF algorithm, and Brian Kulis for the code of the methods described in [11]. We are thankful to Hong Chai for helping us with the processing of the microarray data.

References

1. A. A. Alizadeh, M. B. Eisen, R. E. Davis, and C. Ma et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. Nature, 403, pages 503-511, 2000.
2. A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. *International Conference on Machine Learning*, 2003.

3. S. Basu, A. Banerjee, and R. J. Mooney. Semi-supervised clustering by seeding. *International Conference on Machine Learning*, 2002.
4. S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrainted clustering. *SIAM International conference on Data Mining*, 2004.
5. S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. *International Conference on Knowledge Discovery and Data Mining*, 2004.
6. M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and Metric Learning in semi-supervised clustering. *International Conference on Machine Learning*, 2004.
7. C. L. Blake and C. J. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html, 1998.
8. D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback. TR2003-1892, Cornell University, 2003.
9. X. Z. Fern and C. E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. *International Conference on Machine Learning*, 2004.
10. A. L. N. Fred and A. K. Jain. Data clustering using evidence accumulation. *International Conference on Pattern Recognition*, 2002.
11. B. Kulis, S. Basu, and I. Dhillon, and R. Mooney. Semi-supervised graph clustering: a kernel approach. *International Conference on Machine Learning*, 2005.
12. D. D. Margineantu and T. G. Dietterich. Pruning adaptive boosting. *International Conference on Machine Learning*, 1997.
13. J. McQueen. Some Methods for Classification and Analysis of Multivariate Observation. L. Le Cam and J. Neyman (Eds.), *Berkeley Symposium on Mathematical Statistics and Probability*, pages 281-297, 1967.
14. S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. Machine Learning, 52, pages 91-118, 2003.
15. A Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems 14*, 2002.
16. D. T. Ross, U. Scherf, M. B. Eisen, and C. M. Perou et al. Systematic variation in gene expression patterns in human cancer cell lines. Nature Genetics, 24(3), pages 227-235, 2000.
17. A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. Machine Learning Research, 3, pages 583-417, 2002.
18. A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. *AAAI Workshop on Artificial Intelligence for Web Search*, 2000.
19. S. Theodoridis and K. Koutroubas. Pattern Recognition. Academic Press, 1999.
20. A. Topchy, A. K. Jain, and W. Punch. Combining multiple weak clusterings. *IEEE International Conference of Data Mining*, 2003.
21. A. Topchy, A. K. Jain, and W. Punch. A mixture model for clustering ensembles. *SIAM International Conference on Data Mining*, 2004
22. K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained K-Means clustering with background knowledge. *International Conference on Machine Learning*, 2001.
23. K. Wagstaff. Intelligent Clustering with Instance-Level Constraints. PhD thesis, Cornell University, 2002.
24. E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems 15*, 2003.

An Adaptive Kernel Method for Semi-supervised Clustering

Bojun Yan and Carlotta Domeniconi

Department of Information and Software Engineering
George Mason University
Fairfax, Virginia 22030, USA
byan@gmu.edu, carlotta@ise.gmu.edu

Abstract. Semi-supervised clustering uses the limited background knowledge to aid unsupervised clustering algorithms. Recently, a kernel method for semi-supervised clustering has been introduced, which has been shown to outperform previous semi-supervised clustering approaches. However, the setting of the kernel's parameter is left to manual tuning, and the chosen value can largely affect the quality of the results. Thus, the selection of kernel's parameters remains a critical and open problem when only limited supervision, provided in terms of pairwise constraints, is available. In this paper, we derive a new optimization criterion to automatically determine the optimal parameter of an RBF kernel, directly from the data and the given constraints. Our approach integrates the constraints into the clustering objective function, and optimizes the parameter of a Gaussian kernel iteratively during the clustering process. Our experimental comparisons and results with simulated and real data clearly demonstrate the effectiveness and advantages of the proposed algorithm.

1 Introduction

As a recent emerging technique, semi-supervised clustering has attracted significant research interest. Compared to traditional clustering algorithms, which only use unlabeled data, semi-supervised clustering employs both unlabeled and supervised data to obtain a partitioning that conforms more closely with the user's preferences. Several recent papers have discussed this problem [16,8,1,18, 2,12].

In semi-supervised clustering, limited supervision is provided as input. The supervision can have the form of labeled data or pairwise constraints. In many applications it is natural to assume that pairwise constraints are available [1, 16]. For example, in protein interaction and gene expression data [13], pairwise constraints can be derived from the background domain knowledge. Similarly, in information and image retrieval, it is easy for the user to provide feedback concerning a qualitative measure of similarity or dissimilarity between pairs of objects. Thus, in these cases, although class labels may be unknown, a user can still specify whether pairs of points belong to the same cluster or to different ones. Furthermore, a set of classified points implies an equivalent set of pairwise constraints, but not vice versa.

Recently, a kernel method for semi-supervised clustering has been introduced [12]. This technique extends semi-supervised clustering to a kernel space, thus enabling the discovery of clusters with non-linear boundaries in input space. While a powerful technique, the applicability of a kernel-based semi-supervised clustering approach is limited in practice, due to the critical settings of kernel's parameters. In fact, the chosen parameter values can largely affect the quality of the results. While solutions have been proposed in supervised learning to estimate the optimal kernel's parameters, the problem presents open challenges when no labeled data are provided, and all we have available is a set of pairwise constraints.

In this paper, we derive a new optimization criterion to automatically estimate the optimal parameter of a Gaussian kernel, directly from the data and the given constraints. Our approach integrates the constraints into the clustering objective function, and optimizes the parameter of a Gaussian kernel iteratively during the clustering process. As a result, our technique is able to automatically embed, during the clustering process, the optimal non-linear similarity within the feature space. This makes our adaptive technique capable of discovering clusters with non-linear boundaries in input space with high accuracy, as demonstrated in our experiments. Our proposed method enables the practical utilization of powerful kernel-based semi-supervised clustering approaches by providing a mechanism to automatically set the involved critical parameters.

The rest of the paper is organized as follows. Section 2 provides the necessary background on kernel-based clustering and semi-supervised clustering. Section 3 motivates our approach, and discusses the details of our algorithm. Section 4 describes our experimental settings and results. Section 5 discusses the related work, and finally we provide conclusions and future research directions in Section 6.

2 Background

This section introduces the necessary background on kernel-based clustering and semi-supervised clustering.

2.1 Kernel KMeans

Let X be a dataset of N samples and D dimensions, $X = \{\mathbf{x}_i\}_{i=1}^{N} \subseteq \Re^D$. Let $\phi : \Re^D \rightarrow \Re^{D'}$ be a non-linear mapping function, which maps data from the input (D dimensional) space to a feature space (D' dimensional), with $D' > D$. The Kernel KMeans algorithm generates a partition $\{\pi_c\}_{c=1}^{k}$ of X (π_c represents the c^{th} cluster) so that the objective function $\sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \|\phi(\mathbf{x}_i) - \mathbf{m}_c^{\phi}\|$ is minimized, where $\mathbf{m}_c^{\phi} = \frac{1}{|\pi_c|} \sum_{\mathbf{x}_i \in \pi_c} \phi(\mathbf{x}_i)$ represents the centroid of cluster π_c in feature space. The key issue of Kernel-KMeans is the computation of distances in feature space. The distance of a point $\mathbf{x}_i$ from $\mathbf{m}_c^{\phi}$ in feature space can be expressed as: $\|\phi(\mathbf{x}_i) - \mathbf{m}_c^{\phi}\| = A_{ii} + B_{cc} - D_{ic}$, where $A_{ii} = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_i)$, $D_{ic} = \frac{2}{|\pi_c|} \sum_{\mathbf{x}_j \in \pi_c} \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$, and $B_{cc} = \frac{1}{|\pi_c|^2} \sum_{\mathbf{x}_j, \mathbf{x}_{j'} \in \pi_c} \phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_{j'})$.

Following the standard SVM method, we can represent the dot product of points in kernel space using an appropriate Mercer kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ [15]. Since data points always appear in the form of dot products, the terms for distance computation can be rewritten using the kernel trick: $A_{ii} = K(\mathbf{x}_i, \mathbf{x}_j)$, $D_{ic} = \frac{2}{|\pi_c|} \sum_{\mathbf{x}_j \in \pi_c} K(\mathbf{x}_i, \mathbf{x}_j)$, and $B_{cc} = \frac{1}{|\pi_c|^2} \sum_{\mathbf{x}_j, \mathbf{x}_{j'} \in \pi_c} K(\mathbf{x}_j, \mathbf{x}_{j'})$. We note that A_{ii} is common to every cluster, thus we can avoid calculating it, while B_{cc} must be calculated once in each iteration.

2.2 HMRF Model and Kernel-Based Semi-supervised Clustering

In semi-supervised clustering, we are given a set of pairwise constraints: must-link $ML = \{(\mathbf{x}_i, \mathbf{x}_j)\}$ and cannot-link $CL = \{(\mathbf{x}_i, \mathbf{x}_j)\}$. The goal is to partition the data into k clusters so that a given measure of distorsion between each point and the corresponding cluster representative is minimized, and, at the same time, the smallest number of constraint violation is achieved. Basu et al. (2004) [2] proposed a framework for semi-supervised clustering based on Hidden Markov Random Fields (HMRFs). Considering the squared Euclidean distance as a measure of cluster distortion, and the generalized Potts potential as constraint violation potential, the semi-supervised clustering objective can be expressed as [2]:

$$J_{obj}(\{\pi_c\}_{c=1}^{k}) = \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \|\mathbf{x}_i - \mathbf{m}_c\|^2 + \sum_{\mathbf{x}_i, \mathbf{x}_j \in ML, l_i \neq l_j} w_{ij} + \sum_{\mathbf{x}_i, \mathbf{x}_j \in CL, l_i = l_j} \overline{w}_{ij}$$

where $\mathbf{m}_c$ is the centroid of cluster π_c, ML is the set of must-link constraints, CL is the set of cannot-link constraints, w_{ij} and $\overline{w}_{ij}$ are the penalty costs for violating a must-link and a cannot-link constraint respectively, and l_i represents the cluster label of $\mathbf{x}_i$.

Kulis et al. (2005) [12] extended this framework to a kernel-based semi-supervised clustering. Instead of adding a penalty term for a must-link violation, a reward is given for the satisfaction of the constraint. This is achieved by subtracting the corresponding penalty term from the objective:

$$J_{obj}(\{\pi_c\}_{c=1}^{k}) = \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \|\phi(\mathbf{x})_i - \mathbf{m}_c^{\phi}\|^2 - \sum_{\mathbf{x}_i, \mathbf{x}_j \in ML, l_i = l_j} w_{ij} + \sum_{\mathbf{x}_i, \mathbf{x}_j \in CL, l_i = l_j} \overline{w}_{ij}$$

The algorithm derived in [12] (called SS-Kernel-KMeans), when combined with the Gaussian kernel, is shown to outperform the HMRF-KMeans approach [2], and SS-Kernel-KMeans combined with a linear kernel. However, the setting of the kernel's parameter is left to manual tuning, and the chosen value can largely affect the quality of the results. Thus, the selection of kernel's parameters remains a critical and open problem when only limited supervision is available. This leads to the motivation of our approach discussed in the next Section.

3 Adaptive Kernel-Based Semi-supervised Clustering

In kernel-based learning algorithms it is important that the kernel function in use conforms with the learning target. For classification, the distribution of data in feature space should be correlated to the label distribution. Similarly, in semi-supervised clustering, one wishes to learn a kernel that maps pairs of points subject to a must-link constraint close to each other in feature space, and maps points subject to a cannot-link constraint far apart in feature space.

The authors in [9] introduce the concept of *kernel alignment* to measure the correlation between the groups of data in feature space and the labeling to be learned. In [17], a Fisher discriminant rule is used to estimate the optimal spread parameter of a Gaussian kernel. The selection of kernel's parameters is indeed a critical problem. For example, empirical results in the literature have shown that the value of the spread parameter σ of a Gaussian kernel can strongly affect the generalization performance of an SVM. Values of σ which are too small or too large lead to poor generalization capabilities. When $\sigma \rightarrow 0$, the kernel matrix becomes the identity matrix. In this case, the resulting optimization problem gives Lagrangians which are all 1s, and therefore every point becomes a support vector. On the other hand, when $\sigma \rightarrow \infty$, the kernel matrix has entries all equal to 1, and thus each point in feature space is maximally similar to each other. In both cases, the machine will generalize very poorly.

The problem of setting kernel's parameters, and of finding in general a proper mapping in feature space, is even more difficult when no labeled data are provided, and all we have available is a set of pairwise constraints. In this paper we utilize the given constraints to derive an optimization criterion to automatically estimate the optimal kernel's parameters. Our approach integrates the constraints into the clustering objective function, and optimize the kernel's parameters iteratively while discovering the clustering structure. Specifically, we steer the search for optimal parameter values by measuring the amount of must-link and cannot-link constraint violations in feature space. Following the method proposed in [2, 4], we scale the penalty terms by the distances of points, that violate the constraints, in feature space. That is, for violation of a must-link constraint $(\mathbf{x}_i, \mathbf{x}_j)$, the larger the distance between the two points $\mathbf{x}_i$ and $\mathbf{x}_j$ in feature space, the larger the penalty; for violation of a cannot-link constraint $(\mathbf{x}_i, \mathbf{x}_j)$, the smaller the distance between the two points $\mathbf{x}_i$ and $\mathbf{x}_j$ in feature space, the larger the penalty. According to these rules, we can formulate the penalty terms as follows:

$$P_{ML}(\mathbf{x}_i, \mathbf{x}_j) = w_{ij} \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 1(l_i \neq l_j) \tag{1}$$

$$P_{CL}(\mathbf{x}_i, \mathbf{x}_j) = \overline{w}_{ij} ((D^{\phi}_{max})^2 - \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2) 1(l_i \neq l_j) \tag{2}$$

D^{ϕ}_{max} is the maximum distance between any pair of points in feature space; it ensures that the penalty for violated cannot-link constraints is non-negative.

By combining these two penalty terms with the objective function of Kernel KMeans, we obtain the objective function for our adaptive semi-supervised kernel KMeans (Adaptive-SS-Kernel-KMeans) approach:

$$\begin{aligned} J_{obj} = & \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} (\|\phi(\mathbf{x}_i) - \mathbf{m}_c^{\phi}\|^2) + \sum_{(\mathbf{x}_i,\mathbf{x}_j) \in ML, l_i \neq l_j} w_{ij} \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 \\ & + \sum_{(\mathbf{x}_i,\mathbf{x}_j) \in CL, l_i = l_j} \overline{w}_{ij} ((D_{max}^{\phi})^2 - \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2) \end{aligned} \quad (3)$$

Suppose $\mathbf{x}^{'}$ and $\mathbf{x}^{''}$ are the farthest points in feature space. We use the equality $\sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \|\mathbf{x}_i - \mathbf{m}_c\|^2 = \sum_{c=1}^{k} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \pi_c} \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2|\pi_c|}$ to re-formulate Equation (3) as follows:

$$\begin{aligned} J_{obj} = & \sum_{c=1}^{k} \sum_{(\mathbf{x}_i,\mathbf{x}_j) \in \pi_c} \frac{\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2}{2|\pi_c|} + \sum_{(\mathbf{x}_i,\mathbf{x}_j) \in ML, l_i \neq l_j} w_{ij} \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 \\ & + \sum_{(\mathbf{x}_i,\mathbf{x}_j) \in CL, l_i = l_j} \overline{w}_{ij} (\|\phi(\mathbf{x}^{'}) - \phi(\mathbf{x}^{''})\|^2 - \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2) \end{aligned}$$

By expanding the distance computation in feature space $\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2$, and using the kernel trick $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$, we obtain:

$$\begin{aligned} J_{obj} = & \sum_{c=1}^{k} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \pi_c} \frac{K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)}{2|\pi_c|} \\ & + \sum_{(\mathbf{x}_i,\mathbf{x}_j) \in ML, l_i \neq l_j} w_{ij} (K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)) \\ & + \sum_{(\mathbf{x}_i,\mathbf{x}_j) \in CL, l_i = l_j} \overline{w}_{ij} (K(\mathbf{x}', \mathbf{x}') + K(\mathbf{x}'', \mathbf{x}'') - 2K(\mathbf{x}', \mathbf{x}'') \\ & - K(\mathbf{x}_i, \mathbf{x}_i) - K(\mathbf{x}_j, \mathbf{x}_j) + 2K(\mathbf{x}_i, \mathbf{x}_j)) \end{aligned} \quad (4)$$

Let us consider the Gaussian kernel function: $K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2))$. (From now on we utilize the Gaussian kernel to derive our algorithm, since it has excellent learning properties. Other kernel functions can be used as well.) We want to minimize J_{obj} with respect to the kernel parameter σ. As observed earlier, when $\sigma \to \infty$, all points in feature space are maximally similar to each other, and the objective function (4) is trivially minimized. To avoid this degenerate case, we add the following constraint:

$$\sum_{\mathbf{x}_i \in X} \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_r)\|^2 \geq Const \quad (5)$$

where $\mathbf{x}_r$ is a point randomly selected from X. By incorporating constraint (5) into the objective function, and applying the kernel trick for distance computation in feature space, we finally obtain:

$$J_{kernel-obj} = \sum_{c=1}^{k} \sum_{\mathbf{x}_i,\mathbf{x}_j \in \pi_c} \frac{1-K(\mathbf{x}_i,\mathbf{x}_j)}{|\pi_c|} + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in ML, l_i \neq l_j} 2w_{ij}(1-K(\mathbf{x}_i,\mathbf{x}_j))$$

$$+ \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in CL, l_i = l_j} 2\overline{w}_{ij}(K(\mathbf{x}_i,\mathbf{x}_j) - K(\mathbf{x}',\mathbf{x}'')) - (\sum_{\mathbf{x}_i \in X} 2(1-K(\mathbf{x}_i,\mathbf{x}_r)) - Const)$$

Given $\frac{\partial K(\mathbf{x}_i,\mathbf{x}_j)}{\partial \sigma} = exp(\frac{-\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2})\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\sigma^3}$, we compute $\frac{\partial J_{kernel-obj}}{\partial \sigma}$:

$$\begin{aligned}\frac{\partial J_{kernel-obj}}{\partial \sigma} = & \sum_{c=1}^{k} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in \pi_c} -\frac{1}{|\pi_c|} exp(\frac{-\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2})\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\sigma^3} \\ - & \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in ML, l_i \neq l_j} 2w_{ij} exp(\frac{-\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2})\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\sigma^3}) \\ + & \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in ML, l_i \neq l_j} 2\overline{w}_{ij}(exp(\frac{-\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2})\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\sigma^3} \\ & - exp(\frac{-\|\mathbf{x}'-\mathbf{x}''\|^2}{2\sigma^2})\frac{\|\mathbf{x}'-\mathbf{x}''\|^2}{\sigma^3}) \\ + & \sum_{\mathbf{x}_i \in X} 2exp(\frac{-\|\mathbf{x}_i-\mathbf{x}_r\|^2}{2\sigma^2})\frac{\|\mathbf{x}_i-\mathbf{x}_r\|^2}{\sigma^3}\end{aligned} \tag{6}$$

In the following we derive an EM-based strategy to optimize $J_{kernel-obj}$ by gradient descent.

3.1 EM-Based Strategy

To minimize the objective function $J_{kernel-obj}$, we use an EM-based strategy. We initialize the clusters utilizing the mechanism proposed in [12]: we take the transitive closure of the constraints to form neighborhoods, and then perform a farthest-first traversal on these neighborhoods to get the k initial clusters. We ensure that the same set of constraints is given to the competitive algorithm in our experiments.

E-step: The algorithm assigns data points to clusters so that the objective function $J_{kernel-obj}$ is minimized. Since the objective function integrates the given must-link and cannot-link constraints, it is minimized by assigning each point to the cluster with the closest centroid (first term of $J_{kernel-obj}$) which causes a minimal penalty for violations of constraints (second and third term of $J_{kernel-obj}$). The fourth term of $J_{kernel-obj}$ is constant during the assignment of data points in each iteration. When updating the cluster assignment of a given point, the assignment for the other points is kept fixed [3,19]. During each iteration, data points are re-ordered randomly. The process is repeated until no change in point assignment occurs.

M-step: The algorithm re-computes the cluster representatives. In practice, since we map data in kernel space and do not have access to the coordinates of cluster representatives, we re-compute the term B_{cc} (as discussed in Section 2.1), which will be used to re-assign points to clusters in the E-step. Constraints are not used in this step. Therefore, only the first term of $J_{kernel-obj}$ is minimized.

We note that all the steps so far are executed with respect to the current feature space. We now optimize the feature space by optimizing the kernel parameter σ. To this extent, we apply the gradient descent rule to update the parameter σ of the Gaussian kernel: $\sigma^{(new)} = \sigma^{(old)} - \rho \frac{\partial J_{kernel-obj}}{\partial \sigma}$, where ρ is a scalar step length parameter optimized via a line-search method. The expression for $\frac{\partial J_{kernel-obj}}{\partial \sigma}$ is given in Equation (6).

A description of the algorithm (Adaptive-SS-Kernel-KMeans) is provided in Figure 1.

```
Algorithm: Adaptive-SS-Kernel-KMeans
Input:
  – Set of data points $X = \{\mathbf{x}_i\}_{i=1}^N$
  – Set of must-link constraints $ML$
  – Set of cannot-link constraints $CL$
  – Number of clusters $k$
  – Constraint violation costs $w_{ij}$ and $\overline{w}_{ij}$
Output:
  – Partition of $X$ into $k$ clusters
Method:
  1. Initialize clusters $\{\pi_c^{(0)}\}_{c=1}^k$ using the given constraints; set $t = 0$.
  2. Repeat Step3 - Step6 until convergence.
  3. E-step: Assign each data point $\mathbf{x}_i$ to a cluster $\pi_c^{(t)}$ so that $J_{kernel-obj}$ is minimized.
  4. M-step(1): Re-compute $B_{cc}^{(t)}$, for $c = 1, 2, \cdots, k$.
  5. M-step(2): Optimize the kernel parameter $\sigma$ using gradient descent according to
     the rule: $\sigma^{(new)} = \sigma^{(old)} - \rho \frac{\partial J_{kernel-obj}}{\partial \sigma}$.
  6. Increment $t$ by 1.
```

Fig. 1. Adaptive-SS-Kernel-KMeans

4 Experimental Evaluation

4.1 Datasets

We performed experiments on one simulated dataset and four real datasets. (1) The simulated dataset contains two clusters in two dimensions distributed as concentric circles (See Figure 2(a)). Each cluster contains 200 points. (2) **Digits:** This dataset is the pendigits handwritten character recognition dataset from the

UCI repository[1] [5]. 10% of the data were chosen randomly from the three classes $\{3, 8, 9\}$ as done in [12]. This results in 317 points and 16 dimensions. (3) **Spectf:** This dataset is also from the UCI repository [5]. It describes the diagnosis of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each patient is classified into one of two categories: normal or abnormal. 267 SPECT image sets (patients) were processed to extract features that summarize the original SPECT images. As a result, 44 continuous features were created for each patient. (4) **Vowel:** This dataset concerns the recognition of eleven steady state vowels of British English, using a specified training set of lpc derived log area ratios[2]. Three class corresponding to the vowels "i", "I", and "E" were chosen, for a total of 126 points and 10 dimensions; (5) **Segmentation:** This dataset is from UCI repository [5]. It has 210 points and 19 dimensions. The instances were drawn randomly from a database of 7 outdoor images. The images were hand-segmented to create a classification for every pixel.

4.2 Evaluation Criterion

To evaluate the clustering results, we use the Rand Statistic index [14,18,16]. The Rand Statistic is an external cluster validity measure that estimates the quality of the clustering results with respect to the underlying classes of the data. Let P_1 be the partition of the data X after applying a clustering algorithm, and P_2 be the underlying class structure of the data. We refer to a pair of points $(\mathbf{x}_u, \mathbf{x}_v) \in X \times X$ from the data using the following terms:

- SS: if both points belong to the same cluster of P_1 and to the same group of the underlying class structure P_2.
- SD: if the two points belong to the same cluster of P_1 and to different groups of P_2.
- DS: if the two points belong to different clusters of P_1 and to the same group of P_2.
- DD: if both points belong to different clusters of P_1 and to different groups of P_2.

Assume now that N_{SS}, N_{SD}, N_{DS} and N_{DD} are the number of SS, SD, DS and DD pairs respectively, then $N_{SS} + N_{SD} + N_{DS} + N_{DD} = N_{Pair}$ which is the maximum number of all pairs in the data set[3]. The Rand Statistic index measures the degree of similarity between P_1 and P_2 as follows:

$$RandStatistic = (N_{SS} + N_{DD})/N_{Pair} \tag{7}$$

4.3 Results and Discussion

To evaluate the effectiveness of our proposed method Adaptive-SS-Kernel-KMeans we perform comparisons with SS-Kernel-KMeans [12]. As shown in

[1] http://www.ics.uci.edu/˜mlearn/MLRepository.html

[2] http://www-stat-class.stanford.edu/˜tibs/ElemStatLearn/

[3] $N_{Pair} = N(N-1)/2$, where N is the total number of points in the data set.

[12], SS-Kernel-KMeans combined with a Gaussian kernel outperforms HMRF-KMeans and SS-Kernel-KMeans with a linear kernel. Therefore, the technique SS-Kernel-KMeans with Gaussian kernel was the proper choice for our empirical comparisons. SS-Kernel-KMeans requires in input a predefined value for the Gaussian kernel parameter σ. In absence of labeled data, parameters cannot be cross-validated; thus, we estimate the expected accuracy of SS-Kernel-KMeans by averaging the resulting clustering quality over multiple runs for different values of σ. Specifically, in our experiments, we test the SS-Kernel-KMeans algorithm with the values of σ^2: 0.1, 1, 10, 100, 1000, 10000. We report the average Rand Statistic achieved over the six σ values, as well as the average over the best three performances achieved, in order to show the advantage of our technique also in this latter case. The violation costs w_{ij} and $\overline{w}_{ij}$ are set to 1 in our experiments since we assume no a-priori knowledge on such costs. As value of k, we provide the actual number of classes in the data to both algorithms.

Figures 2-4 show the learning curves using 20 runs of 2-fold cross-validation for each data set (30% for training and 70% for testing). These plots show the improvement in clustering quality on the test set as a function of an increasing amount of pairwise constraints. To study the effect of constraints in clustering, 30% of the data was randomly drawn as the training set at any particular fold, and the constraints are generated only using the training set. The clustering algorithm was run on the whole data set, but we calculated the Rand Statistic only on the test set. Each point on the learning curve is an average of results over 20 runs.

The results shown in Figures 2-4 clearly demonstrate the effectiveness of our proposed technique Adaptive-SS-Kernel-KMeans. For all five datasets, the clustering quality achieved by our adaptive approach significantly outperforms the results provided by SS-Kernel-KMeans, averaged over the σ values tested. In most cases (TwoConcentric, Vowel, Digits, and Segmentation), the Adaptive-SS-Kernel-KMeans technique also outperforms the average top three performances of SS-Kernel-KMeans. For the Stectf data the two approaches show a similar trend. These results show that our adaptive technique is capable of estimating the optimal kernel parameter value from the given constraints. In particular, for the TwoConcentric data (see Figure 2(b)), the Adaptive-SS-Kernel-KMeans technique effectively uses the increased amount of constraints to learn a perfect separation of the two clusters. For the Digits, Spectf, and Segmentation data, the Adaptive-SS-Kernel-KMeans technique provides a clustering quality that is significantly higher than the one given by SS-Kernel-KMeans, even when a small amount of constraints is available. This behavior is very desirable since in practice only a limited amount of supervision might be available. We also emphasize that the cluster initialization mechanism employed in the EM-based strategy mitigates the sensitivity of the result at convergence from the starting point of the search.

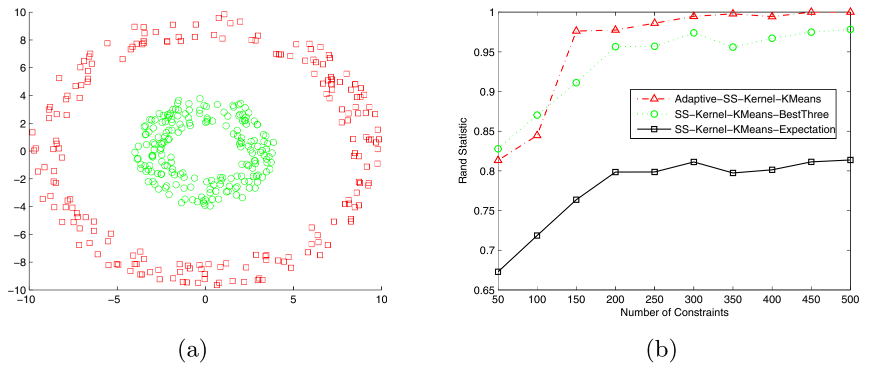


Fig. 2. (a) TwoConcentric data (b) Clustering result on TwoConcentric data

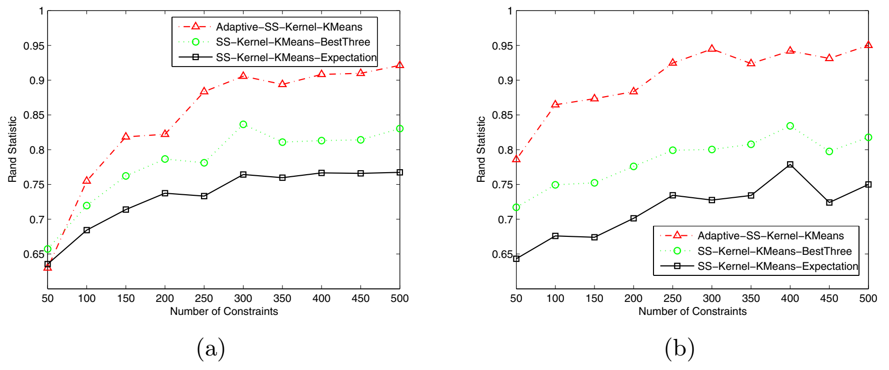


Fig. 3. (a) Clustering result on Vowel data (b) Clustering result on Digits data

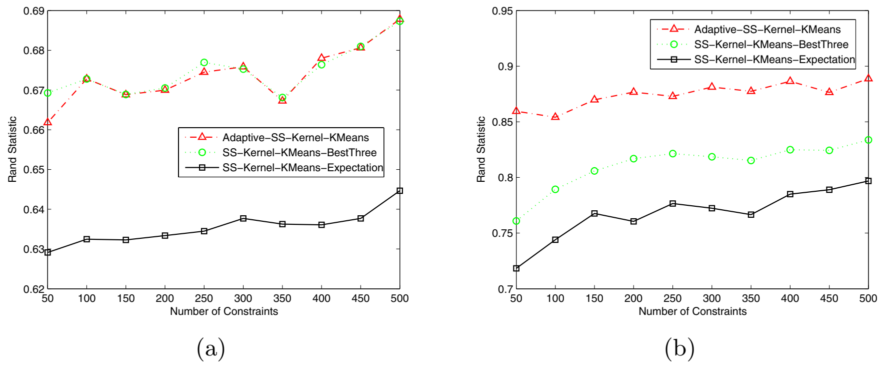


Fig. 4. (a) Clustering result on Spectf data (b) Clustering result on Segmentation data

5 Related Work

In the context of supervised learning, the work in [7] considers the problem of automatically tuning multiple parameters for a support vector machine. This is achieved by minimizing the estimated generalization error achieved by means of a gradient descent approach over the set of parameters. In [17], a Fisher discriminant rule is used to estimate the optimal spread parameter of a Gaussian kernel. The authors in [10] propose a new criterion to address the selection of kernel's parameters within a kernel Fisher discriminant analysis framework for face recognition. A new formulation is derived to optimize the parameters of a Gaussian kernel based on a gradient descent algorithm. This research makes use of labeled data to address classification problems. In contrast, our approach optimizes kernel's parameters based on unlabeled data and pairwise constraints, and aims at solving clustering problems.

In the context of semi-supervised clustering, the authors in [8] use a gradient descent approach combined with a weighted Jensen-Shannon divergence for EM clustering. The authors in [1] propose a method based on Redundant Component Analysis (RCA) that uses must-link constraints to learn a Mahalanobis distance. [18] utilizes both must-link and cannot-link constraints to formulate a convex optimization problem which is local-minima-free. [13] proposes a unified Markov network with constraints. [2] introduces a more general HMRF framework, that works with different clustering distortion measures, including Bregman divergences and directional similarity measures. All these techniques use the given constraints and an underlying (linear) distance metric for clustering points in input space. [12] extends the semi-supervised clustering framework to a non-linear kernel space. However, the setting of the kernel's parameter is left to manual tuning, and the chosen value can largely affect the results. The selection of kernel's parameters is a critical and open problem, which has been the driving force behind the work presented in this paper.

6 Conclusion and Future Work

We proposed a new adaptive semi-supervised Kernel-KMeans algorithm. Our approach integrates the given constraints with the kernel function, and is able to automatically embed, during the clustering process, the optimal non-linear similarity within the feature space. As a result, the proposed algorithm is capable of discovering clusters with non-linear boundaries in input space with high accuracy. Our technique enables the practical utilization of powerful kernel-based semi-supervised clustering approaches by providing a mechanism to automatically set the involved critical parameters. In our future work we will consider active learning as a methodology to generate constraints which are most informative. We will also consider other kernel functions (e.g., polynomial) in our future experiments, as well as combinations of different types of kernels.

Acknowledgements

This work was in part supported by NSF CAREER Award IIS-0447814.

References

1. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning distance functions using equivalence relations. International Conference on Machine Learning, pages 11-18, 2003.
2. Basu, S., Bilenko, M., Mooney, R.J.: A probabilistic framework for semi-supervised clustering. International Conference on Knowledge Discovery and Data Mining, 2004.
3. Besag, J.: On the statistical analysis of dirty pictures. Journal of the Royal Statistical Society, Series B (Methodological), 1986.
4. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and Metric Learning in semi-supervised clustering. International Conference on Machine Learning, 2004.
5. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.
6. Boykov, Y., Veksler, O., Zabih, R.: Markov Random fields with efficient approximations. IEEE Computer Vision and pattern Recognition Conference, 1998.
7. Chapelle, O., Vapnik, V.: Choosing Mutiple Parameters for Support Vector Machines. Machine Learning Vol.46, No. 1. pp.131-159, 2002.
8. Cohn, D., Caruana, R., McCallum, A.: Semi-supervised clustering with user feedback. TR2003-1892, Cornell University, 2003.
9. Cristianini, N., Shawe-Taylor, J., Elisseeff, A.: On Kernel-Target Alignment, Neural Information Processing Systems (NIPS), 2001.
10. Huang, J., Yuen, P.C., Chen, W.S., Lai, J. H.: Kernel Subspace LDA with optimized Kernel Parameters on Face Recognition. The sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004.
11. Kleinberg, J., Tardos, E.: Approximation algorithms for classification problems with pairwise relationships: metric labeling and Markov random fields. The 40th IEEE Symposium on Foundation of Computer Science, 1999.
12. Kulis, B., Basu, S., Dhillon, I., Moony, R.: Semi-supervised graph clustering: a kernel approach. International Conference on Machine Learning, 2005.
13. Segal, E., Wang, H., Koller, D.: Discovering molecular pathways from protein interaction and gene expression data. Bioinformatics, 2003.
14. Theodoridis, S., Koutroubas, K.: Pattern Recognition. Academic Press, 1999.
15. Vapnik., V.: The Nature of Statistical Learning Theory, Wiley, New York, USA, 1995.
16. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained K-Means clustering with background knowledge. International Conference on Machine Learning, pages 577-584, 2001.
17. Wang, W., Xu, Z., Lu W., Zhang, X.: Determination of the spread parameter in the Gaussian Kernel for classification and regression. Neurocomputing, Vol. 55, No. 3, 645, 2002.
18. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. Advances in Neural Information Processing Systems 15, 2003.
19. Zhang, Y., Brady, M., Smith, S.: Hidden Markov random field model and segmentation of brain MR images. IEEE Transactions on Medical Imaging, 2001.

To Select or To Weigh: A Comparative Study of Model Selection and Model Weighing for SPODE Ensembles

Ying Yang[1], Geoff Webb[1], Jesús Cerquides[2], Kevin Korb[1], Janice Boughton[1], and Kai Ming Ting[1]

[1] Clayton School of Information Technology, Monash University, Australia
{ying.yang, geoff.webb, kevin.korb, janice.boughton, kaiming.ting}@infotech.monash.edu.au

[2] Departament de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Spain
cerquide@maia.ub.es

Abstract. An ensemble of Super-Parent-One-Dependence Estimators (SPODEs) offers a powerful yet simple alternative to naive Bayes classifiers, achieving significantly higher classification accuracy at a moderate cost in classification efficiency. Currently there exist two families of methodologies that ensemble candidate SPODEs for classification. One is to select only helpful SPODEs and uniformly average their probability estimates, a type of *model selection*. Another is to assign a weight to each SPODE and linearly combine their probability estimates, a methodology named *model weighing*. This paper presents a theoretical and empirical study comparing model selection and model weighing for ensembling SPODEs. The focus is on maximizing the ensemble's classification accuracy while minimizing its computational time. A number of representative selection and weighing schemes are studied, providing a comprehensive research on this topic and identifying effective schemes that provide alternative trades-off between speed and expected error.

1 Introduction

Semi-naive Bayesian classifiers reduce error by relaxing the attribute independence assumption of naive Bayes [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]. Among alternative semi-naive forms, Super-Parent-One-Dependence Estimators (SPODEs) [2, 3], and particularly ensembles thereof [13] have received a lot of attention [18, 19, 20, 21, 22] because they offer a combination of high training efficiency, high classification efficiency and high classification accuracy. Those merits give SPODEs a great potential to substitute for naive Bayes classifiers in numerous real-world classification systems, including medical diagnosis, fraud detection, email filtering, document classification and webpage prefetching. This paper identifies approaches that can maximize a SPODE ensemble's classification accuracy while minimizing its computational time. This leads to accurate and fast classification algorithms with immediate and significant impact on real-world applications.

1.1 Terminology and Notation

This paper addresses the problem of classification learning using an ensemble of Bayesian probabilistic classifiers. The following terminology and notation will be used throughout the paper. An *instance* $\mathbf{x}\,\langle x_1, x_2, \cdots, x_m\rangle$ is a vector of m attribute values x_i, each observed for an attribute variable X_i $(i \in [1, m])$. It can also have a class label y corresponding to the class variable Y. If its class label is known, an instance is *labeled*. Otherwise, it is *unlabeled*. *Training data* D is a set of labeled instances from which a *classifier* is learned to predict the class labels of unlabeled instances. The number of training instances is n. The number of values for X_i is v_i. X_i's parent variables are $\Phi(i)$. The number of joint states (joint instantiated values) of parents of X_i is $|\phi(i)|$. The r-th joint state of the parents is ϕ_{ir}. When applicable, h indicates a SPODE in general and h_i indicates a particular SPODE whose superparent is X_i.

1.2 SPODE and SPODE Ensemble

A SPODE [2,3] relaxes the naive Bayes (NB) classifier's attribute independence assumption by allowing all attributes to depend on a common attribute, the *superparent*, in addition to the class, as depicted in Figure 1.

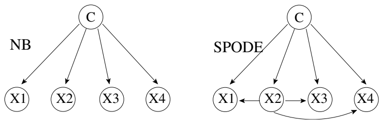


Fig. 1. Illustration of SPODE versus NB. An arc points from a parent to a child. A child only depends on its parents. NB assumes each attribute only depends on the class Y and is independent of other attributes given the class. SPODE assumes that each attribute can depend on both the superparent X_2 and the class.

To classify an instance $\mathbf{x}$, a Bayesian probabilistic classifier calculates $\hat{P}(y \mid \mathbf{x})$, an estimate of the probability of each class label given this instance. The label attaining the highest probability will be assigned to $\mathbf{x}$. Since $\hat{P}(y \mid \mathbf{x}) = \frac{\hat{P}(y,\mathbf{x})}{P(\mathbf{x})}$ and $P(\mathbf{x})$ is invariant across different class labels, one only needs to calculate $\hat{P}(y, \mathbf{x})$. That is, $\text{argmax}_y\, \hat{P}(y \mid \mathbf{x}) = \text{argmax}_y\, \hat{P}(y, \mathbf{x})$.

A SPODE with superparent X_p finds $\text{argmax}_y\, \hat{P}(y, \mathbf{x})$ using $\hat{P}(y, \mathbf{x}) = \hat{P}(y, x_p)\hat{P}(\mathbf{x} \mid y, x_p) = \hat{P}(y, x_p) \prod_{i=1}^{m} \hat{P}(x_i \mid y, x_p)$. The final formula results from SPODEs' assumption that all attributes are independent of each other given Y and X_p.

A SPODE ensemble is a linear combination of multiple SPODEs' probability estimates. It seeks $\text{argmax}_y\, \hat{P}(y, \mathbf{x})$ using: $\hat{P}(y, \mathbf{x}) \approx \sum_{i=1}^{m} w_i \hat{P}_i(y, \mathbf{x})$, where each $\hat{P}_i(y, \mathbf{x})$ is calculated by a SPODE whose superparent being X_i. For a training data set with m attributes, there can be m candidate SPODEs, each taking a different attribute as its superparent.

It has been shown that a SPODE, being a one-dependence estimator, can provide better probability estimates than NB because it involves a weaker attribute independence assumption [1, 2, 3, 10, 13]. It has also been shown that a SPODE ensemble can further improve upon the classification accuracy of a single SPODE by decreasing the classification variance [13,18]. The first approach to ensembling SPODEs was AODE [13] which used equal weight combination of all SPODEs whose parent occurred with a user-specified minimum frequency in the training data. Subsequent research suggested that frequency is not a useful model selection criterion and that appropriate weighting can substantially improve upon equal weighting, proposing weighting schemes such as MAPLMG [18]. On the other hand, it has also been shown that model selection can be very effective when ensembling SPODEs [22]. This paper presents a comprehensive investigation into the relative merits of alternative approaches to weighting and selecting.

2 Model Selection Schemes

The general problem for model selection is, given some sample data, how to decide which are the most effective models within some model space. This paper looks at the space of SPODE models. Only selected SPODEs will be included in the ensemble. Previous research has suggested that cross validation, forward sequential addition and lazy elimination are more effective than alternative selection methods for SPODEs [22, 23].

Cross Validation (CV). [22] scores each individual SPODE by its cross validation error on the training data. In this study, leave-one-out cross validation is employed. Given a SPODE, CV loops through the training data n times, each time training the SPODE from $(n-1)$ instances to classify the remaining 1 instance. The misclassifications are summed and averaged over n iterations. The resulting classification error rate is taken as the metric value of the SPODE. The lower the metric, the higher priority a SPODE should be used. This process is very efficient as the model need only be updated for each instance that is left out, rather than recalculated from scratch. Given a sequence of m SPODEs ordered by their CV values, m ensembles are candidates, from size 1 to size m. Starting with an empty ensemble, each ensemble in turn includes further one SPODE in the queue. Every ensemble's leave-one-out cross validation error is calculated. The ensemble with the lowest error is the one to be selected.[1]

Forward Sequential Addition (FSA). [22] begins with an empty ensemble. It then uses hill-climbing search to iteratively add SPODEs whose individual inclusion results in the lowest classification error. In each iteration, suppose the

[1] If there are multiple ensembles that attain the lowest error, the one with the largest ensemble size is selected as a means to reduce classification variance caused by model selection. The same rule also applies to FSA.

current ensemble is $E_{current}$ with k SPODEs. FSA in turn adds each candidate SPODE, one that has not been included into $E_{current}$, and obtains an ensemble E_{test} of size $(k+1)$. It then calculates the leave-one-out cross validation error of E_{test}. The E_{test} with the lowest error is retained and the corresponding added SPODE is permanently deleted from the candidate list and included into the ensemble. The same process is applied to the new SPODE ensemble of size $(k+1)$ and so on, until every SPODE has been included. The order of addition produces a ranking order for SPODEs. The earlier a SPODE is added, the more merit it possesses and the higher its priority to be used. The ensemble which achieves the lowest error in the adding process is the one to be selected.

Lazy Elimination (LE). CV and FSA select at training time a subset of SPODEs that are used to classify all test instances. An alternative approach delays selection until classification time. LE [23] is based on the observation that $\forall a, b, c : P(a \mid b) = 1.0$ entails $P(c \mid a, b) = P(c \mid b)$. Hence, if it can be inferred that one attribute value entails another, assuming conditional independence between the values is likely to be harmful and the more general value may safely be deleted. To this end, before a test instance is classified LE deletes any attribute value x_i of the instance that occurs in the training data more than a user-defined minimum number of times (in this research, 30) and for which there is another value $x_j, j \neq i$ such that for every training instance containing x_j, x_i is also present. If x_i and x_j are identical, only one is deleted. Effectively, LE performs lazy selection, by not using SPODEs whose superparents are generalizations of other values of the instance to be classified. Note however that it also deletes children from within SPODEs and hence is not solely a SPODE selection algorithm.

3 Model Weighing Schemes

Model weighing focuses on calculating the weight associated with each SPODE to linearly combine their probability estimates of $P(y, \mathbf{x})$.

Information-Theoretic Metrics. provide a combined score for a proposed explanatory model (a SPODE in our context) and for the data given the model. Since they rely upon Shannon information theory [24] for their motivation and interpretation, they should support the inversion of Shannon's law to derive the posterior probability of a model given the data as to be the model's probabilistic weight for purpose of prediction. In principle, the weight w for a SPODE h is[2]:

$$w = \hat{P}(h|D) = e^{-I(h|D)} = e^{-(I(D|h)-I(D)+I(h))} = e^{\left(n\sum_{i=1}^{m+1} H(X_i,\Phi(i))\right)-I(h)}$$

where $H(X_i, \Phi(i))$ is the mutual information between X_i and its parents: $H(X_i, \Phi(i)) = \sum_{j=1}^{v_i} \sum_{r=1}^{|\phi_i|} \left(P(x_{ij}, \phi_{ir}) \log \frac{P(x_{ij},\phi_{ir})}{P(x_{ij})P(\phi_{ir})}\right)$; and $I(h)$ ***varies*** among different schemes, of which two representative ones are presented below.

[2] For simplicity, X_i represents the class variable when $i = m+1$. Generally the log base does not matter. A common practice is to use e or 2.

Bayesian Information Criterion **(BIC).** According to Schwarz [25]: $I_{BIC}(h) = (\log n)\left(\sum_{i=1}^{m+1}(v_i - 1)\prod_{j\in\Phi(i)} v_j\right)$. For any root node X_i (where $\Phi(i) = \emptyset$), the product term on the right should be replaced by 1.

Minimum Message Length **(MML).** According to Korb and Nicholson [26]: $I_{MML}(h) = \log(m+1)! + C_2^{m+1} - \log(m-1)! + \sum_{i=1}^{m+1}\frac{v_i-1}{2}(\log\frac{\pi}{6}+1) - \log\prod_{i=1}^{m+1}\prod_{j=1}^{|\phi_i|}\left(\frac{(v_i-1)!}{(S_{ij}+v_i-1)!}\prod_{l=1}^{v_i}\alpha_{ijl}!\right)$, where S_{ij} is the number of training instances where the parents $\Phi(i)$ take their joint j-th value, and α_{ijl} is the number of training instances where X_i takes its l-th value and $\Phi(i)$ take their j-th joint value. For any root X_i, $|\phi_i|$ should be treated as 1 and every instance should be treated as matching the parents for the purposes of computing S_{ij} and α_{ijl}.

Bayesian Model Averaging (BMA). provides a mechanism to ensemble classification models by accounting for single models' uncertainty of generating the data [27]. Given an instance $\mathbf{x}$ and a set of classifiers h_i, BMA estimates the probability of each class label given $\mathbf{x}$ using: $\hat{P}(y \mid \mathbf{x}) = \sum_{i=1}^{m}\hat{P}(y \mid h_i)\hat{P}(h_i \mid D)$, where $\hat{P}(y \mid h_i)$ is the class probability estimated by a SPODE. One representative approach to estimating the weight $\hat{P}(h_i \mid D)$, used in BMA, was proposed by Cooper and Herskovits [28]: $w_i = \hat{P}(h_i \mid D) = \frac{\hat{P}(h_i,D)}{\sum_{i=1}^{m}\hat{P}(h_i,D)}$, where $P(h_i,D) = \hat{P}(h_i)\prod_{k=1}^{m+1}\prod_{j=1}^{|\phi_i|}\left(\frac{(v_k-1)!}{(S_{kj}+v_k-1)!}\prod_{l=1}^{v_k}\alpha_{kjl}!\right)$, $\hat{P}(h_i) = \frac{1}{m}$ if there are m candidate SPODEs, and S_{kj} and α_{kjl} have the same meanings as for MML.

Maximum a Posteriori Linear Mixture of Generative Distributions (MAPLMG). [18] constructs a SPODE ensemble that maximizes the supervised posterior probability of the weights given the training data. It determines the weighing vector $\mathbf{w}\,\langle w_1,\ldots,w_m\rangle$ as $\mathbf{w} = \mathrm{argmax}_{\mathbf{w}}\,\hat{P}_{LMG}(\mathbf{w}|D)$ where $\hat{P}_{LMG}(\mathbf{w}|D) = \prod\limits_{\langle\mathbf{x},y\rangle\in D}\left(\frac{\sum_{i=1}^{m}w_i\hat{P}_i^{LOO}(y,\mathbf{x})}{\sum_{y\in Y}\sum_{i=1}^{m}w_i\hat{P}_i^{LOO}(y,\mathbf{x})}\prod\limits_{i=1}^{m}w_i\right)$, and $\hat{P}_i^{LOO}(y,\mathbf{x}) = \hat{P}(x_i,y)\prod\limits_{j=1}^{m}\hat{P}(x_j \mid x_i,y)$ whose right hand side is estimated from $(D - \{\langle\mathbf{x},y\rangle\})$ for h_i. The maximization is a constrained nonlinear optimization problem that can be solved by means of a sequence of unconstrained maximizations [29], each of them solved by a Newton-like procedure such as BFGS [30].

4 Time Complexity Analysis

Assume that the number of training instances and attributes are n and m, and number of classes is c. Let the average number of values for an attribute be v.

The ***training time complexity*** of each scheme is listed as follows:

CV	**FSA**	**LE**	**BIC**	**MML or BMA**	**MAPLMG**
$O(m^2nc)$	$O(m^3nc)$	$O(0)$	$O(m^2v^2c)$	$O(m^2n(v+\frac{n}{vc}))$	$O(m^2nc + Kmnc)$

Note that LE does not require any additional information to be gathered at training time and hence has no impact on training time. In practice, MML and BMA often lead to arithmetic overflow when calculating very large exponentials or factorials. One solution is to use the java class *BigDecimal* which unfortunately can be very slow. This is why MML and BMA require large amount of training time as later illustrated in Figure 3. The 'K' in MAPLMG's complexity is a large fixed number that bounds the number of iterations in the maximization step. Since K is fixed, it does not affect the theoretical complexity. However it can dominate the computing time when m,n and c are not large enough.

As for ***classification time complexity***, each scheme's dominating complexity is the linear combination of SPODEs: $O(m^2c)$ that results from the $O(mc)$ SPODE algorithm applied over an $O(m)$ sized ensemble.

5 Experiments

Empirical tests and observations of each selection or weighing scheme for ensembling SPODEs are presented here.

5.1 Design and Results

A large suite of 57 benchmark data sets from the UCI machine learning repository [31], as described in Table 1, are employed to test rival schemes. All missing

Table 1. Statistics of 57 experimental data sets

Data	Ins.	Att.	Data	Ins.	Att.	Data	Ins.	Att.
Abalone	4177	8	Hypothyroid	3772	29	Postoperative	90	8
AE	9961	12	Ionosphere	351	34	PrimaryTumor	339	17
Annealing	898	38	IrisClassification	150	4	Promoter	106	57
Audiology	226	69	KRvsKP	3196	36	Satellite	6435	36
AutosImports85	205	25	LaborNegotiations	57	16	Segment	2310	19
BalanceScale	625	4	LED	1000	7	SickEuthyroid	3772	29
Bands	1078	36	LetterRecognition	20000	16	Sign	12546	8
BreastCancer	699	9	LiverDisorders	345	6	Sonar	208	60
Chess	551	39	LungCancer	32	56	Soybean	683	35
CMC	1473	9	Lymphography	296	18	Splice	3177	60
CreditApproval	690	15	Mfeat-mor	2000	6	Syncon	600	60
Echocardiogram	131	6	Mushroom	8124	22	Thyroid	9169	29
German	1000	20	Musk	476	166	TicTacToe	958	9
GlassIdentification	214	9	NetTalkPhoneme	5438	7	Vehicle	846	18
HeartCleveland	303	13	NewThyroid	215	5	Vowel	990	11
Hepatitis	155	19	OpticalDigits	5620	48	Waveform	5000	40
HorseColic	368	21	PageBlocks	10946	10	Wine	178	13
HouseVotes84	435	16	PenDigits	10992	16	Yeast	1484	8
Hungarian	294	13	PimaDiabetes	768	8	Zoo	101	16

values for nominal and numeric attributes in a data set are replaced with the modes and means from the training data in order to facilitate calculating information metrics. Numeric attributes are discretized using entropy minimization discretization [32]. Each scheme is tested on each data set using a 10-trial 2-fold cross validation, where 5 performance measures are recorded: *training time*, *classification time* and *classification error* that can be decomposed into a *bias* term and a *variance* term [33,34,35,36,37]. We use Kohavi and Wolpert's [35] definitions of bias and variance, and estimate them using Webb's [37] cross-validation method.

It is useful to look into bias and variance of a classifier because they each offer a different perspective of view. Bias describes the component of error that results from systematic error of the learning algorithm. Variance describes the component of error that results from random variation in the training data and from random behavior in the learning algorithm, and thus measures how sensitive an algorithm is to changes in the training data. Moore and McCabe [38] illustrated bias and variance through shooting arrows at a target, as reproduced in Figure 2. We can think of the perfect classifier as the bull's-eye on a target, and the learned classifier as an arrow fired at the bull's-eye. Bias and variance describe what happens when an archer fires many arrows at the target. High bias means that the arrows land consistently off the bull's-eye in the same direction. High variance means that repeated shots differ widely among themselves and are scattered on the target. A good learning scheme, like a good archer, should have both low bias and low variance.

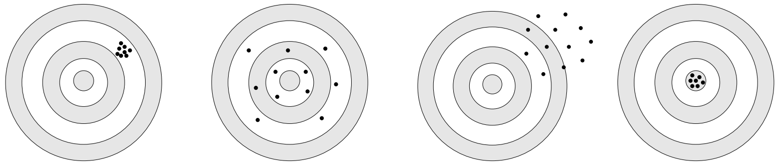

(a) High bias, low variance (b) Low bias, high variance (c) High bias, high variance (d) Low bias, low variance

Fig. 2. Bias and variance in shooting arrows at a target. Bias means that the archer systematically misses the bull's eye in the same direction. Variance means that the arrows are scattered. (Moore and McCabe, 2002)

Statistically a win/lose/tie record (w/l/t) is calculated for each pair of competitors A and B with regard to a performance measure M. The record represents the number of data sets in which A respectively beats, loses to or ties with B on M. A one-tailed binomial sign test can be applied to wins versus losses. If its result is less than the critical level of 0.05, the wins against losses are statistically significant, supporting the claim that the winner has a systematic (instead of by chance) advantage over the loser.

Please be noted that different from our previous research, we no longer impose a frequency threshold on SPODEs. Previously as a means to reduce classification variance, a SPODE was considered a candidate for ensembling only if its frequency was above 30 [22]. However, subsequent research demonstrated better results when the minimum frequency was reduced to 1 [18]. Accordingly, the experimental results of CV and FSA can be different from the previous report [22]. AODE, a complete SPODE ensemble without any selection or weighing applied, is also included to offer a baseline in comparing alternative schemes.

5.2 Observations and Analysis

Experimental results are summarized in Figures 3, 4 and Table 2. Empirical observations reveal the following knowledge.

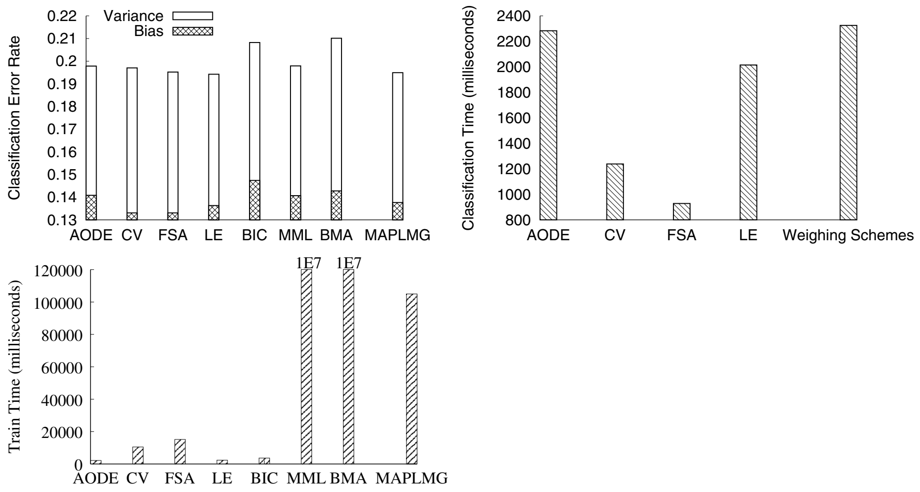


Fig. 3. Compare rival schemes' accuracy and efficiency averaged on 57 data sets. Error can be decomposed into bias and variance. To maintain a proper scale of the graph, the 'train time' bars of MML and BMA are cut short with their true values labeled on top.

LE, best model selection. According to Table 2(a), LE significantly wins against AODE and CV at the 0.05 critical level (w/l/t being 28/6/23 and 34/15/8 respectively). It also wins more often than not when compared with FSA (w/l/t being 31/20/6). As shown in Figure 3, LE achieves the lowest mean error among alternative selection methods. It is also the most efficient method in terms of training time.

MAPLMG, best model weighing. Among model weighing schemes, the best one is MAPLMG. According to Table 2(a), it significantly wins against AODE and every other single weighing scheme. As shown in Figure 3, MAPLMG achieves the lowest mean error among weighing schemes. One factor that may

Table 2. Compare rival schemes' win/lose/tie records with regard to classification error, bias and variance respectively. Each entry indicates that the scheme of the row compares against the scheme of the column. A statistically significant record (at the 0.05 critical level) is indicated in a bold face.

(a) ERROR

w/l/t	AODE	CV	FSA	LE	BIC	MML	BMA
CV	22/26/9						
FSA	27/24/6	26/15/16					
LE	**28/6/23**	**34/15/8**	31/20/6				
BIC	**10/40/7**	**8/41/8**	**7/41/9**	**6/47/4**			
MML	10/10/37	21/22/14	21/27/9	**7/31/19**	**39/11/7**		
BMA	**7/46/4**	**6/45/6**	**5/47/5**	**4/50/3**	**16/29/12**	**8/46/3**	
MAPLMG	**34/7/16**	**37/12/8**	**33/17/7**	26/19/12	**44/9/4**	**35/9/13**	**49/6/2**

(b) BIAS

w/l/t	AODE	CV	FSA	LE	BIC	MML	BMA
CV	**47/4/6**						
FSA	**48/2/7**	22/16/19					
LE	**35/1/21**	**13/35/9**	**11/35/11**				
BIC	**13/34/10**	**4/45/8**	**6/45/6**	**10/41/6**			
MML	15/11/31	**4/46/7**	**4/48/5**	**5/35/17**	**35/14/8**		
BMA	25/25/7	**6/43/8**	**8/44/5**	**12/38/7**	28/20/9	22/28/7	
MAPLMG	**37/2/18**	**11/38/8**	**5/37/15**	19/26/12	**43/11/3**	**38/6/13**	**32/18/7**

(c) VARIANCE

w/l/t	AODE	CV	FSA	LE	BIC	MML	BMA
CV	**3/49/5**						
FSA	**7/44/6**	**30/12/15**					
LE	**7/21/29**	**44/5/8**	**42/8/7**				
BIC	**18/33/6**	27/22/8	25/21/11	19/31/7			
MML	8/17/32	**47/3/7**	**43/7/7**	21/14/22	**32/17/8**		
BMA	**6/46/5**	**15/34/8**	**10/39/8**	**7/46/4**	**10/36/11**	**7/45/5**	
MAPLMG	**11/22/24**	**48/2/7**	**47/4/6**	23/16/18	**32/19/6**	14/23/20	**45/7/5**

contribute to its low error is that MAPLMG optimizes multiple weights simultaneously, while others calculate the weights for individual SPODEs in isolation. On the other hand, this optimization demands time and hence MAPLMG is slower than BIC on training (but still faster than MML and BMA as has been reasoned in Section 4).

AODE, best variance reduction. AODE does not incur sophisticated selection or weighing. It instead simply uniformly averages every SPODE's probability estimate. This simplicity turns out to be the best scheme in terms of variance reduction, as shown in Figure 3. Also according to Table 2(c), AODE always achieves lower variance than every other single scheme, most of which are statistically significant at the critical level of 0.05. In contrast, schemes like CV and FSA are more capable of reducing bias. However, their bias reduction is overshadowed by AODE's variance reduction. As a result, they cannot significantly outperform AODE on error reduction.

To select or to weigh. The best of model selection, LE, and the best of model weighing, MAPLMG both beat AODE at the statistical significance level of 0.05 in terms of classification error. Figure 4 graphs the relative bias, variance and error of the three classifiers. The values on the y-axis are the outcome for LE divided by that for AODE. The values of the x-axis are the outcome for MAPLMG divided by that for AODE. Each point on the graph represents one of the 57 data sets. Points on the left of the vertical line at MAPLMG/AODE=1 in each subgraph are those of which MAPLMG outperforms AODE. Points below the horizontal line at LE/AODE=1 indicate that LE outperforms AODE. Points below the diagonal line X=Y represent that MAPLMG outperforms LE. It is observed that both LE and MAPLMG frequently reduce bias compared with AODE as the majority of points fall within the boundaries X=1 and Y=1 in Figure 4(a). Although AODE is better at reducing variance as shown in Figure 4(b), LE and MAPLMG's bias reduction dominates AODE's variance reduction. Hence both can significantly beat AODE on error reduction as in Figure 4(c).

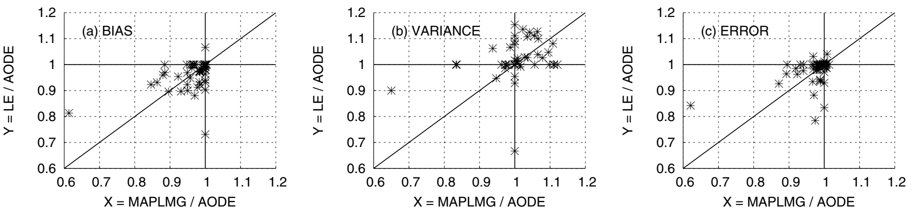


Fig. 4. LE and MAPLMG's performance relative to AODE

Between themselves, as shown in Table 2, LE wins against MAPLMG (although not significantly) on reducing bias (w/l/t being 26/19/12). MAPLMG wins against LE (although not significantly) on reducing variance (w/l/t being 23/16/18). The end effect is that MAPLMG beats LE (although not significantly) on reducing error (w/l/t being 26/19/12). Meanwhile, as shown in Figure 3, LE is more efficient than MAPLMG on both training and classification.

Hence, whether to use model selection or model weighing depends on the specific requirements of a particular classification task. If one needs to maximize accuracy, we recommend MAPLMG. If one seeks both high learning accuracy and efficiency, we recommend LE. If one needs to minimize variance while obtaining a reasonable accuracy, we recommend AODE.

6 Conclusion

We have studied a number of representative model selection and model weighing schemes for SPODE ensemble learning. We have presented the definition, rationale and time complexity of each scheme. We have conducted comprehensive experiments across 57 UCI benchmark data sets to test each scheme's effect on

SPODE ensembles' learning accuracy and efficiency. LE delivers efficient learning and significantly higher accuracy than AODE and thus is identified as the method of choice for model selection. MAPLMG delivers significantly higher accuracy than AODE and thus is identified as the method of choice for model weighing.

Acknowledgment

This research was supported by Australian Research Council grant DP0556279.

References

1. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning **29**(2) (1997) 131–163
2. Keogh, E.J., Pazzani, M.J.: Learning augmented Bayesian classifers: A comparison of distribution-based and classification-based approaches. In: Proceedings of the International Workshop on Artificial Intelligence and Statistics. (1999) 225–230
3. Keogh, E.J., Pazzani, M.J.: Learning the structure of augmented Bayesian classifiers. International Journal on Artificial Intelligence Tools **11**(40) (2002) 587–601
4. Kittler, J.: Feature selection and extraction. In Young, T.Y., Fu, K.S., eds.: Handbook of Pattern Recognition and Image Processing, New York (1986)
5. Kohavi, R.: Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid. In: Proceedings of the 2nd SIGKDD. (1996) 202–207
6. Kononenko, I.: Semi-naive Bayesian classifier. In: Proceedings of the 6th European Working Session on Machine learning. (1991) 206–219
7. Langley, P.: Induction of recursive Bayesian classifiers. In: Proceedings of the 4th ECML. (1993) 153–164
8. Langley, P., Sage, S.: Induction of selective Bayesian classifiers. In: Proceedings of the 10th UAI. (1994) 399–406
9. Pazzani, M.J.: Constructive induction of Cartesian product attributes. ISIS: Information, Statistics and Induction in Science (1996) 66–77
10. Sahami, M.: Learning limited dependence Bayesian classifiers. In: Proceedings of the 2nd SIGKDD. (1996) 334–338
11. Singh, M., Provan, G.M.: Efficient learning of selective Bayesian network classifiers. In: Proceedings of the 13th ICML. (1996) 453–461
12. Webb, G.I.: Candidate elimination criteria for lazy Bayesian rules. In: Proceedings of the 14th Australian AI. (2001) 545–556
13. Webb, G.I., Boughton, J., Wang, Z.: Not so naive Bayes: Aggregating one-dependence estimators. Machine Learning **58**(1) (2005) 5–24
14. Webb, G.I., Pazzani, M.J.: Adjusted probability naive Bayesian induction. In: Proceedings of the 11th Australian AI. (1998) 285–295
15. Xie, Z., Hsu, W., Liu, Z., Lee, M.L.: Snnb: A selective neighborhood based naive Bayes for lazy learning. In: Proceedings of the 6th PAKDD. (2002) 104–114
16. Zheng, Z., Webb, G.I.: Lazy learning of Bayesian rules. Machine Learning **41**(1) (2000) 53–84
17. Zheng, Z., Webb, G.I., Ting, K.M.: Lazy Bayesian rules: A lazy semi-naive Bayesian learning technique competitive to boosting decision trees. In: Proceedings of the 16th ICML. (1999) 493–502

18. Cerquides, J., de Mántaras, R.L.: Robust bayesian linear classifier ensembles. In: Proceedings of the 16th ECML. (2005) 72–83
19. De Ferrari, L.: Mining housekeeping genes with a naive Bayes classifier. MSc Thesis, University of Edinburgh, School of Informatics (2005)
20. Flikka, K., Martens, L., Vandekerckhove, J., Gevaert, K., Eidhammeri, I.: Improving throughput and reliability of peptide identifications through spectrum quality evaluation. In: Proceedings of the 9th Annual International Conference on Research in Computational Molecular Biology. (2005)
21. Nikora, A.P.: Classifying requirements: Towards a more rigorous analysis of natural-language specifications. In: Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering. (2005) 291–300
22. Yang, Y., Korb, K., Ting, K.M., Webb, G.I.: Ensemble selection for superparent-one-dependence estimators. In: Proceedings of the 18th Australian AI. (2005) 102–112
23. Zheng, F., Webb, G.I.: Efficient lazy elimination for averaged one-dependence estimators. In: Proceedings of the 23rd ICML. (2006)
24. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal **27**(3) (1948) 379–423
25. Schwarz, G.: Estimating the dimension of a model. Annals of Statistics **6** (1978) 461–5
26. Korb, K., Nicholson, A.: Bayesian Artificial Intelligence. Chapman & Hall/CRC, Boca Raton, FL (2004)
27. Hoeting, J.A., Madigan, D., Raftery, A.E., Volinsky, C.T.: Bayesian model averaging: A tutorial. Statistical Science **14**(4) (1999) 382417
28. Cooper, G. F., Herskovits, E.: A Bayesian method for constructing Bayesian belief networks from databases. In: Proceedings of the 7th UAI. (1991) 86–94
29. Pedregal, P.: Introduction to Optimization. Number 46 in Texts in Applied Mathematics. Springer (2004)
30. Heath, M.T.: Scientific Computing: An Introductory Survey, 2nd Edition. McGraw-Hill, New York (2002)
31. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/mlrepository.html] (1998) Department of Information and Computer Science, University of California, Irvine.
32. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the 13th IJCAI. (1993) 1022–1027
33. Breiman, L.: Bias, variance and arcing classifiers, technical report 460, Statistics Department, University of California, Berkeley (1996)
34. Friedman, J.H.: On bias, variance, 0/1-loss, and the curse-of-dimensionality. Data Mining and Knowledge Discovery **1**(1) (1997) 55–77
35. Kohavi, R., Wolpert, D.: Bias plus variance decomposition for zero-one loss functions. In: Proceedings of the 13th ICML. (1996) 275–283
36. Kong, E.B., Dietterich, T.G.: Error-correcting output coding corrects bias and variance. In: Proceedings of the 12th ICML. (1995) 313–321
37. Webb, G.I.: Multiboosting: A technique for combining boosting and wagging. Machine Learning **40**(2) (2000) 159–196
38. Moore, D.S., McCabe, G.P.: Introduction to the Practice of Statistics, Fourth Edition. Michelle Julet (2002)

Ensembles of Nearest Neighbor Forecasts

Dragomir Yankov[1], Dennis DeCoste[2], and Eamonn Keogh[1]

[1] University of California, Riverside CA 92507, USA,
{dyankov, eamonn}@cs.ucr.edu,
[2] Yahoo! Research, 3333 Empire Blvd., Burbank CA, USA,
decosted@yahoo-inc.com

Abstract. Nearest neighbor forecasting models are attractive with their simplicity and the ability to predict complex nonlinear behavior. They rely on the assumption that observations similar to the target one are also likely to have similar outcomes. A common practice in nearest neighbor model selection is to compute the globally optimal number of neighbors on a validation set, which is later applied for all incoming queries. For certain queries, however, this number may be suboptimal and forecasts that deviate a lot from the true realization could be produced.

To address the problem we propose an alternative approach of training ensembles of nearest neighbor predictors that determine the best number of neighbors for individual queries. We demonstrate that the forecasts of the ensembles improve significantly on the globally optimal single predictors.

1 Introduction

K-nearest neighbor (k-NN) methods for forecasting work by first identifying the k most similar time series to a given query and then, by combining their historical continuations, evaluate the expected outcome for the query. The methods are linear with respect to the model parameters, and yet they turn out to be suitable for predicting highly nonlinear fluctuations too. This is due to the fact that the identified neighbors themselves could comprise complex nonlinear patterns.

One significant drawback of the k-NN forecasts is their sensitivity to changes in the input parameters, e.g. the number of nearest neighbors, the weighting function, the prediction horizon or the length of the query vector. The impact of the number of neighbors is especially interesting as the resulting models may have intrinsically different characteristics. Namely, a forecast combining too many neighbors, quite often turns out to be biased and one that uses just a few of them, to have large variance. The effect is known as the *bias-variance dilemma* [5] and has been observed before in the context of k-NN forecasting [3,10]. Yet, no consistent approach has been suggested that could improve the forecasts of the method.

Here we propose a procedure, which rather than searching for the globally optimal k-NN predictor, constructs an ensemble of two predictors $\{k_1$-NN,

k_2-NN}, using a different number of neighbors k_1 and k_2 respectively. For each individual query the procedure selects one of the predictors from the ensemble to perform the forecast. Suppose that we have an oracle that looks at the actual continuation of every query and lets the method correctly pick the better of the two predictors. Studying the performance of such 'perfect' ensembles we observed the following effect. The ensembles which perform best, tend to have distant values for k_1 and k_2, with k_1 usually being very small (one or two nearest neighbors). What is also interesting, is that most of the ensembles, even those that are composed of suboptimal predictors, have better accuracy and stability than the globally optimal single predictor.

The observation suggests that it is often the case when we can split the queries into two distinct classes, one that requires a predictor using a small number of neighbors and another one that is better predicted with large number of neighbors. We can look at the globally optimal k-NN predictor as a safe compromise for the two classes, such that does not perform too poorly on each of them, but in general is not optimal for them either. Learning to separate those two classes could give us a powerful tool for improving on the k-NN models and in this work we demonstrate how this can be achieved. The potential of the approach to reduce both the bias and the variance of the NN forecasts is also illustrated.

The rest of the paper is organized as follows. In Section 2 we make an overview of the NN forecasting literature. Section 3 defines the k-NN forecasting framework. Section 4 describes the proposed method. An evaluation of the performance of our approach as compared to the optimal single k-NN predictor is presented in Section 5.

2 Related Work

Nearest neighbor forecasting methods have become popular with the advancement in dynamic systems. The relevance of the methods for time series, arising from such systems, is established by Taken's 'delay coordinate embedding theorem' [12]. It states that if there are enough observations, one can reconstruct the manifold representing the state space of the system. If a query is produced by the same system, then it should be part of the manifold too and its outcome will lie close to the outcome of its nearest neighbors.

The effectiveness of k-NN methods becomes apparent during the Santa Fe forecasting competition [2], when they are demonstrated to be competitive to other more complex methods as feedforward networks. Two entries from the competition that use a k-NN model, submitted by Sauer [10] and Casdagli et al. [3], show very good performance with Sauer taking second place on the Laser generated data set (see Section 5.1). Both Sauer and Casdagli et al. discuss the bias-variance problem of the forecasts but no principled approach for its solution has been suggested.

3 Formalization

Let a time series $\mathbf{y}(t) = (x_1, x_2, ..., x_t)$ be defined as a sequence of scalar observations measured at equal intervals in time. In its general form the forecasting problem targets the estimation of h consecutive future values, i.e. $\mathbf{y}_t(h) = (x_{t+1}, x_{t+2}, ..., x_{t+h})$, using any of the currently available observations from $\mathbf{y}$ (and possibly other time series). Here h is the user specified *prediction horizon.*

The available time series are organized in a training set by running a sliding window of size l along each of them. I.e. the set contains elements of the form $\mathbf{y}_t(l) = (x_{t+1}, x_{t+2}, ..., x_{t+l})$, called *lag* vectors. Note, that if the initial observations are long enough, for most elements $\mathbf{y}_t(l)$ the continuation $\mathbf{y}_{t+l}(h)$ will also be available in the training set. An estimate for the continuation of a query vector $\mathbf{q}(l)$ is then computed by the k-NN predictor as the linear combination:

$$\hat{\mathbf{q}}_c(h) = w_1 \mathbf{y}^1_{c_1}(h) + w_2 \mathbf{y}^2_{c_2}(h) + ... + w_k \mathbf{y}^k_{c_k}(h) \qquad (1)$$

where $\mathbf{y}^i_{c_i}(h)$ is the continuation of the i-th nearest neighbor starting at time point c_i, and $\mathbf{w}(q) = (w_1, w_2, ..., w_k)$ is a preselected weighting function (see Section 3.3).

Equation (1) gives the *direct* forecast of the k-NN algorithm for h steps ahead. A different type of prediction is the iterative one, where a single point is predicted at a time and is afterwards appended to the query vector for subsequent predictions. Iterative predictions are more accurate for short horizons, but as the prediction error accumulates faster, for long horizons, direct predictions tend to outperform them [8]. All results presented here are for direct forecasts, yet the proposed ensemble scheme could as well be applied for the iterative predictions.

3.1 Similarity Metric

The majority of the works on NN forecasting utilize the Euclidean distance [4,10], or some of its modifications. Two popular such modifications are the standardized Euclidean distance in which the series are transformed to have a mean zero and variance one [3], or the weighted Euclidean distance [9] which assigns lower weights to coordinates in the lag vector that are further in time from the target value. The weighted Euclidean distance complicates the model by adding another parameter to it, and it also makes the forecast more sensitive to the sizes of the lag vectors. The standardized Euclidean distance, on the other hand, is not very robust to noise and to non-stationary first and second moments.

In the current implementation the data are also standardized, but then the resulting vectors are compared with the scale-shift invariant distance metric, discussed by Goldin et al. [6]. If $\mathbf{q}$ is the query, $\mathbf{y}$ is the lag vector and $\mathbf{q}_s$ and $\mathbf{y}_s$ are their standardized representations, then the scale-shift transformation further changes $\mathbf{y}_s$ as: $\tilde{\mathbf{y}} = a\mathbf{y}_s + b$. The distance $d(\mathbf{q}, \mathbf{y})$ is now measured as the Euclidean distance (L_2) between the standardized query and the transformed neighbor, i.e. $d(\mathbf{q}, \mathbf{y}) = L_2(\mathbf{q}_s, \tilde{\mathbf{y}})$. The coefficients a and b are estimated using least square linear fit between $\mathbf{q}_s$ and $\mathbf{y}_s$.

3.2 Estimating the Prediction Accuracy

To evaluate the proposed procedure we measure the root mean square error (RMSE) between the actual outcome and the prediction, normalized respectively by the standard deviations of the query and the linear combination of its nearest neighbors: $RMSE = \sqrt{\frac{1}{h}\sum_{i=1}^{h}(\hat{q}_{t+i} - q_{t+i})^2}$. The residual $(\hat{q}_{t+i} - q_{t+i})$ stands for the difference between the scalar prediction and the true outcome for time point $(t + i)$. The RMSE has been the preferred error function for comparing forecasting accuracy in a number of time series prediction competitions. It is symmetric, i.e. both over or underestimating the true value are penalized equally, and measures the loss in the same units as the recorded variable. As the outcome and the forecast are normalized, for stationary time series the RMSE of the simple mean value predictor is equal to one. This further provides a baseline for comparing the goodness of a forecasting algorithm when evaluated on data sets as the Laser oscillations in Section 5.1.

3.3 Weighting Functions

There are two conceptually different weighting schemes, that a NN model can utilize, *kernel regression* and *locally weighted regression* (LWR) [1].

The *kernel* is a function of the distance between the query and its neighbors. One popular kernel is the *uniform* one, assigning equal weights to all neighbors. Atkeson et al. [1] argue that there is no clear evidence for the benefit of using a particular kernel function with NN-learning in general. While our experiments confirmed this observation, we also found out that the uniform kernel behaves more coherently across different data sets and when the input parameters are varied.

Rather than computing a distance function, LWR finds the linear combination of the neighbors that approximates most closely, in a least square sense, the query. The heuristic assumption is then made that the same linear combination of the neighboring continuations will also be the one that approximates best the query outcome. This heuristic holds for short horizons, but as the horizon increases, the assumption gets more unrealistic and the prediction performance becomes considerably poor.

For better consistency across different data sets or input parameters, the presented model utilizes the uniform kernel function.

4 Ensembles of NN Predictors

A general NN model selection procedure computes the globally best number of neighbors on a validation set, and then uses this number for forecasting all subsequent queries. There could be individual queries though, for which the forecasts are way off the true outcome and a different number of neighbors might be more suitable for them. Here we describe a procedure that improves on the error accuracy of the single NN predictors, by adapting to the individual queries.

Suppose that, rather than using a single k-NN predictor for the forecasts, we form the ensemble of two such predictors $Ens = \{k_1\text{-NN}, k_2\text{-NN}\}$. The ensemble works as follows. For every query $\mathbf{q}$, it selects this one of the *subpredictors* k_1-NN or k_2-NN that has a better forecasting accuracy on $\mathbf{q}$. For the time being let us neglect the issue of how exactly that selection is made and assume that Ens always makes the perfect choice. Table 1 lists some results for such ensemble predictors on the Impressions data set (see Section 5.2). It also compares them with the values for several single k-NN predictors[1].

Table 1. Validation error of several ensembles and several simple predictors

k	RMSE(k-NN)	(k_1, k_2)	RMSE(Ens)
1	2.0447	(1,20)	1.5829
2	1.9504	(2,40)	1.5996
6	1.8321	(6,1)	1.6305
100	2.9608	(100,1)	1.6095

The single predictor with the smallest validation error for this data set is k^*-NN = 6-NN. The pairs in column 3 are formed by fixing k_1 and selecting k_2 which minimizes the error of the resulting predictor. Among all such pairs the globally optimal one for the validation set is $Ens^* = \{$1-NN, 20-NN$\}$.

There are two important aspects in the above result. Firstly, note that the subpredictors using a relatively small number of neighbors form optimal ensembles with subpredictors that use a large number of neighbors and vice versa. And secondly, the ensembles, though composed of suboptimal predictors, perform with 10% to 15% better than the globally optimal single predictor 6-NN. To provide some insight on those two effects consider the histograms on Figure 1.

On the left histogram, the difference of the error RMSE(6-NN) - RMSE(1-NN) is spread evenly with much volume on both sides of the median. This means that for almost half of the queries 6-NN is not optimal and their forecasts could be improved. The right histogram shows the difference RMSE(6-NN) - RMSE(10-NN). As the forecasts of 6-NN are similar to these of 10-NN, it cannot clearly identify all those queries for which 6-NN is not optimal. In general, the flatter the histogram, and the more volume it has on both sides of the median, the more improvement will be introduced by the ensemble.

It turns out that it is also easier to find a classifier that might separate well the types of queries inferred by distant predictors, rather than by similar ones. Furthermore, as k-NN has in general smaller bias when k is small and smaller variance when k is large, this suggests that grouping distant predictors might result in ensembles that have the potential for improving both the bias and

[1] We report validation errors here to avoid test selection biases arising during our experimentation. All later experimental work reports true test error performance.

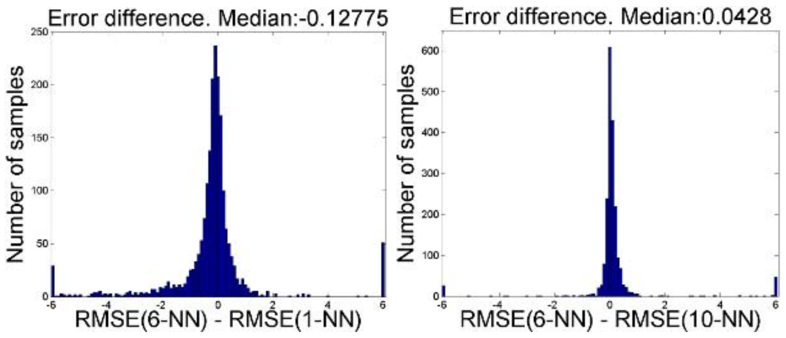


Fig. 1. Distant values have flatter histograms suggesting possible improvement

the variance of the single predictors. In Section 5.3 we provide some empirical evidence to support this conjecture.

4.1 Learning Classes of Queries

So far we have demonstrated that an ensemble of a biased and unbiased NN predictors outperforms the globally optimal single predictor, provided that for every query we correctly select the more accurate of the two subpredictors. Now we demonstrate that the two classes, inferred by those subpredictors, are often largely separable and could be successfully learned by a classifier that will assign them to the right predictor.

Feature Selection. Two type of features have been defined and used in the classifiers that we train in this work: statistical and performance related. The former include some statistical properties of the query and the identified neighbors. The second type deal with the performance of the subpredictors on part of the query or on its nearest neighbors.

Variance of the query, its nearest neighbors and the two forecasts. We measure the variance of all input vectors and that of the two forecasts. When the time series are non stationary (or contain a lot of noise), as $k_1 < k_2$, the forecast of k_2-NN due to the effect of aggregation usually varies less and yields smaller prediction error. On the contrary, when the time series are stationary or have very close neighbors in the training set, then increasing the number of neighbors also increases the chance of identifying an outlier. In those cases the forecasts of k_1-NN have smaller variance and are often closer to the true realization. The feature turns out to be a very strong indicator for the better subpredictor in some of the tested data sets.
Distances between the individual forecasts. When the individual forecasts of the first k_2 neighbors are very similar, then it is reasonable to have higher confidence in their combined forecast. Using k_2-NN will give us smoother and supposedly better forecast.

Performance on the nearest neighbors. We measure the accuracy of k_1-NN and k_2-NN on some of the nearest neighbors to the query. If the space is dense, then the better predictor for the neighbors will likely be the better predictor for the query too.
Step-back forecasts. Looking only at the beginning of the query we test which subpredictor forecasts its ending more accurately. The feature is a strong indicator for short horizon forecasts and for *self-similar* time series.

Classification. We look for a classifier to differentiate between the queries that are better predicted by any of the two subpredictors in the ensemble. Ideally, it should allow to be flexibly tuned between underfitting, when all samples are classified with the safe majority label, and overfitting the data. For the purpose, we use a Support Vector Machine (SVM) with a Gaussian kernel [11].

If $\mathbf{u}_i$ are the vectors of features (see Section 4.1) corresponding to the queries, and $\mathbf{v}_i$ are the respective labels (+1 for the dominant class, and -1 for the other one), then SVM classifies a test sample $\mathbf{u}$ according to the rule:

$$sgn(\mathbf{u}) = sgn\sum_{i=1}^{n}(\alpha_i \mathbf{v}_i K(\mathbf{u}_i, \mathbf{u}) + b) \tag{2}$$

where $0 \leq \alpha_i \leq C, i = 1..n$. In equation (2) α_i are the solution of the dual SVM optimization problem, b is a threshold also learned in the optimization, C is a parameter which determines the trade-off between the complexity and the training error of the classifier and needs to be specified prior to the optimization, and $K(\mathbf{u}_i, \mathbf{u})$ is a kernel function computing the distance between the test sample and a training sample in a highly dimensional feature space. The Gaussian kernel with width σ is defined as $K(\mathbf{u}_i, \mathbf{u})= \mathbf{exp}[-\|\mathbf{u}_i - \mathbf{u}\|^2/(2\sigma^2)]$, where σ also has to be specified in advance.

Using SVM with this type of kernel we can easily obtain the safe asymptotic classifier that underfits the data, for example by letting $C \to 0$ (see [7]). Then, by tuning the two parameters C and σ using cross-validation, a more optimal classification can be found. However, the procedure might become very computationally intensive as a quadratic search has to be performed within a very large set of values.

To find the best (C, σ) pair, we apply the heuristic described by Keerthi et al. [7]. They show that the solution of the equation $\log \sigma^2 = \log C - \log \tilde{C}$, where $\tilde{C}$ is the trade-off parameter for a linear SVM, provides a good approximation of the optimal C and σ. As both checking $\tilde{C}$ and the corresponding solutions of the equation require linear number of steps, the overall time for finding the approximation is linear.

An important aspect concerning the classification of the queries is that we do not need a classifier with perfect accuracy. It can make a lot of errors around the median of the histogram (Figure 1) but as long as the more distant queries are classified correctly the ensemble will still outperform the optimal simple predictor k^*-NN.

5 Empirical Results

The performance of the ensemble predictors is studied on two data sets - the laser oscillation data from the Santa Fe competition and real world data collected from web logs. Three horizons are considered: a relatively short one (30 steps ahead), a medium range one (60) and a long range horizon (100). The query size used is 30 time points.

Unless otherwise specified the best single predictor is compared with the ensemble {1-NN, 10-NN}. It tends to be a good choice in most cases as its subpredictors have distant values for the number of neighbors used, but in general is not globally optimal. When the improvement that we obtain with this ensemble is not significant enough, we also look at the performance of the optimal ensemble as inferred from a validation set. The result of the better of the two is displayed.

5.1 Laser Oscillation Data

The data represents the oscillation of a laser that can be modeled with a Lorenz system. It appears as *DataSet A* in the Santa Fe forecasting competition. The oscillations are comparatively easy to predict, but the transitions between the larger segments occur randomly. The available data are randomly split into a training (6000 elements), validation (2000) and test(2000) sets.

The best single predictor for all three horizons is the 3-NN predictor and the optimal ensemble, again for the three horizons, is {1-NN, 4-NN}. The hypothetically possible improvement in prediction error, by using the best ensemble with an oracle, over the best single predictor is: 16% (horizon 30), 17% (60), 16% (100).

For the single predictors, when increasing k beyond 3 the forecasts steadily begin to worsen. The good performance of the predictors with small number of neighbors is a result of the stationarity and the cleanness of the data. Two

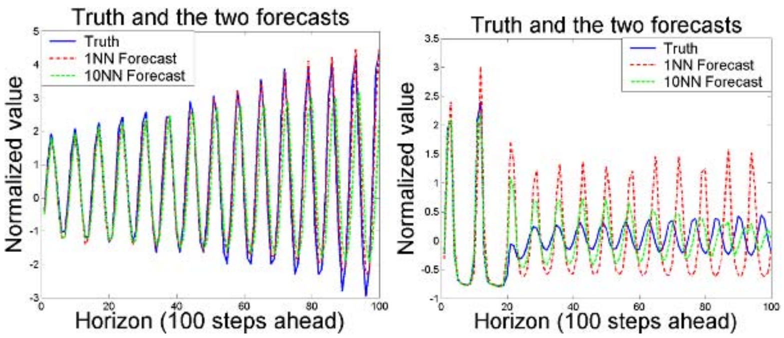


Fig. 2. *Left:* Example where 1-NN works better. *Right:* Example where more nearest neighbors perform better. *(Laser Oscillation Data)*

examples to illustrate the strengths and weaknesses of predictors with small and large number of neighbors are given on Figure 2.

When none of the neighbors predict any transition in the oscillation, using just one neighbor is usually preferable. Adding more neighbors increases the chance of selecting an outlier (Figure 2, *Left*). On the other hand, if many neighbors predict a transition, then increasing k makes it more likely to detect the actual amplitude of the oscillation after the transition (Figure 2, *Right*).

The test set accuracy of the SVM classifier separating the samples into groups, better predicted by k_1-NN or k_2-NN is summarized in Table 2 (column 2). For this data set the samples are equally distributed among the two groups. We found out that this is another premise for a better performance of the ensemble method. The table also lists the prediction test error and its standard deviation for the three horizons. The ensemble improves on both of the components over the optimal 3-NN predictor.

Table 2. Test error and classification accuracy. *(Laser Oscillation Data)*

Horizon	Class. Accuracy	Predictor	Test RMSE	Std
$h = 30$	0.75	3-NN (optimal k)	0.124	0.132
		Ens ={1-NN,10-NN}	**0.120**	**0.130**
$h = 60$	0.74	3-NN (optimal k)	0.207	0.170
		Ens^* ={1-NN,4-NN}	**0.189**	**0.162**
$h = 100$	0.81	3-NN (optimal k)	0.355	0.226
		Ens ={1-NN,10-NN}	**0.329**	**0.213**

For horizon 60 the improvement that we obtained with {1-NN, 10-NN} was not significant. We also check the performance of the globally optimal ensemble according to the validation set, in this case Ens^* ={1-NN, 4-NN}.

Finally, Figure 4 *Left* summarizes the percentage improvement of the ensemble forecast compared to the best single predictor. The improvement in test error is between 25% and 50% of the hypothetically possible improvement measured earlier on the validation set.

5.2 Web Site Impressions Data

The time series represent the number of *impressions* (users that have seen the banner ads), for a set of web sites, recorded over a period of three years.

The series have weekly recurrences, often with seasonal trends, a lot of noise and are highly nonstationary. The training set contains approximately 50 000 vectors, the validation and the test set have 2000 samples each. As a very small portion of the samples increase the error with orders of magnitude, to be fair to the representative majority of the samples we look at the 95%-quantile of the error and its deviation.

The optimal single predictors computed on the validation set are: 10-NN (horizon 30), 8-NN (60), 6-NN (100). The optimal ensembles are: {3-NN, 100-NN} (30), {1-NN, 30-NN} (60) and {1-NN, 20-NN} (100). The hypothetical margin for improvement if using an oracle is: 14% (30), 13% (60), 14% (100).

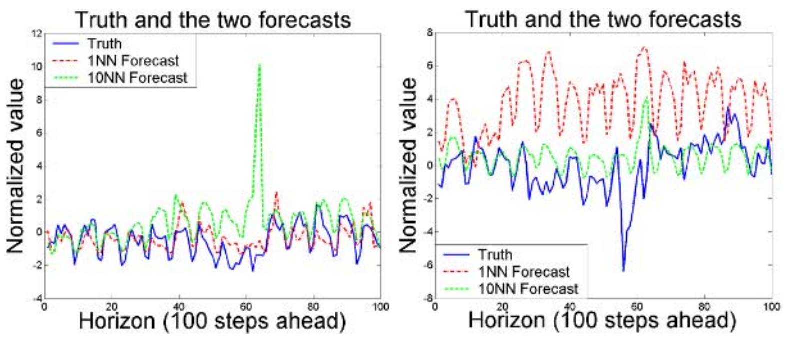


Fig. 3. *Left:* Example where 1-NN works better. *Right:* Example where more nearest neighbors perform better. The smoother forecast is usually the more accurate one. *(Web Impressions Data)*

Due to the large amount of noise, the more conservative, i.e. the smoother forecasts, are usually the more accurate ones (Figure 3). Therefore, the statistical features discriminate quite well between the two classes for the extreme examples, i.e. the examples for which applying the wrong subpredictor increases the error significantly. On the other hand the random spikes and drops in the data are hard to forecast and for these samples the assigned subpredictor is often incorrect. Because of the above two effects, the accuracy of the SVM classifier is comparatively low, but the overall improvement introduced by the ensemble method is quite good (see Table 3). The improvement in the test error and its deviation (Figure 4 *Right*) is between 60% and 70% of the hypothetically possible improvement computed on the validation set.

Table 3. Test error and classification accuracy. *(Web Impressions Data)*

Horizon	Class. Accuracy	Predictor	Test RMSE	Std
$h = 30$	0.58	10-NN (optimal k)	1.1235	0.644
		Ens ={1-NN,10-NN}	**1.021**	**0.452**
$h = 60$	0.77	8-NN (optimal k)	1.549	0.862
		Ens ={1-NN,10-NN}	**1.412**	**0.685**
$h = 100$	0.58	6-NN (optimal k)	1.8676	1.183
		Ens ={1-NN,10-NN}	**1.6881**	**0.961**

5.3 Bias-Variance Improvement

The squared loss of a predictor decomposes into the following two components [5]:

$$\mathcal{E}_{\mathcal{D}}[\{\hat{\mathbf{q}} - \mathbf{q}\}^2] = \underbrace{\{\mathcal{E}_{\mathcal{D}}[\hat{\mathbf{q}}] - \mathbf{q}\}^2}_{bias^2} + \underbrace{\mathcal{E}_{\mathcal{D}}[\{\hat{\mathbf{q}} - \mathcal{E}_{\mathcal{D}}[\hat{\mathbf{q}}]\}^2]}_{variance} \tag{3}$$

where the expectations are computed over a number of different training sets $\mathcal{D}$.

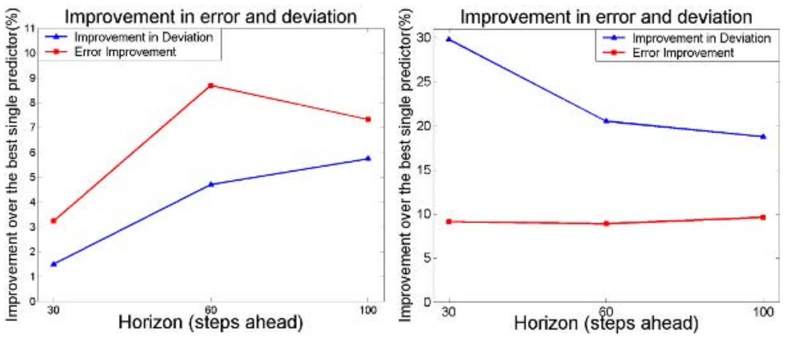


Fig. 4. Test error improvement of the ensemble approach over the best single predictor. *Left*: Laser Oscillation Data. *Right*: Web Impressions Data.

In the previous experiments it was demonstrated that the ensembles can decrease the test error, and hence the overall loss of the single predictors. It is essential to understand whether that improvement originates from one or both of the components in equation 3.

From the larger of the data sets, the Impressions data, we draw 50 random replicas, of size 90% of the original training set size. For every query in the test set, the bias and the variance over the replicas $\mathcal{D}$ are computed. The average bias and variance over all queries, for horizon 100, is presented in Table 4.

Table 4. Bias and variance for horizon 100 on the Impressions data set. The ensemble improves on both of the components.

Predictor	Bias^2	Variance
1-NN	5.468	1.174
6-NN (optimal k)	5.042	0.638
10-NN	5.690	1.96
Ens ={1-NN,10-NN}	**3.721**	**0.204**

As seen from the table, the ensembles can decrease both terms in the squared loss decomposition, which suggests that they are a potentially powerful approach towards the bias-variance problem of the k-NN forecasts.

6 Conclusion

We have presented a method for learning how to separate time series queries into two classes, that are better predicted with one of two possible NN predictors. The experimental evaluation shows that such ensembles have better prediction error, compared to the single globally optimal k-NN predictor.

The work raises some interesting questions. For example, what kind of improvement would one expect, if the ensembles include more than two subpredictors. The results indicate that essential for the performance is the identification of bad cases for the individual predictors. Including more subpredictors adds more alternatives to select from, when forecasting these bad samples. On the other hand, the multiway classification might have lower accuracy. Another interesting research direction is how to combine models that differ with respect to other input parameters, such as weighting function, query lengths, or prediction horizon. In this case a criterion for what models should be combined and different features, characteristic of the new models, need to be derived.

References

1. C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 1996.
2. A.Weigend and N. Gershenfeld. *Time Series Prediction. Forecasting the Future and Understanding the Past.* Addison-Wesley Publishing Company, 1994.
3. M. Casdagli and A. Weigend. Exploring the continuum between deterministic and stochastic modeling. *Time Series Prediction. Forecasting the Future and Understanding the Past*, 59(8):347–366, August 1994.
4. J. Farmer and J. Sidorowich. Predicting chaotic time series. *Physical Review Letters*, 59(8):845–848, August 1987.
5. S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1 – 58, August 1992.
6. D. Goldin and P. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. *Lecture Notes in Computer Science*, 976(7):137–153, January 1995.
7. S. Keerthi and C. Lin. Asymptotic behaviors of Support Vector Machines with Gaussian kernel. *Neural Computation*, (15):1667–1689, 2003.
8. J. McNames, J. Suykens, and J. Vandewalle. Winning entry of the K.U.Leuven time series prediction competition. *Internation Journal of Bifurcation and Chaos,*, 9(8):1485–1500, August 1999.
9. D. Murray. Forecasting a chaotic time series using an improved metric for embedding space. *Physica D*, 68(8):318–325, August 1993.
10. T. Sauer. Time series prediction by using delay coordinate embedding. *Time Series Prediction. Forecasting the Future and Understanding the Past*, 59(8):175–193, August 1994.
11. B. Schölkopf and A. Smola. *Learning with Kernels.* MIT Press, 2002.
12. F. Takens. Detecting strange attractors in turbulence. *Lecture Notes in Mathematics, Dynamical Systems and Turbulence*, 898(7):366–381, January 1981.

Learning Process Models with Missing Data

Will Bridewell, Pat Langley, Steve Racunas, and Stuart Borrett

Computational Learning Laboratory, CSLI,
Stanford University, Stanford, CA 94305 USA
{willb, langley, sracunas, sborrett}@csli.stanford.edu

Abstract. In this paper, we review the task of inductive process modeling, which uses domain knowledge to compose explanatory models of continuous dynamic systems. Next we discuss approaches to learning with missing values in time series, noting that these efforts are typically applied for descriptive modeling tasks that use little background knowledge. We also point out that these methods assume that data are missing at random—a condition that may not hold in scientific domains. Using experiments with synthetic and natural data, we compare an expectation maximization approach with one that simply ignores the missing data. Results indicate that expectation maximization leads to more accurate models in most cases, even though its basic assumptions are unmet. We conclude by discussing the implications of our findings along with directions for future work.

1 Introduction

Consider the challenge of collecting ecological data from the Southern Ocean. The location is remote, the climate can be brutal, and scientists have limited resources, which forces them to carefully plan and prioritize their collection efforts. To accomplish this task, scientists schedule observation cruises when and where they anticipate that phenomena of primary interest will occur. However, the spatial and temporal variability of ecological phenomena further hampers data collection. The phenomena may not occur where anticipated, may happen before or after a cruise, or may last longer than a single research cruise can remain at sea. One strategy to address this issue is to make multiple cruises, but even if this approach is successful, there will be omissions in the data. Yet scientists still want to build models and determine parameter values to explain the data and understand the system.

This scenario highlights a number of issues. First, the gathered data will likely contain large gaps that were engineered from the start. Second, these gaps depend partly on the expected values of the missing data—they are not missing at random. Third, even though the gathering efforts are engineered to contain the most important information, the timing may be off. In addition, instruments may malfunction and some environmental values may fall outside measurable ranges. Despite the scientists' best efforts, important information about system dynamics may be missing.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): ECML 2006, LNAI 4212, pp. 557–565, 2006.

All these situations, which are not unique to large ecological expeditions, leave the scientist with an interesting and complex problem: how can one build a model of a nonlinear, dynamic system when key measurements are missing? Inductive process modeling (Langley et al. 2002) provides a method for building quantitative explanations from time series, but it assumes that the relevant data are available. In comparison, ARIMA methods lead to purely descriptive models, but researchers often augment them with a variant of expectation maximization (EM; Dempster et al. 1977) to handle missing values (Isaksson 1991; Stoica et al. 2005). In this paper, we determine whether EM can be adapted to assist inductive process modeling and to function in realistic scientific settings.

In the pages that follow, we apply an EM variant called EMP to produce an explanatory model of scientific data and compare its behavior to a baseline method that ignores the missing values. The next section describes the inductive process modeling paradigm and introduces the baseline and EM approaches to handling missing observations. After this, we report experiments with synthetic and natural data and present an analysis of the results. In closing, we discuss related work and suggest directions for future research.

2 Handling Missing Data in Inductive Process Modeling

The approach we report here extends earlier work on inductive process modeling (Langley et al. 2002; Todorovski et al. 2005). The discovery task can be stated:

- *Given*: trajectories for a set of continuous variables over time;
- *Given*: background knowledge cast in terms of generic entities and processes;
- *Given*: observable and theoretical entities and variables to be modeled;
- *Find*: a process model that explains the observed trajectories and generalizes accurately to new data.

The task revolves around the notion of a quantitative process model, which provides a causal account of how variables change over time. Todorovski et al. (2005) describe the process model representation, which consists of distinct processes that organize numeric relations among variables that are associated with known entities, and introduce HIPM, a program that induces process models.

Inductive process modeling should lead to a mathematical model of a system that both improves our understanding of that system and enables us to predict its behavior under altered conditions. Additionally, one could use the model to reconstruct unobserved points in a set of trajectories—a use that suggests a solution for handling missing data. For instance, given a model, we could substitute its output for the missing values back into the original data set and learn a new model from the result. This approach falls into the expectation maximization (EM) class of techniques (Dempster et al. 1977).

The EM algorithm is an iterative approach to learning a model from data with missing values that has four main steps:

1. select an initial set of parameters for a model
2. determine the expected values for the missing data

3. induce new model parameters from the union of the expected values and the original data
4. unless the parameters have converged, return to Step 2 using the new model

To be applicable without further complication, EM assumes that the mechanism responsible for the missing data is ignorable. Specifically, the data must be either *missing completely at random*, which means that the mechanism is independent of the observations, or *missing at random*, where the data may influence it (Little and Rubin 2002). Unfortunately, scientific data sets rarely meet these criteria.

Missing values in scientific domains arise from a combination of resource constraints, working hypotheses, and other reasons both practical and accidental. Although variants of EM exist for such "non-ignorable" mechanisms, they require a collection of data sets produced with the same missing-data mechanism—a luxury not typical to scientific research. In response, we chose to violate the assumption of an ignorable mechanism and experimentally evaluate the utility of an EM variant under these conditions. Our specific algorithm, which we call EMP, follows the general EM outline given earlier:

1. substitute linearly-interpolated values for the missing data
2. use HIPM to find the model that minimizes the sum of squared errors
3. simulate the new model
4. substitute the results of the simulation for the missing data
5. if the model has changed and the maximum number of iterations has not been exceeded, go to Step 2

This algorithm differs from the previous outline in that the missing data are initially replaced with rough estimates and that HIPM selects the first model as well as the subsequent ones. In addition, estimating parameters for a nonlinear system remains an unsolved problem, and we can guarantee neither a maximization nor an improved estimate in Step 2. Thus we introduce a maximum number of iterations to force the program to halt. In the next section, we compare the results from EMP to those from a baseline that ignores the missing data.

3 Experimental Evaluation

In the last section, we established that EM is not an ideal fit for the missing data problem that we encounter, but we also saw that a variant of EM may be a reasonable solution in practice. To test this conjecture, we performed experiments with synthetic and natural data for a two population predator–prey system and with natural data from a more complex ecosystem. The synthetic case allows us to control the nature of the noise in the trajectory and to determine how accurately HIPM can recreate the structural and parametric form of the generating model. The natural data gives a more realistic view of EM's capabilities in the presence of complicated and unknown noise models.

To generate synthetic data, we built a predator–prey model that produces a stable oscillation as would be expected in an ideal system. This model includes a

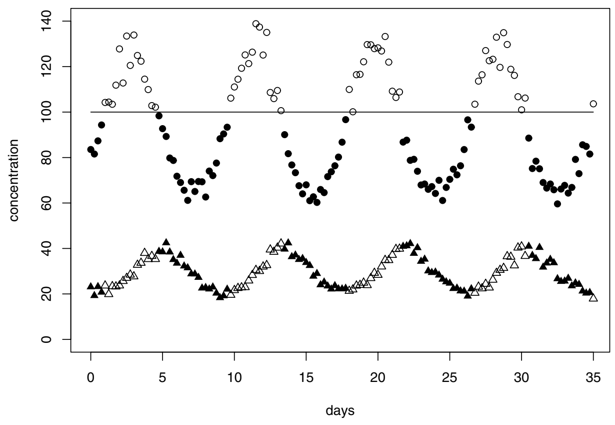


Fig. 1. This figure shows the synthetic predator–prey data used in the experiments. The horizontal line denotes the cut point for the peaks in the prey concentration. Unfilled shapes indicate missing values.

process for logistic growth of the prey, exponential death of the predator, and a Holling type 2 function (Holling 1959) for predation. We simulated the model to mimic experimental conditions where four measurements are taken each day for 35 days, which gave a total of 141 observations including the initial conditions. To generate the final trajectories, we added five percent multiplicative, Gaussian noise to each observation.

After generating the data, we altered the trajectories to produce a plausible worst-case scenario for model induction. For this experiment, we assumed that peaks in the prey population were of primary importance, so we removed them by deleting all observations where the concentration of prey exceeded 100 parts/volume. This operation left roughly half of the data for training purposes, as shown in Figure 1. For the baseline condition, HIPM searched exhaustively through a space of 22 model structures to fit the corrupted training data and tested the resulting model on the original, noise-free trajectories. We used the same search conditions to test EMP, which performed 20 iterations and reported the model with the lowest sum of squared error over all the iterations.

To evaluate EMP on observations from a real system, we used two data sets initially collected by Veilleux (1976) and made available by Jost and Ellner (2000). In his experiments, Veilleux observed the interaction of two protist species in an artificial environment over several days. Since it typically took a few days for these ecosystems to establish a stable frequency, we use a subset of the provided values. Specifically, we use the observations shown in Jost and Ellner's Figure 1a starting at day 8.5 and those in their figure 1c starting at day 11. The resulting data sets contain 54 samples with five full peaks and 30 samples with three full peaks, respectively.

As with the synthetic data, we removed the portions of the Veilleux trajectories that contain the prey's peak values. Here we tested under two conditions.

Table 1. Results on synthetic and natural predator–prey data. The mean squared error (best scores in bold) and coefficient of determination (r^2) are reported for the best models produced by the baseline approach and EMP.

Data	Mean Squared Error		Predator r^2		Prey r^2	
	Base	EMP	Base	EMP	Base	EMP
synthetic	44.04	**13.34**	0.88	0.97	0.89	0.95
1a minor	2073.93	**1925.28**	0.65	0.64	0.63	0.62
1a major	**2580.81**	2636.63	0.55	0.58	0.54	0.55
1c minor	245.42	**231.98**	0.87	0.87	0.91	0.90
1c major	408.67	**249.11**	0.88	0.87	0.90	0.90

In the first case, we cut out the peak value and one or two neighboring points, slightly shaving off the peak. For the second case, we removed the peak value and three to four surrounding points, imposing a deeper cut. Note that a full cycle, from trough to trough, contains ten samples on average, so the second scenario uses roughly half of the total data. HIPM fit each data set independently by searching the same 22 structures used with the synthetic data. We carried out the experiments with EMP in the manner previously described and measured each model's accuracy by testing it against the original, uncut data.

Table 1 shows the results for the predator–prey experiments. In all but one case, EMP produced models with substantially better fits to the original trajectories than did the baseline approach. Interestingly, neither method reproduced the correct model structure for the synthetic data, although HIPM can recover it from perfect data. Notice also that the coefficients of determination for both variables (r^2) are roughly the same across methods. This result suggests that EMP helps HIPM fit the amplitude of the trajectories, but does not affect its ability to fit their shapes. Plots of the trajectories, such as the one in Figure 2, show that, in all cases, the corresponding models provided close visual fits to the frequency in both the 1a and 1c data sets. In addition, the models fit the amplitude of the 1c data quite well, but produced peaks roughly half the height of those observed in the 1a data.

The final experiments evaluate our approach with data from the Ross Sea in the Southern Ocean. This domain differs from the previous two in that the space of model structures is much larger and the data contain a single peak in the primary variable. For the experiments we used two data sets (RS1 and RS2) provided by Kevin Arrigo, the oceanographer in our group. Each set contains 188 preprocessed, daily observations of phytoplankton and nitrate concentrations, along with values for the amount of available light. In both cases the recordings were made over the summer when a single phytoplankton bloom occurred.

We removed 32 samples from the first year of data and 25 samples from the second, based on the location of the phytoplankton peaks. Since light serves as a driving variable, we provided its value in all cases. After preparing the data, we had the program fit each set independently and compare the results against

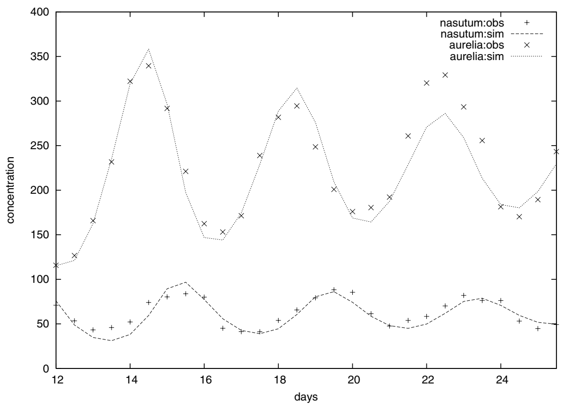


Fig. 2. This figure presents the trajectories produced by EMP's best model for the Veillieux 1c data along with the observed values

the original measurements. Due to the size of the search space defined by the associated generic process library, we ran HIPM in beam-search mode with a beam width of eight. For each training cycle, the program considered an average of 126.7 model structures. As before, EMP ran a total of 20 cycles using the same settings for HIPM in all cases and returned the model with the lowest sum of squared errors.

Table 2 shows the results on the Ross Sea data. In both cases, EMP substantially outperforms the baseline in terms of both mean squared error and r^2. We see more than a 50% reduction in error and, although the r^2 for phytoplankton decreases a bit, the increase for nitrate is phenomenal. Without the use of EMP, HIPM predicted a flat line for the nitrate concentration. In summary, EMP not only reduced error but also improved the conceptual model by accounting for all the observed variables.

The results presented above paint a highly positive picture of the EM approach to handling missing data in inductive process modeling. On both the synthetic and Ross Sea data, the extra computational time led to much better fits, whereas the fits on the Veilleux data were mostly improvements. In the next section, we review the experimental results, suggest further work in this area, and discuss related research.

4 Discussion

Even though EMP is hampered by an unspecified missing data mechanism and asked to operate in a worst-case scenario, it behaved quite well. Looking more closely at the results on the Veilleux data set, we can conjecture why EM was less helpful in some cases and plan studies that could clarify the reasons. First, we note that the behavior of both EMP and the baseline on the 1c data matches what we see when HIPM induces a model from the complete data. Thus, EMP

Table 2. Results on data from the Ross Sea. The mean squared error (best scores in bold) and coefficient of determination (r^2) are reported for the best models produced by the baseline approach and EMP.

Data	Mean Squared Error		Phytoplankton r^2		Nitrate r^2	
	Base	EMP	Base	EMP	Base	EMP
RS1	30.13	**13.56**	0.98	0.96	0.00	0.87
RS2	25.27	**9.93**	0.93	0.80	0.00	0.93

was likely hitting a performance ceiling, and enough information remained in the data for HIPM to build an accurate model even in the baseline condition. This result is somewhat surprising, since over one-third of the data were removed. Second, we could make a similar argument for the 1a data, but performance of both approaches degrades when we remove more of the data. This finding matches intuition and indicates that we corrupted the data enough to affect HIPM's performance.

We should also explain why experiments with the synthetic predator–prey data highlighted the difference between EMP and the baseline more clearly than those using the Veilleux sets. The most obvious difference in these two cases is the nature of the noise in the observations. The synthetic data used a multiplicative, Gaussian model whereas the noise mechanism of the natural data is unknown. Further experiments with alternative noise models may help clarify the effect of noise on both methods and determine whether it accounts for the discrepancies seen in the results.

Although our approach to the missing data problem is related to previous techniques, we have adapted it to the inductive process modeling task and examined its ability to work on scientific data. Process modeling, which we described earlier in the paper, descends from research on equation discovery (e.g., Langley 1981; Żytkow et al. 1990; Džeroski and Todorovski 1995; Washio et al. 2000), but it differs in that it takes background knowledge as input and outputs explanatory models, as opposed to descriptive ones. Our emphasis on differential equations bears some resemblance to work by Bradley and colleagues (2001) and Todorovski (2003), but these approaches lack a a strong attachment to scientifically meaningful processes.

We could also characterize inductive process modeling as a combination of qualitative physics and system identification. In particular, our approach groups equations into a more qualitative, process-based structure like that developed by Forbus (1984), and our use of generic processes to encode background knowledge resembles work in compositional modeling (e.g., Falkenhainer and Forbus 1991), where abstract components are instantiated and assembled to form models. The relationship to system identification (Åström and Eykhoff 1971) lies in our concern with learning parametric models from time series. Inductive process modeling differs from this paradigm in its incorporation of search through a space of model structures.

Although this paper indicates that EM is an appropriate technique for handling missing data when learning process models, more work in this area remains. Here, we concentrated on the case where large portions of data are unavailable, but other situations often arise. In some cases, the variables may be measured at different intervals, which in extreme cases results in a collection of examples that are missing all but one value. In other cases, certain variables may be recorded only at specific times, as occurs when data sets are merged from multiple sources, each reflecting different interests and resource constraints. This situation can cause large gaps in individual variables without affecting the rest of the data. We conjecture that EM-style techniques will be useful in these situations, but we need experiments to test this prediction.

Acknowledgements

This research was supported by Grant No. IIS-0326059 from the National Science Foundation. We thank Ljupčo Todorovski for design and implementation of the HIPM system and Oren Shiran for extensions to the software.

References

Åström, K. J., Eykhoff, P.: System identification—a survey. Automatica **7** (1971) 123–167

Bradley, E., Easley, M., Stolle, R.: Reasoning about nonlinear system identification. Artificial Intelligence **133** (2001) 139–188

Dempster, A. P., Laird, N. M., Rubin, D. B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B (Methodological) **39** (1977) 1–38

Džeroski, S., Todorovski, L.: Discovering dynamics: From inductive logic programming to machine discovery. Journal of Intelligent Information Systems **4** (1995) 89–108

Falkenhainer, B., Forbus, K. D.: Compositional modeling: Finding the right model for the job. Artificial Intelligence **51** (1991) 95–143

Forbus, K.: Qualitative process theory. Artificial Intelligence **24** (1984) 85–168

Holling, C. S.: Some characteristics of simple types of predation and parasitism. Canadian Entomologist **91** (1959) 385–398

Isaksson, A.: System identification subject to missing data. American Control Conference (1991) 693–698

Jost, C., Ellner, S.: Testing for predator dependence in predator-prey dynamics: A non-parametric approach. Proceedings of the Royal Society of London B **267** (2000) 1611–1620

Langley, P.: Data-driven discovery of physical laws. Cognitive Science **5** (1981) 31–54

Langley, P., Sánchez, J., Todorovski, L., Džeroski, S.: Inducing process models from continuous data. The Nineteenth International Conference on Machine Learning (2002) 347–354

Little, R. J., Rubin, D. B.: Statistical analysis with missing data (2nd ed.). Hoboken, NJ: John Wiley

Stoica, P., Xu, L., Li, J.: Parameter estimation with missing data via equalization-maximization. IEEE International Conference on Acoustics, Speech, and Signal Processing (2005) IV–57–IV–60

Todorovski, L.: Using domain knowledge for automated modeling of dynamic systems with equation discovery Doctoral dissertation, Faculty of computer and information science, University of Ljubljana (2003)

Todorovski, L., Bridewell, W., Shiran, O., Langley, P.: Inducing hierarchical process models in dynamic domains. The Twentieth National Conference on Artificial Intelligence (2005) 892–897

Veilleux, B. G.: The analysis of a predatory interaction between didinium and paramecium. Master's thesis, University of Alberta (1976)

Washio, T., Motoda, H., Niwa, Y.: Enhancing the plausibility of law equation discovery. The Seventeenth International Conference on Machine Learning (2000) 1127–1134

Żytkow, J. M., Zhu, J., Hussam, A.: Automated discovery in a chemistry laboratory. The Eighth National Conference on Artificial Intelligence (1990) 889–894

Case-Based Label Ranking

Klaus Brinker and Eyke Hüllermeier

Data and Knowledge Engineering,
Otto-von-Guericke-Universität Magdeburg, Germany
{brinker, huellerm}@iti.cs.uni-magdeburg.de

Abstract. Label ranking studies the problem of learning a mapping from instances to rankings over a predefined set of labels. We approach this setting from a case-based perspective and propose a sophisticated k-NN framework as an alternative to previous binary decomposition techniques. It exhibits the appealing property of transparency and is based on an aggregation model which allows one to incorporate a variety of pairwise loss functions on label rankings. In addition to these conceptual advantages, we empirically show that our case-based approach is competitive to state-of-the-art model-based learners with respect to accuracy while being computationally much more efficient. Moreover, our approach suggests a natural way to associate confidence scores with predictions, a property not being shared by previous methods.

1 Introduction

Label ranking, a particular preference learning scenario, studies the problem of learning a mapping from instances to rankings over a finite number of predefined *labels* (alternatives). It can be considered as a natural generalization of the conventional classification problem, where only a single label (the top-label) is requested instead of a ranking of all labels. Previous *model-based* approaches on extending classification typically decompose ranking problems into multiple binary classification subtasks (e.g., *ranking by pairwise comparison* (RPC) [1]), or generate inflated training sets embedded in a higher dimensional feature space to encode binary preferences among the labels (e.g., *constraint classification* (CC) [2]). Hence, the computational complexity of both learning (and evaluating) a prediction model increases substantially in comparison to standard classification for these methods.

In Sections 2 and 3, we present a novel *case-based* approach to label ranking which is *conceptually simpler* and *computationally less complex.* The essential idea is to view label ranking as a rank aggregation problem within a case-based framework, where the modular architecture allows one to use a variety of procedures for aggregation. Moreover, our approach suggests a natural way to associate confidence scores with predictions, a property not being shared by previous methods. In Section 4, we empirically show that our case-based approach is competitive to state-of-the-art model-based learners with respect to accuracy while being computationally much more efficient, and that our framework indeed provides reliable confidence scores.

2 Framework

In label ranking, the problem is to learn a mapping from instances x of an instance space $\mathcal{X}$ to rankings $\succ_x$ (total strict orders) over a finite set of labels $\mathcal{L} = \{\lambda_1 \dots \lambda_c\}$, where $\lambda_i \succ_x \lambda_j$ means that instance x prefers label λ_i to λ_j. A ranking over $\mathcal{L}$ can be represented by a permutation τ of $\{1 \dots c\}$, where $\tau(i)$ denotes the position of the label λ_i in the ranking. The target space of all permutations over c labels will subsequently be referred to as $\mathcal{S}_c$. Let us make the idealized assumption that the training data submitted to the learning algorithm consists of a set of examples $(x_1, \tau_1) \dots (x_m, \tau_m)$ which contain the *complete* label rankings and therefore the entire sets of pairwise preferences. Of course, in practice it might not always be possible to observe complete rankings. However, by reducing the technical complexity, this assumption will allow us to focus on the main components of case-based label ranking. Later on, we will sketch how to handle the more general case where only a subset of all pairwise preferences is available for each training example x_i.

In the following, we will discuss a general case-based framework for learning label rankings. The k-nearest neighbor algorithm (k-NN) is arguably the most basic case-based learning method [3]. In its simplest version, it assumes all instances to be represented by feature vectors $x = ([x]_1 \dots [x]_N)^\top$ in the N-dimensional space $\mathcal{X} = \mathbb{R}^N$ endowed with the standard Euclidean metric as a distance measure, though an extension to other instance spaces and more general distance measures $d(\cdot, \cdot)$ is straightforward. When a query feature vector x is submitted to the k-NN algorithm, it retrieves the k training instances closest to this point in terms of $d(\cdot, \cdot)$. In the case of classification learning, the k-NN algorithm estimates the query's class label by the most frequent label among these k neighbors. It can be adapted to the regression learning scenario by replacing the majority voting step with computing the (weighted) mean of the target values.

In order to generalize the k-NN algorithm to ranking in a suitable manner one has to incorporate the structured nature of the space of label rankings. Our approach considers aggregation techniques for label ranking which are conceptually related to averaging in k-NN regression learning. To this end, we incorporate a common rank aggregation model in order to combine the k nearest neighbors into a single ranking. The *consensus label ranking* is computed such that it minimizes the sum of pairwise loss values with respect to all k rankings. The corresponding formal model will be detailed in Section 3.

3 Aggregating Label Rankings

The problem of aggregating rankings arises in a variety of applications such as, e.g., the combination of meta-search results [4]. In order to analyze the problem of aggregating label rankings in a formal manner, let $\tau_1 \dots \tau_k$ denote rankings of the c alternatives $\lambda_1 \dots \lambda_c$. A common method to measure the quality of a ranking τ as an aggregation of the set of rankings $\tau_1 \dots \tau_k$ is to compute the sum of pairwise loss values $L(\tau) \overset{\text{def}}{=} \sum_{i=1}^{k} l(\tau, \tau_i)$ with respect to a loss function

$l : \mathcal{S}_c \times \mathcal{S}_c \to \mathbb{R}_{\geq 0}$ defined on pairs of rankings. Having specified a loss function $l(\cdot)$, this model leads to the optimization problem of computing a ranking $\hat{\tau} \in \mathcal{S}_c$ (not necessarily unique) such that

$$L(\hat{\tau}) = \min_{\tau \in \mathcal{S}_c} \sum_{i=1}^{k} l(\tau, \tau_i). \tag{1}$$

Common choices for the loss function are the *Kendall tau loss* [5], the sum of absolute rank distances, which is also referred to as *Spearman footrule loss* [4], and the sum of squared rank distances. The linear transformation of the latter loss function into a $[-1, 1]$-valued similarity measure is well-known as the *Spearman rank correlation coefficient* [6]. The Kendall tau loss l_K essentially calculates the number of pairwise rank inversions on labels to measure the ordinal correlation of two rankings. More formally,

$$l_K(\tau, \tau') \overset{\text{def}}{=} \left|\{(i,j) \mid \tau(i) < \tau(j) \wedge \tau'(i) > \tau'(j)\}\right|. \tag{2}$$

The Spearman footrule loss l_1 and the sum of squared rank distances loss l_2 are formally defined, respectively, as

$$l_1(\tau, \tau') \overset{\text{def}}{=} \sum_{i=1}^{c} |\tau(i) - \tau'(i)| \quad \text{and} \quad l_2(\tau, \tau') \overset{\text{def}}{=} \sum_{i=1}^{c} (\tau(i) - \tau'(i))^2. \tag{3}$$

In the following, we will elaborate on solving the optimization problem (1) depending on the particular choice of the loss function. When using the Kendall tau loss, the associated optimal solution is also referred to as the *Kemeny-optimal ranking.* Kendall's tau is an intuitively quite appealing loss function and Kemeny-optimal rankings have several nice properties. Among those, they satisfy the so-called *Condorcet criterion*, which states that if a certain label defeats every other label in pairwise majority voting among the rankings, this label should be ranked first. However, it has been shown in [7] that the problem of computing Kemeny-optimal rankings is NP-hard.

In the case of the Spearman footrule loss, the optimization problem (1) is equivalent to finding a minimum cost maximum matching in a bipartite graph with c nodes [8]. Fagin et al. [4] proposed a computationally efficient approximate aggregation algorithm which for complete rankings simplifies to ordering the labels according to their median ranks, a task which can be accomplished in $\mathcal{O}(kc + c \log c)$ time. In terms of accuracy, Fagin et al. [4] proved that this algorithm computes a constant-factor approximation $\bar{\tau}$ of the optimal solution for both the l_1 and the l_K loss function. More precisely,

$$\sum_{i=1}^{k} l_1(\bar{\tau}, \tau_i) \leq 2\big(\min_{\tau \in \mathcal{S}_c} \sum_{i=1}^{k} l_1(\tau, \tau_i)\big) \quad \text{and} \quad \sum_{i=1}^{k} l_K(\bar{\tau}, \tau_i) \leq 4\big(\min_{\tau \in \mathcal{S}_c} \sum_{i=1}^{k} l_K(\tau, \tau_i)\big).$$

Moreover, Dwork et al. [8] showed that median ordering indeed finds the optimal solution for the l_1 loss in the case of *unique* median ranks. In the case of equal median ranks, we shall apply random tie breaking.

For the sum of squared rank distances loss, a provably optimal solution of (1) is obtained by ordering alternatives according to the so-called *Borda count* [9], a voting technique well-known in social choice theory. The Borda count of an alternative is the number of (weighted) votes for that alternative in pairwise comparisons with all remaining options. This voting rule requires computational time in the order of $\mathcal{O}(kc + c \log c)$ and thus can be evaluated very efficiently.

In the experimental section, the Borda-count and the median ordering techniques will be incorporated into the learning algorithm as they are computationally efficient and have a sound theoretical basis. However, as the aggregation component is an isolated module within our case-based framework, alternative aggregation techniques which may be suitable for the particular application at hand can be integrated easily such as aggregation techniques which minimize loss functions focusing on correct top ranks rather than distributing equal weight to all positions. Moreover, aggregation methods for partial rankings or rankings with ties provide a means to relax the initial assumption that a complete label ranking needs to be associated with every training example.

As an appealing property of this aggregation model, average loss values, $\frac{1}{k}\sum_{i=1}^{k} l(\tau, \tau_i)$, provide a natural means of associating a (reversed) confidence score with a prediction τ, in contrast to previous approaches where techniques for calculating confidence scores have not been proposed yet. Moreover, it is convenient to rescale to the unit interval by

$$1 - \frac{1}{k}\sum_{i=1}^{k} \frac{l(\tau, \tau_i)}{\max_{\hat{\tau}, \hat{\tau}' \in \mathcal{S}_c} l(\hat{\tau}, \hat{\tau}')} \tag{4}$$

such that higher scores correspond to more reliable predictions. In the experimental section, we will provide empirical evidence that this approach indeed yields a meaningful measure of confidence. Complementing the appealing property of an accessible model, case-based ranking supports critical applications where a transparent and reliable prediction process is a mandatory requirement.

4 Empirical Evaluation

The purpose of this section is to provide an empirical evaluation of case-based label ranking in terms of accuracy and computational complexity. The first series of experiments has been set up in order to compare *case-based* label ranking (k-NN-LR) with the *model-based* pairwise label ranking framework [1]. For case-based ranking, we considered two methods for rank aggregation, namely Borda and median aggregation. For pairwise ranking, support vector machines with linear (PW-SVM-LIN) and RBF kernels (PW-SVM-RBF) were used as the binary base learner. The constraint classification approach [2] has not been included in the experimental evaluation as earlier experiments suggested that it typically achieves a level of accuracy comparable to pairwise label ranking while being computationally far more demanding in general [10].

As benchmark datasets of sufficient size are not publicly available for label ranking, we replicated the setting used in [12] to generate ranking datasets. Here,

Table 1. Empirical comparison of case-based label ranking (KNN-LR) using Borda and median aggregation with the model-based pairwise ranking approach, where support vector machines with linear (PW-SVM-LIN) and RBF kernels (PW-SVM-RBF) were used as the binary base learner. The empirical results are grouped in three separate sections, where the Spearman footrole (first section), the Spearman rank correlation (second section) and the Kendall tau (third section) evaluation measures were computed on the test sets.

	Iris	Wine	Glass	Vowel	Vehicle
PW-SVM-LIN	0.767 ±0.148	0.910 ±0.088	0.827 ±0.054	0.484 ±0.040	0.788 ±0.040
PW-SVM-RBF	0.967 ±0.047	0.905 ±0.083	0.842 ±0.058	0.864 ±0.021	0.857 ±0.027
KNN-LR-MEDIAN	0.940 ±0.066	0.927 ±0.045	0.831 ±0.091	0.864 ±0.023	0.795 ±0.039
KNN-LR-BORDA	0.940 ±0.058	0.933 ±0.051	0.831 ±0.091	0.864 ±0.023	0.793 ±0.051
PW-SVM-LIN	0.867 ±0.074	0.941 ±0.055	0.923 ±0.039	0.769 ±0.028	0.892 ±0.027
PW-SVM-RBF	0.980 ±0.023	0.958 ±0.056	0.929 ±0.037	0.962 ±0.009	0.909 ±0.019
KNN-LR-MEDIAN	0.963 ±0.029	0.949 ±0.039	0.898 ±0.085	0.957 ±0.008	0.876 ±0.037
KNN-LR-BORDA	0.967 ±0.031	0.969 ±0.024	0.892 ±0.080	0.957 ±0.008	0.887 ±0.029
PW-SVM-LIN	0.844 ±0.082	0.933 ±0.063	0.891 ±0.044	0.673 ±0.029	0.861 ±0.024
PW-SVM-RBF	0.978 ±0.031	0.944 ±0.053	0.899 ±0.044	0.922 ±0.014	0.896 ±0.017
KNN-LR-MEDIAN	0.960 ±0.044	0.937 ±0.052	0.882 ±0.075	0.922 ±0.013	0.854 ±0.032
KNN-LR-BORDA	0.960 ±0.044	0.952 ±0.039	0.882 ±0.075	0.922 ±0.013	0.853 ±0.038

the idea is to order all the labels in a multiclass dataset with respect to the class probabilities assigned by a naive Bayes classifier (we used the Weka implementation for numerical datasets [13]). The five underlying multiclass datasets can be found at the UCI Repository of machine learning databases [14] and the Statlog collection [15].

For linear kernels the margin-error penalty C was chosen from $\{2^{-2} \dots 2^{10}\}$ and for RBF kernels the considered sets of hyperparameters are given by $C \in \{0.5, 1, 5, 10, 50, 100, 1000\}$ and $\gamma \in \{10^{-3} \dots 10^{3}\}$. In the case of k-NN learning, the number of nearest neighbors k was selected from $\{1, 3 \dots 15, 21\}$. For all parameters, the optimal values were selected using 10-fold crossvalidation on the training sets where the accuracy was estimated with respect to the metric on label rankings used in the specific experimental run. In order to facilitate interpretability, we employed the Spearman footrule, the squared rank distances and the Kendall tau loss functions (see Section 3) in a common normalized version such that the loss (the similarity value) evaluates to -1 for reversed and to $+1$ for identical label rankings. Hence, for the first set of experiments, the overall experimental setup consists of a nested two level crossvalidation procedure, the inner level for selecting hyperparameters and the outer level for estimating generalization accuracy using an additional crossvalidation step.

As anticipated on behalf of the theoretical results, Borda-aggregation slightly outperforms median-aggregation in the case of the Spearman rank correlation, while a substantial difference between Borda- and Median-aggregation cannot be observed for the Spearman footrule loss. For Kendall's tau, both aggregation techniques achieve similar results. Surprisingly, our k-NN-LR is competitive

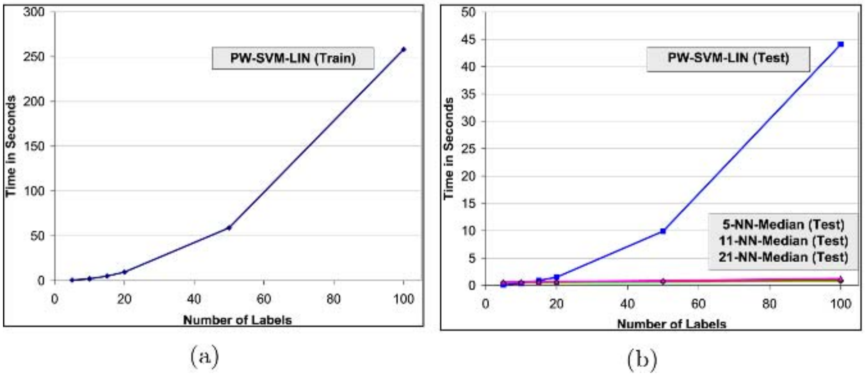


Fig. 1. Computational Time: (a) Training (results for k-NN-LR are omitted as storing the training data requires only negligible effort). (b) Testing.

with PW-SVM-LIN on three datasets (Wine, Glass and Vehicle) and even outperforms it on the remaining two by a large margin (Iris and Vowel). PW-SVM-RBF outperforms our k-NN-LR approach on most datasets, however, typically only by a small margin. Moreover, there are several directions for further improving k-NN-LR, such as weighted aggregation, feature selection and similarity learning which have not been incorporated yet.

We conducted a series of controlled experiments in order to study the computational requirements of the proposed case-based framework using a synthetic experimental setting. This setting is a replication of [1] and operates in the context of expected utility theory. In this setting, the optimal pairwise decision boundaries are hyperplanes, therefore, we selected a linear kernel for PW-SVM with $C = 1000$.[1] Setting both the number of training and test instances to 1000, $k \in \{5, 11, 21\}$ and the input dimension to 10, the training and prediction time of PW-SVM-LIN and k-NN-LR-Median (the difference to k-NN-LR-Borda is negligible) was evaluated for $c \in \{5, 10, 15, 20, 50, 100\}$. The experimental results, depicted in Figure 4, demonstrate that even though we implemented k-NN-LR in a non-optimized straightforward version which does not exploit any sophisticated data structures for supporting efficient nearest neighbor search, it performs very well in terms of computational efficiency. The high computational complexity of pairwise ranking can be attributed to the fact that the number of binary SVMs to be trained is *quadratic* in the number of labels. In contrast to standard classification learning, the training sets for those binary subproblems have the same size as the original dataset, which entails a substantial increase in computational demands. Besides the theoretical difference in testing complexity in this setting ($\mathcal{O}(c^2)$ for pairwise and $\mathcal{O}(kc + c \log c)$ for k-NN ranking), this evaluation underscores the difference in complexity from a practical point of view: k-NN label ranking scales easily to problems with huge numbers of alternatives

[1] Note that this property prohibits a meaningful comparison between pairwise ranking with a *linear* base learner and k-NN label ranking in terms of accuracy.

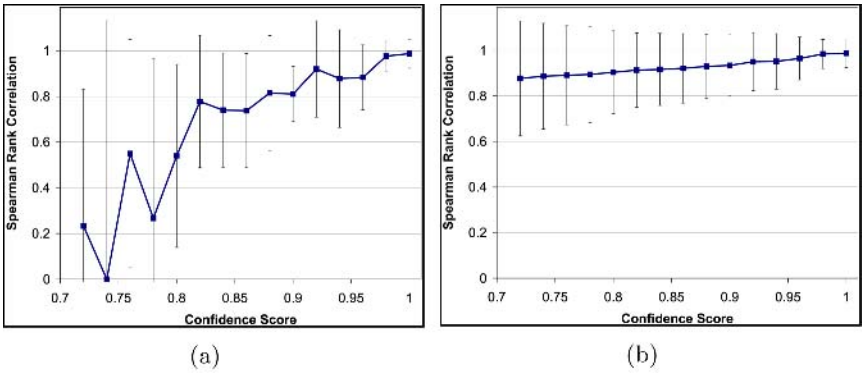


Fig. 2. Accuracy-Confidence Dependence: (a) Average rank correlation for all predictions associated with a particular confidence level. (b) Rank correlation values have been averaged over all predictions associated with *at least* the shown confidence level.

whereas the computational burden involved with pairwise ranking prohibits its application in this regime in many cases.

As mentioned in Section 3, the rescaled accumulated loss (4) can be interpreted as a confidence score for the prediction τ. In order to investigate the dependence of the generalization accuracy on the magnitude of the confidence scores, the k-NN-Borda algorithm ($k = 5$) was evaluated in the above-stated setting: The Vehicle dataset was randomly split into a training and test set of equal size. The predicted label rankings and the associated confidence scores on the test set were used to generate Figure 2, where in (a) the average Spearman rank correlation was averaged over all predictions associated with a *particular* discrete confidence level whereas in (b) the rank correlation was averaged over all predictions with *at least* the specified confidence level. The confidence-accuracy curves clearly indicate that indeed the proposed confidence measure is strongly correlated with the accuracy of predictions. Hence, rejection strategies which refuse to make a prediction if the associated confidence level is below a certain threshold may further increase accuracy, e.g., if we predicted a label ranking for a test instance only if the associated confidence score equals 1.0 (which covers roughly 34% of the entire test set), the Spearman rank correlation would increase from the base level of 0.863 to 0.988 on this subset! Rejecting only the 10% least confident predictions already increases the remaining average rank correlation to 0.913. Similar observations can be made on the other datasets whenever $k > 1$.

5 Concluding Remarks

Despite its conceptual simplicity, case-based learning is one of the most efficient approaches to conventional machine learning problems such as classification and possesses a number of appealing properties. The case-based approach thus lends itself to be applied also in label ranking, a recently introduced more complex type of learning problem. The results in this paper show that case-based label ranking indeed provides a viable alternative to model-based approaches. Beyond the conceptual benefits of flexibility in terms of pairwise loss functions, transparency and confidence computation, our empirical evaluation demonstrates that k-NN label ranking, even in its simplest version, achieves results comparable to state-of-the-art model-based approaches while being amenable to the regime of large-scale problems. Generalizing binary classification techniques to label ranking learning in a model-based methodology suffers substantially from the increased complexity of the target space in ranking (in comparison to classification or regression learning), thus, yielding high computational complexity even for moderately complex problems. This contrasts with the k-NN approach, where the complexity of the target space solely affects the aggregation step and, hence, carries much less weight.

Acknowledgments. This research was supported by the German Research Foundation (DFG) and Siemens Corporate Research (Princeton, USA).

References

1. Fürnkranz, J., Hüllermeier, E.: Pairwise preference learning and ranking. In: Proceedings of ECML 2003, Cavtat, Croatia, Springer-Verlag (2003) 145–156
2. Har-Peled, S., Roth, D., Zimak, D.: Constraint classification: A new approach to multiclass classification and ranking. Proceedings of NIPS 2002. (2002)
3. Dasarathy, B.V.: Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques. IEEE Computer Society Press, Los Alamitos, California (1991).
4. Fagin, R., Kumar, R., Mahdian, M., Sivakumar, D., Vee, E.: Comparing and aggregating rankings with ties. In: Proc. 23rd ACM PODS, 47–58, (2004).
5. Kendall, M.G.: Rank correlation methods. Charles Griffin, London (1955)
6. Lehmann, E.L., D'Abrera, H.J.M.: Nonparametrics: Statistical Methods Based on Ranks, rev. ed. Prentice-Hall, Englewood Cliffs, NJ (1998)
7. Bartholdi, JJ., Tovey, CA., Trick, MA.: Voting schemes for which it can be difficult to tell who won the election. Social Choice and welfare **6**(2) (1989) 157–165
8. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: World Wide Web. (2001) 613–622
9. Hüllermeier, E., Fürnkranz, J.: Ranking by pairwise comparison: A note on risk minimization. In: IEEE International Conference on Fuzzy Systems. (2004)
10. Brinker, K., Fürnkranz, J., Hüllermeier, E.: Label Ranking by Learning Pairwise Preferences. Submitted.
11. Brinker, K., Fürnkranz, J., Hüllermeier, E.: A unified model for multilabel classification and ranking. Proceeding of ECAI 2006. To appear.
12. Brinker, K.: Active learning of label ranking functions. In Greiner, R., Schuurmans, D., eds.: Proceedings of ICML 2004. (2004) 129–136
13. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools with Java implementations. Morgan Kaufmann, San Francisco (2000)
14. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998) Data available at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.
15. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: Machine Learning, Neural and Statistical Classification. Ellis Horwood (1994) Data available at `ftp.ncc.up.pt/pub/statlog/`.

Cascade Evaluation of Clustering Algorithms

Laurent Candillier[1,2], Isabelle Tellier[1], Fabien Torre[1], and Olivier Bousquet[2]

[1] GRAppA - Charles de Gaulle University - Lille 3
`candillier@grappa.univ-lille3.fr`
[2] Pertinence - 32 rue des Jeûneurs -75002 Paris
`olivier.bousquet@pertinence.com`

Abstract. This paper is about the evaluation of the results of clustering algorithms, and the comparison of such algorithms. We propose a new method based on the enrichment of a set of independent labeled datasets by the results of clustering, and the use of a supervised method to evaluate the interest of adding such new information to the datasets.

We thus adapt the *cascade generalization* [1] paradigm in the case where we combine an unsupervised and a supervised learner. We also consider the case where independent supervised learnings are performed on the different groups of data objects created by the clustering [2].

We then conduct experiments using different supervised algorithms to compare various clustering algorithms. And we thus show that our proposed method exhibits a coherent behavior, pointing out, for example, that the algorithms based on the use of *complex* probabilistic models outperform algorithms based on the use of *simpler* models.

1 Introduction

In both supervised and unsupervised learning, the evaluation of the results of a given method, as well as the comparison of various methods, is an important issue. But if cross-validation is a widely accepted method to evaluate supervised learning algorithms, the problem of evaluating unsupervised learning algorithms remains an open issue. The main problem is that the evaluation of clustering results is subjective by nature. Indeed, there are often many different and relevant ways of grouping together some given data objects.

In practice, four main techniques are used to measure the quality of clustering algorithms. But each of these techniques has its own limitations.

1. Use artificial datasets where the desired grouping is known. But the given algorithms are thus evaluated only on the corresponding generated distribution, and results on artificial data can not be generalized to real data.
2. Use labeled datasets and check if the clustering algorithm retrieves the initial classes. But the classes of a supervised problem are not necessarily the classes that have to be found by a clustering algorithm because other groupings can also be meaningful.
3. Work with an expert who evaluates the meaning of the clustering in a particular field. However, if it is possible for an expert to tell if a given clustering

has some meaning, it is much harder to quantify its interest, or to tell if a given result is better than another one. Besides, the relevance of the method can not be generalized to various types of data.

4. Or use some internal criterion, like the intra-cluster inertia and/or the inter-clusters separation. But such pre-defined criteria are also subjective by nature because they use some pre-defined notion of what is a good clustering. For example, inter-clusters separation is not always the best criterion to use : clusters that overlap may sometimes be more relevant.

The main risk in evaluating a clustering method is to consider it as a goal in itself. In fact, what we want to evaluate is how well a given clustering method is able to capture new meaningful and useful information, that is some new knowledge interesting to use for some purpose. We also expect the method to be able to capture such interesting information on various types of problems.

So the main idea of our approach is to consider the clustering as a pre-processing step for another task that we are able to evaluate : supervised learning for instance. Thus the new evaluation method we propose in this paper consists in comparing the results of a supervised algorithm when it is (or not) provided with information coming from a clustering algorithm. If the results of the supervised learning algorithm are improved when some extra-knowledge coming from a clustering process is added, then we conjecture that it means that the clustering process managed to capture some new meaningful and useful information.

This method thus allows us to objectively evaluate the interest of the information captured by a given clustering algorithm. Moreover, the decrease of the error rate of the supervised algorithm when it is helped with the information coming from the clustering algorithm also allows us to quantify this interest. Our evaluation method thus depends on the chosen task, but it allows us to evaluate the contribution of the clustering in the achievement of this objective and real task. Besides, such a bias is less important than when a direct mapping between the clusters and the classes is evaluated.

So our method lies in the framework of classifier combination, in our case the combination of an unsupervised and a supervised method. Many ways of combining classifiers by votes can be found in [3], the two mostly used methods being *bagging* [4] and *boosting* [5]. Some theoretical generalization of these techniques have also been studied, leading to *arcing classifiers* [6], *ensemble methods* [7] and *leveraging methods* [8].

We focus here on techniques that use different learners in a sequential way. In such methods, the output of a learner is an enrichment of the example description, that is then used by the next learner. In that field, *stacked generalization* [9] is a very general framework in which different treatments are stacked : each treatment modifies the example description, and this new dataset is used by the next level. *Cascade generalization* [1] is a special case of stacked generalization. At each level, a classifier is applied on each example $\boldsymbol{x}$ providing probabilities $p(c|\boldsymbol{x})$ that $\boldsymbol{x}$ belongs to class c. These probabilities are then added to the example description and used by the next level classifier. Cascade generalization allows to combine several classifiers but in practice, only two learners are used.

Finally, we also consider the case where we combine an unsupervised and a supervised learner as is done in [2]. In that case, many clusterings are run with different input parameters, leading to different partitions of the set of data objects. For each partition, many independent supervised learnings are executed on the different created groups of data objects and the global error rate is computed. Finally, the partition that leads to the lower error rate is kept.

Based on this principle of sequentially combining an unsupervised and a supervised learner, and then computing the decrease of the error rate of the supervised learner when it is helped by the unsupervised learner, the new evaluation method of clustering algorithms we propose is called *cascade evaluation*. We first describe this new method in section 2. Then section 3 presents some experiments conducted with this new method. Finally, section 4 concludes the paper and suggests topics for future research.

2 Cascade Evaluation Methodology

Being given an initial dataset with classes information, the general steps of our proposed methodology are as follows :

1. learning 1 :
   - perform a supervised learning on the initial dataset;
2. learning 2 :
   - perform a clustering on the dataset without using the classes information;
   - enrich the dataset from the clustering results;
   - and perform a supervised learning on the enriched dataset;
3. compare the results of both learned classifiers.

As we already stated, we consider two different ways of enriching datasets from the results of a given clustering. The first one consists in creating new attributes that represent the information captured by the clustering process, and then adding these new attributes to the initial dataset before running the supervised learning on the enriched dataset. The second way is to consider the new sub-datasets created by the clustering and to run many supervised learnings independently on each sub-dataset.

Concerning the new attributes created from the clustering results in the case of the first combination method, different types of information can be added.

1. As many clustering algorithms provide as output a partition of the initial dataset, we can use the membership of the data objects to the clusters to create new attributes. This information would be represented by a new categorical attribute, each data object being associated with an identifier of the cluster it belongs to.
2. We can also associate to each data object a set of attributes that represent the center of the cluster it belongs to. We would thus double the number of attributes in the dataset.

3. Recently, many *subspace clustering* algorithms [10] emerged that are able to associate to each dimension of each cluster a weight specifying its relevance in determining the membership of the data objects to the cluster. So in such cases, we could add to each data object one new continuous attribute per initial dimension corresponding to the weight, on that dimension, of the cluster it belongs to. Such new attribute would thus allow to differentiate data objects for which a given dimension is relevant from those for which it is not relevant (according to the subspace clustering results).

Besides, as most clustering algorithms need some parameters to tune, we can run these algorithms many times with different input parameter values and enrich the dataset for each clustering results. For example, many clustering methods need as input the number of clusters to be found. In such cases, we could run them many times, varying this parameter from 2 to 10 for example. The supervised algorithm used afterwards would then be able to choose which attribute(s) to use among them.

In the case of the second combination method proposed, we first generate many partitions with different input parameters. We then compute the cross-validation error of independent supervised learners executed on the different groups of data objects created by the clustering. And finally, we select the partition that led to the lowest error rate.

To evaluate the improvement in the results of the supervised learning algorithm with or without the new information coming from the clustering process, we test both methods on various independent datasets. On each dataset, we perform five 2-fold cross-validations, as proposed in [11]. For each 2-fold cross-validation, we compute the balanced error rates of both methods. And we then use four measures to compare them :

– *nb wins*: the number of wins of each method;
– *sign wins*: the number of significant wins, using the *5×2cv F-test* [12] to check if the results are significantly different;
– *wilcoxon*: the wilcoxon signed rank test, that indicates if a method is significantly better than another one on a set of independent problems (if its value is higher than 1.96);
– and *av perf*: the mean balanced error rate.

3 Experiments

We present in this section the results of the comparisons of various clustering algorithms :

– Rand, an algorithm that generates random partitions, being given the number of expected clusters (used as a reference);
– K-means, the well-known full-space clustering algorithm based on the evolution of K centroids that represent the K clusters to be found;

– LAC [13], a subspace clustering algorithm based on K-means that associates with each centroid a vector of weights on each dimension, inversely proportional to the dispersion of the members of the clusters on the dimension;
– SSC [14], that is based on the use of a probabilistic model and the EM algorithm [15] under the assumption that the data follow independent gaussian distributions on each dimension;
– and SuSE [16], an adaptation of SSC that performs *hard feature selection* during the learning process, by selecting for each cluster a subset of the dimensions on which the standard deviation is minimized.

So the algorithms compared here use different models with different complexity levels. K-means uses only one centroid to represent a cluster. LAC adds to each centroid a vector of weights on each dimension. SSC defines a membership probability of each data object to each cluster, in addition to use a gaussian model. And SuSE also considers a subset of relevant dimensions associated to each cluster. All these algorithms need as input parameter the number K of clusters to be found. So as we discussed earlier, we will run them many times with K varying from 2 to 10.

In order to check if the results depend on the supervised algorithm used, we conduct these experiments with various supervised learning algorithms :

– C4.5 [17], the well-known supervised method based on the iterative construction of a decision tree;
– C5 [18] boosted 10 times, that uses the boosting of decision trees, in order to observe if the information added by clustering algorithms also help supervised methods that already combine many classifiers;
– DLG [19], a supervised method that uses *least general generalizations* instead of decision trees, so that many attributes are considered at a time to construct decision surfaces;
– and multi-class Support Vector Machines (SVM) [20], that construct large margin classifiers, in order to check if the information added by clustering algorithms also help supervised methods that use linear combinations of the initial features.

Finally, the datasets used are those of the UCI Machine Learning Repository [21] that contain only numerical attributes.

Table 1 presents the balanced error rates of C4.5 run on the initial dataset, and then run on datasets enriched by the results of the corresponding clustering algorithms. Each measure corresponds to an average over five 2-fold cross-validations. At each time, all the methods are run on the same training set and evaluated on the same test set.

From this table, we can observe that most of the time, the results of C4.5 are improved when some information coming from *real* clustering algorithms are added, whereas adding information from a random clustering degrades the results. Besides, we can note that the results of SSC and SuSE are often better than those of K-means and LAC. Then table 2 presents a summary of the comparison between C4.5 and C4.5 enriched by the clustering algorithms.

Table 1. Balanced error rates (in %) of C4.5 enriched by clustering algorithms. The bold values correspond to the minimum error rates obtained on each dataset.

	C4.5 alone	C4.5 + Rand	C4.5 + K-means	C4.5 + LAC	C4.5 + SSC	C4.5 + SuSE
ecoli	48.5	48.3	42.8	**40.3**	42	43.1
glass	**32.6**	40.8	35.7	37	40.4	34.9
image	4.8	6	4.8	**4.6**	**4.6**	**4.6**
iono	14.1	15.8	14.2	13.1	**9.8**	11.2
iris	7.3	7.9	6.7	**3.7**	5.1	4.7
pima	31	35	32.1	32.1	30.8	**30**
sonar	31	35.2	30	28.8	28.8	**27.2**
vowel	29.5	38.5	25	26.4	24.1	**22.2**
wdbc	5.9	6.8	4.6	3.9	5.1	**3.1**
wine	8.7	8.8	10.4	9.6	**2.7**	3.6

Table 2. Comparison of C4.5 alone with C4.5 enriched by clustering algorithms

	C4.5 alone	C4.5 + Rand	C4.5 + K-means	C4.5 + LAC	C4.5 + SSC	C4.5 + SuSE
nb wins	-	1/9	5/4	7/3	**9/1**	**9/1**
sign wins	-	0/1	0/0	1/0	2/0	**3/0**
wilcoxon	-	-2.67	-0.05	1.31	1.83	**2.56**
av perf	21.3	24.3	20.6	20	19.3	**18.5**

SuSE is the only clustering algorithm that significantly helps C4.5 improve its results, according to the wilcoxon signed rank test. It is significantly better on 3 datasets according to the *5×2cv F-test*. But as SuSE, SSC improves the results of C4.5 nine times over ten, contrary to K-means and LAC. All algorithms improve the results of C4.5 on average, except the random clustering. And when C4.5 is combined with clustering algorithms based on more complex models, then the error rate is lower and the improvements are more significant than when it is combined with clustering algorithms based on simpler models.

Such experiments were also conducted using different supervised algorithms, namely C5 boosted, DLG and SVM, and using the second method for combining unsupervised and supervised algorithms. It is then very interesting to note that, in spite of the use of different supervised and combination methods, the clustering algorithms that best help supervised learners to minimize the cross-validation error rate on the different datasets remain mostly the same. In particular, SSC and SuSE still outperform K-means and LAC in many cases. Moreover, the order in which the clustering methods are ranked remains the same no matter which supervised and combination methods are used.

Finally, as a comparison, we computed the *F-measure* and the *Entropy* between the clusters obtained by the various clustering methods and the initial classes of the various problems in order to measure the mapping between them. We thus first observed that the two measures do not agree on which clustering method leads to the best mapping between the clusters and the classes on each dataset. Then we noted that there is no direct relation between the methods that optimize these values and the methods that better help the supervised learners to improve their results. Besides, such measures do not provide objective information about the interest of the clustering methods, contrary to our proposed evaluation method that shows if the results are significantly better with the help of the given clustering methods.

4 Conclusion

We have presented in this paper a new objective and quantitative evaluation method of clustering algorithms that consists in comparing the results of a supervised algorithm when it is (or not) provided with information coming from a clustering algorithm. We have considered different supervised algorithms to be used in our evaluation method. We have also considered two different ways of combining unsupervised and supervised learning algorithms.

The experiments pointed out that the order in which the clustering methods are ranked remains the same no matter which supervised algorithm and which combination method are used. So it shows the robustness of our proposed evaluation method. The experiments also pointed out that clustering methods based on the use of more complex models outperform methods based on the use of simpler models. This result is not surprising, but rather exhibits coherent results of our new evaluation method.

Although it was not the aim of our investigations, we have also shown that the results of supervised learning algorithms are improved when they use some extra-knowledge coming from non random clustering algorithms. We conjecture supervised learners can benefit from the information added by clustering methods because these new information are of very different nature. In particular, clustering algorithms can help supervised learners to specialize their treatments according to different specific areas in the input space. They can also help supervised learners fit more complex decision surfaces. It thus seems interesting to continue our investigations in the more general framework of classifier combination when one learner is unsupervised.

Our experiments seem to show that using the clustering to partition the object space, and then executing independent supervised learnings on each created group of data objects gives better results than enriching the datasets with new attributes and then executing a supervised learning on the enriched dataset, since the improvements are more important when the second method for combining unsupervised and supervised algorithms is used. But this may be a consequence of the method we have used to create new attributes, that significantly increases the size of the dataset. It would thus be interesting to examine this point in detail in future research.

Finally, in future works, it would also be interesting to find other tasks as objective as supervised learning, and for which clustering would be an interesting pre-processing, in order to conduct other experiments with our proposed evaluation method in such another framework. One possible way would be for example to compute the reduction in the execution time of various requests on OLAP databases that use (or not) a clustering algorithm to create their index.

References

1. Gama, J., Brazdil, P.: Cascade generalization. Machine Learning **41** (2000) 315–343
2. Apte, C.V., Natarajan, R., Pednault, E.P.D., Tipu, F.A.: A probabilistic estimaton framework for predictive model analytics. IBM Systems Journal **41** (2002)
3. Ali, K.M., Pazzani, M.J.: Error reduction through learning multiple descriptions. Machine Learning **24** (1996) 173–202
4. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
5. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Int. Conf. on Machine Learning. (1996) 148–156
6. Breiman, L.: Bias, variance, and arcing classifiers (1996) Technical Report 460, Statistics Department, University of California.
7. Dietterich, T.G.: Ensemble methods in machine learning. In Kittler, J., Roli, F., eds.: 1st Int. Workshop on Multiple Classifier Systems. Volume 1857 of LNCS., Springer-Verlag (2000) 1–15
8. Meir, R., Rätsch, G.: An introduction to boosting and leveraging. In Mendelson, S., Smola, A., eds.: Advanced Lectures on Machine Learning. Number 2600 in LNAI. Springer-Verlag (2003) 119–184
9. Wolpert, D.H.: Stacked generalization. Neural Networks **5** (1992) 241–259
10. Parsons, L., Haque, E., Liu, H.: Evaluating subspace clustering algorithms. In: Workshop on Clustering High Dimensional Data and its Applications, SIAM Int. Conf. on Data Mining. (2004) 48–56
11. Dietterich, T.G.: Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation **10** (1998) 1895–1923
12. Alpaydin, E.: Combined 5x2cv F-test for comparing supervised classification learning algorithms. Neural Computation **11** (1999) 1885–1892
13. Domeniconi, C., Papadopoulos, D., Gunopolos, D., Ma, S.: Subspace clustering of high dimensional data. In: SIAM Int. Conf. on Data Mining. (2004)
14. Candillier, L., Tellier, I., Torre, F., Bousquet, O.: SSC : Statistical Subspace Clustering. In Perner, P., ed.: Machine Learning and Data Mining in Pattern Recognition (MLDM). LNCS, Leipzig, Germany, Springer Verlag (2005) 100–109
15. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B **39** (1977) 1–38
16. Candillier, L., Tellier, I., Torre, F., Bousquet, O.: SuSE : Subspace Selection embedded in an EM algorithm. In Miclet, L., ed.: Actes de la huitième Conférence d'Apprentissage (CAp). (2006)
17. Quinlan, J.R.: C4.5: Programs for Machine Learning. KAUFM (1993)
18. Quinlan, R.: Data mining tools see5 and c5.0 (2004)
19. Webb, G.I., Agar, J.W.M.: Inducing diagnostic rules for glomerular disease with the DLG machine learning algorithm. Artificial Intelligence in Medicine **4** (1992) 419–430
20. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research (JMLR) **6** (2005) 1453–1484
21. Blake, C., Merz, C.: UCI repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/MLRepository.html] (1998)

Making Good Probability Estimates for Regression

Michael Carney and Pádraig Cunningham

Trinity College Dublin, Dublin 2, Ireland
firstname.surname@cs.tcd.ie

Abstract. In this paper, we show that the optimisation of density forecasting models for regression in machine learning can be formulated as a multi-objective problem. We describe the two objectives of *sharpness* and *calibration* and suggest suitable scoring metrics for both. We use the popular negative log-likelihood as a measure of sharpness and the probability integral transform as a measure of calibration.To optimise density forecasting models under multiple criteria we introduce a multi-objective evolutionary optimisation framework that can produce better density forecasts from a prediction user's perspective. Our experiments show improvements over the state-of-the-art on a risk management problem.

1 Introduction

Regression is a supervised learning problem where the fundamental task is to predict some continuous variable. Density forecasting is an important subfield of regression that attempts to tackle the practical problem of uncertainty in predictions of a regression model. To achieve this, a density forecast estimates a complete probability density for the target variable. This is useful primarily because prediction users are generally sensitive to the possible variance around a prediction.

Typically, density forecasting models use a generalisation of maximum likelihood called negative log-likelihood (NLL). This is due to the convenience of being able to use traditional non-linear optimisation techniques such as *conjugate gradient* or *quasi Newton* with minimal adaptations. However, optimising on NLL alone will often result in poor and sometimes misleading density forecasting models [1]. Research suggests that these problems can be identified post training by evaluating the empirical validity of probability estimates [2,3,4], a property of these models that is commonly called *calibration*. We argue that a better way to solve this problem is to directly optimise calibration during training and outline a general framework that will allow this to be achieved for most existing density forecasting techniques.

In this paper, we propose scoring functions for maximising *sharpness* and *calibration* and develop a general framework for optimisation of these two objectives. Sharpness refers to the variance of the prediction around the observation and calibration refers to the empirical validity of the probability estimates (see Section 2). In Section 3 we outline our broad framework based on a multi-objective evolutionary algorithm and describe two different example implementations. In the first, we adapt an Evolutionary Strategy [5] to a multi-objective search method [6] to calibrate a GARCH type model [7]. In the second example we combine a Mixture Density Network [8] with Evolutionary Artificial Neural Networks [9] and Pareto Neural Networks [10] to create a

Pareto Mixture Density Network [11]. Section 4 compares results achieved on foreign exchange data between the state-of-the-art and proposed methods. Finally, Section 5 briefly concludes the paper.

2 Goals of Density Forecasting

We address the regression problem of estimating the parameters for a model given a set of training data $\{(\mathbf{x}_i, t_i)\}_{i=1}^m$, where the ith example is described by the pattern $\mathbf{x}_i \in \Re^n$ and the associated response $t_i \in \Re$. Point forecasting attempts to estimate, $\langle t_i|\mathbf{x}_i\rangle$, the conditional mean of the target variable given an input pattern. Density forecasting models attempt to estimate, $p(t_i|\mathbf{x}_i)$, the conditional probability density that the target is drawn from, a considerably more complex task[1]. The NLL addresses the goal of minimising variance around the target by rewarding models based on the density of the prediction at the target (sharpness).

$$NLL_i = -\log(p(t_i|\mathbf{x}_i)) \tag{1}$$

Calibration, the second goal, refers to the property that if a predicted density function suggests P percent probability of occurrence, the event truly ought to have probability P of occurring [12]. This is a joint property of the target and the predictions. Unfortunately, the assessment of calibration is less straightforward than sharpness.

Diebold et al. [3] show that to adequately assess calibration in a regression scenario it is necessary to make the assumption that you are attempting to find the model that correctly describes the data generating process. It is fair to make this assumption because the correct model *weakly dominates* all other models. In the case where the correct data generating process is described, the set of cumulative densities at the observations will be uniform. Therefore, to determine calibration you must carry out the following,

$$z_i = \int_{-\infty}^{t_i} p(u|\mathbf{x}_i)du \tag{2}$$

where z_i is the cumulative of the predicted density at the target t_i. This is known as the Probability Integral Transform (PIT)[13]. For a data set of length m, this z series should be $\{z_i\}_{i=1}^m \overset{iid}{\sim} U[0, 1]$.

We know $z_i \in [0, 1]$ because it is a value from a cumulative density. Therefore, a test for calibration relates directly to a test for whether the z series is $U[0, 1]$. A useful method for discerning the calibration of a model is to plot a PIT histogram of the z series. For example, in [2,3] PIT histograms are used to assess calibration of a model post training. However, this requires visual assessment, it would be more desirable to have a means of *ranking* a set of models in terms of their uniformity. Fortunately, this is a common problem and relates to testing the goodness-of-fit of a sample of data to a specific distribution. Noceti et al. [14] compared a number of goodness-of-fit tests and concluded that the Anderson-Darling (A^2) [15] test was the most robust among the

[1] Prediction interval estimation is a popular approach to predicting uncertainty. This is a subclass of density forecasting where only a particular interval of predicted density is reported.

most common tests for uniformity. The A^2 test is negatively oriented returning 0 in the case where a model is perfectly calibrated to the data. The formula for A^2 is:

$$A^2 = -m - \frac{1}{m}\sum_{j=1}^{m}(2j-1)[\log(z_i) + \log(1 - z_{m-j})] \tag{3}$$

Where, m, is the number of z values, and the z values are sorted in ascending order. We can now rank the calibration of a set of models based on their A^2 score on a test set.

3 Multi-objective Optimisation Framework

In the preceding sections we presented quality scores for both calibration and sharpness (A^2 and NLL). Implicitly, we have described a multi-objective optimisation problem. There are a number of ways to solve a multi-objective search problem, however, the preferable approach is to use an *a posteriori* multi-objective evolutionary algorithm (MOEA). In the context of MOEA's, *a posteriori* means that the optimisation process maintains an archive of optimal trade-off (non-dominated) solutions known as the Pareto front [16] throughout training and the user selects the model that best optimises their goals from the resulting Pareto front of solutions [6].

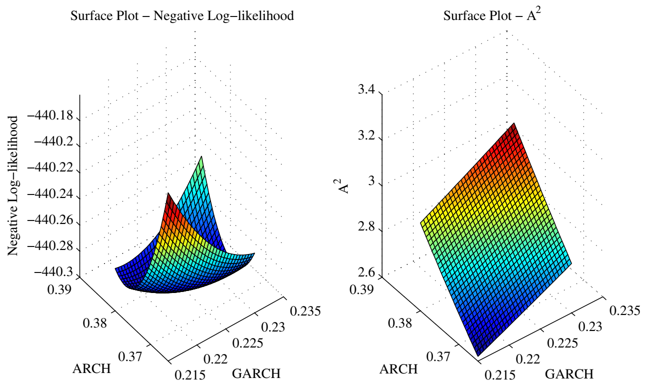


Fig. 1. Comparison of the NLL and A^2 error surfaces in the region of the NLL minimum. This is an error function for a GARCH(1,1) model trained on data from 1,000 observations simulated from an EGARCH(1,1) model that assumes a *Student-t* distribution [17]

Multi-objective search requires objectives to be conflicting i.e. objectives do not converge to the same global minimum in parameter space. To demonstrate that calibration and sharpness are conflicting we analyse their relationship in terms of the parameter space of a model. To do this we have constructed an experiment using a very simple density forecasting model from econometrics called GARCH [7]. The GARCH model has two parameters of importance commonly called the ARCH and GARCH terms that relate to weights applied to the residual and variance for the preceding time-step (see Section 3.1). In this experiment we use a synthetic data set so that we can specify the other parameters of the GARCH model correctly *a priori*. Since we have restricted the model to only two free parameters, an error function will be a surface above a 2-dimensional parameter space. Figure 1 shows plots of the error function surfaces in terms of the two parameters (ARCH and GARCH) of the model around the NLL global minimum. It is clear from the surface plots that they are completely different functions and the minima of the two error functions are located in different regions of the parameter space. This clearly shows that NLL and A^2 are conflicting objectives and multi-objective search is possible.

The standard MOEA algorithm framework such as described in [11,6] provides the basis to our technique. This provides the flexibility, so our method can be applied to almost any density forecasting model that can be represented as a set of parameters. To implement our method the modeller must;

1. Determine a vector representation for the parameters of the density forecasting model.
2. Be able to calculate the A^2 and NLL score for the model's predicted densities.
3. Decide on a mutation and selection strategy for the evolutionary algorithm.

In the following subsections we will briefly describe the implementation of this algorithm for two particular density forecasting models.

3.1 Pareto GARCH

Generalised Autoregressive Conditional Heteroscedasticity (GARCH) models are commonly used in finance to estimate the conditional variances of a time series [7]. Although there have been implementations of GARCH that were optimised using evolutionary algorithms e.g. [18], this is the first time, to our knowledge, that this type of model has been optimised on multiple objectives. In our experiments we use the simplest and most popular model GARCH(1,1), however, this optimisation approach can be applied to any GARCH type model, of which there are many. Our GARCH model predicts a Gaussian, the mean is presumed to be constant and the conditional variance for the next time step is predicted as a weighted sum of the previous time-steps residual, its predicted variance, and the unconditional variance of the series. This very simple model can successfully capture the serial dependence in financial data. GARCH(1,1) can be represented as a vector of 4 parameters. This vector representation is used to encode an individual in our evolutionary algorithm. We use an Evolutionary Strategy (ES) for optimisation because it has a number of advantageous characteristics [5]. Besides being able to optimise a non-differentiable objective function (e.g. A^2 score), ES is attractive because it can solve complex, high dimensional, multi-modal, real valued problems. However, most other evolutionary algorithms could be used instead. For a full description of the Pareto-GARCH model see [19].

3.2 Pareto Mixture Density Network

Details on the Pareto Mixture Density Network are given in [11,1].

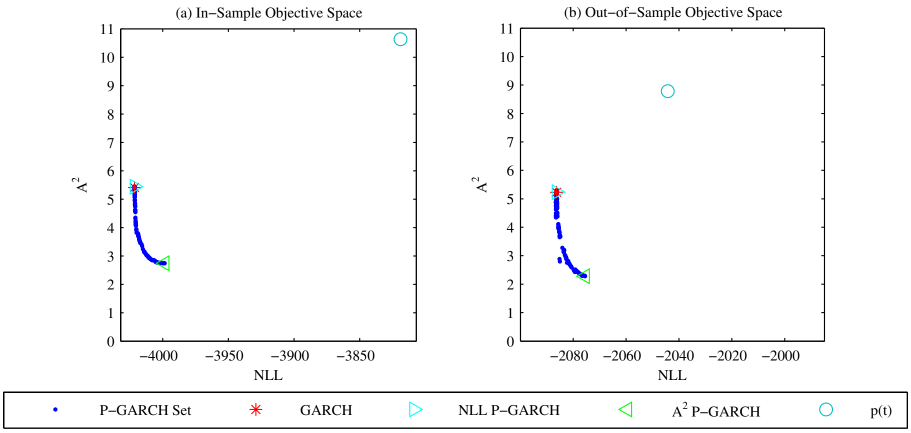


Fig. 2. In-sample and out-of-sample objective spaces after 10,000 iterations of Pareto-GARCH

4 Case Study: Financial Data

In this case study, we analyse the performance of both an MDN and GARCH model on the notoriously difficult domain of foreign exchange data. The data is comprised of 1,501 examples of daily price observations for the DeutscheMark/British Pound foreign exchange rate, from April 1985 to January 1992[2], a particularly volatile period for these two currencies. We transformed the daily prices into a log returns series by $r_i = \log(\frac{p_{i+1}}{p_i})$, where p_i is the price at interval i. The return series was separated into a training set of the first 1,000 observations and a test set comprised of the last 500 observations.

Using a non-linear optimisation technique (*quasi-Newton*) we train a GARCH(1,1) model minimising the NLL. There is no standard implementation of the GARCH optimisation algorithm, however, the error function, NLL, is the same in all cases. Therefore, there is usually negligible difference between the models that are produced by different implementations. In these experiments we use the Matlab GARCH model as our benchmark - it is denoted as *GARCH* in the figures and tables that follow. This model is used as the initial population seed for our ES. We can presume that this model represents a near global optimum solution in terms of the NLL objective and will be present in the Pareto front. The aim of the next step in training, the multi-objective search, is to start from this point on the Pareto front and find as diverse a Pareto front as possible. This process should provide new solutions that improve on calibration.

We carried out 1,000 iterations of the ES algorithm resulting in a set of 316 non-dominated individuals. Figure 2 shows the in-sample and out-of-sample objective spaces for each of the models in the Pareto set. Each point on the objective space represents a model. We have highlighted some models of interest, the *GARCH* model represents the initial solution trained using the standard (Matlab) optimisation procedure. NLL *P-GARCH* is the model that has the best NLL score and A^2 *P-GARCH* is the model

[2] This data is included with the *Mathworks*^{TM} *Matlab*^{TM} *Garch Toolbox.*

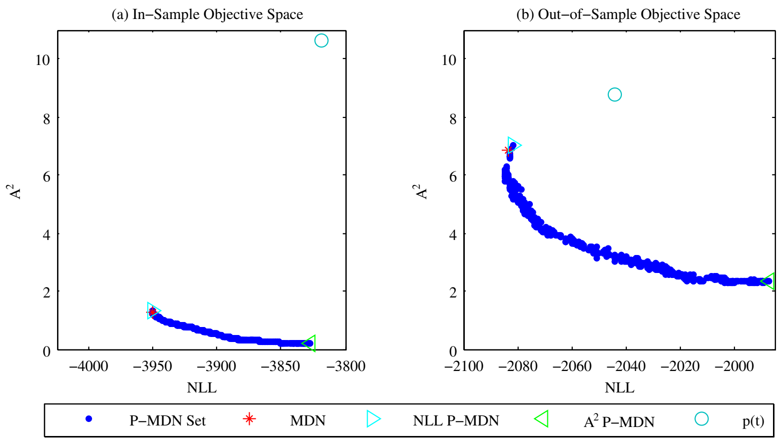


Fig. 3. In-sample and out-of-sample objective spaces after 10,000 iterations of Pareto-MDN

with best A^2 score from the Pareto front. Finally, $p(t)$ represents a model that assumes the same Gaussian distribution at each time-step, the approximated unconditional distribution.

For comparison, we also trained an MDN model on the same data. The MDN was given 6 hidden units and outputs were represented as a 2 component GMM[3]. Again, an initial model was trained on the data using a standard optimisation technique, in this case we use a quais-Newton method. We use this model as our initial starting position in parameter space for our Pareto-MDN evolutionary algorithm. The resulting objective spaces, on both training and test data, after 10,000 epochs of training are shown in Figure 3. The Pareto front has 736 individuals.

We have determined the Spearman rank correlation coefficients between the in-sample and out-of-sample models on each objective function. Both for the GARCH Pareto set and for the MDN Pareto set the model rankings are strongly correlated. This suggests little over-fitting of the models to the data on either objective function. The Spearman rank correlation of NLL scores of all *P-GARCH* models in and out-of sample was 0.99 and in terms of A^2 was 0.99. Similiarly, for *P-MDN* models NLL correlation was 0.86 and the A^2 correlation was 0.99.

As in the GARCH objective space, the MDN models with best NLL and A^2 scores and $p(t)$ are highlighted. The objective space figures suggest that the MDN produces a far better calibrated set of solutions on the in-sample data, however, this advantage is lost in the out-of-sample data. Analysis of the dominance of models shows that in-sample the MDN and GARCH models do not dominate each other. However, out-of-sample there are 70 dominant models, out of the possible 1,052, and 69 of these models are GARCH models. No model is dominated by the unconditional distribution, $p(t)$. Figure 4 shows the out-of-sample PIT histograms of the selected models. The figure clearly shows how the A^2 models are better calibrated than the models optimised on NLL.

[3] These values were chosen after 10 fold cross-validation over several different candidate architectures.

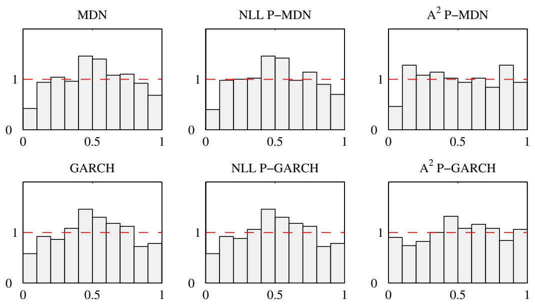


Fig. 4. PIT histograms for selected models on the test data

Table 1. Percentile exceedance ratio. This measure counts the number of days that returns exceed the predicted loss of a model at a specific percentile, normalised over the total number of days. The closer these values are to 10% the better the model. Best results highlighted in bold.

	In-Sample	**Out-of-Sample**
$GARCH$	**9.70%**	5.80%
$NLL\ P-GARCH$	**9.70%**	5.80%
$A^2\ P-GARCH$	11.60%	**9.00%**
MDN	11.10%	3.80%
$NLL\ P-GARCH$	11.10%	3.60%
$A^2\ P-GARCH$	**9.70%**	4.60%

The implications of this result can be illustrated by applying the models from above to a simple financial application. Value-at-Risk (VaR) is a measure of the maximum potential change in value of a portfolio with a given probability over a pre-set horizon. An estimate of the 10% VaR would represent the estimated minimum amount of money that you could loose with 10% probability. Using the models and data from above we demonstrate how our optimisation method improves performance on this metric by showing the percentile exceedance ratios for all models at the 10^{th} percentile in table 1. The A^2 models have the best out-of-sample VaR estimates.

In summary, this strategy allows the user to identify models that score well on both sharpness and calibration. There are many domains such as finance or weather forecasting where calibration is as important as sharpness. If rare events are significant then it is important that the model assigns the correct probability to them. Our approach allows the prediction users to select the model that gives them the best trade-off solution from the Pareto set of solutions.

5 Conclusions

In this paper we have outlined the two goals of density forecasting. Taking only one of these goals into consideration during training of a density forecasting model can result in poor performance. To solve this problem we introduce a new technique for density forecasting optimisation that uses a multi-objective search algorithm to find the best solution. Our framework can be applied to most likelihood based density forecasting models. An attractive advantage of this approach is that the underlying model is not augmented in any way so the model can be interpreted in the normal manner. Our experiments have shown that this optimisation approach can find models that are better calibrated than those found through negative log-likelihood.

References

1. Carney, M., Cunningham, P.: Calibrating probability density forecasts with multi-objective search. Technical Report TCD-CS-2006-07, Trinity College, Dublin (2006)
2. Weigend, A.S., Shi., S.: Predicting daily probability distributions of S&P500 returns. Journal of Forecasting **19**(4) (2000) 375–392
3. Diebold, F.X., Gunther, T.A., Tay, A.S.: Evaluating density forecasts with applications to financial risk management. International Economic Review **39**(4) (1998) 863–83
4. Gneiting, T., Raftery, A.E., Balabdaoui, F., Westveld, A.: Verifying probabilistic forecasts: Calibration and sharpness. In: Proceedings Workshop on Ensemble Forecasting. (2003)
5. Rechenberg, I.: Evolutionsstrategie ’94. Technical report, frommannholzboog (1994)
6. Deb, K.: Multi-Objective Optimisation using Evolutionary Algorithms. Wiley (2001)
7. Bollerslev, T.: Generalized autoregressive conditional hetroskedasticity. Journal of Econometrics **31** (1986) 307–327
8. Bishop, C.: Mixture density networks. Technical report, Neural Computing Research Group, Aston University, Birmingham (1994)
9. Yao, X.: Evolutionary artificial neural networks. In: Proceedings of the IEEE. Volume 87. (1999) 1423–1447
10. Fieldsend, J., Singh, S.: Pareto evolutionary neural networks. IEEE Trans. Neural Networks **16**(2) (2005) 338–354
11. Carney, M., Cunningham, P.: Calibrating probability density forecasts with multi-objective search. In: Proceedings of European Conference in Artificial Intelligence. (2006)
12. Dawid, A.P.: The well calibrated Bayesian. Jour. of the Amer. Stats. Ass. **77** (1982) 605–610
13. Rosenblatt: Remarks on a multivariate transformation. Annals of Math. and Stats. **23** (1952) 470–472
14. Noceti, P., Smith, J., Hodges, S.: An evaluation of tests of distributional forecasts. Journal of Forecasting **22**(6-7) (2003) 447–455
15. Anderson, T., Darling, D.: A test of goodness of fit. Jour. of the Amer. Stats. Ass. **19** (1954) 765–769
16. Pareto, V.: Cours D’Economie Politique. Lausanne (1896)
17. Nelson, D.B.: Conditional heteroskedasticity in asset returns: A new approach. Econometrica **59** (1991) 347–370
18. Jerrell, M.E., Campione, W.A.: Global optimization of econometric functions. Journal of Global Optimization **20** (2001) 273–295
19. Carney, M., Cunningham, P., Lucey, B.M.: Making density forecasting models statistically consistent. Technical Report TCD-CS-2006-04, Dept of Comp Sci, TCD (2006)

Fast Spectral Clustering of Data Using Sequential Matrix Compression

Bo Chen[1,*], Bin Gao[2,*], Tie-Yan Liu[3], Yu-Fu Chen[1], and Wei-Ying Ma[3]

[1] Department of mathematics, Graduate School of Chinese Academic of Science
Beijing, 100080, P.R. China
chbo04@mails.gucas.ac.cn, yfchen@gucas.ac.cn
[2] LMAM, School of Mathematical Sciences, Peking University
Beijing, 100871, P.R. China
gaobin@math.pku.edu.cn
[3] Microsoft Research Asia, Sigma Center, No. 49, Zhichun Road,
Beijing, 100080, P.R. China
{tyliu, wyma}@microsoft.com

Abstract. Spectral clustering has attracted much research interest in recent years since it can yield impressively good clustering results. Traditional spectral clustering algorithms first solve an eigenvalue decomposition problem to get the low-dimensional embedding of the data points, and then apply some heuristic methods such as *k*-means to get the desired clusters. However, eigenvalue decomposition is very time-consuming, making the overall complexity of spectral clustering very high, and thus preventing spectral clustering from being widely applied in large-scale problems. To tackle this problem, different from traditional algorithms, we propose a very fast and scalable spectral clustering algorithm called the sequential matrix compression (SMC) method. In this algorithm, we scale down the computational complexity of spectral clustering by sequentially reducing the dimension of the Laplacian matrix in the iteration steps with very little loss of accuracy. Experiments showed the feasibility and efficiency of the proposed algorithm.

1 Introduction

Spectral clustering [3][6]is one of the most promising methods among existing clustering algorithms. Although a lot of previous work demonstrated the effectiveness of spectral clustering, its applications are mainly restricted to small-scale problems. This is due to the high computational complexity of spectral clustering. In the traditional implementation of spectral clustering, eigenvalue decomposition (EVD) is first conducted and then some additional heuristics such as *k*-means are applied to the eigenvectors to obtain the discrete cluster labels. It is known that the time and space complexities of state-of-the-art EVD solvers (such as Lanczos method [7] and pre-

[*] This work was performed when the first two authors were interns at Microsoft Research Asia.

conditioned conjugate gradient (CG-based) algorithm [4][8]) are $O(mn^2k)$ and $O(n^2k)$ [2], where k is the number of the eigenvectors used, n is the number of data points, and m is the number of iteration steps. It is clear such complexities are too high when the number of data points is large.

In order to extend spectral clustering to large-scale applications, we investigate how to reduce the complexity of spectral clustering in this paper. Our work is based on the following observation. When we compute the eigenvector associated with the second smallest eigenvalue in the EVD process of spectral clustering, we find that some specific elements in this eigenvector (also called the embedding vector or Fiedler vector) get stable very fast after only several steps of iteration and their values will not change by much in the future iteration steps. This observation indicates that it will not cause much loss if one stops the iteration process early for such stable points and fixes their values directly. It is easy to understand that this kind of early stop can reduce the problem scale of spectral clustering and save many computations.

However, one may argue that the embedding values of other data points will be affected if we manually fix the stable points because the optimizations of data points are not independent of each other. To tackle this problem, that is, to save the computations for the stable points but not to affect other points, we propose a matrix compression technology. Take two-way spectral clustering for example. After determining which stable points should be stopped early, we fix their values and merge those fixed data points with positive (negative) embedding values to a single aggregated positive (negative) point. Thus the scale of the Laplacian matrix is reduced. Then we properly adjust the values of the elements in this reduced Laplacian matrix, so that the embedding values of those unfixed points calculated from this reduced Laplacian matrix can be almost the same as their values calculated from the original Laplacian matrix. In this way, we may not only reduce the complexity of spectral clustering, but also successfully minimize the loss caused by the dimensionality reduction.

Actually the above idea can be once again applied to the reduced Laplacian matrix so that the problem scale can be further reduced. By conducting this process recursively, we can eventually get a very efficient implementation of spectral clustering. We name this new technique by sequential matrix compression (SMC). We proved in this paper that the SMC method can preserve enough information to guarantee the accuracy of the solutions. Experimental evaluations on real-world clustering problems showed the effectiveness and efficiency of the proposed SMC method.

2 Sequential Matrix Compression

In this section, we take the two-way ratio-cut spectral clustering for example to describe a fast implementation of spectral clustering based on sequential matrix compression. The same idea can also be extended to the normalized cut.

Suppose $L = \{l_{ij}\}$ is an $n \times n$ Laplacian matrix generated from a certain dataset and ξ is the underlying n-dimensional embedding vector for clustering. That is, $L=D-W$, where W is the adjacency matrix and D is a diagonal matrix with the sum of each row of W assigned to its corresponding diagonal positions. Then the two-way ratio-cut spectral clustering can be formulated as follows.

$$\min \xi^T L \xi \quad \textit{subject to} \quad \xi^T \xi = 1, \; \xi^T e = 0. \tag{1}$$

It is clear that the above optimization problem is equivalent to finding the eigenvector associated with the second smallest eigenvalue of the following EVD problem.

$$L\xi = \lambda \xi. \tag{2}$$

Suppose we have found (*n-k*) stable elements after several steps of iteration, then the rows and columns of the Laplacian matrix *L* can be re-organized as the matrix on the left-hand side of (3) shows so that the first (*n-k*) rows (columns) correspond to the stable elements (which are to be fixed in the subsequent iterations) and the rest *k* rows (columns) correspond to the unfixed elements. As the Laplacian matrix is symmetrical, we have $L_{12}^T = L_{21}$. After applying the matrix compression strategy, the (*n-k*) stable elements are merged to an aggregated positive point and an aggregated negative one. Thus we can compress *L* into a (*k*+2)×(*k*+2) matrix $\hat{L} = \{\hat{l}_{ij}\}$ like the matrix on the right-hand side of (3). To keep $\hat{L}$ symmetrical, we let $\hat{L}_{12}^T = \hat{L}_{21}$. As the matrix compression strategy will not change the interrelations between the unfixed points, L_{22} remains unchanged in $\hat{L}$.

$$L = \begin{array}{c|cc} & n-k & k \\ \hline n-k & L_{11} & L_{12} \\ k & L_{21} & L_{22} \end{array} \quad \overset{\textit{compress}}{\Rightarrow} \quad \hat{L} = \begin{array}{c|cc} & 2 & k \\ \hline 2 & \hat{L}_{11} & \hat{L}_{12} \\ k & \hat{L}_{21} & L_{22} \end{array} \tag{3}$$

After the compression of the Laplacian matrix, the original spectral clustering problem (1) is converted to a smaller-scaled spectral clustering problem as follows.

$$\min \hat{\xi}^T \hat{L} \hat{\xi} \quad \textit{subject to} \quad \hat{\xi}^T \hat{\xi} = 1, \; \hat{\xi}^T e = 0 . \tag{4}$$

Here $\hat{\xi}$ is an (*k*+2)-dimensional embedding vector. It is natural that the solution to (4) is the eigenvector corresponding to the second smallest eigenvalue of the following EVD problem.

$$\hat{L}\hat{\xi} = \hat{\lambda}\hat{\xi} . \tag{5}$$

According to the block structures of L and $\hat{L}$, we rewrite ξ and $\hat{\xi}$ by $\xi = [\xi_1 \quad \xi_2]^T$ and $\hat{\xi} = \left[\hat{\xi}_1 \quad \hat{\xi}_2\right]^T$. Our objective is *to keep the second smallest eigenvalues of* L *and* $\hat{L}$ *equal to each other, and to keep the embedding values of the unfixed elements calculated from* L *and* $\hat{L}$ *exactly the same, i.e., to keep* $\hat{\lambda} = \lambda$ *and* $\hat{\xi}_2 = \xi_2$. For this purpose, we will investigate how to build the Laplacian matrix $\hat{L}$, i.e., how to determine $\hat{L}_{11}$, $\hat{L}_{12}$, and $\hat{L}_{21}$.

Letting $\hat{\xi}_2 = \xi_2$, from the constraints of the original optimization problem (1) and the new optimization problem (4), we can have the following equations.

$$\left.\begin{array}{lll} \xi_1^T\xi_1+\xi_2^T\xi_2=1, & \hat{\xi}_1^T\hat{\xi}_1+\xi_2^T\xi_2=1 & \Rightarrow\ \hat{\xi}_1^T\hat{\xi}_1=\xi_1^T\xi_1 \\ \xi_1^Te+\xi_2^Te=0, & \hat{\xi}_1^Te+\xi_2^Te=0 & \Rightarrow\ \hat{\xi}_1^Te=\xi_1^Te \end{array}\right\}. \tag{6}$$

As we want to fix the stable elements in the embedding vector of the original problem to some discrete values, we can assume $\hat{\xi}_1=(a,-b)^T, a>0, b>0,\ \xi_1=(c_1,c_2,\dots,c_{n-k})^T$ and denote $d_1=\xi_1^T\xi_1=\sum_{i=1}^{n-k}c_i^2,\ d_2=\xi_1^Te=\sum_{i=1}^{n-k}c_i$. Then by substituting the above notations to (6) and solving the corresponding equation set, we have

$$a=\left(d_2+\sqrt{2d_1-d_2^2}\right)\Big/2,\ \ b=\left(-d_2+\sqrt{2d_1-d_2^2}\right)\Big/2\,. \tag{7}$$

Therefore, we can compute $\hat{\xi}_1$ with (7) based on ξ_1, as a necessary condition to guarantee $\hat{\xi}_2=\xi_2$.

Decomposing the matrices in the EVD equations (2) and (5) into blocks, and considering $\hat{\lambda}=\lambda$ and $\hat{\xi}_2=\xi_2$, we can get

$$\begin{array}{llll} L_{11}\xi_1+L_{12}\xi_2=\lambda\xi_1 & (i); & L_{21}\xi_1+L_{22}\xi_2=\lambda\xi_2 & (ii); \\ \hat{L}_{11}\hat{\xi}_1+\hat{L}_{12}\hat{\xi}_2=\lambda\hat{\xi}_1 & (iii); & \hat{L}_{21}\hat{\xi}_1+L_{22}\hat{\xi}_2=\lambda\hat{\xi}_2 & (iv). \end{array} \tag{8}$$

From (8)(*ii*) and (8)(*iv*), we have

$$\hat{L}_{21}\hat{\xi}_1=L_{21}\xi_1\,. \tag{9}$$

With the notations $\hat{\xi}_1,\ \xi_1, d_1, d_2$ defined above, we can rewrite equation (9) as below.

$$a\hat{l}_{i1}-b\hat{l}_{i2}=\sum_{j=1}^{n-k}c_jl_{(i+n-k-2),j},\ \ (i=3,\dots,k+2)\,. \tag{10}$$

The sum of each row (column) of a Laplacian matrix should be zero. Therefore, for the last *k* rows of the two Laplacian matrices L and $\hat{L}$, we have

$$\hat{l}_{i1}+\hat{l}_{i2}=\sum_{j=1}^{n-k}l_{(i+n-k-2),j},\ \ (i=3,\dots,k+2)\,. \tag{11}$$

By solving the equation set as shown in (10) and (11), we can eventually get

$$\begin{cases} \hat{l}_{i1}=\left[\sum_{j=1}^{n-k}(b+c_j)l_{(i+n-k-2),j}\right]\Big/(a+b) \\ \hat{l}_{i2}=\left[\sum_{j=1}^{n-k}(a-c_j)l_{(i+n-k-2),j}\right]\Big/(a+b) \end{cases},\ \ (i=3,\dots,k+2)\cdot \tag{12}$$

From (12), we can see that the weights between the two aggregated points and the unfixed points are the linear combinations of the weights between the original fixed points and the unfixed points. In other words, given ξ_1 and the Laplacian matrix *L*, we can calculate $\hat{L}_{12}$ and $\hat{L}_{21}$ using (12) with very little computational cost.

In the next step, we will discuss how to compute $\hat{L}_{11}$. According to the property of the Laplacian matrix, the sums of the first two columns of $\hat{L}$ should also be zeros. As a result, we have,

$$\begin{cases} \hat{l}_{11}+\hat{l}_{21}=-\sum_{i=3}^{k+2}\hat{l}_{i1}=-\left[\sum_{i=n-k+1}^{n}\sum_{j=1}^{n-k}(b+c_j)l_{ij}\right]\Big/(a+b)=-\left[be^TL_{21}e+e^TL_{21}\xi_1\right]\Big/(a+b) \\ \hat{l}_{12}+\hat{l}_{22}=-\sum_{i=3}^{k+2}\hat{l}_{i2}=-\left[\sum_{i=n-k+1}^{n}\sum_{j=1}^{n-k}(a-c_j)l_{ij}\right]\Big/(a+b)=-\left[ae^TL_{21}e-e^TL_{21}\xi_1\right]\Big/(a+b) \\ \hat{l}_{21}=\hat{l}_{12} \end{cases}. \qquad (13)$$

There are four unknown quantities and three equations in (13), so we have to find another equation to work out the unique solution for this equation set. This additional equation comes from the objective functions of the two optimization problems. Since we would like to keep the second smallest eigenvalues of (2) and (5) equal to each other, we can get the following equation.

$$\xi^T L\xi = \hat{\xi}^T \hat{L}\hat{\xi}\,. \qquad (14)$$

Decomposing the matrices and vectors in (14) into blocks, and considering (9), we can get the following equation.

$$\xi_1^T L_{11}\xi_1 = \hat{\xi}_1^T \hat{L}_{11}\hat{\xi}_1\,. \qquad (15)$$

According to the notations $\hat{\xi}_1$, ξ_1, d_1, d_2 defined above, equation (15) is equivalent to

$$a^2\hat{l}_{11} - 2ab(\hat{l}_{12}+\hat{l}_{21}) + b^2\hat{l}_{22} = \xi_1^T L_{11}\xi_1\,. \qquad (16)$$

Adding (16) to the equation set (13), we eventually have a solvable equation set which solution is shown as follows.

$$\begin{cases} \hat{l}_{11} = \left[\xi_1^T L_{11}\xi_1 - 2be^T L_{21}\xi_1 - b^2 e^T L_{21} e\right]\big/(a+b)^2 \\ \hat{l}_{12} = \hat{l}_{21} = -\left[\xi_1^T L_{11}\xi_1 + (a-b)e^T L_{21}\xi_1 + abe^T L_{21} e\right]\big/(a+b)^2 \\ \hat{l}_{22} = \left[\xi_1^T L_{11}\xi_1 + 2ae^T L_{21}\xi_1 - a^2 e^T L_{21} e\right]\big/(a+b)^2 \end{cases}. \qquad (17)$$

Up to now, we have successfully computed all the elements in the reduced Laplacian matrix $\hat{L}$ based on ξ_1 and L. Actually it is easy to understand what we get is not only a necessary condition but also a sufficient condition for $\hat{\xi}_2 = \xi_2$. That is, if ξ_1 is precise, we can exactly have $\hat{\xi}_2 = \xi_2$. In other words, even if we merge some stable points using the sequential matrix compression strategy, the embedding vector of the unfixed points will not be influenced. Therefore, we have derived a lossless method to scale down the computation cost of spectral clustering problems. We summarized this method as the ratio-cut **S**equential **M**atrix **C**ompression (SMC) algorithm in Table 1.

As for the initialization, we load the Laplacian matrix of the original optimization problem and compute its eigenvector associated with the second smallest eigenvalue by a certain EVD solver. After a certain number of iteration steps, we break off the

EVD solver and check the current status of the eigenvector $\xi^{(t)}$ in order to find out some stable elements (denoted by the sub-vector $\xi_1^{(t)}$). From $\xi_1^{(t)}$, the aggregated sub-vector $\hat{\xi}_1^{(t)}$ can be calculated by (7). Then the elements of the compressed Laplacian matrix $\hat{L}$ can be computed by (12) and (17). If the scale of matrix $\hat{L}$ is still large, another round of matrix compression might be conducted; otherwise, the EVD problem is worked out directly and the clustering result is generated accordingly.

Table 1. The SMC method

1.	Set $t = 0$, and input the original Laplacian matrix $L^{(t)}$.
2.	Compute the eigenvector according to the second smallest eigenvalue of $L^{(t)}$ by CG-based EVD solver, and break off the process after a certain number of iteration steps.
3.	From the current status of the eigenvector $\xi^{(t)}$, select a vector $\xi_1^{(t)}$ whose elements are regarded as stable points. At the same time, $L_{22}^{(t)}$ is obtained and $L_{22}^{(t+1)} = L_{22}^{(t)}$.
4.	Compute $\hat{\xi}_1^{(t)} = (a^{(t)}, -b^{(t)})^T$ by (7).
5.	Compute $\hat{l}_{i1}^{(t)}, \hat{l}_{i2}^{(t)}$, $(i = 3, \ldots, k^{(t)} + 2)$ by (12) so that $L_{21}^{(t+1)}$ is obtained and $L_{12}^{(t+1)} = (L_{21}^{(t+1)})^T$.
6.	Compute $\hat{l}_{11}^{(t)}, \hat{l}_{12}^{(t)}, \hat{l}_{21}^{(t)}, \hat{l}_{22}^{(t)}$ by (17) to obtain $L_{11}^{(t+1)}$.
7.	Build the compressed Laplacian matrix $L^{(t+1)} = \begin{bmatrix} L_{11}^{(t+1)} & L_{12}^{(t+1)} \\ L_{21}^{(t+1)} & L_{22}^{(t+1)} \end{bmatrix}$ and go to Step 2 if the scale of $L^{(t+1)}$ is still large; otherwise, solve the eigenvalue problem of $L^{(t+1)}$ to get the corresponding eigenvector, and output the embedding vector after some necessary post-processing.

It is easy to get that the overall complexity of our proposed SMC method is $O(kn^2)$, where k is the initial iteration steps (usually less than ten), and n is the scale of the original Laplacian matrix. For comparison, the complexity of CG-based EVD solver is $O(mn^2)$, where m is the total iteration steps (usually several hundred or even larger); while Lanczos-based EVD solver takes even more computational burden than CG-based EVD solver.

Note that after an error bound analysis, we can prove that the SMC algorithm is almost lossless. Moreover, the idea of sequential matrix compression can be easily extended to adapt the case of normalized cut, and most of the derivations are quite similar to those of the ratio cut. We omitted the above content due to the space limitation.

3 Experiments

In this section, we report the experiments that we conducted to show the efficiency and effectiveness of the proposed SMC algorithm. Considering that CG-based and

Lanczos-based EVD solvers are among the most popular and efficient algorithms for the EVD of sparse matrices, we use them as the baselines in our experiments. And since our theoretical derivations of the SMC algorithm is based on two-way clustering, all the experiments are also designed for two-way clustering.

When implementing our SMC method, CG-based EVD solver was adopted to compute the embedding vector of the optimization problem (1). After several steps of iteration, we used the following simple strategy to extract the stable sub-vector ξ_1. Suppose there are n_1 positive points and n_2 negative points in ξ . Then the top $p\%$ of the positive points in ξ ($n_1p\%$ in number) were regarded as the positive working set, while the bottom $p\%$ of the negative points in ξ ($n_2p\%$ in number) were regarded as the negative working set. Here $p\%$ is referred to as the proportion of fixed points. In the working sets, if the absolute value of an element's gradient was smaller than a very small threshold (e.g., 0.001.), this element would be regarded as a stable point. All positive stable points were merged to a new aggregated positive point in the next step iteration. Similarly, all negative stable points were merged to a new aggregated negative point. This process can be conducted in a recursive manner until all the data points are merged into two points, which indicates that all the points have been clustered into either of the two clusters.

We ran the SMC algorithm and the reference algorithms on the 20-newsgroups [5] dataset. Each document was represented by a feature vector of term frequency [1], and the weights in the adjacency matrix were calculated by (18), where v_i is the feature vector of the corresponding document.

$$W(i,j) = v_i^T v_j \Big/ \left(\|v_i\|_2 \|v_j\|_2 \right) \tag{18}$$

Table 2. Average clustering error rate and average time cost for 20-newsgroups dataset

Category name	Average clustering error rate			Average time cost		
	CG	Lanczos	SMC	CG	Lanczos	SMC
alt.atheism	0.027	**0.026**	0.039	1.325	5.440	**0.888**
comp.graphics	0.034	**0.005**	0.014	1.299	5.513	**0.881**
comp.os.ms-windows.misc	0.028	**0.011**	0.057	1.321	5.535	**0.902**
comp.sys.ibm.pc.hardware	0.023	**0.006**	0.020	1.173	5.509	**0.903**
comp.sys.mac.hardware	0.009	**0.005**	0.015	1.304	5.498	**0.895**
comp.windows.x.	**0.005**	0.010	0.022	1.280	5.600	**0.887**
misc.forsale	**0.003**	0.004	0.012	1.199	5.518	**0.887**
rec.autos	**0.004**	0.006	0.038	1.251	5.448	**0.896**
rec.motorcycles	0.008	**0.003**	0.010	1.265	5.455	**0.887**
rec.sport.baseball	0.005	**0.003**	0.039	1.279	5.450	**0.887**
rec.sport.hockey	0.006	**0.002**	0.012	1.291	5.476	**0.897**
sci.crypt	0.011	**0.006**	0.007	1.261	5.470	**0.900**
sci.electronics	0.043	**0.015**	0.032	1.376	5.501	**0.905**
sci.med	**0.007**	0.009	0.028	1.300	5.443	**0.895**
sci.space	0.040	**0.004**	0.048	1.451	5.435	**0.902**
soc.religion.christian	0.077	**0.001**	0.006	1.382	5.440	**0.892**
talk.politics.guns	0.044	**0.024**	0.058	1.386	5.435	**0.901**
talk.politics.mideast	**0.051**	0.058	0.079	1.290	5.455	**0.898**
talk.politics.misc	0.058	**0.034**	0.054	1.367	5.440	**0.889**
talk.religion.misc	0.062	**0.041**	0.057	1.291	5.472	**0.894**
Average	0.027	**0.014**	0.032	1.305	5.477	**0.894**

We tested our algorithm and the reference algorithms on every possible pair of categories in the 20-newsgroups dataset. The average performance for between each category and all the other categories are listed in Table 2, where the surpassing values are blackened. For average clustering error rate, we can see that the proposed SMC algorithm performed slightly worse than the reference algorithms. This is reasonable because we used an approximation of ξ_1 instead of the precise vector when compressing the Laplacian matrix. However, on average, the clustering error rates of all these algorithms were all very low and the differences between them were negligible. For average time cost, we can see that the SMC algorithm defeated the other two methods by much. This verified that our algorithm could depress the time complexity of spectral clustering. Overall speaking, the SMC algorithm can achieve higher efficiency with very little accuracy loss compared with the reference algorithms.

4 Conclusions

In this paper, we proposed a sequential matrix compression strategy to accelerate spectral clustering in order to fit the need of large-scale applications. The basic idea is to sequentially depress the scale of the Laplacian matrix in the iteration steps of spectral clustering. Experiments showed the efficiency and feasibility of our method.

References

[1] Baeza-Yates, R., and Ribeiro-Neto, B. Modern information retrieval. ACM Press, a Division of the Association for Computing Machinery, Inc. (ACM). 1999.

[2] Golub, G.H., and Loan, C.F.V. Matrix computations. Johns Hopking University Press, 3rd edition, 1996.

[3] Hagen, L., and Kahng, A.B. New spectral methods for ratio cut partitioning and clustering. IEEE Transactions on CAD, 11:1074-1085, 1992.

[4] Knyazev, A.V. Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method. SIAM Journal on Scientific Computing, Volume 23, Number 2, pp. 517-541, 2001.

[5] Lang, K. News Weeder: Learning to filter netnews. In ICML-95, pp. 331-339, 1995.

[6] Shi, J., and Malik, J. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8):888-905, August 2000.

[7] Sorensen, D.C. Implicitly-restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations. Technical Report TR-96-40, 1996.

[8] Yang, X.P., Sarkar, TK., Arvas, E. A Survey of Conjugate Gradient Algorithms for Solution of Extreme Eigen-problems of a Symmetric Matrix. IEEE Transactions on Acoustics Speech and Singal Processing, Vol 37, NO. 1000, pp.1550-1555, 1989.

An Information-Theoretic Framework for High-Order Co-clustering of Heterogeneous Objects

Antonio D. Chiaravalloti[2], Gianluigi Greco[1], Antonella Guzzo[2], and Luigi Pontieri[2]

[1] Dept. of Mathematics, UNICAL, Via P. Bucci 30B, 87036, Rende, Italy
[2] ICAR, CNR, Via Pietro Bucci 41C, 87036 Rende, Italy
{ggreco}@mat.unical.it
{chiaravalloti, guzzo, pontieri}@icar.cnr.it

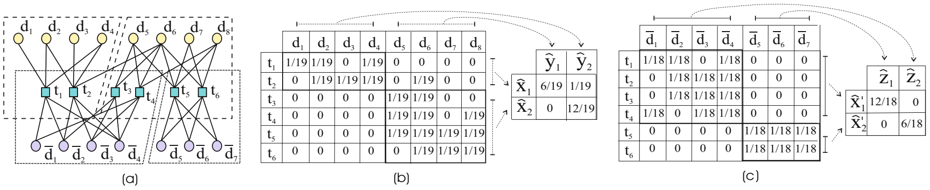
Abstract. The *high-order co-clustering* problem, i.e., the problem of simultaneously clustering several heterogeneous types of domains, is usually faced by minimizing a linear combination of some optimization functions evaluated over pairs of correlated domains, where each weight expresses the reliability/relevance of the associated contingency table. Clearly enough, accurately choosing these weights is crucial to the effectiveness of the co-clustering, and techniques for their automatic tuning are particularly desirable, which are instead missing in the literature. This paper faces this issue by proposing an information-theoretic framework where the co-clustering problem does not need any explicit weighting scheme for combining pairwise objective functions, while a suitable notion of *agreement* among these functions is exploited. Based on this notion, an algorithm for co-clustering a "star-structured" collection of domains is defined.

Keywords: Co-clustering, Mutual Information, Heterogeneous Data.

1 Introduction

The problem of clustering heterogeneous objects has become an active research area in recent years. In particular, there is a great deal of literature addressing the clustering of two different types of objects, hereinafter called *domains* or *dimensions*, such as documents and terms in text corpora (e.g., [7,4]). This task, usually known as (bi-dimensional) *co-clustering* or *bi-clustering*, has been faced by way of different strategies, including spectral [7,4] and information-theoretic approaches [5,3,1].

Some recent works have generalized the bi-clustering problem to the case of more than two domains (short: *high-order* co-clustering problem) [2,6,8]. In particular, [6] considered a co-clustering problem for "star"-structured domains of the form D_X, $D_{Y^1}, ..., D_{Y^N}$ (where $N > 1$ and D_X is the central domain), by defining an objective function f_{X,Y^i}, for each "auxiliary" domain D_{Y^i}, whose optimization should lead to the isolation of the best co-clusters over D_X and D_{Y^i}. In order to integrate all such (bi-dimensional) co-clustering subproblems, a linear combination of all pairwise objective functions is optimized, subject to the constraint that consistent clusters are found for the central (shared) domain. More precisely, for each domain D_{Y^i}, the objective function for co-clustering D_X and D_{Y^i} is weighted with a factor β_i, such that $\sum_{i=1}^{N} \beta_i = 1$. Clearly, extending this setting to arbitrary pairwise interactions mainly requires to equip with a weight, say $\beta_{A,B}$, each pair of (arbitrarily) correlated domains A and B (cf. [2]).



	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
t_1	1/19	1/19	0	1/19	0	0	0	0
t_2	0	1/19	1/19	1/19	0	1/19	0	0
t_3	0	0	0	0	1/19	1/19	0	0
t_4	0	0	0	0	1/19	1/19	0	1/19
t_5	0	0	0	0	1/19	1/19	1/19	1/19
t_6	0	0	0	0	0	1/19	1/19	1/19

	$\widehat{y}_1$	$\widehat{y}_2$
$\widehat{x}_1$	6/19	1/19
$\widehat{x}_2$	0	12/19

	$\overline{d}_1$	$\overline{d}_2$	$\overline{d}_3$	$\overline{d}_4$	$\overline{d}_5$	$\overline{d}_6$	$\overline{d}_7$
t_1	1/18	1/18	0	1/18	0	0	0
t_2	0	1/18	1/18	1/18	0	0	0
t_3	0	1/18	1/18	1/18	0	0	0
t_4	1/18	0	1/18	1/18	0	0	0
t_5	0	0	0	0	1/18	1/18	1/18
t_6	0	0	0	0	1/18	1/18	1/18

	$\widehat{z}_1$	$\widehat{z}_2$
$\widehat{x}'_1$	12/18	0
$\widehat{x}'_2$	0	6/18

Fig. 1. Co-clustering a text corpus: (a) Spectral and (b-c) Information-theoretic approach

For example, Fig. 3(a) reports a chart (discussed in Section 4) that evidences the quality of the clustering (measured via the *loss in mutual information* occurring when replacing the original domains with their clustered versions) at the varying of β_1, over two syntectic datasets having two auxiliary domains. It is easy to see that the best (cf. minimum) value for the loss of mutual information in the first domain is achieved when $\beta_1 = 1$, i.e., when the clustering is performed along this domain only. Similarly, the best value for the second domain is achieved when $\beta_1 = 0$ (and $\beta_2 = 1$). Thus, setting the values for the weights strongly affects the quality of the results over the co-clustering.

In general, there may be not enough knowledge on the reliability/relevance of pair-wise correlation data to set the weights precisely. Hence, some method for their automatic tuning should be defined, as already stated in [6]. In fact, we mainly aim at facing such an open issue, for the specific case of star-structured domains.

In more details, in Section 2, we introduce an information-theoretic framework which generalizes that in [5] and allows to co-cluster an arbitrary number of domains forming a star structure. Notably, this setting fits a wide range of relevant real-world data, like those describing relationships among authors, conferences, papers and keywords in academic publications (with publications constituting the central domain).

In order to address such a problem without using any arbitrary weighting scheme, in Section 3, we propose and study an algorithm that solves the High-Order Co-Clustering by computing Agreements for contrasting Domain objective functions (short: `AD-HOCC` algorithm). The basic idea is to consider a notion of *agreement*, such that a clustering of the central domain is found which guarantees that each partial objective function is not "too far" from its optimal value.

Results from test on both synthetic and real data are finally illustrated in Section 4.

2 Formal Framework

Let $D_X = \{x_1, ..., x_m\}$, $D_{Y^1} = \{y_1^1, ..., y_n^1\}, ..., D_{Y^N} = \{y_1^N, ..., y_n^N\}$ be $N + 1$ domains, i.e., sets of values, and let $X, Y^1, ..., Y^N$ be discrete random variables taking values in D_X, $D_{Y^1}, ..., D_{Y^n}$, resp. The domains are assumed to form a *star* structure, i.e., the auxiliary domains D_{Y^i}, for $i = 1..n$ are pairwise independent, while each of them is correlated with the central domain D_X. Let then $p_i(X, Y^i)$, with $1 \leq i \leq N$, denote the joint probability distribution between X and Y^i, i.e., $p_i(x, y^i)$ is the probability that X takes the value $x \in D_X$ *and* Y^i takes the value $y^i \in D_{Y^i}$.

Assume that the values of D_X are to be clustered into k clusters, say $\widehat{D}_X = \{\widehat{x}_1, \widehat{x}_2, ..., \widehat{x}_k\}$, and those of D_{Y^i} in l_i clusters, say $\widehat{D}_{Y^i} = \{\widehat{y}_1, \widehat{y}_2, ..., \widehat{y}_{l_i}\}$, for each i

in $1..N$. Then, a *high-order co-clustering* for $Y^1, ..., Y^N$ w.r.t. X is a tuple $C = \langle C_X, C_{Y^1}, ..., C_{Y^N} \rangle$, such that $C_X : D_X \mapsto \widehat{D}_X$, and $C_Y : D_{Y^i} \mapsto \widehat{D}_{Y^i}$, for $i = 1 \dots N$. For brevity, for each random variable W and its associated domain D_W, the set of all possible mappings from D_W to its clusters is denoted by $\mathcal{P}(W)$. Moreover $\widehat{W}_{C_W} = C_W(W)$ is the random variable denoting the cluster assigned to W, through the function C_W, defined on D_W and ranging over $\mathcal{P}(W)$. Like in [5], we use lower-case letters for domain elements, and upper-case letters for the random variable ranging over them; in addition, hatted letters are reserved for clusters, and clustered random variables. Also, $\widehat{W}_{C_W}$ will be shortened as $\widehat{W}$ whenever C_W is clear from the context.

Example 1. Let $\mathrm{d}_1, ..., \mathrm{d}_6$ be documents (e.g., academic papers), and $\mathrm{t}_1, ..., \mathrm{t}_8$ be terms. In Fig. 1.(a) the occurrences of terms in documents are represented as edges, while terms and documents are depicted as nodes (of two different types). The problem can be easily modelled in an information-theoretic framework by defining X and Y to be two random variables taking values in $\{\mathrm{d}_1, ..., \mathrm{d}_6\}$ and $\{\mathrm{t}_1, ..., \mathrm{t}_8\}$, respectively. Let $p(X, Y)$ be the joint probability distribution between X and Y represented in a tabular form in Fig. 1(b).[1] E.g., $p(\mathrm{d}_1, \mathrm{t}_2) = \frac{1}{19}$ is the *frequency* of the event of having t_2 occur in document d_1, while $p(\mathrm{d}_1, \mathrm{t}_3) = 0$ indicates that t_3 does not occur in d_1. In this setting we can consider the problem of co-clustering both documents and terms in two clusters, say $\{\widehat{x}_1, \widehat{x}_2\}$ and, resp., $\{\widehat{y}_1, \widehat{y}_2\}$. An example co-clustering $\langle C_X, C_Y \rangle$ is shown in the same figure, where C_X is the function mapping d_1 and d_2 to $\widehat{x}_1$ and every other document to $\widehat{x}_2$, while C_Y maps $\mathrm{t}_1, ..., \mathrm{t}_4$ to $\widehat{y}_1$ and all the other terms to $\widehat{y}_2$.

Assume that some information is available on document authors, say $\mathrm{a}_1, ..., \mathrm{a}_7$. Based on authorship data, reported again in Fig. 1(a), we can consider the problem of co-clustering all of the three domains. This is just a *high-order* co-clustering problem over star-structured domains, which essentially amounts to finding a tuple $\langle C_X, C_Y, C_Z \rangle$, where Z is a random variable taking values in $\{\mathrm{a}_1, ..., \mathrm{a}_7\}$. If one looks at authors independently of terms, a natural co-clustering is $\langle C'_X, C_Z \rangle$, where C'_X is the function mapping $\mathrm{d}_1, ..., \mathrm{d}_4$ to $\widehat{x}_1$ and every other document to $\widehat{x}_2$, and C_Z is the function mapping $\mathrm{a}_1, ..., \mathrm{a}_4$ to $\widehat{z}_1$ and every other author to $\widehat{z}_2$ — see Fig. 1(c). $\lhd$

As the effect of co-clustering can be viewed as a sort of information compression, the co-clustering problem can be turned in the search for a fixed-size compression scheme preserving as much as possible of the original mutual information. To this aim, for any auxiliary domain D_{Y^i}, one can compute the *mutual information* $I(X; Y^i)$ between the random variables X and Y^i, ranging over D_X and D_{Y^i}, respectively. Then, the quality of a multi-dimensional co-clustering can be assessed by taking into account the loss of mutual information that occurs when replacing the original variables X, $Y^1, ..., Y^N$ with their clustered versions $\widehat{X}_{C_X}, \widehat{Y^1}_{C_{Y^1}}, \widehat{Y^N}_{C_{Y^N}}$. Hereinafter, for brevity, the loss of mutual information pertaining the i-th auxiliary dimension will be denoted by $\Delta I_i(C_X, C_{Y^i}) = I(X; Y^i) - I(\widehat{X}_{C_X}; \widehat{Y^i}_{C_{Y^i}})$ (or, shortly, ΔI_i).

For the base case of $N = 1$, an algorithm that computes a (locally) optimal co-clustering has been presented in [5], where it was shown that the mutual information

[1] For the sake of exposition, the joint distribution shown here just results from normalizing a binary association matrix between terms and documents.

loss caused by clustering X and Y^i can be expressed as a dissimilarity between the original joint distribution $p_i(\cdot,\cdot)$ and a function $q_i(\cdot,\cdot)$ that approximates it. More precisely: $\Delta I_i = \mathcal{D}(p_i(X,Y^i)||q_i(X,Y^i))$, where $\mathcal{D}(\cdot||\cdot)$ denotes the Kullback-Leibler (KL) divergence, and $q_i(X,Y^i)$ is a function, preserving all marginals of p_i, of the form $q_i(X,Y) = p_i(\widehat{X},\widehat{Y}^i) \cdot p_i(X|\widehat{X}) \cdot p_i(Y|\widehat{Y}^i)$.

Thus, the pairwise co-clustering problem can be solved by searching for the function q_i that is most similar to p_i, according to $\mathcal{D}$. To this purpose, an alternate minimization scheme is used in [5] which considers only one dimension per time. In the rest of the paper, we investigate the case of $N > 1$, and define a co-clustering approach that ensures as low as possible values for all the information loss functions ΔI_i.

3 Agreement Method

A major problem occurring while optimizing the losses of mutual information ΔI_i, for each $1 \leq i \leq N$, is that the best co-clustering for D_X and D_{Y^i} may not comply with the best co-clustering for D_X and D_{Y^j} for $j \neq i$, over the values in domain D_X.

Example 2. Consider again the data set in Fig. 1. While co-clustering along each auxiliary dimensions independently, the co-clusterings $\langle C_X, C_Y\rangle$ and $\langle C'_X, C_Z\rangle$ introduced in Example 1 seem to be very good candidate solutions. This should be also evident by looking at the graphical representation in Fig. 1(a), where these clusters in fact induce some "optimal" cut over the nodes. However, these two optimal bi-clusterings do not conform with each other, since $C_X \neq C'_X$ and, therefore, there is no immediate way for extending them into a global high-order co-clustering. $\lhd$

As discussed previously, a way for jointly optimizing the losses of mutual information ΔI_i, for each $1 \leq i \leq N$, is to linearly combine these individual functions into a global one. Thus, one can try to minimize the quantity $\sum_{i=1}^{N} \beta_i \cdot \Delta I_i$, where $\beta_1,...,\beta_N$ are suitable weights such that $\sum_{i=1}^{N} \beta_i = 1$.

Since, in general, there may be no knowledge enough to set the coefficients in a precise manner, we abandon the idea of using a pre-fixed weighting scheme and restate the co-clustering problem as a multi-objective optimization of all functions $\Delta I_i = I(X;Y^i) - I(\widehat{X};\widehat{Y}^i)$.

Moreover, it may well be the case that no clustering function C_X exists over the values of the central domain that allows to achieve a minimal information loss over all the other dimensions – this is, e.g., the case of the clusters in Fig. 1. Therefore, we introduce a notion of agreement to represent a sort of optimal "compromise" among the different (and potentially discordant) goals, which are autonomously pursued by all the bi-dimensional co-clustering subproblems.

Actually, it can be shown that computing an *optimal agreement* is NP-hard. Indeed, it requires the computation of the optimal clustering over each dimension alone, which is NP-hard of its own. Accordingly, the following definition states, in a pragmatic way, the notion of agreement under a "local" perspective, only.

Definition 1. *A high-order co-clustering* $C = \langle C_X, C_{Y^1}, ..., C_{Y^N}\rangle$ *is said to be an* α-agreement *for* Y^1,...,Y^N *w.r.t.* X *if, for each* Y^i *with* $1 \leq i \leq N$, *the following conditions hold:*

```
Input:   Domains $D_X$, $D_{Y^1}$,...,$D_{Y^N}$, cluster sets $\widehat{D}_X$, $\widehat{D}_{Y^1}$,...,$\widehat{D}_{Y^N}$,
         a real number $\varepsilon$, and joint distributions $p_1(X,Y^1),...,p_N(X,Y^N)$;
Output:  An $\alpha$-agreement for $Y^1$,...,$Y^N$ w.r.t. $X$;

Define an arbitrary co-clustering $\langle C_X^0, C_{Y^1}^0, \dots, C_{Y^N}^0 \rangle$;
set $\alpha^{(0)} = 0, t = t_i^* = 0, \varDelta I_i^{(0)} = +\infty, \varepsilon_i = \varepsilon, \forall i \in \{1..N\}$;
repeat
  Compute $q_i^{(t)}$, for $i = 1 \dots N$, and set $t = t + 1$;
  for each $Y^i$ and $y \in D_{Y^i}$ do
    $C_{Y^i}^{(t)}(y) = \arg\min_{\widehat{y} \in \widehat{D}_{Y^i}} \mathcal{D}(p_i(X|y)||q_i^{(t-1)}(X|\widehat{y}))$;
  for each $Y^i$ do
    if $\varDelta I_i^{(t)} < \varDelta I_i^{(t_i^*)}$ then $t_i^* = t$ else $\varepsilon_i = \varepsilon_i/2$;
  Compute $q_i^{(t)}$, for $i = 1 \dots N$, and set $t = t + 1$;
  for each $x \in D_X$ do
    let $\delta I_i(x, \widehat{x}_j) = \mathcal{D}(p(Y^i|x)||q_i^{(t-1)}(Y^i|\widehat{x}_j)), \forall \widehat{x}_j \in \widehat{D}_X$;
    let $\alpha(x, \widehat{x}_j) = \min_{Y^i} \frac{\min_{\widehat{x}' \in \widehat{D}_X} \delta I_i(x,\widehat{x}')}{\delta I_i(x,\widehat{x}_j))}, \forall \widehat{x}_j \in \widehat{D}_X$;
    $C_X^{(t)}(x) = \arg\max_{\widehat{x} \in \widehat{D}_X} \alpha(x, \widehat{x})$;
  end for
  let $\alpha^{(t)} = \min_{x \in D_X} \max_{\widehat{x} \in \widehat{D}_X} \alpha(x, \widehat{x})$;
  let $\varDelta I_i^{(t)} = \mathcal{D}(p_i(X^{(t)}, Y^{i(t-1)})||q_i(X^{(t)}, Y^{i(t-1)})), \forall Y^i$;
while ($\alpha^{(t)} \simeq \alpha^{(t-2)}$ and $\nexists Y^i$ s.t. $(1 - \varepsilon_i) \cdot \varDelta I_i^{(t)} > \varDelta I_i^{(t_i^*)}$) or $\alpha^{(t)} > \alpha^{(t-2)}$;
return $\langle C_X^{(t-2)}, C_{Y^1}^{(t-3)}, ..., C_{Y^N}^{(t-3)} \rangle$;
```

Fig. 2. Algorithm AD-HOCC

(a) $\forall C' \in \mathcal{P}(Y^i)$, $\varDelta I_i(C_X, C') \geq \varDelta I_i(C_X, C_{Y^i})$, *and*
(b) $\forall C'' \in \mathcal{P}(X)$, $\varDelta I_i(C'', C_{Y^i}) \geq \alpha \times [\varDelta I_i(C_X, C_{Y^i})]$.

If there exists no $\alpha' > \alpha$ satisfying condition (b), the agreement is said maximal. □

Notably, α is a sort of quality measure assessing the ability of approximating some local optimum of each function $\varDelta I_i$. As an extreme case, when $\alpha = 1$ and $N = 1$, an α-agreement is a local optimum for the two-dimensional co-clustering problem.

Algorithm AD-HOCC. In Fig. 2 an algorithm, named AD-HOCC, is shown that computes a maximal α-agreement, based on a local, alternate, optimization scheme. First an initial arbitrary co-clustering $\langle C_X^0, C_{Y^1}^0, \dots, C_{Y^N}^0 \rangle$ is computed, which is eventually refined in the main loop.

At each repetition t, the optimal clustering $C_{Y^i}^{(t)}$ is computed for each auxiliary domain Y^i ($1 \leq i \leq N$), based on the current clustering for the central domain X. Intuitively, this is carried out with the aim of assigning y to the cluster in $\widehat{D}_{Y^i}$ leading to the minimization of the loss of mutual information. Subsequently, the iteration continues by computing the optimal clustering $C_X^{(t)}$ for the central domain, based on the just computed clusterings for the auxiliary domains. This step is crucial for getting an agreement and is, thus, discussed in more details below.

Prelimiary, for each value $x \in D_X$ and each cluster $\widehat{x}_j \in \widehat{D}_X$, we compute the contribution $\delta I_i(x, \widehat{x}_j)$ that would be given to the information loss over each domain Y^i by assigning x to $\widehat{x}_j$. All these values are normalized w.r.t. the best possible assignment for x, i.e., $\min_{\widehat{x}' \in \widehat{D}_X} \delta I_i(x, \widehat{x}')$, and the worst possible mapping $\alpha(x, \widehat{x}_j)$ is computed.

The algorithm eventually chooses the best over such worst mappings, i.e., an element x is mapped to the cluster $\widehat{x} \in \widehat{D}_X$ maximizing the value $\alpha(x, \widehat{x})$, according to the formula:

$$C_X^{(t)}(x) = \arg \max_{\widehat{x} \in \widehat{D}_X} \alpha(x, \widehat{x}) \tag{1}$$

In addition, the value for $\alpha^{(t)}$ characterizing the guarantee for the agrement is computed according to the formula: $\alpha^{(t)} = \min_{x \in D_X} \max_{\widehat{x} \in \widehat{D}_X} \alpha(x, \widehat{x})$.

The algorithm keeps on iterating till the value $\alpha^{(t)}$ increases, i.e., as long as a better agreement is discovered. Actually, it tolerates that some bi-clustering objective function temporarily get worse, provided that it remains close enough to its best value found so far. To this aim a real number ε is also required in input to denote the range of tolerance admitted. Such a behavior is meant to reduce the risk of underestimating some partial optimum, as the strategy adopted here does not guarantee that every objective function monotonically decreases. Notice, however, that the width of these tolerance ranges is progressively reduced during the search, thus ensuring termination.

Discussion and comparison with related work. Let I be the number of iterations, N be the number of auxiliary domains, and M be is upper bound for both the size of any domain and the number of nonzero elements in any joint distribution matrix.

It is easy to see that algorithm AD-HOCC converges, at a step $t = O(N \cdot M \cdot (k + \sum_i l_i) \cdot I)$, to a high-order co-clustering for Y^1,...,Y^N w.r.t. X such that $\forall C'_X \in \mathcal{P}(X), C'_{Y^1} \in \mathcal{P}(Y^1), ..., C'_{Y^N} \in \mathcal{P}(Y^N)$, and for each i in $1..N$:

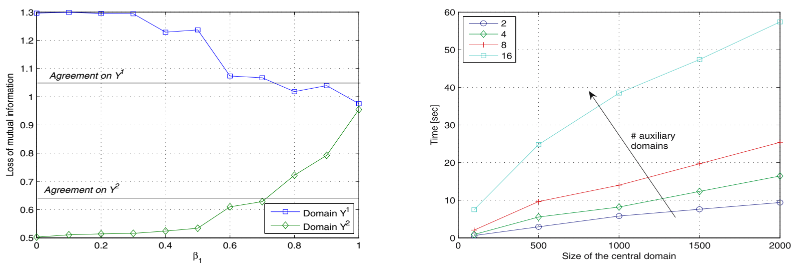
(a) $\Delta I_i(C_X^{(t-2)}, C'_{Y^i}) \geq \Delta I_i(C_X^{(t-2)}, C_{Y^i}^{(t-3)})$, and
(b) $\Delta I_i(C'_X, C_{Y^i}^{(t-3)}) \geq \alpha^{(t)} \Delta I_i(C_X^{(t-2)}, C_{Y^i}^{(t-3)})$.

Moreover, there is no $\bar{\alpha} > \alpha^t$ satisfying condition (b).

Note that when applied over domains D_X, D_{Y^1},...,D_{Y^N}, algorithm AD-HOCC does not, in general, guarantee that all pairwise mutual information losses monotonically decrease. Yet, at each step $t \geq 2$, the decrease in the mutual information I_i along each dimension Y^i is bounded: $\alpha^{(t)} \times \Delta I_i^{(t)} \leq \Delta I_i^{(t-2)}$.

As discussed above, previous approaches to the high-order co-clustering problem[2,6] score individual objective functions via some weights without discussing the problem of their automatic tuning. Besides addressing this issue, algorithm AD-HOCC overcomes some limitations of the approach in [6]. For instance, AD-HOCC can divide the domains in any arbitrary number of clusters, while the approach in [6] was explicitly developed for the *bi-partite* co-clustering problem, only.

Notably, the algorithm in [2] has been designed for general pairwise relationships, and can automatically select the number of cluster for each domain, based on a schema that interleaves top-down clustering of some domains and bottom-up clustering of the others, along with a local correction routine. However, this generality comes with a cost: the algorithm in [2] runs in $O(\max_W \{log|\widehat{D}_W|, log(|D_W|/|\widehat{D}_W|)\} \cdot \max_W \{|D_W|^3\})$,



(a) (b)

Fig. 3. Loss of mutual information (a) and computation time (b) on synthetic data

where W denotes any input domain, while a quadratic dependence on D_W is ensured only when two domains are to be co-clustered.

4 Experiments

This section provides an empirical study of the proposed technique, based on experiments over both synthetic and real data, which were performed by running a Java implementation of the algorithm in Section 3 on a 1600MHz/512MB Pentium IV machine equipped with *Windows XP Professional*. In order to reduce the statistical bias due to the choice of initial clusterings, every measure has been averaged over 10 runs.

Synthetic data have been produced through an ad-hoc Java generator with the following parameters: *(i)* the number $N > 1$ of auxiliary domains, *(ii)* the size of the domains D_X, D_{Y^1}, ..., D_{Y^N}, *(iii)* the number of required clusters along each domain, *(iv)* a noise factor θ, and *(v)* a "disagreement" factor γ. Roughly speaking, the latter basically expresses the maximum percentage of values in D_X that would be assigned to different clusters when considering two different contingency tables.

A first series of experiments have been conducted to asses the behavior of the AD-HOCC algorithm w.r.t. a "prototypical" co-clustering method that optimizes a linear combination of the information losses, with weights β_i. To this aim, a linear combination algorithm has been implemented by modifying the way AD-HOCC selects the cluster $\widehat{x} \in \widehat{D}_X$ to which each element has to be assigned. Indeed, rather than using the strategy in Equation 1, the optimal clustering $C_X^{(t)}$ is computed as $C_X^{(t)}(x) = \arg\min_{\widehat{x}\in\widehat{D}_X} \sum_{i=1}^{N} \beta_i \cdot \mathcal{D}(p_i(Y^i|x)||q_i^{(t-1)}(Y^i|\widehat{x}))$.

Fig. 3(a) shows the mutual information loss at the varying of β_1 ($\beta_2 = 1 - \beta_1$), for a dataset with 2 auxiliary domains, which was built by using a fixed size (1000) and a fixed number of clusters (2) for every domain, and by setting $\gamma = \theta = 0.2$. Note that, for each domain, the information loss produced by AD-HOCC is always ("slightly") higher than the one found by the linear-combination approach when it just considers that domain (i.e., when $\beta_1 = 1$ for Y^1, and $\beta_1 = 0$ for Y^2).

In addition, we computed the time spent against several data sets, all having a fixed size of 100 for auxiliary domains, and generated with $\gamma = 0.1$ and $\theta = 0.05$. Actually, different values have been considered for the size of the central domain (up to 2000

	$agr.$	$\beta_t = 1$	$\beta_t = 0$
lokay-m/williams-w3	0.82	0.62	0.62
kitchen-l/sanders-r	0.77	0.75	0.76

Fig. 4. Mutual information loss and micro-averaged precision on real data

values) and for the total number of domains (up to 16). Results shown in Fig. 3(b), confirm the linear dependence of the computation time on both parameters (cf. Section 3).

In order to validate the proposed approach against real data, we selected two directories, namely *lokay-m* and *williams-w3*, from the preprocessed email datasets available at *http://www.cs.umass.edu/∼ronb/enron_dataset.html*. The resulting data set consists of 275 documents organized in 11 and 18 sub-folders, respectively. Then, the document-by-term and the document-by-category matrix have been built, where categories correspond to the sub-folders. Totally, 1074 terms were selected.

Precision results obtained when partitioning the emails into two clusters are shown in Fig. 4. More specifically, we report the standard *micro-averaged precision* measures (also used, e.g., in [2]) computed by assuming that the main directories *lokay-m* and *williams-w3* represent "ground truth" classes for the emails. The table also shows results for similar experiments conducted over directories *kitchen-l* and *sanders-r*. Notice that, in both cases, AD-HOCC is able to find an accurate solution corresponding to some intermediate value of β_t. Indeed, the precision with AD-HOCC is superior to the one the linear combination achieves in both the extreme scenarios for β_t.

References

1. A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *Proc. 10th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD04)*, 509–514, 2004.
2. R. Bekkerman, R. El-Yaniv, and A. McCallum. Multi-way distributional clustering via pairwise interactions. In *Proc. 22nd Intl. COnf. on Machine Learning (ICML05)*, 41–48, 2005.
3. P. Berkhin and J. D. Becher. Learning simple relations: Theory and applications. In *Proc. 2nd SIAM Conf. on Data Mining (SDM02)*, 2002.
4. I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proc. 7th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD01)*, 269–274, 2001.
5. I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proc. 9th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD03)*, 89–98, 2003.
6. B. Gao, T.Y. Liu, X. Zheng, Q.-S. Cheng, and W.-Y. Ma. Consistent bipartite graph co-partitioning for star-structured high-order heterogeneous data co-clustering. In *Proc. 11th ACM SIGKDD Conf. on Knowledge Discovery and Data mining (KDD05))*, 41–50, 2005.
7. H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. In *Proc. 10th ACM Intl. Conf. on Information and Knowledge Management (CIKM01)*, 25–32, 2001.
8. L. Zhao and M.J. Zaki. TRICLUSTER: an effective algorithm for mining coherent clusters in 3D microarray data. In *Proc. of the 2005 ACM SIGMOD Conf. on Management of Data*, 694–705, 2005.

Efficient Inference in Large Conditional Random Fields

Trevor Cohn*

School of Informatics, University of Edinburgh, EH8 9LE, United Kingdom

Abstract. Conditional Random Fields (CRFs) are widely known to scale poorly, particularly for tasks with large numbers of states or with richly connected graphical structures. This is a consequence of inference having a time complexity which is at best quadratic in the number of states. This paper describes a novel parameterisation of the CRF which ties the majority of clique potentials, while allowing individual potentials for a subset of the labellings. This has two beneficial effects: the parameter space of the model (and thus the propensity to over-fit) is reduced, and the time complexity of training and decoding becomes sub-quadratic. On a standard natural language task, we reduce CRF training time four-fold, with no loss in accuracy. We also show how inference can be performed efficiently in richly connected graphs, in which current methods are intractable.

1 Introduction

Conditional random fields (CRFs) [1] are probabilistic models for labelling structured data. CRFs are undirected graphical models which define a conditional distribution over labellings given an observation. They allow the use of arbitrary, overlapping and non-independent features, avoiding strong independence assumptions over the observation which are typically required by generative models. CRFs have proven very successful in natural language processing [1,2,3].

However, CRFs have typically only been applied to relatively small tasks – those with small label sets, few training instances and using a simple chain. This is because inference in a linear chain CRF has a time complexity quadratic in the number of labels. The complexity is even greater for graphs with larger cliques. This serves to limit the model's scalability to large tasks and prevents the use of richly connected graphs.

This paper describes a novel mechanism that can reduce the complexity of inference. We constrain the labellings considered in each feature function, such that the functions can detect only a relatively small set of labellings. The remaining labellings can be detected only *en masse*, with the features unable to discriminate between each labelling. As such, the clique potentials share the same tied structure, containing many identical values. The sum-product and max-product algorithms can exploit the tied potentials for a significant reduction in runtime, resulting in faster training and decoding.

In many language tasks there are few truly useful labellings; these can be easily enumerated. Accordingly, tying the potentials of all remaining labellings should not noticeably reduce the modelling ability of the CRF, and may make better use of sparse data. We show how this technique can be used to reduce the training and decoding times for part-of-speech tagging, while achieving state-of-the-art results. We also demonstrate that richly connected graphs can be used tractably to perform semantic role labelling.

* Previously from the Department of CSSE, University of Melbourne, VIC 3010, Australia.

2 Conditional Random Fields

CRFs are undirected graphical models which define the conditional probability of an assignment of output labels (states) given an input observation [1]. The joint probability density function of the labelling, $\mathbf{s}$, given the input observation, $\mathbf{o}$, is given by:

$$p_{\Lambda}(\mathbf{s}|\mathbf{o}) = \frac{1}{Z(\mathbf{o})} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{s}_c, \mathbf{o}) = \frac{1}{Z(\mathbf{o})} \exp \sum_{c \in \mathcal{C}} \sum_k \lambda_k f_k(c, \mathbf{s}_c, \mathbf{o}) \tag{1}$$

where $\mathcal{C}$ is the set of cliques and ψ_c are the potential functions, which map the clique labelling and observation into a positive scalar. $Z(\mathbf{o})$ is the partition function which ensures that p is correctly normalised. We use the maximum entropy principle to define the potentials, $\psi_c(\mathbf{s}_c, \mathbf{o}) = \exp \sum_k \lambda_k f_k(c, \mathbf{s}_c, \mathbf{o})$, where λ_k are the parameters of the model, and the functions f_k are feature functions. The feature functions are usually binary valued, and combine an observational test with a labelling test. For example, a typical feature for POS tagging might detect a word suffix of "ing" coupled with a VBG label. The features typically combine observation tests with label unigrams or bigrams.

Training typically involves finding the parameters which maximise the log-likelihood of an *i.i.d.* fully observed training set, $\left(\mathbf{o}^{(i)}, \mathbf{s}^{(i)}\right)_{i=1}^{N}$, consisting of (observation, labelling) pairs. This is subject to a prior over the parameter values, $p(\Lambda)$, yielding an objective of the form:

$$\mathcal{O} = \sum_i \log p_{\Lambda}(\mathbf{s}^{(i)}|\mathbf{o}^{(i)}) + \log p(\Lambda) \ . \tag{2}$$

The parameters which maximise $\mathcal{O}$ cannot be found analytically; instead we need to employ iterative methods. Gradient descent methods have proven the most efficient [4,2]; these require repeated evaluation of objective (2) and its derivatives with respect to each parameter λ_k. The partition function and the derivatives prove costly to calculate requiring the marginal distributions over each clique. These marginals can be computed using sum-product belief propagation (BP) when the graph is acyclic, or loopy BP, an iterative approximation method [5], otherwise. This has a time complexity of $O(ES^C)$ where E are the number of edges in the graph, S is the number of labels and C is the size of the maximal clique.

Decoding uses max-product belief propagation [5] to solve $\mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{s}|\mathbf{o})$. As with the sum-product algorithm, the time complexity is $O(ES^C)$: at least quadratic in the label set.

3 Tied Potentials

We propose the use of tied potentials to simplify both the sum- and max- product algorithms, and thus training and decoding. Firstly we make the standard assumption that all the features have the form $f_k(c, \mathbf{s}_c, \mathbf{o}) = g_k(c, \mathbf{o}) h_k(\mathbf{s}_c)$, where g detects features of the observation and h detects features of the labelling. We then partition the full labelling space into a number of disjoint sets, $S^K = \bigcup_{i=1}^{N} \mathcal{S}_i$, where S is the labels set and K is the factor size. We constrain the functions h_k to detect only whether the labelling is in one of these sets. I.e. the h_k functions are limited to testing set membership rather than testing for exact label configurations. These sets are chosen *a priori*, and typically conflate sets of label configurations which are better modelled together than individually. We assume the last of these sets $\mathcal{S}_N$ is the largest, which call the *remaining labellings*. We call the labellings in the other sets the *selected labellings*.

These constrained features lead to the potentials with the form:

$$\psi_f(\mathbf{s}_f, \mathbf{o}) = \sum_i \omega_i [\![\mathbf{s}_f \in \mathcal{S}_i]\!] \qquad \text{where} \quad \omega_i = \exp \sum_{k \in \mathcal{K}_i} \lambda_k g_k(f, \mathbf{o}) \tag{3}$$

where $[\![\cdot]\!]$ is one if the test succeeds and zero otherwise and $\mathcal{K}_i$ indexes the features which detect a labelling in $\mathcal{S}_i$.

3.1 Sum-Product

Sum-product belief propagation (BP) can exploit the tied potential structure in (3). We describe its application to BP in factor graphs [5], a convenient representation when there are cliques over more than three nodes. BP requires messages to be sent over each edge in the factor graph in both directions. The marginal distributions are calculated from these messages. The messages sent from nodes to factors remain unchanged, while the messages from factors to nodes are updated to reflect the tied potentials:

$$m_{f \to n_i}(s_i) = \sum_{\mathbf{s}'_f : s'_i = s_i} \omega_N M(\mathbf{s}'_f) + \sum_{j=1}^{N-1} \sum_{\mathbf{s}'_f \in \mathcal{S}_j : s'_i = s_i} (\omega_j - \omega_N) M(\mathbf{s}'_f) \tag{4}$$

where $M(\mathbf{s}'_f) = \prod_{n_j \in \mathcal{N}(f) \setminus n_i} m_{n_j \to f}(s'_j)$ is the product of incoming messages. The above formulation introduces the default (tied) potential, ω_N, for every possible label configuration, which is offset by the potential difference, $\omega_j - \omega_N$, for every selected configuration. If we assume that messages are normalised to sum to one, we can further simplify the first term in (4) to ω_N. Here the summation over the full space of labellings is avoided; it is instead replaced by sums over only the selected labellings. This results in a time complexity for sum-product inference of $O(ET)$ where T are the number of selected labellings, regardless of the factor size.[1] The cost of loopy BP cannot be bounded in general; in this case the complexity applies to a single round. The factor marginals can be calculated from the messages using the same trick from (4) to exploit the tied potential values. This allows the efficient calculation of the normalisation constant for each marginal distribution without explicitly enumerating every labelling.

3.2 Max-Product

Max-product belief propagation is used for decoding. The most expensive message is that going from factor to node:

$$m_{f \to n_i}(s_i) = \max_{j=1 \ldots N} \omega_j \max_{\mathbf{s}'_f \in \mathcal{S}_j : s'_i = s_i} M(\mathbf{s}'_f) \tag{5}$$

[1] Larger factors will require more computation, as the number of messages in the product $M(\mathbf{s}_f)$ will increase. However, this effect is linear in the factor size, not exponential.

In order to calculate this maximum, we use a dynamic program over the set of selected labellings. We construct a trie over the selected labellings, as illustrated in Fig. 1. This trie shows three selected sets of labellings and therefore four distinct potentials – one for each selected set and one for the remaining set. The paths leading to remaining labellings are omitted from the tree, shown by dotted edges. By traversing this tree, we can maximise over both the selected labellings and the remaining labelling.

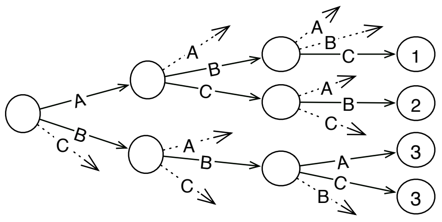


Fig. 1. An example trie with four selected labellings. The leaves are annotated with their potential index, and the dotted lines denote omitted paths leading to remaining labellings.

Maximisation uses a preorder traversal, recording the score for the current partial path. Once we reach a leaf, we have found a selected labelling; its score is the product of its potential ω_i and the leaf score. At each stage in our traversal we may traverse off the trie by following a dotted link, after which no path can yield a selected labelling. Therefore we can maximise the score for subsequent states without constraints. This maximum is simply ω_N if we ensure that messages are normalised such that the maximum message value is 1.

This approach reduces the search space down to the number of nodes in the tree (with an additional fringe). This can be further reduced by traversing the tree in a best-first manner, using A^* search to guide us towards branches of the tree in which we expect high scores. As a simple heuristic we use the product of the node score and the maximum potential value, which often takes a very small number of steps. This does not result in a reduction in the complexity bound, but in practice it considerably reduces the run-time.

4 Experiments

Two experiments were conducted on tasks which test the scalability of the CRF. The first task is POS tagging, which has a large label set and many training instances. In this task, standard CRF training using a linear chain is only just possible with modern hardware, and we show how this can be made considerably faster. The second task is semantic role labelling (SRL), in which we use richly connected cyclic graphs with many large factors. For such graphs, standard loopy belief propagation is impossible due to the large factor size. We show that feature tying can be used to make training and decoding possible for such graphs.

4.1 Part-of-Speech Tagging

The first experiment entailed tagging words with part-of-speech labels, which was modelled with a chain CRF. We used the Penn Treebank III [6], training on sections 2–21, using section 24 for development and section 23 for testing. There are 45,110 training sentences, a total of 1,023,863 tokens and 45 labels. The timing experiments used only the first 1,000 training sentences, while the performance experiments used the full set.

The observation features included word identity, prefixes and suffixes, whether the word contains a number, uppercase letter or a hyphen, nearby words at relative positions $-2, -1, +1$ and $+2$, and the word shape [7]. The word identity and default (always true) observation feature were conjoined with label pairs over pair-wise cliques, while all observation features were conjoined with single node clique labels to form the feature set. Only features seen at least once in training were included.

Timing. Three sets of selected pair-wise labellings were derived from the reduced training set: those transitions occurring at least 50 times, 10 times or once. Each selected transition labelling was modelled with a separate potential, while the remaining transitions were tied. Note that the tied transition features were used alongside standard single node features; the unigram features allow the model to disambiguate between the various tied bigrams. Table 1a shows the results when the models were trained and used for decoding both with the tied-potential optimisations (labelled 'tied') and without ('dense').

The training times for the tied optimised models are considerably lower than the dense model, despite taking more iterations to converge. The tied decoding times increase with the number of selected labellings, and are mostly competitive with the dense decoding, although the cost of constructing and traversing the trie eliminates speed gains when there are many selected labellings. The accuracies of the models trained with each transition threshold were almost identical: clearly no modelling power was sacrificed by excluding rare transitions. Moreover, using ML training, the thresholded transitions (at 10) yielded significantly better accuracy than the complete model (using the McNemar matched-pairs test at 0.1% [8]), indicating that the tying of features is smoothing the model.

Table 1. Part-of-speech tagging timing and performance results

(a) Timings using training subset

Threshold	50	10	1
#-selected	109	302	770
Train – tied	1635s	2084s	3664s
Train – dense	5144s	4326s	4222s
Decode – tied	21s	33s	51s
Decode – dense	36s	26s	24s
MLE accuracy	91.93%	92.47%	91.99%
MAP accuracy	93.09%	93.31%	93.33%

(b) Accuracy using full training set

Threshold	1000	500	100
#-selected	102	227	507
Accuracy dev.	96.86%	96.91%	96.91%
Accuracy test	97.13%	97.22%	97.24%
Iteration time	700s	1160s	2072s

Performance. The tied models were then applied to the full training set, using thresholds of 1000, 500 and 100. Lowering this threshold considerably increased the number of selected labellings and thus the run-time requirements for no tangible gain; therefore we stopped at 100. We could not train a CRF using the standard (dense) training on the full task for comparison, due to its high runtime requirements. The performance of these models on the development and test sets are shown in Table 1b. A Gaussian prior was used with a zero mean and unit variance. Once again, we see that the performance is quite similar across all thresholds (the 500 and 100 results are insignificantly different), showing that a simple frequency based selection policy is adequate. As expected, the per-iteration cost of training increases with the number of selected labellings. Our best accuracy equals that of [9], to our knowledge the best result to date, although we used a different training and test split. Unlike [9], our model achieved this result without using features over tag 3-grams and 4-grams.

4.2 Semantic Role Labelling

The second experiment demonstrates how tied-potentials can be used in order to allow efficient inference on graphs with very large factors. Semantic role labelling (SRL; [10]) is the task of identifying which groups of words act as arguments to a given verbal predicate, and what role they fill – i.e. *agent*, *patient*, etc. We adopt the CoNLL task specification [11], where a predicted syntactic parse tree is provided.

We treat the task as one of labelling each word with a role label which indicates that the word is part of a constituent with this role. The syntax tree specifies the set of syntactic constituents, and each is used as a factor, connected to the node for all words in its yield. Figure 2 shows an example sentence and some induced syntactic factors. The features for a factor were constrained to select only homogeneous labellings over its nodes. Therefore only $|S|$ labellings ($|S|$ is the number of labels) are detected instead of an exponential number in $|S|$. These features can bias the model towards choosing uniform argument labels for all nodes in an argument constituent, while still allowing overlapping constituents to compete for dominance over the labelling. In addition, we introduced adjacency factors linking adjacent constituents (e.g., linking 'I' and 'the' to join NP_1 and NP_2), representing a first order Markov assumption between adjacent argument candidates.

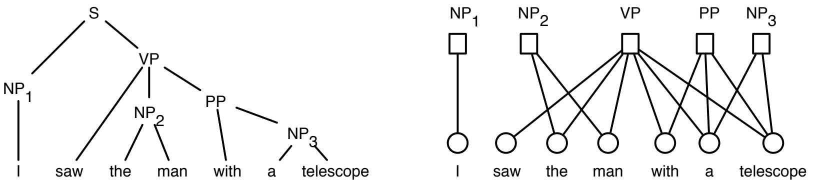


Fig. 2. Example SRL factor graph, showing the syntactic parse and an induced factor graph

The observation features were taken from [12], which included syntactic paths, head words, syntactic category, *etc*. We also used simple pruning [13] to remove irrelevant nodes from the parse tree before creating the factor graph. For further simplification, the factor graph was shrunk by merging adjacent nodes which were both members of the same set of syntactic factors, forcing them to be labelled together. Despite these simplifications, most graphs were cyclic and had large factors.

This model was trained on the proposition-bank corpus [10], using the CoNLL 2005 shared task dataset [11]. We considered only the core argument labels (A0 – A5), which roughly map to the roles of *agent, patient, theme*, *etc*. Sections 2-21 were used for training, and section 24 for development. The F_1 score on the development set was 76.56%. These results can be roughly compared to [14], who applied (constrained) CRFs to the SRL argument labelling task. They report an F_1 score of 74.49%, although on a different version of the data set.

5 Related Work

Siddiqi and Moore present a similar approach for fast inference in hidden Markov models [15]. They encode the state transition matrix by preserving the top K transitions from a given state, while sharing the remaining probability mass evenly between all other transitions from that state. They present optimised versions of the forward-backward algorithm, Viterbi and EM training, all which have considerably reduced complexity. Our work applies a similar method to log-linear undirected graphical models, showing how feature constraints can produce a similar 'mostly-constant' transition matrix (the tied potentials). While Siddiqi and Moore's model requires the specification of the number of dense outgoing transitions, K, our model instead requires explicit enumeration of these configurations. We show how this transition matrix structure can readily be exploited in general (possibly exact) belief propagation for factor graphs, as opposed to just forward-backward and Viterbi on chains.

Other approximate inference techniques have been used to reduce CRF training time, such as beam-search during sum-product or max-product inference [16], or using the voted perceptron algorithm [17]. These approximations are orthogonal to our approach, and could feasibly be used together for further gains. Our approach allows CRF inference in graphs with very large factors. For these graphs, standard training – and approximate training – is intractable.

Similar feature based optimisation has been used for training of log-linear language models [18]. For this task, the label space is extremely large (the vocabulary of words), and the optimisations presented allow efficient grouping of labels with similar feature sets. Our method also groups features, but in the context of belief propagation for globally normalised log-linear models.

6 Conclusion

We have shown how the use of tied potentials can reduce the time complexity of inference for a conditional random field. In a chain, this reduces the complexity to be sub-quadratic in the number of states, making training and decoding faster for many currently difficult tasks. Even for graphs with considerably larger cliques, inference remains tractable, allowing CRFs to be applied to previously impossible tasks. We have shown in our experiments how a small set of selected labellings can be used to reduce training time without sacrificing performance, equalling the state-of-the-art accuracy for POS tagging. We also showed how this technique allows CRFs to be used with richer more densely connected graphs for semantic role labelling.

Acknowledgements. Special thanks to Miles Osborne, Steven Bird, Phil Blunsom and Andrew Smith for their insightful feedback.

References

1. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In: Proceedings of ICML. (2001) 282–289
2. Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: Proceedings of HLT-NAACL. (2003) 213–220
3. Cohn, T., Smith, A., Osborne, M.: Scaling conditional random fields using error-correcting codes. In: Proceedings of ACL. (2005)
4. Malouf, R.: A comparison of algorithms for maximum entropy parameter estimation. In: Proceedings of CoNLL. (2002) 49–55
5. Kschischang, F., Frey, B., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Trans. Inform. Theory **47** (2001) 498–519
6. Marcus, M., Kim, G., Marcinkiewicz, M.A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., Schasberger, B.: The Penn Treebank: Annotating predicate argument structure. In: Proceedings of ARPA Human Language Technology Workshop. (1994)
7. Collins, M.: Ranking algorithms for named entity extraction: Boosting and the voted perceptron. In: Proceedings of ACL. (2002) 489–496
8. Gillick, L., Cox, S.: Some statistical issues in the comparison of speech recognition algorithms. In: Proceedings of ICASSP. (1989) 532–535
9. Toutanova, K., Klein, D., Manning, C., Singer, Y.: Feature rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of HLT-NAACL. (2003) 252–259
10. Palmer, M., Gildea, D., Kingsbury, P.: The proposition bank: An annotated corpus of semantic roles. Computational Linguistics **31**(1) (2005) 71–105
11. Carreras, X., Màrquez, L.: Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In: Proceedings of CoNLL. (2005) 152–164
12. Pradhan, S., Hacioglu, K., Krugler, V., Ward, W., Martin, J., Jurafsky, D.: Support vector learning for semantic argument classification. Machine Learning journal, Special issue on Speech and Natural Language Processing **60**(1) (2005) 11–39
13. Xue, N., Palmer, M.: Calibrating features for semantic role labeling. In: Proceedings of EMNLP. (2004) 88–94
14. Roth, D., Yih, W.: Integer linear programming inference for conditional random fields. In: Proceedings of ICML. (2005) 737–744
15. Siddiqi, S., Moore, A.: Fast inference and learning in large-state-space HMMs. In: Proceedings of ICML. (2005) 800–807
16. Pal, C., Sutton, C., McCallum, A.: Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In: Proceedings of ICASSP. (2006)
17. Roark, B., Saraclar, M., Collins, M., Johnson, M.: Discriminative language modeling with conditional random fields and the perceptron algorithm. In: Proceedings of ACL. (2004) 48–55
18. Wu, J., Khudanpur, S.: Efficient training methods for maximum entropy language modelling. In: Proceedings of the ICSLP. (2000) 114–117

A Kernel-Based Approach to Estimating Phase Shifts Between Irregularly Sampled Time Series: An Application to Gravitational Lenses

Juan C. Cuevas-Tello[1,3], Peter Tiňo[1], and Somak Raychaudhury[2]

[1] School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, United Kingdom
{J.C.Cuevas, P.Tino}@cs.bham.ac.uk

[2] School of Physics and Astronomy, University of Birmingham, Edgbaston, Birmingham B15 2TT, United Kingdom
somak@star.sr.bham.ac.uk

[3] Engineering Faculty, Autonomous University of San Luis Potosí, México

Abstract. Given two scaled, phase shifted and irregularly sampled noisy realisations of the same process, we attempt to recover the phase shift in this contribution. We suggest a kernel-based method that directly models the underlying process via a linear combination of Gaussian kernels. We apply our method to estimate the phase shift between temporal variations, in the brightness of multiple images of the same distant gravitationally lensed quasar, from irregular but simultaneous observations of all images. In a set of controlled experiments, our method outperforms other state-of-art statistical methods used in astrophysics, in particular in the presence of realistic gaps and Gaussian noise in the data. We apply the method to actual observations (at several optical frequencies) of the doubly imaged quasar Q0957+561. Our estimates at various frequencies are more consistent than those of the currently used methods.

Keywords: Kernel methods, time-series, regression.

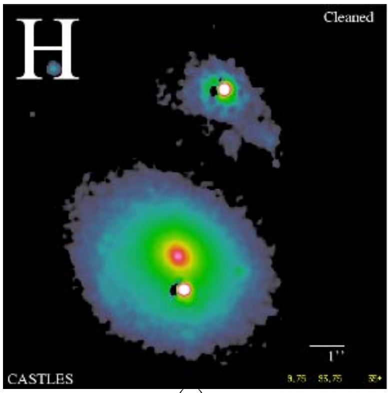
1 Introduction

According to the General theory of Relativity, a ray of light (or any other form of electromagnetic radiation, e.g. radio or x-rays) travels along a geodesic, which could be locally curved due to the gravitational effect of clumps of matter like stars or galaxies. This is known as Gravitational lensing [1] and gives rise to interesting cosmic illusions like magnified and seriously distorted images of distant sources, sometimes splitting into multiple images (e.g. Fig. 1), caused by intervening matter along the line of sight. Since the distortion of the images depends on the distribution of matter in the lensing object, this is the most direct method of measuring matter (which is often dark) in the Universe [2].

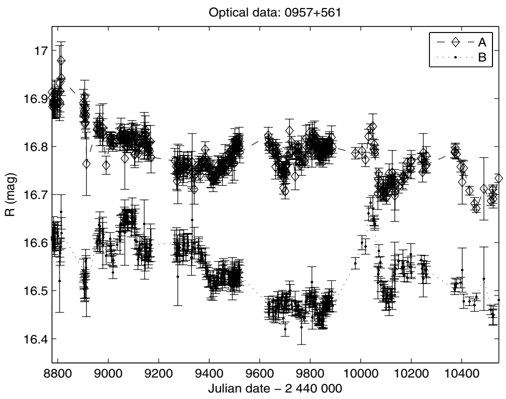
The quasar Q0957+561, an ultra-bright galaxy with a super massive central black hole (see Fig. 1), was the first lensed source to be discovered and it is the most studied so far. The source is 3.2×10^{10} light-years away from us, being lensed by a galaxy (visible in Fig. 1), along the line of sight, only 0.6×10^{10} light-years away. The effect of the lens is to create two distinct images of the same source. The brightness of quasars varies on the time scales of days- and this variation shows up at different times in the two images since the path of light travel is different for them. Since such a *time delay* (phase shift) can provide a rare direct measure of the distances involved, this quantity is of great importance in astronomy, and thus it is not surprising that many attempts have been made to estimate it, e.g. see [3,4,5,6].

The observations can be made by both radio and optical astronomers, since theory predicts that the time delay is independent of the frequency of observation. For our purposes, the data are available as two unevenly sampled time series of fluxes (or logarithm thereof) of the two images. The observations are made at irregular intervals due to weather conditions, equipment availability, object visibility, among other practical considerations.

Elsewhere, we have empirically shown, using artificial irregularly sampled time series with noise and gaps (typical of radio observations), that a kernel-based approach to measure the time delay between two such time series outperforms typical statistical methods used by astronomers [7]. In this contribution, we extend, improve and test this approach to analyse actual optical observations (and artificial data representing such data), which shows high variability compared with radio observations. We compare results with the dispersion spectra method



(a)



(b)

Fig. 1. Quasar Q0957+561. **(a)** Image taken by the Hubble Space telescope (http://www.cfa.harvard.edu/castles). The two point images are of the same distant quasar, 32 billion light-years away, multiply-imaged due to the gravitational effect of the "lensing" galaxy, seen as the extended object, which is along the line of sight, 6 billion light-years away from us. **(b)** The two time series represent the brightness of the two images (in logarithmic units (mag), such that brighter means lower values; see text) as a function time (the abscissa is measured in *days*). Image A is shifted up by 0.2 mag for visualisation purposes. This is data set DS3 with measurement error bars (std. deviations), see §2 and Table 1 for details.

[3,4], which is the most reliable of methods used by astronomers, and thus very widely used [8].

The remainder of this paper is organised as follows: §2 describes the optical data, and §3 the methods. Results of our method are compared to those from the dispersion spectra method in §4, and in §5 we show results from our analysis of optical-like artificial data, followed by comments and conclusions.

2 Astronomical Observations

In this work, we analyse the brightness of the two images of quasar Q0957+561 (Fig. 1) as a function of time, to find the phase shift between the time series. The data sets analysed here are summarised in Table 1. Optical astronomers measure the brightness of a source using imaging devices, with filters to restrict the range of wavelength/frequency of light observed. The flux f of light from a source is expressed in logarithmic units known as magnitudes (mag), defined as mag $= -2.5 \log_{10} f +$ constant. The errors on mag are mainly measurement errors, assumed to be zero-mean Gaussian. The green (g) and red (r) bands represent measurements obtained with filters in the wavelength range 400–550 nm and 550–700 nm, respectively. We use the data sets DS1 and DS2 [5], obtained through a monitoring program at the Apache Point Observatory, New Mexico, USA, and DS3, from images taken at Fred Lawrence Whipple Observatory, Mt. Hopkins, Arizona, USA. [6]. The results of this analysis are presented in §4.

Table 1. 0957+561 optical data sets analysed here

id	band	# Samples	Date	Reference
DS1	g	97	2/12/94 to 6/7/96	[5]
DS2	r	100	2/12/94 to 6/7/96	[5]
DS3	r	422	2/6/92 to 8/4/97	[6]

3 Methods for Time Delay Estimation

We model a pair of time series, obtained by monitoring the brightness (in mag units) of image A and image B, as follows

$$\begin{aligned} x_A(t_i) &= h_A(t_i) + \varepsilon_A(t_i) \\ x_B(t_i) &= h_B(t_i) + M + \varepsilon_B(t_i), \end{aligned} \tag{1}$$

where M is the offset between the two images, and $t_i, i = 1, 2, ..., n$ are discrete observation times. The observation errors $\varepsilon_A(t_i)$ and $\varepsilon_B(t_i)$ are modelled as zero-mean Normal distributions $N(0, \sigma_A(t_i))$ and $N(0, \sigma_B(t_i))$, respectively; $\sigma_A(t_i)$ and $\sigma_B(t_i)$ are given. Now,

$$h_A(t_i) = \sum_{j=1}^{N} \alpha_j K(t_j, t_i) \tag{2}$$

is the "underlying" light curve that underpins image A, whereas

$$h_B(t_i) = \sum_{j=1}^{N} \alpha_j K(t_j + \Delta, t_i) \tag{3}$$

is a time-delayed (by Δ) version of $h_A(t_i)$ underpinning image B. The Gaussian kernels $K(\cdot,\cdot)$ are centred at either t_j, $j = 1, 2, ..., N$ (function f_A), or $t_j + \Delta$, $j = 1, 2, ..., N$ (function f_B) [9,10]. We use widths $\omega_{c_j} > 0$ determining the 'degree of smoothness' of the models h_A and h_B. The widths $\omega_j \equiv \omega_{c_j}$ are determined through the k nearest neighbours of t_j as follows:

$$\omega_j = \sum_{d=1}^{k} (t_j - t_{j-d}) + (t_{j+d} - t_j) = \sum_{d=1}^{k} (t_{j+d} - t_{j-d}).$$

The value of parameter k can be estimated via cross validation.

The weights $\boldsymbol{\alpha}$ in (2-3) are given by

$$\boldsymbol{K\alpha} = \boldsymbol{x}, \tag{4}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_N)^T$,

$$\boldsymbol{K} = \begin{bmatrix} K_A(\cdot,\cdot) \\ \\ K_B(\cdot,\cdot) \end{bmatrix}, \qquad \boldsymbol{x} = \begin{bmatrix} x_A(\cdot)/\sigma_A(\cdot) \\ \\ x_B(\cdot)/\sigma_B(\cdot) \end{bmatrix}, \tag{5}$$

and the kernels $K_A(\cdot,\cdot)$, $K_B(\cdot,\cdot)$ have the form [7]:

$$K_A(t_j, t_i) = \frac{K(t_j, t_i)}{\sigma_A(t_i)}, \qquad K_B(t_j, t_i) = \frac{M + K(t_j + \Delta, t_i)}{\sigma_B(t_i)}. \tag{6}$$

Our aim is to estimate the time delay Δ between the temporal light curves corresponding to images A and B. Given the observed data and a suggested delay Δ ($[\Delta_{min}, \Delta_{max}]$), free parameters of the model (1-3) are determined within the maximum likelihood framework. Since the model is linear in parameters, we regularise $\boldsymbol{K}$ in the model fitting via singular value decomposition (SVD) [11,12].

We use model formulation (1-3), because **(1)** linearity in parameters enables us to use tools of linear algebra in parameter fitting and regularisation, **(2)** Gaussian kernel formulation using variable kernel widths is natural in cases of irregularly sampled data, **(3)** parameter sharing in (2) and (3) provides a transparent tool for coupling the two observed images.

To measure the time delay between time series, astrophysicists often use the Dispersion, which is a weighted sum of squared differences between $x_A(t_i)$ and $x_B(t_i)$. We use the D_1^2 [3] and $D_{4,2}^2$ [4] methods. The latter has a free parameter, *decorrelation length* δ, that signifies the maximum distance between observations we are willing to consider when calculating the correlations. The estimated time delay, Δ, is found by minimising D^2 over a range of time delay trials [Δ_{min}, Δ_{max}].

4 Analysis of Optical Monitoring of Gravitational Lens Q0957+561

We apply both the Dispersion method, which astrophysicists commonly use, and our method on the observational data sets, summarised in Table 1, consisting of measures of the brightness of the two images at irregular intervals.

For the time delay, we use bounds of $\Delta_{min} = 400$ and $\Delta_{max} = 450$ days given that our prior knowledge (from other analyses) is that the best delay is around 420 days [5,6]. So, we evaluate D_1^2 and $D_{4,2}^2$ in this range with increments of one day. The results are in Table 2. The decorrelation length δ in Table 2 is the same adopted by [5] and [6].

The confidence intervals were estimated through 500 Monte Carlo simulations over the observation noise processes by fixing the parameters M and δ to the best values, as in Table 2. The results are in Table 3, where μ_Δ is the mean of the time delay and the confidence intervals are given by the standard deviations (σ_Δ).

Table 2. Results on observed data

	Dispersion spectra		Kernel-based approach
Data set	D_1^2: Δ (M)	$D_{4,2}^2$: Δ ($M;\delta$)	Δ (k)
DS1	417 (0.119)	420 (0.109;7)	420 (3)
DS2	429 (0.210)	446 (0.210;7)	420 (3)
DS3	425 (0.077)	424 (0.077;4)	430 (6)

Quantities in days

Table 3. Confidence intervals: 500 Monte Carlo simulations

	Dispersion spectra		Kernel-based approach	
Data set	D_1^2 : $\mu_\Delta \pm \sigma_\Delta$	$D_{4,2}^2$: $\mu_\Delta \pm \sigma_\Delta$	$\mu_\Delta \pm \sigma_\Delta$	k
DS1	416.7±0.9	419.9±1.3	419.5±0.8	3
DS2	421.6±2.8	443.5±8.2	421.3±3.6	3
DS3	426.7±2.3	438.5±12.7	432.2±5.3	6

μ_Δ and σ_Δ are given in days

When applying our model, we have fixed M to 0.117, 0.21 and 0.076 for DS1, DS2 and DS3 respectively [5,6]. Singular values of $\boldsymbol{K}$ less than a threshold $\lambda = 0.001$ are set to zero to avoid ill-conditioning [11,12] and the smoothing parameter k was chosen through five-fold cross validation (CV) [7]. The results are in Table 2. Again, confidence intervals are estimated through 500 Monte Carlo simulations fixing M, k and λ to their optimal values (see Table 3).

5 Artificial Data

Since the true time delay on the quasar Q0957+561 is unknown, the best way to compare the performance of methods is through a set of controlled experiments where the true time delay is known. We use optical-like artificial data to compare our approach with the commonly used dispersion spectra method. In [7], we used radio-like artificial data with an imposed time delay of 500 days over an observational season of 13.6 years.

Here, the artificial data is generated as in [7], but with an observational season of 1.3 years, 50 irregular samples, a true time delay of 5 days, an offset $M = 0.1$ (considered fixed and known for both methods in §3), three levels of noise of 0.03%, 0.106% and 0.466% of mag (minimum, average and maximum of DS3, respectively), and "observational" gap size of zero to five continuously missing samples per block (five blocks randomly located). We use ten different underlying functions[1], 100 realisations per level of noise and ten realisations per gap size. This gives us an amount of 153,510 data sets under analysis. So, these data sets simulate optical data with low time delay and low offset with high precision [6].

The results are in Fig. 2, mean and standard deviation of time delay estimates are calculated for each underlying function. Then, the mean and standard deviation across all artificial data sets, μ_{Δ} and σ_{Δ}, respectively, are calculated and depicted in Fig. 2. We stress that to make our comparison fair, each method was subjected to the same collection of artificial data sets. In all cases the time delay under analysis is from 0 to 10 days; with increments of 0.1 days. The parameter λ is fixed as above.

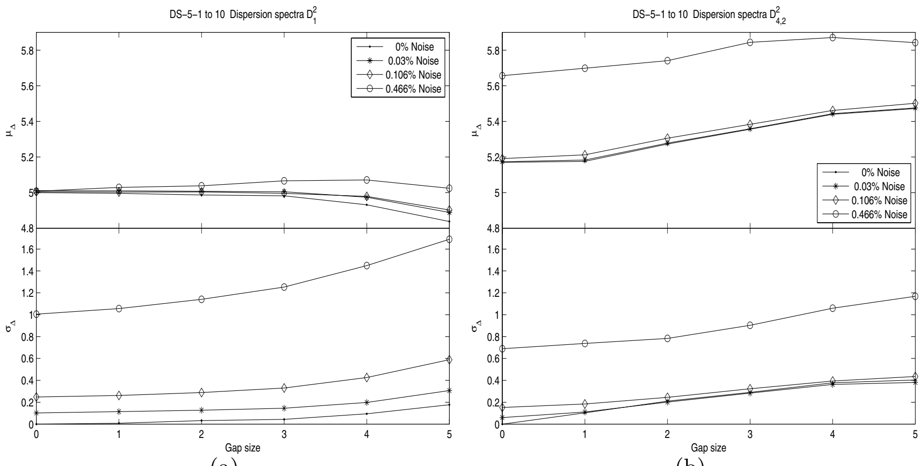
6 Comments and Conclusions

On Monte Carlo simulations, the set DS1 leads to the minimum standard deviation for both dispersion spectra methods, as well as for our kernel-based approach; see Table 3. With our methods, we get consistent results for DS1 and DS2 in Table 2, because they have almost the same sampling. On the other hand, Kundic et al. did not find such a concord with the four methods studied in [5]. Rather, they adopted the time delay of 417 ± 3 days given by *Linear method* [5]. Therefore, the best time delay for DS1 and DS2 is 420 days rather than 417 days [5]. Nevertheless, nobody knows the true time delay for the quasar Q957+561 so far, and as more observations are gathered more time delay estimates appear.

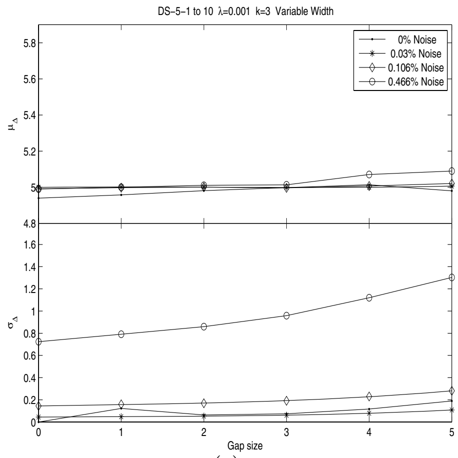
Therefore, in Fig. 2, we have a comparison of our approach against dispersion spectra on artificial data, where the true time delay is known ($\Delta = 5$ days). It appears that the D_1^2 method is less biased than the $D_{4,2}^2$ method. However, compared with $D_{4,2}^2$, the variance of D_1^2 estimates is higher. Compared with D_1^2, our method has less bias and less variance, except for cases of 0% of noise and gap size less than 3, where we observe smaller bias but higher variance. Overall, compared with our method, D_1^2 and $D_{4,2}^2$ seem more vulnerable to observational gaps.

Based on the results in Tables 2 and 3, and in Fig. 2, we conclude that our method is more accurate than dispersion spectra (see caption to Fig. 2). In the future we also plan to investigate options for speeding up parameter estimation in our kernel-based approach.

[1] plots are available at http://www.cs.bham.ac.uk/~jcc/artificial-optical/



(a) (b)



(c)

Fig. 2. Results on all artificial data, see §5 for details. **(a)** Dispersion spectra D_1^2: values of μ_Δ range in [4.83, 5.07], and for σ_Δ in [0, 1.68]. **(b)** Dispersion spectra $D_{4,2}^2$: decorrelation length δ was fixed to 5. The values of μ_Δ range in [5.17, 5.87], and for σ_Δ in [0, 1.16]. **(c)** Kernel-based approach: parameter k was fixed to 3, and the regularisation parameter λ to 0.001. Values of μ_Δ range in [4.94, 5.08], and for σ_Δ in [0, 1.30].

References

1. Saha, P.: Gravitational Lensing. Encyclopedia of Astronomy and Astrophysics (2000)
2. Kochanek, C., Schechter, P.: The Hubble constant from gravitational lens time delays. Carnegie Observatories Astrophysics Series **2** (2004)

3. Pelt, J., Kayser, R., Refsdal, S., Schramm, T.: Time delay controversy on QSO 0957+561 not yet decided. Astronomy and Astrophysics **286**(1) (1994) 775–785
4. Pelt, J., Kayser, R., Refsdal, S., Schramm, T.: The light curve and the time delay of QSO 0957+561. Astronomy and Astrophysics **305**(1) (1996) 97–106
5. Kundic, T., Turner, E., Colley, W., Gott-III, J., Rhoads, J., Wang, Y., bergeron, L., Gloria, K., Long, D., Malhorta, S., Wambsganss, J.: A robust determination of the time delay in 0957+561A,B and a measurement of the global value of Hubble's constant. Astrophysical Journal **482**(1) (1997) 75–82
6. Ovaldsen, J., Teuber, J., Schild, R., Stabell, R.: New aperture photometry of QSO 0957+561; application to time delay and microlensing. Astronomy and Astrophysics **402**(3) (2003) 891–904
7. Cuevas-Tello, J., Tiňo, P., Raychaudhury, S.: How accurate are the time delay estimates in gravitational lensing? Astronomy and Astrophysics (2006) Accepted, http://arxiv.org/abs/astro-ph/0605042.
8. Eigenbrod, A., Courbin, F., Vuissoz, C., Meylan, G., Saha, P., Dye, S.: COSMOGRAIL: The COSmological MOnitoring of GRAvItational Lenses. I. How to sample the light curves of gravitationally lensed quasars to measure accurate time delays. Astronomy and Astrophysics **436** (2005) 25–35
9. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer (2001)
10. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
11. Press, H., Flannery, B., Teukolsky, S., Vetterling, W.: Numerical Recipes. Cambridge University Press (1986)
12. Golub, G., Van-Loan, C.: Matrix Computations. second edn. The Johns Hopkins University Press (1989)

Cost-Sensitive Decision Tree Learning for Forensic Classification

Jason V. Davis, Jungwoo Ha, Christopher J. Rossbach,
Hany E. Ramadan, and Emmett Witchel

Dept. of Computer Sciences, The University of Texas at Austin

Abstract. In some learning settings, the cost of acquiring features for classification must be paid up front, before the classifier is evaluated. In this paper, we introduce the forensic classification problem and present a new algorithm for building decision trees that maximizes classification accuracy while minimizing total feature costs. By expressing the ID3 decision tree algorithm in an information theoretic context, we derive our algorithm from a well-formulated problem objective. We evaluate our algorithm across several datasets and show that, for a given level of accuracy, our algorithm builds cheaper trees than existing methods. Finally, we apply our algorithm to a real-world system, CLARIFY. CLARIFY classifies unknown or unexpected program errors by collecting statistics during program runtime which are then used for decision tree classification after an error has occurred. We demonstrate that if the classifier used by the CLARIFY system is trained with our algorithm, the computational overhead (equivalently, total feature costs) can decrease by many orders of magnitude with only a slight ($< 1\%$) reduction in classification accuracy.

1 Introduction

In the prototypical cost-sensitive classification problem of medical diagnosis, tests are performed sequentially until a diagnosis is made. Classifiers such as decision trees are natural for this problem, as predictions can be made by testing only a small subset of total features (i.e. those features present in the path from the root to the predicted leaf). In this problem, it is acceptable to have very expensive tests present in the decision tree as long as these tests are relatively unlikely to be needed in a typical evaluation of the tree.

However, in many settings, sequential testing is not feasible. In particular, if objects to be classified are transient, then they are not available for further testing when diagnosis (i.e. classifier evaluation) is performed. Consider the problem of classifying software errors: the system can be monitored during run-time, but acquiring additional "after the fact" information requires reproducing the error. Error reproduction can be time consuming and costly because oftentimes system errors are non-deterministic or environment-dependent. To efficiently classify software errors, a system must minimize runtime monitoring costs. Equivalently, the cost of the classifier—i.e. the aggregate cost of monitoring needed to construct any feature that can possibly be tested by the classifier—must be minimized.

In this paper, we present a cost-sensitive decision tree algorithm for forensic classification: the problem of classifying irreproducible events. Here, we assume that all tests (i.e. features) must be acquired before classification; consequently, the classification cost equals the sum of the costs of all features that the classifier may use for testing. We derive our algorithm by expressing the ID3 decision tree algorithm in an information theoretic context; from this, we present a cost-sensitive generalization for the information gain and gain ratio criterion. When used in conjunction with these modified cost-sensitive criteria, the resulting decision tree algorithm minimizes testing costs under the forensic classification problem while simultaneously maximizing accuracy.

For evaluation, we incorporate our cost-sensitive criterion into the C4.5 decision tree algorithm. We compare our algorithm to existing methods across various datasets from the UCI machine learning repository, and show that, for a given level of accuracy, our algorithm builds cheaper trees than existing methods. Finally, we apply our algorithm to a real-world system that classifies program errors, CLARIFY. We give an overview of CLARIFYand the various features available for classification. We propose a cost model to determine feature costs, and show that, for many programs, computational overhead can be reduced by several orders of magnitude with only a slight ($< 1\%$) decrease in classification accuracy.

2 Cost-Sensitive ID3 Decision Tree Algorithm

The ID3 algorithm builds decision trees using a top-down, greedy search procedure and represents the core of Quinlan's highly successful C4.5 decision tree algorithm. Here, we present a cost-sensitive modification to the ID3 algorithm for the forensic classification problem. For simplicity, we will outline the algorithm as a process of building a tree over a nominal feature space with arbitrarily many classes. However, all methods presented can be easily generalized to continuous attributes.

Given a decision tree with k internal nodes $1, ..., k$, each of which split on features $F^1, ..., F^k$, we will denote the tuple (X^i, y^i) to be the set of (instance, label) pairs that will 'pass through' (for internal nodes), or 'end at' (for leaf nodes) node i when the tree is evaluated. We will define $V(f)$ to be the set of values that feature f takes on, and let $(X^j_{[f=v]}, y^j_{[f=v]})$ denote the set of instances in (X^j, y^j) such that feature f takes on value v. Given some leaf node j, the ID3 algorithm splits on the feature f which maximizes the information gain,

$$Gain(X^j, f) = H(y^j) - \sum_{v \in V(f)} \frac{\left|X^j_{[f=v]}\right|}{|X^j|} H\left(y^j_{[f=v]}\right), \tag{1}$$

where $H(y) = -\sum_{\ell \in Classes} \frac{|y_{[Class=\ell]}|}{|y|} \log \frac{|y_{[Class=\ell]}|}{|y|}$, the entropy of the class labels. The information gain can be thought of as the expected decrease in entropy caused by splitting on feature f. Furthermore, if we think of the feature f and class labels y^j as random variables over the set of instances, then the information gain is equivalent to the mutual information between f and y^j, which we denote $I(y^j; f)$. Mutual information is a standard information-theoretic measure of the correlation between two random variables [4].

Since the ID3 algorithm builds the tree in a top-down manner, the split at the root node of the tree is selected using $X^1 = X$, the set of all instances used to train the tree. Recursively applying (1) in terms of $H(y)$, and re-arranging terms yields:

$$\begin{aligned}\sum_{i \in internal} \frac{|X^i|}{|X|} Gain(X^i, F^i) &= H(y) - \sum_{\ell \in leaf} \frac{|X^\ell|}{|X|} H(y^\ell) \\ &= I(y;p), \end{aligned} \tag{2}$$

where p is a random variable that gives the class values as predicted by the tree. Thus, maximizing the mutual information between the true and predicted class labels is equivalent to maximizing the weighted sum of the information gain scores at each internal node of the tree. Furthermore, the ID3 algorithm can be viewed as a greedy method to maximize this mutual information.

In an effort to reduce the cost of the features used to build the ID3 decision tree, we propose the following multi-way objective criteria that maximizes the mutual information while minimizing cost:

$$I(y;p) - \gamma \sum_{f \in F} cost(f), \tag{3}$$

where $F = \cup_{i=1}^{k} F^k$, the set of features used in the tree, *cost* is an arbitrary cost function, and $\gamma \geq 0$ is an adjustable parameter that tunes the tradeoff between maximizing mutual information and minimizing costs.

We optimize this quantity in the same top-down, greedy manner that ID3 operates by maximizing the right hand side of (2) with respect to node i. We get a new cost-sensitive information gain feature selection criteria of the form:

$$CSG(X^i, f) = \frac{|X^i|}{|X|} Gain(X^i, f) - \gamma \cdot cost(f) \mathbf{1}_{[f \notin F]}. \tag{4}$$

The indicator function $\mathbf{1}_{[f \notin F]}$ allows for the re-use of features already added to the tree without incurring additional costs. The normalization for the first term can be factored out if the cost term is not present and reduces to the basic ID3 splitting criteria (1). This normalization results in criteria that are willing to pay for more expensive features at higher levels of the tree, since a larger percentage of the distribution will 'pass through' these nodes. Nodes near the leaves of the tree will be evaluated on a relatively smaller portion of instances, and, consequently, the criteria (4) will seek cheaper features for such nodes.

Quinlan's C4.5 decision tree algorithm [13] uses a modified splitting criteria, called gain ratio, that normalizes the information gain score of splitting on feature f by the entropy of the feature f: $H(X, f) = -\sum_{v \in V(f)} \frac{|X_{[f=v]}|}{|X|} \log \frac{|X_{[f=v]}|}{|X|}$. Using a similar procedure above, this criteria also results in a global objective function, and the resulting cost-sensitive update for our model is:

$$CSGR(X^i, f) = \frac{|X^i|}{|X|} \left(\prod_{j \in Path(i)} \frac{1}{H(X^j, F^j)} \right) Gain(X^i, f) - \gamma \cdot cost(f) \mathbf{1}_{[f \notin F]}. \tag{5}$$

Whereas the $CSGain$ criteria (4) normalizes the $Gain$ term for node j by the probability of an instance arriving at node j, the above criteria normalizes by weights that are a function of both the training set distribution and the split entropies.

3 Experiments

To evaluate our method, we incorporate our cost-sensitive criteria (4) and (5) into a C4.5 decision tree. The C4.5 algorithm builds the decision tree in the same manner as ID3, but incorporates several post-processing heuristics, including a pruning method that removes statistically insignificant leaf nodes after the tree is built. We found that C4.5 yielded trees with significantly higher accuracy than ID3.

We compare our criteria to three existing methods. Nunez [12] proposes a cost-sensitive criteria called the information cost function, $\frac{2^{Gain(X,f)}-1}{(Cost(f)+1)^{\gamma}}$, which is motivated using an economic argument. Mitchell [10] proposes a criteria, $Gain(X^i, f) - \gamma \cdot cost(f)\mathbf{1}_{[f \notin F]}$, which is similar to our $CSGain$ criteria. However, this method does not normalize the $Gain$ function. Note that this criteria is a generalization of Mitchell's method that incorporates a cost/accuracy tradeoff parameter γ to the second term. Norton [11] uses a cost-sensitive criteria, $\frac{Gain(X^i,f)}{Cost(f)^{\gamma}}$, in his proposed IDX algorithm. We also generalize this algorithm to account for varying cost/accuracy tradeoffs. We note that since the Mitchell method incorporates the cost factor using an additive term, we have incorporated the cost/accuracy tradeoff parameter γ as a multiplicative factor. The Norton method incorporates costs using a multiplicative factor, so we use an exponential to adjust this tradeoff.

We present our results in terms of cost ratio, which we define as the sum of the costs of the features in the cost-sensitive decision tree, divided by the total cost of the features in the cost-insensitive tree. We compare our method against existing methods described above using eight datasets from the UCI repository [5], which are outlined in table 1.

For each dataset, we perform 50 trials of the following test. First, we randomly generate costs for each feature in the dataset from a uniform distribution on [0,1]. Second, for each of our algorithms and for each of the 3 existing algorithms, we identify the value of γ that produces the cheapest tree and that also has a 10-fold cross-validated accuracy within 1% of the baseline, cross-validated cost-insensitive C4.5 tree. We use several values of γ ranging from 10^{-6} to 10^{6}. For each algorithm, we then compute the average cost ratio across all 50 trials. Table 1 shows these average ratios for all 5 algorithms. Our cost-sensitive criteria result in significantly lower costs than that of existing algorithms.

4 CLARIFY: Forensic Classification of Confusing Software Error Behavior

In this section, we apply our cost-sensitive decision tree algorithm to a system called CLARIFY. CLARIFY's features are abstractions or *representations* of program control flow, and its classes are error behaviors that are ambiguous or misleading to a program's users. CLARIFY classifies program error behavior via monitored control flow

Table 1. Average cost ratio for our methods (CSGain and CS Gain Ratio) compared to existing methods. The cost ratio is the cost of the cost-sensitive decision tree normalized by the cost of the baseline, cost-insensitive tree. For a given level of accuracy, trees constructed with the cost-sensitive information gain and cost-sensitive gain ratio criterion tend to build much cheaper trees than existing methods.

	Dataset properties			Cost Ratios				
Dataset	# instances	# classes	# features	CSGain	CS Gain Ratio	Nunez	Mitchell	Norton
audiology	226	24	70	0.964	0.980	0.991	5.650	5.650
breast-w	699	2	10	0.647	0.671	0.917	1.106	0.970
credit-a	701	2	16	0.394	0.374	0.557	1.015	0.111
diabetes	768	2	9	0.498	0.541	0.961	0.973	1.123
hepatitis	155	2	20	0.474	0.417	0.558	1.522	0.536
liver-disorders	345	7	2	0.976	0.972	0.997	1.008	1.013
vehicle	849	4	19	0.653	0.790	0.862	0.936	1.051
zoo	107	18	7	0.524	0.507	0.606	1.045	0.542
average	-	-	-	0.641	0.657	0.806	1.657	1.375

forensics to produce more informative error messages. When a program produces an error, CLARIFY uses a classifier to predict the cause of the error from the monitored system forensics. C4.5 decision trees empirically perform very well in this domain [7].

As a testbed for the CLARIFY system, we use six different benchmarks based on the following large, mature programs: `latex` (a typesetting program), `gcc` (GNU C compiler), `mpg321` (mp3 player), `Microsoft Visual FoxPro` (a commercial database management program), `lynx` (a text-based web browser), and `apache` (a web server). For each benchmark, we identified program errors with nondescript, ambiguous, or misleading error handling. For example, such errors include `mpg321` emitting garbled audio resulting from corrupted audio file input—no message is given to the user that any problem has occurred. Benchmarks have 3 (`lynx`) to 9 (`latex`) distinct error cases with 30 (`FoxPro`) to 1,024 (`apache`) instances per error. Dimensionality is also quite high ranging from 3,600 features (`mpg321`) to approximately 100,000 features (`gcc`). For more details, see [7].

4.1 Feature Construction

CLARIFY uses behavior profiles, which are abstractions of program control flow, to monitor program behavior. This paper uses two behavior profiles: function counting (FC) and a novel method called call-tree profiling (CTP). Function counting (sometimes called function call profiling) is a simple count of the number of times each function is called during a program's execution.

Call-tree profiling is a method that captures relations between function calls. Modular software design encourages programmers to create small, simple functions with clear semantics, making function boundaries important. Moreover, the order of function calls and their relationship is a rich source of program behavior data. The dynamic function calling behavior of a program can be represented by a dynamic call tree, where each node is a dynamic instance of a function call, and edges are calls between functions.

Call-tree profiling associates a counter with a depth-bounded subtree rooted at a particular function, and increments the counter when the subtree is executed. Each subtree is a feature and the feature value is the counter value associated to the subtree. In this paper, CTP will refer to the union of the feature spaces at depth bound of at most two (i.e., CTP-D0, CTP-D1, and CTP-D2). Note that CTP-D0 is equivalent to FC.

4.2 Minimizing Overhead Costs

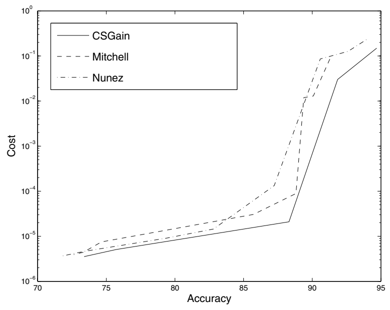
The instrumentation inserted into applications to produce a behavior profile for the CLARIFY decision tree classifier can have significant computational costs. If CLARIFY monitored all features it could monitor, the computational overhead of the system would be high. One way to reduce CLARIFY's computational overhead is to instrument only those features tested in the decision tree. Cost-sensitive learning reduces the amount of required instrumentation even further. Since program instrumentation points must be chosen before the program is executed (i.e. not during prediction), the CLARIFY classification problem is a forensic problem and is thus well-suited for our algorithm. Feature costs vary greatly in this problem domain: features corresponding to frequently executed functions incur overheads many times larger than features corresponding to rarely called functions.

For function counting, instrumentation points are needed only at functions that correspond to nodes in the decision tree. To record function counts, an array of counters is used to track execution for each instrumented function. Let G be the set of monitored functions, and let $E[g]$ be the expected number of times a function g is called in a program's execution. Note that these expectations can be computed from the training set. Then $\sum_{g \in G} E[g]$ gives the expected number of instrumented events per program execution, and will be proportional to overhead cost.

In call-tree profiling (CTP), instrumentation code at the start of each function records function call subtrees. Hence, the cost model accounts for the execution of all functions that appear within any CTP feature. Given a set of CTP subtrees over a set of functions F, we approximate the overhead cost of instrumenting these subtrees as $\sum_{f \in F} E[f]$. Once a function is part of a CTP feature, including it in a different CTP feature does not add significant overhead. Therefore, the cost of each feature must be computed in the context of the features that have already been added to the tree at an earlier stage of the algorithm.

4.3 Results

Figure 1 (left) shows the cost/accuracy tradeoff for the `gcc` benchmark. As a baseline, the cost of the trees built using the two best existing methods (as quantified in section 3) are also plotted. This curve is generated using five-fold cross validation to estimate the classification accuracy of the cost-sensitive decision tree for various values of γ. Among this set of (cost, accuracy) pairs, pareto optimal points are identified to generate the cost/accuracy curve. Since the absolute overhead slowdown is a function of program running time (which varies greatly from benchmark to benchmark), the costs here are normalized by the total instrumentation slowdown incurred if all available features were



Benchmark	Baseline	CSGain w/FC, CTP
mpg321	19.4%	158.3×
gcc	24.2%	1.8×
gzprintf	20.1%	1.7×
latex	44.0%	468.1×
foxpro	3.7%	1,485,943.7×
lynx	1.9%	552.3×
apache	8.9%	4,684.2×

Fig. 1. Left: cost/accuracy tradeoff for the gcc benchmark. Right: costs for six benchmarks with accuracy reductions of at most 1%. The Baseline column gives the decision tree cost when built with the baseline C4.5 algorithm, using CTP, expressed as a percentage of the total cost of instrumenting all features. The remaining columns provide the speedup ratio (defined as baseline cost / cost) for C4.5 using the cost-sensitive gain criteria (CSGain) with FC and CTP features.

instrumented. For example, a cost of .1 corresponds to instrumenting an average of 10% of all function calls in an execution of a program.

Table 1 (right) gives decision tree costs for several benchmarks when trained using the baseline, cost-insensitive C4.5 algorithm (using FC and CTP behavior profiles), and also when trained using C4.5 with the CSGain criteria (4). This improvement is measured as the cost of the tree divided by the cost of the baseline, cost-insensitive tree (note that this is the inverse of the cost ratio term used in section 3). For the cost-sensitive algorithms, results are given for trees with accuracy levels that are no less than 1% lower than the cross validated accuracy of the baseline cost-insensitive classifier trained with FC and CTP representation. Our cost-sensitive algorithm yields reduction in execution of instrumentation points of up to six orders of magnitude.

5 Related Work

Building classifiers that minimize testing costs has received much attention in the field of medical diagnosis. However, the problem of medical diagnosis is fundamentally different from the forensic classification problem. Several cost-sensitive algorithms have been proposed that build decision trees using non-incremental methods, such as a genetic algorithm [14] and a "look ahead" heuristic [11]. These methods are not considered here, as the training time required is several orders of magnitude larger than a C4.5 based incremental algorithm.

In this paper, we have focused on the problem of minimizing test cost while maximizing accuracy. In some settings, it is more appropriate to minimize misclassification costs instead of maximizing accuracy. For the two class problem, Elkan [6] gives a method to minimize misclassification costs given classification probability estimates. Bradford et al. compare pruning algorithms to minimize misclassification costs [1]. As both of these methods act independently of the decision tree growing process, they can be incorporated with our algorithms (although we leave this as future work). Ling et. al. propose a cost-sensitive decision tree algorithm that optimizes both accuracy and cost. However, the cost insensitive version of their algorithm (i.e. the algorithm run if all feature costs are zero), reduces to a splitting criteria that maximizes accuracy, which is well known to be inferior to the information gain and gain ratio criterion [13,10].

Integrating machine learning with program understanding is an active area of current research. Systems that analyze root cause errors in distributed systems [3] and systems that find bugs using dynamic predicates [2,8,9] may both benefit from cost-sensitive learning to decrease overhead monitoring costs.

6 Conclusion

We have introduced two algorithms for the problem of minimizing feature costs for forensic classification. Our algorithms are modifications to the C4.5 decision tree algorithm that use a well motivated cost-sensitive splitting criteria. We provide extensive experiments on real data and objectively demonstrate that our criterion yield algorithms that build cheaper trees than existing methods. Finally, we implement our method in a novel cost-sensitive forensic classification problem, the CLARIFY system. We show our algorithm can reduce overhead costs by many orders of magnitude at only a slight $(< 1\%)$ reduction in classification accuracy.

References

1. J. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C. Brodley. Pruning decision trees with misclassification costs. In *European Conference on Machine Learning*, 1998.
2. Y. Brun and M. D. Ernst. Finding latent code errors via machine learning over program executions. In *ICSE*, 2004.
3. I. Cohen, S. Zhang, M. Goldszmidt, J. Symons, T. Kelly, and A. Fox. Capturing, indexing, clustering, and retrieving system history. In *SOSP*, 2005.
4. T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley Series in Telecommunications, 1991.
5. C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.
6. C. Elkan. The foundations of cost-sensitive learning. In *International joint conference on artifical intelligence*, 2001.
7. J. Ha, H. Ramadan, J. Davis, C. Rossbach, I. Roy, and E. Witchel. Navel: Automating software support by classifying program behavior. Technical Report TR-06-11, University of Texas at Austin, 2006.
8. S. Hangal and M. S. Lam. Tracking down software bugs using automatic anomaly detection. In *ICSE*, 2002.
9. B. Liblit, M. Naik, A. X. Zheng, A. Aiken, and M. I. Jordan. Scalable statistical bug isolation. In *PLDI*, 2005.
10. T. Mitchell. *Machine Learning*. WCB McGraw-Hill, 1997.
11. S.W. Norton. Generating better decision trees. In *International joint conference on artifical intelligence*, 1989.
12. M. Nunez. The use of background knowledge in decision tree induction. In *Machine Learning*, 1991.
13. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers, 1992.
14. P. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. In *Journal of artificial intelligence research*, 1995.

The Minimum Volume Covering Ellipsoid Estimation in Kernel-Defined Feature Spaces

Alexander N. Dolia[1], Tijl De Bie[2], Chris J. Harris[1], John Shawe-Taylor[1], and D.M. Titterington[3]

[1] University of Southampton, Southamptom, SO17 1BJ, UK
[2] OKP Research group, Katholieke Universiteit Leuven, Belgium
[3] Department of Statistics,University of Glasgow

Abstract. Minimum volume covering ellipsoid estimation is important in areas such as systems identification, control, video tracking, sensor management, and novelty detection. It is well known that finding the minimum volume covering ellipsoid (MVCE) reduces to a convex optimisation problem. We propose a regularised version of the MVCE problem, and derive its dual formulation. This makes it possible to apply the MVCE problem in kernel-defined feature spaces. The solution is generally sparse, in the sense that the solution depends on a limited set of points. We argue that the MVCE is a valuable alternative to the minimum volume enclosing hypersphere for novelty detection. It is clearly a less conservative method. Besides this, we can show using statistical learning theory that the probability of a typical point being misidentified as a novelty is generally small. We illustrate our results on real data.

1 Introduction

The minimum volume covering ellipsoid (MVCE) [2,3,10,12], the ellipsoid smallest in volume that covers all of a given set of points, has many applications in areas ranging from systems and control to robust statistics. In this paper we focus on its application to novelty detection (also known as support estimation or domain description): then, all data points from a training set $\{\mathbf{x}_i\}_{i=1}^{\ell}$ are specified to be sampled from an unknown distribution $\mathcal{D}$, and the support of $\mathcal{D}$ is be estimated as the inside region of the MVCE. Points lying outside of the ellipsoid can then subsequently be judged to be *novelties*.

Recently, several results in the machine learning domain have attacked this problem by means of the minimum volume covering hypersphere (MVCS) [7,8,11], fitting a tight hypersphere around the data. A hypersphere being a special type of ellipsoid, the volume of the MVCE will never be larger than the volume of the MVCS. The motivation for the current work is that the additional flexibility in using an ellipsoid is likely to be more sensitive in identifying novelties.

However, specificity problems should be expected for high-dimensional spaces. Indeed, the MVCE becomes vanishingly small for data sets smaller in size than their dimension, and the method would reject (nearly) all test points from $\mathcal{D}$ as outliers, judged not to belong to the support of the distribution. To overcome this problem, we propose a regularised MVCE (RMVCE) method. This allows us to derive the main result of this paper, which is the RMVCE problem in a (possibly infinite-dimensionaly) kernel-defined feature space.

Additionally, we present an in depth statistical analysis of the novelty detection method that is based on the RMVCE problem, and an extension of the RMVCE problem and its kernel version towards a soft-margin variant.

2 The Minimum Volume Ellipsoid

Assume that we have a training dataset containing ℓ samples, $\{\mathbf{x}_i \in \Re^{k\times 1}\}_{i=1}^{\ell}$. The MVCE is specified by the positive definite matrix $\mathbf{M} \in \Re^{k\times k}$ that solves the optimization problem (for conciseness, in this paper we assume the ellipsoid is centred at the origin—extending to a variable centre is trivial [12]):

$$\begin{aligned} &\min_{\mathbf{M},\mu} \log\det\mathbf{M} + \mu, \\ &\text{s.t.} \quad \mathbf{x}_i'\mathbf{M}^{-1}\mathbf{x}_i \leq \mu, \text{ for all } i. \end{aligned} \tag{1}$$

The objective consists of two terms: the logarithm of the volume of the ellipsoid, and the maximal *Mahalanobis distance* $\mathbf{x}_i'\mathbf{M}^{-1}\mathbf{x}_i$ over all data points $\mathbf{x}_i$. This objective as well as the constraints which constrain the data points to be within a Mahalanobis distance μ from the centre of the ellipsoid are both convex in $\mathbf{M}^{-1}$ and μ. Therefore, the optimization problem has a unique optimal solution.

The dual of optimisation problem (1) can be written as [12]:

$$\begin{aligned} &\max_{\boldsymbol{\alpha},\mathbf{M}} \log\det\left(\mathbf{M}\right), \\ &\text{s.t.} \quad \boldsymbol{\alpha} \geq \mathbf{0}, \quad \boldsymbol{\alpha}'\mathbf{e} = 1, \\ &\qquad \mathbf{M} = \textstyle\sum_{i=1}^{\ell} \alpha_i \mathbf{x}_i \mathbf{x}_i'. \end{aligned} \tag{2}$$

from which the variable $\mathbf{M}$ can directly be eliminated to yield an optimisation problem in $\boldsymbol{\alpha}$ only. In the following section we propose a regularised version of the MVCE problem.

3 Regularised Minimum Volume Covering Ellipsoid

As explained in the introduction, we should prevent the ellipsoid to collapse to zero volume in large dimensional spaces. This can be achieved by changing the constraint $\mathbf{M} = \sum_i \alpha_i \mathbf{x}_i \mathbf{x}_i'$ in (2) into $\mathbf{M} = \sum_i \alpha_i \mathbf{x}_i \mathbf{x}_i' + \gamma\mathbf{I}$, which guarantees a minimal diameter of the ellipsoid in all directions. This gives:

$$\begin{aligned} &\max_{\boldsymbol{\alpha},\mathbf{M}} \log\det(\mathbf{M}), \\ &\text{s.t.} \quad \boldsymbol{\alpha} \geq 0, \quad \boldsymbol{\alpha}'\mathbf{e} = 1, \\ &\qquad \mathbf{M} = \textstyle\sum_i \alpha_i \mathbf{x}_i \mathbf{x}_i' + \gamma\mathbf{I}. \end{aligned} \tag{3}$$

The dual of this optimisation problem is given by (without derivation due to space restrictions):

$$\begin{aligned} &\min_{\mathbf{M},\mu} \log\det(\mathbf{M}) + \mu + \gamma\text{trace}(\mathbf{M}^{-1}), \\ &\text{s.t.} \quad \mathbf{x}_i'\mathbf{M}^{-1}\mathbf{x}_i \leq \mu, \text{ for all } i. \end{aligned} \tag{4}$$

For $\gamma = 0$, this is equal to the standard MVCE centred at the origin formulation as discussed in the previous section. Different from the standard formulation is the additional *regularization term* $\gamma\text{trace}(\mathbf{M}^{-1})$. This term ensures that the ellipsoids axes are never extremely small. Indeed, a small diameter in one dimension would result in a small eigenvalue of $\mathbf{M}$, which in turn leads to a large trace of $\mathbf{M}^{-1}$. As we can learn from $\mathbf{M} = \sum_i \alpha_i \mathbf{x}_i \mathbf{x}_i' + \gamma\mathbf{I}$ in (3), the effect is that the diameter along each of the dimensions is at least equal to γ.

Soft margin RMVCE formulation. In the presence of outliers it can be appropriate to introduce slack variables ξ_i and add a corresponding penalty term to the objective:

$$\begin{aligned} &\min_{\mathbf{M},\mu} \log\det(\mathbf{M}) + \mu + \gamma\text{trace}(\mathbf{M}^{-1}) + \frac{1}{\nu l}\textstyle\sum_{i=1}^{l}\xi_i, \\ &\text{s.t.} \quad \mathbf{x}_i'\mathbf{M}^{-1}\mathbf{x}_i \leq \mu + \xi_i, \text{ for all } i, \;\; \boldsymbol{\xi} \geq 0. \end{aligned} \tag{5}$$

where $\nu \in (0.1]$. The dual problem can be written as follows:

$$\begin{aligned} \boldsymbol{\alpha}_\gamma^* &= \text{argmin}_{\boldsymbol{\alpha}} - \log\det\left(\sum_i \alpha_i \mathbf{x}_i \mathbf{x}_i' + \gamma\mathbf{I}\right), \\ &\text{s.t.} \quad \mathbf{e} \geq \nu\ell\boldsymbol{\alpha} \geq 0, \;\; \boldsymbol{\alpha}'\mathbf{e} = 1. \end{aligned}$$

4 Kernel Regularised Minimum Volume Covering Ellipsoid

Let us first define the diagonal matrix $\mathbf{A}$, with $\mathbf{A}_{ii} = a_i = \sqrt{\alpha_i} \geq 0$, such that (with $\mathbf{a} = (a_1 \; a_2 \; \cdots \; a_\ell)'$) from $\mathbf{e}'\boldsymbol{\alpha} = 1$ we have that $\mathbf{a}'\mathbf{a} = 1$. Then we can write $\sum_i \alpha_i \mathbf{x}_i \mathbf{x}_i' + \gamma\mathbf{I} = \mathbf{X}'\mathbf{A}^2\mathbf{X} + \gamma\mathbf{I}$. Note that the matrices $(\mathbf{AX})'(\mathbf{AX}) = \mathbf{X}'\mathbf{A}^2\mathbf{X}$ and $(\mathbf{AX})(\mathbf{AX})' = \mathbf{AXX}'\mathbf{A} = \mathbf{AKA}$ have the same nonzero eigenvalues λ_i, equal to the squares of the singular values of $\mathbf{AX}$ [2]. With d the dimensionality of the space and ℓ the number of data points $\mathbf{x}_i$, it is now easy to show that:

$$\log\det(\mathbf{AKA} + \gamma\mathbf{I}) = \log\det\left(\mathbf{XA}^2\mathbf{X} + \gamma\mathbf{I}\right) + (\ell - d)\log(\gamma).$$

Hence we can optimize $\log\det(\mathbf{AKA} + \gamma\mathbf{I})$ instead of $\log\det\left(\mathbf{XA}^2\mathbf{X} + \gamma\mathbf{I}\right)$. Now define $\mathbf{C}$ to be a Cholesky factor of $\mathbf{K}$ (i.e. $\mathbf{K} = \mathbf{CC}'$). Then, $\mathbf{AKA} = \mathbf{ACC}'\mathbf{A}$ and $\mathbf{C}'\mathbf{A}^2\mathbf{C} = \sum_{i=1}^{l} \alpha_i \mathbf{c}_i \mathbf{c}_i'$ with $\mathbf{c}_i$ the ith row of $\mathbf{C}$ have the same eigenvalues, such that $\log\det(\mathbf{AKA} + \gamma\mathbf{I}) = \log\det\left(\sum_{i=1}^{l} \alpha_i \mathbf{c}_i \mathbf{c}_i' + \gamma\mathbf{I}\right)$. Hence, we obtain the kernel version of the regularized MVCE:

$$\begin{aligned} \boldsymbol{\alpha}_\gamma^* &= \text{argmin}_{\boldsymbol{\alpha}} - \log\det\left(\textstyle\sum_{i=1}^{l}\alpha_i \mathbf{c}_i \mathbf{c}_i' + \gamma\mathbf{I}\right), \\ &\text{s.t.} \quad \mathbf{e} \geq \nu\ell\boldsymbol{\alpha} \geq 0, \;\; \boldsymbol{\alpha}'\mathbf{e} = 1. \end{aligned} \tag{6}$$

Computing the Mahalanobis distance for a test point. We should be able to compute the Mahalanobis distance for a test point exclusively using kernel evaluations and the vector $\boldsymbol{\alpha}$. Recall the eigenvalue decompositions of $\sum_i \alpha_i \mathbf{x}_i \mathbf{x}_i' = \mathbf{X}'\mathbf{A}^2\mathbf{X} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}'$ and $\mathbf{AXX'A} = \mathbf{AKA} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}'$ [2]. We then have that $\sum_i \alpha_i \mathbf{x}_i \mathbf{x}_i' + \gamma \mathbf{I} = \mathbf{U}(\boldsymbol{\Lambda} + \gamma\mathbf{I})\mathbf{U}' + \mathbf{U}^\perp(\gamma\mathbf{I})\mathbf{U}^{\perp\prime}$ (where $\mathbf{U}^\perp$ is an orthonormal basis for the space orthogonal to the column space of $\mathbf{U}$). Thus we can write the Mahalanobis distance as (and we introduce the notation $d_\gamma(\cdot, \boldsymbol{\alpha})$):

$$\begin{aligned}
d_\gamma(\mathbf{x}, \boldsymbol{\alpha}) &\triangleq \mathbf{x}'\mathbf{M}^{-1}\mathbf{x} = \mathbf{x}'(\textstyle\sum_i \alpha_i \mathbf{x}_i \mathbf{x}_i' + \gamma\mathbf{I})^{-1}\mathbf{x} \\
&= \mathbf{x}'\left(\mathbf{U}(\boldsymbol{\Lambda} + \gamma\mathbf{I})^{-1}\mathbf{U}' + \mathbf{U}^\perp(\gamma\mathbf{I})^{-1}\mathbf{U}^{\perp\prime}\right)\mathbf{x} \\
&= \frac{1}{\gamma}\mathbf{x}'\mathbf{x} + \mathbf{x}'\mathbf{U}\left((\boldsymbol{\Lambda} + \gamma\mathbf{I})^{-1} - (\gamma\mathbf{I})^{-1}\right)\mathbf{U}'\mathbf{x} \\
&= \frac{1}{\gamma}k(\mathbf{x}, \mathbf{x}) - \frac{1}{\gamma}\mathbf{x}'\mathbf{U}\left(\boldsymbol{\Lambda}(\boldsymbol{\Lambda} + \gamma\mathbf{I})^{-1}\right)\mathbf{U}'\mathbf{x}, \\
&= \frac{1}{\gamma}\left(k(\mathbf{x}, \mathbf{x}) - \mathbf{k}'\mathbf{AV}\boldsymbol{\Lambda}(\boldsymbol{\Lambda} + \gamma\mathbf{I})^{-1}\mathbf{V}'\mathbf{Ak}\right),
\end{aligned}$$

using $\mathbf{U} = \mathbf{X}'\mathbf{AV}\boldsymbol{\Lambda}^{-\frac{1}{2}}$ and $\mathbf{Xx} = \mathbf{k}$. This is expressed entirely in terms of kernels, since $\mathbf{V}$ and $\boldsymbol{\Lambda}$ can be found using the eigenvalue decomposition of $\mathbf{AKA}$.

5 Statistical Learning Analysis

Theoretically we can view the novelty detection problem in a space $\mathcal{X}$ as the task of finding a set $A \subset \mathcal{X}$ such that most of the support $\text{supp}(\mathcal{D})$ of the distribution $\mathcal{D}$ generating the data is contained in A; that is

$$P_{\mathbf{x}\sim\mathcal{D}}(\mathbf{x} \in \text{supp}(\mathcal{D}) \setminus A) \leq \epsilon, \tag{7}$$

for some small ϵ. This must be achieved while keeping the volume of A as small as possible, where in general the volume could be measured according to some prior distribution though in our case we consider the input space volume.

The motivation for this definition is to 'shrink wrap' the support of the training distribution as tightly as possible to increase the likelihood of detecting novel outliers. The bound of equation (7) upper bounds the probability that a point detected as an outlier (or novelty) is actually generated according to the original training distribution.

Earlier analyses of this type are based on covering number arguments [7] or Rademacher complexities [8], and deal with the case where the set A can be viewed as a hypersphere. However, it seems unnatural to use a spherical shape if the variance of the data varies significantly across different dimensions of the space. One would expect that we can use an elliptical shape with smaller diameters in the dimensions of low variance. The algorithm described in this paper implements just such a shape for the set A through the use of the Mahalanobis distance relative to the matrix $\mathbf{M}$. Introducing such flexibility into the shape of the set A raises the question of whether the algorithm may not be overfitting the data and jeopardizing the confidence with which equation (7) can be asserted. This section will confirm that this concern is unfounded: that is we will prove a bound of the type given in equation (7) that holds with high confidence over the random selection of training sets according to the underlying distribution.

We first observe that the Mahalanobis distance $d_\gamma(\mathbf{x}, \alpha)$ can be viewed as a linear function in the space defined by the kernel $k(\mathbf{x}, \mathbf{z})^2$ where $k(\mathbf{x}, \mathbf{z})$ is the kernel defining the feature space. This follows from the observation that

$$d_\gamma(\mathbf{x}, \alpha) = \mathrm{trace}(\mathbf{M}^{-1}\mathbf{x}\mathbf{x}') = \left\langle \mathbf{M}^{-1}, \mathbf{x}\mathbf{x}' \right\rangle_F ,$$
$$\text{while: } \langle \mathbf{x}\mathbf{x}', \mathbf{z}\mathbf{z}' \rangle = \langle \mathbf{x}, \mathbf{z} \rangle^2 = k(\mathbf{x}, \mathbf{z})^2.$$

Therefore, the critical quantity in analysing the generalization would appear to be the norm of the matrix $\mathbf{M}^{-1}$. Unfortunately this scales with the dimension of the space and so a naive application of standard Rademacher bounds would lead to a bound unsuitable for kernel defined feature spaces.

We will present a bound that uses the PAC-Bayes approach to generalization analysis in order to overcome this difficulty. As far as we are aware this is the first application of this technique to novelty detection.

The general PAC-Bayes theorem assumes a pre-specified ‘prior’ distribution $P(c)$ over the class of classifiers. The learning algorithm returns a distribution $Q(c)$ over the class and classification of an example $\mathbf{x}$ is performed by drawing a classifier c randomly according to $c \sim Q$ and using it to return the label $c(\mathbf{x})$. We denote by $Q_{\mathcal{D}}$ the misclassification probability of Q on an example drawn according to $\mathcal{D}$. For a training set S of n examples, we denote by $\hat{Q}_S$ the empirical misclassification error of Q. We will describe later how such a bound can be applied to the deterministic outlier detector that we consider. We use KL to denote the Kullback-Leibler divergence between two distributions:

$$\mathrm{KL}(Q\|P) = E_{c\sim Q} \ln \frac{Q(c)}{P(c)}.$$

For $p \in [0, 1]$ we overload the notation by using p to represent the binary distribution $\{p, 1 - p\}$. We can now state the theorem in a form due to Langford.

Theorem 1. *[5] For all $\mathcal{D}$, for all priors $P(c)$, and for all $\delta \in (0, 1)$,*

$$P_{S\sim\mathcal{D}^n}\left(\forall Q(c) : \mathrm{KL}(\hat{Q}_S\|Q_{\mathcal{D}}) \leq \frac{\mathrm{KL}(Q\|P) + \ln\frac{n+1}{\delta}}{n}\right) \geq 1 - \delta.$$

Our application of the theorem to the novelty detector will follow closely the application to support vector machines as described in [6] and [5]. This involves choosing P to be a symmetric Gaussian prior of variance 1 but rather than being centered on the origin as in those papers, we choose the prior distribution to be centered at the point $(\mu\gamma^{-1}I, 0)$ for some $\mu > 0$. Note that we are viewing the space as a Euclidean space with the Frobenius inner product with one extra dimension for the threshold. We augment the examples by adding a coordinate equal to -1 in this extra dimension. The posterior distribution $Q(\mu)$

is now a spherically symmetric Gaussian with variance 1 centered at the point $(\mu \mathbf{M}^{-1}, \mu\theta)$, and θ is a threshold such that a novelty is indicated if

$$d_\gamma(\mathbf{x}, \boldsymbol{\alpha}_\gamma^\star) \geq \theta. \tag{8}$$

Clearly, equation (8) can be written as a linear function thresholded at 0 with weight vector $(\mathbf{M}^{-1}, \theta)$. If equation (8) holds for $\mathbf{x}$ then $Q(\mu)$ has probability at least 0.5 of being 1, hence

$$P(d_\gamma(\mathbf{x}, \boldsymbol{\alpha}_\gamma^\star) \geq \theta) \leq 2Q(\mu)_{\mathcal{D}}.$$

It will therefore suffice to obtain an implicit bound on $Q(\mu)_{\mathcal{D}}$ using Theorem 1.

We describe the critical quantities required in the theorem. Following [5] we require the function

$$\tilde{F}(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$

We denote the weight vector $\mathbf{W} = (\mu \mathbf{M}^{-1}, \mu\theta)$. The normalized margin of an example $\mathbf{x}$ is given by

$$g(\mathbf{x}) = \frac{d_\gamma(\mathbf{x}, \alpha_\gamma^\star) - \theta}{\sqrt{\|\mathbf{x}\|^2 + 1}\|\mathbf{W}\|}.$$

The stochastic error rate is then

$$\hat{Q}(\mu)_S = E_{\mathbf{x}\sim S} \tilde{F}(\mu \|\mathbf{W}\| g(\mathbf{x})).$$

Finally, the KL-divergence between prior and posterior is given by

$$\mathrm{KL}(Q\|P) = \frac{\mu^2}{2}(\|\gamma^{-1}\mathbf{I} - \mathbf{M}^{-1}\|^2 + \theta^2) = \frac{\mu^2}{2}\left(\sum_{i=1}^n \frac{\lambda_i^2}{\gamma^2(\lambda_i + \gamma)^2} + \theta^2\right)$$

which critically is independent of the dimension of the feature space.

Putting the pieces together we obtain the following bound on the probability of misidentifying an outlier.

Theorem 2. *Fix $\gamma > 0$ and $\mu > 0$. For all distributions $\mathcal{D}$ and all $\delta \in (0, 1)$, we have with probability at least $1 - \delta$ over the draw of an n-sample S, if $\boldsymbol{\alpha}_\gamma^\star$ is the solution of the novelty detection optimization then*

$$P_{\mathbf{x}\sim\mathcal{D}}(d_\gamma(\mathbf{x}, \boldsymbol{\alpha}_\gamma^\star) \geq \theta) \leq 2Q(\mu)_{\mathcal{D}} \tag{9}$$

where $Q(\mu)_{\mathcal{D}}$ satisfies

$$\mathrm{KL}(\hat{Q}(\mu)_S \| Q(\mu)_{\mathcal{D}}) \leq \frac{\frac{\mu^2}{2}\left(\sum_{i=1}^n \frac{\lambda_i^2}{\gamma^2(\lambda_i+\gamma)^2} + \theta^2\right) + \ln\frac{n+1}{\delta}}{n},$$

$$\text{and: } \quad \hat{Q}(\mu)_S = E_{\mathbf{x}\sim S}\tilde{F}\left(\mu \frac{d_\gamma(\mathbf{x}, \boldsymbol{\alpha}_\gamma^\star) - \theta}{\sqrt{\|\mathbf{x}\|^2 + 1}}\right).$$

Note that in practice one would apply the theorem for a number of different values of μ and possibly different regularization parameter choices. If N applications are made then we should substitute δ/N for δ in the expression for the KL-divergence, but this only enters into the ln term and so has a limited effect.

Proof. The only unresolved part of the proof is the verification of the expression for the stochastic error. We decompose the example $(\mathbf{x}\mathbf{x}', -1)$ into two components $\mathbf{X}_{\|}$ parallel to $\mathbf{W}$ and $\mathbf{X}_{\perp}$ perpendicular. The randomly drawn weight vector can be decomposed into three components $\mathbf{U}_{\|}$ parallel to $\mathbf{W}$ and distributed according to $N(\mu\|\mathbf{W}\|, 1)$, $\mathbf{U}_{\perp}$ parallel to $\mathbf{X}_{\perp}$ distributed according to $N(0, 1)$ and $\mathbf{W}_{\perp\perp}$. Let $w = \|\mathbf{W}\|$, $u_{\|} = \|U_{\|}\|$, $u_{\perp} = \|U_{\perp}\|$, $x_{\|} = \|\mathbf{X}_{\|}\|$, and $x_{\perp} = \|\mathbf{X}_{\perp}\|$. Then we have, as required:

$$\begin{aligned}
\hat{Q}(\mu)_S &= E_{\mathbf{x}\sim S, u_{\|}\sim N(\mu w,1), u_{\perp}\sim N(0,1)} I(u_{\|}x_{\|} + u_{\perp}x_{\perp} \geq 0) \\
&= E_{\mathbf{x}\sim S, z\sim N(0,1), v\sim N(0,1)} I((\mu w + z)x_{\|} + v x_{\perp} \geq 0) \\
&= E_{\mathbf{x}\sim S, z\sim N(0,1), v\sim N(0,1)} I\left(\mu w \geq z + v\frac{x_{\perp}}{x_{\|}}\right) \\
&= E_{\mathbf{x}\sim S, z\sim N\left(0, 1+\frac{x_{\perp}^2}{x_{\|}^2}\right)} I\left(\mu w \geq z\right) = E_{\mathbf{x}\sim S} E_{z\sim N\left(0, \frac{1}{g(\mathbf{x})^2}\right)} I\left(\mu w \geq z\right) \\
&= E_{\mathbf{x}\sim S} \tilde{F}(\mu w g(\mathbf{x})).
\end{aligned}$$

6 Experiment: Condition Monitoring

The purpose of this section is to analyse the comparative performance of the proposed soft margin kernel RMVCE algorithm and the one-class SVM algorithm on a real-life dataset from the Structural Integrity and Damage Assessment Network [1]. There are vibration measurements in this dataset that correspond to "healthy" measurements (without fault) and 4 types of malfunction of machinery: Fault 1, Fault 2, Fault 3 and Fault 4 (see [1] for details). In order to compare the proposed RMVCE method (see (6)) with the one-class SVM method [7,11], we performed experiments in the similar manner as described in [1]: 1) "Healthy" measurements ($\ell = 150$) are used to train the RMVCE (see (6)) and the one-class SVM [7]; 2) one hundred fifty samples (Fault 1) are used to validate the results of training. It can be seen that the proposed RMVCE can be successfully used for novelty detection as a valuable alternative to the minimum volume enclosing hypersphere for novelty detection (see Table 1). The RMVCE method can be also applied to Gaussian Processes to perform optimal experimental design [4].

Table 1. The percentage of correctly labeled classes using one-class SVM and RMVCE methods with Gaussian kernel, $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-0.5||\mathbf{x}_i - \mathbf{x}_j||^2/\sigma^2)$

Method	σ	ν	γ	**Healthy**	**Fault 1**	**Fault 2**	**Fault 3**	**Fault 4**
RMVCE,	320	0.3	0.02	**100%**	**91%**	**100%**	**90%**	**61%**
RMVCE,	320	0.25	0.02	92%	100%	85%	55%	75%
1-SVM	320	0.25	-	79%	100%	98%	85%	93%
1-SVM	320	0.001	-	**90%**	**100%**	**95%**	**68%**	**85%**

7 Conclusions

We have tackled the novelty detection problem using the MVCE. While the MVCE can directly be used in low dimensional spaces, it is problematic in high dimensional spaces. To resolve this, we introduced regularisation, which allowed us to derive a learning theory bound guaranteeing a maximal probability of misidentifying an outlier. Finally, we presented a kernel version allowing to model nonlinearly shaped supports and supports for structured data types.

Acknowledgements. T.De Bie acknowledges support from the CoE EF/05/007 SymBioSys, and from GOA/2005/04, both from the Research Council K.U.Leuven.

References

1. Campbell, C., Bennett, K.P. A Linear Programming Approach to Novelty Detection, Advances in Neural Information Processing Systems 14 (NIPS01), 2002.
2. Dolia, A. N., Harris, C. J., Shawe-Taylor, J. and Titterington, D. M. (2005) Kernel Ellipsoidal Trimming, Internal Report T8.11.10-01/05, School of Electronics and Computer Science, University of Southampton, 11 October 2005.
3. Dolia, A.N., Page, S.F., White, N.M., Harris, C.J. D-optimality for Minimum Volume Ellipsoid with Outliers, Proc. of the 7th International Conference on Signal/Image Processing and Pattern Recognition, (UkrOBRAZ'2004), 73–76, 2004.
4. Dolia, A.N., Harris, C.J., Shawe-Taylor, J., De Bie,T. (2006) Gaussian Processes for Active Sensor Management, Gaussian Processes in Practice Workshop, Bletchley Park, UK 12 - 13 June 2006
5. Langford, J. Tutorial on Practical Prediction Theory for Classification, Journal of Machine Learning Research, **6**:273–306, 2005.
6. Langford, J. and Shawe-Taylor, J. "PAC Bayes and Margins, Advances in Neural Information Processing Systems 15 (NIPS02), 2003.
7. B. Schölkopf and J.C. Platt and J.S. Shawe-Taylor and A.J. Smola and R.C. Williamson. Estimating the Support of a High-Dimensional Distribution, Neural Computation, **13**(7):1443–1471, 2001.
8. Shawe-Taylor, J. and Cristianini, N. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK (2004)
9. Shawe-Taylor, J. and Williams, C. and Cristianini, N. and Kandola, J. S. On the Eigenspectrum of the Gram Matrix and Its Relationship to the Operator Eigenspectrum. Proc. of the 13th International Conference on Algorithmic Learning Theory (ALT2002) **2533**, 2002.
10. Sun, P. and Freund, R.M. Computation of Minimum-Volume Covering Ellipsoids, Operations Research **52**(5):690–706, 2004.
11. Tax, D.M.J., Duin, R.P.W. Data Domain Description by Support vectors. Verleysen, M. (ed.). Proceedings, ESANN. Brussels, 251–256, 1999.
12. Titterington, D.M. Estimation of Correlation Coefficients by Ellipsoidal Trimming, Journal of Royal Statistical Society **C27**(3):227–234, 1078.

Right of Inference: Nearest Rectangle Learning Revisited

Byron J. Gao and Martin Ester

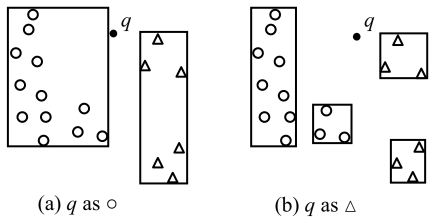
School of Computing Science, Simon Fraser University, Canada
{bgao, ester}@cs.sfu.ca

Abstract. In Nearest Rectangle (*NR*) learning, training instances are generalized into hyperrectangles and a query is classified according to the class of its nearest rectangle. The method has not received much attention since its introduction mainly because, as a hybrid learner, it does not gain accuracy advantage while sacrificing classification time comparing to some other interpretable eager learners such as decision trees. In this paper, we seek for accuracy improvement of *NR* learning through controlling the generation of rectangles, so that each of them has the *right of inference*. Rectangles having the right of inference are compact, conservative, and good for making local decisions. Experiments on benchmark datasets validate the effectiveness of the proposed approach.

1 Introduction

Nearest Rectangle (*NR*) learning [9] is a hybrid inductive learning approach, in which training instances are generalized into axis-parallel hyperrectangles, and a query is classified according to its nearest rectangle. If a query falls inside a rectangle, its distance to that rectangle is zero; if the query lies outside a rectangle, the distance is the (weighted) Euclidean distance from the query to that rectangle. If the query is equidistant to several rectangles, the smallest of which is chosen. The rectangles we mention in this paper are isothetic bounding boxes of the instances they contain, unless otherwise specified.

NR learners belongs to the class of hybrid lazy-eager learning algorithms. Lazy algorithms such as k-Nearest Neighbor (kNN) classifiers are instance-based and non-parametric, where the training data are simply stored in memory and the inductive process is deferred until a query is given. In contrast, eager algorithms such as decision trees, neural networks, and naive Bayes classifiers are model-based and parametric, where the training data are greedily compiled into a concise hypothesis (model) and then completely discarded. Obviously, lazy algorithms incur lower computational costs during training but much higher costs in answering queries also with greater storage requirements, not scaling well to large datasets. They do not generate interpretable models as some eager algorithms, in particular, decision trees can be directly inspected to understand the decision surfaces embedded in data even for non-technical end-users. This ease of comprehension is very appealing in decision support related data mining activities, where insight and explanations are of critical importance [2].



In (a), the rectangles make "wild", inappropriate inferences and the query q is classified as ○;

In (b), the rectangles generalize the same training instances in a compact and conservative fashion, having the right of inference, and q is more appropriately classified as △.

Fig. 1. Right of inference

However, in terms of accuracy, lazy methods can be more advantageous. They do not lose information since all the training data are retained. They have additional information to utilize, the query instances, so that local and adaptive decisions can be made for predictions. On the other hand, eager methods try to make predictions that are good on average using a single global model.

To compromise on some of the distinguishing characteristics of purely lazy or eager methods, hybrid lazy-eager algorithms are studied. As an example, *NR* learning partially processes the training instances and generalizes them into hyperrectangles; these intermediate results are retained and used to answer queries. Nonetheless, the *NR* method has not received much attention mainly because it is considered not accurate enough. The original *NR* learning algorithm as well as several improved versions were experimentally compared with *kNN* [10,11], and it was concluded that the *NR* approach performed well in domains with axis-parallel decision boundaries; while in other occasions it was significantly inferior to *kNN* in terms of accuracy. Comparing to axis-parallel decision trees, which are essentially rectangle-based, the rectangles induced by *NR* learners also offer a level of intuitive interpretability. However, as a hybrid approach, *NR* is slower in answering queries; then with similar accuracy, it has no advantage over decision trees and this line of research discontinued soon after its introduction.

We revisit *NR* learning, and propose that the major reason accounting for its loss of accuracy in previous endeavors was that, the generalized rectangles were not given the *right of inference* that guarantees the appropriateness of rectangles in making inferences. In general, rectangles having the right of inference should be compact, conservative, and good for making local decisions, as illustrated in Fig. 1. By imposing the right of inference on rectangles, *NR* classifiers can potentially be intuitively explanatory, fast, scalable, yet highly accurate, having many combined appealing properties from decision trees and *kNN* classifiers.

1.1 Related Work

Decision trees [6] are typical eager learners while *kNN* classifiers [4] exemplify the simplest form of lazy learners. [1] identified the distinguishing characteristics of eager and lazy learners. Both types of learners have their own desirable properties. To obtain good trade-offs, varied hybrid approaches were proposed, e.g., [7] introduced a method combining instance-based and model-based learning.

As a hybrid approach, nearest rectangle learning was first introduced in [9] under the name of Nested Generalized Exemplar (*NGE*) theory. In *NGE*, an exemplar can be a generalized axis-parallel hyperrectangle or a single training instance, which is a degenerate (trivial) rectangle. Arbitrary overlapping and nesting of rectangles of different classes are allowed. [10,11] challenged the accuracy performance of *NGE* and made several improvement attempts such as disallowing nesting and/or overlapping, modifying the rectangle construction heuristic, and weighting features by mutual information. It was concluded that the major reason leading to the loss of accuracy of *NGE* was the overlapping of rectangles of different classes, yet the best improved version was still significantly inferior to *kNN* in most of the tested datasets. We notice that, all the above attempts did not pay much attention to the quality of the generated rectangles. They allowed rectangles to make wild and inappropriate inferences, which would significantly deteriorate the accuracy performance as illustrated in Fig. 1.

[5] studied cluster description formats, problems and algorithms, which also involved discriminative summarization of labeled data using hyperrectangles. But they considered only a two-class problem concerning objects in or not in the cluster. Moreover, their focus was on description (generalization) instead of classification (inference); the appropriateness of inference of rectangles was not an issue, but the conciseness of descriptions, i.e., the number of rectangles.

In the remaining of the paper, Section 2 discusses the concept of right of inference and its enforcement. Section 3 proposes LearnCovers, an *NR* learning heuristic. Section 4 presents empirical results and Section 5 concludes the paper.

2 Right of Inference and Its Enforcement

2.1 Right of Inference

Rectangle-based classifiers can provide certain degree of insight and understanding into data and the instance space. In fact, consider a closed rectangular instance space, the leaf nodes of an axis-parallel decision tree correspond to a set of isothetic rectangles (not bounding boxes) forming a partition of the instance space. The induction of decision trees generalizes the training data and makes inferences to the entire instance space simultaneously with the disjointness constraint, striving to achieve good-on-average predictions. Intuitively, if we separate generalization and inference into two serial phases and allow same-class rectangles to overlap, we should be able to build classifiers that are more flexible, adaptive and accurate, with the capacity to make local decisions.

Potentially, *NR* learning can induce such explanatory, adaptive and accurate classifiers. However, if in the generalization phase, the rectangles are not constructed in a conservative and compact fashion, they would make wild and improper inferences, similar to the case of decision trees, as demonstrated in Fig. 1 (a). It can be inspected that decision trees would make the same decision for the query in the figure. On the contrary, Fig. 1 (b) illustrates some compact rectangles for the same training data that are good for making local decisions, having the so-called right of inference.

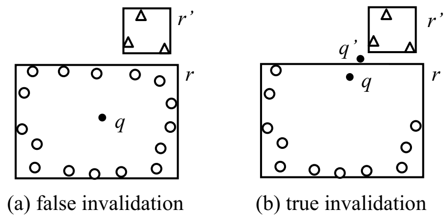


Fig. 2. For Definition 1

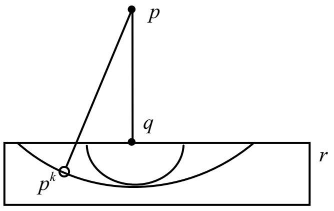


Fig. 3. For Theorem 1

The *right of inference* of a rectangle can be conceptually defined as the privilege that the rectangle has to make sound and local inferences. As we have seen, rectangles having the right of inference should appear compact and saturated. Then, how to define right of inference in a quantitative manner?

Definition 1. (*right of inference*) *A rectangle r has the right of inference if and only if for any query q outside of r, $dist(q, q^k) - dist(q, r) \leq \delta$, where $dist(q, q^k)$ is the Euclidean distance from q to its kth nearest instance in r, $dist(q, r)$ is the Euclidean distance from q to r, and δ is the distance threshold.*

The distance from q to r is equivalent to the line dropped perpendicularly from q to the nearest face, edge, or vertex of r, which is formally defined as follows, without considering rectangle weighting and feature weighting. Let q_{f_i} be the value of q on the *ith* feature, where $1 \leq i \leq m$; let r_{lower,f_i} and r_{upper,f_i} be the lower and upper end values of r on the *ith* feature, then:

$$dist(q, r) = \sqrt{\sum_{i=1}^{m} dif_i^2} \quad \text{where} \quad dif_i = \begin{cases} q_{f_i} - r_{\text{upper},f_i} & \text{when} \quad q_{f_i} > r_{\text{upper},f_i} \\ r_{\text{lower},f_i} - q_{f_i} & \text{when} \quad q_{f_i} < r_{\text{lower},f_i} \\ 0 & \text{otherwise} \end{cases}$$

What is the rationale behind the right of inference thus-defined? Note that, we always have $dist(q, r) \leq dist(q, q^k)$. If $dist(q, q^k) - dist(q, r)$ is too large, the inference on the class of q from r might be inappropriate, since $dist(q, r)$ would alter (bring closer) the locality of the instances in r with respect to q in an intolerable manner; e.g., in Fig. 1 (a), q is very close to the left rectangle, but far away from the instances in it. In contrast, if $dist(q, q^k) - dist(q, r)$ is reasonably small, *NR* classifiers would behave similarly to *kNN*, as shown in Fig. 1 (b).

In Definition 1, we only consider queries lying outside of the rectangle r. This is because some inside query may falsely invalidate a "good" r. In Fig. 2 (a), even if $dist(q, q^k) - dist(q, r)$ is rather large, r is good because q would not be closer to other instances/rectangles of different classes, say r'. Recall that overlapping of rectangles is allowed only if they have the same class label. On the other hand, for a "bad" r as shown in Fig. 2 (b), not considering q or other queries in r would not falsely validate r since if q should invalidate r (closer to r'), there would be another q' outside of r that also invalidates r. We can easily find such q', say, somewhere close to r and on the line joining q and r'.

In Definition 1, it is also reasonable to use the average of distances from q to its k nearest instances in r for $dist(q, q^k)$. k is limited by the number of instances in r, and the choice of k can be a legitimate research issue just as in the case of *kNN* classification. The distance threshold δ has a direct impact on how closely *NR* classifiers would behave to *kNN*. If δ is too large, the rectangles tend to be very large as well making unconstrained inferences. If δ is too small, *NR* learning would induce too many rectangles and become "lazy", losing the desirable properties as a hybrid learner. In the extreme case of $\delta = 0$, *NR* learning would lose the generalization capacity completely and essentially become 1-*NN*.

2.2 Enforcing the Right of Inference

It is not straightforward to enforce the right of inference defined in Definition 1 since there are potentially infinite number of queries to examine. In the following, we discuss some inspiring observations and practical implications.

Theorem 1. *If for any query q that is on the surface of a rectangle r, $dist(q, q^k) \leq \delta$, where q^k is the kth nearest instance of q in r, then r has the right of inference defined in Definition 1 with respect to δ.*

Proof. Let p be any query outside of r. Let p^k be the *kth* nearest instance of p in r and q the projected p on the nearest face of r, as depicted in Fig. 3. According to the definition of point-to-rectangle distance, $dist(p, q) = dist(p, r)$. We use arc_p to denote the intersection of r and the sphere with radius $dist(p, p^k)$ centered at p, and arc_q to denote the intersection of r and the sphere with radius $dist(p, p^k) - dist(p, q)$ centered at q. Clearly, $arc_q \subseteq arc_p$.

Since p^k is the *kth* nearest instance of p in r, the number of instances in arc_p is less or equal to k if not considering ties. Since arc_p and arc_q intersect on only one point, the number of instances in arc_q is less or equal to k even considering ties. That is to say, q^k, the *kth* nearest instance of q in r, lies outside or on the surface of arc_q. In other words, $dist(q, q^k) \geq dist(p, p^k) - dist(p, q) = dist(p, p^k) - dist(p, r)$. Therefore, $\delta \geq dist(q, q^k) \Rightarrow \delta \geq dist(p, p^k) - dist(p, r)$, and the conclusion of Theorem 1 follows.

The implication of Theorem 1 is that, we only need to consider queries on the surface of r to test its right of inference. In the prototype *NR* learner LearnCovers, to be proposed shortly, a simple recursive testing and bisecting enforcement heuristic is embedded. For testing, the query pool consists of a constant number of queries generated according to a ranking scheme that gives high ranks to queries with high probability of invalidating r. Generally, highly ranked queries include vertices that are far away from the mean of the instances in r. Certain positions (say, centers) on long edges or large faces have the next priority to be inserted in the query pool, and then uncertain (random) positions on the surface of r. r is validated (passes the test) if it is not invalidated by any query in the query pool. If r is invalidated, the k-means clustering algorithm with $k = 2$ is applied to bisect the instances in r, and the newly generated rectangles (bounding boxes of the two sections) are tested separately. This recursive testing and bisecting process terminates until all the rectangles pass the test.

3 LearnCovers: Learning the "Right" Rectangles

LearnCovers heuristically constructs a set of rectangles with minimized cardinality and enforced right of inference. The rectangles generalize the given training instances with 100% accuracy, and same-class rectangles are allowed to overlap.

Algorithm 1. LearnCovers

```
1.  $R = \emptyset$; //$R$: the set of generated rectangles
2.  sort $T$; //$T$: the given training set
3.  for each $t \in T$ { //process each training instance $t$ in the sorted order
4.      for each $r \in R$ {
5.          calculate $cost(r, t)$; //$r$ with smaller $cost(r, t)$ is favored in covering $t$
6.          if ($r.class$ != $t.class$ && $cost(r, t)$ == 0)
7.              validateToclose($r$); //$r$ can be closed only after validation
8.          if ($r.class$ == $t.class$ && $r$ is not closed && $cost(r, t)$ == 0)
9.              extend $r$ to cover $t$ and continue to process the next $t$; } //back to line 3
10.     for each $r \in R$ { //$t$ was not covered; fetch $r$ in ascending order of $cost(r, t)$
11.         if ($r.class$ == $t.class$ && $r$ is not closed && violationCheck($r, R$) == no)
12.             expand $r$ to cover $t$ and continue to process the next $t$; } //back to line 3
13.     insert($R, r_{new}$) }; //$t$ cannot be covered; insert the trivial rectangle $r_{new}$ to $R$
14. enforce($R$);
```

The pseudocode is presented in Algorithm 1. R, the rectangle set, is initialized to be empty (line 1). Instances in the given training set T are sorted along a selected feature (line 2) and processed in the sorted order. For each training instance t (line 3), we search through R (line 4) for the best rectangle to accommodate it. Expanding rectangles would incur *covering violations*, i.e., overlapping of rectangles of different classes, which are not allowed. The best rectangle to cover t is the one with the smallest *covering cost* with respect to t, which is defined such that the number of generated rectangles can be minimized.

In line 5, the cost of r in covering t, $cost(r, t)$, is calculated. $cost(r, t) = 0$ only if t lies straightly under r, i.e., by simply extending r along the selected sorting feature, t will be covered by r. If $cost(r, t) = 0$ and r and t are of different classes (line 6), r is closed on condition that it can be validated; otherwise, r is bisected and the two propagated rectangles are inserted into R (line 7). Closed rectangles will not be considered in the remaining procedures, since they cannot be used to cover any further instances without causing violations.

If a non-closed r has the same class label as t with $cost(r, t) = 0$ (line 8), r is an optimal rectangle to cover t. We can simply stop searching and continue to process the next instance (line 9). Note that in this case, violation checking is unnecessary since instances in T are sorted and we only need to extend r along the sorting feature to cover t.

If t has not been covered by such an optimal r (line 10), we need to search through R for the best r with the smallest $cost(r, t)$. The rectangles in R will be considered in the ascending order of $cost(r, t)$, the first available one (line 11) will be used to cover t and we can continue to process the next instance (line 12).

If there is no such $r \in R$ that can cover t without incurring a violation, a trivial rectangle r_{new} for t will be constructed and inserted into R (line 13).

Upon reaching line 14, all the training instances in T have been processed and generalized. The enforcement heuristic discussed previously is applied to all non-closed rectangles in R (closed ones must have been validated), and all the recursively propagated rectangles will be inserted into R after validation. In the actual implementation, we have chosen $k = 1$ for testing the right of inference, that is, any query q on the surface of r with $dist(q, q^1) > \delta$ will invalidate r.

As for the choice of δ, we randomly sample a series of queries. For each query q, we record $dif_q = dist(q, t_q) - dist(q, \overline{t_q})$, where t_q is the nearest training instance of q and $\overline{t_q}$ is the nearest training instance of q that has a different class label from t_q. If we set $\delta = dif_q$, the resulting *NR* classifier behaves the same as 1-*NN* on q and q will not be assigned a class label other than the one of t_q. To see why, let r_{t_q} and $r_{\overline{t_q}}$ be any two rectangles covering t_q and $\overline{t_q}$ respectively, then $dist(q, r_{t_q}) \leq dist(q, t_q)$ and $dist(q, \overline{t_q}) - dist(q, r_{\overline{t_q}}) \leq \delta$ if $r_{\overline{t_q}}$ is enforced the right of inference, from which $dist(q, r_{t_q}) \leq dist(q, r_{\overline{t_q}})$ follows. Intuitively, since the right of inference is enforced on $r_{\overline{t_q}}$, $\overline{t_q}$ will not be brought close enough by the rectangular generalization to challenge the locality of t_q with respect to q. Note that t_q is also brought closer to q by r_{t_q}. After obtaining a series of dif_q's, we use the average value as δ. While how to decide δ deserves further investigations, a more practical situation would be, selecting δ so as to meet a given constraint on the maximum number of rectangles allowed.

The proposed *NR* learner LearnCovers is an extension of Learn2Cover [5], a discriminative summarization heuristic for labeled data, from 2 class to multi-class and with the right of inference enforcement mechanism embedded. Some related issues, such as selecting the sorting feature, handling ties, defining $cost(r, t)$ and so on, are discussed in [5] with more details.

4 Empirical Results

A series of experiments were conducted to evaluate the accuracy performance of the proposed *NR* learner LearnCovers. The notion of rectangle can be extended to tolerate categorical features but not in this prototype version; thus 20 numerical benchmark datasets without missing values from the UCI repository [3] were used to run C4.5 [8], 1-*NN*, *kNN* and LearnCovers. For *kNN*, the highest accuracy was recorded. The datasets were normalized on each feature. For each of the datasets where cross-validation was needed, the averaged result over 3 runs of stratified 10-fold cross-validation was taken.

In Table 1, "Att", "Ins" and "Cla" indicate the numbers of attributes, instances and classes respectively for the datasets. The results show that, LearnCovers outperforms C4.5 in 19, 1-*NN* in 12, and *kNN* in 8 of the 20 datasets. It has the averaged accuracy of 0.857, significantly higher than C4.5 (0.817), better than 1-*NN* (0.843) and comparable to *kNN* (0.864). Recall that, without considering the right of inference, but assisted by some other sophisticated techniques such as rectangle weighting and feature weighting using mutual information, the

Table 1. Accuracy: C4.5, 1-*NN*, *kNN*, and LearnCovers (LC)

Dataset	Att	Ins	Cla	C4.5	1-*NN*	*kNN*	LC	Dataset	Att	Ins	Cla	C4.5	1-*NN*	*kNN*	LC
balance	4	625	3	0.758	0.790	0.900	0.828	pima	8	768	2	0.737	0.701	0.738	0.752
bupa	6	345	2	0.655	0.632	0.652	0.672	satimage	36	6435	6	0.850	0.894	0.906	0.878
car	6	1728	4	0.917	0.917	0.951	0.938	segment	19	2310	7	0.960	0.974	0.974	0.966
ecoli	7	336	8	0.841	0.806	0.871	0.869	sonar	60	208	2	0.702	0.865	0.865	0.794
glass	10	214	6	0.687	0.701	0.712	0.739	spambase	57	4601	2	0.895	0.908	0.908	0.897
iono	34	351	2	0.900	0.869	0.869	0.937	vehicle	18	846	4	0.734	0.696	0.725	0.709
iris	4	150	3	0.953	0.953	0.967	0.973	vowel	10	990	11	0.788	0.989	0.989	0.973
letter	16	20000	26	0.868	0.955	0.955	0.925	waveform	21	5000	3	0.781	0.809	0.833	0.808
new-thyr	5	215	3	0.916	0.968	0.968	0.953	wine	13	178	3	0.936	0.949	0.972	0.977
page-blo	10	5473	5	0.965	0.957	0.959	0.971	yeast	8	1484	10	0.494	0.526	0.574	0.581
								Average				0.817	0.843	0.864	0.857

remedies proposed in [10,11] only achieved moderate accuracy improvement on the original *NR* learner, remaining "significantly inferior to *kNN*".

5 Conclusion

In this paper, we revisited *NR* learning, seeking for its accuracy improvement through imposing the right of reference on rectangles. Experiments on benchmark datasets demonstrated the effectiveness of the proposed approach. For future work, more effective and efficient testing and enforcement mechanisms should be investigated. Grounded on the right of inference of rectangles, there are several interesting directions to further extend *NR* learning. One is to consider k nearest (weighted) rectangles in classification; another is to consider creating a classifier ensemble with multiple rectangle sets that, for example, can be obtained from LearnCovers by varying the sorting feature.

References

1. D. Aha. Lazy learning. *Artificial Intelligence Review*, 11:7-10, 1997.
2. C. Apte and S. Weiss. Data mining with decision trees and decision rules. *Future Generation Computer Systems*, 1997.
3. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
4. B.V. Dasarathy. Nearest Neighbor (NN) norms: NN pattern classification techniques. *IEEE Computer Society Press*, 1991.
5. B.J. Gao and M. Ester. Cluster description formats, problems, and algorithms. In *SIAM International Conference on Data Mining*, 2006.
6. S.K. Murthy. Automatic construction of decision trees from data: a multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2(4):345-389, 1998.
7. J.R. Quinlan. Combining instance-based and model-based learning. In *ICML*, 1993.
8. J.R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, 1993.
9. S. Salzberg. A nearest hyperrectanhgle learning method. *Machine Learning*, 6:251-276, 1991.
10. D. Wettschereck. A hybrid nearest-neighbor and nearest-hyperrectangle algorithm. In *ECML*, 1994.
11. D. Wettschereck and T.G. Dietterich. An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, 19:5-27, 1995.

Reinforcement Learning for MDPs with Constraints

Peter Geibel

Institute of Cognitive Science, AI Group, University of Osnabrück, Germany
`pgeibel@uos.de`
`www.cs.tu-berlin.de\~geibel`

Abstract. In this article, I will consider Markov Decision Processes with two criteria, each defined as the expected value of an infinite horizon cumulative return. The second criterion is either itself subject to an inequality constraint, or there is maximum allowable probability that the single returns violate the constraint. I describe and discuss three new reinforcement learning approaches for solving such control problems.

1 Introduction

Most approaches in reinforcement learning (RL, see e.g. [8]) consider only Markov decision processes (MDPs) with a single criterion, or with several criteria related to hierarchical dependencies between behaviors. On the other hand, in practical applications like robot control, there might exist several possibly *conflicting* objectives requiring a strategy that mediates between them. Problems with multiple, non-hierarchical objectives have hardly been considered in RL, although some articles from the field of dynamic programming (DP, [2]) can be found.

A typical example for a problem with constraints is the accomplishment of some task with a limited amount of energy or time expressed as a second criterion subject to a constraint. Imagine e.g. a robot equipped with a battery. The task of the robot is to collect as much items as possible, but it shouldn't run out of energy.

I will consider two different kinds of constraints. The first group of problems have a constraint on the second criterion function itself, i.e. on the *expected value* of the return. Such problems are typically called constrained MDPs `CMDPs` in the following (see also [1]). The second group contains problems in which we constrain the probability that the return, considered a random variable, violates a constraint. Such problems will be called MDPs with constrained probability of constraint violation (`CPMDP`). Examples are constraints on the probability of resource overutilization as discussed by Dolgov and Durfee in [4,3]. In `CPMDP`s, actually *two* constraints are involved.

Since applications with unequal discount factors are relatively rare and very difficult to solve [5,6,7], we will only consider MDPs with several criteria each based on its own reward function, but using a *common discount factor* γ. We will also focus on MDPs with two criteria only, where the first one is to be optimized, and the second one is subject to a constraint.

The purpose of this paper is to undertake a description and comparison of different approaches for solving constrained problems (including some new ones), and to discuss their respective advantages and shortcomings. In section 2, unconstrained RL problems including Markov Decision Processes, policies, and value functions are introduced. In section 3, I consider MDPs with constraints, i.e. CMDPs and CPMDPs. I will present standard solutions methods as well as three new approaches for solving CMDPs and CPMDPs.

Each method has a parameter that allows to select different behaviours with respect to the first and second criterion function yielding a curve in a 2-dimensional space corresponding to the two criteria in the case of CMDPs I will base the experimental comparison in section 5 on this approach that can also be used for CPMDPs. A concluding discussion can be found in section 6.

2 Unconstrained MDPs

In RL and DP, one considers finite Markov decision processes (MDPs), that are characterized by a finite state set X, a finite action set U, and state transition probabilities $p_{x,u}(x')$ defined as the probabilty that x' is reached when u is executed in x. The value $r_{x,u}$ denotes the reward obtained when executing action u in state x.

A **policy** represents the action selection strategy of the agent. Stationary, deterministic policies are functions π mapping a state x to an action $\pi(x)$. Randomized policies are described using state dependent distributions $\pi(x, .)$ on possible actions.

The aim of the agent is to find a policy π for selecting actions that maximizes the cumulative reward, called the return. The return is defined as $R = \sum_{t=0}^{\infty} \gamma^t r_t$, where the random variable r_t denotes the reward occurring in the t-th time step when the agent uses policy π. Let $x_0, x_1, x_2, \ldots$ denote the corresponding probabilistic sequence of states, and u_i the sequence of actions chosen according to policy π.

The constant $\gamma \in [0, 1]$ is a discount factor that allows to control the influence of future rewards. The expectation of the return $V^\pi(x) = \mathbb{E}\Big[R \,|\, x_0 = x\Big]$ is defined as the **value** of x with respect to π. It is well-known that there exist stationary, deterministic policies π^* for which $V^{\pi^*}(x)$ is optimal (maximal) for *every* state x simultaneously. The optimal values $V^*(x) := V^{\pi^*}(x)$ are the same for every optimal policy π^*.

In order to define optimal stationary policies, let D be an initial distribution on the possible starting states, e.g. the uniform distribution on the set X. We define the **value of a policy** as the expected value of the value function, i.e.

$$\mathcal{V}^\pi = \mathbb{E}_{x \sim D}\,\big[V^\pi(x)\big] = \sum_{x \in X} D(x) V^\pi(x)\,. \tag{1}$$

For a fixed distribution D, this value is maximized by any optimal stationary, deterministic policy.

While DP algorithms often assume a fully know model, RL algorithms like Q-Learning are able to learn in interaction with the real process. We suppose that the reader is familiar with basic RL techniques and leave out the defintion of the algorithm. If the model is known, then standard approaches can be applied. One approach consists in formulating a linear program and solve it with standard techniques, see [1,4].

3 Problems with Constraints

A constrained MDP has an additional **second reward function** $c_{x,u}$ that is used to define the constrained value function C^π, see below. In the case that $c_{x,u} \leq 0$ holds, these values can be considered costs for the actions, but positive values, i.e. rewards, might also occur.

We define constrained MDPs (CMDPs) as problems of the form

$$\begin{aligned} &\max \mathcal{V}^\pi \\ &s.t. \;\; \mathcal{C}^\pi \;\geq\; c \end{aligned}$$

where the threshold c is a real value, and $\mathcal{C}^\pi = \mathbb{E}\; C^\pi(x)$ with $C^\pi(x) = \mathbb{E} \sum_{t=0}^{\infty} \gamma^t c_{x_t,u_t}$. Problems with $\leq$ instead of $\geq$ can be normalized to yield the above form.

From a practical point of view, we often consider (A) *problems with maximum costs*, in which all $c_{x,u} \leq 0$ and $c \leq 0$, e.g. a robot task and risk-sensitive control as discussed by Geibel und Wysotzki in [7]; (B) *Problems with minimum gain,* in which all $c_{x,u} \geq 0$ and $c \geq 0$, e.g. the problem of Buridan's ass, see [6], where a minimum return must be achieved on average; (C) *Mixed Problems* where the $c_{x,u}$ might take on positive as well as negative values and c is arbitrary.

In the robot example, one can argue that the introduction of the expectation operator makes no sense, because we want the robot to *never* run out of energy, or only with a maximum allowable probability p_0, see [4]. Let C denote a random variable that denotes the $c_{x,u}$-based cumulative return occurred in a single run. Now we define MDPs with constrained probability of constraint violation (CPMDPs) as maximizing $\mathcal{V}^\pi$ under the condition

$$P(C \leq c') \leq p_0 \tag{2}$$

with c' being the threshold for the $c_{x,u}$-based return.

It should be noted, that for constrained problems it is no longer the case that stationary deterministic policies are optimal. First of all, we might need to consider randomized policies and a dependence on the initial distribution D. For CMDPs randomized optimal policies can be found by solving a modified linear program.

4 Solution Approaches for MDPs with Constraints

In the following, I will describe and discuss several approaches for solving MDPs with constraints. I will start with the DP approach because it constitutes a baseline for comparing the performance of the approaches.

4.1 `LinMDP`: Linear Programming

In order to solve a standard constrained MDP, a linear program describing optimal solutions of the unconstrained MDP is simply augmented by an additional constraint expressing $\mathcal{C}^\pi \geq c$ is required to hold.This augmented program can again be solved with standard linear programming methods. The method yields a randomized optimal policy dependent on the `CMDP` to be solved, and the initial distribution D. In the following we refer to this method as `LinMDP`.

`LinMDP` is tailored for problems with constraints on the expected costs. As described by Dolgov and Durfee in [4], it can be used for `CPDMPs` by mapping to a `CMDP` with constant $c = p_0 c'$ which is possible in the case of negative values of the $c_{x,u}$ (based on the Markov inequality).

Because `LinMDP` might be suboptimal for solving `CPMDPs`, we propose the following method: the policy π resulting from solving the linear `CMDP`-program has also a specific probability p_π for constraint violation. I.e. given a fixed c' (for the `CPMDP`), instead of setting $c = p_0 c'$ we can vary the c in the corresponding `CMDP`. We then pick that c which results in a feasible policy π for which $p_\pi \leq p_0$ holds and that has the highest $\mathcal{V}^\pi$-value. p_π, $\mathcal{C}^\pi$, and $\mathcal{V}^\pi$ can be estimated using several test runs. In the following, we will refer to this method for solving `CMDPs` as well as `CPMDPs` as `LinMDP`.

4.2 `WeiMDP`: A Weighted Approach

Geibel and Wysotzki in [7] considered the problem of finding policies that have a constrained risk of failing in an MDP with error states. We expressed the probability of entering an undesirable state as an (undiscounted) second value function. This resulted in a constrained MDP with possibly unequal discount factors, which was solved by introducing a weight parameter for risk and value. For solving CMDPs, we suggest to introduce a weight parameter $\xi \in [0, 1]$ and a derived weighted reward function defined as

$$w_{x,u} = \xi r_{x,u} + (1 - \xi) c_{x,u} \, .$$

For a fixed ξ, this new unconstrained MDP can be solved with standard methods, e.g. Q-Learning resulting in an online-method.

Similar to `LinMDP` parameterized with c, using different values of $\xi \in [0, 1]$ will result in different points in the $(\mathcal{V}, \mathcal{C})$-space (`CMDPs`) and $(\mathcal{V}, p_.)$-space (`CPMDPs`), respectively. This method will be called `WeiMDP` in the following.

Again, there is a parameter ξ for choosing a suitable policy dependent on c in the case of `CMDPs` and on `CPMDPs`. Unlike the approach proposed by Dolgov and Durfee, we make no prior assumption on the sign of $c_{x,u}$ and c, i.e. we can naturally treat mixed problems with mixed signs.

In contrast to `LinMDP`, our algorithm performs online learning of a stationary-determinstic optimal policy for the weighted criterion that is a feasible one for the `CMDP` (if the problem has a solution).

4.3 AugMDP: State Space Extension

The rewards $c_{x,u}$ correspond e.g. to costs like energy or time. It is RL folklore to include the status of the battery in the state description. Because we want to deal with finite state MDPs only, the possible values of the so far accumulated costs (e.g. consumed energy since $t = 0$) need to be discretized in an appropriate manner, e.g. by using intervals of equal length covering the possible range of values.

Since we are interested in the costs with respect to a starting state x_0, we need to keep track of the elapsed time if $\gamma < 1$. Otherwise the cost of the successor state cannot be computed correctly. If $i(0)$ is the interval corresponding to zero costs, the process starts in the state $(x_0, i(0), 0)$ where x_0 is a starting state of the original MDP. Given a current state $(x, i(C), t)$, a successor state obtained for action u might be $(x', i(C + \gamma^t c_{x,u}), t + 1)$ where $c_{x,u}$ is the cost incurred by the executed action u, and $i(C + \gamma^t c_{x,u})$ the new interval.

We don't have to consider the time if $\gamma = 1$ holds. But for $\gamma < 1$, the state space is possibly infinite. Assuming a maximum episode length of T and N intervals for discretizing the costs, we arrive at an MDP having $|X|NT$ states if $\gamma < 1$, and $|X|N$ states if $\gamma = 1$ holds.

In order to solve a CPMDP, we apply e.g. Q-learning using $r_{x,u}$ and the augmented state space. An **additional negative reward** $S \leq 0$ is given, when the process enters a state such that the accumlated costs are below c', i.e. when the cost constraint of the CPMDP is violated. Using high absolute values of S will prevent the process from entering such states at all, yielding a policy with a minimal p_0.

In order to deal with CPMDPs and also with CMDPs, we vary S in some sufficiently large interval. Again we have a parameter to "tune the behaviour" until we find a feasible policy for the CMDP or CPMDP, respectively. This way we have a new method method that will be called AugMDP in the following.

The method seems only to be applicable to problems with a constraint on the maximum cost, but not such with a constraint on the minimum profit. The latter start with initial states where the constraint is already violated (the initial gain is zero) resulting in a punishment S right from the start.

4.4 RecMDP: Recursive Reformulation of the Constraint

Gabor, Kalmar, and Szepesvari [6] developed an approach that is suited for dealing with problems of the type (B) described above, i.e. in which $c_{x,u} \geq 0$ and $c \geq 0$ hold.

Gabor, Kalmar, and Szepesvari give a recurrent reformulation of the constraint $C^\pi(x) \geq c$ based on the observation that the actual value of C^π is not really important as long it is above the threshold c (the minimum gain). Note that $\forall x\; C^\pi(x) \geq c$ implies $\mathcal{C}^\pi \geq c$ for every distribution D on the starting states, while the reverse is not true in general. That is, the method will generally arrive at a feasible, suboptimal solution.

Gabor et al. propose the recurrent formulation of a new value function defined by $\bar{C}^\pi(x) = \min\Big(\bar{c}, C^\pi(x)\Big)$ as

$$\bar{C}^\pi(x) = \min\Big(\bar{c}, c_{x,\pi(x)} + \min(\bar{c}, \gamma \sum_{x' \in X} \big[p_{x,\pi(x)}(x')\bar{C}^\pi(x')\big])\Big) \tag{3}$$

It holds $\bar{C}^\pi(x) \leq C^\pi(x)$ holds if we set $\bar{c} = c$. Therefore we might choose a value $\bar{c} \geq c$ in order to cover a larger range of feasible policies π.

Based on the recursive formulations of V^π and $\bar{C}^\pi$, we developed an online algorithm not requiring an initially known model. We leave out the details of the algorithms for reasons of space.

The approach produces necessarily suboptimal stationary, deterministic policies. We have the parameter $\bar{c}$ to adapt the result of the algorithm as in the previous approaches.

Problems with maximum costs can be solved by adding a large enough positive constant k to the values of the $c_{x,u}$ resulting in $c_{x,u} + k \geq 0$ for all x and u. Note that it is not obvious what constant should be added to the threshold $\bar{c}$. But since we adapt $\bar{c}$ anyway, a suitable value of $\bar{c}$ can be found via trial and error. This method will be called `RecMDP` in the following.

5 Experiments

In this section we describe the results of experiments with a series of randomly generated MDPs. We decided to focus on problems with maximum costs (i.e., $c_{x,u}, c \leq 0$) because such problems occur most often in RL applications (e.g., time, energy). In our first experiment, we focused on uniform distributions D. The MDPs were generated in the following manner:

- States: the number of states was selected randomly in the interval $[2, 50]$. With a probability of $\frac{1}{|X|}$, a state was turned into an absorbing state.
- Actions: the number of actions ranged between 2 and 4. We generated randomized actions such that for every state x and action u, $p_{x,u}(x') > 0$ holds for only approximately 25% of the possible successor states x'.
- Rewards: the rewards were selected in the interval $[0, 5]$ with uniform probability.
- Costs: $c_{x,u}$ was selected randomly in the interval $[-r_{x,u} - 1.0, -r_{x,u} + 1.0]$ to ensure that actions with a high reward also tend to have a high cost. Values larger than zero were set to 0.
- Discount factor: γ was selected randomly from the interval $[0, 1]$.

We decided to qualitatively compare the approaches by looking at the possible behaviours that can be generated using different parameter values ($c, \xi, S, \bar{c}$, resp.). We will focus on `CPDMPs` where the curves in the $(\mathcal{V}, p_.)$-space were depicted in Fig. 1. The results for `CMDPs` were quite similar. For reasons of space, we only depict the results for five MDPs being positively representative for all 20 runs performed. For the experiments, we chose c' as $-0.25\frac{6}{1-\gamma}$ where $\frac{6}{1-\gamma}$ is

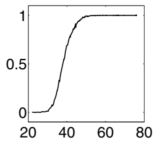
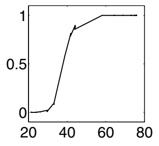
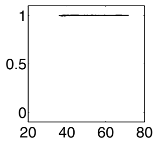
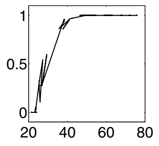
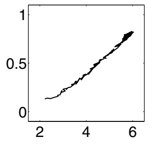
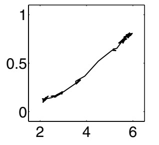
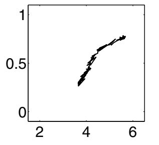
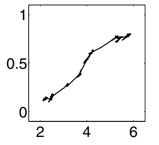
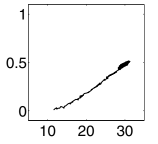
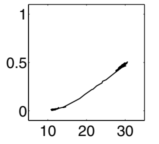
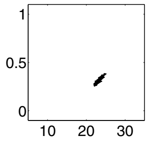
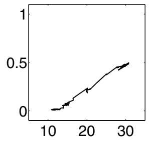
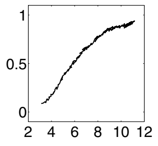
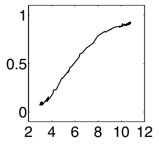
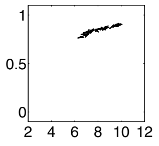
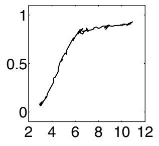
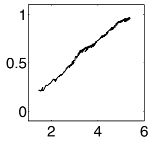
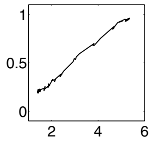
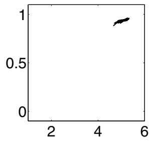
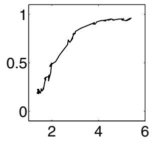
MDP	LinMDP	WeiMDP	AugMDP	RecMDP
$\lvert X\rvert = 12$ $\lvert U\rvert = 3$ $\gamma = 0.95$				
$\lvert X\rvert = 14$ $\lvert U\rvert = 2$ $\gamma = 0.61$				
$\lvert X\rvert = 31$ $\lvert U\rvert = 2$ $\gamma = 0.96$				
$\lvert X\rvert = 32$ $\lvert U\rvert = 4$ $\gamma = 0.69$				
$\lvert X\rvert = 19$ $\lvert U\rvert = 4$ $\gamma = 0.25$				

Fig. 1. CPMDP: curves in the $(\mathcal{V}, p_.)$-space

the theoretical upper bound for the accumulated costs given that the $c_{x,u}$ range in $[-6, 0]$ (see description of the MDPs above).

In Fig. 1, curves are to be considered better that cover a larger range of possible $\mathcal{V}$-values and p_π-values (corresponding to the projection onto the respective axis), and that attain better combinations of the two values, corresponding to curves that run more in the **lower right part** of the diagrams (i.e. with high returns and low probabilites of constraint violation). LinMDP is depicted in the first collumn. It can be seen that our weighted approach WeiMDP has a comparable performance. This is a very surprising result, because LinMDP can find randomized policies with a possibly better performance than the deterministic policies WeiMDP is restricted to. When looking at the policies computed by LinMDP, we found that randomization occurs very rarely which explain the small differences between WeiMDP and LinMDP.

AugMDP performed much worse especially with respect to the possible ranges of values, see Fig. 1. The reason is the very much enlarged state space that has to be considered. RecMDP performs quite well but seems to produce less stable results and worse combinations compared to WeiMDP and LinMDP.

6 Conclusion

All four presented methods have parameters that allow to switch between different behaviours. The parameters can be adapted to produce a feasible policy for the originally given constrained problem. We found that the method `LinMDP` performs best for `CMDP`s as well as `CPMDP`s, although it cannot be applied in an online learning manner. The weighted method `WeiMDP` performs quite well, too, and can be applied for problems with unknown model. Method `WeiMDP`, however, has a higher time complexity than `LinMDP`. Both methods can be extended for more than two criteria. Note that is possible to define handcrafted `CMDP`s, where `LinMDP` will outperform `WeiMDP`.

Encoding the costs in the state space (method `AugMDP`) yields the worst results, because of the much larger state space, and the approximation errors due to the necessary discretization of the possible cost values. We tried a series of different learning strategies with consistently bad results. The recursive method `RecMDP` produced acceptable results. Its performance on gain-constrained problems, for which it actually is designed, is still open and will be investigated in the future.

References

1. E. Altman. *Constrained Markov Decision Processes.* Chapman and Hall/CRC, 1999.
2. D. P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, Massachusetts, 1995. Volumes 1 and 2.
3. D. A. Dolgov and E. H. Durfee. Constructing optimal policies for agents with constrained architectures. In *AAMAS*, pages 974–975, 2003.
4. D.A. Dolgov and E.H. Durfee. Approximating optimal policies for agents with limited execution resources. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1107–1112. AAAI Press, 2004.
5. E.A. Feinberg and A. Shwartz. Constrained markov decision models with weighted discounted rewards. *Math. of Operations Research*, 20(4):302–320, 1995.
6. Zoltan Gabor, Zsolt Kalmar, and Csaba Szepesvari. Multi-criteria reinforcement learning. In *Proc. 15th International Conf. on Machine Learning*, pages 197–205. Morgan Kaufmann, San Francisco, CA, 1998.
7. P. Geibel and F. Wysotzki. Risk-sensitive reinforcement learning applied to chance constrained control. *JAIR*, 24:81–108, 2005.
8. R. S. Sutton and A. G. Barto. *Reinforcement Learning – An Introduction.* MIT Press, 1998.

Efficient Non-linear Control Through Neuroevolution

Faustino Gomez[1], Jürgen Schmidhuber[1,2], and Risto Miikkulainen[3]

[1] Dalle Molle Institute for Artificial Intelligence (IDSIA), Galleria 2, Lugano, CH
[2] Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany
[3] Department of Computer Sciences, University of Texas, Austin, TX 78712 USA

Abstract. Many complex control problems are not amenable to traditional controller design. Not only is it difficult to model real systems, but often it is unclear what kind of behavior is required. Reinforcement learning (RL) has made progress through direct interaction with the task environment, but it has been difficult to scale it up to large and partially observable state spaces. In recent years, neuroevolution, the artificial evolution of neural networks, has shown promise in tasks with these two properties. This paper introduces a novel neuroevolution method called CoSyNE that evolves networks at the level of weights. In the most extensive comparison of RL methods to date, it was tested in difficult versions of the pole-balancing problem that involve large state spaces and hidden state. CoSyNE was found to be significantly more efficient and powerful than the other methods on these tasks, forming a promising foundation for solving challenging real-world control tasks.

1 Introduction

In many decision making processes such as manufacturing, aircraft control, and robotics, researchers are faced with the problem of controlling systems that are highly complex, noisy, and unstable. The problem with designing or programming controllers for such systems by conventional engineering methods is twofold: (1) The environment is often non-linear and noisy so that it is impossible to obtain the kind of accurate and tractable mathematical model required by these methods. (2) The task is complex enough that there is very little *a priori* knowledge of what constitutes a reasonable, much less optimal, control strategy.

These two problems have compelled researchers to explore methods based on reinforcement learning (RL; [1]). Instead of trying to pre-program a response to every likely situation, an agent *learns* the task by interacting with the environment. In principle, RL methods can solve these problems: they do not require a mathematical model of the environment (i.e. the state transition probabilities), and can solve many problems where examples of correct behavior are not available. However, in practice, it has turned out difficult to scale them up to large state spaces and non-Markov tasks where the state of the environment is not fully observable. This is an important challenge because the real-world is continuous (i.e. infinite number of states) and artificial agents, like natural organisms, are necessarily constrained in their ability to fully perceive their environment.

Recently, methods for evolving artificial neural networks or *neuroevolution* [2], especially those that coevolve network functional units [3, 4, 5], have shown promising results on continuous, non-Markov tasks. The method introduced in this paper, Cooperative Synapse NeuroEvolution (CoSyNE), extends the idea of coevolving network components to the level of individual synaptic weights. The goal of this paper is to compare CoSyNE to a wide range of other learning systems in a setting that is challenging yet practical. To this end, a set of pole-balancing tasks is used ranging from the familiar simple versions to versions that are extremely difficult even for the most advanced methods.

2 Cooperative Synapse NeuroEvolution (CoSyNE)

Cooperative Synapse Neuroevolution (CoSyNE) uses cooperative coevolution to construct neural networks, but unlike other methods of this type (e.g. SANE [3] and ESP [5]) it searches at the level of individual network weights rather than neurons.

```
CoSyNE($n, m, \Psi$)
 1: Initialize $\mathcal{P} = \{P_1, \ldots, P_n\}$
 2: repeat
 3:     for $j = 1$ to $m$ do
 4:         $\mathbf{x}_j \Leftarrow (x_{1j}, \ldots, x_{nj})$
 5:         Evaluate($\mathbf{x}_j, \Psi$)
 6:     end for
 7:     $\mathcal{O} \Leftarrow$ Recombine($\mathcal{P}$)
 8:     for $k = 1$ to $l$ do
 9:         $x_{i,m-k} \Leftarrow o_{ik}$
10:     end for
11:     for $i = 1$ to $n$ do
12:         permute($P_i$)
13:     end for
14: until solution is found
```

Fig. 1. The CoSyNE Algorithm

Figure 1 describes the CoSyNE procedure in pseudocode. First (line 1), a population $\mathcal{P}$ consisting of n subpopulations $P_i, i = 1..n$, is created, where n is the number of synaptic weights in the networks to be evolved, determined by a user-specified network architecture Ψ. Each subpopulation is initialized to contain m real numbers, $x_{ij} = \mathcal{P}_{ij} \in P_i, j = 1..m$, chosen from a uniform probability distribution in the interval $[-\alpha, \alpha]$. The population is thereby represented by an $n \times m$ matrix.

CoSyNE then loops through a sequence of *generations* until a sufficiently good network is found (lines 2-14). Each generation starts by constructing a complete network chromosome $\mathbf{x}_j = (x_{1j}, x_{2j}, \ldots, x_{nj})$ from each row in $\mathcal{P}$. The m resulting chromosomes are transformed into networks by assigning their weights to their corresponding synapses, in Ψ.

After all of the networks have been evaluated (line 5) and assigned a fitness, the top quarter with the highest fitness (i.e. the parents) are recombined (line 7) using crossover and mutation. Recombination produces a pool of offspring $\mathcal{O}$ consisting of l new network chromosomes $\mathbf{o}_k$, where $o_{ik} = \mathcal{O}_{ik} \in O_i, k = 1..l$. The weights in each of the offspring chromosomes are then added to $\mathcal{P}$ by replacing the least fit weights in their corresponding subpopulation (lines 8-10).

At this point the algorithm functions as a conventional neuroevolution system that evolves complete network chromosomes. In order to *coevolve* the synaptic weights, the subpopulations are permuted (lines 11-13) so that each weight forms part of a potentially different network in the next generation.

Permuting the subpopulations increases diversity by allowing CoSyNE to sample networks that would not be generated through recombination alone. This means that which weights are retained in the population from one generation to the next is not determined only by which networks scored well in the previous generation, but rather by a broader sampling of the possible m^n networks that can be formed by selecting a weight from each subpopulation. More precisely, each generation the offspring lie within a subspace that is defined by all possible applications of the genetic operators to the set of parents (i.e. the subspace *spanned* by the parents). Each successive generation produces offspring from a subspace contained within the previous one (except for some sampling outside due to mutation), as the population converges to virtually a single search point. With permutation, points can be sampled outside of this subspace by forming networks from weights not found in the current set of parents. The overall effect is to make the algorithm less greedy because weights have a chance to reproduce even if they were not part of the parent chromosomes in previous generations.

Cooperative coevolution in general can delay convergence through this process, but because CoSyNE evolves at the lowest possible level of granularity (the individual parameter) the number of possible networks m^n is maximized. Therefore, there are a maximum number of ways to sample outside of the parent subspace and delay convergence, allowing CoSyNE more time to put the pieces together to form a good network.

The basic CoSyNE framework does not specify how the weights are grouped in the chromosomes (i.e. which entry in the chromosome corresponds to which synapse) or which genetic operators are used. In the implementation used in this paper, the weights of each neuron are grouped together (i.e. form a substring) and are separated into input, output, and recurrent weight segments. For the genetic operators we use multi-point crossover where 1-point crossover is applied to each neuron segment of the chromosome is used to generate the offspring, and mutation where each weight in $\mathcal{P}$ has a small probability of being changed to a new value chosen at random from the initial weight range $[-\alpha, \alpha]$.

3 Experiments in Pole-Balancing

CoSyNE was compared experimentally to a broad range of learning algorithms on a sequence of increasingly difficult versions of the pole-balancing task. The basic pole-balancing system consists of a pole hinged to a wheeled cart on a finite stretch of track that must be balance by applying a force to the cart at regular intervals. Although this task has been a popular artificial learning testbed for over 30 years, it turns out that the basic pole-balancing problem can be solved easily by random search. To make the problem more challenging, four task configurations of increasing difficulty (due to [6]) were used: one pole with complete state information (1a) and incomplete state information (1b), and two poles with complete state information (2a) and incomplete state information (2b). Task 1a is the classic one-pole configuration where the controller receives all four state variables: the position and velocity of the cart $(x, \dot{x})$, and the angle and angular velocity of the pole $(\theta_1,\dot{\theta}_1)$. In 1b, the controller only has access to x and θ_1; it does not receive the velocities $(\dot{x}, \dot{\theta}_1)$. In 2a, the system now has a second pole $(\theta_2, \dot{\theta}_2)$ next to the first, making the state-space six-dimensional, and non-linear. Task 2b, like 1b, is non-Markov with the controller only seeing x, θ_1, and θ_2. Fitness was determined by the number of time steps a network could keep both poles within a specified failure angle from vertical and the cart between the ends of the track. The failure angle was 12° and 36° for the one and two pole tasks, respectively. The initial angle of the long pole was set to 4° from vertical for all trials. A task was considered solved if a network could balance the pole(s) for 100,000 time steps, which is equal to over 30 minutes in simulated time. CoSyNE evolved networks with one hidden unit, 20 weights per subpopulation for the one-pole tasks, and 30 weights for the two-pole tasks. Mutation was set to 5% in all of the experiments. All simulations were run on a 1.50GHz Intel Xeon.

The pole-balancing environment was implemented using a realistic physical model with friction, and fourth-order Runge-Kutta integration with a step size of 0.01s (see [6] for the equations of motion and parameters used). At each time-step (0.02 seconds of simulated time) the network receives the state variable values scaled to [-1.0, 1.0]. This input activation is propagated through the network to produce a signal from the output unit that represents the amount of force used to push the cart. The force is then applied and the system transitions to the next state, which becomes the new input to the controller. This cycle is repeated until a pole falls or the cart goes off the end of the track.

3.1 Other Methods

CoSyNE was compared to 14 other learning methods: seven *single-agent* and seven *evolutionary* methods. Due to space limitations, the reader is referred to the original papers and [5] for parameter settings and implementation details.
Single-Agent Methods

Random Weight Guessing (RWG) where the network weights are chosen at random (i.d.d) from a uniform distribution. This approach gives us an idea of how difficult each task is to solve by simply guessing a good set of weights.
Policy Gradient RL (PGRL; [7]) where sampled Q-values are used to differentiate the performance of the policy with respect to its parameters. The policy was implemented by a feed-forward network (FNN) with one hidden layer.
Value and Policy Search (VAPS; [8]) extends the work of Baird et al. [9] to policies that can make use of memory. The algorithm uses stochastic gradient descent to search the space of finite policy graph parameters.
Q-learning with MLP (Q-MLP)**:** The basic Q-learning algorithm [10] using a an FNN trained with backpropagation to map state–action pairs to Q-values.

Sarsa(λ) with Case-Based function approximator (SARSA-CABA; [11])**:** This method uses on-policy Temporal Difference control with eligibility traces that uses a case-based memory to approximate the Q-function.

Sarsa(λ) with CMAC function approximator (SARSA-CMAC; [11])**:** This method is the same as SARSA-CABA except that it uses a Cerebellar Model Articulation Controller instead of a case-based memory.

Adaptive Heuristic Critic (AHC; [12])**:** uses a learning agent composed of two components: an *actor* (policy) and a *critic* (value-function), both implemented using an FNN trained with a variant of backpropagation.

Evolutionary Methods

Symbiotic, Adaptive Neuro-Evolution (SANE; [3]) is a cooperative coevolutionary method that evolves neurons in a single population.

Conventional Neuroevolution (CNE) is our implementation of single-population neuroevolution similar to the algorithm used in [6], where each chromosome in the population represents a complete neural network.

Evolutionary Programming (EP; [13]) is a general mutation-based evolutionary method that can be used to search the space of neural networks.

Cellular Encoding (CE; [14]) uses Genetic Programming to evolve graph-rewriting programs that control how neural networks are constructed.

Covariance Matrix Adaptation Evolutionary Strategies (CMA-ES; [15]) evolves the covariance matrix of the mutation operator in evolutionary strategies. The results in the pole-balancing domain were obtained by Igel [16].

NeuroEvolution of Augmenting Topologies (NEAT; [17]) is a neuroevolution method that evolves topology as well as synaptic weights.

Enforced SubPopulation (ESP; [5]) cooperatively coevolves neurons in a separate subpopulation for each network unit.

For Q-MLP, SANE, CNE, ESP, NEAT, and CoSyNE, experiments were run using our own code. For PGRL, AHC, SARSA, publicly available code from [18], [12], and [11], was used respectively, modified for the pole-balancing domain. For VAPS, EP, CMA-ES, and CE, the results were taken from the papers cited above.

3.2 Results

Balancing one pole is a relatively easy problem that gives us a base performance measurement before moving on to the much harder two-pole task. It has also been solved with many other methods and therefore serves to put the results in perspective with prior literature. Table 1 shows the results for the this task for both complete and incomplete state information.

The results for task 1a show that simply choosing weights at random (RWG) is sufficient to solve this task efficiently. CoSyNE was the only method that solved the task in fewer evaluations. The other single-agent methods were all significantly slower than the evolutionary methods, especially in terms of CPU time. Depending on the kind of function approximator, the amount of computation required to evaluate and update the value-functions used by AHC, SARSA, and Q-learning can prove costly.

In contrast, the evolutionary methods do not update any agent parameters during interaction with the environment and only need to evaluate a function approximator once per state transition since the policy is represented explicitly.

Task 1b is notably harder since in addition to controlling the system, the concomitant problem of velocity calculation must also be solved. Despite considerable effort, we were unable to solve this task with AHC and PGRL.

To make Q-MLP and the SARSA methods effective, their inputs were extended to include also the immediately previous cart position, pole angle, and action $(x_{t-1}, \theta_{t-1}, a_{t-1})$. This *delay window* of depth 1 is sufficient to disambiguate process states.

Table 1. Results for balancing one pole. Average of 50 simulations. All differences are statistically significant ($p < 0.01$).

Method	with velocities		w/out velocities	
	Evals	CPU	Evals	CPU
VAPS	—	—	(500k)	(5days)
AHC	189,500	95	failed	
PGRL	28,779	1,163	failed	
Q-MLP	2,056	53	11,311	340
SARSA-CABA	965	1,713	15,617	6,754
SARSA-CMAC	540	487	13,562	2,034
NEAT	743	7	1,523	6
CNE	352	5	724	5
SANE	302	5	1,212	6
ESP	289	4	589	5
CMA-ES	283	—	—	—
RWG	199	2	8,557	3
CoSyNE	98	1	127	2

VAPS is the slowest method in this comparison, with the single reported run (in parentheses) only balancing the pole for around 1 minute of simulated time after several days of computation [8]. Results for the SARSA methods are the average of successful runs only. Of the single-agent methods Q-MLP fared the best, reliably solving the task and doing so much more rapidly than SARSA.

The performance of the six evolutionary methods degrades only slightly compared to the previous task. CoSyNE, CNE, and ESP were two orders of magnitude faster than VAPS and SARSA, one order of magnitude faster than Q-MLP, and approximately twice as fast as SANE and NEAT. CoSyNE was able to balance the pole for over 30 minutes of simulated time usually within 2 seconds of learning CPU time, and do so reliably.

Table 2. Two poles with velocities Average of 50 simulations. EP results taken from [13], CMA-ES from [16]. All differences are statistically significant ($p < 0.001$) except the number of evaluations for NEAT and ESP.

Method	Evaluations	CPU time
RWG	474,329	70
EP	307,200	—
CNE	22,100	73
SANE	12,600	37
Q-MLP	10,582	153
NEAT	3,600	31
ESP	3,800	22
CoSyNE	954	4
CMA-ES	895	—

Balancing two poles represents a significant jump in difficulty. For task 2a, CoSyNE was compared with Q-MLP, CNE, SANE, NEAT, ESP, and the published results of EP and CMA-ES. Despite extensive experimentation with many different parameter settings, we were unable to get the SARSA methods to solve this task within 12 hours of computation.

Table 2 shows the results for this task. Q-MLP compares very well to the evolutionary methods with respect to evaluations, in fact, better than on task 1b, but again lags behind SANE, NEAT, and ESP by nearly an order of magnitude in CPU time. ESP and NEAT are statistically even in terms of evaluations, requiring roughly three times fewer evaluations than SANE. CMA-ES required the fewest number of evaluations, just edging out CoSyNE.

For task 2b, none of the single-agent methods made noticeable progress after 12 hours of computation. Therefore, only neuroevolution methods are compared. To allow for a comparison with CE, controllers were evolved using both the standard fitness function used in the previous tasks, and a the "damping" fitness function (used in [14]) that prevents controllers from solving the task by simply swinging the poles back and forth.

Table 3 compares the six neuroevolution methods for both fitness functions. To determine when the task was solved for the damping fitness function, the best controller from each generation was tested using the standard fitness to see if it could balance the poles for 100,000 time steps. The results for CE are in parentheses in the table because only a single run was reported in [14].

Table 3. Two poles without velocities. Average of 50 simulations. All results are statistically significant except for the difference between ESP and NEAT using the standard fitness.

Method	Evaluations	
	Standard fit.	Damping fit.
RWG	415,209	1,232,296
CE	—	(840,000)
SANE	262,700	451,612
CNE	76,906	87,623
ESP	7,374	26,342
NEAT	6,929	24,543
CMA-ES	3,521	6,061
CoSyNE	1,249	3,416

Using the damping fitness, ESP, CNE, NEAT, and CoSyNE required an order of magnitude fewer evaluations than SANE and CE. ESP and NEAT were three times faster than CNE using either fitness function, with CNE failing to solve the task about 40% of the time. CoSyNE was the most efficient method for both fitness measures, outperforming ESP and NEAT by a factor of six on the standard fitness, and CMA-ES by a factor of two on the damping fitness.

4 Discussion and Conclusion

The comparison results show that the evolutionary methods are more efficient than the single-agent methods in this set of tasks. The best single-agent method in task 1a required an order of magnitude more CPU time than NE, and the transition from 1a to 1b represented a significant challenge, causing some of them to fail and others to take 30 times longer than NE. Only Q-MLP was able to solve task 2a and none of the single-agent methods could solve task 2b. In contrast, all of the evolutionary methods scaled up to the most difficult tasks, with CoSyNE beating the next fastest method by a wide margin.

The most challenging task exhibits many of the dimensions of difficulty found in real-world control problems: (1) continuous state and action spaces, (2) partial observability, and (3) non-linearity. The first two are problematic for conventional reinforcement learning methods because they either complicate the representation of the value function or the access to it. Neuroevolution deals with them by evolving recurrent networks; the networks can compactly represent arbitrary temporal, non-linear mappings. The success of CoSyNE on tasks of this complexity suggests that it can be applied to the control of real systems that manifest similar properties—specifically, non-linear, continuous systems such as aircraft control, satellite detumbling, and robot bipedal walking.

Acknowledgments

This research was funded by the following grants: NSF EIA-0303609, NSF IIS-0083776, THECB (Texas Higher Education Coordinating Board) ARP-003658-476-2001, and CSEM Alpnach and the EU MindRaces project: FP6 511931.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)
2. Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE **87**(9) (1999)
3. Moriarty, D.E.: Symbiotic Evolution of Neural Networks in Sequential Decision Tasks. PhD thesis, University of Texas at Austin (1997) Tech. Rep. UT-AI97-257.
4. Potter, M.A., De Jong, K.A.: Evolving neural networks with collaborative species. In: Proceedings of the 1995 Summer Computer Simulation Conference. (1995)
5. Gomez, F.J.: Robust Nonlinear Control through Neuroevolution. PhD thesis, University of Texas at Austin (2003) Tech. Rep. AI-TR-03-303.
6. Wieland, A.: Evolving neural network controllers for unstable systems. In: Proceedings of the International Joint Conference on Neural Networks (Seattle, WA), Piscataway, NJ: IEEE (1991) 667–673
7. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems 12. Volume 12., MIT Press (2000) 1057–1063
8. Meuleau, N., Peshkin, L., Kim, K.E., Kaelbling, L.P.: Learning finite state controllers for partially observable environments. In: 15th International Conference of Uncertainty in AI. (1999)
9. Baird, L.C., Moore, A.W.: Gradient descent reinforcement learning. In: Advances in Neural Information Processing Systems 12. (1999)
10. Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning **8**(3) (1992) 279–292
11. Santamaria, J.C., Sutton, R.S., Ram, A.: Experiments with reinforcement learning in problems with continuous state and action spaces. Adaptive Behavior **6**(2) (1998)

12. Anderson, C.W.: Strategy learning with multilayer connectionist representations. Technical Report TR87-509.3, GTE Labs, Waltham, MA (1987)
13. Saravanan, N., Fogel, D.B.: Evolving neural control systems. IEEE Expert (1995)
14. Gruau, F., Whitley, D., Pyeatt, L.: A comparison between cellular encoding and direct encoding for genetic neural networks. NC-TR-96-048, NeuroCOLT (1996)
15. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation **9**(2) (2001) 159–195
16. Igel, C.: Neuroevolution for reinforcement learning using evolution strategies. In Proceedings of the Congress on Evolutionary Computation. IEEE (2003)
17. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10** (2002) 99–127
18. Grudic, G.: http://www.cis.upenn.edu/ grudic/PGRLSim/ (2000)

Efficient Prediction-Based Validation for Document Clustering

Derek Greene and Pádraig Cunningham

University of Dublin, Trinity College,
Dublin 2, Ireland
{derek.greene, padraig.cunningham}@cs.tcd.ie

Abstract. Recently, stability-based techniques have emerged as a very promising solution to the problem of cluster validation. An inherent drawback of these approaches is the computational cost of generating and assessing multiple clusterings of the data. In this paper we present an efficient *prediction-based* validation approach suitable for application to large, high-dimensional datasets such as text corpora. We use kernel clustering to isolate the validation procedure from the original data. Furthermore, we employ a *prototype reduction* strategy that allows us to work on a reduced kernel matrix, leading to significant computational savings. To ensure that this condensed representation accurately reflects the cluster structures in the data, we propose a *density-biased* strategy to select the reduced prototypes. This novel validation process is evaluated on real-world text datasets, where it is shown to consistently produce good estimates for the optimal number of clusters.

1 Introduction

The task of evaluating the output of a clustering algorithm, referred to as cluster validation, is a fundamental problem in unsupervised learning. One common application of validation is in the identification of suitable values for algorithm parameters, such as the optimal number of clusters $\hat{k}$. Internal validation indices, which make assessments based on intrinsic properties of the raw data, have frequently been used for this task in the past [1]. However, these tend to be model dependent in the sense that they make assumptions about the structure of clusters in data [2]. On the other hand, *external* validation techniques, which assess the degree to which a clustering corresponds to the "natural classes" in the data, are not directly applicable for parameter selection since external knowledge will typically be unavailable during the clustering process.

Validation techniques based on stability analysis have recently been shown to be particularly effective in determining the optimal number of clusters in data [2]. The *stability* of a clustering model refers to its ability to consistently replicate similar solutions on data originating from the same source. In practice, this involves repeatedly clustering subsamples of the original dataset. A high level of agreement between the clusterings indicates that the model is appropriate for the data. A related approach for estimating $\hat{k}$ was proposed by Tibshirani *et al.* [3], which was motivated by the concept of prediction accuracy in supervised learning. This *prediction-based* scheme assesses the stability of a clustering model by measuring the degree to which the model allows us to consistently construct a classifier on a training set that will successfully predict the assignment of objects in a clustering of a corresponding test set.

Prototype reduction techniques have been extensively used in supervised learning for tasks involving large datasets. These techniques are concerned with producing a minimal set of objects or prototypes to represent the data, while ensuring that a classifier applied to this set will perform approximately as well as on the original dataset. In the literature, these techniques are generally divided into two categories: *prototype selection* techniques seek to identify a subset of representative objects from the original data, while *prototype extraction* techniques involve the creation of an entirely new set of objects. Most work in prototype reduction has focused on supervised learning tasks, although the concept has been used implicitly as part of many clustering algorithms (*e.g.* [4]). A comprehensive overview of supervised reduction schemes is provided in [5].

A significant drawback of stability-based validation methods is the computational cost of generating and comparing multiple clusterings. Consequently, these methods have rarely been applied to sparse, large-scale datasets such as text corpora. In this paper, we tackle these issues by proposing an efficient prediction-based validation scheme. Our approach makes use of kernel clustering methods so that we no longer need to generate clusterings in the original high-dimensional space. Furthermore, we propose a novel *density-biased* prototype reduction strategy that allows us to construct a condensed kernel matrix, leading to substantial efficiency improvements. Our evaluation on text data shows that this strategy results in a 16-20 fold speed-up without significantly impacting upon the validation procedure's ability to correctly identify $\hat{k}$. Note that an extended version of this paper is available as a technical report with the same title [6].

2 Proposed Method

For small datasets, stability-based validation techniques offer an attractive option for inferring a value for $\hat{k}$. However, as the number of dimensions grows, the time required to repeatedly apply an algorithm such as standard k-means will greatly increase. The number of objects n will also be a limiting factor, as a larger value for n will substantially increase the computational cost of the clustering and the stability assessment procedures, which typically run in $O(n^2)$ time or slower. To address these issues, we now present an efficient prediction-based validation method suitable for use on text corpora.

2.1 Kernel-Based Stability Analysis

To avoid having to work in the original high-dimensional feature space, we make use of recently proposed kernel clustering methods. A kernel function is usually represented by an $n \times n$ kernel matrix $\mathbf{K}$, where K_{ij} indicates the affinity between objects x_i and x_j. The advantage of using kernel methods in the context of stability analysis stems from the fact that, having constructed a single kernel matrix, we may subsequently generate multiple partitions without referring back to the original data. As the standard k-means algorithm has commonly been used in both stability analysis and document clustering, we focus here on the use of the corresponding kernelised k-means algorithm.

To form the basis for our validation scheme, we choose the prediction-based method proposed by Tibshirani *et al.* [3] due to its empirical success and computational advantage over other stability analysis methods. The latter benefit derives from the fact that each run of the clustering algorithm works on a sample of $\frac{n}{2}$ objects only. Formally, the validation process involves applying two-fold cross-validation to randomly split the dataset $\mathcal{X} = \{x_1, \ldots, x_n\}$ into disjoint training and test sets, denoted $\mathcal{X}_a$ and $\mathcal{X}_b$ respectively. Both sets are subsequently clustered using kernel k-means with random initialisation. A prediction for the assignment of objects in the test set is then produced by assigning each $x_i \in \mathcal{X}_b$ to the nearest centroid in training clustering. The accuracy of this prediction is assessed by measuring the degree to which it agrees with the original clustering of $\mathcal{X}_b$. For this task, we employ an adjusted version of the *prediction strength* measure described in [3]. However, rather than heuristically choosing from among the potential values, we select the value k that leads to the maximum average score over τ runs. Since prediction strength exhibits a bias toward smaller values of k, we employ the widely-used adjustment technique described in [7] to correct for chance agreement.

As discussed in [2], the choice of classifier used to make predictions should complement the clustering algorithm. To "mimic" the assignment behaviour of the kernel k-means algorithm, we employ a kernel nearest centroid classifier, such that each object in $\mathcal{X}_b$ is classified as being a member of the class represented by the nearest kernel pseudo-centroid in the training clustering. Subsequently, we use corrected prediction strength to evaluate the degree to which the predicted classification agrees with the clustering of $\mathcal{X}_b$ as produced by kernel k-means.

2.2 Kernel Reduction

In the previous section, we described a stability-based validation method suitable for use on high-dimensional data. However, the validation process still requires τ runs consisting of clustering and prediction assessment phases, which both run in $O((\frac{n}{2})^2)$ time. Clearly, decreasing n will make the process significantly less computationally expensive. Motivated by the large-scale clustering technique described in [4], an intuitive solution is to create a reduced set of $n' < n$ objects, upon which the validation procedure may be applied. However, any such reduction must be performed in a way that preserves the structures in the data.

Many supervised prototype reduction approaches process each class separately. As a result, the reduced prototypes will be "meaningful" in the sense that they will represent regions from a single class only. In the absence of class labels, we must rely upon intrinsic properties of the data to ensure that all structures in the data are adequately represented. Unfortunately, text corpora often contain unbalanced cluster sizes, which may also differ in their relative densities, making the task particularly problematic. To address these issues, we propose a reduction scheme consisting of two phases. In the first phase, prototype extraction is used to generate a set of candidate prototypes formed from small homogeneous regions of the data. The second phase selects from among these a subset of n' prototypes to build a reduced $n' \times n'$ kernel matrix, denoted by $\mathbf{K}'$.

Firstly, we create a set of extracted prototypes $\mathcal{S} = \{s_1, \ldots, s_n\}$ in a manner similar to that employed by the supervised BTS reduction scheme [8], where new prototypes are formed by locally combining subsets of the original dataset $\mathcal{X}$. Formally, we define a *neighbourhood* $\mathcal{N}_i$ as a subset of $\mathcal{X}$ consisting of a seed object x_i together with its set of p nearest neighbours. A new prototype s_i may be constructed from the mean of these $p+1$ objects. Since we wish to work in the kernel-induced space only, we consider s_i to be the pseudo-centroid of the subset $\mathcal{N}_i$ as calculated from the values in $\mathbf{K}$. We note that, as regions forming cluster structures will normally be locally homogeneous, the majority of the set of neighbours of each object are likely belong to the same cluster as that object. Therefore, prototypes constructed from the centroid of sufficiently small neighbourhoods will generally be representative of a single natural class.

The problem remains of selecting a subset $\mathcal{S}'$ of n' optimal prototypes from the n possible candidates. A possible solution is to apply unbiased random sampling to choose $\mathcal{S}'$. However, this approach has several drawbacks in the context of validation. Ideally, we wish to select a fraction of prototypes from each class that is proportional to the size of that class in the original data. A single random sample from $\mathcal{S}$ is not guaranteed to achieve this. To illustrate this problem, we consider a small subset of the well-known 20 newsgroups collection, consisting of 300 documents from the 'cryptography' group and 150 documents from the 'hockey' group. Figures 1(a) and 1(b) respectively show the block-ordered matrices corresponding to the full kernel matrix and a reduced matrix produced by randomly selecting seed objects. From the latter, it is evident that the smaller 'hockey' class is not adequately represented by the random reduction process. We observe that reduced prototypes chosen in this way frequently fail to produce a true proxy for the dataset, resulting in poor estimations for $\hat{k}$ in the subsequent validation process. In these cases, the failure is often due to poor sampling of smaller clusters or important sub-regions within clusters. While we could run the process multiple times and aggregate the results, the computational cost would typically negate the benefits of performing prototype reduction.

As an alternative, in the second phase of our reduction procedure we employ a density-biased strategy to select $\mathcal{S}'$. This procedure has similar goals to existing density-biased sampling techniques (*e.g.* [9]), but is deterministic and significantly less computationally demanding. Firstly, we define the *compactness* of a neighbourhood $\mathcal{N}_a$ as the average of the pairwise affinities between its members:

$$C(\mathcal{N}_a) = \frac{\sum_{x_i, x_j \in \mathcal{N}_a} K_{ij}}{|\mathcal{N}_a|^2} \tag{1}$$

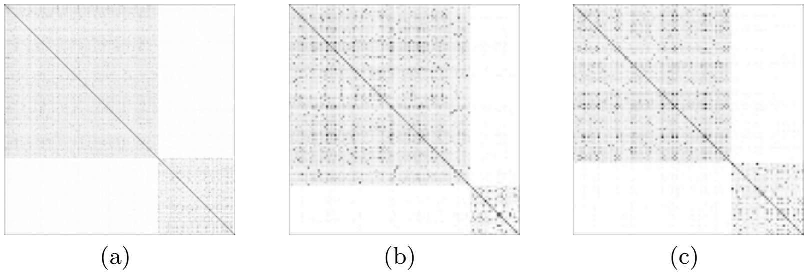


Fig. 1. Gram matrix for (a) full kernel; (b) kernel reduced by random sampling; (c) kernel reduced by density-biased selection

where $|\mathcal{N}_a| = p + 1$. This is equivalent to the "self-similarity" of the pseudo-centroid formed from $\mathcal{N}_a$. In the selection process, the prototypes in $\mathcal{S}$ are ranked in descending order according to their compactness. From these, we uniformly choose $n' = \frac{n}{\rho}$ prototypes, where ρ is the *reduction rate* that determines the degree to which the number of data objects should be reduced. Specifically, we select every ρ-th prototype from the ordered list, thereby ensuring that we represent all density patterns in the data. We then build the reduced kernel matrix $\mathbf{K}'$ based on these n' prototypes. Rather than computing explicit representations for the new prototypes in the original feature space, we can make use of the values in the original kernel matrix to directly construct $\mathbf{K}'$. Formally, the affinity between a pair of reduced prototypes s_i and s_j is defined as:

$$K'_{ij} = \frac{\sum_{x_a \in S_i, x_b \in S_j} K_{ab}}{(p+1)^2} \tag{2}$$

While it is possible that a matrix constructed in this way may not always be positive semi-definite, it has previously been shown in [10] that this does not pose a significant problem for the kernel k-means algorithm.

Referring back to our previous example, we see that, unlike the matrix in Figure 1(b), the reduced kernel matrix shown in Figure 1(c) is clearly representative of the two classes in the original dataset. In practice, we consistently observe that this density-biased selection strategy produces a set of extracted prototypes that accurately summarise the underlying structures in the data. We contend that this is due to the inclusion of regions representing clusters of varying densities and all sub-regions within those clusters.

Once we have constructed the reduced kernel matrix, the validation scheme proceeds as described in Section 2.1. The proposed reduction strategy results in a significant decrease in the computational cost of the validation process. Our approach does involve a once-off initialisation step, requiring time $O(n \log n)$ for prototype extraction and $O(n'^2 p^2)$ for the construction of $\mathbf{K}'$. However, the computational gains made in the subsequent validation process are substantial. For each of the τ runs, the costs associated with clustering and prediction assessment are both reduced to $O((\frac{n}{2\rho})^2)$.

2.3 Application to Document Clustering

While our proposed method may be used in conjunction with any valid kernel function, for document clustering we make use of a linear kernel that has been normalised according to the approach described in [11]. This results in a kernel matrix that is equivalent to that produced by the standard cosine similarity measure. Since this matrix will often suffer from *diagonal dominance* [10], we address the problem by applying a negative shift to the diagonal of the kernel matrix so as to minimise its trace. A summary of the complete validation process is provided in Figure 2.

Initialisation Phase

- Construct full $n \times n$ kernel matrix $\mathbf{K}$ using the original data.
- Extract candidate prototypes $\mathcal{S}$, consisting of n neighbourhood pseudo-centroids.
- Evaluate compactness of candidates in $\mathcal{S}$ and sort accordingly in descending order.
- Uniformly select subset $\mathcal{S}'$ of n' reduced prototypes from the ordered list.
- Construct the $n' \times n'$ reduced kernel matrix $\mathbf{K}'$ based on $\mathcal{S}'$ using Eqn. 2.
- Apply zero-trace diagonal shift to $\mathbf{K}'$.

Validation Phase

- Produce τ splits of $\mathcal{S}'$ into training and test sets.
- For each value of $k \in [k_{min}, k_{max}]$:
  1. For each split $(\mathcal{X}_a, \mathcal{X}_b)$:
     (a) Apply kernel k-means to training set $\mathcal{X}_a$ using kernel $\mathbf{K}'$.
     (b) Predict the assignment of documents in $\mathcal{X}_b$ based on centroids from clustering of $\mathcal{X}_a$.
     (c) Apply kernel k-means to test set $\mathcal{X}_b$ using kernel $\mathbf{K}'$.
     (d) Evaluate prediction strength and correct for chance.
  2. Compute mean corrected prediction strength for k.
- Estimate $\hat{k}$ by selecting candidate k with highest mean prediction strength.

Fig. 2. Complete kernel prediction-based validation scheme, with prototype reduction

As mentioned previously, we assume that regions will generally be locally homogeneous, which should be the case when an appropriate kernel function is chosen. To maximise homogeneity, we select a low value for the number of neighbours (*e.g.* $p = 5$). Empirical evidence suggests that a value of $\rho = 4$ for the reduction rate substantially reduces the time required for the validation process, without significantly affecting its accuracy. The selection of ρ is also related to the maximum number of runs τ. The computational gains resulting from prototype reduction facilitate the use of a larger value (*e.g.* $\tau = 200$) to guarantee the robustness of the overall validation procedure. It must be stressed that, in practice, the use of these general purpose parameter values proved to be effective on a diverse range of datasets, indicating that the proposed validation method is quite robust to the choice of values for these parameters. This allows us to focus on the more immediate task of selecting the number of clusters.

3 Empirical Evaluation

In this section, we evaluate the newly proposed validation scheme on a set of real-world corpora that have previously been used in document clustering. For further details on these corpora and a comprehensive evaluation on artificial data, consult [6]. The experimental process involved applying four prediction-based validation schemes to each corpus across a reasonable range of k values ($k_{min} = 2$, $k_{max} = 10$) and comparing their output with the "true" number of natural classes $\hat{k}$. The first pair of schemes employ the standard k-means algorithm with cosine similarity. In the former (KM-S), prediction evaluations are made using *prediction strength* corrected for chance agreement. In the latter (KM-P), evaluations are made using the *partition similarity* criterion as described in [12]. The second pair of schemes are those introduced in this paper: kernel k-means with prediction strength (KKM-S), and kernel k-means with prediction strength after prototype reduction (RED-S).

Table 1. Summary of top-3 estimations for $\hat{k}$ on real datasets

Dataset	$\hat{k}$	KM-S	KM-P	KKM-S	RED-S
bbc	5	**5**, 4, 6	**5**, 6, 7	**5**, 6, 4	**5**, 6, 4
bbcsport	5	4, **5**, 3	**5**, 6, 4	**5**, 6, 4	**5**, 4, 6
classic3	3	**3**, 2, 4	**3**, 2, 4	**3**, 4, 5	**3**, 4, 5
classic	4	3, 5, 2	3, 5, 2	5, **4**, 2	5, **4**, 2
cstr	4	3, 2, **4**	3, **4**, 2	3, **4**, 5	3, **4**, 5
ng17-19	3	5, 4, 6	5, 4, 6	5, 4, **3**	4, 5, **3**
ng3	3	**3**, 4, 2	**3**, 4, 5	**3**, 4, 2	**3**, 2, 4
reviews	5	2, 3, 6	2, 8, 9	2, **5**, 4	2, 6, 3

Table 1 shows the results of the comparison, indicating the top three estimated values for $\hat{k}$ on the real corpora. In almost all cases, the reduced validation method (RED-S) recommended the same value of k as that chosen when validating based on the full kernel matrix (KKM-S). Only in the case of the *reviews* dataset, which contains significantly overlapping clusters, did it fail to rate $\hat{k}$ among its top three choices. However, KM-S and KM-P also performed poorly on this corpus. It is interesting to note that both kernel-based techniques consistently outperformed those employing standard k-means. In these cases, our evaluations indicate that the application of diagonal dominance reduction prior leads to a non-trivial improvement in validation accuracy.

We observed that, when using kernel-based validation, the application of prototype reduction with $\rho = 4$ resulted in a 16-20 fold decrease in the time required for the entire process. For example, on a standard Pentium IV desktop computer, selecting a value for k on the *bbc* corpus took 55 minutes when using the full kernel matrix, while the same procedure using RED-S took only 3 minutes. In general, RED-S was significantly faster than any of the other strategies considered.

4 Conclusion

We have proposed a practical approach to stability-based validation suitable for the task of estimating the number of clusters in large, high-dimensional datasets such as text corpora. The use of kernel clustering methods allows us to work on a single kernel matrix rather than repeatedly computing distances in the original feature space. Moreover, we have demonstrated that we can significantly decrease the computational demands of the validation process by employing a form of prototype reduction to construct a reduced kernel matrix. Experimental evaluations have shown this validation process to be effective on real-world text corpora, where it consistently produced good estimates for the optimal number of clusters, often outperforming existing methods that are significantly more computationally expensive.

While we have particularly focused on validation in document clustering, we believe that our approach is applicable for a wide variety of other domains and kernel functions, where large datasets would otherwise make stability analysis unfeasible.

References

1. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall, Inc. (1988)
2. Lange, T., Roth, V., Braun, M.L., Buhmann, J.M.: Stability-based validation of clustering solutions. Neural Comput. **16** (2004) 1299–1323
3. Tibshirani, R., Walther, G., Botstein, D., Brown, P.: Cluster validation by prediction strength. Technical Report 2001-21, Stanford University (2001)
4. Cutting, D.R., Pedersen, J.O., Karger, D., Tukey, J.W.: Scatter/gather: A cluster-based approach to browsing large document collections. In: Proc. 15th Annual International ACM/SIGIR Conference. (1992) 318–329
5. Bezdek, J.C., Kuncheva, L.: Nearest prototype classifier designs: An experimental study. International Journal of Intelligent Systems **16** (2001) 1445–1473
6. Greene, D., Cunningham, P.: Efficient prediction-based validation for document clustering. Technical Report CS-2006-22, Trinity College Dublin (2006)
7. Hubert, L.J., Arabie, P.: Comparing partitions. Journal of Classification **2** (1985) 193–218
8. Hamamoto, Y., Uchimura, S., Tomita, S.: A bootstrap technique for nearest neighbor classifier design. IEEE Tran. Pattern Anal. Machine Intell. **19** (1997) 73–79
9. Palmer, C.R., Faloutsos, C.: Density biased sampling: an improved method for data mining and clustering. In: Proc. 2000 ACM SIGMOD International Conference on Management of Data. (2000) 82–92
10. Greene, D., Cunningham, P.: Practical solutions to the problem of diagonal dominance in kernel document clustering,. In: Proc. 23rd International Conference on Machine Learning. (2006)
11. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press (2001)
12. Giurcaneanu, C., Tabus, I.: Cluster structure inference based on clustering stability with applications to microarray data analysis. EURASIP Journal on Applied Signal Processing **1** (2004) 64–80

On Testing the Missing at Random Assumption

Manfred Jaeger

Institut for Datalogi, Aalborg Universitet, Fredrik Bajers Vej 7E, DK-9220 Aalborg Ø
jaeger@cs.aau.dk

Abstract. Most approaches to learning from incomplete data are based on the assumption that unobserved values are missing at random (mar). While the mar assumption, as such, is not testable, it can become testable in the context of other distributional assumptions, e.g. the naive Bayes assumption. In this paper we investigate a method for testing the mar assumption in the presence of other distributional constraints. We present methods to (approximately) compute a test statistic consisting of the ratio of two profile likelihood functions. This requires the optimization of the likelihood under no assumptions on the missingness mechanism, for which we use our recently proposed AI & M algorithm. We present experimental results on synthetic data that show that our approximate test statistic is a good indicator for whether data is mar relative to the given distributional assumptions.

1 Introduction

Most commonly used statistical learning methods are based on the assumption that missing values are *missing at random (mar)* [7]. For many datasets this assumption is not completely realistic. However, even when there are doubts as to the exact validity of the *mar* assumption, pragmatic considerations often lead one to adopt *mar*-based techniques like the ubiquitous EM algorithm. To help decide whether a method like EM should be applied, it would be very valuable to know whether the data at hand is *mar* or not. In this paper we investigate a method for performing statistical tests for *mar*.

We have to start with a caveat: *mar* is not testable [2,6]. The exact technical content behind this statement has to be interpreted carefully: it only says that the *mar*-assumption per se – without any further assumptions about the data – cannot be refuted from the data. However, in many machine learning scenarios certain distributional assumptions about the data are being made. For example, when learning a Naive Bayes model, then the specific independence assumptions underlying Naive Bayes are made. As pointed out in [4], the *mar*-assumption can become refutable in the context of such existing assumptions on the underlying complete data distribution.

In this paper we show that a suitably defined likelihood ratio provides a test statistic that allows us to discriminate between *mar* and non-*mar* models relative to restricted parametric models for the complete data distribution. A crucial component in the computation of the likelihood ratio is the optimization of the likelihood under no assumptions on the coarsening mechanism. For this purpose we employ our recently introduced AI&M procedure [5].

2 The Likelihood Ratio Statistic

We shall work with the coarse data model [3], which allows to consider other forms of incompleteness than missing values. In the coarse data model the *mar*-assumption has its counterpart in the *coarsened at random (car)* assumption.

Incomplete data is a partial observation of some underlying complete data represented by a random variable X with values in a finite state space $W = \{x_1, \ldots, x_n\}$. X has a distribution P_θ for some θ in a parameter space Θ.

The value of X is observed only incompletely. In the general coarse data model such incomplete observations of X can be given by any subset of the state space W. Formally, these observations are the values of a random variable Y with state space 2^W. It is assumed that the observations Y always contain the true value of X. The joint distribution of X and Y, then can be parameterized by $\theta \in \Theta$ and a parameter vector λ from the parameter space

$$\Lambda_{sat} := \{(\lambda_{x,U})_{x \in W, U \in 2^W : x \in U} \mid \lambda_{x,U} \geq 0, \; \forall x \in W : \sum_{U : x \in U} \lambda_{x,U} = 1\},$$

so that $P_{\theta,\lambda}(X = x, Y = U) = P_\theta(X = x)\lambda_{x,U}$.

The parameter space Λ_{sat} represents the *saturated* coarsening model, i.e. the one that does not encode any assumptions on how the data is coarsened. Specific assumptions on the coarsening mechanism can be made by limiting admissible λ-parameters to some subset of Λ_{sat}. The *car* assumption corresponds to the subset

$$\Lambda_{car} := \{\lambda \in \Lambda_{sat} \mid \forall U \forall x, x' \in U : \lambda_{x,U} = \lambda_{x',U}\}.$$

When $\lambda \in \Lambda_{car}$ one can simply write λ_U for $\lambda_{x,U}$.

Let $\boldsymbol{U} = (U_1, \ldots, U_N)$ be a sample of realizations of Y. The log-likelihood ratio based on $\boldsymbol{U}$ for testing the *car*-assumption against the unrestricted alternative is

$$LR(\boldsymbol{U}) := \frac{1}{N}(\max_{\theta \in \Theta} \max_{\lambda \in \Lambda_{car}} \sum_{i=1}^{N} \log P_{\theta,\lambda}(Y = U_i) - \max_{\theta \in \Theta} \max_{\lambda \in \Lambda_{sat}} \sum_{i=1}^{N} \log P_{\theta,\lambda}(Y = U_i)) \tag{1}$$

(for convenience we normalize the ratio by the sample size).

The *profile log-likelihood* (over Θ) given a coarsening model $\Lambda \subseteq \Lambda_{sat}$ is defined as

$$LL_\Lambda(\theta \mid \boldsymbol{U}) := \max_{\lambda \in \Lambda} \sum_{i=1}^{N} \log P_{\theta,\lambda}(Y = U_i).$$

In this paper we will only be concerned with $\Lambda = \Lambda_{sat}$ (no assumptions on the coarsening mechanism) and $\Lambda = \Lambda_{car}$ (*car* assumption). We call the resulting profile likelihoods simply *profile(sat)*-, respectively *profile(car)-likelihood*, denoted LL_{sat}, LL_{car}.

Using profile likelihoods, (1) can be rewritten as

$$LR(\boldsymbol{U}) := \frac{1}{N}(LL_{car}(\hat{\theta} \mid \boldsymbol{U}) - LL_{sat}(\hat{\hat{\theta}} \mid \boldsymbol{U})), \tag{2}$$

where $\hat{\theta}$ and $\hat{\hat{\theta}}$ are the maxima of $LL_{car}(\cdot \mid \boldsymbol{U})$, respectively $LL_{sat}(\cdot \mid \boldsymbol{U})$.

The profile(car) likelihood factors as

$$LL_{car}(\theta \mid \boldsymbol{U}) = Lf(\boldsymbol{U}) + LL_{FV}(\theta \mid \boldsymbol{U}),$$

where $LL_{FV}(\theta \mid \boldsymbol{U}) = \sum_{i=1}^{N} logP_{\theta}(X \in U_i)$ is the *face-value* log-likelihood [1], i.e. the likelihood obtained by ignoring the missingness mechanism, and

$$Lf(\boldsymbol{U}) := \max_{\lambda \in \Lambda_{car}} \sum_{i=1}^{N} log\lambda_{U_i}. \tag{3}$$

We wish to compute $LR(\boldsymbol{U})$ by computing the three components $Lf(\boldsymbol{U})$, $LL_{FV}(\hat{\theta} \mid \boldsymbol{U})$, and $LL_{sat}(\hat{\hat{\theta}} \mid \boldsymbol{U}))$. In most cases it will be impossible to obtain exact, closed-form solutions for any of these three terms. We therefore have to use approximate methods.

Approximating LL_{FV}. To compute $LL_{FV}(\hat{\theta} \mid \boldsymbol{U})$ one has to find $\hat{\theta}$, i.e. optimize the face-value likelihood. This can typically be accomplished by some version of the EM algorithm. In our experiments we use the EM implementation for Bayesian networks provided by the Hugin system (www.hugin.com). Since we are not guaranteed to find a global maximum of LL_{FV}, we obtain only a lower bound on $LL_{FV}(\hat{\theta} \mid \boldsymbol{U})$.

Approximating $Lf(\boldsymbol{U})$. To approximate the term $1/NLf(\boldsymbol{U})$ in (2) we have to find

$$\max_{\lambda \in \Lambda_{car}} \frac{1}{N} \sum_{i=1}^{N} log(\lambda_{U_i}) = \max_{\lambda \in \Lambda_{car}} \sum_{j=1}^{K} m(\bar{U}_j) log(\lambda_{\bar{U}_j}), \tag{4}$$

where $\bar{\boldsymbol{U}} := \bar{U}_1, \ldots, \bar{U}_K$ is an enumeration of the distinct $U_i \in \boldsymbol{U}$, and $m(\bar{U}_j)$ is the empirical probability of $\bar{U}_j$ in $\boldsymbol{U}$. Thus, computing $Lf(\boldsymbol{U})$ is a convex optimization problem under linear constraints of the form $\lambda_U \geq 0$ and $\sum_{U:x\in U} \lambda_U = 1$. However, since there is one constraint of the latter form for each $x \in W$, the number of constraints is manageable only for very small state spaces W.

As a first simplification of the problem, we observe that since the objective function only depends on $\lambda_{\bar{U}}$ for $\bar{U} \in \bar{\boldsymbol{U}}$, we can restrict the optimization problem to these $\lambda_{\bar{U}}$ under the linear constraints

$$C(x): \quad \sum_{\bar{U} \in \bar{\boldsymbol{U}}: x \in \bar{U}} \lambda_{\bar{U}} \leq 1 \qquad (x \in W). \tag{5}$$

An optimal solution $\hat{\lambda}$ for $(\lambda_{\bar{U}_1}, \ldots, \lambda_{\bar{U}_K})$ can be extended to an optimal solution $\hat{\lambda} \in \Lambda_{car}$ by setting $\hat{\lambda}_{\{x\}} := 1 - \sum_{\bar{U} \in \bar{\boldsymbol{U}}: x \in \bar{U}} \hat{\lambda}_{\bar{U}}$ for all $x \in W$ with $\{x\} \notin \bar{\boldsymbol{U}}$, and $\hat{\lambda}_U := 0$ for all other $U \notin \bar{\boldsymbol{U}}$.

The optimization problem now is reduced to a manageable number of parameters. In order to also obtain a manageable number of constraints, we perform the optimization of (4) only under a subset $C(x_1), \ldots, C(x_m)$ of the constraints (5), which is obtained by randomly sampling $x_i \in W$. Since we are thus relaxing the feasible region, we obtain an over-estimate of (4). In our experiments we found that it is sufficient to sample approximately $m = 2K$ constraints, i.e. adding more constraints tended not to change the computed maximum (4) significantly.

Once a set of constraints has been generated we employ the standard Lagrange multiplier approach to perform the optimization. For the unconstrained optimization of the dual function the PAL Java package is used (http://ftp.cse.sc.edu/bioinformatics/PAL/pal-1.4/).

Approximating LL_{sat}. To compute $LL_{sat}(\hat{\hat{\theta}} \mid \boldsymbol{U}))$ we have to maximize the profile (sat)-likelihood. For this we use the AI&M procedure as introduced in [5]. AI&M resembles the EM procedure for maximizing LL_{FV}. Like EM, AI&M is a generic method that has to be implemented by concrete computational procedures for specific types of parametric models. In our experiments we use the AI&M implementation for Bayesian networks as described in [5]. The AI&M procedure will usually not find a global maximum of LL_{sat}, so that as for LL_{FV} we obtain a lower bound on the correct value.

Combining our approximations for the components of (2), we obtain an approximation $\bar{LR}(\boldsymbol{U})$ of $LR(\boldsymbol{U})$. Since we over-estimate $Lf(\boldsymbol{U})$, and under-estimate LL_{FV} and LL_{sat}, one cannot say whether $\bar{LR}(\boldsymbol{U})$ will over- or under-estimate $LR(\boldsymbol{U})$. However, when our approximation for LL_{FV} is better than our approximation for LL_{sat} (as can be expected), then we will obtain an over-estimate of $LR(\boldsymbol{U})$.

3 Generating Non-*car* Data

In our experiments we want to investigate how effective our computed $\bar{LR}(\boldsymbol{U})$ is for testing *car*. To this end we generate incomplete data from Bayesian network models following the general procedure described in [5]: to a Bayesian network with nodes $V_1, \ldots, V_k$ Boolean *observation nodes* $obsV_1, \ldots, obsV_k$ are added. The observation nodes are randomly connected with the original nodes and among themselves. The conditional probability tables for the observation nodes are randomly filled in by independently sampling the rows from a Beta distribution with mean μ and variance σ. Then complete instantiations of the extended network are sampled, giving an incomplete observations of $V_1, \ldots, V_k$ by omitting the values for which $obsV_i = false$.

This general procedure allows us to control in various ways how non-*car* the generated data will be. The first way is by setting the variance σ of the Beta distribution: $\sigma = 0$ means that all rows in all conditional probability tables will be identical, and so the $obsV_i$ nodes become actually independent of their parents, meaning that the data becomes *car* (indeed, missing completely at random). Large values of σ lead to nearly deterministic, complex dependency patterns of the $obsV_i$ variables on their parents, which allows for highly non-*car* mechanisms.

A second way of controlling *car* is by taking mixtures of several coarsening mechanisms: to generate a sample of size N, $l \geq 1$ different coarsening models are generated by our standard method, and from each a sample of size N/l is generated. By the following theorem, the data thus generated becomes *car* for $l \to \infty$.

Theorem 1. *Let μ be a probability distribution on Λ_{sat} such that*

$$\textit{for all } U \subseteq W,\ x, x' \in U:\ E_\mu[\lambda_{x,U}] = E_\mu[\lambda_{x',U}]. \tag{6}$$

There exists $\lambda_\infty \in \Lambda_{\text{car}}$ such that for $\lambda_1, \lambda_2, \ldots \in \Lambda_{\text{sat}}$ iid sampled according to μ: $P_\mu(lim_{n\to\infty} 1/n \sum_{i=1}^{n} \lambda_i = \lambda_\infty) = 1$. Furthermore, for fixed $\theta \in \Theta$: $P_\mu(lim_{n\to\infty} 1/n \sum_{i=1}^{n} P_{\theta,\lambda_i} = P_{\theta,\lambda_\infty}) = 1$.

The proof of the theorem is almost immediate by an appeal to the strong laws of large numbers. The theorem is of some independent interest in that it says that random mixtures of coarsening mechanisms tend to become *car*. This is relevant for real-life datasets, which often can be assumed to be produced by mixtures of coarsening mechanisms (for example, different employees entering customer's records into a database may exhibit different data coarsening mechanisms). However, one must also take into account that the symmetry condition (6) is satisfied for mathematically natural sampling distributions like Lebesgue measure, but not for most real-life sampling distributions over coarsening mechanisms.

Since our random construction of coarsening mechanisms is completely symmetric with respect to different values of the random variables, the symmetry condition (6) is satisfied, and our data becomes *car* for $l \to \infty$.

4 Experiments

In all our experiments we first select a Bayesian network from which incomplete data then is generated as described in the preceding section. In all experiments the structure of the network used for generating the data also defines the parametric model Θ used in computing $\bar{LR}(\boldsymbol{U})$. Thus, in our experiments the assumptions made for the underlying complete data distribution are actually correct. Most of our experiments are based on the standard benchmark 'Asia' and 'Alarm' Bayesian networks. Asia has 8 nodes, defining a state space W of size 256. Alarm has 37 nodes with $|W| = 1.7 \cdot 10^{16}$.

As a reference point for further experiments we use the following base experiment: using Asia as the underlying complete data model, 100 incomplete datasets are generated from 100 different coarsening models. Each dataset is of size 5000, and the parameters of the Beta distribution used in constructing the coarsening model are $\mu = 0.1, \sigma = 0.05$. This setting gives a distribution over parameters that is quite highly concentrated near extreme values 0 and 1, leading to an incomplete data distribution that is strongly non-*car* according to our heuristic described in Section 3.

Figure 1 a) shows the distribution over computed $\bar{LR}(\boldsymbol{U})$ values for the different datasets. The variance in the results is due to variations at three different levels: first, the different randomly generated coarsening models lead to a different expected value $E[LR(\boldsymbol{U})]$; second, the value $LR(\boldsymbol{U})$ for the actually sampled dataset varies from $E[LR(\boldsymbol{U})]$; third, our approximation $\bar{LR}(\boldsymbol{U})$ varies from $LR(\boldsymbol{U})$. Figure 1 b) shows the result of sampling 100 different datasets each from only three different coarsening models. This clearly indicates that the primary source of variance in a) is the difference in the coarsening models (of course, this can change for smaller sample-sizes). Finally, for the model inducing the leftmost cluster in b), and one dataset sampled from that

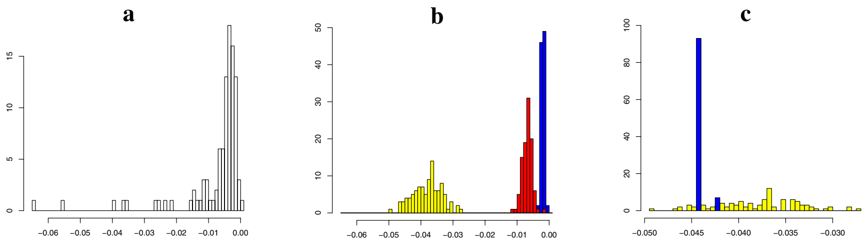


Fig. 1. Likelihood ratio distribution

model, the computation of $LR(\boldsymbol{U})$ was repeated 100 times. Figure 1 c) shows the result (black histogram), and, for comparison the result from 100 different datasets (light gray histogram – this is the same as in b)). From this we infer that the variance observed in b) for fixed models is mostly due to the sample variance of the datasets, and less to the variance in the randomized computation of $\bar{LR}(\boldsymbol{U})$. The bi-modality observed in the black histogram in c) can be traced back to the computation of $LL_{FV}(\hat{\theta}\boldsymbol{U})$, i.e. there appear to have been (at least) two different convergence points of EM.

In summary, the results shown in Figure 1 show that our computed $\bar{LR}(\boldsymbol{U})$ measure actual properties of the given model and data, and is not dominated by noise in the computation. We can now proceed to investigate how good an indicator for *car*-ness this value is. To this end we now vary the coarse data generation of the base experiment in two ways: in one experiment we use smaller variance parameters $\sigma = 0.02$ and $\sigma = 0$ in the Beta distribution. In a second experiment we leave the Beta distribution unchanged, but create mixtures with $l = 2$ and $l = 10$ components. Figure 2 shows the results. The left histograms in both rows are just the results from the base experiment again. As we move from left to right (in both rows), the data becomes more *car* according to our heuristic *car*-measures σ and l (it is truly *car* in the $\sigma = 0$ experiment). The corresponding increasing concentration of $\bar{LR}(\boldsymbol{U})$ near 0 shows that it can indeed serve as statistic for discriminating between *car* and non-*car* models.

Due to its quite small state space, models based on the Asia network do not pose the full computational challenge of computing $\bar{LR}(\boldsymbol{U})$. An experiment with two different variance settings has also been conducted for the Alarm network. Figure 3 shows the result. We observe that here the computed $\bar{LR}(\boldsymbol{U})$ values are all positive. Since the actual $LR(\boldsymbol{U})$ values must be ≤ 0, this means that the over-estimate of $Lf(\boldsymbol{U})$, combined with the under-estimate of LL_{sat} here lead to a significant over-estimate of $LR(\boldsymbol{U})$. Nevertheless, the computed $\bar{LR}(\boldsymbol{U})$ discriminates quite successfully between the *car* and non-*car* models.

As a final test for the $\bar{LR}(\boldsymbol{U})$ computation we use data from three different artificially constructed Bayesian networks: all networks contain seven binary nodes. The first network ('simple') contains no edges; the second network ('medium') contains 14 randomly inserted edges, and the third ('dense') is a fully connected network (21 edges). The conditional probability tables are randomly generated. The three models represent decreasingly restrictive distributional assumptions, with the dense network not encoding any restrictions. We again sample 100 datasets from 100 different random

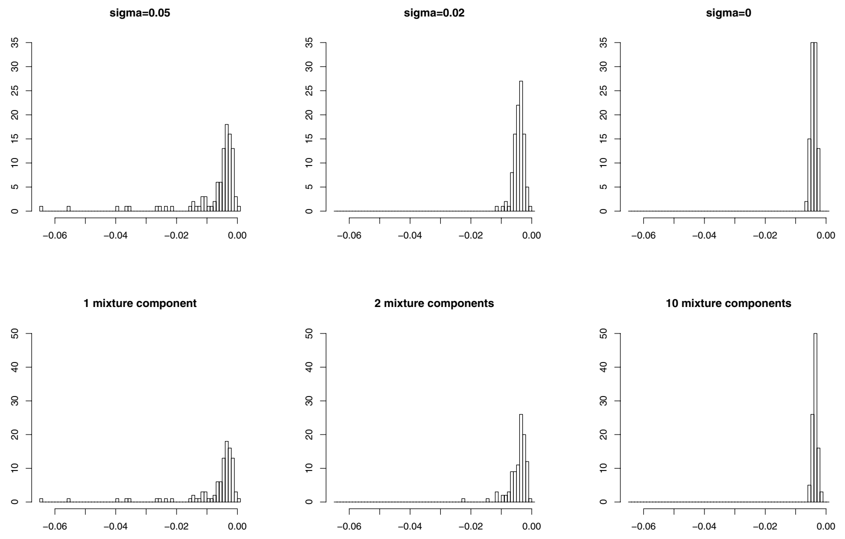


Fig. 2. Computed likelihood ratios and heuristic *car*-measures (Asia)

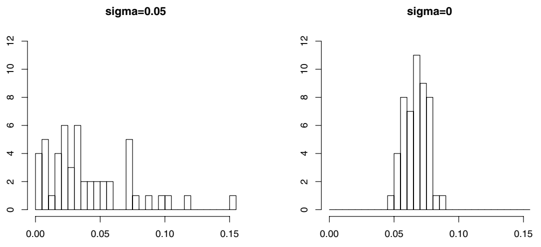


Fig. 3. Computed likelihood ratios and heuristic *car*-measures (Alarm)

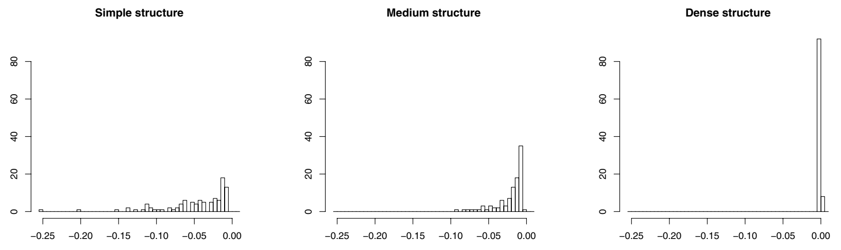


Fig. 4. Likelihood ratio and model structure

coarsening models (sample-size 5000, $\mu = 0.1$, $\sigma = 0.05$). Figure 4 shows the result. As required by the fact that *car* is not testable without any restrictive assumptions on the full data model, we observe that the $\bar{LR}(\boldsymbol{U})$-values for the dense network show no indication that the data is not *car*. The more restricted the model, the easier it becomes to refute the *car*-assumption based on $\bar{LR}(\boldsymbol{U})$.

5 Conclusion

Utilizing our recently introduced AI&M procedure for optimizing the profile(sat)-likelihood, we have shown how to compute an approximate likelihood-ratio statistic for testing the *car* assumption in the context of distributional constraints on the underlying complete data distribution. Initial experiments show that we obtain a quite effective measure for discriminating between *car* and non-*car* incomplete data distributions.

To obtain a practical test for a particular dataset under consideration, one will also need a way to specify a critical value κ, so that the *car* hypothesis will be accepted iff $\bar{LR}(\boldsymbol{U}) \geq \kappa$. At this point there exist no general, theoretically well-founded rules for setting κ. The best way to proceed, therefore, is to empirically determine for a given state space W and a parametric model Θ, the sampling distribution of $\bar{LR}(\boldsymbol{U})$ under the *car* assumption, and to set κ according to the observed empirical distribution and the desired confidence level.

Tests for *car* can also play a role in model selection: when *car* is rejected relative to a current parametric complete data model Θ, one may either retain Θ and employ techniques not relying on *car*, or one can relax the parametric model, thus hoping to make it consistent with *car* (which ultimately it will, as illustrated in Figure 4).

References

1. A. P. Dawid and J. M. Dickey. Likelihood and bayesian inference from selectively reported data. *Journal of the American Statistical Association*, 72(360):845–850, 1977.
2. R. D. Gill, M. J van der Laan, and J. M. Robins. Coarsening at random: Characterizations, conjectures, counter-examples. In D. Y. Lin and T. R. Fleming, editors, *Proceedings of the First Seattle Symposium in Biostatistics: Survival Analysis*, Lecture Notes in Statistics, pages 255–294. Springer-Verlag, 1997.
3. D. F. Heitjan and D. B. Rubin. Ignorability and coarse data. *The Annals of Statistics*, 19(4):2244–2253, 1991.
4. M. Jaeger. Ignorability for categorical data. *The Annals of Statistics*, 33(4):1964–1981, 2005.
5. M. Jaeger. The AI&M procedure for learning from incomplete data. In *Proceedings of UAI-06*, 2006. To appear.
6. C. Manski. *Partial Identification of Probability Distributions*. Springer, 2003.
7. D.B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

B-Matching for Spectral Clustering

Tony Jebara and Vlad Shchogolev

Department of Computer Science, Columbia University, New York NY 10027, USA

Abstract. We propose preprocessing spectral clustering with b-matching to remove spurious edges in the adjacency graph prior to clustering. B-matching is a generalization of traditional maximum weight matching and is solvable in polynomial time. Instead of a permutation matrix, it produces a binary matrix with rows and columns summing to a positive integer b. The b-matching procedure prunes graph edges such that the in-degree and out-degree of each node is b, producing a more balanced variant of k-nearest-neighbor. The combinatorial algorithm optimally solves for the maximum weight subgraph and makes subsequent spectral clustering more stable and accurate. Experiments on standard datasets, visualizations, and video data support the use of b-matching to prune graphs prior to spectral clustering.

1 Introduction

Clustering is an important tool in the machine learning portfolio, particularly in unsupervised settings. Traditional approaches to clustering include iterative methods such as k-means and Expectation Maximization (EM) which make parametric assumptions about the data and can be easily confounded by local minima during their typically greedy optimizations. Recently, spectral clustering [9,6] methods have gained prominence as principled relaxations of the NP normalized cut clustering problem. These algorithms typically involve finding the top eigenvectors after processing an affinity matrix built from pairwise similarities between points in a dataset. This affinity matrix can be seen as a weighted graph. Essentially, spectral clustering makes an appeal to spectral graph theory [1] and approximates the NP-complete normalized cut procedure. Since a user need only specify a similarity function, spectral clustering methods are non-parametric and avoid explicit assumptions about the generative model of the data. Furthermore, spectral clusterings are not plagued by local minima and often outperform traditional greedy parametric clustering methods. Also, unlike many greedy methods, spectral methods enjoy polynomial run-time guarantees. Finally, spectral clustering produces the same result despite permutation and reordering of input points (unlike some greedy methods such as single-linkage clustering that process data-points in a sequential manner).

Currently, a gamut of spectral clustering algorithms are available and exhibit some variability in their performance on real-world datasets. In this article, we propose a pre-processing of the weighted graph of pairwise similarities. This can be done prior to *any* spectral clustering method. This pre-processing involves b-matching [5], a permutationally invariant (i.e. independent of the ordering of the points in the dataset) combinatorial procedure which eliminates edges in the weighted graph. Conveniently, b-matching on a weighted graph is solvable in polynomial time. The procedure finds the maximum weight subgraph in the original graph where each vertex has an in-degree and an out-degree of b. By preceding spectral clustering with b-matching, we reduce the mismatch the spectral clustering has to an exact normalized cut solution. We conjecture that removing edges optimally via b-matching makes the spectral clustering relaxation more closely approach the NP-complete normalized cut solution. Another argument which is often cited in the nonlinear manifold embedding literature is that the similarity metric being used is only locally reliable. It is unreliable if points are distant or produce low edge weight in the graph [8]. In fact, embedding methods typically resort to a k-nearest neighbor method for pruning the weighted similarity graph which can be seen as a greedy variant of b-matching. Therefore, a b-matching graph pruning procedure can compensate for a poor choice of the similarity metric used in spectral clustering.

2 Spectral Clustering

Assume we are given N samples, $\mathbf{x}_1, \ldots, \mathbf{x}_N$ where each datum is in a sample space $\mathbf{x}_i \in \mathcal{X}$. Also assume we can readily compute an affinity between pairs of samples via the function $k(\mathbf{x}_i, \mathbf{x}_j)$. Consider a matrix $A \in \mathbb{R}^{N \times N}$ of affinities between all pairs of points in the dataset such that $A_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $A_{ii} = 0$. This matrix describes a fully connected graph G with N vertices V and $N \times N$ edges E. The edge between node i and node j has weight A_{ij}. Without loss of generality, assume that the points $\mathbf{x}_i$ are in d-dimensional Euclidean space $\mathbb{R}^d$ and the similarity function is merely a radial basis function kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$. Therefore, $A_{ij} = A_{ji} \geq 0$. Although we focus on binary clustering, multi-category extensions are straightforward. Given a weighted graph G, a good clustering criterion is the minimum normalized cut [7]. Consider a subset $B \subset V$ of the full vertex set V. Normalized cut is the NP-complete minimization of the cost:

$$NCUT(B) = \frac{\sum_{i \in B, j \in V/B} A_{ij}}{\sum_{i \in B, j \in V} A_{ij}} + \frac{\sum_{i \in V/B, j \in B} A_{ij}}{\sum_{i \in V/B, j \in V} A_{ij}}.$$

To make the problem tractable, [9] provide a relaxation by solving for a real valued solution instead of a discrete vertex selection (or cut). This approach efficiently approximates normalized cut. First, we compute the $N \times N$ diagonal matrix $D_{ii} = \sum_j A_{ij}$. We represent the discrete set B via the indicator vector $\mathbf{y} \in \mathbb{R}^N$. This vector is defined as $\mathbf{y}(i) = \sqrt{d_{V/B}/d_B d}$ if node i is in B and otherwise $\mathbf{y}(i) = -\sqrt{d_B/d_{V/B} d}$. Here, we take $d = \sum_i D_{ii}$ and $d_B = \sum_{i \in B} D_{ii}$. Normalized cut finds a discrete $\mathbf{y}$ vector that minimizes $\mathbf{y}^T(D - A)\mathbf{y}$ subject to $\mathbf{y}^T D\mathbf{y} = 1$ and $\mathbf{y}^T D\mathbf{e} = 0$. Since this is an intractable, we solve for a real-valued $\mathbf{y}$ vector instead. This is done by via the generalized eigenvalue system $(D - A)\mathbf{y} = \lambda D\mathbf{y}$. We get $\mathbf{y}$ as the second smallest eigenvector of the eigensystem. The scalar values of $\mathbf{y}$ determine which nodes belong to the cut.

In practice, we will use a variant of the spectral clustering algorithm above [6]. This variant is as follows. First, compute A and D and the normalized Laplacian $D^{-1/2}AD^{-1/2}$. Second, find the k largest eigenvectors of L and form the matrix $X \in \mathbb{R}^{N\times k}$ by stacking them. Third, form the matrix Y from X by $Y_{ij} = X_{ij}/\sqrt{\sum_j X_{ij}^2}$. Fourth, treat each row of Y as a point in $\mathbb{R}^k$ and cluster them into k clusters using k-means. Fifth, assign the i'th point in the dataset the same cluster label that the i'th row of the Y matrix was assigned. There is evidence this variant of spectral clustering has better empirical performance as well as theoretical justification. This is because it exploits a larger eigengap in the eigensystem which prevents eigenvectors from rotating arbitrarily. Eigenvectors with similar eigenvalues can be rotated within the subspace and can then become unreliable for clustering. The most computationally demanding aspect of these spectral clustering methods is the $\mathcal{O}(N^3)$ eigensystem solution.

Clearly, the performance of spectral clustering algorithms hinges on the input weight matrix A. But, this matrix may be corrupted with poor pairwise affinity values. This intuition is also relevant for embedding methods [8] which also encourage pruning spurious edges from a weighted graph, typically using a k-nearest-neighbor method. One argument for pruning is that the similarity metric is only locally valid and becomes unreliable when points are distant from each other or produce low edge weight in the graph. Furthermore, a large eigengap is desirable for eigenvector stability and would emerge from an affinity matrix A that had a binary clustering structure. For instance, consider the binary clustering problem with an indicator vector $\mathbf{y} \in \mathbb{R}^N$ where $\mathbf{y}(i) = \pm 1$. Also, assume that the clustering is balanced such that $\sum_i \mathbf{y}(i) = 0$. The resulting *ideal affinity matrix* is $A_{ij} = \frac{1}{2}(\mathbf{y}(i)\mathbf{y}(j) + 1)$. In other words, use high affinity between points in the same class and none between points from different classes. The matrix A will be binary with rows and columns summing to $b = N/2$. Clearly, the L matrix in the spectral clustering procedure will exhibit a large and stable eigengap if fed such an *idealized affinity matrix*. Is it possible to achieve these goals by pre-processing the A matrix without straying too far from the original information it carries? We suggest using b-matching, a combinatorial optimization procedure that deletes some low-edge weight entries from the A matrix to produce an *idealized affinity matrix*. This is done while keeping strong edges to produce the maximum total weight sub-graph from the original A.

3 Weighted B-Matching

We will use b-matching to prune and binarize the affinity matrix A in the previous section. Assume we are given a weighted general graph G with nodes V and edges E. An edge connecting node i to j has weight A_{ij}. The maximum weight b-matching problem [7] is a maximum weight subgraph of G such that the degree of each vertex in the subgraph is b. This is a special case of the *degree-constrained subgraph problem*. We have the following combinatorial optimization.

Given a weight matrix $A \in \mathbb{R}^{|V|\times|V|}$, find a binary matrix $P \in \mathcal{B}$ where $\mathcal{B}$ is the space of binary matrices $\{0,1\}^{|V|\times|V|}$ that maximizes the following:

$$\max_{P\in\mathcal{B}} \sum_{ij} P_{ij} A_{ij} \ \ s.t. \sum_{i} P_{ij} \ = \ \sum_{j} P_{ij} \ = \ b. \tag{1}$$

The above is essentially a balanced variant of k-nearest neighbor. Nearest-neighbor methods only enforce the row constraint $\sum_i P_{ij} = b$ instead of enforcing both row and column summation. Meanwhile, b-matching ensures that each point has b neighbors and only b other points may choose it as a neighbor. This prevents, for example, a single centrally-located point from dominating the data and acting as a neighbor to too many other points. B-matching is also a generalization of the 1-matching problem or linear assignment problem (LAP) which finds a permutation matrix P (i.e. $b = 1$). LAP is solvable via the Kuhn-Munkres or Hungarian method in $\mathcal{O}(|V|^3)$ time. Tutte [10] shows that is possible to transform an instance of the b-matching problem into a 1-matching problem on general graphs. The later can be solved by the Blossom algorithm, a linear programming formulation developed by Edmonds [2]. A direct linear programming approach to general matching is not possible because one cannot guarantee that the algorithm will produce integral solutions. Edmonds solves this issue by adding an exponential number of constraints to the linear program. The added constraints enforce that no odd-length circuit (called a blossom) could have more matched edges than appropriate. These constraints are unnecessary for bipartite matching since bipartite graphs don't have odd circuits. The Blossom algorithm uses the primal-dual method to solve the linear program. A special search procedure avoids working explicitly with the exponential number of constraints. In particular, the algorithm works by shrinking blossoms into single pseudo-nodes, ensuring that the algorithm can detect a way to improve the current matching. The primal-dual method starts with a feasible dual solution and searches for a feasible primal solution that satisfies the complementary slackness conditions. The search is performed on a restricted primal (RP) problem. If unsuccessful, the dual solution is updated via an update rule, and the entire procedure is repeated. The algorithm moves from one basic feasible RP solution to another. The cost decreases monotonically and no basis is repeated. Hence the algorithm will terminate in polynomial time. The b-matching problem can also be solved directly by a variant of the blossom algorithm. One difference during b-matching is the blossom structure is more complicated than just odd circuits [5]. The current best exact algorithm for b-matching is due to Gabow [4] and runs in $\mathcal{O}(\min(|E|\log|V|, |V|^2)|V|b)$ time. Recently, this and other matching problems have been reformulated in terms of Balanced Network Flow problems [3]. We used the b-matching software package: www.math.uni-augsburg.de/opt/goblin.html.

4 B-Matching for Spectral Clustering

We will use the b-matching procedure to find an *idealized affinity matrix* that is closest to (in the Frobenius norm sense) the original affinity matrix. Suppose we have N objects and we wish to find a vector $\mathbf{y} \in \{-1, 1\}^N$ which gives a binary clustering of the objects. If $\mathbf{y}$ were the true labeling of the objects, then we would like our kernel matrix to be $P^* = \frac{1}{2}(\mathbf{y} \cdot \mathbf{y}^T + 1)$. However, the kernel or affinity matrix is normally computed from real world data, and hence does not have the simple structure of P^*. We address this by finding an approximation to A that has the structure of matrix P^*. In particular, we find a binary symmetric matrix with rows and columns summing to b which we will typically set to $b = \frac{N}{2}$. We thus have the following minimization:

$$\min_P \|P - A\|_F^2 \;\text{ subject to }\; \sum_i P_{ij} \;=\; \sum_j P_{ij} \;=\; b \;\; P_{ij} \in \{0, 1\}.$$

The cost function is simplified as follows:

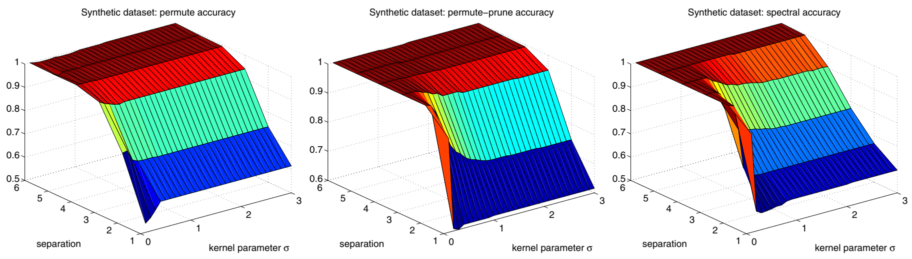
$$\|P - A\|_F^2 \;=\; tr[PP^T] + tr[AA^T] - 2\; tr[P^T A] \;=\; Nb + \|A\|_F^2 - 2\; tr[P^T A].$$

Only the last term depends on P. Note that $tr[P^T A] = \sum_{i,j} P_{ij} A_{ij}$ Therefore, the Frobenius norm minimization problem is exactly equivalent to b-matching maximization in Equation 1. In other words, we can equivalently solve for the b-matching that is most correlated with the input affinity weight matrix A. Thus, we precede spectral clustering algorithms with this b-matching procedure applied to the A affinity matrix and get the binary b-matching matrix P. For binary clustering, we set $b = N/2$ just as in the *idealized affinity matrix.* This essentially encourages a balanced clustering problem. We then use the P matrix in a spectral clustering procedure instead of the original A matrix. Since this matrix is binary, this may result in lost information so we also consider using the matrix $P \otimes A$ (where $\otimes$ is the element-wise product of the two matrices) as the affinity matrix for spectral clustering. This leads to two variations on spectral clustering. We find P via b-matching with weights in A and with $b = N/2$. One approach, **permute** performs spectral clustering with P as the affinity matrix. The other approach, **permute-prune** instead uses the element-wise product of $P \otimes A$ as the affinity matrix. We compare these two approaches to direct spectral clustering called **spectral** on the original affinity matrix. We also compare the performance where we replace the b-matching procedure with a k-nearest-neighbor procedure. Here, k is set to equal b to obtain a similar effect. This approach yields two more competitor methods, namely **knn** and **knn-prune**. Note, to go beyond binary clustering we find M clusters. This is done by setting $b = N/M$ in the b-matching procedure and using multi-class spectral clustering tools. We next show experiments with these various spectral clustering methods.

5 Experiments

To evaluate the clustering schemes, we applied them to classification problems where labels are hidden from the learning algorithm. The labels are used afterward to report a classification accuracy by seeing how well the clustering algorithms agree with the true labeling.

In a synthetic experiment, we generated data along two S-shaped curves with a different spread value c between them. The larger the value of c is, the further apart the two S-curves are. The desired classification is to separate each S-curve from the other. As c gets small, clustering algorithms will misplace a point from one curve onto another. We also vary the σ parameter in the RBF to see its effect on the algorithms. Figure 1 shows the accuracy of (a) the permute method, (b) the permute-prune method which does best for the range of settings for c and σ and (c) the traditional spectral clustering method which does worst.



(a) Permute (b) Permute-Prune (c) Spectral

Fig. 1. Classification accuracy for c and σ settings

We evaluated the clustering methods on UCI binary classification problems across varying RBF σ settings. Figure 2(a) shows the accuracy on the UCI OptDigits dataset. The average accuracy over ten folds of size $N = 100$ training examples is shown. Permute is the top algorithm throughout and peaks at over 90% at the best choice of σ. In the PenDigits dataset in Figure 2(b), we down-sampled to $N = 80$ training examples and show the mean accuracy for all algorithms. Permute-prune is the top performer here. For the UCI Vote dataset with 80 training examples, Figure 2(c), shows both permute and permute-prune performed well and above the other methods.

We finally evaluated the clustering methods on video sequences composed of two scenes. In a scene, actors move and cameras pan smoothly. However, more abrupt changes occur during a scene transition (i.e. a sudden cut). If images are represented as vector coordinates in a Euclidean space (i.e. by rasterizing each image into a vector) a video sequence of two scenes looks like two nonlinear strands that are highly intertwined yet disconnected. Therefore, a good clustering algorithm should separate the two scenes. We obtained video sequence [1] and identified scene transitions manually to obtain a labeled classification problem. We evaluated the various clustering schemes as they discover the labeling just from RBF kernels between pairs of vectorized images. One class is the first scene and the other is the second scene in a video sequence. The order of the frames is randomized and we select $N = 48$ random samples. Interestingly, nearby frames

[1] The video, a real-life parody of the Simpsons television show, is available at: video.google.com/videoplay?docid=-2231271827736577327&q=simpsons+intro.

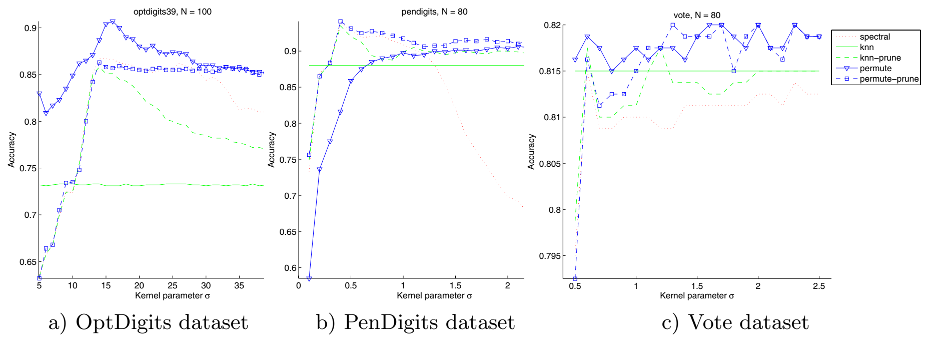
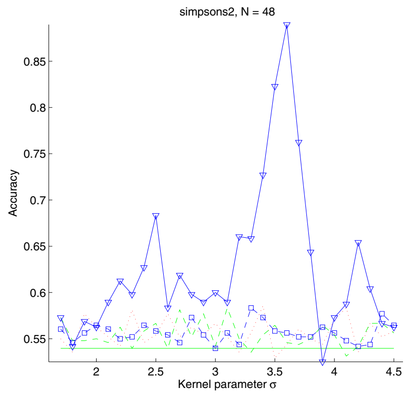


Fig. 2. Accuracy of Spectral Clustering Procedures on UCI Data

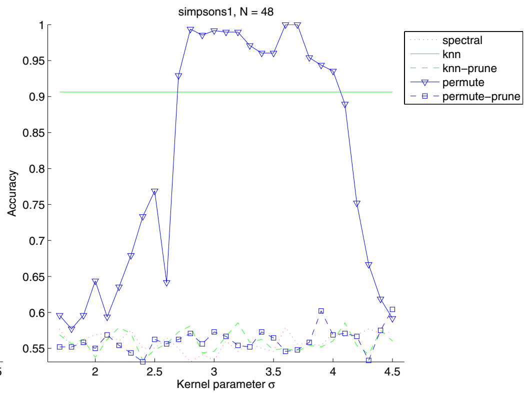
have a strong RBF similarity values with each other yet only weak similarity values to other frames in the dataset. Thus, if the video sequence was sorted in time, we would expect the affinity matrix A to look like a thin banded matrix. We randomize the set of training examples to obtain an average classification accuracy for all five algorithms over 10 folds under various settings of the RBF σ. The goal is to split the video into the two scenes, one with the actress Maggie and one with the actress Marge. Figure 3(a) shows that permute is the only strong clustering method with almost 90% accuracy while the other methods perform close to random chance. A similar experiment was performed with another pair of scenes in the video sequence. One scene contains the actor Bart and one scene contains the actor Homer. Figure 3(b) shows the results of the various clustering approaches. Clearly permute does best and can achieve 100% accuracy (although knn does approach it somewhat). The b-matching solution is able to lock onto the two clusters or threads of video sequences. We conjecture that these video examples are actually reminiscent of our toy S-curves example since each scene forms a windy nonlinear curve from the nearby adjacent frames. These two video curves may be so close together that it is impossible to uncover the clustering unless an aggressive b-matching procedure finds the binary maximum weight b-matching and propagates information across the nearby neighbors on the non-linear curves.

6 Discussion

We showed how b-matching, a polynomial time combinatorial algorithm, can prune and binarize the weighted affinity graph prior to spectral clustering. This procedure improves the accuracy of spectral clustering in applied problems and does better than a k-nearest-neighbor pre-processing. The preprocessing also reduces the variability of spectral clustering over different splits of a dataset. Furthermore, the performance advantage is maintained over a wider range of parameters of the similarity function (i.e. the RBF σ parameter). The b-matching algorithm takes cubic time which is close to the cost of the eigensystem solvers



(a) Maggie vs. Marge Scene



(b) Homer vs. Bart Scene

Fig. 3. Clustering Accuracy on Simpsons Video Sequences

that underly spectral clustering. Nevertheless, we are investigating faster approximate b-matching algorithms to further reduce computational limitation. Finally, we are also looking into theoretical arguments for b-matching. [2]

References

1. Chung, F.: Spectral Graph Theory. American Mathematical Society's CBMS Regional Conference Series in Mathematics **92** (1997)
2. Edmonds, J.: Paths, trees and flowers. Canadian Journal of Mathematics **17** (1965)
3. Fremuth-Paeger, C., Jungnickel, D.: Balanced network flows. I. A unifying framework for design and analysis of matching algorithms. Networks **33** 1 (1999) 1-28
4. Gabow, H.: An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. STOC '83: Proceedings of the fifteenth annual ACM symposium on theory of computing (1983) 448-456
5. Muller-Hannemann, M., Schwartz, A.: Implementing weighted b-matching algorithms: insights from a computational study. J. Exp. Algorithmics **5** (2000) 1084-6654
6. Ng. A., Jordan, M., Weiss, Y.: On spectral clustering: Analysis and an algorithm. Neural Information Processing Systems **14** (2001)
7. Papadimitriou, C., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall (1982)
8. Roweis, S., Saul, L.: Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science, **290** 5500 (2000)
9. Shi, F., Malik, J.: Normalized Cuts and Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, **22** 8 (2000) 888-905
10. Tutte, W.T.: A short proof of the factor theorem for finite graphs. Canadian Journal of Mathematics **6** (1954) 347-352

[2] This research was funded by NSF grants CCR-0312690, IIS-0347499 and IIS-0093302.

Multi-class Ensemble-Based Active Learning

Christine Körner[1] and Stefan Wrobel[1,2]

[1] Fraunhofer Institut Intelligente Analyse- und Informationssysteme, Germany
{christine.koerner, stefan.wrobel}@iais.fraunhofer.de
[2] Dept. of Computer Science III, University of Bonn, Germany

Abstract. Ensemble-based active learning has been proven to efficiently reduce the number of training instances and thus the cost of data acquisition. To determine the utility of a candidate training instance, the disagreement about its class value among the ensemble members is used. While the disagreement for binary classification is easily determined using margins, the adaption to multi-class problems is not straightforward and little studied in the literature. In this paper we consider four approaches to measure ensemble disagreement, including margins, uncertainty sampling and entropy, and evaluate them empirically on various ensemble strategies for active learning. We show that margins outperform the other disagreement measures on three of four active learning strategies. Our experiments also show that some active learning strategies are more sensitive to the choice of disagreement measure than others.

1 Introduction

Ensemble-based active learning is well-known to effectively choose training instances when resources for labeled data are limited. Its most prominent representatives are query-by-bagging and query-by-boosting [1], co-testing [2] and active-decorate [3]. All four strategies choose training instances based on the disagreement among their ensemble members. For binary-class learning problems ensemble disagreement is simply measured by the difference between positive and negative votes. However, it is not obvious how this approach can be generalized to determine ensemble disagreement in multi-class learning problems. Existing literature has consequently proposed a variety of techniques, including margins [2], uncertainty sampling [4, 5] and entropy [6, 7, 3, 8]. Surprisingly, no study exists that evaluates which of these methods is most suitable for ensemble-based active learning or whether the application of a method depends on the chosen ensemble strategy.

In this paper we compare the three disagreement measures proposed in the literature along with a "control" measure that combines different aspects of existing measurements. In a comprehensive set of experiments on 12 different learning problems, we evaluate all four disagreement measures empirically in the context of the four most prominent ensemble-based active learning strategies, namely query-by-bagging and query-by-boosting [1], co-testing [2] and active-decorate [3]. We show that margins outperform other query selection strategies on three of four active learning strategies. At the same time, we observe that for query-by-bagging and co-testing the choice of disagreement measure is essential to the success of the active learner, while query-by-boosting and active-decorate perform quite robust using different disagreement measures. The results of our experiments clearly demonstrate that from the existing disagreement measures considered in the literature, the margin-based approach should be chosen as a standard approach for multi-class ensemble-based active learning.

The paper is organized as follows. Section 2 reviews active learning strategies for ensembles. Section 3 presents the disagreement measures for the multi-class case. We evaluate the presented disagreement measures in Section 4 on 12 UCI domains and conclude the paper with future work.

2 Ensemble-Based Active Learning

In this section we review the idea of ensemble-based active learning and describe four active learning strategies which we will use in our experiments. Ensemble-based active learning originates in the query-by-committee approach by Seung et al. [9]. Query-by-committee is a form of query filtering where a stream of unlabeled instances is provided from which the algorithm chooses the most profitable for labeling [10]. The utility of a candidate instance is evaluated by an ensemble of randomly selected hypotheses from the version space, the subset of all hypotheses consistent with the training data. The stronger the committee disagrees on a class label the more valuable is the query. Assuming an infinite number of committee members and an equal number of positive and negative votes (maximal disagreement in binary classification), the knowledge about the query's true class label will halve the version pace. Query-by-committee is an iterative algorithm that adds the queried instance and its label to the training set and repeats until a desired accuracy or the quota for labeling is reached.

In the remainder of this section we review four acknowledged strategies for active learning that spring from the idea of query-by-committee but use different randomization strategies in order to create the ensembles. The presented strategies are query-by-bagging, query-by-boosting, co-testing and active-decorate.

Query-by-bagging and query-by-boosting [1] rely on sampling strategies that randomize the training data before a deterministic learning algorithm (typically C4.5) builds one classifier from each subsample. As the name implies, query-by-bagging utilizes Bagging [11] as sampling strategy, drawing each subsample with replacement. Once the ensemble is formed, the committee votes on the (binary) class values of all unlabeled instances and randomly selects a query from all instances that split the committee most evenly.

Query-by-boosting proceeds similar to query-by-bagging but uses AdaBoost [12] to create differing training sets. AdaBoost is in itself an iterative algorithm that, starting from the original sample distribution, builds a classifier but adapts the distribution to emphasize misclassified instances before the next training set is drawn. Again, an instance with the smallest margin between the number of positive and negative votes is chosen as query, but the votes are weighted according to the training error of each committee member.

Co-testing [2] is an active learning strategy that is inspired by the multi-view approach called co-training [13,14]. It utilizes two redundant views of the training data to create an ensemble and selects, in its naive approach, a query among all unlabeled instances where the two classifiers disagree. Although co-testing relies on independent views, it has been shown to perform well using random splits in domains without redundant attributes [2, 15].

Active-decorate [3] is a recent approach to ensemble-based active learning and uses artificially enhanced training sets. The underlying principle, Decorate [16], increases the size of the ensemble iteratively, starting with one classifier based on the original training set. Afterwards, it constructs new training instances assuming independent attribute distributions and labels them inversely proportional to the prediction of the current ensemble. A new classifier build from the original and artificial training data is added to the ensemble if it reduces the training error of the ensemble. Active-decorate uses margins to measure ensemble disagreement but generalizes the idea to multi-class problems.

3 Disagreement Measures for Multi-class Ensembles

For binary classification ensemble disagreement can be easily determined. It is large if the number of positive and negative votes of the ensemble are evenly split. It is small if one class prevails. However, the generalization to multi-class problems is not that straightforward. Assume that an ensemble of ten classifiers votes on two instances with four possible class values. The vote distributions for instance one and two are $d_1 = (3, 3, 2, 2)$ and $d_2 = (5, 5, 0, 0)$ respectively. Which instance should the active learner recommend? Obviously, the distribution for instance one is very homogeneous. Yet, the contradiction for instance two is fiercer as it targets class one and two.

This section describes four techniques to measure ensemble disagreement in multi-class problems, which we evaluate in Section 4. The first three techniques, margins, uncertainty sampling and entropy, are commonly used in literature. The fourth, specific disagreement, is a "control" measure which we developed to contrast existing approaches. For all measures we assume that an ensemble returns a probability distribution of the class value for each unlabeled instance obtained either by majority vote or by averaging the class distributions of the committee members. Fig. 1 visualizes all four disagreement measures for a three-class problem. Each picture illustrates the disagreement for all possible class distributions $d = (p_1, p_2, p_3)$. The x- and y-axis contain the class probabilities p_1 and p_2 respectively while p_3 is indirectly depicted by isolines with a gradient of -1. Dark colors indicate preferred sections for query selection.

Margin-based disagreement: Following the generalization of binary margins as proposed by Melville and Mooney [3], the margin in multi-class problems is calculated as difference between the first and second highest class probability. A strategy that chooses an instance with minimum margin thus evaluates the competitiveness of the most likely class label. Nevertheless, it does not consider any information about the remaining class probabilities or the level of probability on which a margin occurs.

Uncertainty sampling-based disagreement: Uncertainty sampling [4, 5] provides a second way to generalize the binary margin approach and can be applied to any classifier that provides a class label along with an estimate about the confidence in its prediction. Uncertainty sampling simply queries an instance of which the predicted class value possesses a minimum probability among all candidate instances. Thus, uncertainty sampling accounts for the level of probability. It indirectly prefers candidates with a balanced class distribution but again does not benefit from information about the remaining class probabilities.

Entropy-based disagreement: Entropy is a well-known measure in information theory to determine the disorder of a system. In ensemble-based active learning various forms, ranging from ordinary entropy [6] to Kullback-Leibler divergence [7] and Jensen-Shannon divergence [8], have been applied to train probabilistic classifiers. In our experiments we focus on ordinary entropy defined as $E = -\sum_{i=1}^{k} p_i log_2 p_i$ for a k-class problem. Again, entropy generalizes disagreement as defined for binary classification.

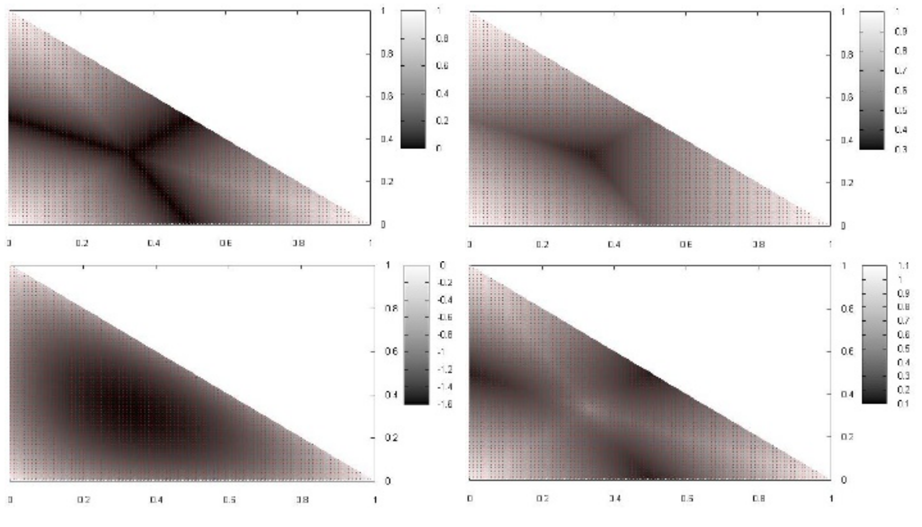


Fig. 1. Visualization of disagreement measures for a three-class problem; top left: margin-based; top right: uncertainty sampling-based; bottom left: entropy-based; bottom right: specific disagreement

Specific disagreement ("control"): The above approaches select queries either by degree of competition between the first two predominant strategies (dark lines, Fig. 1) or according to homogeneity of distribution (dark centers, Fig. 1). Yet, Muslea [15] pointed out that disagreement between two differing predictions also increases with the level of confidence. We therefore designed a "control" measure, specific disagreement, which combines different aspects of the above measures to indicate disagreement on a narrow subset of class values. Our measure combines margin-based disagreement (*margin*) with the maximal class probability (*max*), normalized with the total number of class values ($|c|$): $specific\,disagreement = margin + 0.5 \frac{1}{(|c| \cdot max)^3}$.

4 Experiments

We evaluated the disagreement measures introduced in Section 3 on 12 data sets from the UCI repository [17] as given in Table 1[1]. We applied all measures to the ensemble-based active learning strategies query-by-bagging, query-by-boosting, co-testing and active-decorate. With the exception of co-testing, each ensemble consisted of 20 committee members. The ensembles used C4.5 as base learner and were configured according to their default parameters in the WEKA toolkit [18]. We initiated the active learners with 50 randomly drawn training instances and evaluated the experiments based on 2x10-fold cross validation. During each iteration we added 1 query to the training set and proceeded until all available data was used or a maximum of 250 queries were issued. In order to ascertain the effect of active learning, each ensemble strategy was additionally evaluated on a random sequence of training instances.

Table 1. Characteristics of UCI data sets

	attributes				accuracy		attributes				accuracy
data set	num	sym	class	inst	C4.5	data set	num	sym	class	inst	C4.5
abalone	7	1	3	4177	60.25	optdigits	64	0	10	5620	90.69
bupa	6	0	2	345	68.70	pima	8	0	2	768	73.83
car	0	6	4	1728	92.65	segment.	19	0	7	2310	97.23
ecoli	7	0	8	336	85.07	vehicle	18	0	4	846	71.95
glass	9	0	7	214	65.88	wdbc	30	0	2	569	94.01
letter	16	0	26	20000	88.06	yeast	8	0	10	1484	56.10

We apply two techniques to compare the performance of disagreement measures. The first performs a pairwise comparison of disagreement measures. The second ranks the disagreement measures according to their number of queries that are necessary to reach some target error rate. Both techniques are conducted independently for each active learning strategy.

During pairwise comparison we conduct a z-test on the (over 20 trials averaged) ensemble accuracies after a new training instance has been added. We count for the first 200 queries how often each strategy significantly outperforms the other.[2] If the difference between the individual counts exceeds a threshold

[1] For letter and optdigits we used only the first 5000 and 2500 instances respectively.
[2] Note, that the z-tests are not independent because queries are added sequentially.

of 20, we assign one point to the superior disagreement measure on the given data set. We aggregate the scores over all UCI data sets and calculate total wins, losses and ties per disagreement measure. Table 2 shows the results of pairwise comparison separately for each active learning strategy. A score of 2 in cell (2, 1) means, for example, that the disagreement measure in row 2 outperformed the disagreement measure in column 1 on 2 out of 12 UCI data sets.

The second evaluation technique estimates how efficient an active learner uses the data and is similar to measures used in [1, 3]. It compares the number of training instances necessary to reach a certain target error rate, calculated as average error of the last 50 training examples given a random sequence of queries. In contrast to [3] we record the training set size on the third occurrence the target error rate is reached. This correction proved necessary because the error rate on consecutive queries showed great variation. We ranked the results for each UCI data set and calculated average ranks as shown in Table 3.

Table 2. Pairwise comparison of disagreement measures

query-by-boosting

	mar.	unc.	ent.	spe.	ran.	total: +	-	0
mar.	0	1	1	0	7	9	6	45
unc.	2	0	1	0	6	9	3	48
ent.	2	0	0	1	7	10	4	46
spe.	2	1	1	0	7	11	1	48
ran.	0	1	1	0	0	2	27	31

query-by-bagging

	mar.	unc.	ent.	spe.	ran.	total: +	-	0
mar.	0	4	4	7	9	24	0	36
unc.	0	0	4	5	5	14	4	42
ent.	0	0	0	2	7	9	10	41
spe.	0	0	1	0	5	6	16	38
ran.	0	0	1	2	0	3	26	31

co-testing

	mar.	unc.	ent.	spe.	ran.	total: +	-	0
mar.	0	5	6	3	5	19	0	41
unc.	0	0	3	0	3	6	12	42
ent.	0	0	0	0	3	3	20	37
spe.	0	4	5	0	5	14	5	41
ran.	0	3	6	2	0	11	16	33

active-decorate

	mar.	unc.	ent.	spe.	ran.	total: +	-	0
mar.	0	2	2	6	7	17	1	42
unc.	1	0	1	6	7	15	3	42
ent.	0	0	0	5	7	12	5	43
spe.	0	0	1	0	5	6	19	35
ran.	0	1	1	2	0	4	26	30

Table 3. Comparison by number of training instances using ranks

active learner	margin	unc. samp.	entropy	specific	random
query-by-boosting	2.79	3.08	2.38	2.00	4.75
query-by-bagging	1.92	2.42	2.88	3.38	4.42
co-testing	1.50	3.17	4.00	2.88	3.46
active-decorate	2.25	2.50	2.67	3.67	3.92

Before we compare the performance of disagreement measures for each active learning strategy, we would like to direct the attention on the consistent results of both evaluation techniques. High scores in pairwise comparison correspond to first ranks and vice versa. Furthermore, insignificant differences in the total scores of pairwise comparison are reflected in small variation between ranks.

When query-by-boosting serves as active learner, all disagreement measures are distinct superior to a random sequence of queries. The differences between individual measures though is very small and shows, except for a slight advantage of specific disagreement, no distinction. The results for query-by-bagging support again the general superiority of any disagreement measure over random query selection. Yet, a clear distinction between the disagreement measures exists. Margins achieve the best results, followed by uncertainty sampling, entropy and finally specific disagreement. The performance of co-testing is closely connected to the applied disagreement measure. Again, margins dominate the other approaches. Note, that only margins and specific disagreement perform better than a random strategy. The results for active-decorate show a general superiority of all disagreement measures over a random sequence, in the case of specific disagreement the distinction is only marginal. The descending order of margin-, uncertainty sampling- and entropy-based disagreement as found in query-by-bagging and co-testing is preserved, although the distance between the methods is much smaller.

To summarize the results, whenever a distinction between disagreement measures is obvious, margins receive the best results followed by uncertainty sampling and entropy. The quality of specific disagreement varies for different active learning strategies. While query-by-bagging and co-testing react very sensitive to different disagreement measures (in case of co-testing the application of uncertainty sampling and entropy even leads to worse results than a random query selection), query-by-boosting and active-decorate perform robust on all disagreement measures.

How can we explain the results? The poor performance of entropy-based disagreement may have already been anticipated from Fig. 1. It shows a broad and unspecific selection of queries. In fact, to give an example, the entropy of two distributions $d_1 = (0.5, 0.5, 0)$ and $d_2 = (0.77, 0.015, 0.015)$ is equal. Yet, d_1 is a good candidate for querying while d_2 is not. The good performance of margin-based disagreement can be explained by its focus on competitive strategies, among which it selects equally between uniform and non-uniform distributions. Neither uncertainty sampling-based nor specific disagreement, which shift the focus to a more or less uniform distribution respectively, perform as well. It implies that both, very undirected ensemble decisions as well as decisions which focus on a few highly confident choices, are essential to active learning and should not be disregarded. The robustness of query-by-boosting and active-decorate took us by surprise. We believe that as both ensemble methods interfere with the distribution of their training data, they are able to compensate the choice of less informative queries. However, a definite answer to this behavior needs further research.

5 Conclusion and Future Work

In this paper we present a detailed study which compares commonly used disagreement measures for multi-class ensemble-based active learning. We compare the measures empirically on four active learning strategies, namely query-by-boosting, query-by-bagging, co-testing and active-decorate. In a comprehensive set of experiments on 12 UCI domains we show the superiority of margin-based disagreement, which should be used as a standard approach. In addition, our evaluation shows that the sensibility to disagreement measures varies between active learning strategies. In future work we would like to improve the margin-based approach by enhancing it with further information on the class distribution. We also plan to expand our studies to include disagreement measures that base on the individual class probability distributions of the ensemble members.

References

1. Abe, N., Mamitsuka, H.: Query learning strategies using boosting and bagging. In: Proc. of ICML-98, Morgan Kaufmann (1998) 1–9
2. Muslea, I., Minton, S., Knoblock, C.A.: Selective sampling with redundant views. In: Proc. of AAAI-00, AAAI Press / The MIT Press (2000) 621–626
3. Melville, P., Mooney, R.: Diverse ensembles for active learning. In: Proc. of ICML-04, ACM (2004) 584–591
4. Lewis, D.D., Catlett, J.: Heterogeneous uncertainty sampling for supervised learning. In: Proc. of ICML-94, ACM (1994) 148–156
5. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: Proc. of SIGIR-94, ACM / Springer (1994) 3–12
6. Dagan, I., Engelson, S.: Committee-based sampling for training probabilistic classifiers. In: Proc. of ICML-95, Morgan Kaufmann (1995) 150–157
7. McCallum, A., Nigam, K.: Employing em and pool-based active learning for text classification. In: Proc. of ICML-98, Morgan Kaufmann (1998) 350–358
8. Melville, P., Yang, S.M., Saar-Tsechansky, M., Mooney, R.: Active learning for probability estimation using jensen-shannon divergence. In: Proc. of ECML-05, Springer (2005) 268–279
9. Seung, H.S., Opper, M., Sompolinsky, H.: Query by committee. In: Proc. of COLT-92, ACM (1992) 287–294
10. Freund, Y., Seung, H.S., Shamir, E., Tishby, N.: Selective sampling using the query by committee algorithm. Machine Learning **28**(2-3) (1997) 133–168
11. Breiman, L.: Bagging predictors. Technical report 421, University of California, Berkeley (1994)
12. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Proc. of ICML-96, Morgan Kaufmann (1996) 148–156
13. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proc. of COLT-98, ACM (1998) 92–100
14. Nigam, K., Ghani, R.: Analyzing the effectiveness and applicability of co-training. In: Proc. of CIKM-00, ACM (2000) 86–93
15. Muslea, I.: Active Learning with Multiple Views. PhD thesis, University of Southern California (2002)
16. Melville, P., Mooney, R.: Constructing diverse classifier ensembles using artificial training examples. In: Proc. of IJCAI-03, Morgan Kaufmann (2003) 505–510
17. Blake, C.L., Merz, C.J.: Uci repository of of machine learning databases. (http://www.ics.uci.edu/∼mlearn/MLRepository.html)
18. Witten, I.H., Frank, E.: Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco (2000)

Active Learning with Irrelevant Examples

Dominic Mazzoni, Kiri L. Wagstaff, and Michael C. Burl

Jet Propulsion Laboratory, California Institute of Technology,
Mail Stop 126-347, 4800 Oak Grove Drive, Pasadena CA 91109, USA,
kiri.wagstaff@jpl.nasa.gov

Abstract. Active learning algorithms attempt to accelerate the learning process by requesting labels for the most informative items first. In real-world problems, however, there may exist unlabeled items that are irrelevant to the user's classification goals. Queries about these points slow down learning because they provide no information about the problem of interest. We have observed that when irrelevant items are present, active learning can perform worse than random selection, requiring more time (queries) to achieve the same level of accuracy. Therefore, we propose a novel approach, **Relevance Bias**, in which the active learner combines its default selection heuristic with the output of a simultaneously trained relevance classifier to favor items that are likely to be both informative and relevant. In our experiments on a real-world problem and two benchmark datasets, the Relevance Bias approach significantly improves the learning rate of three different active learning approaches.

1 Introduction

For many classification problems, class labels must be generated by a domain expert and are therefore expensive to acquire. Active learning [1] attempts to reduce this burden by incrementally selecting the most useful items for labeling. Current active learning approaches assume that the expert can provide a valid label for any item in the data set. In this work, we investigate what happens when this assumption is violated due to the presence of *irrelevant* examples (items that cannot be assigned to any of the valid classes). This may occur because the example belongs to an irrelevant class or because it is ambiguous. For example, in the realm of handwritten digit recognition, irrelevant examples include scanning errors, such as smudges or non-digit characters, or truly ambiguous examples, such as a '7' with a very short upper bar that therefore looks like a '1'. Existing active learning methods have no mechanism for handling irrelevant examples.

An obvious strategy for dealing with irrelevant items is to filter the data before training the classifier. While this is a reasonable solution for a benchmark data set, it is unrealistic as a general solution, especially with very large data sets. The process of filtering is as labor-intensive as labeling the entire data set. Since the purpose of active learning is to achieve high performance without requiring that every item be labeled, it is necessary for active learners to be able to work with the unfiltered data.

This paper offers two main contributions. First, we propose an active learning framework where “irrelevant” is a valid response from the expert labeler (Section 2). In this framework, we demonstrate that several popular active learning methods are sufficiently sensitive to irrelevant examples that in some cases they perform worse (that is, learn more slowly) than a random selection (passive) strategy. We also discuss why placing the irrelevant items into a new class and using a multi-class active learning method is ineffective. We then present our second contribution, **Relevance Bias**, a method by which any active learning method can learn to avoid querying irrelevant items (Section 3). Finally, we present experimental results in Section 4 that demonstrate the improvements achieved by active learners with a Relevance Bias and conclude in Section 5.

2 Active Learning and Irrelevant Items

We focus on *pool-based* active learning, where the learner has access to a (fixed) pool of items for which it can request labels. We assume the existence of a pool $\mathcal{U} = \{x_i\}$ of unlabeled items. Each x_i is a d-dimensional vector in Euclidean space, and the items are assumed to be i.i.d. according to an unknown fixed distribution $P(x)$. For simplicity, we will discuss active learning in the context of binary classification. In traditional active learning, there exists a classification label $y_i \in \{\pm 1\}$ for each x_i that is available, upon request, from the expert labeler. We refer to the expert's labeling of x as $f(x)$. Let $\mathcal{L}$ be the set of items for which the learner has already requested labels. In each round, the active learner selects an unlabeled item x from $\mathcal{U}$ and receives its label, $y = f(x)$. The learner then updates its classifier based on $\mathcal{L} \cup \{(x, y)\}$. In this section, we describe several active learning methods and then discuss how the active learning problem changes when irrelevant items exist.

2.1 Active Learning Algorithms

Although active learning is not restricted to any single inductive learning technique, much of the recent work in this area has focused on active learning for support vector machines (SVMs) [2] due to their strong performance on a variety of problems. An SVM is a binary classifier that constructs a hyperplane in d dimensions to separate the two classes. In particular, it seeks the hyperplane that will maximize the *margin*, or distance between each class and the hyperplane. For classes that are not linearly separable, the SVM implicitly maps each point into a higher-dimensional space via a *kernel function*, which often improves separability. All active learning methods seek to select the item x which, when labeled, provides the greatest accuracy improvement. In this work, we consider four data selection methods: three active learning methods and passive (random) selection.

1. **Simple Margin (“Simple”):** [3] Rank each example $x \in \mathcal{U}$ by its distance from the hyperplane and then choose item x with the smallest distance.

2. **MaxMin Margin:** [3] Empirically test which item x will be most effective, in terms of maximizing the separation between the two classes (and therefore minimizing the size of the version space), regardless of which label it receives.
3. **Diverse:** [4] Choose item x that simultaneously minimizes distance to the hyperplane and maximizes the diversity of the new training set, $\mathcal{L} \cup \{(x, y)\}$.
4. **Random:** Choose item x randomly.

Probabilistic Active Learning. Because each of the active learning algorithms described above proceeds heuristically, it will not always be advantageous to select the top-ranked query. Therefore, for each algorithm, we instead used a variant that sorts all of the examples in the pool according to the active learning algorithm's heuristic, and then chooses an item at random from the top $p\%$ of the pool instead of choosing the top-ranked example. In our experiments, with $p = 10\%$, we determined that these probabilistic active learners outperformed the "strict" algorithms by 1–6% on all three data sets and never resulted in decreased performance. Therefore, in the remainder of the paper we will report results using probabilistic active learning.

2.2 A Framework for Active Learning with Irrelevant Items

Applying active learners when irrelevant items are present requires a modified learning framework. We model the new expert labeler as a function h that maps items to *three* possible values, $y_i \in \{-1, 0, +1\}$. A value of 0 indicates that the label is irrelevant to the learning task at hand. Again, on each round, the active learner applies a selection function to choose an unlabeled item x from $\mathcal{U}$. The learner proceeds normally unless $h(x) = 0$, in which case it acquires no new information and must wait until the following round to make a new request. For some problems, waiting until the next round can be extremely expensive. For example, we have investigated using active learning to select initial conditions for an asteroid collision simulator [5]. Determining the "label" (outcome) for each set of initial conditions requires running a numerical simulation algorithm for days, and a significant amount of time is lost due to an irrelevant query.

Using this framework, we are able to study each active learning method's response to the presence of irrelevant items. The ideal active learner would avoid the irrelevant items completely, since they cannot help improve the margin or reduce the size of the feature space. However, if the active learner's heuristic for item selection coincides with the kind of irrelevant examples that are present, the learner may devote the majority of its queries to the irrelevant items. In Section 4, we will show experimentally that this problem is significant enough that it can cause active learning methods to perform worse than random selection. In keeping with the classification goals, we evaluate performance on the relevant items only.

An intuitive approach to dealing with irrelevant items would be to use a multi-class active learner and provide it with three classes: the positive, negative, and irrelevant items. However, this approach requires that the active learner devote resources (queries) to accurately modeling the irrelevant class, which detracts

```
SELECT-RB(active learner $\mathcal{A}$, relevance classifier $C^*$)
1. Rank all items $x$ in the unlabeled pool $\mathcal{U}$ using $\mathcal{A}$.
2. Normalize all $\mathcal{A}(x)$ scores into the range $[0, 1]$.
3. Calculate $P(\text{relevant}|x)$ using $C^*$.
4. Select $x$ such that $\mathcal{A}(x) \times P(\text{relevant}|x)$ is maximized, and request $y = h(x)$,
the label for $x$.
5. If $y = 0$ (irrelevant), add $x$ to $\mathcal{R}$; otherwise, add $x$ to $\mathcal{L}$. Re-train $C^*$.
```

Fig. 1. Pseudo-code for adding a Relevance Bias to active learner $\mathcal{A}$

from the real classification goal. In the next section, we introduce our approach, which cleanly separates the goals of (1) high performance on the relevant classes and (2) minimizing the number of irrelevant queries.

3 Solution: Active Learning Relevance Bias

We propose an active learner that, in addition to selecting the most informative items for labeling, also learns to separate relevant from irrelevant items. This approach can be adopted by any existing active learning method. The new active learner collects irrelevant items x in a set $\mathcal{R}$ of rejected queries, then trains an additional classifier to distinguish between the set of examples in $\mathcal{L}$ (the labeled set, both positive and negative examples) and the examples in $\mathcal{R}$. The active learner uses this relevance classifier, C^*, to influence its decision about which example should be chosen next. Classifier C^* is unlikely to be 100% accurate, especially in early rounds, so relying on it to strictly filter the pool $\mathcal{U}$ may not yield the best results. Instead, we use C^* to influence the active learner's *rankings* of items in the pool, creating a **Relevance Bias (RB)**. Figure 1 outlines pseudo-code that replaces a normal active learner's item selection method. In step 4, the learner combines its ranking of the items with the probability that they are relevant to yield a final decision about which item to query. In our experiments, using C^* as a modifier rather than a filter increased performance by up to 30%.

We trained an SVM for C^* using the same parameters (kernel function and regularization parameter) as were used for $\mathcal{A}$. However, any learning method that outputs a probability can be used. When C^* is an SVM, $P(\text{relevant}|x)$ can be approximated in several ways, such as clipping values outside the range $[-1, 1]$ and mapping them to the range $[0, 1]$, or using Platt's technique of mapping the SVM outputs to a sigmoid probability model [6]. We used the former method in this work.

4 Experimental Results

To evaluate the effectiveness of our proposed approach, we conducted experiments on a real-world data set and two benchmark problems. Key data set characteristics, including SVM parameters, are shown in Table 1. The disjoint test sets are composed solely of relevant items.

Table 1. Summary of the data sets, including the SVM parameters used, the choice of positive, negative, and irrelevant classes, and number of items in each class. All experiments used an RBF kernel.

	Number of	Training set (# items)			SVM params	
Data set	features	Positive	Negative	Irrelevant	γ	C
MISR	156	cloudy (100)	clear (100)	dusty (100)	1.0	1.0
DNA	180	EI (50)	IE (50)	neither (50)	2^{-6}	8.0
MNIST	784	'1' (100)	'7' (100)	others (800)	1.0	1.0

MISR. This data set consists of the pixels in an image that was collected by the Multi-angle Imaging SpectroRadiometer (MISR) instrument in Earth orbit over the Sahara Desert on February 6, 2004. The goal is to build a classifier that distinguishes cloudy and clear pixels. This particular image also contains several dusty pixels, which fall into neither class and are therefore considered irrelevant. For each pixel, we extracted 156 features: the bidirectional reflectance factor for the pixel and a subset of the pixels within a 5x5 neighborhood, from four different spectral bands and from cameras viewing the scenes from three different angles. We selected 300 pixels randomly from the image to form the training set, 100 each of the cloudy, clear, and dusty classes.

DNA. This is the `dna` data set from the StatLog repository [7]. The goal is to use the DNA sequence on either side of a splice junction to distinguish between exon/intron boundaries (EI sites, or "donors") and intron/exon boundaries (IE sites, or "acceptors"). Some splice junctions fall into neither category (irrelevant). Each feature vector consists of 180 features (60 DNA base pairs, each encoded using three binary features). We ran experiments on 150 examples chosen randomly from the training set of 2000 examples, using the SVM hyperparameters as in Hsu and Lin's one-vs-one experiments [8].

MNIST. This data set consists of scanned images of handwritten digits [9]. Each image is composed of 28x28 pixels. We used a subset of the full data set that contained 1000 items, 100 from each class (digit). For these experiments, we focus on learning to distinguish between digits 1 and 7, which is one of the more difficult cases. The 800 items representing the eight other digits are all irrelevant items, since they are neither 1's nor 7's.

4.1 Active Learning Results

We performed an empirical comparison for all three active learning methods against random selection and the RB-enhanced versions of each method. In addition to plotting learning curves, we also compute a scalar value that captures overall performance, AUC (area under the curve). The AUC of algorithm $\mathcal{A}$ after n rounds is the sum of the accuracy it obtained at each round from 1 to n, normalized by n.

$$\mathrm{AUC}_n(\mathcal{A}) = \frac{1}{n}\sum_{t=1}^{n} \mathrm{Acc}_t(\mathcal{A}) \tag{1}$$

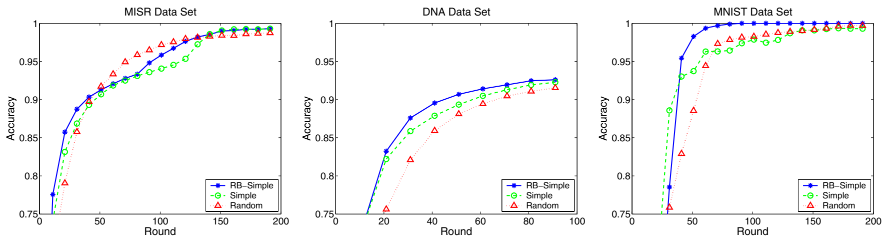


Fig. 2. Active learning (Simple) with and without a Relevance Bias (100 trials)

If n is clear, we omit the subscript. In general, a higher AUC indicates faster, more efficient learning.

Figure 2 shows the behavior of Simple, RB-Simple, and Random on each data set (the other active learners are omitted from the figures to reduce clutter). The quantified AUCs observed for all three active learners are shown in Table 2. We observe several cases where regular active learning performs worse than random selection. The MaxMin algorithm is particularly sensitive to the presence of irrelevant items in the MISR and MNIST data sets. Overall, we find that RB-Diverse yields the best performance.

With the sole exception of Simple's performance on the MNIST data set, adding a Relevance Bias to each method improves performance (AUC), sometimes dramatically. To more clearly illustrate this phenomenon, Figure 3 (top) shows the (cumulative) number of irrelevant items that each algorithm selected. We can see that Simple has a definite bias towards selecting irrelevant items (more than expected by random chance). In contrast, RB-Simple chooses far fewer irrelevant items and correspondingly achieves higher performance.

Because the performance of an RB-enhanced active learner is dependent on the accuracy of its C^* classifier, we also examined the C^* learning curves. Figure 3 (bottom) shows these curves for Simple, RB-Simple, and Random (evaluation is over a separate set of items not included in $\mathcal{U}$). C^* quickly achieves a high level of performance on the MISR and DNA data sets. However, C^* learns more slowly on the more complex MNIST problem (80% of the data set is irrelevant), explaining why RB-Simple's AUC is not an improvement over Simple.

We also tested the multi-class approach (Section 2.2) for each of our data sets. In each case, we trained a multi-class active learner [10] to discriminate between

Table 2. AUC for three active learners compared to Random, the RB-enhanced versions, and Multi-class active learning (100 trials). The best result for each data set is in bold, and results that are significantly worse than Random (95% confidence level) are italicized.

			Simple		MaxMin		Diverse		
Data set	# Rounds	Random	Regular	RB	Regular	RB	Regular	RB	Multi-class
MISR	200	90.6	90.3	91.3	*65.5*	*76.7*	90.7	**92.0**	*86.8*
DNA	100	79.3	81.7	**82.8**	78.1	78.8	81.7	**82.8**	*73.5*
MNIST	200	87.1	88.0	88.0	*54.0*	*65.0*	89.4	**90.5**	*73.2*

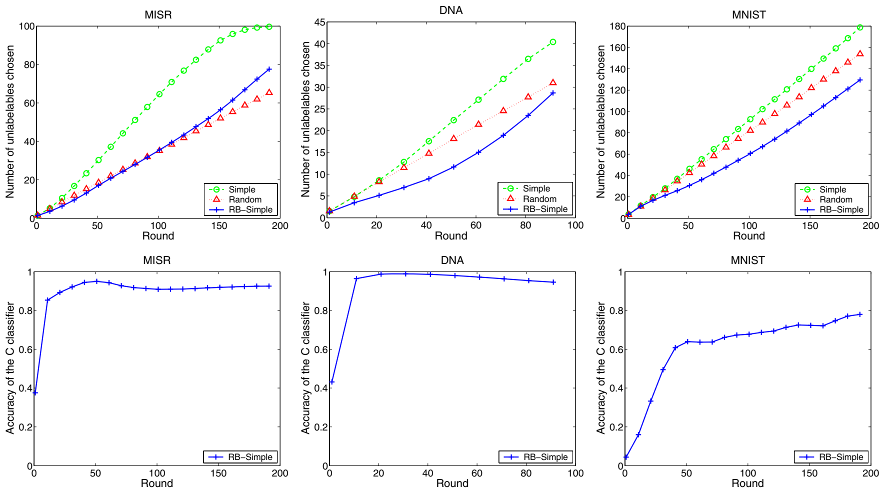


Fig. 3. Top: Comparison between Simple, Random, and RB-Simple in terms of number of irrelevant examples queried; lower values are better. Bottom: Learning curves for RB-Simple's C^* (relevance) classifier. All results are averaged over 100 trials.

the positive, negative, and irrelevant classes. As shown in Table 2, the multi-class active learner also performs worse (learns more slowly) than Random, because it aims to simultaneously maximize performance over all three classes and must therefore devote a large number of queries to the third (irrelevant) class. In contrast, the Relevance Bias approach can provide performance gains even when C^* is not yet fully accurate, because the majority of its effort is devoted to modeling the two critical classes. This intuition is confirmed experimentally: we find that RB-Simple strongly outperforms the multi-class learner, by 5–19%.

5 Conclusions and Future Work

Active learning enables the application of machine learning methods to problems where it is difficult or expensive to acquire expert labels. Real-world data sets may contain items that are irrelevant to the user's classification goals. When filtering these items is not a realistic option, it is essential that active learning methods be able to cope with the presence of irrelevant items. However, existing active learning algorithms can perform worse than passive learning in this situation.

We have proposed a new active learning framework that allows for any item to be assigned to an "irrelevant" class. We presented a novel method, Relevance Bias, by which any active learning algorithm can be modified to avoid irrelevant examples by training a second classifier to distinguish between the relevant and irrelevant items. This method consistently improves the performance of active learners when irrelevant items are present. We have also shown that the multi-class approach does not perform as well as the Relevance Bias approach.

An important extension to this work will be to add a mechanism that compensates for the fact that C^* cannot perfectly distinguish relevant from irrelevant items (especially early on). For example, the algorithm could estimate C^*'s accuracy via leave-one-out cross-validation and adjust the strength of the relevance bias over time. Thus, initial rounds will be more exploratory, while later ones can rely more strongly on C^*'s recommendations.

Acknowledgments. We thank Dennis DeCoste for suggesting the initial idea for this work and for valuable suggestions and feedback along the way. We also thank David Diner, Roger Davies, and Michael Garay for working with us on MISR cloud classification, and Rebecca Castaño and Robert Granat for other valuable feedback. The research described in this paper was funded by the NASA AIST Program and was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

1. Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. Machine Learning **15**(2) (1994) 201–221
2. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20** (1995) 273–297
3. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. Journal of Machine Learning Research **2** (2002) 45–66
4. Brinker, K.: Incorporating diversity in active learning with support vector machines. In: Proceedings of the Twentieth International Conference on Machine Learning, Washington, D. C. (2003) 59–66
5. Burl, M.C., DeCoste, D., Enke, B., Mazzoni, D., Merline, W.J., Scharenbroich, L.: Automated knowledge discovery from simulators. In: Proceedings of the Sixth SIAM International Conference on Data Mining. (2006)
6. Platt, J.C.: Probabilities for SV Machines. In Smola, A.J., Bartlett, P., Schölkopf, B., Schuurmans, D., eds.: Advances in Large Margin Classifiers, MIT Press (1999) 61–74
7. Michie, D., Spiegelhalter, D., Taylor, C.: Machine Learning, Neural and Statistical Classification. Prentice Hall, Englewood Cliffs, N.J. (1994) Data available at `http://www.liacc.up.pt/ML/statlog/`.
8. Hsu, C., Lin, C.: A comparison of methods for multi-class support vector machines. IEEE Transactions on Neural Networks **13** (2002) 415–425
9. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324
10. Tong, S.: Active Learning: Theory and Applications. PhD thesis, Stanford University (2001)

Classification with Support Hyperplanes

Georgi I. Nalbantov[1,2], Jan C. Bioch[2], and Patrick J.F. Groenen[2]

[1] ERIM, Erasmus University Rotterdam
[2] Econometric Institute, Erasmus University Rotterdam
{nalbantov, bioch, groenen}@few.eur.nl

Abstract. A new classification method is proposed, called Support Hyperplanes (SHs). To solve the binary classification task, SHs consider the set of all hyperplanes that do not make classification mistakes, referred to as semi-consistent hyperplanes. A test object is classified using that semi-consistent hyperplane, which is farthest away from it. In this way, a good balance between goodness-of-fit and model complexity is achieved, where model complexity is proxied by the distance between a test object and a semi-consistent hyperplane. This idea of complexity resembles the one imputed in the width of the so-called margin between two classes, which arises in the context of Support Vector Machine learning. Class overlap can be handled via the introduction of kernels and/or slack variables. The performance of SHs against standard classifiers is promising on several widely-used empirical data sets.

Keywords: Kernel Methods, Large Margin and Instance-based Classifiers.

1 Introduction

Consider the task of separating two classes of objects from each other on the basis of some shared characteristics. In general, this separation problem is referred to as the (binary) classification task. Some well-known approaches to this task include (binary) Logistic Regression, k-Nearest Neighbor, Decision Trees, Naive Bayes classifier, Linear and Quadratic Discriminant Analysis, Neural Networks, and more recently, Support Vector Machines (SVMs).

Support Hyperplanes (SHs) is a new instance-based large margin classification technique that provides an implicit decision boundary using a set of explicitly defined functions. For SHs, this set consists of all hyperplanes that do not misclassify any of the data objects. Each hyperplane that belongs to this set is called a *semi-consistent* hyperplane with respect to the data. We first treat the so-called separable case – the case where the classes are perfectly separable by a hyperplane. Then we deal with the nonseparable case via the introduction of kernels and slack variables, similarly to SVMs. The basic motivation behind SHs is the desire to classify a given test object with that semi-consistent hyperplane, which is most likely to classify this particular object correctly. Since for each new object there is a different such semi-consistent hyperplane, the produced decision surface between the classes is implicit.

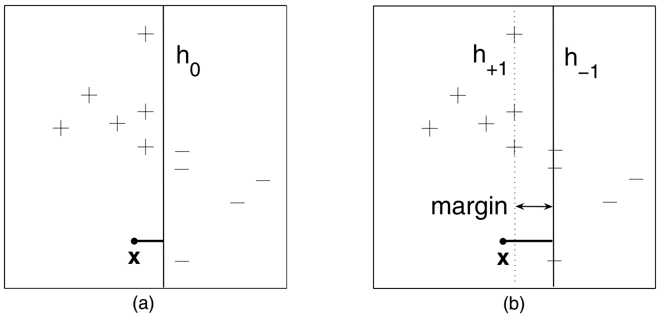


Fig. 1. Two equivalent ways to apply the SVMs classification rule. In Panel (a), the test point $\mathbf{x}$ receives the label (+1) assigned using hyperplane h_0, $\sum_{i=1}^{l} y_i \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}) + b = 0$. In Panel (b), the same test point receives the label (+1) assigned using the farthest away semi-consistent hyperplane from it (h_{-1}, $\sum_{i=1}^{l} y_i \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}) + b = -1$), which is parallel to another semi-consistent hyperplane (h_{+1}, $\sum_{i=1}^{l} y_i \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}) + b = 1$) in such a way that the distance between these two hyperplanes is maximal.

An advantage of the SHs method is that it is robust against outliers and avoids overfitting. Further, we demonstrate empirically that the SHs decision boundary appears to be relatively insensitive to the choice of kernel applied to the data. The SHs approach is more conservative than SVMs, for instance, in the sense that the hyperplane determining the classification of a new object is more distant from it than any of the hyperplanes forming the so-called margin in SVMs. It can be argued that the SHs approach is more general than SVMs by means of a formulation of the SHs decision boundary that is nested into the formulation of the SVMs decision boundary.

2 Support Vector Machines for Classification

We start with an account of the SVM classifier, developed by Vapnik ([11]) and co-workers. SVMs for binary classification solve the following task: given training data $\{\mathbf{x}_i, y_i\}_{i=1}^{l}$ from $\mathbb{R}^n \times \{-1, 1\}$, estimate a function $f : \mathbb{R}^n \rightarrow \{-1, 1\}$ such that f will classify correctly unseen observations $\{\mathbf{x}_j, y_j\}_{j=l+1}^{l+1+m}$. In SVMs, the input vectors $\{\mathbf{x}_i\}_{i=1}^{l}$ are usually mapped from $\mathbb{R}^n$ into a higher-dimensional space via a mapping φ, in which the vectors are denoted as $\{\varphi(\mathbf{x}_i)\}_{i=1}^{l}$. In this higher-dimensional (or, feature) space, the SVM method finds the hyperplane that maximizes the closest distance between the observations from the two classes, the so-called margin, while at the same time minimizes the amount of training errors ([2], [4], [11]). The optimal SVM hyperplane is found by solving the following quadratic optimization problem:

$$\begin{array}{ll} \max\limits_{\boldsymbol{\alpha}} & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} & 0 \leq \alpha_i \leq C, \ i = 1, 2, \ldots, l, \ \text{and} \ \sum_{i=1}^{l} y_i \alpha_i = 0, \end{array} \qquad (1)$$

where $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)'\varphi(\mathbf{x}_j)$ is a *Mercer* kernel that calculates the inner product of input vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ mapped in feature space. Using the optimal α's of (1) the SVM hyperplane h_0, $\mathbf{w}'\varphi(\mathbf{x}) + b = 0$, can be expressed as $\sum_{i=1}^{l} y_i\alpha_i\kappa(\mathbf{x}_i, \mathbf{x}) + b = 0$. Here, $\mathbf{w}$ is a vector of hyperplane coefficients, and b is the intercept. A test observation $\mathbf{x}$ receives the class label assigned using h_0, as shown in Fig. 1a. Stated equivalently, $\mathbf{x}$ is classified using the farthest-away hyperplane that is semi-consistent with the training data, which is parallel to another semi-consistent hyperplane in such a way that the distance between these two hyperplanes is maximal (see Fig. 1b).

3 Support Hyperplanes

3.1 Definition and Motivation

Just like SVMs, the SHs address the classification task. Let us focus on the so-called linearly separable case, where the positive and negative observations of a training data set D are perfectly separable from each other by a hyperplane. Consider the set of semi-consistent hyperplanes. Formally, a hyperplane with equation $\mathbf{w}'\mathbf{x}+b = 0$ is defined to be semi-consistent with a given data set if for all data points $i = 1, 2, \ldots, l$, it holds that $y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 0$; the same hyperplane is defined to be consistent with the data if for all data points $i = 1, 2, \ldots, l$, it holds that $y_i(\mathbf{w}'\mathbf{x}_i + b) > 0$. The basic motivation behind Support Hyperplanes (SHs) is the desire to classify a test observation $\mathbf{x}$ with that semi-consistent hyperplane, which is in some sense the most likely to assign the correct label to $\mathbf{x}$. The extent of such likeliness is assumed to be positively related to the distance between $\mathbf{x}$ and any semi-consistent hyperplane. Thus, if $\mathbf{x}$ is more distant from hyperplane h_a than from hyperplane h_b, both of which are semi-consistent with D, then h_a is considered more likely to classify $\mathbf{x}$ correctly than hyperplane h_b. This leads to the following classification rule of SHs: *a test point* $\mathbf{x}$ *should be classified using the farthest-away hyperplane from* $\mathbf{x}$ *that is semi-consistent with the training data.* Intuitively, this hyperplane can be called the "support hyperplane" since it supports its own judgement about the classification of $\mathbf{x}$ with greatest self-confidence; hence the name Support Hyperplanes for the whole method. For each test point $\mathbf{x}$ the corresponding support hyperplane is different. Therefore, the entire decision boundary between the two classes is not explicitly computed. A point is defined to lie on the SHs decision boundary if there exist two different semi-consistent hyperplanes that are farthest away from it. SHs consider the distance between a test point $\mathbf{x}$ and a semi-consistent hyperplane as a proxy for complexity associated with the classification of $\mathbf{x}$. Under this circumstance, the best generalizability is achieved when one classifies $\mathbf{x}$ with the so-called support hyperplane: the semi-consistent hyperplane that is most distant from $\mathbf{x}$. If one however considers the width of the margin as a proxy for complexity, then the SVM hyperplane achieves the best generalizability. Notice that by definition the support hyperplane is at least as distant from $\mathbf{x}$ as any of the two semi-consistent hyperplanes that form the margin of the optimal SVM hyperplane, which makes

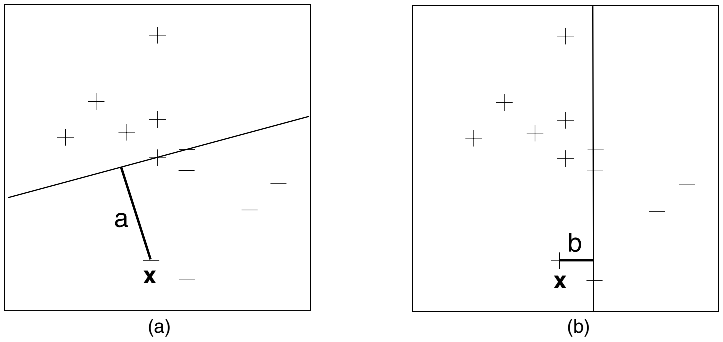


Fig. 2. Classification with Support Hyperplanes in two steps. At stage one (Panel (a)), a test point $\mathbf{x}$ is added as class "–" to the original data set that consists of "+" and "–" labeled points, and the distance a from $\mathbf{x}$ to the farthest away semi-consistent hyperplane is computed. At stage two (Panel (b)), $\mathbf{x}$ is added to the original data set as class "+", and the distance b from $\mathbf{x}$ to the farthest away semi-consistent hyperplane is computed. If $a > b$ ($a < b$), then $\mathbf{x}$ is assigned to class "–" ("+").

the SHs method relatively more conservative. Let us now argue more formally that the SH approach generalizes SVM by means of a formulation of the SHs decision boundary that is part of a formulation of the SVMs decision boundary. A point $\mathbf{x}$ is defined to lie on the implicit SHs separation surface if the following three conditions are met: (1) $\mathbf{x}$ is equally distant from two hyperplanes, (2) these two hyperplanes are semi-consistent with the training data, and (3) the distance between point $\mathbf{x}$ and any of the two hyperplanes is maximal. Next, observe that a point $\mathbf{x}$ is defined to lie on the explicit SVMs optimal hyperplane if and only if the three conditions above plus an additional fourth condition are all satisfied: (4) the two (semi-consistent) hyperplanes are parallel to each other.

3.2 Estimation

Given a linearly separable data set D, $\{\mathbf{x}_i, y_i\}_{i=1}^{l}$, from $\mathbb{R}^n \times \{-1, 1\}$, SHs classify a test point $\mathbf{x}_{l+1}$ using that semi-consistent hyperplane with respect to D, which is most distant from $\mathbf{x}_{l+1}$. Formally, in order to find the support hyperplane $\mathbf{w}'\mathbf{x} + b = 0$ of point $\mathbf{x}_{l+1}$, one solves the following quadratic optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,y_{l+1}} \quad & \frac{1}{2}\mathbf{w}'\mathbf{w} \\ \text{s.t.} \quad & y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 0,\ \ i = 1, 2, \ldots, l \\ & y_{l+1}(\mathbf{w}'\mathbf{x}_{l+1} + b) = 1 \end{aligned} \tag{2}$$

The distance between the support hyperplane $\mathbf{w}'\mathbf{x} + b = 0$ and $\mathbf{x}_{l+1}$ is defined as $1/\sqrt{\mathbf{w}'\mathbf{w}}$ by the last constraint of (2), irrespective of the label y_{l+1}. This distance is maximal when $\frac{1}{2}\mathbf{w}'\mathbf{w}$ is minimal. The role of the first l inequality constraints is to ensure that the support hyperplane is semi-consistent with the training data.

Optimization problem (2) is partially combinatorial, since not all variables are continuous: the label y_{l+1} can take only two discrete values. Therefore, in order to solve (2), two distinct optimization subproblems should we solved (see Fig. 2). One time (2) is solved when y_{l+1} equals $+1$, and another time when y_{l+1} equals -1. Each of these optimization subproblems has a unique solution, provided that the extended data set $\{\mathbf{x}_i, y_i\}_{i=1}^{l+1}$ is separable. In case the two solutions yield the same value for the objective function $\frac{1}{2}\mathbf{w}'\mathbf{w}$, the test point $\mathbf{x}_{l+1}$ lies on the SHs decision boundary and the classification label is undetermined. If the extended data set has become nonseparable when y_{l+1} is labeled, say, $+1$, then the respective optimization subproblem does not have a solution. Then, $\mathbf{x}_{l+1}$ is assigned the opposite label, here -1. A way to detect whether a subproblem has become nonseparable from separable will be described in [8]. The implicit nature of SHs provides for the property that the SHs decision boundary is in general nonlinear, even in case the original data is not mapped into a higher-dimensional space. Figure 3 demonstrates that this property does not hold in general for SVMs. This figure also illustrates that the SHs decision boundary appears to be less sensitive to the choice of kernel and kernel parameters than the respective SVMs boundary.

We now treat the so-called (linearly) nonseparable case. A training data set is said to be nonseparable if there does not exist a single hyperplane that is consistent with it. SHs deal with the nonseparable case in the same way as SVMs: by introducing so-called slack variables. For SHs, this procedure amounts to solving the following quadratic optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,y_{l+1},\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}'\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 0 - \xi_i,\ \ \xi_i \geq 0,\ \ i = 1, 2, \ldots, l \\ & y_{l+1}(\mathbf{w}'\mathbf{x}_{l+1} + b) = 1. \end{aligned} \tag{3}$$

Note that in (3) the points that are incorrectly classified are penalized linearly via $\sum_{i=1}^{l}\xi_i$. If one prefers a quadratic penalization of the classification errors, then the sum of squared errors $\sum_{i=1}^{l}\xi_i^2$ should be substituted for $\sum_{i=1}^{l}\xi_i$ in (3). One can go even further and extend the SHs algorithm in a way analogical to LS-SVM ([5]) by imposing in (3) that constraints $y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 0 - \xi_i$ hold as equalities, on top of substituting $\sum_{i=1}^{l}\xi_i^2$ for $\sum_{i=1}^{l}\xi_i$.

Each of the two primal subproblems pertaining to (3) can be expressed in dual form[1] as:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \alpha_{l+1} - \tfrac{1}{2}\textstyle\sum_{i,j=1}^{l+1}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i'\mathbf{x}_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C,\ \ i = 1, 2, \ldots, l,\ \text{ and } \textstyle\sum_{i=1}^{l+1} y_i\alpha_i = 0, \end{aligned} \tag{4}$$

[1] The derivation of the dual problem resembles the one used in SVMs (see, e.g., [2]).

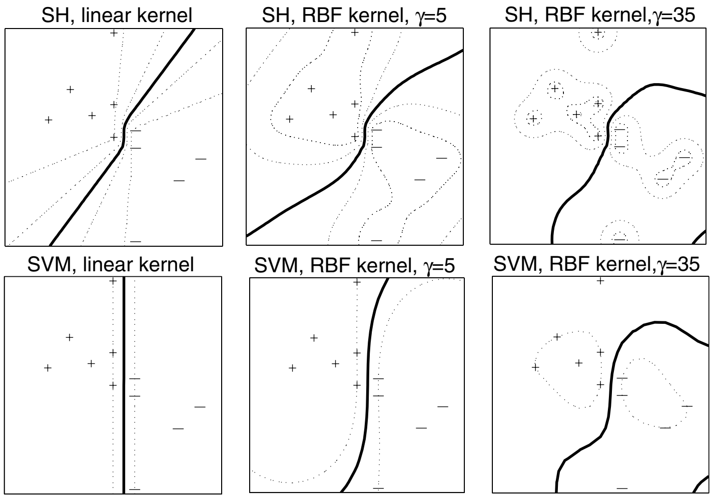


Fig. 3. Decision boundaries for SHs and SVMs using the linear, $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i'\mathbf{x}_j$, and the RBF, $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \parallel \mathbf{x}_i - \mathbf{x}_j \parallel^2)$, kernels on a linearly separable data set. The dashed contours for the SHs method are iso-curves along which the ratio of two distances is constant: the distance from a test point to the farthest semi-consistent hyperplane when it is added to the data set one time as "+", and another time as "–".

where the α's are the Lagrange multipliers associated with the respective subproblem. In the first subproblem $y_{l+1} = 1$, while in the second subproblem $y_{l+1} = -1$. The advantage of the dual formulation (4) is that different *Mercer* kernels can be employed to replace the inner product $\mathbf{x}_i'\mathbf{x}_j$ in (4), just like in the SVMs case. The $(l+1) \times (l+1)$ symmetric positive-definite matrix with elements $\varphi(\mathbf{x}_i)'\varphi(\mathbf{x}_j)$ on the i^{th} row and j^{th} column is called the kernel matrix.

The SHs approach can also be theoretically justified by observing that a kernel matrix used by SHs can be modified to represent the original SHs optimization problem as an SVM problem. It turns out that the theoretical underpinnings for SVMs can also be transferred to the SHs method. More details will be provided in [8].

4 Experiments on Some UCI and SlatLog Data Sets

The basic optimization algorithm for Support Hyperplanes (4) is implemented via a modification of the freely available LIBSVM software ([3]). We tested the performance of Support Hyperplanes on several small- to middle-sized binary data sets that are freely available from the SlatLog and UCI repositories ([9]) and have been analyzed by many researchers and practitioners (e.g. [1], [6], [7], [10] and others): *Sonar*, *Voting*, *Wisconsin Breast Cancer* (W.B.C.), *Heart*, *Australian Credit Approval* (A.C.A.), and *Hepatitis* (Hep.). Detailed information on these data sets can be found on the web sites of the respective repositories.

Table 1. Leave-one-out accuracy rates (in %) of the Support Hyperplanes classifier as well as some standard methods on several binary data sets. Rbf, 2p and lin stand for Radial Basis Function, second-degree polynomial and linear kernel, respectively.

	SH rbf	SH 2p	SH lin	SVM rbf	SVM 2p	SVM lin	NB	LR	LDA	QDA	MLP	*k*NN	DS	C4.5
Sonar	**91.4**	88.0	79.8	88.9	82.2	80.8	67.3	73.1	75.5	74.9	81.3	86.5	73.1	71.2
Voting	96.8	96.3	96.8	96.5	96.3	96.8	90.3	96.5	95.9	94.2	94.9	93.3	95.9	**97.0**
W.B.C.	**97.4**	96.9	97.0	97.0	96.9	96.9	96.0	96.1	96.0	91.4	95.0	97.0	92.4	95.3
Heart	**85.6**	81.9	**85.6**	**85.6**	81.1	**85.6**	83.0	83.7	83.7	81.5	78.9	84.4	76.3	75.2
A.C.A.	**87.4**	86.7	86.8	**87.4**	79.9	87.1	77.1	86.4	85.8	85.2	84.8	85.9	85.5	83.8
Hep.	**87.7**	86.5	86.5	86.5	86.5	86.5	83.2	83.9	85.8	83.9	79.4	85.8	79.4	80.0

We compare the results of SHs to those of several state-of-art techniques: Linear and Quadratic Discriminant Analysis (LDA and QDA), Logistic Regression (LR), Multi-layer Perceptron (MLP), *k*-Nearest Neighbor (*k*NN), Naive Bayes classifier (NB) and two types of Decision Trees – Decision Stump (DS) and C4.5. The experiments for the NB, LR, MLP, *k*NN, DS and C4.5 methods have been carried out with the WEKA learning environment using default model parameters, except for *k*NN. We refer to [12] for additional information on these classifiers and their implementation. We measure model performance by the leave-one-out (LOO) accuracy rate. For our purposes – comparison between the methods – LOO seems to be more suitable than the more general *k*-fold cross-validation (CV), because it always yields one and the same error rate estimate for a given model, unlike the CV method (which involves a random split of the data into several parts).

Table 1 presents performance results for all methods considered. Some methods, namely *k*NN, SHs and SVMs, require tuning of model parameters. In these cases, we report only the highest LOO accuracy rate obtained by performing a grid search for tuning the necessary parameters. Overall, the accuracy rates of Support Hyperplanes exhibit first-rate performance on all six data sets: five times out of six the accuracy rate of SHs is the highest one. SVMs follow closely, and the rest of the techniques show relatively less favorable and more volatile results. For example, the C4.5 classifier performs best on the *Voting* data set, but achieves rather low accuracy rates on two other data sets – *Sonar* and *Heart.* Note that not all data sets are equally easy to handle. For instance, the performance variation over all classifiers on the *Voting* and *Breast Cancer* data sets is rather low, whereas on the *Sonar* data set it is quite substantial.

5 Conclusion

We have introduced a new technique that can be considered as a type of an instance-based large margin classifier, called Support Hyperplanes (SHs). SHs induce an implicit and generally nonlinear decision surface between the classes by using a set of (explicitly defined) hyperplanes. SHs classify a test observation using the farthest-away hyperplane from it that is semi-consistent with the data used for training. This results in a good generalization quality. Although we have treated just the binary case, the multi-class extension can easily be carried out by means of standard methods such as one-against-one or one-against-all classification. A potential weak point of SHs, also applying to SVMs, is that it is not clear a priori which type of kernel and what value of the tuning parameters should be used. Furthermore, we do not address the issue of attribute selection and the estimation of class-membership probabilities. Further research could also concentrate on the application of SHs in more domains, on faster implementation suitable for analyzing large-scale data sets, and on the derivation of theoretical test-error bounds.

References

1. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
2. Burges, C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery **2** (1998) 121–167
3. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2006) Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
4. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press (2000)
5. van Gestel, T.V., Suykens, J.A.K., Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., Moor, B.D., Vandewalle, J.: Benchmarking least squares support vector machine classifiers. Machine Learning **24** (2004) 5–32
6. King, R.D., Feng, C., Sutherland, A.: STATLOG: comparison of classification algorithms on large real-world problems. Applied Artificial Intelligence **9(3)** (1995) 289–334
7. Lim, T., Loh, W., Shih, Y.: A comparison of prediction accuracy, complexity, and training time for thirtythree old and new classification algorithms. Machine Learning **40** (1995) 203–228
8. Nalbantov, G.I., Bioch, J.C., Groenen, P.J.F.: Instance-based classification with support hyperplanes. Econometric Institute technical report, Erasmus University Rottedam (to appear)
9. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI Repository of machine learning databases (1998) `http://www.ics.uci.edu/~mlearn/MLRepository.html` University of California, Irvine, Dept. of Information and Computer Sciences.
10. Perlich, C., Provost, F., Simonoff, J.S.: Tree induction vs. logistic regression: a learning-curve analysis. Journal Of Machine Learning Research **4** (2003) 211–255
11. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer-Verlag New York, Inc. (1995) 2nd edition, 2000.
12. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufman, San Francisco (2005) 2nd edition.

(Agnostic) PAC Learning Concepts in Higher-Order Logic

K.S. Ng

Symbolic Machine Learning and Knowledge Acquisition
National ICT Australia Limited
kee.siong@nicta.com.au

Abstract. This paper studies the PAC and agnostic PAC learnability of some standard function classes in the learning in higher-order logic setting introduced by Lloyd et al. In particular, it is shown that the similarity between learning in higher-order logic and traditional attribute-value learning allows many results from computational learning theory to be 'ported' to the logical setting with ease. As a direct consequence, a number of non-trivial results in the higher-order setting can be established with straightforward proofs. Our satisfyingly simple analysis provides another case for a more in-depth study and wider uptake of the proposed higher-order logic approach to symbolic machine learning.

1 Introduction

Symbolic machine learning is traditionally studied in the field of Inductive Logic Programming (ILP). Within ILP, there is a rich body of work on the PAC-learnability (and non-PAC-learnability) of different classes of first-order logic programs. See, for surveys, [9] and [5]. The arguments used in this kind of analyses are usually intimately and intricately linked to the computation model of first-order logic programming. It is not clear whether these results, which reflect the nature of learning with a first-order language like Prolog, reflect the nature of learning with rich expressive languages in general. To bridge this gap in our understanding, we need to explore learnability issues in formalisms other than first-order logic programming. This paper is an attempt in this endeavour.

In particular, we will look at the higher-order logic approach to symbolic learning expounded in [12]. We will examine the PAC and agnostic PAC learnability of several common function classes definable in this new logical setting. Our main observation is that the similarity in nature between learning in higher-order logic and traditional attribute-value learning allows many results from computational learning theory to be 'ported' with ease to the higher-order setting. A direct consequence of this is that a number of non-trivial results in the logical setting can be shown with relatively straightforward proofs. The simplicity of our analysis, when compared to similar but more technical analyses in ILP, provides another piece of evidence that symbolic machine learning can be fruitfully studied in the higher-order setting proposed in [12].

The paper is organized as follows. I review the learning in higher-order logic setting in §2. The main results of this paper are in §3. §4 then concludes.

2 Learning in Higher-Order Logic

The logic underlying our learning setting is a polymorphically typed, higher-order logic based on Church's simple theory of types. The form of the language is similar to that of a standard functional programming language like Haskell. Indeed, the approach grew out of research into a functional logic programming language called Escher [11]. In what follows, I will assume the reader is familiar with the syntax and terminology of functional programming languages.

We consider only binary classification problems in this paper. In standard attribute-value learning, the set of individuals X is a subset of $\mathbb{R}^m$ for some m, and the hypothesis space H consists of predicates (boolean functions) defined on $\mathbb{R}^m$. The logical setting introduced in [12] extends this basic setup in two ways.

1. The set X is equated with a class of terms called basic terms in a higher-order logic. This class of terms includes $\mathbb{R}^m$ and just about every data type in common use in computer science.
2. The set H is extended to admit any subset of computable predicates on basic terms definable by composing simpler functions called transformations.

We now examine these two points in some detail.

Representation of Individuals. We first look at how training examples are represented in the logic. The basic idea is that each individual x in a labelled example (x, y) should be represented as a closed term. All information is captured in one place. In this sense, learning in higher-order logic is close to the learning from interpretations [6] and learning from propositionalized data [10] settings in ILP. The formal basis for this is provided by the concept of a *basic term*. Essentially, one first defines the concept of a term in higher-order logic. A suitably rich subset is then identified for data modelling. The details of this development can be found in [12]. For the purpose of this paper, it is sufficient to know that a rich catalogue of data types is provided via basic terms, and these include integers, floating-point numbers, strings, tuples, sets, multisets, lists, trees, graphs and composite types that can be built up from these.

We now introduce a simple multiple-instance problem to illustrate the representation language. More complicated applications can be found in [3] and [15]. We have a collection of bunches of keys and a door. A bunch of keys is labelled true ($\top$) iff it contains at least one key that opens the door, false ($\bot$) otherwise. Given the bunches of keys and their labels, the problem is to learn a function to predict whether any given bunch of keys opens the door.

We model a bunch of keys as a set of keys. Each key, in turn, is modelled as a tuple capturing two of its properties: the company that makes it and its length. This leads to the following type declarations.

$$type\ Bunch = \{Key\} \qquad type\ Key = Make \times Length$$

The types *Make* and *Length* have the following data constructors.

$$Abloy, Chubb, Rubo, Yale : Make \qquad Short, Medium, Long : Length$$

A training example may look like this.

$$\mathit{opens}\ \{(\mathit{Abloy}, \mathit{Medium}), (\mathit{Chubb}, \mathit{Long}), (\mathit{Rubo}, \mathit{Short})\} = \top$$

Given a set of such examples, we want to learn the function $\mathit{opens} : \mathit{Bunch} \rightarrow \Omega$. Here and in the following, Ω denotes the type of the booleans.

Given such data, one can proceed with distance-based learning methods by plugging into standard learning algorithms suitable kernels and distance measures defined on basic terms. In this paper, however, we will adopt a more symbolic approach to the problem. For each learning problem, we will need to explicitly define a space of predicates in which to search for a suitable candidate solution. We next describe the mechanism used to define predicate spaces.

Predicate Construction. Predicates are constructed incrementally by composing basic functions called transformations. Composition is handled by the function $\circ : (a \rightarrow b) \rightarrow (b \rightarrow c) \rightarrow (a \rightarrow c)$ defined by $((f \circ g)\ x) = (g\ (f\ x))$.

Definition 1. *A* transformation f *is a function having a signature of the form*

$$f : (\varrho_1 \rightarrow \Omega) \rightarrow \cdots \rightarrow (\varrho_k \rightarrow \Omega) \rightarrow \mu \rightarrow \sigma$$

for $k \geqslant 0$*. The type* μ *is called the* source *of the transformation; and* σ*, the* target *of the transformation. The number* k *is called the* rank *of the transformation.*

Every function is potentially a transformation — just put $k = 0$. Transformations are used to define a particular class of predicates called standard predicates.

Definition 2. *A* standard predicate *is a term of the form*

$$(f_1\ p_{1,1} \dots p_{1,k_1}) \circ \cdots \circ (f_n\ p_{n,1} \dots p_{n,k_n})$$

for some $n \geq 1$*, where* f_i *is a transformation of rank* k_i*, the target of* f_n *is* Ω*, and each* p_{i,j_i} *is a standard predicate.*

In applications, we first identify a class of transformations H of relevance to the problem domain. Having done that, we then build up a class of standard predicates by composing transformations taken from H in appropriate ways. To illustrate this process, we will first look at some useful transformations for the multiple-instance Keys problem introduced earlier.

Example 3. The transformation $\mathit{top} : a \rightarrow \Omega$ defined by $(\mathit{top}\ x) = \top$ for each x is the weakest predicate on the type a one can define.

Example 4. Given terms of type Key, which are tuples, we can introduce the function $\mathit{projMake} : \mathit{Key} \rightarrow \mathit{Make}$ defined by $(\mathit{projMake}\ (t_1, t_2)) = t_1$ to project out the first element. Likewise, we can define $\mathit{projLength}$.

Example 5. Given terms of type Key, we can introduce the transformation $\wedge_2 : (\mathit{Key} \rightarrow \Omega) \rightarrow (\mathit{Key} \rightarrow \Omega) \rightarrow \mathit{Key} \rightarrow \Omega$ defined by $(\wedge_2\ p\ q) = \lambda x.((p\ x) \wedge (q\ x))$ to conjoin predicates on Key.

Example 6. Given terms of type *Bunch*, which are sets of keys, we can introduce the transformation $\mathit{setExists}_1 : (\mathit{Key} \to \Omega) \to \mathit{Bunch} \to \Omega$ defined by $(\mathit{setExists}_1\ p\ q) = \exists x.((x \in q) \wedge (p\ x))$. The term $(\mathit{setExists}_1\ p\ q)$ evaluates to $\top$ iff there exists a key in q that satisfies p. This transformation directly capture the notion of multiple-instance learning.

Example 7. Given the transformations identified so far, we can construct (complex) standard predicates on *Bunch*. For example, the predicate

$$\mathit{setExists}_1\ (\wedge_2\ (\mathit{projMake} \circ (= \mathit{Rubo}))\ (\mathit{projLength} \circ (= \mathit{Medium})))$$

evaluates a set s of keys to $\top$ iff there exists a Rubo key of medium length in s. One can easily construct other examples.

To efficiently enumerate a class of predicates for a particular application, we use a construct called predicate rewrite systems. It is similar in design to Cohen's antecedent description grammars [4]. We give an informal description of it now. A *predicate rewrite* is an expression of the form $p \rightarrowtail q$, where p and q are standard predicates. The predicate p is called the *head* of the predicate rewrite; q, the *body*. A *predicate rewrite system* is a finite set of predicate rewrites. The following is a simple predicate rewrite system for the Keys problem.

$$\mathit{top} \rightarrowtail \mathit{setExists}_1(\wedge_2\ \mathit{top}\ \mathit{top})$$
$$\mathit{top} \rightarrowtail \mathit{projMake} \circ (= C) \quad \text{for each constant } C : \mathit{Make}$$
$$\mathit{top} \rightarrowtail \mathit{projLength} \circ (= C) \quad \text{for each constant } C : \mathit{Length}$$

Roughly speaking, predicate generation works as follows. Starting from an initial predicate r, all predicate rewrites that have r (of the appropriate type) in the head are selected to make up child predicates that consist of the bodies of these predicate rewrites. Then, for each child predicate and each redex in that predicate, all child predicates are generated by replacing each redex by the body of the predicate rewrite whose head is identical to the redex. This generation of predicates continues to produce the predicate class. For example, the following is a path in the predicate space defined by the rewrite system given above.

$$\mathit{top} \rightsquigarrow \mathit{setExists}_1(\wedge_2\ \mathit{top}\ \mathit{top}) \rightsquigarrow \mathit{setExists}_1(\wedge_2\ (\mathit{projMake} \circ (= \mathit{Abloy})\ \mathit{top})$$
$$\rightsquigarrow \mathit{setExists}_1(\wedge_2\ (\mathit{projMake} \circ (= \mathit{Abloy})\ (\mathit{projLength} \circ (= \mathit{Short})))$$

The space of predicates defined by a predicate rewrite system $\rightarrowtail$ is denoted $S_{\rightarrowtail}$.

Given a predicate rewrite system $\rightarrowtail$, we can define more complex function classes in terms of predicates defined by $\rightarrowtail$. Two function classes in actual use [15] we will look at in this paper are

1. k-DT($\rightarrowtail$) – the class of all decision trees of maximum depth k taking predicates in $S_{\rightarrowtail}$ as tests in internal nodes; and
2. k-DL($\rightarrowtail$) – the class of all decision lists [16] where each test in an internal node is a conjunction of at most k predicates in $S_{\rightarrowtail}$. (We can close $S_{\rightarrowtail}$ under negation if we wish to.)

Other common function classes can be defined (and analysed) in a similar way.

3 PAC Learnability of Higher-Order Concepts

We examine the PAC learnability of k-DL($\rightarrowtail$) and k-DT($\rightarrowtail$) in this section, focusing mainly on the former. We assume familiarity with the standard definitions of PAC and agnostic PAC learning. The efficient computability of higher-order predicates is studied in §3.1. Sample complexity questions are briefly looked at in §3.2. With these in place, some results on the efficient (agnostic) PAC learnability of k-DL($\rightarrowtail$) and k-DT($\rightarrowtail$) are given in §3.3 and §3.4.

3.1 Efficiently Computable Predicates

As pointed out in [5], polynomial computability of concept classes is a prerequisite for efficient PAC learnability. We now give a sufficient condition on predicate rewrite systems that will ensure the production of only polynomially computable predicates. Predicate classes defined on such restricted rewrite systems can then be shown to contain only concepts that can be efficiently evaluated.

Definition 8. *A transformation $f : (\varrho_1 \to \Omega) \to \cdots \to (\varrho_k \to \Omega) \to \alpha \to \sigma$ is said to be* polynomial-time computable qua transformation *if $(f\, p_1 \ldots p_k)$ is polynomial-time computable given that each p_i is polynomial-time computable.*

Proposition 9. *Let $f : \alpha \to \sigma$ and $g : \sigma \to \phi$ be polynomial-time computable functions. Then the function $f \circ g$ is polynomial-time computable.*

Proposition 10. *Let T be a set of transformations and let S_T be the set of all standard predicates that can be formed using transformations in T. If every $f \in T$ is polynomial-time computable qua transformation, then every $p \in S_T$ composed of a finite number of transformations is polynomial-time computable.*

Proof. The proof proceeds by induction on the number of transformations in p and uses Proposition 9. □

In defining a predicate rewrite system $\rightarrowtail$, if we restrict ourselves to transformations that are polynomial-time computable qua transformation, then we can be assured that every $p \in S_{\rightarrowtail}$ we will ever construct is polynomial-time computable since $S_{\rightarrowtail} \subseteq S_T$. Further, given such a $\rightarrowtail$, it's easy to show that k-DL($\rightarrowtail$) and k-DT($\rightarrowtail$) contain only polynomially computable predicates.

3.2 Sample Complexity

It is well-known that the (agnostic) PAC learnability of a function class is tightly governed by its VC dimension [2]. The problem of calculating the VC dimension of general predicate classes definable using predicate rewrite systems was previously considered in [14]. The main conclusion reached there is that in the higher-order logic setting, the VC dimension of a predicate class is usually not much lower than the upper bound given by the logarithm of the size of the predicate class. Given this observation, we consider only finite function classes here. A rewrite system $\rightarrowtail$ is said to be *finite* if $S_{\rightarrowtail}$ is finite. We assume from now onwards all predicate rewrite systems are finite. Since all finite function classes are (agnostic) PAC learnable, we are interested primarily in *efficient* learnability.

3.3 Efficient (Agnostic) PAC Learnability of k-DL($\rightarrowtail$)

We now look at the learnability of higher-order decision lists, starting with the PAC learnability of k-DL($\rightarrowtail$). Under reasonable assumptions on the encoding sizes of individuals and the target function t ($size(t) = \log_2 |k\text{-DL}(\rightarrowtail)|$), one can show the following is true; Rivest's proof [16] goes through essentially unchanged.

Proposition 11. *Let X be a set of individuals and $\rightarrowtail$ a finite predicate rewrite system made up of only transformations that satisfy Definition 8. Then the class k-DL($\rightarrowtail$) is efficiently PAC learnable with sample complexity*

$$m(\epsilon, \delta) \leqslant \frac{1}{\epsilon}(O((S_{\rightarrowtail})^l) + \ln \frac{1}{\delta})$$

for some constant l.

The ease with which Proposition 11 can be established should not belie its importance. It extends Rivest's theorem beyond simple decision lists defined on boolean vectors to arbitrarily rich efficiently computable higher-order decision lists defined on arbitrarily complex structured data. In fact, it's worth pointing out that k-DL($\rightarrowtail$) is probably the largest class of functions that has ever been shown to be efficiently PAC learnable.

We now study the learnability of k-DL($\rightarrowtail$) in the more realistic agnostic PAC learning setting. The correct (and only) strategy in this setting is simple: find the predicate in the predicate class with the lowest error on the training examples. (See, for details, [2, Chap. 23].) Computing the decision list with the lowest empirical error given a set of training examples is, unfortunately, a computationally difficult problem. Indeed, one can show that there is no efficient algorithm for this optimization problem in the propositional setting, which is a special case of the general problem, unless P=NP. We now sketch a proof.

The argument is an adaptation of the proof for [2, Thm 24.2]. Consider the following two decision problems.

1. Vertex-Cover : Instance: A graph $G = (V, E)$ and an integer $k \leqslant |V|$.
   Question: Is there a vertex cover $U \subseteq V$ such that $|U| \leqslant k$?
2. DL-Fit : Instance: $z \in (\{0,1\}^n \times \{0,1\})^m$ and an integer $k \in \{1, \ldots, m\}$.
   Question: Is there $h \in$ 1-DL(n) such that $\hat{er}(h, z) \leqslant k/m$?

A vertex cover of a graph $G = (V, E)$ is a set $U \subseteq V$ of vertices such that at least one vertex of every edge in E is in U. In the definition of DL-Fit , $\hat{er}(h, z)$ is defined to be $|\{(x, y) \in z : h(x) \neq y\}|/m$ and 1-DL(n) is as defined in [16].

It is known that Vertex-Cover is NP-hard. One can show that every Vertex-Cover problem can be reduced in polynomial time to a DL-Fit problem. Consider an instance $G = (V, E)$ of Vertex-Cover where $|V| = n$ and $|E| = r$. We assume that each vertex in V is labelled with a number from $\{1, 2, \ldots, n\}$ and we denote by ij an edge in E connecting vertex i and vertex j. The size of the instance is $\Omega(r + n)$. We construct $z(G) \in (\{0,1\}^n \times \{0,1\})^{r+n}$

as follows. For any two integers i, j between 1 and n, let $e_{i,j}$ denote the binary vector of length n with ones in positions i and j and zeroes everywhere else. The sample $z(G)$ consists of the labelled examples $(e_{i,i}, 1)$ for $i = 1, 2, \ldots, n$ and, for each edge $ij \in E$, the labelled example $(e_{i,j}, 0)$. The size of $z(G)$ is $(r+n)(n+1)$, which is polynomial in the size of the original Vertex-Cover instance.

Example 12. Consider the graph $G = \{\{1, 2, 3, 4\}, \{11, 12, 13, 14, 23, 33\}\}$. Then

$$z(G) = \{(1000, 1), (0100, 1), (0010, 1), (0001, 1),$$
$$(1000, 0), (1100, 0), (1010, 0), (1001, 0), (0110, 0), (0010, 0)\}.$$

Using the fact that 1-DL(n) is a subset of linear threshold functions [1], it is easy to show that the following is true.

Proposition 13. *Given any graph $G = (V, E)$ with n vertices and r edges and any integer $k \leqslant n$, let $z(G)$ be as defined above. There is $h \in \mathtt{1\text{-}DL}(n)$ such that $\hat{er}(h, z(G)) \leqslant k/(n + r)$ iff there is a vertex cover of G of cardinality at most k.*

Now, if there is an algorithm that, given a set $\mathcal{E}$ of examples, can find in polynomial time $\arg\min_{h \in 1\text{-}DL(n)} \hat{er}(h, \mathcal{E})$, then it can be used to solve DL-Fit in polynomial time. But since DL-Fit is NP-hard, such an algorithm cannot exists unless P=NP. This means k-DL($\rightarrowtail$) is not efficiently agnostic PAC learnable under standard complexity-theoretic assumptions.

3.4 Efficient (Agnostic) PAC Learnability of k-DT($\rightarrowtail$)

We end the section with brief remarks on the learnability of 1-DT($\rightarrowtail$) (higher-order decision stumps) and k-DT($\rightarrowtail$) for $k > 1$ (higher-order decision trees).

Given a set X of individuals and a predicate rewrite system $\rightarrowtail$, the class 1-DT($\rightarrowtail$) is not efficiently (agnostic) PAC learnable. This is unsurprising since (agnostic) PAC learning 1-DT($\rightarrowtail$) entails an exhaustive search of $S_{\rightarrowtail}$. The run time thus cannot be bounded by a polynomial in $size(t)$.

The efficient learnability of decision trees remains one of the longest-standing open problems in computational learning theory. In particular, it is not known whether the problem of computing the most accurate decision tree given a set of examples is hard. The weak learning framework provides probably the best chance for obtaining positive results; see [8].

4 Discussion and Conclusion

We conclude by comparing our analysis with similar studies in ILP. The comparison is centred around the basic setup of a learning problem. There are two main logical settings in first-order learning: learning from entailment [13] and learning from interpretations [7]. Learning in higher-order logic, being a direct generalization of attribute-value learning, is closer to the latter. The two share the following features with attribute-value learning, which are not found in the learning from entailment setting:

1. examples and background knowledge are separated;
2. examples are separated from one another.

The advantage of making such separations is argued convincingly in [6]. In particular, making such separations allows many results and algorithms from propositional learning to be easily 'upgraded' to the richer settings. This paper provides further evidence in support of this general observation.

In the learning from interpretations setting, there is a price in making such separations in that recursive predicates cannot be learned. This limitation can be overcome in the higher-order setting via the use higher-order functions like `foldr` that package up recursion into convenient forms. This suggests that the basic setup of the learning from interpretations setting is right, but one needs to work in a richer language. Viewed this way, learning in higher-order logic can be understood as taking the natural next step in the direction suggested by the learning from interpretations formulation. This is of course only a retrospective viewpoint; the two formulations were developed very much independently.

References

1. M. Anthony. Decision lists and threshold decision lists. Technical Report LSE-CDAM-2002-11, London School of Economics, 2002.
2. M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations.* Cambridge University Press, 1999.
3. A. F. Bowers, C. Giraud-Carrier, and J. W. Lloyd. A knowledge representation framework for inductive learning. `http://rsise.anu.edu.au/~jwl/`, 2001.
4. W. Cohen. Grammatically biased learning: Learning logic programs using an explicit antecedent description language. *Artificial Intelligence*, 68(2):303–366, 1994.
5. W. W. Cohen and D. Page. Polynomial learnability and inductive logic programming: Methods and results. *New Generation Computing*, 13:369–409, 1995.
6. L. De Raedt. Logical settings for concept learning. *Artificial Intelligence*, 95:187–201, 1997.
7. L. De Raedt and S. Džeroski. First order jk-clausal theories are PAC-learnable. *Artificial Intelligence*, 70:375–392, 1994.
8. M. Kearns and Y. Mansour. On the boosting ability of top-down decision tree learning algorithms. *J. of Computer and System Sciences*, 58(1):109–128, 1999.
9. J.-U. Kietz and S. Džeroski. Inductive logic programming and learnability. *SIGART Bulletin*, 5(1):22–32, 1994.
10. S. Kramer, N. Lavrač, and P. Flach. Propositionalization approaches to relational data mining. In *Relational Data Mining*, chapter 11. Springer, 2001.
11. J. W. Lloyd. Programming in an integrated functional and logic language. *Journal of Functional and Logic Programming*, 3, 1999.
12. J. W. Lloyd. *Logic for Learning: Learning Comprehensible Theories from Structured Data.* Cognitive Technologies. Springer, 2003.
13. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.
14. K. S. Ng. Generalization behaviour of Alkemic decision trees. In *Proc. of the 15th International Conference on Inductive Logic Programming*, pages 246–263, 2005.
15. K. S. Ng. *Learning Comprehensible Theories from Structured Data.* PhD thesis, Computer Sciences Laboratory, The Australian National University, 2005.
16. R. L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

Evaluating Feature Selection for SVMs in High Dimensions

Roland Nilsson[1], José M. Peña[1], Johan Björkegren[2], and Jesper Tegnér[1]

[1] IFM Computational Biology, Linköping University, SE58183 Linköping, Sweden, {rolle, jmp, jespert}@ifm.liu.se

[2] Gustav V Research Institute, Karolinska Institute, SE17177 Stockholm, Sweden johan.bjorkegren@ki.se

Abstract. We perform a systematic evaluation of feature selection (FS) methods for support vector machines (SVMs) using simulated high-dimensional data (up to 5000 dimensions). Several findings previously reported at low dimensions do not apply in high dimensions. For example, none of the FS methods investigated improved SVM accuracy, indicating that the SVM built-in regularization is sufficient. These results were also validated using microarray data. Moreover, all FS methods tend to discard many relevant features. This is a problem for applications such as microarray data analysis, where identifying all biologically important features is a major objective.

1 Introduction

In pattern recognition, feature selection (FS) is traditionally viewed as a preprocessing step that simplifies the task of learning classifiers, rather than a learning objective in itself [1]. On the other hand, there is currently considerable interest within the bioinformatics community to apply FS methods to discover biologically important genes (features) that are not captured by traditional statistical testing [2]. For example, in cancer research, the primary interest is to identify all cancer-related genes from microarray data [3]. This is a fundamentally different problem, because many of the biologically important features may not be needed for classification [4], and will therefore be discarded by FS methods that optimize for classification accuracy.

To assess the applicability of FS methods to microarray data, we performed a systematic evaluation of a number of FS methods in conjunction with SVMs. To our knowledge, this study is the first systematic evaluation of feature set accuracy, and the first to simulate high-dimensional data of the order found in microarray studies.

2 Methods

Throughout, we assume that examples $(x^{(i)}, y^{(i)})$ are independent observations of the random variable pair (X, Y) with distribution $f(x, y)$ on the domain $\mathcal{X} \times \mathcal{Y}$.

We will denote components of a vector x by x_i, and restrictions to a given feature set S by $x_S = \{x_i : i \in S\}$. A classifier is defined simply as a function $g(x) : \mathcal{X} \mapsto \mathcal{Y} = \{-1, +1\}$, predicting a category y for each observed example x. For a given sample size l, a classifier is induced from data $D^l = \{(x^{(i)}, y^{(i)})\}_{i=1}^{l} \in (\mathcal{X} \times \mathcal{Y})^l$ by an inducer (a learning algorithm), defined as a function $I : (\mathcal{X} \times \mathcal{Y})^l \mapsto \mathcal{G}$, where $\mathcal{G}$ is some set of possible classifiers. The optimal (Bayes) classifier g^* is defined as the one that minimizes the *risk* functional

$$R(g) = \sum_{y \in \mathcal{Y}} p(y) \int_{\mathcal{X}} 1\{g(x) \neq y)\} f(x|y) dx \ , \tag{1}$$

In our simulations we use gaussian densities, for which $R(g)$ is easy to calculate directly from (1), and g^* is unique and can be derived analytically. For comparing learning algorithms, are we interested in the overall performance of an inducer I, rather than the performance of a particular g [5]. We evaluate the performance of I using the *expected* risk

$$\rho = E_D[R(I(D^l))] \ . \tag{2}$$

We will also evaluate accuracy with respect to the set of relevant features. This set is defined as [4]

$$S_{\text{REL}} = \{i : \exists S : p(y|x_i, x_S) \neq p(y|x_S)\} \ , \tag{3}$$

where $p(y|x_S)$ is the conditional density of Y after observing $X_S = x_S$. Informally, a feature is relevant if it carries "information" about the target variable Y. Relevant features are either *strongly* or *weakly* relevant; the latter may be "redundant" in the sense that they are not required for optimal classification [4]. The optimal feature set with respect to classification accuracy is defined as

$$S_{\text{OPT}} = \arg\min_S E_{D_S}[R(I_S(D_S^l))] \ , \tag{4}$$

where I_S is a suitable inducer for the data D_S^l, using the features S. In general, S_{OPT} may not be unique, and the minimization may require an exhaustive search among all subsets of S_{REL}, which is an NP-complete problem [6]. However, in our simulations S_{OPT} was identical to the features used by the Bayes rule g^*. Therefore, we are able to measure feature set precision and recall against both S_{REL} and S_{OPT}. In analogue with the risk measure above, for describing the performance of a FS algorithm we consider the distribution of these measures when D^l is a random variable, and estimate the expected value of this distribution.

An overview of our simulation/evaluation procedure is shown in figure 1. For a given data distribution $f(x, y)$, we first take a sample D^l to be used as training data (step 1). We then perform FS and classifier (SVM) induction. Finally, we calculate classifier error probabilities (steps 3,4), the feature sets S_{REL} and S_{OPT}(step 5) and the precision and recall (step 6) with respect to these sets. For each FS algorithm, this process is repeated 100 times and averaged values are reported.

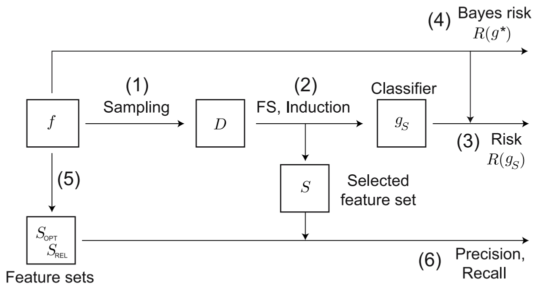


Fig. 1. A schematic view of the evaluation process

We chose five well-known FS methods for evaluation: Pearson Correlation (PC) [1], SVM Naive Weight Rank (WR) [7], Recursive Feature Elimination (RFE) [7], Linear Programming-SVM (LPSVM) [8] and Approximation of the zeRO-norm Minimization (AROM) [9]. PC is a filter method, WR and RFE are wrapper methods, while AROM and LPSVM are embedded methods. We also refer PC, WR and RFE as "ranking methods" since they merely output a ranking of features, while LPSVM and AROM output a set S, thus determining $|S|$ automatically. Throughout, we used a linear SVM [10] as the inducer $I(D^l)$.

For more detailed method descriptions, we refer to the extended version of this paper, available at `www.ifm.liu.se/∼rolle/ecml2006.pdf`.

3 Results

We used a gaussian data distribution for all simulations. This was designed so that a subset of m features $X_1, \dots, X_m$ were relevant to Y, while $X_{m+1}, \dots, X_n$ were irrelevant. Of the m relevant features, $m/2$ were in the optimal feature set; further, half these ($m/4$) were *marginally* independent of Y and thus undetectable by univariate filter methods like PC. We sampled 100 training data points and normalized data to zero mean and unit variance before applying each method. The key parameters to the "difficulty" of the learning problems represented by this data distribution are m and n. We chose a parameter grid $8 \leq m \leq 500$, $20 \leq n \leq 5000$, with values evenly spaced on a logarithmic scale (figure 2A).

The expected risk ρ for the SVM without FS on the (m, n) parameter grid is shown in figure 2A. We find that ρ increases slightly with n, but decreases rapidly with respect to m. Thus, more features is in general better for the SVM: as long as we can obtain a few more relevant features, we can afford to include many irrelevant ones. Therefore, improving SVM performance by FS is very difficult. In particular, an FS method must provide very good recall, or SVM performance will degrade quickly.

To validate our results, we also tested the FS methods on a large microarray data set consisting of 12,600 features (genes) and 136 samples [11]. For

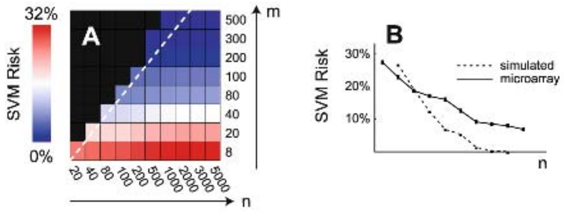


Fig. 2. **A:** Plot of expected SVM risk ρ *vs.* number of relevant features m and total number of features n. **B:** Dotted line, plot of SVM risk *vs.* n for simulations, corresponding to the dotted diagonal in (A). Solid line, SVM risk *vs.* n for microarray data. Here m is unknown but proportional to n.

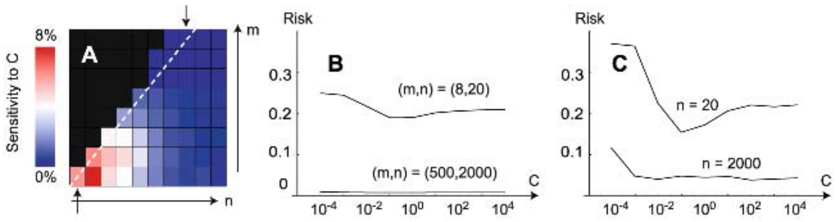


Fig. 3. Sensitivity to the SVM C-parameter. **A:** Sensitivity defined as $\max_C \hat{R} - \min_C \hat{R}$ plotted against m and n. **B:** For simulated data, detailed plot of $\hat{R}$ against C for the cases (m, n) marked by arrows in (A), roughly corresponding to the cases in (C). **C:** For microarray data, plot of $\hat{R}$ against C for $n = 20$ and $n = 2000$.

comparison with our simulations, we first extracted the 5000 features with largest variance and then extracted random subsets of sizes $10, \ldots, 5000$ from these. Although m is unknown in this case, in expectation this procedure gives a constant m/n ratio, since we draw features with equal probability. This roughly corresponds to a diagonal in figure 2A. We selected random training sets of $l = 100$ samples, estimated $\hat{R}(g)$ on the remaining 36 samples, and repeated this process 300 times for each n. The resulting risk estimate was found to agree qualitatively with our simulations (figure 2B).

The value of the regularization parameter C has been found to strongly impact SVM classification accuracy in low dimensions [12]. In our simulations, we optimized C over a range $10^{-4}, \ldots, 10^4$ for each (m, n). We found that C was no longer important in higher dimensions (figure 3A), regardless of the value of m. At lower dimensions, $C \approx 1$ provided good performance (figure 3B), so we fixed $C = 1$ for the remainder of this study. We observed the same (and even stronger) trend for the microarray data (figure 3C). We conclude that the parameter C has virtually no impact on classification accuracy in high dimensions.

Next, we investigated the accuracy of the feature rankings produced by PC, WR and RFE. To address this question without involving the problem of choosing $|S|$, we constructed ROC-curves (figure 4). We found that RFE outperforms

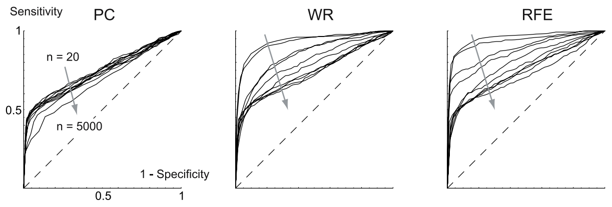


Fig. 4. ROC-curves for the PC, WR and RFE methods. Here we fixed $m = 8$ relevant features and varied $n = 20, \dots, 5000$ as indicated by grey arrows. Dashed diagonals indicate expected result for randomly selected features.

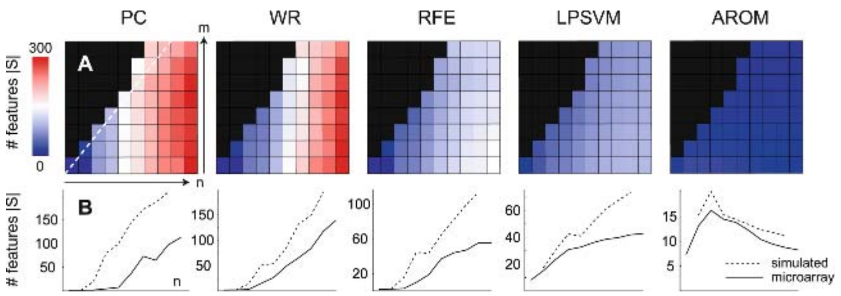


Fig. 5. A: Number of selected features $|S|$ for each (m, n) for simulated data. All plots are scaled equally to (0,300). **B:** Number of selected features $|S|$ *vs.* n, corresponding to the dashed diagonal in (A), for simulated and microarray data. Plots are scaled differently.

WR, which in turn outperforms PC, in agreement with Guyon et al. [7]. This was expected, since 1/4 of the relevant features are not detectable by PC. However, these differences diminished with increasing n. At $n = 5000$, the simpler WR method was as accurate as RFE.

To use ranking methods in practise, a critical issue is how to determine $|S|$ (LPSVM and AROM decide this automatically by heuristics that favor small feature sets [8,9]). A common strategy is to minimize some risk estimate $\hat{R}(g_S)$ for the classifier g_S induced using the feature set S, over a number of possible choices of $|S|$ [13]. For this purpose, we chose the radius-margin bound [14, section 10.7]. Overall, we found that the ranking methods tend to select more features than AROM or LPSVM (figure 5A). We also found that $|S|$ tends to *increase* with n. This can be explained by noting that with better rankings we reach low classifier risk $R(g_S)$ for small $|S|$. Therefore, PC chooses the largest $|S|$, and RFE the smallest. This was also verified for the microarray data (figure 5B). More surprisingly, there was also a tendency for $|S|$ to *decrease with* m (most evident for RFE). This can be understood in the same fashion: the FS problem becomes harder as m decreases, so rankings become more inaccurate and a larger $|S|$ must be used. We conclude that, by selecting $|S|$ to minimize

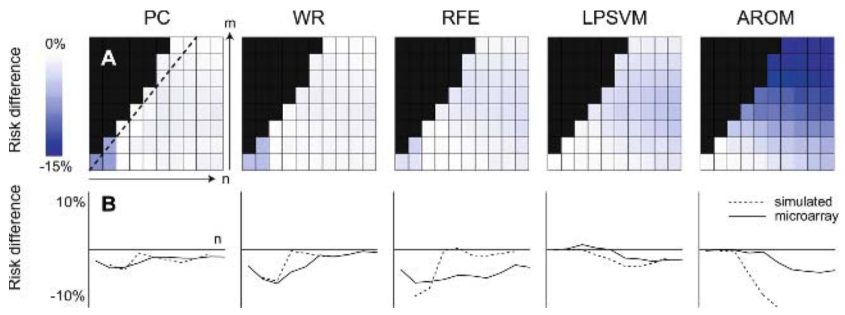


Fig. 6. A: Risk difference $\rho(g) - \rho(g_S)$, using each respective FS method to obtain S (negative means worse accuracy with FS), simulated data. **B:** Estimated risk *vs.* n, corresponding to the dashed diagonal in (A), for simulated and microarray data.

risk, we obtain methods that attempt to control recall but sacrifice precision. LPSVM produced smaller feature sets than RFE, but otherwise exhibited the same tendencies discussed above. Again, the simulation results were consistent with microarray data (figure 5B).

In principle, if $\hat{R}(g_S)$ is accurate, then optimizing this estimate over $|S|$ should at least guarantee that ranking methods do not increase classifier risk. Our simulations verified this: in figure 6A, the difference $\rho(g) - \rho(g_S)$ is close to 0. Thus, the radius-margin bound seems to be accurate, so our results should be attributed to the rankings themselves. LPSVM and AROM worked best around $n \approx 100$ (figure 6A,B), corresponding to the data sets used in the original publications [8,9]. In higher dimensions however, these methods tend to increase the SVM risk. AROM in particular increased ρ by up to 15%, probably because it insisted on very small feature sets. Results on microarray data were similar (figure 6B) except possibly for RFE, which was less accurate on microarray data. In summary, none of the FS methods improved SVM accuracy.

For the simulated data, we measured the accuracy of selected feature sets by precision and recall *vs.* S_{OPT} (figure 7A,B) and S_{REL} (figure 7C,D). There are interesting differences between these two feature sets. Concerning recall *vs.* S_{REL}, we see that PC $>$ WR $>$ RFE $>$ LPSVM $>$ AROM. The filter method PC presumably performs best here since it does not distinguish between strong and weak relevance. In fact, we see that PC selects more weakly than strongly relevant features (since it gives lower recall *vs.* S_{OPT}). In contrast, RFE, LPSVM and AROM have higher recall *vs.* S_{OPT} than *vs.* S_{REL}. All of these methods involve some form of risk optimization, and therefore target S_{OPT}. Consequently, they tend to miss (or, avoid, depending on one's perspective) many of the weakly relevant features. All methods have low precision; AROM provided the best precision, but at the price of lower recall. This is natural since AROM was biased towards very small $|S|$ (figure 5). The remaining methods were comparable.

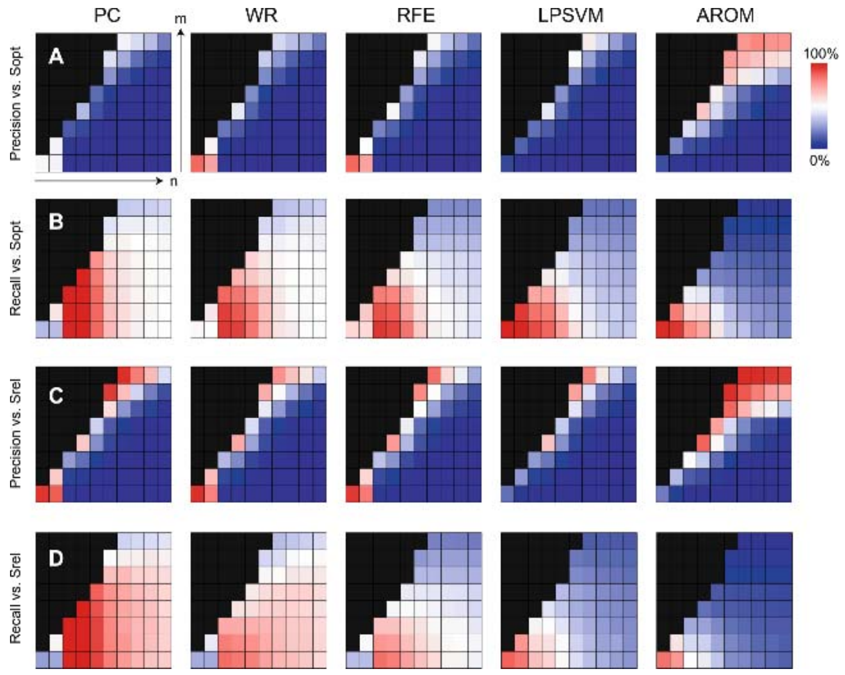


Fig. 7. Feature set accuracy measures for each FS method

4 Discussion

A striking trend in our results is that both classification and feature selection (FS) methods behave very differently in high *vs.* low dimensions. For example, while AROM and LPSVM improve classification accuracy for SVMs for lower n and $m \ll n$ [8,9], we find no improvement at high n (figure 6). Also, while the C-parameter is crucial for low n, it has little or no influence at high n (figure 3). Thus, we recommend that simulation studies of FS methods are performed with dimension comparable to that of real data.

None of the FS methods tested improved SVM classification accuracy in high dimensions. To explain this, one may consider FS as a minimization of the L_0-norm of a vector of feature weights, while the SVM minimizes the L_2-norm [15]. Our results the imply that in high dimensions, the L_2-norm is simply the better choice. For multi-class problems however, there are indications that FS may improve SVM performance [16].

In microarray data analysis, it is often desirable to control precision while maximizing recall [17]. It is clear from figure 7 that none of the methods tested provide such control. A feature selection method that solves this problems would be most useful for microarray data analysis.

Acknowledgements

This work was supported by grants from the Ph.D. Programme in Medical Bioinformatics, the Swedish Research Council (VR-621-2005-4202), Clinical Gene Networks AB and Linköping University.

References

1. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research **3** (2003) 1157–1182
2. Dougherty, E.R.: The fundamental role of pattern recognition for the gene-expression/microarray data in bioinformatics. Pattern Recognition **38** (2005) 2226–2228 Editorial.
3. Golub, T.R., et al.: Molecular classifiation of cancer: class discovery and class prediction by gene expression monitoring. Science **286** (1999) 531–537
4. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence **97** (1997) 273–324
5. Dietterich, T.G.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation **10** (1998) 1895–1923
6. Davies, S., Russel, S.: NP-completeness of searches for smallest possible feature sets. In: Proceedings of the 1994 AAAI fall symposium on relevance, AAAI Press (1994) 37–39
7. Guyon, I., et al.: Gene selection for cancer classification using support vector machines. Machine Learning **46** (2002) 389–422
8. Fung, G., Mangasarian, O.L.: A feature selection newton method for support vector machine classification. Computational Optimization and Applications **28** (2004) 185–202
9. Weston, J., et al.: Use of the zero-norm with linear models and kernel methods. Journal of Machine Learning Research **3** (2003) 1439–1461
10. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3) (1995) 273–297
11. Singh, D., et al.: Gene expression correlates of clinical prostate cancer behavior. Cancer Cell **1** (2002) 203–209
12. Keerthi, S.S.: Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms. IEEE Transactions on Neural Networks **13**(5) (2002) 1225–1229
13. Ambroise, C., McLachlan, G.J.: Selection bias in gene extraction on the basis of microarray gene-expression data. PNAS **99**(10) (2002) 6562–6566
14. Vapnik, V.N.: Statistical Learning Theory. John Wiley and Sons, Inc. (1998)
15. Perkins, S., et al.: Grafting: Fast, incremental feature selection by gradient descent in function space. Journal of Machine Learning Research **3** (2003) 1333–1356
16. Statnikov, A. et al.: A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. Bioinformatics **21**(5) (2005), 631–643
17. Speed, T. (ed.): Statistical Analysis of Gene Expression Microarray Data. Chapman & Hall (2003)

Revisiting Fisher Kernels for Document Similarities

Martin Nyffenegger[1], Jean-Cédric Chappelier[1], and Éric Gaussier[2]

[1] Ecole Polytechnique Fédérale de Lausanne, Switzerland
[2] Xerox Research Center Europe, Meylan, France

Abstract. This paper presents a new metric to compute similarities between textual documents, based on the Fisher information kernel as proposed by T. Hofmann. By considering a new point-of-view on the embedding vector space and proposing a more appropriate way of handling the Fisher information matrix, we derive a new form of the kernel that yields significant improvements on an information retrieval task. We apply our approach to two different models: Naive Bayes and PLSI.

1 Introduction

This paper presents a new way of computing similarities between textual documents based on Fisher kernels and inspired from the method introduced by Hofmann for Probabilistic Latent Semantic Indexing (PLSI) [1]. Fisher kernels define similarities between probabilistic models, based on information-geometry considerations [2]. When applied to documents, they allow the underlying semantics of the documents being compared to be taken into account. They can furthermore be used in both supervised and unsupervised learning contexts.

The next section of this paper introduces the Fisher kernel and the probabilistic models we consider for documents. We then propose two improvements: one arising from considerations about the underlying vector spaces in Sect. 3, and one from the Fisher information matrix approximation in Sect. 4. Finally, the new formulas we derive are evaluated on several Information Retrieval (IR) test collections in Sect. 5.

2 Fisher Kernels for Document Models

Let $\mathcal{D}$ be a collection of N documents: $\mathcal{D} = \{d_1, ..., d_N\}$. Each document consists of a bag-of-words taken from a finite vocabulary $\mathcal{W}$ of M words: $\mathcal{W} = \{w_1, ..., w_M\}$. Latent class models rely on unobserved "semantic classes", represented by some unobserved class variables out of $\mathcal{Z} = \{z_1, ..., z_K\}$. A class represents a set of words describing the same (or a few related) concept(s) or topic(s). We here focus on two latent class models: Naive Bayes (also called "mixture of unigrams") [3,4] and PLSI [5,1], but our method can directly be applied to other latent class models, and more generally to mixture models, for textual documents.

In the Naive Bayes model, each document is generated from a mixture of multinomials: $p(d) = \sum_{z \in \mathcal{Z}} p(d|z)\, p(z)$, with $p(d|z) = \prod_{w \in \mathcal{W}} p(w|z)^{n(d,w)}$, where $n(d,w)$ is the number of occurrences of word w in document d. The parameters of this model are $\theta^{\text{NB}} = \Big(p(z_k), p(w_j|z_k)\Big)$, for $k = 1...K$ and $j = 1...M$.[1]

In the PLSI model, a collection of documents is modelled as a bag of co-occurring (document, word) pairs, the log-likelihood of the collection being $l_d^{\text{PLSI}}(\theta) = \sum_{d \in \mathcal{D}} \sum_{w \in \mathcal{W}} n(d,w) \log p(d,w)$. In this model, documents and words are assumed to be independent conditionally on latent classes, thus: $p(d,w) = \sum_{z \in \mathcal{Z}} p(z)\ p(d|z)\ p(w|z) = p(d)\ \sum_{z \in \mathcal{Z}} p(w|z)\ p(z|d)$. The parameters of this model are $\theta^{\text{PLSI}} = \Big(p(z_k), p(w_j|z_k), p(d_i|z_k)\Big)$, for $k \le K$, $j \le M$, and $i \le N$.[1]

An important difference between Naive Bayes and PLSI lies in the fact that the latter is not, strictly speaking, a generative model: Its reliance on the parameters $p(d_i|z_k)$ prevents the generation of new documents (for which no d_i exist). Yet, this model has proved useful in many practical situations, and has been successfully used in different fields [7,8].

The Fisher kernel, first introduced by Jaakkola and Haussler [2], constitutes a similarity function between data points, derived from a probabilistic model of the data. It measures to which extent two data points "stretch" the model in the same direction: $\mathcal{K}\langle d, d'\rangle = (\nabla_\theta l_d(\theta))^T\, G(\theta)^{-1}\, \nabla_\theta l_{d'}(\theta)$, where $l_d(\theta)$ is the log-likelihood of the parameters θ for document d, and $G(\theta)$ is the "Fisher information matrix", defined as the covariance matrix of the Fisher score.

The Fisher information matrix plays an important role in the above definition as it makes the kernel independent of the chosen parameterization (for equivalent parameterizations, i.e. related through diffeomorphisms). As the Fisher information matrix is difficult to compute, and as its form changes according to the parameterization adopted, it is natural to use a parameterization in which the information matrix can be approximated by the identity matrix. For both Naive Bayes and PLSI, the "square-root re-parameterization" has been deemed to play this role [2,1]. We also use this parameterization here.

For Naive Bayes, $l_d^{\text{NB}}(\theta) = \log \sum_{z \in \mathcal{Z}} p(z) \prod_{w \in \mathcal{W}} p(w|z)^{n(d,w)}$, and the associated Fisher kernel is[2]:

$$\mathcal{K}^{\text{NB}}\langle d, d'\rangle = \sum_{z \in \mathcal{Z}} \frac{p(z|d)p(z|d')}{p(z)} + \sum_{w \in \mathcal{W}} n(d,w)n(d',w) \sum_{z \in \mathcal{Z}} \frac{p(z|d)p(z|d')}{p(w|z)}. \quad (1)$$

For PLSI, $l_d(\theta)$ amounts to:

$$l_d^{\text{PLSI}}(\theta) = \sum_{w \in \mathcal{W}} n(d,w) \log \sum_{z \in \mathcal{Z}} p(z)\ p(w|z)\ p(d|z).$$

[1] These parameters are in fact related through normalization constraints. However, as already pointed out in [1], and confirmed by our own experiments in [6], these constraints can experimentally be ignored in the kernel derivations and have thus, for the sake of clarity, not been included here.

[2] We here omit the derivation, which is mainly technical and similar to the one in [9], even though from a different document likelihood.

Introducing $\widehat{p}(d,w) = n(d,w)/W$, where $W = \sum_d \sum_w n(d,w)$ is the total number of words in the collection, and making the assumption $\sum_w \frac{\widehat{p}(w,d)}{p(w,d)} p(w|z) \approx 1$, the Fisher kernel becomes:

$$\mathcal{K}^{\text{PLSI}}\langle d,d' \rangle = \sum_{z \in \mathcal{Z}} \frac{p(z,d)p(z,d')}{p(z)} + \sum_{w \in \mathcal{W}} \widehat{p}(d,w)\widehat{p}(d',w) \sum_{z \in \mathcal{Z}} \frac{p(z|d,w)p(z|d',w)}{p(w|z)}. \quad (2)$$

Note that the above assumption differs slightly from the one made in [1]: $\sum_w \frac{\widehat{p}(w|d)}{p(w|d)} p(w|z) \approx 1$. As the parameters of PLSI are obtained by maximizing the log-likelihood $l_d^{\text{PLSI}}(\theta) = \sum_{d,w} n(d,w) \log p(d,w)$, they also minimize the KL-divergence between $\widehat{p}(d,w)$ and $p(d,w)$. PLSI thus aims at finding estimates $p(d,w)$ that approximate $\widehat{p}(d,w)$, which naturally lead us to replace the approximation in [1] by the above one.

The above Fisher kernels suffer from the fact that the word counts, $n(d,w)$ or $\widehat{p}(d,w)$, are not normalized by the document length, so that documents are compared on the basis of raw counts instead of frequencies, as results from the IR community suggest (e.g. [10]). In order to get to normalized versions, researchers have considered different pseudo-likelihood functions, either a normalized expected log-likelihood for Naive Bayes [9], or a likelihood normalized by the document length for PLSI [1]. There is however no real theoretical justification for deriving Fisher kernels from the former two pseudo-likelihood functions.

3 Underlying Vector Spaces

Both (1) and (2) exhibit two distinct contributions: one from the class probabilities $p(z)$, and the other from the conditional word probabilities $p(w|z)$. The associated kernels implement a dot product in a feature space, the dimensions of which correspond to the former sets of parameters. There are fundamentally two different types of vector spaces involved in this computation: First, a *class-related* K-dimensional subspace, the dimensions of which correspond to the class-probabilities, then K M-dimensional *word-related* subspaces, the dimensions of which correspond to the vocabulary words.

In the class-related vector space, documents d, with components $[p(z_k|d)]$ or $[p(z_k,d)]$, $k \leq K$, are compared on the basis of the latent topics they contain.

In each of the K word-related subspaces, vectors of word counts $[n(d,w_j)]$ or $[\widehat{p}(d,w_j)]$, $j \leq M$, weighted with the word and document contributions to the current class z_k ($p(w_j|z_k)$ and $p(z_k|d)$ or $p(z_k|d,w_j)$), are compared. The representations of a document d in these K different subspaces only differ on the weighting of its standard bag-of-words representation, $(n(d,w_1), \cdots, n(d,w_M))$, here based on raw counts. Note that neither this representation nor the weighting used in each subspace makes use of any document length normalization, so that longer documents (i.e. containing more words) tend to yield higher similarities with other documents. To compensate this effect, the IR community rather relies on word frequency vectors, leading to the basic document representation $(\widehat{p}(w_1|d), \cdots, \widehat{p}(w_M|d))$, which can then be reweighted in different ways (e.g. with an IDF coefficient, the role of which is here played by the factors $p(w_j|z_k)$) [10].

The above considerations lead us to revisit the original kernels (1) and (2), and introduce a normalization by the document length only in the K word-related vector spaces. This leads to:

$$\mathcal{K}^{\text{NB}'}\langle d, d'\rangle = \sum_{z\in\mathcal{Z}} \frac{p(z|d)p(z|d')}{p(z)} + \sum_{w\in\mathcal{W}} \widehat{p}(w|d)\widehat{p}(w|d') \sum_{z\in\mathcal{Z}} \frac{p(z|d)p(z|d)}{p(w|z)}, \tag{3}$$

$$\mathcal{K}^{\text{PLSI}'}\langle d,d'\rangle = \sum_{z\in\mathcal{Z}} \frac{p(z,d)p(z,d')}{p(z)} + \sum_{w\in\mathcal{W}} \widehat{p}(w|d)\widehat{p}(w|d') \sum_{z\in\mathcal{Z}} \frac{p(z|w,d)p(z|w,d')}{p(w|z)}. \tag{4}$$

Interestingly, the form we arrive at for the Naive Bayes model is equivalent to the one derived in [9], but originates from the actual Fisher kernel. For PLSI, the form we obtain resembles the one obtained in [1], but involves a joint, rather than a conditional, (document, topic) distribution in the *class-related* vector space (first term).

4 Impact of the Fisher Information matrix

As already noticed, most authors approximate the Fisher information matrix $G(\theta)$ by the identity matrix. However, theoretical and experimental considerations show that this approximation is not entirely founded for the models here considered, all the more so that the final form we consider integrates normalization by the document length. The complete computation of the Fisher information matrix however remains too complex to be achieved efficiently in practice. We thus resort to an intermediate form, and assume that the Fisher information matrix could be approximated by a diagonal matrix, which corresponds to the diagonal of the actual Fisher information matrix.

In the case of i.i.d. variables, the Fisher information matrix can be approximated, at a maximum likelihood point, by the following quantity (e.g. [11]): $G(\theta) \approx \sum_{d\in\mathcal{D}} (\nabla_\theta l_d(\theta))^T (\nabla_\theta l_d(\theta))$. The diagonal of this matrix can be efficiently computed as it amounts, for each coefficient, to sum the square of the Fisher scores of the individual documents: $G(\theta)_{(ii)} \approx \sum_{d\in\mathcal{D}} (\nabla_\theta l_d(\theta))^2_{(ii)}$. Using this approximation in the above-defined kernels lead for PLSI to:

$$\mathcal{K}^{\text{PLSI}''}\langle d_i, d_n\rangle = \sum_z \frac{p(z,d_i)p(z,d_n)}{p(z)} \overbrace{\frac{1}{\sum_d \left(\frac{p(z,d)}{\sqrt{p(z)}}\right)^2}}^{\text{contribution from } G(\theta)^{-1}}$$

$$+ \sum_w \widehat{p}(w|d_i)\widehat{p}(w|d_n) \sum_z \frac{p(z|w,d_i)p(z|w,d_n)}{p(w|z)} \overbrace{\frac{1}{\sum_d \left(\widehat{p}(w,d)\frac{p(z|w,d)}{\sqrt{p(w|z)}}\right)^2}}^{\text{contribution from } G(\theta)^{-1}} \tag{5}$$

and to a similar formula for the Naive Bayes model, where the first term is divided by $\sum_d \left(\frac{p(z|d)}{\sqrt{p(z)}}\right)^2$ instead (conditionnal instead of joint).

5 Experimental Results

5.1 Single Kernel

As done in [1], we have tested our approach on four standard IR benchmark collections [12]: CISI (1460 doc.), CRAN (1400 doc.), MED (1033 doc.) and CACM (3204 doc.). For all the tests, documents have been lemmatized and stemmed. As a baseline, we use a standard *tf-idf* kernel, which roughly corresponds to the kernels we have defined (eq. (3) and (4)) for $K = 1$.

In order to validate the different developments presented in the former sections, we have tested the different forms of the Naive Bayes and PLSI kernels for different values of K. The obtained results are shown in Table 1.[3]

The first conclusion we can draw from these results is that the normalization based on the underlying vector spaces (VS) significantly improves the performance of the kernel, and leads to results on par with (CACM & MED) and significantly better than (CISI & CRAN) the standard *tf-idf* approach. The second conclusion is that taking into account the diagonal Fisher information matrix, as presented in Sect. 4, further improves the Naive Bayes kernel. However, for PLSI, the use of the diagonal Fisher information matrix does not yield further improvements[4]. This difference is possibly due to the fact that Naive Bayes yields "harder" assignments of documents to classes (one class only for each document) and is thus expected to have less correlations between its parameters. The diagonal of the Fisher information matrix should thus capture most of the information in this case, which does not seem to be true for PLSI.

5.2 Combination of Kernels

One of the major issues with the use of kernels derived from latent class models is the determination of the number of classes. Two main strategies can address this problem: either resort to model selection techniques to find the optimal value of K, or combine kernels obtained with different values of K. The results presented below follow the latter approach. They are based on a simple linear combination $\mathcal{K}_{\text{full}}\langle d, d'\rangle = \sum_r \alpha_r \; \mathcal{K}_r\langle d, d'\rangle$, where r refers to different values for K.

The weights α_r are determined so as to maximize the R-Precision of the complete kernel $\mathcal{K}_{\text{full}}\langle d, d'\rangle$. In order to keep the computation reasonable, we considered only three kernels: $\mathcal{K}_1$, $\mathcal{K}_{16}$ and $\mathcal{K}_{64}$. The value $K = 1$ was chosen since it yields a kernel similar to a simple *tf-idf* kernel, the good behavior of which is exemplified in Table 1, and was also retained in previously reported experiments [5]. As the combination we rely on should not decrease the results of any of the individual kernels (the weights for the others could be set to 0), we expected

[3] Notice that we used the more usual R-Precision measure instead of the quite unusual 9-pt average precision used in [5] and [1]. Although correlated, the results vary in their absolute values.

[4] A Wilcoxon test shows in this case that the lines VS and DFIM are not significantly different, except for CISI with $K = 32$, for which VS is superior to DFIM.

Table 1. Comparison of the R-Precision for different kernels for both Naive Bayes and PLSI: VS refers to the normalization associated with the different underlying vector spaces, DFIM refers to the use of the diagonal Fisher information matrix. Numbers in bold correspond to the best results, significantly better than the original kernels, as measured with a Wilcoxon test at 1%.

Kernel		CISI		CRAN		MED		CACM	
		$\mathcal{K}_{32}$	$\mathcal{K}_{64}$	$\mathcal{K}_{32}$	$\mathcal{K}_{64}$	$\mathcal{K}_{32}$	$\mathcal{K}_{64}$	$\mathcal{K}_{32}$	$\mathcal{K}_{64}$
tf-idf ($\simeq \mathcal{K}_1$)		.1862		.2381		.4258		.2169	
Naive Bayes									
original	(eq. 1)	.0591	.0575	.0857	.0804	.1131	.1176	.0381	.0388
VS	(eq. 3)	.0834	.0857	.1962	**.1940**	**.1518**	.1470	.1202	.1338
DFIM		**.1017**	**.0998**	**.2238**	**.2005**	**.1554**	**.1570**	**.1411**	**.1548**
PLSI									
Hofmann's (eq. 6 & 7 in [1])		.1013	.1164	.1631	.1505	.3146	.3897	.1100	.1189
VS	(eq. 4)	**.1971**	**.1779**	**.2961**	**.2904**	**.4203**	**.4283**	**.2118**	**.1787**
DFIM	(eq. 5)	.1650	**.1679**	**.2918**	**.2971**	**.4023**	**.4285**	**.1871**	**.2244**

Table 2. R-Precision for the combination $\mathcal{K}_{\text{full}} = \alpha_1\ \mathcal{K}_1 + \alpha_{16}\ \mathcal{K}_{16} + \alpha_{64}\ \mathcal{K}_{64}$, with the corresponding weight values, for the original Hoffman's and DFIM formulations of the PLSI model

	Hofmann's					DFIM				
	α_1	α_{16}	α_{64}	$\mathcal{K}_{\text{full}}$	best($\mathcal{K}_1$,$\mathcal{K}_{16}$, $\mathcal{K}_{64}$)	α_1	α_{16}	α_{64}	$\mathcal{K}_{\text{full}}$	best($\mathcal{K}_1$,$\mathcal{K}_{16}$, $\mathcal{K}_{64}$)
CISI	0.9	0.1	0.0	.2246	.1862	0.1	0.5	0.4	.2003	.1862
CRAN	0.8	0.2	0.0	.3110	.2381	0.2	0.4	0.4	.3395	.2971
MED	0.9	0.1	0.0	.4754	.4248	0.1	0.5	0.4	.4790	.4285
CACM	0.9	0.1	0.0	.2133	.2169	0.0	0.6	0.4	.2350	.2244

that considering additional kernels would improve the results of the *tf-idf* kernel. To assess this, we arbitrarily chose the values $K = 16$ and $K = 64$ for the additional two kernels. We then first normalized the kernels, to have similarities in [0...1] and, setting the constraint $\alpha_1 + \alpha_{16} + \alpha_{64} = 1$, we computed all possible values of the kernel combination by varying α_1 and α_{16} in [0...1] (see Fig. 1). We tested this approach on PLSI, as this model yields the best individual kernel performance. Table 2 summarizes the results obtained.

The first conclusion that can be drawn from these results is that the combination, especially when it is coupled with the DFIM version, has a positive effect on the results, as the R-Precision has improved for almost all collections. The only collection on which it is not the case is CISI, which is not so surprising as CISI is known to be a "difficult case" for content-only approaches, with only a few common words between queries and documents (leading to usually low precision) [12, pp 91–94].

The second conclusion concerns the role of the kernel with $K = 1$, which is predominant in the original Hofmann's version, but somewhat minor in our DFIM

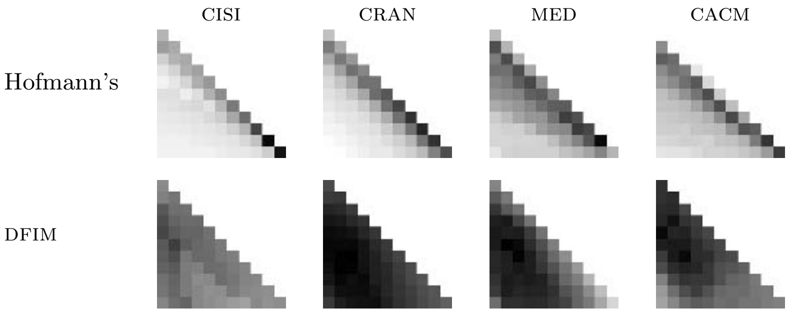


Fig. 1. Graphical representation of the performance of a combination of three kernels with different number of classes, $K = 1$, 16 and 64, for different values of the combinations. The horizontal axis corresponds to α_1 (between 0 and 1) and the vertical axis to α_{16} (also between 0 and 1; $\alpha_{64} = 1 - \alpha_1 - \alpha_{16}$). The greyscale represents the different values of the R-Precision, with white for low values and black for the high values.

version. In order to visualize the impact of each kernel on the combination, we plotted the R-Precision on a 2−dimensional space corresponding to the different values for α_1 (x-Axis, $0 \leq \alpha_1 \leq 1$) and α_{16} (y-Axis, $0 \leq \alpha_{16} \leq 1$). The results obtained are displayed in Fig. 1.

Interestingly, it can be seen from these results that our approach is more robust and homogeneous than the original (Hofmann's) formulation. Indeed, most of the performance of the original formulation is obtained on the $\alpha_{64} = 0$ line (top-left to bottom-right line) and $\alpha_1 \simeq 1$ area (bottom-left part), which shows the predominance of the baseline $K = 1$ system. The peek performance of the DFIM model is, however, obtained in the "average area" ($\alpha_1 \simeq \alpha_{16} \simeq \alpha_{64} \simeq \frac{1}{3}$). Moreover, the area on which the performance is high (dark area) is significantly larger with the DFIM model, which shows that this model is more robust to the values α_r of the combination.

Note however that we did not optimize the combination weights on an independent validation set, but on the entire collection, as, for two of the four collections (MED, 30 queries, and CACM, 52 queries), we did not have enough queries to constitute distinct sets of reasonable size. The robustness of the DFIM model however suggests that the obtained results should be largely independent of the training/validation/test split retained.

6 Conclusion

Starting from the original theory of Fisher kernels and including specific considerations on the underlying vector space for models of documents, we derived new kernel versions for both the Naive Bayes and PLSI models, which we validated experimentally in IR tasks. In these experiments, the new versions outperform the versions previously proposed. The investigation we made on the underlying vector spaces in which kernels are computed, and the different normalizations we introduced in these spaces, yield a general method to adapt Fisher kernels to different situations and tasks. These considerations also allowed us to justify the form of the Fisher kernel for Naive Bayes used in previous works. We also proposed a new approximation to the inverse Fisher information matrix, based on the diagonal of the empirical Fisher information matrix. Although this new approximation does not improve the Fisher kernel for PLSI, it significantly improves the one for Naive Bayes. In the future, we plan to investigate whether the same approach can be applied to other tasks (as document clustering or categorization) and other "generative" kernels.

References

1. Hofmann, T.: Learning the similarity of documents: An information-geometric approach to document retrieval and categorization. In: Advances in Neural Information Processing Systems (NIPS). Volume 12. (2000) 914–920
2. Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. In: Advances in Neural Information Processing Systems (NIPS). Volume 11. (1999) 487–493
3. Lewis, D.: Naive (Bayes) at forty: The independence assumption in information retrieval. In: Proc. of 10th European Conference on Machine Learning (ECML'98). Volume 1398 of Lecture Notes In Computer Science., Springer (1998) 4–15
4. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. Machine Learning **39** (2000) 103–134
5. Hofmann, T.: Probabilistic latent semantic indexing. In: Proc. of 22th International Conference on Research and Development in Information Retrieval. (1999) 50–57
6. Nyffenegger, M.: Similarités textuelles à base de noyaux de Fisher. Master's thesis, Ecole Polytechnique Fédérale de Lausanne, Switerland (2005)
7. Jin, X., Zhou, Y., Mobasher, B.: Web usage mining based on probabilistic latent semantic analysis. In: Proc. of the 10th ACM SIGKDD International Conference on Knowledge discovery and Data Mining (KDD'04). (2004) 197–205
8. Ahrendt, P., Goutte, C., Larsen, J.: Co-occurrence models in music genre classification. In: IEEE Int. Workshop on Machine Learning for Signal Processing. (2005)
9. Vinokourov, A., Girolami, M.: A probabilistic framework for the hierarchic organisation and classification of document collections. Journal of Intelligent Information Systems **18** (2002) 153–172
10. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill, New York (1983)
11. McLachlan, G., Peel, D.: Finite Mixture Models. Wiley (2000)
12. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. Addison-Wesley (1999)

Scaling Model-Based Average-Reward Reinforcement Learning for Product Delivery

Scott Proper[1] and Prasad Tadepalli[2]

[1] Oregon State University, Corvallis, OR 97331-3202, USA,
proper@cs.orst.edu
[2] tadepall@cs.orst.edu

Abstract. Reinforcement learning in real-world domains suffers from three curses of dimensionality: explosions in state and action spaces, and high stochasticity. We present approaches that mitigate each of these curses. To handle the state-space explosion, we introduce "tabular linear functions" that generalize tile-coding and linear value functions. Action space complexity is reduced by replacing complete joint action space search with a form of hill climbing. To deal with high stochasticity, we introduce a new algorithm called ASH-learning, which is an afterstate version of H-Learning. Our extensions make it practical to apply reinforcement learning to a domain of product delivery - an optimization problem that combines inventory control and vehicle routing.

1 Introduction

Reinforcement Learning (RL) provides a nice framework to model a variety of stochastic optimization problems [1]. However, table-based approaches to large RL problems suffer from three "curses of dimensionality": explosions in state and action spaces, and a large number of possible next states of an action due to stochasticity [2]. We propose ways to mitigate these curses in a moderately-sized domain of real-time delivery of products using multiple vehicles with stochastic demands. While RL has been applied separately to inventory control [3] and vehicle routing [4,5,2,6] in the past, we are not aware of any applications of RL to the integrated problem of real-time delivery of products that includes both.

We introduce methods that effectively address each of the curses of dimensionality in the product delivery domain. To mitigate the exploding state-space problem, we introduce "tabular linear functions," (TLFs) which can be viewed as linear functions over some features, whose weights are functions of other features. TLFs generalize tables, linear functions, and tile coding, and allow for a fairly flexible mechanism for specifying the space of potential value functions. We show particular uses of these functions in the product delivery domain that achieve a compact representation of the value function and faster learning.

Second, to reduce the computational cost of searching the action space, which is exponential in the number of agents, we introduce a hill climbing algorithm that scales to a larger number of agents without sacrificing solution quality.

Third, since our base algorithm is model-based, we must calculate the expected value of the next state each step. Many domains have a large stochastic branching factor, i.e., large number of possible next states for a given state-action pair. To mitigate this problem, we introduce ASH-Learning, which is an "afterstate" version of H-learning, and learns by distinguishing between the action-dependent and action-independent effects of the action [1].

In Section 2, we give background on Average-reward Reinforcement Learning (ARL) and H-learning. In Section 3 we illustrate the three curses of dimensionality in the product delivery domain which motivates our research. In Section 4, we describe our solutions to the three curses of dimensionality. In Section 5, we present experimental results and in Section 6 we discuss our results.

2 Average-Reward Reinforcement Learning

We assume that the learner's environment is modeled by a Markov Decision Process (MDP), defined by a 5-tuple $\langle S, A, U, p, r\rangle$, where S is a discrete set of N states, and A is a discrete set of actions. $U(s)$ is the set of actions applicable in state s. By the Markovian assumption, an action u in a given state $s \in S$ results in state s' with some fixed probability $p(s'|s,u)$ and a finite immediate reward $r(s,u)$. A *policy* μ is a mapping from states to actions, such that $\mu(s) \in U(s)$. In Average-reward Reinforcement Learning (ARL), we seek to optimize the average expected reward per time-step, which is called the *gain* [7]. For a given starting state s_0 and policy μ, the gain is given by Equation 1 where $r^\mu(s_0,t)$ is the total reward in t steps when policy μ is used starting at state s_0, and $E(r^\mu(s_0,t))$ is its expected value:

$$\rho^\mu(s_0) = \lim_{t\to\infty} \frac{1}{t} E(r^\mu(s_0,t)) \tag{1}$$

The goal of ARL is to learn a policy that achieves near-optimal gain by executing actions, receiving rewards and learning from them. For any fixed policy, the limit of the difference between the total reward accumulated in time t from a state s and the total reward expected in time t on the average as t tends to infinity is called the *bias* of s and is denoted by $h(s)$. For an optimal policy, the gain ρ and the bias $h(.)$ satisfy the following Bellman equation [7]:

$$h(s) = \max_{u\in U(s)} \left\{ r(s,u) + \sum_{s'=1}^{N} p(s'|s,u)h(s') \right\} - \rho \tag{2}$$

The optimal policy chooses actions maximizing the right hand side of this equation. We use an ARL method called "H-Learning" which is model-based in that it learns and uses explicitly represented action models $p(s'|s,u)$ and $r(s,u)$ [8].

At every step, the H-learning algorithm updates the parameters of the value function in the direction of reducing the temporal difference error (TDE), i.e., the difference between the r.h.s. and the l.h.s. of the Bellman Equation 2.

$$TDE(s) = \max_{u\in U(s)} \left\{ r(s,u) + \sum_{s'=1}^{N} p(s'|s,u)h(s') \right\} - \rho - h(s) \tag{3}$$

We use ϵ-greedy exploration in all our experiments. From Equation 2, it can be seen that $r(s,u) + h(s') - h(s)$ gives an unbiased estimate of ρ, when action u is greedy in state s and s' is the next state. Hence, H-Learning updates ρ as follows in every greedy step:

$$\rho \leftarrow (1-\alpha)\rho + \alpha(r(s,a) - h(s) + h(s')) \tag{4}$$

3 The Product Delivery Domain

To conduct our experiments, we used a version of the product delivery domain (shown in Figure 1) that combines aspects of the vehicle routing problem [9] and inventory control problems. A single product is to be delivered to shops from a depot using several trucks. Shop inventories and truck loads are discretized into 5 levels 0-4. We used 4 trucks, 5 shops, and 10 locations, giving a state-space size of $(5^5)(5^4)(10^4) = 19,531,250,000$, which illustrates the first curse of dimensionality. Each truck has 9 actions available at each time step: unload up to 4 units, move in up to 4 directions, or wait. With 4 trucks, we must consider $9^4 = 6561$ joint actions each step, thus illustrating the second curse of dimensionality. Trucks are loaded automatically upon reaching the depot. A small negative reward of -0.1 is given for every "move" action to reflect fuel costs. We give a reward of -5 if a customer enters a store and finds the shelves empty. We model customer consumption at shops by decreasing the inventory level by 1 unit with some probability, which independently varies from shop to shop. Thus the number of possible next states for a state and an action, called the "stochastic branching factor," is exponential in the number of shops, illustrating the third curse of dimensionality.

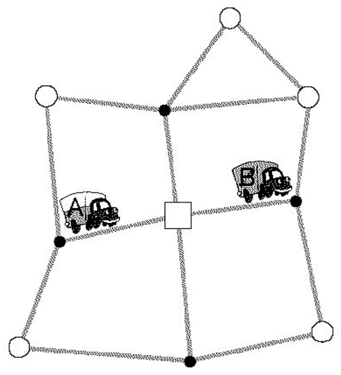


Fig. 1. The Product Delivery Domain: The square in the center is the depot and the circles are the shops

4 Taming the Three Curses of Dimensionality

This section describes our methods for taming the three curses of dimensionality.

4.1 Tabular Linear Functions

We introduce "tabular linear functions," (TLFs) which generalize linear functions, tables, and tile coding. A TLF is a linear function of a set of "linear" features of the state, where the weights of the linear function are arbitrary functions of other discretized or "nominal" features. Hence the weights can be stored in a table indexed by the nominal features, and when multiplied with the linear features of the state and summed, produce the final value function.

More formally, a tabular linear function TLF is represented by Equation 5, which is a sum of n terms. Each term is a product of a linear feature ϕ_i and a weight θ_i. The linear features ϕ_i need not be distinct, although they usually are. Each weight θ_i is a function of m_i nominal features $f_{i,1}, \ldots, f_{i,m_i}$.

$$h(s) = \sum_{i=1}^{n} \theta_i(f_{i,1}(s), \ldots, f_{i,m_i}(s))\phi_i(s) \tag{5}$$

A TLF reduces to a linear function when there are no nominal features, i.e., $\theta_1, \ldots, \theta_n$ are scalar values. One can also view any TLF as a purely linear function where there is a term for every possible set of values of the nominal features:

$$h(s) = \sum_{i=1}^{n} \sum_{k \in K} \theta_{i,k}\phi_i(s)I(f_i(s) = k) \tag{6}$$

Here $I(f_i(s) = k)$ is 1 if $f_i(s) = k$ and 0 otherwise. $f_i(s)$ is an abbreviation of the m_i-component function in Equation 5, K is the set of all of its possible values. TLFs reduce to a table when there is a single term and no linear features, i.e., $n = 1$ and $\phi_1 = 1$ for all states. They reduce to tile coding or coarse coding when there are no linear features, but there are multiple terms, i.e., $\phi_i = 1$ for all i and $n \geq 1$. The nominal features of each term can be viewed as defining a tiling or partition of the state space into non-overlapping regions and the terms are simply added up to yield the final value of the state [1].

Consider, for example, one particular application of TLFs to our product delivery domain. We can represent the value function $h(s)$ by Equation 7:

$$h(s) = \sum_{t=1}^{k} \sum_{x=1}^{n} \theta_{t,x}(p_t, l_t, i_x) \tag{7}$$

where there are k trucks, n shops, and no linear features. The value function has kn terms, each term corresponding to a truck-shop pair (t, x). The nominal features are truck position p_t, truck load l_t, and shop inventory i_x. This is the form of TLF we use in our experiments.

In general, the value function $h(.)$ in ARL is represented as a parameterized functional form of Equation 5 with weights $\theta_1, \ldots, \theta_n$ and linear features $\phi_1, \ldots, \phi_n$. Each weight θ_i is a function of m_i nominal features $f_{i,1}, \ldots, f_{i,m_i}$.

Then each θ_i is updated using the following equation:

$$\theta_i(f_{i,1}(s), \ldots, f_{i,m_i}(s)) \leftarrow \theta_i(f_{i,1}(s), \ldots, f_{i,m_i}(s)) + \beta(TDE(s))\nabla_{\theta_i}h(s) \tag{8}$$

where $\nabla_{\theta_i}h(s) = \phi_i(s)$ and β is the learning rate.

The above update suggests that the value of $h(s)$ would be adjusted to reduce the temporal difference error in state s. The update is exactly the same as that of linear value functions except that only those parameters that belong to the entry that corresponds to the current state's nominal features get the update in proportion to the value of the linear feature.

4.2 Hill Climbing for Action Space Search

To mitigate the exponential growth of the joint action space and the time required to search this action space, we implemented a simple form of hill climbing which greatly sped up the process without a loss in the quality of the resulting policy. We used hill climbing only during training, and used complete action-space search during the evaluation.

Note that every joint action $\boldsymbol{a}$ is a vector of sub-actions, each by a single truck, i.e., $\boldsymbol{a} = (a_1, \ldots, a_k)$. This vector is initialized with all "wait" actions. Starting at a_1, we consider a neighborhood of 8 actions (one for each possible action a_1 may take, other than the action it is currently set to), and $\boldsymbol{a}$ is set to the best action. This process is repeated for each truck $a_2, \ldots, a_k$. The process then starts over at a_1, repeating until $\boldsymbol{a}$ has converged to a local optimum (all agent actions stay the same on one pass over the actions).

4.3 ASH-Learning

One of the drawbacks of model-based RL methods is that they require stepping through all possible next states of a given action to compute the expected value of the next state. Optimizing this step improves the speed of the algorithm considerably. Consider the fact that we need to compute the term $\sum_{s'=1}^{N} p(s'|s,u)h(s')$ in Equation 3 to compute the Bellman error and update the parameters. Since there are usually an exponential number of possible next states in parameters such as the number of shops, doing this calculation by brute-force is expensive.

A method for optimizing the calculation of the expectation is an algorithm we call *ASH-Learning*, or Afterstate H-Learning. This is based on the notion of *afterstates* [1] also called "post-decision states" [2]. We create afterstates by conceptually splitting the effects of an agent's action into "action-dependent" and "action-independent" (or environmental) effects. The afterstate is the state that results by taking into account only the action-dependent effects. We can view the progression of states/afterstates as $s \xrightarrow{a} s_a \rightarrow s' \xrightarrow{a'} s'_{a'} \rightarrow s''$. The "$a$" suffix used here indicates that s_a is the afterstate of state s and action a. The action-independent effects of the environment have created state s' from afterstate s_a. The agent chooses action a' leading to afterstate $s'_{a'}$ and receiving reward $r(s', a')$. The environment again stochastically selects a state, and so on. The h-values may now be redefined in these terms:

$$h(s_a) = E(h(s')) \tag{9}$$

$$h(s') = \max_{u \in U(s')} \left\{ r(s', u) + \sum_{s'_u=1}^{N} p(s'_u|s', u)h(s'_u) \right\} - \rho \tag{10}$$

If we substitute Equation 10 into Equation 9, we obtain this Bellman equation:

$$h(s_a) = E\left[\max_{u \in U(s')} \left\{ r(s', u) + \sum_{s'_u=1}^{N} p(s'_u|s', u)h(s'_u) \right\} - \rho \right] \tag{11}$$

```
1. Find an action $u \in U(s')$ that maximizes $\left\{ r(s',u) + \sum_{s'_u=1}^{N} p(s'_u|s',u)h(s'_u) \right\}$
2. Take an exploratory action or a greedy action in the state $s'$. Let $a'$ be the action
   taken, $s'_{a'}$ be the afterstate, and $s''$ be the resulting state.
3. Update the model parameters $p(s'_{a'}|s',a')$ and $r(s',a')$ using the immediate re-
   ward received.
4. If a greedy action was taken, then
   (a) $\rho \leftarrow (1-\alpha)\rho + \alpha(r(s',a') - h(s_a) + h(s'_{a'}))$
   (b) $\alpha \leftarrow \frac{\alpha}{\alpha+1}$
5. $h(s_a) \leftarrow (1-\beta)h(s_a) + \beta \left( \max_{u \in U(s')} \left\{ r(s',u) + \sum_{s'_u=1}^{N} p(s'_u|s',u)h(s'_u) \right\} - \rho \right)$
6. $s' \leftarrow s''$
7. $s_a \leftarrow s'_{a'}$
```

Fig. 2. The ASH-learning algorithm. The agent executes steps 1-7 when in state s'.

The s'_u notation indicates the afterstate obtained by taking action u in state s'. We estimate the expectation of equation 11 via sampling in step 5 of Figure 2. Since this avoids looping through all possible next states, the algorithm is faster. In our domain, the afterstate is deterministic given the agent's actions, but the stochastic effects of customer actions are unknown. Hence we do not need to learn $p(s'_u|s',u)$, providing a significant savings in computation time.

The temporal difference error for the ASH-learning algorithm is given by:

$$TDE(s_a) = \max_{u \in U(s')} \left\{ r(s',u) + \sum_{s'_u=1}^{N} p(s'_u|s',u)h(s'_u) \right\} - \rho - h(s_a) \tag{12}$$

which we use in Equation 8 when using TLFs for function approximation.

5 Experimental Results

We conducted several experiments testing the methods discussed in Section 4. Tests are averaged over 30 runs of 10^6 time steps for all results. Runs are divided into 20 phases of 48,000 training steps and 2,000 evaluation steps each. During evaluation steps, exploration is turned off and complete search is used to select actions. 4 trucks and 5 shops were used in all the tests. In Figure 3 we compare the results of H-learning and ASH-learning, and complete search of the joint action space vs. hill climbing search. We also compare these results to a fairly sophisticated handcoded non-learning greedy algorithm. This algorithm works by first prioritizing the shops by the expected time until each shop is empty due to customer actions, and then assigning trucks to the highest-priority shops. Once an assignment is made it is straightforward to assign each truck an optimal action. Our results show that ASH-learning outperforms the hand-coded algorithm and H-learning, converging faster to a better average reward.

From Table 1, we see that ASH-learning is very successful at ameliorating the explosion in stochastic branching factor. The largest gains in execution time

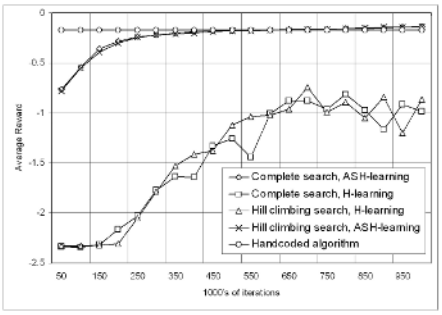


Fig. 3. Comparison of Handcoded algorithm, complete search, Hill climbing, and H- and ASH-learning

Table 1. Comparison of execution times for one run

Search	Algorithm	Seconds
Complete	H-learning	148
Complete	ASH-learning	92
Hill climbing	H-learning	26
Hill climbing	ASH-learning	15

were seen using hill climbing. Combined with ASH-learning, this led to speedups of an order of magnitude. This is because the average number of joint actions considered using hill climbing was about 44, whereas the number of legal truck actions considered by complete search was about 385. Despite this, there was no significant difference in the rate of convergence or the value of the final learned policy. As the number of agents increases beyond 4, we would expect to see even greater improvements in execution times of hill climbing search.

We obtained similar good results using Tabular Linear Functions and ASH-learning in other domains we tested including a taxi domain which is popular in hierarchical RL [10] and a competitive team game domain (which requires strong coordination between agents). We have also developed a multi-agent version of ASH-learning which allowed us to scale up to 10 agents in the team game domain without significant slow down.

6 Discussion and Future Work

We illustrated the three curses of dimensionality of reinforcement learning and showed effective techniques to address them in certain domains. TLFs offer an attractive alternative to other nonlinear forms of function approximation such as neural nets. They converge faster and allow meaningful prior knowledge to be provided. Hill climbing is a cheap but effective technique to mitigate the action-space explosion due to multiple agents. Another promising avenue to explore is the coordination graph approach, where the value function is decomposed in a way that makes the coordination opportunities among agents explicit [11].

We introduced ASH-learning, which is an afterstate version of model-based real-time dynamic programming. It is similar to R-learning in that action- independent effects are not learned or used [12]. However, the value function is state-based, so is more compact than R-learning, much more so for multiple agents. Thus it combines the nice features of both model-based and model-free methods and has proved itself quite well in our domain. Any afterstate-based method is expected to be quite effective in domains where action-dependent effects can be conceptually separated from environmental effects.

In summary, our methods and results add to the accumulating evidence of the effectiveness of average-reward RL, and suggest that the explosions in state space, action space, and high stochasticity may all be ameliorated.

Acknowledgments

This work was supported by the National Science Foundation under grant number: IIS-0329278. The authors would like to thank Hong Tang for his initial experiments and code, and Alan Fern, Neville Mehta, Jason Tracy, and Rasaratnam Logendran for many useful discussions.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. MIT Press (1998)
2. Powell, W.B., Van Roy, B.: Approximate Dynamic Programming for High-Dimensional Dynamic Resource Allocation Problems. In Si, J., Barto, A.G., Powell, W.B., Wunsch, D., eds.: Handbook of Learning and Approximate Dynamic Programming. Wiley-IEEE Press, Hoboken, NJ (2004)
3. Van Roy, B., Bertsekas, D.P., Lee, Y., Tsitsiklis, J.N.: A Neuro-Dynamic Programming Approach to Retailer Inventory Management. In: Proceedings of the IEEE Conference on Decision and Control. (1997)
4. Secamondi, N.: Comparing Neuro-Dynamic Programming Algorithms for the Vehicle Routing Problem with Stochastic Demands. Computers and Operations Research **27**(11-12) (2000)
5. Secamondi, N.: A Rollout Policy for the Vehicle Routing Problem with Stochastic Demands. Operations Research **49**(5) (2001) 768–802
6. Strens, M., Windelinckx, N.: Combining planning with reinforcement learning for multi-robot task allocation. In: Lecture Notes in Computer Science. Volume 3394. (2005) 260–274
7. Puterman, M.L.: Markov Decision Processes: Discrete Dynamic Stochastic Programming. John Wiley (1994)
8. Tadepalli, P., Ok, D.: Model-based Average Reward Reinforcement Learning. Artificial Intelligence **100** (1998) 177–224
9. Bräysy, O., Gendreau, M.: Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. Working Paper, SINTEF Applied Mathematics, Department of Optimisation, Norway (2003)
10. Ghavamzadeh, M., Mahadevan, S.: Learning to communicate and act using hierarchical reinforcement learning. In: AAMAS, IEEE Computer Society (2004) 1114–1121
11. Guestrin, C., Lagoudakis, M., Parr, R.: Coordinated reinforcement learning. In: Proceedings of the 19th International Conference on Machine Learning, San Francisco, CA, Morgan Kaufmann (2002)
12. Schwartz, A.: A Reinforcement Learning Method for Maximizing Undiscounted Rewards. In: Proceedings of the 10th International Conference on Machine Learning, Amherst, Massachusetts, Morgan Kaufmann (1993) 298–305

Robust Probabilistic Calibration

Stefan Rüping

Fraunhofer AIS, Schloss Birlinghoven, 53754 St. Augustin, Germany*
stefan.rueping@ais.fraunhofer.de
http://www.ais.fraunhofer.de

Abstract. Probabilistic calibration is the task of producing reliable estimates of the conditional class probability $P(class|observation)$ from the outputs of numerical classifiers. A recent comparative study [1] revealed that Isotonic Regression [2] and Platt Calibration [3] are most effective probabilistic calibration technique for a wide range of classifiers. This paper will demonstrate that these methods are sensitive to outliers in the data. An improved calibration method will be introduced that combines probabilistic calibration with methods from the field of robust statistics [4]. It will be shown that the integration of robustness concepts can significantly improve calibration performance.

1 Introduction

Given a binary classification task described by an unknown probability distribution $P(X, Y)$ on an input space X and a set of labels $Y = \{-1, 1\}$, a probabilistic classifier is a function $f_{prob} : X \to [0, 1]$ that returns an estimate of the conditional class probability, i.e.

$$f_{prob}(x) \approx P(Y = 1|x).$$

This paper deals with probabilistic classification by calibrating a numerical classifier. That is, for a numerical classification function

$$cl(x) = sign(f_{num}(x))$$

the task is to find an appropriate scaling function $\sigma : \mathcal{R} \to [0, 1]$ such that

$$\sigma(f_{num}(x)) \approx P(Y = 1|x)$$

holds. A recent comparative study [1] revealed that Platt Calibration [3] and Isotonic Regression [2] are very effective probabilistic calibration techniques for a wide range of classifiers. This paper will show that learning a probabilistic classifier is sensitive to outliers in the data and that the performance of a probabilistic classification method can be improved by taking concepts of robust statistics into account.

* The work presented in this paper was done while the author was working at LS Informatik 8, Universität Dortmund.

This paper is structured as follows: Section 2 will give an introduction to performance measures for probabilistic classification and existing calibration methods. In Section 3, the new contribution of this paper will be presented, an investigation of the benefits of robustification in calibration algorithms. Section 4 gives an empirical evaluation of the calibration methods and Section 5 concludes.

2 Probabilistic Classifiers

What is a good probabilistic classifier? This is not trivial to answer, because the true conditional class probability $P(Y = 1|x)$ for an observation x is not known. One requirement is that a probabilistic classifier should be well-calibrated. That is, for each interval of probabilities $[p_1, p_2]$ the probability of drawing a positive example given the classifier predicts $f_{prob}(x) \in [p_1, p_2]$ should also be in $[p_1, p_2]$. However, calibration is not sufficient because it is easy to perfectly calibrate a classifier by assigning the default probability $P(Y = 1)$ to all examples.

A better approach is to measure the error of a probabilistic classifier on a set of examples (x_i, y_i) by a loss function, e.g. the squared loss (Brier score)

$$L_2 = \frac{1}{n} \sum_i \left(f_{prob}(x_i) - \frac{1 + y_i}{2} \right)^2$$

or the cross-entropy loss

$$L_{cre} = -\frac{1}{n} \sum_i \left(\frac{y_i + 1}{2} \log f_{prob}(x_i) + \frac{1 - y_i}{2} \log(1 - f_{prob}(x_i)) \right).$$

For these two loss functions it can be shown that a small error corresponds to a small distance of the distributions $P(Y = 1|x)$ and $f_{prob}(x)$. Both losses differ in the costs they assign to high prediction errors. In the extreme case, the cross-entropy loss is infinite when f_{num} falsely predicts a probability of 0 or 1. This does not happen with the squared loss, which is upper-bounded by 1.

2.1 Probabilistic Calibration Methods

In general, a probabilistic scaler works by letting the numerical base classifier predict the examples x_i in the training set and then fitting a scaling function that maps the predicted values to probabilities in a way that is optimal with respect to the true classes y_i. Special care must be taken to avoid over-fitting the training data. Usually, the predictions that are used as input for the calibration step are generated in an intermediate cross-validation step.

Many probabilistic calibration techniques have been proposed in the literature, for example Softmax Scaling, Binning, using the classifier's precision [5], calibration by Gaussian modeling of the decision function, Beta Scaling [6], Isotonic Regression [2], and Platt Calibration [3]. The two latter methods are the most popular of these methods and will be investigated in this paper. Both methods rely on the following assumption:

Monotonicity Assumption: The true conditional class probability $P(Y = 1|x)$ is an isotonic (monotonically increasing) function of the value of the learners decision function $f_{num}(x)$.

A recent empirical study [1] investigated the dependency between learning algorithms and their optimal calibration strategy. One of the main results of the study was that Platt Calibration works well for maximum margin methods like SVMs or Boosting, which show a similar distortion in the uncalibrated probability estimates, while it is less well suited for Naive Bayes, which shows a different type of distortion. Isotonic Regression was shown to perform consistently well for large data sets, but may suffer from overfitting on small data sets.

3 Robust Probabilistic Calibration Methods

The goal of classification algorithms is to predict the class label itself, not its probability. It follows that in order to do its job, the classifier must get a good estimation of the critical region $P(Y = 1|x) \approx \frac{1}{2}$, while it is irrelevant to get a better fit on very high and very low class probabilities. The form of the decision function far away from the class boundary may be less influenced by the data than by requirements of low complexity, sparsity, or the form of hypothesis space of the learner. For example, in Figure 1 the decision function of a radial-basis SVM is plotted. One can clearly see that near the decision boundary the function resembles a straight line, while on the outside the function is very curvy. This is an effect of the sparsity of the SVM outside the margin.

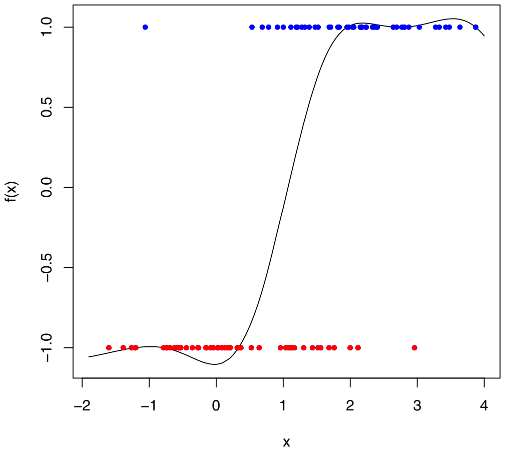


Fig. 1. Artificial one-dimensional data set and decision function of a radial-basis SVM

The general problem of scalers which implement the monotonicity assumption in the presence of outliers is that outliers may receive extreme values of the decision function, which gives them the highest influence on the form of the scaling function. Accordingly, by removing a certain number of the points with extreme decision function values may lead to a more robust scaling function. This effect can be seen in Figure 2. It shows an one-dimensional data set, with most examples generated by two Gaussians at 0 and 2 and a small number of points generated by a Gaussian at 5. The latter points cannot be adequately modeled by the linear learner that was used and become outliers. The linear model was calibrated in two different ways, first by standard Platt Calibration and then by Platt Calibration with the outliers removed. One can clearly see that standard approach does not fit the true conditional class probability $P(Y = 1|x)$ at all, while the robustified version shows a very good fit on most of the examples.

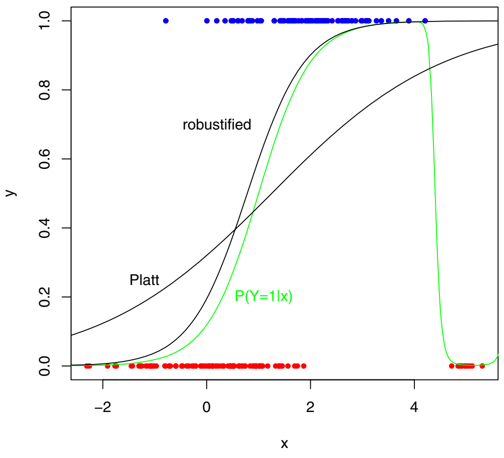


Fig. 2. Linear SVM on an artificial data set calibrated with Platt Calibration and a robustified version thereof

This discussion raises the question whether it is advisable to drop the monotonicity assumption and allow non-monotonic scaling functions. The drawback of this idea is that this allows much more complex functions and, hence, fitting the optimal function becomes much harder. In particular, while a small set of outliers can have a drastic effect on the estimated function, it is very hard to reliably identify a better function based on only a small number of examples. However, even when there is not enough data to correctly model all examples, in many situations there at least may be enough information to remove the hazardous influence of the outlying examples.

The field of Robust Statistics [4] deals with the problem of how much influence small sets of outliers can have. Methods from this field can be used to construct a robust calibration method, i.e. one that is only minimally influenced by outliers. Even if a robust probability estimate may not be sufficient to optimally calibrate all examples, it is a practicable method to limit potential problems and to reliably identify problematic examples.

3.1 Robust Calibration

The results of the discussion so far motivate a robustification of monotonic calibration techniques by adapting the approach of least trimmed loss and least median loss [7,8]. The main idea is to bound the influence of large outliers by removing a fraction τ of the training examples with highest absolute value $|f(x)|$, and then apply regular calibration. This approach is motivated by the least trimmed loss estimator and is based on the idea that the most deteriorating effect comes from outliers that the classifier is most sure about. Ignoring this small number of highly influential points allows the scaler to better concentrate on the bulk of the data.

In order to find an optimal value of τ, it is optimized over values in $[\tau_{min}, 1]$ for some $\tau_{min} > 0.5$. In order to get a reliable estimate of the overall error, the mean of the errors of all training examples is used to select τ. This approach, which is called Trainset-Trimmed Calibration, allows the scaler to adapt to different numbers of outliers in the data in a robust way. A lower bound of $\tau_{min} > 0.5$ is chosen in order to force the learner to predict the larger part of the data and not to fit only a small subset. In the experiments, τ_{min} was set to 0.9.

A more sophisticated, but computationally more expensive way is to select τ by means of an internal cross-validation experiment. That is, several values of τ are tried out using cross-validation of the calibration method on the training set and the optimal τ is used to calibrate to complete training set. This approach is called CV-Trimmed Calibration. Note that this internal cross-validation only requires to repeatedly execute the calibrator, not the actual learner. In the experiments, 10-fold cross-validation has been used. On the other hand, an internal cross-validation reduces the number of available examples, which may have a bad effect on small data sets.

Trimmed Calibration is independent of the underlying calibration method. In the following, Trimmed Calibration applied to Platt Calibration and Isotonic Regression will be called Trimmed Platt and Trimmed Isotonic Regression, respectively.

4 Empirical Comparison

The new probabilistic calibration algorithms have been compared with existing approaches on a total of 21 data sets, with 16 data sets from the well-known UCI machine learning repository [9], and 5 data sets from several real-world applications. The data sets were chosen as to cover a wide range of number of attributes, dimensionality and complexity. For all data sets, continuous attributes were z-scaled, and nominal attributes have been dichotomized. Multi-class data sets have been converted to binary tasks by selecting the two largest the classes or by joining several smaller classes to one. The LETTER data set has been transformed into two classification tasks the same way as in [1]. For data sets with more than 5000 examples, a random sample of size 5000 has been drawn to limit the runtime of the experiments. Table 1 gives some statistics on the data sets.

Table 1. Description of the data sets used in the experiments

Id	Name	Size	Dimension	Id	Name	Size	Dimension
1	adult	32561	104	9	iris	150	4
2	balance	576	4	10	letterP1	20000	16
3	breast-cancer	683	9	11	letterP2	20000	16
4	covtype	4951	48	12	liver	345	6
5	dermatology	184	33	13	mushroom	8124	126
6	diabetes	768	8	14	promoters	106	228
7	digits	776	64	15	voting	435	16
8	ionosphere	351	34	16	wine	178	13
17	business	157	13	20	medicine	6610	18
18	insurance	10000	135	21	garageband	1885	552
19	physics	5000	78				

The base learners in the experiments were a linear Support Vector Machine, a SVM with Radial Basis Kernel, Boosted Decision Stumps, Boosted Decision Trees, Random Forests, Decision Trees, k Nearest Neighbor and Naive Bayes. Parameters for the learners have been set to default values. All results have been obtained using 10-fold cross-validation.

In order to test the significance of the results, the Wilcoxon Signed-rank Test, which has been recently suggested in [10] for performance comparisons over multiple data sets, has been employed. In particular, it is important to compare the approaches over all data sets and all base learners combined, because it may very well be the case that in several experiments there are no outliers that distort calibration, and in this case the robustified version is intended to be identical to the standard version. Hence, finding that both methods are statistically identical in some of the cases is not considered to be a problem.

For Trainset-Trimmed Platt Calibration and both the MSE and CRE error measures the new approach is significantly better (p-values of 0.002 and 0.033) than regular Platt Calibration when compared over all bases learners (detailed results not shown here due to space constraints). However, a closer look into the results reveals that there are several learners for which it performs worse. This problem is healed by the CV-trimmed Platt Calibration, whose results are shown in Table 2; it can be seen that it not only achieves even better p-values over all learners, it also performs clearly better for most of the individual learners. The notable exception are Decision Trees and k Nearest Neighbor for the CRE measure. This is in accordance with [1], which reported that Decision Trees suffer from high variance, which Platt Calibration – in contrast to Isotonic Regression – is not able to deal with. This indicates that the functional form of Platt Calibration is not adequate for Decision Trees in the first place and hence one cannot expect to get meaningful results from this combination at all. Indeed, in the following experiments we will see that robustified Isotonic Regression works clearly better for Decision Trees.

Table 2. Standard Platt Calibration versus CV-Trimmed Platt Calibration

Learner	MSE			CRE		
	% better	% worse	p-value	% better	% worse	p-value
Linear SVM	76.2	23.8	0.006	66.7	33.3	0.010
RBF SVM	76.2	23.8	0.016	57.1	28.6	0.054
Boosted Stumps	61.9	28.6	0.083	38.1	19.0	0.305
Boosted Trees	52.4	38.1	0.152	38.1	42.9	0.370
Random Forest	57.1	33.3	0.023	38.1	38.1	0.449
Decision Tree	42.9	47.6	0.848	38.1	52.4	0.866
k Nearest Neighbor	52.4	38.1	0.118	33.3	47.6	0.630
Naive Bayes	66.7	19.0	0.012	28.6	33.3	0.221
all	60.7	31.5	0.001	42.3	36.9	0.028

Table 3. Standard Isotonic Regression versus Trainset-Trimmed Isotonic Regression

Learner	MSE			CRE		
	% better	% worse	p-value	% better	% worse	p-value
Linear SVM	0.0	0.0	0.500	0.0	0.0	0.500
RBF SVM	0.0	0.0	0.500	0.0	0.0	0.500
Boosted Stumps	28.6	19.0	0.238	23.8	23.8	0.270
Boosted Trees	33.3	33.3	0.450	38.1	28.6	0.265
Random Forest	52.4	9.5	0.006	52.4	9.5	0.002
Decision Tree	47.6	19.0	0.116	52.4	14.3	0.026
k Nearest Neighbor	28.6	23.8	0.175	28.6	19.0	0.093
Naive Bayes	14.3	0.0	0.091	14.3	0.0	0.091
all	25.6	13.1	0.002	26.2	11.9	0.001

The robustification of Isotonic Regression performs slightly different than Platt Calibration. For the trainset-trimmed approach in Table 3 one can see that for both SVMs it performs exactly identical to the standard version. It is important to notice that this is not a bug, but a feature - it may very well be the case that there are no outliers which distort the calibration and hence a value of $\tau = 1$ may be optimal. In total, one can see that for every base learner the trainset-trimmed version of Isotonic Regression performs better than its standard counterpart. Over all learners, it is significantly better with a p-value of 0.002 (MSE) and 0.001 (CRE).

Surprisingly, the CV-trimmed version of Isotonic Regression performs worse than the trainset-trimmed version. An explanation can be found in the learning curve analysis in [1], which found that Isotonic Regression has much problems with overfitting when there is not enough data. Further reducing the number of available examples in an internal cross-validation scheme worsens this situation.

In summary, the mixed approach avoids a breakdown in the case of few examples, but performs worse than selecting the best robustifier for each calibrator. Best results are obtained with CV-trimmed robustification for Platt Calibration and with the Trainset-trimmed method for Isotonic Regression.

5 Conclusions

This paper presented an analysis of the robustification of the two standard probabilistic calibration techniques, Platt Calibration and Isotonic Regression. It was shown that it is possible to significantly improve both methods by safeguarding against the occurrence of outliers in the data. In particular, this paper highlighted the importance of considering the concept of robustness in the design of probabilistic calibration methods.

Acknowledgments

Many thanks to Katharina Morik for her helpful advice in preparing this paper.

References

1. Niculescu-Mizil, A., Caruana, R.: Predicting good probabilities with supervised learning. In: Proceedings of the 22nd International Conference on Machine Learning. (2005) 625–632
2. Zadrozny, B., Elkan, C.: Transforming classifier scores into accurate multiclass probability estimates. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. (2002) 694–699
3. Platt, J.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Smola, A., Bartlett, P., Schölkopf, B., Schuurmans, D., eds.: Advances in Large Margin Classifiers. MIT Press (1999)
4. Huber, P.J.: Robust Statistics. John Wiley & Sons (1981)
5. Rüping, S.: A simple method for estimating conditional probabilities in SVMs. In Abecker, A., Bickel, S., Brefeld, U., Drost, I., Henze, N., Herden, O., Minor, M., Scheffer, T., Stojanovic, L., Weibelza hl, S., eds.: LWA 2004 - Lernen - Wissensentdeckung - Adaptivität, Humboldt-Universität Berlin (2004)
6. Garczarek, U.: Classification Rules in Standardized Partition Spaces. PhD thesis, Universität Dortmund (2002)
7. Rousseeuw, P.J.: Least median of squares regression. J. Am. Stat. Assoc. **79** (1984) 871–880
8. Rousseeuw, P.J., Leroy, A.M.: Robust Regression and Outlier Detection. Wiley (1987)
9. Murphy, P.M., Aha, D.W.: UCI repository of machine learning databases (1994)
10. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7** (2006) 1–30

Missing Data in Kernel PCA

Guido Sanguinetti and Neil D. Lawrence

Department of Computer Science, University of Sheffield
211 Portobello Street, Sheffield S1 4DP, U.K.
{guido, neil}@dcs.shef.ac.uk

Abstract. Kernel Principal Component Analysis (KPCA) is a widely used technique for visualisation and feature extraction. Despite its success and flexibility, the lack of a probabilistic interpretation means that some problems, such as handling missing or corrupted data, are very hard to deal with. In this paper we exploit the probabilistic interpretation of linear PCA together with recent results on latent variable models in Gaussian Processes in order to introduce an objective function for KPCA. This in turn allows a principled approach to the missing data problem. Furthermore, this new approach can be extended to reconstruct corrupted test data using fixed kernel feature extractors. The experimental results show strong improvements over widely used heuristics.

1 Introduction

Kernel PCA is a non-linear feature selection technique which extends the linear statistical method of Principal Component Analysis (PCA) by elegantly using the so called *kernel trick* [1]. However, while the flexibility of Kernel PCA has led to very good performance on a number of problems, the lack of a probabilistic interpretation for it means that it can be very difficult to adapt it in the presence of missing or corrupted data.

In this paper we suggest a simple way of estimating missing data in Kernel PCA. We start by reformulating Kernel PCA along the lines suggested in [2][3], we then show how the derived objective function can be used in the face of missing data. We demonstrate the resulting approach on two widely used data sets: the *Tobamovirus* data set used in [4] and [5] (where a missing data comparision was also made) and the oil flow data set used in [6]. We compare our results with other possible approaches: the crude but widely used heuristic of replacing a missing value with the mean of the corresponding component across the data set, a nearest neighbour approach and a reconstruction using linear probabilistic PCA. Both the reconstruction error and the visualisation improve dramatically through our approach.

We also consider the related problem of reconstructing missing test data: assuming we have trained a Kernel PCA feature extractor, what is the best guess for a data point with partially missing data? Our approach turns out to produce a very reasonable solution to this problem, providing again dramatic improvements in visualisation and reconstruction error.

The remainder of the paper is organised as follows: we start by briefly reviewing the probabilistic interpretation of PCA (PPCA, [5]) and its dual formulation. We then show how a kernel version of dual PPCA leads naturally to an objective function for KPCA and discuss how to use this information to deal with missing data. In the third section, we present our experimental results. In the fourth section we turn to the somewhat complementary problem of estimating missing data in test data. We finally conclude by discussing the merits and limits of our approach.

2 Cross Entropy and Reconstructing Missing Data

The key idea in PCA is to identify the directions of maximal variance in a data set. This can be shown to be equivalent to an eigenvalue problem for the empirical covariance matrix constructed from the data. Probabilistic PCA [5] assumes a linear relationship between the observed variables $\mathbf{y}_i$ and a latent variable $\mathbf{x}_i$,

$$\mathbf{y}_i = \mathbf{W}\mathbf{x}_i + \boldsymbol{\epsilon}, \tag{1}$$

where $\mathbf{W}$ is a $d \times q$ matrix (d being the dimension of the observed variable and q that of the latent variables) and $\boldsymbol{\epsilon}$ is an error term assumed to be Gaussian distributed with spherical covariance, $\boldsymbol{\epsilon} \sim N\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$. For dimensonal reduction we have $d > q$. Equation 1 then implies a Gaussian likelihood for the observed variable,

$$\mathbf{y}_i \sim N\left(\mathbf{W}\mathbf{x}_i, \sigma^2\mathbf{I}\right). \tag{2}$$

Placing a Gaussian prior on the latent variables $\mathbf{x}$ leads to the marginal likelihood

$$\mathbf{y}^{(j)} \sim N\left(\mathbf{0}, \mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I}\right). \tag{3}$$

It can be proved that the maximum of the marginal likelihood is achieved when the columns of $\mathbf{W}$ span the directions of maximal variance in the data.

This picture can be reversed leading to the dual approach to probabilistic PCA taken in [2][3]. We place a prior distribution on $\mathbf{W}$ in which each element of $\mathbf{W}$ is Gaussian distributed, $w_{ij} \sim N(0,1)$, the likelihood of equation 2 can be marginalised with respect to $\mathbf{W}$ to yield a marginal likelihood for the data set of the form

$$\mathbf{y}^{(j)} \sim N\left(\mathbf{0}, \mathbf{X}\mathbf{X}^{\mathrm{T}} + \sigma^2\mathbf{I}\right), \tag{4}$$

where $\mathbf{y}^{(j)}$ is the jth column of $\mathbf{Y}$ and each column is independent. Maximum likelihood estimation with respect to the embeddings, $\mathbf{X}$, leads to an eigenvalue problem for the inner product matrix $\mathbf{K} = \frac{1}{d}\mathbf{Y}\mathbf{Y}^{\mathrm{T}}$, which is well known to be mathematically equivalent to the eigenvalue problem for the empirical covariance matrix.

The likelihood for both PPCA and dual PPCA can be given an interesting interpretation as the *cross entropy* between two Gaussian distributions, one

Algorithm 1. The Missing Data reconstruction algorithm

```
Initialise the missing data;
Select the dimension of the latent space $q$;
repeat
   Compute the kernel matrix $\mathbf{K}$;
   Compute the approximating matrix $\mathbf{C} = \mathbf{X}\mathbf{X}^{\mathrm{T}} + \sigma^2 I$ by computing the principal
   components of $\mathbf{K}$;
   Minimise the cross entropy between $\mathbf{K}$ and $\mathbf{C}$ with respect to the missing data;
until convergence
```

specified by the empirical covariance $\mathbf{S}$ and the other by the approximating covariance $\Sigma = \mathbf{W}\mathbf{W}^T + \sigma^2 I$ in the case of PPCA and $\mathbf{C} = \mathbf{X}\mathbf{X}^T + \sigma^2 I$ in the case of dual PPCA. This is given, up to an additive constant, by the formula

$$\mathcal{L}\left(N\left(\mathbf{0},\mathbf{C}\right)||N\left(\mathbf{0},\mathbf{K}\right)\right) = -\frac{1}{2}\left(\log|\mathbf{C}| + \operatorname{trace}\left(\mathbf{K}\mathbf{C}^{-1}\right)\right). \tag{5}$$

We note in passing that, when $N > q$, $\mathbf{K}$ will not be positive definite, however this situation can be rectified without significant effect on the algorithm by adding a spherical term to $\mathbf{K}$ (see [7]).

Kernel PCA can be viewed as dual PCA on the images of the data set in a (possibly infinite dimensional) feature space. As the inner product matrix in (4) scales with the number of data points and not with their dimensionality, the computational burden will remain unchanged by pre-applying a feature map. Using the kernel trick, we have that the inner product matrix of the images of the data via the feature map is given by the kernel matrix $K\left(\mathbf{x}_i, \mathbf{x}_j\right)$, whose spectral decomposition provides the nonlinear feature extractors.

Therefore, it is natural to consider the cross entropy of equation (5) as an objective function for Kernel PCA. The implicit idea behind this is that nonlinear data in the observed space can be mapped, through the feature map, to a high dimensional space where the implied generative structure becomes approximately Gaussian[1]. While we are not aware of a general proof of this fact, there has been experimental evidence supporting it (see e.g. [8]).

Having obtained an objective function for Kernel PCA, we are in a position to give principled answers to a number of problems. In particular, this suggests a method for dealing with missing or corrupted data: the objective function can be optimised with respect to both the images and the values of the missing points (which are particular elements of $\mathbf{Y}$).

We chose to take an iterative approach to the optimisation, using spectral decomposition to compute principal components and a scaled conjugate gradient algorithm to optimise with respect to the missing points. This is summed up schematically in Algorithm 1.

[1] More precisely, the generative structure becomes approximately Gaussian after projection onto a suitable finite dimensional space.

3 Experimental Results

To test our approach we tried our algorithm on two well known Tobamovirus data set. This was used in [4] to demonstrate PCA and further used in [5] to demonstrate PPCA in the presence of missing data. It consists of 38 data points, each of them 18 dimensional. In our experiment we removed at random 130 values by sampling from a uniform distribution. To capture 95% of the initial variability we selected a latent dimension, q, of 8. We used an MLP kernel with weight variance and bias both equal to 10 [9]. Further experimental results are reported in [10].

Figure 1 (a-c) compares the reconstruction obtained with our method (b) with the underlying truth (KPCA on the full data set,(c)) and with the widely used heuristic of replacing missing components with the mean across the data set (a). The improvement in visualisation is dramatic.

To quantify the effectiveness of our algorithm, we repeated the experiment with ten different probabilities (from 0.05 to 0.5) and for ten different random seeds. To measure the quality of the reconstruction, we estimated the squared reconstruction error (given that we know the true positions of the points). We compared our results with three different methods: the widely used heuristic of the mean as above, a 1 nearest neighbour (1NN)[2] method which replaces the missing values with the values of the point with the nearest values in the known features, and missing point estimation for linear probabilistic PCA (initialised with the mean). The results for the Tobamovirus data set are summarised in Figure 1 (d), plotting the deletion probabilities on the x-axis versus the reconstruction error. The solid line is the mean initialisation, the dotted line is the reconstruction using our method, the dashed line shows the reconstruction errors using PPCA and the dotted and dashed line shows the reconstruction using 1NN (notice that 1NN is viable only up to deletion probabilities of 0.15).

4 Reconstructing Corrupted Test Data

Having introduced an objective function for Kernel PCA, the next question is the following: suppose we have trained a KPCA feature extractor on some training data set. If we are given a test point, we can use our feature extractors on it. Suppose though the test data has some missing components, can we use the knowledge of the feature extractors to deduce something about the missing data? We are assuming that the test point comes from the same (unknown) generative distribution as the training set; also, we do not want to recompute the feature extractors anew (which would reduce us to the previous problem).

We can again draw inspiration by the linear picture; a trained PPCA feature extractor gives us a generative distribution for the data

$$\mathbf{y}|W,\sigma \sim N\left(\boldsymbol{\mu}, WW^T + \sigma^2 I\right). \tag{6}$$

[2] It could be argued that a more sensible choice would be to use k-nearest neighbours. However, when the deletion probability is high, it is impossible to find sufficient uncorrupted data points to make k-NN viable.

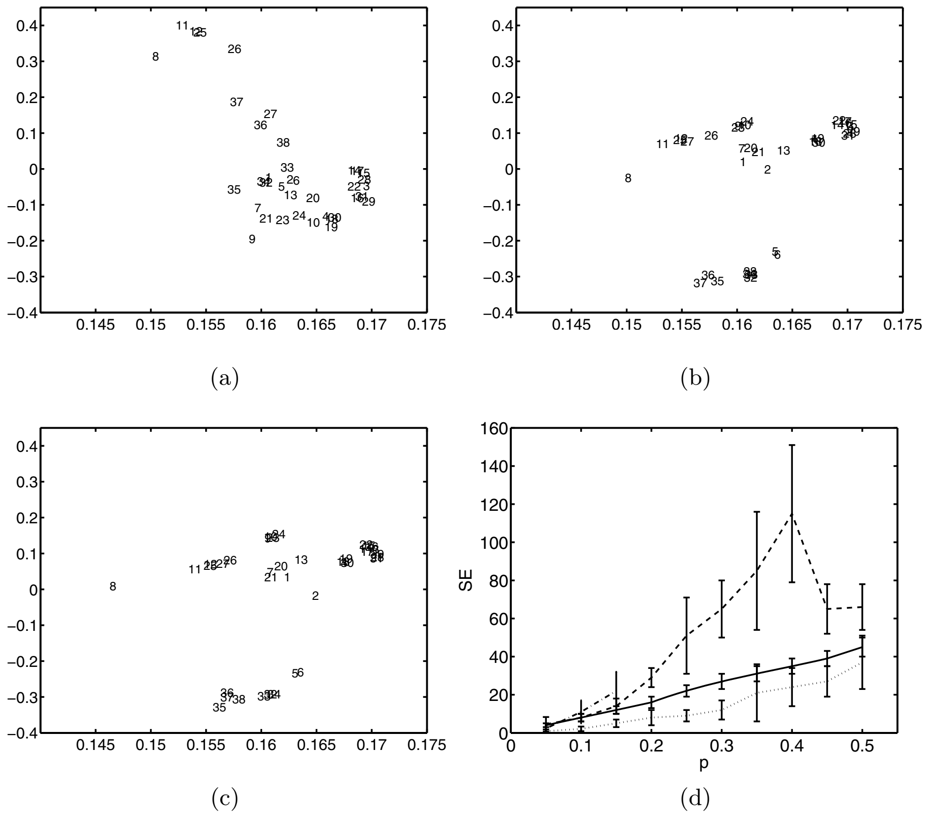


Fig. 1. KPCA with missing data. (a) shows the projection on the first two principal components of the initialisation with 20% of the values removed and initialised to the mean for the Tobamovirus data set. (b) shows the projection on the first two principal components of the optimal reconstruction of the missing data for the Tobamovirus data set. (c) shows KPCA on the Tobamovirus data set. (d) shows a comparison of the reconstruction squared errors using different methods for different deletion probabilities: the mean substitution (solid line), PPCA (dashed), 1 nearest neighbour (dotted and dashed) and our approach (dotted).

If we are given some of the entries in the test point $\mathbf{y}_t$, call them $\mathbf{y}_{t\mathrm{Known}}$, the obvious best guess for the unknown entries would be given by the maximum of the conditional probability $p\left(\mathbf{y}_{t\mathrm{notKnown}}|\mathbf{y}_{t\mathrm{Known}}\right)$ (notice that this will also provide an estimate of the uncertainty on the guess).

Although it is in general impossible to estimate the conditional distribution (6) for Kernel PCA, we can still obtain a kernel version of the optimisation problem by looking back at PPCA from a geometric perspective. The maximum of the conditional probability is given by the minimum of the Mahalanobis distance of $\mathbf{y}_t$ from the mean $\boldsymbol{\mu}$, the Mahalanobis distance being measured with the inverse covariance matrix

$$C^{-1} = \left(WW^T + \sigma^2 I\right)^{-1} .$$

Therefore we can recover the maximum by optimising the quantity

$$\mathbf{y}_t^T C^{-1}\mathbf{y}_t = \sum_{i=1}^{q}\left(\lambda_i^{-1} - \sigma^{-2}\right)\left(\mathbf{y}_t \cdot \mathbf{u}_i\right)^2 + \sigma^{-2}\|\mathbf{y}_t\|^2 \tag{7}$$

where q is the number of principal components included in the model, $\mathbf{u}_i$ are the principal eigenvectors and λ_i are the corresponding eigenvalues.

As equation (7) makes clear, this distance can be expressed uniquely in terms of dot products of the test point with the principal components (and with itself), hence it is readily transferred to the kernel situation. In the RBF case, there is the further advantage that $k(\mathbf{y}, \mathbf{y}) = 1\ \forall \mathbf{y}$ so that the second term in (7) needs not be included.

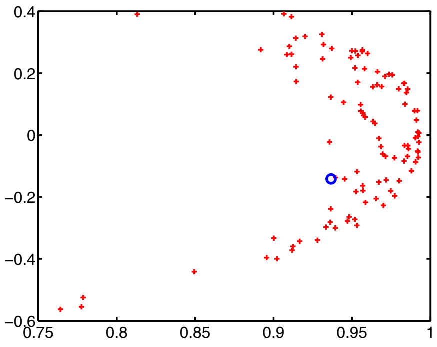
Recalling that the KPCA feature extractors in feature space are given by $\mathbf{u}_i = \sum_{j=1}^{N\text{train}} \alpha_j^i \boldsymbol{\Phi}(\mathbf{y}_j)$ where $\boldsymbol{\alpha}^i$ is the i-th eigenvector of the Gram matrix $k(\mathbf{y}_i, \mathbf{y}_j)$ (normalised so that $\lambda_i \left(\boldsymbol{\alpha}^i \cdot \boldsymbol{\alpha}^i\right) = 1$), we obtain the following objective function for a missing test point

$$\mathcal{L} = \sum_{i=1}^{q}\left(\lambda_i^{-1} - \sigma^{-2}\right)\left(\sum_{j=1}^{N\text{train}} \alpha_j^i k(\mathbf{y}_j, \mathbf{y}_t)\right)^2 . \tag{8}$$

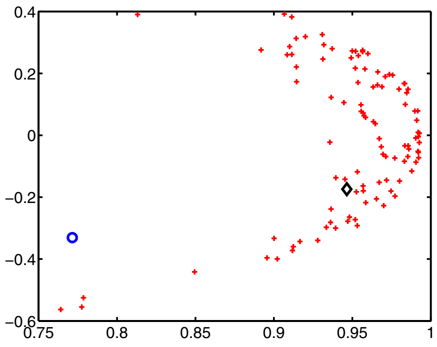
Notice that we need both the KPCA feature extractors and the off subspace variance σ^2 to formulate our optimisation problem, which can be obtained using our approach to Kernel PCA but not using the standard non-probabilistic formulation.

To test this approach we used the oil flow data set of [6]. This consists of 1000 12 dimensional synthetically generated data points modelling the flow of a mixture of oil, water and gas in a pipeline. The points are labelled in three different classes, according to the flow being laminar, annular or homogeneous. In this case we used an RBF kernel with inverse width 0.075. The results are shown in Figure 2. We selected the points corresponding to a laminar flow in the oil flow data set. We removed a point at random and performed KPCA on the remaining data set, retaining two principal components. We then treated the point we removed as a test point and artificially corrupted its first five coordinates by multiplying them by a constant factor. The point recovered through optimising the objective function (8) is very close indeed.

To quantify the efficacy of our method, we repeated the example of Figure 2 removing a different point at random fifty times and replacing its first five coordinates with random numbers. We also increased the number of features extracted from two to ten. The results are summarised in Table 1, where a comparison with the mean substitution and 1 nearest neighbour is made. Notice that the reconstruction error tends to decrease as the number of extracted features is increased, as well as the reconstruction becoming more consistent (smaller fluctuations in the mean error).



(a)



(b)

Fig. 2. Reconstructing test points with Kernel PCA:(a) training points (crosses) and original position of the test point (circle); (b) corrupted position of the test point (circle) and reconstructed position of the test point (diamond)

Table 1. Reconstructing corrupted test points using KPCA feature extractors. The first column shows the number of principal components retained, the second to fourth columns show the mean reconstruction error across 50 runs using our method, mean substitution and 1 nearest neighbour respectively. Notice that the reconstruction error using our method decreases as the number of principal components is increased; with more than three retained components our method gives the best performance.

Features extracted	KPCA	Mean	1NN
2	0.55±0.28	0.76±0.33	**0.29±0.20**
3	0.38±0.22	0.76±0.33	**0.29±0.20**
4	**0.28±0.17**	0.76±0.33	0.29±0.20
5	**0.24±0.16**	0.76±0.33	0.29±0.20

5 Discussion

In this paper we introduced an objective function for Kernel PCA, building on previous work on probabilistic PCA [5] and latent variable models in Gaussian Processes [2] [3]. This in turns allows to extend important inference techniques, such as the estimation of missing data, to the case where the features are nonlinear.

Experimental results on two benchmark data sets show that this approach yields far better results than the often recommended heuristic of replacing a missing value with the mean (which we used as our initialisation), and consistently outperforms other methods such as 1 NN and probabilistic PCA. Furthermore, the same ideas lead to a very natural solution of the related problem of estimated missing or corrupted components in test data.

Despite these positive results, our approach still falls short of providing a full probabilistic interpretation for Kernel PCA. The Gordian knot of the feature map has been severed by integrating out the nonlinear mapping. This comes at the cost of no longer being able to predict the positions of new observed points from the latent ones. The link between the primal and the dual PCA problems in the kernelised case requires the explicit knowledge of the feature map. Similarly, the elegant interpretation in terms of probability distributions is harder to recover.

Acknowledgements. G.S. gratefully acknowledges support from a BBSRC award "Improved processing of microarray data using probabilistic models".

References

1. Schölkopf, B., Smola, A.J., Müller, K.R.: Kernel principal component analysis. In: Proceedings 1997 International Conference on Artificial Neural Networks, ICANN'97, Lausanne, Switzerland (1997) 583
2. Lawrence, N.D.: Gaussian process models for visualisation of high dimensional data. In Thrun, S., Saul, L., Schölkopf, B., eds.: Advances in Neural Information Processing Systems. Volume 16., Cambridge, MA, MIT Press (2004) 329–336
3. Lawrence, N.D.: Probabilistic non-linear principal component analysis with Gaussian process latent variable models. Journal of Machine Learning Research **6** (2005) 1783–1816
4. Ripley, B.D.: Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge, U.K. (1996)
5. Tipping, M.E., Bishop, C.M.: Probabilistic principal component analysis. Journal of the Royal Statistical Society, B **6** (1999) 611–622
6. Bishop, C.M., Svensén, M., Williams, C.K.I.: GTM: the Generative Topographic Mapping. Neural Computation **10** (1998) 215–234
7. Lawrence, N.D., Sanguinetti, G.: Matching kernels through Kullback-Leibler divergence minimisation. Technical Report CS-04-12, The University of Sheffield, Department of Computer Science (2004)
8. Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press (2001)
9. Williams, C.K.I.: Computing with infinite networks. In Mozer, M.C., Jordan, M.I., Petsche, T., eds.: Advances in Neural Information Processing Systems. Volume 9., Cambridge, MA, MIT Press (1997)
10. Sanguinetti, G., Lawrence, N.D.: Missing data in kernel PCA. Technical Report CS-06-08, The University of Sheffield, Department of Computer Science (2006)

Exploiting Extremely Rare Features in Text Categorization[⋆]

Péter Schönhofen and András A. Benczúr

Informatics Laboratory
Computer and Automation Research Institute
Hungarian Academy of Sciences
Lagymanyosi u 11, H-1111 Budapest
schonhofen@gmail.com, benczur@sztaki.hu

Abstract. One of the first steps of document classification, clustering and many other information retrieval tasks is to discard words occurring only a few times in the corpus, based on the assumption that they have little contribution to the bag of words representation. However, as we will show, rare n-grams and other similar features are able to indicate surprisingly well if two documents belong to the same category, and thus can aid classification. In our experiments over four corpora, we found that while keeping the size of the training set constant, 5-25% of the test set can be classified essentially for free based on rare features without any loss of accuracy, even experiencing an improvement of 0.6-1.6%.

1 Introduction

Document categorization and clustering is a well studied area, several papers survey the available methods and their performance [21, 20, 4, 3]. In most results both frequent and rare words are discarded as part of pre-processing. The only measurement which takes rarity into account is the inverse document frequency in the tf-idf weighting scheme. In their classical paper Yang and Pedersen [21] disprove the widely held belief that common terms are non-informative for text categorization. In this paper we observe the same about *rare terms*; more precisely we show how rare words and n-grams ($n \leq 6$) [2] can be exploited to improve quality of classification. A feature instance w has **frequency** f if it is present in exactly f documents; the feature is **rare** if $f \leq 10$. For experiments with more features including skipping n-grams and contextual bigrams see the full version of the paper.

Our results indicate that topical similarity between two documents sharing the same extremely rare n-gram can be much stronger than those between their bag of words vectors [18] exploited by traditional classifiers. A possible explanation of this phenomenon can be given based on the assumption that a rare feature usually has some bias towards a certain topic and is not spread completely

⋆ Support from the NKFP 2005 project MOLINGV. Full version available at www.ilab.sztaki.hu/websearch-data/Publications/rarewords-techrep.pdf

uniformly across the documents. If the probability that the feature is present in a document is only by a small margin above the threshold required for the appearance in the corpus, then it is likely to appear in some of the documents about the characteristic topic of the feature but not but not in others.

In order to exploit rare words and n-grams in categorization we have to resolve the computational burden of handling a very large number of features. As the majority of features are rare and a single one of them is known to have little effect on the output, we may neither give them all nor a selected part of them as input to classifiers or clustering algorithms. Instead we preprocess the corpus by forcing documents with a sufficient number of rare features in common into the same category. This can be realized in several ways: we may pre-classify documents that share rare features with training set documents as the simplest use. We may either merge the content of documents that share rare features to mutually enrich their vocabulary or represent one with the text of the other. In Section 2 we give various methods for prioritizing among different features in common with different topics and documents, filtering out less reliable pairs, and resolving conflicts when features exist in common with more than one category.

Although the usefulness of rare features for classification has been already pointed out by Price and Thelwall [12], the approach proposed in this paper is essentially different from theirs. We emphasize that our method does not carry out feature extraction in the conventional sense, since we do not use rare features as document representatives. Instead, after exploiting them to discover rare feature instances, they are removed from documents before passing them to the classification algorithm.

To prove that rare words and features indeed reflect general topical similarity between documents and their usability does not depend on any peculiar characteristics of corpora, we tested our method on four text collections, namely Reuters-21578, Reuters Corpus Volume 1 (RCV1), Ken Lang's 20 Newsgroups, and the abstracts of patents contained in the World International Property Organization's (WIPO) corpus. Results are discussed in Section 3. The effect of our method on clustering accuracy is shown in the full paper.

1.1 Related Results

The idea to consider high and low frequency words separately originates in Luhn's [7] intuition (see also [18]) that middle-ranking words are the most indicative of the content. For example, [15] shows that words with the highest average discriminatory power tend to occur in 1%-90% of documents. Therefore infrequent words, usually thought to be typos or obscure phrases [18, 19], are often ignored in IR systems. When measuring the effect of removing rare and frequent words prior to clustering, Rigouste et al. [13] found that while the former hurts, rare words can be safely discarded. Similarly Yang and Pedersen [21] acknowledge that rare words have no significant influence on classification.

Several authors only partly accept that rare words are completely useless for classification. Price and Thelwall [12] show that words of frequency even as low as 2 are useful for academic domain clustering, suggesting that they

Algorithm 1. Connecting documents via rare features.

```
1: Discard words with frequency above threshold *cutoff*; select rare features with
   frequency $f \leq rarity$ based on remaining words
2: Weight each document pair by the number of common selected features.
3: Form the graph over documents with edges for pairs with weight at least $w_{\min}$.
4: for all pairs $d$, $d'$ in order of decreasing weight do
5:    if $d$ and $d'$ belongs to no pair then
6:       form component $(d, d')$
7: for all components do
8:    represent the component by either a random document of the component or the
      union of all text in the component
```

Table 1. Parameters of Algorithm 1

stemming	on or off
rarity	frequency f threshold to consider a feature rare
cutoff	maximum frequency of a word allowed to appear in a rare feature
$w_{\min}$	minimum no. rare features needed in common to connect two documents
$\text{dist}_{\max}$	maximum distance of indirection to form composite documents
merging	choice of passing merged text or a sample document to the classifier

contain subject-related information. However they describe no efficient method to train a classifier based on rare terms; for future work they envision an artificial intelligence or natural language processing approach which would discard useless ones [1]. Similarly [21] mentions that discarding rare words too aggressively can be counterproductive but gives no solution to the computational issues.

A problem of rare words is that due to their large number they cannot be fed to computationally hard methods that would be able to separate useful words from useless ones; algorithms such as vocabulary spectral analysis [17] are infeasible over rare words. In addition, rare words often cause noise that confuse term weighing and feature selection such as χ^2 [14] or mutual information [21,11]. The only exception is the inverse document frequency or idf [16] that is commonly used in summarization, feature extraction and dimensionality reduction.

2 Algorithm for Connecting Documents Via Rare Features

Next we describe our algorithm that, prior to classification, preprocesses documents by connecting them via rare features in common. The algorithm can be tuned by several parameters to maximize the gain in classification quality. We may exclude very frequent words from rare n-grams (*cutoff*), limit the maximum feature frequency (*rarity*) to consider a feature rare; require several features in common to connect two documents ($w_{\min}$) and limit the distance ($\text{dist}_{\max}$) between connected documents. The parameters are described in detail next and summarized in Table 1.

Algorithm 2. Kruskal's algorithm in the special case of connecting test set documents to train set ones (supervised case).

```
for distance = 1, ..., $\mathrm{dist}_{\max}$ do
    for all remaining test set documents $d$ do
        if $d$ has edge to at least one $d'$ in the train set then
            merge $d$ with all of its edges into $d'$ where the weight of $(d, d')$ maximum
for all documents $d'$ in the train set do
    for all $d$ merged with $d'$ do
        Classify $d$ into the category of $d'$
        Expand $d'$ with the text of $d$ /*optional*/
        Remove $d$ from test set
```

The main idea of the algorithm is to greedily pair documents in the order of decreasing number of rare features in common. The algorithm hence uses the simplest form of single linkage clustering [10, and many others]; we tested a few more complicated versions but achieved no improvements. The general method is described in Algorithm 1 while a slightly stronger version of steps 4–8 is given in detail for the simpler special case of connecting test set documents to train set ones in Algorithm 2. Later in Fig. 1-c we will justify the choice of this simple clustering algorithm by showing that pairs connected by rare features are in isolation and larger components appear only sporadically.

The main computational effort in our Algorithm 1 is devoted to identifying rare features (line 1), a very large fraction of all features. It is easy to implement the selection by external memory sorting; in our experiments we choose the simpler and faster internal memory solution that poses limits on the corpus size.

Given the collection of rare features together with the documents containing them, we build an undirected graph over documents as nodes (lines 2–3). We iterate over the features and add a new edge candidate whenever we discover a pair of documents sharing a rare feature. We weight edges by the number of rare features in common and discard edge candidates below weight $w_{\min}$.

We connect documents by iterating through edges in the order of decreasing weight. If the next highest weight edge (d, d') is such that neither d nor d' belongs to a component formed earlier in line 6, then we add this pair as a new component. More complex algorithms may form components of chains of length up to some $\mathrm{dist}_{\max}$, replace the greedy choice of the loop in line 4 for example by a maximum weight matching algorithm, or form larger components by iteratively merging them into new ones.

Finally we pass the corpus to the classifier; here for the components we have the choice to pass the merged text or just a randomly selected document to the classifier. Optionally we may enrich the content of each document d of the train set with the text of all or some documents d' merged with d during the algorithm. If we train the classifier with the extended documents, we observe that their richer content characterize categories better.

While our preprocessing algorithm is also suited for unsupervised clustering, if exists, we may prefer train–test document connections. Documents connected

Table 2. Characteristics of the four corpora used in our experiments

Corpus	Reuters-21578	RCV1	20 Newsgroups	WIPO abstr.
No. of docs	10,944	199,835	18,828	75,250
No. of categories	36	91	20	114
Avg doc length	69 words	122 words	125 words	62 words

to the train set can then be classified "for free". This special case is described in Algorithm 2 where we iterate through all test set documents d; whenever d is connected to another in the train set, we merge d with d' such that they are connected by the largest number of features in common.

We may also consider indirect connections to the train set. If document d is connected to another in the test set that is in turn connected to d' of the train set, we may also pre-classify d into the category of d'. We set a distance threshold $\text{dist}_{\max}$; this turns into $\text{dist}_{\max}$ iterations of the first **for** loop of Algorithm 2.

3 Experiments

We tested our algorithm on four corpora of different domain and nature. Table 3 shows their most important properties. For all corpora, we removed stop words and performed stemming with the Morph component of WordNet [9].

Reuters-21578 [5] and Reuters Corpus Volume 1 (RCV1) [6] contains news about politics, economics and trade. In RCV1 we characterized documents solely by their topic codes, industry and country classifications were ignored. If a document was assigned to multiple topics, it was re-assigned to the smallest, provided that it covered at least 50 documents; documents without topic indication were discarded. Due to performance limitations, only the first 200,000 documents were considered from RCV1. However, to demonstrate our method's scalability, in the full version of the paper we also processed RCV1 in a partitioned way.

Ken Lang's 20 Newsgroups, or more precisely its slightly modified and cleaned variant made available by Jason Rennie (`http://people.csail.mit.edu/jrennie/20Newsgroups/20news-18828.tar.gz`), consists of short Usenet postings evenly distributed among 20 domains. The World Internet Property Organization (WIPO) corpus is a collection of patents organized in a strict multilevel classification system. Because the full text of documents is unnecessarily long for our purposes, we only utilize abstracts. Furthermore due to the very large number of categories, we only keep the top two levels of the hierarchy.

First let us explore the quality of rare words and n-grams for $2 \leq n \leq 6$ with frequency $2 \leq f \leq 10$. Let $q(w)$ be the probability that two random documents containing w belong to the same category (that may be apriori assumed in an unsupervised experiment). For a fixed feature type, **feature quality** is the average of $q(w)$ over all feature instances w of the given type with frequency $f \geq 2$. Quality is inversely proportional to the number of pairs formed by the preprocessing algorithm; the expected performance of our preprocessing algorithm based on these two values is explored in the full version of the paper.

As Fig. 1-d shows, quality quickly decays with increasing frequency f for short features (words, bigrams), while for 5- and 6-grams, quality remains high even at frequencies close to 10. We also observed a slight increase in quality when lowering the *cutoff* limit to exclude frequent words from rare features (Fig. 1-a); however, this affect classification and clustering accuracy only marginally, since coverage becomes very low.

Fig. 1-b shows that out of the document pairs generated by our algorithm, what percentage of pairs connected by selected features have Jaccard similarity less than or equal to the threshold specified over the horizontal axis. The figure supports our claim that documents share rare features due to a general topical similarity and not just because of some side effect of (near) replication or quoting from other documents. If rare features all arose from duplicates, the curve would proceed close to 0, jumping to 1 only when the similarity threshold is increased to its maximum. If, on the other hand, rare features appeared in very dissimilar documents in common, then the curve would jump already at a fairly low similarity level. In fact, in 20 Newsgroups our method pairs the least similar documents, while in RCV1 we observe just the contrary, pairings are between the most similar ones.

As seen in Fig. 1-c, the largest fraction of selected rare features connect pairs of documents that remain in isolation. arising in line 3 of Algorithm 1, the number of connected components of various sizes decays exponentially and components of more than four documents occur only sporadically.

For classification we used the naive Bayes component of the Bow toolkit [8] with default parameters, as this enhanced naive Bayes implementation often outperformed the SVM classifier. Improvements achieved by our method are shown on Fig 1-e–f and h–i for the four corpora. In the figures, each measurement point represents the average accuracy of five random choices of training documents. The values for the free parameters of Table 1 are *rarity* = 2, *cutoff* = 1000, $w_{\min} = 3$ for all except WIPO where $w_{\min} = 2$, and merging set on. In Fig. 1-g we see that our algorithm significantly reduces the number of documents passed to the classifier.

In Fig. 1-e–f and h–i we see that except from 20 Newsgroups, the usefulness of the various feature types relative to each other reflect the ranking as expected by their feature quality: words are the least efficient, with n-grams providing better results; bigrams and 3-grams however perform unexpectedly well.

Note that improvements for 20 Newsgroups roughly stabilize beyond 10% training set ratio, possibly because for larger training sets the accuracy of the classification algorithm approaches the quality of features, diminishing their power.

4 Conclusion and Future Work

This paper presented a novel approach in which extremely rare n-grams, mostly neglected by previous research, are exploited to aid classification and clustering. In our experiments on four different corpora we found that even simple features

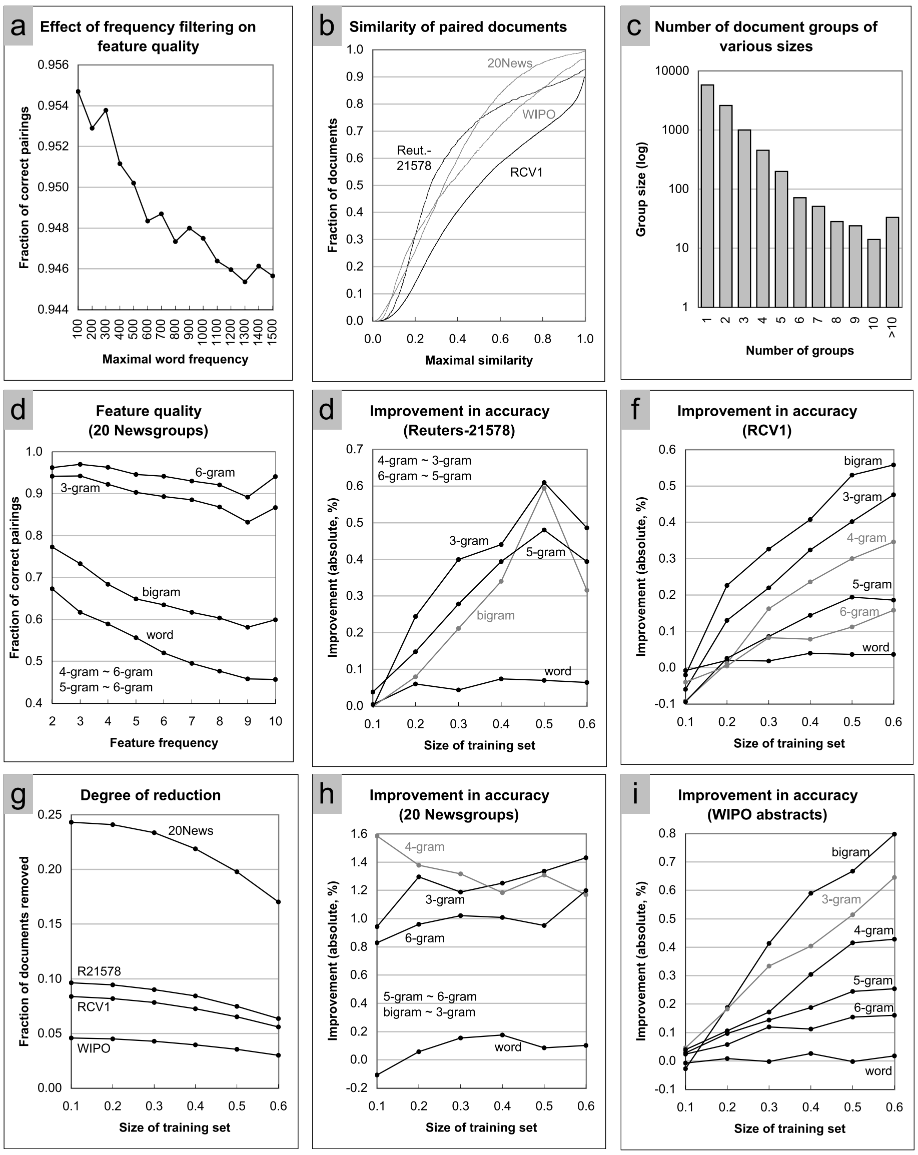


Fig. 1. a: Feature quality of selected features in 20 Newsgroups. **b:** Histogram of the Jaccard similarity of document pairs connected by the co-occurrence of a frequency 2 trigram. **c:** The number of components of various sizes connected by the frequency 2 trigrams of 20 Newsgroups. **d:** Feature quality as the function of *cutoff* in frequency 2 trigrams over 20 Newsgroups.**e**–**f** and **h**–**i:** Improvement over the baseline classification accuracy. For the sake of clarity we merged very close lines into one. **g:** The fraction of documents paired by our algorithm. Features used are 4-grams for 20 Newsgroups, trigrams for Reuters-21578 and bigrams for RCV1 and WIPO.

like rare bigrams and 3-grams are able to improve accuracy. Future directions may include new features and more sophisticated merging algorithms.

References

1. D. C. Comeau and W. J. Wilbur. Non-word identification or spell checking without a dictionary. *J. Am. Soc. Inf. Sci. Technol.*, 55(2):169–177, 2004.
2. J. Goodman. A bit of progress in language modeling. *CoRR*, cs.CL/0108005, 2001.
3. M. Iwayama and T. Tokunaga. Cluster-based text categorization: a comparison of category search strategies. In *SIGIR '95*, pages 273–280, 1995.
4. T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proc. European Conference on Machine Learning*, pages 137–142, 1998.
5. D. D. Lewis. Reuters-21578 text categorization test collection, distribution 1.0, available at `http://www.daviddlewis.com/resources/ testcollections/ reuters21578` , 1997.
6. D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, 2004.
7. H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM J. Research and Development*, 1(4):309–317, 1957.
8. A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/˜mccallum/bow, 1996.
9. G. A. Miller. Wordnet: A lexical database for English. *Commun. ACM*, 38(11):39–41, 1995.
10. P. Pantel and D. Lin. Discovering word senses from text. In *Porc. SigKDD*, 2002.
11. V. Pekar and M. Krkoska. Weighting distributional features for automatic semantic classification of words. In *International Conference on Recent Advances In Natural Language Processing*, pages 369–373, 2003.
12. L. Price and M. Thelwall. The clustering power of low frequency words in academic webs: Brief communication. *J. Am. Soc. Inf. Sci. Technol.*, 56(8):883–888, 2005.
13. L. Rigouste, O. Cappe, and F. Yvon. Evaluation of a probabilistic method for unsupervised text clustering. In *International Symposium on Applied Stochastic Models and Data Analysis (ASMDA)*, 2005.
14. M. Rogati and Y. Yang. High-performing feature selection for text classification. In *Proc. International Conference on Information and Knowledge Management*, pages 659–661, 2002.
15. G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. Technical report, Ithaca, NY, USA, 1974.
16. K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.
17. M. Thelwall. Vocabulary spectral analysis as an exploratory tool for scientific web intelligence. In *Proc. Information Visualisation*, pages 501–506, 2004.
18. C. J. Van Rijsbergen. *Information Retrieval, 2nd edition.* Dept. of Computer Science, University of Glasgow, 1979.
19. M. Weeber, R. Vos, and R. H. Baayen. Extracting the lowest frequency words: Pitfalls and possibilities. *Computational Linguistics*, 26(3):301–317, 2000.
20. P. Willett. Recent trends in hierarchic document clustering: A critical review. *Inf. Process. Manage.*, 24(5):577–597, 1988.
21. Y. Yang and J. Pedersen. A comparative study on feature selection in text categorization. In *Proc. ICML-97*, pages 412–420, 1997.

Efficient Large Scale Linear Programming Support Vector Machines

Suvrit Sra

Dept. of Comp. Sciences, The University of Texas at Austin
Austin, TX 78712, USA

Abstract. This paper presents a decomposition method for efficiently constructing ℓ_1-norm Support Vector Machines (SVMs). The decomposition algorithm introduced in this paper possesses many desirable properties. For example, it is provably convergent, scales well to large datasets, is easy to implement, and can be extended to handle support vector regression and other SVM variants. We demonstrate the efficiency of our algorithm by training on (dense) synthetic datasets of sizes up to 20 million points (in $\mathbb{R}^{32}$). The results show our algorithm to be several orders of magnitude faster than a previously published method for the same task. We also present experimental results on real data sets—our method is seen to be not only very fast, but also highly competitive against the leading SVM implementations.

1 Introduction

Traditionally Support Vector Machines (SVMs) are constructed by maximizing an ℓ_2-norm margin, which is achieved by solving an associated quadratic program. Researchers have also looked at maximizing margins measured using other ℓ_p norms [1]—most notably the ℓ_1 and ℓ_∞ norms, both of which lead to linear programming formulations[1]. The book chapter by Bennett [1] lists some further useful references related to such formulations. Some recent relevant papers studying the ℓ_1-norm SVM are [2, 13, 19].

Most work on SVMs, however, ends up focusing on the details of the ℓ_2-norm SVM, relegating the solution of the ℓ_1 (or ℓ_∞) norm SVM to an off-the-shelf linear programming (LP) solver such as CPLEX$^{\text{TM}}$. For real world data, especially for large-scale data, such an approach can be very expensive, if not impractical. The ℓ_2-SVM has on the other hand witnessed a lot of research and efficient implementations for solving it are available (e.g., SVM$^{\text{light}}$ [10], SMO [16], LIBSVM [4]). It is desirable that some of the algorithmic progress made for the ℓ_2-SVM be carried over to the ℓ_1-SVM too.

We are aware of one previous work, namely that of Bradley and Mangasarian [2] that attempts to make learning ℓ_1-SVMs practical for large data sets. These authors introduce a method called Linear Programming Chunking (LPC) that

[1] In general, maximizing margin using an ℓ_q norm can be done by minimizing $\|\boldsymbol{w}\|_p$, where $\frac{1}{p} + \frac{1}{q} = 1$.

decomposes a linear program into smaller chunks and solves them using any LP solver. However, despite its efficiencies, LPC can be prohibitively slow for large problems. Thus, an efficient method for solving LP based SVM formulations is needed, and this paper presents such a method. Our approach yields a scalable and efficient decomposition method for solving the ℓ_1-SVM, which is several orders of magnitude faster than the LPC approach and makes learning large scale ℓ_1-SVMs practical. Furthermore, our method is simple to implement, provably convergent, easily extensible to solve other SVM variants, and yields accuracies competitive with well established SVM software.

Much of the speed of our algorithm lies in the fact that we solve the primal as opposed to the dual formulation of the ℓ_1-SVM.[2] Recently Chapelle [5] has provided motivation for reconsidering the solution of the primal formulation of the SVM. Since many years, owing to its simplicity and extensibility to nonlinear cases, researchers have focused on solving the dual problem for SVMs. For example, when number of training points greatly exceeds the dimensionality of a single point, it is advantageous to solve the primal rather than the dual (despite the novel efficiencies introduced in [14]).

2 The ℓ_1-Norm SVM

As per the standard two-class classification problem, we assume the input to be the set $\{(\boldsymbol{x}_i, y_i) : 1 \le i \le N\}$, where $\boldsymbol{x}_i \in \mathbb{R}^M$ are the training points, and $y_i \in \{\pm 1\}$ are their associated class labels. The aim is to learn a function or classifier $f(\boldsymbol{x})$ such that given a new data point $\boldsymbol{x}$, we can accurately predict its class label. A linear classifier is commonly constructed by computing a function $f(\boldsymbol{x}) = \operatorname{sgn}(\boldsymbol{w}^T\boldsymbol{x} + b)$. The corresponding ℓ_1-SVM problem may be written as

$$\begin{aligned} &\min_{\boldsymbol{w},b} \ \|\boldsymbol{w}\|_1 + C\sum_i \xi_i, \\ \text{subject to} \quad &y_i(\langle \boldsymbol{x}_i, \boldsymbol{w}\rangle + b) \ge 1 - \xi_i, \qquad \xi_i \ge 0, \quad 1 \le i \le N. \end{aligned} \tag{2.1}$$

The parameter C is a cost (penalty) parameter and is provided as input (normally after having been determined using cross-validation). Observe that in (2.1) we seek to minimize $\|\boldsymbol{w}\|_1$ instead of $\|\boldsymbol{w}\|_2^2$, as is done in the traditional ℓ_2-SVM. Minimizing $\|\boldsymbol{w}\|_1$ leads to sparser solutions, which in turn imply better dimension reduction, greater robustness, and faster classifiers [1, 2, 19].

We introduce an auxiliary variable $\boldsymbol{f} = |\boldsymbol{w}|$ (elementwise) to write (2.1) as the linear program (LP)

$$\begin{aligned} &\min_{\boldsymbol{w},b,\boldsymbol{f},\boldsymbol{\xi}} \ \mathbf{1}^T(\boldsymbol{f} + C\boldsymbol{\xi}) \\ &y_i(\langle \boldsymbol{x}_i, \boldsymbol{w}\rangle + b) \ge 1 - \xi_i, \quad 1 \le i \le N, \\ &-\boldsymbol{w} - \boldsymbol{f} \le \mathbf{0}, \quad \boldsymbol{w} - \boldsymbol{f} \le \mathbf{0}, \quad \boldsymbol{\xi} \ge \mathbf{0}. \end{aligned} \tag{2.2}$$

[2] Our approach has a more *primal-dual* flavor, but since we never form the dual, we continue referring to it as a *primal* approach.

For large-scale data solving the LP (2.2) using off-the-shelf software can be very expensive. Bradley and Mangasarian [2] described a method called Linear Programming Chunking for efficiently solving (2.2). However, despite their "chunking" approach, their method can still be extremely slow for large data sets. In Section 3 below, we describe a fast decomposition procedure for solving (2.2). We remark that our techniques can be easily adapted to solve the ℓ_∞-norm SVM. Furthermore, nonlinear SVMS via the LP-machines [8] can also be handled with equal ease. We omit the details due to space limitations (these will be published elsewhere).

3 Algorithm

It may not seem obvious how to solve (2.2) using a simple decomposition method. Problem (2.2) lacks strict convexity, a necessary ingredient for the application of many decomposition techniques. Fortunately, we can exploit a very useful result of Mangasarian [12, Theorem 2.1-a-i] (adapted as Theorem 1 below) that permits us to transform (2.2) into an equivalent quadratic program that has the necessary strict convexity.

Theorem 1 (ℓ_1 SVM). *Let $\boldsymbol{g} = [\boldsymbol{w}; b; \boldsymbol{f}; \boldsymbol{\xi}]$ and $\boldsymbol{c} = [\boldsymbol{0}; 0; \boldsymbol{1}; C\boldsymbol{1}]$ be partitioned conformally. If* (2.2) *has a solution, then there exists an $\epsilon_0 > 0$, such that for all $\epsilon \le \epsilon_0$,*

$$\underset{\boldsymbol{g}\in G}{\operatorname{argmin}} \ \|\boldsymbol{g} + \epsilon^{-1}\boldsymbol{c}\|_2^2 \quad = \quad \underset{\boldsymbol{g}\in G^\star}{\operatorname{argmin}} \ \|\boldsymbol{g}\|_2^2, \tag{3.1}$$

where G is the feasible set for (2.2) *and $G^\star$ is the set of optimal solutions to* (2.2)*. The minimizer of* (3.1) *is unique.*

Theorem 1 essentially states that the solution of (3.1) yields the minimum ℓ_2-norm solution out of all the possible solutions of (2.2). This seemingly counter-intuitive replacement of a linear program by a corresponding quadratic program lies at the heart of building a decomposition method for the ℓ_1-SVM.

3.1 Decomposition

To permit a clearer description we rewrite (3.1) in the more explicit form

$$\min_{\boldsymbol{g}=[\boldsymbol{w};b;\boldsymbol{f},\boldsymbol{\xi}]} \ \tfrac{1}{2}\|\boldsymbol{g} - (-\tfrac{1}{\epsilon})\boldsymbol{c}\|^2, \tag{3.2}$$

$$\begin{aligned} \text{subject to} \qquad -y_i\boldsymbol{x}_i^T\boldsymbol{w} - y_i b + \boldsymbol{0}^T\boldsymbol{f} - \xi_i &\le -1 && \text{(TR)}\\ -w_i + 0b - f_i + 0\xi_i &\le \ \ 0 && \text{(A1)}\\ w_i + 0b + f_i + 0\xi_i &\le \ \ 0 && \text{(A2)}\\ 0w_i + 0b + 0f_i - \xi_i &\le \ \ 0, && \text{(XI)} \end{aligned}$$

where $\boldsymbol{g}$ and $\boldsymbol{c}$ are as in Theorem 1, and $1 \le i \le N$. Let $\boldsymbol{z}$ denote the vector of dual variables associated with the training (TR), absolute value (A1), (A2), and soft-margin (XI) constraints. Further, let $\boldsymbol{A}$ denote the matrix of all these constraints put together.

Let $L(\boldsymbol{g}, \boldsymbol{z})$ denote that Lagrangian for (3.2). A first order necessary condition of optimality is

$$\frac{\partial}{\partial \boldsymbol{g}} L(\boldsymbol{g}, \boldsymbol{z}) = \boldsymbol{g} + \epsilon^{-1}\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{z} = 0, \qquad \boldsymbol{z} \geq \boldsymbol{0}. \tag{3.3}$$

The decomposition procedure that we use consists of the following main steps:

1. Start with a dual feasible solution, and obtain a corresponding primal solution so that the first-order necessary conditions (3.3) are satisfied. For example, $\boldsymbol{z} = \boldsymbol{0}$ and $\boldsymbol{g} = -\epsilon^{-1}\boldsymbol{c}$ is a valid initialization.
2. Go through each constraint individually and enforce it. Enforcing each constraint is equivalent to updating the corresponding dual variable z_j $(1 \leq j \leq 4N)$ so that $z_j \geq 0$ is maintained, while recomputing $\boldsymbol{g}$ to ensure that (3.3) remains satisfied.
3. Repeat Step 2 until some convergence condition is satisfied (such as small net violation of all the KKT constraints, change below a certain threshold to the objective function etc.).

This decomposition procedure is based upon Bregman's method [3][3], which is a generic decomposition method for minimizing a strictly convex function subject to linear inequality constraints. This procedure generates a sequence of primal ($\{\boldsymbol{g}^t\}_{t\geq 0}$) and dual ($\{\boldsymbol{z}^t\}_{t\geq 0}$) iterates that converge to the optimal solution of the associated problem (see [3, Chapter 6] for a proof). Pseudo-code and associated implementation details for this algorithm are omitted from this paper due to space limitations.

So far we have not remarked upon the selection of the parameter ϵ. Two approaches are possible. One can test the accuracy on a hold-out subset of the training data and perform a search for a good value of ϵ. One can also pick an ϵ from within a predefined range of values. The former approach might increase the running time (albeit minimally), whereas the latter is simple and fast. We tried both approaches, and found that usually a value of ϵ in [0.1–100] worked well (i.e., resulted in high training and test accuracy, as well as rapid convergence).

4 Experimental Results

In this section we describe some of the experiments that we performed to assess the quality of our implementation. We consider two types of experiments. The first type is on real data (Section 4.1), while the second is on synthetic data (Section 4.2). The purpose of the former is to illustrate that our implementation performs competitively on real world data sets when compared to some of the

[3] For the quadratic case, Bregman's method reduces to Hildreth's method [9], however we continue using the name Bregman's method to indicate that the same ideas could be applied to handle other convex penalties too.

leading SVM implementations. The latter set of experiments show two things: i) the efficiency of our ℓ_1-norm SVM implementation, and ii) the importance of solving the *primal* problem for data with highly skewed dimensions.

We implemented our algorithm in C++ using the the sparse matrix library SSLib [18]. The experiments reported in Section 4.1 were performed on a Pentium 4, 3GHz Linux machine equipped with 1GB RAM, whereas those in Section 4.2 were performed on a Pentium Xeon 3.2GHz Linux machine with 8GB RAM.

4.1 Classification Experiments on Real Data

Below we report classification results for several real data sets. Table 1 presents these results, wherein we report both training and test accuracies, as well as the respective running times of the algorithms tested (excluding I/O). As attested to by the results, our ℓ_1-SVM performs competitively against standard SVM packages such as SVM$^{\text{light}}$ and LIBSVM (version 2.82). The results given in Table 1 are merely illustrative. More extensive parameter tuning would definitely lead to better accuracies than reported.

We selected some of the data sets made available on the LIBSVM [4] webpage and the UCI machine learning repository [7]. The datasets used were

1. Liver-UCI [7]—345 $\times$ 7. No test set.
2. W7A [16]—24,692 training points with 300 features; 25,057 test points.
3. Ijcnn [17]—49,990 training points with 22 features; 91,701 test points.
4. RCV1 [11]—20,242 training points with 47,236 features; 677,399 test points.

Table 1. Training accuracies and associated running times for our ℓ_1-SVM implementation with both soft and hard margins ($C = \infty$), and comparative numbers for SVM$^{\text{light}}$ and LIBSVM. The running times reported are exclusive of the time spent in I/O. We added timing computation code to LIBSVM. For the data sets that came with a separate test collection, we also report test accuracies and test times. Our implementation was overall faster in both training and testing, with a marginal sacrifice in terms of accuracies.

Data set	ℓ_1-SVM (hard)	ℓ_1-SVM (soft)	SVM$^{\text{light}}$	LIBSVM
Liver-UCI	69.6% (0.02s)	71.2% (0.04s)	74.8% (0.12s)	70.4% (0.03s))
W7A	97.7% (1.49s)	98.6% (3.3s)	98.7% (6.2s)	98.7% (12.6s)
W7A (test)	97.6% (0s)	98.6% (0s)	98.7% (0.01s)	98.7% (3.4s)
Ijcnn	90.3% (0.04s)	92.1% (1.5s)	92.4% (22.4s)	92.4% (84s)
Ijcnn (test)	90.1% (0.01s)	91.7% (0.01s)	92.1% (0.08s)	92.1% (85.1s)
RCV1	99.6% (0.08s)	98.8% (2.2s)	98.9% (19s)	98.9% (318s)
RCV1 (test)	96.0% (0.29s)	96.3% (0.30s)	96.3% (0.90s)	96.3%($\sim$ 82min)

For all the implementations tested, we used the same value for the cost parameter C. Further, for both SVM$^{\text{light}}$ and LIBSVM we used the linear kernels. Observe that our ℓ_1-SVM outperforms both SVM$^{\text{light}}$ and LIBSVM in terms of training and testing speed, with a small drop in accuracy—except for the RCV1 dataset, where the ℓ_1-norm SVM has higher training accuracy.

4.2 Scalability Experiments on Synthetic Data

We performed a series of experiments on synthetically generated data to test the scalability of our ℓ_1-SVM. We sampled an equal number of points from two multidimensional von Mises-Fisher distributions [15], with overlapping means, so that the data were not linearly separable. We compare our implementation

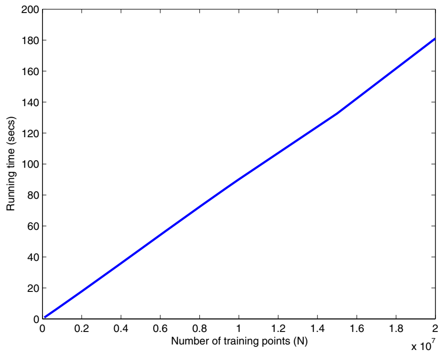


Fig. 1. Running time of our ℓ_1-SVM to demonstrate its scalability. The plot ranges from $N = 100{,}000$ to 20 million, and the corresponding times range from 0.89s to 181s. The run time can be seen to grow linearly in the number of training points (because the number of training points $N \gg M$, no effect of dimensionality is discernible).

against the Linear Programming Chunking (LPC) approach of [2]. Empirically, our method for solving the ℓ_1-SVM is several orders of magnitude faster than the LPC method. An exact number representing the speedup is not possible to provide since the authors of LPC ran their experiments on a different platform than ours. Nevertheless, we offer a conservative estimate of speed to permit a rough comparison. LPC was run on a cluster with 64 Sun UltraSPARC II processors, with a total of 8GB of RAM. We ran our ℓ_1-SVM code on an Intel Xeon 3.2GHz Linux machine with 8GB RAM. Conservatively assuming that both machines run at approximately the same speed (i.e., disregarding the fact that the cluster had 64 processors)[4] we can give a crude comparison between the two implementations. Note that these numbers are merely indicative of the speedup, since LPC and ℓ_1-SVM were run on different machines. The LPC method consumed 6.94, 25.91, and 231.32 hours, for 200,000, 500,000, and 1 million points, respectively. In comparison, our method took 1.76, 4.43, and 8.8 seconds for the same sized datasets. These numbers are compelling and show that our ℓ_1-SVM runs several orders of magnitude faster than the LPC algorithm while trying to solve the same problem. Hence, for such large scale datasets, it should be the

[4] This estimate is conservative because if one compares the performance of the cluster with that of a single Xeon based machine, the cluster is a few times faster—as can be ascertained by going through CPU/System comparison benchmarks.

method of choice. Our speed gains come from two sources: i) the decomposition approach, and ii) solving the primal problem instead of the dual.

We mention in passing that Mangasarian and Musicant [14] solve a particular version of the ℓ_2-norm SVM problem by attacking the dual formulation using an active set approach. Our ℓ_1-norm implementation can be modified to solve the primal version of their problem and once again outperforms the dual. Mangasarian and Musicant [14] reported running times of (on a 400MHz Pentium II Xeon processor with 2GB RAM) of 38 minutes for a problem size of 4 million points (in $\mathbb{R}^{32}$), and 96.57 minutes for 7 million points. Comparative numbers for our implementation can be obtained from Figure 1. Interpolation yields the estimates 36 seconds for 4 million points and 63 seconds for 7 million points.

5 Discussion

In this paper we treated the solution of a linear programming based ℓ_1-norm SVM problem by converting it into a quadratic program, which was then solved by an efficient decomposition method. As far as we know, nobody has applied this idea to the solution of ℓ_1-SVMs before. We saw that the decomposition method permits efficient solution of extremely large-scale problems. Furthermore, our results corroborate the non-surprising, but often overlooked fact that for problems where the number of training points vastly outnumbers their dimensionality, solving the primal problem is more efficient.

Since the decomposition procedures that we invoked are quite general, they can easily be adapted to solve related problems such as SV-regression, linear programming SVMs [6, 8], and the so-called ϵ- and ν-SVM variants.

5.1 Future Work

The ℓ_1-SVM presented in this paper is a preliminary piece of work. Many further refinements need to be incorporated into it to make it a highly competitive and accurate SVM training engine. Notable extensions to it that are currently under preparation include:

- Automatic determination of a good values for the cost parameter C and the control parameter ϵ.
- Improved methods for automatically determining early stopping criteria so that the algorithm takes minimum amount of running time without sacrificing too much accuracy.
- Additional improvements to the algorithm itself to improve its rate of convergence, and decrease errors due to numerical difficulties.

Acknowledgments

I thank Brian Kulis for initial discussions about the ℓ_1-SVM.

References

[1] K. Bennett. Combining support vector and mathematical programming methods for classification. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*, pages 307–326. MIT Press, 1999.

[2] P. S. Bradley and O. L. Mangasarian. Massive data discrimination via linear support vector machines. *Optimization Methods and Software*, 13(1), 2000.

[3] Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications.* Oxford University Press, 1997.

[4] C-C. Chang and C-J. Lin. LIBSVM: a libary for support vector machines, 2001. *http://www.csie.ntu.edu.tw/˜cjlin/libsvm.*

[5] O. Chapelle. Training a support vector machine in the primal. Technical report, Max Planck Institute for Biological Cybernetics, 2006.

[6] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[7] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[8] T. Graepel, R. Herbrich, B. Schölkopf, A. Smola, P. Bartlett, K.-R. Müller, K. Obermayer, and R. Williamson. Classification on proximity data with lp-machines. In *9th Int. Conf. on Artificial Neural Networks: ICANN*, 1999.

[9] C. Hildreth. A quadratic programming procedure. *Naval Res. Logist. Quarterly*, 4, 1957.

[10] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*, pages 42–56. MIT Press, 1999.

[11] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[12] O. L. Mangasarian. Normal solutions of linear programs. *Mathematical Programming Study*, 22:206–216, 1984.

[13] O. L. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research*, 2006.

[14] O. L. Mangasarian and D. R. Musicant. Active support vector machine classification. In *NIPS*, 2001.

[15] K. V. Mardia and P. Jupp. *Directional Statistics.* John Wiley and Sons Ltd., second edition, 2000.

[16] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*, pages 185–208. MIT Press, 1999.

[17] D. Prokhorov. Slide presentation in IJCNN'01, Ford Research Laboratory. IJCNN 2001 neural network competition, 2001.

[18] S. Sra and S. S. Jegelka. SSLib: Sparse Matrix Manipulation Library. *http://www.cs.utexas.edu/users/suvrit/work/progs/sparselib.html*, 2006.

[19] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *NIPS*, 2004.

An Efficient Approximation to Lookahead in Relational Learners

Jan Struyf[1], Jesse Davis[2], and David Page[2]

[1] Katholieke Universiteit Leuven, Dept. of Computer Science
Celestijnenlaan 200A, 3001 Leuven, Belgium
Jan.Struyf@cs.kuleuven.be

[2] University of Wisconsin-Madison, Dept. of Biostatistics and Medical Informatics and Dept. of Computer Sciences
1300 University Avenue, Madison, WI 53706, USA
jdavis@cs.wisc.edu, page@biostat.wisc.edu

Abstract. Greedy machine learning algorithms suffer from shortsightedness, potentially returning suboptimal models due to limited exploration of the search space. Greedy search misses useful refinements that yield a significant gain only in conjunction with other conditions. Relational learners, such as inductive logic programming algorithms, are especially susceptible to this problem. Lookahead helps greedy search overcome myopia; unfortunately it causes an exponential increase in execution time. Furthermore, it may lead to overfitting. We propose a heuristic for greedy relational learning algorithms that can be seen as an efficient, limited form of lookahead. Our experimental evaluation shows that the proposed heuristic yields models that are as accurate as models generated using lookahead. It is also considerably faster than lookahead.

1 Introduction

Symbolic machine learning algorithms, such as rule or decision tree induction algorithms, search a large space of candidate models to find a suitable model [8]. Each search step consists of generating a new model and evaluating it on the set of training examples. This is repeated until a sufficiently accurate model is found. Naïvely enumerating all possible models is generally too computationally expensive, therefore, machine learning algorithms employ intelligent search strategies, such as greedy, randomized, or branch-and-bound search.

Greedy or hill-climbing search is often used because of its computational efficiency. Greedy search constructs a sequence of models such that each model is a locally optimal refinement of the previous model in the sequence. The search ends after meeting a stop criterion, such as no refinement significantly increases the evaluation score. The main disadvantage of greedy search is its shortsightedness. Greedy search misses refinements that yield a high evaluation score only in combination with further refinements. Inductive logic programming (ILP) [7] algorithms are especially susceptible to this problem. ILP algorithms use first-order logic to represent the data and models. Data are represented with facts and rules are represented by clauses. Consider the clause `happy(C)`$\leftarrow$`account(C,A,B)`$\wedge$`B=high`, which states that a customer of a bank is happy if one of the customer's accounts has a high balance. Greedy search will fail to find the first literal of this clause because it is non-discriminating, that is, it does not alter the evaluation score of the clause. The evaluation score (e.g., the accuracy of the clause) usually only depends on the number of positive and negative examples covered by the clause. In our example, all customers have at least one account. Therefore the examples covered, and consequently the evaluation scores of the clauses `happy(C)`$\leftarrow$`true` and `happy(C)`$\leftarrow$`account(C,A,B)`, are identical. Greedy search will thus never select the latter clause and therefore never find our example clause.

Lookahead [9] helps greedy search overcome myopia. Instead of adding only one literal at each search step, lookahead adds the best conjunction consisting of at most $n + 1$ literals, where n is the lookahead depth. With depth one lookahead, our example clause could be learned in one refinement step. Unfortunately, execution time increases exponentially with the lookahead depth.

In this paper, we propose a feature based evaluation score for literals that is comparable to a limited form of depth one lookahead. Our score can be computed efficiently from a set of tables that can be pre-computed with one pass over the data. The resulting approach is computationally more efficient than lookahead and yields models that have a comparable accuracy.

2 An Efficient Approximation to Lookahead

We first illustrate the idea behind our approach with the example task of predicting when a bank customer is happy. Consider evaluating the clause `happy(C)` $\leftarrow$ `account(C,A,B)`. In this case, variables `A` and `B` serve as output variables. When computing the score of the clause, we can potentially glean information from which constants bind to these variables. This information about bindings can help guide the search towards which refinement to pick.

Assume we use accuracy as the scoring function. Then the refined clause `happy(C)`$\leftarrow$`account(C,A,B)` yields no benefit over the clause `happy(C)`$\leftarrow$`true` (because it covers the same examples). Even though both clauses have the same score, intuitively the refined clause appears more promising. By looking at the bindings for variable `B` we could observe, for example, that people who have an account with a high account balance tend to be happy: variable `B` takes a binding of *high* more frequently for *Happy* cases compared to *Not Happy* cases. Thus, we can see that by placing a condition on the variable `B` in a future step – that is, adding a literal with `B` as input – we might be able to get a clause with a high accuracy.

By looking at the bindings for `B`, we have leveraged the fact that the constants that bind to this variable are shared across several examples. It is more difficult to perform this analysis for a variable such as `A` because it has a unique value (account number) for every account. To illustrate how we might handle this type of variables, assume that we have three unary predicates that take an account number as input: `savingAcc(A)`, `checkingAcc(A)`, and `moneyMarketAcc(A)`.

Even if all customers with money market accounts are happy, and all others are not happy, the individual account numbers tell us nothing about whether a customer is happy. The benefit comes from the fact that these account numbers refer to money market accounts. Thus, we can assess the clause's quality based on which predicates, like `moneyMarketAcc(A)`, hold for the bindings of variables. Nevertheless, we wish to do this assessment without the full cost of lookahead.

2.1 Features and Evaluation Scores

In the previous section, we have shown that class-wise counts for variable bindings, computed either directly or indirectly via the predicates that hold for each binding, yield extra information about the quality of a clause. In this section, we formalize this idea.

Suppose we have clause c and need to evaluate the quality of extending c with the literal l. Instead of just counting the number of examples in each class that the resulting clause covers, we also compute information based on the bindings for the output variables of l. We define a set of features for these bindings, which can be pre-computed prior to running the learning algorithm. The first feature is l itself. The other features are conjunctions of l with a second literal. The second literal uses exactly one of l's output variables as an input variable and does not share any other variables with clause c[1].

Definition 1 (Literal Features). *Given a clause c and a literal l, the set of features* $\mathrm{F}(c,l)$ *for l given c is defined as follows.*

$$\mathrm{F}(c,l) = \{\, l \,\} \,\cup\, \{\, l \wedge l_i \mid \mathrm{legal}(c,l,l_i) \,\}$$

$$\mathrm{legal}(c,l,l_i) = \#\{(\mathrm{vars}(l) - \mathrm{vars}(c)) \cap \mathrm{vars}(l_i)\} = 1 \,\wedge\, \mathrm{vars}(l_i) \cap \mathrm{vars}(c) = \emptyset$$

with $\mathrm{vars}(x)$ *the set of variables appearing in x. (# denotes set cardinality.)*

Consider again the bank domain and suppose that the learner starts with clause $c =$ `happy(C)`$\leftarrow$`true` and is about to evaluate literal $l =$ `account(C,A,B)`. The first column of Table 1 lists the features $\mathrm{F}(c,l)$.

We define an evaluation score that incorporates information from the feature set. Several feature representations are possible and each representation allows different evaluation scores to be defined. The most general representation can be found in column R1 of Table 1. The columns for customer c_1 to customer c_n indicate for each customer which features hold. Column R2 of Table 1 shows a second representation, which stores class-wise counts of the examples for which each feature holds. Column R3 contains the most restrictive representation. Here we only compute some score for each feature.

In the most general representation of the features (R1), each class of customers (happy or unhappy) can be seen as a cluster with each instance being the feature representation of one of the training examples (i.e., a column of R1).

[1] The constraints are introduced to limit the number of features.

Table 1. Feature representations for the literal `account(C,A,B)` in the bank application. (We assume that predicate arguments are typed; the same variable can only appear at argument positions of the same type.).

	R1			R2		R3
F(c, l)	c_1	...	c_n	#Happy	#Unhappy	Score
`account(C,A,B)`	1	...	1	3000	3000	0.50
`account(C,A,B)`$\wedge$`B=high`	1	...	0	2600	0	0.93
`account(C,A,B)`$\wedge$`B=medium`	0	...	1	2000	1000	0.67
`account(C,A,B)`$\wedge$`B=low`	0	...	1	200	3000	0.03
`account(C,A,B)`$\wedge$`card(A,R)`	1	...	1	3000	3000	0.50
`account(C,A,B)`$\wedge$`loan(A,L)`	0	...	1	1000	2000	0.33
...	...	...	...	...	...	...

If the cluster of the happy customers is far apart from the cluster of unhappy customers according to some distance metric, then the literal is potentially good at separating the happy from the unhappy customers and should be assigned a high score. For example, one could define the quality of the literal as the Euclidean distance between the prototypes of the happy and the unhappy customers.

In this paper, we restrict ourselves to the least general representation (R3). R3 represents each feature by a score (e.g., its classification accuracy). The score of a literal is computed by aggregating the scores of its features. We choose to compute the score of a literal as the maximum of the scores of the features. Note that this choice can be compared to a limited form of depth one lookahead. Depth one lookahead, however, imposes fewer restrictions. It allows conjunctions with literals sharing no variables with l, literals that share more than one variable with l, and that share variables with the rest of the clause.

2.2 Efficiently Computing Class-Wise Counts

In this section, we show how to efficiently compute class-wise example counts for the features, which corresponds to representation R2 in Table 1. Note that R3 can be easily computed from R2. The algorithm for finding the class-wise counts relies on a set of pre-computed tables.

We construct one table for each type. Each of these tables contains one column for each predicate argument of that type. The table is indexed by the constants in the domain of the type. Each cell $T_t[x, p, a]$ of the table for type t stores a Boolean indicating that predicate p holds if constant x is substituted for argument a.

The ComputeCounts algorithm in Fig. 1 uses the pre-computed tables to calculate the class-wise counts for the features. The algorithm creates a table called Counts, which has a row for each feature and a column for each class. The main loop of the algorithm iterates over the training examples. For each training example, it computes which features hold in the array Holds (the array Holds corresponds to a column of R2 in Table 1). Next, it increments the count for each feature that holds, conditioning on the example's class.

```
**procedure** ComputeCounts($c, l$)
1: $c' := c$ extended with $l$; $V :=$ vars($l$) − vars($c$)
2: for each training example $e$
3:     if the condition of $c'$ holds for $e$
4:         Holds[1] := 1; $\forall i > 1$ : Holds[$i$] := 0
5:         for each binding $X/x$ of a variable $X \in V$
6:             for each column $(p, a)$ of $T_{\text{type}(X)}$
7:                 $i$ such that predicate$(l_i) = p \wedge$ argument$(l_i, a) = X$
8:                 if $T_{\text{type}(X)}[x, p, a] = 1$ then Holds[$i$] := 1
9:         $\forall i$ : Counts[$i$][class($e$)] := Counts[$i$][class($e$)] + Holds[$i$]
10: return Counts
```

Fig. 1. An algorithm computing the class-wise example counts for the features

To compute which of the features hold for a given training example, the algorithm executes the given clause c extended with literal l on the example. If it covers the example, then the first element of Holds is set to one. We assume here that the first feature is the literal itself. To compute whether or not the other features hold, the algorithm looks at the bindings that are obtained for the output variables V of l while executing the clause. For each binding of $X \in V$ to a constant x, it looks up the corresponding row in the pre-computed table for X's type. Each element in this row indicates if a given feature holds for this binding. The algorithm records this in the array Holds. After all bindings have been processed, Holds indicates the features that hold for the example. Holds can now be used to update the class-wise counts.

ComputeCounts is more efficient than computing the required counts for each of the features separately for two reasons. First, it computes the counts for all features in one pass over the data. This is, clause c extended with l needs to be executed only once on each example instead of once for each feature. Second, it caches in pre-computed tables whether or not a given feature holds.

3 Experimental Evaluation

We compare the feature based evaluation of literals (FBE) presented in this paper to lookahead. The conjecture is that (1) models built using FBE have a comparable accuracy to models built using lookahead and (2) FBE is considerably faster.

We test FBE in the ILP algorithm Tilde [2], that is available in ACE 1.2.9 [3]. Tilde induces first-order logical decision trees. Briefly, these are decision trees similar to the ones of C4.5 [10], but the tests in the internal nodes are expressed in first-order logic, meaning that each test is a conjunction of one or more literals. We use the exhaustive lookahead feature of Tilde. For a lookahead depth of n each node can contain at most $n + 1$ literals and these are found by means of exhaustive search. The lookahead algorithm implemented in Tilde provides a challenging baseline for comparison because it employs query-pack execution [3], which has been shown to yield large gains in execution time in combination with lookahead.

Table 2. Comparison of Tilde with our new FBE approach to Tilde with exhaustive lookahead of depth 0 to 2. The columns represent the data set and its number of positive/negative examples, the accuracy and AUPRC measured using cross-validation (with 90% confidence intervals), the CPU time for the cross-validation (not including loading of the data and pre-computing tables - the latter are small and range from 0.01 to 0.32 sec/fold), and the average tree size. Significant wins/losses of lookahead versus FBE are indicated with $\oplus$ and $\ominus$ (significance level 0.01). All experiments are performed on an Intel Xeon 3.3GHz / 4GB Linux system.

Data	Method	Accuracy		AUPRC		Time (sec)	Size (nodes)
Muta188	L0	69.1 ± 7.5		70 ± 8	$\ominus$	1	1.0
#p = 125	L1	74.5 ± 4.7		84 ± 7		62	11.8
#n = 63	L2	73.9 ± 6.7		79 ± 6		1455	11.8
	FBE	76.6 ± 5.3		85 ± 8		13	14.8
Muta230	L0	63.9 ± 3.3	$\ominus$	65 ± 4	$\ominus$	1	1.7
#p = 138	L1	74.8 ± 5.8		84 ± 3		321	18.8
#n = 92	L2	73.5 ± 3.4		81 ± 7		2482	15.3
	FBE	74.8 ± 4.7		86 ± 4		36	19.2
Financial	L0	86.8 ± 0.7	$\ominus$	13 ± 1	$\ominus$	0	0.0
#p = 31	L1	96.6 ± 1.5		84 ± 9		25	2.0
#n = 203	L2	96.2 ± 1.8		81 ± 9		2716	1.4
	FBE	96.6 ± 1.5		84 ± 9		13	2.0
Sisyphus A	L0	62.1 ± 0.0	$\ominus$	62 ± 0	$\ominus$	3	0.0
#p = 10723	L1	94.9 ± 0.5		97 ± 1		4302	18.6
#n = 6544	L2	96.6 ± 0.2	$\oplus$	98 ± 0	$\oplus$	161253	22.7
	FBE	94.8 ± 0.3		97 ± 1		779	17.4
Sisyphus B	L0	71.4 ± 0.0	$\ominus$	29 ± 0	$\ominus$	1	0.0
#p = 3705	L1	75.9 ± 0.7		59 ± 1		5544	57.2
#n = 9229	L2	92.0 ± 0.3	$\oplus$	86 ± 1	$\oplus$	92053	14.6
	FBE	76.1 ± 0.7		59 ± 2		223	43.1
UWCSE	L0	93.6 ± 2.3		39 ± 17		20	25.4
#p = 113	L1	94.0 ± 2.3		29 ± 14		163	31.6
#n = 2711	L2	94.3 ± 2.3		33 ± 13		8098	24.6
	FBE	95.0 ± 1.3		34 ± 19		74	45.2
Yeast	L0	87.7 ± 0.4	$\ominus$	68 ± 2		1479	82.0
#p = 1299	L1	88.0 ± 0.6		63 ± 2	$\ominus$	3630	65.8
#n = 5456	L2	88.0 ± 0.5	$\ominus$	62 ± 2	$\ominus$	436062	62.4
	FBE	88.8 ± 0.4		71 ± 1		3023	95.1
Carc	L0	62.1 ± 4.5		66 ± 4		23	15.0
#p = 182	L1	60.3 ± 4.1		67 ± 4		1843	32.4
#n = 148	L2	60.0 ± 3.4		64 ± 4		2183	17.5
	FBE	60.3 ± 4.3		67 ± 5		262	35.3
Bongard	L0	98.1 ± 0.4	$\ominus$	98 ± 1	$\ominus$	90	11.4
#p = 671	L1	99.6 ± 0.3		100 ± 0		215	9.9
#n = 864	L2	100.0 ± 0.0		100 ± 0		22637	5.0
	FBE	99.5 ± 0.3		100 ± 0		31	14.1

We have implemented FBE in Tilde. To compute the conjunction for a node of the tree, we use greedy search with the FBE score to find a conjunction of at most two literals. Therefore, our results are comparable to depth one lookahead. The evaluation score of a literal is computed as the maximum of the information gain ratios [10] computed for its features (the latter are computed using ComputeCounts shown in Fig. 1).

We perform experiments on nine data sets: two versions of Mutagenesis (Muta [7], p. 344), Financial [1], Sisyphus task A and B [4], UWCSE [5], Yeast [5], Carcinogenesis (Carc [7], p. 345), and Bongard ([7], p. 136). Details of the data sets can be found in the listed references. We run Tilde with FBE and with exhaustive lookahead of depth 0 to 2. We estimate the predictive accuracy and area under the precision-recall curve (AUPRC) [6] of the obtained models using 10 fold stratified cross-validation for all data sets except UWCSE. For this data set, we use the 5 folds provided by the original authors.

Table 2 presents the results. Most results confirm our hypothesis. The results obtained with FBE have comparable accuracy and AUPRC to those with lookahead depth one (L1) and are never significantly worse. For six data sets the accuracy (AUPRC) of FBE is significantly better than that of L0. Note that for some data sets, Tilde fails to build a model without lookahead (cf. the Size column). The reason is that none of the evaluated clauses yields a non-zero gain in these cases. For the Sisyphus data sets, L2 performs significantly better than FBE. Note that L2 is more expressive (it allows two literals in each node). It is also 200-400 times slower on these data sets.

The FBE approach is always faster than L1 and L2. It is on average 7 times faster than L1 and 200 times faster than L2. Of course, our approach trades time for memory: it makes use of pre-computed tables. The memory required for storing these tables was, however, limited: the memory overhead over the space required for loading the system and the data was at most 12%.

4 Conclusions

Greedy machine learning algorithms and in particular Inductive Logic Programming (ILP) algorithms suffer from shortsightedness resulting in accuracy-wise suboptimal models. Lookahead helps greedy search overcome this shortcoming, but incurs an exponential increase in execution time. In this paper, we propose an alternative termed feature based evaluation (FBE). The idea behind feature based evaluation is to compute the score of a refinement based on a number of features that are defined for it. The particular instantiation of FBE that is considered in this paper can be seen as a restricted form of lookahead. In an experimental evaluation of the approach, we show that FBE yields models with an accuracy comparable to that of models built with lookahead and that FBE is considerably faster.

Other researchers have considered the problem of myopia in greedy ILP systems. Most approaches can be seen as a limited form of lookahead. These include determinate literals, template based lookahead, and macro-operators. Besides lookahead, beam-search has also been used. A comparison of systems implementing these different approaches appears in [9]. Skewing [11] also reduces myopia of greedy learners. Skewing is, however, less applicable to the type of myopia faced by relational learners, which occurs for non-discriminating literals that introduce useful new variables.

Interesting directions for further work include evaluating FBE in the context of a rule learner, investigating other evaluation scores based on FBE (e.g., the Euclidean distance mentioned in Section 2.1), and testing FBE with higher lookahead depths (e.g., to approximate depth two lookahead, one would add features consisting of two literals in the set of pre-computed tables described in Section 2.2).

Acknowledgments. Jan Struyf is a postdoctoral fellow of the Fund for Scientific Research of Flanders (FWO-Vlaanderen). David Page is supported in part by the U.S. National Science Foundation grant IIS 0534908.

References

1. P. Berka. Guide to the financial data set. In *ECML/PKDD 2000 Discovery Challenge*, 2000.
2. H. Blockeel and L. De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.
3. H. Blockeel, L. Dehaspe, B. Demoen, G. Janssens, J. Ramon, and H. Vandecasteele. Improving the efficiency of inductive logic programming through the use of query packs. *Journal of Artificial Intelligence Research*, 16:135–166, 2002.
4. H. Blockeel and J. Struyf. Frankenstein classifiers: Some experiments on the Sisyphus data set. In *Workshop on Integration of Data Mining, Decision Support, and Meta-Learning (IDDM'01)*, 2001.
5. J. Davis, E. Burnside, I. C. Dutra, D. Page, and V. Santos Costa. An integrated approach to learning bayesian networks of rules. In *16th European Conference on Machine Learning*, pages 84–95, 2005.
6. J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. In *23nd International Conference on Machine Learning*, 2006. To appear.
7. S. Džeroski and N. Lavrač, editors. *Relational Data Mining*. Springer, 2001.
8. T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.
9. L. Peña Castillo and S. Wrobel. A comparative study on methods for reducing myopia of hill-climbing search in multirelational learning. In *21st International Conference on Machine Learning*, 2004.
10. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann, 1993.
11. S. Ray and D. Page. Generalized skewing for functions with continuous and nominal attributes. In *22nd International Conference on Machine Learning*, pages 705–712, 2005.

Improvement of Systems Management Policies Using Hybrid Reinforcement Learning

Gerald Tesauro[1], Nicholas K. Jong[2], Rajarshi Das[1], and Mohamed N. Bennani[3]

[1] IBM TJ Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532 USA
[2] Dept. of Computer Sciences,Univ. of Texas, Austin, TX 78712 USA
[3] Dept. of Computer Science, George Mason Univ., Fairfax, VA 22030 USA

Abstract. Reinforcement Learning (RL) holds particular promise in an emerging application domain of performance management of computing systems. In recent work, online RL yielded effective server allocation policies in a prototype Data Center, without explicit system models or built-in domain knowledge. This paper presents a substantially improved and more practical "hybrid" approach, in which RL trains offline on data collected while a queuing-theoretic policy controls the system. This approach avoids potentially poor performance in live online training. Additionally we use nonlinear function approximators instead of tabular value functions; this greatly improves scalability, and surprisingly, eliminated the need for exploratory actions. In experiments using both open-loop and closed-loop traffic as well as large switching delays, our results show significant performance improvement over state-of-art queuing model policies.

1 Introduction

The ongoing rapid growth in scale and complexity of the world's IT infrastructure has motivated intense focus on automating management of computing systems. Major IT vendors and universities have recently initiated research on "autonomic" computing systems, seeking means by which computing systems may dynamically reconfigure themselves, continually optimize their performance, detect and repair faults, and protect themselves from external attacks. Machine learning may prove to be of great benefit in developing such capabilities. While standard approaches to systems management rely on extensive domain knowledge, ML may evade the knowledge bottleneck by automatically learning high-quality management policies based solely on observed data.

Reinforcement Learning (RL) methods hold particular promise for systems performance management. RL can obtain high-quality policies without explicit models or extensive built-in system knowledge. Also, by accounting for the long-range consequences of decisions, RL can surpass other methods that treat dynamical effects only approximately, or ignore them altogether (e.g. traditional steady-state queuing theory), or cast the decision problem as a series of unrelated instantaneous optimizations.

Initial work [1,2,3] applying RL to systems management shows promise, but we are concerned with two potentially significant practical problems. First, the above studies suggest that tens of thousands of observations may be required, at natural intervals ranging from several seconds to several minutes of real time. This implies online training times could be as long as several months, which would be unacceptable in many applications. Second and perhaps more importantly, the performance obtained during live online training may be unacceptably poor, due to two factors: (a) an arbitrarily bad initial policy in the absence of domain knowledge or good heuristics; (b) in general RL procedures also need a certain amount of exploration of suboptimal actions, which may be exceedingly costly to implement in a live system.

To address the above practical limitations, we devise in this paper a hybrid method combining the advantages of both explicit model-based methods and *tabula rasa* RL. Instead of training an RL module online on the consequences of its own decisions, we propose offline training on data collected using a stationary external policy (based e.g. on an appropriate queuing model) to manage the system. This assures acceptable performance while gathering training data, assuming a decent initial policy.

Offline sweeps through the acquired dataset may be orders of magnitude faster than the underlying physical time scales. This permits multiple sweeps through the dataset, enabling training of sophisticated nonlinear value function approximators which learn too slowly to be trained online. Function approximators generalize training experience across states and actions, reducing the need for extensive exploration.

The idea of combining RL with an external policy is not new. Our hybrid RL method is similar to the "implicit imitation" framework of [4], in which an agent learns based on the state transitions and rewards generated using the policy of another agent.

The main contributions of our work are as follows:

1. We demonstrate effectiveness of hybrid RL in a prototype system comprising real servers, realistic Web-based workloads, and realistic time-varying demand. Our results significantly outperform state-of-art queuing models and are much better than our prior online RL approach [2] that only achieved equality to queuing models. To our knowledge this is the first time that queuing models have been surpassed in a transactional performance management task, an achievement considered both significant and impressive by systems experts [5].

2. We provide the first demonstration in a systems management application that RL can deal effectively and automatically with dynamic consequences of management decisions – transients and switching delays – associated with server reallocation.

3. Several aspects of our results may be counterintuitive or surprising to a machine learning audience and hence may spur further research. One such finding is that our methodology works quite well in practice, despite lacking rigorous convergence guarantees. Another surprising finding is that we did not need to add any exploration to the base queuing model policies. Finally, it seems non-intuitive that hybrid RL can outperform queuing models while receiving only a subset of the available inputs.

4. Our approach is much more practical and scalable than the online approach of [2], since we avoid poor performance during online training, and function approximators offer much better scalability than lookup tables. Consequently, we anticipate possible wide usage of hybrid RL in many different types of systems management applications.

The rest of the paper is organized as follows. Section 2 describes our prototype Data Center. Section 3 presents specifics of our hybrid RL approach. Sections 4 and 5 give performance results and discuss why RL outperforms the queuing models. Conclusions are given in Section 6. Our queuing model policies are detailed in [6].

2 Experimental Setup

Our experimental testbed, illustrated in Fig. 1, follows the scenario in [1] for dynamically allocating a set of identical servers among multiple web applications. Each application has its own Application Manager module which communicates with a Resource Arbiter module regarding resource needs. Allocation decisions are made in five-second time intervals as follows: Each Application Manager *i* reports to the Arbiter a commonly scaled utility curve $V_i(\cdot)$ estimating expected business value as a function of number of allocated servers. Business value is defined in monetary units by a Service Level Agreement (SLA), which stipulates payments or penalties as a function of performance. Upon receipt of the utility curves, the Arbiter solves for the globally optimal allocation maximizing total expected value. It then conveys to each application a list of assigned servers, which are used in dedicated fashion until the next allocation decision.

Our standard testbed uses eight HTTP servers (3.06GHz Xeon machines) and three applications. Two of the applications are separate instantiations of "Trade3," a web application that provides a realistic emulation of online trading. Each Trade3 SLA is a sigmoidal function of mean response time over the allocation interval. The third application is a long-running "Batch" workload that can be paused and restarted as servers are added and removed. This emulates a CPU-intensive task such as Monte Carlo portfolio simulations. Since Batch has steady need for resource, its SLA is a simple increasing function of number of assigned servers.

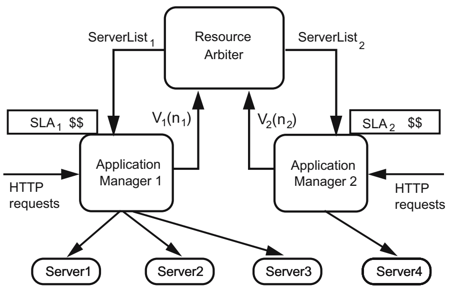


Fig. 1. Resource allocation scenario

Demand in each Trade3 environment is driven by a separate workload generator, which can operate either in open-loop or closed-loop mode. The open-loop mode generates Poisson HTTP requests with an adjustable mean arrival rate. The closed-loop mode simulates an adjustable, finite number of customers who alternate between waiting for a request to complete and thinking about sending the next request, with fixed think time distributions. To emulate stochastic bursty time-varying demand, we use a realistic Web traffic model [7] to adjust once per second either the closed-loop number of customers or the open-loop mean arrival rate.

3 Hybrid RL Approach

In the first part of our hybrid RL procedure, given in Algorithm 1, one obtains some initial policy and would expect that adding some form of off-policy exploration (e.g. softmax or ε-greedy) would be necessary to facilitate learning. However, we were surprised to discover that we can learn substantially improved policies without any added exploration, so this step may not be necessary. One then runs the initial policy in the system and records a set of $(T+1)$ observations $\{(s_t, a_t, r_t), 0 \leq t \leq T\}$, where (s_t, a_t, r_t) are the observed state, action and immediate reward at time t. Using these data, we train a value function $Q(s,a)$ estimating long-range expected value taking initial action a in state s. $Q(s,a)$ defines a new RL-based policy which then replaces the original policy.

Algorithm 1 envisions batch training, wherein for each observation (s_t, a_t, r_t) we compute in Line 7 a target Q-value and then regress the current $Q(s_t, a_t)$ values toward their targets. Our targets derive from the well-known Sarsa rule [8]: $\Delta Q(s_t, a_t) = \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$. Sarsa was needed for technical reasons in our specific application, as detailed below, but in other applications it may well be possible to use Q-Learning instead. The batch training details will depend on the function approximator used. In some cases one can do incremental training at each observation; in other cases the entire batch may be needed, e.g., to construct a regression tree. For our problem, we use a standard direct gradient method to train neural net weights, which works well in practice but carries a theoretical risk of divergence. In this case, we could use instead a residual gradient method [9], which guarantees convergence to local Bellman error minima. Typically the batch training will proceed for some number of sweeps through the training set until some error criterion (e.g., SSE) reaches an asymptotic value.

Algorithm 1. Hybrid RL procedure

```
 1: Add exploration mechanism to initial policy if necessary
 2: Run initial policy and record $\{(s_t, a_t, r_t), 0 \leq t \leq T\}$
 3: Initialize $Q$-function approximator (e.g. randomly)
 4: repeat
 5:    $SSE \leftarrow 0$ {sum squared error}
 6:    for all $t$ such that $0 \leq t < T$ do
 7:       $target \leftarrow r_t + \gamma Q(s_{t+1}, a_{t+1})$
 8:       $error \leftarrow target - Q(s_t, a_t)$
 9:       $SSE \leftarrow SSE + error \cdot error$
10:       Train $Q(s_t, a_t)$ towards $target$
11:    end for
12: until CONVERGED($SSE$)
```

3.1 Experimental Implementation

We could implement hybrid RL at the global Arbiter level, but the global state space scales exponentially with the number of applications. Hence we adopt the decompositional approach of [2] in which RL was implemented separately within each Trade3 application, using only local state and local number of allocated servers. This scales to many applications and worked well empirically despite lacking convergence guarantees.

The separation of learning from decision-making necessitates an on-policy algorithm like Sarsa in Line 7 of Algorithm 1: local Q-Learning's $\max_a Q(s_{t+1}, a)$ term incorrectly assumes at the next time step that each application gets all the resources.

In each application, the action a_t comprises the local number of servers n_t allocated at time t. To represent the state s_t, many sensor readings could be used, but for simplicity we follow [2] in using only the current demand λ_t. We are also particularly interested in the dynamic consequences of allocation decisions. For example, when a server is added there may be initial transient suboptimal performance, or switching delays, during which the server is unavailable. To handle such effects, we employ a "delay-aware" representation that adds the previous allocation n_{t-1} to the state representation at time t. As long as such effects last no more than one allocation interval, this should suffice to learn the impact on expected value (longer delays would require n_{t-2}, etc.).

We represent the Q-function $Q(\lambda_t, n_{t-1}, n_t)$ with neural networks, due to their robust high-dimensional generalization and prior RL application successes. In each Trade3 application, we train by backpropagation a standard MLP with three input units, 12 sigmoidal hidden units, and one linear output unit. We scale the inputs to the interval $[0,1]$, and using a learning rate of 0.005, 10-20 thousand sweeps through the dataset suffice to converge empirically. We set the discount parameter $\gamma = 0.5$.

4 Results

We first present results for open-loop and closed-loop systems without switching delays. The performance measure is total SLA revenue per allocation decision summed over all three applications. Each data point represents performance in a 12-hour run using a specific demand time series in each Trade3 that is repeated in all of the experiments. For a variety of initial model-based policies, we compare the initial policy performance with that of its corresponding hybrid RL trained policy.

The open-loop and closed-loop results are shown in Fig. 2. The percentage figures denote relative improvement of hybrid RL policies over their corresponding initial queuing models. (For the random initial policies, such percentages are shown in brackets as they have dubious meaning in our opinion.) The error bars denote 95% confidence intervals for the reported values; this calculation does not reflect the nearly identical demand traces used in each experiment. To address this factor we also performed paired T-test when comparing the hybrid RL results with each corresponding initial policy.

In the open-loop case we examine three initial policies: our open-loop models with and without exponentially smoothed parameter estimates [6], and for a baseline comparison, a uniform random allocation policy. We see substantial improvement of hybrid RL over each corresponding initial policy in both relative and absolute terms. In all pairs of experiments, a paired T-test rejects the null hypothesis that there is no difference between the performance means at 1% significance level with P-value $\leq 10^{-6}$.

In the closed-loop case we again examine a random initial policy, plus four different queuing model polices whose details are described in [6]. Once again we find substantial improvement of hybrid RL over each initial policy. The improvement over the random policy is enormous, while improvement over the queuing models is consistently at a double-digit percentage level with high statistical significance (rejected each null hypothesis using paired T-test at 1% significance level with P-value $\leq 4 \times 10^{-3}$).

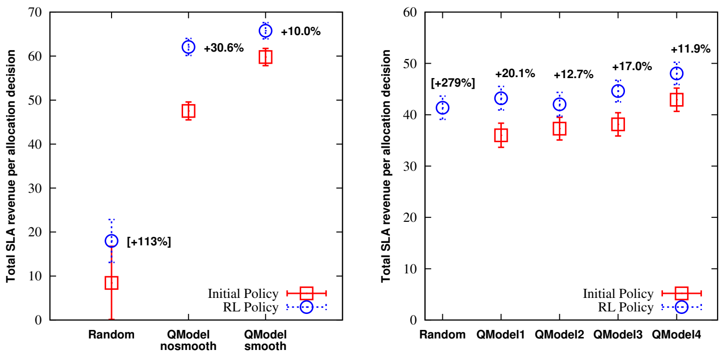


Fig. 2. (a)Performance of policies in open-loop zero-delay scenario. (b) Performance of policies in closed-loop zero-delay scenario. (The random policy performance lies off the scale at -23.0.).

We also note that, in replicating the online method of [2], we obtained results ∼5-10% worse than our best queuing models, so that hybrid RL also outperforms online RL. This may reflect difficulty in scaling online RL to larger state/action spaces.

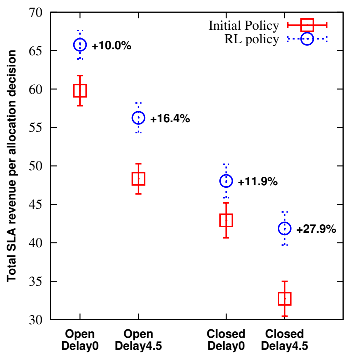


Fig. 3. Comparison of delay=4.5 sec with delay=0 results in open-loop and closed-loop scenarios

Fig. 3 compares our zero-delay results, using our best open-loop and closed-loop queuing models, with corresponding experiments that impose a delay of 4.5 seconds when a server is reassigned to a different application. The delay is asymmetric in that the server is immediately unavailable to the old application, but does not become available to the new application until 4.5 seconds have elapsed. We chose the delay to be a huge fraction of the five second allocation interval so that its empirical effects would be as clear as possible. Modeling of large delays is also of practical importance, since reallocation of servers in real Data Centers could entail several minutes of downtime. Fig. 3 shows that imposing this delay does in fact substantially harm the average performance in all cases. However, the amount of policy improvement of hybrid RL over its initial policy increases in both absolute and relative terms. In the open-loop scenario the improvement increases from 10.0% to 16.4%, while in the closed-loop scenario the improvement jumps from 11.9% to 27.9%.

5 Insights into Hybrid RL Outperformance

We offer three insights as to how hybrid RL is able to outperform the initial queuing model policies. The first has to do with estimation bias. For reasons too technical to detail here, the queuing model policies end up having a bias toward overprovisioning. due to interactions of their response time estimates with the nonlinear SLA function. However, the RL nets, by learning to estimate utility directly, are able to achieve less biased estimation errors. This leads to the Trade3 applications receiving slightly fewer servers on average, with a slight loss of Trade3 utility, but the loss is more than made up by substantially greater Batch utility.

The second point is that our RL nets are able to properly treat transients and switching delays, unlike the steady-state queuing models. The learned policies exhibit hysteresis, a tendency to prefer steady allocations over switching based on instantaneous state. Some evidence for this is seen in Table 1, which exhibits basic statistics averaged over the two Trade3 applications $T1$ and $T2$ from the eight experiments shown in Fig. 3. The quantity $<n_T>= (<n_{T1}> + <n_{T2}>)/2$ is the average number of assigned servers, while $<\delta n_T>= (<\delta n_{T1}> + <\delta n_{T2}>)/2$ is the RMS change in number of assigned servers from one time step to the next. We see that $<n_T>$ is slightly less for the RL nets than for the queuing models, and there is a further slight reduction for the RL nets for 4.5 sec delay compared to zero delay. More importantly, the $<\delta n_T>$ statistics reveal noticeably less server swapping when using RL nets compared to queuing models, with the effect becoming quite pronounced ($>\sim$50% reduction) in the 4.5 sec delay case.

Table 1. Mean number of servers $< n_T >$ assigned to a Trade3 application, and mean change in number of assigned servers $< \delta n_T >$ per time step, in the eight experiments plotted in Fig. 3

Experiment	$< n_T >$	$< \delta n_T >$
Open-loop Delay=0 QM	2.27	0.578
Open-loop Delay=0 RL	2.04	0.464
Open-loop Delay=4.5 QM	2.31	0.581
Open-loop Delay=4.5 RL	1.86	0.269
Closed-loop Delay=0 QM	2.38	0.654
Closed-loop Delay=0 RL	2.24	0.486
Closed-loop Delay=4.5 QM	2.36	0.736
Closed-loop Delay=4.5 RL	1.95	0.331

The reduction in $<\delta n_T>$ generally reflects hysteresis in the RL policies, and relates to our third insight, that RL policies exhibit greatly reduced thrashing. In experiments with 4.5 sec delay, massive thrashing under high load is a significant problem using the queuing model policies. An example of this is given in Fig. 4, which shows a five-minute interval in which $T1$ is moderately loaded and $T2$ is very heavily loaded. The

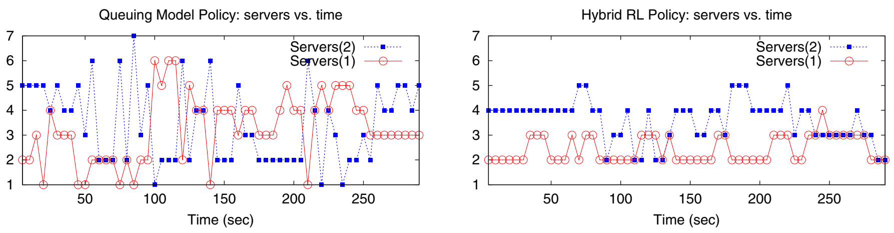


Fig. 4. Reduction of thrashing using Hybrid RL in closed-loop, 4.5 sec delay experiment

left plot shows the queuing model allocations, while the right plot shows the hybrid RL allocations under the same demand trace. We see much steadier allocations in the latter case. This is due partly to the RL value functions' general preference to have no more than five servers, and partly to their projected high switching cost which inhibits a large instantaneous increase in servers (say, from 2 to 5-6) within an application.

6 Conclusions

Our hybrid RL approach neatly takes advantage of RL's ability to learn in a knowledge-free manner, without an explicit system model or traffic model, and requiring little or no domain knowledge built into its state-space and value-function representations. However, our approach can exploit any available knowledge contained in an external policy, without having to interface to such knowledge. Moreover, using a simple "delay-aware representation" including the previous allocation decision, our approach also naturally handles transients and switching delays, which are dynamic consequences of reallocation lying outside the scope of traditional steady-state queuing models. On the other hand, hybrid RL also exploits the ability of model-based policies to achieve decent performance levels within a given system. This maintains acceptable performance while gathering training data, and avoids poor live performance expected in using online RL. We may also exploit robustness of model-based policies under various types of system changes, e.g. hardware upgrades or changes in the SLA, which require retraining of the RL value functions. When such changes occur, we can fall back on the model-based policy to deliver acceptable performance while accumulating new training data.

Hybrid RL may have wide applicability in many other areas of systems management. The most promising applications would exhibit: (a) a tractable state-space representation; (b) frequent online decision making depending upon time-varying system state; (c) frequent observation of numerical rewards in an immediate or moderately delayed relation to management actions; (d) pre-existing policies that obtain acceptable performance levels. Many applications clearly have such properties: among them are dynamic allocation of other types of resources, e.g., bandwidth, memory, LPARs, etc. We would also include performance-based online tuning of system control parameters, such as web server, OS or DB parameters. Finally, hybrid RL could conceivably encompass simultaneous management to multiple criteria (e.g. performance and availability), as long as the rewards pertaining to each criterion are equivalently scaled.

In future work we plan further study of the scalability of hybrid RL/function approximation to larger state spaces. We will also study whether further performance improvements can be obtained via multiple iterations of hybrid RL. Our testbed may also provide an interesting test of novel theoretical research on learning good policies without explicit exploration [10], and on exploration-exploitation tradeoffs.

References

1. Das, R., Tesauro, G., Walsh, W.E.: Model-based and model-free approaches to autonomic resource allcation. Technical Report RC23802, IBM Research (2005)
2. Tesauro, G.: Online resource allocation using decompositional reinforcement learning. In: Proc. of AAAI-05. (2005)
3. Vengerov, D., Iakovlev, N.: A reinforcement learning framework for dynamic resource allocation: First results. In: Proc. of ICAC-05. (2005)
4. Price, B., Boutilier, C.: Accelerating reinforcement learning through implicit imitation. J. of AI Research **19** (2003) 569–629
5. Lavenberg, S.S.: Personal communication (2006)
6. Tesauro, G., Jong, N.K., Das, R., Bennani, M.N.: A hybrid reinforcement learning approach to autnomic resource allocation. In: Proc. of ICAC-06. (2006) 65–73
7. Squillante, M.S., Yao, D.D., Zhang, L.: Internet traffic: Periodicity, tail behavior and performance implications. In Gelenbe, E., ed.: System Performance Evaluation: Methodologies and Applications, CRC Press (1999)
8. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
9. Baird, L.: Residual algorithms: Reinforcement learning with function approximation. In: Proc. of ICML-95. (1995)
10. Abbeel, P., Ng, A.Y.: Exploration and apprenticeship learning in reinforcement learning. In: Proc. of ICML-05. (2005)

Diversified SVM Ensembles for Large Data Sets

Ivor W. Tsang[1], Andras Kocsor[2], and James T. Kwok[1]

[1] Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
{ivor, jamesk}@cse.ust.hk
[2] Research Group on Artificial Intelligence
Hungarian Academy of Sciences and University of Szeged
H-6720 Szeged, Aradi vrt. 1., Hungary
kocsor@inf.u-szeged.hu

Abstract. Recently, the core vector machine (CVM) has shown significant speedups on classification and regression problems with massive data sets. Its performance is also almost as accurate as other state-of-the-art SVM implementations. By incorporating the orthogonality constraints to diversify the CVM ensembles, this turns out to speed up the maximum margin discriminant analysis (MMDA) algorithm. Extensive comparisons with the MMDA ensemble along with bagging on a number of large data sets show that the proposed diversified CVM ensemble can improve classification performance, and is also faster than the original MMDA algorithm by more than an order of magnitude.

1 Introduction

Support vector machines (SVMs) have been highly successful in many machine learning problems. Recently, the core vector machines (CVM) [1] is proposed for scaling up SVM. The main idea is to formulate the learning problem as a minimum enclosing ball (MEB) problem, and then apply an $(1+\epsilon)$-approximation algorithm. It has a provably asymptotic time complexity that is *linear* in m and a space complexity that is *independent* of m. Experiments on large classification [1] and regression [2] data sets demonstrate that the CVM is much faster and can handle much larger data sets than existing scale-up methods.

However, while a single SVM is often good in most cases, it is not always perfect. In particular, when there are many noisy patterns, they may corrupt the optimal decision boundary of a single SVM hyperplane. To address this problem, several ensemble methods, such as bagging, boosting and nonlinear ensemble approaches [3,4], have been proposed to improve SVM performance by combining multiple SVMs. However, these SVM ensemble methods require having many SVMs as base classifiers [4].

On the other hand, AdaBoost [5] has achieved good generalization performance by constructing weak classifier ensembles. The key idea is to update the probability distribution d_i's over the training set subject to the corrective constraint that the new distribution is orthogonal to the vector of the margin errors $-y_i f_t(\mathbf{x}_i)$. Consider the following weak classifier that is a variant of the Parzen window classifier, with the patterns weighted by d_i's: $f_t(\mathbf{x}) = \sum_{i=1}^m d_i^t y_i k(\mathbf{x}_i, \mathbf{x}) = \mathbf{w}_t' \varphi(\mathbf{x})$, where φ is the feature map associated with the kernel k, and $\mathbf{w}_t$ is the current weight vector. Then, the constraint for the new d_i^{t+1} distribution is $\sum_{i=1}^m d_i^{t+1} y_i f_t(\mathbf{x}_i) = 0$ or $\mathbf{w}_{t+1}' \mathbf{w}_t = 0$. This implies that the weight vector of the two consecutive weak classifiers are orthogonal. Moreover, Kivinen *et al.* [5] suggested finding the new distribution subject to the totally corrective constraints, i.e., the new distribution is orthogonal to the vectors of margin errors of all existing classifiers ($\mathbf{w}_{t+1}' \mathbf{w}_r = 0$ for $r = 1, \ldots, t$). Thus, usually only a few weak classifiers are required in constructing an ensemble with good classification performance.

The diversity of the base classifiers can improve the performance of the ensembles [4,6]. Intuitively, the orthogonality constraints can also be exploited to diversify the base SVM classifiers. By adding orthogonality constraints to the CVM ensemble, we will show in this paper that this can be seen as integrating maximum margin discriminant analysis MMDA [7] with the CVM. However, in order to apply the CVM algorithm, the QP problem corresponding to the kernel method of interest has to take a particular form. This, however, is not met by the MMDA, as the original CVM does not allow orthogonality constraints on the weight vectors. Thus, we propose an extension of the MEB problem by placing orthogonality constraints on the center of the MEB. We can then obtain orthogonal CVM ensembles on large data sets efficiently.

The rest of this paper is organized as follows. Section 2 first reviews MMDA. Section 3 then describes the proposed extension of the MEB problem, the modified CVM algorithm, and other variants of MMDA. Experimental results are presented in Section 4, followed by some concluding remarks in the last section.

2 Maximum Margin Discriminant Analysis (MMDA)

Given a training set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \pm 1$. Consider the following variant of the Lagrangian SVM [8], where the weight $\mathbf{w}$ is orthogonal to $\mathbf{u}_q = \mathbf{w}_q / \|\mathbf{w}_q\|$ for $q = 1, \ldots, s$:

$$\min \|\mathbf{w}\|^2 + b^2 + C \sum_{i=1}^m \xi_i^2 \quad : \quad y_i(\mathbf{w}'\varphi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \ \ \mathbf{u}_q' \mathbf{w} = 0. \qquad (1)$$

Here, φ is the nonlinear feature map associated with kernel k, ξ_i's are slack variables and C is a regularization parameter. Introducing Lagrangian multipliers $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]'$ and $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_s]'$ for the inequality and equality constraints, we obtain the dual:

$$\max 2\boldsymbol{\alpha}'\mathbf{1} - \boldsymbol{\alpha}'\hat{\mathbf{K}}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}'\mathbf{Y}\boldsymbol{\Phi}'\mathbf{U}\boldsymbol{\gamma} - \boldsymbol{\gamma}'\mathbf{U}'\mathbf{U}\boldsymbol{\gamma} \quad : \quad \boldsymbol{\alpha} \geq \mathbf{0}, \qquad (2)$$

where $\mathbf{0}, \mathbf{1} \in \mathbb{R}^m$ are vectors of zeros and ones, $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_s]$, $\mathbf{K} = \boldsymbol{\Phi}'\boldsymbol{\Phi}$ (where $\boldsymbol{\Phi} = [\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_m)]$) is the kernel matrix, $\mathbf{Y} = \text{diag}(y_1, \ldots, y_m)$, and

$$\hat{\mathbf{K}} = \mathbf{Y}\left(\mathbf{K} + \mathbf{1}\mathbf{1}' + \mathbf{I}/C\right)\mathbf{Y}, \tag{3}$$

is the transformed "kernel" matrix. By using the Karush-Kuhn-Tucker (KKT) conditions, the primal variables $\mathbf{w}, b$ can be recovered from the optimal $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$, and $\mathbf{u}_q$. Using (1), MMDA then extracts the weights ($\mathbf{w}$'s) one by one, and each of these can be expressed as a linear combination of $\varphi(\mathbf{x}_i)$'s. Note, however, that this MMDA formulation does not fit the existing MEB models in [1,2].

3 Core Vector Machine Ensembles

3.1 MEB with Multiple Projection Constraints on the Center

The center-constrained MEB problem in [2] constrains the center $\mathbf{c}$ to lie on the hyperplane $[\mathbf{0}'\ 1]\mathbf{c} = 0$. Here, we instead confine $\mathbf{c}$ to lie on multiple hyperplanes defined by $\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2, \ldots, \tilde{\mathbf{u}}_s$:

$$\min R^2 \quad : \quad \|\mathbf{c} - \tilde{\varphi}(\mathbf{x}_i)\|^2 \leq R^2, \ \ \tilde{\mathbf{u}}_q'\mathbf{c} = v_q. \tag{4}$$

Introducing Lagrangian multipliers $\tilde{\boldsymbol{\alpha}} = [\tilde{\alpha}_1, \ldots, \tilde{\alpha}_m]'$ and $\tilde{\boldsymbol{\gamma}} = [\tilde{\gamma}_1, \ldots, \tilde{\gamma}_s]'$ for the inequality and equality constraints, we obtain the dual:

$$\max \tilde{\boldsymbol{\alpha}}'\text{diag}(\tilde{\mathbf{K}}) + \tilde{\boldsymbol{\gamma}}'\mathbf{v} - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{\gamma}}'\tilde{\mathbf{U}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} \quad : \quad \tilde{\boldsymbol{\alpha}} \geq \mathbf{0}, \ \ \tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1, \tag{5}$$

where $\mathbf{v} = [v_1, \ldots, v_s]'$, $\tilde{\mathbf{U}} = [\tilde{\mathbf{u}}_1, \ldots, \tilde{\mathbf{u}}_s]$, $\tilde{\mathbf{K}} = \tilde{\boldsymbol{\Phi}}'\tilde{\boldsymbol{\Phi}}$ and $\tilde{\boldsymbol{\Phi}} = [\tilde{\varphi}(\mathbf{x}_1), \ldots, \tilde{\varphi}(\mathbf{x}_m)]$. Assume that for any pattern $\mathbf{x}$, $\tilde{k}$ satisfies

$$\tilde{k}(\mathbf{x}, \mathbf{x}) = \tilde{\kappa}, \tag{6}$$

a constant. Using the constraint $\tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1$, we obtain $\tilde{\boldsymbol{\alpha}}'\text{diag}(\tilde{\mathbf{K}}) = \tilde{\kappa}$. Dropping this constant from the objective in (5), we obtain a simpler QP:

$$\max \tilde{\boldsymbol{\gamma}}'\mathbf{v} - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{\gamma}}'\tilde{\mathbf{U}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} \quad : \quad \tilde{\boldsymbol{\alpha}} \geq \mathbf{0}, \ \ \tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1. \tag{7}$$

The radius $R = \sqrt{\tilde{\boldsymbol{\alpha}}'\text{diag}(\tilde{\mathbf{K}}) + \tilde{\boldsymbol{\gamma}}'\mathbf{v} - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{\gamma}}'\tilde{\mathbf{U}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}}}$ and the center $\mathbf{c} = \sum_{i=1}^m \tilde{\alpha}_i\tilde{\varphi}(\mathbf{x}_i) + \sum_{q=1}^s \tilde{\gamma}_q\tilde{\mathbf{u}}_q$ are recovered from the optimal $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\gamma}}$. Conversely, any QP in the form of (7) can be regarded as a MEB problem.

Once we have a MEB problem, one can apply the core-set approximation and probabilistic speedup techniques in CVM [1,2] to obtain an approximate solution of the MEB problem efficiently. The CVM procedure can be easily adapted to cater for this center $\mathbf{c}$. Each iteration then becomes the solving of the subproblem MEB($\mathcal{S}_t$) defined on the core-set $\mathcal{S}_t$.

Notice that finding MEB($\mathcal{S}_t$) still involves a QP. Instead of solving a QP with the equality constraint in (7), we follow the trick in [9] and remove the constraints by introducing Lagrangian multipliers $\tilde{\mu}_i$'s (where $\tilde{\mu}_i \geq 0$) for the nonnegative constraints $\tilde{\alpha}_i \geq 0$ and β for the equality constraint $\tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1$ in (7). Then the Lagrangian becomes $\tilde{L}(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\gamma}}, \tilde{\boldsymbol{\mu}}, \beta) = \tilde{\boldsymbol{\gamma}}'\mathbf{v} - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{\gamma}}'\tilde{\mathbf{U}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} + 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\mu}} + 2\beta(\tilde{\boldsymbol{\alpha}}'\mathbf{1} - 1)$, where $\tilde{\boldsymbol{\mu}} = [\tilde{\mu}_1, \ldots, \tilde{\mu}_m]'$. We set its derivatives w.r.t. $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\gamma}}$ to zero. Since $\tilde{\mathbf{K}} \succeq 0$ is pd and $\tilde{\mathbf{u}}_q$'s are independent, $\tilde{\mathbf{U}}'\tilde{\mathbf{U}} \succ 0$, and so $\tilde{\mathbf{G}} = \begin{bmatrix} \tilde{\mathbf{K}} & \tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}} \\ \tilde{\mathbf{U}}'\tilde{\boldsymbol{\Phi}} & \tilde{\mathbf{U}}'\tilde{\mathbf{U}} \end{bmatrix} \succ 0$. Hence, the optimal solution is:

$$[\tilde{\boldsymbol{\alpha}}' \ \tilde{\boldsymbol{\gamma}}']' = \tilde{\mathbf{G}}^{-1}[(\beta\mathbf{1} + \tilde{\boldsymbol{\mu}})' \ \mathbf{v}']', \tag{8}$$

where $\tilde{\boldsymbol{\mu}}$ and β are such that $\tilde{\boldsymbol{\alpha}} \geq \mathbf{0}, \tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1, \tilde{\boldsymbol{\alpha}} \odot \tilde{\boldsymbol{\mu}} = \mathbf{0}$ and $\tilde{\boldsymbol{\mu}} \geq \mathbf{0}$ (here, $\tilde{\boldsymbol{\alpha}} \odot \tilde{\boldsymbol{\mu}}$ is the elementwise product of $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\mu}}$).

3.2 Connection to MMDA

We now return to the QP problem associated with MMDA in (2). Introduce Lagrangian multipliers $\mu_i \geq 0$'s for the nonnegative constraints $\alpha_i \geq 0$ in (2), then the Lagrangian is $L(\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\mu}) = 2\boldsymbol{\alpha}'\mathbf{1} - \boldsymbol{\alpha}'\hat{\mathbf{K}}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}'\mathbf{Y}\boldsymbol{\Phi}'\mathbf{U}\boldsymbol{\gamma} - \boldsymbol{\gamma}'\mathbf{U}'\mathbf{U}\boldsymbol{\gamma} + 2\boldsymbol{\alpha}'\boldsymbol{\mu}$, where $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_m]'$. Since $\mathbf{K} \succeq 0$, $\hat{\mathbf{K}}$ in (3) is pd and $\mathbf{u}_q$'s are independent, $\mathbf{U}'\mathbf{U} \succ 0$, and so $\mathbf{G} = \begin{bmatrix} \hat{\mathbf{K}} & \mathbf{Y}\boldsymbol{\Phi}'\mathbf{U} \\ \mathbf{U}'\boldsymbol{\Phi}\mathbf{Y} & \mathbf{U}'\mathbf{U} \end{bmatrix} \succ 0$. Analogous to (8), an optimal solution is obtained as:

$$[\boldsymbol{\alpha}' \ \boldsymbol{\gamma}']' = \mathbf{G}^{-1}[(\mathbf{1} + \boldsymbol{\mu})' \ \mathbf{0}']', \tag{9}$$

where $\boldsymbol{\mu} \geq \mathbf{0}$, $\boldsymbol{\alpha} \geq \mathbf{0}$ and $\boldsymbol{\alpha} \odot \boldsymbol{\mu} = \mathbf{0}$. Alternatively, the optimal values for $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ can be solved by using the trick in [8]: $\mathbf{0} \leq \mathbf{a} \perp \mathbf{b} \geq \mathbf{0} \Leftrightarrow \mathbf{a} = (\mathbf{a} - \tau\mathbf{b})_+$ for $\tau > 0$, then $\boldsymbol{\mu} = ((\hat{\mathbf{K}}\boldsymbol{\alpha} + \mathbf{Y}\boldsymbol{\Phi}'\mathbf{U}\boldsymbol{\gamma} - \mathbf{1}) - \tau\boldsymbol{\alpha})_+$ by choosing a learning rate $\tau = 1.9/C$ as suggested in [8] (here, $\mathbf{a} \perp \mathbf{b}$ means $\mathbf{a}$ and $\mathbf{b}$ are perpendicular).

When the kernel k satisfies (6), $\hat{k}$ for the kernel matrix in (3) also satisfies (6), as $\hat{k}(\mathbf{x}, \mathbf{x}) = k(\mathbf{x}, \mathbf{x}) + 1 + 1/C$ is a constant for any $\mathbf{x}$. We set $v_q = 0$, $\tilde{\mathbf{U}} = [\mathbf{U}' \ \mathbf{0}']'$ (where $\mathbf{0}$ is the $s \times (m+1)$ zero matrix), and $\tilde{\varphi}(\mathbf{z}_i) = [y_i\varphi(\mathbf{x}_i)', y_i, y_i/\sqrt{C}\mathbf{e}_i']'$ (where $\mathbf{e}_i$ is the m-dimensional vector which has all zeros except that the ith entry is equal to one). Then $\tilde{\mathbf{K}} = \tilde{\boldsymbol{\Phi}}'\tilde{\boldsymbol{\Phi}} = [\tilde{k}(\mathbf{z}_i, \mathbf{z}_j)]$ with $\tilde{k}(\mathbf{z}_i, \mathbf{z}_j) = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \delta_{ij} y_i y_j / C$, $\tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}} = \mathbf{Y}\boldsymbol{\Phi}'\mathbf{U}$ and $\tilde{\mathbf{U}}'\tilde{\mathbf{U}} = \mathbf{U}'\mathbf{U}$. Multiplying $[\mathbf{1}' \ \mathbf{0}']$ on both sides of (8) and (9): $\tilde{\boldsymbol{\alpha}}'\mathbf{1} - \mathbf{1}'\mathbf{H}\tilde{\boldsymbol{\mu}} = \beta\mathbf{1}'\mathbf{H}\mathbf{1} = \beta(\boldsymbol{\alpha}'\mathbf{1} - \mathbf{1}'\mathbf{H}\boldsymbol{\mu})$, where $\mathbf{H}$ is the left top $m \times m$ submatrix of $\tilde{\mathbf{G}}^{-1}$. Using $\tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1$, and assuming that $\boldsymbol{\alpha}'\mathbf{1} > 0$, we have

$$\beta = \frac{1 - \mathbf{1}'\mathbf{H}\tilde{\boldsymbol{\mu}}}{\boldsymbol{\alpha}'\mathbf{1} - \mathbf{1}'\mathbf{H}\boldsymbol{\mu}} = \frac{1}{\boldsymbol{\alpha}'\mathbf{1}} \frac{\boldsymbol{\alpha}'\mathbf{1} - \mathbf{1}'\mathbf{H}\tilde{\boldsymbol{\mu}}\boldsymbol{\alpha}'\mathbf{1}}{\boldsymbol{\alpha}'\mathbf{1} - \mathbf{1}'\mathbf{H}\boldsymbol{\mu}} = \frac{1}{\boldsymbol{\alpha}'\mathbf{1}}, \tag{10}$$

where $\tilde{\boldsymbol{\mu}} = \frac{\boldsymbol{\mu}}{\boldsymbol{\alpha}'\mathbf{1}} \geq \mathbf{0}$. Furthermore, from (8), (9) and (10), we obtain

$$[\tilde{\boldsymbol{\alpha}}' \ \tilde{\boldsymbol{\gamma}}']' = \beta\tilde{\mathbf{G}}^{-1}[(\mathbf{1} + \boldsymbol{\mu})' \ \mathbf{0}']' = [\boldsymbol{\alpha}' \ \boldsymbol{\gamma}']'/\boldsymbol{\alpha}'\mathbf{1} \tag{11}$$

such that $\tilde{\boldsymbol{\alpha}}'\mathbf{1} = \frac{\boldsymbol{\alpha}'\mathbf{1}}{\boldsymbol{\alpha}'\mathbf{1}} = 1$, $\tilde{\boldsymbol{\alpha}} \odot \tilde{\boldsymbol{\mu}} = \frac{\boldsymbol{\alpha}}{\boldsymbol{\alpha}'\mathbf{1}} \odot \frac{\boldsymbol{\mu}}{\boldsymbol{\alpha}'\mathbf{1}} = \mathbf{0}$, and $\tilde{\boldsymbol{\alpha}} \geq \mathbf{0}$. Hence, using (11), the solutions of $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\gamma}}$ in (5) can be recovered from the optimal values for $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ in (2). In other words, the optimization problem associated with MMDA in (1) can now be viewed as a constrained MEB problem in (4), with $\tilde{\varphi}$ being replaced by the new feature map $\hat{\varphi}$ and the associated kernel $\hat{k}$ satisfying (6).

3.3 Other Variants of MMDA

Other variants of MMDA that generate a non-orthogonal basis where the data is uncorrelated (but do not use the orthogonality constraints) can also use this new

MEB model. As discussed in [10], the uncorrelated constraints consider the relationship between patterns, and minimize redundancy among the weight vectors in the reduced space. We can replace the orthogonality constraints on $\mathbf{w}$ in (1) by uncorrelated constraints, and the primal becomes: $\min \|\mathbf{w}\|^2 + b^2 + C\sum_{i=1}^{m} \xi_i^2 \ : \ y_i(\mathbf{w}'\varphi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \ \hat{\mathbf{u}}_q'\mathbf{w} = \mathbf{u}_q'\boldsymbol{\Phi}\boldsymbol{\Phi}'\mathbf{w} = 0$. The corresponding dual is $\max 2\boldsymbol{\alpha}'\mathbf{1} - \boldsymbol{\alpha}'\hat{\mathbf{K}}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}'\mathbf{Y}\boldsymbol{\Phi}'\hat{\mathbf{U}}\boldsymbol{\gamma} - \boldsymbol{\gamma}'\hat{\mathbf{U}}'\hat{\mathbf{U}}\boldsymbol{\gamma} \ : \ \boldsymbol{\alpha} \geq \mathbf{0}$, where $\hat{\mathbf{U}} = [\hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_s]$. Using the same construction as in Section 3.2, this is also a MEB problem with multiple projection constraints on the center.

4 Experiments

4.1 Experimental Setup

Experiments are performed on a number of real-world data sets[1] (Table 1). All the different base classifier variants are run N_c times using the one-vs-all scheme (where N_c is the number of classes). The following base classifiers are compared: 1) Orthogonal SVM: SVM with orthogonality constraints with all previous SVM classifiers. This is the same as MMDA; 2) Orthogonal CVM: the proposed ensemble; 3) Bagged SVM (the base SVMs are trained by LIBSVM[2]).

As suggested in [3], a double-layer hierarchical combination scheme using nonlinear classifiers can have improved performance. In this experiment, we combine the base SVMs by the following classifiers: 1) SVM; 2) artificial neural network (ANN), with a single layer of 10 hidden units; 3) CVM; 4) Majority voting [3]. To demonstrate the usefulness of the extra orthogonal SVMs, we also compare with the standard SVM and ANN classifiers. The C parameter in (1) is always fixed at 1. We use the Gaussian kernel $\exp(-\|\mathbf{x}-\mathbf{z}\|^2/\beta)$, where $\beta = \frac{1}{m^2}\sum_{i,j=1}^{m} \|\mathbf{x}_i - \mathbf{x}_j\|^2$ is the average squared distance between patterns. Experiments are implemented in MATLAB (except for the bagged SVM which is in C++) and are performed on an AMD Athlon 4400+ PC with 4GB of RAM.

Table 1. Data sets used in the experiments

	optdigits	satimage	pendigits	letters	mnist	usps	face
# classes	10	6	10	26	10	2	2
# attributes	64	36	16	16	780	676	361
# training patterns	3,823	4,435	7,494	16,000	60,000	266,079	346,260
# testing patterns	1,797	2,000	3,498	4,000	10,000	75,383	24,045

The performance of ensemble methods depend critically on the number of base SVMs used, so we first perform some preliminary experiments on this. Figure 1 shows the results on the smaller data sets using the ANN as the final

[1] The first five data sets are from the UCI machine learning repository, while the last two are from http://www.cs.ust.hk/~ivor/cvm.html.

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

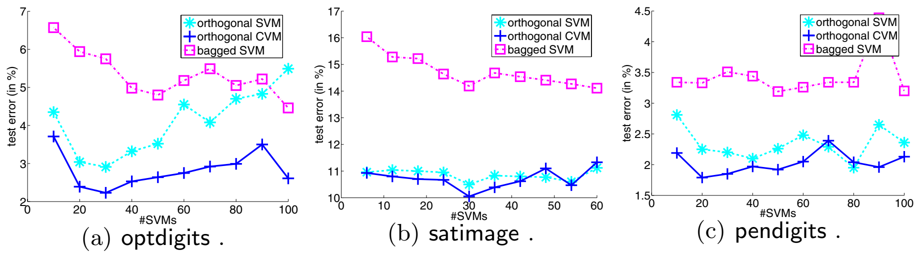


Fig. 1. Testing error of the different SVM ensembles vs #SVMs

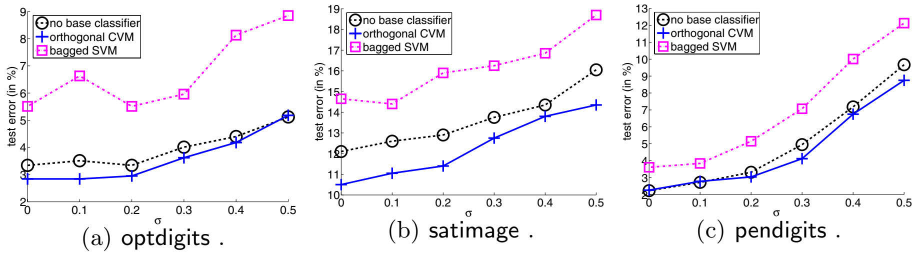


Fig. 2. Testing error of the different SVM ensembles at different noise levels

classifier. We observe that the performance of the bagged SVM first improves as more base SVMs are used, and then becomes more stable or even degraded. The performance using both the orthogonal CVM and SVM ensemble are better than the others when there are around $3N_c$ to $5N_c$ base SVMs. So, in the sequel, $N_c/3N_c/5N_c$ base SVMs are used.

4.2 Experimental Results

First, we show the proposed orthogonal CVM ensemble is more robust than the single SVM classifier and bagged SVMs. We run the orthogonal CVM ensemble and bagged SVM on the first three small data sets in Table 1. The input features are corrupted by zero-mean Gaussian noise at different noise levels (σ). For simplicity, we fix the number of base SVMs at $5N_c$, and the final classifier is a SVM. From Figure 2, we observe that the orthogonal CVM ensemble is more resistant to noise than the single SVM classifier and bagged SVMs.

As can be seen from Table 2, SVM ensembles can improve classification performance. In particular, nonlinear ensemble schemes using orthogonal SVMs outperform a single SVM. Moreover, the orthogonality constraints used in both the SVM and CVM base classifiers lead to lower testing errors than the bagged SVMs when using a few ($3N_c - 5N_c$) base SVMs.

As mentioned in Section 2, each base SVM can be expressed as a linear combination of kernel evaluations. Figure 3 shows the number of kernel evaluations involved in each base SVM. As can be seen, the CVM implementation produces SVMs that are sparser than the original one. As kernel evaluations are relatively

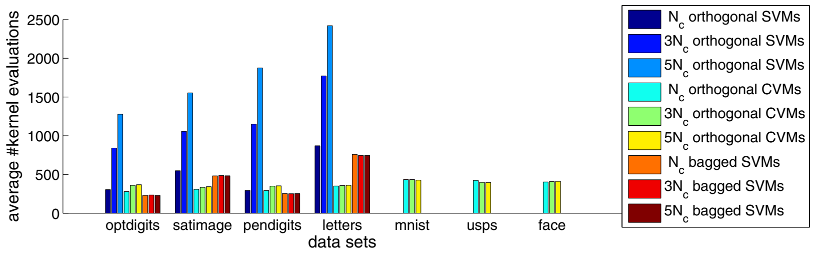


Fig. 3. Average number of kernel evaluations involved in each base SVM

Table 2. Testing errors on the various data sets

base classifier		final classifier	optdigits	satimage	pendigits	letters	mnist	usps	face
no base classifier		SVM	3.34	10.4	3.26	9.05	4.88	–	–
		ANN	5.63	12.6	4.81	29.05	9.61	0.88	2.6
orthogonal SVM	#base=N_c	SVM	3.07	10.55	2.18	7.15	–	–	–
	$3N_c$		3.07	11.45	2.15	6.05	–	–	–
	$5N_c$		3.07	10.35	2.09	5.98	–	–	–
	#base=N_c	ANN	4.35	11.05	2.81	19.8	–	–	–
	$3N_c$		2.91	10.35	2.2	17.98	–	–	–
	$5N_c$		3.52	11.3	2.26	17.35	–	–	–
	#base=N_c	CVM	3.07	10.55	2.18	7.1	–	–	–
	$3N_c$		3.07	10.45	2.12	6.05	–	–	–
	$5N_c$		3.07	10.35	2.09	5.78	–	–	–
orthogonal CVM	#base=N_c	SVM	2.84	10.25	2.29	**5.15**	5.46	–	–
	$3N_c$		2.9	10.5	2.26	5.4	4.27	–	–
	$5N_c$		2.95	10.7	2.21	5.65	**4.08**	–	–
	#base=N_c	ANN	3.73	10.25	2.78	19.03	7.01	0.7	1.72
	$3N_c$		**2.23**	10.7	2.15	17.55	6.72	0.67	**1.61**
	$5N_c$		2.64	**10.05**	**1.92**	17.33	6.66	**0.66**	1.66
	#base=N_c	CVM	2.84	10.25	2.29	5.18	5.46	0.69	1.9
	$3N_c$		2.84	10.5	2.26	5.43	4.28	0.67	1.66
	$5N_c$		2.95	10.7	2.21	5.7	4.09	0.7	1.65
bagged SVM	#base=N_c	SVM	6.35	16.0	3.98	29.6	–	–	–
	$3N_c$		6.07	14.8	3.61	25.8	–	–	–
	$5N_c$		5.63	14.85	3.72	24.83	–	–	–
	#base=N_c	ANN	6.57	16.04	3.34	30.22	–	–	–
	$3N_c$		5.75	15.22	3.51	26.36	–	–	–
	$5N_c$		4.8	15.19	3.19	25.21	–	–	–
	#base=N_c	CVM	6.4	15.75	3.66	29.7	–	–	–
	$3N_c$		5.79	14.6	3.44	25.95	–	–	–
	$5N_c$		5.51	14.65	4.61	25.25	–	–	–
	#base=N_c	voting	7.8	20.5	4.18	32.2	–	–	–
	$3N_c$		6.52	19.3	3.75	28.15	–	–	–
	$5N_c$		5.29	18.45	3.35	26.8	–	–	–

Table 3. CPU time (in seconds) required in the ensemble learning of base SVMs

base classifier	optdigits	satimage	pendigits	letters	mnist	usps	face
orthogonal SVM #base=N_c	84	121	127	1,911	–	–	–
$3N_c$	476	421	570	9,646	–	–	–
$5N_c$	1,495	900	1,674	20,860	–	–	–
orthogonal CVM #base=N_c	41	23	20	92	1,610	2,359	105
$3N_c$	181	78	95	301	4,928	6,585	337
$5N_c$	332	136	174	512	8,179	10,630	556

expensive, the orthogonal CVM is generally faster than the original implementation during testing.

Table 3 lists the CPU time needed in the ensemble learning of base SVMs. As can be seen, the proposed method is often faster than the original MMDA by one to two orders of magnitude. In particular, note that the bagged SVM and orthogonal SVM ensembles cannot finish training on the three largest data sets in 24 hours (indicated by "-" in the tables), while the proposed method obtain ensembles for the final classifier in usually less than several thousand seconds.

5 Conclusions

In this paper, we investigate ensemble learning in large scale classification tasks. The use of orthogonality constraints in the SVM ensemble leads to more robust performance than bagging. Moreover, the training time complexity depends only linearly on the training set size. In practice, it is 10-100 times faster than the original SVM ensemble. The proposed method produces sparser base SVMs and with better performance. It also involves fewer kernel evaluations. This in turn allows the combined classifier to be computed much faster during testing. In the future, we will investigate other different constraints on the SVM ensemble.

References

1. Tsang, I.W., Kwok, J.T., Cheung, P.M.: Core vector machines: Fast SVM training on very large data sets. Journal of Machine Learning Research **6** (2005) 363–392
2. Tsang, I.W., Kwok, J.T., Lai, K.T.: Core vector regression for very large regression problems. In: Proceedings of the Twentieth-Second International Conference on Machine Learning, Bonn, Germany (2005) 913–920
3. Kim, H.C., Pang, S., Je, H.M., Kim, D., Bang, S.: Constructing support vector machine ensemble. Pattern Recognition **36** (2003) 2757–2767
4. Valentini, G., Dietterich, T.: Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods. Journal of Machine Learning Research **5** (2004) 725–775
5. Kivinen, J., Warmuth, M.K.: Boosting as entropy projection. In: Proceedings of the twelfth annual conference on Computational learning theory, Santa Cruz, California, United States (1999) 134 – 144

6. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning **51** (2003) 181–207
7. Kocsor, A., Kovács, K., Szepesvári, C.: Margin maximizing discriminant analysis. In: Proceedings of the 15th European Conference on Machine Learning, Pisa, Italy (2004) 227–238
8. Mangasarian, O., Musicant, D.: Lagrangian support vector machines. Journal of Machine Learning Research **1** (2001) 161–177
9. Kienzle, W., Schölkopf, B.: Training support vector machines with multiple equality constraints. In: Proceedings of the European Conference on Machine Learning. (2005)
10. Ye, J., Li, T., Xiong, T., Janardan, R.: Using uncorrelated discriminant analysis for tissue classification with gene expression d. IEEE/ACM Transactions on Computational Biology and Bioinformatics **1** (2004) 181–190

Dynamic Integration with Random Forests

Alexey Tsymbal[1], Mykola Pechenizkiy[2], and Pádraig Cunningham[1]

[1] Dept of Computer Science, Trinity College Dublin, Ireland
{Alexey.Tsymbal, Padraig.Cunningham}@cs.tcd.ie
[2] Dept of Math IT, University of Jyväskylä, Finland
mpechen@it.jyu.fi

Abstract. Random Forests (RF) are a successful ensemble prediction technique that uses majority voting or averaging as a combination function. However, it is clear that each tree in a random forest may have a different contribution in processing a certain instance. In this paper, we demonstrate that the prediction performance of RF may still be improved in some domains by replacing the combination function with dynamic integration, which is based on local performance estimates. Our experiments also demonstrate that the RF Intrinsic Similarity is better than the commonly used Heterogeneous Euclidean/Overlap Metric in finding a neighbourhood for local estimates in the context of dynamic integration of classification random forests.

1 Introduction

Random Forests (RF) are a relatively young (they were introduced in 2001), but effective and popular ensemble technique [5]. RF were demonstrated to compare favourably with boosting in terms of predictive performance and to be more robust with respect to overfitting noisy instances in various classification and regression domains.

In the standard RF algorithm [5] simple majority voting or averaging are used to combine the base predictions. A natural possible extension to RF is to improve the combination of trees by taking into account their local performance. One such combination technique, which could be used here, is dynamic integration (DI) [12]. In DI, local performance is estimated for each base model based on the performance on similar instances, and then this is used to calculate a corresponding weight for combining predictions with locally weighted voting, or to simply select a model with the best local performance. RF provide us with an Intrinsic Similarity metric (RFIS), which could be used in DI. The proportion of the base trees where two instances appear together in the same leaves can be used as a measure of similarity between the instances [5]. In this paper we evaluate the two alternative combination functions. We find that DI does improve the performance of RF. We also find that RFIS is very effective; this is not surprising as it is *in tune* with the dynamics of the ensemble.

This paper is organized as follows: in Section 2 we review RF, in Section 3 we consider how they can be augmented with DI, in Section 4 we present the results of our experiments, and in Section 5 we conclude with a brief summary.

2 Random Forests

Breiman in his paper [5] demonstrated that optimal ensemble performance could be achieved by injecting randomness in order to minimize correlation between base models while maintaining their accuracy. In RF this is achieved by combining two sources of randomness. First, instances used to grow each tree are sampled randomly without replacement from the original training set. Second, RF randomly select features at each node to grow each tree [5]. Using the Strong Law of Large Numbers, Breiman demonstrated that RF always converge so that overfitting is not a problem, that is RF never overfit as more trees are added.

RF have a set of desirable properties [5]:

(1) their predictive performance is as good as boosting and sometimes better;
(2) they are relatively robust to outliers and noise;
(3) they are faster than many other ensembles, bagging and boosting in particular;
(4) due to the use of bootstrapping, they give useful internal (so-called *out-of-bag*) estimates of error, strength (margin), correlation and feature importance;
(5) they are simple and easily parallelized.

RF were demonstrated to produce error rates not far above the Bayes rate in different application domains [5]. However, in some domains their accuracy can still be improved. For example, Robnik-Šikonja in [7] considered two ways of improving RF: (1) a combination of several feature selection criteria in order to reduce correlation in the forests, and (2) replacement of majority voting with locally weighted voting.

3 Improving Random Forests with Dynamic Integration

A number of *selection* and *combination* approaches to ensemble integration have been proposed [6, 8, 9, 12]. The most popular *combination* technique, also used in RF, is simple majority voting [1]. Weighted Voting (WV), where each vote has a weight proportional to the estimated generalization performance of the corresponding classifier, usually has better predictive performance [1].

A number of *selection* techniques have also been proposed to address the task of integration. One of the most popular and simplest selection techniques is Cross-Validation Majority (CVM), where the classifier with the highest cross-validation accuracy is selected [9].

The approaches described above are *static*. They select one model or combine the models uniformly. In *dynamic* integration information about each new instance is taken into account [7, 10, 12]. Three DI techniques based on the same local performance estimates; Dynamic Selection (DS), Dynamic Voting (DV), and Dynamic Voting with Selection (DVS), were considered in [12]. First, the errors of each base classifier on each instance of the training set are estimated using cross validation. This demands $O(Mn)$ additional space for saving information about the errors of M base classifiers on n training instances. The application phase begins with determining k-nearest neighbours for a new instance. After that, weighted nearest neighbour learning is used to predict the local performance of each base classifier.

Then, DS simply selects a classifier with the least local error. In DV, each base classifier receives a weight that is proportional to its estimated local performance. In DVS, the base classifiers with the errors that fall into the upper half of the error interval are discarded and DV is applied to the remaining set of classifiers.

DI was successfully applied in a number of contexts, outperforming other integration methods. In [12] DS, DV and DVS were used with bagging and boosting. In [10] DI was applied to ensembles of base classifiers generated on different feature subsets. In [8] an adaptation of the DI techniques to regression was considered.

RF have the very appealing property that each tree is built on a bootstrap replicate. The remaining (out-of-bag) instances are useful for the evaluation of the base trees' accuracy, margin, correlation, and even feature importance [1]. This property can be used in DI as well. With out-of-bag instances there is no need for cross validation or for a separate validation set.

Different distance functions can be used in DI. The simplest and most common way is to use the Euclidean distance with numeric features, and the overlap distance with categorical features, as in the heterogeneous Euclidean/overlap metric (HEOM) [13]. HEOM was demonstrated to be robust and difficult to compete with in many domains [13]. However, RF provide us with RFIS, which could be used in DI as well. The proportion of the trees where two instances appear together in the same leaves can be used as a measure of similarity between them [5]. It is important to note that two instances that are close together in the HEOM space might have relatively small RFIS if they are near the classification boundary. RFIS demands $O(nK)$ additional space for saving information about n training instances in the leaves of the K trees.

In order to calculate the weight in model i in DI for a new instance $\mathbf{x}$, we use:

$$w_i(\mathbf{x}) = \sum_{j=1}^{k}\left(I_{OOB_i}(\mathbf{x}_j)\cdot\sigma(\mathbf{x},\mathbf{x}_j)\cdot mr_i(\mathbf{x}_j)\right)\Big/\sum_{j=1}^{k}\left(I_{OOB_i}(\mathbf{x}_j)\cdot\sigma(\mathbf{x},\mathbf{x}_j)\right) \tag{1}$$

where k is the size of the neighbourhood, OOB_i is the set of out-of-bag instances for model i and $I()$ is an indicator function, $\sigma(\mathbf{x},\mathbf{x}_j)$ is a distance-based relevance coefficient, and $mr_i(\mathbf{x}_j)$ is the margin of model i on jth nearest neighbour of $\mathbf{x}$. Margin can be defined as follows for a classifier with crisp outputs:

$$mr_i(\mathbf{x}) = \begin{cases} 1, & h_i(\mathbf{x}) = y(\mathbf{x}) \\ -1, & h_i(\mathbf{x}) \neq y(\mathbf{x}) \end{cases} \tag{2}$$

In fact, weight (1) represents the expected margin of model i on instance $\mathbf{x}$. We normalize weights (1) to be non-negative and to sum to one in order to apply them in DI. In our experiments with the two distance metrics we use the inverse HEOM distance and the cube of RFIS as the corresponding distance-based weight coefficients:

$$\sigma_{HEOM}(\mathbf{x},\mathbf{x}_j) = 1/HEOM(\mathbf{x},\mathbf{x}_j) \tag{3}$$

$$\sigma_{RFIS}(\mathbf{x},\mathbf{x}_j) = RFIS^3(\mathbf{x},\mathbf{x}_j) \tag{4}$$

In our experiments we also consider a non-weighted variant of (1), demonstrating that the use of weights is usually superior for both of the distance metrics.

4 Experimental Studies

In our experiments we use an implementation based on the machine learning library WEKA 3.4.2 [14]. Information Gain is used as the splitting criterion, and the number of randomly selected features in each node is $\lfloor \log_2 M + 1 \rfloor$, where M is the number of features in the dataset.

In our experiments we use 27 benchmark datasets. 24 of these datasets are from the UCI ML repository [3]. The Parity2 and Parity3 datasets were considered in [7]. They have 2 and 3 binary parity features respectively and 10 random binary features. The Images dataset consists of 1000 image windows drawn from 2 monochrome images of natural scenes. These images were previously considered in [2]. We estimate accuracy and margin after 30 runs of hold-out cross validation with 70/30% train/test split of each dataset.

As was mentioned before, RF often give an error rate comparable to the Bayes rate. This is especially so for the relatively simple UCI datasets. Thus, it is no surprise that their accuracy is difficult to beat for any technique, including DI. In our experiments, on 12 of the 27 datasets there was a statistically significant accuracy improvement due to the use of DI. With the remaining datasets the difference in accuracy was insignificant. We continue the analysis of experimental results focusing on these 12 datasets.

The first surprising tendency we could observe was that the accuracy of DS was very poor. On most datasets DS significantly decreased the accuracy of RF with any local learning scheme. Only with 2 datasets was its accuracy better; MONK-2 and Parity2. These datasets represent artificial concepts "well suitable" for dynamic selection. Such a poor behaviour of DS is surprising, because much research in the area of ensembles is concentrated on (dynamic) classifier selection, and this is justified by its good performance in many application domains. However, in the context of RF, the base models are usually weak and diverse, which makes the task of classifier selection difficult. We continue the analysis of experimental results focusing on DV and DVS. They give close results, with DVS being a little better on average.

Naturally, accuracy with DI usually decreases with the increase in the size of neighbourhood, becoming closer to simple static majority voting. DI is not very sensitive to the size of neighbourhood. 15 and 31 instances give close results, both for the weighted and non-weighted cases, with 15 being a slightly better neighbourhood on average. We continue our analysis focusing on the size of neighbourhood equal to 15.

Now let us consider different local learning schemes for DI. In Fig. 1 the accuracy of plain RF with static voting (*SV*) is compared with RF with DVS for the 4 different local learning schemes; HEOM, equally-weighted (DVS_{HEOM}) and locally weighted (DVS_{HEOMW}), and RFIS, equally-weighted (DVS_{RF}) and locally weighted (DVS_{RFW}) for 4 different ensemble sizes (10, 25, 50 and 100), averaged over the 12 datasets.

This figure reveals a few interesting tendencies. First, it shows the average improvement due to DI, which is more than 1.5% with any local learning scheme for the ensembles with 100 trees. Second, all the schemes give close results. However, it can be seen that the locally weighed schemes out-perform their equally weighted counterparts, and RFIS results usually out-perform the corresponding HEOM results. An interesting result is that the locally weighted RFIS scheme clearly stands out on the figure. As we shall later see, this superiority will also be supported by tests for statistical significance and the analysis of classification margin for each dataset.

In Table 1 accuracy results are given for the ensembles of 100 trees for plain RF with static voting (*SV*) and for the four local learning schemes with DVS (*DVS_{HEOM}*, *DVS_{HEOMW}*, *DVS_{RF}* and *DVS_{RFW}*) for the 12 datasets. The table includes the dataset name, the minimum, average and maximum accuracy of ensemble members, and accuracies for the five integration strategies. Numbers given in bold represent the significant wins of corresponding DVS strategies over SV (according to the paired *t*-test with 0.95 level of significance).

This table demonstrates the fact that RF contain weak and highly diverse base classifiers. In many domains RF out-perform the best component decision tree (except the Glass, Zoo and Parity problems). Of the four local learning strategies, locally weighted RFIS (DVS_{RFW}) demonstrates the most robust behaviour with the best average accuracy and 9 wins (with 8 wins for DVS_{HEOM} and 7 wins for DVS_{RF} and

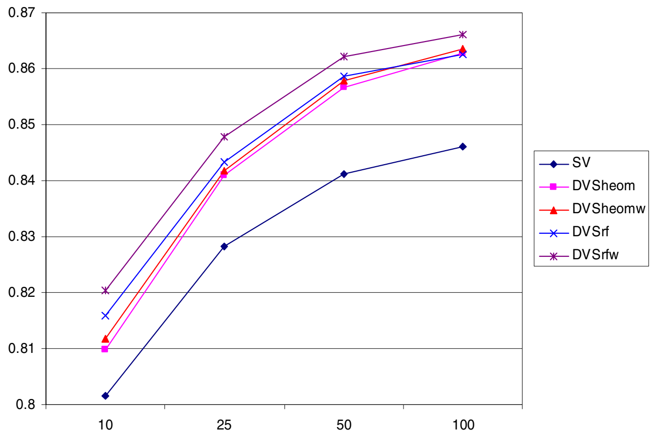


Fig. 1. Accuracy of plain and DVS RF for different local learning schemes and ensemble sizes

Table 1. Accuracy of plain RF and DVS RF for the four local learning schemes

Dataset	Min	Aver	Max	SV	DVS_{HEOM}	DVS_{HEOMW}	DVS_{RF}	DVS_{RFW}
Audiology	0.316	0.507	0.707	0.727	**0.741**	**0.740**	**0.739**	**0.739**
Car	0.755	0.830	0.888	0.935	**0.938**	**0.937**	0.936	0.937
DNAp	0.385	0.636	0.872	0.908	0.914	0.911	0.908	0.913
Glass	0.495	0.637	0.770	0.762	0.765	0.764	0.770	**0.772**
Images	0.566	0.639	0.708	0.85	**0.859**	**0.86**	**0.857**	**0.859**
MONK-1	0.624	0.824	0.989	0.997	**0.999**	**1.000**	**1.000**	**1.000**
Parity2	0.397	0.658	0.999	0.925	**0.973**	**0.974**	**0.978**	**0.977**
Parity3	0.350	0.543	0.860	0.639	**0.716**	**0.724**	**0.713**	**0.724**
Sonar	0.524	0.689	0.835	0.830	**0.841**	0.840	**0.840**	**0.844**
Tic-tac-toe	0.673	0.765	0.845	0.936	**0.960**	**0.961**	**0.961**	**0.966**
Vehicle	0.605	0.675	0.738	0.746	0.748	0.748	0.749	0.749
Zoo	0.709	0.844	0.959	0.898	0.898	0.904	0.899	**0.912**
Average	*0.533*	*0.687*	*0.848*	*0.846*	*0.863*	*0.864*	*0.863*	*0.866*

DVS_{HEOMW}). DI (DVS strategy) always gives similar or better accuracy than SV in these domains. The same situation holds true with DV and with the ensembles of other sizes (10, 25 and 50).

Besides the accuracy for the different integration techniques considered we also measured *classification margin* for SV, DV and DVS. The margin of a classifier h on instance $\mathbf{x}$ can be measured as the extent to which the average vote for the right class $y(\mathbf{x})$ exceeds the maximal average vote for any other class [5]. Average margin over the test instances represents an estimate of expected margin for the classification problem considered and is an important characteristic for any learning algorithm [5,7].

In Table 2 margin is given for plain RF and for the 4 local learning schemes with DVS. From this table one can see that DI always increases the margin of plain RF on these 12 datasets. Interestingly, this increase is always significant. Besides, DI often increased the margin even when the accuracy of DI remained the same with SV (on the rest of 27 datasets). This behaviour is not so surprising, as the notion of a diverse ensemble is somewhat at odds with the concept of a high margin, i.e. diversity can be achieved by *squeezing* the margin.

Interestingly, the margins of weighted schemes are always greater than the corresponding non-weighted margins, and the margins using RFIS are always greater than the corresponding HEOM margins. Another interesting and somewhat surprising tendency when one considers separate local learning schemes is that while all the other three schemes usually give pretty close results, the margin with locally weighted RFIS, DVS_{RFW}, usually clearly stands out and is always statistically significantly higher than all the other corresponding margins, supporting its relative superiority in accuracy shown in Table 1. The results in Table 2 clearly show the superiority of RFIS over HEOM in finding a neighbourhood for DI. The fact that the locally weighted RFIS always produces a statistically significantly greater margin, even though the corresponding accuracy may not always be significantly different in comparison with the other local learning schemes, demonstrates its greater strength in this context.

Table 2. Classification margin for plain RF and for the four local learning schemes with DVS

Dataset	SV	DVS_{HEOM}	DVS_{HEOMW}	DVS_{RF}	DVS_{RFW}
Audiology	0.255	0.291	0.302	0.296	0.314
Car	0.701	0.729	0.733	0.735	0.754
DNAp	0.267	0.298	0.302	0.310	0.322
Glass	0.370	0.386	0.394	0.392	0.411
Images	0.276	0.287	0.289	0.289	0.295
MONK-1	0.614	0.689	0.700	0.716	0.754
Parity2	0.315	0.434	0.454	0.449	0.484
Parity3	0.092	0.160	0.172	0.165	0.187
Sonar	0.377	0.397	0.406	0.406	0.420
Tic-tac-toe	0.513	0.571	0.575	0.582	0.608
Vehicle	0.418	0.426	0.427	0.429	0.434
Zoo	0.752	0.761	0.775	0.763	0.782
Average	*0.413*	*0.452*	*0.461*	*0.461*	*0.480*

In our experiments, we considered *two bias/variance decompositions*; those of Kohavi and Wolpert [6] and Breiman [4]. They closely capture the original squared loss definitions and have a behaviour that corresponds with intuition. Analysing the behaviour of DS, we could see that DS tries to reduce bias at the expense of the considerable increase in variance. The increase in variance was huge, and on some datasets bias was increased too. DV and DVS reduce error by reducing bias while trying to keep variance the same. DV and DVS, on these datasets, always decrease bias and this decrease is always significant. Sometimes this is accompanied by an insignificant increase in variance. Comparing DV and DVS, we could see that DVS, as a technique involving classifier selection, tries to further decrease bias. Interestingly, this is not always accompanied by an increase in variance, and on average the variance terms of DV and DVS are the same. More detailed experimental results for the present study including numbers for the bias/variance decomposition (which are not included here due to space limitations) are made available online as a technical report [11].

5 Conclusions

One way for improving RF is to replace majority voting with a more sophisticated combination function such as DI. Our experiments demonstrated that DI was able to improve the accuracy of RF on 12 out of 27 datasets.

More detailed experimental analysis revealed a few interesting tendencies. DV and DVS were demonstrated to always increase margin in comparison with the usual RF – a characteristic that is similar to that of boosting. Bias/variance analysis demonstrated that DV and DVS tended to decrease bias while keeping variance the same. DS was proven to be inappropriate in this context, always significantly increasing variance.

Among the distance functions and local learning schemes considered in DI, the best combination was RFIS with locally weighted learning. Interestingly, this combination usually resulted in a significantly greater margin than all the other techniques, even when accuracy remained the same. In general, the RFIS metric demonstrated very promising behaviour, and it is an interesting question for further research whether this superiority will hold true in other data mining tasks.

Acknowledgments. This material is based upon work supported by the Science Foundation Ireland under Grant S.F.I.-02/N.1/111 and by the Academy of Finland.

References

1. Bauer E., Kohavi R.: An empirical comparison of voting classification algorithms: bagging, boosting, and variants, *Machine Learning*, 36 (1,2) (1999) 105-139
2. Bingham E., Mannila H.: Random projection in dimensionality reduction: applications to image and text data. In: *Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining KDD 01*, ACM Press (2001) 245-250
3. Blake C.L., E. Keogh, C.J. Merz: UCI repository of machine learning databases [http:// www.ics.uci.edu/ ~mlearn/ MLRepository.html], Dept. of Information and Computer Science, University of California, Irvine, CA (1999)

4. Breiman L.: Bias, Variance, and Arcing Classifiers, Tech. Report 486, Statistics Dept., University of California, Berkeley, USA (1996)
5. Breiman L.: Random Forests, *Machine Learning*, 45(1) (2001) 5-32
6. Kohavi R., Wolpert D.: Bias plus variance decomposition for zero-one loss functions. In: *Proc. 13th Int. Conf. on Machine Learning*, Morgan Kaufmann (1996) 275-283
7. Robnik-Šikonja M.: Improving random forests. In: J.F. Boulicaut et al. (eds.), *Proc. 15th European Conf. on Machine Learning ECML 04*, Springer, LNCS 3201 (2004) 359-370
8. Rooney N., Patterson D., Anand S., Tsymbal A.: Dynamic integration of regression models. In: *Proc. 5th Int. Workshop on Multiple Classifier Systems MCS 04,* LNCS 3181, Springer (2004) 164-173
9. Schaffer C.: Selecting a classification method by cross-validation, *Machine Learning*, 13 (1993) 135-143
10. Tsymbal A., Pechenizkiy M., Cunningham P.: Sequential genetic search for ensemble feature selection. In: *Proc. 19th Int. Joint Conf. on Artificial Intelligence IJCAI 05*, Morgan Kaufmann (2005) 877-882
11. Tsymbal A., Pechenizkiy M., Cunningham P.: Dynamic integration with random forests. Tech Report TCD-CS-2006-23, Dept of Computer Science, Trinity College Dublin, Ireland (2006) (available online at http://www.cs.tcd.ie/publications/tech-reports/reports.06/)
12. Tsymbal A., Puuronen S.: Bagging and boosting with dynamic integration of classifiers. In: D.A. Zighed, J. Komorowski, J. Zytkow (eds.), *Principles of Data Mining and Knowledge Discovery, Proceedings of PKDD 00*, Springer, LNAI 1910 (2000) 116-125
13. Wilson D.R., Martinez T.R.: Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research,* 6(1) (1997) 1-34
14. Witten I., Frank E.: Data Mining: Practical Machine Learning Tools With Java Implementations, San Francisco: Morgan Kaufmann (2000)

Bagging Using Statistical Queries

Anneleen Van Assche and Hendrik Blockeel

Computer Science Department, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium

Abstract. Bagging is an ensemble method that relies on random resampling of a data set to construct models for the ensemble. When only statistics about the data are available, but no individual examples, the straightforward resampling procedure cannot be implemented. The question is then whether bagging can somehow be simulated. In this paper we propose a method that, instead of computing certain heuristics (such as information gain) from a resampled version of the data, estimates the probability distribution of these heuristics under random resampling, and then samples from this distribution. The resulting method is not entirely equivalent to bagging because it ignores certain dependencies among statistics. Nevertheless, experiments show that this "simulated bagging" yields similar accuracy as bagging, while being as efficient and more generally applicable.

1 Introduction

Ensemble methods build a set of different models and combine their predictions to classify new examples. These methods vary in the way they build the separate models and in the way they combine their predictions. Bagging [1] builds several models on replicate training sets that are produced by sampling with replacement from the original training set. By doing so it is able to improve the predictive accuracy.

In some learning settings, direct access to individual examples of the data is not available; instead, the learning algorithm has access only to summary statistics about the data. Several authors have described how various predictive models (such as decision trees) can be learned from precomputed statistics such as itemset frequencies [2] or AD-trees [3], and what the advantages are of doing so. When trying to apply bagging in the context of learning from statistics, one is confronted with the fact that statistics are needed for several resampled versions of the dataset, rather than for the dataset itself. Since no direct access to the data is available, resampling the dataset by randomly selecting (with replacement) individual examples from it is not possible, so the straightforward approach of resampling the dataset and computing the statistics for these resampled versions is not possible. An alternative is to simulate the computation of statistics from randomly resampled versions of the dataset by computing the distribution (under random resampling of the dataset) of the statistics themselves, and then sampling values for these statistics from those distributions. This approach, applied to the specific case of bagging decision trees, is the subject of this paper.

The paper is organized as follows. First we describe the learning problem in Sect. 2. Then Sect. 3 deals with bagging: we explain the basic algorithm we use in the bagging procedure, namely decision trees, and show how it is applied using statistical queries. Next we briefly focus on bagging. In Sect. 4 we provide a new way to simulate the use of bootstraps and indicate how this method differs from the original bagging procedure. Section 5 describes experimental results comparing this new method to the original bagging procedure. These results show that bagging can be simulated efficiently with the proposed approach. In Sect. 6 we conclude and provide some directions for future work.

2 Learning from Statistics

We start with defining and motivating the exact problem setting. We address the problem of classification using only statistical information about the data set to learn a classification model. We assume the data set is described by m attributes $(A_1 ... A_m)$ from which A_m is the class attribute C consisting of n_c possible class labels $(c_1 \dots c_{n_c})$. The learning algorithm does not have access to the training set E but is provided an oracle O that can be queried for frequencies $fr(Cr) = |\{e \in E | Cr(e)\}|$, where $Cr \in \mathcal{C}$ is taken from some (possibly restricted) space of conditions. For instance, $\mathcal{C}$ might contain all conditions of the form $A_i = a$ with a some value of A_i; all conjunctions of such conditions; all such conjunctions with at most K conjuncts; all such conjunctions of which the frequency is above some predefined threshold; etc. Note that if there are no restrictions to $\mathcal{C}$ (more specifically, if $\mathcal{C}$ is sufficiently expressive to identify any individual instance), then full information on the dataset is available and then this setting does not differ essentially from learning from the dataset itself, except for implementation issues.

Several researchers have worked in this setting. For instance, after running an algorithm such as Apriori [4] on a dataset, the frequencies of all itemsets with a frequency above Apriori's minimal support parameter have been computed and stored; Panov et al. [2] show how predictive models such as decision trees, decision rules and Naive Bayes can be computed from only this information. Similarly, Moore and Lee [3] proposed AD-trees, a data structure to efficiently store the frequency of any conjunction of attribute-value combinations in a dataset, and showed how for instance decision trees can be learned from AD-trees.

The advantage of this setting is that, once an efficient oracle is available (the AD-tree has been built, or the itemsets mined), predictive models can be built with far greater speed than when having to recompute all the necessary statistics (which involves at least one pass over the whole dataset) each time they are needed. The setting is also useful when mining databases where for reasons of privacy only statistical queries are allowed.

The oracle knows certain statistics about a single data set, the training set. But some learners employ randomly resampled versions of the training set. For instance, as Panov et al. [2] mention, Ripper prunes its rule sets by means of random resampling. Also bagging uses resampled versions of the training set. It is not obvious how such methods can be used in the setting of learning from frequencies. For bagging, we investigate this in this paper.

3 Bagging

3.1 Decision Trees

Decision trees are usually constructed top-down, from the root to the leaves. At each node of the tree a "most informative" test for that node is selected using some heuristic. Different heuristics have been proposed to select the best test [5,6]. In the remainder of this text we use information gain, but our method also works for other heuristics. The information gain $\mathcal{IG}_A(E)$ of an attribute A relative to a set of examples E is defined as $\mathcal{IG}_A(E) = \mathcal{E}(E) - \sum_{v \in \text{Values}(A)} \frac{E_v}{E} \mathcal{E}(E_v)$ where $\mathcal{E}(E)$ is the entropy of E ($\mathcal{E}(E) = \sum_{i=1}^{c} p_i \text{log}_2 p_i$, where p_i is the proportion of E belonging to class i). We see that the heuristic is determined by the class distributions p_i within a node, which in turn directly follow from frequencies $fr(c \wedge L)/fr(L)$ where L represents the conditions on the path from the tree's root to the current node, $fr(L)$ is the number of examples covered by this node, and $fr(c \wedge L)$ is the number of such examples of class c. Similarly, other statistics used by tree learners (e.g., for the stopping criterion) can be defined in terms of such frequencies. Panov et al. [2] describe how a good decision tree can be learned even from incomplete statistics (when the frequency of itemsets is unknown if it is below a certain threshold). In this paper we assume that decision trees can be learned in the described setting, and we focus on the bagging procedure.

3.2 Bagging

Bagging [1] operates by repeatedly resampling the training set and building trees on these resamples. The resamples E_i form replicate data sets (also called bootstraps), each consisting of n examples (with n equal to the size of the original data set E), drawn at random, but with replacement, from E. So each example e from E may appear repeated times or not at all in any particular E_i.

But since in our setting, examples cannot be accessed individually, straightforward sampling with replacement on the original data cannot be performed anymore. We only have statistics about the original data available. In the next section we explain how we will sample the statistics instead of the dataset.

4 Sampling the Statistics

The key idea to bagging without bootstrapping is the following. In classical bagging, the information gain $\mathcal{IG}_A$ (or some other heuristic) of an attribute A is computed from a random resample E_i of the original data set E. Since the resample is chosen randomly, it might just as well have been another one, which would have lead to a different value for $\mathcal{IG}_A$. Clearly, computing the exact $\mathcal{IG}_A$ on a random resample E_i is equivalent to sampling $\mathcal{IG}_A$ from its own distribution. Put differently: since $\mathcal{IG}_A$ is a function of the data set, and considering E_i a stochastic variable of which the distribution is known, the distribution of $\mathcal{IG}_A(E_i)$ can be computed. Generating values for $\mathcal{IG}_A$ from this distribution is equivalent to generating E_i from its own distribution and computing $\mathcal{IG}_A$ from it. We will refer to this procedure as "resampling the statistics" as opposed to "resampling the data sets" (and computing the statistics from them).

In our approach, we will not resample information gain directly; the statistics that we resample are frequencies of class and attribute values, from which information gain but also other useful statistics can be computed efficiently.

Resampling Statistics for a Single Test. Assume for the sake of illustration that we have 3 boolean attributes (A, B, C) and a binary class variable with values +, -. Figure 1a shows for each attribute the joint distribution of that attribute and the class attribute in a training set with 100 examples. (Note that in this paper the term distribution will usually refer to a sample distribution, measured by absolute frequencies, and not a population distribution.) From the frequencies shown there, the information gain of A, B and C can easily be computed. If we take a bootstrap sample of 100 instances from the training set, what will the joint distribution of the attributes and the class variable look like? Each instance, being randomly taken from the training set, has a $60/100 = 0.6$ chance of being $(+, \neg A)$, a 0.2 probability of being $(-, A)$, a 0.5 probability of being $(+, B)$, etc., and hence has the same probability of ending up in the corresponding cell in the joint distribution table of the resampled set. Clearly, the vector $(n_{+,A}, n_{+,\neg A}, n_{-,A}, n_{-,\neg A})$ is multinomially distributed with parameter (0, 0.6, 0.2, 0.2), and similarly for B and C.

Generally, if we have n_c classes $c_1, \ldots, c_{n_c}$ and n_a values $a_1, \ldots, a_{n_a}$ of an attribute A, then the $n_a n_c$-dimensional vector $(X_{11}, \ldots, X_{n_a n_c})$ where X_{ij} denotes the number of instances in dataset E with $A = a_i$ and class value c_j, is multinomially distributed with parameters $(p_{11}, \ldots, p_{n_a n_c})$ where $p_{ij} = X_{ij}/n$, with $n = \sum X_{ij}$ (the total number of instances in E):

$$P(X'_{11} = x_1, ..., X'_{n_a n_c} = x_k) = \frac{n!}{x_1!..x_k!} p_{11}^{x_1} ... p_{n_a n_c}^{x_k}$$

where $k = n_c * n_a$. We use the method proposed by Devroye [7] to generate this vector; this method consists of repeatedly generating a number for each separate component X_{ij} (so $n_a * n_c$ times) according to a binomial distribution, using the BTPE algorithm from [8].

Choosing the Best Test for a Single Node. With the above method for generating the X'_{ij}, we can accurately model how the $\mathcal{IG}$ of each attribute will be distributed, under the resampling conditions used by bagging. Figure 1c shows the distribution of the $\mathcal{IG}$s of attributes A, B and C under bootstrap resampling of the training set summarized in Fig. 1a. It can be seen, for instance, that the probability that C is the best test in any random bootstrap is very low.

	A	$\neg A$
$+$	0	60
$-$	20	20

	B	$\neg B$
$+$	50	10
$-$	10	30

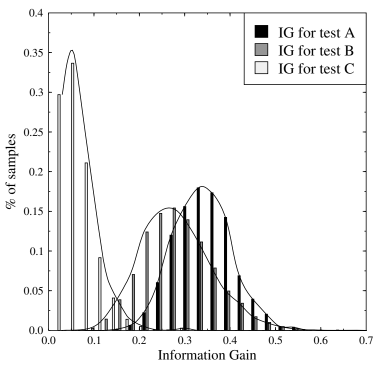
	C	$\neg C$
$+$	30	30
$-$	30	10

(a) Example of statistics $S[t]$

	ABC	$A\neg BC$	$AB\neg C$	$A\neg B\neg C$
$+$	0	0	0	0
$-$	5	0	12	3

	$\neg ABC$	$\neg A\neg BC$	$\neg AB\neg C$	$\neg A\neg B\neg C$
$+$	30	0	20	10
$-$	3	10	2	5

(b) Example of a joint distribution of Fig. 1a



(c) The distribution of information gains on random resamples of the original data for test A, B and C according to Fig. 1a

Fig. 1.

Unfortunately, the situation is not so clear for A and B. It might seem that we get a good estimate of the probability that $\mathcal{IG}_A > \mathcal{IG}_B$ by just generating random $\mathcal{IG}_A$ and $\mathcal{IG}_B$ and checking how often $\mathcal{IG}_A > \mathcal{IG}_B$. But with this procedure we are assuming that $\mathcal{IG}_A$ and $\mathcal{IG}_B$ are independent. This is not necessarily the case: A and B may be correlated and so may their $\mathcal{IG}$s. Consider for instance the joint distribution over A, B, C and the class, shown in Fig. 1b. This distribution gives the same marginal distributions as shown in Fig. 1a and in Fig. 1c. Yet, this joint distribution is such that $P(\mathcal{IG}_A > \mathcal{IG}_B)$ is approximately 1, whereas sampling $\mathcal{IG}_A$ and $\mathcal{IG}_B$ independently gives $P(\mathcal{IG}_A > \mathcal{IG}_B) = 0.72$.

It is impossible to correctly compute $P(\mathcal{IG}_A > \mathcal{IG}_B)$ from the distributions of $\mathcal{IG}_A$ and $\mathcal{IG}_B$; we need the joint distribution. To compute the latter we need the joint frequency distribution of A, B and the class. Generally, to correctly mimic bagging, the full joint distribution of all attributes and the class is required.

Unfortunately, using the full joint distribution may be infeasible in practice. First of all, knowledge of the full distribution corresponds to perfect knowledge of the training dataset; it may not really be interesting to try to simulate bagging in that context (unless, for instance, it would yield an efficiency gain). Further, the number of X' variables that need to be generated grows exponentially in the number of attributes, which makes this approach practically infeasible.

Of course, while we know that our approach is theoretically only an approximation of what happens in bagging, the question remains how much this matters in practice. In practical datasets, the correlations between attributes may be such that the probability of an attribute having the highest information gain under our approximate method or true bagging seldom differs strongly; and if these probabilities differ, there is still the question of how much this affects the quality of the bagged ensemble. The latter question is of more practical importance and we will focus on that in our experiments.

Choosing the Best Test in Multiple Nodes. The procedure described above assumes we are in the root node of the tree. Deeper in the tree, the statistics we resample for each possible test A_i in a certain node with path L from the root are the frequencies $fr(L \wedge A_i = a \wedge C = c)$ for each value a of A_i and each value c of the class attribute C. This means that for the different paths in the tree we need joint distributions over several attributes, but this is usually only a small subset of the full joint distribution. Note that while in real bagging one complete tree is built on one particular bootstrap, here new samples are drawn at each node. Again, we see that a perfect simulation of bagging is only possible by using the full joint distribution, which may not be feasible. So it remains to be seen experimentally how much our simulated bagging procedure differs from actual bagging.

5 Experiments

We implemented this simulated bagging method in the WEKA data mining system [9]. As a basic decision tree algorithm we started from ID3 [10]. For the binomial random number generation we used the CERN Java library [11] containing a high performance implementation of the BTPE algorithm [8]. More details about the implementation can be found in [12].

We compared the accuracy of simulated bagging (using resampling of the statistics) to original bagging (using resampling the data) on 12 UCI data sets[1]. We selected data sets with a varying data set size (from 286 to 67557 examples), all consisting of mainly nominal attributes, as this is required by the ID3 decision tree algorithm. We performed discretization where still necessary and removed missing values. To allow accuracy comparison between simulated bagging and the original bagging approach, no constraints were imposed on the frequencies that simulated bagging could query. Both methods are performed using 20 iterations of bagging and are evaluated by averaging over 5 stratified ten-fold cross-validations. Results on accuracy, size of the ensembles and runtimes are shown in Table 1 for simulated bagging (SB), simulated bagging with an adapted stopping criterion (SB_stop, discussed below) and regular bagging (Bag). As can be seen from the upper part of the table, in general the accuracy results are not significantly different from each other (also when taking standard deviations into account). Also, despite the fact that simulated bagging neglects dependencies among attributes, we found that the number of different tests chosen among the different iterations at a certain level in the trees was very similar for the two methods. The only point where trees of the two methods really differ is in their size. The middle part of Table 1 shows that sizes of the ensembles output by simulated bagging are in general larger than those of the ensembles of regular bagging. This is because in simulated bagging the frequencies that are queried in all nodes are always computed on the complete data set and not on a bootstrap,

[1] breast-cancer (br), kr-vs-kp (kr), primary-tumor (pr), soybean (so), splice (sp), waveform-5000 (wa), credit-a (cr), hypothyroid (hy), mushroom (mu), vote (vo), nursery (nu), connect-4 (co)

Table 1. Results on accuracy, size of the ensembles and runtime of simulated bagging (SB), SB with stopping criterion (SB_stop) and normal bagging (Bag) on 12 UCI data sets (abbreviations of data sets are mentioned in text above)

	br	kr	pr	so	sp	wa	cr	hy	mu	vo	nu	co
	Accuracy											
SB	65.7	99.6	39.8	91.7	94.3	79.1	82.7	99.5	100.0	95.3	99.1	79.6
SB_stop	65.9	99.6	39.8	91.7	94.3	79.6	84.2	99.5	100.0	95.7	99.0	80.0
Bag	66.0	99.6	36.4	90.1	93.8	77.4	82.6	99.4	100.0	94.6	99.0	79.3
	Size (number of nodes)											
SB	7262	1931	7287	4351	19941	100863	8346	1933	575	1400	17471	968167
SB_stop	4919	1720	5231	3155	14127	66582	5970	1592	566	1089	13432	671063
Bag	4523	1302	4980	2545	8416	42042	4276	1113	502	732	16006	534225
	Time (seconds)											
SB	0.436	1.102	1.616	1.050	6.396	6.466	0.556	0.440	0.252	0.214	1.156	209.786
SB_stop	0.296	0.986	1.178	0.788	5.140	4.154	0.404	0.400	0.246	0.180	0.946	100.384
Bag	0.282	2.216	0.842	1.078	5.692	5.574	0.566	1.298	2.060	0.224	3.074	187.624

which only contains 63% of the original examples. Table 1 also shows results when applying a stopping criterion that only looks at 63% of the data while checking the minimal leaf size condition to decide whether to stop or not, which results in shorter trees.

From the efficiency results, we can see that even in a normal learning setting where the data set is provided and regular bagging can be applied, simulated bagging can be useful as it yields similar accuracy often in less time. In [12] we discuss the time complexity of using this simulated bagging in a normal learning setting where the data set is available and compare it to regular bagging and a more efficient implementation of bagging [13]. We point out conditions under which simulated bagging might be preferred over regular bagging.

We conclude that ignoring dependencies between tests when sampling the statistics does not have a detrimental effect, while making bagging more generally applicable.

6 Conclusions

We have proposed a new method for approximating bagging in a learning setting where only statistics about the data and not the individual data instances themselves are available. In this context resampling the data set (with replacement), as is usually done by bagging, cannot be applied anymore. Here we propose a simulation of bagging that resamples the statistics instead of the data. Although it ignores certain dependencies among the statistics, it shows to be a good and efficient approximation to bagging.

The applicability of the described technique is not restricted to bagging. It can be applied to all methods using resampling, such as certain pruning methods (Panov et al.[2] described the need for that), some methods for error assessment, etc.

Our current implementation, based on ID3 [10], only handles data sets with nominal attributes. For numerical attributes the basic ideas presented here still apply, but there are some technical details that are more complicated. When computing statistics for each possible threshold for a numeric attribute, it suffices to go over the data set only once if examples are sorted according to the attribute. Then for each of the iterations of bagging we could take a sample of each of these statistics and compute the information gains on them. But statistics of successive thresholds are strongly correlated and by randomly sampling we would lose this characteristic. So after we have taken a sample for the first threshold we might want to impose some constraints on the samples of the next thresholds. This method will be investigated in the near future.

Acknowledgements. Anneleen Van Assche is supported by the Institute for the Promotion of Innovation by Science and Technology in Flanders (I.W.T.-Vlaanderen). Hendrik Blockeel is a Postdoctoral Fellow of the Fund for Scientific Research - Flanders (Belgium) (F.W.O.-Vlaanderen). We would like to thank the reviewers and chairs for their valuable comments and suggestions.

References

1. Breiman, L.: Bagging predictors. Machine Learning **24**(2) (1996) 123–140
2. Panov, P., Džeroski, S., Blockeel, H., Loškovska, S.: Predictive data mining using itemset frequencies. In: Proc. of the 7th Int. Multiconf. Information Society. (2005)
3. Moore, A.W., Lee, M.S.: Cached sufficient statistics for efficient machine learning with large datasets. Journal of Artificial Intelligence Research **8** (1998) 67–91
4. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In Buneman, P., Jajodia, S., eds.: Proc. of ACM SIGMOD Conf. on Management of Data, Washington, D.C., USA, ACM (1993) 207–216
5. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann (1993)
6. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth, Belmont (1984)
7. Devroye, L.: Non-Uniform Random Variate Generation. Springer-Verlag (1986)
8. Kachitvichyanukul, V., Schmeiser, B.: Binomial random variate generation. Communications of the ACM **31** (1988) 216–222
9. Witten, I., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2005) 2nd Edition.
10. Quinlan, J.R.: Induction of decision trees. Machine Learning **1** (1986) 81–106
11. Cern: European org. for nuclear research (1999) `http://dsd.lbl.gov/~hoschek/colt/`.
12. Van Assche, A., Blockeel, H.: Simulating bagging without bootstrapping. In Saeys, Y., Tsiporkova, E., De Baets, B., Van de Peer, Y., eds.: Proc. of the 15th Annual Machine Learning Conf. of Belgium and the Netherlands. (2006) 25–32
13. Blockeel, H., Struyf, J.: Efficient algorithms for decision tree cross-validation. Journal of Machine Learning Research **3**(Dec) (2002) 621–650

Guiding the Search in the NO Region of the Phase Transition Problem with a Partial Subsumption Test

Samuel Wieczorek[1,*], Gilles Bisson[2], and Mirta B. Gordon[2]

[1] Laboratoire Biologie, Informatique, Mathématiques, CEA-DSV-DRDC
17, avenue des martyrs, 38054 Grenoble Cedex 9, France
samuel.wieczorek@cea.fr
[2] Laboratoire Leibniz-IMAG, UMR 5522
46, Avenue Félix Viallet, 38031 Grenoble Cedex, France
{gilles.bisson, mirta.gordon}@imag.fr

Abstract. We introduce a test, named *π-subsumption*, which computes partial subsumptions between a hypothesis *h* and an example *e,* as well as a measure, the *subsumption index*, which quantifies the covering degree between *h* and *e*. The behavior of this measure is studied on the phase transition problem.

Keywords: Inductive Logic Programming, similarity index, *θ*-subsumption, partial subsumption, CSP, phase transition.

1 Introduction

In the learning systems using a first order logic representation, the *covering test* is a critical operation. It allows to check whether a given hypothesis *h* is consistent with a given example *e*. In ILP [15], the standard covering test is the *θ*-subsumption [17] which returns a boolean value indicating if the hypothesis covers an example or not. In such context, [4], [6] have shown that there is a *phase transition*, associated to the use of the *θ-subsumption*. These works also have shown the existence of failure regions where the learning algorithms are unable to find the actual concepts. These failures occur both around the phase transition (PT) region (*i.e.* the mushy region between the YES and NO phases) and in some parts of the NO region.

In this paper, we extend the classical covering test by introducing the notion of *partial covering* between a hypothesis and an example. Our covering test does not indicate whether a hypothesis covers an example by a boolean value, but assesses the importance of the partial covering by means of a "subsumption degree" index. The evaluation of our index is done in the same framework as in [4]. We show that it is possible to create a gradient in the NO region which smoothes the abrupt jump of the PT due to the *θ*-subsumption test. We expect such gradient will allow to efficiently guide the learning algorithms when they need to explore the NO region, for instance to be able to learn some complex conjunctive concepts.

[*] This work is done in the context of a PhD thesis in the laboratoire Leibniz.

The paper is organized as follows. Section 2 provides some definitions and gives a brief review of the solutions to compute partial subsumptions. In Section 3 we present a heuristic to compute a partial covering and the associated subsumption degree. In Section 4 the properties of this measure are studied in the phase transition problem.

2 Substitution and Generalization in ILP

We restrict here to the Datalog language[1] where the expressions are conjunctions of literals, each literal is built on a n-ary predicate which defines a property or relation between its arguments (variables or constants). First, let's give some definitions:

Definition 1. A *substitution* θ is a finite list of bindings X_i/t_i, where X_i is a variable in h and t_i a term in e. The application $h\theta$ of a substitution θ to a clause h is obtained by replacing all the occurrences of each variable X_i in h by the corresponding term t_i in e.

Definition 2. A clause h *θ-subsumes* a clause e (denoted $h \leq e$) iff there exists a substitution θ such that $h\theta \subseteq e$.

Definition 3. Given two clauses h and *e whose sets of variables are distinct*, one says that h *π-subsumes* e (π stands for partial) iff there exists a sub-expression h' in h ($h' \subseteq h$) and a substitution θ of the variables in h' such that $h'\theta \subseteq e$.

Definition 4. Given two clauses h and e such that $h \leq e$, the *subsumption index* (denoted I_π) is a function from (h, e) on the interval [0, 1] that quantifies to which extent h partially subsumes e, or in other terms the covering degree of h' relatively to h, with $h' \subseteq h$. The index satisfies the following condition: if $h'=h$ then $h \leq e$ and $I_\pi=1$; when no sub-expression $h' \leq e$, then h and e are completely dissimilar and $I_\pi=0$. A computational definition of I_π is given in section 3.2.

To search for a partial subsumption, a solution consists in generalizing the clauses h and *e, in order to build the sub-expression h'*. In ILP, the most studied generalization method is the *lgg* (Least General Generalization), defined by Plotkin [17], which builds the most specific generalization that θ-subsumes h and e. Once the lgg has been built, it becomes possible to evaluate the covering degree between h and e by calculating the ratio between the size of the *lgg* and the size of h. Unfortunately, as the *lgg* contains a large number of redundant literals, a *reduction step* is needed to obtain the minimal form h' relevant for evaluation of the size. This reduction can be carried out by using a complete algorithms such that [7], [12] or [1], which allow to search exhaustively for a minimal solution. However, the number of possible solutions may be potentially large since it corresponds to the size of the generalization lattice. In the NO region, this problem becomes crucial since the size of *lgg*(h, e) is huge. To perform the reduction, some authors [5], [8] propose heuristics whose the complexity is still high: $O(n^4)$ where n, according to the method, is either the number of variables or the number of literals.

[1] This restriction is done for the sake of simplicity. In FOL, the functional terms can be managed by using the "flattening techniques" turning functional terms into new predicates.

3 Partial Subsumption and Subsumption Index

In this section, we propose a heuristic that searches in *polynomial time* the substitutions θ from which the sub-expression *h'* can be deduced. In the rest of the paper we consider, *without any loss of generality*[2], that the arity of the predicates of *h* and *e* equals 2. We use a variant of the method proposed in [2], [3] to compute a *degree of subsumption* between any two clauses. This method is based on two steps:

- *a comparison step* to evaluate a local *degree of subsumption* between all pairs of variables in *h* and *e*. These values are stored in a subsumption matrix SUB;
- *a matching step* between the variables of *h* and *e*, guided by the content of SUB. This second step computes the substitutions θ and the *subsumption index*.

3.1 Local Comparison Between Variables

The work [2] allows to compute a *similarity index* between any two clauses *h* and *e*. For a given pair of variables x_i in *h* and a_m in *e*, the main idea is to consider that their similarity is larger when they appear in common predicates at the same position, and it increases further if their *neighbouring variables* (*i.e.* the variables occurring in the same literals as x_i and a_m) are themselves *similar to each other*. This recursive definition allows to formulate the calculus under the form of a *system of equations* and to ensure that the computed values reflect the similarity between the relational structures of *h* and *e*. We have enhanced the original work in several keypoints, the main modification was to replace the notion of similarity between x_i and a_m by the notion of *degree of subsumption*. Thus, we have a *referent* (the hypothesis *h*) and a *target* (the example *e*) and the index is no more symmetrical. Here is the algorithm:

```
function assess_subsumption_matrix (h, e, Nsub)
(L1)for k=1 to Nsub
(L2)   for each pair ($x_i$,$a_m$) of matchable variables in h and e
          sum := 0
(L3)      for each variable $x_j$ neighbour of $x_i$
             best_neighbour := 0
(L4)         for each variable $a_n$ neighbour of $a_m$
                neighbour := 0
(L5)            for each pair of literals {$p_r$($x_i$,$x_j$),$p_r$($a_m$,$a_n$)}
                   neighbour:=neighbour + Lsim($p_r$($x_i$,$x_j$),$p_r$($a_m$,$a_n$))
                best_neighbour:= max (best_neighbour, neighbour)
             sum = sum + best_neighbour
          $SUB_k$[$x_i$,$a_m$] = sum / card(literals($x_i$)) //normalisation
   return $SUB_{Nsub}$
```

Initially all the elements of SUB_0 are initialized to 1. The loop L1 evaluates the matrix SUB through an iterative *Jacobi's method* since the *system of equations* used to estimate the degree of subsomption between variables is not linear. At each iteration, for each pair of variables (x_i, a_m), we look for the best matching (*i.e.* the one maximizing the *degree of subsumption*) of the neighbours $\{x_1\ldots x_j\}$ of x_i with the

[2] This restriction is done to simplify the description of the algorithms. In practice, the results presented here hold for predicates of any arity (including the complexity evaluation in 3.3).

neighbours $\{a_1 \ldots a_n\}$ of a_m. The central step of the procedure is the comparison of the pairs of matchable literals $\{p_r(x_i, x_j), p_r(a_m, a_n)\}$ (*i.e.* which share the same predicate symbol p_r) through the function Lsim (see [2] for further details):

$$Lsim\big(p_r(x_i,x_j),p_r(a_m,a_n)\big)=\big(1+SUB_{k-1}[x_j,a_n]\big)/2 \tag{1}$$

This function expresses the fact that the *subsumption* between two literals is the average of the *subsumption degree* between their arguments. The *subsumption value* between x_i and a_m equals 1 since they appear in the same predicate and position. For x_j and a_n we use the *subsumption* value evaluated at the previous iteration (*i.e.* the value of $SUB_{k-1}[x_j, a_n]$); conversely, the evaluation of $SUB_k[x_j, a_n]$ for the same literals will recursively involve the value of $SUB_{k-1}[x_i, a_m]$. The algorithm iterates N_{sub} times, each iteration k taking implicitly into account the of *k-th* neigbours of the current variable x_i and a_m. However, since the influence of the neighborhood geometrically decreases according to the length of the path between the variables, the values of SUB converge after a small number iterations (we use N_{sub}=5 in our experiments).

3.2 Global Matching of Variables

The second step of the method looks for an *optimal (partial) matching* (i.e. maximizing the subsumption index) between the variables in the hypothesis and in the example. As the θ-subsumption test is equivalent to a Constraint Satisfaction Problem (CSP) [6], [16], [13] we use the *Minimizing Conflicts* heuristic [14], which is based on a local search algorithm. The method is initialized by assigning a value to every variable and, then iteratively changes the assignment of the variables one by one in order to minimize the number of conflicts[3]. It stops either when a complete assignment ($I_\pi = 1$) has been found or when the maximum number of iterations N_{max} has been reached and then returns the best assignment found. Two variants have been implemented: MCA_nonGuided (the assignments are not guided by SUB) and MCA_Guided (the assignments are guided by the values of SUB). We will discuss in section 4.2, the optimal value for the parameter N_{max}. For a given substitution θ, the subsumption index $I_\pi \in [0, 1]$, corresponding to a matching of h and e, is proportional to the sum of the *degree of subsumption* of the matched variables:

$$I_\pi(h,e)=\left[\sum_i^{V_{matched}} SUB_{Nsub}\big[x_i,\theta(x_i)\big]\times\left|literals(x_i)\right|\right] / \sum_k^{V}\left|literals(x_k)\right| \tag{2}$$

where:

- V (resp. $V_{matched}$) is the set of (resp. matched) variables in the hypothesis h;
- SUB_{Nsub} denotes the matrix SUB at the last iteration;
- $\theta(x_i)$ is the variable in e matched with the variable x_i in h by the substitution θ;
- $|literals(x_i)|$ is the cardinal of the set of literals in which the variable x_i appears.

In the numerator, we weight the value of $SUB_{Nsub}[x_i, \theta(x_i)]$ by the number of literals of x_i. The denominator yields the index I_π non-symmetrical and normalized in [0, 1].

[3] Given θ, a conflict occurs if there exists a literal $p_r(x, y)$ in h but no literal $p_r(\theta(x), \theta(y))$ in e.

3.3 Computational Complexity

The computational complexity of I_π is the sum of the complexities of the two steps needed for its evaluation: the evaluation of the matrix SUB and the search of the optimal match between the variables. Let's consider the following parameters:

- V_h and V_e: average number of variables in h and e;
- L_h and L_e: average number of literals in h and e;
- N_{sub}: iteration number for the computation of the matrix SUB;
- N_{max}: maximum number of iterations in the MCA algorithm.

The elementary step in the SUB calculus, is the call to the function Lsim. The algorithm is composed of a loop of N_{sub} iterations with $V_h \times V_e$ pairs of variables to assess. Each of these pairs needs to compute the *subsumption* between all couples of neighbouring variables. This number is of the order of $2\times(L_h/V_h)$ for the hypothesis and $2\times(L_e/V_e)$ for the examples, thus the number of comparisons is $4\times(L_h\times L_e / V_h\times V_e)$ for each pair. The complexity of this step is $O(N_{sub}\times L_h\times L_e)$, with N_{sub} being small (cf. 3.1). For the matching step, we consider the complexity of the algorithm MCA, composed of a main loop of N_{max} iterations. At each iteration, two functions are called, each one compares the sets of literals in h and e, thus giving a complexity of $O(2\times L_h\times L_e)$. The average complexity of this step is $O(N_{max}\times L_h\times L_e)$. Thus, the overall computational complexity of I_π index is $O((N_{sub} + N_{max}) \times L_h\times L_e)$.

4 Study of the Index Behavior

It has been shown [4], [6], [9] that upon the search of a substitution θ in the *θ*-subsumption test, a phenomenon named *phase transition* exists for some critical values of the *order parameters*, namely: the *constraint density* in the hypothesis and the *constraint tightness* in the example. Given two random formulas h and e, this transition separates [10], in the plane whose axes are the order parameters, a phase (called YES region) where h has a probability close to 1 to cover e and a phase (called NO region) where the solutions are sparse and the coverage probability is close to 0. In addition, the PT region concentrates the most complex problems and correspondingly there is a peak in computational complexity [18].

4.1 Study of the Phase Transition

In order to compare the behavior of the partial subsumption I_π with the *θ*-subsumption in the context of the PT, we use the experimental protocol defined by [4]. This protocol is based on a set random problems, each one being a pair (h, e) defined by a quadruplet (n, m, L, N) where the hypothesis h contains n variables and m literals (built on m different binary predicate symbols) and the example e contains L constants and N literals for each of the m predicate symbols. As in [4], we explore points in the $<m, L>$ plane, for values of $m \in [10, 50]$, $L \in [10, 50]$, n=6 and N=100. For each pair (m, L), we generate 100 random problems and compute the subsumption index I_π. The averages over these problems are reported on figure 1.

As expected, in the YES region we observe a flat zone with $I_\pi = 1$ consistent with the fact that the problems have a solution with probability 1. In addition, the index I_π regularly decreases on penetrating in the NO region where the algorithm identifies partial solutions *h'*. The value of I_π is proportional to the *degree of subsumption* between *h* and *e*. However, the decrease of I_π *is not linear* showing that the behavior of our comparison method is different from the one of a classical Levenstein distance.

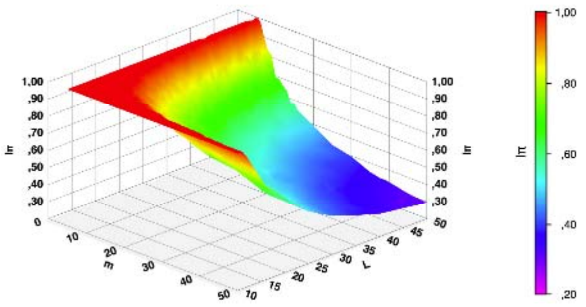


Fig. 1. Average values of index I_π on <*m*, *L*> plane with MCA_Guided (N_{max} =100 iterations)

4.2 Completeness of the Approach

Since the proposed method is based on a heuristic, it does not guarantee that an optimal solution to the problem will always be found (*i.e.* a complete substitution of variables in the YES region and the largest partial solutions in the NO region). Moreover, we need to evaluate the sensibility of our method with respect to the parameter N_{max} which controls the extension of search space explored to find the substitution. To answer both questions, we have modified the previous experimental protocol in such a way that, for each random problem (*h*, *e*), the example *e* is always θ-subsumed by *h* (*h* is named a *hidden solution*). In this test, as the *subsumption index* I_π should be equal to 1 for all problems, we can evaluate the minimal value of N_{max} that is needed to retrieve this solution. The results are summarized in figure 2.

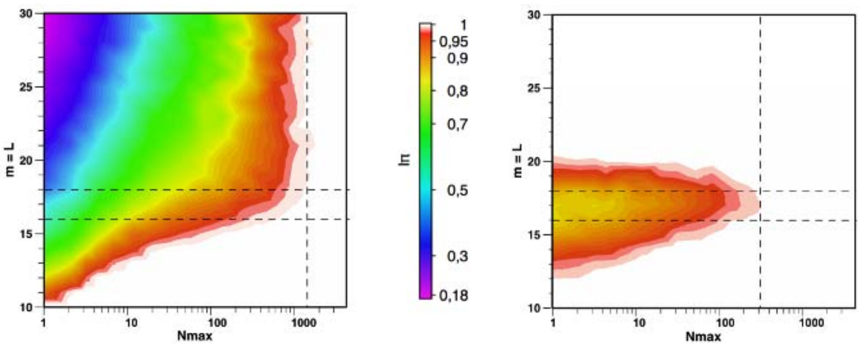


Fig. 2. Average value of index I_π according to the number of iterations N_{max} for the problems along the line *m* = *L*; the horizontal axis has a logarithmic scale. The behavior of the algorithms MCA_nonGuided is shown on the left (figure 2a) and MCA_Guided on the right (figure 2b).

In the YES region (m=$L \leq 16$), both algorithms are able to find a solution with a low number of iterations (usually N_{max}<100) and when m=$L \leq 12$, the MCA_Guided finds a solution from the start with the information contained in SUB.

In the PT region ($16 \leq m$=$L \leq 18$), we also observe a peak in computational complexity as described in [18], however the guided version is clearly better than the unguided one: not only it succeeds in retrieving a complete solution with a lower number of iterations ($N_{max}\approx 200$ versus $N_{max}\approx 1000$) but, for a given N_{max}, the found solution is better than the one returned by MCA_nonGuided. A closer examination of the matrix SUB_{Nsub} reveals that several subsumption values between the variables in h and e equal 1. That means that our method is unable to distinguish "from the local point of view" the different pairs of variables. Thus, it should explore more thoroughly the search space to find a satisfying solution in the matching step.

Finally, in the NO region (m=$L \geq 18$), the two algorithms have a very different behavior. The MCA_nonGuided algorithm needs many iterations (N_{max}>1000) to retrieve the solution. On the contrary, MCA_Guided finds the solution faster and for m=$L \geq 20$, the solution is found from the initial state (N_{max}=0) generated using SUB_{Nsub}. These results clearly show that the information contained in the subsumption matrix SUB improves the exploration of the search space in the PT region and particularly in the NO region. For the latter, the method seems really efficient since the overall complexity just depends on the product of the size, in terms of literals in h and in e: $O(|h|\text{x}|e|)$. This complexity is arguably lower than the one obtained by computing a *lgg* between h and e and doing the subsequent reduction.

5 Conclusion

In this paper, we have introduced the notion of *partial subsumption* (named π-subsumption) which, coupled with a *subsumption index* I_π, builds and quantifies the covering between two clauses h and e. We have studied the behavior of the covering test in the different regions of the *phase diagram* on random problems.

The *subsumption index* suppresses the abrupt jump of the θ-subsumption at the *phase transition*. This result is interesting since several works in ILP have shown that widely used search criteria do not allow to learn a complex conjunctive concept which belongs to the NO region [4]: the search remains confined in the PT region and the learned concept is over-generalized leading to a bad *generalization performance*.

Concerning the computational complexity, our approach is very efficient when the concept belongs to the NO region. However, it is necessary to explore more thoroughfully the search space when the problem is close to the *phase transition*. To speed-up our method in the YES and PT regions, we may use a simple meta-algorithm: first, to call an efficient system, like Django [13], to test if h θ-subsumes e then, if the answer is "false", to call our method to estimate the partial subsumption.

We plan to pursue the present study in two directions. First, we expect that the number of iterations N_{max} may decrease in the PT region by coupling the two steps of our method. Second, we will implement our heuristic in some *top-down* and *bottom-up* learning tools in order to verify the assumption that the gradient induced in the hypothesis space allows finding complex concepts in the NO region.

Acknowledgments

This work is part of the ACCAMBA project (`http://accamba.imag.fr`) and have been funded by the ACI IMPBio. We gratefully acknowledge S. Roy and S. Aci for helpful discussions, J. Maloberti who provided his implementation of Django and the reviewers of ECML-PKDD 2006 whose comments helped to clarify this work.

References

1. Arias M., Khardon R., Maloberti J.: Learning horn expressions with LOGAN-H. Technical report 2005-4 of the Department of Computer Science, Tufts University (2005)
2. Bisson G.: Learning in FOL with a similarity measure. In *Proceeding of 10th AAAI Conference*, San-Jose. (1992) 82-87
3. Bisson G.: Why and how to define a similarity measure for object-based representation systems. Proceedings of 2nd international conference on building and sharing very large-scale knowledge bases (KBKS). IOS press. Enschede (NL). (1995), 236-246
4. Botta M., Saitta L., Sebag M.: Relational learning as search in a critical region. *Journal of Machine Learning research*, 4 (2003) 431–463.
5. Costa V. S., Srinivasan A., Camacho, R., Blockeel H., Demoen B., Janssens G., Struyf J., Vandecastaele H., Van Laer, W.: Query transformations for improving the efficiency of ilp systems. *Machine Learning Research*, 4 (2003) 465–491
6. Giordana A., Saitta L.: Phase transitions in relational learning. *Machine Learning* 41 (2000) 17-25
7. Gottlob G., Fermüller C. G.: Removing redundancy from a clause. *Artificial Intelligence*, 61(2) (1993) 263–289
8. Hirata K.: On condensation of a clause. In Horvath T. and Yamamoto A., Eds, *Proceedings of the 15th Int. Conf. on Inductive Logic Programming, LNAI*, Springer-Verlag (2003) 164–179
9. Hogg T., Huberman B. A., Williams C. P.: Eds, *Artificial Intelligence: Special Issue on Frontiers in Problem Solving: Phase Transition and Complexity*, Elsevier 81 (1996) 1-2
10. Hogg T.: Refining the phase transition in combinatorial search. *Artificial Intelligence*, 81 (1996) 127–154
11. Khardon R.: Learning function-free horn expressions. *Machine Learning*, 37(3) (1999) 241–275
12. Maloberti J.: *Improving Inductive Logic Programming with Constraint Satisfaction Techniques: Applications to Frequent Query Discovery.* PhD thesis, LRI, Université d'Orsay (2005)
13. Maloberti J., Sebag M.: Fast theta-subsumption with constraint satisfaction algorithms. *Machine Learning*, 55(2) (2004) 137–174
14. Minton S., Philips A., Johnston M. D., Laird P.: Minimizing conflicts: A heuristic repair method for constraint-satisfaction and scheduling problems. *Journal of Artificial Intelligence Research*, 1 (1993) 1–15
15. Muggleton S., De Raedt L.: Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19 (1994) 629–679
16. Prosser P.: An empirical study of phase transitions in binary constraint satisfaction problems. *Artificial Intelligence*, 81 (1996) 81–109
17. Plotkin G.: A note on inductive generalization. *Machine Intelligence*, 5 (1970)
18. Smith B. M., Dyer M. E.: Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81 (1996) 155-181

Spline Embedding for Nonlinear Dimensionality Reduction

Shiming Xiang[1], Feiping Nie[1], Changshui Zhang[1], and Chunxia Zhang[2]

[1] State Key Laboratory of Intelligent Technology and Systems,
Department of Automation, Tsinghua University, Beijing 100084, China
{xsm, nfp03, zcs}@mail.tsinghau.edu.cn
[2] School of Computer Science, Software School,
Beijing Institute of Technology, Beijing 100081, China
cxzhang@bit.edu.cn

Abstract. This paper presents a new algorithm for nonlinear dimensionality reduction (NLDR). Smoothing splines are used to map the locally-coordinatized data points into a single global coordinate system of lower dimensionality. In this work setting, we can achieve two goals. First, a global embedding is obtained by minimizing the low-dimensional coordinate reconstruction error. Second, the NLDR algorithm can be naturally extended to deal with out-of-sample data points. Experimental results illustrate the validity of our method.

1 Introduction

Recently, nonlinear dimensionality reduction(NLDR) is a hot topic in machine learning. The motivation behind NLDR is to discover an informative representation hidden in high-dimensional data points. Generally, the task is to evaluate the embedding coordinates of the data points on the low-dimensional manifold.

Many algorithms have been proposed to learn a low-dimensional embedding: Isomap [1], locally linear embedding (LLE) [2], Laplacian eigenmap [3], Hessian LLE (HLLE) [4], local tangent space alignment (LTSA) [5], etc.. They are developed to embed a set of given data points. However, most of them are not capable of embedding new data points into the learned manifold in a natural way. This is known as *the out-of-sample problem.*

This paper presents a new algorithm for NLDR. The algorithm is developed from the conceptual framework of *compatible mapping.* A data point in the manifold can be represented in different local coordinate systems, but it has a single global coordinate on the low-dimensional manifold. Compatible mappings under different local coordinate systems map a data point into the low-dimensional manifold such that its global coordinate is unique. Smoothing splines are used to construct such mappings.

Our algorithm has many parallels to manifold learning by Isomap, LLE, Laplacian eigenmap, HLLE, LTSA, etc.. Most notably, the objective function is based on the reconstruction error, and a global low-dimensional embedding is achieved by solving an eigenvalue problem.

Our algorithm can be naturally extended to deal with out-of-sample data points, due to spline interpolation. Here, a natural extension mean that the algorithm can embed new data points in way of how to treat the training data points. For most manifold learning algorithms [6], however, it is difficult to achieve this goal in such a natural way.

2 Mappings

The NLDR problem can be formulated as follows. Given a set of n scattered data points $\boldsymbol{x}_i \in \mathbb{R}^m$ lying on a manifold M embedded in a m-dimensional Euclidean space. The goal is to invert an underlying generative model $\boldsymbol{x} = f(\boldsymbol{y})$ to find the corresponding low-dimensional parameters (embedding coordinates) $\boldsymbol{y}_i \in \mathbb{R}^d$ such that $\boldsymbol{x}_i = f(\boldsymbol{y}_i)$, say, construct $\mathcal{Y} = \{\boldsymbol{y}_i\}_{i=1}^n$ from $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$.

2.1 Compatible Mappings

Generally, given a parameter space $\Omega \subset \mathbb{R}^d$ and a mapping $f : \Omega \mapsto \mathbb{R}^m$ with $m > d$, then $M = f(\Omega)$ is called a parameterized manifold embedded in $\mathbb{R}^m$. In view of Riemannian geometry, it can be locally coordinatized. Thus we can use local coordinates to exploit the local geometrical structures of the data. Then we face a task—the points with local coordinates should be aligned together into the low-dimensional manifold with a single global coordinate system.

Denote the neighborhood of $\boldsymbol{x}_i$ $(i = 1, \cdots, n)$ by $\mathcal{N}_i$ $(\subset \mathcal{X} \subset M)$. According to the above analysis, for each $\mathcal{N}_i$, we need two mappings, $h_i : \mathcal{N}_i \rightarrow \mathcal{T}_i$ and $g_i : \mathcal{T}_i \rightarrow \mathcal{Y}_i$ $(\subset \mathcal{Y} \subset \Omega)$. The task of h_i is to locally coordinatize the data points in $\mathcal{N}_i$, while the task of g_i is to transform the local coordinates in $\mathcal{T}_i$ into the global coordinates. Let $\varphi_i = g_i \circ h_i$ be a compound mapping of h_i and g_i, then φ_i will be used to align the data points in $\mathcal{N}_i$ into a global coordinate system.

To avoid alignment conflicts, we need to consider the data points in the intersections of the neighborhoods. Without loss of generality, we consider $\mathcal{N}_1$ and $\mathcal{N}_2$. For $\boldsymbol{x} \in \mathcal{N}_1 \cap \mathcal{N}_2$, since it has a unique $\boldsymbol{y} \in \mathcal{Y}$, we have

$$\varphi_1(\mathcal{N}_1 \cap \mathcal{N}_2) = \varphi_2(\mathcal{N}_1 \cap \mathcal{N}_2) \tag{1}$$

Definition: Let φ_1 and φ_2 be two mappings. We say φ_1 and φ_2 are compatible with each other if $\varphi_1(\boldsymbol{x}) = \varphi_2(\boldsymbol{x})$ holds for any $\boldsymbol{x} \in \mathcal{N}_1 \cap \mathcal{N}_2$.

Now we can see that we need n mappings $\{\varphi_i\}_{i=1}^n$ which are compatible with each other. It seems quite complex to construct such mappings since there are lots of constraints as eq. (1). However, we have the following proposition:

Proposition: Let $\mathcal{N}_i$ contains k data points in $\mathcal{X}$ and denote $\mathcal{N}_i = \{\boldsymbol{x}_{i_j}\}_{j=1}^k$. Here, subscript i_j stands for an index and $i_j \in \{1, \cdots, n\}$. For each $\varphi_i = g_i \circ h_i$, if h_i is an injection and for $\boldsymbol{t}_i^j = h_i(\boldsymbol{x}_{i_j})$, we have

$$\boldsymbol{y}_{i_j} = g_i(\boldsymbol{t}_i^j), \ \ j = 1, 2, \cdots, k \tag{2}$$

then φ_i, $i = 1, \cdots, n$, are compatible with each other.

Since h_i is an injection, then $\boldsymbol{t}_i^1 = h_i(\boldsymbol{x}_{i_1}) \neq \boldsymbol{t}_i^2 = h_i(\boldsymbol{x}_{i_2})$ holds if $\boldsymbol{x}_{i_1} \neq \boldsymbol{x}_{i_2}$. Thus, each $\boldsymbol{x}_{i_j} \in \mathcal{N}_i$ has a unique local coordinate $\boldsymbol{t}_i^j$ in $\mathcal{T}_i = h_i(\mathcal{N}_i)$. Although $\boldsymbol{x}_{i_j}$ has different local coordinates in different coordinate systems, it has a unique global coordinate $\boldsymbol{y}_{i_j}$. According to eq. (2), the above proposition holds.

Based on this proposition, the constraints in eq. (1) can be naturally satisfied. Thus, we can construct h_i and g_i by only starting with $\mathcal{N}_i$.

We will treat h_i as a linear projection to a tangent space of M defined at $\boldsymbol{x}_i$ (subsection 2.2). In addition, smoothing splines will be used to construct g_i so that the constraints in eq. (2) can be faithfully satisfied (subsections 2.3-2.5).

2.2 Tangent Space Projection

We suppose that M is a smooth manifold, then the tangent space $T_x(M) \subset \mathbb{R}^m$ can be well defined at each point $\boldsymbol{x} \in M$. We use this subspace to define the local coordinates [4,5] and construct h_i. The steps can be summarized as follows:

First, for each data point $\boldsymbol{x}_i \in \mathcal{X}$, $i = 1, \cdots, n$, we identify its k-nearest neighbors in Euclidean distance to construct a neighborhood $\mathcal{N}_i$.

Then, for each $\mathcal{N}_i$, we let $X_i = [\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \cdots, \boldsymbol{x}_{i_k}] \in \mathbb{R}^{m\times k}$, and perform a singular value decomposition of the centralized matrix of X_i:

$$X_i(I - \tfrac{1}{k}\boldsymbol{e}\cdot\boldsymbol{e}^T) = U_i \Sigma_i V_i^T, \quad i = 1, \cdots, n \tag{3}$$

where I is a $k \times k$ identity matrix, $\boldsymbol{e} = [1, 1, \cdots, 1]^T \in \mathbb{R}^k$, $\Sigma_i = [\Sigma, 0]^T \in \mathbb{R}^{m\times k}$, in which $\Sigma = \mathrm{diag}(\sigma_1, \cdots, \sigma_k)$, U_i contains m left singular vectors, and V_i contains k right singular vectors. Finally, we get the local coordinate $\boldsymbol{t}_i^j \in \mathbb{R}^d$:

$$\boldsymbol{t}_i^j = (\tilde{U}_i)^T \cdot (\boldsymbol{x}_{i_j} - \bar{\boldsymbol{x}}_i) = \tilde{\Sigma}_i \cdot \boldsymbol{v}_i^j, \;\; j = 1, \cdots, k \tag{4}$$

here $\tilde{U}_i$ is a $m \times d$ sub-matrix which contains the first d column vectors of U_i, $\bar{\boldsymbol{x}}_i = \frac{1}{k}\sum_{j=1}^k \boldsymbol{x}_{i_j}$, $\tilde{\Sigma}_i = \mathrm{diag}(\sigma_1, \cdots, \sigma_d)$, $\boldsymbol{v}_i^j$ is a d-dimensional vector which contains the first d components of the j-th row of matrix V_i.

Now h_i can be defined as a subspace projection operator from $\mathbb{R}^m$ to $\mathbb{R}^d$, i.e., $h_i \overset{\triangle}{=} (\tilde{U}_i)^T$. In view of mappings, it is an injection since $(\tilde{U}_i)^T$ has full row rank.

2.3 Interpolation Conditions

Think momentarily of $\varphi_i = g_i \circ h_i$ as the inverse mapping of f near point $\boldsymbol{x}_i$. Note that f^{-1} is usually highly nonlinear and h_i is a linear operator, which is used to exploit the locally linear geometrical structure of the data. Then it is suitable to construct g_i as a nonlinear mapping. But g_i should meet the k constraints as formulated in eq. (2). To this end, we use nonlinear splines to construct g_i.

Now we let $Y_i = [\boldsymbol{y}_{i_1}, \boldsymbol{y}_{i_2}, \cdots, \boldsymbol{y}_{i_k}] \in \mathbb{R}^{d\times k}$ contain k global coordinates of data points in $\mathcal{N}_i$. Denote its r-th row by $[y_{i_1}^{(r)}, \cdots, y_{i_k}^{(r)}]$, $r = 1, \cdots, d$. In terms of splines, we divide the constraints in eq. (2) into d groups of *interpolation conditions*. The r-th group corresponds to the r-th coordinate components:

$$y_{i_j}^{(r)} = g_i^r(\boldsymbol{t}_i^j), \;\; j = 1, 2, \cdots, k \tag{5}$$

Finally, g_i can be constructed as a vector function, which contains d splines, namely, $g_i = [g_i^1, g_i^2, \cdots, g_i^d]^T$.

2.4 Smoothing Splines

To be clear, we use $\boldsymbol{\theta}_j$, z_j and g to replace $\boldsymbol{t}_i^j$, $y_{i_j}^{(r)}$ and g_i^r in eq. (5). Then

$$z_j = g(\boldsymbol{\theta}_j), \quad j = 1, 2, \cdots, k \tag{6}$$

For eq. (6), we can minimize the following objective functional:

$$J(g) = \tfrac{1}{k} \textstyle\sum_{j=1}^{k} (z_j - g(\boldsymbol{\theta}_j))^2 + J_s^d(g) \tag{7}$$

where $J_s^d(g)$ is a penalty functional, which stipulates the continuity of functions. We choose to optimize the functional $J(g)$ in Sobolev space [7,8]. Then, a smoothing spline solution can be formulated as follows [7,8]:

$$g(\boldsymbol{\theta}) = \textstyle\sum_{j=1}^{k} \alpha_j \, \phi_j(\boldsymbol{\theta}) + \sum_{i=1}^{l} \beta_i \, p_i(\boldsymbol{\theta}) \tag{8}$$

where $\phi_j(\boldsymbol{\theta})$ is a Green's function and $p_i(\boldsymbol{\theta})$ is a polynomial term. All such $p_i(\boldsymbol{\theta})$, $i = 1, \cdots, l$, constitute a basis of a polynomial space with degree less than s. Considering the second term in eq. (8) as a linear polynomial and denoting $\boldsymbol{\theta} = [\theta_1, \theta_2]^T$ in case of $d = 2$, for example, then we have $p_1(\boldsymbol{\theta}) = 1$, $p_2(\boldsymbol{\theta}) = \theta_1$ and $p_3(\boldsymbol{\theta}) = \theta_2$. In this case, l is equal to 3. In addition, $\phi_j(\boldsymbol{\theta}) = (||\boldsymbol{\theta} - \boldsymbol{\theta}_j||)^{2s-d} \cdot \log(||\boldsymbol{\theta} - \boldsymbol{\theta}_j||)$ if d is even; $\phi_j(\boldsymbol{\theta}) = (||\boldsymbol{\theta} - \boldsymbol{\theta}_j||)^{2s-d}$ if d is odd.

To avoid degeneracy, we add conditionally positive definition constraints:

$$\textstyle\sum_{j=1}^{k} \alpha_j \cdot p_i(\boldsymbol{\theta}_j) = 0, \quad i = 1, \cdots, l \tag{9}$$

Now substituting the interpolation conditions into eq. (8) and eq. (9), we can get a linear system for solving the coefficients $\boldsymbol{\alpha} \in \mathbb{R}^k$ and $\boldsymbol{\beta} \in \mathbb{R}^l$:

$$\begin{pmatrix} K & P \\ P^T & 0 \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{z} \\ 0 \end{pmatrix}, \quad \text{i.}e., \quad A \cdot \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{z} \\ 0 \end{pmatrix} \tag{10}$$

where K is a $k \times k$ symmetrical matrix with elements $K_{ij} = \phi(||\boldsymbol{\theta}_i - \boldsymbol{\theta}_j||)$, P is a $k \times l$ matrix with elements $P_{ij} = p_i(\boldsymbol{\theta}_j)$, $\boldsymbol{z} = [z_1, \cdots, z_k]^T$, and A is used to denote the coefficient matrix.

The spline as formulated in eq. (8) has several good properties. First, $g(\boldsymbol{\theta}_j) = z_j$ holds for all j when P has full row rank. This can be satisfied in the work setting of NLDR. In case of $d = 2$, for example, the scattered points should not be in a line. Faithfully satisfying the interpolation conditions is necessary for us to construct compatible mappings. Second, it is smooth and we can use it to interpolate new points near $\{\boldsymbol{\theta}_j\}_{j=1}^k$. This yields a mechanism to treat the out-of-samples. Third, the interpolation error can be estimated in form of k interpolation values z_j, $j = 1, \cdots, k$. Thus, we can construct an objective function. By minimizing it, a global optimal embedding can be achieved.

Actually, $J_s^d(g)$ can be approximated as $\boldsymbol{\alpha}^T K \boldsymbol{\alpha} = \boldsymbol{z}^T (B^T K B) \boldsymbol{z}$ [9], in which B is the upper left $k \times k$ subblock of A^{-1}. By some mathematical deductions, it turns out the following interpolation error for $J(g)$:

$$J(g) \approx \boldsymbol{z}^T B \boldsymbol{z} \tag{11}$$

2.5 Mapping Local Coordinates to Global Coordinates

The steps for constructing a vector function g_i are summarized as follows:

(1) Use $\boldsymbol{t}_i^j$ to replace $\boldsymbol{\theta}_j$ in eq. (6), and calculate the coefficient matrix A in eq. (10). Adding a subscript i, we get A_i;

(2) Use eq. (10) d times for d coordinate components. Actually, we have

$$A_i \cdot \begin{pmatrix} \boldsymbol{\alpha}_1, & \cdots, & \boldsymbol{\alpha}_d \\ \boldsymbol{\beta}_1, & \cdots, & \boldsymbol{\beta}_d \end{pmatrix} = \begin{pmatrix} Y_i^T \\ 0 \end{pmatrix} \tag{12}$$

(3) Respectively substitute coefficients $(\boldsymbol{\alpha}_1, \boldsymbol{\beta}_1), \cdots, (\boldsymbol{\alpha}_d, \boldsymbol{\beta}_d)$ into eq. (8) to construct d splines: $g_i^1, \cdots, g_i^d$. Thus we obtain $g_i = [g_i^1, \cdots, g_i^d]^T$, which is used to map the local coordinates to the global coordinates.

Finally, based on eq. (11), the reconstruction error for d coordinate components can be calculated as follows:

$$e_i = tr(Y_i B_i Y_i^T) \tag{13}$$

where tr is a trace operator, and B_i is the upper left $k \times k$ subblock of A_i^{-1}.

3 Spline Embedding

3.1 Global Embedding

In Section 2, we discuss how to map the data points in each local neighborhood into a single global coordinate system. Now our task is to assemble the local treatments together to obtain an optimal global embedding $\mathcal{Y} = \{\boldsymbol{y}_i\}_{i=1}^n$.

For k data points $\{\boldsymbol{x}_{i_j}\}_{j=1}^k$ in each $\mathcal{N}_i$, $i = 1, \cdots, n$, we first calculate their local coordinates $\{\boldsymbol{t}_i^j\}_{j=1}^k$ according to the method introduced in subsection 2.2.

When mapping such k local coordinates into a single global coordinate system, we get a reconstruction error e_i as formulated in eq. (13). Finally, we can obtain the following global coordinate reconstruction error:

$$E(Y) = \textstyle\sum_{i=1}^n e_i = \sum_{i=1}^n tr(Y_i B_i Y_i^T) \tag{14}$$

Further let $Y = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n] \in \mathbb{R}^{d\times n}$ and $S_i \in \mathbb{R}^{n\times k}$ be a 0-1 selection matrix such that $Y_i = YS_i$. Then we have

$$E(Y) = \textstyle\sum_{i=1}^n tr(YS_i B_i S_i^T Y^T) = tr(YSBS^T Y^T) \tag{15}$$

where $S = [S_1, \cdots, S_n] \in \mathbb{R}^{n\times nk}$ and $B = \mathrm{diag}(B_1, \cdots, B_n) \in \mathbb{R}^{nk\times nk}$.

We use $E(Y)$ as the objective function. To avoid degenerate solutions, we add a constraint $YY^T = I$. Then, the minimum of $E(Y)$ for the d-dimensional global embedding can be obtained from the d eigenvectors of symmetrical sparse matrix SBS^T, which are associated to $d + 1$ smallest eigenvalues. We leave out the eigenvector corresponding to eigenvalue 0 and use the next d eigenvectors to construct the matrix Y. Finally, we obtain a d-dimensional global embedding for n scattered data points $\boldsymbol{x}_i$, $i = 1, 2, \cdots, n$.

3.2 Extension for Out-of-Sample Data Points

Given a low-dimensional embedding $\mathcal{Y} = \{\boldsymbol{y}_i\}_{i=1}^n$ of n data points $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$. Now the task is to embed a new data point $\boldsymbol{x} \in M$.

Let $\mathcal{X}' = \mathcal{X} \cup \{\boldsymbol{x}\}$. We first construct a neighborhood $\mathcal{N}_x$ from $\mathcal{X}'$. Without loss of generality, we let $\mathcal{N}_x = \{\boldsymbol{x}, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_k\}$, with $\boldsymbol{x}_j \in \mathcal{X}$, $j = 1, \cdots, k$.

For $k+1$ data points in $\mathcal{N}_x$, then we calculate their local coordinates and denote them by $\boldsymbol{t}, \boldsymbol{t}_1, \cdots, \boldsymbol{t}_k$. Finally, according to the k known coordinates $\boldsymbol{y}_j \in \mathcal{Y}$, $j = 1, \cdots, k$, we re-construct a vector function g such that $\boldsymbol{y}_j = g(\boldsymbol{t}_j)$. Based on g, we can map $\boldsymbol{t}$ into the d-dimensional global coordinate system and thus embed the new data point $\boldsymbol{x}$ into the learned manifold.

4 Experimental Results

In this section, we validate spline embedding and its extension for out-of-samples, using synthetic data points and several real-world image data sets.

Fig. 1(a) shows $n = 1000$ data points sampled from a Swiss roll surface. The parameter domain of this surface is a 2D rectangular segment. The learned results by spline embedding using $k = 12$ nearest neighbors are shown in Fig. 1(b). The color coding shows that it yields a faithful embedding. Fig. 1(c) and Fig. 1(d) demonstrate another experiment. The 1000 data points are also sampled from the same surface, but with a hole. As can be seen that the topology is well learned. Fig. 1(e) shows 1500 data points on a S-surface with four holes. Among those data points, only 750 data points are randomly selected from the half part of the whole S-surface, in which two holes are cut. The rest 750 points are generated deterministically according to the symmetry of the surface. The results learned by spline embedding with $k = 12$ are shown in Fig. 1(f). We can see the learned 2D data points are roughly symmetrical about the line in the middle.

Fig. 2(a) shows the results of spline embedding which is applied to $n = 1965$ grayscale images of faces, with $k = 12$. The size of the images is 28×20. Thus the dimensionality of the data is 560. Representative faces are given to illustrate the poses and facial expressions. The results in Fig. 2(b) are learned from $n = 698$ images of a 3D statue, with $k = 12$. Each image includes 64×64 grayscale pixels. The manifold is embedded in $\mathbb{R}^{4096}$. Representative images are shown to illustrate the lighting conditions and the camera directions. The results in Fig. 3(c) are learned from $n = 400$ color teapot images [10], with $k = 5$. The size of the images is 76×101. Thus the manifold is embedded in $\mathbb{R}^{23028}$. The results roughly spread along a circle, which reveal the rotations of the teapot.

Now we demonstrate an example of embedding out-of-sample data points. The original data points are taken from a group of synthetic images with 64×64 pixels. Each such image contains a 16×16 white square (see Fig. 3(a)).

First, we let the x and y coordinates of the center of the white square vary from 8 to 56, both with translation step $t_x = t_y = 2$. We get 576 images for training. Fig. 3(b) shows the 2D embedding results for these images, with $k = 12$.

Then, we generate 576 new samples, changing x and y from 9 to 57, also with step $t_x = t_y = 2$. Based on the learned results as shown in Fig. 3(b), such 576

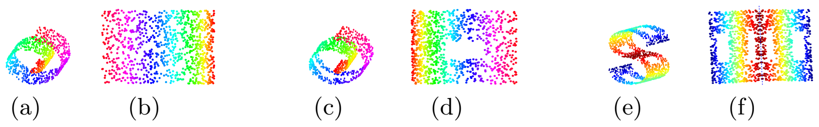


Fig. 1. (a). the 1000 data points; (b). 2D embedding results of the data points shown in (a); (c). the 1000 data points; (d). 2D embedding results of the data points in (c); (e). the 1500 data points; (f). 2D embedding results of the data points in (e)

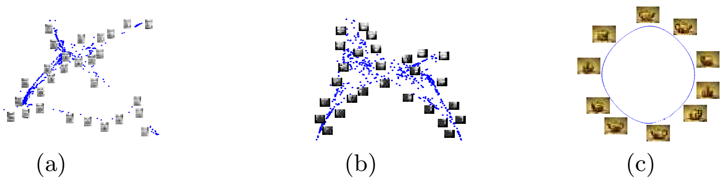


Fig. 2. (a). 2D embedding of 1965 graysacle face images; (b). 2D embedding of 698 graysacle statue images; (c). 2D embedding of 400 color teapot images

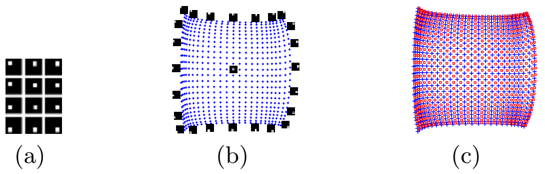


Fig. 3. (a). Examples of the synthetic binary images; (b). 2D embedding of 576 training images; (c). The embedded 576 out-of-samples, indicated by (red) square points

new data points are embedded one-by-one by using our method. The results are demonstrated in Fig. 3(c), indicated by the (red) square points. As can be seen that the new points are all faithfully embedded into the right positions.

5 Discussion and Conclusion

Like other manifold learning algorithms, such as Isomap, LLE, Laplacian Eigenmap, LTSA, HLLE, etc., it is necessary for us to select a proper k to explore the locally geometrical structures. Generally, k is not too small and too big since manifold is a combination of locally linear patches. If the data points are densely sampled from the manifold, we can take a bigger k.

To embed new data points, it needs smoothing splines. To obtain a C^p continuous spline, it requires $2s - d > p$. Given s and d, there have $l = (d + s - 1)!/(d!(s - 1)!)$ terms in the polynomial. Thus the number of the nearest neighbors k should be equal to or greater than l. In case of $s = 2$ and $d = 2$, it needs $k \geq 3$. To learn a global manifold embedding from a set of given data points, we can take the second term in eq. (8) as a linear polynomial.

Compared with LTSA, our method can achieve nonlinear alignments during coordinate mapping. In computation, it needs to construct the coefficient matrix A in eq. (10). The computational complexity is quadratic in the number of neighbors k. The complexities of other computations are roughly same as those of LTSA. A 2D embedding of $n = 1500$ data points in $\mathbb{R}^3$ by spline embedding with $k = 12$ takes about 10.1s on a 1.7GHz CPU using Matlab. For the same task, LTSA needs about 9.2s, while HLLE needs about 92.5s.

In this paper, we propose a new nonlinear dimensionality reduction algorithm, spline embedding, which is developed from compatible mappings. Splines are used to construct such mappings. A global embedding is finally achieved by minimizing the low-dimensional global reconstruction error. Due to the spline interpolation, the proposed methods can be naturally extended to deal with out-of-samples. In further, we would like to study semi-supervised spline embedding.

Acknowledgements

This work is supported by the Projection (60475001) of the National Nature Science Foundation of China. The anonymous reviewers have helped to improve the representation of this paper.

References

1. Tenenbaum, J. B., de Silva, V., Langford, J. C.: A global geometric framework for nonlinear dimensionality reduction. Science. **290** (2000) 2319–2323
2. Roweis, S. T., Saul, L. K.: Nonlinear dimensionality reduction by locally linear embedding. Science. **290** (2000) 2323–2326
3. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation. **15** (2003) 1373–1396
4. Donoho, D. L., Grimes, C. E.: Hessian eigenmaps: locally linear embedding techniques for highdimensional data. Proceedings of the National Academy of Arts and Sciences. **100** (2003) 5591–5596
5. Zhang, Z.Y., Zha, H.Y.: Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. SIAM Scientific Computing. **26** (2004) 313–338
6. Bengio, Y., Paiement, J., Vincent, P.: Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps and spectral clustering. Advances in Neural Information Processing Systems 16. Cambridge, MA: MIT Press (2004)
7. Duchon, J.: Splines minimizing rotation-invariant semi-norms in Sobolev spaces. In: Schempp, W., Zeller, K. (eds): Constructive Theory of Functions of Several Variables. Berlin: Springer-Verlag (1977) 85–100
8. Wahba, G.: Spline models for observsatonal data. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM Press (1990)
9. Bookstein, F. L.: Principal warps: thine-plate splines and the decomposition of deformations. IEEE Transactions on PAMI. **11**(1989) 567–585
10. Weinberger, K. Q., Saul, L. K.: Unsupervised learning of image manifolds by semidefinite programming. In: CVPR. Washington DC, USA (2004) 988–995

Cost-Sensitive Learning of SVM for Ranking

Jun Xu[1], Yunbo Cao[2], Hang Li[2], and Yalou Huang[1]

[1] College of Software, Nankai University, Weijin Road 94, Tianjin, China
nkxj@hotmail.com, yellow@nankai.edu.cn
[2] Microsoft Research Asia, Zhichun Road 49, Beijing, China
{yucao, hangli}@microsoft.com

Abstract. In this paper, we propose a new method for learning to rank. 'Ranking SVM' is a method for performing the task. It formulizes the problem as that of binary classification on instance pairs and performs the classification by means of Support Vector Machines (SVM). In Ranking SVM, the losses for incorrect classifications of instance pairs between different rank pairs are defined as the same. We note that in many applications such as information retrieval the negative effects of making errors between higher ranks and lower ranks are larger than making errors among lower ranks. Therefore, it is natural to bring in the idea of cost-sensitive learning to learning to rank, or more precisely, to set up different losses for misclassification of instance pairs between different rank pairs. Given a cost-sensitive loss function we can construct a Ranking SVM model on the basis of the loss function. Simulation results show that our method works better than Ranking SVM in practical settings of ranking. Experimental results also indicate that our method can outperform existing methods including Ranking SVM on real information retrieval tasks such as document search and definition search.

1 Introduction

Learning to rank is an important research topic in machine learning, because many issues in information retrieval and data mining can be formalized as ranking problems. For example, in information retrieval, the user types a query, and the system calculates the relevance scores of documents with respect to the query and returns the documents in descending order of the relevance scores. The relevance scores can be calculated by a ranking function constructed with machine learning.

Machine learning approaches to ranking have been proposed. Ranking SVM [14] is such a method. It formulizes the learning to rank as that of learning for classifying instance pairs into two categories and trains a SVM model to perform the task.

We note that in many applications such as information retrieval, the negative effects of making errors between higher ranks and lower ranks are larger than making errors among lower ranks[5]. In Ranking SVM, however, the losses for incorrect classifications of instance pairs between different rank pairs are defined as the same (i.e., only correctly ranked and incorrectly ranked are considered). Therefore, to make Ranking SVM more useful in practice, it is necessary to change the formation of the loss function.

In the paper, we propose a new method for learning 'Ranking SVM' based on cost-sensitive learning. Specifically, we set different losses for misclassification of instance pairs between different rank pairs. We find it is feasible to make a generalization of the learning algorithm of Ranking SVM such that given a cost-sensitive loss function we construct a Ranking SVM model on the basis of the loss function.

2 Related Work

The problem of learning to rank can be formulized as classification [10],[12], regression [17], or ordinal regression. The ordinal regression methods can be further classified into two groups: referred to, in this paper, as 'point-wise training' (c.f. [6], [7]) and 'pair-wise training' (c.f. [4], [14]) respectively. We consider pair-wise training in this paper. Ranking SVM [14] is a typical method of ordinal regression based on pair-wise training. It formulizes the learning to rank problem as that of learning for classifying instance pairs into two categories (correctly ranked and incorrectly ranked) and trains a SVM model to perform the task.

Cost-sensitive learning is a sub-field of supervised learning in which classifiers are constructed when different penalties (losses) are needed for different types of classification errors [9]. Recent research on cost-sensitive learning can be grouped into three categories: 1) creating particular classifiers in cost-sensitive learning (e.g., [3], [11]), 2) assigning each example to its lowest expected cost class (e.g., [8], [24]), and 3) modifying the distribution of training examples prior to performing learning (e.g., [1], [18]).

3 Learning of SVM for Ranking

Assume that a training set of labeled data is available. Each instance $(\vec{x}_i, y_i) \in R^n \times Y$ has been generated independently from an unknown distribution, where $\vec{x}$ denotes a feature vector and y denotes a label of rank. In the space of ranks $Y = \{r_1, r_2, \cdots, r_q\}$, there exists a total order between the ranks $r_q \succ r_{q-1} \succ \cdots \succ r_1$, where '$\succ$' denotes preference relationship. The goal of ranking learning is to induce a ranking function $f \in F$, which can determine preference relation between instances:

$$\vec{x}_i \succ \vec{x}_j \Leftrightarrow f(\vec{x}_i) > f(\vec{x}_j) \tag{1}$$

Herbrich et al propose transforming the learning task into that of learning for classification *on pairs of instances* (Ranking SVM)[14]. First, we assume that f is a linear function:

$$f_{\vec{w}}(\vec{x}) = \langle \vec{w}, \vec{x} \rangle, \tag{2}$$

where $\vec{w}$ denotes a vector of weights and $< \cdot, \cdot >$ stands for an inner product. Plugging (2) into (1) we obtain

$$\vec{x}_i \succ \vec{x}_j \Leftrightarrow \langle \vec{w}, \vec{x}_i - \vec{x}_j \rangle > 0 \tag{3}$$

Note that the relation $\vec{x}_i \succ \vec{x}_j$ between instance pairs $\vec{x}_i$ and $\vec{x}_j$ is expressed by a new vector $\vec{x}_i - \vec{x}_j$. Next, we take any instance pair and their relation to create a new vector and a new label. Let $\vec{x}^{(1)}$ and $\vec{x}^{(2)}$ denote the first and second instances, and let $y^{(1)}$ and $y^{(2)}$ denote their ranks, then we have

$$\left(\vec{x}^{(1)} - \vec{x}^{(2)}, z = \begin{cases} +1 & y^{(1)} \succ y^{(2)} \\ -1 & y^{(2)} \succ y^{(1)} \end{cases}\right) \tag{4}$$

From the given training data set *S,* we create a new training data set *S′* containing ℓ labeled vectors:

$$S' = \left\{\left(\vec{x}_i^{(1)} - \vec{x}_i^{(2)}, z_i\right)\right\}_{i=1}^{\ell} \tag{5}$$

Next, we take *S′* as classification data and construct a SVM model that can assign either positive label $z = +1$ or negative label $z = -1$ to any vector $\vec{x}^{(1)} - \vec{x}^{(2)}$.

Constructing the SVM model is equivalent to solving the following quadratic optimization problem:

$$\begin{aligned} &\min_{\vec{w}} M(\vec{w}) = \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i \\ &\text{subject to } \xi_i \geq 0, \quad z_i\left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)}\right\rangle \geq 1 - \xi_i \quad i = 1, \cdots, \ell. \end{aligned} \tag{6}$$

Note that the optimization in (6) is equivalent to that in (7), when $\lambda = \frac{1}{2C}$ [13].

$$\min_{\vec{w}} \sum_{i=1}^{\ell}\left[1 - z_i\left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)}\right\rangle\right]_+ + \lambda\|\vec{w}\|^2, \tag{7}$$

where subscript '+' indicates positive part. The first term is the so-called 'empirical hinge loss' and the second term is a regularizer.

Suppose that $\vec{w}^*$ is the weights in the SVM solution. We utilize $\vec{w}^*$ to form a ranking function $f_{\vec{w}^*}(\vec{x}) = \left\langle \vec{w}^*, \vec{x}\right\rangle$.

In many applications such as information retrieval the negative effects of making errors between higher ranks and lower ranks are larger than making errors among lower ranks[5]; for example, usability studies show that search users usually only look at the top ranked results [22].

Let us consider an example in information retrieval. Suppose that there are seven documents to rank and there are three possible ranks: definitely relevant, possibly relevant, and not relevant, denoted as 'd', 'p', and 'n' respectively. The perfect ranking is given and two rankings 1 and 2 are supposed to be made.

Perfect ranking:	d p p n n n n
Ranking 1:	*p d* p n n n n
Ranking 2:	d p *n p* n n n

Both ranking 1 and ranking 2 are not perfect. From practical viewpoint, the 'loss' in ranking 1 should be larger than that in ranking 2, because the error in ranking 1 is between the highest rank d and the middle rank p, while the error in ranking 2 is between the middle rank p and the lowest rank n. That is to say, a larger penalty should be added to the former. However, this is not reflected in the conventional Ranking SVM or any other existing ranking methods.

To deal with the problem, we propose a new type of SVM model for ranking, which retains different losses for different rank pairs. Specifically, we introduce penalty weights τ's into different rank pairs in the loss function in (7). That is to say, we re-formalize SVM learning problem as that of minimizing the following loss function.

$$\min_{\vec{w}} L(\vec{w}) = \sum_{i=1}^{\ell}\tau_{k(i)}\left[1 - z_i\left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)}\right\rangle\right]_+ + \lambda\|\vec{w}\|^2, \tag{8}$$

where $k(i)$ represents the type of rank pairs $y_i^{(1)}$ and $y_i^{(2)}$ with regard to $\vec{x}_i^{(1)} - \vec{x}_i^{(2)}$; $\tau_{k(i)}$ represents penalty weight for $k(i)$.

If it is to penalize errors between a rank pair, then we can assign a larger weight (larger than 1.0). Note that our method contains Ranking SVM as a special case in which all τ's equal 1.0.

In our method, instead of directly solving (8), we solve the equivalent quadratic optimization problem as described below.

$$\begin{aligned} &\min_{\vec{w}} M(\vec{w}) = \frac{1}{2}\|\vec{w}\|^2 + \sum_{i=1}^{\ell} C_{k(i)} \cdot \xi_i \\ &\text{subject to } \xi_i \geq 0, \quad z_i \left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \right\rangle \geq 1 - \xi_i \quad i = 1, \cdots, \ell \end{aligned} \tag{9}$$

This is because the following proposition 1 holds. Note that in (9) we use different C's for different rank pairs.

Proposition 1: *The problems in* (8) *and* (9) *are equivalent, when* $C_{k(i)} = \tau_{k(i)} / 2\lambda$.

The Lagrange dual function of problem (9) is

$$L_D = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{i'=1}^{\ell} \alpha_i \alpha_{i'} z_i z_{i'} \left\langle \vec{x}_i^{(1)} - \vec{x}_i^{(2)}, \vec{x}_{i'}^{(1)} - \vec{x}_{i'}^{(2)} \right\rangle. \tag{10}$$

We maximize L_D subject to the constraints $0 \leq \alpha_i \leq C_{k(i)} \quad i = 1, \cdots, \ell$.

4 Experimental Results

We conducted three experiments: simulation, document search, and definition search.

As measures for evaluating the results of ranking methods, we used Normalized Discounted Cumulative Gain (NDCG)[16] and Mean Average Precision (MAP)[2]. They are both widely used in information retrieval. NDCG is based on the assumption that there are more than two ranks for relevance ranking while MAP is based on the assumption that there are two categories: relevant and irrelevant.

As baselines, we used Ranking SVM in all the experiments. In the document search experiment, we also compared our method with BM25 [21] and Language Model for Information Retrieval (LMIR) [20]. In the definition search experiment, we also used SVM classifier as a baseline.

One important issue for our method is to determine the values of the penalty parameters τ's. It appears to be hard to derive the ideal values of τ's analytically. In this paper we propose a heuristic method (c.f. Fig. 1) for estimating the parameter values. In experiments we used training data sets for the estimation.

4.1 Simulation Experiments

We conducted a simulation to verify the effectiveness of our method. First, we assumed that there are three ranks: r_1, r_2, and r_3, and instances in the two dimensional Euclidean space are generated according to Gaussian distributions $N(m_k, \boldsymbol{I})$. We set the centers as m_1=(0, -0.5), m_2=(0, 2), and m_3=(2, 2.5) for r_1, r_2, and r_3, respectively.

We assume that the standard deviations ***I's*** are represented by the 2×2 identity matrix. Next, we randomly generated n_k instances for each rank r_k (k =1, 2, 3) according to its distribution. We set n_1 = 1000, n_2 = 200, and n_3 = 100 (c.f., Fig. 2).

```
Method: Given a rank pair ($r_1$, $r_2$), estimate value of the penalty parameter $\tau$ for the rank
        pair.
$Drop$ = 0, $T$ = number of iterations
Query set $Q = \{q_1, q_2, \ldots, q_k\}$,
for $i$ = 1 to $k$ do
    Get document set $D_i = \{d_{i1}, d_{i2}, \ldots, d_{in}\}$ retrieved by $q_i$
    Get corresponding rank labels $L_i = \{l_{i1}, l_{i2}, \ldots, l_{in}\}$
    Create a perfect ranking for $q_i$ and calculate NDCG@1: $M_{perfect}$
    $Drop_i = 0$
    for $j$ = 1 to $T$ do
        Randomly pick up two documents $d_1$ and $d_2$ whose labels are $r_1$ and $r_2$ respectively
        Swap $d_1$ and $d_2$ and calculate NDCG@1 for the new ranking: $M_{ij}$
        $Drop_i = Drop_i + (M_{perfect} - M_{ij})$
    End for
    $Drop = Drop + (Drop_i / T)$
End for
Return $Drop / k$
```

Fig. 1. Heuristic Method for setting the penalty parameters τ 's

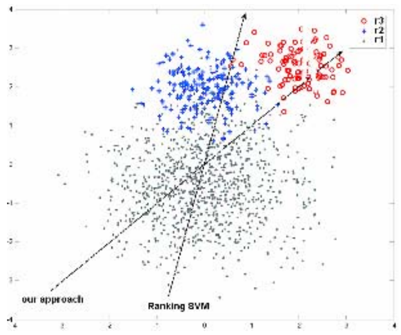


Fig. 2. Two ranking functions in simulation

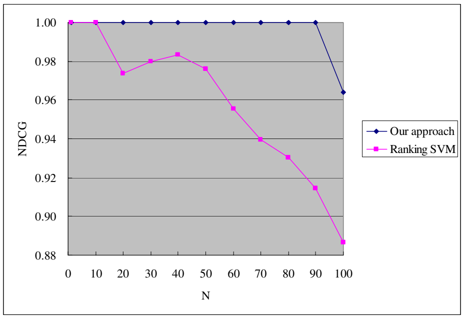


Fig. 3. NDCG curves in simulation

We applied our method and Ranking SVM to learn ranking functions from the data. In our method we set large penalty values for rank pairs r_3-r_1 and r_3-r_2, and set a small penalty value for rank pair r_2-r_1. The ranking function $f(\vec{x}) = 2.85x_1 + 3.01x_2$ was created with our method. The ranking function obtained by conventional Ranking SVM was $f(\vec{x}) = 0.53x_1 + 2.04x_2$. Fig. 2 shows the ranking functions. We calculated the NDCG values at the positions of 1, 10, 20, …, 100 and the results are given in Fig. 3. We can see that the NDCG scores of our method stay at 1.0 until when position N reaches 90. The results demonstrate that our method works better than Ranking SVM for enhancing ranking accuracy.

4.2 Experiment on Document Search

In the experiment, we tested whether our proposed methods can work well for document search. We made use of the OHSUMED collection [15]. The relevance judgments of OHSUMED are either ‘d’ (*definitely relevant*), ‘p’ (*possibly relevant*), or ‘n’ (*not relevant*). Rank ‘n’ has the largest number of documents, followed by ‘p’ and ‘d’.

Each instance consists of a vector of features, determined by a query and a document. We adopted the feature set of [19]. Table 1 shows all the features. For example, *tf* (term frequency), *idf* (inverse document frequency), document length, and their combinations are features. BM25 score is another feature, which is calculated using the ranking method of BM25[21]. For the baseline methods of BM25 and LMIR, we used the tool Lemur (http://www.lemurproject.org/).

We compared the performances of our method and Ranking SVM with the data through 4-fold cross-validation. The result reported in Fig. 4 are those averaged over four trials. From the figure, we see that our method outperforms baselines in terms of all the measures. The result indicates that our method is effective for the task of document retrieval.

We conducted Sign Test on the improvements of our method over BM25, LMIR, and Ranking SVM in terms of NDCG@1. The results indicate that the improvements are statistically significant (p-value < 0.05).

We analyzed the results and found that our method can indeed make better rankings than Ranking SVM. For example, for query 9, the top five documents returned by ranking SVM and our method are listed in Table 2 (the scores of NDCG@1 and NDCG@5 are also given). We note that both of the two ranking methods have two document pairs incorrectly ranked. Two (d, p) pairs reversed in the ranking by Ranking SVM and one (d, p) and one (p, n) pairs reversed by our method. However, the errors in our method are less hurtful, as they are not between the ranks d and p That is to say, our method can meet the requirement in practical ranking problems better.

4.3 Experiment on Definition Search

In this experiment, we verify whether our method can also be used in other information retrieval tasks such as search of definitions [23]. The problem is defined as retrieving and ranking definitions found from documents for a given query term. The key issue here is to rank definitions (or definition candidates) extracted from documents. The definitions can be in paragraphs or sentences and are extracted by patterns and rules. The definition candidates are categorized into three levels: 'good', 'indifferent', and 'bad', according to their goodness as definitions. This is another multiple ranks ranking problem. SVM and Ranking SVM are used for performing the task and it is empirically proved that both of the methods can work well. We tried to see whether it is possible to improve the results by using the cost sensitive learning method proposed in this paper.

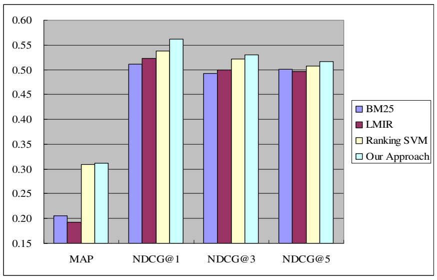
We conducted 5-fold cross validation and Fig. 5 shows the results averaged over the five trials. In the experiment, we also used SVM classifier as baseline. We did not

Table 1. Features for building ranking model. $C(w, d)$ represents the raw count of word w in document d; C represents the collection; n is the number of terms in the query; $|.|$ is the size of a set; and $idf(.)$ is the inverse document frequency.

1	$\sum_{q_i \in q \cap d} \log(c(q_i, d)+1)$	2	$\sum_{q_i \in q \cap d} \log(\frac{\lvert C\rvert}{c(q_i, C)}+1)$	3	$\sum_{q_i \in q \cap d} \log(idf(q_i))$
4	$\sum_{q_i \in q \cap d} \log(\frac{c(q_i, d)}{\lvert d\rvert}+1)$	5	$\sum_{q_i \in q \cap d} \log(\frac{c(q_i, d)}{\lvert d\rvert} \cdot idf(q_i)+1)$	6	$\sum_{q_i \in q \cap d} \log(\frac{c(q_i, d)}{\lvert d\rvert} \cdot \frac{\lvert C\rvert}{c(q_i, C)}+1)$
7	$\log(BM25\ score)$				

Table 2. Top 5 documents ranked by Ranking SVM and our method with respect to query 9

	Ranking SVM	Our Method
Top 5 ranked docs	*p* ***d d*** *p n*	***d*** *p* ***d*** *n p*
NDCG@1	0.3333	1.0
NDCG@5	0.5453	0.6238



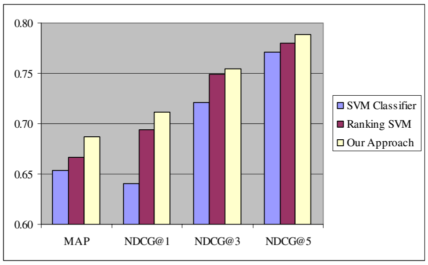


Fig. 4. Ranking accuracies in document search **Fig. 5.** Ranking accuracies in definition search

use BM25 and LMIR, because they are not suitable for conducting definition search. From Fig. 5, we conclude that our method outperforms the baseline methods of using SVM classifier and Ranking SVM. This indicates again that our method is effective for improving real ranking problems.

5 Conclusion

In the paper, we have proposed a novel cost-sensitive method to learn Support Vector Machines for ranking. We note that in many applications such as information retrieval the negative effects of making errors between higher ranks and lower ranks are much larger than making errors among lower ranks. Therefore, in learning methods for ranking, it is necessary to set up different losses for incorrectly ranking instances between different ranks. All the existing methods did not take the issue into consideration. In this paper, we take Ranking SVM as an example and have developed a new method to deal with the problem. We find that it is possible to make a generalization of the learning algorithm of Ranking SVM with a new cost-sensitive loss function. Simulation results show that our method can indeed reduce errors between higher ranks and lower ranks and thus perform better than Ranking SVM in practical settings of ranking. Experimental results verify that our method significantly outperforms Ranking SVM and other baseline methods for performing real Information Retrieval tasks.

References

1. Abe, N., Zadrozny, B., Langford, J.: An Iterative Method for Multi-class Cost-sensitive Learning. In: Proc. of 10th Inter. Conf. on KDD, Seattle, Washington, USA. (2004) 3–11
2. Baeza-Yates, R. A., Ribeiro-Neto B.: Modern Information Retrieval, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1999

3. Bradford, J., Kunz, C., Kohavi, R., Brunk, C., Brodley, C.: Pruning decision trees with misclassification costs. In: Proc. of ECML. Chemnitz, Germany. (1998) 131-136
4. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to Rank using Gradient Descent. In: Proc. of 22nd ICML. Bonn, Germany. (2005)
5. Cao, Y., Xu, J., Liu, T., Li, H., Huang, Y., Hon, H. W.: An Ordinal Regression Method for Document Retrieval. Proc. of 29th Inter. ACM SIGIR Conf., (2006), to appear
6. Chu, W., Keerthi, S.: New Approaches to Support Vector Ordinal Regression. In: Proc. of ICML, Bonn, Germany. (2005) 145–152
7. Crammer, K., Singer, Y.: PRanking with Ranking. Advances in NIPS 14, Cambridge, MA: MIT Press. (2002) 641–647
8. Domingos, P.: MetaCost : A general method for making classifiers cost sensitive. In: Proc. of the 5th KDD, San Diego, CA, USA. (1999) 155–164
9. Elkan, C.: The Foundations of Cost-Sensitive Learning. In: Proc. of the 17th Inter. Joint Conf. on Artificial Intelligence, (2001) 973–978
10. Frank, E., Hall, M.: A Simple Approach to Ordinal Classification. In: Proc. of ECML, Freiburg, Germany. (2001) 145-165
11. Geibel, P., Wysotzki, F.: Perceptron based learning with example dependent and noisy costs. In: Proc. of 12th ICML, Washington DC, USA. (2003)
12. Har-Peled, S., Roth, D., Zimak, D.: Constraint classification: A new approach to multiclass classification and ranking. Advances in NIPS 15. (2002)
13. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: data mining, inference and prediction. Springer-Verlag. (2001)
14. Herbrich, R., Graepel, T., Obermayer, K.: Large Margin Rank Boundaries for Ordinal Regression. Advances in Large Margin Classifiers. (2000) 115-132.
15. Hersh, W. R., Buckley, C., Leone, T. J., Hickam, D. H.: OHSUMED: An interactive retrieval evaluation and new large test collection for research. In: Proc. of the 17th Inter. ACM SIGIR Conf. (1994) 192-201
16. Jarvelin, K., Kekalainen, J.: IR evaluation methods for retrieving highly relevant documents. In: Proc. of the 23rd Inter. ACM SIGIR Conf. (2000) 41-48
17. Kramer, S., Widmer, G., Pfahringer, B., & Degroeve, M. (2001). Prediction of ordinal classes using regression trees. Fundamenta Informaticae, vol. 47, 1-13.
18. Monard, M. C. & Batista, G. E. A. P. A.: Learning with Skewed Class Distribution. Advances in Logic, Artificial Intelligence and Robotics, Sao Paulo, SP. (2002) 173-180
19. Nallapati, R.: Discriminative models for information retrieval. Proc. of the 27th Inter. ACM SIGIR Conf., Sheffield, United Kingdom. (2004) 64-71
20. Ponte J. and Croft W. B.: A language model approach to information retrieval. Proc. of the Inter. ACM SIGIR Conf. (1998) 275-281.
21. Robertson, S., Hull, D. A.: The TREC-9 Filtering Track Final Report. Proc. of the 9th TREC, (2000) 25-40.
22. Spink, A., Jansen B.J., Wolfram, D., Saracevic, T.: From e-sex to e-commerce: web search changes. IEEE Computer,35(3), (2002) 107-109
23. Xu, J., Cao, Y., Li, H., Zhao, M.: Ranking Definitions with Supervised Learning Methods. Proc. of the 14th Inter. Conf. on World Wide Web, (2005) 811-819
24. Zadrozny, B., Elkan, C.: Learning and making decisions when costs and probabilities are both unknown. In: Proc. of the 7th Inter. Conf. on KDD, (2001) 204–213

Variational Bayesian Dirichlet-Multinomial Allocation for Exponential Family Mixtures

Shipeng Yu[1,2], Kai Yu[2], Volker Tresp[2], and Hans-Peter Kriegel[1]

[1] Institute for Computer Science, University of Munich, Germany
[2] Siemens Corporate Technology, Munich, Germany

Abstract. This paper studies a Bayesian framework for density modeling with mixture of exponential family distributions. *Variational Bayesian Dirichlet-Multinomial allocation* (VBDMA) is introduced, which performs inference and learning efficiently using variational Bayesian methods and performs automatic model selection. The model is closely related to Dirichlet process mixture models and demonstrates similar automatic model selection in the variational Bayesian context.

1 Introduction

In statistical analysis and artificial intelligence, there has been a strong interest in finite mixture distributions for density estimation. The model offers a natural framework to handle the heterogeneity in clustering analysis, which is often of central importance in many applications. Among all the choices, exponential family mixtures are extremely useful in practice, since they cover a broader scope of characteristics of random variables, and the existence of conjugate priors often makes inference easier [1,7].

Previously much work has been done with a fixed number of components. The efforts include estimating parameters of each component by EM algorithms or via MCMC in a Bayesian way. Model selection, i.e., choosing the number of components, remains a fundamental challenge for mixture modeling. A frequentist treatment typically tests the hypotheses about this number. On the other side, a Bayesian way computes the *a posteriori* over the model space. Recently, there are increasing interests in *Bayesian nonparametric statistics*, which apply *Dirichlet process* to handle infinite number of components (e.g., [5,2]).

This paper focuses on a fully Bayesian mixture model with finite K exponential family components. The interesting point is that variational learning in the model tends to end up with a sparsity of mixing weights when K is sufficiently large. This is because the model approaches a Dirichlet process mixture model in the limiting case. A few authors explored this point in Bayesian statistics [8,9], but it is not sufficiently noticed. In this paper we propose the variational Bayesian Dirichlet-Multinomial allocation (VBDMA) for model selection in finite mixture models. This on one hand offers tractability because of the finite dimensionality and variational methods, and on the other hand provides general solutions to mixture of exponential-family distributions which covers a wide range of real-world problems.

2 Mixture of Exponential Family Distributions

Exponential Family. A probability distribution of $\mathbf{x} \in \mathcal{X}$ given parameters $\boldsymbol{\theta}$ is in the *exponential family* if it takes the form

$$P(\mathbf{x}|\boldsymbol{\theta}) = h(\mathbf{x}) \exp\left\{\boldsymbol{\theta}^\top \phi(\mathbf{x}) - A(\boldsymbol{\theta})\right\}, \tag{1}$$

where $\phi(\mathbf{x})$ is the *sufficient statistics* of $\mathbf{x}$, and $\boldsymbol{\theta}$ is called the *natural parameter*. The quantity $A(\boldsymbol{\theta})$, known as the *log partition function*, is defined as a normalization factor independent of $\mathbf{x}$: $A(\boldsymbol{\theta}) = \log \int_{\mathcal{X}} h(\mathbf{x}) \exp\left\{\boldsymbol{\theta}^\top \phi(\mathbf{x})\right\} d\mathbf{x}$. It is well-known that $A(\boldsymbol{\theta})$ plays an important role for exponential family distributions. In particular, it can be identified as the *moment generating function* of $\phi(\mathbf{x})$. One important example of this is given as:

$$\frac{\partial A(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{\theta}}[\phi(\mathbf{x})] := \int_{\mathcal{X}} \phi(\mathbf{x}) P(\mathbf{x}|\boldsymbol{\theta})\, d\mathbf{x}, \tag{2}$$

which gives the mean of the sufficient statistics.

Conjugate Family. The *conjugate family* defines a prior family for exponential family distributions as

$$P(\boldsymbol{\theta}|\boldsymbol{\gamma}, \eta) = g(\boldsymbol{\theta}) \exp\left\{\boldsymbol{\theta}^\top \boldsymbol{\gamma} - \eta A(\boldsymbol{\theta}) - B(\boldsymbol{\gamma}, \eta)\right\}, \tag{3}$$

where $(\boldsymbol{\gamma}, \eta)$ are the parameters for the prior, i.e., *hyperparameters*, with $\boldsymbol{\gamma}$ having dimensionality $\dim(\boldsymbol{\theta})$, and η a scalar. It is conjugate in that the posterior distribution takes the same form as the prior, calculated by Bayes' rule:

$$P(\boldsymbol{\theta}|\mathbf{x}, \boldsymbol{\gamma}, \eta) \propto P(\mathbf{x}|\boldsymbol{\theta}) P(\boldsymbol{\theta}|\boldsymbol{\gamma}, \eta) \propto g(\boldsymbol{\theta}) \exp\left\{\boldsymbol{\theta}^\top (\boldsymbol{\gamma} + \phi(\mathbf{x})) - (\eta + 1) A(\boldsymbol{\theta}) - B(\boldsymbol{\gamma}, \eta)\right\}.$$

It is easy to check that conjugate family (3) also belongs to exponential family, with sufficient statistics $\binom{\boldsymbol{\theta}}{-A(\boldsymbol{\theta})}$ and natural parameter $\binom{\boldsymbol{\gamma}}{\eta}$. Then we have

$$\frac{\partial B(\boldsymbol{\gamma}, \eta)}{\partial \boldsymbol{\gamma}} = \mathbb{E}_{\boldsymbol{\gamma}, \eta}[\boldsymbol{\theta}], \quad \frac{\partial B(\boldsymbol{\gamma}, \eta)}{\partial \eta} = \mathbb{E}_{\boldsymbol{\gamma}, \eta}[-A(\boldsymbol{\theta})] \tag{4}$$

by applying (2). These results turn out to be useful for subsequent sections.

Exponential Family Mixtures. In mixture modeling, each data point is sampled from a fixed but unknown *component distribution*, which belongs to exponential family here. At the moment we fix the number of components in the mixture to be K, a finite positive integer. We will focus on the case that all the component distributions take the same form, e.g., Gaussian. Then the likelihood given N *i.i.d.* sampled data points $\mathcal{D} := \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is formally written as

$$P(\mathcal{D}|\boldsymbol{\pi}, \boldsymbol{\Theta}) = \textstyle\prod_{i=1}^N \sum_{k=1}^K P(c_i = k|\boldsymbol{\pi}) P(\mathbf{x}_i|\boldsymbol{\theta}_k),$$

where $P(c_i = k|\boldsymbol{\pi}) = \pi_k$ is a Multinomial with parameters $\boldsymbol{\pi}$, and $P(\mathbf{x}_i|\boldsymbol{\theta}_k)$ takes the general form (1). The K-dimensional vector $\boldsymbol{\pi} := \{\pi_k\}_{k=1}^K$ gives the weights for the component distributions and sums to 1. $\boldsymbol{\Theta} := \{\boldsymbol{\theta}_k\}_{k=1}^K$ contain the natural parameters of all component distributions. c_i is seen as a random variable of *indicator* for data $\mathbf{x}_i$, saying which component $\mathbf{x}_i$ is sampled from.

We need to assign priors to all the parameters. For $\boldsymbol{\Theta}$ we assign conjugate prior (3) to each $\boldsymbol{\theta}_k$ independently, with the same hyperparameters $(\boldsymbol{\gamma}_0, \eta_0)$: $P(\boldsymbol{\theta}_k|\boldsymbol{\gamma}_0, \eta_0) = g(\boldsymbol{\theta}_k) \exp\left\{\boldsymbol{\theta}_k^\top \boldsymbol{\gamma}_0 - \eta_0 A(\boldsymbol{\theta}_k) - B(\boldsymbol{\gamma}_0, \eta_0)\right\}$. For the Multinomial parameters $\boldsymbol{\pi}$ we assign a Dirichlet distribution $\boldsymbol{\pi} \sim \mathrm{Dir}(\frac{\alpha}{K}, \ldots, \frac{\alpha}{K})$. Here we make the constraint that all the parameters in this Dirichlet are the same and sum to a scalar that is independent of K, the number of components.

With these priors, the final data likelihood can be obtained by integrating out *latent variables* $\boldsymbol{\pi}$ and $\boldsymbol{\Theta}$ (see plate model in Fig. 1 left):

$$P(\mathcal{D}) = \int_{\boldsymbol{\pi}} P(\boldsymbol{\pi}|\alpha) \int_{\boldsymbol{\Theta}} \prod_{k=1}^K P(\boldsymbol{\theta}_k|\boldsymbol{\gamma}_0, \eta_0) \left\{\prod_{i=1}^N \sum_{k=1}^K \pi_k P(\mathbf{x}_i|\boldsymbol{\theta}_k)\right\} d\boldsymbol{\Theta} d\boldsymbol{\pi}.$$

The model has two parameters: α is a positive scalar; $(\boldsymbol{\gamma}_0, \eta_0)$ has dimensionality $\dim(\phi(\mathbf{x})) + 1$. [1,4] studied the special case of Gaussian mixtures.

3 Model Inference and Learning

Inference in the proposed model is intractable and needs Markov chain Monte Carlo (MCMC) sampling. In this paper we instead focus on *variational Bayesian* methods, which are motivated by approximating the *a posteriori* distribution of latent variables with a tractable family, and then maximizing a lower-bound of data likelihood with respect to some *variational parameters* [10,7]. One common way of achieving this is to assume a *factorized* distribution for the latent variables, which indicates that for exponential family mixtures we use distribution

$$Q(\boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{c}|\boldsymbol{\lambda}, \boldsymbol{\gamma}, \boldsymbol{\eta}, \boldsymbol{\varphi}) := Q(\boldsymbol{\pi}|\boldsymbol{\lambda}) \prod_{k=1}^K Q(\boldsymbol{\theta}_k|\boldsymbol{\gamma}_k, \eta_k) \prod_{i=1}^N Q(c_i|\boldsymbol{\varphi}_i)$$

to approximate the true posterior $P(\boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{c}|\mathcal{D}, \alpha, \boldsymbol{\gamma}_0, \eta_0)$. Here $Q(\boldsymbol{\pi}|\boldsymbol{\lambda})$ is K-dim. Dirichlet, $Q(\boldsymbol{\theta}_k|\boldsymbol{\gamma}_k, \eta_k)$ the conjugate family (3), and $Q(c_i|\boldsymbol{\varphi}_i)$ K-dim. Multinomial. Applying Jensen's inequality yields a lower bound of the log-likelihood: $\mathcal{L}(\mathcal{D}) = \mathbb{E}_Q[\log P(\boldsymbol{\pi}|\alpha)] + \sum_{k=1}^K \mathbb{E}_Q[\log P(\boldsymbol{\theta}_k|\boldsymbol{\gamma}_0, \eta_0)] + \sum_{i=1}^N \mathbb{E}_Q[\log P(c_i|\boldsymbol{\pi})] + \sum_{i=1}^N \mathbb{E}_Q[\log P(\mathbf{x}_i|\boldsymbol{\theta}, c_i)] - \mathbb{E}_Q[\log Q(\boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{c})]$. Variational Bayesian methods in the literature maximize this lower bound *only* with respect to variational parameters $\boldsymbol{\lambda}, \boldsymbol{\gamma}, \boldsymbol{\eta}, \boldsymbol{\varphi}$, and thus fix the model parameters $\alpha, \boldsymbol{\gamma}_0, \eta_0$ (see [1,7]). This paper will however treat it as the E-step of the algorithm, and estimate the model parameters in the M-step.

In the E-step, it is straightforward to obtain the following updates by setting the partial derivatives with respect to each variational parameter to be zero:

$$\varphi_{i,k} \propto \exp\left\{\mathbb{E}_{\boldsymbol{\gamma}_k, \eta_k}[\boldsymbol{\theta}_k^\top \phi(\mathbf{x}_i) - A(\boldsymbol{\theta}_k)] + \mathbb{E}_{\boldsymbol{\lambda}}[\log \pi_k]\right\}, \tag{5}$$

$$\boldsymbol{\gamma}_k = \sum_{i=1}^N \varphi_{i,k} \phi(\mathbf{x}_i) + \boldsymbol{\gamma}_0, \quad \eta_k = \sum_{i=1}^N \varphi_{i,k} + \eta_0, \quad \lambda_k = \sum_{i=1}^N \varphi_{i,k} + \frac{\alpha}{K}, \tag{6}$$

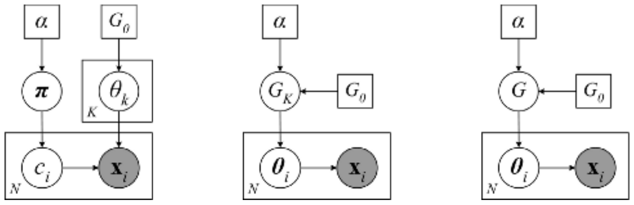


Fig. 1. Plate models for exponential family finite mixtures (left and middle), and the DP mixture model (right). G_K denotes the finite discrete prior for $\boldsymbol{\theta}_k$'s.

where the first expectation in (5) can be calculated using (4), and $\mathbb{E}_{\boldsymbol{\lambda}}[\log \pi_k] = \left\{\Psi(\lambda_k) - \Psi\left(\sum_{j=1}^{K} \lambda_j\right)\right\}$, with $\Psi(\cdot)$ the digamma function. This expectation is obtained by applying (2) to Dirichlet distribution $Q(\boldsymbol{\pi}|\boldsymbol{\lambda})$. Since these equations are coupled, they should be updated iteratively until convergence. In variational Bayes, (5) is called *variational E-step*, and (6) is called *variational M-step.* This yields the algorithm given in [1] for mixture of Gaussians.

These equations recover the theorem in [7] for exponential family mixture models, and turn out to be very intuitive and explainable. For instance, $\varphi_{i,k}$ measures the *a posteriori* probability that data $\mathbf{x}_i$ comes from component k, and can be written from (5) as $\varphi_{i,k} \propto \exp\left\{\mathbb{E}_{\boldsymbol{\gamma}_k,\eta_k}[\log P(\mathbf{x}_i|\boldsymbol{\theta}_k)]\right\} \exp\left\{\mathbb{E}_{\boldsymbol{\lambda}}[\log \pi_k]\right\}$ which can be seen as a *likelihood* term (left term) multiplied by a *prior* (right term), with other parameters fixed. This is analogous to a direct application of Bayes' rule. Other updates (6) also combine the *empirical observations* (the sum terms) with the prior (the model parameters).

In the M-step, we maximize the lower-bound with respect to the model parameters. For α we obtain $\sum_{k=1}^{K} \mathbb{E}_{\alpha}[\log \pi_k] = \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{\lambda}}[\log \pi_k]$, which turns out to match the *sufficient statistics* of Dirichlet distributions. Similar results hold for $\boldsymbol{\gamma}_0$ and η_0: $\sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{\gamma}_0,\eta_0}[\boldsymbol{\theta}_k] = \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{\gamma}_k,\eta_k}[\boldsymbol{\theta}_k]$, $\sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{\gamma}_0,\eta_0}[-A(\boldsymbol{\theta}_k)] = \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{\gamma}_k,\eta_k}[-A(\boldsymbol{\theta}_k)]$. These expectations can be calculated using (4). Analytical solutions for these equations are in general not obtainable, so we need computational methods such as Newton-Raphson method to solve the problem.

4 Variational Bayesian Dirichlet-Multinomial Allocation

Model selection for mixture modeling, i.e., choosing the number K, is an important problem. This can be done via cross-validation; a Bayesian way selects the model with the largest *a posteriori* likelihood. However in both cases we have to retrain the model with different K's, which is normally very expensive.

In this section we investigate the functionality of α and show that the learning algorithm in Sec. 3 can lead to sparse mixtures. The algorithm has strong connections to *Dirichlet process* (DP) [6], and can be viewed as a variational algorithm for inference in DP mixture models. Therefore we call it *variational Bayesian Dirichlet-Multinomial allocation* (VBDMA), and it turns out that K can be automatically obtained after training, with the sparsity controlled by α.

Connections to Dirichlet Process. Denote $\boldsymbol{\theta}$ the natural parameter that generates data $\mathbf{x}$. In the mixture model we see that $\boldsymbol{\theta}$ is sampled from distribution $G_K(\boldsymbol{\theta}) := P(\boldsymbol{\theta}|\boldsymbol{\pi}, \boldsymbol{\Theta}) = \sum_{k=1}^{K} \pi_k \delta_{\boldsymbol{\theta}_k}(\boldsymbol{\theta})$, where $\delta_{\boldsymbol{\theta}_k}(\boldsymbol{\theta})$ is the point mass distribution and takes value 1 for $\boldsymbol{\theta} = \boldsymbol{\theta}_k$ and 0 otherwise. $G_K(\cdot)$ defines a *discrete prior* for $\boldsymbol{\theta}$, and model parameters α and $(\boldsymbol{\gamma}_0, \eta_0)$ now take the role of tuning the discrete but *unknown* distribution $G_K(\cdot)$.

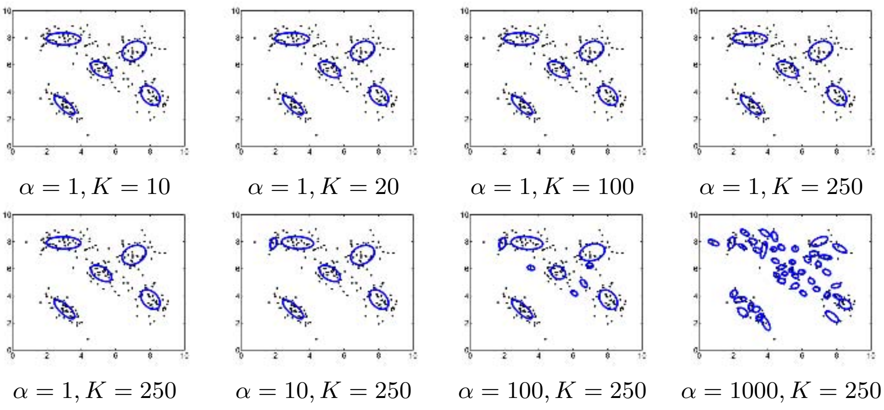
When we let $K \to \infty$, it is known in statistics that the unknown distribution G_K tends to be a sample from a Dirichlet process, constrained by the *concentration parameter* (a positive scalar) and a *base distribution* [11]. In our model, the concentration parameter is just α, and the base distribution G_0 is given by (3). This model is illustrated in Fig. 1 middle. Following the convention for Dirichlet process, all the parameters $\boldsymbol{\theta}_i$ for data $\mathbf{x}_i$ are sampled as:

$$\boldsymbol{\theta}_i \overset{\text{iid}}{\sim} G, \quad \text{for } i = 1, \ldots, N; \qquad G \sim \text{DP}(\alpha, G_0).$$

Dirichlet process is well-known for the property of obtaining a nonparametric and discrete prior, and thus is widely applied for mixture modeling (see, e.g., [9]). When K is finite, however, the model is not equivalent to defining a Dirichlet process prior for $\boldsymbol{\theta}_i$'s, but is shown to be a good approximation if K is sufficiently large. This finite approximation is sometimes referred to as *Dirichlet-Multinomial allocation* (DMA), and is used for approximated sampling for Dirichlet processes [8]. In both DP and DMA, model selection can be done automatically via sampling methods, and the concentration parameter α is known to control the flexibility of generating new mixture components.

Sparsity of Infinite Mixture. Let us first fix α and focus on the E-step (5)$\sim$(6). With an uninformative initialization of variational parameters (e.g., we choose $\boldsymbol{\gamma}_k = \boldsymbol{\gamma}_0$, $\eta_k = \eta_0$ and $\lambda_k = \alpha/K$, for all k), we first fit the mixture membership $\varphi_{i,k}$ from (5), and then update the Dirichlet parameters using (6). Since all the components have the same prior terms $\mathbb{E}_{\boldsymbol{\lambda}}[\log \pi_k]$ initially, in (5) the assignment probabilities $\varphi_{i,k}$ will solely depend on the *empirical explanation* of $\mathbf{x}_i$ given component parameter $\boldsymbol{\theta}_k$. This will make the updated $\varphi_{i,k}$ unevenly distributed, and the constraints $\sum_k \varphi_{i,k} = 1, \forall i$ will lead to some "unlikely" components with very small assignment probabilities, i.e., $\sum_{i=1}^{N} \varphi_{i,k}$. When these values are fed into (6), these components will get smaller values for λ_k, and thus the prior term $\mathbb{E}_{\boldsymbol{\lambda}}[\log \pi_k]$ in (5) will also get smaller, which makes $\varphi_{i,k}$ more sharply distributed. Eventually, these components will get $\varphi_{i,k} = 0$, for all data points $\mathbf{x}_i$. This in turn leads to $\boldsymbol{\gamma}_k = \boldsymbol{\gamma}_0$, $\eta_k = \eta_0$ and $\lambda_k = \alpha/K$, all equal to the hyperparameters. When K is very large, α/K is very small, and these components almost have no chance to get bigger $\varphi_{i,k}$ in the future for some data $\mathbf{x}_i$, as seen from (5). Finally when the algorithm converges, we obtain only a small number of *effective* components.

This phenomenon is illustrated in Fig. 2 (upper row) for Gaussian mixtures, where we sampled 250 data points from 5 Gaussians. When we fix $\alpha = 1$, sparsity is obtained for all K's, even if K is only 10. When K becomes larger, the fitted number does not vary, but tends to be stable. As will be seen next, the strength of sparsity is not random, but depends strongly on parameter α.


$\alpha = 1, K = 10$ $\alpha = 1, K = 20$ $\alpha = 1, K = 100$ $\alpha = 1, K = 250$

$\alpha = 1, K = 250$ $\alpha = 10, K = 250$ $\alpha = 100, K = 250$ $\alpha = 1000, K = 250$

Fig. 2. Fitting VBDMA on a toy Gaussian mixture data with different α and K values

Functionality of $\boldsymbol{\alpha}$. Now we investigate the situation that K is fixed, and α is allowed to change. When α is small, updates for λ_k will mostly depend on the empirical assignments $\sum_{i=1}^{N} \varphi_{i,k}$ in (6), and thus quickly get unbalanced. Then similar to the previous discussion, $\varphi_{i,k}$ will get an even sharp distribution in the next update, and the algorithm quickly converges to a small number of components that fit the data best. In the limiting case that $\alpha = 0$, λ_k's are purely determined by empirical updates, and we are making a *maximum likelihood* estimate for the mixing weights $\boldsymbol{\pi}$.

On the other hand when α is relatively large, the prior term α/K will dominate the update equation (6), and thus λ_k will not be very unbalanced in one step. This will in turn make the update equation (5) smooth for $\varphi_{i,k}$, and more components will survive than that with small α. As the iteration continues, certainly some components will be "dead" because of their poor fit to the data, but the death rate is much slower and we could expect more components left after convergence. A limiting case for this is to let $\alpha \to \infty$, which corresponds to fix the $\boldsymbol{\pi}$ *a priori* to be $\{\frac{1}{K}, \ldots, \frac{1}{K}\}$, and does not change it in the whole learning process. This normally leads to non-sparsity of the learned model.

Fig. 2 (bottom row) shows how α controls the sparsity of mixture modeling. With K fixed as 250, smaller α (e.g., 1) leads to higher sparsity, and larger α (e.g., 1000) results in more components. Therefore choosing a suitable α means choosing a desired number of mixture components.

Discussions. Previous discussions suggest that the algorithm in Sec. 3 can be viewed as a variational algorithm for DP mixture model, which we call the VBDMA. A nice property of VBDMA is that decrease of K is a natural consequence of model fitting with the data, and can be controlled by α. This is in contrast to post-processing methods (e.g., [12]) where heuristics must be used. VBDMA also provides explanations to [4], and can be extended to more complicated mixture models like mixture of factor analyzers [7].

Table 1. The number of learned mixture components (means and standard deviations) in VBDMA (top) and VBTDP (bottom) for the toy Gaussian data with different initial K and α values. The experiments are repeated 20 times independently.

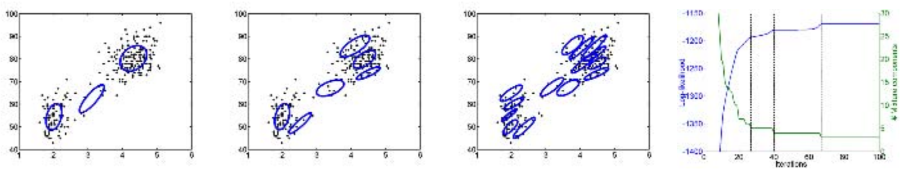
	$K = 5$	$K = 10$	$K = 20$	$K = 50$	$K = 100$	$K = 250$
$\alpha = 1$	4.45 ± 0.60	6.00 ± 1.03	6.70 ± 0.86	7.15 ± 1.27	6.85 ± 1.42	6.25 ± 1.16
$\alpha = 10$	4.95 ± 0.22	7.80 ± 1.01	8.65 ± 1.14	7.35 ± 1.04	7.10 ± 1.37	6.45 ± 1.10
$\alpha = 100$	5.00 ± 0.00	10.00 ± 0.00	19.90 ± 0.31	21.20 ± 1.58	11.40 ± 1.76	7.80 ± 1.40
$\alpha = 1000$	5.00 ± 0.00	10.00 ± 0.00	20.00 ± 0.00	49.65 ± 0.49	69.05 ± 2.19	45.05 ± 2.06
$\alpha = 10000$	5.00 ± 0.00	10.00 ± 0.00	20.00 ± 0.00	49.90 ± 0.31	85.10 ± 2.47	87.75 ± 2.07

	$K = 5$	$K = 10$	$K = 20$	$K = 50$	$K = 100$	$K = 250$
$\alpha = 1$	4.50 ± 0.61	6.30 ± 1.03	7.35 ± 1.46	8.15 ± 1.39	8.55 ± 1.23	9.00 ± 1.62
$\alpha = 10$	4.65 ± 0.49	6.75 ± 0.91	7.85 ± 1.14	8.50 ± 1.24	8.80 ± 1.32	9.15 ± 1.09
$\alpha = 100$	4.60 ± 0.60	7.55 ± 1.15	8.95 ± 1.79	9.60 ± 1.70	9.90 ± 1.21	10.10 ± 1.33
$\alpha = 1000$	4.65 ± 0.49	7.80 ± 1.01	10.45 ± 1.47	10.80 ± 2.07	11.15 ± 2.06	11.10 ± 2.31
$\alpha = 10000$	4.60 ± 0.50	7.75 ± 1.02	10.20 ± 1.32	11.05 ± 2.01	11.50 ± 1.82	11.40 ± 2.19

Another variational algorithm for DP is proposed in [2] which is based on truncated DP (we denote it VBTDP). The idea is similar to VBDMA, but they put variational distributions directly on the stick-breaking parameters (see the definition in [9]). It is known that the variational form in VBTDP induces a *generalized Dirichlet distribution* to weights $\boldsymbol{\pi}$, and uses twice as many parameters as a Dirichlet distribution [3]. Some properties of generalized Dirichlet distribution include that each dimension of $\boldsymbol{\pi}$ is not always *negatively correlated* to other dimensions (i.e,. observing a sample from one dimension will surely increase the expected value of the parameter for this dimension, but decrease those for the other dimensions) as in Dirichlet distribution, and that the order of these dimensions is important for sampling and learning [13]. Both properties are however unnecessary for mixture modeling, and the latter is even contradictory to Bayesian exchangeability in this context.

In Tab. 1 we show the numbers of learned components for VBDMA and VBTDP on the toy data with different α and K values. For both methods, increasing α leads to more components, and sparsity is achieved for all K's when α is small. However, while varying α yields quite different sparsity for VBDMA, in VBTDP α seems to be insensitive to the results. Please refer to [14] for more detailed discussion about these two methods.

Empirical Study. Due to space limit we only consider the VBDMA with Gaussian mixtures on the "Old Faithful" data set. For more results on real data sets please refer to [14]. "Old Faithful" contains 272 2D observations from the Old Faithful Geyser in the Yellowstone National Park. Each observation consists of the duration of the eruption and the waiting time to the next eruption. We set K to 272 initially, and setting α to 100, 500 and 1000 results in 3, 6 and 15 Gaussians, respectively (see Fig. 3). All of them fit the data well, but in different granularities. The final log likelihoods of the three model fitting are -1174.55, -1187.75 and -1253.17, respectively. One can do a model selection using this likelihood and prefer the first one, but now there is no need to choose K *a priori* because this number is automatically determined by the VBDMA



$\alpha = 100, K = 272$ $\alpha = 500, K = 272$ $\alpha = 1000, K = 272$ $\alpha = 100$

Fig. 3. Fitting a mixture of Gaussians on the "Old Faithful" data set

algorithm with a learned α which is approximately 100. We also see that each time the effective mixture number decreases, the likelihood has a noticeable increase (we mark three of them using dashed lines).

References

1. H. Attias. A variational Bayesian framework for graphical models. In *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.
2. D. M. Blei and M. I. Jordan. Variational methods for the Dirichlet process. Proceedings of the 21st International Conference on Machine Learning, 2004.
3. R. J. Connor and J. E. Mosimann. Concepts of independence for proportions with a generalization of the Dirichlet distribution. *J. Amer. Stat. Ass.*, 64, 1969.
4. A. Corduneanu and C. M. Bishop. Variational Bayesian model selection for mixture distributions. In *Workshop AI and Statistics*, pages 27–34, 2001.
5. M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430), June 1995.
6. T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1:209–230, 1973.
7. Z. Ghahramani and M. J. Beal. Graphical models and variational methods. In *Advanced mean Field Methods — Theory and Practice*. MIT Press, 2000.
8. P. J. Green and S. Richardson. Modelling heterogeneity with and without the Dirichlet process. unpublished paper, 2000.
9. H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173, 2001.
10. M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.
11. R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9:249–265, 2000.
12. N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton. SMEM algorithm for mixture models. In *Neural Computation*, 1999.
13. T.-T. Wong. Generalized Dirichlet distribution in Bayesian analysis. *Appl. Math. Comput.*, 97(2-3):165–181, 1998.
14. S. Yu. *Advanced Probabilistic Models for Clustering and Projection*. PhD thesis, University of Munich, 2006.

Author Index