

LNAI 6925

Jyrki Kivinen
Csaba Szepesvári
Esko Ukkonen
Thomas Zeugmann (Eds.)

Algorithmic Learning Theory

22nd International Conference, ALT 2011
Espoo, Finland, October 2011
Proceedings



LNAI Series Editors

Randy Goebel

University of Alberta, Edmonton, Canada

Yuzuru Tanaka

Hokkaido University, Sapporo, Japan

Wolfgang Wahlster

DFKI and Saarland University, Saarbrücken, Germany

LNAI Founding Series Editor

Joerg Siekmann

DFKI and Saarland University, Saarbrücken, Germany

Jyrki Kivinen Csaba Szepesvari
Esko Ukkonen Thomas Zeugmann (Eds.)

Algorithmic Learning Theory

22nd International Conference, ALT 2011
Espoo, Finland, October 5-7, 2011
Proceedings

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada

Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Jyrki Kivinen

University of Helsinki, Department of Computer Science

P.O. Box 68 (Gustaf Hällströmin katu 2b), 00014 Helsinki, Finland

E-mail: jyrki.kivinen@cs.helsinki.fi

Csaba Szepesvári

University of Alberta, Department of Computing Science

Edmonton, AB, T6G 2E8, Canada

E-mail: szepesva@ualberta.ca

Esko Ukkonen

University of Helsinki, Department of Computer Science

P.O. Box 68 (Gustaf Hällströmin katu 2b), 00014 Helsinki, Finland

E-mail: ukkonen@cs.helsinki.fi

Thomas Zeugmann

Hokkaido University, Division of Computer Science

N-14, W-9, Sapporo 060-0814, Japan

E-mail: thomas@ist.hokudai.ac.jp

ISSN 0302-9743

e-ISSN 1611-3349

ISBN 978-3-642-24411-7

e-ISBN 978-3-642-24412-4

DOI 10.1007/978-3-642-24412-4

Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011936965

CR Subject Classification (1998): I.2, F.4.1, F.1, F.2, I.2.3, I.2.6

LNCS Sublibrary: SL 7 – Artificial Intelligence

© Springer-Verlag Berlin Heidelberg 2011

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains the papers presented at the 22nd International Conference on Algorithmic Learning Theory (ALT 2011), which was held in Espoo, Finland, October 5–7, 2011. The conference was co-located with the 14th International Conference on Discovery Science (DS 2011). The technical program of ALT 2011 contained 28 papers selected from 61 submissions, and 5 invited talks. The invited talks were presented in joint sessions at both conferences.

ALT 2011 was dedicated to the theoretical foundations of machine learning and took place on the campus of Aalto University, Espoo, Finland. ALT provides a forum for high-quality talks with a strong theoretical background and scientific interchange in areas such as inductive inference, universal prediction, teaching models, grammatical inference, formal languages, query learning, complexity of learning, on-line learning and relative loss bounds, semi-supervised and unsupervised learning, clustering, active learning, statistical learning, regression, bandit problems, support vector machines, Vapnik-Chervonenkis dimension, probably approximately correct learning, Bayesian and causal networks, boosting and bagging, information-based methods, minimum description length, Kolmogorov complexity, kernels, graph learning, decision tree methods, Markov decision processes, reinforcement learning, intelligent agents, and real-world applications of algorithmic learning theory.

DS 2011 was the 14th International Conference on Discovery Science and focused on the development and analysis of methods for intelligent data analysis, knowledge discovery and machine learning, as well as their application to scientific knowledge discovery. Traditionally, it was co-located and held in parallel with Algorithmic Learning Theory.

The present volume contains the texts of the 28 papers presented at ALT 2011, divided into groups of papers on inductive inference, regression, bandit problems, online learning, kernels and margin-based methods, intelligent agents, and on other learning models. The volume also contains the texts or abstracts of the invited talks:

- Peter Auer (Montanuniversität Leoben, Austria), “Models for Autonomously Motivated Exploration in Reinforcement Learning” (invited speaker for ALT 2011)
- Yoshua Bengio (Université de Montréal, Canada), “On the Expressive Power of Deep Architectures” (joint invited speaker for ALT 2011 and DS 2011)
- Jorma Rissanen (Helsinki Institute for Information Technology, Finland), “Optimal Estimation” (invited speaker for ALT 2011)
- Eyke Hüllermeier jointly with Johannes Fürnkranz (Philipps-Universität Marburg, Germany, and Technische Universität Darmstadt, Germany, respectively), “Learning from Label Preferences” (invited speaker for DS 2011)
- Ming Li (University of Waterloo, Canada), “Information Distance and Its Extensions” (invited speaker for DS 2011).

Papers presented at DS 2011 are contained in the DS 2011 proceedings.

Since 1999, ALT has been awarding the *E. M. Gold Award* for the most outstanding student contribution. This year, the award was given to Malte Darnstt for his paper “Supervised Learning and Co-training,” co-authored with Hans Ulrich Simon and Balzs Szrenyi.

ALT 2011 was the 22nd in the ALT conference series, established in Japan in 1990. A second root is the conference series Analogical and Inductive Inference, previously held in 1986, 1989, 1992, which merged with the conference series ALT after a collocation in the year 1994. From then on, ALT became an international conference series which kept its strong links to Japan but was also regularly held in other destinations including Australia, Germany, Hungary, Italy, Portugal, Singapore, Spain, the USA, and Finland.

The ALT series was supervised by its Steering Committee: Naoki Abe (IBM Thomas J. Watson Research Center, Yorktown, USA), Shai Ben-David (University of Waterloo, Canada), Jyrki Kivinen (University of Helsinki, Finland), Philip M. Long (Google, Mountain View, USA), Akira Maruoka (Ishinomaki Senshu University, Japan), Takeshi Shinohara (Kyushu Institute of Technology, Iizuka, Japan), Frank Stephan (National University of Singapore, Republic of Singapore), Einoshin Suzuki (Kyushu University, Fukuoka, Japan), Csaba Szepesvri (University of Alberta, Canada), Eiji Takimoto (Kyushu University, Fukuoka, Japan), Gyrgy Turn (University of Illinois at Chicago, USA and University of Szeged, Hungary), Osamu Watanabe (Tokyo Institute of Technology, Japan), Thomas Zeugmann (Chair, Hokkaido University, Japan), and Sandra Zilles (Publicity Chair, University of Regina, Saskatchewan, Canada).

We would like to thank the many people and institutions who contributed to the success of the conference. In particular, we want to thank our authors for contributing to the conference and for coming to Espoo in October 2011. Without their efforts and their willingness to choose ALT 2011 as a forum to report on their research, this conference would not have been possible.

We would like to thank the Aalto University, School of Science, Department of Information and Computer Science, the University of Helsinki, Department of Computer Science, the Helsinki Institute for Information Technology, and Algodan – Finnish Centre of Excellence for Algorithmic Data Analysis Research for generously sponsoring the conference.

We are furthermore grateful to Aalto University for hosting the event. The support of Aalto University, the University of Helsinki, the Helsinki Institute for Information Technology and Algodan was a great help, organizationally and financially, for the ALT 2011 and DS 2011 conferences.

We also thank the journal *Artificial Intelligence* for its generous financial support of ALT 2011 and DS 2011.

We are also grateful that we could use the excellent conference management system EasyChair for putting together the program for ALT 2011; EasyChair was developed mainly by Andrei Voronkov and is hosted at the University of Manchester. The system is cost-free.

The conference series ALT was this year, as in many previous years, co-located with the series Discovery Science. We are grateful for this continuous collaboration. In particular, we would like to thank the Conference Chair Heikki Mannila and the Program Committee Chairs Tapio Elomaa and Jaakko Hollmén of Discovery Science 2011.

We would like to thank Olli Simula for organizing the conference and the tremendous amount of work he put into making ALT 2011 a success. We want to extend our thanks to the other members of the local Organizing Committee, who were there to organize the reception, to sit at the information desk and to carry out the other duties connected to organizing and hosting a conference.

We are grateful to the members of the ALT 2011 Program Committee and the subreferees for their hard work in selecting a good program for ALT 2011. Reviewing papers and checking the correctness of results is demanding in time and skills and we very much appreciate this contribution to the conference. Last but not least we thank Springer for their support in preparing and publishing this volume of the *Lecture Notes in Artificial Intelligence* series.

July 2011

Jyrki Kivinen
Csaba Szepesvári
Esko Ukkonen
Thomas Zeugmann

Organization

Conference Chair

Esko Ukkonen

University of Helsinki, Finland

Program Committee

Dana Angluin	Yale University, USA
Jean-Yves Audibert	Université Paris Est, France
Shai Ben-David	University of Waterloo, Canada
Avrim Blum	Carnegie Mellon University, USA
Nader Bshouty	Technion, Israel
John Case	University of Delaware, USA
Ricard Gavaldà	Universitat Politècnica de Catalunya, Spain
András György	Hungarian Academy of Sciences, Hungary
Sanjay Jain	National University of Singapore
Jyrki Kivinen (Chair)	University of Helsinki, Finland
Gábor Lugosi	Pompeu Fabra University, Spain
Rémi Munos	Institut National de Recherche en Informatique et en Automatique Lille, France
Ronald Ortner	University of Leoben, Austria
John Shawe-Taylor	University College London, UK
Hans Ulrich Simon	Ruhr-Universität Bochum, Germany
Frank Stephan	National University of Singapore
Csaba Szepesvári (Chair)	University of Alberta, Edmonton, Canada
Eiji Takimoto	Kyushu University, Japan
Vladimir Vovk	Royal Holloway, University of London, UK
Akihiro Yamamoto	Kyoto University, Japan
Sandra Zilles	University of Regina, Canada

Local Arrangements Chair

Olli Simula

Aalto University, Espoo, Finland

Subreferees

Jacob Abernethy	Robert Kleinberg
Margareta Ackerman	Timo Kötzing
Andras Antos	Mikhail Langovoy
Peter Auer	Tor Lattimore
Maria-Florina Balcan	François Laviolette
Andras A. Benczur	Alessandro Lazaric
Alina Beygelzimer	Shane Legg
Antoine Bordes	Guy Lever
Sébastien Bubeck	Julien Mairal
Christos Dimitrakakis	Andre Martins
Frank Drewes	David McAllester
Aurelien Garivier	Gergely Neu
Claudio Gentile	Alexandru Niculescu-Mizil
Oded Goldreich	Daniel Reidenbach
Peter Grünwald	Afshin Rostamizadeh
Joe Halpern	Cynthia Rudin
Zaid Harchaoui	Daniil Ryabko
Kohei Hatano	Yevgeny Seldin
Daniel Hsu	Ambuj Tewari
Rodolphe Jenatton	Peter Torma
Hachem Kadri	Jia Yuan Yu
Yuri Kalnishkan	Thomas Zeugmann

Sponsoring Institutions

Aalto University, School of Science, Department of Information and Computer Science
University of Helsinki, Department of Computer Science
Helsinki Institute for Information Technology
Algordan – Finish Centre of Excellence for Algorithmic Data Analysis Research
The journal *Artificial Intelligence*

Table of Contents

Editors' Introduction	1
<i>Jyrki Kivinen, Csaba Szepesvari, Esko Ukkonen, and Thomas Zeugmann</i>	

Invited Papers

Models for Autonomously Motivated Exploration in Reinforcement Learning (Extended Abstract)	14
<i>Peter Auer, Shiao Hong Lim, and Chris Watkins</i>	
On the Expressive Power of Deep Architectures	18
<i>Yoshua Bengio and Olivier Delalleau</i>	
Optimal Estimation	37
<i>Jorma Rissanen</i>	
Learning from Label Preferences	38
<i>Eyke Hllermeier and Johannes Frnkranz</i>	
Information Distance and Its Extensions	39
<i>Ming Li</i>	

Inductive Inference

Iterative Learning from Positive Data and Counters	40
<i>Timo Ktzing</i>	
Robust Learning of Automatic Classes of Languages	55
<i>Sanjay Jain, Eric Martin, and Frank Stephan</i>	
Learning and Classifying	70
<i>Sanjay Jain, Eric Martin, and Frank Stephan</i>	
Learning Relational Patterns	84
<i>Michael Geilke and Sandra Zilles</i>	

Regression

Adaptive and Optimal Online Linear Regression on ℓ^1 -Balls	99
<i>S�bastien Gerchinovitz and Jia Yuan Yu</i>	

Re-adapting the Regularization of Weights for Non-stationary Regression	114
<i>Nina Vaits and Koby Crammer</i>	
Competing against the Best Nearest Neighbor Filter in Regression	129
<i>Arnak S. Dalalyan and Joseph Salmon</i>	

Bandit Problems

Lipschitz Bandits without the Lipschitz Constant	144
<i>Sébastien Bubeck, Gilles Stoltz, and Jia Yuan Yu</i>	
Deviations of Stochastic Bandit Regret	159
<i>Antoine Salomon and Jean-Yves Audibert</i>	
On Upper-Confidence Bound Policies for Switching Bandit Problems ...	174
<i>Aurélien Garivier and Eric Moulines</i>	
Upper-Confidence-Bound Algorithms for Active Learning in Multi-armed Bandits	189
<i>Alexandra Carpentier, Alessandro Lazaric, Mohammad Ghavamzadeh, Rémi Munos, and Peter Auer</i>	

Online Learning

The Perceptron with Dynamic Margin	204
<i>Constantinos Panagiotakopoulos and Petroula Tsampouka</i>	
Combining Initial Segments of Lists	219
<i>Manfred K. Warmuth, Wouter M. Koolen, and David P. Helmbold</i>	
Regret Minimization Algorithms for Pricing Lookback Options	234
<i>Eyal Gofer and Yishay Mansour</i>	
Making Online Decisions with Bounded Memory	249
<i>Chi-Jen Lu and Wei-Fu Lu</i>	
Universal Prediction of Selected Bits	262
<i>Tor Lattimore, Marcus Hutter, and Vaibhav Gavane</i>	
Semantic Communication for Simple Goals Is Equivalent to On-line Learning	277
<i>Brendan Juba and Santosh Vempala</i>	

Kernel and Margin Based Methods

Accelerated Training of Max-Margin Markov Networks with Kernels....	292
<i>Xinhua Zhang, Ankan Saha, and S.V.N. Vishwanathan</i>	

Domain Adaptation in Regression	308
<i>Corinna Cortes and Mehryar Mohri</i>	
Approximate Reduction from AUC Maximization to 1-Norm Soft Margin Optimization	324
<i>Daiki Suehiro, Kohei Hatano, and Eiji Takimoto</i>	

Intelligent Agents

Axioms for Rational Reinforcement Learning	338
<i>Peter Sunehag and Marcus Hutter</i>	
Universal Knowledge-Seeking Agents	353
<i>Laurent Orseau</i>	
Asymptotically Optimal Agents	368
<i>Tor Lattimore and Marcus Hutter</i>	
Time Consistent Discounting	383
<i>Tor Lattimore and Marcus Hutter</i>	

Other Learning Models

Distributional Learning of Simple Context-Free Tree Grammars	398
<i>Anna Kasprzak and Ryo Yoshinaka</i>	
On Noise-Tolerant Learning of Sparse Parities and Related Problems	413
<i>Elena Grigorescu, Lev Reyzin, and Santosh Vempala</i>	
Supervised Learning and Co-training	425
<i>Malte Darnstädt, Hans Ulrich Simon, and Balázs Szörényi</i>	
Learning a Classifier when the Labeling Is Known	440
<i>Shalev Ben-David and Shai Ben-David</i>	

Erratum

Erratum: Learning without Coding	452
<i>Samuel E. Moelius III and Sandra Zilles</i>	

Author Index	453
-------------------------------	-----

Editors' Introduction

Jyrki Kivinen, Csaba Szepesvari, Esko Ukkonen, and Thomas Zeugmann

The ALT-conference series is focuses on studies of learning from an algorithmic and mathematical perspective. During the last decades various models of learning emerged and a main goal is to investigate how various learning problems can be formulated and solved in some of the abstract models.

The general study of scenarios in which computer programs learn from information provided to them involves a considerable interaction between different branches of mathematics and computer science such as analysis, statistics, probability theory, combinatorics, theory of computation, Kolmogorov complexity, and analysis of algorithms. There are also close connections to the more empirical oriented disciplines of machine learning.

This wide variety is also nicely reflected in the papers contained in this volume. In the following, we shall introduce the five invited lectures and the regular contributions in some more detail.

Invited Talks. It is a good tradition of the co-located conferences ALT and DS to have five joint invited speakers. For ALT 2011 and DS 2011 the invited speakers are eminent researchers and they present either an introduction to their specific research area or give a lecture of wide general interest.

Peter Auer (University of Leoben) has worked on various topics in learning theory. One important theme has been exploration-exploitation trade-off in reinforcement learning, as modeled for example by the so-called multiarmed bandit problem. His invited presentation *Models for Autonomously Motivated Exploration in Reinforcement Learning* (joint work with Shiao Hong Lim and Chris Watkins) proposed some settings where the learning agent is not just maximizing some externally given reward function, as usually in reinforcement learning, but actively tries to find out things about the environment that might be useful in the future.

Yoshua Bengio (Universite de Montral) received his PhD from McGill University in 1991. His work covers a variety of topics in machine learning and neural networks. Much of his recent work has focused on deep architectures, which were also the topic of his invited presentation *On the Expressive Power of Deep Architectures* (joint work with Olivier Delalleau). Using representations with several layers allows building higher-level abstractions on top of some simple underlying structure, which might be needed to solve challenging AI problems. For a long time the study of deep architectures was discouraged by the lack of good learning algorithms for them, but recently there have been some striking successes that have brought the topic back into mainstream. The invited presentation gave theoretical and practical motivations for deep architectures, surveyed some of the successful algorithms and considered ideas for further challenges.

Jorma Rissanen (Helsinki Institute for Information Technology) received his PhD from Helsinki University of Technology in 1965. During his long career

he has made several highly influential contributions to information theory and its applications. He received the IBM Outstanding Innovation Award in 1988, the IEEE Richard W. Hamming Medal in 1993, and the Kolmogorov Medal in 2006. Furthermore, his honors include receiving in 2009 the Claude E. Shannon Award for his work in developing arithmetic coding. Within the machine learning community he is perhaps best known for introducing and developing the Minimum Description Length principle. His invited talk *Optimal Estimation* presented new information-theoretic methods for estimating parameters, their number and structure, with results about their optimality properties.

Eyke Hüllermeier (Universität Marburg) received his PhD in 1997 from the Computer Science Department of the University of Paderborn, and his Habilitation degree in 2002 from the same university. From 1998 to 2000, he spent two years as a Marie Curie fellow at the Institut de Recherche en Informatique de Toulouse. Currently, he is also the head of the IEEE/CIS ETTC Task Force on Machine Learning.

Johannes Fürnkranz (Technical University of Darmstadt) obtained his PhD in 1994, and the Habilitation degree in 2001 both from the Technical University of Vienna. In 2002 he received a prestigious APART stipend of the Austrian Academy of Sciences. His main research interest is machine learning. He also received an “Outstanding Editor Award” of the *Machine Learning journal*.

In their invited talk *Learning from Label Preferences* Eyke Hüllermeier and Johannes Fürnkranz studied a particular instance of preference learning. They addressed this problem by reducing it to the learning by pairwise comparison paradigm. This allows to decompose a possibly complex prediction problem into learning problems of a simpler type, i.e., binary classification.

Ming Li (University of Waterloo) received his PhD from Cornell University in 1985. His research interests cover a wide range of topics including bioinformatics algorithms and software, Kolmogorov complexity and its applications, analysis of algorithms, computational complexity, and computational learning theory. His outstanding contributions have been widely recognized. In 2006 he became an ACM fellow, a fellow of the Royal Society of Canada, and an IEEE fellow. Furthermore, he received the Award of Merit from the Federation of Chinese Canadian Professionals in 1997, the IEEE Pioneer Award for Granular Computing in 2006, the Premier’s Discovery Award for Innovation Leadership in 2009, the Outstanding Contribution Award from IEEE for Granular Computing in 2010, and the Killam Prize in 2010. The invited talk *Information Distance and Its Extensions* by Ming Li presented two extensions to the general theory of information distance concerning multiple objects and irrelevant information. The theory of information distance emerged during the last two decades and it found numerous applications during the past ten years.

Inductive Inference. A formal language is just a set of strings over some fixed finite alphabet. Inductive inference of formal languages is the study of algorithms that map evidence on a language into hypotheses about it. In general, one studies scenarios in which the sequence of computed hypotheses stabilizes to an accurate and finite description (e.g., a grammar) of the target language.

The following sources of information are distinguished. The learner receives augmenting initial segments of any sequence of all positive and negative examples (all strings over the underlying alphabet are classified with respect to their containment in the target language). In general, there is no requirement concerning the order in which the labeled strings are presented. If this source of information is used then we refer to it as *informant*. Instead of having potential access to all labeled strings, the learner may be required to learn from positive data only. Then the learner is fed augmenting initial segments of any infinite sequence of strings exhausting just the target language in the limit. We refer to it as learning from *text*. Learning from examples usually results in identification in the limit, i.e., after having seen only finitely many examples the learner stabilizes its output to a correct description of the target. Note that it is usually not decidable whether or not the learner has already converged. A learner identifies a target language if it learns the target from any text (informant) for it. A learner learns a class of languages if it identifies every language in the class.

Allowing the learner to compute its actual hypothesis from all the data seen so far is somehow unrealistic, since it requires memory capacities to process text segments of unbounded length. Therefore, one has also studied the variation that the learner has exclusively access to the new datum coming in and its previously computed hypothesis. The criterion of success remains unchanged, i.e., the learner stabilizes its output to a correct description of the target. The resulting learning model is referred to as *iterative learning*. It is well known that the collection of all classes of languages that are iteratively learnable is strictly smaller than the collection of all classes of languages learnable in the limit. Therefore, one has also studied variations of iterative learning.

The paper *Iterative Learning from Positive Data and Counters* by Timo Kötzing considers the variation that an iterative learner has additionally access to a counter. While it was known that this additional information yields a strictly more powerful learning model, it remained open why and how such a counter augments the learner power. To answer this question, six different types of a counter are distinguished. In the previously studied case, the counter was incremented in each iteration, i.e., counting from zero to infinity (i.e., $c(i+1) = c(i) + 1$). Further possibilities include *strictly increasing* counters (i.e., $c(i+1) > c(i)$), and *increasing and unbounded* (i.e., $c(i+1) \geq c(i)$ and the limit inferior of the sequence of counter values is infinity). The paper completely characterizes the relative learning power of iterative learners in dependence on the counter type allowed. It is shown that strict monotonicity and unboundedness are the only properties of the counters that augment the learner power in the iterative setting. The situation changes if other learning criteria are considered. For example, the learner may be required to never abandon a correct hypothesis, or its hypotheses should not depend on the order and the number of repetitions of the examples. It is then shown that for each choice of two different counter types there is a learning criterion that, when augmented with one of the counter types, yields different learnable classes than the same criterion when augmented with the other counter type.

The setting studied by Sanjay Jain, Eric Martin, and Frank Stephan in their paper *Robust Learning of Automatic Classes of Languages* is different from the general one described above in that the classes of target languages considered are required to be *automatic* ones. That is, the authors consider classes of regular languages of the form $(L_i)_{i \in I}$ such that $\{(i, x) \mid x \in L_i\}$ and I itself are regular sets. So automatic classes of languages are a particular type of an automatic structure. Note that automatic structures have received considerable attention recently in learning theory and elsewhere. Furthermore, automatic classes are also a special case of indexed families that have been intensively studied in learning theory. To explain what is meant by *robust learning*, let us assume that we know a class \mathcal{L} to be learnable. The interesting question is then what can be said about the classes \mathcal{T} that are obtainable by applying an algorithmic transformation to \mathcal{L} . If all these classes \mathcal{T} are learnable then we call \mathcal{L} robustly learnable. Clearly, the answer may depend on the type of transformation allowed. Knowing such an invariant is of great importance as we all know from mathematics. In the case of inductive inference of recursive functions this problem has been studied intensively. There it turned out that general recursive operators, that is, operators mapping every total function to a total one, are the most appropriate transformations. In the setting of inductive inference of languages from text, so far the attempts to find the appropriate class of transformations failed. In their paper, Jain, Martin, and Stephan resolve this problem for automatic classes by using automatic systems to define the class of admissible transformations. Then characterizations of robust learning with respect to several natural learning criteria are provided. Last but not least the authors extend their results to the case where the source of information is changed to queries. That is, instead of passively receiving initial segments of a text, the learner is allowed to ask particular types of questions to gain information concerning the target. Commonly used types of questions are membership queries (asking whether or not a particular string belongs to the target language) and equivalence queries (asking whether or not a particular finite description generates the target language and nothing else). In addition also subset queries and superset queries are studied. Note that the criterion of success has to be modified, too. Instead of learning in the limit, learning via queries requires the learner to indicate that it has learned by stopping to ask questions and outputting a correct description of the target.

The paper *Learning and Classifying* by Sanjay Jain, Eric Martin, and Frank Stephan sheds additional light on our understanding of language learning by relating it to classification. The model of classification considered is new. The authors define a so-called P -classifier which takes as input a finite sequence of elements of a language L and a finite sequence of predicates from the set P . It then outputs either the special symbol “?” indicating that the classifier makes no guess or a finite sequence of truth values intended as guesses of the values of the predicates on the language L . A computable P -classifier M is said to classify a language L if for every text for L and for every finite sequence of predicates from P , the guesses of M are the correct truth values of the finite sequence of predicates on L for all but finitely many initial segments of the text. Again, a

computable P -classifier is said to classify a class of languages if it classifies every language in the class. So it remains to specify the set P of allowed predicates. The basic predicates are membership of a particular element in L . The remaining predicates are Boolean combinations of these basic predicates.

The paper then compares P -classification with different criteria of learning from text. These criteria are learning in the limit as described above, behaviorally correct learning and finite identification. *Behaviorally correct learning* differs from learning in the limit in that the learner has to output for all but finitely many inputs a correct hypothesis, but not necessarily the same one. *Finite identification* is a model, where it is demanded that convergence of the sequence of hypotheses is decidable. So the learner outputs again the special symbol “?” indicating that it does not make a guess or a hypothesis. Once a hypothesis is output, it must be correct and learning is over. The main motivation for these investigations and the main insight obtained by these studies is the exploration of the idea that learning may be viewed as the limit of an increasing set of classification tasks.

The paper *Learning Relational Patterns* by Michael Geilke and Sandra Zilles studies the learnability of a special target class. Patterns are a very intuitive way to define languages. A pattern is just a non-empty string over $(\Sigma \cup X)$, where Σ is finite alphabet (the so-called constants) and $X = \{x_1, x_2, \dots\}$ is a countable set of variables. For example, $ax_1abbx_2cx_1$ is a pattern provided $a, b, c \in \Sigma$. The language $L(\pi)$ generated by a pattern π is the set of all strings obtainable by substituting strings over Σ for the variables occurring in π , where in each substitution step the same string has to be used for all occurrences of the same variable. So, $abbabbaacb \in L(\pi)$, where $\pi = ax_1abbx_2cx_1$ and obtained by substituting bb for x_1 and aa for x_2 . Note that it makes a huge difference whether or not only non-empty strings are allowed as substitutions. If this is the case then the class of all pattern languages is learnable from text. On the other hand, if empty strings are allowed as substitutions then the class of all pattern languages is *not* learnable from text. The present paper considers the case that empty substitutions are not allowed. While the class of all such pattern languages is very interesting and has attracted a lot of attention, it may be also too general for several applications. Thus, the authors introduce a new class by allowing (a finite number of) relations between the variables in the pattern and call the new class relational patterns. For instance, a relation can be used to express the demand that the substitutions for the variables x_1 and x_2 used in one substitution step have always the same length (as in our example above). It is then shown that the class of relational pattern languages is learnable from text, where the hypotheses output are also relational patterns. The authors also study the complexity of the membership problem which is NP -complete for the original class. For relational patterns it is shown to be NP -hard. Finally, probabilistic relational patterns are considered. Now for each variable type (expressed by the given relations) a distribution over the set of allowed substitutions is specified. This induces a probability distribution over the strings of the language, and learning has to be performed with respect to all texts obtainable in this way. The success

criterion is then relaxed to δ -learning, meaning that the learner has to succeed with probability at least $1 - \delta$. Under fairly natural conditions on a class of all relational patterns it is shown that δ -learning can be achieved.

Regression. In regression problems one is concerned with learning to predict a real-valued response given some inputs. The learner predicting some value suffers some loss, which is usually the square of the prediction error. The problem can be studied in the online learning framework or under some statistical assumptions; in both cases the main issue is to design algorithms which keep the prediction loss as small as possible.

Of considerable interest is to learn a linear mapping from a d -dimensional Euclidean space to the set of real numbers, i.e., the task of linear prediction. In many practical problems one suspects that the weight vector w^* defining the linear mapping is sparse because not all inputs are relevant. This also implies that the weight vector will have a small 1-norm. How to design algorithms that can exploit this prior information has been the subject of intense investigation in recent years. Sébastien Gerchinovitz and Jia Yuan Yu study this problem in the online learning framework in their paper *Adaptive and Optimal Online Linear Regression on ℓ^1 -balls*. Their main contribution is showing that the best achievable regret is subject to a phase transition depending on the value of the “intrinsic quantity” $\kappa = \sqrt{T} \|w^*\|_1 X / (2dY)$: For $\kappa < 1$, the best achievable regret scales as $d\kappa$, whereas for $\kappa > 1$ it behaves as $d \ln \kappa$. Here, T is the sample-size, X is the size of the ℓ^∞ -ball that the inputs lie in, and Y is a bound on the size of the responses. They also give computationally efficient algorithms that essentially achieve this bound without knowing the values of $\|w^*\|_1$, X , Y or T .

Nina Vaits and Koby Crammer in their paper *Re-Adapting the Regularization of Weights for Non-Stationary Regression* consider the problem of tracking the best sequence of weights also in the context of linear prediction with a squared loss. They develop an algorithm which uses per-feature learning rates and prove a regret bound with respect to the best sequence of functions. Under some technical assumption and with proper tuning, the regret is shown to be of order $O(T^{(p+1)/2p} \log T)$ when the best weight sequence’s “cumulative deviation” is of order $O(T^{1/p})$ for some $p > 1$. They also show that by running multiple instances in parallel, prior knowledge of p can be avoided.

Oftentimes, analyzing the “in-sample” or training error is the first step in the analysis of the risk of regression methods. Moreover, the behavior of the training error is also of major interest in signal or image processing when the goal of learning is to reject noise at the input points in the training data. The main novelty in the paper of Arnak S. Dalalyan and Joseph Salmon (*Competing Against the Best Nearest Neighbor Filter in Regression*) is that the authors prove a sharp oracle inequality for the expected training error of a procedure that they suggest. The oracle inequality is called sharp as its leading constant, multiplying the risk of the best predictor within the considered set of predictors, is one and the additive residual term decays at the rate of $O(1/n)$. The procedure itself uses aggregation with exponential weights over a set of symmetrized linear estimators, a special case of which are nearest neighbor filters. In particular, the

procedure avoids the need to choose the number of neighbors k to be considered and yet its performance is guaranteed to be almost as good as that of the nearest neighbor filter with the best choice of k . The procedure assumes the knowledge of the covariance matrix underlying the noise or an unbiased estimate of this covariance matrix which is independent of the responses used in the training procedure.

Bandit Problems. Bandit problems provide the simplest model to study learning in interactive, sequential scenarios with limited feedback: The learner takes actions, resulting in some reward that the learner observes. However, the learner gains no information about the rewards associated with the action not taken, hence the feedback about the environment is limited. The goal of the learner is to achieve as much reward as possible. Performance is measured in terms of the regret, i.e., the loss as compared to using the single best action from the beginning of time.

In their paper *Lipschitz Bandits without the Lipschitz Constant* Sébastien Bubeck, Gilles Stoltz, and Jia Yuan Yu study bandit problems when the set of actions is the d -dimensional hypercube and the payoff function is known to be globally Lipschitz with respect to the maximum-norm. They develop an algorithm which works as well as the Lipschitz constant was available though their algorithm does not need to know the Lipschitz constant. This is in contrast to previous works which either assumed that the Lipschitz constant is known *a priori* or the regret of the algorithm scaled suboptimally as a function of the unknown Lipschitz constant. The strategy proposed is based on a discretization argument, assumes the knowledge of the horizon and proceeds in two phases. In the first phase, the Lipschitz constant is estimated by exploring the points in the hypercube uniformly. By the end of this phase, the Lipschitz constant is estimated based on the obtained data. By biasing the estimate obtained this way upwards, it is ensured that the the Lipschitz constant will not be underestimated. In the second phase, the hypercube is discretized based on the estimated biased Lipschitz constant and then a standard multi-armed bandit strategy is used in the resulting finite-armed problem.

Most papers concerned with the stochastic version of bandit problem study the expected regret. However, a decision maker might also be interested in the risk, i.e., whether the regret is small not only in expectation, but also with high probability. In their paper *Deviations of Stochastic Bandit Regret* Antoine Salomon and Jean-Yves Audibert show that in the classical setting of finite-armed stochastic bandit problems whether “small risk” policies exist hinges upon whether the total number of plays is known beforehand. That small risk is possible to achieve when this knowledge is available was known beforehand. The new result is that without this knowledge, no algorithm can achieve small risk except when the class of distributions that can be assigned to the actions is restricted in some way.

Bandit algorithms designed for static environments are not expected to work well when the environment changes from time to time, for example when the environment changes abruptly. In the case of adversarial stochastic bandits, the

corresponding problem is called the tracking problem and the so-called Exp3.S algorithm was shown to achieve a regret of $O(\sqrt{sT \log T})$ on a horizon T when the number of abrupt changes is at most s . Aurélien Garivier and Eric Moulines study the same problem in a stochastic setting. In their paper titled *On Upper-Confidence Bound Policies for Switching Bandit Problems* they prove that already when a single switch between two stochastic environments $\mathcal{E}_1, \mathcal{E}_2$ is allowed, no algorithm can achieve better than $\sqrt{C(\mathcal{E}_1, \mathcal{E}_2)T}$ regret, where $C(\mathcal{E}_1, \mathcal{E}_2)$ is a constant that depends on the two environments $\mathcal{E}_1, \mathcal{E}_2$ only. They also study the so-called discounted UCB algorithm and one version which uses sliding windows. They show that with appropriate tuning these algorithms are able to match the regret of Exp3.S. These theoretical findings are complemented with results of numerical experiments that indicate that the UCB-type algorithms might be advantageous in stochastic environments compared to Exp3.S.

Online learning with bandit information can be studied under various criteria. In their paper *Upper-Confidence-Bound Algorithms for Active Learning in Multi-Armed Bandits* Alexandra Carpentier, Alessandro Lazaric, Mohammad Ghavamzadeh, Rémi Munos, and Peter Auer study the problem of estimating the mean values of a finite number of actions uniformly well. In earlier work, a specific algorithm based on a forced exploration strategy was designed and analyzed for this problem. However, it is suspected that forced exploration with a fixed exploration rate can lead to suboptimal performance. In their paper Carpentier *et al.* study algorithms which avoid fixed rate forced exploration schemes. The performance bounds developed are indeed better than that of developed for the forced exploration scheme.

Online Learning. In online learning the training data is presented sequentially and the learner updates its hypothesis after each data point. Sequential decision making tasks naturally require online learning, but it can also be used for computational reasons even when all the training data is available at once but manipulating the entire data set is computationally too expensive.

The classical perceptron algorithm takes very little time per iteration and is guaranteed to find a hyperplane separating the positive and negative data points if one exists, but the margin of the hyperplane is not in any way optimized. Various algorithms for optimizing the margin are known, but they get computationally quite demanding for large data sets. Constantinos Panagiotakopoulos and Petroula Tsampouka in their paper *The Perceptron with Dynamic Margin* contribute to the line of research that tries to combine the computational simplicity of the perceptron algorithm with guaranteed approximation bounds for the margin. The algorithm is based on maintaining a dynamic upper bound on the maximum margin. Besides the theoretical approximation bound, also experiments show the algorithm to perform well compared to previous ones for the same problem.

The paper *Combining Initial Segments of Lists* by Manfred K. Warmuth, Wouter M. Koolen, and David P. Helmbold falls broadly within the framework of predicting with expert advice. As an example, suppose you have K different policies for maintaining a memory cache of size N . Different policies work well

for different access sequences, and you would like to combine the caches of the K policies dynamically into your own cache of size N such that no matter what the access sequence, you do not incur many more misses than the best of the K policies for that particular sequence. A naive implementation of the well-known Hedge algorithm for predicting with expert advice is not computationally feasible, since there are roughly N^K ways of picking your combined cache. However, the paper comes up with efficient algorithms based on the special combinatorial structure of this set of N^K combinations. Also some lower bounds and hardness results are presented.

The paper *Regret Minimization Algorithms for Pricing Lookback Options* by Eyal Gofer and Yishay Mansour applies tools from online prediction to finance. The final goal of the paper is to get upper bounds for the values of a certain type of an option. Assuming an arbitrage-free market, such upper bounds can be derived from regret bounds for trading algorithms. The trading algorithm considered here combines one-way trading (selling stock over time but never buying more) with regret minimization that tries to follow which performs better, cash or stock. A simulation on real stock data demonstrates how the bound works in practice.

The paper *Making Online Decisions with Bounded Memory* by Chi-Jen Lu and Wei-Fu Lu is concerned with the problem of prediction with expert advice for $0 - 1$ losses when the predictor is a finite state machine. Assuming that the number of actions is n , it is shown that any predictor with m^{n-1} states must have regret $\Omega(T/m)$ in T time steps (note that remembering the exact number of mistakes of each expert would use T^n states). In the paper the authors propose two new algorithms for this problem: the first one is based on exponential weighting and achieves $O(m+T/m \ln(nm))$ regret (for small m), while the second algorithm, based on gradient descent, achieves $O(n\sqrt{m} + T/\sqrt{m})$ regret. Note that the first algorithm achieves an almost optimal $\sqrt{T \ln n}$ regret using roughly half the memory that standard algorithms would use.

In sequence prediction the problem is to predict the next symbol of a sequence given the past. As it is well known Solomonoff induction solves this problem, but only if the entire sequence is sampled from a computable distribution. In the paper *Universal Prediction of Selected Bits* Tor Lattimore, Marcus Hutter, and Vaibhav Gavane consider the more general problem of predicting only parts of a sequence, lifting the restriction that the sequence is computable (or sampled from a computable distribution). For example, in an online classification problem, the side information available to predict the next outcome can be arbitrary, the only part that needs to be predicted are the labels. They show that the normalized version of Solomonoff induction can still be used in this more general problem, and in fact it can detect any recursive sub-pattern (regularity) within an otherwise completely unstructured sequence. It is also shown that the unnormalized version can fail to predict very simple recursive sub-patterns.

Consider two computers communicating using protocols designed and implemented by different parties. In such settings, the possibility of incompatibility arises. Thus, it is desirable if one (or both) computers utilize a communication

strategy that automatically corrects mistakes. Previous work has shown that if the user can sense progress, such a strategy exists if there exists a protocol at all that would achieve the goal of communication. The drawback of the actual constructions is that they rely on enumerating protocols until a successful one is discovered, leading to the potential for exponential overhead in the length of the desired protocol. Brendan Juba and Santosh Vempala in their paper *Semantic Communication for Simple Goals is Equivalent to On-line Learning* consider the problem of reducing this overhead for some reasonably general special cases. Most interestingly, this is done by establishing an equivalence between these special cases and the usual model of mistake-bounded on-line learning. The results motivate the study of sensing with richer kinds of feedback.

Kernels and Margin-Based Methods. Kernels are a powerful mathematical tool that have gained popularity in machine learning, among other reasons, because they sometimes allow computationally efficient implementation of algorithms that otherwise would require manipulating very high-dimensional feature vectors. Learning algorithms that operate in a high-dimensional feature space often employ some form of margin maximization as a means of avoiding overfitting.

The paper *Accelerated Training of Max-Margin Markov Networks with Kernels* by Xinhua Zhang, Ankan Saha, and S.V.N. Vishwanathan considers structured output prediction, where in addition to the inputs, also the outputs to be predicted can have a complicated structure. Using the kernel paradigm this can be modeled assuming a joined feature map ϕ that maps input-output pairs (\mathbf{x}, \mathbf{y}) into the feature space. One way of proceeding from there, and the one adopted in this paper, is max-margin Markov networks, which leads to a minimization problem where the objective function is convex but not smooth. Non-smoothness rules out some of the faster optimization methods. This paper shows how some known techniques for this kind of optimization can be modified so that they retain their convergence speed, getting to within ϵ of the optimum in $O(1/\sqrt{\epsilon})$ iterations, and allow the iteration step to be implemented in an efficient manner that utilizes the structure of the outputs.

Corinna Cortes and Mehryar Mohri in their paper *Domain Adaptation in Regression* consider the situation when the training and test data come from different distributions. We assume there is little or no labeled data about the *target domain* where we actually wish to learn, but unlabeled data is available, as well as labeled data from a different but somehow related *source domain*. Previous work has introduced a notion of discrepancy such that a small discrepancy between the source and target domains allows learning in this scenario. This paper sharpens and simplifies the previous results for a large class of domains related to kernel regression. It then goes on to develop an algorithm for finding a source distribution that minimizes the discrepancy and shows empirically that the new algorithm allows domain adaptation on much larger data sets than previous methods.

The paper *Approximate Reduction from AUC Maximization to 1-Norm Soft Margin Optimization* by Daiki Suehiro, Kohei Hatano, and Eiji Takimoto

considers the problem of obtaining a good ranking function as a convex combination of a given set of basic ranking functions. Area under the ROC curve (AUC) is a popular performance measure for ranking, and known results bounds it in terms of a margin-based criterion for pairs of positive and negative examples. However, using this reduction directly to optimize AUC leads to a problem of size $O(pn)$, where p is the number of positive examples and n the number of negative examples in the original problem. This is computationally infeasible for even moderately large data sets. The paper presents an alternative reduction that leads to a problem of size $O(p + n)$. The problem thus obtained is not equivalent with the original problem, but the paper provides some approximation guarantees, and shows empirically that the proposed approach is practical.

Intelligent Agents. Intelligent agents need to adapt to their environment to achieve their goals. The problem is made especially difficult by the fact that the actions taken may have long term effects.

In their paper *Axioms for Rational Reinforcement Learning*, following Savage's pioneering work, Peter Sunehag and Marcus Hutter define a notion of rational agents and show that the so-defined rational agents act as if they maintained a probabilistic world model. The simplest rationality concept considered in the paper from which the other concepts are derived concerns agents who have preferences above payoff schemes corresponding to a single uncertain outcome. The authors also investigate the subtleties of countably infinite outcomes and asymptotic optimality when an agent faces countably many environments.

Laurent Orseau studies the question of how to design agents which are “knowledge seeking” in the paper titled *Universal Knowledge-Seeking Agents*. The knowledge-seeking agents are those who have a probabilistic world model. In a rather unorthodox manner, the immediate cost suffered by such an agent at some time step is defined as the conditional probability assigned to future outcomes based on the probabilistic world model that the agent chose to use. Arguably, an agent that uses an appropriate world model and that acts so as to minimize the long-term cost will choose actions that allow it to “discard” as many environments as quickly as possible. Performance is compared to the expected total cost suffered by the optimal agent that uses the probability distribution of the true environment as its world model. The main result, which is proven for certain “horizon functions,” shows that the so-called AIXI agent’s performance converges to the optimal performance provided that the environment is deterministic. A cost defined using the logarithm of the conditional probabilities, i.e., a Shannon-type cost, is also studied.

A recent result by Orseau published at ALT 2010 showed that Hutter’s universal Bayesian agent AIXI fails to be weakly asymptotically optimal when the environment is chosen to be some computable deterministic environment. The main observation in the paper *Asymptotically Optimal Agents* by Tor Lattimore and Marcus Hutter is that a similar result holds true for arbitrary agents. In particular, the authors study this question in general discounted deterministic reinforcement problems. It is shown that no learning algorithm can be strongly asymptotically optimal for this class of environments, while the existence of

weakly asymptotically optimal algorithms depends on the considered discount function. However, weakly asymptotically optimal algorithms are necessarily incomputable. One such algorithm is presented for geometric discounting.

A discount matrix d is an $\infty \times \infty$ matrix: At time step t an agent using d would “discount” future rewards using the values in the t th column of d . A discount matrix leads to time-consistent behavior if for any environment the optimal policy given some history up to time t uses the same action as the optimal policy that is computed with a column of the discount matrix corresponding to some previous time instance (ties are assumed to be broken in an arbitrary, systematic manner). Tor Lattimore and Marcus Hutter prove a characterization of what discount matrices lead to time consistent discounting in their paper *Time Consistent Discounting*. They also study the sensitivity of behaviors to perturbations of a time-consistent discount matrix. Finally, using a game theoretic approach, they show that there is a rational policy even if the discount matrix is time-inconsistent.

Other Learning Models. *Identification in the limit from positive data* is one of the earliest learning paradigms considered in computer science. The learner receives an infinite sequence that consists of the strings belonging to an unknown (infinite) language. After each input string, the learner outputs a grammar, and learning is successful if after some finite amount of steps, the grammar is correct for the unknown language.

Probably approximately correct (PAC) learning is another model that has served as the framework for many fundamental results and also inspired a large number of other models. In the basic PAC setting, the unknown quantities are a target concept $f: X \rightarrow \{0, 1\}$ and a probability measure P over X . The learner receives a set of labeled examples $(x, f(x))$ and outputs a hypothesis $h: X \rightarrow \{0, 1\}$. For given ε and δ , the hypothesis must satisfy with probability at least $1 - \delta$ the property $P(f(x) \neq h(x)) \leq \varepsilon$. The analysis of a learning algorithm involves estimating the required number of examples and the computation time in terms of ε , δ and other relevant parameters of the problem.

The paper *Distributional Learning of Simple Context-Free Tree Grammars* by Anna Kasprzik and Ryo Yoshinaka considers learning of languages consisting of trees, not strings. Context-free tree grammars generalize the notion of context-free grammars from strings to trees. While learning general context-free languages seems difficult, efficient learning algorithms for several subclasses are known. The present paper, in particular, takes as a starting point the known results for *substitutable* context-free languages. A context-free language L over is substitutable, if any two strings z_1 and z_2 that satisfy $uz_1v \in L$ and $uz_2v \in L$ for some pair of strings (u, v) , also satisfy the condition $u'z_1v' \in L \Leftrightarrow u'z_2v' \in L$ for any pair (u', v') . Intuitively, if z_1 and z_2 can both appear in one context (u, v) , then they can appear in exactly the same contexts (u', v') . The learning techniques based on analyzing such interactions between strings and their context are known as distributional. In the present paper, the notion of substitutability is appropriately generalized to tree languages. This leads to learning algorithms for several classes of context-free tree languages.

Elena Grigorescu, Lev Reyzin and Santosh Vempala in their paper *On Noise-Tolerant Learning of Sparse Parities and Related Problems* consider PAC learning with random classification noise. In this model, the examples are affected by noise with some fixed rate $\eta < 1/2$, so an example (x, y) of target concept f satisfies $y = f(x)$ with probability $1 - \eta$ and $y = 1 - f(x)$ with probability η . The situation where $X = \{0, 1\}^n$ and the target concept is known to be the parity of some unknown subset of the n input bits is of particular interest, since it is perhaps the most basic case that is known to be learnable in the noise-free setting, but not known to be learnable with random classification noise. In contrast, most known learning algorithms for the noise-free PAC model have been generalized to allow random classification noise by using the statistical query model. The present paper shows that parities of at most r variables are learnable in the PAC model with random classification noise in time $\text{poly}(1/\varepsilon, \ln(1/\delta), 1/(1-2\eta))n^{(1/2+2\eta^2+o(1))r}$, which is the first known improvement over the brute-force bound $O(n^r)$. The results of this paper can be combined with earlier work to get bounds for general r -juntas (functions that depend on only r input bits) and for s -term DNF formulas.

Co-training under the conditional independence assumption is a model often used in PAC-style analysis of semisupervised learning. In this model, access to a large number of unlabeled examples can lead to a drastic reduction in the required number of labeled examples. The paper *Supervised Learning and Co-Training* by Malte Darnstädt, Hans Ulrich Simon, and Balázs Szörényi poses the question of how much of this reduction is due to the unlabeled examples, and how much would result from the conditional independence assumption even without access to any unlabeled examples. It turns out that under this assumption, the number of labeled examples needed to co-train two concept classes, having VC-dimensions d_1 and d_2 , is $O(\sqrt{d_1 d_2}/\varepsilon)$. For small ε this is significantly smaller than the lower bound $\Omega(d/\varepsilon)$ for learning a concept class of VC-dimension d without the conditional independence assumption.

A labeled random example $(x, f(x))$ gives information both about the target function f and the distribution of x . The paper *Learning a Classifier When the Labeling is Known* by Shalev Ben-David and Shai Ben-David focuses on the second aspect by assuming that the target function f is actually known to the learner beforehand. The learning problem is still nontrivial if we require that the hypothesis h belongs to some restricted hypothesis class H that does not include the target f . In practice, such restrictions might arise because the hypothesis must be very efficient to evaluate, or in a form understandable to a human expert. The paper establishes a combinatorial property of H based on shattering that tells us which of three cases holds: the required number of samples is either zero, $\Theta(1/\varepsilon)$, or $\Omega(1/\varepsilon^2)$.

Models for Autonomously Motivated Exploration in Reinforcement Learning (Extended Abstract)

Peter Auer¹, Shiau Hong Lim¹, and Chris Watkins²

¹ Chair for Information Technology, Montanuniversität Leoben, Austria
<http://institute.unileoben.ac.at/infotech>

² Department of Computer Science, Royal Holloway University of London, UK
<http://www.rhul.ac.uk/computerscience>

Abstract. We discuss some models for autonomously motivated exploration and present some recent results.

Keywords: Reinforcement learning, autonomous exploration, intrinsic rewards.

1 Introduction

One of the striking differences between current reinforcement learning algorithms and early human learning is that animals and infants appear to explore their environments with autonomous purpose, in a manner appropriate to their current level of skills. An important intuition for autonomously motivated exploration was proposed in [1, 2]: an agent should be interested in making observations that reduce its uncertainty about future observations. Inherently random observations are not interesting because the agent can never learn to predict them; and observations that the agent can already predict are not interesting because they convey no new information. This criterion is valuable but essentially retrospective: at the end of its life, an agent can look back and use the criterion to rigorously assess which of its past explorations were useful in increasing the precision of predictions, and which explorations were useless. It is harder to identify useful explorations prospectively, but there are a range of plausible heuristics for doing so, in the form of setting up intrinsic rewards for explorations which give preliminary signs of success. Some of these ideas have already been used in reinforcement learning (3) to identify intermediate skills that should be learned. A taxonomy of intrinsic reward systems that encourage an agent to make such “interesting” observations is proposed in [4]; an implementation of a simple learning system exhibiting “intelligent adaptive curiosity” is described in [5]. However, there is not yet a systematic theoretical analysis of possible heuristics for prospective identification of useful explorations. It is also evident that improvement in prediction is only one of many rigorous retrospective criteria for identifying useful explorations: there has been no systematic examination of other criteria for retrospective and prospective identification of useful explorations.

2 Preliminaries

For a formal model of autonomous exploration that is accessible to theoretical analysis, we consider learning a Markov Decision Process (MDP) without external rewards. The learning agent still has to gather relevant information about the unknown state transitions in the MDP, but relevance is not determined by a predefined reward function on the states. Instead, we propose other notions for relevant information, that are derived from various objectives for the learning agent. In general, the goal of the learning agent is to gather as much relevant information as possible in the available exploration time, or — vice versa — to use as little exploration time as possible to gather the required information.

3 Learning to Reach States

An immediate and simple objective for a learning agent in an MDP with finite state space \mathcal{S} , is to find for each of the states $s \in \mathcal{S}$ a reliable policy for reaching this state from a defined start state s_0 . The learning protocol is that the learning agent may explore the MDP by taking actions in the MDP for a number of steps.¹ After the exploration phase, for each of the states $s \in \mathcal{S}$ the learning agent needs to output a policy π_s , which is evaluated by the average time $L(s|s_0, \pi_s)$ that it takes the agent to reach the goal state s from start state s_0 , when following policy π_s . The overall utility of the learning agent \mathcal{A} after T exploration steps can then be measured by some utility function,

$$U(\mathcal{A}, T) = \sum_{s \in \mathcal{S}} u(s, L(s|s_0, \pi_s)) ,$$

where $u(s, L)$ is a function decreasing in L . As an example, the particular choice

$$u_{\Lambda, \alpha}(s, L) = \begin{cases} 1 & \text{if } L \leq \alpha \Lambda \text{ and } \exists \pi_s^* : L(s|s_0, \pi_s^*) \leq \Lambda \\ 0 & \text{else} \end{cases}$$

counts the states which are reachable in Λ steps by an optimal policy and for which the learning agent has found a sufficiently good policy. We are interested in a bound on the number of necessary exploration steps, such that for all states reachable in Λ steps a sufficiently good policy has been found. A straightforward adaptation of the RMAX algorithm [6, 7] gives bounds polynomial in the number of states, but better bounds can be derived from more recent work [8]. We will also discuss further improvements of these bounds.

3.1 Infinite State Spaces

The problem of reaching states becomes more even interesting and intriguing when an infinite but discrete state space is considered. A simple adaptation of

¹ To avoid MDPs where the exploring agent may get stuck in a state, we assume that the agent has a RESET action that deterministically takes it back to the start state s_0 .

known reinforcement algorithms that depend on the size of the state space is not possible anymore. And even for a finite state space the dependency of the exploration time on the size of the complete state space is unfavorable. Instead, a dependency on the number of states that can actually be reached in Λ steps would be preferable. Unfortunately, this is not possible in general.

Counter Example. For any $n \geq 1$, consider the state space

$$\mathcal{S} = \{s_0, s_1^{(1)}, \dots, s_1^{(n)}, \dots, s_{\Lambda-1}^{(1)}, \dots, s_{\Lambda-1}^{(n)}, s_\Lambda\}$$

and actions a_0, a_1 . Assume for the transition probabilities that

- $p(s_1^{(k)}|s_0, a_i) = 1/n$ for $k = 1, \dots, n$ and $i = 0, 1$,
- for all $\ell = 1, \dots, \Lambda - 1$ and $k = 1, \dots, n$ there is an action a_i with $p(s_{\ell+1}^{(k)}|s_\ell^{(k)}, a_i) = 1$ and $p(s_0|s_\ell^{(k)}, a_{1-i}) = 1$, (for notational convenience $s_\Lambda^{(1)} = s_\Lambda^{(n)} = s_\Lambda$).

Then for $\Lambda \ll n$, s_Λ is the only state that can be reached in Λ steps by an optimal policy. But $\Omega(n\Lambda)$ exploration steps are necessary to actually find a policy which gets to s_Λ in $O(\Lambda)$ steps. \square

Thus we will consider also more constrained versions of learning to reach states.

3.2 Other Learning Objectives

Another natural goal for autonomous exploration is to enable the agent to adapt quickly to any externally assigned reward function. As it turns out, this requires — under mild conditions and in an adversarial setting — uniformly good estimates of all transition probabilities, since otherwise the rewards can be assigned such that the learning agent is hurt most by the most inaccurately estimated transition probabilities.

4 Further Research Directions

We believe that there are at least two very interesting directions for further research: one is to consider autonomous exploration in continuous state spaces with parametrized transition probabilities, see e.g. [9]. The other research direction is about further objectives for autonomous exploration, in particular learning and representing multiple skills. The competence of an agent can be considered as a function over a set of skills, and through exploration and learning the agent improves its competence.

Acknowledgments. This research has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr. 231495 (CompLACS) and nr. 216886 (PASCAL2).

References

- [1] Schmidhuber, J.: A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers. In: Meyer, J.A., Wilson, S.W. (eds.) International Conference on Simulation of Adaptive Behavior: From Animals to Animats, pp. 222–227. MIT Press, Cambridge (1991)
- [2] Schmidhuber, J.: Developmental Robotics, Optimal Artificial Curiosity, Creativity, Music, and the Fine Arts. *Connection Science* 18(2), 173–187 (2006)
- [3] Singh, S., Barto, A.G., Chentanez, N.: Intrinsically Motivated Reinforcement Learning. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) *Advances in Neural Information Processing Systems*, vol. 17, pp. 1281–1288. MIT Press, Cambridge (2005)
- [4] Oudeyer, P.-Y., Kaplan, F.: What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics*, 1 (2007)
- [5] Oudeyer, P.-Y., Kaplan, F., Hafner, V.: Intrinsic Motivation Systems for Autonomous Mental Development. *IEEE Transactions on Evolutionary Computation* 11(1), 265–286 (2007)
- [6] Brafman, R.I., Tennenholtz, M.: R-max - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *JMLR* 3, 213–231 (2003)
- [7] Kakade, S.M.: On the Sample Complexity of Reinforcement Learning. PhD thesis, University College London (2003)
- [8] Jaksch, T., Ortner, R., Auer, P.: Near-optimal Regret Bounds for Reinforcement Learning. *JMLR* 99, 1563–1600 (2010)
- [9] Engel, Y., Mannor, S., Meir, R.: Reinforcement learning with Gaussian processes. In: 22nd International Conference on Machine Learning, ICML 2005, pp. 201–208. ACM, New York (2005)

On the Expressive Power of Deep Architectures

Yoshua Bengio and Olivier Delalleau

Dept. IRO, Université de Montréal. Montréal (QC), H3C 3J7, Canada

Abstract. Deep architectures are families of functions corresponding to deep circuits. Deep Learning algorithms are based on parametrizing such circuits and tuning their parameters so as to approximately optimize some training objective. Whereas it was thought too difficult to train deep architectures, several successful algorithms have been proposed in recent years. We review some of the theoretical motivations for deep architectures, as well as some of their practical successes, and propose directions of investigations to address some of the remaining challenges.

1 Learning Artificial Intelligence

An intelligent agent takes good decisions. In order to do so it needs some form of knowledge. Knowledge can be embodied into a function that maps inputs and states to states and actions. If we saw an agent that always took what one would consider as the good decisions, we would qualify the agent as intelligent. Knowledge can be explicit, as in the form of symbolically expressed rules and facts of expert systems, or in the form of linguistic statements in an encyclopedia. However, knowledge can also be implicit, as in the complicated wiring and synaptic strengths of animal brains, or even in the mechanical properties of an animal's body. Whereas Artificial Intelligence (AI) research initially focused on providing computers with knowledge in explicit form, it turned out that much of our knowledge was not easy to express formally. What is a *chair*? We might write a definition that can help another human understand the concept (if he did not know about it), but it is difficult to make it sufficiently complete for a computer to translate into the same level of competence (e.g. in recognizing chairs in images). Much so-called *common-sense knowledge* has this property.

If we cannot endow computers with all the required knowledge, an alternative is to let them learn it from examples. Machine learning algorithms aim to extract knowledge from examples (i.e., data), so as to be able to properly generalize to new examples. Our own implicit knowledge arises either out of our life experiences (lifetime learning) or from the longer scale form of learning that evolution really represents, where the result of adaptation is encoded in the genes. Science itself is a process of learning from observations and experiments in order to produce actionable knowledge. Understanding the principles by which agents can capture knowledge through examples, i.e., learn, is therefore a central scientific question with implications not only for AI and technology, but also to understand brains and evolution.

Formally, a learning algorithm can be seen as a functional that maps a dataset (a set of examples) to a function (typically, a decision function). Since the dataset is itself a random variable, the learning process involves the application of a procedure to a target distribution from which the examples are drawn and for which one would like to infer a good decision function. Many modern learning algorithms are expressed as an optimization problem, in which one tries to find a compromise between minimizing empirical error on training examples and minimizing a proxy for the richness of the family of functions that contains the solution. A particular challenge of learning algorithms for AI tasks (such as understanding images, video, natural language text, or speech) is that such tasks involve a large number of variables with complex dependencies, and that the amount of knowledge required to master these tasks is very large. Statistical learning theory teaches us that in order to represent a large body of knowledge, one requires a correspondingly large number of degrees of freedom (or richness of a class of functions) and a correspondingly large number of training examples. In addition to the statistical challenge, machine learning often involves a computational challenge due to the difficulty of optimizing the training criterion. Indeed, in many cases, that training criterion is not convex, and in some cases it is not even directly measurable in a deterministic way and its gradient is estimated by stochastic (sampling-based) methods, and from only a few examples at a time (online learning).

One of the characteristics that has spurred much interest and research in recent years is **depth of the architecture**. In the case of a multi-layer neural network, depth corresponds to the number of (hidden and output) layers. A fixed-kernel Support Vector Machine is considered to have depth 2 (Bengio and LeCun, 2007) and boosted decision trees to have depth 3 (Bengio *et al.*, 2010). Here we use the word *circuit* or *network* to talk about a directed acyclic graph, where each node is associated with some output value which can be computed based on the values associated with its predecessor nodes. The arguments of the learned function are set at the input nodes of the circuit (which have no predecessor) and the outputs of the function are read off the output nodes of the circuit. Different families of functions correspond to different circuits and allowed choices of computations in each node. Learning can be performed by changing the computation associated with a node, or rewiring the circuit (possibly changing the number of nodes). The depth of the circuit is the length of the longest path in the graph from an input node to an output node.

This paper also focuses on **Deep Learning**, i.e., learning *multiple levels of representation*. The intent is to discover more *abstract* features in the higher levels of the representation, which hopefully make it easier to *separate from each other the various explanatory factors extent in the data*. Theoretical results (Yao, 1985; Håstad, 1986; Håstad and Goldmann, 1991; Bengio *et al.*, 2006; Bengio and Delalleau, 2011; Braverman, 2011), reviewed briefly here (see also a previous discussion by Bengio and LeCun, 2007) suggest that in order to learn the kind of complicated functions that can represent high-level abstractions (e.g., in vision, language, and other AI-level tasks) associated with functions with

many variations but an underlying simpler structure, one may need *deep architectures*. The recent surge in experimental work in the field seems to support this notion, accumulating evidence that in challenging AI-related tasks – such as computer vision (Bengio *et al.*, 2007; Ranzato *et al.*, 2007; Larochelle *et al.*, 2007; Ranzato *et al.*, 2008; Lee *et al.*, 2009; Mobahi *et al.*, 2009; Osindero and Hinton, 2008), natural language processing (NLP) (Collobert and Weston, 2008; Weston *et al.*, 2008), robotics (Hadsell *et al.*, 2008), or information retrieval (Salakhutdinov and Hinton, 2007; Salakhutdinov *et al.*, 2007) – deep learning methods significantly out-perform comparable but shallow competitors (e.g. winning the Unsupervised and Transfer Learning Challenge; Mesnil *et al.*, 2011), and often match or beat the state-of-the-art.

In this paper we discuss some of the theoretical motivations for deep architectures, and quickly review some of the current layer-wise unsupervised feature-learning algorithms used to train them. We conclude with a discussion of principles involved, challenges ahead, and ideas to face them.

2 Local and Non-local Generalization: The Challenge and Curse of Many Factors of Variation

How can learning algorithms generalize from training examples to new cases? It can be shown that there are no completely universal learning procedures, in the sense that for any learning procedure, there is a target distribution on which it does poorly (Wolpert, 1996). Hence, all generalization principles exploit some property of the target distribution, i.e., some kind of prior. The most exploited generalization principle is that of *local generalization*. It relies on a *smoothness assumption*, i.e., that the target function (the function to be learned) is smooth (according to some measure of smoothness), i.e., changes slowly and rarely (Barron, 1993). Contrary to what has often been said, what mainly hurts many algorithms relying only on this assumption (pretty much all of the non-parametric statistical learning algorithms) is not the dimensionality of the input but instead the insufficient smoothness of the target function¹.

To make a simple picture, imagine the supervised learning framework and a target function that is locally smooth but has many ups and downs in the domain of interest. We showed that if one considers a straight line in the input domain, and counts the number of ups and downs along that line, then a learner based purely on local generalization (such as a Gaussian kernel machine) requires at least as many examples as there are ups and downs (Bengio *et al.*, 2006).

Manifold learning algorithms are unsupervised learning procedures aiming to characterize a low-dimensional manifold near which the target distribution concentrates. Bengio and Monperrus (2005) argued that many real-world manifolds (such as the one generated by translations or rotations of images, when the image is represented by its pixel intensities) are highly curved (translating by 1

¹ But of course additional noisy dimensions, although they do not change smoothness of the target function, require more examples to cancel the noise.

pixel can change the tangent plane of the manifold by about 90 degrees). The manifold learning algorithms of the day, based implicitly or explicitly on non-parametric estimation of the local tangent planes to the manifold, are relying on purely local generalization. Hence they would require a number of examples that grows linearly with the dimension d of the manifold and the number of patches $O\left(\frac{D}{r}\right)^d$ needed to cover its nooks and crannies, i.e., in $O\left(d\left(\frac{D}{r}\right)^d\right)$ examples, where D is a diameter of the domain of interest and r a radius of curvature.

3 Expressive Power of Deep Architectures

To fight an exponential, it seems reasonable to arm oneself with other exponentials. We discuss two strategies that can bring a potentially exponential statistical gain thanks to a combinatorial effect: distributed (possibly sparse) representations and depth of architecture. We also present an example of the latter in more details in the specific case of so-called sum-product networks.

3.1 Distributed and Sparse Representations

Learning algorithms based on *local generalization* can generally be interpreted as creating a number of *local regions* (possibly overlapping, possibly with soft rather than hard boundaries), such that each region is associated with its own degrees of freedom (parameters, or examples such as prototypes). Such learning algorithms can then learn to discriminate between these regions, i.e., provide a different response in each region (and possibly doing some form of smooth interpolation when the regions overlap or have soft boundaries). Examples of such algorithms include the mixture of Gaussians (for density estimation), Gaussian kernel machines (for all kinds of tasks), ordinary clustering (such as k-means, agglomerative clustering or affinity propagation), decision trees, nearest-neighbor and Parzen windows estimators, etc... As discussed in previous work (Bengio *et al.*, 2010), all of these algorithms will generalize well only to the extent that there are enough examples to cover all the regions that need to be distinguished from each other.

As an example of such algorithms, the way a clustering algorithm or a nearest-neighbor algorithm could partition the input space is shown on the left side of Fig. II. Instead, the right side of the figure shows how an algorithm based on distributed representations (such as a Restricted Boltzmann Machine; Hinton *et al.*, 2006) could partition the input space. Each binary hidden variable identifies on which side of a hyper-plane the current input lies, thus breaking out input space in a number of regions that could be exponential in the number of hidden units (because one only needs a few examples to learn where to put each hyper-plane), i.e., in the number of parameters. If one assigns a binary code to each region, this is also a form of clustering, which has been called *multi-clustering* (Bengio, 2009).

Distributed representations were put forward in the early days of connectionism and artificial neural networks (Hinton, 1986, 1989). More recently, a variation on distributed representations has been explored by many researchers,

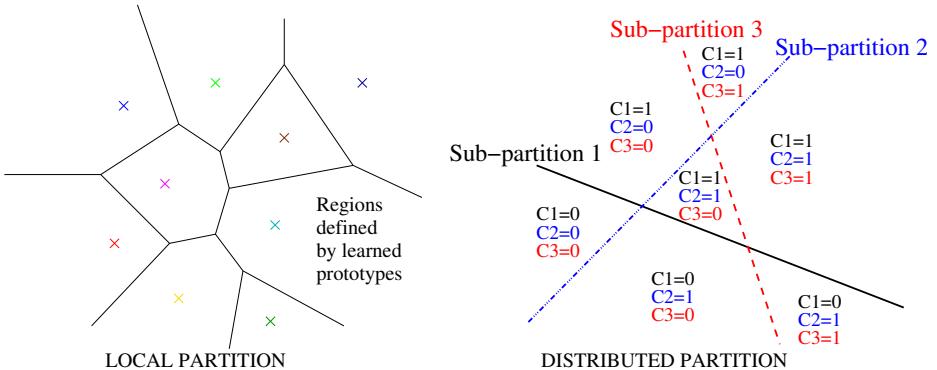


Fig. 1. Contrast between learning algorithms such as clustering (left side), based on local generalization, with examples required in each input region distinguished by the learner, and algorithms based on distributed representations, such as a Restricted Boltzmann Machine (right side). The latter also splits up the input space in regions but where the number of parameters or examples required can be much smaller (potentially exponentially smaller) than the number of regions one can distinguish. This is what grants the possibility of generalizing to regions where no data have been observed.

which is somehow in between purely local representations and the traditional dense distributed representations: sparse representations. The idea is that only a few dimensions of the representations are “active”, with the inactive dimensions basically set to 0 or close to 0. Neurons in the cortex are believed to have a distributed and sparse representation (Olshausen and Field, 1997), with around 1-4% of the neurons active at any one time (Attwell and Laughlin, 2001; Lennic, 2003). With k out of d active dimensions in a representation, one still gets (potentially) exponentially more representational power than a local representation, with the number of regions that can be distinguished now being in the order of n choose k . See Bengio (2009) for a brief overview of the literature on sparse representations.

3.2 Depth

Depth is a notion borrowed from complexity theory, and that is defined for *circuits*. A circuit is a directed acyclic graph where each node is associated with a computation, and whose output results are used by the successors of that node. Input nodes have no predecessor and output nodes have no successor. The depth of a circuit is the longest path from an input to an output node. A long-standing question in complexity theory is the extent to which depth-limited circuits can represent functions as efficiently as deeper circuits. A depth-2 circuit (with appropriate choice of computational elements, e.g. logic gates or formal neurons) can compute or approximate any function, but it may require an exponentially large number of nodes. This is a relevant question for machine learning, because many learning algorithms learn “shallow architectures” (Bengio and LeCun, 2007), typically of depth 1 (linear predictors) or 2 (most non-parametric predictors). If AI tasks

require deeper circuits (and human brains certainly appear deep), then we should find ways to incorporate depth into our learning algorithms. The consequences of using a too shallow predictor would be that it may not generalize well, unless given huge numbers of examples and capacity (i.e., computational resources and statistical resources).

The early results on the limitations of shallow circuits regard functions such as the parity function (Yad, 1985), showing that logic gates circuits of depth-2 require exponential size to implement d -bit parity where a deep circuit of depth $O(\log(d))$ could implement it with $O(d)$ nodes. Håstad (1986) then showed that there are functions computable with a polynomial-size logic gate circuit of depth k that require exponential size when restricted to depth $k - 1$ (Håstad, 1986). Interestingly, a similar result was proven for the case of circuits made of linear threshold units (formal neurons; Håstad and Goldmann, 1991), when trying to represent a particular family of functions. A more recent result brings an example of a very large class of functions that cannot be efficiently represented with a small-depth circuit (Braverman, 2011). It is particularly striking that the main theorem regards the representation of functions that capture dependencies in joint distributions. Basically, dependencies that involve more than r variables are difficult to capture by shallow circuits. An r -independent distribution is one that cannot be distinguished from the uniform distribution when looking only at r variables at a time. The proof of the main theorem (which concerns distribution over bit vectors) relies on the fact that order- r polynomials over the reals cannot capture r -independent distributions. The main result is that bounded-depth circuits cannot distinguish data generated by r -independent distributions from independent noisy bits. We have also recently shown (Bengio and Delalleau, 2011) results for *sum-product networks* (where nodes either compute sums or products, over the reals). We present these results in more details below as an example of the advantage brought by depth in terms of the efficiency of the representation: we found two families of polynomials that can be efficiently represented with depth- d circuits, but require exponential size with depth-2 circuits. Interestingly, sum-product networks were recently proposed to efficiently represent high-dimensional joint distributions (Poon and Domingos, 2010).

Besides the complexity-theory hints at their representational advantages, there are other motivations for studying learning algorithms which build a deep architecture. The earliest one is simply inspiration from brains. By putting together anatomical knowledge and measures of the time taken for signals to travel from the retina to the frontal cortex and then to motor neurons (about 100 to 200ms), one can gather that at least 5 to 10 feedforward levels are involved for some of the simplest visual object recognition tasks. Slightly more complex vision tasks require iteration and feedback top-down signals, multiplying the overall depth by an extra factor of 2 to 4 (to about half a second).

Another motivation derives from what we know of cognition and abstractions: as argued by Bengio (2009), it is natural for humans to represent concepts at one level of abstraction as the *composition* of concepts at lower levels. Engineers often craft representations at multiple levels, with higher levels obtained by

transformation of lower levels. Instead of a flat `main` program, software engineers structure their code to obtain plenty of *re-use*, with functions and modules reusing other functions and modules. This inspiration is directly linked to machine learning: deep architectures appear well suited to *represent higher-level abstractions* because they lend themselves to *re-use*. For example, some of the features that are useful for one task may be useful for another, making Deep Learning particularly well suited for *transfer learning* and *multi-task learning* (Caruana, 1995; Collobert and Weston, 2008; Bengio *et al.*, 2011; Bengio, 2011). Here one is exploiting the existence of underlying common explanatory factors that are useful for multiple tasks. This is also true of *semi-supervised learning*, which exploits connections between the input distribution $P(X)$ and a target conditional distribution $P(Y|X)$ (see Weston *et al.* (2008) for a first application of Deep Learning to semi-supervised learning). In general these two distributions, seen as functions of x , may be unrelated to each other. But in the world around us, it is often the case that some of the factors that shape the input variables X are predictive of the output variables Y . Deep Learning relies heavily on unsupervised or semi-supervised learning, and assumes that *representations of X that are useful to capture $P(X)$ are also in part useful to capture $P(Y|X)$* . An extensive study by Erhan *et al.* (2010) has explored the question of whether and how this prior may explain the success of the *greedy layer-wise unsupervised pre-training* recipe followed in many Deep Learning algorithms, and explained in Sect. 4.

3.3 A Deep Sum-Product Networks Case Study

Poon and Domingos (2010, 2011) introduced deep **sum-product networks** as a method to compute partition functions of tractable graphical models. These networks are analogous to traditional artificial neural networks but with nodes that compute either products or weighted sums of their inputs. In this setting the advantage brought by depth may not be obvious: after all, the output value can always be written as a sum of products of input variables (possibly raised to some power), and consequently it is easily rewritten as a shallow network with a sum output unit and product hidden units.

The argument supported by our theoretical analysis (Bengio and Delalleau, 2011) is that a deep architecture is able to compute some functions much more efficiently than a shallow one. Here we measure “efficiency” in terms of the number of computational units in the network. Bengio (2009) suggested that some polynomials could be represented more efficiently by deep sum-product networks, but without providing any formal statement or proofs. We partly addressed this void by demonstrating families of circuits for which a deep architecture can be exponentially more efficient than a shallow one in the context of real-valued polynomials².

² Here we restrict our definition of “sum-product networks” to those networks whose summation units have positive incoming weights, even though some of our results still hold for networks with non-positive weights (Bengio and Delalleau, 2011).

In the following we briefly review our main results, in which we consider two families of functions represented by deep sum-product networks (denoted by \mathcal{F} and \mathcal{G}). For each family, we establish a lower bound on the minimal number of hidden units a shallow (depth-2) sum-product network would require to represent a function of this family, showing it is much less efficient than the deep representation.

The first family of functions we study is $\mathcal{F} = \cup_{n \geq 4} \mathcal{F}_n$, where \mathcal{F}_n is made of functions built from deep sum-product networks with $n = 2^k$ inputs and (even) depth k that alternate binary product and sum layers (Fig. 2 for the simplest case, \mathcal{F}_4).

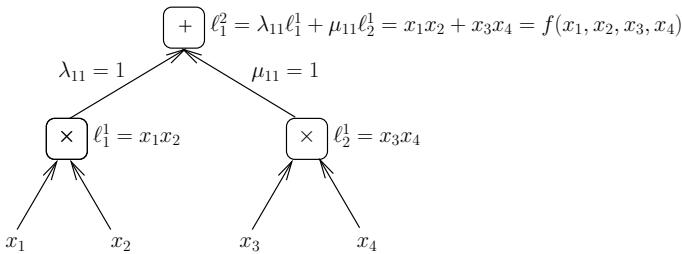


Fig. 2. Sum-product network computing some $f \in \mathcal{F}_4$, i.e., with $n=4$ inputs and depth $k = \log_2 n = 2$

The second family of functions is $\mathcal{G} = \cup_{n \geq 2, i \geq 0} \mathcal{G}_{in}$ such that the sub-family \mathcal{G}_{in} is made of sum-product networks with n input variables and depth $2i + 1$, that alternate sum and product layers. Each sum or product unit takes $n - 1$ units from the previous layer as inputs. An example of a network belonging to $\mathcal{G}_{1,3}$ is shown in Fig. 3 (it has unit summation weights to keep the figure easy to read). Note that contrary to family \mathcal{F} , depth and input size can be varied independently for networks in \mathcal{G} .

The main result for family \mathcal{F} is that any shallow sum-product network computing a function in \mathcal{F}_n must have at least $2\sqrt{n}-1$ hidden units. The high-level proof sketch consists in the following steps (Bengio and Delalleau, 2011):

1. Show that the number of unique products found in the expanded polynomial representation of $f \in \mathcal{F}_n$ is $2\sqrt{n}-1$.
2. Prove that the only possible architecture for a shallow sum-product network to compute f is to have a hidden layer made of product units, with a sum unit as output.
3. Conclude that the number of hidden units in step 2 must be at least the number of unique products computed in step 1.

For family \mathcal{G} , we obtain that a shallow sum-product network computing $g_{in} \in \mathcal{G}_{in}$ must have at least $(n-1)^i$ hidden units. The proof relies on a similar idea, i.e. we use a lower bound on the number of products found in the expanded polynomial expansion of g to bound the number of hidden units in a shallow sum-product

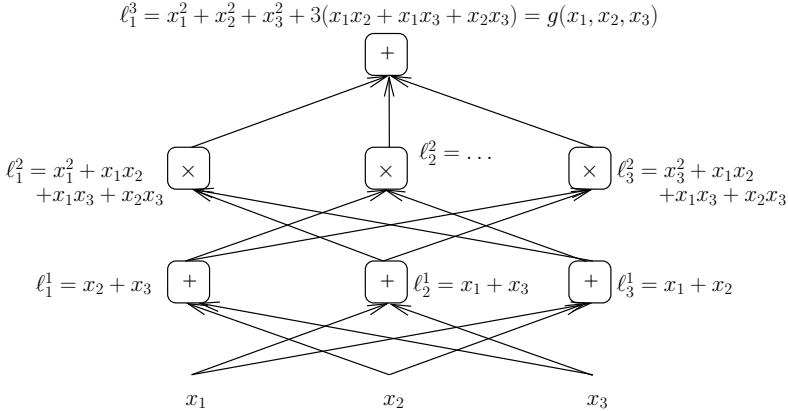


Fig. 3. Sum-product network computing some $g \in \mathcal{G}_{1,3}$

network with summation output. In addition, the final result uses the degree of the output polynomial, which is $(n - 1)^i$, to bound the number of hidden units in a shallow sum-product network with product output (also proving there can be no product units in the hidden layer).

In summary, we obtain that **functions in families \mathcal{F} and \mathcal{G} can be computed by a deep sum-product network with exponentially less units than when computed by a shallow sum-product network**. This motivates using deep sum-product networks to obtain more efficient representations.

4 A Zoo of Learning Algorithms

Greedy Layer-Wise Unsupervised Feature Learning

Whereas early efforts at training deep architectures were unsuccessful (Bengio and LeCun, 2007), a major breakthrough in Deep Learning methods came about with the use of *layer-wise unsupervised learning* (Hinton *et al.*, 2006; Bengio *et al.*, 2007; Ranzato *et al.*, 2007), as a way to initialize a deep supervised neural network.

Deep Learning usually occurs in two phases: first, unsupervised, layer-wise training, and second, supervised training of a classifier that exploits what has been done in the first phase. In the unsupervised phase, each layer is added and trained greedily, i.e., keeping the earlier layers fixed and ignoring the future interactions with additional layers. Each layer uses the representation learned by the previous layer as input that it tries to model and transform to a new and better representation. Many unsupervised learning algorithms are being explored for the first phase, including various methods to train Restricted Boltzmann Machines (RBMs) (Freund and Haussler, 1994; Hinton *et al.*, 2006; Tieleman, 2008; Salakhutdinov and Hinton, 2009; Desjardins *et al.*, 2010) or Deep Boltzmann Machines (Salakhutdinov and Hinton, 2010; Lee *et al.*, 2009), different flavours of auto-encoders (Bengio *et al.*, 2007; Vincent *et al.*,

2008; Larochelle *et al.*, 2009), and other sparse encoder-decoder systems (Ranzato *et al.*, 2007; Kavukcuoglu *et al.*, 2009).

The objective stated in the Deep Learning literature is to discover powerful representation-learning algorithms, mostly thanks to unsupervised learning procedures. Ideally, such representations should somehow capture the salient factors of variation that explain the data, and this can be tested by attempting to use these learned representations to predict some of these factors, e.g., in a classification problem.

Boltzmann Machines. The first unsupervised learning algorithm (Hinton and Salakhutdinov, 2006; Hinton *et al.*, 2006) that has been proposed for training each layer of a deep architecture is based on a Restricted Boltzmann Machine (Smolensky, 1986), which is an undirected graphical model that is a particular form of Boltzmann Machine (Hinton *et al.*, 1984). A Boltzmann Machine is an undirected graphical model for observed variable x based on latent variable h is specified by an *energy function* $\mathbb{E}(x, h)$:

$$P(x, h) = \frac{e^{-\mathbb{E}(x, h)}}{Z}$$

where Z is a normalization constant called the partition function. A Boltzmann machine is one where $\mathbb{E}(x, h)$ is a second-order polynomial in (x, h) , e.g.,

$$\mathbb{E}(x, h) = h'Wx + h'Uh + x'Vx + b'h + c'x$$

and in general both x and h are considered to be binary vectors, which makes Z intractable except when both x and h have very few components. The coefficients $\theta = (W, U, V, b, c)$ of that second-order polynomial are the parameters of the model. Given an observed x , the inference $P(h|x)$ is generally intractable but can be estimated by sampling from a Monte-Carlo Markov Chain (MCMC), e.g. by Gibbs sampling, or using loopy belief, variational or mean-field approximations. Even though computing the energy is easy, marginalizing over h in order to compute the likelihood $P(x)$ is generally intractable, so that the exact log-likelihood gradient is also intractable. However, several algorithms have been proposed in recent years to estimate the gradient, most of them based on the following decomposition into the so-called “positive phase part” (x is fixed to the observed value, the gradient term tends to decrease the associated energy) and “negative phase part” (both x and h are sampled according to P , and the gradient term tends to increase their energy):

$$\frac{\partial}{\partial \theta}(-\log P(x)) = E_h \left[\frac{\partial \mathbb{E}(x, h)}{\partial \theta} | x \right] - E_{x,h} \left[\frac{\partial \mathbb{E}(x, h)}{\partial \theta} \right].$$

Even though a Boltzmann Machine is a parametric model when we consider the dimensionality n_h of h to be fixed, in practice one allows n_h to vary, making it a non-parametric model. With n_h large enough, one can model any discrete distribution: Le Roux and Bengio (2008) showed that Restricted Boltzmann Machines (described below) are universal approximators, and since they are special cases

of Boltzmann Machines, Boltzmann Machines also are universal approximators. On the other hand with $n_h > 0$ the log-likelihood is not anymore convex in the parameters, and training can potentially get stuck in one of many local minima.

The Restricted Boltzmann Machine (RBM) is a Boltzmann machine without lateral interactions, i.e., $U = 0$ and $V = 0$. It turns out that the positive phase part of the gradient can be computed exactly and tractably in the easier special case of the RBM, because $P(h|x)$ factorizes into $\prod_i P(h_i|x)$. Similarly $P(x|h)$ factorizes into $\prod_j P(x_j|h)$, which makes it possible to apply blocked Gibbs sampling (sampling h given x , then x given h , again h given x , etc.). For a trained RBM, the learned representation $R(x)$ of its input x is usually taken to be $E[h|x]$, as a heuristic.

RBM s are typically trained by stochastic gradient descent, using a noisy (and generally biased) estimator of the above log-likelihood gradient. The first gradient estimator that was proposed for RBMs is the Contrastive Divergence estimator (Hinton, 1999; Hinton *et al.*, 2006), and it has a particularly simple form: the negative phase gradient is obtained by starting a very short chain (usually just one step) at the observed x and replacing the above expectations by the corresponding samples. In practice, it has worked very well for unsupervised pre-training meant to initialize each layer of a deep supervised (Hinton *et al.*, 2006; Bengio *et al.*, 2007; Erhan *et al.*, 2010) or unsupervised (Hinton and Salakhutdinov, 2006) neural network.

Another common way to train RBMs is based on the Stochastic Maximum Likelihood (SML) estimator (Younes, 1999) of the gradient, also called Persistent Contrastive Divergence (PCD; Tielemans, 2008) when it was introduced for RBMs. The idea is simply to keep sampling negative phase x 's (e.g. by blocked Gibbs sampling) even though the parameters are updated once in a while, i.e., without restarting a new chain each time an update is done. It turned out that SML yields RBMs with much better likelihood, whereas CD updates sometimes give rise to worsening likelihood and suffer from other issues (Desjardins *et al.*, 2010). Theory suggests (Younes, 1999) this is a good estimator if the parameter changes are small, but practice revealed (Tielemans, 2008) that it worked even for large updates, in fact giving rise to faster mixing (Tielemans and Hinton, 2009; Breuleux *et al.*, 2011). This is happening because learning actually interacts with sampling in a useful way, pushing the MCMC out of the states it just visited. This principle may also explain some of the fast mixing observed in a related approach called Herding (Welling, 2009; Breuleux *et al.*, 2011).

RBM s can be stacked to form a Deep Belief Network (DBN), a hybrid of directed and undirected graphical model components, which has an RBM to characterize the interactions between its top two layers, and then generates the input through a directed belief network. See Bengio (2009) for a deeper treatment of Boltzmann Machines, RBMs, and Deep Belief Networks.

Auto-encoders are neural networks which are trained to reconstruct their input (Rumelhart *et al.*, 1986; Bourlard and Kamp, 1988; Hinton and Zemel, 1994). A one-hidden layer auto-encoder is very similar to an RBM and its reconstruction error gradient can be seen as an approximation of the RBM log-likelihood

gradient (Bengio and Delalleau, 2009). Both RBMs and auto-encoders can be used as one-layer unsupervised learning algorithms that give rise to a new representation of the input or of the previous layer. In the same year that RBMs were successfully proposed for unsupervised pre-training of deep neural networks, auto-encoders were also shown to help initialize deep neural networks much better than random initialization (Bengio *et al.*, 2007). However, ordinary auto-encoders generally performed worse than RBMs, and were unsatisfying because they could potentially learn a useless identity transformation when the representation size was larger than the input (the so-called “overcomplete” case).

Sparse Coding was introduced in computational neuroscience (Olshausen and Field, 1997) and produced filters very similar to those observed in cortex visual area V1 (before similar filters were achieved with RBMs, sparse predictive decomposition, and denoising auto-encoders, below). It corresponds to a linear directed graphical model with a continuous-valued latent variable associated with a sparsity prior (Student or Laplace, the latter corresponding to an L1 penalty on the value of the latent variable). This is like an auto-encoder, but without a parametric encoder, only a parametric decoder. The “encoding” corresponds to inference (finding the most likely hidden code associated with observed visible input) and involves solving a lengthy but convex optimization problem and much work has been devoted to speeding it up. A very interesting way to do so is with **Predictive Sparse Decomposition** (Kavukcuoglu *et al.*, 2008), in which one learns a parametric encoder that approximates the result of the sparse coding inference (and in fact changes the solution so that both approximate encoding and decoding work well). Such models based on approximate inference were the first successful examples of stacking a sparse encoding (Ranzato *et al.*, 2007; Jarrett *et al.*, 2009) into a deep architecture (fine-tuned for supervised classification afterwards, as per the above greedy-layerwise recipe).

Score Matching is an alternative statistical estimation principle (Hyvärinen, 2005) when the maximum likelihood framework is not tractable. It can be applied to models of continuous-valued data when the probability function can be computed tractably up to its normalization constant (which is the case for RBMs), i.e., it has a tractable energy function. The *score* of the model is the partial derivative of the log-likelihood with respect to the input, and indicates in which direction the likelihood would increase the most, from a particular input x . Score matching is based on minimizing the squared difference between the score of the model and a target score. The latter is in general unknown but the score match can nonetheless be rewritten in terms of the expectation (under the data generating process) of first and (diagonal) second derivatives of the energy with respect to the input, which correspond to a tractable computation.

Denoising Auto-encoders were first introduced by Vincent *et al.* (2008) to bypass the frustrating limitations of auto-encoders mentioned above. Auto-encoders are only meant to learn a “bottleneck”, a reduced-dimension representation. The idea of Denoising Auto-Encoders (DAE) is simple: feed the encoder/decoder system with a *stochastically corrupted input*, but ask it to *reconstruct the clean input* (as one would typically do to train any denoising system).

This small change turned out to systematically yield better results than those obtained with ordinary auto-encoders, and similar or better than those obtained with RBMs on a benchmark of several image classification tasks (Vincent *et al.*, 2010). Interestingly, the denoising error can be linked in several ways to the likelihood of a generative model of the distribution of the uncorrupted examples (Vincent *et al.*, 2008; Vincent, 2011), and in particular through the Score Matching proxy for log-likelihood (Vincent, 2011): the denoising error corresponds to a form of *regularized* score matching criterion (Kingma and LeCun, 2010). The link also sheds light on why a denoising auto-encoder captures the input distribution. The difference vector between the reconstruction and the corrupted input is the model’s guess as to the direction of greatest increase in the likelihood (starting from a corrupted example), whereas the difference vector between the corrupted input and the clean original is nature’s hint of a direction of greatest increase in likelihood (since a noisy version of a training example is very likely to have a much lower probability than the original under the data generating distribution). The difference of these two differences is just the denoising reconstruction error residue.

Noise-Contrastive Estimation is another estimation principle which can be applied when the energy function can be computed but not the partition function (Gutmann and Hyvärinen, 2010). It is based on training not only from samples of the target distribution but also from samples of an auxiliary “background” distribution (e.g. a flat Gaussian). The partition function is considered like a free parameter (along with the other parameters) in a kind of logistic regression trained to predict the probability that a sample belongs to the target distribution vs the background distribution.

Semi-supervised Embedding is an interesting and different way to use unlabeled data to learn a representation (e.g., in the hidden layers of a deep neural network), based on a hint about *pairs of examples* (Weston *et al.*, 2008). If two examples in a pair are expected to have a similar semantic, then their representations should be encouraged to be similar, whereas otherwise their representations should be at least some distance away. This idea was used in unsupervised and semi-supervised contexts (Chopra *et al.*, 2005; Hadsell *et al.*, 2006; Weston *et al.*, 2008), and originates in the much older idea of *siamese networks* (Bromley *et al.*, 1993).

Contractive Autoencoders (Rifai *et al.*, 2011) minimize a training criterion that is the sum of a reconstruction error and a “contraction penalty”, which encourages the learnt representation $h(x)$ to be as invariant as possible to the input x , while still allowing to distinguish the training examples from each other (i.e., to reconstruct them). As a consequence, the representation is faithful to changes in input space in the directions of the manifold near which examples concentrate, but it is highly contractive in the orthogonal directions. This is similar in spirit to a PCA (which only keeps the leading directions of variation and completely ignores the others), but is softer (no hard cutting at a particular dimension), is non-linear and can contract in different directions depending on where one looks in the input space (hence can capture non-linear manifolds).

To prevent a trivial solution in which the encoder weights go to zero and the decoder weights to infinity, the contractive autoencoder uses tied weights (the decoder weights are forced to be the transpose of the encoder weights). Because of the contractive criterion, what we find empirically is that for any particular input example, many of the hidden units saturate while a few remain sensitive to changes in the input (corresponding to changes in the directions of changes expected under the data distribution). That subset of active units changes as we move around in input space, and defines a kind of *local chart*, or local coordinate system, in the neighborhood of each input point. This can be visualized to some extent by looking at the singular values and singular vectors of the Jacobian matrix J (containing the derivatives of each hidden unit output with respect to each input unit). Contrary to other autoencoders, one tends to find only few dominant eigenvalues, and their number corresponds to a local rank or local dimension (which can change as we move in input space). This is unlike other dimensionality reduction algorithms in which the number of dimensions is fixed by hand (rather than learnt) and fixed across the input domain. In fact the learnt representation can be overcomplete (larger than the input): it is only in the sense of its Jacobian that it has an effective small dimensionality for any particular input point. The large number of hidden units can be exploited to model complicated non-linear manifolds.

5 Principles, Challenges and Ideas Ahead

What lies beyond the principle of local generalization which has already been very successful in machine learning? The principles of distributed (possibly sparse) representations and deep circuits make possible other ways to generalize. Both can exploit a combinatorial effect in order to characterize and differentiate a number of input regions that is exponentially larger than the number of parameters.

However, these principles also come with challenges, notably a more difficult non-convex optimization problem. The greedy layer-wise unsupervised pre-training trick has served well, but more needs to be done in order to globally train these deep architectures. This optimization difficulty means that the optimization problem is not cleanly decoupled from the modeling choices. Some models may work well in practice because the optimization is easier.

Representation learning algorithms have been found empirically to partly disentangle the underlying factors of variation, such as geometric factors of variation (Goodfellow *et al.*, 2009), or domain vs sentiment in sentiment analysis (Glorot *et al.*, 2011). This means that some learned features (some component of the representation) are more invariant to some factors of variation (compared to the raw input) and more sensitive to others. Perhaps the most exciting challenge ahead is the following: why is this apparent disentangling happening, and can we go even further in that direction? We already know some tricks which seem to help this disentangling: independence (e.g., as in ICA), sparsity (e.g., as in sparse auto-encoders, sparse RBMs, or sparse denoising auto-encoders),

grouping (forcing some groups of learned features to behave as a group, e.g., encouraging all of the units in a group to be off together), and slowness (forcing some units to respond in a temporally coherent manner). We propose to invest more towards understanding all of these better and exploiting this understanding to yield learning algorithms that better disentangle the underlying factors of variation in AI-related tasks.

References

- Attwell, D., Laughlin, S.B.: An energy budget for signaling in the grey matter of the brain. *Journal of Cerebral Blood Flow And Metabolism* 21, 1133–1145 (2001)
- Barron, A.E.: Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. on Information Theory* 39, 930–945 (1993)
- Bengio, Y.: Learning deep architectures for AI. *Foundations and Trends in Machine Learning* 2(1), 1–127 (2009); Also published as a book. Now Publishers
- Bengio, Y.: Deep learning of representations for unsupervised and transfer learning. In: *Workshop on Unsupervised and Transfer Learning, ICML 2011* (2011)
- Bengio, Y., Delalleau, O.: Justifying and generalizing contrastive divergence. *Neural Computation* 21(6), 1601–1621 (2009)
- Bengio, Y., Delalleau, O.: Shallow versus deep sum-product networks. In: *The Learning Workshop*, Fort Lauderdale, Florida (2011)
- Bengio, Y., LeCun, Y.: Scaling learning algorithms towards AI. In: Bottou, L., Chapelle, O., DeCoste, D., Weston, J. (eds.) *Large Scale Kernel Machines*. MIT Press, Cambridge (2007)
- Bengio, Y., Monperrus, M.: Non-local manifold tangent learning. In: Saul, L., Weiss, Y., Bottou, L. (eds.) *Advances in Neural Information Processing Systems, NIPS 2004*, vol. 17, pp. 129–136. MIT Press, Cambridge (2005)
- Bengio, Y., Delalleau, O., Le Roux, N.: The curse of highly variable functions for local kernel machines. In: Weiss, Y., Schölkopf, B., Platt, J. (eds.) *Advances in Neural Information Processing Systems (NIPS 2005)*, vol. 18, pp. 107–114. MIT Press, Cambridge (2006)
- Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) *Advances in Neural Information Processing Systems (NIPS 2006)*, vol. 19, pp. 153–160. MIT Press, Cambridge (2007)
- Bengio, Y., Delalleau, O., Simard, C.: Decision trees do not generalize to new variations. *Computational Intelligence* 26(4), 449–467 (2010)
- Bengio, Y., Bastien, F., Bergeron, A., Boulanger-Lewandowski, N., Breuel, T., Chherawala, Y., Cisse, M., Côté, M., Erhan, D., Eustache, J., Glorot, X., Muller, X., Pannetier Lebeuf, S., Pascanu, R., Rifai, S., Savard, F., Sicard, G.: Deep learners benefit more from out-of-distribution examples. In: *JMLR W&CP: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011* (2011)
- Bourlard, H., Kamp, Y.: Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics* 59, 291–294 (1988)
- Braverman, M.: Poly-logarithmic independence fools bounded-depth boolean circuits. *Communications of the ACM* 54(4), 108–115 (2011)
- Breuleux, O., Bengio, Y., Vincent, P.: Quickly generating representative samples from an rbm-derived process. *Neural Computation* 23(8), 2058–2073 (2011)

- Bromley, J., Benz, J., Bottou, L., Guyon, I., Jackel, L., LeCun, Y., Moore, C., Sackinger, E., Shah, R.: Signature verification using a siamese time delay neural network. In: Advances in Pattern Recognition Systems using Neural Network Technologies, pp. 669–687. World Scientific, Singapore (1993)
- Caruana, R.: Learning many related tasks at the same time with backpropagation. In: Tesauro, G., Touretzky, D., Leen, T. (eds.) Advances in Neural Information Processing Systems (NIPS 1994), vol. 7, pp. 657–664. MIT Press, Cambridge (1995)
- Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR 2005). IEEE Press, Los Alamitos (2005)
- Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Cohen, W.W., McCallum, A., Roweis, S.T. (eds.) Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML 2008), pp. 160–167. ACM, New York (2008)
- Desjardins, G., Courville, A., Bengio, Y., Vincent, P., Delalleau, O.: Tempered Markov chain monte carlo for training of restricted Boltzmann machine. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010), pp. 145–152 (2010)
- Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? Journal of Machine Learning Research 11, 625–660 (2010)
- Freund, Y., Haussler, D.: Unsupervised learning of distributions on binary vectors using two layer networks. Technical Report UCSC-CRL-94-25, University of California, Santa Cruz (1994)
- Glorot, X., Bordes, A., Bengio, Y.: Domain adaptation for large-scale sentiment classification: A deep learning approach. In: Proceedings of the Twenty-eighth International Conference on Machine Learning, ICML 2011 (2011)
- Goodfellow, I., Le, Q., Saxe, A., Ng, A.: Measuring invariances in deep networks. In: Bengio, Y., Schuurmans, D., Williams, C., Lafferty, J., Culotta, A. (eds.) Advances in Neural Information Processing Systems (NIPS 2009), vol. 22, pp. 646–654 (2009)
- Gutmann, M., Hyvärinen, A.: Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In: Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010 (2010)
- Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR 2006), pp. 1735–1742. IEEE Press, Los Alamitos (2006)
- Hadsell, R., Erkan, A., Sermanet, P., Scoffier, M., Muller, U., LeCun, Y.: Deep belief net learning in a long-range vision system for autonomous off-road driving. In: Proc. Intelligent Robots and Systems (IROS 2008), pp. 628–633 (2008)
- Håstad, J.: Almost optimal lower bounds for small depth circuits. In: Proceedings of the 18th Annual ACM Symposium on Theory of Computing, pp. 6–20. ACM Press, Berkeley (1986)
- Håstad, J., Goldmann, M.: On the power of small-depth threshold circuits. Computational Complexity 1, 113–129 (1991)
- Hinton, G.E.: Learning distributed representations of concepts. In: Proceedings of the Eighth Annual Conference of the Cognitive Science Society, Amherst, pp. 1–12. Lawrence Erlbaum, Hillsdale (1986)
- Hinton, G.E.: Connectionist learning procedures. Artificial Intelligence 40, 185–234 (1989)

- Hinton, G.E.: Products of experts. In: Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN), vol. 1, pp. 1–6. IEE, Edinburgh (1999)
- Hinton, G.E., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. *Science* 313(5786), 504–507 (2006)
- Hinton, G.E., Zemel, R.S.: Autoencoders, minimum description length, and helmholtz free energy. In: Cowan, D., Tesauro, G., Alspector, J. (eds.) *Advances in Neural Information Processing Systems (NIPS 1993)*, vol. 6, pp. 3–10. Morgan Kaufmann Publishers, Inc., San Francisco (1994)
- Hinton, G.E., Sejnowski, T.J., Ackley, D.H.: Boltzmann machines: Constraint satisfaction networks that learn. Technical Report TR-CMU-CS-84-119, Carnegie-Mellon University, Dept. of Computer Science (1984)
- Hinton, G.E., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. *Neural Computation* 18, 1527–1554 (2006)
- Hyvärinen, A.: Estimation of non-normalized statistical models using score matching. *Journal of Machine Learning Research* 6, 695–709 (2005)
- Jarrett, K., Kavukcuoglu, K., Ranzato, M., LeCun, Y.: What is the best multi-stage architecture for object recognition? In: Proc. International Conference on Computer Vision (ICCV 2009), pp. 2146–2153. IEEE, Los Alamitos (2009)
- Kavukcuoglu, K., Ranzato, M., LeCun, Y.: Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Computational and Biological Learning Lab, Courant Institute, NYU. Tech Report CBLL-TR-2008-12-01 (2008)
- Kavukcuoglu, K., Ranzato, M., Fergus, R., LeCun, Y.: Learning invariant features through topographic filter maps. In: Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR 2009), pp. 1605–1612. IEEE, Los Alamitos (2009)
- Kingma, D., LeCun, Y.: Regularized estimation of image statistics by score matching. In: Lafferty, J., Williams, C.K.I., Shawe-Taylor, J., Zemel, R., Culotta, A. (eds.) *Advances in Neural Information Processing Systems*, vol. 23, pp. 1126–1134 (2010)
- Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: Ghahramani, Z. (ed.) *Proceedings of the Twenty-fourth International Conference on Machine Learning (ICML 2007)*, pp. 473–480. ACM, New York (2007)
- Larochelle, H., Erhan, D., Vincent, P.: Deep learning using robust interdependent codes. In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, pp. 312–319 (2009)
- Le Roux, N., Bengio, Y.: Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation* 20(6), 1631–1649 (2008)
- Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Bottou, L., Littman, M. (eds.) *Proceedings of the Twenty-sixth International Conference on Machine Learning (ICML 2009)*. ACM, Montreal (2009)
- Lennie, P.: The cost of cortical computation. *Current Biology* 13(6), 493–497 (2003)
- Mesnil, G., Dauphin, Y., Glorot, X., Rifai, S., Bengio, Y., Goodfellow, I., Lavoie, E., Muller, X., Desjardins, G., Warde-Farley, D., Vincent, P., Courville, A., Bergstra, J.: Unsupervised and transfer learning challenge: a deep learning approach. In: Workshop on Unsupervised and Transfer Learning, ICML 2011 (2011)
- Mobahi, H., Collobert, R., Weston, J.: Deep learning from temporal coherence in video. In: Bottou, L., Littman, M. (eds.) *Proceedings of the 26th International Conference on Machine Learning*, pp. 737–744. Omnipress, Montreal (2009)

- Olshausen, B.A., Field, D.J.: Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Research* 37, 3311–3325 (1997)
- Osindero, S., Hinton, G.E.: Modeling image patches with a directed hierarchy of markov random field. In: Platt, J., Koller, D., Singer, Y., Roweis, S. (eds.) *Advances in Neural Information Processing Systems (NIPS 2007)*, vol. 20, pp. 1121–1128. MIT Press, Cambridge (2008)
- Poon, H., Domingos, P.: Sum-product networks: A new deep architecture. In: *NIPS, Workshop on Deep Learning and Unsupervised Feature Learning*, Whistler, Canada (2010)
- Poon, H., Domingos, P.: Sum-product networks for deep learning. In: *Learning Workshop*. FL, Fort Lauderdale (2011)
- Ranzato, M., Poultney, C., Chopra, S., LeCun, Y.: Efficient learning of sparse representations with an energy-based model. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) *Advances in Neural Information Processing Systems (NIPS 2006)*, vol. 19, pp. 1137–1144. MIT Press, Cambridge (2007)
- Ranzato, M., Boureau, Y.-L., LeCun, Y.: Sparse feature learning for deep belief networks. In: Platt, J., Koller, D., Singer, Y., Roweis, S. (eds.) *Advances in Neural Information Processing Systems (NIPS 2007)*, vol. 20, pp. 1185–1192. MIT Press, Cambridge (2008)
- Rifai, S., Vincent, P., Muller, X., Glorot, X., Bengio, Y.: Contractive auto-encoders: Explicit invariance during feature extraction. In: *Proceedings of the Twenty-eighth International Conference on Machine Learning, ICML 2011* (2011)
- Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. *Nature* 323, 533–536 (1986)
- Salakhutdinov, R., Hinton, G.E.: Semantic hashing. In: *Proceedings of the 2007 Workshop on Information Retrieval and Applications of Graphical Models (SIGIR 2007)*. Elsevier, Amsterdam (2007)
- Salakhutdinov, R., Hinton, G.E.: Deep Boltzmann machines. In: *Proceedings of The Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, vol. 5, pp. 448–455 (2009)
- Salakhutdinov, R., Hinton, G.E.: An efficient learning procedure for deep Boltzmann machines. Technical Report MIT-CSAIL-TR-2010-037, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology (2010)
- Salakhutdinov, R., Mnih, A., Hinton, G.E.: Restricted Boltzmann machines for collaborative filtering. In: Ghahramani, Z. (ed.) *Proceedings of the Twenty-fourth International Conference on Machine Learning (ICML 2007)*, pp. 791–798. ACM, New York (2007)
- Smolensky, P.: Information processing in dynamical systems: Foundations of harmony theory. In: Rumelhart, D.E., McClelland, J.L. (eds.) *Parallel Distributed Processing*, vol. 1, ch. 6, pp. 194–281. MIT Press, Cambridge (1986)
- Tieleman, T.: Training restricted boltzmann machines using approximations to the likelihood gradient. In: Cohen, W.W., McCallum, A., Roweis, S.T. (eds.) *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML 2008)*, pp. 1064–1071. ACM, New York (2008)
- Tieleman, T., Hinton, G.: Using fast weights to improve persistent contrastive divergence. In: Bottou, L., Littman, M. (eds.) *Proceedings of the Twenty-sixth International Conference on Machine Learning (ICML 2009)*, pp. 1033–1040. ACM, New York (2009)
- Vincent, P.: A connection between score matching and denoising autoencoders. *Neural Computation* 23(7), 1661–1674 (2011)

- Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.-A.: Extracting and composing robust features with denoising autoencoders. In: Cohen, W.W., McCallum, A., Roweis, S.T. (eds.) *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML 2008)*, pp. 1096–1103. ACM, New York (2008)
- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11, 3371–3408 (2010)
- Welling, M.: Herding dynamic weights for partially observed random field models. In: *Proceedings of the 25th Conference in Uncertainty in Artificial Intelligence (UAI 2009)*. Morgan Kaufmann, San Francisco (2009)
- Weston, J., Ratle, F., Collobert, R.: Deep learning via semi-supervised embedding. In: Cohen, W.W., McCallum, A., Roweis, S.T. (eds.) *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML 2008)*, pp. 1168–1175. ACM, New York (2008)
- Wolpert, D.H.: The lack of a priori distinction between learning algorithms. *Neural Computation* 8(7), 1341–1390 (1996)
- Yao, A.: Separating the polynomial-time hierarchy by oracles. In: *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, pp. 1–10 (1985)
- Younes, L.: On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics and Stochastic Reports* 65(3), 177–228 (1999)

Optimal Estimation

Jorma Rissanen

Helsinki Institute for Information Technology and Tampere University of Technology,
Finland

1 Modeling Problem

Data $Y = \{y_t : t = 1, 2, \dots, n\}$, or $Y|X = \{(y_t, x_{1,t}, x_{2,t}, \dots)\}$, X explanatory variables. Want to learn properties in Y expressed by set of distributions as models: $f(Y|X_s; \theta, s)$, where $\theta = \theta_1, \dots, \theta_{k(s)}$ real-valued parameters, s structure parameter: for picking the most important variables in X .

1.1 Models and Estimators

To simplify notations write $y_t, x_{1,t}, x_{2,t}, \dots$ as x_t ; structures determined by number k of real-valued parameters.

Classes of parametric models

$$\begin{aligned}\mathcal{M}_k &= \{f(x^n; \theta, k) : \theta \in \Omega^k \subset R^k\}; k \leq n \\ \mathcal{M} &= \{\mathcal{M}_k : k = 1, 2, \dots, K, K \leq n\}.\end{aligned}$$

Sets of estimator functions $\bar{\theta}(\cdot), \bar{k}(\cdot)$. Consider the distributions defined by estimators

$$\begin{aligned}\text{for fixed } k : \quad &\bar{f}(x^n; k) = f(x^n; \bar{\theta}(x^n), k) / \bar{C}_{k,n} \\ &\bar{C}_{k,n} = \int f(y^n; \bar{\theta}(y^n), k) dy^n \\ \text{in general : } &\bar{f}(x^n) = \bar{f}(x^n; \bar{k}(x^n)) / \bar{C}_n \\ &\bar{C}_n = \sum_k \int_{\bar{k}(y^n)=k} \bar{f}(y^n; k) dy^n\end{aligned}$$

Let $\hat{\theta}(\cdot), \hat{k}(\cdot)$ be the estimator that maximizes \bar{C}_n :

$$\hat{C}_n = \max_{\bar{\theta}(\cdot), \bar{k}(\cdot)} \bar{C}_n. \tag{1}$$

It also maximizes the probability or density $\hat{f}(x^n)$ on the observed data, which is taken as the single postulate for this theory of estimation. The maximum \hat{C}_n is called the maximum capacity, and it is also the maximum mutual information that any estimator can obtain about the models in the class.

Learning from Label Preferences*

Eyke Hüllermeier¹ and Johannes Fürnkranz²

¹ Philipps-Universität Marburg, Germany

² Technische Universität Darmstadt, Germany

Abstract. In this paper, we review the framework of learning (from) label preferences, a particular instance of preference learning. Following an introduction to the learning setting, we particularly focus on our own work, which addresses this problem via the learning by pairwise comparison paradigm. From a machine learning point of view, learning by pairwise comparison is especially appealing as it decomposes a possibly complex prediction problem into a certain number of learning problems of the simplest type, namely binary classification. We also discuss how a number of common machine learning tasks, such as multi-label classification, hierarchical classification or ordinal classification, may be addressed within the framework of learning from label preferences. We also briefly address theoretical questions as well as algorithmic and complexity issues.

Preference learning is a recent addition to the suite of learning tasks in machine learning [1], where the training information is typically not given in the form of scalar outputs, but instead in the form of pairwise comparisons expressing *preferences* between different objects. One can distinguish learning from *object preferences*, where the training data is given in the form of pairwise comparisons between objects, and learning from *label preferences*, where the training data is given in the form of pairwise comparisons between labels that are attached to the objects. In the former case, a common performance task is to rank a new set of objects (*object ranking*), whereas in the latter case, the performance task is to rank the set of labels for a new object (*label ranking*). Besides, the training information may also be given in the form of (ordinal) *preference degrees* attached to the objects, indicating an *absolute* (as opposed to a relative/comparative) assessment. If the task is to rank a new set of objects according to their preference degrees, we also speak of *instance ranking*. In this talk, we focus on the task of label ranking, and, in particular, on our own work in this area, which concentrates on learning a set of pairwise comparators (LPC) [2].

References

1. Fürnkranz, J., Hüllermeier, E. (eds.): Preference Learning. Springer, Heidelberg (2010)
2. Fürnkranz, J., Hüllermeier, E.: Preference learning and ranking by pairwise comparison. In: [1], pp. 65–82

* The full version of this paper is published in the Proceedings of the 14th International Conference on Discovery Science, Lecture Notes in Artificial Intelligence Vol. 6926.

Information Distance and Its Extensions*

Ming Li

School of Computer Science, University of Waterloo,
Waterloo, Ont. N2L 3G1, Canada
mli@uwaterloo.ca
<http://www.cs.uwaterloo.ca/~mli>

Abstract. Consider, in the most general sense, the space of all information carrying objects: a book, an article, a name, a definition, a genome, a letter, an image, an email, a webpage, a Google query, an answer, a movie, a music score, a Facebook blog, a short message, or even an abstract concept. Over the past 20 years, we have been developing a general theory of information distance in this space and applications of this theory. The theory is object-independent and application-independent. The theory is also unique, in the sense that no other theory is “better”. During the past 10 years, such a theory has found many applications. Recently we have introduced two extensions to this theory concerning multiple objects and irrelevant information. This expository article will focus on explaining the main ideas behind this theory, especially these recent extensions, and their applications. We will also discuss some very preliminary applications.

* The full version of this paper is published in the Proceedings of the 14th International Conference on Discovery Science, Lecture Notes in Artificial Intelligence Vol. 6926.

Iterative Learning from Positive Data and Counters

Timo Kötzing^{*,**}

Max-Planck-Institut für Informatik, Campus E1 4, 66123 Saarbrücken, Germany
`koetzing@mpi-inf.mpg.de`

Abstract. We analyze iterative learning in the limit from positive data with the additional information provided by a *counter*. The simplest *type* of counter provides the current iteration number (counting up from 0 to infinity), which is known to improve learning power over plain iterative learning.

We introduce five other (weaker) counter types, for example only providing some unbounded and non-decreasing sequence of numbers. Analyzing these types allows for understanding what properties of a counter can benefit learning.

For the iterative setting, we completely characterize the relative power of the learning criteria corresponding to the counter types. In particular, for our types, the only properties improving learning power are *unboundedness* and *strict monotonicity*.

Furthermore, we show that each of our types of counter improves learning power over weaker ones in *some* settings, and that, for iterative learning criteria with one of these types of counter, separations of learning criteria are necessarily witnessed by classes containing only infinite languages.

Keywords: Inductive Inference.

1 Introduction

We analyze the problem of algorithmically learning a description for a formal language (a computably enumerable subset of the natural numbers) when presented successively all and only the elements of that language. For example, a learner h might be presented more and more even numbers. After each new number, h may output a description of a language as its conjecture. The learner h might decide to output a program for the set of all multiples of 4, as long as no even number not divisible by 4 has been presented. Later, when h sees an even number not divisible by 4, it might change this guess to a program for the set of all multiples of 2.

* Timo Kötzing was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant no. NE 1182/5-1.

** The author would like to thank John Case, Sanjay Jain, Frank Stephan and Sandra Zilles for valuable and fruitful discussions.

Many criteria for deciding whether a learner h is *successful* on a language L have been proposed in the literature. Gold, in his seminal paper [Gol67], gave a first, simple learning criterion, *TxtEx-learning*¹, where a learner is *successful* iff, on every *text* for L (listing of all and only the elements of L) it eventually stops changing its conjectures, and its final conjecture is a correct description for the input sequence.

Trivially, each single, describable language L has a suitable constant function as an Ex-learner (this learner constantly outputs a description for L). Thus, we are interested for which *classes of languages* \mathcal{L} is there a *single learner* h learning *each* member of \mathcal{L} . This framework is known as *language learning in the limit* and has been studied extensively, using a wide range of learning criteria similar to TxtEx-learning (see, for example, the text book [JORS99]).

In this paper we are concerned with a memory limited variant of TxtEx-learning, namely *iterative learning* [Wie76, LZ96] (It). While in TxtEx-learning a learner may arbitrarily access previously presented data points, in iterative learning the learner only sees its previous conjecture and the latest data point. It is well known that this setting allows for learning strictly fewer classes of languages. The successive literature analyzed iterative learners with some additional resources, for example a *bounded example memory* [LZ96]; “long term” *finite memory states* [FKS95]; or *feedback learning*, i.e. the ability to ask for the containment of examples in previously seen data [LZ96, CJLZ99].

A different option for providing additional learning power for iterative learning was suggested in [CM08b], where *iterative with counter learning* was introduced. In this setting, a learner, in each iteration, has access to its previous conjecture, the latest datum, and the current iteration number (counting up from 0 to infinity). [CM08b] shows that this learning criterion is strictly more powerful than plain iterative learning, strictly less powerful than TxtEx-learning, and incomparable to *set-driven* learning [WC80].

In set-driven learning, the learner has access only to the (unordered) set of data seen so far, with duplicates removed. Consider now a learning criterion, where the learner has access to the set of data seen so far, just as in set-driven learning, but also to the current iteration number (just as in iterative with counter learning as introduced in [CM08b]). It is easy to see that this learning criterion is equivalent to *partially set-driven* (or *rearrangement independent*) learning [SR84]; it is well known that partially set-driven learning is equivalent to TxtEx-learning.

The main aim of this paper is to discuss how and why such a counter improves learning power. In particular, we want to understand what properties of a counter can be used in a learning process to increase learning power. Is it the higher and higher counter values, which we can use to time-bound computations? Is it knowing the number of data items seen so far? Is it the complete enumeration of all natural numbers which we can use to divide up tasks into infinitely many subtasks to be executed at the corresponding counter value?

¹ *Txt* stands for learning from a *text* of positive examples; *Ex* stands for *explanatory*.

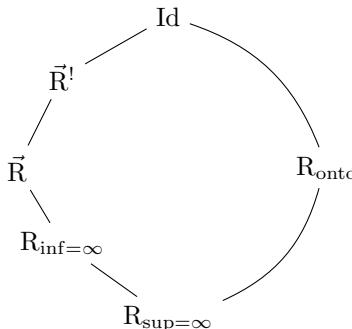
We approach these questions by introducing different *counter types*, each modeling some of the possibly beneficial properties mentioned above. Formally, a counter type is a set of *counters*; a *counter* is a mapping from the natural numbers to itself. Instead of giving the learner the current iteration number, we will map this number with a counter drawn from the counter type under consideration.

We define the following counter types²

- (i) Complete and ordered: $\text{Id} = \{\text{id}_{\mathbb{N}}\}$ ³
- (ii) Strictly monotone: $\vec{R}^! = \{c \mid \forall i : c(i+1) > c(i)\};$
- (iii) Monotone & unbounded: $\vec{R} = \{c \mid \forall i : c(i+1) \geq c(i) \wedge \liminf_{i \rightarrow \infty} c(i) = \infty\};$
- (iv) Eventually above any number: $R_{\inf=\infty} = \{c \mid \liminf_{i \rightarrow \infty} c(i) = \infty\};$
- (v) Unbounded: $R_{\sup=\infty} = \{c \mid \limsup_{i \rightarrow \infty} c(i) = \infty\};$
- (vi) Complete: $R_{\text{onto}} = \{c \mid \text{range}(c) = \mathbb{N}\}.$

By requiring a learner to succeed regardless of what counter was chosen from the counter type, we can provide certain beneficial properties of a counter, while not providing others. For example, counters from R_{onto} provide a complete enumeration of all natural numbers, but do not allow to infer the number of data items seen so far.

We illustrate the inclusion properties of the different sets of counters with the following diagram (inclusions are top to bottom; thus, inclusions of learning power when such counters are used are bottom to top).



The weakest type of counter is $R_{\sup=\infty}$, the unbounded counter. The advantage over having no counter at all is to be able to make computations with higher and higher time bounds; in fact, it is easy to see that set-driven learning merely requires a counter from $R_{\sup=\infty}$ to gain the full power of TxtEx-learning. John Case pointed out that any text for an infinite language implicitly provides a counter from $R_{\sup=\infty}$.

A somewhat stronger counter type is $R_{\inf=\infty}$; the intuitive advantage of this counter is that a learner will not repeat mistakes made on small counter values

² The counter types (i), (iii) and (v) were suggested by John Case in private communication.

³ “Id” stands for identity; \mathbb{N} denotes the natural numbers and $\text{id}_{\mathbb{N}}$ the identity on \mathbb{N} .

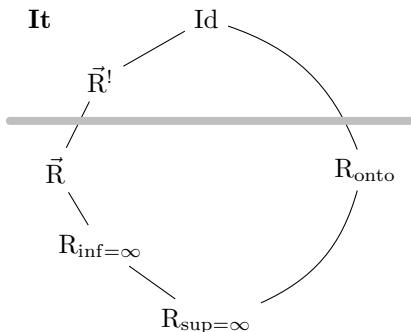
indefinitely, but only the behavior on large counter values affects the learning process in the limit.

For the monotone counters from \vec{R} , the advantage is again that early mistakes are not repeated once learning has proceeded to a later stage (as in, higher counter value), as well as a monotonicity in advancing through these stages.

Counters from $\vec{R}^!$ have the additional benefit of providing an upper bound on the number of examples seen so far.

Id is the strongest type of counter providing exactly the number of data elements presented so far. Also, all natural numbers are listed, which allows a learner to divide up tasks into infinitely many subtasks to be executed at the corresponding counter value; the counter type R_{onto} models this latter advantage while dropping the order restriction.

The main results of this paper consider iterative learning and are as follows. Even adding the weakest type of counter, $R_{sup=\infty}$, to plain iterative learning allows for an increase in learning power; however, there is no increase on learning classes of infinite languages only (Theorem 4). Furthermore, the criteria corresponding to the six counter types are divided into two groups of criteria of equal learning power as depicted by the following diagram (the gray line divides the two groups).



In particular, only the strict monotonicity of a counter gives additional learning power over $R_{sup=\infty}$ counters. The proofs for the claims inherent in the diagram can be found in Section 5.

Theorem 9 in Section 5 shows the separation depicted in the above diagram; its proof uses a self-learning class of languages [CK10, CK11] and Case's *Operator Recursion Theorem* (ORT) [Cas74, JORS99]. Because of space limitations, we only sketch that argument below.

Extending these results to settings where learners have additional resources is ongoing work; preliminary results show that, for adding a finite number of memory states, we get a similar diagram as for iterative learning above.

One may wonder whether some two of the counter types introduced above always yield the same learning power (as many did in the case of iterative learning), across all possible settings. In Section 2 we show and discuss that this is not the case.

After this, the paper is organized as follows. Section 3 gives some mathematical preliminaries. In Section 4 we establish that any separation of learning criteria power will necessarily be witnessed by a class containing only infinite languages, if the considered learning criteria have access to any of the six counter types. As already mentioned, Section 5 gives some details for the diagram above.

2 Differences in Counters

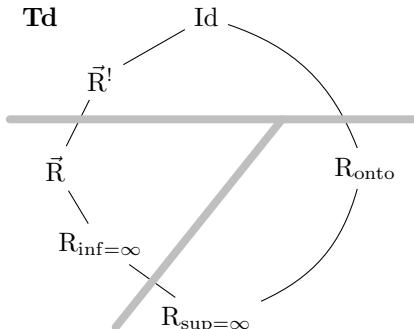
In this section we show that, for any choice of two different counter types, there is a learning criterion which, when augmented with one of the counter types, yields different classes of languages learnable than when augmented with the other.

We already saw some such separations in the setting for iterative learning. Now we will give some other settings witnessing other separations.

First, consider iterative learning with one additional feedback query (see [LZ96, CJLZ99]). In this setting, in each iteration, the learner may ask about one datum whether it has been presented previously. Frank Stephan and Sanjay Jain (private communication) have a proof that, in this setting, there are classes of languages learnable with R_{onto} counters which are not learnable with $\vec{R}^!$ counters. Thus, there are settings where Id separates from $\vec{R}^!$, and where R_{onto} separates from $R_{\text{sup}=\infty}$.

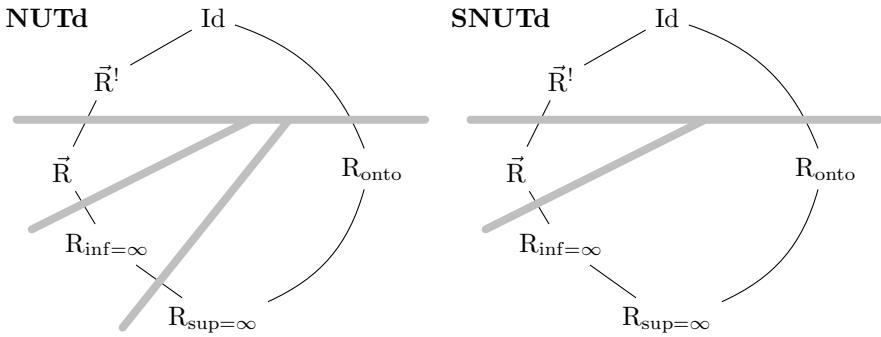
For more separations, we turn to very simple learning criteria. We consider *transductive* learning (**Td**), that is, learning without memory (which equals a degenerate case of memoryless learning with bounded memory states, where the bound on the number of states is 1; [CCJS07, CK08]). In this somewhat artificial toy setting a learner is presented a datum (and possibly a counter value) in each iteration, and not more. Note that, learners are allowed to output the special symbol ? to, in effect, keep the previous conjecture as the latest guess.

It is not hard to see that, for transductive learning, adding an $R_{\text{sup}=\infty}$ or R_{onto} counter does not improve learning power. However, other types of counter do provide increases. The general result is depicted in the following diagram, using the same format as in the diagram on iterative learning above.



The intuitive reasons for the separations are as follows. An infinite limit inferior guarantees that mistakes on early counter values are not repeated infinitely often. With a strictly monotone counter, any mistake on a counter value z is guaranteed to be preceded by at most z other data items; thus, if the language contains at least $z + 1$ data items giving the correct output, the mistake will be rectified.

The situation changes if we require of the learner additionally to never abandon correct conjectures – either only not semantically (called non-U-shaped learning [BCM⁺08]) or not even syntactically (strongly non-U-shaped learning [CM08a]). The resulting groupings and separations are depicted in the following two diagrams.



Intuitively, for learning criteria requiring non-U-shapedness, order plays an important role (wrong conjectures may only come before correct ones), leading to the separations between $R_{inf=\infty}$ and \vec{R} . For strongly non-U-shaped learning with $R_{inf=\infty}$ counter, a learner may not give two different conjectures for any two pairs of datum/counter value.

All the above settings together show that, for each two different types of counter, there are settings of associated learning criteria where the learning power separates.

Because of space restrictions, the only theorem regarding transductive learning we will prove in this paper is given in Theorem 10, giving a flavor of the proofs concerning transductive learning.

3 Mathematical Preliminaries

Unintroduced notation follows [Rog67].

\mathbb{N} denotes the set of natural numbers, $\{0, 1, 2, \dots\}$. The symbols \subseteq , \subset , \supseteq , \supset respectively denote the subset, proper subset, superset and proper superset relation between sets. For any set A , we let $Pow(A)$ denote the set of all subsets of A . \emptyset denotes both the empty set and the empty sequence.

With dom and $range$ we denote, respectively, domain and range of a given function. We sometimes denote a partial function f of $n > 0$ arguments x_1, \dots, x_n in lambda notation (as in Lisp) as $\lambda x_1, \dots, x_n \cdot f(x_1, \dots, x_n)$. For example, with $c \in \mathbb{N}$, $\lambda x \cdot c$ is the constantly c function of one argument.

We fix any computable 1-1 and onto pairing function $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ ⁴. Whenever we consider tuples of natural numbers as input to a function, it is understood that the general coding function $\langle \cdot, \cdot \rangle$ is used to code the tuples into a single natural number. We similarly fix a coding for finite sets and sequences, so that we can use those as input as well.

If a function f is not defined for some argument x , then we denote this fact by $f(x)\uparrow$, and we say that f on x *diverges*; the opposite is denoted by $f(x)\downarrow$, and we say that f on x *converges*. If f on x converges to p , then we denote this fact by $f(x)\downarrow = p$.

The special symbol ? is used as a possible hypothesis (meaning “no change of hypothesis”). We write $f \rightarrow p$ to denote that $f \in \mathfrak{P}$ *converges to* p , i.e., $\exists x_0 : f(x_0) = p \wedge \forall x \geq x_0 : f(x)\downarrow \in \{?, p\}$ ⁵

\mathcal{P} and \mathcal{R} denote, respectively, the set of all partial computable and the set of all computable functions (mapping $\mathbb{N} \rightarrow \mathbb{N}$).

We let φ be any fixed acceptable programming system for \mathcal{P} . Further, we let φ_p denote the partial computable function computed by the φ -program with code number p .

A set $L \subseteq \mathbb{N}$ is *computably enumerable (ce)* iff it is the domain of a computable function. Let \mathcal{E} denote the set of all ce sets. We let W be the mapping such that $\forall e : W(e) = \text{dom}(\varphi_e)$. For each e , we write W_e instead of $W(e)$. W is, then, a mapping from \mathbb{N} onto \mathcal{E} . We say that e is an index, or program, (in W) for W_e .

In this paper, an *operator* is a mapping from any fixed number of arguments from \mathcal{P} into \mathcal{P} .

The symbol # is pronounced *pause* and is used to symbolize “no new input data” in a text. For each (possibly infinite) sequence q with its range contained in $\mathbb{N} \cup \{\#\}$, let $\text{content}(q) = (\text{range}(q) \setminus \{\#\})$.

3.1 Learning Criteria

A *learner* is a partial computable function.

A *language* is a ce set $L \subseteq \mathbb{N}$. Any total function $T : \mathbb{N} \rightarrow \mathbb{N} \cup \{\#\}$ is called a *text*. For any given language L , a *text for L* is a text T such that $\text{content}(T) = L$. This kind of text is what learners usually get as information. We will extend the notion of texts to include counters as follows.

For any type of counters R , we let $\mathbf{TxtCtr}[R]$ be the set of all functions $\langle T, c \rangle = \lambda i. \langle T(i), c(i) \rangle$ with T a text and $c \in R$. We call an element from $\mathbf{TxtCtr}[R]$ a *text/counter*, and the content of any text/counter is the content of its text component.

A *sequence generating operator* is an operator β taking as arguments a function h (the learner) and a text/counter T and that outputs a function p . We call p the *learning sequence* of h given T . Intuitively, β defines how a learner can interact with a given text/counter to produce a sequence of conjectures.

⁴ For a linear-time example, see [RC94, Section 2.3].

⁵ $f(x)$ converges should not be confused with f converges to.

We define the sequence generating operators **It** and **Td** (corresponding to the learning criteria discussed in the introduction) as follows. For all h, T, i ,

$$\begin{aligned}\mathbf{It}(h, T)(i) &= \begin{cases} h(\emptyset), & \text{if } i = 0; \\ h(\mathbf{It}(h, T)(i-1), T(i-1)), & \text{otherwise.} \end{cases}^6 \\ \mathbf{Td}(h, T)(i) &= \begin{cases} h(\emptyset), & \text{if } i = 0; \\ h(T(i-1)), & \text{otherwise.} \end{cases}\end{aligned}$$

Thus, in iterative learning, the learner has access to the previous conjecture, but not so in transductive learning.

Successful learning requires the learner to observe certain restrictions, for example convergence to a correct index. These restrictions are formalized in our next definition.

A *sequence acceptance criterion* is a predicate δ on a learning sequence and a text/counter. We give the examples of explanatory (**Ex**), non-U-shaped (**NU**) and strongly non-U-shaped (**SNU**) learning, which were discussed in Sections 1 and 2. Formally, we let, for all p, T ,

$$\begin{aligned}\mathbf{Ex}(p, T) &\Leftrightarrow [\exists q : p \text{ converges to } q \wedge W_q = \text{content}(T)]; \\ \mathbf{NU}(p, T) &\Leftrightarrow [\forall i : W_{p(i)} = \text{content}(T) \Rightarrow W_{p(i+1)} = W_{p(i)}]; \\ \mathbf{SNU}(p, T) &\Leftrightarrow [\forall i : W_{p(i)} = \text{content}(T) \Rightarrow p(i+1) = p(i)].\end{aligned}$$

We combine any two sequence acceptance criteria δ and δ' by intersecting them.

For any set of text/counters α , any sequence generating operator β and any combination of sequence acceptance restrictions δ , $\alpha\beta\delta$ is a *learning criterion*. A learner h $\alpha\beta\delta$ -*learns* the set

$$\alpha\beta\delta(h) = \{L \in \mathcal{E} \mid \forall T \in \alpha : \text{content}(T) = L \Rightarrow \delta(\beta(h, T), T)\}.$$

Abusing notation, we also use $\alpha\beta\delta$ to denote the set of all $\alpha\beta\delta$ -learnable classes (learnable by some learner).

4 Separations by Classes of Infinite Languages

In this section we show that, for iterative learning, all separations between the learning criteria corresponding to the different counter types are necessarily witnessed by sets of infinite languages. The reasoning for this can be extended to include many other learning criteria.

For an operator Θ , a learning criterion I is called Θ -robust iff, for any class of languages \mathcal{L} , I -learnability of \mathcal{L} is equivalent to I -learnability of $\Theta(\mathcal{L})$ (element wise application of Θ).⁷

We let Θ_0 be the mapping $L \mapsto 2L \cup (2\mathbb{N} + 1)$. Obviously, there is a function f_0 such that $\forall e : \Theta_0(W_e) = W_{f_0(e)}$. Note that Θ_0 has an inverse Θ_0^{-1} for which a function analogous to f_0 exists.

⁶ $h(\emptyset)$ denotes the *initial conjecture* made by h .

⁷ [CK11] explores some notions of robustness for function learning.

Theorem 1. Let $R \in \{R_{\sup=\infty}, R_{\inf=\infty}, \vec{R}, \vec{R}^!, \text{Id}, R_{\text{onto}}\}$. Then we have that the learning criterion $\mathbf{TxtCtr}[R]\mathbf{ItEx}$ is Θ_0 -robust.

Proof. Let $\mathcal{L} \in \mathbf{TxtCtr}[R]\mathbf{ItEx}$. Obviously, $\Theta_0(\mathcal{L})$ can be learned using the learner for \mathcal{L} by ignoring odd data (considering them as $\#$) and halving all even data, mapping all conjectures with f_0 . Conversely, let a learner h_0 for $\Theta_0(\mathcal{L})$ be given. Consider first the case of $R = \text{Id}$. Define the following function h' .

$$\forall e, x, z : h'(e, x, z) = h_0(h_0(e, 2x, 2z), 2z + 1, 2z + 1).$$

Intuitively, on a text T , h' simulates h_0 on the text where $2T$ is interleaved with odd data. We use 1-1 s-m-n to get a function to turn conjectures for a language from $\Theta_0(\mathcal{L})$ into the corresponding language from \mathcal{L} (we use 1-1 so that we can extract and use the conjectures of h_0 from the previous generation as input to h_0), resulting in a learner h . Note that, for $R = R_{\text{onto}}$, the above construction of h works just as well. All other cases are similar as follows.

For $R \in \{R_{\sup=\infty}, R_{\inf=\infty}, \vec{R}\}$, when we see counter value of z , we simulate h_0 on all odd data $\leq z$ and on the current datum times two, using a counter value of z for all of them.

For $R = \vec{R}^!$, when we see counter value of z , we simulate h_0 on all odd data $< z$ and on the current datum times two, using a counter value of $z^2 + i$ for the i th run of h_0 . Thus, within these batches of data, the counter values are strictly increasing. The next batch will start with a counter value of $(z+1)^2 = z^2 + 2z + 1$. This exceeds the last counter used in the previous batch, as the previous batch had a size $\leq z + 1$. \square

Theorem 2. Let I and I' be Θ_0 -robust learning criteria. Then I and I' separate in learning power iff they separate on classes of infinite languages.

Proof. Suppose a class of languages \mathcal{L} separates I and I' . Then $\Theta_0(\mathcal{L})$, a class of infinite languages, also witnesses this separation, as I and I' are Θ_0 -robust. \square

From what we saw in this section we get the following corollary.

Corollary 3. Let $R, R' \in \{R_{\sup=\infty}, R_{\inf=\infty}, \vec{R}, \vec{R}^!, \text{Id}, R_{\text{onto}}\}$. Then the learning criteria $\mathbf{TxtCtr}[R]\mathbf{ItEx}$ and $\mathbf{TxtCtr}[R']\mathbf{ItEx}$ separate iff the separation is witnessed by a class of infinite languages. Furthermore, it is witnessed by a class of languages all containing all odd numbers.

5 Comparison of Counter Types

In this section we present the proofs for the results regarding iterative learning with counter. First we compare the weakest counter with no counter at all (Theorem 4). The Theorems 5 through 8 give equivalences of learning power as indicated in Section 1. Finally, Theorem 9 gives the separation between strictly monotone counters and weaker counters.

Looking into the proof of Theorem 4 in [CM08b] (showing that an Id counter allows for learning languages which cannot be learned set-drivenly), we see that even the counters from $R_{\sup=\infty}$ allow for learning more than learning set-drivenly. This leads to the first part of the next theorem. However, this proof makes use of finite languages. John Case remarked that $R_{\sup=\infty}$ -counters are provided by texts for infinite languages for free, leading to the second part of the theorem. We let \mathcal{E}_∞ denote the set of all *infinite* ce sets.

Theorem 4. We have

$$\mathbf{Txt}\mathbf{It}\mathbf{Ex} \subset \mathbf{Txt}\mathbf{Ctr}[R_{\sup=\infty}]\mathbf{It}\mathbf{Ex}$$

and

$$\text{Pow}(\mathcal{E}_\infty) \cap \mathbf{Txt}\mathbf{It}\mathbf{Ex} = \text{Pow}(\mathcal{E}_\infty) \cap \mathbf{Txt}\mathbf{Ctr}[R_{\sup=\infty}]\mathbf{It}\mathbf{Ex}.$$

For the next step up the hierarchy of counter types, we don't get an increase in learning power.

Theorem 5. We have

$$\mathbf{Txt}\mathbf{Ctr}[R_{\sup=\infty}]\mathbf{It}\mathbf{Ex} = \mathbf{Txt}\mathbf{Ctr}[R_{\inf=\infty}]\mathbf{It}\mathbf{Ex}.$$

Proof. Clearly we get " \subseteq ". The intuitive reason for the inclusion " \supseteq " is as follows. We can use the max of counter value, hypothesis and datum as a new counter to work with. If the conjecture changes infinitely often, then, without loss of generality, the new counter is from $R_{\inf=\infty}$. Hence, the learning will converge; furthermore, for any fixed number z , only finitely many data points are evaluated with a counter value below z .

Let $\mathcal{L} \in \mathbf{Txt}\mathbf{Ctr}[R_{\inf=\infty}]\mathbf{It}\mathbf{Ex}$ as witnessed by h_0 . By Corollary 3, we can assume, without loss of generality, that \mathcal{L} contains only infinite languages. Obviously, using standard padding arguments, we can assume the sequence of h_0 's conjectures, on any text, to be non-decreasing in numeric value. Furthermore, we can assume that, whenever h_0 would make a mind change when the present datum was replaced with a $\#$, then it would also change its mind on the actual datum.

Let h be such that

$$\begin{aligned} h(\emptyset) &= h_0(\emptyset); \\ \forall e, x, z : h(e, x, z) &= h_0(e, x, \max(e, x, z)). \end{aligned}$$

That is, h has the same initial conjecture as h_0 and uses the maximum of current conjecture, current datum (we let $\#$ count as 0) and current counter value as new counter value.

Let $L \in \mathcal{L}$, T a text for L and $c \in R_{\sup=\infty}$ a counter. Suppose, by way of contradiction, h on T and c does not converge. Then h simulates h_0 on T and a counter from $R_{\sup=\infty}$; this converges, a contradiction.

Suppose, by way of contradiction, h on T and c does not converge to an index for L . We focus on the text after h 's convergence to some (wrong) conjecture e .

Consider first the case where there is a finite s such that, for all $s' \geq s$, $h_0(e, \#, s') \neq e$, that is, h_0 changes its mind on $\#$ for all but finitely many counter values. Then, clearly, at some point after the convergence of h on T , we get a counter value $\geq s$ so that h will change its mind, a contradiction.

Consider now the case where, for infinitely many s , $h_0(e, \#, s) = e$. Let T' be the text derived from T where we do not change anything before the point of h 's convergence on T , and afterwards replace all repeated data with $\#$ es. Let c' be such that, for all i , $c'(i) = \max_t[h_0(e, T'(i), t) = e]$ (possibly ∞) – that is, c' denotes the maximum counter value for h_0 to not change its mind. As e is incorrect and needs to be changed by h_0 eventually on any counter with infinite limit inferior, c' has *finite* limit inferior. Thus, there is a bound s such that, for infinitely many i , $\max_t[h_0(e, T'(i), t) = e] \leq s$. Because of the case we consider now, we know that there are infinitely many i with $T'(i) \neq \#$ and $\max_t[h_0(e, T'(i), t) = e] \leq s$. One of these pairwise different $T'(i) = T(i)$ will be larger than s , leading to a mind change with h , a contradiction. \square

Note that the just above proof is not entirely a simulation argument – h_0 is being simulated, but not on counters for which we have immediate performance guarantees.

Also the next step in the counter hierarchy does not yield a difference in learning power.

Theorem 6. We have

$$\mathbf{Txt}\mathbf{Ctr}[R_{inf=\infty}]\mathbf{ItEx} = \mathbf{Txt}\mathbf{Ctr}[\vec{R}]\mathbf{ItEx}.$$

Proof. Clearly we get “ \subseteq ”. Let $\mathcal{L} \in \mathbf{Txt}\mathbf{Ctr}[\vec{R}]\mathbf{ItEx}$ as witnessed by h_0 . Obviously, using standard padding arguments, we can assume the sequence of h_0 's conjectures, on any text, to be non-decreasing in numeric value. Furthermore, we can assume each conjecture to exceed the counter value on which it was first output.

For all e, z , we let $f(e, x, z)$ be the least t with $e \leq t \leq \max(e, z)$ and $h_0(e, x, t) \neq e$, if existent (undefined otherwise). Note that the domain of f is decidable.

Let h be such that, for all e, x, z ,

$$h(\emptyset) = h_0(\emptyset);$$

$$h(e, x, z) = \begin{cases} h_0(e, x, f(e, x, z)), & \text{if } f(e, x, z) \downarrow, \\ e, & \text{otherwise.} \end{cases}$$

Let $L \in \mathcal{L}$, T a text for L and $c \in R_{inf=\infty}$ a counter. We define a counter c' on argument i thus. Let e be the conjecture of h after $T[i]$; if, in the definition of $h(e, T(i), c(i))$, the first case holds with some t , then $c'(i) = t$. Otherwise, if there will be a mind change of h on T with counter c later, then $c'(i) = e$, else $c'(i) = \max(e, \min\{c(j) \mid j \geq i\})$.

It is easy to see that $c' \in \vec{R}$ and h on T and c simulates h_0 on T and c' . \square

Note that, in the proof just above, the argument is again not entirely a simulation – defining the counter c' requires knowledge of future mind changes and of infinitely many future counter values.

Next we show that complete counters do not give an advantage over $R_{\sup=\infty}$ counters.

Theorem 7. We have

$$\mathbf{Txt}\mathbf{Ctr}[R_{\sup=\infty}]\mathbf{ItEx} = \mathbf{Txt}\mathbf{Ctr}[R_{\text{onto}}]\mathbf{ItEx}.$$

Proof. The inclusion “ \subseteq ” is trivial. Suppose, by way of contradiction, a set \mathcal{L} separates the two criteria considered by this theorem. Then, using Corollary 3, we get that a class of languages all containing all odd data witness the separation as well. From a text for such a language we can extract a complete counter (by dividing each datum by 2, rounding down), a contradiction. \square

Last we show that also the top-most in the counter hierarchy gives no difference in learning power.

Theorem 8. We have

$$\mathbf{Txt}\mathbf{Ctr}[\vec{R}^!]\mathbf{ItEx} = \mathbf{Txt}\mathbf{Ctr}[\text{Id}]\mathbf{ItEx}.$$

Proof. Clearly we get “ \subseteq ”. The intuitive idea for “ \supseteq ” is as follows. The learner can store in the conjecture the last counter on which it changed the conjecture and fill up all the gaps in between two counter values with #es.

Let $\mathcal{L} \in \mathbf{Txt}\mathbf{Ctr}[\text{Id}]\mathbf{ItEx}$ as witnessed by h_0 . Without loss of generality, we assume that h_0 will change its mind on any datum whenever it would change its mind on a # (this is not as trivial as for other counter types, but straightforward to show). Using 1-1 s-m-n, we fix any 1-1 function pad such that, for all e, x , $W_{\text{pad}(e,x)} = W_e$. We use this function for a learner to memorize certain information (at the cost of a mind change).

We define a function h_0^* inductively as follows. For all e, z ,

$$\begin{aligned} h_0^*(e, \emptyset, z) &= e; \\ \forall \sigma, x : h_0^*(e, \sigma x, z) &= h_0(h_0^*(e, \sigma, z), x, z + \text{len}(\sigma)). \end{aligned}$$

Let h be such that, for all e, x, z, z' ,

$$\begin{aligned} h(\emptyset) &= \text{pad}(h_0(\emptyset), 0); \\ h(\text{pad}(e, z'), x, z) &= \begin{cases} \text{pad}(h_0^*(e, \#^{z-z'} x, z'), z + 1), & \text{if } h_0^*(e, \#^{z-z'} x, z') \neq e; \\ e, & \text{otherwise.} \end{cases} \end{aligned}$$

Let $L \in \mathcal{L}$, T a text for L and $c \in \vec{R}^!$ a counter. We define a text T' thus.

$$\forall i : T'(i) = \begin{cases} T(k), & \text{if } i = c(k); \\ \#, & \text{otherwise.} \end{cases}$$

Clearly, T' is a text for L and $T' \circ c = T$. Let p be the sequence of outputs of h on T and p' the sequence of outputs of h_0 on T' . Now we have $p' \circ c = p$, as h makes mind changes on data whenever it would make a mind change on a pause with the same counter value. \square

The next theorem shows that the remaining two classes of learning power do separate.

Theorem 9. We have

$$\mathbf{Txt}\mathbf{Ctr}[\vec{R}]\mathbf{ItEx} \subset \mathbf{Txt}\mathbf{Ctr}[\vec{R}^!]\mathbf{ItEx}.$$

Informal sketch of the argument. The inclusion is trivial. The intuitive idea of the separation is as follows. We use a self-learning class of languages (see the definition of \mathcal{L} below for an example of a self-learning class; these classes are discussed in more detail in [CK10, CK11]). We will then suppose that this class can be learned with \vec{R} counters by some function h . We will suggest to h to change its mind on certain data (we call them $a(0), a(1), \dots$); if h never changes its mind, then this will suffice to make h fail. Otherwise, we are still free to suggest to h not to change its mind on this data for high counter values (above some threshold we call c_0). We now add some other data (we call them $b(0), b(1), \dots$) on which h should not change, unless it has changed on some a -values previously. In fact, we add exactly c_0 such data points. With a counter from \vec{R} , these data points may all come very early, on a low and always the same counter value; h will not be able to change its mind then (otherwise we just start over and find infinitely many mind changes on data where no mind change was suggested). Furthermore, h will change its mind later on some a , leading to no success in identification.

With a strictly increasing counter, however, h would have no problem: if all b -values come before the a -values, then the counter would be pushed high enough such that on a -values there need not be a mind change. \square

Finally, we give two theorems regarding transductive learning.

Theorem 10. We have

$$\mathbf{Txt}\mathbf{Ctr}[R_{\text{onto}}]\mathbf{TdEx} = \mathbf{Txt}\mathbf{TdEx} = \mathbf{Txt}\mathbf{Ctr}[R_{\inf=\infty}]\mathbf{SNUTdEx}.$$

Proof. We start with the first equality, where the inclusion “ \supseteq ” is trivial. Let \mathcal{L} be $\mathbf{Txt}\mathbf{Ctr}[R_{\text{onto}}]\mathbf{TdEx}$ -learnable, as witnessed by h . Without loss of generality, we can assume h to output $?$ on $\#$ with any counter value. Otherwise, the number output on pause may be infinitely output on learning any language, in which case \mathcal{L} can have at most one element, which we can learn using only the initial conjecture and outputting $?$ on $\#$ with any counter value.

We claim that, for all $L \in \mathcal{L}$, there is a p such that

- $L = W_{h(\emptyset)}$ or $\exists x \in L \forall z : h(x, z) = p$;
- $\forall x \in L, z \in \mathbb{N} : h(x, z) \in \{?, p\}$.

Suppose, by way of contradiction, that we can get two different outputs p and p' on two datum/counter pairs. Then we can force infinite oscillation between p and p' , a contradiction. Thus, there is at most one p ever output on a datum. Suppose that, for each $x \in L$, there is a z such that $h(x, z) = ?$. Then we can list all $x \in L$ such that h always outputs $?$ and fill up missing counter values with $\#$. As L is learned by h , $L = W_{h(\emptyset)}$.

Clearly, any h as above might as well ignore the counter and learn without such additional help.

Regarding $\mathbf{Txt}\mathbf{Ctr}[R_{inf=\infty}]\mathbf{SNUTdEx}$, it is easy to see that it includes all $\mathbf{Txt}\mathbf{TdEx}$ -learnable classes (using the characterization of learnability as given by the above list). Furthermore, note that any two syntactically different outputs of a learner lead to a syntactic U-shape on some text/counter, with the counter from $R_{inf=\infty}$. Thus, the above characterization for languages from $\mathbf{Txt}\mathbf{Ctr}[R_{onto}]\mathbf{TdEx}$ also characterizes the languages from that are learnable in the sense of $\mathbf{Txt}\mathbf{Ctr}[R_{inf=\infty}]\mathbf{SNUTdEx}$. \square

Theorem 11. We have

$$\mathbf{Txt}\mathbf{Ctr}[R_{inf=\infty}]\mathbf{TdEx} = \mathbf{Txt}\mathbf{Ctr}[\vec{R}]\mathbf{TdEx}.$$

Proof. The inclusion “ \subseteq ” is trivial. Let \mathcal{L} be $\mathbf{Txt}\mathbf{Ctr}[\vec{R}]\mathbf{TdEx}$ -learnable, as witnessed by h .

We show that h witnesses $\mathcal{L} \in \mathbf{Txt}\mathbf{Ctr}[R_{inf=\infty}]\mathbf{TdEx}$. Let $L \in \mathcal{L}$, T a text for L and $c \in R_{inf=\infty}$. Permute $\langle T, c \rangle$ into a text/counter $\langle T', c' \rangle$ such that c' is non-decreasing. Note that h on $\langle T', c' \rangle$ converges to an index for L .

We distinguish two cases. Either h on $\langle T', c' \rangle$ makes infinitely many non-? outputs. Then h on $\langle T, c \rangle$ makes the same infinitely many non-? outputs, and all of those are equal and correct.

Otherwise h on $\langle T', c' \rangle$ makes only finitely many non-? outputs. Then all those finitely many outputs are correct, as we could permute all later elements before any given output (and decrease the counter value as required to retain monotonicity of the counter). Thus, h on $\langle T, c \rangle$ converges to an index for L . \square

References

- [BCM⁺08] Baliga, G., Case, J., Merkle, W., Stephan, F., Wiehagen, W.: When unlearning helps. *Information and Computation* 206, 694–709 (2008)
- [Cas74] Case, J.: Periodicity in generations of automata. *Mathematical Systems Theory* 8, 15–32 (1974)
- [CCJS07] Carlucci, L., Case, J., Jain, S., Stephan, F.: Results on memory-limited U-shaped learning. *Information and Computation* 205, 1551–1573 (2007)
- [CJLZ99] Case, J., Jain, S., Lange, S., Zeugmann, T.: Incremental concept learning for bounded data mining. *Information and Computation* 152, 74–110 (1999)
- [CK08] Case, J., Kötzting, T.: Dynamic modeling in inductive inference. In: Freund, Y., Györfi, L., Turán, G., Zeugmann, T. (eds.) ALT 2008. LNCS (LNAI), vol. 5254, pp. 404–418. Springer, Heidelberg (2008)

- [CK10] Case, J., Kötzing, T.: Strongly non-U-shaped learning results by general techniques. In: Proceedings of COLT (Conference on Learning Theory), pp. 181–193 (2010)
- [CK11] Case, J., Kötzing, T.: Measuring learning complexity with criteria epitomizers. In: Proceedings of STACS (Symposium on Theoretical Aspects of Computer Science), pp. 320–331 (2011)
- [CM08a] Case, J., Moelius, S.: Optimal language learning. In: Freund, Y., Györfi, L., Turán, G., Zeugmann, T. (eds.) ALT 2008. LNCS (LNAI), vol. 5254, pp. 419–433. Springer, Heidelberg (2008)
- [CM08b] Case, J., Moelius, S.: U-shaped, iterative, and iterative-with-counter learning. Machine Learning 72, 63–88 (2008)
- [FKS95] Freivalds, R., Kinber, E., Smith, C.: On the impact of forgetting on learning machines. Journal of the ACM 42, 1146–1168 (1995)
- [Gol67] Gold, E.: Language identification in the limit. Information and Control 10, 447–474 (1967)
- [JORS99] Jain, S., Osherson, D., Royer, J., Sharma, A.: Systems that Learn: An Introduction to Learning Theory, 2nd edn. MIT Press, Cambridge (1999)
- [LZ96] Lange, S., Zeugmann, T.: Incremental learning from positive data. Journal of Computer and System Sciences 53, 88–103 (1996)
- [RC94] Royer, J., Case, J.: Subrecursive Programming Systems: Complexity and Succinctness. Research Monograph in Progress in Theoretical Computer Science. Birkhäuser, Boston (1994)
- [Rog67] Rogers, H.: Theory of Recursive Functions and Effective Computability. McGraw Hill, New York (1987); Reprinted by MIT Press, Cambridge (1987)
- [SR84] Schäfer-Richter, G.: Über Eingabeabhängigkeit und Komplexität von Inferenzstrategien. PhD thesis, RWTH Aachen (1984)
- [WC80] Wexler, K., Culicover, P.: Formal Principles of Language Acquisition. MIT Press, Cambridge (1980)
- [Wie76] Wiehagen, R.: Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. Elektronische Informationverarbeitung und Kybernetik 12, 93–99 (1976)

Robust Learning of Automatic Classes of Languages

Sanjay Jain^{1,*}, Eric Martin², and Frank Stephan^{3,**}

¹ School of Computing, National University of Singapore,
Singapore 117417, Republic of Singapore
sanjay@comp.nus.edu.sg

² School of Computer Science and Engineering,
University of New South Wales, Sydney 2052, Australia
emartin@cse.unsw.edu.au

³ Department of Mathematics and Department of Computer Science,
National University of Singapore, Singapore 119076
fstephan@comp.nus.edu.sg

Abstract. This paper adapts and investigates the paradigm of robust learning, originally defined in the inductive inference literature for classes of recursive functions, to learning languages from positive data. Robustness is a very desirable property, as it captures a form of invariance of learnability under admissible transformations on the object of study. The classes of languages of interest are automatic — a formal concept that captures the notion of being recognisable by a finite automaton. A class of first-order definable operators — called translators — is introduced as natural transformations that preserve automaticity of languages in a given class and the inclusion relations between languages in the class. For many learning criteria, we characterise the classes of languages all of whose translations are learnable under that criterion. The learning criteria have been chosen from the literature on both explanatory learning from positive data and query learning, and include consistent and conservative learning, strong-monotonic learning, strong-monotonic consistent learning, finite learning, learning from subset queries, learning from superset queries, and learning from membership queries.

Keywords: inductive inference, learning in the limit, query learning, robust learning, translations.

1 Introduction

The present work introduces and studies a notion of robust learning in the context of Gold-style language learning (that is, learning in the limit from positive data only). Essentially, robust learning means that learnability of a class is invariant under any admissible transformation: that is, not only the class itself, but also each of its images under an admissible transformation, is learnable.

* Supported in part by NUS grant numbers C252-000-087-001 and R252-000-420-112.

** Supported in part by NUS grant numbers R146-000-114-112 and R252-000-420-112.

The search for invariants is quite prominent in many fields of mathematics. For example, Hermann Weyl described Felix Klein’s famous Erlangen programme on the algebraic foundation of geometry in these words [21]: “If you are to find deep properties of some object, consider all natural transformations that preserve your object.” Bārzdiņš had a similar interest in relation to learning a class of recursive functions, and he conjectured the following, see [3, 23]. Let a class of recursive functions be given. Then every image of the class under a general recursive operator is learnable iff the class is a subclass of a recursively enumerable (uniformly recursive) class of functions. Recursively enumerable classes of functions can be easily identified by a technique called “learning by enumeration” [4]. Using this technique, one just conjectures the first function in the list which is consistent with all data seen so far. Learnability of classes of functions by such algorithms cannot be destroyed by general recursive operators. Bārzdiņš’ conjecture, see [23], essentially says that the enumeration technique also captures all cases where robust learnability holds. Fulk [3] disproved the conjecture and this led to a rich field of exploration within the field of function learning [9, 10, 19]. Further refinements, such as uniform robust learnability [10] (where the learner for a transformed class has to be computable in a description of the transformation) and hyperrobust learnability [19] (learnability, by the same learner, of all the transformations of a certain kind — more precisely, transformations given by primitive recursive operators) have also been investigated.

It is natural to try and generalise robust learning to the case of language learning, which was the first object of study in inductive inference and has been more broadly investigated than function learning. However, the natural extension of the definition as in function learning does not work well for language learning, as even the class of singletons would not be robustly learnable based on that definition. This paper proposes a modified approach to robust language learning, focusing on specific classes of languages — to be introduced in the next paragraph — that enjoy good properties. This approach also provides appealing characterisations. Nevertheless, all concepts defined in this paper are meaningful even for arbitrary families of r.e. languages.

Before we introduce the specific classes of languages that are the object of study of this paper, recall that sets of finite strings over some finite alphabet are regular if they are recognisable by a finite state automaton. Sets of pairs of finite strings over respective alphabets are regular if they are recognisable by a finite state multi-input automaton that uses two different inputs to read both coordinates of the pair, with a special symbol (say \star) being used to pad a shorter coordinate. For instance, to accept the pair $(010, 45)$ an automaton should read 0 from the first input and 4 on the second input and change its state from the start state to some state q_1 , then read 1 from the first input and 5 from the second input and change its state from q_1 to some state q_2 , finally read 0 from the first input and \star from the second input and change its state from q_2 to an accepting state. It is essential that all inputs involved are read synchronically — one character per input and cycle. The classes of languages that we focus on in this paper are classes of regular languages of the form $(L_i)_{i \in I}$ such that I and

$\{(i, x) : x \in L_i\}$ are regular sets; we refer to such a class as an automatic family of languages. An automatic family of languages is actually a particular kind of automatic structure [5, 13, 14].

Learnability of automatic families has recently been studied [6, 7]. Moreover, these classes are a special case of indexed families, which have been extensively studied in learning theory [1, 16, 17]. One major advantage of automatic families over indexed families is that their first-order theory is decidable [5, 6, 7, 13] and that many important properties of them can be first-order defined. In particular, the inclusion structure of the classes can be first-order defined and plays an important role in this paper. This is a fundamental difference to the case of indexed families.

We consider any transformation given by an operator Φ which maps subsets of the source domain D to subsets of an image domain D' such that the automatic family $(L_i)_{i \in I}$ to be learnt is mapped to a family $(L'_i)_{i \in I} = (\Phi(L_i))_{i \in I}$, Φ is definable by a first-order formula, Φ preserves inclusions of all sets and Φ preserves noninclusions between members of the class. We call such a Φ a translator. An important special case is given by text-preserving translators for which $\Phi(L)$ is the union of all $\Phi(E)$ where E ranges over the finite subsets of L . A key result of the theory of automatic structures is that the image $(\Phi(L_i))_{i \in I}$ of an automatic family under such an operator is again an automatic family [13]. We study the impact of such translations on learnability.

We proceed as follows. In Sections 2 and 3, we introduce the necessary notation and notions. In Section 4, we illustrate the notions with a few examples and provide a general characterisation of robust learnability in the limit of automatic families of languages. In Section 5, we provide many further characterisations of robust learnability for some of the learning criteria that have been studied in the literature: consistent and conservative learning, strong-monotonic learning, strong-monotonic consistent learning, finite learning. In Section 6, we consider learning from subset queries, learning from superset queries, and learning from membership queries.

2 Automatic Structures, Languages and Translations

The languages considered in inductive inference [8] consist of numbers implicitly coding some underlying structure, but the coding is not made explicit. In the context of this work though, where languages have to be recognised by finite automata, a minimum of structure has to be given to the members of a language: they are assumed to be strings over an *alphabet* denoted by Σ . It is assumed that Σ is finite but large enough so that all regular sets considered are included in the set Σ^* of strings over Σ . For $x \in \Sigma^*$, the length of x , denoted $|x|$, is the number of symbols occurring in x ; for example, $|00121| = 5$. We write xy for the concatenation of two strings x and y . We denote the empty string by ε .

We denote by D and I two regular subsets of Σ^* . We assume that the members of Σ are strictly ordered and given $x, y \in \Sigma^*$, we write $x <_u y$ iff x is length-lexicographically smaller than y , that is, if either $|x| < |y|$ or $|x| = |y|$ and x comes lexicographically before y . We write $x \leq_u y$ iff $x = y$ or $x <_u y$.

In order to capture the constraint that a class of languages is uniformly recognisable by a finite automaton, we make use of a particular kind of automatic structures [14], that for simplicity, is still referred to as automatic structures. The structures under consideration offer enough expressive power to refer to the target language that a learner will be given a presentation of and has to eventually correctly identify, and to refer to the whole class of languages that are the object of learning. A unary predicate symbol and a binary predicate symbol are used to refer to the target language and the class of languages, respectively.

Definition 1. We call *automatic structure* any \mathcal{V} -structure \mathfrak{M} whose domain is Σ^* , with \mathcal{V} some relational vocabulary with the following properties.

- \mathcal{V} contains the unary predicate symbol X and the binary predicate symbol Y (and possibly more predicate symbols of any arity).
- The interpretation of X in \mathfrak{M} is included in D and the interpretation of Y in \mathfrak{M} is included in $I \times D$.
- The interpretation of all predicate symbols in \mathcal{V} in \mathfrak{M} is regular.

By *language* we mean a subset of Σ^* ; by *source language* we mean a subset of D .

Definition 2. By an *automatic class* we mean a repetition-free regular I -family $(L_i)_{i \in I}$ of source languages (so $\{(i, x) : i \in I, x \in L_i\}$ is recognisable by a multi-input automaton). Members of I are referred to as *indices*.

Here are some examples of automatic classes: (i) The class of sets with up to k elements for a constant k ; (ii) The class of all finite and cofinite subsets of $\{0\}^*$; (iii) The class of all intervals of an automatic linear order on a regular set. On the other hand, the class of all finite sets over $\{a, b\}$ is not automatic. The constraint that automatic classes be repetition-free is not standard when one considers learning of indexed families. However, it is without loss of generality in the context of this work and allows one to substantially simplify the arguments in most proofs. We consider transformations of languages that are definable in the language used to describe the target language and the class of languages to be learnt. In the next definition, x denotes any member of the target language.

Definition 3. Let Φ be any first-order formula over the vocabulary of some automatic structure with the distinguished variable x as unique free variable (this allows one to denote such a formula by Φ rather than by $\Phi(x)$). Let an automatic class $\mathbf{I} = (L_i)_{i \in I}$ be given. For all source languages L , denote by $\Phi_{\mathbf{I}}\langle L \rangle$ the language consisting of all $s \in \Sigma^*$ such that $\Phi[s/x]$ is true in all automatic structures in which the interpretation of $X(w)$ is $w \in L$ and the interpretation of $Y(i, w)$ is $i \in I \wedge w \in L_i$. We say that Φ is an *automatic \mathbf{I} -translator* if:

- for all source languages L and L' , if $L \subseteq L'$ then $\Phi_{\mathbf{I}}\langle L \rangle \subseteq \Phi_{\mathbf{I}}\langle L' \rangle$;
- for all members i and j of I , if $L_i \not\subseteq L_j$ then $\Phi_{\mathbf{I}}\langle L_i \rangle \not\subseteq \Phi_{\mathbf{I}}\langle L_j \rangle$.

Given two terms t and t' , we write $t \in X$ for $X(t)$ and $t' \in Y_t$ for $Y(t, t')$ (equivalently, $Y_t = \{t' : Y(t, t')\}$).

Given an automatic **I**-translator Φ , we let $\Phi(\mathbf{I})$ denote $(\Phi_{\mathbf{I}}(L_i))_{i \in I}$; we refer to any such family as a *translation of **I***. Note that a translation is always defined with respect to **I**-translators for a particular automatic class **I**. We drop the reference to **I** for ease of notation. The advantage of the definability of translators via first-order formulas is automaticity preservation. Indeed, by results in [13]:

Theorem 4. *For all automatic classes **I**, all translations of **I** are automatic.*

3 Texts and Learnability

Let us recall the basic concepts in inductive inference as originally defined in [4] and fix some notation. The only difference with the classical framework of learning from positive data is that we consider languages over strings rather than natural numbers.

Let $\# \notin \Sigma$ be a special symbol. We denote by SEQ the set of finite sequences of members of $\Sigma^* \cup \{\#\}$. Given $\sigma \in \text{SEQ}$, we denote by $\text{rng}(\sigma)$ the set of members of Σ^* that occur in σ (e.g., if $\Sigma = \{a, b, c\}$ and $\sigma = (ab, \#, ab, cacba)$ then $\text{rng}(\sigma) = \{ab, cacba\}$). Given $\sigma \in \text{SEQ}$ and a family $\mathbf{I} = (L_i)_{i \in I}$ of languages, we say that σ is *for **I*** iff $\text{rng}(\sigma) \subseteq L_i$ for some $i \in I$. Given a language L , a *text* for L refers to an enumeration $(e_k)_{k \in \mathbb{N}}$ of $L \cup \{\#\}$ where $\#$ might not occur. The concatenation of $\sigma \in \text{SEQ}$ with $s \in \Sigma^* \cup \{\#\}$ is written $\sigma \diamond s$. We also write $\sigma \diamond \tau$ for the concatenation of $\sigma, \tau \in \text{SEQ}$. An initial segment of a member σ of SEQ is a member τ of SEQ such that σ is of the form $\tau \diamond \tau'$ for some $\tau' \in \text{SEQ}$; in this case σ is also said to extend τ .

The notion of translator is quite general and it is worthwhile to examine to which extent it can be constrained to continuous transformations, that is, translators such that any member of the translation can be determined from a finite subset of the original language:

Definition 5. Let an automatic class $\mathbf{I} = (L_i)_{i \in I}$ and an automatic **I**-translator Φ be given. We say that Φ is *text-preserving* iff for all source languages L and for all $s \in \Phi_{\mathbf{I}}(L)$, there is a finite subset F of L with $s \in \Phi_{\mathbf{I}}(F)$.

We talk about *text-preserving translation of **I*** to refer to any family of the form $\Phi(\mathbf{I})$ where Φ is a text-preserving automatic **I**-translator.

Example 6. Given an automatic class $\mathbf{I} = (L_i)_{i \in I}$, let a formula Φ^{nc} with x as unique free variable express $x \in I \wedge \exists y(y \in X \setminus Y_x)$, that is, $x \in I \wedge X \not\subseteq L_x$. Then for all languages L , $\Phi_{\mathbf{I}}^{nc}(L) = \{i \in I : L \not\subseteq L_i\}$. Moreover, Φ^{nc} is text-preserving. Note that I is expressible, as there is at most one index for \emptyset ; hence I is the set of all x 's with $\exists y(Y(x, y))$, or $\exists y(Y(x, y)) \vee x = i_0$, where i_0 denotes the index of \emptyset in case $\emptyset \in I$.

Almost all results will involve recursive learners, with one exception (Theorem 29) where we had to drop the recursiveness requirement. This result will be expressed in terms of general learners. Both kinds of learners are defined next.

Definition 7. A *general learner* is a partial function from SEQ into I . A *learner* is any partial recursive function from SEQ into I with a recursive domain.

In the context of automatic structures, the fact that a learner is undefined on some input signals that the learner cannot make a reasonable guess, rather than the learner being unable to make a guess due to computational infeasibility. This justifies the constraint on the domain of a learner in the definition above. We could also let a learner output some special symbol rather than being undefined.

Definition 8 (Gold [4]). Let $\mathbf{I} = (L_i)_{i \in I}$ be an automatic class. A learner M is said to *learn* \mathbf{I} iff for all $i \in I$ and for all texts $(e_k)_{k \in \mathbb{N}}$ for L_i , $M((e_0 \dots, e_k))$ is defined and equal to i for cofinitely many $k \in \mathbb{N}$. We say that \mathbf{I} is *learnable* iff some learner learns \mathbf{I} .

Note that for simplicity, we use the term “learning” to refer to the notion that in the literature is more precisely called *explanatory learning*. Furthermore, observe that the definition above takes advantage of the one-one indexing of the automatic families considered. We now recall some of the restrictions on learnability that have been investigated in the literature [12, 22, 11] and that will be considered in this paper, individually or combined.

Definition 9. Let an automatic class $\mathbf{I} = (L_i)_{i \in I}$ and a learner M that learns \mathbf{I} be given.

M is *consistent* iff for all $\sigma \in \text{SEQ}$, if σ is for \mathbf{I} then $M(\sigma)$ is defined and $\text{rng}(\sigma) \subseteq L_{M(\sigma)}$.

M is *conservative* iff for all $\sigma, \tau \in \text{SEQ}$, if $\sigma \diamond \tau$ is for \mathbf{I} , both $M(\sigma)$ and $M(\sigma \diamond \tau)$ are defined and $L_{M(\sigma \diamond \tau)} \neq L_{M(\sigma)}$, then $\text{rng}(\sigma \diamond \tau) \setminus L_{M(\sigma)} \neq \emptyset$.

M is *confident* iff for all texts e of languages, there exists $m \in \mathbb{N}$ such that for all $n \geq m$, $M((e(0) \dots e(n)))$ is undefined or equal to $M((e(0) \dots e(m)))$.

M is *strong-monotonic* iff for all $\sigma, \tau \in \text{SEQ}$, if σ is an initial segment of τ , τ is for \mathbf{I} and both $M(\sigma)$ and $M(\tau)$ are defined, then $L_{M(\sigma)} \subseteq L_{M(\tau)}$.

Definition 10. An automatic class \mathbf{I} is said to be *consistently, conservatively, confidently or strong-monotonically learnable* iff some consistent, conservative, confident or strong-monotonic learner learns \mathbf{I} , respectively.

For robust learning, one requires that each translation $\Phi(\mathbf{I})$ of the family \mathbf{I} is learnable (according to the given criterion), where Φ ranges over all automatic \mathbf{I} -translators. Note that requiring the learnability of each translation implies requiring the learnability of \mathbf{I} itself, as the translator can be the identity-operator. In some cases, we consider learnability only of $\Phi(\mathbf{I})$, for all *text-preserving* automatic \mathbf{I} -translators Φ .

The proof of the learnability of indexed families of languages in terms of tell-tales given by Angluin [1] can easily be adapted to the current setting, with indexed families replaced by automatic classes.

Definition 11 (Angluin [1]). Let $\mathbf{I} = (L_i)_{i \in I}$ be an automatic class. Given $i \in I$, a *tell-tale for L_i* is a finite $F \subseteq L_i$ such that for all $i' \in I$, if $F \subseteq L_{i'} \subseteq L_i$ then $L_i = L_{i'}$.

Theorem 12 (Jain, Luo and Stephan [6]; based on Angluin [1]). Let $\mathbf{I} = (L_i)_{i \in I}$ be an automatic class. Then \mathbf{I} is learnable iff for all $i \in I$, there

exists a tell-tale for L_i . Moreover, if \mathbf{I} is learnable then \mathbf{I} is consistently and conservatively learnable by a set-driven learner (whose conjecture on an input σ only depends on $\text{rng}(\sigma)$).

Alternatively, one could also describe the tell-tale by an upper bound in order to get a first-order definable formula which expresses learnability. An automatic class $\mathbf{I} = (L_i)_{i \in I}$ is learnable iff for all members i of I , there is a bound $b_i \in D$ (which recall, denotes the domain of the source languages) such that no $j \in I$ satisfies that $\{y \in L_i : y \leq_{ll} b_i\} \subseteq L_j \subset L_i$. This is equivalent to

$$(\forall i \in I) (\exists b_i \in D) (\forall j \in I) \left[(\exists y \in D[y \leq_{ll} b_i \wedge y \in L_i \setminus L_j]) \vee (\exists y \in D[y \in L_j \setminus L_i]) \vee (\forall y \in D[y \in L_i \Rightarrow y \in L_j]) \right].$$

In order not to clutter notation, we will from now on abstain from breaking subset-relations down into first-order formulas as exemplified with the previous formula; we leave it to the reader to formalise subset-relations via quantified predicates using membership.

Example 13. Let an automatic class $\mathbf{I} = (L_i)_{i \in I}$ be given. There are two learners, M_{smon} and M_{ex} (which use automatic description of \mathbf{I} as a parameter), which learn \mathbf{I} whenever \mathbf{I} is strong-monotonically and explanatorily learnable, respectively. These two learners are defined as follows.

- In response to $\sigma \in \text{SEQ}$, M_{smon} outputs the unique $i \in I$ that satisfies $\text{rng}(\sigma) \subseteq L_i$ and such that for all $j \in I$, if $\text{rng}(\sigma) \subseteq L_j$ then $L_i \subseteq L_j$; if such an i does not exist then M_{smon} is undefined.
- In response to $\sigma \in \text{SEQ}$, M_{ex} outputs the unique $i \in I$ such that $\text{rng}(\sigma) \subseteq L_i$ and there is no $j \in I$ with (1) $\text{rng}(\sigma) \subseteq L_j$ and (2) either $L_j \subset L_i$ or $j <_{ll} i$ and $L_i \not\subseteq L_j$; if such an index i does not exist then M_{ex} is undefined.

Note that not all explanatorily learnable classes are strong-monotonically learnable, hence the learner M_{smon} is not as powerful as M_{ex} . An example of a class which is explanatorily learnable but not strong-monotonically learnable is $\{\{0, 1\}^* \setminus \{x\} : x \in \{0, 1\}^*\}$. Furthermore, the above learners can of course also run on classes of the form $\Phi\langle \mathbf{I} \rangle$ (using $\Phi\langle \mathbf{I} \rangle$ as a parameter instead of \mathbf{I}) whenever such a class is learnable under the corresponding condition.

4 General Characterisation

We start with two examples of conditions that guarantee robust learnability.

Theorem 14. Let $\mathbf{I} = (L_i)_{i \in I}$ be an automatic class.

- If $(\{L_i : i \in I\}, \supset)$ is well ordered (for the superset relation, not the subset relation), then all translations of \mathbf{I} are learnable.
- If for all $i, j \in I$, $L_i \subseteq L_j \Leftrightarrow L_i = L_j$ then all translations of \mathbf{I} are learnable.

As can be expected, learning does not imply robust learning, even if restricted to text-preserving translations:

Theorem 15. *Some strong-monotonically, consistently and confidently learnable automatic class has a text-preserving translation which is not learnable.*

Theorem 16 offers a general characterisation of robust learning in this framework along the lines of Theorem 12.

Theorem 16. *Given an automatic class $\mathbf{I} = (L_i)_{i \in I}$, the three conditions below are equivalent.*

1. *Every translation of \mathbf{I} is learnable.*
2. *Every text-preserving translation of \mathbf{I} is learnable.*
3. *For all $i \in I$, there exists $b_i \in I$ such that for all $j \in I$, either $L_j \not\subseteq L_i$ or there exists $k \in I$ with $k \leq_{ll} b_i$, $L_i \not\subseteq L_k$ and $L_j \subseteq L_k$.*

5 Characterisations of Learnability Variously Constrained

Consistency is a rather weak constraint on learners, and is often combined with other desirable properties. We first combine consistency with conservativeness. Later we will combine it with strong-monotonicity. Note that here “a class is consistently and conservatively learnable” means that the class is learnable by a learner which is both consistent and conservative (rather than having two different learners, one satisfying consistency and the other satisfying conservativeness). A similar convention applies to combining other constraints on learners. Let us first illustrate the notion with an example.

Example 17. Take I equal to $\{1^n, 2^n : n \in \{1, 2, 3, \dots\}\}$. Let $\mathbf{I} = (L_i)_{i \in I}$ be defined by $L_{1^n} = \{0^m : m > n\}$ and $L_{2^n} = \{0^m : m < n\}$ for all $n > 0$. Note that \mathbf{I} is an automatic class that is neither \subset - nor \supset -well founded. Let Φ be a text-preserving automatic \mathbf{I} -translator. Some consistent and conservative learner M learns $\Phi(\mathbf{I})$, proceeding as follows in response to $\sigma \in \text{SEQ}$.

If σ extends τ , $M(\tau)$ is defined and $\text{rng}(\sigma) \subseteq L_{M(\tau)}$

Then M outputs $M(\tau)$

Else If there is $n \in \mathbb{N}$ with $\text{rng}(\sigma) \subseteq \Phi_{\mathbf{I}}(L_{1^n})$ and $\text{rng}(\sigma) \not\subseteq \Phi_{\mathbf{I}}(L_{1^{n+1}})$

Then M outputs 1^n

Else M conjectures 2^n for the least $n > 0$ with $\text{rng}(\sigma)$ included in $\Phi_{\mathbf{I}}(L_{2^n})$.

Since Φ is text-preserving, for all $n > 0$, every finite subset of $\Phi_{\mathbf{I}}(L_{1^n})$ is contained in $\Phi_{\mathbf{I}}(L_{2^m})$ for some $m \in \mathbb{N}$; hence M is consistent. By the first clause in the definition of M , M is conservative.

We now show that M learns $\Phi(\mathbf{I})$. Let $n > 0$ be given. Presented with a text for $\Phi_{\mathbf{I}}(L_{1^n})$, M eventually observes a datum outside $\Phi_{\mathbf{I}}(L_{1^{n+1}})$, at which point M either conjectures 1^n or outputs the previous hypothesis — of the form 2^m for some $m > 0$ — until $\Phi_{\mathbf{I}}(L_{2^m})$ becomes inconsistent with the data observed, at which point M makes a mind change to 1^n . Presented with a text for $\Phi_{\mathbf{I}}(L_{2^n})$, M eventually conjectures 2^n as soon as the data observed become inconsistent with $\Phi_{\mathbf{I}}(L_1)$ and $\Phi_{\mathbf{I}}(L_{2^m})$ for all nonzero $m < n$, which is guaranteed to happen as there are only finitely many languages of the latter type.

Combined with Theorem 16, the next result characterises robust learnability by consistent, conservative learners.

Theorem 18. *Let \mathbf{I} be a learnable automatic class all of whose translations are learnable. Then every translation of \mathbf{I} is consistently and conservatively learnable iff the set of members of \mathbf{I} is well founded under inclusion.*

For the learning criterion of strong-monotonicity, we first consider the concept by itself and then combined with consistency. Again, we can characterise robust learnability under these restrictions, and provide further insights.

Theorem 19. *Given an automatic class $\mathbf{I} = (L_i)_{i \in I}$, clauses 1–3 are equivalent.*

1. *Every translation of \mathbf{I} is strong-monotonically learnable.*
2. *Every text-preserving translation of \mathbf{I} is strong-monotonically learnable.*
3. *For all $i \in I$, there exists $b_i \in I$ such that for all $j \in I$ with $L_i \not\subseteq L_j$, there exists $k \in I$ with $k \leq_L b_i$, $L_i \not\subseteq L_k$ and $L_j \subseteq L_k$.*

Theorem 20. *Every automatic class has some strong-monotonically learnable translation.*

Theorem 21. *If some text-preserving translation of an automatic class \mathbf{I} is strong-monotonically learnable, then \mathbf{I} itself is strong-monotonically learnable.*

We now consider learners which are both strong-monotonic and consistent. The following example shows that consistency adds a genuine constraint to strong-monotonicity.

Example 22. Take $I = \{0, 1\} \cup \{2\}^*$. Let $\mathbf{I} = (L_i)_{i \in I}$ be defined by $L_0 = \{0\}$, $L_1 = \{1\}$ and $L_{2^n} = \{0, 1\} \cup \{2^m : m \geq n\}$ for all $n \in \mathbb{N}$. Then \mathbf{I} is an automatic class. Let Φ be an automatic \mathbf{I} -translator. Then M_{smon} from Example 13 (using $\Phi(\mathbf{I})$ as a parameter) learns $\Phi(\mathbf{I})$, due to the following behaviour on $\sigma \in \text{SEQ}$: if $\text{rng}(\sigma)$ is contained in exactly one of $\Phi_{\mathbf{I}}(L_0)$ and $\Phi_{\mathbf{I}}(L_1)$, then M_{smon} outputs 0 or 1, respectively. If there is $n \in \mathbb{N}$ with $\text{rng}(\sigma)$ contained in $\Phi_{\mathbf{I}}(L_{2^n})$ but not in $\Phi_{\mathbf{I}}(L_{2^{n+1}})$, then $M_{smon}(\sigma)$ outputs 2^n . In any other case, $M_{smon}(\sigma)$ is undefined. Clearly, M_{smon} is a strong-monotonic learner that learns $\Phi(\mathbf{I})$.

But no consistent, strong-monotonic learner M learns $\Phi(\mathbf{I})$. Indeed, suppose otherwise, and let $\sigma \in \text{SEQ}$ be such that $\text{rng}(\sigma)$ is a subset of the union of $\Phi_{\mathbf{I}}(L_0)$ with $\Phi_{\mathbf{I}}(L_1)$, but not a subset of either of the sets. Then $M(\sigma)$ is equal to 2^n for some $n \in \mathbb{N}$, and $M(\tau)$ remains equal to 2^n for all $\tau \in \text{SEQ}$ that extend σ and that are initial segments of a text for $\Phi_{\mathbf{I}}(L_{2^{n+1}})$ (the learner cannot change its mind as $L_{2^{n+1}} \subset L_{2^n}$); therefore M fails to learn $\Phi(\mathbf{I})$.

Theorem 23. *Given an automatic class $\mathbf{I} = (L_i)_{i \in I}$, the three conditions below are equivalent.*

1. *Every translation of \mathbf{I} is strong-monotonically consistently learnable.*
2. *Every text-preserving translation of \mathbf{I} is strong-monotonically consistently learnable.*
3. *$\{L_i : i \in I\}$ is \subset -well-ordered and of type ω at most.*

Theorem 24. Let an automatic class $\mathbf{I} = (L_i)_{i \in I}$ be given. The following

- (\star) For all finite $F \subseteq I$, if there exists $i \in I$ with $L_k \subseteq L_i$ for all $k \in F$, then there exists $i \in I$ such that $L_k \subseteq L_i$ for all $k \in F$, and for all $j \in I$, $L_i \subseteq L_j \Leftrightarrow \forall k \in F (L_k \subseteq L_j)$

expresses that every finite subset of \mathbf{I} which is \subseteq -bounded in \mathbf{I} has a (necessarily unique) \subseteq -least upper bound in \mathbf{I} . Then statements 1,2 below hold.

1. There exists an automatic \mathbf{I} -translator Φ such that $\Phi(\mathbf{I})$ is consistently and strongly-monotonically learnable iff (\star) holds.
2. Suppose that the class \mathbf{I} is strongly-monotonically learnable. Then some text-preserving automatic translation of \mathbf{I} is strongly-monotonically and consistently learnable iff (\star) holds.

Note that a class that contains an ascending infinite chain is not confidently learnable. The following theorem follows from this observation along with the results about strong-monotonic learning shown above.

Theorem 25. Given an automatic class $\mathbf{I} = (L_i)_{i \in I}$, statements 1–4 below hold.

1. A strong-monotonically learnable class \mathbf{I} is confidently learnable iff it contains no infinite ascending chain.
2. Every translation of \mathbf{I} is strong-monotonically and confidently learnable iff \mathbf{I} does not contain infinite ascending chains and for all $i \in I$, there exists $b_i \in I$ such that for all $j \in I$, if $L_i \not\subseteq L_j$, then there is $k \leq_{ll} b_i$ with $L_j \subseteq L_k$ and $L_i \not\subseteq L_k$.
3. Some translation of \mathbf{I} is strong-monotonically and confidently learnable iff \mathbf{I} contains no ascending infinite chain.
4. If some text-preserving translation of \mathbf{I} is strong-monotonically and confidently learnable, then \mathbf{I} itself is strong-monotonically and confidently learnable.

These results give a full characterisation on how confident learnability combines with strong-monotonic learning. We should also observe the fact that every translation of a class is confidently learnable does not imply that the class is strong-monotonically learnable:

Example 26. Consider the automatic class \mathbf{I} which contains $\{0\}^*$ and for all $n > 0$, $\{0^m : m < n\} \cup \{1^n\}$ and $\{0\}^* \cup \{1^m : m \geq n\}$. This class is not strong-monotonically learnable as any learner that learns \mathbf{I} must output the index for $\{0\}^*$ after seeing a finite sequence σ of suitable examples. But then, for sufficiently large n , a mind change to the index for $\{0^m : m < n\} \cup \{1^n\}$ would be necessary to learn it from any text which extends σ .

Still for every automatic \mathbf{I} -translator Φ , $\Phi(\mathbf{I})$ is confidently learnable by a learner M that proceeds as follows. As long as the data is consistent with $\Phi_{\mathbf{I}}\langle\{0\}^*\rangle$, M conjectures the index for $\Phi_{\mathbf{I}}\langle\{0\}^*\rangle$. If there exists (a necessarily unique) $n > 0$ such that the data seen so far is consistent with the set $\Phi_{\mathbf{I}}\langle\{0^m : m < n\} \cup \{1^n\}\rangle$ but not with $\Phi_{\mathbf{I}}\langle\{0\}^* \cup \{1^m : m > n\}\rangle$, then M

outputs the index for $\Phi_{\mathbf{I}}\langle\{0^m : m < n\} \cup \{1^n\}\rangle$. Otherwise, if presented with some data that is consistent with $\Phi_{\mathbf{I}}\langle 0^* \cup \{1^m : m \geq n\}\rangle$ for all $n \in \mathbb{N}$, but not with $\Phi_{\mathbf{I}}\langle 0^*\rangle$, M outputs its previous hypothesis. Otherwise, M conjectures the index for $\Phi_{\mathbf{I}}\langle\{0\}^* \cup \{1^m : m \geq n\}\rangle$ where $n > 0$ is largest for which the set is consistent with the input data; n might go down as more data are presented, but will eventually stabilise. Hence $\Phi(\mathbf{I})$ is confidently learnable.

A characterisation of classes whose every translation is confidently learnable by a computable learner is open. Theorem 29 deals with the case of general learners.

Theorem 27. *Every translation of an automatic class \mathbf{I} is confidently, conservatively and consistently learnable iff \mathbf{I} is finite.*

A more restrictive notion of learning is finite learning where the very first conjecture output by the learner has to correctly identify the set to be learnt. Obviously, finitely learnable classes are antichains as otherwise one could see the data for a set L_i and conjecture an index for this set only to find out later that the set to be learnt is actually a superset of L_i . So a key question is to characterise the size of these antichains.

Theorem 28. *Let an automatic class \mathbf{I} be given. Statements 1–3 below hold.*

1. *Every text-preserving translation of \mathbf{I} is finitely learnable iff \mathbf{I} is a finite antichain.*
2. *Some translation of \mathbf{I} is finitely learnable iff \mathbf{I} is an antichain.*
3. *If \mathbf{I} has a finitely learnable text-preserving translation, then \mathbf{I} itself is finitely learnable.*

The following result is the only one that involves general learners rather than computable learners. Recall the definition of Φ^{nc} in Example 6.

Theorem 29. *Let $\mathbf{I} = (L_i)_{i \in I}$ be an automatic class all of whose translations are learnable. Then both conditions below are equivalent.*

- *Every translation of I is confidently learnable by some general learner.*
- *There exists no nonempty subset J of I such that, for all $i \in J$ and finite subsets F of $\Phi_{\mathbf{I}}^{nc}\langle L_i\rangle$, there exists $j \in J$ with $F \cup \{i\} \subseteq \Phi_{\mathbf{I}}^{nc}\langle L_j\rangle$.*

6 Learning from Queries

Whereas learnability in the limit offers a model of passive learning, learning from queries allows agents to play an active role by questioning an oracle on some properties that the target language might have, and sometimes getting further clues [2]. Four kinds of queries are usually used, alone or in combination. In a given context, the selection of queries that the learner is allowed to make is dictated by the desire to obtain natural, elegant and insightful characterisations. Some studies have compared the passive and active models of learning, which usually turn out to be different [18]. Interestingly, in our model both families of paradigms bind tightly as learnability of a class of languages from superset queries is equivalent to learnability from positive data of every translation of the class (Corollary 40 below).

Definition 30. Let T be the set of queries of at least one of the following types:

$$\begin{array}{ll} \text{Membership query: is } x \in L? & \text{Subset query: is } L_e \subseteq L? \\ \text{Superset query: is } L_e \supseteq L? & \text{Equivalence query: is } L_e = L? \end{array}$$

Let an automatic class $\mathbf{I} = (L_i)_{i \in I}$ be given.

An \mathbf{I} -query learner of type T is a machine M such that, for all $i \in I$, when learning L_i , M makes finitely many queries from T , possibly taking into account the answers to earlier queries, with all queries answered correctly w.r.t. $L = L_i$, and eventually outputs a member of I .

An \mathbf{I} -query learner of type T learns \mathbf{I} iff for all $i \in I$, i is the member of I that M eventually outputs when learning L_i . A query learner of type T for \mathbf{I} is an \mathbf{I} -query learner of type T that learns \mathbf{I} .

\mathbf{I} is learnable from queries of type T iff a query learner of type T for \mathbf{I} exists. When T is clear from the context, we omit to mention “of type T .”

Remark 31. Let an automatic class $\mathbf{I} = (L_i)_{i \in I}$ and an automatic \mathbf{I} -translator Φ be given. Since Φ preserves inclusion and no counterexamples are involved, “learnability of \mathbf{I} from queries of type T ” and “learnability of $\Phi(\mathbf{I})$ from queries of type T ” are equivalent notions as long as T does not include membership queries. Observe that subset, superset and equivalence queries are only with reference to languages in \mathbf{I} , or $\Phi(\mathbf{I})$, respectively.

Since no complexity bound is imposed, we immediately have the following:

Theorem 32. Every automatic class is learnable from equivalence queries.

We illustrate query learning with a few examples of automatic classes.

Example 33. All translations of the classes below are learnable from membership queries:

- $\{x \in \{0,1\}^*: x \text{ is a prefix of } y \vee y \text{ is a prefix of } x\} : y \in \{0,1\}^*$.
- $\{\{0\}^* \cup \{1^m : m \leq n\} : n > 0\} \cup \{\{0^m : m \geq n\} : n > 0\}$.
- Any finite class.

Example 34. Given an automatic class \mathbf{I} , let Φ^{nc} be the text-preserving automatic \mathbf{I} -translator defined in Example 6. Then $\Phi^{nc}(\mathbf{I})$ is learnable from membership and subset queries by searching for the unique $i \in I$ which satisfies that $i \notin \Phi_I^{nc}(L) \wedge \Phi_I^{nc}(L_i) \subseteq \Phi_I^{nc}(L)$. Indeed, a negative answer to the membership query for i implies $\Phi_I^{nc}(L) \subseteq \Phi_I^{nc}(L_i)$ and so $\Phi_I^{nc}(L_i) = \Phi_I^{nc}(L)$.

Example 35. Let an automatic class \mathbf{I} be given and let Φ be a, not necessarily text-preserving, automatic \mathbf{I} -translator satisfying $\Phi_{\mathbf{I}}(L) = \{i \in I : L_i \subseteq L\}$ for all languages L . Then $\Phi(\mathbf{I})$ is learnable from membership and superset queries: a $\Phi(\mathbf{I})$ -query learner can search for the unique $i \in I \cap \Phi_{\mathbf{I}}(L)$ with $\Phi_{\mathbf{I}}(L) \subseteq \Phi_{\mathbf{I}}(L_i)$. This i satisfies $\Phi_{\mathbf{I}}(L_i) = \Phi_{\mathbf{I}}(L)$ and can be found using both kinds of queries.

Example 36. Consider the automatic class \mathbf{I} consisting of $\{0, 1\}^*$ and all singletons of the form $\{0, 1\}^* \setminus \{x\}$ with $x \in \{0, 1\}^*$. Then none of \mathbf{I} 's text-preserving translations is learnable from superset and membership queries. Let Φ be a text-preserving \mathbf{I} -translator, and assume for a contradiction that a query learner M for $\Phi(\mathbf{I})$ outputs an index for $\Phi_{\mathbf{I}}(\{0, 1\}^*)$ after finitely many superset and membership queries on x_1, x_2, \dots, x_n . If L is any member of $\Phi(\mathbf{I})$, then the superset query “is $\Phi_{\mathbf{I}}(\{0, 1\}^*) \supseteq L$?” necessarily receives the answer “yes”, and for all $i \in I$ with $L_i \neq \{0, 1\}^*$ and $L_i \neq L$, the superset query “is $\Phi_{\mathbf{I}}(L_i) \supseteq L$?” necessarily receives the answer “no”. Furthermore, the membership queries “is $x_k \in L$?” necessarily receive the answer “yes” when answered with respect to $L = \Phi_{\mathbf{I}}(\{0, 1\}^*)$. Now for each $x_k \in \Phi_{\mathbf{I}}(\{0, 1\}^*)$, there is a finite subset E_k of $\{0, 1\}^*$ with $x_k \in \Phi_{\mathbf{I}}(E_k)$. Consider any $y \in \{0, 1\}^*$ such that:

- for all $k \in I$ such that M has queried the membership of x_k to the target language when learning $\Phi_{\mathbf{I}}(\{0, 1\}^*)$, $y \notin E_k$;
- the superset query “is $\Phi_{\mathbf{I}}(L) \subseteq \Phi_{\mathbf{I}}(\{0, 1\}^* \setminus \{y\})$?” has not been asked by M when learning $\Phi_{\mathbf{I}}(\{0, 1\}^*)$.

Then all queries would receive the same answer if the language L to be learnt was $\Phi_{\mathbf{I}}(\{0, 1\}^* \setminus \{y\})$; therefore M cannot distinguish $\Phi_{\mathbf{I}}(\{0, 1\}^* \setminus \{y\})$ from $\Phi_{\mathbf{I}}(\{0, 1\}^*)$. Hence M is incorrect and $\Phi(\mathbf{I})$ is not learnable from superset and membership queries.

Theorem 37. *Every automatic class has a translation learnable using membership queries.*

The theorem and corollary that follow characterise learnability from subset and superset queries. These results have a similar flavour as Theorems 4, 5 and 10 in [15], obtained in the context of indexable classes of r.e. languages and a broader class of queries.

Theorem 38. *Let an automatic class $\mathbf{I} = (L_i)_{i \in I}$ be given. Then \mathbf{I} is learnable from subset queries iff for all $i \in I$, there exists $b_i \in I$ such that, for all $j \in I$ with $L_i \subset L_j$, there exists $k \in I$ with $k \leq_{ll} b_i$ and $L_k \subseteq L_j \wedge L_k \not\subseteq L_i$.*

Corollary 39. *Let an automatic class $\mathbf{I} = (L_i)_{i \in I}$ be given. Then \mathbf{I} is learnable from superset queries iff for all $i \in I$, there exists $b_i \in I$ such that for all $j \in I$ with $L_i \supset L_j$, there exists $k \in I$ with $k \leq_{ll} b_i$ and $L_k \supseteq L_j \wedge L_k \not\supseteq L_i$.*

Corollary 40. *An automatic class \mathbf{I} is learnable from superset queries iff every translation of \mathbf{I} is learnable from positive data.*

Given an automatic class \mathbf{I} of languages all of whose text-preserving translations are learnable from superset and membership queries, \mathbf{I} -query learners that ask superset queries do not benefit from also asking membership queries:

Theorem 41. *If every text-preserving translation of an automatic class \mathbf{I} is learnable from membership and superset queries then \mathbf{I} itself is learnable from superset queries.*

One has an analogous result for subset queries, but considering all translations rather than all text-preserving translations of the class, thanks to a (non text-preserving) automatic **I**-translator Φ that satisfies $\Phi_{\mathbf{I}}(L) = \{i \in I : L_i \subseteq L\}$ for all languages L . Indeed a membership query of the form “is $i \in \Phi_{\mathbf{I}}(L)$?” is then equivalent to the subset query “is $\Phi_{\mathbf{I}}(L_i) \subseteq \Phi_{\mathbf{I}}(L)$?”:

Theorem 42. *If every translation of an automatic class **I** is learnable from membership and subset queries then **I** itself is learnable from subset queries only.*

In the previous result, restriction to text-preserving translations is impossible:

Theorem 43. *Let **I** be the automatic class $\{\emptyset\} \cup \{\{0, 1\}^* \setminus \{x\} : x \in \{0, 1\}^*\}$.*

1. *Every text-preserving translation of **I** is learnable using membership and subset queries.*
2. *Some translation of **I** is not learnable using membership queries only.*
3. **I** is not learnable using subset queries only.

Theorem 44. *Given automatic class $\mathbf{I} = (L_i)_{i \in I}$, every translation of **I** is learnable from membership queries iff $(\forall i)(\exists b_i)(\forall j \neq i)(\exists k \leq_u b_i)[(L_j \subseteq L_k \wedge L_i \not\subseteq L_k) \vee (L_k \subseteq L_j \wedge L_k \not\subseteq L_i)]$.*

7 Conclusion

A notion of learnability is robust if it is immune to natural transformations of the class of objects to be learned. The associated notion of transformation of languages has been defined as a function, called a translator, that maps languages to languages and preserves the inclusion structure of the languages in the original class. Our study has focused on automatic classes of languages, as automaticity is invariant under translation and as this restriction allows one to obtain appealing characterisations of robust learning under many classical learning criteria, namely: consistent and conservative learning, strong-monotonic learning, strong-monotonic consistent learning, finite learning, learning from subset queries, learning from superset queries, and learning from membership queries. The characterisations are natural as they express a particular constraint on the inclusion structure of the original class. In many cases, they are especially strong as they also deal with learnability under those of the translations that are text-preserving, in that they can be generated from an enumeration of a language without necessitating the latter to be “seen as a whole.” In some of the characterisations, learning from every translation turned out to be equivalent to learning from every text-preserving translation (Theorem 16 (standard learnability), Theorem 19 (strong-monotonic learnability), and Theorem 23 (consistent learnability)). Though there are some similarities in the proof, we do not know of a general characterisation of learning criteria for which such a result applies. Open questions remain. For instance, for confident learning, we only found a characterisation with respect to nonrecursive learners.

References

- [1] Angluin, D.: Inductive inference of formal languages from positive data. *Information and Control* 45(2), 117–135 (1980)
- [2] Angluin, D.: Learning regular sets from queries and counterexamples. *Information and Computation* 75, 87–106 (1987)
- [3] Fulk, M.: Robust separations in inductive inference. In: Proc. of the 31st Annual Symposium on Foundations of Computer Science (FOCS), pp. 405–410 (1990)
- [4] Mark Gold, E.: Language identification in the limit. *Information and Control* 10(5), 447–474 (1967)
- [5] Hodgson, B.R.: Décidabilité par automate fini. *Annales des sciences mathématiques du Québec* 7(1), 39–57 (1983)
- [6] Jain, S., Luo, Q., Stephan, F.: Learnability of Automatic Classes. In: Dediu, A.-H., Fernau, H., Martín-Vide, C. (eds.) LATA 2010. LNCS, vol. 6031, pp. 321–332. Springer, Heidelberg (2010)
- [7] Jain, S., Ong, Y.S., Pu, S., Stephan, F.: On automatic families. Manuscript, 2010. Technical Report TRB1/10, School of Computing, National University of Singapore (2010)
- [8] Jain, S., Osherson, D., Royer, J.S., Sharma, A.: Systems That Learn, 2nd edn. MIT Press, Cambridge (1999)
- [9] Jain, S., Stephan, F.: A tour of robust learning. In: Computability and Models. Perspectives East and West, pp. 215–247. Kluwer Academic / Plenum Publishers (2003)
- [10] Jain, S., Smith, C.H., Wiegagen, R.: Robust learning is rich. *Journal of Computer and System Sciences* 62(1), 178–212 (2001)
- [11] Jantke, K.P.: Monotonic and non-monotonic inductive inference. *New Generation Computing* 8, 349–360 (1991)
- [12] Osherson, D.N., Weinstein, S.: Criteria of language learning. *Information and Control* 52, 123–138 (1982)
- [13] Khoussainov, B., Nerode, A.: Automatic presentations of structures. In: Leivant, D. (ed.) LCC 1994. LNCS, vol. 960, pp. 367–392. Springer, Heidelberg (1995)
- [14] Khoussainov, B., Rubin, S.: Automatic structures: overview and future directions. *Journal of Automata, Languages and Combinatorics* 8(2), 287–301 (2003)
- [15] Lange, S., Nessel, J., Zilles, S.: Learning languages with queries. In: Proceedings of the FGML Meeting 2002, pp. 92–99 (2002)
- [16] Lange, S., Zeugmann, T.: Language learning in dependence on the space of hypotheses. In: COLT 1993, pp. 127–136. ACM Press, New York (1993)
- [17] Lange, S., Zeugmann, T., Zilles, S.: Learning indexed families of recursive languages from positive data: a survey. *Theoretical Computer Science* 397, 194–232 (2008)
- [18] Lange, S., Zilles, S.: Formal language identification: Query learning vs. Gold-style learning. *Information Processing Letters* 91(6), 285–292 (2004)
- [19] Ott, M., Stephan, F.: Avoiding coding tricks by hyperrobust learning. *Theoretical Computer Science* 284(1), 161–180 (2002)
- [20] Rabin, M.: Automata on Infinite Objects and Church’s Problem. AMS, Providence (1972)
- [21] Weyl, H.: Symmetry. Princeton University Press, Princeton (1952)
- [22] Wiehagen, R., Zeugmann, T.: Learning and consistency. In: Lange, S., Jantke, K.P. (eds.) GOSLER 1994. LNCS (LNAI), vol. 961, pp. 1–24. Springer, Heidelberg (1995)
- [23] Zeugmann, T.: On Bārzdiņš’ Conjecture. In: Jantke, K.P. (ed.) AII 1986. LNCS, vol. 265, pp. 220–227. Springer, Heidelberg (1987)

Learning and Classifying

Sanjay Jain^{1,*}, Eric Martin², and Frank Stephan^{3,**}

¹ Department of Computer Science,
National University of Singapore, Singapore 117417, Republic of Singapore
sanjay@comp.nus.edu.sg

² School of Computer Science and Engineering,
The University of New South Wales, Sydney NSW 2052, Australia
emartin@cse.unsw.edu.au

³ Department of Mathematics and Department of Computer Science,
National University of Singapore, Singapore 117543, Republic of Singapore
fstephan@comp.nus.edu.sg

Abstract. We define and study a learning paradigm that sits between identification in the limit and classification. More precisely, we expect that a learner be able to identify in the limit which members of a set D of n possible data belong to a target language, where n and D are arbitrary. We show that Ex- and BC-learning are often more difficult than performing this classification task, taking into account desirable constraints on how the learner behaves, such as bounding the number of mind changes and being conservative. Special attention is given to various forms of consistency. We provide a fairly comprehensive set of results that demonstrate the fruitfulness of the approach and the richness of the paradigm.

1 Introduction

The main purpose of the field of inductive inference is to study whether it is possible, following a mechanical procedure subjected to various constraints in its computing abilities, to identify a device (usually a Turing machine or one of its equivalents) from an enumeration of the data it can produce, from an analysis of the finite sets or sequences of those data (see [BB75], [CS83], [Gol67], [JORS99], [ZZ08] for example). Successful learning or identification in the limit requires that the procedure succeeds after enough data have been seen. Success usually means that either a syntactical description of the device has been correctly discovered and will be issued from the point of convergence onwards (called explanatory or Ex-identification, [Gol67]), or that the behaviour of the device has been correctly identified and that from the point of convergence onwards, a syntactical description of the device will be issued but this syntactical description may vary in the face of new data (called behaviourally correct or BC-identification, [Bár74b], [CL82], [CS83], [OW82]).

* Supported in part by NUS grants C252-000-087-001 and R252-000-420-112.

** Supported in part by NUS grant R252-000-420-112.

Another line of research, which has received substantial though considerably less attention from the community, consists in discovering in the limit some property of the device that generates the data being analysed, and is referred to as classification (see for example, [FH94, WS95, SWZ97, CKSS97, Ste01, Jai01, JMS08]). Think of multiclassification as the task of finding out whether the device that generates the data to be observed has property P , for any P in a given set of properties. Learning is in some sense the case of synchronous multiclassification for (usually infinitely many) disjoint classification tasks: it amounts to discovering (the description of) a device characterized as having or not the property of producing d , for any potential datum d . Therefore, it has to be expected that learning is in many cases more difficult than classification. This is still not as obvious as might first appear as we deal with full classification and not *positive* only classification: the work of a classifier is to output either 1 or 0 to indicate whether, on the basis of the sequence of data currently observed, the property holds or does not hold, respectively.

The simplest property one might conceive of is that of membership of some datum to the set of data that the device can produce. The task is trivial with one mind change at most, as a classifier just has to output 0 until the datum appears, if ever. Even if the focus is on the simple properties expressed as data membership, more complex tasks naturally come to mind. In this paper, we will investigate the task of multiclassification—but we will still talk about a “classifier” rather than “multiclassifier”. For instance, the task to classify a device in one of the four classes determined by whether or not it produces the datum 0 and whether or not it produces the datum 1 might require two mind changes sometimes, unless the presence of one number guarantees the presence of the other in the set of data that can be generated by any of the target devices. A natural generalisation of this idea then requires from the classifier that it deals with any finite set of basic classification tasks, be they of the form of data membership or more complex ones. For instance, one might want to consider Boolean combinations of data membership of a certain kind as basic classification tasks.

We mentioned that learning is a kind of synchronous multiclassification, where each classification task expresses the property of generating or not precisely this or that set of data, for all possible such sets. More precisely, learning can be seen as multiclassification of the basic property of data membership, taking together the infinitely many possible data. So what we are describing is a framework where learning is somehow the limit of multiclassification, where the number of basic classification tasks taken together, and required to be finite, grows towards infinity: identifying a device is the limiting case of discovering whether $0, 1, \dots, n$ belong to the set of data that can be generated by that device when n becomes larger and larger.¹ One of the aims of this paper is to make that statement precise and investigate its consequences. In other words, we examine to which extent successful multiclassification makes learning possible.

¹ Considering classification tasks individually is trivial, as we have observed already, whereas considering infinitely many tasks together is too hard and too close to the limiting case of considering all predicates.

We proceed as follows. In Section 2 we introduce the background notation and notions, illustrated with a few examples. This covers the classical concepts of finite, Ex- and BC-learnability, and the new concept of classification, possibly constrained by counterparts to the usual notions of mind changes and conservativeness, and three different notions of consistency. Section 3 presents negative results on BC-learnability, shown to be harder than classification variously constrained. In contrast, Section 4 shows how finite or Ex-learnability can be obtained from various classification strategies. Section 5 investigates in more depth the relationships between the various forms of classification.

2 Definitions and Examples

2.1 General Notions and Learnability

We denote by \mathbb{N} the set of natural numbers. The length of a finite sequence σ is denoted $\text{len}(\sigma)$. Given a sequence σ , for $i < \text{len}(\sigma)$, $\sigma(i)$ represents the i th member of σ (starting with the 0th element); for $i \leq \text{len}(\sigma)$, $\sigma|_i$ represents the initial segment of σ of length i . We use \star to denote concatenation between finite sequences, and identify a sequence of length 1 with its unique member. Given $x, i \in \mathbb{N}$, we denote by x^i the unique member of $\{x\}^i$. Given a set E , we denote by E^* the set of finite sequences of members of E . Given $E \subseteq \mathbb{N}$, we let χ_E denote the characteristic function of E .

We fix an acceptable enumeration $(\varphi_e)_{e \in \mathbb{N}}$ of the partial recursive functions over \mathbb{N} , and for all $e \in \mathbb{N}$, denote by W_e the domain of φ_e (see, for example [Rog67]). We fix a computable coding of all finite sequences of members of \mathbb{N} into \mathbb{N} , and denote by $\langle n_1, \dots, n_k \rangle$ the image of (n_1, \dots, n_k) by this coding. Also, we fix a computable coding of all pairs of members of \mathbb{N} into \mathbb{N} , and denote by $\langle n_1, n_2 \rangle_2$ the image of (n_1, n_2) by this coding for all $n_1, n_2 \in \mathbb{N}$, with the property that $\langle n_1, n_2 \rangle_2 \leq \langle n'_1, n'_2 \rangle_2$ and $\langle n_1, n_2 \rangle_2 \leq \langle n_1, n'_2 \rangle_2$ whenever $n_1 \leq n'_1$ and $n_2 \leq n'_2$. Given a total recursive function g , the *retraceable function determined by* g is the total recursive function f such that $f(n) = \langle f(0), \dots, f(n-1), g(n) \rangle$ for all $n \in \mathbb{N}$. A *retraceable function* is the retraceable function determined by some given recursive function.

A language is a set of the form W_e . We denote by \mathcal{L} a class of languages. Given a language L , a *text for* L is an infinite enumeration of the members of L , possibly with repetitions of some members of L and possibly with occurrences of $\#$, the “pause” symbol [Gö67]. Given a recursive function f , the *canonical text for* f is the infinite sequence whose i th term, $i \in \mathbb{N}$, is $\langle i, f(i) \rangle_2$. A *text* is a text for some language. We denote $(\mathbb{N} \cup \{\#\})^*$ by SEQ . Given $\sigma \in \text{SEQ}$, $\text{rng}(\sigma)$ denotes the set of natural numbers that occur in σ . A member σ of SEQ is said to be *for* a language L if $\text{rng}(\sigma) \subseteq L$, and *for* \mathcal{L} if it is for some member of \mathcal{L} .

A *learner* is a computable function from SEQ into $\mathbb{N} \cup \{\text{?}\}$ (where ? is a distinguished symbol which allows the learner to express that it makes no hypothesis). Given a language L , a learner M *Ex-learns* L , see [Gö67], iff for all texts t for L , there exists $e \in \mathbb{N}$ such that $W_e = L$ and $M(\sigma) = e$ for cofinitely many finite initial segments σ of t . A learner *Ex-learns* \mathcal{L} if it Ex-learns all members of \mathcal{L} .

Given a language L , a learner M *BC-learns* L , see [Bár74b, CL82, OW82], iff for all texts t for L , $W_{M(\sigma)} = L$ for cofinitely many finite initial segments σ of t . A learner *BC-learns* \mathcal{L} if it BC-learns all members of \mathcal{L} . Given a language L , a learner M *finitely learns* L , see [Gol67], iff M Ex-learns L and for all texts t for L and for all finite initial segments σ_1 and σ_2 of t , if $M(\sigma_1)$ and $M(\sigma_2)$ are both distinct from $?$ then they are equal. A learner *finitely learns* \mathcal{L} if it finitely learns all members of \mathcal{L} . Given $c \in \mathbb{N}$, we say that a learner M that Ex-learns \mathcal{L} *makes at most c mind changes*, see [CS83], iff there is no infinite sequence t of members of $\mathbb{N} \cup \{\#\}$ and no strictly increasing sequence (i_0, \dots, i_c) of $c+1$ integers such that $M(t|_{i_0}) \neq ?$ and $M(t|_{i_k}) \neq M(t|_{i_{k+1}})$ for all $k \leq c$; when $c=0$ we rather say that M *makes no mind change*.

2.2 Classification

A *predicate* is a property that can be true or false for any given language (1 or 0 is then the *truth value* of that predicate *for* that language, respectively). Given a sequence τ of predicates, a language L and a sequence τ' of members of $\{0, 1\}$, we say that τ' is the *sequence of truth values of τ for L* if τ' has the same length as τ and for all $i < \text{len}(\tau)$, $\tau'(i)$ is the truth value of $\tau(i)$ for L . For $n \in \mathbb{N}$, let \in_n be the predicate that is true for languages that contain n , and false for others. Let In denote the set of all predicates of the form \in_n , $n \in \mathbb{N}$. Let Boole denote the set of all predicates that are Boolean combinations of predicates in In .

Definition 1. Let P be a set of predicates.

A *general P -classifier* is a total function M on $\text{SEQ} \times P^*$ such that for all $\sigma \in \text{SEQ}$, for all $n \in \mathbb{N}$ and for all $\tau \in P^n$, $M(\sigma, \tau)$ is a member of $\{0, 1\}^n \cup \{?\}$.

A *P -classifier* is a computable general P -classifier.

Definition 2. Let P be a set of predicates and let M be a general P -classifier.

Given a language L , we say that M *classifies* L iff the following holds. For any text t for L and any $\tau \in P^*$, for cofinitely many finite initial segments σ of t , $M(\sigma, \tau)$ is the sequence of truth values of τ for L .

We say that M *classifies* \mathcal{L} iff M classifies all members of \mathcal{L} .

Given a set P of predicates, a *classification task (with respect to P)* refers to any finite sequence of members of P . To say that a general P -classifier M is successful on a classification task τ then means that for all texts t for a member L of \mathcal{L} and for cofinitely many finite initial segments σ of t , $M(\sigma, \tau)$ is the sequence of truth values of τ for L .

Classification, especially when requiring the classifier to decide whether the input language belongs to one of several disjoint classes of languages, has been studied by several authors earlier, see [FH94, WS95, SWZ97, CKSS97, Ste01, Jai01, JMS08] for example. The definitions above are about multi-classification, in which the classifier has to simultaneously perform several classification tasks.

We now consider the important constraint of consistency. In the context of this paper, and as opposed to the classical notion of consistency used for learnability (see [Ang80, Bár74a, JB81, WL76]) it is natural to impose that the decisions on

the predicates to be dealt with be consistent not only with the available data (which can always be easily achieved when deciding a finite number of predicates), but also with a language in the class under consideration. As classifiers have to deal with arbitrarily large finite set of predicates, one can further consider whether the classification, on a particular input, is compatible with some fixed language L when the predicate set is large enough. This L above may or may not be in the class of languages under consideration. The following definition explores these possibilities.

Notions of conservativeness [Ang80] and mind changes [CS83] will also play an important role in this paper, but they are straightforward adaptations of the classical notions considered in inductive inference.

Definition 3. Let a class of languages \mathcal{L} , a set of predicates P , and a general P -classifier M be given.

We say that M is *consistent on* \mathcal{L} iff for all $\sigma \in \text{SEQ}$ and $\tau \in P^*$, if σ is for \mathcal{L} then there exists $L \in \mathcal{L}$ such that σ is for L and $M(\sigma, \tau)$ is the sequence of truth values of τ for L .

We say that M is *strongly consistent on* \mathcal{L} iff M is consistent on \mathcal{L} and for all $\sigma \in \text{SEQ}$, if σ is for \mathcal{L} then there exists a language L and a $\tau \in P^*$ such that σ is for L and for all $\tau' \in P^*$ satisfying $\text{rng}(\tau) \subseteq \text{rng}(\tau')$, $M(\sigma, \tau')$ is the sequence of truth values of τ' for L .

We say that M is *strongly consistent within class on* \mathcal{L} iff M is consistent on \mathcal{L} and for all $\sigma \in \text{SEQ}$, if σ is for \mathcal{L} then there exists $L \in \mathcal{L}$ and a $\tau \in P^*$ such that σ is for L and for all $\tau' \in P^*$ satisfying $\text{rng}(\tau) \subseteq \text{rng}(\tau')$, $M(\sigma, \tau')$ is the sequence of truth values of τ' for L .

(Based on [Ang80]) We say that M is *conservative on* \mathcal{L} iff for all $\sigma \in \text{SEQ}$, $n \in \mathbb{N} \cup \{\#\}$ and $\tau \in P^*$, if there exists $L \in \mathcal{L}$ such that $\sigma \star n$ is for L and $M(\sigma, \tau)$ is the sequence of truth values of τ for L then $M(\sigma, \tau) = M(\sigma \star n, \tau)$.

When \mathcal{L} is clear from the context, we omit “on \mathcal{L} ” in the previous expressions.

Definition 4. (Based on [CS83]) Let a set P of predicates, a general P -classifier M that classifies \mathcal{L} , and $c \in \mathbb{N}$ be given. We say that M *makes at most c mind changes* iff there is no text t for \mathcal{L} , no $\tau \in P^*$ and no strictly increasing sequence (i_0, \dots, i_c) of $c+1$ integers such that $M(t|_{i_0}, \tau) \neq ?$ and $M(t|_{i_k}, \tau) \neq M(t|_{i_{k+1}}, \tau)$ for all $k \leq c$; when $c = 0$ we rather say that M *makes no mind change*.

2.3 Illustrative Examples

A successful classifier of a learnable class does not immediately provide a learner of that class. For an illustration, the next example exhibits a class of languages \mathcal{L} and a consistent and conservative In-classifier that classifies \mathcal{L} using at most one mind change, and for all finite initial segments of a text for some member of \mathcal{L} , is incorrect on infinitely many classification tasks.

Example 5. For all $i \in \mathbb{N}$, let $L_i = \{2j \mid j \leq i\} \cup \{2i + 1\}$ and suppose that \mathcal{L} is equal to $\{2\mathbb{N}\} \cup \{L_i \mid i \in \mathbb{N}\}$. Let M be an In-classifier such that for all $\sigma \in \text{SEQ}$ for \mathcal{L} and sequences of the form $(\in_{n_0}, \dots, \in_{n_k})$, the following holds.

If there exists $i \in \mathbb{N}$ with $2i + 1 \in \text{rng}(\sigma)$ then $M(\sigma, (\in_{n_0}, \dots, \in_{n_k}))$ is equal to $(\chi_{L_i}(n_0), \dots, \chi_{L_i}(n_k))$. Otherwise, if the greatest number in $\{n_0, \dots, n_k\}$, say n , is both odd and greater than any number in $\text{rng}(\sigma)$, then for all $j \leq k$, $M(\sigma, (\in_{n_0}, \dots, \in_{n_k}))(j) = 1$ iff n_j is even or $n_j = n$. Otherwise, for all $j \leq k$, $M(\sigma, (\in_{n_0}, \dots, \in_{n_k}))(j) = 1$ iff n_j is even. It is easy to see that M In-classifies \mathcal{L} , is consistent and conservative, and makes at most one mind change; also, for all $\sigma \in \text{SEQ}$, if σ is for $2\mathbb{N}$ then there exists infinitely many sequences of the form $(\in_{n_0}, \dots, \in_{n_k})$ such that $M(\sigma, (\in_{n_0}, \dots, \in_{n_k})) \neq (\chi_{2\mathbb{N}}(n_0), \dots, \chi_{2\mathbb{N}}(n_k))$.

Successful classification is possible with respect to a nonlearnable class, as illustrated in the next example.

Example 6. Suppose that \mathcal{L} consists of \mathbb{N} and all cosingletons (sets of the form $\mathbb{N} \setminus \{n\}$). Then no learner BC-learns \mathcal{L} (this follows easily from Angluin's tell-tale characterization [Ang80] as \mathbb{N} does not have a tell-tale in \mathcal{L}). Let a Boole-classifier M be defined as follows. Let $\sigma \in \text{SEQ}$ be for \mathcal{L} . Let $\tau \in \text{Boole}^*$ be given. If there exists a unique integer $n \notin \text{rng}(\sigma)$ such that \in_n is used in one of τ 's predicates, then $M(\sigma, \tau)$ is the sequence of truth values of τ in $\mathbb{N} \setminus \{n\}$. Otherwise, $M(\sigma, \tau)$ is the sequence of truth values of τ in \mathbb{N} . It is immediately verified that M is consistent and classifies \mathcal{L} making at most two mind changes.

3 Classification Versus BC-Learnability

We start our investigations with three results on Boole-classification. Almost all other results deal with In-classification.

Theorem 7. There exists a class \mathcal{L} of languages that some Boole-classifier classifies making no mind change, but no learner BC-learns \mathcal{L} .

Proof. For any set E , let f_E denote the retraceable function determined by χ_E . Let $S_E = \{\langle n, f_E(n) \rangle_2 : n \in \mathbb{N}\}$. Let \mathcal{L} denote the set of all S_E , where E is a recursive set. Then, since the class of recursive sets cannot be BC-learnt from informant [CL82], we immediately have that \mathcal{L} is not in BC.

Let M be a Boole-classifier such that for all $\sigma \in \text{SEQ}$ that are for \mathcal{L} and for all $\tau \in \text{Boole}^*$, the following holds. If there exists an integer n such that \in_n is used in one of τ 's predicates and every member of $\text{rng}(\sigma)$ is of the form $\langle m, x \rangle_2$ with $m < n$, then $M(\sigma, \tau) = ?$. Otherwise, $M(\sigma, \tau)$ is the sequence of truth values of τ for S_E for any finite set E such that σ is for S_E (note that all E , such that σ is for S_E , will give the same result for $M(\sigma, \tau)$). It is easily verified that M classifies \mathcal{L} making no mind change. \square

Theorem 8. There exists a class \mathcal{L} of languages such that some strongly consistent within class general Boole-classifier classifies \mathcal{L} making at most one mind change, but no learner BC-learns \mathcal{L} .

Proof. Let $(M_e)_{e \in \mathbb{N}}$ denote an effective enumeration of all learners. Given a subset E of \mathbb{N} , let f_E denote the retraceable function determined by χ_E . We

inductively define a sequence $(E_{e,i})_{e,i \in \mathbb{N}}$ of finite subsets of \mathbb{N} such that for all $e, i \in \mathbb{N}$, $E_{e,i} \subseteq E_{e,i+1}$, in such a way that the set of triples $\langle e, i, x \rangle$ with $e \in \mathbb{N}$, $i \in \mathbb{N}$ and $x \in E_{e,i}$ is recursively enumerable. For all $e \in \mathbb{N}$, set $E_{e,0} = \{e\}$. Let natural numbers e and i be given, and assume that $E_{e,i}$ has been defined. Look for an initial segment σ of the canonical text for $f_{E_{e,i}}$, with $\text{len}(\sigma) > \max(E_{e,i})$, and for $j \geq \text{len}(\sigma)$ such that $M_e(\sigma)$ is defined and $\langle j, f_{E_{e,i}}(j) \rangle_2 \in W_{M_e(\sigma)}$, and set $E_{e,i+1} = E_{e,i} \cup \{j\}$; if there is no such σ and j then set $E_{e,i+1} = E_{e,i}$. For all $e \in \mathbb{N}$, let S_e denote $\{\langle n, f_{\bigcup_{i \in \mathbb{N}} E_{e,i}}(n) \rangle_2 \mid n \in \mathbb{N}\}$. Denote $\{\langle n, f_\emptyset(n) \rangle_2 \mid n \in \mathbb{N}\}$ by S . Finally, set $\mathcal{L} = \{S\} \cup \{S_e \mid e \in \mathbb{N}\}$.

Let M be a general Boole-classifier such that for all $\sigma \in \text{SEQ}$ that are for \mathcal{L} , the following holds. If σ is for S then for all $\tau \in \text{Boole}^*$, $M(\sigma, \tau)$ is the sequence of truth values of τ for S . Otherwise, there exists a unique $e \in \mathbb{N}$ such that σ is for S_e , and for all $\tau \in \text{Boole}^*$, $M(\sigma, \tau)$ is the sequence of truth values of τ in S_e . Trivially, M is strongly consistent within class and classifies \mathcal{L} with at most one mind change. Moreover, it follows immediately from the definition of \mathcal{L} that for all $e \in \mathbb{N}$, M_e does not BC-learn S_e . So no learner BC-learns \mathcal{L} . \square

Theorem 9. *There exists a class \mathcal{L} of languages such that some strongly consistent Boole-classifier classifies \mathcal{L} making at most two mind changes, but no learner BC-learns \mathcal{L} .*

We now focus on In-classification. The last result of this section exhibits a case where BC-learnability fails whereas consistent and conservative classification succeeds, even when the latter is constrained to using very few mind changes.

Theorem 10. *There exists a class \mathcal{L} of languages such that some consistent and conservative In-classifier classifies \mathcal{L} making at most 2 mind changes, but no learner BC-learns \mathcal{L} .*

4 Classification Versus Finite and Ex-learnability

The first result of this section exhibits a class of languages that is easy to learn, as it is learnable with no mind change, whereas classification requires sometimes to go through all possibilities of making n predicates true or false before converging to the correct answer.

Theorem 11. *There exists a class \mathcal{L} of finite languages such that some learner finitely learns \mathcal{L} and some consistent and conservative In-classifier classifies \mathcal{L} . Moreover, for all consistent In-classifiers M and for all $n \in \mathbb{N}$, there is $\tau \in \text{In}^n$ and a text t for \mathcal{L} such that $\{M(t|_i, \tau) \mid i \in \mathbb{N}\}$ has cardinality 2^n .*

The next results in this section show how to construct an Ex-learner from a classifier constrained in the maximum number of mind changes it is allowed to make, and by consistency and conservativeness requirements.

Theorem 12. *Let \mathcal{L} be a class of languages that some strongly consistent and conservative In-classifier classifies making at most k mind changes for some k in \mathbb{N} . Then some learner Ex-learns \mathcal{L} . Moreover, all members of \mathcal{L} are recursive.*

Proof. For all $n \in \mathbb{N}$, set $\tau_n = (\epsilon_0, \dots, \epsilon_n)$. Let C be a strongly consistent and conservative In-classifier that classifies \mathcal{L} making at most k mind changes. Define a learner M as follows. Let $\sigma \in \text{SEQ}$ be given. Then $M(\sigma)$ is an integer e such that for all $n \in \mathbb{N}$, $n \in W_e$ iff either $n \leq \text{len}(\sigma)$ and $C(\sigma, \tau_{\text{len}(\sigma)})(n) = 1$, or $n > \text{len}(\sigma)$ and $C(\sigma, \tau_n)(n) = 1$. As the process describing W_e in the previous statement gives a decision procedure, we can make sure that the learner makes a mind change only when the conjectured language changes. So to complete the proof of the theorem, it suffices to verify that M BC-learns \mathcal{L} . Let a text t for $L \in \mathcal{L}$ be given. We inductively define a sequence $(\sigma_i)_{i \leq k}$ of finite initial segments of t and a sequence $(n_i)_{i \leq k}$ of natural numbers, such that for all $i < k$, $\text{len}(\sigma_i) \leq n_i \leq \text{len}(\sigma_{i+1})$. Let σ_0 be the empty sequence, and let $n_0 \in \mathbb{N}$ be such that for all $n \geq n_0$ and $n' \geq n$, $C(\sigma_0, \tau_n)$ is an initial segment of $C(\sigma_0, \tau_{n'})$ (such an n_0 exists since M is strongly consistent). Let $i < k$ be given and assume that for all $j \leq i$, σ_j and n_j have been defined. If for all initial segments σ of t and all n such that $n \geq \text{len}(\sigma) \geq n_i$, $C(\sigma, \tau_n) = C(\sigma_i, \tau_n)$, then clearly $M(\sigma)$ is an index for L , for all initial segments σ of t with $\text{len}(\sigma) \geq n_i$. So, suppose that there exists a finite initial segment σ of t and n such that $n \geq \text{len}(\sigma) \geq n_i$ and $C(\sigma, \tau_n) \neq C(\sigma_i, \tau_n)$. Since C is conservative, $C(\sigma_i, \tau_n)$ is not the sequence of truth values of τ_n for L' , for any $L' \in \mathcal{L}$ which contains $\text{rng}(\sigma)$. So, since C is consistent, $C(\sigma_i, \tau_n)$ is not an initial segment of $C(\sigma, \tau_{n'})$ for any $n' \geq n$. We then set $\sigma_{i+1} = \sigma$ and let $n_{i+1} \geq n$ be such that for all $n'' \geq n_{i+1}$ and $n''' \geq n''$, $C(\sigma_{i+1}, \tau_{n''})$ is an initial segment of $C(\sigma_{i+1}, \tau_{n'''})$ (such an n_{i+1} exists since C is strongly consistent). Note that for all $m \geq n_{i+1}$, $C(\sigma_{i+1}, \tau_m) \neq C(\sigma_i, \tau_m)$: this follows from the respective definitions of σ_i , σ_{i+1} , n_i and n_{i+1} , the fact that $n_i \leq n \leq n_{i+1}$, and the fact that $C(\sigma_i, \tau_n)$ is an initial segment of $C(\sigma_i, \tau_m)$, but not of $C(\sigma_{i+1}, \tau_m)$. Since C makes no more than k mind changes, we conclude that for all finite initial segments σ of t that extend σ_k and are of length greater than n_k , $M(\sigma)$ is an index of the language L . \square

Theorem 13. *Let \mathcal{L} be a class of languages such that some consistent In-classifier classifies \mathcal{L} making at most one mind change. Then some strongly consistent In-classifier classifies \mathcal{L} making at most one mind change.*

Proof. Let M be a consistent In-classifier that classifies \mathcal{L} making at most one mind change. Given $i \in \mathbb{N}$, let τ_i denote $(\epsilon_0, \dots, \epsilon_i)$, and let A_i be the set of all $j \leq i$ with $M(((), \tau_i))(j) = 1$. Let B denote the set of all members of $\bigcup \mathcal{L}$ that do not belong to A_i for infinitely many $i \in \mathbb{N}$, and let C denote the set of members of $\bigcup \mathcal{L}$ that belong to A_i for all but finitely many $i \in \mathbb{N}$. Given $x \in \mathbb{N}$ and $i \geq x$, let S_x^i denote $M(((), \tau_i))$ if $x \in A_i$, and $M((x), \tau_i)$ otherwise.

First note that for all $x \in B$, there exists a unique language L_x in \mathcal{L} such that $x \in L_x$, and for all $i \geq x$, S_x^i is the sequence of truth values of τ_i for L_x . Indeed, since M makes at most one mind change, $M((x), \tau_i)$ is necessarily the sequence of truth values of τ_i for L_x for the infinitely many $i \in \mathbb{N}$ such that $x \notin A_i$, which uniquely determines L_x . Since M is consistent, $M(((), \tau_i))$ is then necessarily the sequence of truth values of τ_i for L_x for all $i \in \mathbb{N}$ such that $x \in A_i$.

We first show that C is r.e. There is nothing to prove if C is recursive, so assume otherwise. For a contradiction, suppose that there exists $x \in C$ such

that for all $i \geq x$, S_x^i is an initial segment of S_x^{i+1} . Since $S_x^i = M((\lambda), \tau_i)$ for cofinitely many $i \in \mathbb{N}$, it follows from the definition of A_i , $i \in \mathbb{N}$, that for all $y \geq x$, either $y \in C$ and $S_x^y(y) = 1$, or $y \notin C$ and $S_x^y(y) = 0$, which contradicts the hypothesis that C is not recursive. Together with the previous observation on the members of B , this implies that C is the set of all $x \in \bigcup_{i \in \mathbb{N}} A_i$ such that S_x^i is not an initial segment of S_x^{i+1} for some $i \geq x$, hence is r.e.

We now show that C is recursive. Assume that there exists a sequence $(x_i)_{i \in \mathbb{N}}$ of pairwise distinct members of C and a sequence $(n_i)_{i \in \mathbb{N}}$ of members of \mathbb{N} such that for all $i \in \mathbb{N}$, $n_i \geq x_i$ and $x_i \notin A_{n_i}$. One can assume without loss of generality that the sequences $(x_i)_{i \in \mathbb{N}}$ and $(n_i)_{i \in \mathbb{N}}$ are recursive. For all $i \in \mathbb{N}$, set $S^i = M((x_i), \tau_{n_i})$. Then for all $i \in \mathbb{N}$, S^i is the sequence of truth values of τ_{n_i} in every language in \mathcal{L} that contains x_i ; moreover, since x_i belongs to A_j for cofinitely many $j \in \mathbb{N}$, S^i is an initial segment of $M((\lambda), \tau_j)$ for cofinitely many $j \in \mathbb{N}$. Hence for all $i, j \in \mathbb{N}$, one of S^i and S^j is an initial segment of the other, and for all $y \in B$, $S^i(y) = 0$ (as otherwise, y belongs to A_j for cofinitely many $j \in \mathbb{N}$). Thus C is the language L such that for all $i \in \mathbb{N}$, S^i is the sequence of truth values of τ_{n_i} for L . Moreover, since the sequence $(S^i)_{i \in \mathbb{N}}$ is r.e., C is recursive. On the other hand, if there exists no infinite sequence $(x_i)_{i \in \mathbb{N}}$ of pairwise distinct members of C and corresponding sequence $(n_i)_{i \in \mathbb{N}}$ of members of \mathbb{N} such that for all $i \in \mathbb{N}$, $n_i \geq x_i$ and $x_i \notin A_{n_i}$, then $\bigcup_i A_i \cap B$ is r.e.; thus C is recursive (as both, C and $\{i : i \in A_i\} \setminus C$ are recursively enumerable and the recursive set $\{i : i \in A_i\}$ contains all but finitely many elements in C).

Now observe that no member L of \mathcal{L} is strictly included in C . Otherwise, there would be a member σ of SEQ , with $\text{rng}(\sigma) \subseteq L$, and $i \in \mathbb{N}$ such that $x \in C \cap A_i \setminus L$ but $M(\sigma, \tau_i)(x) = 0$. But then there exists a $\sigma' \in \text{SEQ}$ extending σ such that $\text{rng}(\sigma') \subseteq C$ and $M(\sigma', \tau_i)(x) = 1$, which contradicts that M makes at most 1 mind change.

We can now define a strongly-consistent In-classifier N as follows. Let $\sigma \in \text{SEQ}$ be for \mathcal{L} , and let $\tau \in \text{In}^*$ be given. If σ is for C then $N(\sigma, \tau)$ is the sequence of truth values of τ for C . Otherwise, by the previous remark, there exists a least $x \in \text{rng}(\sigma)$ that does not belong to C , and $N(\sigma, \tau)$ is the sequence of truth values of τ for L_x , the unique member of \mathcal{L} that contains x (by the definition of L_x , $x \in B$, $N(\sigma, \tau)$ can be effectively determined using the sequences S_x^i , $i \geq x$). Obviously, N classifies \mathcal{L} with at most one mind change. \square

Corollary 14. *Let \mathcal{L} be a class of languages. If some consistent In-classifier classifies \mathcal{L} making at most one mind change then some learner Ex-learns \mathcal{L} .*

5 Limitations of Classification Variously Constrained

The results in this section have the flavour of the standard results in inductive inference that compare how various constraints imposed on the learners affect their power to learn as opposed to other constraints. These matters are investigated here in the context of our notion of classification.

Theorem 15. *There exists a class \mathcal{L} of languages such that some conservative In-classifier classifies \mathcal{L} using at most one mind change, but no consistent In-classifier classifies \mathcal{L} .*

Theorem 16. *There exists a class \mathcal{L} of languages such that some conservative and strongly consistent In-classifier classifies \mathcal{L} using at most one mind change, but no strongly consistent within class In-classifier classifies \mathcal{L} .*

The above diagonalization is optimal, as any consistent classifier using no mind changes outputs only correct classifications (for any language in the class) on empty σ . It is thus also strongly consistent within class making no mind change. Note that by Theorem 13, the following is also optimal.

Theorem 17. *There exists a class \mathcal{L} of languages such that some conservative and consistent In-classifier classifies \mathcal{L} using at most two mind changes, but no strongly consistent In-classifier classifies \mathcal{L} .*

Proof. Let $(M_e)_{e \in \mathbb{N}}$ be an effective enumeration of all In-classifiers. For the sake of defining \mathcal{L} , we first define a family $(F_e^t)_{e,t \in \mathbb{N}}$ of finite subsets of \mathbb{N} of the form $\{0, p_1, q_1, \dots, p_n, q_n\}$ where $n \in \mathbb{N}$, p_1, \dots, p_n are even, q_1, \dots, q_n are odd, and $1 < p_1 < q_1 < \dots < p_n < q_n$. It will be the case that:

- for each e, t , F_e^t is finite and can be recursively determined from e, t ;
- for all $e, t, t' \in \mathbb{N}$, if $t \leq t'$ then all even numbers in F_e^t belong to $F_e^{t'}$;
- for all $e \in \mathbb{N}$, if $\bigcup_{t \in \mathbb{N}} F_e^t$ is infinite then for all $t, t' \in \mathbb{N}$ with $t \leq t'$, $F_e^t \subseteq F_e^{t'}$;
- F_e^t contains all even numbers in $\bigcup_{t' \in \mathbb{N}} F_e^{t'}$ which are $\leq t$, except perhaps for the largest such even number.

For all $e, t \in \mathbb{N}$, let τ_e^t denote $\langle \langle e, 0 \rangle_2, \dots, \langle e, t \rangle_2 \rangle$. Let $e \in \mathbb{N}$ be given. Define F_e^0 as $\{0\}$. Let $t \in \mathbb{N}$ be given, and assume that F_e^t has been defined, but F_e^{t+1} has not been defined yet. If there exists an odd number $q \leq t$ such that within t computation steps, $M_e(\langle e, 0 \rangle_2, \tau_e^t)$ assigns the value 1 to $\in_{\langle e, q \rangle_2}$, then for all $t' > t$, let $F_e^{t'}$ be defined as F_e^t if $q \notin F_e^t$, and as $(F_e^t \setminus \{q\}) \cup \{q+2\}$ otherwise. Suppose that there is no such odd number. Let p be equal to 2 if $F_e^t = \{0\}$, and to 3 plus the largest odd number in F_e^t otherwise. If

- $M_e(\langle e, 0 \rangle_2, \tau_e^t)$ does not halt within t steps, or $M_e(\langle e, 0 \rangle_2, \tau_e^t)$ does not assign the value 1 to $\in_{\langle e, n \rangle_2}$ for some even number $n \in F_e^t$,
- or if $p > t$,
- or if $p \leq t$ and within t computation steps, $M_e(\langle e, 0 \rangle_2, \tau_e^t)$ does not assign the value 0 to $\in_{\langle e, p \rangle_2}$,

then set $F_e^{t+1} = F_e^t$. Otherwise, define F_e^{t+1} as the union of F_e^t with $\{p, 8t+1\}$. This completes the definition of $(F_e^t)_{e,t \in \mathbb{N}}$. For all $e, t \in \mathbb{N}$, we now define a language S_e^t and set $\mathcal{L} = \{\emptyset\} \cup \{S_e^t \mid e \in \mathbb{N}, t \in \mathbb{N}\}$. Let $e, t \in \mathbb{N}$ be given. Let E denote the set of all even numbers in F_e^t .

- Suppose that $\bigcup_{t' > t} F_e^{t'} \cap 2\mathbb{N} = E$. Then set $S_e^t = \{\langle e, x \rangle_2 : x \in E\}$.
- Suppose that $\bigcup_{t' > t} F_e^{t'} \cap 2\mathbb{N} \neq E$. Let $t' > t$ be least such that $F_e^{t'} \cap 2\mathbb{N} \neq F_e^t \cap 2\mathbb{N}$. Let q be the largest odd number of $F_e^{t'}$. If $\bigcup_{t'' > t} F_e^{t''}$ does not contain $q + 2$ then set $S_e^t = \{\langle e, x \rangle_2 : x \in E \cup \{q\}\}$; otherwise, set $S_e^t = \{\langle e, x \rangle_2 : x \in E \cup \{q + 2\}\}$.

We now show that \mathcal{L} satisfies the claim of the theorem. Define an In-classifier M as follows. Let $\sigma \in \text{SEQ}$ be for \mathcal{L} , and let $\tau \in \text{In}^*$ be given.

If $\text{rng}(\sigma) = \emptyset$ then let $M(\sigma, \tau)$ be the sequence of truth values of τ for \emptyset .

If $e \in \mathbb{N}$ is such that σ contains an element of the form $\langle e, x \rangle_2$ but no element of the form $\langle e, 2y + 1 \rangle_2$, and if $t \in \mathbb{N}$ is largest with $\in_{\langle e, t \rangle_2}$ occurring in τ , then let $M(\sigma, \tau)$ be the sequence of truth values of τ in the set of all elements of the form $\langle e, 2x \rangle_2$ that occur in $F_e^t \cup \text{rng}(\sigma)$. Note that if some (necessarily unique) element of the form $\langle e, 2x \rangle_2$ with $2x \leq t$ belongs to $\text{rng}(\sigma) \setminus F_e^t$, then $2x$ is larger than all even numbers in F_e^t and $M(\sigma, \tau)$ is the sequence of truth values of τ for L for any $L \in \mathcal{L}$ such that σ is for L ; hence M will not make any further mind change on any extension of σ that is for \mathcal{L} .

If $e \in \mathbb{N}$ is such that σ contains an element of the form $\langle e, 2x + 1 \rangle_2$ (which is necessarily unique), then let $M(\sigma, \tau)$ be the sequence of truth values of τ for S_e^t for any t such that S_e^t contains $2x + 1$. Note that if $2x + 1$ belongs to S_e^t for some $t \in \mathbb{N}$, then one can effectively find such a t by looking for a t' such that $2x - 1$ or $2x + 1$ belongs to $F_e^{t'}$; moreover, M will not make any further mind change on any extension of σ that is for \mathcal{L} . It is easily verified that M is consistent and conservative and classifies \mathcal{L} making at most 2 mind changes.

Let $e, t \in \mathbb{N}$ be given. If there exists an odd number $x \leq t$ such that $M_e(\langle e, 0 \rangle_2, \tau_e^t)$ assigns the value 1 to $\in_{\langle e, x \rangle_2}$, then by construction, M_e is not consistent. Otherwise, if there exists $t \in \mathbb{N}$ such that $F_e^t = \bigcup_{s \in \mathbb{N}} F_e^s$, then by construction again, M_e is not consistent. Otherwise, still by construction, there are infinitely many $t \in \mathbb{N}$ such that $M_e(\langle e, 0 \rangle_2, \tau_e^{t+1})$ is not an extension of $M_e(\langle e, 0 \rangle_2, \tau_e^t)$ (because when M_e deals with $\langle e, 0 \rangle_2$ as first argument and with τ_e^t as second argument with larger and larger values of t , M_e is forced to give the value false and then the value true to \in_p for infinitely many $p \in 2\mathbb{N}$). Hence M_e is not strongly consistent. Thus, no strongly consistent In-classifier classifies \mathcal{L} . \square

Theorem 18. *Let $k > 1$ be given. There exists a class \mathcal{L} of languages such that some strongly consistent within class and conservative In-classifier classifies \mathcal{L} making at most k mind changes, but no In-classifier classifies \mathcal{L} making at most $k - 1$ mind changes.*

If conservative classifiers were required to output an initial hypothesis different to ? (in line with what is required of consistent classifiers), then the number of mind changes for conservative classifiers could be one more than the number of mind changes needed by strongly consistent classifiers.

Similarly, if conservative learners could not output ?, then the number of mind changes by the conservative learner in the next theorem would increase by one.

Theorem 19. Let $k \in \mathbb{N}$ and a class of languages \mathcal{L} be given. If some total In-classifier classifies \mathcal{L} making at most k mind changes, then some conservative In-classifier classifies \mathcal{L} making at most k mind changes.

Proof. Let M be a total In-classifier that classifies \mathcal{L} making at most k mind changes. Let an In-classifier N be defined as follows. Let $\tau \in \text{In}^*$ be given. Set $N((\), \tau) = ?$. For all $\sigma \in \text{SEQ}$ and $x \in \mathbb{N} \cup \{\#\}$, let $N(\sigma \star x, \tau)$ be equal to $M(\sigma \star x, \tau)$ if the latter is the sequence of truth values of τ in $\text{rng}(\sigma)$, and to $N(\sigma, \tau)$ otherwise. It is immediately seen that M is conservative and classifies \mathcal{L} making at most k mind changes. \square

The last result of the paper deals with the hierarchy of the number of mind changes that might be needed as a function of the number of membership predicates to decide. It shows that for any choice of such a function f , for some class of languages \mathcal{L} , it is impossible for any consistent classifier for \mathcal{L} to decide n predicates using at most $f(n)$ mind changes, though for some particular choices of f , some classifier might do so with $f(n+1)$ mind changes at most.

Theorem 20. For every strictly increasing total function f from \mathbb{N} into \mathbb{N} there exists a class \mathcal{L} of languages, which satisfies the following:

- (a) For all consistent In-classifiers M for \mathcal{L} and $n > 0$, there exists $\tau \in \text{In}^n$, $L \in \mathcal{L}$ and a text t for L such that $\{M(t|_{i+1}, \tau) \neq M(t|_i, \tau) \mid i \in \mathbb{N}\}$ has cardinality greater than $f(n)$.
- (b) There exists a consistent In-classifier M for \mathcal{L} such that for all $n \in \mathbb{N}$, for all $\tau \in \text{In}^n$, for all $L \in \mathcal{L}$ and for all texts t for L , the set $\{M(t|_{i+1}, \tau) \neq M(t|_i, \tau) \mid i \in \mathbb{N}\}$ has cardinality at most $3(f(n) + 2)$.

Proof. Let $(M_e)_{e \in \mathbb{N}}$ be an enumeration of all In-classifiers. Let $e \in \mathbb{N}$ and $n > 0$ be given. Let $\tau_{e,n}$ denote the sequence $(\in_{\langle e, n, 0 \rangle}, \dots, \in_{\langle e, n, n-1 \rangle})$. If there exists $p \in \mathbb{N}$ such that for all $x \in \mathbb{N}$, $M_e(\#^p, \tau_{e,n})$ is a sequence of nothing but 0's, then let $g(e, n)$ denote such a p ; otherwise, let $g(e, n)$ denote 0. Given $e \in \mathbb{N}$, $n > 0$ and $r \in \{n, \dots, n + f(n) + 1\}$, let $L_{e,n}^r$ denote $\{\langle e, n, x \rangle \mid n \leq x \leq r\}$, $\sigma_{e,n}^r$ the concatenation of $\#^{g(e,n)}$ with $(\langle e, n, n \rangle, \dots, \langle e, n, r \rangle)$, and for all $i < n$, $L_{e,n}^{r,i}$ the set $L_{e,n}^r \cup \{\langle e, n, i \rangle\}$.

Let \mathcal{L} consist of \emptyset and, for $e \in \mathbb{N}$, $n > 0$ and $r \in \{n, \dots, n + f(n) + 1\}$, sets of the form $L_{e,n}^r$ or $L_{e,n}^{r,i}$ such that the following holds.

- Suppose that $r \leq n + f(n)$ and $r - n$ is even. Then for all $i < n$, \mathcal{L} contains $L_{e,n}^{r,i}$ iff \mathcal{L} contains $L_{e,n}^s$ for all $s \in \{n, \dots, r-1\}$. Moreover, \mathcal{L} contains $L_{e,n}^r$ iff (i) \mathcal{L} contains $L_{e,n}^{r,i}$ for all $i < n$ and (ii) $M_e(\sigma_{e,n}^r, \tau_{e,n})$ is defined and is a sequence where 1 occurs once and only once.
- Suppose that $r \leq n + f(n)$ and $r - n$ is odd. Then \mathcal{L} contains $L_{e,n}^r$ iff \mathcal{L} contains $L_{e,n}^{s,i}$ for all $s \in \{n, \dots, r-1\}$ and $i < n$. Moreover, for all $i < n$, \mathcal{L} contains $L_{e,n}^{r,i}$ iff (i) \mathcal{L} contains $L_{e,n}^r$ and (ii) $M_e(\sigma_{e,n}^r, \tau_{e,n})$ is defined and is a sequence of nothing but 0's.
- Suppose that $r = n + f(n) + 1$. Then \mathcal{L} contains $L_{e,r}^r$ and $L_{e,r}^{n,i}$ for all $i < n$ iff for all $s \in \{n, \dots, r-1\}$, \mathcal{L} contains $L_{e,n}^s$ and $L_{e,n}^{s,i}$ for all $i < n$.

Let $e \in \mathbb{N}$ be such that M_e is a consistent In-classifier that classifies \mathcal{L} . Let $n > 0$ be given. Since M_e classifies \emptyset , the definition of $g(e, n)$ implies that $M_e(\#^g(e, n), \tau_{e,n})$ is a sequence of nothing but 0's. Moreover, for all members r of $\{n, \dots, n + f(n)\}$, either $r - n$ is even and $M(\sigma_{e,n}^r, \tau_{e,n})$ is defined and is a sequence where 1 occurs once and only once, or $r - n$ is odd and $M(\sigma_{e,n}^r, \tau_{e,n})$ is defined and is a sequence of nothing but 0's. Indeed, suppose otherwise for a contradiction. Let $r \in \{n, \dots, n + f(n)\}$ be least such that the previous condition does not hold. Assume that $r - n$ is even (the case where $r - n$ is odd is similar). According to its definition, \mathcal{L} contains $L_{e,n}^{r,i}$ for all $i < n$, but it contains no $L_{e,n}^s$ with $s \in \{r, \dots, n + f(n) + 1\}$. Since M_e classifies \mathcal{L} and is consistent, we infer that $M_e(\sigma_{e,n}^r, \tau_{e,n})$ is defined and is a sequence where 1 occurs once and only once. This is a contradiction. Hence any text t for $L_{e,n}^{n+f(n)+1}$ having $\sigma_{e,n}^r$ as initial segment for all $r \in \{n, \dots, n + f(n) + 1\}$ is such that the cardinality of the set $\{M_e(t|_{i+1}, \tau_{e,n}) \neq M_e(t|_i, \tau_{e,n}) \mid i \in \mathbb{N}\}$ is greater than $f(n)$. This completes the proof of part (a). We omit the proof of part (b). \square

6 Conclusion

Our aim was to close the gap between classification and learning, using a notion of multiclassification where arbitrarily large finite sets of membership queries have to be dealt with synchronously. Learning implicitly permits to multiclassify the membership of all numbers in the limit and therefore our result that in many cases learnability is more difficult to achieve than classification is not unexpected. We have also shown that multiclassification is interesting in its own right. In particular combining it with conservativeness and various variants of consistency gives a complex and interesting picture. Furthermore, multiclassification permits in many cases severe constant mind change bounds and we explored the interaction between these constant mind change bounds on the one hand and the learnability of the corresponding classes on the other hand.

References

- [Ang80] Angluin, D.: Inductive inference of formal languages from positive data. *Information and Control* 45, 117–135 (1980)
- [Bār74a] Bārzdiņš, J.: Inductive inference of automata, functions and programs. In: Proceedings of the 20th International Congress of Mathematicians, Vancouver, pp. 455–560 (1974) (in Russian); English translation in American Mathematical Society Translations: Series 2, 109, 107–112 (1977)
- [Bār74b] Bārzdiņš, J.: Two theorems on the limiting synthesis of functions. In: Theory of Algorithms and Programs, vol. 1, pp. 82–88. Latvian State University (1974) (in Russian)
- [BB75] Blum, L., Blum, M.: Toward a mathematical theory of inductive inference. *Information and Control* 28(2), 125–155 (1975)
- [CKSS97] Case, J., Kinber, E., Sharma, A., Stephan, F.: On the classification of computable languages. In: Reischuk, R., Morvan, M. (eds.) STACS 1997. LNCS, vol. 1200, pp. 225–236. Springer, Heidelberg (1997)

- [CL82] Case, J., Lynes, C.: Machine inductive inference and language identification. In: Nielsen, M., Schmidt, E.M. (eds.) ICALP 1982. LNCS, vol. 140, pp. 107–115. Springer, Heidelberg (1982)
- [CS83] Case, J., Smith, C.: Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science* 25, 193–220 (1983)
- [FH94] Freivalds, R., Hoffmann, A.: An inductive inference approach to classification. *Journal of Experimental and Theoretical Artificial Intelligence* 6, 63–72 (1994)
- [Gol67] Gold, E.M.: Language identification in the limit. *Information and Control* 10(5), 447–474 (1967)
- [Jai01] Jain, S.: On an open problem in classification of languages. *Journal of Experimental and Theoretical Artificial Intelligence* 13(2), 113–118 (2001)
- [JB81] Jantke, K.P., Beick, H.-R.: Combining postulates of naturalness in inductive inference. *Journal of Information Processing and Cybernetics (EIK)* 17, 465–484 (1981)
- [JMS08] Jain, S., Martin, E., Stephan, F.: Absolute versus probabilistic classification in a logical setting. *Theoretical Computer Science A* 397(1-3), 114–128 (2008); Special Issue on Forty Years of Inductive Inference
- [JORS99] Jain, S., Osherson, D., Royer, J., Sharma, A.: *Systems that Learn: An Introduction to Learning Theory*, 2nd edn. MIT Press, Cambridge (1999)
- [OW82] Osherson, D., Weinstein, S.: Criteria of language learning. *Information and Control* 52, 123–138 (1982)
- [Rog67] Rogers, H.: *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York (1967); Reprinted by MIT Press in 1987
- [Ste01] Stephan, F.: On one-sided versus two-sided classification. *Archive for Mathematical Logic* 40, 489–513 (2001)
- [SWZ97] Smith, C., Wiegagen, R., Zeugmann, T.: Classifying predicates and languages. *International Journal of Foundations of Computer Science* 8, 15–41 (1997)
- [WL76] Wiegagen, R., Liepe, W.: Charakteristische Eigenschaften von erkennbaren Klassen rekursiver Funktionen. *Journal of Information Processing and Cybernetics (EIK)* 12, 421–438 (1976)
- [WS95] Wiegagen, R., Smith, C.H.: Generalization versus classification. *Journal of Experimental and Theoretical Artificial Intelligence* 7, 163–174 (1995)
- [ZZ08] Zeugmann, T., Zilles, S.: Learning recursive functions: A survey. *Theoretical Computer Science A* 397(1-3), 4–56 (2008); Special Issue on Forty Years of Inductive Inference. Dedicated to the 60th Birthday of Rolf Wiegagen

Learning Relational Patterns^{*}

Michael Geilke¹ and Sandra Zilles²

- ¹ Fachbereich Informatik, Technische Universität Kaiserslautern
D-67653 Kaiserslautern, Germany
geilke.michael@gmail.com
- ² Department of Computer Science, University of Regina
Regina, SK, Canada S4S 0A2
zilles@cs.uregina.ca

Abstract. Patterns provide a simple, yet powerful means of describing formal languages. However, for many applications, neither patterns nor their generalized versions of typed patterns are expressive enough. This paper extends the model of (typed) patterns by allowing relations between the variables in a pattern. The resulting formal languages are called *Relational Pattern Languages* (RPLs). We study the problem of learning RPLs from positive data (text) as well as the membership problem for RPLs. These problems are not solvable or not efficiently solvable in general, but we prove positive results for interesting subproblems.

We further introduce a new model of learning from a restricted pool of potential texts. Probabilistic assumptions on the process that generates words from patterns make the appearance of some words in the text more likely than that of other words. We prove that, in our new model, a large subclass of RPLs can be learned with high confidence, by effectively restricting the set of likely candidate patterns to a finite set after processing a single positive example.

1 Introduction

After Angluin [2] introduced the pattern languages, they became a popular object of study in algorithmic learning theory. Patterns are strings consisting of constant and variable symbols; substituting variables by strings of constant symbols generates words in the corresponding pattern languages. Thus patterns provide a simple and intuitive, yet powerful means of describing formal languages.

One focus has been on learning pattern languages in the limit from positive data: a learner is given access to a stream of all and only the words in the target language L and is supposed to generate a sequence of patterns that eventually stabilizes on a pattern generating L [3, 2, 9]. Two central concerns are here

- (a) the apparent trade-off between the expressiveness of pattern languages and the existence of algorithms for learning such languages from positive data [10],
- (b) the lack of efficient algorithms for fundamental tasks involved in many intuitive learning procedures, like solving the membership problem for pattern

* This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

languages [1]. Any learning algorithm that uses membership tests to construct patterns consistent with the known data will suffer from the *NP*-hardness of the membership problem.

The first concern is best illustrated when comparing non-erasing pattern languages [1] to erasing pattern languages [13], the latter ones differing from the former ones only in the detail that they allow to replace variables in a pattern with the empty string. This little additional detail makes patterns more expressive, but at the same time, in general, non-learnable in the limit from positive data [10]. Furthermore, even erasing pattern languages are often not expressive enough to model interesting real-world applications. To this end, Wright [16] and Koshiba [6] introduced an extension of pattern languages, called typed pattern languages. Typed patterns restrict the set of allowed substitutions separately for each variable, so as to model languages in which, *e.g.*, the variable x_3 in the pattern “author: x_1 title: x_2 year: x_3 ” should be replaced only by 4-digit strings, whereas x_1 and x_2 can be replaced by strings containing letters. Unfortunately, little is known about general conditions under which typed pattern languages are learnable. Moreover, for many applications, neither pattern languages nor typed pattern languages are sufficient to model the complex structure in textual data. Below we give examples of bioinformatics applications in which it is obvious that (typed) pattern languages lack the ability to express that the substitutions for two or more distinct variables are dependent on each other.

This paper extends the model of (typed) pattern languages by allowing that certain variables in a pattern are in a particular relation with each other. The resulting formal languages are called *Relational Pattern Languages*; both classical pattern languages and typed pattern languages are special cases of relational pattern languages. Moreover, relational pattern languages overcome the limitations observed in terms of expressiveness of (typed) pattern languages.

We study relational pattern languages both with respect to their learnability in the limit from positive data (text) and with respect to the complexity of the membership problem. Our contributions along these lines are as follows:

(1) Considering Gold’s model of learning in the limit from arbitrary texts [3], relational pattern languages can be learned as long as the set of allowed relations between variables is finite and no variable can be replaced by the empty string. The conditions are essential for learnability.

(2) The membership problem for relational pattern languages is *NP*-complete in general, but we show a number of interesting sub-problems that can be solved efficiently. Most notably, we prove that the membership problem for relational patterns over finitely many polynomial-time decidable relations is solvable in polynomial time if the words for which to test membership are bounded in length. This is not only a very interesting sub-problem from an application point of view, but also not trivial, since we deal with potential empty substitutions.¹

Considering practical applications, Gold’s model of learning in the limit can be criticized: often there is no step in the learning process after which the set

¹ If only non-empty substitutions for variables are allowed, the membership problem restricted to a finite set of words becomes trivial.

of candidate patterns can be reduced to a finite set, thus forcing the learner to make a best guess within an infinite version space—this might make it difficult for a user to decide when to interrupt the infinite learning process. Despite allowing the learning algorithm this limit behaviour, there is no general positive learnability result for the case that empty substitutions are allowed, *cf.* [10]. This is partly due to the fact that a learning algorithm in Gold’s model is required to be successful on any text for the target language. As a step towards a practical model for learning a large class of (erasing) relational pattern languages, we make the following contributions:

(3) We introduce a new model of learning from a restricted pool of potential texts, since in practice not all texts (data streams) are equally likely. We assume that there are probability distributions over the strings that can be substituted for pattern variables, thus making the appearance of some words in the text more likely than that of other words. As we will explain below, our model differs from previous approaches that were based on a similar motivation [5, 12]. We refer to the underlying patterns as *Probabilistic Relational Patterns*.

(4) We prove that in our new model, a large class of (erasing) probabilistic relational pattern languages can be learned with high confidence, by effectively restricting the set of likely candidate patterns to a finite set after processing only a single positive example.

(5) Our model results in a simple but potentially practical method for testing membership correctly with high confidence, for all probabilistic relational patterns using a finite set of recursive relations.

2 Learning (Typed) Pattern Languages

Languages are defined with respect to a non-empty *alphabet* Σ . A *word* w is a finite, possibly empty, sequence of symbols from Σ the length of which is denoted by $|w|$ ². ϵ refers to the empty word, *i.e.*, the word of length 0. The set of all words over Σ is denoted by Σ^* , and the set of all non-empty words over Σ by Σ^+ ; hence $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. A *language* L is a subset of Σ^* . By $w_1 \circ w_2$ we denote the concatenation of two words w_1 and w_2 (where, for ease of presentation, we allow w_1 and/or w_2 to be written as $\sigma \in \Sigma$ rather than a word (σ) of length 1). In what follows, we always assume Σ to be a finite set of cardinality at least 2. We denote the set of all non-zero natural numbers by \mathbb{N}_+ .

In Gold’s model of learning in the limit from positive data [3], a class of languages is learnable if there is a learner that “identifies” every language in the class from any of its texts, where a text for a language L is an infinite sequence $\tau(0), \tau(1), \tau(2), \dots$ of words such that $\{\tau(i) \mid i \in \mathbb{N}\} = L$.

Definition 1 (Gold [3]). Let \mathcal{L} be a class of languages. \mathcal{L} is learnable in the limit from positive data if there is a hypothesis space $\{L_i \mid i \in \mathbb{N}\} \supseteq \mathcal{L}$ and a partial recursive mapping S such that, for any $L \in \mathcal{L}$ and any text $(\tau(i))_{i \in \mathbb{N}}$ for

² “Word” refers to a finite sequence of symbols exclusively from Σ , whereas “string” refers to any other sequence of symbols.

$L, S(\tau(0), \dots, \tau(n))$ is defined for all $n \in \mathbb{N}$ and there is a $j \in \mathbb{N}$ with $L_j = L$ and $S(\tau(0), \dots, \tau(n)) = j$ for all but finitely many n .

A class of languages that has been studied in the formal language theory community as well as in the learning theory community is Angluin's class of non-erasing pattern languages [1], defined as follows. Let $X = \{x_1, x_2, \dots\}$ be a countable set of symbols called variables; we require that X be disjoint from Σ . A *pattern* is a non-empty finite string over $\Sigma \cup X$. The set of all patterns over $\Sigma \cup X$ will be denoted by Pat_{Σ} . A *substitution* is a string homomorphism $\theta : Pat_{\Sigma} \rightarrow \Sigma^*$ that is the identity when restricted to Σ . The set of all substitutions with respect to Σ is denoted by Θ_{Σ} . The non-erasing language $L_{NE}(p)$ of a pattern $p \in (\Sigma \cup X)^+$ is defined by $L_{NE}(p) = \{w \in \Sigma^* \mid \exists \theta \in \Theta_{\Sigma} [\theta(p) = w \wedge \forall x \in X [\theta(x) \neq \epsilon]]\}$, i.e., it consists of all words that result from substituting all variables in p by non-empty words over Σ^* . The set $\mathcal{L}_{\Sigma, NE}$ of non-erasing pattern languages is given by $\mathcal{L}_{\Sigma, NE} = \{L_{NE}(p) \mid p \in Pat_{\Sigma}\}$. Angluin showed that $\mathcal{L}_{\Sigma, NE}$ is learnable in the limit from positive data [1].

Shinohara extended pattern languages by permitting the substitution of variables by ϵ [13]. We denote the erasing pattern language of a pattern p by $L_E(p)$, where $L_E(p) = \{w \in \Sigma^* \mid \exists \theta \in \Theta_{\Sigma} [\theta(p) = w]\}$, and refer to the class of erasing pattern languages by $\mathcal{L}_{\Sigma, E}$. For $|\Sigma| \in \{2, 3, 4\}$, $\mathcal{L}_{\Sigma, E}$ is not learnable in the limit from positive data [10]. Wright [16] studied a subclass of erasing pattern languages under restricted substitutions, leading to Koshiba's typed pattern languages [6]. In Koshiba's model, each variable x in a pattern p is assigned a particular *type* $T(x) = t$, where $t \subseteq \Sigma^*$ is recursive. (p, T) is then called a typed pattern. The words generated by (p, T) are formed by substituting any variable of type t in p only with words from t , resulting in the typed pattern language $L(p, T)$.³

Types make pattern languages more expressive and more suitable for applications. For example, a system for entering bibliographic data as described by Shinohara [13] might use patterns like $p = \text{author: } x_1 \text{ title: } x_2 \text{ year: } x_3$. One would expect x_3 to be substituted only by certain two or four digit integers—a property that becomes expressible when using types.

In fact, every recursive language L can trivially be written as a typed pattern language, generated by the pattern $p = x_1$ where the type of x_1 is L . Thus, a typed pattern is not always a useful description of a language, from an application point of view. Ideally, one would like to keep the types themselves simple, to make the pattern understandable by humans.

Consider for example patterns describing RNA sequences formed out of bases A, C, G, U . The secondary structure of molecules contains information about bonds between base pairs in the sequence; C can bond with G , A with U . For example, Fig. 1 shows potential intramolecular bonds for the sequence $CUUU$

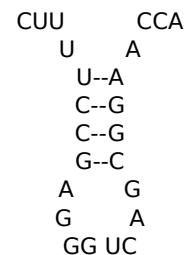


Fig. 1. RNA sequence with bonds

³ Actually, Koshiba did not allow substituting variables by the empty string, but we relax this condition here. We also deviate slightly from his notation.

UCCG AGGG UCAG CGGA ACCA. Obviously, the bonding partners towards one end of the sequence must appear in reverse order of the corresponding bases towards the opposite end of the sequence. To express this with a typed pattern would require the whole subsequence *UCCG AGGG UCAG CGGA* to be the substitution for a single variable and thus an element of a complex type.

A formally simpler example is the language $L_1 = \{a^n b^n \mid n \geq 1\}$, which is context-free but not regular. To express L_1 as a typed pattern language requires “complex” types; regular types are not sufficient.

Proposition 2. *Let (p, T) be a typed pattern. If $L(p, T) = L_1$ then there is some $x \in X$ occurring in p such that $T(x)$ is not a regular language.*

Proof. Let (p, T) be a typed pattern generating L_1 , and k be the number of variables that occur more than once in p . Assume $T(x)$ was regular for all $x \in X$ occurring in p . We deduce a contradiction by induction on k .

For $k = 1$, $L(p, T)$ would be the concatenation of regular languages and as such regular—a contradiction.

For $k \geq 2$, let x be a variable that occurs l times in p with $l \geq 2$. W.l.o.g., we may assume that $L(x, T) \neq \{\epsilon\}$. Since $L(p, T) = L_1$ and x occurs multiple times, one obtains $L(x, T) \subseteq \{a^i \mid i \in \mathbb{N}_+\}$ or $L(x, T) \subseteq \{b^i \mid i \in \mathbb{N}_+\}$. For $L(x, T) \subseteq \{a^i \mid i \in \mathbb{N}_+\}$, let y be a variable not occurring in p , $T'(y) = \{a^{i-l} \mid a^i \in L(x, T)\}$, and $T'(z) = T(z)$ for all variables z occurring in p . Then $L(p', T') = L(p, T) = L_1$, where p' is the pattern that results when removing all occurrences of x from p and adding y as a prefix. The inductive hypothesis yields the required contradiction. The case $L(x, T) \subseteq \{b^i \mid i \in \mathbb{N}_+\}$ is analogous. \square

3 Relational Patterns

In order to model interdependencies between the substitutions of variables, we introduce relations between variables into the definition of patterns.

Definition 3. *Let R be a set of relations over Σ^* . Then, for any $n \in \mathbb{N}_+$, R_n denotes the set of n -ary relations in R . A relational pattern with respect to Σ and R is a pair (p, v_R) where p is a pattern over Σ and $v_R \subseteq \{(r, y_1, \dots, y_n) \mid n \in \mathbb{N}_+, r \in R_n, \text{ and } y_1, \dots, y_n \text{ are variables in } p\}$. The set of relational patterns with respect to R will be denoted by $\text{Pat}_{\Sigma, R}$.*

The set of all possible substitutions for (p, v_R) is denoted by $\Theta_{(p, v_R), \Sigma}$. It contains all substitutions $\theta \in \Theta_\Sigma$ that fulfill, for all $n \in \mathbb{N}_+$:

$$\forall r \in R_n \ \forall y_1, \dots, y_n \in X \ [(r, y_1, \dots, y_n) \in v_R \Rightarrow (\theta(y_1), \dots, \theta(y_n)) \in r].$$

The language of (p, v_R) , denoted by $L(p, v_R)$, is defined as $\{w \in \Sigma^ \mid \exists \theta \in \Theta_{(p, v_R), \Sigma} : \theta(p) = w\}$. The set of all languages of relational patterns with respect to R will be denoted by $\mathcal{L}_{\Sigma, R}$.*

For instance, $r = \{(w_1, w_2) \mid w_1, w_2 \in \Sigma^* \wedge |w_1| = |w_2|\}$ is a binary relation, which, applied to two variables x_1 and x_2 in a relational pattern (p, v_R) , ensures

that the substitutions of x_1 and x_2 generating words from p always have the same length. Formally, this is done by including (r, x_1, x_2) in v_R .

We assume, without loss of generality, that for every variable x occurring in a relational pattern (p, v_R) , there is exactly one $r \in R_1$ such that $(r, x) \in v_R$. In fact, this unary relation r represents the type of variable x . If there is no $r \in R_1$ with $(r, x) \in v_R$, we can include (r_*, x) in (p, v_R) , where $w \in r_* \leftrightarrow w \in \Sigma^*$. If R_1 contains several r_i (for i in some index set I) with $(r_i, x) \in v_R$, we can replace them by the single relation $(r_{\cap I}, x)$ where $w \in r_{\cap I} \leftrightarrow \forall i \in I [w \in r_i]$. We will use the terms “type” and “unary relation” interchangeably. Similarly, without loss of generality, each set of n variables is included in v_R with at most one n -ary relation. We further assume that relational patterns do not contain any variable twice. This is no restriction, since repetition of variables can be expressed by an equality relation between two distinct variables.

We are only going to consider the case that R is finite, which seems sufficient for many practical applications. It is easy to see that $\mathcal{L}_{\Sigma, NE}$ and $\mathcal{L}_{\Sigma, E}$, as well as the class of typed pattern languages over finitely many types, are subclasses of $\mathcal{L}_{\Sigma, R}$, for respective suitably defined finite sets R .

The gain in expressiveness shows for example in L_1 , which, by Proposition 2, cannot be expressed as a typed pattern language using only regular type languages. Using relational patterns, regular types are sufficient to describe L_1 .

Proposition 4. *There is a finite set R of relations such that R_1 contains only regular languages and $L_1 \in \mathcal{L}_{\Sigma, R}$.*

Proof. If $R = \{r_1, r_2, r\}$, $r_1 = \{a^i \mid i \geq 1\}$, $r_2 = \{b^i \mid i \geq 1\}$, $r = \{(w_1, w_2) \mid |w_1| = |w_2|\}$, and $v_R = \{(r_1, x_1), (r_2, x_2), (r, x_1, x_2)\}$ then $L(x_1 x_2, v_R) = L_1$. \square

Since erasing pattern languages can be expressed as relational pattern languages, Reidenbach’s non-learnability results for erasing pattern languages [10] immediately yield the following theorem.

Theorem 5. *There is a finite alphabet Σ and finite set R of recursive relations such that $\mathcal{L}_{\Sigma, R}$ is not learnable in the limit from positive data.*

However, if we disallow empty substitutions, we get positive learnability results for any set of recursive relations.

Theorem 6. *Let R be a finite set of recursive relations with $r \subseteq \Sigma^+$ for all $r \in R_1$. Then $\mathcal{L}_{\Sigma, R}$ is learnable in the limit from positive data.*

To prove this, we use a well-known result due to Angluin [2], according to which every *indexable class* of languages that has *finite thickness* is learnable in the limit from positive data. A class \mathcal{L} of languages is indexable if there is an enumeration $(L_i)_{i \in \mathbb{N}}$ with $\mathcal{L} = \{L_i \mid i \in \mathbb{N}\}$ and an effective procedure d that decides, given any $i \in \mathbb{N}$ and $w \in \Sigma^*$, whether or not $w \in L_i$ [2]. $(L_i)_{i \in \mathbb{N}}$ is then called an indexing for \mathcal{L} . An indexable class \mathcal{L} has finite thickness if, for every $w \in \Sigma^*$, the set of languages in \mathcal{L} that contain w is finite. One can establish both indexability and finite thickness to prove Theorem 6. The proofs

of both these properties use standard techniques and are omitted because of space constraints. It should be noted though that our results show in particular, that Theorem 6 can be witnessed by a learner that returns relational patterns as its hypotheses—a desirable feature from an application point of view, since relational patterns provide an intuitive representation of a language.

Lemma 7. *Let R be a finite set of recursive relations with $r \subseteq \Sigma^+$ for all $r \in R_1$. There exists an effective enumeration $f : \mathbb{N} \rightarrow \text{Pat}_{\Sigma, R}$ of all relational patterns over R such that $(L(f(i)))_{i \in \mathbb{N}}$ is an indexing for $\mathcal{L}_{\Sigma, R}$.*

Lemma 8. *Let R be a finite set of recursive relations with $r \subseteq \Sigma^+$ for all $r \in R_1$. Then $\mathcal{L}_{\Sigma, R}$ has finite thickness.*

In fact, Lemma 8 can be strengthened. An indexable class \mathcal{L} is said to have recursive finite thickness [8] if there is an indexing $(L_i)_{i \in \mathbb{N}}$ for \mathcal{L} and a recursive procedure c such that, for any $w \in \Sigma^*$, $c(w)$ is a finite subset of \mathbb{N} fulfilling $[w \in L_i \leftrightarrow \exists j \in c(w) [L_j = L_i]]$, i.e., for every word w a finite list of indices for all languages in \mathcal{L} containing w can be effectively determined.

Theorem 9. *Let R be a finite set of recursive relations with $r \subseteq \Sigma^+$ for all $r \in R_1$. Then $\mathcal{L}_{\Sigma, R}$ has recursive finite thickness.*

The proof is omitted due to space constraints. Theorem 9 has some nice consequences, which follow immediately from the literature on recursive finite thickness [8, 7]. For the following corollary, note that an iterative learner [14] is restricted to learn without access to prior data at any point in time. Its hypothesis on a text segment $(\tau(0), \dots, \tau(n))$ is determined only by $\tau(n)$ and its hypothesis generated on $(\tau(0), \dots, \tau(n-1))$ (or a dummy hypothesis in case $n=0$).

Corollary 10. *Let R be a finite set of recursive relations with $r \subseteq \Sigma^+$ for all $r \in R_1$. Let $k \in \mathbb{N}$. Then the class of all unions of up to k languages from $\mathcal{L}_{\Sigma, R}$ is learnable in the limit from positive data using an iterative learner.*

4 The Membership Problem

Many algorithms for learning formal languages make a hypothesis only if the corresponding language is proven to be consistent with the observed data. This typically requires solving several instances of the *membership problem*. If $\mathcal{P} \subseteq \text{Pat}_{\Sigma}$ is a set of patterns and $W \subseteq \Sigma^*$ a set of words, then the erasing (non-erasing) membership problem for (\mathcal{P}, W) is decidable if there is an effective procedure that, given any $p \in \mathcal{P}$ and any $w \in W$, decides whether or not $w \in L_{NE}(p)$ ($w \in L_E(p)$, resp.). Similarly, we can define the membership problem for relational pattern languages.

The fact that both the non-erasing and the erasing membership problem for $(\text{Pat}_{\Sigma}, \Sigma^*)$ are NP -complete [1, 4] is an obstacle for the design of efficient learning algorithms for pattern languages. In this section, we study the complexity of subproblems of the membership problem for relational pattern languages.

The following consequence of Angluin's result is straightforward.

Theorem 11. *Let R be a finite set of recursive relations. Then the membership problem for $(\text{Pat}_{\Sigma, R}, \Sigma^*)$ is NP-hard.*

However, when the number of variables occurring in a relational pattern is bounded a priori, we get a positive result, which generalizes Angluin's result that the non-erasing membership problem is polynomial-time decidable if the patterns contain at most k variables, for some k [1].

Theorem 12. *Let R be a set of finite relations, each of which is decidable in polynomial time. Let $k \in \mathbb{N}$. Then the membership problem for $(\{(p, v_R) \in \text{Pat}_{\Sigma, R} \mid p \text{ contains at most } k \text{ distinct variables}\}, \Sigma^*)$ is decidable in polynomial time.*

Proof. Given a relational pattern (p, v_R) over R and a word w , the following procedure decides whether $w \in L(p, v_R)$ in time polynomial in $|p|$, $|v_R|$, and $|w|$.

1. Let $z \leq k$ be the number of distinct variables in p . List all tuples (w_1, \dots, w_z) of up to z many substrings of w , for which $w_1 \circ \dots \circ w_z$ is a subsequence of w . (Note: as k is constant, the number of such tuples is polynomial in $|w|$.)
2. For each tuple (w_1, \dots, w_z) thus listed, define a substitution θ by substituting the z variables in p in order with the words w_1, \dots, w_z ; then test whether (i) $\theta \in \Theta_{(p, v_R), \Sigma}$ and (ii) $\theta(p) = w$. (Note: these tests can be done in polynomial time, since all relations in R can be decided in polynomial time.)
3. If there is one tuple (w_1, \dots, w_z) for which the test on both (i) and (ii) is positive, return *yes*, otherwise return *no*.

The correctness and efficiency of the procedure follow immediately. \square

In contrast to this, in the context of relational patterns, it is impossible to get an equivalent of Shinohara's [13] result stating that the (non-)erasing membership problem is polynomial-time decidable when restricted to the class of all patterns in which no variable occurs twice, so called *regular patterns*. Since relational patterns can always be expressed equivalently without using any variable twice, Theorem 11 yields NP-hardness of the membership problem for relational patterns with recursive relations and without repetition of variables.

For erasing regular pattern languages, Shinohara's result can be extended:

Theorem 13. *Let $k \in \mathbb{N}$. Then the erasing membership problem for $(\{p \in \text{Pat}_{\Sigma} \mid \text{there are at most } k \text{ variables that occur multiple times in } p\}, \Sigma^*)$ is decidable in polynomial time.*

Proof. For a given $p \in \text{Pat}_{\Sigma}$ with at most k repeated variables and for $w \in \Sigma^*$, the number of ways in which only the repeated variables in p can be replaced by subwords of w is (loosely) upper-bounded by $\binom{|w|}{2k}$ (for each of the up to k repeated variables, one fixes a start position and an end position of the first location in w in which the variable could be substituted), which is polynomial in $|w|$. Replacing only the repeated variables by words in this way maps p to a regular

pattern whose length is polynomial in $|w| + |p|$. Obviously, $w \in L_E(p)$ iff w is generated by one of these regular patterns. Since, according to Shinohara [13], the erasing membership problem for regular patterns is polynomial-time decidable, it follows that $w \in L_E(p)$ can be decided in polynomial time as well. \square

To our knowledge, the literature has so far not dealt with the question of the complexity of the membership problem when the class of patterns is not severely restricted, yet the set of words is. Since many real-world applications deal with an a priori restricted set of words, it seems reasonable to focus our attention on such problems. For example, in bioinformatics applications, one often has an upper bound on the length of RNA sequences or amino acid sequences that will occur in a database, due to restrictions on either molecular size or the length of snippets collected in experiments. We hence focus on the membership problem for $(\mathcal{P}, \Sigma^{\leq m})$ for $m \in \mathbb{N}$ and for large classes \mathcal{P} of (relational) patterns. Here $\Sigma^{\leq m}$ denotes the set of words of length at most m over Σ .

For classical patterns, the non-erasing membership problem for $(\text{Pat}_\Sigma, \Sigma^{\leq m})$ clearly is polynomial-time decidable, since the length of a pattern generating a word w is upper-bounded by $|w|$. However, for erasing pattern languages and for the general case of relational pattern languages, a similar statement does not follow that obviously. The following main result of this section states that, for words of length at most m , the membership problem for a very general class of relational patterns is polynomial-time decidable. Note that this does not provide practical solutions in general, since the length bound m , which occurs in the exponent in our complexity bound, might be rather large in practice.

Theorem 14. *Let R be a set of polynomial-time decidable relations and $m \in \mathbb{N}$. Then the membership problem for $(\text{Pat}_{\Sigma, R}, \Sigma^{\leq m})$ is decidable in polynomial time.*

Proof. Let $(p, v_R) \in \text{Pat}_{\Sigma, R}$ and $w \in \Sigma^{\leq m}$. Let R' be the set of relations resulting from R when every unary relation t is replaced by $t \setminus \{\epsilon\}$, i.e., $R' = (R \setminus R_1) \cup \{(t \setminus \{\epsilon\}) \mid t \in R_1\}$.

We say that $((p, v_R), w)$ fulfills Property $(*)$ if there is a relational pattern $(q, v_{R'}) \in Q_{(p, v_R)}$ with $|q| \leq |w|$, and a substitution $\theta_q \in \Theta_{(q, v_{R'}), \Sigma}$, such that $\theta_q(q) = w$ and $\theta \in \Theta_{(p, v_R), \Sigma}$, where θ restricted to the variables in q equals θ_q and $\theta(x) = \epsilon$ for all other variables. Here $Q_{(p, v_R)}$ is the set of all relational patterns $(q, v_{R'}) \in \text{Pat}_{\Sigma, R'}$ where

1. q results from p by removing arbitrarily many variables from p
2. $v_{R'} = \{(r, y_1, \dots, y_n) \in v_R \mid y_i \text{ occurs in } q \text{ for all } 1 \leq i \leq n, n \geq 2\} \cup \{(t \setminus \{\epsilon\}, x) \mid (t, x) \in v_R \text{ and } x \text{ occurs in } q\}$.

First, we claim that $w \in L(p, v_R)$ iff $((p, v_R), w)$ fulfills Property $(*)$: If $w \in L(p, v_R)$, then there is a substitution $\theta_p \in \Theta_{(p, v_R), \Sigma}$ such that $\theta_p(p) = w$. Let q be the pattern resulting from p after deletion of all variables x with $\theta(x) = \epsilon$. Obviously, $|q| \leq |w|$ and there is a $\theta_q \in \Theta_{(q, v_{R'}), \Sigma}$ with $\theta_q(q) = w$, where $v_{R'}$ is defined as in Property $(*)$.2. Clearly, $(q, v_{R'}) \in Q_{(p, v_R)}$. It remains to show that

$\theta \in \Theta_{(p, v_R), \Sigma}$, where θ restricted to the variables in q equals θ_q and $\theta(x) = \epsilon$ for all other variables. This follows from $\theta = \theta_p$. Hence $((p, v_R), w)$ fulfills Property (*). If $((p, v_R), w)$ fulfills Property (*), it follows just as easily that $w \in L(p, v_R)$.

Second, we show that Property (*) can be tested in polynomial time in $|p|$ and $|v_R|$: To construct a list of all $(q, v_{R'}) \in Q_{(p, v_R)}$ with $|q| \leq |w|$, it suffices to consider all sets S of at most $|w|$ many distinct variables occurring in p and to list, for each such S , the relational pattern $(\theta'(p), v_{R'})$ with $v_{R'}$ as in Property (*).2, where

$$\theta'(x) := \begin{cases} x, & \text{if } x \in S \cup \Sigma, \\ \epsilon, & \text{otherwise.} \end{cases}$$

With $|S| \leq |w| \leq m$, it follows that at most

$$\sum_{i=0}^{|w|} \binom{|p|}{i} \leq \sum_{i=0}^{|w|} |p|^i \leq (m+1) \cdot |p|^m = O(|p|^m)$$

many relational patterns $(q, v_{R'})$ are listed. Theorem I2 implies that, for these listed relational patterns $(q, v_{R'})$, membership of w in $L(q, v_{R'})$ can be tested in time polynomial in m . If all these membership tests are negative, then Property (*) is not fulfilled. Each positive membership test yields a substitution $\theta_q \in \Theta_{(q, v_{R'})}$ with $\theta_q(q) = w$. One then tests whether $\theta \in \Theta_{(p, v_R), \Sigma}$, where θ is defined as in Property (*); each of these tests can be done in $O(|v_R| \cdot |p|)$. Property (*) is fulfilled if and only if one of these tests is positive. With m being fixed, the total run-time is polynomial in $|v_R|$ and $|p|$. \square

5 Probabilistic Relational Patterns

We have identified two problems that are impossible or hard to solve for complex classes of relational patterns: (i) learning such patterns from text and (ii) the membership problem for such patterns for all words $w \in \Sigma^*$. There is reason to assume that real-world instantiations of these problems can be solved more easily. To model realistic scenarios more closely, we assume that certain words in a (relational) pattern language are more likely than others, and thus introduce the class of *probabilistic relational patterns*. Our approach addresses the two issues above in the following way.

Learning from text. In applications, not all words are equally likely to occur in a text for a target language. Hence it seems unnecessary to demand from a learner to be successful on all texts of a target language. In this section, we modify Gold's success criterion by considering probability distributions over types in the pattern language and demanding learnability only from texts that are "sufficiently likely." Studying the learnability of *probabilistic* relational pattern languages in our new model, we obtain upper bounds on the number of relational patterns that have to be considered to identify the target language.

The membership problem. As a side-effect of our model of probabilistic relational patterns, we obtain a simple and potentially practical mechanism for testing membership of words of arbitrary length.

We choose to model probabilistic relational pattern languages via probability distributions on *types*, *i.e.*, for any type r and any subset $A \subseteq r$, a variable of type r is substituted by a word from subset A with a certain probability. We have to make sure though that the collection of words substituted for distinct variables does not violate any of the higher-order relations in the pattern.

Definition 15. Let R be a set of relations. A probabilistic relational pattern over R is a triple (p, v_R, pr_{R_1}) , where (p, v_R) is a relational pattern over R and, for each $r \in R_1$, pr_{R_1} contains exactly one probability distribution on r . For $r \in R_1$ and $A \subseteq r$, the probability of A with respect to r is then denoted by $pr_r(A)$. A probabilistic text for (p, v_R, pr_{R_1}) , if existent, is an infinite sequence of words in $L(p, v_R)$, each of which is formed by the following stochastic process:

1. For each variable x in p , for $(r, x) \in v_R$, draw a substitution $\theta(x) \in r$ according to pr_r .
2. If, for all $n \in \mathbb{N}_+$, all $r' \in R_n$, and all $(r', x_{i_1}, \dots, x_{i_n}) \in v_R$, the substitutions drawn fulfill $(\theta(x_{i_1}), \dots, \theta(x_{i_n})) \in r'$, then return the word resulting from the substitutions drawn in step 1. Otherwise, repeat from step 1.

Unlike Gold's definition, we define the success criterion with respect to a class of patterns rather than pattern languages. This is necessary since two relational patterns could describe the same language L but differ in the probability distribution defined on L .

Definition 16. Let \mathcal{P} be a class of probabilistic relational patterns and $\delta \in (0, 1)$. \mathcal{P} is δ -learnable in the limit from likely texts if there is a partial recursive mapping S such that, for any $(p, v_R, pr_{R_1}) \in \mathcal{P}$, if $(\tau(i))_{i \in \mathbb{N}}$ is any probabilistic text for (p, v_R, pr_{R_1}) , then, $S(\tau(0), \dots, \tau(n))$ is defined for all n and, with probability at least $1 - \delta$, there is a relational pattern (q, v'_R) such that $L(q, v'_R) = L(p, v_R)$ and $S(\tau(0), \dots, \tau(n)) = q$ for all but finitely many n .

The parameter δ determines a confidence level with which the learner S is supposed to identify a pattern from a probabilistic text, if this text is generated according to Definition 15. Learning is still a limit process, but success is only required on a portion of all possible texts.

Our model has some similarities with stochastic finite learning [T1, T2]. In the latter model though, learning is a finite process and the data stream is generated from a probability distributions over the learning domain (which is in our case Σ^*) rather than over types. Kearns and Pitt [5] studied PAC learning of pattern languages generated by patterns with a bounded number of variables, restricting the class of possible distribution of words by considering only a particular kind of substitution. The way they restrict substitutions differs from our model, and their learner expects both positive and negative examples to learn from.

Prior research on (efficient) learning of subclasses of erasing pattern languages mostly studied patterns in which the number of variables is a priori upper-bounded [T1, T2, T5]. However, in many applications, the number of variables in target patterns is huge, but the arity of relations is limited. For instance, in

our simplified bioinformatics model, the maximal arity of relations in relational patterns that describe the two-dimensional structure of RNA sequences is 2. Therefore, we study relational patterns in which the arity of relations is bounded and the number of distinct variables is unbounded.

Definition 17. Let $k \in \mathbb{N}$, R a finite set of recursive relations, and (p, v_R) a relational pattern. (p, v_R) is called k -simple, if

1. $R_i = \emptyset$ for all $i > k$,
2. for every $x \in X$, $|\{(r, y_1, \dots, y_{i-1}, x, y_{i+1}, \dots, y_n) \in v_R \mid n \geq 2, 1 \leq i \leq n\}| \leq 1$, and
3. if $r \in R$, $(w_1, \dots, w_i) \in r$, and $w_j = \epsilon$ for some $j \in \{1, \dots, i\}$ then $w_1 = \dots = w_i = \epsilon$.

A k -simple relational pattern contains at most 1 multi-variable relation per variable, with an arity of at most k . These relations, as well as all types in a k -simple pattern, are recursive. Erasing pattern languages generated by patterns with a bounded number of variable repetitions can also be generated by k -simple relational patterns.⁴ In addition, relational patterns generating RNA sequences as shown in Figure 11 are 2-simple, if each base participates in at most one bond.

Definition 18 requires additional notation. For a relational pattern (p, v_R) , any n -ary relation r , and any $i \in \{1, \dots, n\}$, $r[i]$ denotes the set $\{w \in \Sigma^* \mid \exists w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_n \in \Sigma^* [(w_1, \dots, w_{i-1}, w, w_{i+1}, \dots, w_n) \in r]\}$ and

$$\text{Allowed}(x) = \bigcap_{r(y_1, \dots, y_{i-1}, x, y_{i+1}, \dots, y_n) \in v_R} r[i].$$

Intuitively, $\text{Allowed}(x)$ is the set of all words that can potentially be substituted for x in the relational pattern (p, v_R) .

Definition 18. Let $\pi \in (0, 1)$, $k \in \mathbb{N}$, R a finite set of recursive relations, and (p, v_R, pr_{R_1}) a probabilistic relational pattern. (p, v_R, pr_{R_1}) is called (k, π) -good if (p, v_R) is k -simple and, for all $t \in R_1$ and all x with $(t, x) \in v_R$

1. $pr_t(\text{Allowed}(x)) > 0$,
2. there is a set $A \subseteq t \setminus \{\epsilon\}$ with $\frac{pr_t(\text{Allowed}(x) \cap A)}{pr_t(\text{Allowed}(x))} > 1 - \pi$.

The main result of this section shows that (k, π) -good probabilistic relational patterns are learnable with high confidence, by cutting down to a finite set of likely candidate patterns after processing only a single positive example.

Theorem 19. Let $\delta, \pi \in (0, 1)$, $k \in \mathbb{N}$, R a finite set of recursive relations, and \mathcal{P} any class of (k, π) -good probabilistic relational patterns with respect to R . Then \mathcal{P} is δ -learnable in the limit from likely texts.

⁴ To our knowledge, learnability of this class has not been studied in the literature before—our positive learnability result presented below in Corollary 21 immediately yields learnability of such erasing pattern languages from likely texts.

Proof. Let $(p, v_R, pr_{R_1}) \in \mathcal{P}$ be arbitrary and v be the number of *independent variables* in (p, v_R) , i.e., variables x such that for all $n \in \mathbb{N}$ and for all $r \in R_n : (r, y_1, \dots, y_{i-1}, x, y_{i+1}, \dots, y_n) \in v_R$ implies $i = 1$. Let τ be an arbitrary probabilistic text for $L(p, v_R)$. Since (p, v_R, pr_{R_1}) is (k, π) -good, for all $t \in R_1$ and for all variables x of type t in p , there is a set $A \subseteq t \setminus \{\epsilon\}$ fulfilling Properties 1 and 2 of Definition 18. $m_1 = |\tau(0)|$ is an upper bound for the number of independent variables in (p, v_R) that have been substituted by $w \in A$ in $\tau(0)$. Since $pr_t(\text{Allowed}(x) \cap A)/pr_t(\text{Allowed}(x)) > 1 - \pi$, each of the v many independent variables in p are substituted with probability $1 - \pi$ by a word from A .

Using Chernoff bounds with $\mu = v(1 - \pi)$, it follows that the probability that fewer than $(1 - \lambda_1)\mu = (1 - \lambda_1)v(1 - \pi)$ independent variables are replaced by a word from A is less than $\exp(-\frac{\mu\lambda_1^2}{2}) = \exp(-\frac{v(1-\pi)\lambda_1^2}{2})$ where $\lambda_1 \in (0, 1)$.

First, we determine an upper bound for v with a confidence of at least $1 - \delta_1$ for some $\delta_1 \in (0, \frac{\delta}{2})$. We want to upper-bound the probability $Pr[m_1 < (1 - \lambda_1)v(1 - \pi)]$ (which is $< \exp(-\frac{v(1-\pi)\lambda_1^2}{2})$) by δ_1 . This can be achieved when $\exp(-\frac{\mu}{2} + m_1 - \frac{m_1^2}{2\mu}) \leq \delta_1$ (details are omitted), which holds when $\mu + \frac{m_1^2}{\mu} \geq 2m_1 - 2 \ln \delta_1$ and in particular when $\mu \geq 2m_1 - 2 \ln \delta_1$. The latter is equivalent to $v \geq \frac{1}{1-\pi}(2 \cdot m_1 - 2 \cdot \ln \delta_1)$. Thus, with confidence level $1 - \delta_1$, $\frac{1}{1-\pi}(2 \cdot m_1 - 2 \cdot \ln \delta_1)$ is an upper bound for the number of independent variables in p .

Second, we compute an upper bound for the number m_2 of independent variables that are substituted by ϵ , with confidence level $1 - \delta_2$ for $\delta_2 = \frac{\delta}{2}$. Using Chernoff bounds with an expected value of $\mu' = v\pi$, it follows that

$$Pr[m_2 > (1 + \lambda_2) \cdot \mu'] = Pr[m_2 > (1 + \lambda_2) \cdot v \cdot \pi] < \left[\frac{\exp(\lambda_2)}{(1 + \lambda_2)^{(1+\lambda_2)}} \right]^{v \cdot \pi}.$$

If $\lambda_2 = \min\{\lambda > 0 \mid \left[\frac{\exp(\lambda)}{(1+\lambda)^{(1+\lambda)}} \right]^{v\pi} \leq \delta_2\}$, then $(1 + \lambda_2)v\pi + 1 \geq m_2$ with confidence level $1 - \delta_2$. Since (p, v_R) is k -simple, a variable can be substituted by ϵ only if it is in relation with a variable that is substituted by ϵ . As $R_i = \emptyset$ for $i > k$ and because of Property 2 in Definition 17, with confidence level $1 - \delta_2$, the number of variables in p that are substituted by ϵ is at most m_2 .

Thus, with confidence at least $(1 - \delta_1)(1 - \delta_2) > (1 - \delta)$, $m_1 + m_2$ is an upper bound for $|p|$. Since R is finite, the set of possible candidates for the target language (with confidence at least $1 - \delta$) is finite and, therefore, the target pattern can be identified in the limit from positive data with probability $1 - \delta$. \square

For typed pattern languages defined over a finite number of types and with a bounded number of variable repetitions, we obtain a general positive learnability result. They can be represented by a particular kind of k -simple relational pattern, for which we can prove learnability from likely texts using Theorem 19. The proof is omitted because of space constraints. We require that every variable has a non-zero probability of being replaced with a non-empty word.

Theorem 20. Let $k \in \mathbb{N}$ and let \mathcal{L} be the class of all typed pattern languages generated using finitely many types and patterns with at most k repetitions per variable. Then there is a set R of relations such that

1. $\mathcal{L} \subseteq \mathcal{L}_{\Sigma, R}$, and
2. for all $\delta, \pi \in (0, 1)$, the class $\{(p, v_R, pr_{R_1}) \mid (p, v_R) \in \mathcal{L}_{\Sigma, R} \text{ and } \forall t \in R_1 [pr_t(t \setminus \{\epsilon\}) > 1 - \pi]\}$ is δ -learnable from likely texts.

Corollary 21. Let $k \in \mathbb{N}$ and let \mathcal{L} be the class of all erasing pattern languages generated by patterns with at most k repetitions per variable. Then there is a set R of relations such that

1. $\mathcal{L} \subseteq \mathcal{L}_{\Sigma, R}$, and
2. for all $\delta, \pi \in (0, 1)$, the class $\{(p, v_R, pr_{R_1}) \mid (p, v_R) \in \mathcal{L}_{\Sigma, R} \text{ and } \forall t \in R_1 [pr_t(t \setminus \{\epsilon\}) > 1 - \pi]\}$ is δ -learnable from likely texts.

The model of probabilistic relational patterns also yields a straightforward method for testing membership correctly with high confidence.

Observation 22. Let R be a finite set of recursive relations and let pr_{R_1} contain a probability distribution for each $r \in R_1$. Let $\delta, \pi \in (0, 1)$. Let \mathcal{A} be the following algorithm, given a probabilistic relational pattern (p, v_R, pr_{R_1}) and $w \in \Sigma^*$:

1. Let W be the set of words obtained from $2 - 2 \ln(\delta)/\pi$ independent draws from $L(p, v_R)$ as in Definition 15.
2. If $w \in W$, return yes, else return no.

Then, for any (p, v_R) and any $w \in L(p, v_R)$ whose probability of being generated in (p, v_R, pr_{R_1}) by the process in Definition 15 is at least π , \mathcal{A} detects the membership of w in $L(p, v_R)$ with probability at least $1 - \delta$. For any (p, v_R) and any $w \notin L(p, v_R)$, \mathcal{A} declares non-membership of w in $L(p, v_R)$.

The proof is a simple application of Chernoff bounds that implies that, after $2 - 2 \ln(\delta)/\pi$ independent trials, an event with probability at least π occurs with probability at least $1 - \delta$. For example, to detect membership with a confidence of 0.9 for all words whose probability is at least $\pi = 0.00001$, a set W of about 660,518 words would be generated. How efficient such a method would be in practice depends on the efficiency of generating words from relational patterns.

6 Conclusion

We extended the model of (typed) pattern languages by introducing relational pattern languages, whose expressiveness seems more apt for many applications. We studied relational pattern languages both with respect to their learnability from positive data and with respect to the complexity of the membership problem. We identified interesting sub-problems of the membership problem that can be solved efficiently, in particular the sub-problem of bounded word length, which, to our knowledge, has not even been studied for classical patterns yet.

Our probabilistic version of relational patterns, and the corresponding model of learning from likely texts provide an adaptation for more realistic text mining and bioinformatics scenarios than Gold's original model can deal with. In our new model, a large class of (erasing) probabilistic relational pattern languages can be learned with high confidence. After seeing just one positive example, the learner can effectively restrict the set of likely candidate patterns to a finite set. Finally, we proposed a simple yet potentially efficient method for testing membership for probabilistic relational patterns correctly with high confidence.

References

- [1] Angluin, D.: Finding patterns common to a set of strings. *J. Comput. Syst. Sci.* 21, 46–62 (1980)
- [2] Angluin, D.: Inductive inference of formal languages from positive data. *Inform. Control* 45, 117–135 (1980)
- [3] Gold, E.M.: Language identification in the limit. *Inform. Control* 10, 447–474 (1967)
- [4] Jiang, T., Kinber, E., Salomaa, A., Salomaa, K., Yu, S.: Pattern languages with and without erasing. *Int. J. Comput. Math.* 50, 147–163 (1994)
- [5] Kearns, M., Pitt, L.: A polynomial-time algorithm for learning k -variable pattern languages from examples. In: COLT, pp. 57–71 (1989)
- [6] Koshiba, T.: Typed pattern languages and their learnability. In: Vitányi, P.M.B. (ed.) EuroCOLT 1995. LNCS, vol. 904, pp. 367–379. Springer, Heidelberg (1995)
- [7] Lange, S.: Algorithmic learning of recursive languages. Habilitationsschrift, University of Leipzig (2000)
- [8] Lange, S., Zeugmann, T.: Incremental learning from positive data. *J. Comput. Syst. Sci.* 53, 88–103 (1996)
- [9] Lange, S., Zeugmann, T., Zilles, S.: Learning indexed families of recursive languages from positive data: A survey. *Theor. Comput. Sci.* 397, 194–232 (2008)
- [10] Reidenbach, D.: Discontinuities in pattern inference. *Theor. Comput. Sci.* 397, 166–193 (2008)
- [11] Reischuk, R., Zeugmann, T.: An average-case optimal one-variable pattern language learner. *J. Comput. Syst. Sci.* 60, 302–335 (2000)
- [12] Rossmanith, P., Zeugmann, T.: Stochastic finite learning of the pattern languages. *Mach. Learn.* 44, 67–91 (2001)
- [13] Shinohara, T.: Polynomial time inference of extended regular pattern languages. In: Goto, E., Furukawa, K., Nakajima, R., Nakata, I., Yonezawa, A. (eds.) RIMS 1982. LNCS, vol. 147, pp. 115–127. Springer, Heidelberg (1983)
- [14] Wiehagen, R.: Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. *Elektronische Informationsverarbeitung und Kybernetik* 12, 93–99 (1976)
- [15] Wright, K.: Identification of unions of languages drawn from an identifiable class. In: COLT, pp. 328–333 (1989)
- [16] Wright, K.: Inductive identification of pattern languages with restricted substitutions. In: COLT, pp. 111–121 (1990)

Adaptive and Optimal Online Linear Regression on ℓ^1 -Balls^{*}

Sébastien Gerchinovitz¹ and Jia Yuan Yu^{1,2}

¹ École Normale Supérieure, Paris, France

² HEC Paris, CNRS, Jouy-en-Josas, France

Abstract. We consider the problem of online linear regression on individual sequences. The goal in this paper is for the forecaster to output sequential predictions which are, after T time rounds, almost as good as the ones output by the best linear predictor in a given ℓ^1 -ball in \mathbb{R}^d . We consider both the cases where the dimension d is small and large relative to the time horizon T . We first present regret bounds with optimal dependencies on the sizes U , X and Y of the ℓ^1 -ball, the input data and the observations. The minimax regret is shown to exhibit a regime transition around the point $d = \sqrt{TUX}/(2Y)$. Furthermore, we present efficient algorithms that are adaptive, i.e., that do not require the knowledge of U , X , Y , and T , but still achieve nearly optimal regret bounds.

1 Introduction

In this paper, we consider the problem of online linear regression against arbitrary sequences of input data and observations, with the objective of being competitive with respect to the best linear predictor in an ℓ^1 -ball of arbitrary radius. This extends the task of convex aggregation. We consider both low- and high-dimensional input data. Indeed, in a large number of contemporary problems, the available data can be high-dimensional—the dimension of each data point is larger than the number of data points. Examples include analysis of DNA sequences, prediction with sparse data (e.g., Netflix problem), times series of seismic activity. In such high-dimensional problems, even linear regression on a small ℓ^1 -ball is sufficient if the best predictor is sparse. Our goal is, in both low and high dimensions, to provide online linear regression algorithms along with bounds on ℓ^1 -balls that characterize their robustness to worst-case scenarios.

1.1 Setting

We consider the online version of linear regression, which unfolds as follows. First, the environment chooses a sequence of observations $(y_t)_{t \geq 1}$ in \mathbb{R} and a sequence of input vectors $(\mathbf{x}_t)_{t \geq 1}$ in \mathbb{R}^d , both initially hidden from the forecaster. At each time instant $t \in \mathbb{N}^* = \{1, 2, \dots\}$, the environment reveals the data $\mathbf{x}_t \in \mathbb{R}^d$; the

* This research was carried out within the INRIA project CLASSIC hosted by École Normale Supérieure and CNRS.

forecaster then gives a prediction $\hat{y}_t \in \mathbb{R}$; the environment in turn reveals the observation $y_t \in \mathbb{R}$; and finally, the forecaster incurs the square loss $(y_t - \hat{y}_t)^2$. The dimension d can be either small or large relative to the number T of time steps: we consider both cases.

In the sequel, $\mathbf{u} \cdot \mathbf{v}$ denotes the standard inner product between $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, and we set $\|\mathbf{u}\|_\infty \triangleq \max_{1 \leq j \leq d} |u_j|$ and $\|\mathbf{u}\|_1 \triangleq \sum_{j=1}^d |u_j|$. The ℓ^1 -ball of radius $U > 0$ is the following bounded subset of \mathbb{R}^d :

$$B_1(U) \triangleq \{\mathbf{u} \in \mathbb{R}^d : \|\mathbf{u}\|_1 \leq U\}.$$

Given a fixed radius $U > 0$ and a time horizon $T \geq 1$, the goal of the forecaster is to predict almost as well as the best linear forecaster in the reference set $\{\mathbf{x} \in \mathbb{R}^d \mapsto \mathbf{u} \cdot \mathbf{x} \in \mathbb{R} : \mathbf{u} \in B_1(U)\}$, i.e., to minimize the regret on $B_1(U)$ defined by

$$\sum_{t=1}^T (y_t - \hat{y}_t)^2 - \min_{\mathbf{u} \in B_1(U)} \left\{ \sum_{t=1}^T (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \right\}.$$

We shall present algorithms along with bounds on their regret that hold uniformly over all sequences¹ $(\mathbf{x}_t, y_t)_{1 \leq t \leq T}$ such that $\|\mathbf{x}_t\|_\infty \leq X$ and $|y_t| \leq Y$ for all $t = 1, \dots, T$, where $X, Y > 0$. These regret bounds depend on four important quantities: U , X , Y , and T , which may be known or unknown to the forecaster.

1.2 Contributions and Related Works

The literature on online linear regression is extensive, we can only situate our work with those closest to ours.

Our first contribution consists of a minimax analysis of online linear regression on ℓ^1 -balls in the arbitrary sequence setting. We first provide a refined regret bound expressed in terms of Y , d , and a quantity $\kappa = \sqrt{TUX}/(2dY)$. This quantity κ is used to distinguish two regimes: we show a distinctive regime transition² at $\kappa = 1$ or $d = \sqrt{TUX}/(2Y)$. Namely, for $\kappa < 1$, the regret is of the order of \sqrt{T} , whereas it is of the order of $\ln T$ for $\kappa > 1$.

This regret bound matches the optimal risk bounds for stochastic settings³ [BM01, Tsy03, RWY09]. Hence, linear regression is just as hard in the stochastic setting as in the arbitrary sequence setting. Using the standard online to batch conversion, we make the latter statement more precise by establishing a lower bound for all κ at least of the order of $\sqrt{\ln d}/d$. This lower bound extends those of [CB99, KW97], which only hold for small κ of the order of $1/d$.

¹ Actually our results hold whether $(\mathbf{x}_t, y_t)_{t \geq 1}$ is generated by an oblivious environment or a non-oblivious opponent since we consider deterministic forecasters.

² In high dimensions (i.e., when $d > \omega T$, for some absolute constant $\omega > 0$), we do not observe this transition (cf. Figure 1).

³ For example, $(\mathbf{x}_t, y_t)_{1 \leq t \leq T}$ may be i.i.d., or \mathbf{x}_t can be deterministic and $y_t = f(\mathbf{x}_t) + \varepsilon_t$ for an unknown function f and an i.i.d. sequence $(\varepsilon_t)_{1 \leq t \leq T}$ of Gaussian noise.

The algorithm achieving our minimax regret bound is both computationally inefficient and non-adaptive (i.e., it requires prior knowledge of the quantities U , X , Y , or T that may be unknown in practice). The next two contributions tackle these issues.

In the same setting as ours, [CBLW96] presents a gradient descent algorithm with regret bounds relative to predictors in an ℓ^2 -ball. For the regret relative to predictors in an ℓ^1 -ball, the EG $^\pm$ algorithm of [KW97] achieves a regret bound of $2UXY\sqrt{2T\ln(2d)} + 2U^2X^2\ln(2d)$. This algorithm is efficient, and our lower bound in terms of κ shows that it is optimal up to logarithmic factors in the regime $\kappa \leq 1$. However, the EG $^\pm$ algorithm requires prior knowledge of U , X , Y , and T .

Our second contribution is a generic method, called *loss Lipschitzification*, which enables to adapt automatically to X , Y , and T when U is known. Our method transforms the loss function $\mathbf{u} \mapsto (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2$ into a Lipschitz continuous function and adapts to the unknown Lipschitz constant. The LEG algorithm (Section 3) illustrates this technique by modifying the EG $^\pm$ algorithm [KW97] to yield an algorithm of the same computational complexity that also achieves the minimax regret without needing to know X , Y , and T beforehand.

Our third contribution is a simple method to achieve minimax regret uniformly over all ℓ^1 -balls $B_1(U)$ for $U > 0$. This robustness property is similar to that of the p -norm algorithms [GL03], but our method guarantees a better regret bound⁴. This method aggregates instances of an algorithm that require prior knowledge of U . For the sake of simplicity, we assume that X , Y , and T are known, but explain in the discussions how to extend the method to a fully adaptive algorithm that requires neither of U , X , Y , nor T .

The SMIDAS algorithm [SST09] and the COMID algorithm [DSSST10], which generalize p -norm algorithms, can be shown to achieve the minimax regret if U , X , Y and T are known. The LEG algorithm (Section 3) does so without prior knowledge of the problem parameters X , Y and T . When U is unknown, the Scaling algorithm (Section 4) has a better bound than the SMIDAS algorithm⁵.

The paper is organized as follows. In Section 2 we establish our refined upper and lower bounds in terms of the intrinsic quantity κ . In Section 3, we present an efficient and adaptive algorithm that achieves the optimal regret on $B_1(U)$ when U is known. In Section 4, we use an aggregating strategy to achieve an optimal regret uniformly over all ℓ^1 -balls $B_1(U)$, for $U > 0$, when X , Y , and T are known. Finally, in Section 5, we discuss as an extension a fully automatic algorithm that requires no prior knowledge of U , X , Y , or T .

2 Optimal Rates

In this section, we first present a refined upper bound on the minimax regret on $B_1(U)$ for an arbitrary $U > 0$. In Corollary 1, we express this upper bound in

⁴ Our regret bound grows as U instead of U^2 .

⁵ Same comment.

terms of an intrinsic quantity $\kappa \triangleq \sqrt{TUX}/(2dY)$. The optimality of the latter bound is shown in Section 2.2.

2.1 Upper Bound

Theorem 1 (Upper bound). *Let $d, T \in \mathbb{N}^*$, and $U, X, Y > 0$. The minimax regret on $B_1(U)$ for bounded base predictions and observations satisfies*

$$\begin{aligned} & \inf_F \sup_{\|\mathbf{x}_t\|_\infty \leq X, |y_t| \leq Y} \left\{ \sum_{t=1}^T (y_t - \hat{y}_t)^2 - \inf_{\|\mathbf{u}\|_1 \leq U} \sum_{t=1}^T (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \right\} \\ & \leq \begin{cases} 3UXY\sqrt{2T\ln(2d)} & \text{if } U < \frac{Y}{X}\sqrt{\frac{\ln(1+2d)}{T\ln 2}}, \\ 26UXY\sqrt{T\ln\left(1 + \frac{2dY}{\sqrt{TUX}}\right)} & \text{if } \frac{Y}{X}\sqrt{\frac{\ln(1+2d)}{T\ln 2}} \leq U \leq \frac{2dY}{\sqrt{TUX}}, \\ 32dY^2\ln\left(1 + \frac{\sqrt{TUX}}{dY}\right) + dY^2 & \text{if } U > \frac{2dY}{X\sqrt{T}}, \end{cases} \end{aligned}$$

where the infimum is taken over all forecasters F and where the supremum extends over all sequences $(\mathbf{x}_t, y_t)_{1 \leq t \leq T} \in (\mathbb{R}^d \times \mathbb{R})^T$ such that $|y_1|, \dots, |y_T| \leq Y$ and $\|\mathbf{x}_1\|_\infty, \dots, \|\mathbf{x}_T\|_\infty \leq X$.

Theorem 1 improves the bound of [KW97, Theorem 5.11] for the EG $^\pm$ algorithm. First, our bound depends logarithmically—as opposed to linearly—on U for $U > 2dY/(\sqrt{T}X)$. Secondly, it is smaller by a factor ranging from 1 to $\sqrt{\ln d}$ when

$$\frac{Y}{X}\sqrt{\frac{\ln(1+2d)}{T\ln 2}} \leq U \leq \frac{2dY}{\sqrt{T}X}. \quad (1)$$

Hence, Theorem 1 answers a question⁶ raised in [KW97] about the gap of $\sqrt{\ln(2d)}$ between the upper and lower bounds.

The proof appears in [GY11]. It uses a Maurey-type argument: we randomize over a discretization of $B_1(U)$. Although this argument was used in the stochastic setting (cf. [Nem00, Tsy03, BN08, SSSZ10]), we adapt it to the deterministic setting. This is yet another technique that can be applied to both the stochastic and individual sequence settings.

The following corollary expresses the upper bound of Theorem 1 in terms of an intrinsic quantity $\kappa \triangleq \sqrt{TUX}/(2dY)$ that relates $\sqrt{TUX}/(2Y)$ to the ambient dimension d .

Corollary 1 (Upper bound in terms of an intrinsic quantity). *Let $d, T \in \mathbb{N}^*$, and $U, X, Y > 0$. The upper bound of Theorem 1 expressed in terms of d, Y , and the intrinsic quantity $\kappa \triangleq \sqrt{TUX}/(2dY)$ reads:*

⁶ The authors of [KW97] asked: “For large d there is a significant gap between the upper and lower bounds. We would like to know if it possible to improve the upper bounds by eliminating the $\ln d$ factors.”

$$\begin{aligned} & \inf_F \sup_{\|\mathbf{x}_t\|_\infty \leq X, |y_t| \leq Y} \left\{ \sum_{t=1}^T (y_t - \hat{y}_t)^2 - \inf_{\|\mathbf{u}\|_1 \leq U} \sum_{t=1}^T (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \right\} \\ & \leq \begin{cases} 6dY^2\kappa\sqrt{2\ln(2d)} & \text{if } \kappa < \frac{\sqrt{\ln(1+2d)}}{2d\sqrt{\ln 2}}, \\ 52dY^2\kappa\sqrt{\ln(1+1/\kappa)} & \text{if } \frac{\sqrt{\ln(1+2d)}}{2d\sqrt{\ln 2}} \leq \kappa \leq 1, \\ 32dY^2(\ln(1+2\kappa)+1) & \text{if } \kappa > 1. \end{cases} \end{aligned}$$

The upper bound of Corollary II is shown in Figure II. Observe that, in low dimension (Figure 1(b)), a clear transition from a regret of the order of \sqrt{T} to one of $\ln T$ occurs at $\kappa = 1$. This transition is absent for high dimensions: for $d \geq \omega T$, where $\omega \triangleq (32(\ln 3) + 1)^{-1}$, the regret bound $32dY^2(\ln(1+2\kappa)+1)$ is worse than a trivial bound of TY^2 when $\kappa \geq 1$.

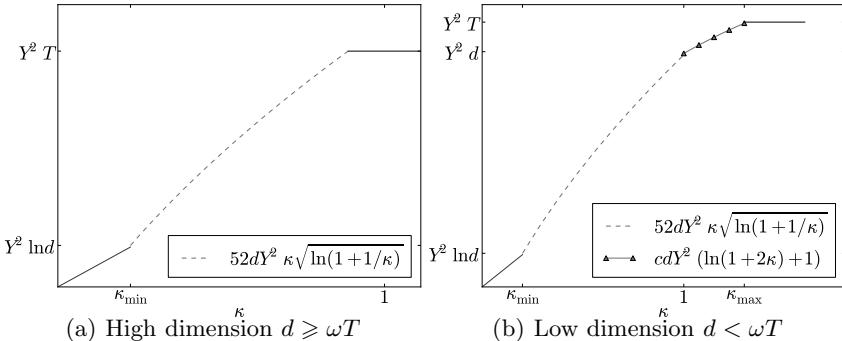


Fig. 1. The regret bound of Corollary II over $B_1(U)$ as a function of $\kappa = \sqrt{TUX}/(2dY)$. The constant c is chosen to ensure continuity at $\kappa = 1$, and $\omega \triangleq (32(\ln 3) + 1)^{-1}$. We define: $\kappa_{\min} = \sqrt{\ln(1+2d)}/(2d\sqrt{\ln 2})$ and $\kappa_{\max} = (e^{(T/d-1)/c} - 1)/2$.

2.2 Lower Bound

Corollary II gives an upper bound on the regret in terms of the quantities d , Y , and $\kappa \triangleq \sqrt{TUX}/(2dY)$. We now show that for all $d \in \mathbb{N}^*$, $Y > 0$, and $\kappa \geq \sqrt{\ln(1+2d)}/(2d\sqrt{\ln 2})$, the upper bound can not be improved⁷ up to logarithmic factors.

Theorem 2 (Lower bound). *For all $d \in \mathbb{N}^*$, $Y > 0$, and $\kappa \geq \frac{\sqrt{\ln(1+2d)}}{2d\sqrt{\ln 2}}$, there exist $T \geq 1$, $U > 0$, and $X > 0$ such that $\sqrt{TUX}/(2dY) = \kappa$ and*

⁷ For T sufficiently large, we may overlook the case $\kappa < \sqrt{\ln(1+2d)}/(2d\sqrt{\ln 2})$ or $\sqrt{T} < (Y/(UX))\sqrt{\ln(1+2d)/\ln 2}$. Observe that in this case, the minimax regret is already of the order of $Y^2 \ln(1+d)$ (cf. Figure II).

$$\begin{aligned} & \inf_F \sup_{\|\mathbf{x}_t\|_\infty \leq X, |y_t| \leq Y} \left\{ \sum_{t=1}^T (y_t - \hat{y}_t)^2 - \inf_{\|\mathbf{u}\|_1 \leq U} \sum_{t=1}^T (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \right\} \\ & \geq \begin{cases} \frac{c_1}{\ln(2+16d^2)} dY^2 \kappa \sqrt{\ln(1+1/\kappa)} & \text{if } \frac{\sqrt{\ln(1+2d)}}{2d\sqrt{\ln 2}} \leq \kappa \leq 1, \\ \frac{c_2}{\ln(2+16d^2)} dY^2 & \text{if } \kappa > 1, \end{cases} \end{aligned}$$

where $c_1, c_2 > 0$ are absolute constants. The infimum is taken over all forecasters F and the supremum extends over all sequences $(\mathbf{x}_t, y_t)_{1 \leq t \leq T} \in (\mathbb{R}^d \times \mathbb{R})^T$ such that $|y_1|, \dots, |y_T| \leq Y$ and $\|\mathbf{x}_1\|_\infty, \dots, \|\mathbf{x}_T\|_\infty \leq X$.

The above lower bound extends those of [CB99, KW97], which hold for small κ of the order of $1/d$. The proof appears in [GY11]. We perform a reduction to the stochastic batch setting—via the standard online to batch conversion, and employ a version of a lower bound of [Tsy03].

3 Adaptation to Unknown X , Y and T

Although the proof of Theorem 1 already gives an algorithm that achieves the minimax regret, the latter takes as inputs U , X , Y , and T , and it is inefficient in high dimensions. In this section, we present a new method that achieves the minimax regret both efficiently and without prior knowledge of X , Y , and T provided that U is known. Adaptation to an unknown U is considered in Section 4. Our method consists of modifying an underlying linear regression algorithm such as the EG $^\pm$ algorithm [KW97] or the sequential Ridge forecaster [Vov01, AW01]. Next, we show that the EG $^\pm$ algorithm with Lipschitzified losses achieves the minimax regret for the regime $d > \sqrt{TUX}/(2Y)$. A simpler modification (without loss Lipschitzification) can be applied to the Ridge forecaster to achieve a nearly optimal regret bound of order $dY^2 \ln\left(1 + d\left(\frac{\sqrt{TUX}}{dY}\right)^2\right)$ in the regime $d < \sqrt{TUX}/(2Y)$. The latter analysis is more technical and hence is omitted.

3.1 Lipschitzification of the Loss Function

The second algorithm of the proof of Theorem 1 is computationally inefficient because it aggregates approximately $d\sqrt{T}$ experts. In contrast, the EG $^\pm$ algorithm has a manageable computational complexity that is linear in d . We now describe a version of the EG $^\pm$ algorithm that is minimax optimal but does not require prior knowledge of X and Y —as opposed to the EG $^\pm$ algorithm. Our key technique consists of transforming the loss functions $\mathbf{u} \mapsto (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2$ into functions $\tilde{\ell}_t$ that are Lipschitz continuous with respect to $\|\cdot\|_1$. Afterward, adaptation to the unknown Lipschitz constants is carried out using the techniques of [CBMS07].

We point out that our Lipschitzification method can be applied to other algorithms, such as the p -norm algorithm and its regularized variants (SMIDAS and COMID) [GLO03, SST09, DSSST10]. The method may also apply to loss functions other than the square loss, e.g., convex but non-Lipschitz functions.

The Lipschitzification proceeds as follows. At each time t , we set

$$B_t \triangleq \left(2^{\lceil \log_2(\max_{1 \leq s \leq t-1} y_s^2) \rceil} \right)^{1/2},$$

so that $y_s \in [-B_t, B_t]$ for all $s = 1, \dots, t-1$. The modified loss function $\tilde{\ell}_t : \mathbb{R}^d \rightarrow \mathbb{R}$ is constructed as follows:

- if $|y_t| > B_t$, then

$$\tilde{\ell}_t(\mathbf{u}) = 0 \quad \text{for all } \mathbf{u} \in \mathbb{R}^d;$$

- if $|y_t| \leq B_t$, then $\tilde{\ell}_t$ is the convex function that coincides with the square loss when $|\mathbf{u} \cdot \mathbf{x}_t| \leq B_t$ and is linear elsewhere. This function is shown in Figure 2 and can be formally defined as

$$\tilde{\ell}_t(\mathbf{u}) \triangleq \begin{cases} (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 & \text{if } |\mathbf{u} \cdot \mathbf{x}_t| \leq B_t, \\ (y_t - B_t)^2 + 2(B_t - y_t)(\mathbf{u} \cdot \mathbf{x}_t - B_t) & \text{if } \mathbf{u} \cdot \mathbf{x}_t > B_t, \\ (y_t + B_t)^2 + 2(-B_t - y_t)(\mathbf{u} \cdot \mathbf{x}_t + B_t) & \text{if } \mathbf{u} \cdot \mathbf{x}_t < -B_t. \end{cases}$$

Observe that in both cases $|y_t| > B_t$ and $|y_t| \leq B_t$, the function $\tilde{\ell}_t$ is continuously differentiable and Lipschitz continuous with respect to $\|\cdot\|_1$ with Lipschitz constant

$$\|\nabla \tilde{\ell}_t\|_\infty \leq 2(|y_t| + B_t) \|\mathbf{x}_t\|_\infty \leq 2(1 + \sqrt{2}) \|\mathbf{x}_t\|_\infty \max_{1 \leq s \leq t} |y_s|, \quad (2)$$

where we used the fact that $B_t \leq \sqrt{2} \max_{1 \leq s \leq t-1} |y_s|$. We can also glean from Figure 2 that, when $|y_t| \leq B_t$, we have

$$\forall \mathbf{u} \in \mathbb{R}^d, \quad (y_t - [\mathbf{u} \cdot \mathbf{x}_t]_{B_t})^2 \leq \tilde{\ell}_t(\mathbf{u}) \leq (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2, \quad (3)$$

where for all $B > 0$, we define the clipping operator $[\cdot]_B$ by

$$[x]_B \triangleq \min\{B, \max\{-B, x\}\} \quad \text{for all } x \in \mathbb{R}.$$

3.2 Lipschitzifying Exponentiated Gradient Algorithm

Consider the LEG algorithm of Figure 3. Let $(\mathbf{e}_j)_{1 \leq j \leq d}$ denote the canonical basis of \mathbb{R}^d and $\pm U \mathbf{e}_j$ denote the vertices of $B_1(U)$. We use as a blackbox the exponentially weighted majority forecaster of [CBMS07] on $2d$ experts—namely, $\{\pm U \mathbf{e}_j : j = 1, \dots, d\}$ —as in [KW97]. It adapts to the unknown Lipschitz constant $\max_{1 \leq t \leq T} \|\nabla \tilde{\ell}_t\|_\infty$ by the particular choice of η_t .

We first need some notations. Following the tuning provided by [CBMS07], the parameter η_t of the LEG algorithm (see Figure 3) is defined by

$$\eta_t = \min \left\{ \frac{1}{\widehat{E}_{t-1}}, C \sqrt{\frac{\ln K}{V_{t-1}}} \right\}, \quad (4)$$

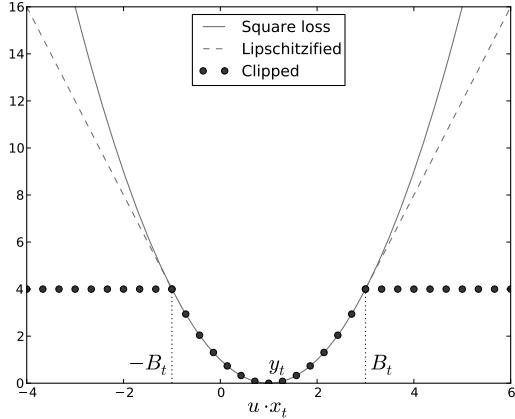


Fig. 2. Example when $|y_t| \leq B_t$. The square loss $(y_t - \mathbf{u} \cdot \mathbf{x}_t)^2$, its clipped version $(y_t - [\mathbf{u} \cdot \mathbf{x}_t]_{B_t})^2$ and its Lipschitzified version $\tilde{\ell}_t(\mathbf{u})$ are plotted as a function of $\mathbf{u} \cdot \mathbf{x}_t$.

where $C \triangleq \sqrt{2(\sqrt{2}-1)/(e-2)}$ and

$$\begin{aligned} z_{j,\varepsilon}^s &\triangleq (-1)^\varepsilon \nabla \tilde{\ell}_s(\hat{\mathbf{u}}_s) \cdot U e_j, \quad s \in \{1, \dots, T\}, j \in \{1, \dots, d\}, \varepsilon \in \{0, 1\}, \\ \hat{E}_{t-1} &\triangleq \inf_{k \in \mathbb{Z}} \left\{ 2^k : 2^k \geq \max_{1 \leq s \leq t-1} \left| \max_{\substack{1 \leq j \leq d \\ \varepsilon \in \{0,1\}}} z_{j,\varepsilon}^s - \min_{\substack{1 \leq j \leq d \\ \varepsilon \in \{0,1\}}} z_{j,\varepsilon}^s \right| \right\}, \\ V_{t-1} &\triangleq \sum_{s=1}^{t-1} \sum_{j,\varepsilon} w_{2j-1+\varepsilon,s} \left(z_{j,\varepsilon}^s - \sum_{k,\gamma} w_{2k-1+\gamma,s} z_{k,\gamma}^s \right)^2. \end{aligned}$$

Note that \hat{E}_{t-1} approximates the range of the $z_{j,\varepsilon}^s$ up to time $t-1$, while V_{t-1} is the corresponding cumulative variance of the forecaster.

The next theorem bounds the regret of the LEG algorithm on $B_1(U)$. This algorithm is efficient and adaptive in X and Y ; it achieves approximately the regret bound of Theorem II in the regime $\kappa \leq 1$ or $d \geq \sqrt{TUX}/(2Y)$.

Theorem 3. Let $U > 0$ and $T \geq 1$. Then, for all individual sequences $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T) \in \mathbb{R}^d \times \mathbb{R}$, the Lipschitzifying Exponentiated Gradient algorithm tuned with U satisfies the regret bound

$$\begin{aligned} \sum_{t=1}^T (y_t - \hat{y}_t)^2 - \inf_{\|\mathbf{u}\|_1 \leq U} \sum_{t=1}^T (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \\ \leq c_1 UXY \left(\sqrt{T \ln(2d)} + 8 \ln(2d) \right) + c_2 Y^2, \end{aligned}$$

where $c_1 \triangleq 8(\sqrt{2}+1)$ and $c_2 \triangleq 4(1+1/\sqrt{2})^2$, and where the quantities $X = \max_{1 \leq t \leq T} \|\mathbf{x}_t\|_\infty$ and $Y = \max_{1 \leq t \leq T} |y_t|$ are unknown to the forecaster.

Parameter: radius $U > 0$.

Initialization: $B_1 \triangleq 0$, $\hat{\mathbf{w}}_0 \triangleq (1/(2d), \dots, 1/(2d)) \in \mathbb{R}^{2d}$.

At each time round $t \geq 1$,

1. Compute the linear combination

$$\hat{\mathbf{u}}_t \triangleq U \sum_{j=1}^d (\hat{w}_{2j-1,t} - \hat{w}_{2j,t}) \mathbf{e}_j \in B_1(U); \quad (5)$$

2. Get $\mathbf{x}_t \in \mathbb{R}^d$ and output the clipped prediction $\hat{y}_t \triangleq [\hat{\mathbf{u}}_t \cdot \mathbf{x}_t]_{B_t}$;
3. Get $y_t \in \mathbb{R}$ and define the modified loss $\tilde{\ell}_t : \mathbb{R}^d \rightarrow \mathbb{R}$ as above;
4. Update the parameter η_{t+1} according to (4);
5. Update the weight vector $\mathbf{w}_{t+1} = (w_{1,t+1}, \dots, w_{2d,t+1})$ defined for all $j = 1, \dots, d$ and $\varepsilon \in \{0, 1\}$ by

$$w_{2j-1+\varepsilon,t+1} \triangleq \frac{\exp\left(-\eta_{t+1} \sum_{s=1}^t (-1)^\varepsilon \nabla \tilde{\ell}_s(\hat{\mathbf{u}}_s) \cdot U \mathbf{e}_j\right)}{\sum_{\substack{1 \leq k \leq K \\ \varepsilon' \in \{0,1\}}} \exp\left(-\eta_{t+1} \sum_{s=1}^t (-1)^{\varepsilon'} \nabla \tilde{\ell}_s(\hat{\mathbf{u}}_s) \cdot U \mathbf{e}_k\right)};$$

6. Update the threshold $B_{t+1} \triangleq \left(2^{\lceil \log_2(\max_{1 \leq s \leq t} y_s^2) \rceil}\right)^{1/2}$.

Fig. 3. The Lipschitzifying Exponentiated Gradient (LEG) algorithm

Proof (of Theorem 3). By definition of \hat{y}_t and $B_{t+1} \geq |y_t|$ we have

$$\begin{aligned} \sum_{t=1}^T (y_t - \hat{y}_t)^2 &\leq \sum_{\substack{t=1 \\ t: |y_t| \leq B_t}}^T \left(y_t - [\hat{\mathbf{u}}_t \cdot \mathbf{x}_t]_{B_t}\right)^2 + \sum_{\substack{t=1 \\ t: |y_t| > B_t}}^T (B_{t+1} + B_t)^2 \\ &\leq \sum_{\substack{t=1 \\ t: |y_t| \leq B_t}}^T \tilde{\ell}_t(\hat{\mathbf{u}}_t) + \left(1 + \frac{1}{\sqrt{2}}\right)^2 \sum_{\substack{t=1 \\ t: B_{t+1} > B_t}}^T B_{t+1}^2 \\ &\leq \sum_{t=1}^T \tilde{\ell}_t(\hat{\mathbf{u}}_t) + 4 \left(1 + \frac{1}{\sqrt{2}}\right)^2 Y^2, \end{aligned}$$

where the second inequality follows from the fact that:

- if $|y_t| \leq B_t$ then $(y_t - [\hat{\mathbf{u}}_t \cdot \mathbf{x}_t]_{B_t})^2 \leq \tilde{\ell}_t(\hat{\mathbf{u}}_t)$ by Equation (3);
- if $|y_t| > B_t$, which is equivalent to $B_{t+1} > B_t$ by definition of B_{t+1} , then $B_t \leq B_{t+1}/\sqrt{2}$, so that $B_{t+1} + B_t \leq (1 + 1/\sqrt{2})B_{t+1}$.

As for the third inequality above, we used the non-negativity of $\tilde{\ell}_t(\hat{\mathbf{u}}_t)$ and upper bounded the geometric sum $\sum_{t: B_{t+1} > B_t} B_{t+1}^2$ in the same way as in [CBMS07, Theorem 6], i.e., setting $K \triangleq \lceil \log_2 \max_{1 \leq t \leq T} y_t^2 \rceil$,

$$\sum_{t:B_{t+1}>B_t}^T B_{t+1}^2 \leq \sum_{k=-\infty}^K 2^k = 2^{K+1} \leq 4Y^2.$$

Since $\tilde{\ell}_t(\mathbf{u}) \leq (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2$ for all $\mathbf{u} \in \mathbb{R}^d$ (by Equation (3) if $|y_t| \leq B_t$, obvious otherwise), the last inequality yields

$$\begin{aligned} & \sum_{t=1}^T (y_t - \hat{y}_t)^2 - \inf_{\|\mathbf{u}\|_1 \leq U} \sum_{t=1}^T (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \\ & \leq \sum_{t=1}^T \tilde{\ell}_t(\hat{\mathbf{u}}_t) - \inf_{\|\mathbf{u}\|_1 \leq U} \sum_{t=1}^T \tilde{\ell}_t(\mathbf{u}) + 4 \left(1 + \frac{1}{\sqrt{2}}\right)^2 Y^2. \end{aligned} \quad (6)$$

But, by convexity and continuous differentiability of $\tilde{\ell}_t$,

$$\begin{aligned} & \sum_{t=1}^T \tilde{\ell}_t(\hat{\mathbf{u}}_t) - \inf_{\|\mathbf{u}\|_1 \leq U} \sum_{t=1}^T \tilde{\ell}_t(\mathbf{u}) \leq \sup_{\|\mathbf{u}\|_1 \leq U} \sum_{t=1}^T \nabla \tilde{\ell}_t(\hat{\mathbf{u}}_t) \cdot (\hat{\mathbf{u}}_t - \mathbf{u}) \\ & \leq \sum_{t=1}^T \nabla \tilde{\ell}_t(\hat{\mathbf{u}}_t) \cdot \hat{\mathbf{u}}_t - \min_{\substack{1 \leq j \leq d \\ \gamma \in \{\pm 1\}}} \sum_{t=1}^T \nabla \tilde{\ell}_t(\hat{\mathbf{u}}_t) \cdot (U\gamma \mathbf{e}_j), \end{aligned} \quad (7)$$

where the second inequality holds by linearity of $\mathbf{u} \mapsto \nabla \tilde{\ell}_t(\hat{\mathbf{u}}_t) \cdot (\hat{\mathbf{u}}_t - \mathbf{u})$ on the polytope $B_1(U)$. Next we use the particular form of $\hat{\mathbf{u}}_t$. By (5) we get

$$\begin{aligned} & \sum_{t=1}^T \nabla \tilde{\ell}_t(\hat{\mathbf{u}}_t) \cdot \hat{\mathbf{u}}_t - \min_{\substack{1 \leq j \leq d \\ \gamma \in \{\pm 1\}}} \sum_{t=1}^T \nabla \tilde{\ell}_t(\hat{\mathbf{u}}_t) \cdot (U\gamma \mathbf{e}_j) \\ & = \sum_{t=1}^T \sum_{\substack{1 \leq j \leq d \\ \varepsilon \in \{0,1\}}} w_{2j-1+\varepsilon,t} (-1)^\varepsilon \nabla \tilde{\ell}_t(\hat{\mathbf{u}}_t) \cdot U \mathbf{e}_j - \min_{\substack{1 \leq j \leq d \\ \varepsilon \in \{0,1\}}} \sum_{t=1}^T (-1)^\varepsilon \nabla \tilde{\ell}_t(\hat{\mathbf{u}}_t) \cdot U \mathbf{e}_j \\ & \leq 2U \max_{1 \leq t \leq T} \left\| \nabla \tilde{\ell}_t \right\|_\infty \left(2\sqrt{T \ln(2d)} + 4 \ln(2d) + 6 \right) \end{aligned} \quad (8)$$

$$\leq 8(\sqrt{2} + 1)UXY \left(\sqrt{T \ln(2d)} + 2 \ln(2d) + 3 \right), \quad (9)$$

where (8) follows straightforwardly⁸ from [CBMS07, Corollary 1], and where (9) follows from $\|\nabla \tilde{\ell}_t\|_\infty \leq 2(\sqrt{2} + 1)XY$ by (2). Putting Equations (6), (7), and (9) together and noting that $3 \leq 6 \ln(2d)$ concludes the proof. \square

4 Adaptation to Unknown U

In the previous section, the forecaster is given a radius $U > 0$ and asked to ensure a low worst-case regret on the ℓ^1 -ball $B_1(U)$. In this section, U is no

⁸ The weight vectors $\mathbf{w}_t \in \mathbb{R}^{2d}$ are exactly those of [CBMS07, Corollary 1] when applied to the loss vectors $(\nabla \tilde{\ell}_t(\hat{\mathbf{u}}_t) \cdot U \mathbf{e}_j, -\nabla \tilde{\ell}_t(\hat{\mathbf{u}}_t) \cdot U \mathbf{e}_j)_{1 \leq j \leq d} \in \mathbb{R}^{2d}$, $t = 1, \dots, T$.

longer given: the forecaster is asked to be competitive against all balls $B_1(U)$, for $U > 0$. Namely, its worst-case regret on each $B_1(U)$ should be almost as good as if U were known beforehand. For simplicity, we assume that X , Y , and T are known: we discuss in Section 5 how to simultaneously adapt to all parameters.

Parameters: $X, Y, \eta > 0$, $T \geq 1$, and $c > 0$ (a constant).
Initialization: $R = \lceil \log_2(2T/c) \rceil_+$, $w_1 = \mathbf{1}/(R+1) \in \mathbb{R}^{R+1}$.
For time steps $t = 1, \dots, T$:

1. For experts $r = 0, \dots, R$:
 - Run the sub-algorithm $\mathcal{A}(U_r)$ on the ball $B_1(U_r)$ and obtain the prediction $\hat{y}_t^{(r)}$.
2. Output the prediction $\hat{y}_t = \sum_{r=0}^R \frac{w_t^{(r)}}{\sum_{r'=0}^R w_t^{(r')}} [\hat{y}_t^{(r)}]_Y$.
3. Update $w_{t+1}^{(r)} = w_t^{(r)} \exp(-\eta(y_t - [\hat{y}_t^{(r)}]_Y)^2)$ for $r = 0, \dots, R$.

Fig. 4. The Scaling algorithm

We define

$$R \triangleq \lceil \log_2(2T/c) \rceil_+ \quad \text{and} \quad U_r \triangleq \frac{Y}{X} \frac{2^r}{\sqrt{T \ln(2d)}}, \quad \text{for } r = 0, \dots, R, \quad (10)$$

where $c > 0$ is a known absolute constant and

$$\lceil x \rceil_+ \triangleq \min\{k \in \mathbb{N} : k \geq x\} \quad \text{for all } x \in \mathbb{R}.$$

The Scaling algorithm of Figure 4 works as follows. We have access to a sub-algorithm $\mathcal{A}(U)$ which we run simultaneously for all $U = U_r$, $r = 0, \dots, R$. Each instance of the sub-algorithm $\mathcal{A}(U_r)$ performs online linear regression on the ℓ^1 -ball $B_1(U_r)$. We employ an exponentially weighted forecaster to aggregate these $R+1$ sub-algorithms to perform online linear regression simultaneously on the balls $B_1(U_0), \dots, B_1(U_R)$.

The following regret bound follows by exp-concavity of the square loss.

Theorem 4. Suppose that $X, Y > 0$ are known. Let $c, c' > 0$ be two absolute constants. Suppose that for all $U > 0$, we have access to a sub-algorithm $\mathcal{A}(U)$ with regret against $B_1(U)$ of at most

$$cUXY\sqrt{T\ln(2d)} + c'Y^2 \quad \text{for } T \geq T_0, \quad (11)$$

uniformly over all sequences (\mathbf{x}_t) and (y_t) bounded by X and Y . Then, for a known $T \geq T_0$, the Scaling algorithm with $\eta = 1/(8Y^2)$ satisfies

$$\begin{aligned} \sum_{t=1}^T (y_t - \hat{y}_t)^2 &\leq \inf_{\mathbf{u} \in \mathbb{R}^d} \left\{ \sum_{t=1}^T (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 + 2c \|\mathbf{u}\|_1 XY\sqrt{T\ln(2d)} \right\} \\ &\quad + 8Y^2 \ln(\lceil \log_2(2T/c) \rceil_+ + 1) + (c + c')Y^2. \end{aligned} \quad (12)$$

In particular, for every $U > 0$,

$$\begin{aligned} \sum_{t=1}^T (y_t - \hat{y}_t)^2 &\leq \inf_{\mathbf{u} \in B_1(U)} \left\{ \sum_{t=1}^T (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \right\} + 2cUXY\sqrt{T \ln(2d)} \\ &\quad + 8Y^2 \ln(\lceil \log_2(2T/c) \rceil + 1) + (c + c')Y^2. \end{aligned}$$

Remark 1. By Theorem 3 the LEG algorithm satisfies assumption (II) with $T_0 = \ln(2d)$, $c \triangleq 9c_1 = 72(\sqrt{2} + 1)$, and $c' \triangleq c_2 = 4(1 + 1/\sqrt{2})^2$.

Proof. Since the Scaling algorithm is an exponentially weighted average forecaster (with clipping) applied to the $R + 1$ experts $\mathcal{A}(U_r) = (\hat{y}_t^{(r)})_{t \geq 1}$, $r = 0, \dots, R$, we have, by Lemma 1 in the appendix,

$$\begin{aligned} \sum_{t=1}^T (y_t - \hat{y}_t)^2 &\leq \min_{r=0, \dots, R} \sum_{t=1}^T \left(\hat{y}_t^{(r)} - \hat{y}_t \right)^2 + 8Y^2 \ln(R + 1) \\ &\leq \min_{r=0, \dots, R} \left\{ \inf_{\mathbf{u} \in B_1(U_r)} \left\{ \sum_{t=1}^T (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \right\} + cU_r XY\sqrt{T \ln(2d)} \right\} + z, \quad (13) \end{aligned}$$

where the last inequality follows by assumption (II), and where we set

$$z \triangleq 8Y^2 \ln(R + 1) + c'Y^2.$$

Let $\mathbf{u}_T^* \in \arg \min_{\mathbf{u} \in \mathbb{R}^d} \left\{ \sum_{t=1}^T (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 + 2c \|\mathbf{u}\|_1 XY\sqrt{T \ln(2d)} \right\}$. Next, we proceed by considering three cases depending on the value of $\|\mathbf{u}_T^*\|_1$.

Case 1: $U_0 < \|\mathbf{u}_T^*\|_1 < U_R$. Let $r^* \triangleq \min\{r = 0, \dots, R : U_r \geq \|\mathbf{u}_T^*\|_1\}$. Note that $r^* \geq 1$ since $\|\mathbf{u}_T^*\|_1 > U_0$. By (13) we have

$$\begin{aligned} \sum_{t=1}^T (y_t - \hat{y}_t)^2 &\leq \inf_{\mathbf{u} \in B_1(U_{r^*})} \left\{ \sum_{t=1}^T (y_t - \mathbf{u} \cdot \mathbf{x}_t)^2 \right\} + cU_{r^*} XY\sqrt{T \ln(2d)} + z \\ &\leq \sum_{t=1}^T (y_t - \mathbf{u}_T^* \cdot \mathbf{x}_t)^2 + 2c \|\mathbf{u}_T^*\|_1 XY\sqrt{T \ln(2d)} + z, \end{aligned}$$

where the last inequality follows from $\mathbf{u}_T^* \in B_1(U_{r^*})$ and from the fact that $U_{r^*} \leq 2 \|\mathbf{u}_T^*\|_1$ (since, by definition of r^* , $\|\mathbf{u}_T^*\|_1 > U_{r^*-1} = U_{r^*}/2$). Finally, we obtain (12) by definition of \mathbf{u}_T^* and $z \triangleq 8Y^2 \ln(R + 1) + c'Y^2$.

Case 2: $\|\mathbf{u}_T^*\|_1 \leq U_0$. By (13) we have

$$\sum_{t=1}^T (y_t - \hat{y}_t)^2 \leq \left\{ \sum_{t=1}^T (y_t - \mathbf{u}_T^* \cdot \mathbf{x}_t)^2 + cU_0 XY\sqrt{T \ln(2d)} \right\} + z, \quad (14)$$

which yields (12) since $cU_0 XY\sqrt{T \ln(2d)} = cY^2$ (by definition of U_0), by adding $2c \|\mathbf{u}_T^*\|_1 XY\sqrt{T \ln(2d)} \geq 0$, and by definition of \mathbf{u}_T^* and z .

Case 3: $\|\mathbf{u}_T^*\|_1 \geq U_R$. By construction, we have $\hat{y}_t \in [-Y, Y]$, and by assumption, we have $y_t \in [-Y, Y]$, so that

$$\begin{aligned} \sum_{t=1}^T (y_t - \hat{y}_t)^2 &\leq 4Y^2 T \leq \sum_{t=1}^T (y_t - \mathbf{u}_T^* \cdot \mathbf{x}_t)^2 + 2cU_R XY \sqrt{T \ln(2d)} \\ &\leq \sum_{t=1}^T (y_t - \mathbf{u}_T^* \cdot \mathbf{x}_t)^2 + 2c \|\mathbf{u}_T^*\|_1 XY \sqrt{T \ln(2d)}, \end{aligned}$$

where the second inequality follows by $2cU_R XY \sqrt{T \ln(2d)} = 2cY^2 2^R \geq 4Y^2 T$ (since $2^R \geq 2T/c$ by definition of R), and the last inequality uses the assumption $\|\mathbf{u}_T^*\|_1 \geq U_R$. We finally get (12) by definition of \mathbf{u}_T^* .

This concludes the proof of the first claim (12). The second claim follows by bounding $\|\mathbf{u}\|_1 \leq U$. \square

5 Extension to a Fully Adaptive Algorithm and Other Discussions

The Scaling algorithm of Section 4 uses prior knowledge of Y , Y/X , and T . In order to obtain a fully automatic algorithm, we need to adapt efficiently to these quantities. Adaptation to Y is possible via a technique already used for the LEG algorithm, i.e., by updating the clipping range B_t based on the past observations $|y_s|$, $s \leq t-1$.

In parallel to adapting to Y , adaptation to Y/X can be carried out as follows. We replace the exponential sequence $\{U_0, \dots, U_R\}$ by another exponential sequence $\{U'_0, \dots, U'_{R'}\}$:

$$U'_r \triangleq \frac{1}{T^k} \frac{2^r}{\sqrt{T \ln(2d)}} , \quad r = 0, \dots, R' , \quad (15)$$

where $R' \triangleq R + \lceil \log_2 T^{2k} \rceil = \lceil \log_2(2T/c) \rceil + \lceil \log_2 T^{2k} \rceil$, and where $k > 1$ is a fixed constant. On the one hand, for $T \geq T_0 \triangleq \max\{(X/Y)^{1/k}, (Y/X)^{1/k}\}$, we have (cf. (10) and (15)),

$$[U_0, U_R] \subset [U'_0, U'_{R'}] .$$

Therefore, the analysis of Theorem 4 applied to the grid $\{U'_0, \dots, U'_{R'}\}$ yields⁹ a regret bound of the order of $UXY\sqrt{T \ln d} + Y^2 \ln(R'+1)$. On the other hand, clipping the predictions to $[-Y, Y]$ ensures the crude regret bound $4Y^2 T_0$ for small $T < T_0$. Hence, the overall regret for all $T \geq 1$ is of the order of

$$UXY\sqrt{T \ln d} + Y^2 \ln(k \ln T) + Y^2 \max\{(X/Y)^{1/k}, (Y/X)^{1/k}\} .$$

Adaptation to an unknown time horizon T can be carried out via a standard doubling trick on T . However, to avoid restarting the algorithm repeatedly, we can use a time-varying exponential sequence $\{U'_{-R'(t)}(t), \dots, U'_{R'(t)}(t)\}$ where

⁹ The proof remains the same by replacing $8Y^2 \ln(R+1)$ with $8Y^2 \ln(R'+1)$.

$R'(t)$ grows at the rate of $k \ln(t)$. This gives¹⁰ us an algorithm that is fully automatic in the parameters U , X , Y and T . In this case, we can show that the regret is of the order of

$$UXY\sqrt{T \ln d} + Y^2 k \ln(T) + Y^2 \max \left\{ \left(\sqrt{T} X/Y \right)^{1/k}, \left(Y/(\sqrt{T} X) \right)^{1/k} \right\},$$

where the last two terms are negligible when $T \rightarrow +\infty$ (since $k > 1$).

Next we discuss another possible improvement. There is a logarithmic gap between the upper bound of Theorem 1 and the lower bound of Theorem 2. This gap comes from a concentration argument on a specific sequence of (unbounded) normal random variables in the proof of the lower bound. We think that in the interval $\kappa \geq cd$ (for some large enough absolute constant $c > 0$), we can recover the missing $\ln(1 + 2\kappa)$ in our lower bound by using the argument of [Vov01, Theorem 2] instead. As for the interval $\kappa \leq cd$, we could use a different sequence of random variables with bounded support, and, e.g., Assouad's Lemma.

Acknowledgments. The authors would like to thank Gilles Stoltz for his helpful comments and suggestions. This work was supported in part by French National Research Agency (ANR, project EXPLO-RA, ANR-08-COSI-004) and the PASCAL2 Network of Excellence under EC grant no. 216886. J. Y. Yu was partly supported by a fellowship from Le Fonds québécois de la recherche sur la nature et les technologies.

References

- [AW01] Azoury, K.S., Warmuth, M.K.: Relative loss bounds for on-line density estimation with the exponential family of distributions. *Mach. Learn.* 43(3), 211–246 (2001)
- [BM01] Birgé, L., Massart, P.: Gaussian model selection. *J. Eur. Math. Soc.* 3, 203–268 (2001)
- [BN08] Bunea, F., Nobel, A.: Sequential procedures for aggregating arbitrary estimators of a conditional mean. *IEEE Trans. Inform. Theory* 54(4), 1725–1735 (2008)
- [CB99] Cesa-Bianchi, N.: Analysis of two gradient-based algorithms for on-line regression. *J. Comput. System Sci.* 59(3), 392–411 (1999)
- [CBL06] Cesa-Bianchi, N., Lugosi, G.: Prediction, learning, and games. Cambridge University Press, Cambridge (2006)
- [CBLW96] Cesa-Bianchi, N., Long, P.M., Warmuth, M.K.: Worst-case quadratic loss bounds for prediction using linear functions and gradient descent. *IEEE Transactions on Neural Networks* 7(3), 604–619 (1996)
- [CBMS07] Cesa-Bianchi, N., Mansour, Y., Stoltz, G.: Improved second-order bounds for prediction with expert advice. *Mach. Learn.* 66(2/3), 321–352 (2007)
- [DSSST10] Duchi, J.C., Shalev-Shwartz, S., Singer, Y., Tewari, A.: Composite objective mirror descent. In: Proceedings of the 23rd Annual Conference on Learning Theory (COLT 2010), pp. 14–26 (2010)

¹⁰ Each time the exponential sequence (U'_r) expands, the weights assigned to the existing points U'_r are appropriately reassigned to the whole new sequence.

- [GL03] Gentile, C., Littlestone, N.: The robustness of the p -norm algorithms. *Mach. Learn.* 53(3), 265–299 (2003)
- [GY11] Gerchinovitz, S., Yu, J.Y.: Adaptive and optimal online linear regression on ℓ^1 -balls. Technical report (2011), <http://arxiv.org/abs/1105.4042>
- [KW97] Kivinen, J., Warmuth, M.K.: Exponentiated gradient versus gradient descent for linear predictors. *Inform. and Comput.* 132(1), 1–63 (1997)
- [Nem00] Nemirovski, A.: Topics in Non-Parametric Statistics. Springer, Heidelberg (2000)
- [RWY09] Raskutti, G., Wainwright, M.J., Yu, B.: Minimax rates of convergence for high-dimensional regression under ℓ^q -ball sparsity. In: Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton 2009), pp. 251–257 (2009)
- [SSSZ10] Shalev-Shwartz, S., Srebro, N., Zhang, T.: Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM J. Optim.* 20(6), 2807–2832 (2010)
- [SST09] Shalev-Shwartz, S., Tewari, A.: Stochastic methods for ℓ^1 -regularized loss minimization. In: Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009), pp. 929–936 (2009)
- [Tsy03] Tsybakov, A.B.: Optimal rates of aggregation. In: Proceedings of the 16th Annual Conference on Learning Theory (COLT 2003), pp. 303–313 (2003)
- [Vov01] Vovk, V.: Competitive on-line statistics. *Internat. Statist. Rev.* 69, 213–248 (2001)

A Lemma

Next we recall a regret bound satisfied by the standard exponentially weighted average forecaster applied to clipped base forecasts. Assume that at each time $t \geq 1$, the forecaster has access to $K \geq 1$ base forecasts $\hat{y}_t^{(k)} \in \mathbb{R}$, $k = 1, \dots, K$, and that for some known bound $Y > 0$ on the observations, he predicts at time t as

$$\hat{y}_t \triangleq \sum_{k=1}^K p_{k,t} [\hat{y}_t^{(k)}]_Y .$$

In the equation above, $[x]_Y \triangleq \min\{Y, \max\{-Y, x\}\}$ for all $x \in \mathbb{R}$, and the weight vectors $\mathbf{p}_t \in \mathbb{R}^K$ are given by $\mathbf{p}_1 = (1/K, \dots, 1/K)$ and, for all $t = 2, \dots, T$, by

$$p_{k,t} \triangleq \frac{\exp\left(-\eta \sum_{s=1}^{t-1} \left(y_s - [\hat{y}_s^{(k)}]_Y\right)^2\right)}{\sum_{j=1}^K \exp\left(-\eta \sum_{s=1}^{t-1} \left(y_s - [\hat{y}_s^{(j)}]_Y\right)^2\right)}, \quad 1 \leq k \leq K,$$

for some parameter $\eta > 0$ to be chosen below. The next lemma is a straightforward consequence of Theorem 3.2 and Proposition 3.1 of [CBL06].

Lemma 1 (Exponential weighting with clipping). *Assume that the forecaster knows beforehand a bound $Y > 0$ on the observations $|y_t|$, $t = 1, \dots, T$. Then, the above forecaster tuned with $\eta \leq 1/(8Y^2)$ satisfies*

$$\sum_{t=1}^T (y_t - \hat{y}_t)^2 \leq \min_{1 \leq k \leq K} \sum_{t=1}^T (y_t - \hat{y}_t^{(k)})^2 + \frac{\ln K}{\eta} .$$

Re-adapting the Regularization of Weights for Non-stationary Regression

Nina Vaits and Koby Crammer

Department of Electrical Engineering,
The Technion, Haifa, Israel

nvaits@tx.technion.ac.il, koby@ee.technion.ac.il

Abstract. The goal of a learner in standard online learning is to have the cumulative loss not much larger compared with the best-performing prediction-function from some fixed class. Numerous algorithms were shown to have this gap arbitrarily close to zero compared with the best function that is chosen off-line. Nevertheless, many real-world applications (such as adaptive filtering) are non-stationary in nature and the best prediction function may not be fixed but drift over time. We introduce a new algorithm for regression that uses per-feature-learning rate and provide a regret bound with respect to the best sequence of functions with drift. We show that as long as the cumulative drift is sub-linear in the length of the sequence our algorithm suffers a regret that is sub-linear as well. We also sketch an algorithm that achieves the best of the two worlds: in the stationary settings has $\log(T)$ regret, while in the non-stationary settings has sub-linear regret. Simulations demonstrate the usefulness of our algorithm compared with other state-of-the-art approaches.

1 Introduction

We consider the classical problem of online learning for regression. On each iteration, the algorithm receives a new instance (for example, input from an array of antennas) and outputs a prediction of a real value (for example distance to the source). The correct value is then revealed, and the algorithm suffers a loss based on both its prediction and the correct output value.

In general, the goal of the learner is to achieve an average loss that is not *too big* compared with the loss it would have received if it had chosen to predict according to the single best-performing function from some fixed class. It is well-known that as the number of time steps grows, very simple aggregation algorithms are able to achieve an average loss arbitrarily close to that of the best function in retrospect. Furthermore, such guarantees hold even if the input and output pairs are chosen in a fully adversarial manner with no distributional assumptions [Q].

Despite the extensive and impressive guarantees that can be made for algorithms in such settings, competing with the best *fixed* function is not always good enough. In many real-world applications, the true target function is not *fixed*, but is slowly changing over time. Consider a filter designed to cancel echoes in an hall. Over time, people enter and leave the hall, furniture are being moved, microphones are replaced and so on. When this drift occurs, the predictor itself must also change in order to remain relevant.

These reasons led to the development of algorithms and accompanying analysis for drifting or shifting settings (for example [24][20][23] and the references therein). In this setting, the performance of an algorithm is compared with a sequence of functions (and not a single function). Often such a sequence is either drifting, where each function is close in some sense to its predecessor, or shifting, where conceptually the sequence can be partitioned into few segments, for each there is a single function that performs well on all examples of that segment.

Recently there is an increased amount of interest in algorithms that exploits second order information. For example the second order perceptron algorithm [5], confidence-weighted learning [10][8], adaptive regularization of weights (AROW) [9], all designed for classification; and AdaGrad [11] and FTPRL [25] for general loss functions.

In this paper we build on the AROW algorithm and develop an algorithm for regression. Such algorithms are known to work well in practice and converge fast for stationary-problems. However, for non-stationary problems AROW and other similar algorithms gradually stop updating their prediction rule, even though their performance deteriorates. We thus modify the algorithm to overcome these shortcomings. We analyze the algorithm in the worst-case regret framework and show, that as long as the amount of average-drift is sublinear, the average-loss of our algorithm will converge to the average-loss of the best sequence of functions. Specifically, we show that if the cumulative deviation is of order $\mathcal{O}(T^{1/p})$ for some known $p > 1$, then the cumulative regret is $\mathcal{O}(T^{(p+1)/(2p)} \log(T))$. We also show that for stationary setting the algorithm suffers logarithmic regret, similar to previous results [13].

Additionally, we sketch an algorithm that does not employ such prior knowledge. Specifically, this algorithm runs $C + 1$ copies of the algorithm mentioned above, each copy with a parameter chosen to fit a specific assumption of the amount of non-stationarity in the data. The algorithm then feeds these $C + 1$ outputs to an additional copy that computes a linear combination of these $C + 1$ outputs. Thus, the cumulative loss of the later algorithm is bounded by the cumulative loss of the best copy (ie the one with the “best” choice of parameter) with additional regret of $\mathcal{O}(\log T)$. Therefore, this algorithm for the non-stationary setting will suffer a polynomial sub-linear regret.

Notation: For a symmetric matrix Σ we denote its j th eigenvalue by $\lambda_j(\Sigma)$. Similarly we denote its smallest eigenvalue by $\lambda_{\min}(\Sigma) = \min_j \lambda_j(\Sigma)$, and its largest eigenvalue by $\lambda_{\max}(\Sigma) = \max_j \lambda_j(\Sigma)$.

2 Problem Setting

We work in the online setting for regression evaluated using the square loss. On each iteration our algorithm receives an instance $\mathbf{x}_t \in \mathbb{R}^d$ and predicts a real value $\hat{y}_t \in \mathbb{R}$ it then receives the correct label $y_t \in \mathbb{R}$, suffers loss $\ell(y_t, \hat{y}_t) = (\hat{y}_t - y_t)^2$. The algorithm then updates its prediction rule, and proceeds to the next round.

Our algorithms employs linear functions (with bounded norm), $\hat{y}_t = \mathbf{x}_t^\top \mathbf{w}_{t-1}$. The goal of the learning algorithm is to suffer low loss compared with the best linear function. Formally, we define the regret of an algorithm to be,

$$\mathcal{R}(T) = \sum_t^T (\mathbf{x}_t^\top \mathbf{w}_{t-1} - y_t)^2 - \inf_{\mathbf{u}} \sum_t^T (\mathbf{x}_t^\top \mathbf{u} - y_t)^2 .$$

The goal of the algorithm is to have $\mathcal{R}(T) = o(T)$, such that the average loss will converge to the average loss of the best linear function \mathbf{u} . We use an extended notion of evaluation, comparing the performance of an algorithm to the performance of a sequence of functions,

$$\mathcal{R}(T) = \sum_t^T (\mathbf{x}_t^\top \mathbf{w}_{t-1} - y_t)^2 - \inf_{\mathbf{u}_1, \dots, \mathbf{u}_T} \sum_t^T (\mathbf{x}_t^\top \mathbf{u}_t - y_t)^2.$$

It is reasonable to assume that the total deviation of the compared sequence is sub-linear in T , that is, for which $\sum_t \|\mathbf{u}_{t-1} - \mathbf{u}_t\| = o(T)$. Clearly, if the total deviation is $\Omega(T)$ we can not expect a learning algorithm to achieve a vanishing averaged regret. Yet, it may be the case that the sequence of functions is not converging $\lim_{t \rightarrow \infty} \sum_{s=1}^t \|\mathbf{u}_{s-1} - \mathbf{u}_s\|$ yet the algorithm will have vanishing average regret.

3 Algorithm

As in CW [10,8] and AROW [9] our algorithm maintains a Gaussian distribution parameterized by a mean $\mathbf{w}_t \in \mathbb{R}^d$ and a full covariance matrix $\Sigma_t \in \mathbb{R}^{d \times d}$. Intuitively, the mean \mathbf{w}_t represents the current linear function, while the covariance matrix Σ_t captures the uncertainty in the function \mathbf{w}_t . Given a new example (\mathbf{x}_t, y_t) the algorithm uses its current mean to make a prediction $\hat{y}_t = \mathbf{x}_t^\top \mathbf{w}_{t-1}$. Our algorithm then sets the new distribution to be the solution of the following optimization problem,

$$D_{KL}(\mathcal{N}(\mathbf{w}, \Sigma) \| \mathcal{N}(\mathbf{w}_{t-1}, \Sigma_{t-1})) + \frac{1}{2r} \ell(y_t - \mathbf{w}^\top \mathbf{x}_t) + \frac{1}{2r} (\mathbf{x}_t^\top \Sigma \mathbf{x}_t)$$

This optimization problem is similar to the one of AROW [9] for classification, except we use the square loss rather than squared-hinge loss used in AROW. Intuitively, the optimization problem trades off between three requirements. The first term forces the parameters not to change much per example, as the entire learning history is encapsulated within them. The second term requires that the new vector \mathbf{w}_t should perform well on the current instance, and finally, the third term reflects the fact that the uncertainty about the parameters reduces as we observe the current example \mathbf{x}_t . Writing the KL explicitly, we get the following.

$$\begin{aligned} \frac{1}{2} \log \left(\frac{\det \Sigma_{t-1}}{\det \Sigma} \right) + \frac{1}{2} \text{Tr}(\Sigma_{t-1}^{-1} \Sigma) + \frac{1}{2} (\mathbf{w}_{t-1} - \mathbf{w})^\top \Sigma_{t-1}^{-1} (\mathbf{w}_{t-1} - \mathbf{w}) - \frac{d}{2} \\ + \frac{1}{2r} \ell(y_t - \mathbf{w}^\top \mathbf{x}_t) + \frac{1}{2r} (\mathbf{x}_t^\top \Sigma \mathbf{x}_t). \end{aligned} \quad (1)$$

We now develop the update rule of (1) explicitly. Taking the derivative of (1) with respect to \mathbf{w} and setting it to zero, we get $\Sigma_{t-1}^{-1} (\mathbf{w} - \mathbf{w}_{t-1}) - \frac{1}{2r} 2(\mathbf{w}_t \cdot \mathbf{x}_t - y_t) \mathbf{x}_t = 0$. Therefore, if Σ_{t-1} is non-singular, the update for the mean parameters is given by

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \frac{1}{r} (\mathbf{w}_t \cdot \mathbf{x}_t - y_t) \Sigma_{t-1} \mathbf{x}_t. \quad (2)$$

We solve for \mathbf{w}_t by taking the dot product of each side of the equality with \mathbf{x}_t : $\mathbf{w}_t \cdot \mathbf{x}_t = \mathbf{w}_{t-1} \cdot \mathbf{x}_t - \frac{1}{r}(\mathbf{w}_t \cdot \mathbf{x}_t - y_t) \mathbf{x}_t^\top \Sigma_{t-1} \mathbf{x}_t$. Rearranging the terms yields, $(\mathbf{w}_t \cdot \mathbf{x}_t)(r + \mathbf{x}_t^\top \Sigma_{t-1} \mathbf{x}_t) = (\mathbf{w}_{t-1} \cdot \mathbf{x}_t)r + (y_t \mathbf{x}_t^\top \Sigma_{t-1} \mathbf{x}_t)$, and substituting back in (2), we get

$$\begin{aligned}\mathbf{w}_t &= \mathbf{w}_{t-1} - \frac{1}{r} \left(\frac{(\mathbf{w}_{t-1} \cdot \mathbf{x}_t)r + (y_t \mathbf{x}_t^\top \Sigma_{t-1} \mathbf{x}_t)}{r + \mathbf{x}_t^\top \Sigma_{t-1} \mathbf{x}_t} - y_t \right) \Sigma_{t-1} \mathbf{x}_t \\ &= \mathbf{w}_{t-1} - \left(\frac{(\mathbf{w}_{t-1} \cdot \mathbf{x}_t) - y_t}{r + \mathbf{x}_t^\top \Sigma_{t-1} \mathbf{x}_t} \right) \Sigma_{t-1} \mathbf{x}_t.\end{aligned}\quad (3)$$

We compute the update of the confidence parameters by setting the derivative of (1) with respect to Σ to zero, $-\frac{1}{2}\Sigma^{-1} + \frac{1}{2}\Sigma_{t-1}^{-1} + \frac{1}{2r}\mathbf{x}_t\mathbf{x}_t^\top = 0$. From this we obtain the following update for the confidence parameters.

$$\Sigma_t^{-1} = \Sigma_{t-1}^{-1} + \frac{1}{r}\mathbf{x}_t\mathbf{x}_t^\top. \quad (4)$$

Our goal is to develop algorithms for non-stationary settings. As observed in the context of CW [10], AROW [9], AdaGrad [11] and FTPRL [25] the matrix Σ can be interpreted as adaptive learning rate. Therefore, due to the update of (4) the eigenvalues of Σ_t goes to zero with t (equivalently, the update (4) forces the eigenvalues of Σ_t^{-1} to increase) and the effective learning rate goes to zero. As a consequence the algorithm will gradually stop updating using instances which lie in the subspace of examples that were previously observed numerous times. This property leads to fast convergence in the stationary case, but at the same time to poor performance in the non-stationary case. As it might happen there is a need to update the prediction rule with using some instance, yet the learning rate for this specific update is too small, and no useful update may be performed.

We propose two modifications to the above algorithm, that combined together overcome the problem that learning rate gradually goes to zero. The modified algorithm operates on segments of the input sequence. In each segment indexed by i , the algorithm checks whether the lowest eigenvalue of Σ_t is greater than a given lower bound Λ_i . Once the lowest eigenvalue of Σ_t is smaller than Λ_i the algorithm resets $\Sigma_t = I$ and updates the value of the lower bound Λ_{i+1} . Formally, the algorithm uses the update (4) to compute an intermediate candidate for Σ_t denote by $\tilde{\Sigma}_t = (\Sigma_{t-1}^{-1} + \frac{1}{r}\mathbf{x}_t\mathbf{x}_t^\top)^{-1}$. If indeed $\tilde{\Sigma}_t \succeq \Lambda_i I$ then it sets $\Sigma_t = \tilde{\Sigma}_t$ otherwise it sets $\Sigma_t = I$ and the segment index is increased by 1.

Additionally, before this modification, the norm of the weight vector \mathbf{w}_t did not increase much as the ‘‘effective’’ learning rate (the matrix Σ_t) went to zero. After our update, as the learning rate is effectively bounded from below, the norm of \mathbf{w}_t may increase too fast, which in turn will cause a low update-rate in non-stationarity inputs.

We thus employ one more modification exploited later by the analysis. After updating the mean \mathbf{w}_t as in (3) we project it into a ball B around the origin with radius R_B using a Mahalanobis distance. Formally, we define the function $\text{proj}(\tilde{\mathbf{w}}, \Sigma, R_B)$ to be the solution of the following optimization problem,

$$\arg \min_{\|\mathbf{w}\| \leq R_B} \frac{1}{2} (\mathbf{w} - \tilde{\mathbf{w}})^\top \Sigma^{-1} (\mathbf{w} - \tilde{\mathbf{w}}) \quad (5)$$

Parameters: $0 < r, R_B$, a sequence $1 > \Lambda_1 \geq \Lambda_2 \dots$

Initialize: Set $\mathbf{w}_0 = 0$, $\Sigma_0 = I$, $i = 1$

For $t = 1, \dots, T$ **do**

- Receive an instance \mathbf{x}_t
- Output prediction $\hat{y}_t = \mathbf{x}_t^\top \mathbf{w}_{t-1}$
- Receive the correct label y_t
- Update:

$$\tilde{\Sigma}_t^{-1} = \Sigma_{t-1}^{-1} + \frac{1}{r} \mathbf{x}_t \mathbf{x}_t^\top \quad (6)$$

$$\tilde{\mathbf{w}}_t = \mathbf{w}_{t-1} + \frac{(y_t - \mathbf{x}_t^\top \mathbf{w}_{t-1}) \Sigma_{t-1} \mathbf{x}_t}{r + \mathbf{x}_t^\top \Sigma_{t-1} \mathbf{x}_t} \quad (7)$$

- Update Σ_t :
If $\tilde{\Sigma}_t \succeq \Lambda_i I$ set $\Sigma_t = \tilde{\Sigma}_t$ else set $\Sigma_t = I$, $i = i + 1$
- Update \mathbf{w}_t :
 $\mathbf{w}_t = \text{proj}(\tilde{\mathbf{w}}_t, \Sigma_t, R_B)$

Output: \mathbf{w}_T , Σ_T

Fig. 1. ARCOR: adaptive regularization of weights for regression with covariance reset

We write the Lagrangian,

$$\mathcal{L} = \frac{1}{2} (\mathbf{w} - \tilde{\mathbf{w}})^\top \Sigma^{-1} (\mathbf{w} - \tilde{\mathbf{w}}) + \alpha \left(\frac{1}{2} \|\mathbf{w}\|^2 - \frac{1}{2} R_B^2 \right).$$

Setting to zero the gradient with respect to \mathbf{w} we get, $\Sigma^{-1} (\mathbf{w} - \tilde{\mathbf{w}}) + \alpha \mathbf{w} = 0$. Solving for \mathbf{w} we get

$$\mathbf{w} = (\alpha I + \Sigma^{-1})^{-1} \Sigma^{-1} \tilde{\mathbf{w}} = \frac{1}{\alpha} \left(I + (\alpha \Sigma)^{-1} \right)^{-1} \Sigma^{-1} \tilde{\mathbf{w}} = (I + \alpha \Sigma)^{-1} \tilde{\mathbf{w}}.$$

From KKT conditions we get that If $\|\tilde{\mathbf{w}}\| \leq R_B$ then $\alpha = 0$ and $\mathbf{w} = \tilde{\mathbf{w}}$. Otherwise, α is the unique positive scalar that satisfy the equality,

$$\| (I + \alpha \Sigma)^{-1} \tilde{\mathbf{w}} \| = R_B.$$

The value of α can be found using binary search and eigen-decomposition of the matrix Σ . We write explicitly $\Sigma = V \Lambda V^\top$ for a diagonal matrix Λ . By denoting $\mathbf{u} = V^\top \tilde{\mathbf{w}}$ we get that the last equation equals to, $\| (I + \alpha \Lambda)^{-1} \mathbf{u} \| = R_B$. We thus wish to find α such that $\sum_j^d \frac{u_j^2}{(1 + \alpha \Lambda_{j,j})^2} = R_B^2$. It can be done using a binary search in the range $\alpha \in [0, a]$ where $a = (\|\mathbf{u}\|/R_B - 1)/\lambda_{\min}(\Lambda)$. To summarize the projection step can be performed in time cubic in d and logarithmic in R_B and Λ_i . We call the algorithm ARCOR for adaptive regularization with covariance reset. A pseudo-code of the algorithm is summarized in Fig. 1. We note that it can be combined with Mercer kernels, and omit the details due to lack of space.

4 Analysis

We turn to analyze the algorithm in the non-stationary case, computing the regret with respect to more than a single comparison vector. Before we proceed with the bound we define additional notation. We denote by t_i the example index for which a restart was performed for the i th time, that is $\Sigma_{t_i} = I$ for all i . We define by n the total number of restarts, or intervals. We denote by $T_i = t_i - t_{i-1}$ the number of examples between two consecutive restarts. Clearly $T = \sum_{i=1}^n T_i$. Finally, we denote by $\Sigma^{i-1} = \Sigma_{t_{i-1}}$ just before the i th restart, and we note that it depends on exactly T_i examples (since the last restart).

In what follows we compare the performance of the algorithm to the performance of a sequence of weight vectors $\mathbf{u}_t \in \mathbb{R}^d$ all of which are of bounded norm R_B . In other words, all the vectors \mathbf{u}_t belong to $B = \{\mathbf{u} : \|\mathbf{u}\|_2 \leq R_B\}$.

Theorem 1. *Assume the algorithm of Fig. 1 is run with an input sequence $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$, that all the inputs are of unit norm $\|\mathbf{x}_t\| = 1$, and that the outputs are bounded $Y = \max_t |y_t|$. Let \mathbf{u}_t be any sequence of bounded weight vectors $\|\mathbf{u}_t\| \leq R_B$. The cumulative loss is bounded by,*

$$\begin{aligned} \sum_t (\mathbf{x}_t^\top \mathbf{w}_{t-1} - y_t)^2 &\leq \sum_t (\mathbf{x}_t^\top \mathbf{u}_t - y_t)^2 + 2(R_B^2 + Y^2) \sum_{i=1}^n \log \det \left((\Sigma^i)^{-1} \right) \\ &\quad + r \mathbf{u}_T^\top \Sigma_T^{-1} \mathbf{u}_T + 2R_B r \sum_t \frac{1}{\Lambda_{i(t)}} \|\mathbf{u}_{t-1} - \mathbf{u}_t\|, \end{aligned} \quad (8)$$

where n is the number of covariance restarts and Σ^{i-1} is the value of the covariance matrix just before the i th restart.

Note that the number of restarts n is not fixed but depends both on the total number of examples T and the scheme used to set the values of the lower bound of the eigenvalues Λ_i . In general, the lower the values of Λ_i are, the smaller the number of covariance-restarts occur, yet the larger the value of the last term of the bound is, which scales inversely proportional to Λ_i . A more precise statement is given in the next corollary.

Corollary 1. *Assume the algorithm of Fig. 1 made n restarts. Under the conditions of Theorem 1 we have,*

$$\begin{aligned} \sum_t (\mathbf{x}_t^\top \mathbf{w}_{t-1} - y_t)^2 &\leq \sum_t (\mathbf{x}_t^\top \mathbf{u}_t - y_t)^2 + 2(R_B^2 + Y^2) dn \log \left(1 + \frac{T}{nrd} \right) \\ &\quad + r \mathbf{u}_T^\top \Sigma_T^{-1} \mathbf{u}_T + 2R_B r \Lambda_n^{-1} \sum_t \|\mathbf{u}_{t-1} - \mathbf{u}_t\|, \end{aligned} \quad (9)$$

Proof. By definition we have $(\Sigma^i)^{-1} = I + \frac{1}{r} \sum_{t=t_i}^{T_i+t_i} \mathbf{x}_t \mathbf{x}_t^\top$. Denote the eigenvalues of $\sum_{t=t_i}^{T_i+t_i} \mathbf{x}_t \mathbf{x}_t^\top$ by $\lambda_1, \dots, \lambda_d$. Since $\|\mathbf{x}_t\| = 1$ their sum is $\text{Tr} \left(\sum_{t=t_i}^{T_i+t_i} \mathbf{x}_t \mathbf{x}_t^\top \right) = T_i$.

We use the concavity of the log function to bound $\log \det \left((\Sigma^i)^{-1} \right) = \sum_j^d \log \left(1 + \frac{\lambda_j}{r} \right) \leq d \log \left(1 + \frac{T_i}{rd} \right)$. Applying concavity again we bound the following sum,

$$\sum_i^n \log \det \left((\Sigma^i)^{-1} \right) \leq \sum_i^n d \log \left(1 + \frac{T_i}{rd} \right) \leq dn \log \left(1 + \frac{T}{nrd} \right),$$

where we used the fact that $\sum_i^n T_i = T$. Substituting the last inequality in Theorem 15 as well as using the monotonicity of the coefficients, $\Lambda_i \geq \Lambda_n$ for all $i \leq n$, yields the desired bound. ■

Implicitly, the second and fourth terms of the bound have opposite dependence on n . The second term is increasing with $n \ll T$, while the fourth is decreasing with n . If n is small it means that the lower bound Λ_n is very low (otherwise we would make more restarts) and thus Λ_n^{-1} is large. We now make this implicit dependence explicit.

Our goal is to bound the number of restarts n as a function of the number of examples T . This depends on the exact sequence of values Λ_i used. The following lemma provides a bound on n given a specific sequence of Λ_i .

Lemma 1. *Assume the algorithm of Fig. 15 is run with some sequence of Λ_i , then the number of restarts is upper bounded by,*

$$n \leq \min_N \left\{ N : T \leq r \sum_i^N (\Lambda_i^{-1} - 1) \right\}.$$

Proof. Since $\sum_{i=1}^n T_i = T$, then the number of restarts is maximized when the number of examples between restarts T_i is minimized. We prove now a lower bound on T_i for all $i = 1 \dots n$. A restart occurs for the i th time when the smallest eigenvalue of Σ_t is larger (for the first time) than Λ_i .

As before, by definition we have, $(\Sigma^i)^{-1} = I + \frac{1}{r} \sum_{t=t_i}^{T_i} \mathbf{x}_t \mathbf{x}_t^\top$. By Theorem 8.1.8 of [15] we have that there exists a matrix $A \in \mathbb{R}^{d \times T_i}$ with each column belongs to the $d-1$ -dimensional simplex (that is $a_{k,l} \geq 0$ and $\sum_k a_{k,l} = 1$ for $l = 1, \dots, T_i$) such that the k th eigenvalue λ_k^i of $(\Sigma^i)^{-1}$ equals to $\lambda_k^i = 1 + \frac{1}{r} \sum_{l=1}^{T_i} a_{k,l}$. The value of T_i is defined when the largest eigenvalue of $(\Sigma^i)^{-1}$ hits Λ_i^{-1} . Formally, we get the following lower bound on T_i ,

$$\begin{aligned} \arg \min_{\{a_{k,l}\}} s \quad & \text{s.t.} & \max_k \left(1 + \frac{1}{r} \sum_{l=1}^s a_{k,l} \right) \geq \Lambda_i^{-1} \\ & a_{k,l} \geq 0 \quad \text{for } k = 1, \dots, d, l = 1, \dots, s \\ & \sum_k a_{k,l} = 1 \quad \text{for } l = 1, \dots, s \end{aligned}$$

For a fixed s a maximal value $\max_k \left(1 + \frac{1}{r} \sum_{l=1}^s a_{k,l} \right)$ is obtained if all the “mass” is concentrated in one value k . That is we have $a_{k,l} = 1$ for $k = k_0$ and $a_{k,l} = 0$

otherwise. In this case $\max_k (1 + \frac{1}{r} \sum_{l=1}^s a_{k,l}) = (1 + \frac{1}{r}s)$ and the lower bound is obtained when $(1 + \frac{1}{r}s) = \Lambda_i^{-1}$. Solving for s we get that the shortest possible length of the i th interval is bounded by, $T_i \leq r(\Lambda_i^{-1} - 1)$. Summing over the last equation we get, $T = \sum_i^n T_i \leq r \sum_i^n (\Lambda_i^{-1} - 1)$. Thus, the number of restarts is upper bounded by the minimal value n that satisfy the last inequality. ■

Combining Lemma 1 with Corollary 1 we get,

Corollary 2. *Assume the algorithm of Fig. 1 is ran with a polynomial schema, that is $\Lambda_i^{-1} = i^{q-1} + 1$ for some $q > 1$. Under the conditions of Theorem 1 we have,*

$$\begin{aligned} \sum_t (\mathbf{x}_t^\top \mathbf{w}_{t-1} - y_t)^2 &\leq \sum_t (\mathbf{x}_t^\top \mathbf{u}_t - y_t)^2 + r \mathbf{u}_T^\top \Sigma_T^{-1} \mathbf{u}_T \\ &\quad + 2(R_B^2 + Y^2) d (q(T+1) + 1)^{\frac{1}{q}} \log \left(1 + \frac{T}{nrd} \right) \quad (10) \end{aligned}$$

$$+ 2R_B r \left((q(T+1) + 1)^{\frac{q-1}{q}} + 1 \right) \sum_t \|\mathbf{u}_{t-1} - \mathbf{u}_t\|. \quad (11)$$

Proof. Substituting $\Lambda_i^{-1} = i^{q-1} + 1$ in Lemma 1 we get,

$$r \sum_i^n (\Lambda_i^{-1} - 1) = r \sum_{i=1}^n i^{q-1} \geq \int_1^n x^{q-1} dx = \frac{1}{q} (n^q - 1). \quad (12)$$

Setting the last term to $T+1$ we get an upper bound on n ,

$$n \leq (q(T+1) + 1)^{1/q} \Rightarrow \Lambda_n^{-1} \leq (q(T+1) + 1)^{(q-1)/q} + 1. \quad (13)$$

Comparing the last two terms of the bound of Corollary 2 we observe a natural tradeoff in the value of q . The third term of (10) is decreasing with large values of q , while the fourth term of (11) is increasing with q .

Assuming a bound on the deviation $\sum_t \|\mathbf{u}_{t-1} - \mathbf{u}_t\| \leq C T^{1/p}$. We set $q = 2p/(p+1)$ and get that the sum of (10) and (11) is of order $\mathcal{O}(T^{(p+1)/(2p)} \log(T))$. As a consequence we get that, as long as $p > 1$ the sum of (10) and (11) is $o(T)$ and thus is vanishing. Furthermore, when the noise is very low we have $p \approx -(1 + \epsilon)$ in this case $q \approx 2 + (2/\epsilon)$, and we get that the algorithm will not make any restarts and retrieve the bound of $\mathcal{O}(\log T)$ for the stationary case. Intuitively, for this choice of q the algorithm will have only one interval, and there will be no restarts.

Intuitively, this schema to set Λ_i balances between the amount of noise (need for many restarts) and the property that using the covariance matrix for updates achieves fast-convergence. We note that an exponential schema $\Lambda_i = 2^{-i}$ will lead to very few restarts, and very small eigenvalues of the covariance matrix. This is because the last segment will be about half the length of the entire sequence.

5 Proof of Theorem 1

We first state the following lemmas, for which we define

$$d_t(\mathbf{z}, \mathbf{v}) = (\mathbf{z} - \mathbf{v})^\top \Sigma_t^{-1}(\mathbf{z} - \mathbf{v}), \quad d_{\tilde{t}}(\mathbf{z}, \mathbf{v}) = (\mathbf{z} - \mathbf{v})^\top \tilde{\Sigma}_t^{-1}(\mathbf{z} - \mathbf{v}), \quad \chi_t = \mathbf{x}_t^\top \Sigma_{t-1} \mathbf{x}_t.$$

Lemma 2. Let $\tilde{\mathbf{w}}_t$ and $\tilde{\Sigma}_t$ be defined in Fig. 1 and Fig. 2 in Fig. 1 then,

$$d_{t-1}(\mathbf{w}_{t-1}, \mathbf{u}_{t-1}) - d_{\tilde{t}}(\tilde{\mathbf{w}}_t, \mathbf{u}_{t-1}) = \frac{1}{r} \ell_t - \frac{1}{r} g_t - \frac{\ell_t \chi_t}{r(r + \chi_t)}. \quad (14)$$

The proof follows a chain of algebraic equalities and is omitted due to lack of space.

Lemma 3. Denote by $\Delta_t = d_{t-1}(\mathbf{w}_{t-1}, \mathbf{u}_{t-1}) - d_t(\mathbf{w}_t, \mathbf{u}_t)$ then

$$\begin{aligned} \Delta_t &\geq \frac{1}{r} (\ell_t - g_t) - \ell_t \frac{\chi_t}{r(r + \chi_t)} \\ &\quad + \mathbf{u}_{t-1}^\top \Sigma_{t-1}^{-1} \mathbf{u}_{t-1} - \mathbf{u}_t^\top \Sigma_t^{-1} \mathbf{u}_t - 2R_B \Lambda_i^{-1} \|\mathbf{u}_{t-1} - \mathbf{u}_t\| \end{aligned} \quad (15)$$

where $i - 1$ is the number of restarts performed before example t .

Proof. We write Δ_t as a telescopic sum of four terms as follows,

$$\begin{aligned} \Delta_{t,1} &= d_{t-1}(\mathbf{w}_{t-1}, \mathbf{u}_{t-1}) - d_{\tilde{t}}(\tilde{\mathbf{w}}_t, \mathbf{u}_{t-1}) & \Delta_{t,2} &= d_{\tilde{t}}(\tilde{\mathbf{w}}_t, \mathbf{u}_{t-1}) - d_t(\tilde{\mathbf{w}}_t, \mathbf{u}_{t-1}) \\ \Delta_{t,3} &= d_t(\tilde{\mathbf{w}}_t, \mathbf{u}_{t-1}) - d_t(\mathbf{w}_t, \mathbf{u}_{t-1}) & \Delta_{t,4} &= d_t(\mathbf{w}_t, \mathbf{u}_{t-1}) - d_t(\mathbf{w}_t, \mathbf{u}_t) \end{aligned}$$

We lower bound each of the four terms. Since, the value of $\Delta_{t,1}$ was computed in Lemma 2 we start with the second term. If no reset occurs then $\Sigma_t = \tilde{\Sigma}_t$ and $\Delta_{t,2} = 0$. Otherwise, we use the facts that $0 \preceq \tilde{\Sigma}_t \preceq I$, and $\Sigma_t = I$ and get,

$$\begin{aligned} \Delta_{t,2} &= (\tilde{\mathbf{w}}_t - \mathbf{u}_{t-1})^\top \tilde{\Sigma}_t^{-1} (\tilde{\mathbf{w}}_t - \mathbf{u}_{t-1}) - (\tilde{\mathbf{w}}_t - \mathbf{u}_{t-1})^\top \Sigma_t^{-1} (\tilde{\mathbf{w}}_t - \mathbf{u}_{t-1}) \\ &= \text{Tr} \left((\tilde{\mathbf{w}}_t - \mathbf{u}_{t-1}) (\tilde{\mathbf{w}}_t - \mathbf{u}_{t-1})^\top \left(\tilde{\Sigma}_t^{-1} - \Sigma_t^{-1} \right) \right) \\ &\geq \text{Tr} \left((\tilde{\mathbf{w}}_t - \mathbf{u}_{t-1}) (\tilde{\mathbf{w}}_t - \mathbf{u}_{t-1})^\top (I - I) \right) = 0. \end{aligned}$$

To summarize, $\Delta_{t,2} \geq 0$. We can lower bound $\Delta_{t,3}$ by using the fact that \mathbf{w}_t is a projection of $\tilde{\mathbf{w}}_t$ onto a closed set (a ball of radius R_B around the origin), which by our assumption contains \mathbf{u}_t . Employing Corollary 3 of [20] we get, $d_t(\tilde{\mathbf{w}}_t, \mathbf{u}_{t-1}) \geq d_t(\mathbf{w}_t, \mathbf{u}_{t-1})$ and thus $\Delta_{t,3} \geq 0$.

Finally, we lower bound the fourth term $\Delta_{t,4}$,

$$\begin{aligned} \Delta_4 &= (\mathbf{w}_t - \mathbf{u}_{t-1})^\top \Sigma_t^{-1} (\mathbf{w}_t - \mathbf{u}_{t-1}) - (\mathbf{w}_t - \mathbf{u}_t)^\top \Sigma_t^{-1} (\mathbf{w}_t - \mathbf{u}_t) \\ &= \mathbf{u}_{t-1}^\top \Sigma_t^{-1} \mathbf{u}_{t-1} - \mathbf{u}_t^\top \Sigma_t^{-1} \mathbf{u}_t - 2\mathbf{w}_t^\top \Sigma_t^{-1} (\mathbf{u}_{t-1} - \mathbf{u}_t) \end{aligned} \quad (16)$$

We use Hölder inequality and then Cauchy-Schwartz inequality to get the following lower bound,

$$\begin{aligned} -2\mathbf{w}_t^\top \Sigma_t^{-1} (\mathbf{u}_{t-1} - \mathbf{u}_t) &= -2\text{Tr} \left(\Sigma_t^{-1} (\mathbf{u}_{t-1} - \mathbf{u}_t) \mathbf{w}_t^\top \right) \\ &\geq -2\lambda_{\max}(\Sigma_t^{-1}) \mathbf{w}_t^\top (\mathbf{u}_{t-1} - \mathbf{u}_t) \geq -2\lambda_{\max}(\Sigma_t^{-1}) \|\mathbf{w}_t\| \|\mathbf{u}_{t-1} - \mathbf{u}_t\|. \end{aligned}$$

We use the facts that $\|\mathbf{w}_t\| \leq R_B$ and that $\lambda_{max}(\Sigma_t^{-1}) = 1/\lambda_{min}(\Sigma_t) \leq \Lambda_i^{-1}$, where i is the current segment index, and get

$$-2\mathbf{w}_t^\top \Sigma_t^{-1} (\mathbf{u}_{t-1} - \mathbf{u}_t) \geq -2\Lambda_i^{-1} R_B \|\mathbf{u}_{t-1} - \mathbf{u}_t\|. \quad (17)$$

Substituting (17) in (16) and using $\Sigma_t \preceq \Sigma_{t-1}$ we get,

$$\begin{aligned} \Delta_{t,4} &\geq \mathbf{u}_{t-1}^\top \Sigma_t^{-1} \mathbf{u}_{t-1} - \mathbf{u}_t^\top \Sigma_t^{-1} \mathbf{u}_t - 2R_B \Lambda_i^{-1} \|\mathbf{u}_{t-1} - \mathbf{u}_t\| \\ &\geq \mathbf{u}_{t-1}^\top \Sigma_{t-1}^{-1} \mathbf{u}_{t-1} - \mathbf{u}_t^\top \Sigma_t^{-1} \mathbf{u}_t - 2R_B \Lambda_i^{-1} \|\mathbf{u}_{t-1} - \mathbf{u}_t\|. \end{aligned} \quad (18)$$

Combining (18) with Lemma 2 concludes the proof. ■

Lemma 4. *During the runtime of the algorithm of Fig. 1 we have*

$$\sum_{t=t_i}^{t_i+T_i} \frac{\chi_t}{(\chi_t + r)} \leq \log \left(\det \left(\Sigma_{t_{i+1}-1}^{-1} \right) \right) = \log \left(\det \left((\Sigma^i)^{-1} \right) \right). \quad (19)$$

We remind the reader that: (1) t_i is the first example index after the i th restart (2) T_i is the number of examples observed before the next restart (3) the notation $\Sigma^i = \Sigma_{t_{i+1}-1}$ is the covariance matrix just before the next restart.

The proof of the lemma is similar to the proof of Lemma 4 [9] and thus omitted. We now turn to prove Theorem 1.

Proof. We bound the sum $\sum_t \Delta_t$ from above and below, and start with an upper bound using the property of telescopic sum and get,

$$\sum_t \Delta_t = (d_0(\mathbf{w}_0, \mathbf{u}_0) - d_T(\mathbf{w}_T, \mathbf{u}_T)) \leq d_0(\mathbf{w}_0, \mathbf{u}_0) \quad (20)$$

We compute a lower bound by applying Lemma 3 and get,

$$\begin{aligned} \sum_t \Delta_t &\geq \sum_t \left(\frac{1}{r} (\ell_t - g_t) - \ell_t \frac{\chi_t}{r(r + \chi_t)} \right. \\ &\quad \left. + \mathbf{u}_{t-1}^\top \Sigma_{t-1}^{-1} \mathbf{u}_{t-1} - \mathbf{u}_t^\top \Sigma_t^{-1} \mathbf{u}_t - 2R_B \Lambda_{i(t)}^{-1} \|\mathbf{u}_{t-1} - \mathbf{u}_t\| \right), \end{aligned}$$

where $i(t)$ is the number of restarts occurred before observing the t th example. We continue to develop the last equation and get,

$$\begin{aligned} \sum_t \Delta_t &\geq \frac{1}{r} \sum_t \ell_t - \frac{1}{r} \sum_t g_t - \sum_t \ell_t \frac{\chi_t}{r(r + \chi_t)} \\ &\quad + \sum_t \left(\mathbf{u}_{t-1}^\top \Sigma_{t-1}^{-1} \mathbf{u}_{t-1} - \mathbf{u}_t^\top \Sigma_t^{-1} \mathbf{u}_t \right) - \sum_t 2R_B \Lambda_{i(t)}^{-1} \|\mathbf{u}_{t-1} - \mathbf{u}_t\| \\ &= \frac{1}{r} \sum_t \ell_t - \frac{1}{r} \sum_t g_t - \sum_t \ell_t \frac{\chi_t}{r(r + \chi_t)} \\ &\quad + \mathbf{u}_0^\top \Sigma_0^{-1} \mathbf{u}_0 - \mathbf{u}_T^\top \Sigma_T^{-1} \mathbf{u}_T - 2R_B \sum_t \Lambda_{i(t)}^{-1} \|\mathbf{u}_{t-1} - \mathbf{u}_t\|. \end{aligned} \quad (21)$$

Combining (20) with (21) and using $d_0(\mathbf{w}_0, \mathbf{u}_0) = \mathbf{u}_0^\top \Sigma_0^{-1} \mathbf{u}_0$ ($\mathbf{w}_0 = \mathbf{0}$) we get,

$$\begin{aligned} \frac{1}{r} \sum_t \ell_t - \frac{1}{r} \sum_t g_t - \sum_t \ell_t \frac{\chi_t}{r(r + \chi_t)} - \mathbf{u}_T^\top \Sigma_T^{-1} \mathbf{u}_T \\ - 2R_B \sum_t A_{i(t)}^{-1} \|\mathbf{u}_{t-1} - \mathbf{u}_t\| \leq 0. \end{aligned}$$

Rearranging the terms we get,

$$\sum_t \ell_t \leq \sum_t g_t + \sum_t \ell_t \frac{\chi_t}{r + \chi_t} + r \mathbf{u}_T^\top \Sigma_T^{-1} \mathbf{u}_T + 2R_B r \sum_t \frac{1}{A_{i(t)}} \|\mathbf{u}_{t-1} - \mathbf{u}_t\| \quad (22)$$

Since $\|\mathbf{w}_t\| \leq R_B$, (and we assume that) $\|\mathbf{x}_t\| = 1$ and $\sup_t |y_t| = Y$, we get that $\sup_t \ell_t \leq 2(R_B^2 + Y^2)$. The second term in the right-hand-side is bounded using the last inequality and Lemma 4.

$$\begin{aligned} \sum_t \ell_t \frac{\chi_t}{r + \chi_t} &= \sum_i^n \sum_{t=t_i}^{t_i+T_i} \ell_t \frac{\chi_t}{r + \chi_t} \\ &\leq \sum_i^n \left(\sup_t \ell_t \right) \log \det \left((\Sigma^i)^{-1} \right) \\ &\leq 2(R_B^2 + Y^2) \sum_i^n \log \det \left((\Sigma^i)^{-1} \right). \end{aligned} \quad (23)$$

■

To conclude, we showed that if the algorithm is given an upper bound on the amount of drift, which is sub-linear in T , it can achieve sub-linear regret. Furthermore, if it is known that there is no non-stationarity in the reference vectors, then running the algorithm with $q = \infty$ will have a regret logarithmic in T . We use this property in the next section, where we describe a version of the algorithm when such a bound is not known, or is very loose.

6 Algorithm for Unknown Amount of Drift

We sketch now an algorithm, which does not assuming knowing the exact drift level, yet achieves $\log(T)$ regret in the stationary case, and sub-linear regret otherwise.

In a nutshell, the algorithm runs $C + 1$ copies of ARCOR, one with $q = \infty$ (no reset) and the others with $q = 1, 2, 3, \dots, C$. On each iteration the input vector \mathbf{x}_t is fed into the $C + 1$ copies each of which computes a prediction $\hat{y}_{t,c}$. These $C + 1$ predictions are collected into a vector in \mathbb{R}^{C+1} . This vector is then fed into another copy of the algorithm which is run with $q = \infty$ and $R_B = 1$. Denote its weight vector by $\mathbf{v}_t \in \mathbb{R}^{C+1}$. The output of our algorithm is thus $\hat{y}_t = \mathbf{v}_{t-1}^\top \hat{y}_t$. Given the feedback, the algorithm updates all $C + 1$ copies using ARCOR, as well as the additional copy. Conceptually, we position $C + 2$ copies of the algorithm in a network of depth 2. The

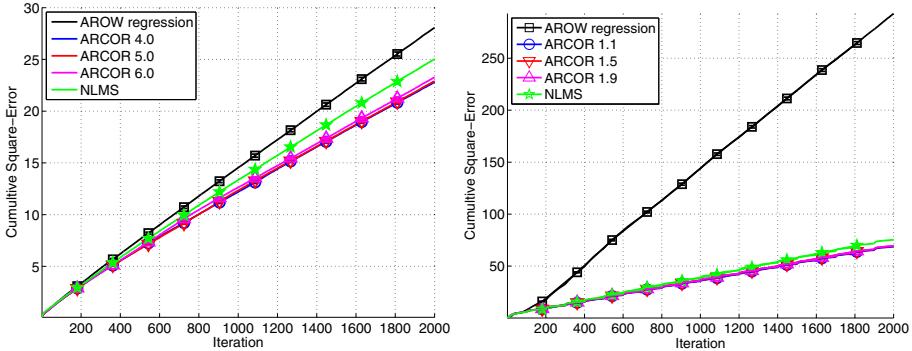


Fig. 2. Cumulative squared loss for AROW for regression, ARCOR with few value of q and NLMS vs iteration. Left panel shows result for dataset with no shifts and the right panel for dataset with shifts.

first layer is composed of $C + 1$ nodes, each runs its own copy of the algorithm under a specific assumption of drift value (as above). The outputs if the first layer are fed into the second layer, that integrates them linearly into a final output.

Intuitively, since the value of v_t can be any member of the standard basis (ie $(0..1..0)$), we get that the loss of the additional copy is bounded with the loss of the best copy with additional $\log(T)$ term (the regret for the stationary case). Thus if the best copy is the one with $q = \infty$ (ARCOR for the stationary case), the total regret is logarithmic in T . Otherwise, in the non-stationary case, the regret of the best copy would be polynomial in T , which is the final regret. We just sketched the proof of the following theorem,

Theorem 2. *Assuming the algorithm just presented is run with $C + 1$ copies of AR-COR and the additional copy. Then, the total loss of the algorithm is bounded by $\sum_t (\hat{y}_t - y_t)^2 \leq \min_{c=1}^{C+1} \sum_t (\hat{y}_{c,t} - y_t)^2 + D \log(T)$, where D is a constant depending on the number of copies C and the parameter used r .*

We note that Theorem 1 or one of its corollaries can be used to bound each of the terms $\sum_t^T (\hat{y}_{c,t} - y_t)^2$. Specifically, if the minima is obtained for the stationary case we get a logarithmic regret all together, otherwise the polynomial term is dominant in the bound.

7 Simulations

We illustrate the algorithms with two synthetic datasets, one with drifting only, and the other also with switching. We generated 2,000 points in \mathbb{R}^{20} where the first ten coordinates were grouped into five groups of size two. Each such pair was drawn from a 45° rotated Gaussian distribution with standard deviation 10 and 1. The remaining 10 coordinates were drawn from independent Gaussian distributions $\mathcal{N}(0, 2)$. The first dataset was generated using a sequence of vectors $u_t \in \mathbb{R}^{20}$ for which the only non-zero coordinates are the first two. This vector in \mathbb{R}^2 is of unit norm $\|u_t\| = 1$ and rotating in a rate of $t^{-0.01}$.

Similarly, the second dataset was generated with a sequence of rate $t^{-0.5}$, but with one additional twist. Every 50 time-steps the two-dimensional vector defined above was “embedded” in different pair of coordinates of the reference vector \mathbf{u}_t , for the first 50 steps it were coordinates 1, 2 in the next 50 examples, coordinates 3, 4 and so on. This change causes a switch in the reference vector \mathbf{u}_t . Finally, the target label y_t was set to be $\mathbf{x}_t^\top \mathbf{u}_t + \xi_t$ where $\xi_t \sim \mathcal{N}(0, 2)$.

Three algorithms are evaluated: NLMS (normalized least mean square) [20] which is a state-of-the-art first-order algorithm, AROW for regression, as developed in Sec. 3 with no restarting nor projection and ARCOR for various value of q . All algorithms have one parameter to tune, which was performed using a single random sequence. We repeat each experiment 100 reporting the mean cumulative square-loss and 95% confidence interval. We note that Aggregating Algorithm (AA) and Ridge Regression(RR) algorithm are very close algorithmically and in performance to AROW for regression and thus omitted.

The results are summarized in Fig. 2. In a nutshell, AROW for regression performs worst, NLMS is second and ARCOR is the best. This is surprising, as AROW for classification outperforms many algorithms that are related in spirit to NLMS. Yet, as mentioned above, the algorithm drives its learning rate to zero, not allowing for the ability to track drifting concepts. For both datasets, and mainly for the one with switching (right panel), AROW for regression is sensitive to the non-stationary properties of the data, and thus suffers very large loss, as its tracking ability is very slow. NLMS has nice tracking properties, but its learning rate is relatively slow. ARCOR tracks as fast as AROW, yet it bounds the learning rate and thus allows fast tracking rate. Note that in both datasets the “gap” between the cumulative error of all algorithms increases with time, this means that ARCOR tracks better both on drifting and switching settings.

8 Related Work and Summary

There is a large body of research in online learning for regression problems. Almost half a century ago, Widrow and Hoff [28] developed a variant of the least mean squares (LMS) algorithm for adaptive filetering and noise reduction. The algorithm was further developed and analyzed extensively (see e.g. [12]). The normalized least mean squares filter (NLMS) [20] builds on LMS and performs better to scaling of the input. The recursive least squares (RLS) [19] is the closest to our algorithm in the signal processing literature, it also maintains a weight-vector and a covariance-like positive semi-definite (PSD) matrix used to re-weight the input.

In the machine learning literature the problem of online regression was studied extensively, and clearly we can not cover all the relevant work. Cesa-Bianchi et al. [4] studied gradient descent based algorithms for regression with the squared loss. Kivinen and Warmuth [22] proposed various generalizations for general regularization functions. We refer the reader to a comprehensive book in the subject [6].

Foster [14] studied an online version of the ridge regression algorithm in the worst-case setting. Vovk [18] proposed a related algorithm called the Aggregating Algorithm (AA), and later Forster [13] improved its regret analysis for the square loss. Both algorithms employ second order information. ARCOR for the separable case is very similar

to these algorithms, although has alternative derivation. Recently, few algorithms were proposed either for classification [5][10][8][9] or for general loss functions [11][25] in the online convex programming framework. Our work shares the same design principles of AROW [9] yet it is aimed for regression. Furthermore, it has two important modifications which makes it work in the drifting or shifting setting. These modifications make the analysis more complex than of AROW.

Two of the approaches used in previous algorithms for non-stationary setting are to bound the weight vector and covariance reset. Bounding the weight vector was performed either by projecting it into a bonded set [20], shrinking it by multiplication [23] or subtracting previously seen examples [3]. These three methods (or at least most of their variants) can be combined with kernel operators, and in fact, the last two approaches were designed and motivated by kernels.

The Covariance Reset RLS algorithm (CR-RLS) [26][17][7] was designed for adaptive filtering. CR-RLS makes covariance reset every fixed amount of data points, while AR-COR performs restarts based on the actual properties of the data: the eigenspectrum of the covariance matrix. Furthermore, as far as we know, there is no analysis in the mistake bound model for this algorithm. Both ARCOR and CR-RLS are motivated from the property that the covariance matrix goes to zero and becomes rank deficient.

In both algorithms the information encapsulated in the covariance matrix is lost after restarts. In a rapidly varying environments, like a wireless channel, this loss of memory can be beneficial, as previous contributions to the covariance matrix may have little correlation with the current structure. Recent versions of RLS+CR [16][27] employ covariance reset to have numerically stable computations.

Our work combines both techniques with online learning with second order algorithm for regression. In this aspect we have the best of all worlds, fast convergence rate due to the usage of second order information, and the ability to adapt in non-stationary environments due to projection and resets. Current work includes extending the algorithm for general loss function, efficient implementation of the algorithm and automatically detecting the level of non-stationarity.

Acknowledgments. This research was supported in part by the Israeli Science Foundation grant ISF-1567/10 any by the European Unions Seventh Framework Programme (FP7/2007-2013) under PASCAL2 (PUMP PRIMING). KC is a Horev Fellow, supported by the Taub Foundations.

References

1. Auer, P., Warmuth, M.K.: Tracking the best disjunction. *Electronic Colloquium on Computational Complexity (ECCC)* 7(70) (2000)
2. Bershad, N.J.: Analysis of the normalized lms algorithm with gaussian inputs. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34(4), 793–806 (1986)
3. Cavallanti, G., Cesa-Bianchi, N., Gentile, C.: Tracking the best hyperplane with a simple budget perceptron. *Machine Learning* 69(2-3), 143–167 (2007)
4. Cesa-Bianchi, N., Long, P.M., Warmuth, M.K.: Worst case quadratic loss bounds for on-line prediction of linear functions by gradient descent. Technical Report IR-418, University of California, Santa Cruz, CA, USA (1993)

5. Cesa-Bianchi, N., Conconi, A., Gentile, C.: A second-order perceptron algorithm. *Siam Journal of Computation* 34(3), 640–668 (2005)
6. Cesa-Bianchi, N., Lugosi, G.: *Prediction, Learning, and Games*. Cambridge University Press, New York (2006)
7. Chen, M.-S., Yen, J.-Y.: Application of the least squares algorithm to the observer design for linear time-varying systems. *IEEE Transactions on Automatic Control* 44(9), 1742–1745 (1999)
8. Crammer, K., Dredze, M., Pereira, F.: Exact confidence-weighted learning. In: *NIPS*, vol. 22 (2008)
9. Crammer, K., Kulesza, A., Dredze, M.: Adaptive regularization of weighted vectors. In: *Advances in Neural Information Processing Systems*, vol. 23 (2009)
10. Dredze, M., Crammer, K., Pereira, F.: Confidence-weighted linear classification. In: *ICML* (2008)
11. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. In: *COLT*, pp. 257–269 (2010)
12. Feuer, A., Weinstein, E.: Convergence analysis of lms filters with uncorrelated Gaussian data. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 33(1), 222–230 (1985)
13. Forster, J.: On relative loss bounds in generalized linear regression. In: Ciobanu, G., Păun, G. (eds.) *FCT 1999. LNCS*, vol. 1684, pp. 269–280. Springer, Heidelberg (1999)
14. Foster, D.P.: Prediction in the worst case. *The Annals of Statistics* 19(2), 1084–1090 (1991)
15. Golub, G.H., Van Loan, C.F.: *Matrix computations*, 3rd edn. Johns Hopkins University Press, Baltimore (1996)
16. Goodhart, S.G., Burnham, K.J., James, D.J.G.: Logical covariance matrix reset in self-tuning control. *Mechatronics* 1(3), 339–351 (1991)
17. Goodwin, G.C., Teoh, E.K., Elliott, H.: Deterministic convergence of a self-tuning regulator with covariance resetting. *Control Theory and App.*, IEE Proc. D 130(1), 6–8 (1983)
18. Vovk, V.G.: Aggregating strategies. In: *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pp. 371–383. Morgan Kaufmann, San Francisco (1990)
19. Hayes, M.H.: 9.4: Recursive least squares. In: *Statistical Digital Signal Processing and Modeling*, p. 541 (1996)
20. Herbster, M., Warmuth, M.K.: Tracking the best linear predictor. *Journal of Machine Learning Research* 1, 281–309 (2001)
21. Itmead, R.R., Anderson, B.D.O.: Performance of adaptive estimation algorithms in dependent random environments. *IEEE Transactions on Automatic Control* 25, 788–794 (1980)
22. Kivinen, J., Warmuth, M.K.: Exponential gradient versus gradient descent for linear predictors. *Information and Computation* 132, 132–163 (1997)
23. Kivinen, J., Smola, A.J., Williamson, R.C.: Online learning with kernels. In: *NIPS*, pp. 785–792 (2001)
24. Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. *Inf. Comput.* 108(2), 212–261 (1994)
25. McMahan, H.B., Streeter, M.J.: Adaptive bound optimization for online convex optimization. In: *COLT*, pp. 244–256 (2010)
26. Salgado, M.E., Goodwin, G.C., Middleton, R.H.: Modified least squares algorithm incorporating exponential resetting and forgetting. *International Journal of Control* 47(2), 477–491 (1988)
27. Song, H.-S., Nam, K., Mutschler, P.: Very fast phase angle estimation algorithm for a single-phase system having sudden phase angle jumps. In: *Industry Applications Conference. 37th IAS Annual Meeting*, vol. 2, pp. 925–931 (2002)
28. Widrow, B., Hoff Jr., M.E.: Adaptive switching circuits (1960)

Competing against the Best Nearest Neighbor Filter in Regression

Arnak S. Dalalyan and Joseph Salmon

¹ Université Paris Est, Ecole des Ponts ParisTech,
77455 Marne-la-Vallée Cedex 2, France
dalalyan@imagine.enpc.fr

² Electrical and Computer Engineering,
Duke University, 7P.O. Box 90291
Durham, NC 27708
joseph.salmon@duke.edu

Abstract. Designing statistical procedures that are provably almost as accurate as the best one in a given family is one of central topics in statistics and learning theory. Oracle inequalities offer then a convenient theoretical framework for evaluating different strategies, which can be roughly classified into two classes: selection and aggregation strategies. The ultimate goal is to design strategies satisfying oracle inequalities with leading constant one and rate-optimal residual term. In many recent papers, this problem is addressed in the case where the aim is to beat the best procedure from a given family of linear smoothers. However, the theory developed so far either does not cover the important case of nearest-neighbor smoothers or provides a suboptimal oracle inequality with a leading constant considerably larger than one. In this paper, we prove a new oracle inequality with leading constant one that is valid under a general assumption on linear smoothers allowing, for instance, to compete against the best nearest-neighbor filters.

Keywords: adaptive smoothing, nonparametric regression, supervised learning.

1 Introduction

Linear procedures are omnipresent in machine learning. Sliding windows estimators, nearest neighbor smoothers, support vector machines with L^2 loss, etc., are popular examples of learning procedures obtained from the data vector by a linear transform. However, the performance of these procedures is, in general, severely affected by the choice of various tuning parameters. A typical example is presented in Figure 1: among the three linear estimators of a regression function, the two up-most estimators perform very poorly while the third one leads to an almost perfect recovery. The goal of the present paper is to propose a strategy which is able to estimate a regression function almost as well as the best linear procedure in a given family. Such a family may be obtained by considering, for instance, different values for the number of neighbors in nearest neighbor

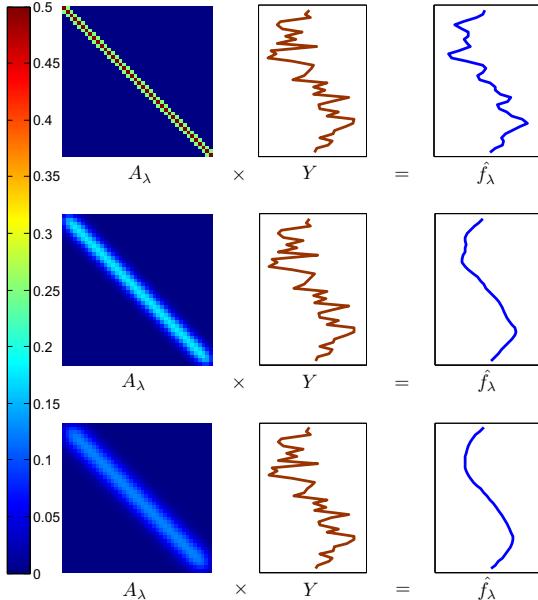


Fig. 1. The effect of the smoothing matrix A_λ on the resulting estimator. In this example, the true signal is the sine function over $[-\pi, \pi]$ and the three matrices represented in the leftmost column are some powers of one convolution matrix. Large powers correspond to stronger smoothing. It is clearly seen that the third matrix leads to an almost perfect recovery of the original signal.

smoothing. It is also possible to make vary the metric in which the proximity is measured.

We will mainly focus on the theoretical guarantees expressed in terms of oracle inequalities for the expected squared loss. Interestingly, despite the fact that several recent papers [1, 4, 18, 21] discuss the paradigm of competing against the best linear procedure from a given family, none of them provide oracle inequalities with leading constant equal to one. Furthermore, most existing results involve some constants depending on different parameters of the setup. In contrast, the oracle inequality that we prove herein is with leading constant one and admits a very simple formulation. It is established for a suitably symmetrized version of the exponentially weighted aggregate [16, 8, 14] with arbitrary prior (see Figure 2) and a temperature parameter which is not too small. The result is completely non-asymptotic and leads to asymptotically optimal residual term in the case where the sample size, as well as the cardinality of the family of competitors, tends to infinity.

More precisely, let \mathbf{f} be an unknown function defined on some set \mathcal{X} (called feature space) and taking values in \mathbb{R} . To fix the ideas, assume that \mathcal{X} is equipped with a metric d . We consider the setting where only noisy evaluations of the function \mathbf{f} at n points x_1, \dots, x_n of \mathcal{X} are available. The observations are then $\mathcal{D} = \{(x_1, Y_1), \dots, (x_n, Y_n)\}$. We are interested here in recovering the true values

$\mathbf{f}(x_i)$ for $i = 1, \dots, n$ based on the data \mathcal{D} . In this context, if we assume that \mathbf{f} is smooth in some sense, a common estimator of $\mathbf{f}(x_i)$ is given by the k -nearest neighbor (k NN) filter $\hat{\mathbf{f}}_{k,d}(x_i)$. In order to define it, let us denote by $j_{i,0}, j_{i,1}, \dots, j_{i,(n-1)}$ a permutation of $\{1, \dots, n\}$ that leads to a rearrangement of the design points x_j from the closest to x_i to the farthest, *i.e.*, $0 = d(x_i, x_{j_{i,0}}) \leq d(x_i, x_{j_{i,1}}) \leq \dots \leq d(x_i, x_{j_{i,(n-1)}})$. The k NN smoothing filter is then defined by

$$\hat{\mathbf{f}}_{k,d}(x_i) = \frac{1}{k} \sum_{m=0}^{k-1} Y_{j_{i,m}}. \quad (1)$$

In most applications, one can define different metrics d on the feature space and obtain different estimators of $\mathbf{f}(x_i)$ with very different statistical properties. The choice of the parameter k and the metric d that leads to the smallest possible estimation error is an important problem both from practical and theoretical viewpoints. A natural question arises: assume that we are given several metrics d_1, \dots, d_L on the feature space, is it possible to design a statistical procedure that estimates each of $\mathbf{f}(x_i)$ nearly as well as the best k NN-filter from the family $\{\hat{\mathbf{f}}_{k,d_\ell} : k = 1, \dots, n; \ell = 1, \dots, L\}$? We show that the answer to this question is affirmative, but there is a price to pay for not knowing the optimal metric and the optimal value of k . In the present work, we address this issues by aggregating the estimators $\hat{\mathbf{f}}_{k,d}$ over the set of all possible values of k and the metric d . Our results imply that the price to pay for not knowing the best values of these parameters is of the order $\log(L(n-1))/n$.

Note that the estimator (1) can be written as $\hat{\mathbf{f}}_{k,d}(x_i) = \sum_{j=1}^n a_{ij} Y_j$, with a_{ij} being equal to $1/k$ if $j \in \{j_{i,0}, \dots, j_{i,(k-1)}\}$ and 0 otherwise. Thus, the weights a_{ij} depend exclusively on the features x_1, \dots, x_n and not on the Y_i s. Therefore, the k NN filter is a particular instance of linear estimators defined by

$$\hat{\mathbf{f}} = \begin{bmatrix} \hat{\mathbf{f}}(x_1) \\ \vdots \\ \hat{\mathbf{f}}(x_n) \end{bmatrix} = A\mathbf{Y},$$

where A is a $n \times n$ weight matrix and $\mathbf{Y} = [Y_1, \dots, Y_n]^\top$ is the vector of observed responses. The main results of this paper hold for this general class of estimators under some condition on the weight matrix A . This condition is satisfied for a k NN estimator whatever the metric d and the parameter k are.

From the perspective of learning theory, oracle inequalities constitute a valuable theoretical tool for assessing the performance of procedures elaborated in the context of agnostic learning introduced by [20], see also [19] for a recent contribution to the subject. Note also that the problem of competing against the best procedure in a given family has been extensively studied in the context of online learning and prediction with expert advice [21, 9, 10, 5]. A remarkable connection between the results on online learning and the statistical oracle inequalities has been recently established by [17]. The case of linear estimators has been studied by [24, 26, 12] for projection matrices A and by [26, 12] for diagonal

Input: data vector $\mathbf{Y} \in \mathbb{R}^n$, $n \times n$ noise covariance matrix Σ and a family of linear smoothers $\{\hat{\mathbf{f}}_\lambda = \mathbf{A}_\lambda \mathbf{Y}; \lambda \in \Lambda\}$.

Output: estimator $\hat{\mathbf{f}}_{\text{SEWA}}$ of the true function f .

Parameter: prior probability distribution π on Λ , temperature parameter $\beta > 0$.

Strategy:

1. For every λ , compute the risk estimate $\hat{r}_\lambda^{\text{unb}} = \|\mathbf{Y} - \hat{\mathbf{f}}_\lambda\|_n^2 + \frac{2}{n} \text{Tr}(\Sigma \mathbf{A}_\lambda) - \frac{1}{n} \text{Tr}[\Sigma]$.
2. Define the probability distribution $\hat{\pi}(d\lambda) = \theta(\lambda)\pi(d\lambda)$ with $\theta(\lambda) \propto \exp(-n\hat{r}_\lambda^{\text{unb}}/\beta)$.
3. For every λ , build the symmetrized linear smoothers $\tilde{\mathbf{f}}_\lambda = (\mathbf{A}_\lambda + \mathbf{A}_\lambda^\top - \mathbf{A}_\lambda^\top \mathbf{A}_\lambda) \mathbf{Y}$.
4. Average out the symmetrized smoothers w.r.t. posterior $\hat{\pi}$: $\hat{\mathbf{f}}_{\text{SEWA}} = \int_{\Lambda} \tilde{\mathbf{f}}_\lambda \hat{\pi}(d\lambda)$.

Fig. 2. The symmetrized exponentially weighted aggregation strategy for competing against the best linear smoother in a given family

matrices A . However, these result do not cover several important classes of linear estimators including the kNN filter.

We should mention that the numerical experiments we have carried out on a number of synthetic datasets have shown that the symmetrized exponentially weighted aggregate performs as predicted by our theoretical result. Interestingly, these experiments show also that the standard (non-symmetrized) exponentially weighted aggregate is not worse than the symmetrized one. However, we are not able so far to provide theoretical guarantees for the non-symmetrized strategy.

Outline. The rest of the paper is organized as follows. We introduce the main notation along with a short background on oracle inequalities and on linear filtering in Section 2. The main contribution of the paper, a sharp oracle inequality for the symmetrized exponentially weighted aggregate, is stated in Section 3 while Section 4 contains some numerical results. Section 5 summarizes the content of the paper and provides some perspectives. The proofs are postponed to the Appendix.

2 Notation and Background

Throughout this work, we focus on the heteroscedastic regression model with Gaussian additive noise. More precisely, we assume that we are given a vector $\mathbf{Y} = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$ obeying the model:

$$y_i = f_i + \xi_i, \quad \text{for } i = 1, \dots, n, \tag{2}$$

where $\boldsymbol{\xi} = (\xi_1, \dots, \xi_n)^\top$ is a centered Gaussian random vector, $f_i = \mathbf{f}(x_i)$ where \mathbf{f} is an unknown function $\mathcal{X} \rightarrow \mathbb{R}$ and $x_1, \dots, x_n \in \mathcal{X}$ are deterministic points.

Here, no assumption is made on the set \mathcal{X} . Our objective is to recover the vector $\mathbf{f} = (f_1, \dots, f_n)$, often referred to as *signal*, based on the data y_1, \dots, y_n . In our work the noise covariance matrix $\Sigma = \mathbb{E}[\xi\xi^\top]$ is assumed to be finite and known, with a possible extension to the case of estimated covariance matrix discussed in Remark 5. We measure the performance of an estimator $\hat{\mathbf{f}}$ by its expected empirical quadratic loss: $r = \mathbb{E}(\|\mathbf{f} - \hat{\mathbf{f}}\|_n^2)$, where $\|\mathbf{f} - \hat{\mathbf{f}}\|_n^2 = \frac{1}{n} \sum_{i=1}^n (f_i - \hat{f}_i)^2$. We also denote by $\langle \cdot | \cdot \rangle_n$ the corresponding empirical inner product. For any matrix \mathbf{B} , $\|\mathbf{B}\|$ stands for the spectral norm of \mathbf{B} , *i.e.*, its largest singular value.

2.1 Oracle Inequalities

In this subsection we describe the paradigm of selection/aggregation of estimators in a data-driven manner from a given family of estimators. The task of aggregation consists in estimating \mathbf{f} by a suitable combination of the elements of a family of *constituent estimators* $\mathcal{F}_\Lambda = (\hat{\mathbf{f}}_\lambda)_{\lambda \in \Lambda} \in \mathbb{R}^n$, while the task of selection is just to choose a data-dependent value $\hat{\lambda}$ of λ for which the estimator $\hat{\mathbf{f}}_{\hat{\lambda}}$ is close to \mathbf{f} . The target objective of the selection/aggregation is to build an estimator $\hat{\mathbf{f}}_{\text{select}}/\hat{\mathbf{f}}_{\text{aggr}}$ that mimics the performance of the best constituent estimator, called *oracle* (because of its dependence on the unknown function f). In what follows, we assume that Λ is a measurable subset of \mathbb{R}^M , for some $M \in \mathbb{N}$.

The theoretical tool commonly used for evaluating the quality of an aggregation procedure is the oracle inequality (OI), generally written in the following form:

$$\mathbb{E}\|\hat{\mathbf{f}}_{\text{aggr}} - \mathbf{f}\|_n^2 \leq C_n \inf_{\lambda \in \Lambda} \left(\mathbb{E}\|\hat{\mathbf{f}}_\lambda - \mathbf{f}\|_n^2 \right) + R_n, \quad (3)$$

with *residual* term R_n tending to zero, and *leading constant* C_n being bounded. The OIs with leading constant one—called sharp OIs—are of central theoretical interest since they allow to bound the excess risk and to assess the aggregation-rate-optimality.

2.2 Nearest Neighbor Filtering

When the unknown function \mathbf{f} is smooth or can be well approximated by a smooth function, it is reasonable to estimate it by computing the moving averages or k -Nearest Neighbor (k NN) filters, see *e.g.* [15]. More precisely, let us fix an index i and consider the problem of estimating the value f_i of \mathbf{f} at x_i . Let x_{j_1}, \dots, x_{j_k} be the set of k points from $\{x_1, \dots, x_n\}$ which are at smallest distance (in some metric) from x_i . The idea of k NN filtering is to estimate the unknown value f_i by taking the average of k values Y_{j_ℓ} , $\ell = 1, \dots, k$. This approach is particularly popular, for instance, in stereo-vision for reconstructing 3D scenes from 3D point clouds.

A crucial point when estimating a function by k NN-filtering is the choice of the tuning parameter k . This parameter allows the user to control the trade-off between the bias and the variance of estimation. If the value of k is too small, the resulting estimator is very oscillating, whereas large values of k lead to over-smoothed estimators. Many strategies for selecting k in a data driven manner have been proposed

in the literature [25, 23, 22, 18, 1]. However, to the best of our knowledge, none of these procedures is proved to satisfy a sharp oracle inequality in the sense made precise in the previous section. In the present work, we propose a strategy—for which a sharp oracle inequality is established—based on data-driven aggregation of k NN filters rather than on (data-driven) selection of the parameter k .

2.3 General Linear Smoothing

More generally, we will focus on *linear estimators* $\hat{\mathbf{f}}_\lambda$, *i.e.*, estimators that are linear transforms of the data $\mathbf{Y} = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$. Using the convention that all vectors are one-column matrices, linear estimators can be defined by

$$\hat{\mathbf{f}}_\lambda = \mathbf{A}_\lambda \mathbf{Y}, \quad (4)$$

where the $n \times n$ real matrix \mathbf{A}_λ is deterministic. This means that the entries of \mathbf{A}_λ may depend on the points x_1, \dots, x_n but not on the data vector \mathbf{Y} . Let \mathbf{I}_n denote the identity matrix of size $n \times n$. It is well-known that the risk of the estimator (4) is given by

$$\mathbb{E}[\|\hat{\mathbf{f}}_\lambda - \mathbf{f}\|_n^2] = \|(\mathbf{A}_\lambda - \mathbf{I}_n)\mathbf{f}\|_n^2 + \frac{\text{Tr}(\mathbf{A}_\lambda \Sigma \mathbf{A}_\lambda^\top)}{n} \quad (5)$$

and that $\hat{r}_\lambda^{\text{unb}}$, defined by

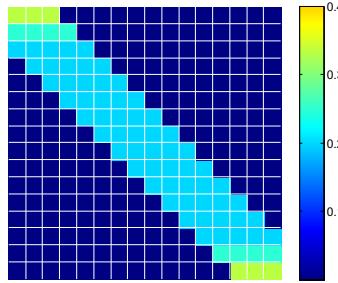
$$\hat{r}_\lambda^{\text{unb}} = \|\mathbf{Y} - \hat{\mathbf{f}}_\lambda\|_n^2 + \frac{2}{n} \text{Tr}(\Sigma \mathbf{A}_\lambda) - \frac{1}{n} \text{Tr}[\Sigma] \quad (6)$$

is an unbiased estimator of $r_\lambda = \mathbb{E}[\|\hat{\mathbf{f}}_\lambda - \mathbf{f}\|_n^2]$. In order to get a sharp oracle inequality with a simple residual term, we will need the following assumption.

[C(λ)] The matrix \mathbf{A}_λ satisfies $\text{Tr}(\Sigma \mathbf{A}_\lambda) \leq \text{Tr}(\Sigma \mathbf{A}_\lambda^\top \mathbf{A}_\lambda)$.

Let us present now several classes of widely used linear estimators for which this condition is satisfied.

1. The simplest class of matrices \mathbf{A}_λ for which condition **C(λ)** holds true are orthogonal projections. Indeed, if \mathbf{A}_λ is a projection matrix, it satisfies $\mathbf{A}_\lambda^\top \mathbf{A}_\lambda = \mathbf{A}_\lambda$ and, therefore, $\text{Tr}(\Sigma \mathbf{A}_\lambda) = \text{Tr}(\Sigma \mathbf{A}_\lambda^\top \mathbf{A}_\lambda)$.
2. If the matrix Σ is diagonal, then a sufficient condition for **C(λ)** is $a_{ii} \leq \sum_{j=1}^n a_{ji}^2$. Consequently, for the matrices having only zeros on the main diagonal **C(λ)** holds true. For instance, the k NN filter in which the weight of the observation Y_i is replaced by zero, *i.e.*, $a_{ij} = \mathbf{1}_{j \in \{j_{i,1}, \dots, j_{i,k}\}}/k$ satisfies this condition.
3. Under a little bit more stringent assumption of homoscedasticity, *i.e.*, when $\Sigma = \sigma^2 \mathbf{I}_n$, if the matrices \mathbf{A}_λ are such that all the non-zero elements of each row are equal and sum up to one (or a quantity larger than one) then $\text{Tr}(\mathbf{A}_\lambda) = \text{Tr}(\mathbf{A}_\lambda^\top \mathbf{A}_\lambda)$ and **C(λ)** is fulfilled. A notable example of linear estimators that satisfy this condition are Nadaraya-Watson estimators with rectangular kernel and nearest neighbor filters. Below is a visual illustration of a matrix defining a Nadaraya-Watson estimator:



3 Main Result

Let $r_\lambda = \mathbb{E}[\|\hat{\mathbf{f}}_\lambda - \mathbf{f}\|_n^2]$ denote the risk of the estimator $\hat{\mathbf{f}}_\lambda$, for any $\lambda \in \Lambda$, and let \hat{r}_λ be an estimator of r_λ . For any probability distribution π over the set Λ and for any $\beta > 0$, we define the probability measure of exponential weights, $\hat{\pi}$, by the following formula: $\hat{\pi}(d\lambda) = \theta(\lambda)\pi(d\lambda)$ with

$$\theta(\lambda) = \frac{\exp(-n\hat{r}_\lambda/\beta)}{\int_{\Lambda} \exp(-n\hat{r}_\omega/\beta)\pi(d\omega)}. \quad (7)$$

The corresponding exponentially weighted aggregate, henceforth denoted by $\hat{\mathbf{f}}_{\text{EWA}}$, is the expectation of the $\hat{\mathbf{f}}_\lambda$ w.r.t. the probability measure $\hat{\pi}$:

$$\hat{\mathbf{f}}_{\text{EWA}} = \int_{\Lambda} \hat{\mathbf{f}}_\lambda \hat{\pi}(d\lambda). \quad (8)$$

It is convenient and customary to use the terminology of Bayesian statistics: the measure π is called *prior*, the measure $\hat{\pi}$ is called *posterior* and the aggregate $\hat{\mathbf{f}}_{\text{EWA}}$ is then the *posterior mean*. The parameter β will be referred to as the *temperature parameter*. In the framework of aggregating statistical procedures, the use of such an aggregate can be traced back to [16].

The density $\theta(\cdot)$ assigns weights to the estimators according to their performance, measured in terms of the risk estimate \hat{r}_λ . The temperature parameter reflects the confidence we have in this criterion: if $\beta \approx 0$ the posterior concentrates on the estimators achieving the smallest value for \hat{r}_λ , whereas if $\beta \rightarrow +\infty$ then the posterior approaches to the prior π , and the data do not modify our confidence in the estimators. It should also be noted that averaging w.r.t. the posterior $\hat{\pi}$ is not the only way of constructing an estimator of \mathbf{f} based on $\hat{\pi}$; some alternative randomized estimators have been studied, for instance, in [2].

To state our main results, we denote by \mathcal{P}_{Λ} the set of all probability measures on Λ and by $\mathcal{K}(p, p')$ the Kullback-Leibler divergence between two probability measures $p, p' \in \mathcal{P}_{\Lambda}$:

$$\mathcal{K}(p, p') = \begin{cases} \int_{\Lambda} \log\left(\frac{dp}{dp'}(\lambda)\right)p(d\lambda) & \text{if } p \ll p', \\ +\infty & \text{otherwise.} \end{cases}$$

Theorem 1. Let $\{\mathbf{A}_\lambda : \lambda \in \Lambda\}$ be any family of $n \times n$ matrices satisfying condition $\mathbf{C}(\lambda)$ on a set of π -measure one. Let $\hat{\mathbf{f}}_{\text{SEWA}}$ denote the symmetrized exponentially weighted aggregate, i.e. the exponentially weighted aggregate acting on symmetrized estimators $\hat{\mathbf{f}}_\lambda = (\mathbf{A}_\lambda + \mathbf{A}_\lambda^\top - \mathbf{A}_\lambda^\top \mathbf{A}_\lambda) \mathbf{Y}$ with the weights (7) defined via the risk estimate $\hat{r}_\lambda^{\text{unb}}$. Then, for every $\beta \geq 4\|\Sigma\|$, it holds that

$$\mathbb{E}[\|\hat{\mathbf{f}}_{\text{SEWA}} - \mathbf{f}\|_n^2] \leq \inf_{p \in \mathcal{P}_\Lambda} \left\{ \int_\Lambda \mathbb{E}[\|\hat{\mathbf{f}}_\lambda - \mathbf{f}\|_n^2] p(d\lambda) + \frac{\beta}{n} \mathcal{K}(p, \pi) \right\}.$$

A first observation that one can make is that, in the particular case of a finite collection of projection estimators (i.e., $\mathbf{A}_\lambda = \mathbf{A}_\lambda^\top = \mathbf{A}_\lambda^2$ for every λ) this result reduces to Corollary 6 in [24]. Furthermore, Theorem II handles the general noise covariances while [24] deals only with i.i.d. Gaussian noise.

Note also that the result of Theorem II applies to the estimator $\hat{\mathbf{f}}_{\text{EWA}}$ that uses the full knowledge of the covariance matrix Σ . Indeed, even if for the choice of β only an upper bound on the spectral norm of Σ is required, the entire matrix Σ enters in the definition of the unbiased risk $\hat{r}_\lambda^{\text{unb}}$ that is used for defining $\hat{\mathbf{f}}_{\text{SEWA}}$. We will discuss in Remark 5 some extensions of the proposed methodology to the case of unknown Σ .

Remark 1. We decided in this paper to focus on the case of Gaussian errors, in order to put the emphasis on the possibility of efficiently aggregating broad families of *linear estimators* without spending time and space on other technical aspects. The result stated in this section can be generalized to some other noise distributions by following the approach developed in [13].

Remark 2. We prove a result that is stronger than the one stated in Theorem II. In particular, it holds for any matrices \mathbf{A}_λ and boils down to the elegant inequality stated in Theorem II when condition $\mathbf{C}(\lambda)$ is π -a.e. satisfied. The precise form of this more general result is the following. Let $\hat{\mathbf{f}}_{\text{SEWA}}$ denote the aggregate defined in Figure 2. Then, for every $\beta \geq 4\|\Sigma\|$, the risk $\mathbb{E}[\|\hat{\mathbf{f}}_{\text{SEWA}} - \mathbf{f}\|_n^2]$ of $\hat{\mathbf{f}}_{\text{SEWA}}$ is bounded from above by

$$\inf_{p \in \mathcal{P}_\Lambda} \left\{ \int_\Lambda \mathbb{E}[\|\hat{\mathbf{f}}_\lambda - \mathbf{f}\|_n^2] p(d\lambda) + \frac{\beta}{n} \mathcal{K}(p, \pi) \right\} + R_n \quad (9)$$

with the residual term $R_n = \frac{\beta}{n} \log \left[\int_\Lambda e^{\frac{\beta}{n} \text{Tr}[\Sigma(\mathbf{A}_\lambda - \mathbf{A}_\lambda^\top \mathbf{A}_\lambda)]} \pi(d\lambda) \right]$.

Remark 3. Using the previous remark, one can also get the risk bound (9), when condition $\mathbf{C}(\lambda)$ is only approximately satisfied. More precisely, if condition $\mathbf{C}(\lambda)$ is replaced by :

[$\mathbf{C}(\lambda, \varepsilon)$] The matrix \mathbf{A}_λ satisfies $\text{Tr}(\Sigma \mathbf{A}_\lambda) \leq \text{Tr}(\Sigma \mathbf{A}_\lambda^\top \mathbf{A}_\lambda) + \varepsilon$,

then the residual term R_n in Inequality (9) simply becomes $\frac{2\varepsilon}{n}$.

In order to demonstrate that Theorem II can be reformulated in terms of an OI as defined by (3), let us consider the case when the prior π is discrete. That is, we assume that $\pi(\Lambda_0) = 1$ for a countable set $\Lambda_0 \subset \Lambda$. Without loss of generality, we assume that $\Lambda_0 = \mathbb{N}$. Then, the following result holds true.

Proposition 1. If π is supported by \mathbb{N} and condition $\mathbf{C}(\lambda)$ is satisfied for every $\lambda \in \mathbb{N}$, then the aggregate $\hat{\mathbf{f}}_{\text{SEWA}}$ satisfies the inequality

$$\mathbb{E}[\|\hat{\mathbf{f}}_{\text{SEWA}} - \mathbf{f}\|_n^2] \leq \inf_{\lambda: \pi_\lambda > 0} \left\{ \mathbb{E}\|\hat{\mathbf{f}}_\lambda - \mathbf{f}\|_n^2 + \frac{\beta \log(1/\pi_\lambda)}{n} \right\} \quad (10)$$

provided that $\beta \geq 4\|\Sigma\|$.

Proof. It suffices to apply Theorem 1 and to bound the right hand side from above by the minimum over all Dirac measures $p = \delta_\lambda$ with λ such that $\pi_\lambda > 0$.

This inequality can be compared to Corollary 2 in Section 4.3 of [4]. Our inequality has the advantage of being sharp, *i.e.*, having factor one both in front of the expectation of the LHS of (10) and in front of the inf of the RHS. To the best of our knowledge, there is no other result in the literature providing such a sharp OI for linear estimators which are not of projection type. In particular, in [4] the risk in the LHS of the OI is multiplied by a constant which is smaller than one and depends on different parameters of the problem. It should be noted, however, that we consider the noise covariance matrix as known, whereas [4] estimates the noise covariance along with the regression function.

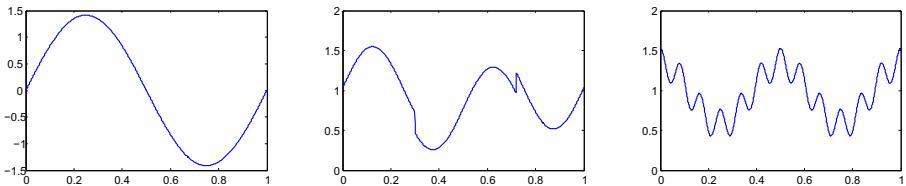


Fig. 3. Three test signals used in experimental evaluation. From left to right : the sine function, HeaviSine and Wave functions [7].

Remark 4. A particular case of Proposition 1 is the situation where π is the uniform probability over a finite set of cardinality M . In such a situation, the remainder term in (10) becomes of the form $(\beta \log M)/n$. The rate $(\log M)/n$ of the remainder term in the OI has been proven [28] unavoidable in the context of aggregating data-independent estimators. By similar arguments, it is possible to prove that this rate is optimal in the case of aggregating linear smoothers as well.

Remark 5. The symmetrized exponentially weighted aggregate $\hat{\mathbf{f}}_{\text{SEWA}}$ is easily extended to handle the more realistic situation where an unbiased estimate $\hat{\Sigma}$, independent of \mathbf{Y} , of the covariance matrix Σ is available. Simply replace Σ by $\hat{\Sigma}$ in the definition of the unbiased risk estimate (6). When the matrices \mathbf{A}_λ satisfy π -a.e. condition $\mathbf{C}(\lambda)$, it is easy to see that the claim of Theorem 1 remains valid. Of course, the condition $\beta \geq 4\|\Sigma\|$ should be replaced by $\beta \geq 4\|\hat{\Sigma}\|$ and β should be replaced by $\mathbb{E}[\beta]$ in the right hand side of the oracle inequality.

4 Numerical Experiments

We have implemented the symmetrized exponentially weighted aggregate (SEWA) in Matlab in the case of combining k NN filters with varying values of k . Along with SEWA we have also implemented the classical exponentially weighted aggregate (EWA) as defined for instance in [24, 14] and the empirical risk minimization (ERM) algorithm, the latter consisting in choosing the value of k minimizing the unbiased estimator of the risk (6). Following the philosophy of reproducible research, a toolbox containing the code we used for getting the results reported in this section will be made available by the date of the conference at the authors' home pages.

In our experiments, we compared the aforementioned three strategies, ERM, EWA and SEWA, on three common 1D signals depicted in Figure 3. Each signal has been beforehand normalized to have an L^2 norm equal to one. We have chosen several sample sizes $n \in \{30, 50, 100\}$ and noise levels $\sigma^2 \in \{0.2, 0.5, 1, 1.5, 2\}$ and randomly generated the data vector $\mathbf{Y} = (Y_1, \dots, Y_n)$ by the formula $Y_i = \mathbf{f}(i/n) + \epsilon_i$, where $(\epsilon_1, \dots, \epsilon_n)$ is a Gaussian random vector $\mathcal{N}(0, \sigma^2 \mathbf{I}_n)$. We then computed the three estimators ERM, EWA and SEWA and repeated the experiment 10^4 times. As preliminary estimators we used the k NN filters with $k \in \{1, \dots, [n/2]\}$. The prior was chosen to be uniform and the temperature parameter is the one suggested by the theory: $\beta = 4\sigma^2$. The medians and the inter-quartile ranges of the errors¹ $\|\hat{\mathbf{f}}_\bullet - \mathbf{f}\|_2$ are summarized in Tables 1, 2 and 3 below.

Table 1. Sine function: the values of the median error and the inter-quartile range (in parentheses) over 10^4 trials are reported

	$n = 30$			$n = 50$			$n = 100$		
	ERM	EWA	SEWA	ERM	EWA	SEWA	ERM	EWA	SEWA
$\sigma^2 = 0.2$	1.4397 (0.40)	1.3906 (0.40)	1.3469 (0.35)	1.5290 (0.40)	1.4685 (0.39)	1.4663 (0.38)	1.6768 (0.40)	1.5984 (0.38)	1.6471 (0.38)
$\sigma^2 = 0.5$	2.0301 (0.57)	1.8806 (0.48)	1.7861 (0.53)	2.1395 (0.65)	2.0800 (0.59)	2.0086 (0.56)	2.3634 (0.63)	2.2661 (0.62)	2.2786 (0.59)
$\sigma^2 = 1$	2.4966 (0.69)	2.2161 (0.61)	2.1933 (0.71)	2.8026 (0.81)	2.5501 (0.67)	2.4487 (0.74)	3.0561 (0.93)	2.9287 (0.83)	2.8590 (0.81)
$\sigma^2 = 1.5$	2.7930 (0.94)	2.4966 (0.83)	2.5046 (0.96)	3.1521 (0.94)	2.8125 (0.84)	2.7660 (0.95)	3.5679 (1.09)	3.3088 (0.92)	3.2167 (0.96)
$\sigma^2 = 2$	3.0113 (1.08)	2.7180 (1.02)	2.7793 (1.17)	3.3930 (1.10)	3.0757 (0.93)	3.0413 (1.06)	3.9748 (1.19)	3.5854 (1.00)	3.4970 (1.09)

A first observation is that the aggregation strategies, EWA and SEWA, are always better than the selection strategy ERM. This is essentially explained

¹ In this expression the norm is the classical Euclidean one and $\hat{\mathbf{f}}_\bullet$ is either one of the estimators ERM, EWA or SEWA.

Table 2. HeaviSine function [7]: the values of the median error and the inter-quartile range (in parentheses) over 10^4 trials are reported

	$n = 30$			$n = 50$			$n = 100$		
	ERM	EWA	SEWA	ERM	EWA	SEWA	ERM	EWA	SEWA
$\sigma^2 = 0.2$	1.6552 (0.37)	1.5906 (0.36)	1.5708 (0.35)	1.8157 (0.37)	1.7274 (0.39)	1.7306 (0.38)	2.0170 (0.39)	1.9359 (0.37)	1.9921 (0.39)
$\sigma^2 = 0.5$	2.2783 (0.55)	2.1604 (0.57)	2.0845 (0.58)	2.4834 (0.59)	2.3370 (0.54)	2.2589 (0.57)	2.7984 (0.62)	2.6620 (0.59)	2.6611 (0.59)
$\sigma^2 = 1$	2.9039 (0.82)	2.7275 (0.81)	2.6416 (0.85)	3.1558 (0.85)	2.9446 (0.83)	2.8783 (0.84)	3.5533 (0.86)	3.3284 (0.80)	3.2715 (0.82)
$\sigma^2 = 1.5$	3.3554 (1.08)	3.1526 (0.99)	3.0878 (0.97)	3.5758 (1.02)	3.3576 (0.95)	3.2583 (1.00)	4.0708 (1.05)	3.7886 (0.97)	3.7106 (1.00)
$\sigma^2 = 2$	3.7266 (1.34)	3.4729 (1.19)	3.4443 (1.22)	4.0147 (1.30)	3.7368 (1.23)	3.6694 (1.24)	4.4888 (1.24)	4.1560 (1.13)	4.0723 (1.16)

Table 3. Wave function: the values of the median error and the inter-quartile range (in parentheses) over 10^4 trials are reported

	$n = 30$			$n = 50$			$n = 100$		
	ERM	EWA	SEWA	ERM	EWA	SEWA	ERM	EWA	SEWA
$\sigma^2 = 0.2$	1.4340 (0.37)	1.3814 (0.29)	1.3724 (0.30)	1.5887 (0.41)	1.5725 (0.33)	1.5580 (0.33)	1.9720 (0.34)	1.8696 (0.30)	1.8612 (0.33)
$\sigma^2 = 0.5$	1.8300 (0.45)	1.6868 (0.41)	1.7159 (0.47)	2.1004 (0.53)	1.9571 (0.41)	1.9608 (0.47)	2.4045 (0.67)	2.3730 (0.49)	2.3462 (0.52)
$\sigma^2 = 1$	2.1727 (0.74)	2.0073 (0.65)	2.0976 (0.73)	2.4719 (0.69)	2.2784 (0.60)	2.3351 (0.68)	2.9898 (0.77)	2.7755 (0.58)	2.7716 (0.66)
$\sigma^2 = 1.5$	2.4395 (1.00)	2.2637 (0.84)	2.4013 (0.94)	2.7554 (0.93)	2.5266 (0.77)	2.6331 (0.89)	3.2993 (0.88)	3.0282 (0.72)	3.0761 (0.83)
$\sigma^2 = 2$	2.6845 (1.23)	2.5068 (1.01)	2.6809 (1.12)	2.9950 (1.15)	2.7495 (0.94)	2.8961 (1.06)	3.5428 (1.05)	3.2290 (0.86)	3.3133 (0.99)

by a relative lack of stability of selection strategies thoroughly discussed in [6]. A second observation is that there is no clear winner among the aggregation strategies EWA and SEWA. Both of them are quite accurate with very little difference in the error of estimation. This raises the following question: is it possible to prove a sharp oracle inequality for the standard EWA without applying the symmetrization trick? To date, we are unable to answer this question.

It is important to stress that the medians reported in Tables 13 are those of estimation errors without normalization by the sample size n . Therefore, it is quite natural that these errors increase with n (more and more parameters are estimated). It is however clear from the reported results that the non-normalized accuracy increases very slowly when n increases. This is in agreement with our theoretical result stating that the error increases at most logarithmically.

5 Conclusion and Outlook

We have suggested a new strategy for aggregating linear smoothers in order to denoise a signal corrupted by an additive Gaussian noise. We proved a sharp oracle inequality for the proposed strategy, termed SEWA for symmetrized exponentially weighted aggregation. A few experimental results are also reported that allow to illustrate our theoretical result and to quantify the advantage of aggregation as compared to selection.

The SEWA results may have profitable application to classification and pattern recognition. As proved in [3], fast rates in classification can be obtained by plugging-in efficient regression estimators. We are experimenting with the use of a procedure analogous to SEWA to perform binary classification. The results, to date, have been as encouraging as in the regression case.

Acknowledgments. This work has been partially supported by ANR grant Parcimonie.

References

- [1] Arlot, S., Bach, F.: Data-driven calibration of linear estimators with minimal penalties. In: NIPS, pp. 46–54 (2009)
- [2] Audibert, J.-Y.: Fast learning rates in statistical inference through aggregation. *Ann. Statist.* 37(4), 1591–1646 (2009)
- [3] Audibert, J.-Y., Tsybakov, A.B.: Fast learning rates for plug-in classifiers. *Ann. Statist.* 35(2), 608–633 (2007)
- [4] Baraud, Y., Giraud, C., Huet, S.: Estimator selection in the gaussian setting (2010) (submitted)
- [5] Ben-David, S., Pal, D., Shalev-Shwartz, S.: Agnostic online learning. In: COLT (2009)
- [6] Breiman, L.: Better subset regression using the nonnegative garrote. *Technometrics* 37, 373–384 (1995)
- [7] Buckheit, J.B., Donoho, D.L.: Wavelab and reproducible research. In: Wavelets and Statistics. Lect. Notes Statist., vol. 103, pp. 55–81. Springer, New York (1995)
- [8] Catoni, O.: Statistical learning theory and stochastic optimization. Lecture Notes in Mathematics, vol. 1851. Springer, Berlin (2004)
- [9] Cesa-Bianchi, N., Lugosi, G.: Prediction, learning, and games. Cambridge University Press, Cambridge (2006)
- [10] Cesa-Bianchi, N., Mansour, Y., Stoltz, G.: Improved second-order bounds for prediction with expert advice. *Mach. Learn.* 66, 321–352 (2007)
- [11] Cornillon, P.-A., Hengartner, N., Matzner-Løber, E.: Recursive bias estimation for multivariate regression smoothers (2009) (submitted)
- [12] Dalalyan, A.S., Salmon, J.: Sharp oracle inequalities for aggregation of affine estimators. technical report, arXiv:1104.3969v2 [math.ST] (2011)
- [13] Dalalyan, A.S., Tsybakov, A.B.: Aggregation by exponential weighting, sharp pac-bayesian bounds and sparsity. *Mach. Learn.* 72(1-2), 39–61 (2008)
- [14] Dalalyan, A.S., Tsybakov, A.B.: Sparse regression learning by aggregation and Langevin Monte-Carlo. In: COLT (2009)
- [15] Devroye, L., Györfi, L., Lugosi, G.: A probabilistic theory of pattern recognition. Applications of Mathematics, vol. 31. Springer, New York (1996)
- [16] George, E.I.: Minimax multiple shrinkage estimation. *Ann. Statist.* 14(1), 188–205 (1986)

- [17] Gerchinovitz, S.: Sparsity regret bounds for individual sequences in online linear regression (submitted, 2011)
- [18] Goldenshluger, A., Lepski, O.V.: Universal pointwise selection rule in multivariate function estimation. Bernoulli 14(4), 1150–1190 (2008)
- [19] Kalai, A., Klivans, A., Mansour, Y., Servedio, R.: Agnostically learning halfspaces. SIAM J. Comput. 37(6), 1777–1805 (2008)
- [20] Kearns, M.J., Schapire, R.E., Sellie, L.: Toward efficient agnostic learning. Machine Learning 17(2-3), 115–141 (1994)
- [21] Kivinen, J., Warmuth, M.K.: Averaging expert predictions. In: Fischer, P., Simon, H.U. (eds.) EuroCOLT 1999. LNCS (LNAI), vol. 1572, pp. 153–167. Springer, Heidelberg (1999)
- [22] Lafferty, J., Wasserman, L.: Rodeo: sparse, greedy nonparametric regression. Ann. Statist. 36(1), 28–63 (2008)
- [23] Lepski, O.V., Mammen, E., Spokoiny, V.G.: Optimal spatial adaptation to inhomogeneous smoothness: an approach based on kernel estimates with variable bandwidth selectors. Ann. Statist. 25(3), 929–947 (1997)
- [24] Leung, G., Barron, A.R.: Information theory and mixing least-squares regressions. IEEE Trans. Inf. Theory 52(8), 3396–3410 (2006)
- [25] Li, K.-C.: From Stein’s unbiased risk estimates to the method of generalized cross validation. Ann. Statist. 13(4), 1352–1377 (1985)
- [26] Salmon, J., Dalalyan, A.S.: Optimal aggregation of affine estimators. In: COLT (2011)
- [27] Stein, C.M.: Estimation of the mean of a multivariate distribution. In: Proc. Prague Symp. Asymptotic Statist (1973)
- [28] Tsybakov, A.B.: Optimal rates of aggregation. In: COLT, pp. 303–313 (2003)

A Proof of Theorem 1

The proof of our main result relies on the well-known Stein lemma [27] providing an unbiased risk estimate for any estimator that depends sufficiently smoothly on the data vector \mathbf{Y} . For the convenience of the reader, we recall Stein’s lemma in the case of heteroscedastic Gaussian regression.

Lemma 1. *Let \mathbf{Y} be a random vector drawn from the Gaussian distribution $\mathcal{N}_n(\mathbf{f}, \Sigma)$. If the estimator $\hat{\mathbf{f}}$ is a.e. differentiable in \mathbf{Y} and the elements of the matrix $\nabla \cdot \hat{\mathbf{f}}^\top := (\partial_i \hat{f}_j)$ have finite first moment, then $\hat{r}_\Sigma = \|\mathbf{Y} - \hat{\mathbf{f}}\|_n^2 + \frac{2}{n} \text{Tr}[\Sigma(\nabla \cdot \hat{\mathbf{f}}^\top)] - \frac{1}{n} \text{Tr}[\Sigma]$, is an unbiased estimate of r , i.e., $\mathbb{E}\hat{r}_\Sigma = r$. Moreover, if $\hat{\Sigma}$ is an unbiased estimator of Σ such that \mathbf{Y} and $\hat{\Sigma}$ are independent, then*

$$\hat{r} = \|\mathbf{Y} - \hat{\mathbf{f}}\|_n^2 + \frac{2}{n} \text{Tr}[\hat{\Sigma}(\nabla \cdot \hat{\mathbf{f}}^\top)] - \frac{1}{n} \text{Tr}[\hat{\Sigma}], \quad (11)$$

is an unbiased estimator of the risk r as well.

We apply Stein’s lemma to the estimator $\hat{\mathbf{f}}_\lambda = \mathbf{A}_\lambda \mathbf{Y}$, where \mathbf{A}_λ is an $n \times n$ matrix. We get that $\hat{r}_{\lambda, \Sigma}^{\text{unb}} = \|\mathbf{Y} - \hat{\mathbf{f}}_\lambda\|_n^2 + \frac{2}{n} \text{Tr}[\Sigma \mathbf{A}_\lambda] - \frac{1}{n} \text{Tr}[\Sigma]$ is an unbiased estimator of the risk $r_\lambda = \mathbb{E}[\|\hat{\mathbf{f}}_\lambda - \mathbf{f}\|_n^2] = \|(\mathbf{A}_\lambda - \mathbf{I}_n)\mathbf{f}\|_n^2 + \frac{1}{n} \text{Tr}[\mathbf{A}_\lambda \Sigma \mathbf{A}_\lambda^\top]$.

Furthermore, if $\widehat{\Sigma}$ is an unbiased estimator of Σ then $\hat{r}_\lambda^{\text{unb}} = \|\mathbf{Y} - \tilde{\mathbf{f}}_\lambda\|_n^2 + \frac{2}{n} \text{Tr}[\widehat{\Sigma} \mathbf{A}_\lambda] - \frac{1}{n} \text{Tr}[\widehat{\Sigma}]$ is also an unbiased estimator of r_λ .

Prior to proceeding with the proof of main theorems, we prove an important auxiliary result which is the central ingredient of the proof for our main result.

Lemma 2. *Let assumptions of Lemma 1 be satisfied. Let $\{\tilde{\mathbf{f}}_\lambda : \lambda \in \Lambda\}$ be a family of estimators of \mathbf{f} and $\{\tilde{r}_\lambda : \lambda \in \Lambda\}$ a family of risk estimates such that the mapping $\mathbf{Y} \mapsto (\tilde{\mathbf{f}}_\lambda, \tilde{r}_\lambda)$ is a.e. differentiable $\forall \lambda \in \Lambda$. Let $\hat{r}_\lambda^{\text{unb}}$ be the unbiased risk estimate of $\tilde{\mathbf{f}}_\lambda$ given by Stein's lemma (cf. Eq. (1)).*

1. *For every $\mu \in \mathcal{P}_\Lambda$ and for any $\beta > 0$, the estimator $\tilde{\mathbf{f}}_{\text{EWA}}$ defined as the average of $\tilde{\mathbf{f}}_\lambda$ w.r.t. the probability measure $\hat{\mu}(\mathbf{Y}, d\lambda) = \theta(\mathbf{Y}, \lambda)\mu(d\lambda)$ with $\theta(\mathbf{Y}, \lambda) \propto \exp\{-n\tilde{r}_\lambda(\mathbf{Y})/\beta\}$ admits*

$$\hat{r}_{\text{EWA}} = \int_{\Lambda} \left(\hat{r}_\lambda^{\text{unb}} - \|\tilde{\mathbf{f}}_\lambda - \tilde{\mathbf{f}}_{\text{EWA}}\|_n^2 - \frac{2n}{\beta} \langle \nabla_{\mathbf{Y}} \tilde{r}_\lambda | \widehat{\Sigma} (\tilde{\mathbf{f}}_\lambda - \tilde{\mathbf{f}}_{\text{EWA}}) \rangle_n \right) \hat{\mu}(d\lambda)$$

as unbiased estimator of the risk.

2. *If furthermore $\tilde{r}_\lambda \geq \hat{r}_\lambda^{\text{unb}}$, $\forall \lambda \in \Lambda$ and $\int_{\Lambda} \langle \nabla \tilde{r}_\lambda | \widehat{\Sigma} (\tilde{\mathbf{f}}_\lambda - \tilde{\mathbf{f}}_{\text{EWA}}) \rangle_n \hat{\mu}(d\lambda) \geq -a \int_{\Lambda} \|\tilde{\mathbf{f}}_\lambda - \tilde{\mathbf{f}}_{\text{EWA}}\|_n^2 \hat{\mu}(d\lambda)$ for some random $a > 0$ independent of \mathbf{Y} , then for every $\beta \geq 2na$ it holds that*

$$\mathbb{E}[\|\tilde{\mathbf{f}}_{\text{EWA}} - \mathbf{f}\|_n^2] \leq \inf_{p \in \mathcal{P}_\Lambda} \left\{ \int_{\Lambda} \mathbb{E}[\tilde{r}_\lambda] p(d\lambda) + \frac{\mathbb{E}[\beta] \mathcal{K}(p, \mu)}{n} \right\}.$$

Proof. According to the Lemma 1, the quantity

$$\hat{r}_{\text{EWA}} = \|\mathbf{Y} - \tilde{\mathbf{f}}_{\text{EWA}}\|_n^2 + \frac{2}{n} \text{Tr}[\widehat{\Sigma}(\mathbf{V} \cdot \tilde{\mathbf{f}}_{\text{EWA}}(\mathbf{Y}))] - \frac{1}{n} \text{Tr}[\widehat{\Sigma}] \quad (12)$$

is an unbiased estimate of the risk $r_n = \mathbb{E}(\|\tilde{\mathbf{f}}_{\text{EWA}} - \mathbf{f}\|_n^2)$. Using simple algebra, one checks that

$$\|\mathbf{Y} - \tilde{\mathbf{f}}_{\text{EWA}}\|_n^2 = \int_{\Lambda} \left(\|\mathbf{Y} - \tilde{\mathbf{f}}_\lambda\|_n^2 - \|\tilde{\mathbf{f}}_\lambda - \tilde{\mathbf{f}}_{\text{EWA}}\|_n^2 \right) \hat{\mu}(d\lambda). \quad (13)$$

By interchanging the integral and differential operators, we get the following relation: $\partial_{y_i} \tilde{\mathbf{f}}_{\text{EWA}, j} = \int_{\Lambda} \{(\partial_{y_j} \tilde{\mathbf{f}}_\lambda^j(\mathbf{Y})) \theta(\mathbf{Y}, \lambda) + \tilde{\mathbf{f}}_\lambda^j(\mathbf{Y}) (\partial_{y_i} \theta(\mathbf{Y}, \lambda))\} \mu(d\lambda)$. This equality, combined with Equations (12) and (13) implies that

$$\hat{r}_{\text{EWA}} = \int_{\Lambda} (\hat{r}_\lambda^{\text{unb}} - \|\tilde{\mathbf{f}}_\lambda - \tilde{\mathbf{f}}_{\text{EWA}}\|_n^2) \hat{\mu}(d\lambda) + \frac{2}{n} \int_{\Lambda} \text{Tr}[\widehat{\Sigma} \tilde{\mathbf{f}}_\lambda \nabla_{\mathbf{Y}} \theta(\mathbf{Y}, \lambda)^\top] \mu(d\lambda).$$

Taking into account the fact that the differentiation and the integration can be interchanged, $\int_{\Lambda} \tilde{\mathbf{f}}_{\text{EWA}}(\nabla_{\mathbf{Y}} \theta(\mathbf{Y}, \lambda))^\top \mu(d\lambda) = \tilde{\mathbf{f}}_{\text{EWA}} \nabla_{\mathbf{Y}} (\int_{\Lambda} \theta(\mathbf{Y}, \lambda) \mu(d\lambda)) = 0$, and we come up with the following expression for the unbiased risk estimate:

$$\begin{aligned} \hat{r}_{\text{EWA}} &= \int_{\Lambda} \left(\hat{r}_\lambda^{\text{unb}} - \|\tilde{\mathbf{f}}_\lambda - \tilde{\mathbf{f}}_{\text{EWA}}\|_n^2 + 2 \langle \nabla_{\mathbf{Y}} \log \theta(\lambda) | \widehat{\Sigma} (\tilde{\mathbf{f}}_\lambda - \tilde{\mathbf{f}}_{\text{EWA}}) \rangle_n \right) \hat{\mu}(d\lambda) \\ &= \int_{\Lambda} \left(\hat{r}_\lambda^{\text{unb}} - \|\tilde{\mathbf{f}}_\lambda - \tilde{\mathbf{f}}_{\text{EWA}}\|_n^2 - 2n\beta^{-1} \langle \nabla_{\mathbf{Y}} \tilde{r}_\lambda | \widehat{\Sigma} (\tilde{\mathbf{f}}_\lambda - \tilde{\mathbf{f}}_{\text{EWA}}) \rangle_n \right) \hat{\mu}(d\lambda). \end{aligned}$$

This completes the proof of the first assertion of the lemma.

To prove the second assertion, let us observe that under the required condition and in view of the first assertion, for every $\beta \geq 2na$ it holds that $\hat{r}_{\text{EWA}} \leq \int_A \tilde{r}_\lambda^{\text{unb}} \hat{\mu}(d\lambda) \leq \int_A \tilde{r}_\lambda \hat{\mu}(d\lambda) \leq \int_A \tilde{r}_\lambda \hat{\mu}(d\lambda) + \frac{\beta}{n} \mathcal{K}(\hat{\mu}, \mu)$. To conclude, it suffices to remark that $\hat{\mu}$ is the probability measure minimizing the criterion $\int_A \tilde{r}_\lambda p(d\lambda) + \frac{\beta}{n} \mathcal{K}(p, \mu)$ among all $p \in \mathcal{P}_A$. Thus, for every $p \in \mathcal{P}_A$, it holds that $\hat{r}_{\text{EWA}} \leq \int_A \tilde{r}_\lambda p(d\lambda) + \frac{\beta}{n} \mathcal{K}(p, \mu)$. Taking the expectation of both sides, the desired result follows.

Proof of Remark 2 and Theorem 1

Let now $\tilde{\mathbf{f}}_\lambda = \tilde{\mathbf{A}}_\lambda \mathbf{Y}$ with a symmetric $\tilde{\mathbf{A}}_\lambda = \mathbf{A}_\lambda + \mathbf{A}_\lambda^\top - \mathbf{A}_\lambda^\top \mathbf{A}_\lambda$. We apply Lemma 2 with the prior $\mu(d\lambda) \propto \exp\{2 \text{Tr}[\Sigma(\mathbf{A}_\lambda^\top - \mathbf{A}_\lambda^\top \mathbf{A}_\lambda)]/\beta\} \pi(d\lambda)$, with $\hat{\mathbf{f}}_\lambda = \mathbf{A}_\lambda \mathbf{Y}$ and with the risk estimate

$$\tilde{r}_\lambda = \|\mathbf{Y} - \hat{\mathbf{f}}_\lambda\|_n^2 + \frac{2}{n} \text{Tr}[\Sigma \tilde{\mathbf{A}}_\lambda] - \frac{1}{n} \text{Tr}[\Sigma] = \hat{r}_\lambda^{\text{unb}} + \frac{2}{n} \text{Tr}[\Sigma \mathbf{A}_\lambda^\top \mathbf{A}_\lambda - \Sigma \mathbf{A}_\lambda]. \quad (14)$$

One easily checks that this choice leads to the posterior $\hat{\mu}$ that is equal to $\hat{\pi}$ defined in Figure 2. Therefore, the aggregate $\tilde{\mathbf{f}}_{\text{EWA}}$ based on the prior μ coincides with $\hat{\mathbf{f}}_{\text{SEWA}}$ based on the prior π . Thus we obtain the following inequality:

$$\mathbb{E}[\|\hat{\mathbf{f}}_{\text{SEWA}} - \mathbf{f}\|_n^2] \leq \inf_{p \in \mathcal{P}_A} \left\{ \int_A \mathbb{E}[\tilde{r}_\lambda] p(d\lambda) + \frac{\beta \mathcal{K}(p, \mu)}{n} \right\}. \quad (15)$$

Furthermore, easy algebra yields that all the conditions required in the second part of Lemma 2 are fulfilled with $a = \frac{2\|\Sigma\|}{n}$ as soon as $\beta \geq 4\|\Sigma\|$. Indeed, one can notice that $\nabla_{\mathbf{Y}} \tilde{r}_\lambda = \frac{2}{n}(\mathbf{Y} - \tilde{\mathbf{f}}_\lambda)$. This leads to

$$\begin{aligned} \int_A \langle \nabla_{\mathbf{Y}} \tilde{r}_\lambda | \Sigma(\tilde{\mathbf{f}}_\lambda - \tilde{\mathbf{f}}_{\text{EWA}}) \rangle_n \hat{\mu}(d\lambda) &= \frac{2}{n} \int_A \langle \tilde{\mathbf{f}}_{\text{EWA}} - \tilde{\mathbf{f}}_\lambda | \Sigma(\tilde{\mathbf{f}}_\lambda - \tilde{\mathbf{f}}_{\text{EWA}}) \rangle_n \hat{\mu}(d\lambda) \\ &\leq \frac{2\|\Sigma\|}{n} \int_A \|\tilde{\mathbf{f}}_\lambda - \tilde{\mathbf{f}}_{\text{EWA}}\|_n^2 \hat{\mu}(d\lambda). \end{aligned} \quad (16)$$

Hence the conclusion of the second part of Lemma 2 holds true. To prove the claim of Remark 2, one can notice that:

$$\begin{aligned} \mathcal{K}(p, \mu) &= - \int_A \log\left(\frac{d\mu}{dp}(\lambda)\right) p(d\lambda) \\ &= \int_A \frac{2}{\beta} \text{Tr}[\Sigma(\mathbf{A}_\lambda^\top \mathbf{A}_\lambda - \mathbf{A}_\lambda)] p(d\lambda) + \log \left[\int_A e^{\frac{2}{\beta} \text{Tr}[\Sigma(\mathbf{A}_\lambda - \mathbf{A}_\lambda^\top \mathbf{A}_\lambda)]} \pi(d\lambda) \right] + \mathcal{K}(p, \pi). \end{aligned} \quad (17)$$

Then, by taking the expectation and combining together relations (14), (15) and (17), one gets $\mathbb{E}[\|\hat{\mathbf{f}}_{\text{SEWA}} - \mathbf{f}\|_n^2] \leq \inf_{p \in \mathcal{P}_A} \left\{ \int_A r_\lambda p(d\lambda) + \frac{\beta \mathcal{K}(p, \pi)}{n} \right\} + R_n$, and the claim of Remark 2 follows.

Finally, if condition $\mathbf{C}(\lambda)$ is satisfied for π -almost all values of λ , then R_n is non-positive, and we get the sharp oracle inequality stated in Theorem 1.

Lipschitz Bandits without the Lipschitz Constant

Sébastien Bubeck¹, Gilles Stoltz^{2,3}, and Jia Yuan Yu^{2,3}

¹ Centre de Recerca Matemàtica, Barcelona, Spain

² Ecole normale supérieure, CNRS, Paris, France

³ HEC Paris, CNRS, Jouy-en-Josas, France

Abstract. We consider the setting of stochastic bandit problems with a continuum of arms indexed by $[0, 1]^d$. We first point out that the strategies considered so far in the literature only provided theoretical guarantees of the form: given some tuning parameters, the regret is small with respect to a class of environments that depends on these parameters. This is however not the right perspective, as it is the strategy that should adapt to the specific bandit environment at hand, and not the other way round. Put differently, an adaptation issue is raised. We solve it for the special case of environments whose mean-payoff functions are globally Lipschitz. More precisely, we show that the minimax optimal orders of magnitude $L^{d/(d+2)} T^{(d+1)/(d+2)}$ of the regret bound over T time instances against an environment whose mean-payoff function f is Lipschitz with constant L can be achieved without knowing L or T in advance. This is in contrast to all previously known strategies, which require to some extent the knowledge of L to achieve this performance guarantee.

1 Introduction

In the (stochastic) bandit problem, a gambler tries to maximize the revenue gained by sequentially playing one of a finite number of arms that are each associated with initially unknown (and potentially different) payoff distributions [Rob52]. The gambler selects and pulls arms one by one in a sequential manner, simultaneously learning about the machines' payoff-distributions and accumulating rewards (or losses). Thus, in order to maximize his gain, the gambler must choose the next arm by taking into consideration both the urgency of gaining reward ("exploitation") and acquiring new information ("exploration"). Maximizing the total cumulative payoff is equivalent to minimizing the (total) *regret*, that is, minimizing the difference between the total cumulative payoff of the gambler and that of another clairvoyant gambler who chooses the arm with the best mean-payoff in every round. The quality of the gambler's strategy can be characterized by the rate of growth of his expected regret with time. In particular, if this rate of growth is sublinear, the gambler in the long run plays as well as his clairvoyant counterpart.

Continuum-Armed Bandit Problems. Although the early papers studied bandits with a finite number of arms, researchers soon realized that bandits with

infinitely many arms are also interesting, as well as practically significant. One particularly important case is when the arms are identified by a finite number of continuous-valued parameters, resulting in online optimization problems over continuous finite-dimensional spaces. During the last decades numerous contributions have investigated such continuum-armed bandit problems, starting from the early formulations of [Agr95, Cop09, Kle04] to the more recent approaches of [AOS07, KSU08, BMSS11]. A special case of interest, which forms a bridge between the case of a finite number of arms and the continuum-armed setting, is the problem of bandit linear optimization, see [DHK08] and the references therein.

Not the Right Perspective! We call an environment f the mapping that associates with each arm $x \in \mathcal{X}$ the expectation $f(x)$ of its associated probability distribution. The theoretical guarantees given in the literature mentioned above are of the form: given some tuning parameters, the strategy is competitive, and sometimes even minimax optimal, with respect to a large class of environments that unfortunately depends on these parameters. But of course, this is not the right perspective: it is the strategy that should adapt to the environment, not the other way round!

More precisely, these parameters describe the smoothness of the environments f in the class at hand in terms of a global regularity and/or local regularities around the global maxima of f . The issues raised by some of the works mentioned above can be roughly described as follows:

- The class of environments for the CAB1 algorithm of [Kle04] is formed by environments that are (α, L, δ) -uniformly locally Lipschitz and the strategy CAB1 needs to know α to get the optimal dependency in the number T of arms pulled;
- For the Zooming algorithm of [KSU08], it is formed by environments that are 1-Lipschitz with respect to a fixed and known metric L ;
- The HOO algorithm of [BMSS11] basically needs to know the pseudo-metric ℓ with respect to which f is weakly Lipschitz continuous, with Lipschitz constant equal to 1;
- Other examples include the UCB-air algorithm (which relies on a smoothness parameter β , see [WAM09]), the OLOP algorithm (smoothness parameter γ , see [BM10]), the LSE algorithm (smoothness parameter C_L , see [YM11]), the algorithm presented in [AOS07] and so on.

Adaptation to the Unknown Smoothness is Needed. In a nutshell, adaptive methods are required. By adaptive methods, we mean—as is done in the statistical literature—agnostic methods, i.e., with minimal prior knowledge about f , that nonetheless obtain almost the same performance against a given environment f as if its smoothness were known beforehand.

More precisely, given a fixed (possibly vector-valued) parameter L lying in a set \mathcal{L} and a class of allowed environments \mathcal{F}_L , where $L \in \mathcal{L}$, existing works present algorithms that are such that their worst-case regret bound over T time steps against environments in \mathcal{F}_L ,

$$\sup_{f \in \mathcal{F}_L} R_T(f) \leq \varphi(T, L),$$

is small and even minimax optimal, i.e., such that it has the optimal dependencies on T and L . However, to do so, the knowledge of L is required. In this work, we are given a much larger class of environments $\mathcal{F} = \cup_{L \in \mathcal{L}} \mathcal{F}_L$, and our goal is an algorithm that adapts in finite time to every instance f of \mathcal{F} , in the sense that for all T and $f \in \mathcal{F}$, the regret $R_T(f)$ is at most of the order of $\min \varphi(T, L)$, where the minimum is over the parameters L such that $f \in \mathcal{F}_L$.

Since we are interested in worst-case bounds, we will have to consider distribution-free bounds (i.e., bounds that only depend on a given class \mathcal{F}_L); of course, the orders of magnitude of the latter, even when they are minimax optimal, are often far away –as far as the dependencies in T are concerned– with respect to distribution-dependent bounds (i.e., bounds that may depend on a specific instance $f \in \mathcal{F}_L$).

Links with Optimization Algorithms. Our problem shares some common points with the maximization of a deterministic function f (but note that in our case, we only get to see noisy observations of the values of f). When the Lipschitz constant L of f is known, an approximate maximizer can be found with well-known Lipschitz optimization algorithms (e.g., Shubert’s algorithm). The case of unknown L has been studied in [JPS93, Hor06]. The DIRECT algorithm of [JPS93] carries out Lipschitz optimization by using the smallest Lipschitz constant that is consistent with the observed data; although it works well in practice, only asymptotic convergence can be guaranteed. The algorithm of [Hor06] iterates over an increasing sequence of possible values of L ; under an additional assumption on the minimum increase in the neighborhood of the maximizers, it guarantees a worst-case error of the order of $L^2 T^{-2/d}$ after taking T samples of the deterministic function f .

Adaptation to a Global Lipschitz Smoothness in Bandit Problems. We provide in this paper a first step toward a general theory of adaptation. To do so, we focus on the special case of classes \mathcal{F}_L formed by all environments f that are L -Lipschitz with respect to the supremum norm over a subset of \mathbb{R}^d : the hypercube $[0, 1]^d$ for simplicity. This case covers partially the settings of [BMSS11] and [KSU08], in which the Lipschitz constant was equal to 1, a fact known by the algorithms. (Extensions to Hölderian-type assumptions as in [Kle04, AOS07] will be considered in future work.)

As it is known, getting the minimax-optimal dependency on T is easy, the difficult part is getting that on L without knowing the latter beforehand.

Our Contributions. Our algorithm proceeds by discretization as in [Kle04]. To determine the correct discretization step, it first resorts to an uniform exploration yielding a rather crude estimate of the Lipschitz constant (that is however sufficient for our needs); in a second phase, it finds the optimal interval using a standard exploration-exploitation strategy. Our main assumptions are (essentially) that f and its derivative are Lipschitz continuous in the hypercube.

We feel that this two-step approach can potentially be employed in more general settings well beyond ours: with the notation above, the uniform-exploration phase performs a model-selection step and recommends a class $\mathcal{F}_{\tilde{L}}$, which is used in the second phase to run a continuum-armed bandit strategy tuned with the optimal parameters corresponding to $\tilde{L} \in \mathcal{L}$. However, for the sake of simplicity, we study only a particular case of this general methodology.

Outline of the Paper. In Section 2, we describe the setting and the classes of environments of interest, establish a minimax lower bound on the achievable performance (Section 2.1), and indicate how to achieve it when the global Lipschitz parameter L is known (Section 2.2). Our main contribution (Section 3) is then a method to achieve it when the Lipschitz constant is unknown for a slightly restricted class of Lipschitz functions.

2 Setting and Notation

We consider a d -dimensional compact set of arms, say, for simplicity, $\mathcal{X} = [0, 1]^d$, where $d \geq 1$. With each arm $\underline{x} \in [0, 1]^d$ is associated a probability distribution $\nu_{\underline{x}}$ with known bounded support, say $[0, 1]$; this defines an environment. A key quantity of such an environment is given by the expectations $f(\underline{x})$ of the distributions $\nu_{\underline{x}}$. They define a mapping $f : [0, 1]^d \rightarrow [0, 1]$, which we call the mean-payoff function.

At each round $t \geq 1$, the player chooses an arm $\underline{I}_t \in [0, 1]^d$ and gets a reward Y_t sampled independently from $\nu_{\underline{I}_t}$ (conditionally on the choice of \underline{I}_t). We call a strategy the (possibly randomized) rule that indicates at each round which arm to pull given the history of past rewards.

We write the elements \underline{x} of $[0, 1]^d$ in columns; \underline{x}^T will thus denote a row vector with d elements.

Assumption 1. We assume that f is twice differentiable, with Hessians uniformly bounded by M in the following sense: for all $\underline{x} \in [0, 1]^d$ and all $\underline{y} \in [0, 1]^d$,

$$\left| \underline{y}^T H_f(\underline{x}) \underline{y} \right| \leq M \|\underline{y}\|_\infty^2.$$

The ℓ^1 -norm of the gradient $\|\nabla f\|_1$ of f is thus continuous and it achieves its maximum on $[0, 1]^d$, whose value is denoted by L . As a result, f is Lipschitz with respect to the ℓ^∞ -norm with constant L (and L is the smallest¹ constant for which it is Lipschitz): for all $\underline{x}, \underline{y} \in [0, 1]^d$,

$$|f(\underline{x}) - f(\underline{y})| \leq L \|\underline{x} - \underline{y}\|_\infty.$$

In the sequel we denote by $\mathcal{F}_{L,M}$ the set of environments whose mean-payoff functions satisfy the above assumption.

We also denote by \mathcal{F}_L the larger set of environments whose mean-payoff functions f is only constrained to be L -Lipschitz with respect to the ℓ^∞ -norm.

¹ The proof of the approximation lemma will show why this is the case.

2.1 The Minimax Optimal Orders of Magnitude of the Regret

When f is continuous, we denote by

$$f^* = \sup_{\underline{x} \in [0,1]^d} f(\underline{x}) = \max_{\underline{x} \in [0,1]^d} f(\underline{x})$$

the largest expected payoff in a single round. The expected regret \bar{R}_T at round T is then defined as

$$\bar{R}_T = \mathbb{E} \left[T f^* - \sum_{t=1}^T Y_t \right] = \mathbb{E} \left[T f^* - \sum_{t=1}^T f(L_t) \right]$$

where we used the tower rule and where the expectations are with respect to the random draws of the Y_t according to the ν_{L_t} as well as to any auxiliary randomization the strategy uses.

In this article, we are interested in controlling the worst-case expected regret over all environments of \mathcal{F}_L . The following minimax lower bound follows from a straightforward adaptation of the proof of [BMSS11, Theorem 13], which is omitted from this extended abstract and may be found in [BSY11]. (The adaptation is needed because the hypothesis on the packing number is not exactly satisfied in the form stated in [BMSS11].)

Theorem 1. *For all strategies of the player and for all*

$$T \geq \max \left\{ L^d, \left(\frac{0.15 L^{2/(d+2)}}{\max\{d, 2\}} \right)^d \right\},$$

the worst-case regret over the set \mathcal{F}_L of all environments that are L -Lipschitz with respect to the ℓ^∞ -norm is larger than

$$\sup_{\mathcal{F}_L} \bar{R}_T \geq 0.15 L^{d/(d+2)} T^{(d+1)/(d+2)}.$$

The multiplicative constants are not optimized in this bound (a more careful proof might lead to a larger constant in the lower bound).

2.2 How to Achieve a Minimax Optimal Regret When L is Known

In view of the previous section, our aim is to design strategies with worst-case expected regret $\sup_{\mathcal{F}_L} \bar{R}_T$ less than something of order $L^{d/(d+2)} T^{(d+1)/(d+2)}$ when L is unknown. A simple way to do so when L is known was essentially proposed in the introduction of [Kle04] (in the case $d = 1$); it proceeds by discretizing the arm space. The argument is reproduced below and can be used even when L is unknown to recover the optimal dependency $T^{(d+1)/(d+2)}$ on T (but then, with a suboptimal dependency on L).

We consider the approximations \bar{f}_m of f with m^d regular hypercube bins in the ℓ^∞ -norm, i.e., m bins are formed in each direction and combined to

form the hypercubes. Each of these hypercube bins is indexed by an element $\underline{k} = (k_1, \dots, k_d) \in \{0, \dots, m-1\}^d$. The average value of f over the bin indexed by \underline{k} is denoted by

$$\bar{f}_m(\underline{k}) = m^d \int_{\underline{k}/m + [0, 1/m]^d} f(\underline{x}) d\underline{x}.$$

We then consider the following two-stage strategy, which is based on some strategy MAB for multi-armed bandits; MAB will refer to a generic strategy but we will instantiate below the obtained bound. Knowing L and assuming that T is fixed and known in advance, we may choose beforehand $m = \lceil L^{2/(d+2)} T^{1/(d+2)} \rceil$. The decomposition of $[0, 1]^d$ into m^d bins thus obtained will play the role of the finitely many arms of the multi-armed bandit problem. At round $t \geq 1$, whenever the MAB strategy prescribes to pull bin $\underline{K}_t \in \{0, \dots, m-1\}^d$, then first, an arm \underline{I}_t is pulled at random in the hypercube $\underline{K}_t/m + [0, 1/m]^d$; and second, given \underline{I}_t , the reward Y_t is drawn at random according to $\nu_{\underline{I}_t}$. Therefore, given \underline{K}_t , the reward Y_t has an expected value of $\bar{f}_m(\underline{K}_t)$. Finally, the reward Y_t is returned to the underlying MAB strategy.

Strategy MAB is designed to control the regret with respect to the best of the m^d bins, which entails that

$$\mathbb{E} \left[T \max_{\underline{k}} \bar{f}_m(\underline{k}) - \sum_{t=1}^T Y_t \right] \leq \psi(T, m^d),$$

for some function ψ that depends on MAB. Now, whenever f is L -Lipschitz with respect to the ℓ^∞ -norm, we have that for all $\underline{k} \in \{0, \dots, m-1\}^d$ and all $\underline{x} \in \underline{k}/m + [0, 1/m]^d$, the difference $|f(\underline{x}) - \bar{f}_m(\underline{k})|$ is less than L/m ; so that²

$$\max_{\underline{x} \in [0, 1]^d} f(\underline{x}) - \max_{\underline{k}} \bar{f}_m(\underline{k}) \leq \frac{L}{m}.$$

All in all, for this MAB-based strategy, the regret is bounded by the sum of the approximation term L/m and of the regret term for multi-armed bandits,

$$\begin{aligned} \sup_{\mathcal{F}_L} \bar{R}_T &\leq T \frac{L}{m} + \psi(T, m^d) \\ &\leq L^{d/(d+2)} T^{(d+1)/(d+2)} + \psi\left(T, \left(\lceil L^{2/(d+2)} T^{1/(d+2)} \rceil\right)^d\right). \end{aligned} \quad (1)$$

We now instantiate this bound.

² Here, one could object that we only use the local Lipschitzness of f around the point where it achieves its maximum; however, we need f to be L -Lipschitz in an $1/m$ -neighborhood of this maximum, but the optimal value of m depends on L . To solve the chicken-egg problem, we restricted our attention to globally L -Lipschitz functions, which, anyway, in view of Theorem I, comes at no cost as far as minimax-optimal orders of magnitude of the regret bounds in L and T are considered.

The INF strategy of [AB10] (see also [ABL11]) achieves $\psi(T, m') = 2\sqrt{2Tm'}$ and this entails a final $O(L^{d/(d+2)} T^{(d+1)/(d+2)})$ bound in (I). Note that for the EXP3 strategy of [ACBFS02] or the UCB strategy of [ACBF02], extra logarithmic terms of the order of $\ln T$ would appear in the bound.

3 Achieving a Minimax Optimal Regret Not Knowing L

In this section, our aim is to obtain a worst-case regret bound of the minimax-optimal order of $L^{d/(d+2)} T^{(d+1)/(d+2)}$ even when L is unknown. To do so, it will be useful to first estimate L ; we will provide a (rather crude) estimate suited to our needs, as our goal is the minimization of the regret rather than the best possible estimation of L . Our method is based on the following approximation results.

For the estimation to be efficient, it will be convenient to restrict our attention to the subset $\mathcal{F}_{L,M}$ of \mathcal{F}_L , i.e., we will consider the additional assumptions on the existence and boundedness of the Hessians asserted in Assumption I. However, the obtained regret bound (Theorem 2) will suffer from some (light) dependency on M but will have the right orders of magnitude in T and L ; the forecaster used to achieve it depends neither on L nor on M and is fully adaptive.

3.1 Some Preliminary Approximation Results

We still consider the approximations \bar{f}_m of f over $[0, 1]^d$ with m^d regular bins. We then introduce the following approximation of L :

$$\bar{L}_m = m \max_{\underline{k} \in \{1, \dots, m-2\}^d} \max_{\underline{s} \in \{-1, 1\}^d} |\bar{f}_m(\underline{k}) - \bar{f}_m(\underline{k} + \underline{s})|.$$

This quantity provides a fairly good approximation of the Lipschitz constant, since $m(\bar{f}_m(\underline{k}) - \bar{f}_m(\underline{k} + \underline{s}))$ is an estimation of the (average) derivative of f in bin \underline{k} and direction \underline{s} .

The lemma below relates precisely \bar{L}_m to L : as m increases, \bar{L}_m converges to L .

Lemma 1. *If $f \in \mathcal{F}_{L,M}$ and $m \geq 3$, then*

$$L - \frac{7M}{m} \leq \bar{L}_m \leq L.$$

Proof. We note that for all $\underline{k} \in \{1, \dots, m-2\}^d$ and $\underline{s} \in \{-1, 1\}^d$, we have by definition

$$\begin{aligned} |\bar{f}_m(\underline{k}) - \bar{f}_m(\underline{k} + \underline{s})| &= m^d \left| \int_{\underline{k}/m + [0, 1/m]^d} \left(f(\underline{x}) - f(\underline{x} + \underline{s}/m) \right) d\underline{x} \right| \\ &\leq m^d \int_{\underline{k}/m + [0, 1/m]^d} \left| f(\underline{x}) - f(\underline{x} + \underline{s}/m) \right| d\underline{x}. \end{aligned}$$

Now, since f is L -Lipschitz in the ℓ^∞ -norm, it holds that

$$\left| f(\underline{x}) - f(\underline{x} + \underline{s}/m) \right| \leq L \|\underline{s}/m\|_\infty = \frac{L}{m};$$

integrating this bound entails the stated upper bound L on \overline{L}_m .

For the lower bound, we first denote by $\underline{x}_* \in [0, 1]^d$ a point such that $\|\nabla f(\underline{x}_*)\|_1 = L$. (Such a point always exists, see Assumption II) This point belongs to some bin in $\{0, \dots, m-1\}^d$; however, the closest bin \underline{k}_m^* in $\{1, \dots, m-2\}^d$ is such that

$$\forall \underline{x} \in \underline{k}_m^*/m + [0, 1/m]^d, \quad \|\underline{x} - \underline{x}_*\|_\infty \leq \frac{2}{m}. \quad (2)$$

Note that this bin \underline{k}_m^* is such that all $\underline{k}_m^* + \underline{s}$ belong to $\{0, \dots, m-1\}^d$ and hence legally index hypercube bins, when $\underline{s} \in \{-1, 1\}^d$. Now, let $\underline{s}_m^* \in \{-1, 1\}^d$ be such that

$$\nabla f(\underline{x}_*) \cdot \underline{s}_m^* = \|\nabla f(\underline{x}_*)\|_1 = L, \quad (3)$$

where \cdot denotes the inner product in \mathbb{R}^d . By the definition of \overline{L}_m as some maximum,

$$\begin{aligned} \overline{L}_m &\geq m \left| \overline{f}_m(\underline{k}_m^*) - \overline{f}_m(\underline{k}_m^* + \underline{s}_m^*) \right| \\ &= m \times m^d \left| \int_{\underline{k}_m^*/m + [0, 1/m]^d} \left(f(\underline{x}) - f(\underline{x} + \underline{s}_m^*/m) \right) d\underline{x} \right|. \end{aligned} \quad (4)$$

Now, Taylor's theorem (in the mean-value form for real-valued twice differentiable functions of possibly several variables) shows that for any $\underline{x} \in \underline{k}_m^*/m + [0, 1/m]^d$, there exists two elements ξ and ζ , belonging respectively to the segments between \underline{x} and \underline{x}_* , on the one hand, between \underline{x}_* and $\underline{x} + \underline{s}_m^*/m$ on the other hand, such that

$$\begin{aligned} &f(\underline{x}) - f(\underline{x} + \underline{s}_m^*/m) \\ &= (f(\underline{x}) - f(\underline{x}_*)) + (f(\underline{x}_*) - f(\underline{x} + \underline{s}_m^*/m)) \\ &= \nabla f(\underline{x}_*) \cdot (\underline{x} - \underline{x}_*) + \frac{1}{2} (\underline{x} - \underline{x}_*)^T H_f(\xi) (\underline{x} - \underline{x}_*) \\ &\quad - \nabla f(\underline{x}_*) \cdot (\underline{x} + \underline{s}_m^*/m - \underline{x}_*) - \frac{1}{2} (\underline{x} + \underline{s}_m^*/m - \underline{x}_*)^T H_f(\zeta) (\underline{x} + \underline{s}_m^*/m - \underline{x}_*) \\ &= -\nabla f(\underline{x}_*) \cdot \frac{\underline{s}_m^*}{m} + \frac{1}{2} (\underline{x} - \underline{x}_*)^T H_f(\xi) (\underline{x} - \underline{x}_*) \\ &\quad - \frac{1}{2} (\underline{x} + \underline{s}_m^*/m - \underline{x}_*)^T H_f(\zeta) (\underline{x} + \underline{s}_m^*/m - \underline{x}_*). \end{aligned}$$

Using B3 and substituting the bound on the Hessians stated in Assumption II, we get

$$f(\underline{x}) - f(\underline{x} + \underline{s}_m^*/m) \leq -\frac{L}{m} + \frac{M}{2} \|\underline{x} - \underline{x}_*\|_\infty^2 + \frac{M}{2} \|\underline{x} + \underline{s}_m^*/m - \underline{x}_*\|_\infty^2;$$

substituting (2), we get

$$f(\underline{x}) - f(\underline{x} + \underline{s}_m^*/m) \leq -\frac{L}{m} + \frac{M}{2m^2}(2^2 + 3^2) \leq -\frac{L}{m} + \frac{7M}{m^2} \leq 0,$$

where the last inequality holds with no loss of generality (if it does not, then the lower bound on \bar{L}_m in the statement of the lemma is trivial). Substituting and integrating this equality in (1) and using the triangle inequality, we get

$$\bar{L}_m \geq L - \frac{7M}{m}.$$

This concludes the proof. \square

3.2 A Strategy in Two Phases

Our strategy is described in Figure 1; several notation that will be used in the statements and proofs of some results below are defined therein. Note that we proceed in two phases: a pure exploration phase, when we estimate L by some \tilde{L}_m , and an exploration-exploitation phase, when we use a strategy designed for the case of finitely-armed bandits on a discretized version of the arm space. (The discretization step depends on the estimate obtained in the pure exploration phase.)

The first step in the analysis is to relate \tilde{L}_m and \hat{L}_m to the quantity they are estimating, namely \bar{L}_m .

Lemma 2. *With probability at least $1 - \delta$,*

$$|\hat{L}_m - \bar{L}_m| \leq m \sqrt{\frac{2}{E} \ln \frac{2m^d}{\delta}}.$$

Proof. We consider first a fixed $\underline{k} \in \{0, \dots, m-1\}^d$; as already used in Section 2.2, the $Z_{\underline{k},j}$ are independent and identically distributed according to a distribution on $[0, 1]$ with expectation $\bar{f}_m(\underline{k})$, as j varies between 1 and E . Therefore, by Hoeffding's inequality, with probability at least $1 - \delta/m^d$

$$|\hat{\mu}_{\underline{k}} - \bar{f}_m(\underline{k})| \leq \sqrt{\frac{1}{2E} \ln \frac{2m^d}{\delta}}.$$

Performing a union bound and using the triangle inequality, we get that with probability at least $1 - \delta$,

$$\forall \underline{k}, \underline{k}' \in \{0, \dots, m-1\}^d, \quad \left| |\hat{\mu}_{\underline{k}} - \hat{\mu}_{\underline{k}'}| - |\bar{f}_m(\underline{k}) - \bar{f}_m(\underline{k}')| \right| \leq \sqrt{\frac{2}{E} \ln \frac{2m^d}{\delta}}.$$

This entails the claimed bound. \square

By combining Lemmas 1 and 2, we get the following inequalities on \tilde{L}_m , since the latter is obtained from \hat{L}_m by adding a deviation term.

Parameters:

- Number T of rounds;
- Number m of bins (in each direction) considered in the pure exploration phase;
- Number E of times each of them must be pulled;
- A multi-armed bandit strategy MAB (taking as inputs a number m^d of arms and possibly other parameters).

Pure exploration phase:

1. For each $\underline{k} \in \{0, \dots, m-1\}^d$
 - pull E arms independently uniformly at random in $\underline{k}/m + [0, 1/m]^d$ and get E associated rewards $Z_{\underline{k},j}$, where $j \in \{1, \dots, E\}$;
 - compute the average reward for bin \underline{k} ,

$$\hat{\mu}_{\underline{k}} = \frac{1}{E} \sum_{j=1}^E Z_{\underline{k},j};$$

2. Set

$$\hat{L}_m = m \max_{\underline{k} \in \{1, \dots, m-2\}^d} \max_{\underline{s} \in \{-1, 1\}^d} |\hat{\mu}_{\underline{k}} - \hat{\mu}_{\underline{k}+\underline{s}}|$$

and define $\tilde{L}_m = \hat{L}_m + m \sqrt{\frac{2}{E} \ln(2m^d T)}$ as well as $\tilde{m} = \lceil \tilde{L}_m^{2/(d+2)} T^{1/(d+2)} \rceil$.

Exploration-exploitation phase:

Run the strategy MAB with \tilde{m}^d arms as follows; for all $t = Em + 1, \dots, T$,

1. If MAB prescribes to play arm $\underline{K}_t \in \{0, \dots, \tilde{m}-1\}^d$, pull an arm \underline{L}_t at random in $\underline{K}_t/m + [0, 1/m]^d$;
 2. Observe the associated payoff Y_t , drawn independently according to $\nu_{\underline{L}_t}$;
 3. Return Y_t to the strategy MAB.
-

Fig. 1. The considered strategy

Corollary 1. *If $f \in \mathcal{F}_{L,M}$ and $m \geq 3$, then, with probability at least $1 - 1/T$,*

$$L - \frac{7M}{m} \leq \tilde{L}_m \leq L + 2m\sqrt{\frac{2}{E} \ln(2m^d T)}.$$

We state a last intermediate result; it relates the regret of the strategy of Figure I to the regret of the strategy MAB that it takes as a parameter.

Lemma 3. *Let $\psi(T', m')$ be a distribution-free upper bound on the expected regret of the strategy MAB, when run for T' rounds on a multi-armed bandit problem with m' arms, to which payoff distributions over $[0, 1]$ are associated. The expected regret of the strategy defined in Figure I is then bounded from above as*

$$\sup_{\mathcal{F}_L} \bar{R}_T \leq Em^d + \mathbb{E} \left[\frac{LT}{\tilde{m}} + \psi(T - Em^d, \tilde{m}^d) \right].$$

Proof. As all payoffs lie in $[0, 1]$, the regret during the pure exploration phase is bounded by the total length Em^d of this phase.

Now, we bound the (conditionally) expected regret of the MAB strategy during the exploration-exploitation phase; the conditional expectation is with respect to the pure exploration phase and is used to fix the value of \tilde{m} . Using the same arguments as in Section 2.2, the regret during this phase, which lasts $T - Em^d$ rounds, is bounded against any environment in \mathcal{F}_L by

$$L \frac{T - Em^d}{m} + \psi(T - Em^d, \tilde{m}^d).$$

The tower rule concludes the proof. \square

We are now ready to state our main result. Note that it is somewhat unsatisfactory as the main regret bound (8) could only be obtained on the restricted class $\mathcal{F}_{L,M}$ and depends (in a light manner, see comments below) on the parameter M , while having the optimal orders of magnitude in T and L . These drawbacks might be artifacts of the analysis and could be circumvented, perhaps in the light of the proof of the lower bound (Theorem I), which exhibits the worst-case elements of \mathcal{F}_L (they seem to belong to some set \mathcal{F}_{L,M_L} , where M_L is a decreasing function of L).

Note however that the dependency on M in (8) is in the additive form. On the other hand, by adapting the argument of Section 2.2, one can prove distribution-free regret bounds on $\mathcal{F}_{L,M}$ with an improved order of magnitude as far as T is concerned but at the cost of getting M as a multiplicative constant in the picture. While the corresponding bound might be better in some regimes, we consider here $\mathcal{F}_{L,M}$ instead of \mathcal{F}_L essentially to remove pathological functions, and as such we want the weakest dependency on M .

Theorem 2. When used with the multi-armed strategy INF, the strategy of Figure 1 ensures that

$$\sup_{\mathcal{F}_{L,M}} \overline{R}_T \leq T^{(d+1)/(d+2)} \left(9L^{d/(d+2)} + 5 \left(2m \sqrt{\frac{2}{E} \ln(2T^{d+1})} \right)^{d/(d+2)} \right) + Em^d + 2\sqrt{2Td^d} + 1 \quad (5)$$

as soon as

$$m \geq \frac{8M}{L}. \quad (6)$$

In particular, for

$$0 < \gamma < \frac{d(d+1)}{(3d+2)(d+2)} \quad \text{and} \quad \alpha = \frac{1}{d+2} \left(\frac{d+1}{d+2} - \gamma \frac{3d+2}{d} \right) > 0, \quad (7)$$

the choices of $m = \lfloor T^\alpha \rfloor$ and $E = m^2 \lceil T^{2\gamma(d+2)/d} \rceil$ yield the bound

$$\sup_{\mathcal{F}_{L,M}} \overline{R}_T \leq \max \left\{ \left(\frac{8M}{L} + 1 \right)^{1/\alpha}, L^{d/(d+2)} T^{(d+1)/(d+2)} (9 + \varepsilon(T, d)) \right\}, \quad (8)$$

where

$$\varepsilon(T, d) = 5T^{-\gamma} (\ln(2T^d))^{d/(d+2)} + T^{-\gamma} + \frac{2\sqrt{2d^d T} + 1}{T^{-(d+1)/(d+2)}}$$

vanishes as T tends to infinity.

Note that the choices of E and m solely depend on T , which may however be unknown in advance; standard arguments, like the doubling trick, can be used to circumvent the issue, at a minor cost given by an additional constant multiplicative factor in the bound.

Remark 1. There is a trade-off between the value of the constant term in the maximum, $(1+8M/L)^{1/\alpha}$, and the convergence rate of the vanishing term $\varepsilon(T, d)$ toward 0, which is of order γ . For instance, in the case $d = 1$, the condition on γ is $0 < \gamma < 2/15$; as an illustration, we get

- a constant term of a reasonable size, since $1/\alpha \leq 4.87$, when the convergence rate is small, $\gamma = 0.01$;
- a much larger constant, since $1/\alpha = 60$, when the convergence rate is faster, $\gamma = 2/15 - 0.01$.

Proof. For the strategy INF, as recalled above, $\psi(T', m') = 2\sqrt{2T'm'}$. The bound of Lemma 3 can thus be instantiated as

$$\sup_{\mathcal{F}_{L,M}} \overline{R}_T \leq Em^d + \mathbb{E} \left[\frac{LT}{\tilde{m}} + 2\sqrt{2T\tilde{m}^d} \right].$$

We now substitute the definition

$$\tilde{m} = \left\lceil \tilde{L}_m^{2/(d+2)} T^{1/(d+2)} \right\rceil \leq \tilde{L}_m^{2/(d+2)} T^{1/(d+2)} \left(1 + \frac{1}{\tilde{L}_m^{2/(d+2)} T^{1/(d+2)}} \right)$$

and separate the cases depending on whether $\tilde{L}_m^{2/(d+2)} T^{1/(d+2)}$ is smaller or larger than d to handle the second term in the expectation. In the first case, we simply bound \tilde{m} by d and get a $2\sqrt{2Td^d}$ term. When the quantity of interest is larger than d , then we get the central term in the expectation below by using the fact that $(1 + 1/x)^d \leq (1 + 1/d)^d \leq e$ whenever $x \geq d$. That is, $\sup_{\mathcal{F}_{L,M}} \bar{R}_T$ is less than

$$Em^d + \mathbb{E} \left[T^{(d+1)/(d+2)} \frac{L}{\tilde{L}_m^{2/(d+2)}} + 2\sqrt{2T e \left(T^{1/(d+2)} \tilde{L}_m^{2/(d+2)} \right)^d} + 2\sqrt{2Td^d} \right].$$

We will now use the lower and upper bounds on \hat{L}_m stated by Corollary 11. In the sequel we will make repeated use of the following inequality linking α -norms and 1-norms: for all integers p , all $u_1, \dots, u_p > 0$, and all $\alpha \in [0, 1]$,

$$(u_1 + \dots + u_p)^\alpha \leq u_1^\alpha + \dots + u_p^\alpha. \quad (9)$$

By resorting to (9), we get that with probability at least $1 - 1/T$,

$$\begin{aligned} 2\sqrt{2T e \left(T^{1/(d+2)} \tilde{L}_m^{2/(d+2)} \right)^d} &= 2\sqrt{2e} T^{(d+1)/(d+2)} \tilde{L}_m^{d/(d+2)} \\ &\leq 2\sqrt{2e} T^{(d+1)/(d+2)} \left(L + 2m \sqrt{\frac{2}{E} \ln(2m^d T)} \right)^{d/(d+2)} \\ &\leq 2\sqrt{2e} T^{(d+1)/(d+2)} \left(L^{d/(d+2)} + \left(2m \sqrt{\frac{2}{E} \ln(2m^d T)} \right)^{d/(d+2)} \right). \end{aligned}$$

On the other hand, with probability at least $1 - 1/T$,

$$\tilde{L}_m \geq L - \frac{7M}{m} \geq \frac{L}{8},$$

where we assumed that m and E are chosen large enough for the lower bound of Corollary 11 to be larger than $L/8$. This is indeed the case as soon as

$$\frac{7M}{m} \leq \frac{7L}{8}, \quad \text{that is,} \quad m \geq \frac{8M}{L},$$

which is exactly the condition (6).

Putting all things together (and bounding m by T in the logarithm), with probability at least $1 - 1/T$, the regret is less than

$$\begin{aligned} Em^d + T^{(d+1)/(d+2)} \frac{L}{(L/8)^{2/(d+2)}} + 2\sqrt{2Td^d} \\ + 2\sqrt{2e} T^{(d+1)/(d+2)} \left(L^{d/(d+2)} + \left(2m \sqrt{\frac{2}{E} \ln(2T^{d+1})} \right)^{d/(d+2)} \right); \quad (10) \end{aligned}$$

on the event of probability smaller than $\delta = 1/T$ where the above bound does not necessarily hold, we upper bound the regret by T . Therefore, the expected regret is bounded by (10) plus 1. Bounding the constants as $8^{2/(d+2)} \leq 8^{2/3} = 4$ and $2\sqrt{2e} \leq 5$ concludes the proof of the first part of the theorem.

The second part follows by substituting the values of E and m in the expression above and by bounding the regret by T_0 for the time steps $t \leq T_0$ for which the condition (6) is not satisfied.

More precisely, the regret bound obtained in the first part is of the desired order $L^{d/(d+2)} T^{(d+1)/(d+2)}$ only if $E \gg m^2$ and $Em^d \ll T^{(d+1)/(d+2)}$. This is why we looked for suitable values of m and E in the following form:

$$m = \lfloor T^\alpha \rfloor \quad \text{and} \quad E = m^2 \lceil T^{2\gamma(d+2)/d} \rceil,$$

where α and γ are positive. We choose α as a function of γ so that the terms

$$\begin{aligned} Em^d &= (\lfloor T^\alpha \rfloor)^{d+2} \lceil T^{2\gamma(d+2)/d} \rceil \\ \text{and} \quad T^{(d+1)/(d+2)} \left(\frac{m}{\sqrt{E}} \right)^{d/(d+2)} &= T^{(d+1)/(d+2)} \left(\lceil T^{2\gamma(d+2)/d} \rceil \right)^{-d/(2(d+2))} \end{aligned}$$

are approximatively balanced; for instance, such that

$$\alpha(d+2) + 2\gamma(d+2)/d = (d+1)/(d+2) - \gamma,$$

which yields the proposed expression (7). The fact that α needs to be positive entails the constraint on γ given in (7).

When condition (6) is met, we substitute the values of m and E into (5) to obtain the bound (8); the only moment in this substitution when taking the upper or lower integer parts does not help is for the term Em^d , for which we write (using that $T \geq 1$)

$$\begin{aligned} Em^d &= m^{d+2} \lceil T^{2\gamma(d+2)/d} \rceil \leq T^{\alpha(d+2)} (1 + T^{2\gamma(d+2)/d}) \\ &\leq 2 T^{\alpha(d+2)} T^{2\gamma(d+2)/d} = 2 T^{(d+1)/(d+2)-\gamma}. \end{aligned}$$

When condition (6) is not met, which can only be the case when T is such that $T^\alpha < 1 + 8M/L$, that is, $T < T_0 = (1 + 8M/L)^{1/\alpha}$, we upper bound the regret by T_0 . \square

Acknowledgements. This work was supported in part by French National Research Agency (ANR, project EXPLO-RA, ANR-08-COSI-004) and the PASCAL2 Network of Excellence under EC grant no. 216886.

References

- [AB10] Audibert, J.-Y., Bubeck, S.: Regret bounds and minimax policies under partial monitoring. Journal of Machine Learning Research 11, 2635–2686 (2010)

- [ABL11] Audibert, J.-Y., Bubeck, S., Lugosi, G.: Minimax policies for combinatorial prediction games. In: Proceedings of the 24th Annual Conference on Learning Theory. Omnipress (2011)
- [ACBF02] Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multi-armed bandit problem. *Machine Learning Journal* 47(2-3), 235–256 (2002)
- [ACBFS02] Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.: The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing* 32(1), 48–77 (2002)
- [Agr95] Agrawal, R.: The continuum-armed bandit problem. *SIAM Journal on Control and Optimization* 33, 1926–1951 (1995)
- [AOS07] Auer, P., Ortner, R., Szepesvári, C.: Improved rates for the stochastic continuum-armed bandit problem. In: Bshouty, N.H., Gentile, C. (eds.) COLT. LNCS (LNAI), vol. 4539, pp. 454–468. Springer, Heidelberg (2007)
- [BM10] Bubeck, S., Munos, R.: Open-loop optimistic planning. In: Proceedings of the 23rd Annual Conference on Learning Theory. Omnipress (2010)
- [BMSS11] Bubeck, S., Munos, R., Stoltz, G., Szepesvári, C.: \mathcal{X} -armed bandits. *Journal of Machine Learning Research* 12, 1655–1695 (2011)
- [BSY11] Bubeck, S., Stoltz, G., Yu, J.Y.: Lipschitz bandits without the lipschitz constant (2011), <http://arxiv.org/pdf/1105.5041>
- [Cop09] Cope, E.: Regret and convergence bounds for immediate-reward reinforcement learning with continuous action spaces. *IEEE Transactions on Automatic Control* 54(6), 1243–1253 (2009)
- [DHK08] Dani, V., Hayes, T.P., Kakade, S.M.: Stochastic linear optimization under bandit feedback. In: Proceedings of the 21st Annual Conference on Learning Theory, pp. 355–366. Omnipress (2008)
- [Hor06] Horn, M.: Optimal algorithms for global optimization in case of unknown Lipschitz constant. *Journal of Complexity* 22(1) (2006)
- [JPS93] Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications* 79(1), 157–181 (1993)
- [Kle04] Kleinberg, R.: Nearly tight bounds for the continuum-armed bandit problem. In: Advances in Neural Information Processing Systems, pp. 697–704 (2004)
- [KSU08] Kleinberg, R., Slivkins, A., Upfal, E.: Multi-armed bandits in metric spaces. In: Proceedings of the 40th ACM Symposium on Theory of Computing (2008)
- [Rob52] Robbins, H.: Some aspects of the sequential design of experiments. *Bulletin of the American Mathematics Society* 58, 527–535 (1952)
- [WAM09] Wang, Y., Audibert, J.Y., Munos, R.: Algorithms for infinitely many-armed bandits. In: Advances in Neural Information Processing Systems, pp. 1729–1736 (2009)
- [YM11] Yu, J.Y., Mannor, S.: Unimodal bandits. In: Proceedings of the 28th International Conference on Machine Learning (2011)

Deviations of Stochastic Bandit Regret

Antoine Salomon¹ and Jean-Yves Audibert^{1,2}

¹ Imagine, LIGM

École des Ponts ParisTech

Université Paris Est

salomona@imagine.enpc.fr

audibert@imagine.enpc.fr

² Sierra, CNRS/ENS/INRIA, Paris, France

Abstract. This paper studies the deviations of the regret in a stochastic multi-armed bandit problem. When the total number of plays n is known beforehand by the agent, Audibert et al. (2009) exhibit a policy such that with probability at least $1 - 1/n$, the regret of the policy is of order $\log n$. They have also shown that such a property is not shared by the popular UCB1 policy of Auer et al. (2002). This work first answers an open question: it extends this negative result to any anytime policy. The second contribution of this paper is to design anytime robust policies for specific multi-armed bandit problems in which some restrictions are put on the set of possible distributions of the different arms.

1 Introduction

Bandit problems illustrate the fundamental difficulty of sequential decision making in the face of uncertainty: a decision maker must choose between following what seems to be the best choice in view of the past (“exploitation”) or testing (“exploration”) some alternative, hoping to discover a choice that beats the current empirical best choice. More precisely, in the stochastic multi-armed bandit problem, at each stage, an agent (or decision maker) chooses one action (or arm), and receives a reward from it. The agent aims at maximizing his rewards. Since he does not know the process generating the rewards, he does not know the best arm, that is the one having the highest expected reward. He thus incurs a regret, that is the difference between the cumulative reward he would have got by always drawing the best arm and the cumulative reward he actually got. The name “bandit” comes from imagining a gambler in a casino playing with K slot machines, where at each round, the gambler pulls the arm of any of the machines and gets a payoff as a result.

The multi-armed bandit problem is the simplest setting where one encounters the exploration-exploitation dilemma. It has a wide range of applications including advertisement [BSS09], [DK09], economics [BV08], [LPT04], games [GW06] and optimization [Kle05], [CM07], [KSU08], [BMSS09]. It can be a central building block of larger systems, like in evolutionary programming [Hol92] and reinforcement learning [SB98], in particular in large state space Markovian

Decision Problems [KS06]. Most of these applications require that the policy of the forecaster works well *for any time*. For instance, in tree search using bandit policies at each node, the number of times the bandit policy will be applied at each node is not known beforehand (except for the root node in some cases), and the bandit policy should thus provide consistently low regret whatever the total number of rounds is.

Most previous works on the stochastic multi-armed bandit [Rob52], [LR85], [Agr95], [ACBF02] (among others) focused on the expected regret, and showed that after n rounds, the expected regret is of order $\log n$. So far, the analysis of the upper tail of the regret was only addressed in [AMS09]. The two main results there about the deviations of the regret are the following. First, after n rounds, for large enough constant $C > 0$, the probability that the regret of UCB1 (and also its variant taking into account the empirical variance) exceeds $C \log n$ is upper bounded by $1/(\log n)^{C'}$ for some constant C' depending on the distributions of the arms and on C (but not on n). Besides, for most bandit problems, this upper bound is tight to the extent that the probability is also lower bounded by a quantity of the same form. Second, a new upper confidence bound policy was proposed: it requires to know the total number of rounds in advance and uses this knowledge to design a policy which essentially explores in the first rounds and then exploits the information gathered in the exploration phase. Its regret has the advantage of being more concentrated to the extent that with probability at least $1 - 1/n$, the regret is of order $\log n$. The problem left open by [AMS09] is whether it is possible to design an anytime robust policy, that is a policy such that for any n , with probability at least $1 - 1/n$, the regret is of order $\log n$. In this paper, we answer negatively to this question when the reward distributions of all arms are just assumed to be uniformly bounded, say all rewards are in $[0, 1]$ for instance (Corollary 7). We then study which kind of restrictions on the set of probabilities defining the bandit problem allows to answer positively. One of our positive results is the following: if the agent knows the value of the expected reward of the best arm (but does not know which arm is the best one), the agent can use this information to design an anytime robust policy (Theorem 12).

The paper is organised as follows: in Section 2, we formally describe the problem we address and give the corresponding definitions and properties. In Section 3, we present our main impossibility result. In Section 4, we provide restrictions under which it is possible to design anytime robust policies. Section 5 is devoted to the proof of our main result. All other proofs are available at <http://hal.archives-ouvertes.fr/hal-00579607/en/>.

2 Problem Setup and Definitions

In the stochastic multi-armed bandit problem with $K \geq 2$ arms, at each time step $t = 1, 2, \dots$, an agent has to choose an arm I_t in the set $\{1, \dots, K\}$ and obtains a reward drawn from ν_{I_t} independently from the past (actions and observations). The environment is thus parameterized by a K -tuple of probability distributions

$\theta = (\nu_1, \dots, \nu_K)$. The agent aims at maximizing his rewards. He does not know θ but knows that it belongs to some set Θ . We assume for simplicity that $\Theta \subset \bar{\Theta}$, where $\bar{\Theta}$ denotes the set of all K -tuple of probability distributions on $[0, 1]$. We thus assume that the rewards are in $[0, 1]$.

For each arm k and all times $t \geq 1$, let $T_k(t) = \sum_{s=1}^t \mathbb{1}_{I_s=k}$ denote the number of times arm k was pulled from round 1 to round t , and by $X_{k,1}, X_{k,2}, \dots, X_{k,T_k(t)}$ the sequence of associated rewards. For an environment parameterized by $\theta = (\nu_1, \dots, \nu_K)$, let \mathbb{P}_θ denote the distribution on the probability space such that for any $k \in \{1, \dots, K\}$, the random variables $X_{k,1}, X_{k,2}, \dots$ are i.i.d. realizations of ν_k , and such that these K infinite sequence of random variables are independent. Let \mathbb{E}_θ denote the associated expectation.

Let $\mu_k = \int x d\nu_k(x)$ be the mean reward of arm k . Let $\mu^* = \max_{k \in \{1, \dots, K\}} \mu_k$ and fix an arm k^* such that $\mu_{k^*} = \mu^*$, that is k^* has the best expected reward. The suboptimality of arm k is measured by $\Delta_k = \mu^* - \mu_k$. The agent aims at minimizing its regret defined as the difference between the cumulative reward he would have got by always drawing the best arm and the cumulative reward he actually got. At time $n \geq 1$, its regret is thus

$$\hat{R}_n = \sum_{t=1}^n X_{k^*,t} - \sum_{t=1}^n X_{I_t, T_{I_t}(t)}. \quad (1)$$

The expectation of this regret has a simple expression in terms of the suboptimalities of the arms and the expected sampling times of the arms at time n . Precisely, we have

$$\begin{aligned} \mathbb{E}_\theta \hat{R}_n &= n\mu^* - \sum_{t=1}^n \mathbb{E}_\theta(\mu_{I_t}) = n\mu^* - \mathbb{E}_\theta \left(\sum_{k=1}^K T_k(n) \mu_k \right) \\ &= \mu^* \sum_{k=1}^K \mathbb{E}_\theta[T_k(n)] - \sum_{k=1}^K \mu_k \mathbb{E}_\theta[T_k(n)] = \sum_{k=1}^K \Delta_k \mathbb{E}_\theta[T_k(n)]. \end{aligned}$$

Other notions of regret exists in the literature: the quantity $\sum_{k=1}^K \Delta_k T_k(n)$ is called the pseudo regret and may be more practical to study, and the quantity $\max_k \sum_{t=1}^n X_{k,t} - \sum_{t=1}^n X_{I_t, T_{I_t}(t)}$ defines the regret in adversarial settings. Results and ideas we want to convey here are more suited to definition (II), and taking another definition of the regret would only bring some more technical intricacies.

Our main interest is the study of the *deviations* of the regret \hat{R}_n , i.e. the value of $\mathbb{P}_\theta(\hat{R}_n \geq x)$ when x is larger and of order of $\mathbb{E}_\theta \hat{R}_n$. If a policy has small deviations, it means that the regret is small with high probability and in particular, if the policy is used on some real data, it is very likely to be small on this specific dataset. Naturally, small deviations imply small expected regret since we have

$$\mathbb{E}_\theta \hat{R}_n \leq \mathbb{E}_\theta \max(\hat{R}_n, 0) = \int_0^{+\infty} \mathbb{P}_\theta(\hat{R}_n \geq x) dx.$$

To a lesser extent it is also interesting to study the deviations of the sampling times $T_n(k)$, as this shows the ability of a policy to match the best arm. Moreover our analysis is mostly based on results on the deviations of the sampling times, which then enables to derive results on the regret. We thus define below the notion of being f -upper tailed for both quantities.

Define $\mathbb{R}_+^* = \{x \in \mathbb{R} : x > 0\}$, and let $\Delta = \min_{k \neq k^*} \Delta_k$ be the gap between the best arm and second best arm.

Definition 1 (f -T and f -R). Consider a mapping $f : \mathbb{R} \rightarrow \mathbb{R}_+^*$. A policy has f -upper tailed sampling Times (in short, we will say that the policy is f -T) if and only if

$$\exists C, \tilde{C} > 0, \forall \theta \in \Theta \text{ such that } \Delta \neq 0,$$

$$\forall n \geq 2, \forall k \neq k^*, \mathbb{P}_\theta \left(T_k(n) \geq C \frac{\log n}{\Delta_k^2} \right) \leq \frac{\tilde{C}}{f(n)}.$$

A policy has f -upper tailed Regret (in short, f -R) if and only if

$$\exists C, \tilde{C} > 0, \forall \theta \in \Theta \text{ such that } \Delta \neq 0, \forall n \geq 2, \mathbb{P}_\theta \left(\hat{R}_n \geq C \frac{\log n}{\Delta} \right) \leq \frac{\tilde{C}}{f(n)}.$$

We will sometimes prefer to denote $f(n)$ -T (resp. $f(n)$ -R) instead of f -T (resp. f -R) for readability. Note also that, for sake of simplicity, we leave aside the degenerated case of Δ being null (i.e. when there are at least two optimal arms).

In this definition, we considered that the number K of arms is fixed, meaning that C and \tilde{C} may depend on K . The thresholds considered on $T_k(n)$ and \hat{R}_n directly come from known tight upper bounds on the expectation of these quantities for several policies. To illustrate this, let us recall the definition and properties of the popular UCB1 policy. Let $\hat{X}_{k,s} = \frac{1}{s} \sum_{t=1}^s X_{k,t}$ be the empirical mean of arm k after s pulls. In UCB1, the agent plays each arm once, and then (from $t \geq K + 1$), he plays

$$I_t \in \operatorname{argmax}_{k \in \{1, \dots, K\}} \left\{ \hat{X}_{k,T_k(t-1)} + \sqrt{\frac{2 \log t}{T_k(t-1)}} \right\}. \quad (2)$$

While the first term in the bracket ensures the exploitation of the knowledge gathered during steps 1 to $t - 1$, the second one ensures the exploration of the less sampled arms. For this policy, [ACBF02] proved:

$$\forall n \geq 3, \mathbb{E}[T_k(n)] \leq 12 \frac{\log n}{\Delta_k^2} \quad \text{and} \quad \mathbb{E}_\theta \hat{R}_n \leq 12 \sum_{k=1}^K \frac{\log n}{\Delta_k} \leq 12K \frac{\log n}{\Delta}.$$

[LR85] showed that these results cannot be improved up to numerical constants. [AMSO9] proved that UCB1 is \log^3 -T and \log^3 -R where \log^3 is the function $x \mapsto [\log(x)]^3$. Besides, they also study the case when $2 \log t$ is replaced by $\rho \log t$ in (2) with $\rho > 0$, and proved that this modified UCB1 is $\log^{2\rho-1}$ -T and $\log^{2\rho-1}$ -R for $\rho > 1/2$, and that $\rho = \frac{1}{2}$ is actually a critical value, since for $\rho < 1/2$, the

policy does not even have a logarithmic regret guarantee in expectation. Another variant of UCB1 proposed by Audibert et al. is to replace $2 \log t$ by $2 \log n$ in (2) when we want to have low and concentrated regret at a fixed given time n . We refer to it as UCB-H as its implementation requires the knowledge of the horizon n of the game. The behaviour of UCB-H on the time interval $[1, n]$ is significantly different to the one of UCB1, as UCB-H will explore much more at the beginning of the interval, and thus avoids exploiting the suboptimal arms on the early rounds. Audibert et al. showed that UCB-H is $n\text{-T}$ and $n\text{-R}$ (as it will be recalled in Theorem 8).

We now introduce the weak notion of f -upper tailed as this notion will be used to get our strongest impossibility results.

Definition 2 ($f\text{-wT}$ and $f\text{-wR}$). Consider a mapping $f : \mathbb{R} \rightarrow \mathbb{R}_+^*$. A policy has weak f -upper tailed sampling Times (in short, we will say that the policy is $f\text{-wT}$) if and only if

$$\forall \theta \in \Theta \text{ such that } \Delta \neq 0,$$

$$\exists C, \tilde{C} > 0, \forall n \geq 2, \forall k \neq k^*, \mathbb{P}_\theta \left(T_k(n) \geq C \frac{\log n}{\Delta_k^2} \right) \leq \frac{\tilde{C}}{f(n)}.$$

A policy has weak f -upper tailed Regret (in short, $f\text{-wR}$) if and only if

$$\forall \theta \in \Theta \text{ such that } \Delta \neq 0, \exists C, \tilde{C} > 0, \forall n \geq 2, \mathbb{P}_\theta \left(\hat{R}_n \geq C \frac{\log n}{\Delta} \right) \leq \frac{\tilde{C}}{f(n)}.$$

The only difference between $f\text{-T}$ and $f\text{-wT}$ (and between $f\text{-R}$ and $f\text{-wR}$) is the interchange of “ $\forall \theta$ ” and “ $\exists C, \tilde{C}$ ”. Consequently, a policy that is $f\text{-T}$ (respectively $f\text{-R}$) is $f\text{-T}$ (respectively $f\text{-wR}$). Let us detail the links between the $f\text{-T}$, $f\text{-R}$, $f\text{-wT}$ and $f\text{-wR}$.

Proposition 3. Assume that there exists $\alpha, \beta > 0$ such that $f(n) \leq \alpha n^\beta$ for any $n \geq 2$. We have

$$f\text{-T} \Rightarrow f\text{-R} \Rightarrow f\text{-wR} \Leftrightarrow f\text{-wT}.$$

The proof of this proposition is technical but rather straightforward. Note that we do not have $f\text{-R} \Rightarrow f\text{-T}$, because the agent may not regret having pulled a suboptimal arm if the latter has delivered good rewards. Note also that f is required to be at most polynomial: if not some rare events such as unlikely deviations of rewards towards their actual mean can not be neglected, and none of the implications hold in general (except, of course, $f\text{-R} \Rightarrow f\text{-wR}$ and $f\text{-T} \Rightarrow f\text{-wT}$).

3 Impossibility Result

From now on, we mostly deal with anytime policies (i.e. policies that do not have the knowledge of the horizon n) and the word policy (or algorithm) implicitly refers to anytime policy.

In the previous section, we have mentioned that for any $\rho > 1/2$, there is a variant of UCB1 (obtained by changing $2 \log t$ into $\rho \log t$ in (2)) which is $\log^{2\rho-1} \mathcal{T}$. This means that, for any $\alpha > 0$, there exists a $\log^\alpha \mathcal{T}$ policy, and hence $\log^\alpha \mathcal{R}$ policy. The following result shows that it is impossible to find an algorithm that would have better deviation properties than these UCB policies. For many usual settings (e.g., when Θ is the set $\bar{\Theta}$ of all K -tuples of measures on $[0, 1]$), with not so small probability, the agent gets stuck drawing a suboptimal arm he believes best. Precisely, this situation arises when simultaneously:

- (a) an arm k delivers payoffs according to a same distribution ν_k in two distinct environments θ and $\tilde{\theta}$,
- (b) arm k is optimal in θ but suboptimal in $\tilde{\theta}$,
- (c) in environment $\tilde{\theta}$, other arms may behave as in environment θ , i.e. with positive probability other arms deliver payoffs that are likely in both environments.

If the agent suspects that arm k delivers payoffs according to ν_k , he does not know if he has to pull arm k again (in case the environment is θ) or to pull the optimal arm of $\tilde{\theta}$. The other arms can help to point out the difference between θ and $\tilde{\theta}$, but then they have to be chosen often enough. This is in fact this kind of situation that has to be taken into account when balancing a policy between exploitation and exploration.

Our main result is the formalization of the leads given above. In particular, we give a rigorous description of conditions (a), (b) and (c). Let us first recall the following results, which are needed in the formalization of condition (c). One may look at [Rud86], p.121 for details (among others). Those who are not familiar with measure theory can skip to the non-formal explanation just after the results.

Theorem 4 (Lebesgue–Radon–Nikodym theorem). *Let μ_1 and μ_2 be σ -finite measures on a given measurable space. There exists a μ_2 -integrable function $\frac{d\mu_1}{d\mu_2}$ and a σ -finite measure m such that m and μ_2 are singular¹ and*

$$\mu_1 = \frac{d\mu_1}{d\mu_2} \cdot \mu_2 + m.$$

The density $\frac{d\mu_1}{d\mu_2}$ is unique up to a μ_2 -negligible event.

We adopt the convention that $\frac{d\mu_1}{d\mu_2} = +\infty$ on the complementary of the support of μ_2 .

Lemma 5. *We have*

- $\mu_1\left(\frac{d\mu_1}{d\mu_2} = 0\right) = 0$.
- $\mu_2\left(\frac{d\mu_1}{d\mu_2} > 0\right) > 0 \Leftrightarrow \mu_1\left(\frac{d\mu_2}{d\mu_1} > 0\right) > 0$.

¹ Two measures m_1 and m_2 on a measurable space (Ω, \mathcal{F}) are singular if and only if there exists two disjoint measurable sets A_1 and A_2 such that $A_1 \cup A_2 = \Omega$, $m_1(A_2) = 0$ and $m_2(A_1) = 0$.

Proof. The first point is a clear consequence of the decomposition $\mu_1 = \frac{d\mu_1}{d\mu_2} \cdot \mu_2 + m$ and of the convention mentioned above. For the second point, one can write by uniqueness of the decomposition:

$$\mu_2 \left(\frac{d\mu_1}{d\mu_2} > 0 \right) = 0 \Leftrightarrow \frac{d\mu_1}{d\mu_2} = 0 \text{ a.s.} \Leftrightarrow \mu_1 = m \Leftrightarrow \mu_1 \text{ and } \mu_2 \text{ are singular.}$$

And by symmetry of the roles of μ_1 and μ_2 :

$$\mu_2 \left(\frac{d\mu_1}{d\mu_2} > 0 \right) > 0 \Leftrightarrow \mu_1 \text{ and } \mu_2 \text{ are not singular} \Leftrightarrow \mu_1 \left(\frac{d\mu_2}{d\mu_1} > 0 \right) > 0.$$

Let us explain what these results has to do with condition (c).

One may be able to distinguish environment θ from $\tilde{\theta}$ if a certain arm ℓ delivers a payoff that is infinitely more likely in $\tilde{\theta}$ than in θ . This is for instance the case if $X_{\ell,t}$ is in the support of $\tilde{\nu}_\ell$ and not in the support of ν_ℓ , but our condition is more general. If the agent observes a payoff x from arm ℓ , the quantity $\frac{d\nu_\ell}{d\tilde{\nu}_\ell}(x)$ represents how much the observation of x is more likely in environment θ than in $\tilde{\theta}$. If ν_k and $\tilde{\nu}_k$ admit density functions (say, respectively, f and \tilde{f}) with respect to a common measure, then $\frac{d\nu_\ell}{d\tilde{\nu}_\ell}(x) = \frac{f(x)}{\tilde{f}(x)}$. Thus the agent will almost never make a mistake if he removes θ from possible environments when $\frac{d\nu_\ell}{d\tilde{\nu}_\ell}(x) = 0$. This may happen even if x is in both supports of ν_ℓ and $\tilde{\nu}_\ell$, for example if x is an atom of $\tilde{\nu}_\ell$ and not of ν_ℓ (i.e. $\tilde{\nu}_\ell(x) > 0$ and $\nu_\ell(x)=0$). On the contrary, if $\frac{d\nu_\ell}{d\tilde{\nu}_\ell}(x) > 0$ both environments θ and $\tilde{\theta}$ are likely and arm ℓ 's behaviour is both consistent with θ and $\tilde{\theta}$.

Now let us state the impossibility result. Here and throughout the paper we find it more convenient to denote $f \gg_{+\infty} g$ rather than the usual notation $g = o(f)$, which has the following meaning:

$$\forall \varepsilon > 0, \exists N \geq 0, \forall n \geq N, g(n) \leq \varepsilon f(n).$$

Theorem 6. Let $f : \mathbb{N} \rightarrow \mathbb{R}_+^*$ be greater than any \log^α , that is $f \gg_{+\infty} \log^\alpha$ for any $\alpha > 0$. Assume that there exists $\theta, \tilde{\theta} \in \Theta$, and $k \in \{1, \dots, K\}$ such that:

- (a) $\nu_k = \tilde{\nu}_k$,
- (b) k is the index of the best arm in θ but not in $\tilde{\theta}$,
- (c) $\forall \ell \neq k, \mathbb{P}_{\tilde{\theta}} \left(\frac{d\nu_\ell}{d\tilde{\nu}_\ell}(X_{\ell,1}) > 0 \right) > 0$.

Then there is no f -wT policy, and hence no f -R policy.

Let us give some hints of the proof (see Section 5 for details). The main idea is to consider a policy that would be f -wT, and in particular that would “work well” in environment θ in the sense given by the definition of f -wT. The proof exhibits a time N at which arm k , optimal in environment θ and thus often drawn with high \mathbb{P}_θ -probability, is drawn too many times (more than the logarithmic threshold $C \log(N)/\Delta_k^2$) with not so small $\mathbb{P}_{\tilde{\theta}}$ -probability, which shows the nonexistence of such a policy. More precisely, let n be large enough and

consider a time N of order $\log n$ and above the threshold. If the policy is f -w \mathcal{T} , at time N , sampling times of suboptimal arms are of order $\log N$ at most, with \mathbb{P}_θ -probability at least $1 - \tilde{C}/f(N)$. In this case, at time N , the draws are concentrated on arm k . So $T_k(N)$ is of order N , which is more than the threshold. This event holds with high \mathbb{P}_θ -probability. Now, from (a) and (c), we exhibit constants that are characteristic of the ability of arms $\ell \neq k$ to “behave as if in θ' ”: for some $0 < a, \eta < 1$, there is a subset ξ of this event such that $\mathbb{P}_\theta(\xi) \geq a^T$ for $T = \sum_{\ell \neq k} T_\ell(N)$ and for which $\frac{d\mathbb{P}_\theta}{d\mathbb{P}_{\theta'}}$ is lower bounded by η^T . The event ξ on which the arm k is sampled N times at least has therefore a $\mathbb{P}_{\theta'}$ -probability of order $(\eta a)^T$ at least. This concludes this sketchy proof since T is of order $\log N$, thus $(\eta a)^T$ is of order $\log^{\log(\eta a)} n$ at least.

Note that the conditions given in Theorem 6 are not very restrictive. The impossibility holds for very basic settings, and may hold even if the agent has great knowledge of the possible environments. For instance, the setting

$$K = 2 \text{ and } \Theta = \left\{ \left(\text{Ber}\left(\frac{1}{4}\right), \delta_{\frac{1}{2}} \right), \left(\text{Ber}\left(\frac{3}{4}\right), \delta_{\frac{1}{2}} \right) \right\},$$

where $\text{Ber}(p)$ denotes the Bernoulli distribution of parameter p and δ_x the Dirac measure on x , satisfies the three conditions of the theorem.

Nevertheless, the main interest of the result regarding the previous literature is the following corollary.

Corollary 7. *If Θ is the whole set $\bar{\Theta}$ of all K -tuples of measures on $[0, 1]$, then there is no f - \mathcal{R} policy, where f is any function such that $f \gg_{+\infty} \log^\alpha$ for all $\alpha > 0$.*

This corollary should be read in conjunction with the following result for UCB-H which, for a given n , plays at time $t \geq K + 1$,

$$I_t \in \operatorname{argmax}_{k \in \{1, \dots, K\}} \left\{ \hat{X}_{k, T_k(t-1)} + \sqrt{\frac{2 \log n}{T_k(t-1)}} \right\}.$$

Theorem 8. *For any $\beta > 0$, UCB-H is n^β - \mathcal{R} .*

For $\rho > 1$, Theorem 8 can easily be extended to the policy UCB-H(ρ) which starts by drawing each arm once, and then at time $t \geq K + 1$, plays

$$I_t \in \operatorname{argmax}_{k \in \{1, \dots, K\}} \left\{ \hat{X}_{k, T_k(t-1)} + \sqrt{\frac{\rho \log n}{T_k(t-1)}} \right\}. \quad (3)$$

Naturally, we have $n^\beta \gg_{n \rightarrow +\infty} \log^\alpha(n)$ for all $\alpha, \beta > 0$ but this does not contradict our theorem, since UCB-H(ρ) is not an *anytime* policy. UCB-H will work fine if the horizon n is known in advance, but may perform poorly at other rounds. In particular and as any policy, in view of Corollary 7, it cannot achieve anytime polynomial regret concentration.

Corollary 7 should also be read in conjunction with the following result for the policy UCB1(ρ) which starts by drawing each arm once, and then at time $t \geq K + 1$, plays

$$I_t \in \operatorname{argmax}_{k \in \{1, \dots, K\}} \left\{ \hat{X}_{k, T_k(t-1)} + \sqrt{\frac{\rho \log t}{T_k(t-1)}} \right\}. \quad (4)$$

Theorem 9. For any $\rho > 1/2$, UCB1(ρ) is $\log^{2\rho-1} \mathcal{R}$.

Thus, any improvements of existing algorithms which would for instance involve estimations of variance (see [AMS09]), of Δ_k , or of many characteristics of the distributions cannot beat the variants of UCB1 regarding deviations.

4 Positive Results

The intuition behind Theorem 6 suggests that, if one of the three conditions (a), (b), (c) does not hold, a robust policy would consist in the following: at each round and for each arm k , compute a distance between the empirical distribution of arm k and the set of distribution ν_k that makes arm k optimal in a given environment θ . As this distance decreases with our belief that k is the optimal arm, the policy consists in taking the k minimizing the distance. Thus, the agent chooses an arm that fits better a winning distribution ν_k . He cannot get stuck pulling a suboptimal arm because there are no environments $\tilde{\theta}$ with $\nu_k = \tilde{\nu}_k$ in which k would be suboptimal. More precisely, if there exists such an environment $\tilde{\theta}$, the agent is able to distinguish θ from $\tilde{\theta}$: during the first rounds, he pulls every arm and at least one of them will never behave as if in θ if the current environment is $\tilde{\theta}$. Thus, in $\tilde{\theta}$, he is able to remove θ from the set of possible environments Θ (remember that Θ is a parameter of the problem which is known by the agent).

Nevertheless such a policy cannot work in general, notably because of the three following limitations:

- If $\tilde{\theta}$ is the current environment and even if the agent has identified θ as impossible (i.e. $\frac{d\nu_k}{d\tilde{\nu}_k}(X_{k,1}) = 0$), there still could be other environments θ' that are arbitrary close to θ in which arm k is optimal and which the agent is not able to distinguish from $\tilde{\theta}$. This means that the agent may pull arm k too often because distribution $\tilde{\nu}_k = \nu_k$ is too close to a distribution ν'_k that makes arm k the optimal arm.
- The ability to identify environments as impossible relies on the fact that the event $\frac{d\nu_k}{d\tilde{\nu}_k}(X_{k,1}) > 0$ is almost sure under \mathbb{P}_θ (see Lemma 5). If the set of all environments Θ is uncountable, such a criterion can lead to exclude the actual environment. For instance, assume an agent has to distinguish a distribution among all Dirac measures δ_x ($x \in [0, 1]$) and the uniform probability λ over $[0, 1]$. Whatever the payoff x observed by the agent, he

will always exclude λ from the possible distributions, as x is always infinitely more likely under δ_x than under λ :

$$\forall x \in [0, 1], \frac{d\lambda}{d\delta_x}(x) = 0.$$

- On the other hand, the agent could legitimately consider an environment θ as unlikely if, for $\varepsilon > 0$ small enough, there exists $\tilde{\theta}$ such that $\frac{d\nu_k}{d\tilde{\nu}_k}(X_{k,1}) \leq \varepsilon$. Criterion (c) only considers as unlikely an environment θ when there exists $\tilde{\theta}$ such that $\frac{d\nu_k}{d\tilde{\nu}_k}(X_{k,1}) = 0$.

Despite these limitations, we give in this section sufficient conditions on Θ for such a policy to be robust. This is equivalent to finding conditions on Θ under which the converse of Theorem 6 holds, i.e. under which the fact one of the conditions (a), (b) or (c) does not hold implies the existence of a robust policy. This can also be expressed as finding which kind of knowledge of the environment enables to design anytime robust policies.

We estimate distributions of each arm by means of their empirical cumulative distribution functions, and distance between two c.d.f. is measured by the norm $\|\cdot\|_\infty$, defined by $\|f\|_\infty = \sup_{x \in [0,1]} |f(x)|$ where f is any function $[0, 1] \rightarrow \mathbb{R}$. The empirical c.d.f of arm k after having been pulled t times is denoted $\hat{F}_{k,t}$. The way we choose an arm at each round is based on confidence areas around $\hat{F}_{k,T_k(n-1)}$. We choose the greater confidence level (GCL) such that there is still an arm k and a winning distribution ν_k such that F_{ν_k} , the c.d.f. of ν_k , is in the area of $\hat{F}_{k,T_k(n-1)}$. We then select the corresponding arm k . By means of Massart's inequality (1990), this leads to the c.d.f. based algorithm described in Figure 1. Let Θ_k denote the set $\{\theta \in \Theta \mid k \text{ is the optimal arm in } \theta\}$, i.e. the set of environments that makes k the index of the optimal arm.

Proceed as follows:

- Draw each arm once.
- Remove each $\theta \in \Theta$ such that there exists $\tilde{\theta} \in \Theta$ and $\ell \in \{1, \dots, K\}$ with $\frac{d\nu_\ell}{d\tilde{\nu}_\ell}(X_{\ell,1}) = 0$.
- Then at each round t , play an arm

$$I_t \in \operatorname{argmin}_{k \in \{1, \dots, K\}} T_k(t-1) \inf_{\theta \in \Theta_k} \|\hat{F}_{k,T_k(t-1)} - F_{\nu_k}\|_\infty^2.$$

Fig. 1. A c.d.f.-based algorithm: GCL

4.1 Θ is Finite

When Θ is finite the limitations presented above do not really matter, so that the converse of Theorem 6 is true and our algorithm is robust.

Theorem 10. *Assume that Θ is finite and that for all $\theta = (\nu_1, \dots, \nu_K)$, $\tilde{\theta} = (\tilde{\nu}_1, \dots, \tilde{\nu}_K) \in \Theta$, and all $k \in \{1, \dots, K\}$, at least one of the following holds:*

- $\nu_k \neq \tilde{\nu}_k$,
- k is suboptimal in θ , or is optimal in $\tilde{\theta}$.
- $\exists \ell \neq k, \mathbb{P}_{\tilde{\theta}} \left(\frac{d\nu_\ell}{d\tilde{\nu}_\ell}(X_{\ell,1}) > 0 \right) = 0$.

Then GCL is n^β - \mathcal{T} (and hence n^β - \mathcal{R}) for all $\beta > 0$.

4.2 Bernoulli Laws

We assume that any ν_k ($k \in \{1, \dots, K\}$, $\theta \in \Theta$) is a Bernoulli law, and denote by μ_k its parameter. We also assume that there exists $\gamma \in (0, 1)$ such that $\mu_k \in [\gamma, 1]$ for all k and all θ .² Moreover we may denote arbitrary environments $\theta, \tilde{\theta}$ by $\theta = (\mu_1, \dots, \mu_K)$ and $\tilde{\theta} = (\tilde{\mu}_1, \dots, \tilde{\mu}_K)$.

In this case $\frac{d\nu_\ell}{d\tilde{\nu}_\ell}(1) = \frac{\mu_\ell}{\tilde{\mu}_\ell} > 0$, so that for any $\theta, \tilde{\theta} \in \Theta$ and any $l \in \{1, \dots, K\}$ one has

$$\mathbb{P}_{\tilde{\theta}} \left(\frac{d\nu_\ell}{d\tilde{\nu}_\ell}(X_{\ell,1}) > 0 \right) \geq \mathbb{P}_{\tilde{\theta}}(X_{\ell,1} = 1) = \tilde{\mu}_\ell > 0.$$

Therefore condition (c) of Theorem 6 holds, and the impossibility result only relies on conditions (a) and (b). Our algorithm can be made simpler: there is no need to try to exclude unlikely environments and computing the empirical c.d.f. is equivalent to computing the empirical mean (see Figure 2). The theorem and its converse are expressed as follows. We will refer to our policy as GCL-B as it looks for the environment matching the observations at the Greatest Confidence Level, in the case of Bernoulli distributions.

Proceed as follows:

- Draw each arm once.
- Then at each round t , play an arm

$$I_t \in \operatorname{argmin}_{k \in \{1, \dots, K\}} T_k(t-1) \inf_{\theta \in \Theta_k} \left(\mu_k - \hat{X}_{k,T_k(t-1)} \right)^2.$$

Fig. 2. A c.d.f.-based algorithm in case of Bernoulli laws: GCL-B

Theorem 11. For any $\theta \in \Theta$ and any $k \in \{1, \dots, K\}$, let us set

$$d_k = \inf_{\tilde{\theta} \in \Theta_k} |\mu_k - \tilde{\mu}_k|.$$

GCL-B is such that: $\forall \beta > 0, \exists C, \tilde{C} > 0, \forall \theta \in \Theta, \forall n \geq 2$,

$$\forall k \in \{1, \dots, K\}, \mathbb{P}_\theta \left(T_k(n) \geq \frac{C \log n}{d_k^2} \right) \leq \frac{\tilde{C}}{n^\beta}.$$

Let $f : \mathbb{N}^* \rightarrow \mathbb{R}_+^*$ be greater than any \log^α , that is $\forall \alpha > 0, f \gg_{+\infty} \log^\alpha$. If there exists k such that

² The result also holds if all parameters μ_k are in a given interval $[0, \gamma]$, $\gamma \in (0, 1)$.

$$(a') \inf_{\theta \in \Theta \setminus \Theta_k} d_k = \inf_{\substack{\theta \in \Theta_k \\ \tilde{\theta} \in \Theta \setminus \Theta_k}} |\mu_k - \tilde{\mu}_k| = 0,$$

then there is no policy such that:

$$\exists C, \tilde{C} > 0, \forall \theta \in \Theta, \forall n \geq 2, \forall k \neq k^*, \mathbb{P}_\theta(T_k(n) \geq C \log n) \leq \frac{\tilde{C}}{f(n)}.$$

Note that we do not adopt the former definitions of robustness ($f\text{-R}$ and $f\text{-T}$), because the significant term here is d_k (and not Δ_k ³, which represents the distance between Θ_k and $\Theta \setminus \Theta_k$). Indeed robustness lies on the ability to distinguish environments, and this ability is all the more stronger as the distance between the parameters of these environments is greater. Provided that the density $\frac{d\nu}{d\tilde{\nu}}$ is uniformly bounded away from zero, the theorem holds for any parametric model, with d_k being defined with a norm on the space of parameters (instead of $|.|$). Note also that the second part of the theorem is a bit weaker than Theorem 6, because of the interchange of “ $\forall \theta$ ” and “ $\exists C, \tilde{C}$ ”. The reason for this is that condition (a) is replaced by a weaker assumption: ν_k does not equal $\tilde{\nu}_k$, but condition (a') means that such ν_k and $\tilde{\nu}_k$ can be chosen arbitrarily close.

4.3 μ^* Is Known

This section shows that the impossibility result also breaks down if μ^* is known by the agent. This situation is formalized as μ^* being constant over Θ . Conditions (a) and (b) of Theorem 6 do not hold: if a distribution ν_k makes arm k optimal in an environment θ , it is still optimal in any environment $\tilde{\theta}$ such that $\tilde{\nu}_k = \nu_k$. In this case, our algorithm can be made simpler (see Figure 3). At each round we choose the greatest confidence level such that at least one empirical mean $\hat{X}_{k,T_k(t-1)}$ has μ^* in its confidence interval, and select the corresponding arm k . This is similar to the previous algorithm, deviations being evaluated according to Hoeffding’s inequality instead of Massart’s one. We will refer to this policy as GCL*.

Proceed as follows:

- Draw each arm once.
- Then at each round t , play an arm

$$I_t \in \operatorname{argmin}_{k \in \{1, \dots, K\}} T_k(t-1) \left(\mu^* - \hat{X}_{k,T_k(t-1)} \right)^2.$$

Fig. 3. GCL*: a variant of c.d.f.-based algorithm when μ^* is known

Theorem 12. When μ^* is known, GCL* is $n^\beta\text{-T}$ (and hence $n^\beta\text{-R}$) for all $\beta > 0$.

³ There is no need to leave aside the case of $d_k = 0$: with the convention $\frac{1}{0} = +\infty$, the corresponding event has zero probability.

5 Proof of Theorem 6

Let us first notice that we can remove the Δ_k^2 denominator in the the definition of $f\text{-w}\mathcal{T}$ without loss of generality. This would not be possible for the $f\text{-T}$ definition owing to the different position of “ $\forall\theta$ ” with respect to “ $\exists C, \tilde{C}$ ”.

Thus, a policy is $f\text{-w}\mathcal{T}$ if and only if

$\forall\theta \in \Theta$ such that $\Delta \neq 0$,

$$\exists C, \tilde{C} > 0, \forall n \geq 2, \forall k \neq k^*, \mathbb{P}_\theta(T_k(n) \geq C \log n) \leq \frac{\tilde{C}}{f(n)}.$$

Let us assume that the policy has the f -upper tailed property in θ , i.e., there exists $C, \tilde{C} > 0$

$$\forall N \geq 2, \forall \ell \neq k, \mathbb{P}_\theta(T_\ell(N) \geq C \log N) \leq \frac{\tilde{C}}{f(N)}. \quad (5)$$

Let us show that this implies that the policy cannot have also the f -upper tailed property in $\tilde{\theta}$. To prove the latter, it is enough to show that for any $C', \tilde{C}' > 0$

$$\exists n \geq 2, \mathbb{P}_{\tilde{\theta}}(T_k(n) \geq C' \log n) > \frac{\tilde{C}'}{f(n)}. \quad (6)$$

since k is suboptimal in environment $\tilde{\theta}$. Note that proving (6) for $C' = C$ is sufficient. Indeed if (6) holds for $C' = C$, it a fortiori holds for $C' < C$. Besides, when $C' > C$, (5) holds for C replaced by C' , and we are thus brought back to the situation when $C = C'$. So we only need to lower bound $\mathbb{P}_{\tilde{\theta}}(T_k(n) \geq C \log n)$.

From Lemma 5, $\mathbb{P}_{\tilde{\theta}}\left(\frac{d\nu_\ell}{d\nu_k}(X_{\ell,1}) > 0\right) > 0$ is equivalent to $\mathbb{P}_\theta\left(\frac{d\tilde{\nu}_\ell}{d\nu_k}(X_{\ell,1}) > 0\right) > 0$. By independence of $X_{1,1}, \dots, X_{K,1}$ under \mathbb{P}_θ , condition (c) in the theorem may be written as

$$\mathbb{P}_\theta\left(\prod_{\ell \neq k} \frac{d\tilde{\nu}_\ell}{d\nu_\ell}(X_{\ell,1}) > 0\right) > 0.$$

Since $\left\{\prod_{\ell \neq k} \frac{d\tilde{\nu}_\ell}{d\nu_\ell}(X_{\ell,1}) > 0\right\} = \cup_{m \geq 2} \left\{\prod_{\ell \neq k} \frac{d\tilde{\nu}_\ell}{d\nu_\ell}(X_{\ell,1}) \geq \frac{1}{m}\right\}$, this readily implies that

$$\exists \eta \in (0, 1), \mathbb{P}_\theta\left(\prod_{\ell \neq k} \frac{d\tilde{\nu}_\ell}{d\nu_\ell}(X_{\ell,1}) \geq \eta\right) > 0.$$

Let $a = \mathbb{P}_\theta\left(\prod_{\ell \neq k} \frac{d\tilde{\nu}_\ell}{d\nu_\ell}(X_{\ell,1}) \geq \eta\right)$.

Let us take n large enough such that $N = \lfloor 4C \log n \rfloor$ satisfies $N < n$, $C \log N < \frac{N}{2K}$ and $f(n)\eta^t(a^t - \frac{(K-1)\tilde{C}}{f(N)}) > \tilde{C}'$ for $t = \lfloor C \log N \rfloor$. For any \tilde{C}' , such a n does exist since $f \gg_{+\infty} \log^\alpha$ for any $\alpha > 0$.

The idea is that if until round N , arms $\ell \neq k$ have a behaviour that is typical of θ , then the arm k (which is suboptimal in $\tilde{\theta}$) may be pulled about $C \log n$ times at round N . Precisely, we prove that $\forall \ell \neq k, \mathbb{P}_\theta(T_\ell(N) \geq C \log N) \leq \frac{\tilde{C}}{f(N)}$

implies $\mathbb{P}_{\tilde{\theta}}(T_k(n) \geq C' \log n) > \frac{\tilde{C}'}{f(n)}$. Let $A_t = \cap_{s=1..t} \left\{ \prod_{\ell \neq k} \frac{d\tilde{\nu}_\ell}{d\nu_\ell}(X_{\ell,s}) \geq \eta \right\}$. By independence and by definition of a , we have $\mathbb{P}_\theta(A_t) = a^t$. We also have

$$\begin{aligned} \mathbb{P}_{\tilde{\theta}}(T_k(n) \geq C \log n) &\geq \mathbb{P}_{\tilde{\theta}}\left(T_k(N) \geq \frac{N}{2}\right) \\ &\geq \mathbb{P}_{\tilde{\theta}}\left(\bigcap_{\ell \neq k} \left\{ T_\ell(N) \leq \frac{N}{2K} \right\}\right) \\ &\geq \mathbb{P}_{\tilde{\theta}}\left(\bigcap_{\ell \neq k} \left\{ T_\ell(N) < C \log N \right\}\right) \\ &\geq \mathbb{P}_{\tilde{\theta}}\left(A_t \cap \left\{ \bigcap_{\ell \neq k} \left\{ T_\ell(N) < C \log N \right\} \right\}\right). \end{aligned}$$

Introduce $B_N = \bigcap_{\ell \neq k} \{T_\ell(N) < C \log N\}$, and the function q such that

$$\mathbb{1}_{A_t \cap B_N} = q((X_{\ell,s})_{\ell \neq k, s=1..t}, (X_{k,s})_{s=1..N}).$$

Since $\tilde{\nu}_k = \nu_k$, by definition of A_t and by standard properties of density functions $\frac{d\tilde{\nu}_\ell}{d\nu_\ell}$, we have

$$\begin{aligned} &\mathbb{P}_{\tilde{\theta}}\left(A_t \cap \left\{ \bigcap_{\ell \neq k} \{T_\ell(N) < C \log N\} \right\}\right) \\ &= \int q((x_{\ell,s})_{\ell \neq k, s=1..t}, (x_{k,s})_{s=1..N}) \prod_{\substack{\ell \neq k \\ s=1..t}} d\tilde{\nu}_\ell(x_{\ell,s}) \prod_{s=1..N} d\tilde{\nu}_k(x_{k,s}) \\ &\geq \eta^t \int q((x_{\ell,s})_{\ell \neq k, s=1..t}, (x_{k,s})_{s=1..N}) \prod_{\substack{\ell \neq k \\ s=1..t}} d\nu_\ell(x_{\ell,s}) \prod_{s=1..N} d\nu_k(x_{k,s}) \\ &= \eta^t \mathbb{P}_\theta\left(A_t \cap \left\{ \bigcap_{\ell \neq k} \{T_\ell(N) < C \log N\} \right\}\right) \geq \eta^t \left(a^t - \frac{(K-1)\tilde{C}'}{f(N)} \right) > \frac{\tilde{C}'}{f(n)}, \end{aligned}$$

where the one before last inequality relies on a union bound with (5) and $\mathbb{P}_\theta(A_t) = a^t$, and the last inequality uses the definition of n . We have thus proved that (6) holds, and thus the policy cannot have the f -upper tailed property simultaneously in environment θ and $\tilde{\theta}$.

References

- [ACBF02] Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multi-armed bandit problem. *Mach. Learn.* 47(2-3), 235–256 (2002)
- [Agr95] Agrawal, R.: Sample mean based index policies with $o(\log n)$ regret for the multi-armed bandit problem. *Advances in Applied Mathematics* 27, 1054–1078 (1995)

- [AMS09] Audibert, J.-Y., Munos, R., Szepesvári, C.: Exploration-exploitation trade-off using variance estimates in multi-armed bandits. *Theoretical Computer Science* 410(19), 1876–1902 (2009)
- [BMSS09] Bubeck, S., Munos, R., Stoltz, G., Szepesvari, C.: Online optimization in X-armed bandits. In: *Advances in Neural Information Processing Systems*, vol. 21, pp. 201–208 (2009)
- [BSS09] Babaioff, M., Sharma, Y., Slivkins, A.: Characterizing truthful multi-armed bandit mechanisms: extended abstract. In: *Proceedings of the Tenth ACM Conference on Electronic Commerce*, pp. 79–88. ACM, New York (2009)
- [BV08] Bergemann, D., Valimaki, J.: Bandit problems. In: *The New Palgrave Dictionary of Economics*, 2nd edn. Macmillan Press, Basingstoke (2008)
- [CM07] Coquelin, P.A., Munos, R.: Bandit algorithms for tree search. In: *Uncertainty in Artificial Intelligence* (2007)
- [DK09] Devanur, N.R., Kakade, S.M.: The price of truthfulness for pay-per-click auctions. In: *Proceedings of the Tenth ACM Conference on Electronic Commerce*, pp. 99–106. ACM, New York (2009)
- [GW06] Gelly, S., Wang, Y.: Exploration exploitation in go: UCT for Monte-Carlo go. In: *Online trading between exploration and exploitation Workshop, Twentieth Annual Conference on Neural Information Processing Systems, NIPS 2006* (2006)
- [Hol92] Holland, J.H.: *Adaptation in natural and artificial systems*. MIT Press, Cambridge (1992)
- [Kle05] Kleinberg, R.D.: Nearly tight bounds for the continuum-armed bandit problem. In: *Advances in Neural Information Processing Systems*, vol. 17, pp. 697–704 (2005)
- [KS06] Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) *ECML 2006. LNCS (LNAI)*, vol. 4212, pp. 282–293. Springer, Heidelberg (2006)
- [KSU08] Kleinberg, R., Slivkins, A., Upfal, E.: Multi-armed bandits in metric spaces. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pp. 681–690 (2008)
- [LPT04] Lamberton, D., Pagès, G., Tarrès, P.: When can the two-armed bandit algorithm be trusted? *Annals of Applied Probability* 14(3), 1424–1454 (2004)
- [LR85] Lai, T.L., Robbins, H.: Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics* 6, 4–22 (1985)
- [Mas90] Massart, P.: The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality. *The Annals of Probability* 18(3), 1269–1283 (1990)
- [Rob52] Robbins, H.: Some aspects of the sequential design of experiments. *Bulletin of the American Mathematics Society* 58, 527–535 (1952)
- [Rud86] Rudin, W.: *Real and complex analysis*, 3rd edn. McGraw-Hill Inc., New York (1986)
- [SB98] Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT Press, Cambridge (1998)

On Upper-Confidence Bound Policies for Switching Bandit Problems

Aurélien Garivier and Eric Moulines

Institut Telecom, Telecom ParisTech, Laboratoire LTCI, CNRS UMR 5141
46 rue Barrault, 75634 Paris Cedex 13

Abstract. Many problems, such as cognitive radio, parameter control of a scanning tunnelling microscope or internet advertisement, can be modelled as non-stationary bandit problems where the distributions of rewards changes abruptly at unknown time instants. In this paper, we analyze two algorithms designed for solving this issue: *discounted UCB* (D-UCB) and *sliding-window UCB* (SW-UCB). We establish an upper-bound for the expected regret by upper-bounding the expectation of the number of times suboptimal arms are played. The proof relies on an interesting Hoeffding type inequality for self normalized deviations with a random number of summands. We establish a lower-bound for the regret in presence of abrupt changes in the arms reward distributions. We show that the discounted UCB and the sliding-window UCB both match the lower-bound up to a logarithmic factor. Numerical simulations show that D-UCB and SW-UCB perform significantly better than existing soft-max methods like EXP3.S.

1 Introduction

Multi-armed bandit (MAB) problems, modelling allocation issues under uncertainty, are fundamental to stochastic decision theory. The archetypal MAB problem may be stated as follows: there is a bandit with K independent arms. At each time step, the agent chooses one arm and receives a reward accordingly. In the stationary case, the distribution of the rewards are initially unknown, but are assumed to remain constant during all games. The agent aims at minimizing the expected *regret* over T rounds, which is defined as the expectation of the difference between the total reward obtained by playing the best arm and the total reward obtained by using the algorithm. For several algorithms in the literature (e.g. [20, 1]), as the number of plays T tends to infinity, the expected total reward asymptotically approaches that of playing a policy with the highest expected reward, and the regret grows as the logarithm of T . More recently, finite-time bounds for the regret and improvements have been derived (see [5, 2, 16]), but those improvements do not address the issue of non-stationarity.

Though the stationary formulation of the MAB allows to address exploration versus exploitation challenges in a intuitive and elegant way, it may fail to be adequate to model an evolving environment where the reward distributions undergo changes in time. As an example, in the cognitive medium radio access

problem [19], a user wishes to opportunistically exploit the availability of an empty channel in a multiple channel system; the reward is the availability of the channel, whose distribution is unknown to the user. Another application is real-time optimization of websites by targetting relevant content at individuals, and maximizing the general interest by learning and serving the most popular content (such situations have been considered in the recent Exploration versus Exploitation (EvE) PASCAL challenge by [14], see also [18] and the references therein). These examples illustrate the limitations of the stationary MAB models. The probability that a given channel is available is likely to change in time. The news stories a visitor of a website is most likely to be interested in vary in time.

To model such situations, non-stationary MAB problems have been considered (see [17, 14, 22, 24]), where distributions of rewards may change in time. Motivated by the problems cited above, and following a paradigm widely used in the change-point detection literature (see [12, 21] and references therein), we focus on non-stationary environments where the distributions of the rewards undergo abrupt changes. We show in the following that, as expected, policies tailored for the stationary case fail to track changes of the best arm.

Section 2 contains the formal presentation of the non-stationary setting we consider, together with two algorithms addressing this exploration/exploitation dilemma : D-UCB and SW-UCB. D-UCB had been proposed in [17] with empirical evidence of efficiency, but no theoretical analysis. SW-UCB is a new UCB-like algorithm that appears to perform slightly better in switching environments. In Section 3, we provide upper-bounds on the performance of D-UCB and SW-UCB; moreover, we provide a lower-bound on the performance of any algorithm in abruptly changing environments, that almost matches the upper-bounds. As a by-product, we show that any policy (like UCB-1) that achieves a logarithmic regret in the stationary case cannot reach a regret of order smaller than $T/\log(T)$ in the presence of switches. D-UCB is analyzed in Section 4, it relies on a novel deviation inequality for self-normalized averages with random number of summands which is stated in Section 7 together with some technical results. A lower bound on the regret of any algorithm in an abruptly changing environment is given in Section 5. In Section 6, two simple Monte-Carlo experiments are presented to support our findings.

2 Algorithms

In the sequel, we assume that the set of arms is $\{1, \dots, K\}$, and that the rewards $\{X_t(i)\}_{t \geq 1}$ for arm $i \in \{1, \dots, K\}$ are modeled by a sequence of independent random variables from potentially different distributions (unknown to the user) which may vary across time but remain bounded by $B > 0$. For each $t > 0$, we denote by $\mu_t(i)$ the expectation of the reward $X_t(i)$ for arm i . Let i_t^* be the arm with highest expected reward at time t (in case of ties, let i_t^* be one of the arms with highest expected rewards). The regret of a policy π is defined as the expected difference between the total rewards collected by the optimal policy

π^* (playing at each time instant the arm i_t^*) and the total rewards collected by the policy π . Note that, in this paper, the non-stationary regret is not defined with respect to the best arm on average, but with respect to a strategy tracking the best arm at each step (this notion of regret is similar to the “regret against arbitrary strategies” introduced in Section 8 of [3] for the non-stochastic bandit problem).

We consider *abruptly changing environments*: the distributions of rewards remain constant during periods and change at unknown time instants called *breakpoints* (which do not depend on the policy of the player or on the sequence of rewards). In the following, we denote by Υ_T the number of breakpoints in the reward distributions that occur before time T . Another type of non-stationary MAB, where the distribution of rewards changes continuously, is considered in [22].

Standard soft-max and UCB policies are not appropriate for abruptly changing environments: as stressed in [14], “empirical evidence shows that their Exploration versus Exploitation trade-off is not appropriate for abruptly changing environments”. To address this problem, several methods have been proposed.

In the family of softmax action selection policies, [3] and [8, 9] have proposed an adaptation of the Fixed-Share algorithm referred to as EXP3.S (see [15, 6] and the references therein). Theorem 8.1 and Corollary 8.3 in [3] state that when EXP3.S is tuned properly (which requires in particular that Υ_T is known in advance), the expected regret satisfies $\mathbb{E}_\pi[R_T] \leq 2\sqrt{e-1}\sqrt{KT(\Upsilon_T \log(KT) + e)}$. Despite the fact that it holds uniformly over all reward distributions, such an upper-bound may seem deceptive in comparison to the stationary case; the rate $O(\sqrt{T} \log T)$ is much larger than the $O(\log T)$ achievable for a fixed distribution in the absence of changes. But actually, we prove in Section 5 that no policy can always achieve an average fixed-game regret smaller than $O(\sqrt{T})$ in the non-stationary case. Hence, EXP3.S matches the best achievable rate up to a factor $\sqrt{\log T}$. By construction, this algorithm can as well be used in an adversarial setup; but, in a stochastic environment, it is not guaranteed to be optimal (think that, in the stationary case, UCB outperforms EXP3 in the stochastic setup), and specific methods based on probabilistic estimation have to be considered.

In fact, in the family of UCB policies, several attempts have been made; see for examples [22] and [17]. In particular, [17] have proposed an adaptation of the UCB policies that relies on a discount factor $\gamma \in (0, 1)$. This policy constructs an UCB $\bar{X}_t(\gamma, i) + c_t(\gamma, i)$ for the instantaneous expected reward, where the discounted empirical average is given by

$$\bar{X}_t(\gamma, i) = \frac{1}{N_t(\gamma, i)} \sum_{s=1}^t \gamma^{t-s} X_s(i) \mathbb{1}_{\{I_s=i\}}, \quad N_t(\gamma, i) = \sum_{s=1}^t \gamma^{t-s} \mathbb{1}_{\{I_s=i\}},$$

where the discounted exploration bonus is $c_t(\gamma, i) = 2B\sqrt{\xi \log n_t(\gamma)/N_t(\gamma, i)}$, with $n_t(\gamma) = \sum_{i=1}^K N_t(\gamma, i)$, for an appropriate parameter ξ . Using these notations, discounted-UCB (D-UCB) is defined in Algorithm 1. For $\gamma = 1$, D-UCB boils down to the standard UCB-1 algorithm.

Algorithm 1. Discounted UCB

for t from 1 to K , play arm $I_t = t$;
 for t from $K + 1$ to T , play arm

$$I_t = \arg \max_{1 \leq i \leq K} \bar{X}_t(\gamma, i) + c_t(\gamma, i).$$

In order to estimate the instantaneous expected reward, the D-UCB policy averages past rewards with a discount factor giving more weight to recent observations. We propose in this paper a more abrupt variant of UCB where averages are computed on a fixed-size horizon. At time t , instead of averaging the rewards over the whole past with a discount factor, *sliding-window UCB* relies on a local empirical average of the observed rewards, using only the τ last plays. Specifically, this algorithm constructs an UCB $\bar{X}_t(\tau, i) + c_t(\tau, i)$ for the instantaneous expected reward; the local empirical average is given by

$$\bar{X}_t(\tau, i) = \frac{1}{N_t(\tau, i)} \sum_{s=t-\tau+1}^t X_s(i) \mathbb{1}_{\{I_s=i\}}, \quad N_t(\tau, i) = \sum_{s=t-\tau+1}^t \mathbb{1}_{\{I_s=i\}},$$

and the exploration bonus is defined as $c_t(\tau, i) = B \sqrt{\xi \log(t \wedge \tau) / (N_t(\tau, i))}$, where $t \wedge \tau$ denotes the minimum of t and τ , and ξ is an appropriate constant. The policy defined in Algorithm 2 is denoted *Sliding-Window UCB* (SW-UCB).

Algorithm 2. Sliding-Window UCB

for t from 1 to K , play arm $I_t = t$;
 for t from $K + 1$ to T , play arm

$$I_t = \arg \max_{1 \leq i \leq K} \bar{X}_t(\tau, i) + c_t(\tau, i),$$

3 Regret Bounds

In this section, we provide upper-bounds on the regret of D-UCB and SW-UCB, as well as an almost matching lower-bound on the regret of any algorithm facing an abruptly changing environment.

Let γ_T denote the number of breakpoints before time T , and let $\tilde{N}_T(i) = \sum_{t=1}^T \mathbb{1}_{\{I_t=i \neq i_t^*\}}$ denote the number of times arm i was played when it was not the best arm during the T first rounds. Denote by $\Delta\mu_T(i)$ the minimum of the difference of expected reward of the best arm $\mu_t(i_t^*)$ and the expected reward $\mu_t(i)$ of arm i for all times $t \in \{1, \dots, T\}$ such that arm i is not optimal:

$$\Delta\mu_T(i) = \min \left\{ \mu_t(i_t^*) - \mu_t(i) : t \in \{1, \dots, T\}, \mu_t(i) < \mu_t(i_t^*) \right\}. \quad (1)$$

We denote by \mathbb{P}_γ and \mathbb{E}_γ the probability distribution and expectation under the policy D-UCB using the discount factor γ . As the expected regret is

$$\mathbb{E}_\gamma [R_T] = \mathbb{E}_\gamma \left[\sum_{t=1}^T \sum_{i: \mu_t(i) < \mu_t(i_t^*)} (X_t(i_t^*) - X_t(i)) \mathbb{1}_{\{I_t=i\}} \right] \leq B \sum_{i=1}^K \mathbb{E}_\gamma [\tilde{N}_T(i)] ,$$

it is sufficient to upper-bound the expected number of times an arm i is played when this arm is suboptimal.

Theorem 1. *Let $\xi \in (1/2, 1)$ and $\gamma \in (1/2, 1)$. For any $T \geq 1$ and for any arm $i \in \{1, \dots, K\}$:*

$$\mathbb{E}_\gamma [\tilde{N}_T(i)] \leq C_1 T(1-\gamma) \log \frac{1}{1-\gamma} + C_2 \frac{\Upsilon_T}{1-\gamma} \log \frac{1}{1-\gamma} , \quad (2)$$

where

$$C_1 = \frac{32\sqrt{2}B^2\xi}{\gamma^{1/(1-\gamma)}(\Delta\mu_T(i))^2} + \frac{4}{(1-\frac{1}{e}) \log \left(1 + 4\sqrt{1-1/2\xi} \right)}$$

and

$$C_2 = \frac{\gamma-1}{\log(1-\gamma) \log \gamma} \times \log((1-\gamma)\xi \log n_K(\gamma)) .$$

When γ goes to 1, $C_2 \rightarrow 1$ and

$$C_1 \rightarrow \frac{16eB^2\xi}{(\Delta\mu_T(i))^2} + \frac{2}{(1-e^{-1}) \log \left(1 + 4\sqrt{1-1/2\xi} \right)} .$$

Algorithm SW-UCB shows a similar behavior, but the absence of infinite memory makes it slightly more suited to abrupt changes of the environment. Denote by \mathbb{P}_τ and \mathbb{E}_τ the probability distribution and expectation under policy SW-UCB with window size τ . The following bound holds:

Theorem 2. *Let $\xi > 1/2$. For any integer τ and any arm $i \in \{1, \dots, K\}$,*

$$\mathbb{E}_\tau [\tilde{N}_T(i)] \leq C(\tau) \frac{T \log \tau}{\tau} + \tau \Upsilon_T + \log^2(\tau) , \quad (3)$$

where

$$\begin{aligned} C(\tau) &= \frac{4B^2\xi}{(\Delta\mu_T(i))^2} \frac{\lceil T/\tau \rceil}{T/\tau} + \frac{2}{\log \tau} \left[\frac{\log(\tau)}{\log(1 + 4\sqrt{1-(2\xi)^{-1}})} \right] \\ &\rightarrow \frac{4B^2\xi}{(\Delta\mu_T(i))^2} + \frac{2}{\log(1 + 4\sqrt{1-(2\xi)^{-1}})} \quad \text{as } \tau \text{ and } T/\tau \text{ go to infinity.} \end{aligned}$$

3.1 Tuning the Parameters

If horizon T and the growth rate of the number of breakpoints Υ_T are known in advance, the discount factor γ can be chosen so as to minimize the RHS in Equation 2. Choosing $\gamma = 1 - (4B)^{-1}\sqrt{\Upsilon_T/T}$ yields $\mathbb{E}_\gamma [\tilde{N}_T(i)] = O(\sqrt{T\Upsilon_T \log T})$. Assuming that $\Upsilon_T = O(T^\beta)$ for some $\beta \in [0, 1]$, the regret is upper-bounded as $O(T^{(1+\beta)/2} \log T)$. In particular, if $\beta = 0$, the number of breakpoints Υ_T is upper-bounded by T independently of T , taking $\gamma = 1 - (4B)^{-1}\sqrt{\Upsilon_T/T}$, the regret is bounded by $O(\sqrt{T\Upsilon_T \log T})$. Thus, D-UCB matches the lower-bound of Theorem 3 stated below, up to a factor $\log T$.

Similary, choosing $\tau = 2B\sqrt{T \log(T)/\Upsilon_T}$ in SW-UCB yields $\mathbb{E}_\tau [\tilde{N}_T(i)] = O(\sqrt{T\Upsilon_T \log T})$. Assuming that $\Upsilon_T = O(T^\beta)$ for some $\beta \in [0, 1]$, the average regret is upper-bounded as $O(T^{(1+\beta)/2} \sqrt{\log T})$. If $\beta = 0$, the number of breakpoints Υ_T is upper-bounded by T independently of T , then with $\tau = 2B\sqrt{T \log(T)/T}$ the upper-bound is $O(\sqrt{T\Upsilon_T \log T})$. Thus, SW-UCB matches the lower-bound of Theorem 3 up to a factor $\sqrt{\log T}$, slightly better than the D-UCB.

On the other hand, if the breakpoints have a positive density over time (say, if $\Upsilon_T \leq rT$ for a small positive constant r), then γ has to remain lower-bounded independently of T ; Theorem 1 gives a linear, non-trivial bound on the regret and allows to calibrate the discount factor γ as a function of the density of the breakpoint: with $\gamma = 1 - \sqrt{r}/(4B)$ we get an upper-bound with a dominant term in $-\sqrt{r} \log(r)O(T)$.

Concerning SW-UCB, τ has to remain lower-bounded independently of T . For instance, if $\Upsilon_T \leq rT$ for some (small) positive rate r , and for the choice $\tau = 2B\sqrt{-\log r/r}$, Theorem 2 gives $\mathbb{E}_\tau [\tilde{N}_T(i)] = O(T\sqrt{-r \log(r)})$. If the growth rate of Υ_T is known in advance, but not the horizon T , then we can use the “doubling trick” to set the value of γ and τ . Namely, for t and k such that $2^k \leq t < 2^{k+1}$, take $\gamma = 1 - (4B)^{-1}(2^k)^{(\beta-1)/2}$.

If there is no breakpoint ($\Upsilon_T = 0$), the best choice is obviously to make the window as large as possible, that is $\tau = T$. Then the procedure is exactly standard UCB. A slight modification of the preceeding proof for $\xi = \frac{1}{2} + \epsilon$ with arbitrary small ϵ yields $\mathbb{E}_{\text{UCB}} [\tilde{N}_T(i)] \leq \frac{2B^2}{(\Delta\mu(i))^2} \log(T)(1 + o(1))$. This result improves by a constant factor the bound given in Theorem 1 in [5]. In [13], another constant factor is gained by using a different proof.

4 Analysis of D-UCB

Because of space limitations, we present only the analysis of D-UCB, i.e. the proof of Theorem 1. The case of SW-UCB is similar, although slightly more simple because of the absence of bias at a large distance of the breakpoints.

Compared to the standard regret analysis of the stationary case (see e.g. [5]), there are two main differences. First, because the expected reward changes,

the discounted empirical mean $\bar{X}_t(\gamma, i)$ is now a *biased* estimator of the expected reward $\mu_t(i)$. The second difference stems from the deviation inequality itself: instead of using a Chernoff-Hoeffding bound, we use a novel tailored-made control on a self-normalized mean of the rewards with a random number of summands, which is stated in Section 7. The proof is in 5 steps:

Step 1. The number of times a suboptimal arm i is played is:

$$\tilde{N}_T(i) = 1 + \sum_{t=K+1}^T \mathbb{1}_{\{I_t=i \neq i_t^*, N_t(\gamma, i) < A(\gamma)\}} + \sum_{t=K+1}^T \mathbb{1}_{\{I_t=i \neq i_t^*, N_t(\gamma, i) \geq A(\gamma)\}},$$

where $A(\gamma) = 16B^2\xi \log n_T(\gamma) / (\Delta\mu_T(i))^2$. Using Lemma 1 (see Section 7), we may upper-bound the first sum in the RHS as $\sum_{t=K+1}^T \mathbb{1}_{\{I_t=i \neq i_t^*, N_t(\gamma, i) < A(\gamma)\}} \leq \lceil T(1-\gamma) \rceil A(\gamma) \gamma^{-\frac{1}{1-\gamma}}$. For a number of rounds (which depends on γ) following a breakpoint, the estimates of the expected rewards can be poor for $D(\gamma) = \log((1-\gamma)\xi \log n_K(\gamma)) / \log(\gamma)$ rounds. For any positive T , we denote by $\mathcal{T}(\gamma)$ the set of all indices $t \in \{K+1, \dots, T\}$ such that for all integers $s \in]t-D(\gamma), t]$, for all $j \in \{1, \dots, K\}$, $\mu_s(j) = \mu_t(j)$. In other words, t is in $\mathcal{T}(\gamma)$ if it does not follow too soon after a state transition. This leads to the following bound:

$$\sum_{t=K+1}^T \mathbb{1}_{\{I_t=i \neq i_t^*, N_t(\gamma, i) \geq A(\gamma)\}} \leq \Upsilon_T D(\gamma) + \sum_{t \in \mathcal{T}(\gamma)} \mathbb{1}_{\{I_t=i \neq i_t^*, N_t(\gamma, i) \geq A(\gamma)\}}.$$

Putting everything together, we obtain:

$$\tilde{N}_T(i) \leq 1 + \lceil T(1-\gamma) \rceil A(\gamma) \gamma^{-1/(1-\gamma)} + \Upsilon_T D(\gamma) + \sum_{t \in \mathcal{T}(\gamma)} \mathbb{1}_{\{I_t=i \neq i_t^*, N_t(\gamma, i) \geq A(\gamma)\}}. \quad (4)$$

Step 2. Let $t \in \mathcal{T}(\gamma)$. If the following three things were true:

$$\begin{cases} \bar{X}_t(\gamma, i) + c_t(\gamma, i) < \mu_t(i) + 2c_t(\gamma, i) \\ \mu_t(i) + 2c_t(\gamma, i) < \mu_t(i_t^*) \\ \mu_t(i_t^*) < \bar{X}_t(\gamma, i_t^*) + c_t(\gamma, i_t^*) \end{cases}$$

then $\bar{X}_t(\gamma, i) + c_t(\gamma, i) < \bar{X}_t(\gamma, i_t^*) + c_t(\gamma, i_t^*)$, and arm i^* would be chosen. Thus,

$$\{I_t = i \neq i_t^*, N_t(\gamma, i) \geq A(\gamma)\} \subseteq \begin{cases} \{\mu_t(i_t^*) - \mu_t(i) \leq 2c_t(\gamma, i), N_t(\gamma, i) \geq A(\gamma)\} \\ \cup \{\bar{X}_t(\gamma, i_t^*) \leq \mu_t(i_t^*) - c_t(\gamma, i_t^*)\} \\ \cup \{\bar{X}_t(\gamma, i) \geq \mu_t(i) + c_t(\gamma, i)\} \end{cases} \quad (5)$$

In words, playing the suboptimal arm i at time t may occur in three cases: if $\mu_t(i)$ is substantially over-estimated, if $\mu_t(i_t^*)$ is substantially under-estimated, or if $\mu_t(i)$ and $\mu_t(i_t^*)$ are close to each other. But for the choice of $A(\gamma)$ given above, we have $c_t(\gamma, i) \leq 2B\sqrt{(\xi \log n_t(\gamma)) / A(\gamma)} \leq \Delta\mu_T(i)/2$, and the event $\{\mu_t(i_t^*) - \mu_t(i) < 2c_t(\gamma, i), N_t(\gamma, i) \geq A(\gamma)\}$ never occurs.

In Steps 3 and 4 we upper-bound the probability of the first two events of the RHS of (5). We show that for $t \in \mathcal{T}(\gamma)$, that is at least $D(\gamma)$ rounds after a breakpoint, the expected rewards of all arms are well estimated with high probability. For all $j \in \{1, \dots, K\}$, consider the event $\mathcal{E}_t(\gamma, j) = \{\bar{X}_t(\gamma, i) \geq \mu_t(j) + c_t(\gamma, j)\}$. The idea is the following: we upper-bound the probability of $\mathcal{E}_t(\gamma, j)$ by separately considering the fluctuations of $\bar{X}_t(\gamma, j)$ around $M_t(\gamma, j)/N_t(\gamma, j)$, and the ‘bias’ $M_t(\gamma, j)/N_t(\gamma, j) - \mu_t(j)$, where $M_t(\gamma, j) = \sum_{s=1}^t \gamma^{t-s} \mathbb{1}_{\{I_s=j\}} \mu_s(j)$.

Step 3. Let us first consider the bias. First note that $M_t(\gamma, j)/N_t(\gamma, j)$, as a convex combination of elements $\mu_s(j) \in [0, B]$, belongs to interval $[0, B]$. Hence, $|M_t(\gamma, j)/N_t(\gamma, j) - \mu_t(j)| \leq B$. Second, for $t \in \mathcal{T}(\gamma)$,

$$\begin{aligned} |M_t(\gamma, j) - \mu_t(j)N_t(\gamma)| &= \left| \sum_{s=1}^{t-D(\gamma)} \gamma^{t-s} (\mu_s(j) - \mu_t(j)) \mathbb{1}_{\{I_s=j\}} \right| \\ &\leq \sum_{s=1}^{t-D(\gamma)} \gamma^{t-s} |\mu_s(j) - \mu_t(j)| \mathbb{1}_{\{I_s=j\}} \leq B \gamma^{D(\gamma)} N_{t-D(\gamma)}(\gamma, j). \end{aligned}$$

As obviously $N_{t-D(\gamma)}(\gamma, j) \leq (1-\gamma)^{-1}$, we get that $|M_t(\gamma, j)/N_t(\gamma, j) - \mu_t(j)| \leq B \gamma^{D(\gamma)} ((1-\gamma)N_t(\gamma))^{-1}$. Altogether,

$$\left| \frac{M_t(\gamma, j)}{N_t(\gamma, j)} - \mu_t(j) \right| \leq B \left(1 \wedge \frac{\gamma^{D(\gamma)}}{(1-\gamma)N_t(\gamma)} \right).$$

Hence, using the elementary inequality $1 \wedge x \leq \sqrt{x}$ and the definition of $D(\gamma)$, we obtain for $t \in \mathcal{T}(\gamma)$:

$$\left| \frac{M_t(\gamma, j)}{N_t(\gamma, j)} - \mu_t(j) \right| \leq B \sqrt{\frac{\gamma^{D(\gamma)}}{(1-\gamma)N_t(\gamma, i)}} \leq B \sqrt{\frac{\xi \log n_K(\gamma)}{N_t(\gamma, j)}} \leq \frac{1}{2} c_t(\gamma, j).$$

In words: $D(\gamma)$ rounds after a breakpoint, the ‘bias’ is smaller than the half of the exploration bonus. The other half of the exploration bonus is used to control the fluctuations. In fact, for $t \in \mathcal{T}(\gamma)$:

$$\begin{aligned} \mathbb{P}_\gamma(\mathcal{E}_t(\gamma, j)) &\leq \mathbb{P}_\gamma\left(\bar{X}_t(\gamma, j) > \mu_t(j) + B \sqrt{\frac{\xi \log n_t(\gamma)}{N_t(\gamma, j)}} + \left| \frac{M_t(\gamma, j)}{N_t(\gamma, j)} - \mu_t(j) \right|\right) \\ &\leq \mathbb{P}_\gamma\left(\bar{X}_t(\gamma, j) - \frac{M_t(\gamma, j)}{N_t(\gamma, j)} > B \sqrt{\frac{\xi \log n_t(\gamma)}{N_t(\gamma, j)}}\right). \end{aligned}$$

Step 4. Denote the discounted total reward obtained with arm j by

$$S_t(\gamma, j) = \sum_{s=1}^t \gamma^{t-s} \mathbb{1}_{\{I_s=j\}} X_s(j) = N_t(\gamma, j) \bar{X}_t(\gamma, j).$$

Using Theorem 4 and the fact that $N_t(\gamma, j) \geq N_t(\gamma^2, j)$, we get:

$$\begin{aligned} \mathbb{P}_\gamma(\mathcal{E}_t(\gamma, j)) &\leq \mathbb{P}_\gamma\left(\frac{S_t(\gamma, j) - M_t(\gamma, j)}{\sqrt{N_t(\gamma^2, j)}} > B\sqrt{\frac{\xi N_t(\gamma, j) \log n_t(\gamma)}{N_t(\gamma^2, j)}}\right) \\ &\leq \mathbb{P}_\gamma\left(\frac{S_t(\gamma, j) - M_t(\gamma, j)}{\sqrt{N_t(\gamma^2, j)}} > B\sqrt{\xi \log n_t(\gamma)}\right) \\ &\leq \left\lceil \frac{\log n_t(\gamma)}{\log(1 + \eta)} \right\rceil n_t(\gamma)^{-2\xi(1 - \frac{\eta^2}{16})}. \end{aligned}$$

Step 5. Hence, we finally obtain from Equation (4) that for all positive η :

$$\begin{aligned} \mathbb{E}_\gamma \left[\tilde{N}_T(i) \right] &\leq 1 + \lceil T(1 - \gamma) \rceil A(\gamma) \gamma^{-1/(1-\gamma)} + D(\gamma) \Upsilon_T \\ &\quad + 2 \sum_{t \in \mathcal{T}(\gamma)} \left\lceil \frac{\log n_t(\gamma)}{\log(1 + \eta)} \right\rceil n_t(\gamma)^{-2\xi(1 - \frac{\eta^2}{16})}. \end{aligned}$$

When $\Upsilon_T \neq 0$, γ is taken strictly smaller than 1. As $\xi > \frac{1}{2}$, we take $\eta = 4\sqrt{1 - 1/2\xi}$, so that $2\xi(1 - \eta^2/16) = 1$. For that choice, with $\tau = (1 - \gamma)^{-1}$,

$$\begin{aligned} \sum_{t \in \mathcal{T}(\gamma)} \left\lceil \frac{\log n_t(\gamma)}{\log(1 + \eta)} \right\rceil n_t(\gamma)^{-2\xi(1 - \frac{\eta^2}{16})} &\leq \tau - K + \sum_{t=\tau}^T \left\lceil \frac{\log n_\tau(\gamma)}{\log(1 + \eta)} \right\rceil n_\tau(\gamma)^{-1} \\ &\leq \tau - K + \left\lceil \frac{\log n_\tau(\gamma)}{\log(1 + \eta)} \right\rceil \frac{T}{n_\tau(\gamma)} \leq \tau - K + \left\lceil \frac{\log \frac{1}{1-\gamma}}{\log(1 + \eta)} \right\rceil \frac{T(1 - \gamma)}{1 - \gamma^{1/(1-\gamma)}} \end{aligned}$$

and we obtain the statement of the Theorem.

5 A Lower-Bound on the Regret in Abruptly Changing Environment

In this section, we consider a particular non-stationary bandit problem where the distributions of rewards on each arm are piecewise constant and have two breakpoints. Given any policy π , we derive a lower-bound on the number of times a sub-optimal arm is played (and thus, on the regret) in at least one such game. Quite intuitively, the less explorative a policy is, the longer it may keep a suboptimal policy after a breakpoint. Theorem 3 gives a precise content to this statement.

As in the previous section, K denotes the number of arms, and the rewards are assumed to be bounded in $[0, B]$. Consider any deterministic policy π of choosing the arms I_1, \dots, I_T played at each time depending to the past rewards $G_t \triangleq X_t(I_t)$, and recall that I_t is measurable with respect to the sigma-field $\sigma(G_1, \dots, G_t)$ of the past observed rewards. Denote by $N_{s:t}(i)$ the number of times arm i is played between times s and t $N_{s:t}(i) = \sum_{u=s}^t \mathbb{1}_{\{I_u=i\}}$, and

$N_T(i) = N_{1:T}(i)$. For $1 \leq i \leq K$, let P_i be the probability distribution of the outcomes of arm i , and let $\mu(i)$ denote its expectation. Assume that $\mu(1) > \mu(i)$ for all $2 \leq i \leq K$. Denote by \mathbb{P}_π the distribution of rewards under policy π , that is: $d\mathbb{P}_\pi(g_{1:T}|I_{1:T}) = \prod_{t=1}^T dP_{I_t}(g_t)$. For any random variable W measurable with respect to $\sigma(G_1, \dots, G_T)$, denote by $\mathbb{E}_\pi[W]$ its expectation under \mathbb{P}_π .

In the sequel, we divide the period $\{1, \dots, T\}$ into epochs of the same size $\tau \in \{1, \dots, T\}$, and we modify the distribution of the rewards so that on one of those periods, arm K becomes the one with highest expected reward. Specifically: let Q be a distribution of rewards with expectation $\nu > \mu(1)$, let $\delta = \nu - \mu(1)$ and let $\alpha = D(P_K; Q)$ be the Kullback-Leibler divergence between P_K and Q . For all $1 \leq j \leq M = \lfloor \frac{T}{\tau} \rfloor$, we consider the modification \mathbb{P}_π^j of \mathbb{P}_π such that on the j -th period of size τ , the distribution of rewards of the K -th arm is changed to ν . That is, for every sequence of rewards $g_{1:T}$,

$$\frac{d\mathbb{P}_\pi^j}{d\mathbb{P}_\pi}(g_{1:T}|I_{1:T}) = \prod_{t=1+(j-1)\tau, I_t=K}^{j\tau} \frac{dQ}{dP_K}(g_t).$$

Besides, let $N^j(i) = N_{1+(j-1)\tau:j\tau}(i)$ be the number of times arm i is played in the j -th period. For any random variable W in $\sigma(G_1, \dots, G_T)$, denote by $\mathbb{E}_\pi^j[W]$ its expectation under distribution \mathbb{P}_π^j . Now, denote by \mathbb{P}_π^* the distribution of rewards when j is chosen uniformly at random in the set $\{1, \dots, M\}$, i.e. \mathbb{P}_π^* is the (uniform) mixture of the $(\mathbb{P}_\pi^j)_{1 \leq j \leq M}$, and denote by $\mathbb{E}_\pi^*[\cdot]$ the expectation under \mathbb{P}_π^* : $\mathbb{E}_\pi^*[W] = M^{-1} \sum_{j=1}^M \mathbb{E}_\pi^j[W]$. In the following, we lower-bound the expected regret of any policy π under \mathbb{P}_π^* in terms of its regret under \mathbb{P}_π .

Theorem 3. *For any horizon T such that $64/(9\alpha) \leq \mathbb{E}_\pi[N_T(K)] \leq T/(4\alpha)$ and for any policy π ,*

$$\mathbb{E}_\pi^*[R_T] \geq C(\mu) \frac{T}{\mathbb{E}_\pi[R_T]},$$

where $C(\mu) = 2\delta(\mu(1) - \mu(K))/(3\alpha)$.

Proof. The main ingredients of this reasoning are inspired by the proof of Theorem 5.1 in [3]. First, note that the Kullback-Leibler divergence $D(\mathbb{P}_\pi; \mathbb{P}_\pi^j)$ is:

$$\begin{aligned} D(\mathbb{P}_\pi; \mathbb{P}_\pi^j) &= \sum_{t=1}^T D(\mathbb{P}_\pi(G_t|G_{1:t-1}); \mathbb{P}_\pi^j(G_t|G_{1:t-1})) \\ &= \sum_{t=1+(j-1)\tau}^{j\tau} \mathbb{P}_\pi(I_t=K) D(P_K; Q) = \alpha \mathbb{E}_\pi[N_{1+(j-1)\tau:j\tau}(K)]. \end{aligned}$$

But $\mathbb{E}_\pi^j[N^j(K)] - \mathbb{E}_\pi[N^j(K)] \leq \tau d_{TV}(\mathbb{P}_\pi^j, \mathbb{P}_\pi) \leq \tau \sqrt{D(\mathbb{P}_\pi; \mathbb{P}_\pi^j)/2}$ by Pinsker's inequality, and thus $\mathbb{E}_\pi^j[N^j(K)] \leq \mathbb{E}_\pi[N^j(K)] + \tau \sqrt{\alpha \mathbb{E}_\pi[N^j(K)]/2}$. Consequently, since $\sum_{j=1}^M N^j(K) \leq N_T(K)$,

$$\sum_{j=1}^M \mathbb{E}_\pi^j[N^j(K)] - \mathbb{E}[N_T(K)] \leq \tau \sum_{j=1}^M \sqrt{\frac{\alpha \mathbb{E}_\pi[N^j(K)]}{2}} \leq \tau \sqrt{\frac{\alpha M \mathbb{E}_\pi[N_T(K)]}{2}}.$$

Thus, there exists $1 \leq j \leq M$ such that

$$\begin{aligned}\mathbb{E}_\pi^*[N^j(K)] &\leq \frac{1}{M}\mathbb{E}_\pi[N_T(K)] + \frac{\tau}{M}\sqrt{\frac{\alpha}{2}M\mathbb{E}_\pi[N_T(K)]} \\ &\leq \frac{\tau}{T-\tau}\mathbb{E}_\pi[N_T(K)] + \sqrt{\frac{\alpha}{2}\frac{\tau^3}{T-\tau}\mathbb{E}_\pi[N_T(K)]}.\end{aligned}$$

Now, the expectation under \mathbb{P}_π^* of the regret R_T is lower-bounded as:

$$\frac{\mathbb{E}_\pi^*[R_T]}{\delta} \geq \tau - \mathbb{E}_\pi^*[N_T(K)] \geq \left(\tau - \frac{\tau}{T-\tau}\mathbb{E}_\pi[N_T(K)] - \sqrt{\frac{\alpha}{2}\frac{\tau^3}{T-\tau}\mathbb{E}_\pi[N_T(K)]} \right).$$

Maximizing the right hand side of the previous inequality by choosing $\tau = 8T/(9\alpha\mathbb{E}_\pi[N_T(K)])$ or equivalently $M = 9\alpha/(8\mathbb{E}_\pi[N_T(K)])$ leads to the lower-bound:

$$\mathbb{E}_\pi^*[R_T] \geq \frac{16\delta}{27\alpha} \left(1 - \frac{\alpha\mathbb{E}_\pi[N_T(K)]}{T} \right)^2 \left(1 - \frac{8}{9\alpha\mathbb{E}_\pi[N_T(K)]} \right) \frac{T}{\mathbb{E}_\pi[N_T(K)]}.$$

To conclude, simply note that $\mathbb{E}_\pi[N_T(K)] \leq \mathbb{E}_\pi[R_T]/(\mu(1) - \mu(K))$. We obtain that $\mathbb{E}_\pi^*[R_T]$ is lower-bounded by

$$\frac{16\delta(\mu(1) - \mu(K))}{27\alpha} \left(1 - \frac{\alpha\mathbb{E}_\pi[N_T(K)]}{T} \right)^2 \left(1 - \frac{8}{9\alpha\mathbb{E}_\pi[N_T(K)]} \right) \frac{T}{\mathbb{E}_\pi[R_T]},$$

which directly leads to the statement of the Theorem.

The following corollary states that no policy can have a non-stationary regret of order smaller than \sqrt{T} . It appears here as a consequence of Theorem 3, although it can also be proved directly.

Corollary 1. *For any policy π and any positive horizon T ,*

$$\max\{\mathbb{E}_\pi(R_T), \mathbb{E}_\pi^*(R_T)\} \geq \sqrt{C(\mu)T}.$$

Proof. If $\mathbb{E}_\pi[N_T(K)] \leq 16/(9\alpha)$, or if $\mathbb{E}_\pi[N_T(K)] \geq T/\alpha$, the result is obvious. Otherwise, Theorem 3 implies that:

$$\max\{\mathbb{E}_\pi(R_T), \mathbb{E}_\pi^*(R_T)\} \geq \max\{\mathbb{E}_\pi(R_T), C(\mu)\frac{T}{\mathbb{E}_\pi(R_T)}\} \geq \sqrt{C(\mu)T}.$$

In words, Theorem 3 states that for any policy not playing each arm often enough, there is necessarily a time where a breakpoint is not seen after a long period. For instance, as standard UCB satisfies $\mathbb{E}_\pi[N(K)] = \Theta(\log T)$, then $\mathbb{E}_\pi^*[R_T] \geq cT/\log(T)$ for some positive c depending on the reward distribution. To keep notation simple, Theorem 3 is stated and proved here for deterministic policy. It is easily verified that the same results also holds for randomized strategies (such as EXP3-P, see [3]).

This result is to be compared with standard minimax lower-bounds on the regret. On one hand, a *fixed-game lower-bound* in $O(\log T)$ was proved in [20] for the stationary case, when the distributions of rewards are fixed and T is allowed to go to infinity. On the other hand, a finite-time *minimax lower-bound* for individual sequences in $O(\sqrt{T})$ is proved in [3]. In this bound, for each horizon T the worst case among all possible reward distributions is considered, which explains the discrepancy. This result is obtained by letting the distance between distributions of rewards tend to 0 (typically, as $1/\sqrt{T}$). In Theorem 3, no assumption is made on the distributions of rewards P_i and Q , their distance actually remains lower-bounded independently of T . In fact, in the case considered here minimax regret and fixed-game minimal regret appear to have the same order of magnitude.

6 Simulations

The scope of this section is to present two simple, archetypal settings that show the interest of UCB methods in non-stationary stochastic environments. In the first example, there are $K = 3$ arms and the time horizon is set to $T = 10^4$. The rewards of arm $i \in \{1, \dots, K\}$ at time t are independent Bernoulli random variables with success probability $p_t(i)$, with $p_t(1) = 0.5$, $p_t(2) = 0.3$ and for $t \in \{1, \dots, T\}$, $p_t(3) = 0.4$ for $t < 3000$ or $t \geq 5000$, and $p_t(3) = 0.9$ for $3000 \leq t < 5000$. The optimal policy for this bandit task is to select arm 1 before the first breakpoint ($t = 3000$) and after the second breakpoint ($t = 5000$). In Figure 1, we represent the evolution of the cumulated regret. We compare the UCB-1 algorithm with $\xi = \frac{1}{2}$, the EXP3.S algorithm described in [3] with the tuned parameters given in Corollary 8.3 (with the notations of this paper $\alpha = T^{-1}$ and $\gamma = \sqrt{K(\Upsilon_T \log(KT) + e)}/[(e-1)T]$ with $\Upsilon_T = 2$), the D-UCB algorithm with $\xi = 0.6$ and $\gamma = 1 - 1/4\sqrt{T}$ and the SW-UCB with $\xi = 0.6$ and $\tau = 4\sqrt{T \log T}$ (chosen according to Section 3).

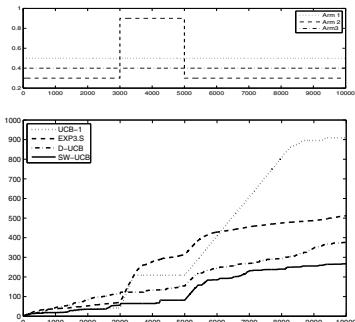


Fig. 1. Bernoulli MAB problem with two swaps. Above: evolution of the reward distributions. Below: cumulative regret of each policy.

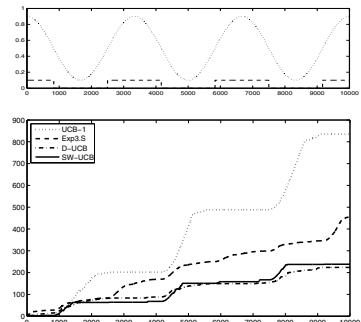


Fig. 2. Bernoulli MAB problem with periodic rewards. Above: evolution of reward distribution of arm 1. Below: cumulative regret of each policy.

As can be seen in Figure 1 (and as can be consistently observed over all simulations), D-UCB performs almost as well as SW-UCB. Both of them waste significantly less time than EXP3.S and UCB-1 to detect the breakpoints, and quickly concentrate their pulls on the optimal arm. Observe that policy UCB-1, initially the best, reacts very fast to the first breakpoint ($t = 3000$), as the confidence interval for arm 3 at this step is very loose. On the contrary, it takes a very long time after the second breakpoint ($t = 5000$) for UCB-1 to play arm 1 again.

In the second example, we test the behaviour of D-UCB and SW-UCB by investigating their performance in a slowly-varying environment. This environment is made of $K = 2$ arms, the rewards are still Bernoulli random variables with parameters $p_t(i)$ but they are in persistent, continuous evolution. Arm 2 is taken as a reference ($p_t(2) = 1/2$ for all t), and the parameter of arm 1 evolves periodically as: $p_t(1) = 0.5 + 0.4 \cos(6\pi R t / T)$. Hence, the best arm to pull changes cyclically and the transitions are smooth (regularly, the two arms are statistically indistinguishable). In Figure 2, the evolutions of the cumulative regrets under the four policies are shown: in this continuously evolving environment, the performance of D-UCB and SW-UCB are almost equivalent while UCB-1 and the Exp3.S algorithms accumulate larger regrets. Continuing the experiment or multiplying the changes only confirms this conclusion.

These modest and yet representative examples suggest that, despite the fact that similar regret bounds are proved for D-UCB, SW-UCB and EXP3.S, the two former methods are significantly more reactive to changes in practice and have a better performance, whether the environment is slowly or abruptly changing. EXP3.S, on the other hand, is expected to be more robust and more adapted to non stochastic (and non-oblivious) environments.

7 Technical Results

We first state a deviation bound for self-normalized discounted average, of independent interest, that proves to be a key ingredient in the analysis of D-UCB. Let $(X_t)_{t \geq 1}$ be a sequence of non-negative independent random variables bounded by B defined on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$, and we denote $\mu_t = \mathbb{E}[X_t]$. Let \mathcal{F}_t be an increasing sequence of σ -fields of \mathcal{A} such that for each t , $\sigma(X_1, \dots, X_t) \subset \mathcal{F}_t$ and for $s > t$, X_s is independent from \mathcal{F}_t . Consider a previsible sequence $(\epsilon_t)_{t \geq 1}$ of Bernoulli variables (for all $t > 0$, ϵ_t is \mathcal{F}_{t-1} -measurable). For $\gamma \in [0, 1)$, let $S_t(\gamma) = \sum_{s=1}^t \gamma^{t-s} X_s \epsilon_s$, $M_t(\gamma) = \sum_{s=1}^t \gamma^{t-s} \mu_s \epsilon_s N_t(\gamma) = \sum_{s=1}^t \gamma^{t-s} \epsilon_s$, and $n_t(\gamma) = \sum_{s=1}^t \gamma^{t-s}$.

Theorem 4. *For all integers t and all $\delta, \eta > 0$,*

$$\mathbb{P} \left(\frac{S_t(\gamma) - M_t(\gamma)}{\sqrt{N_t(\gamma^2)}} > \delta \right) \leq \left\lceil \frac{\log n_t(\gamma)}{\log(1 + \eta)} \right\rceil \exp \left(-\frac{2\delta^2}{B^2} \left(1 - \frac{\eta^2}{16} \right) \right).$$

The following lemma is required in the analysis of SW-UCB and D-UCB:

Lemma 1. *Let $i \in \{1, \dots, K\}$; for any positive integer τ , let $N_{t-\tau:t}(1, i) = \sum_{s=t-\tau+1}^t \mathbb{1}_{\{I_s=i\}}$. Then for any positive m ,*

$$\sum_{t=K+1}^T \mathbb{1}_{\{I_t=i, N_{t-\tau:t}(1,i) < m\}} \leq \lceil T/\tau \rceil m.$$

Thus, for any $\tau \geq 1$ and $A > 0$, $\sum_{t=K+1}^T \mathbb{1}_{\{I_t=i, N_t(\gamma, i) < A\}} \leq \lceil T/\tau \rceil A \gamma^{-\tau}$.

The proof of these results is omitted due to space limitations.

8 Conclusion and Perspectives

This paper theoretically establishes that the UCB policies can be successfully adapted to cope with non-stationary environments. It is shown introducing two breakpoints is enough to move from the $\log(T)$ performance of stationary bandits to the $\sqrt{T \log(T)}$ performance of adversarial bandits. The upper bound of the D-UCB and SW-UCB in an abruptly changing environment are similar to the upper bounds of the EXP3.S algorithm, and numerical experiments show that UCB policies outperform the softmax methods in stochastic environments. The focus of this paper is on an abruptly changing environment, but it is believed that the theoretical tools developed to handle the non-stationarity can be applied in different contexts. In particular, using a similar bias-variance decomposition of the discounted or windowed-rewards, continuously evolving reward distributions can be analysed. Furthermore, the deviation inequality for discounted martingale transforms given in Section 7 is a powerful tool of independent interest.

As the previously reported Exp3.S algorithm, the performance of the proposed policy depends on tuning parameters (discount factor or window). Designing a fully adaptive algorithm, able to actually detect the changes as they occur with no prior knowledge of a typical inter-arrival time, is not an easy task and remains the subject of on-going research. A possibility may be to tune adaptively the parameters by using data-driven approaches, as in [24]. Another possibility is to use margin assumptions on the gap between the distributions before and after the changes, as in [21]: at the price of this extra assumption, one obtains improved bounds without the need for the knowledge of the number of changes.

References

- [1] Agrawal, R.: Sample mean based index policies with $O(\log n)$ regret for the multi-armed bandit problem. *Adv. in Appl. Probab.* 27(4), 1054–1078 (1995)
- [2] Audibert, J.Y., Munos, R., Szepesvari, A.: Tuning bandit algorithms in stochastic environments. In: Hutter, M., Servedio, R.A., Takimoto, E. (eds.) ALT 2007. LNCS (LNAI), vol. 4754, pp. 150–165. Springer, Heidelberg (2007)
- [3] Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multi-armed bandit problem. *SIAM J. Comput.* 32(1), 48–77 (2002)

- [4] Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.* 3(Spec. Issue Comput. Learn. Theory), 397–422 (2002)
- [5] Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47(2/3), 235–256 (2002)
- [6] Cesa-Bianchi, N., Lugosi, G.: *Prediction, Learning, and Games*. Cambridge University Press, New York (2006)
- [7] Cesa-Bianchi, N., Lugosi, G.: On prediction of individual sequences. *Ann. Statist.* 27(6), 1865–1895 (1999)
- [8] Cesa-Bianchi, N., Lugosi, G., Stoltz, G.: Regret minimization under partial monitoring. *Math. Oper. Res.* 31(3), 562–580 (2006)
- [9] Cesa-Bianchi, N., Lugosi, G., Stoltz, G.: Competing with typical compound actions (2008)
- [10] Devroye, L., Györfi, L., Lugosi, G.: *A probabilistic theory of pattern recognition*. Applications of Mathematics, vol. 31. Springer, New York (1996)
- [11] Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. System Sci.* 55(1, part 2), 119–139 (1997); In: Vitányi, P.M.B. (ed.) EuroCOLT 1995. LNCS, vol. 904. Springer, Heidelberg (1995)
- [12] Fuh, C.D.: Asymptotic operating characteristics of an optimal change point detection in hidden Markov models. *Ann. Statist.* 32(5), 2305–2339 (2004)
- [13] Garivier, A., Cappé, O.: The kl-ucb algorithm for bounded stochastic bandits and beyond. In: Proceedings of the 24rd Annual International Conference on Learning Theory (2011)
- [14] Hartland, C., Gelly, S., Baskiotis, N., Teytaud, O., Sebag, M.: Multi-armed bandit, dynamic environments and meta-bandits. In: nIPS-2006 Workshop, Online Trading Between Exploration and Exploitation, Whistler, Canada (2006)
- [15] Herbster, M., Warmuth, M.: Tracking the best expert. *Machine Learning* 32(2), 151–178 (1998)
- [16] Honda, J., Takemura, A.: An asymptotically optimal bandit algorithm for bounded support models. In: Proceedings of the 23rd Annual International Conference on Learning Theory (2010)
- [17] Kocsis, L., Szepesvári, C.: Discounted UCB. In: 2nd PASCAL Challenges Workshop, Venice, Italy (April 2006)
- [18] Koulouriotis, D.E., Xanthopoulos, A.: Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems. *Applied Mathematics and Computation* 196(2), 913–922 (2008)
- [19] Lai, L., El Gamal, H., Jiang, H., Poor, H.V.: Cognitive medium access: Exploration, exploitation and competition (2007)
- [20] Lai, T.L., Robbins, H.: Asymptotically efficient adaptive allocation rules. *Adv. in Appl. Math.* 6(1), 4–22 (1985)
- [21] Mei, Y.: Sequential change-point detection when unknown parameters are present in the pre-change distribution. *Ann. Statist.* 34(1), 92–122 (2006)
- [22] Slivkins, A., Upfal, E.: Adapting to a changing environment: the brownian restless bandits. In: Proceedings of the Conference on 21st Conference on Learning Theory, pp. 343–354 (2008)
- [23] Whittle, P.: Restless bandits: activity allocation in a changing world. *J. Appl. Probab. Special* 25A, 287–298 (1988) a celebration of applied probability
- [24] Yu, J.Y., Mannor, S.: Piecewise-stationary bandit problems with side observations. In: ICML 2009: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 1177–1184. ACM, New York (2009)

Upper-Confidence-Bound Algorithms for Active Learning in Multi-armed Bandits

Alexandra Carpentier¹, Alessandro Lazaric¹, Mohammad Ghavamzadeh¹, Rémi Munos¹, and Peter Auer²

¹ INRIA Lille - Nord Europe, Team SequeL, France

² University of Leoben, Franz-Josef-Strasse 18, 8700 Leoben, Austria

Abstract. In this paper, we study the problem of estimating the mean values of all the arms uniformly well in the multi-armed bandit setting. If the variances of the arms were known, one could design an optimal sampling strategy by pulling the arms proportionally to their variances. However, since the distributions are not known in advance, we need to design adaptive sampling strategies to select an arm at each round based on the previous observed samples. We describe two strategies based on pulling the arms proportionally to an upper-bound on their variance and derive regret bounds for these strategies. We show that the performance of these allocation strategies depends not only on the variances of the arms but also on the full shape of their distribution.

1 Introduction

Consider a marketing problem where the objective is to estimate the potential impact of several new products or services. A common approach to this problem is to design active online polling systems, where at each time step a product is presented (e.g., via a web banner on Internet) to random customers from a population of interest, and feedbacks are collected (e.g., whether the customer clicks on the advertisement or not) and used to estimate the average preference of all the products. It is often the case that some products have a general consensus of opinion (low variance) while others have a large variability (high variance). While in the former case very few votes would be enough to have an accurate estimate of the value of the product, in the latter the system should present the product to more customers in order to achieve the same level of accuracy. Since the variability of the opinions for different products is not known in advance, the objective is to design an active strategy that selects which product to display at each time step in order to estimate the values of all the products uniformly well.

The problem of online polling can be seen as an online allocation problem with several options, where the accuracy of the estimation of the quality of each option depends on the quantity of resources allocated to it and also on some (initially unknown) intrinsic variability of the option. This general problem is closely related to the problems of active learning (Cohn et al., 1996, Castro et al., 2005), sampling and Monte-Carlo methods (Étotoré and Jourdain, 2010), and optimal experimental design (Fedorov, 1972, Chaudhuri and Mykland, 1995). A particular

instance of this problem is introduced in Antos et al. (2010) as an active learning problem in the framework of stochastic multi-armed bandits. More precisely, the problem is modeled as a repeated game between a learner and a stochastic environment, defined by a set of K unknown distributions $\{\nu_k\}_{k=1}^K$, where at each round t , the learner selects an option (or arm) k_t and as a consequence receives a random sample from ν_{k_t} (independent of the past samples). Given a budget of n samples, the goal is to define an allocation strategy over arms so as to estimate their expected values uniformly well (using a squared loss to evaluate the accuracy). Note that if the variances $\{\sigma_k^2\}_{k=1}^K$ of the arms were initially known, the optimal allocation strategy would be to sample the arms proportionally to their variances, or more precisely, proportionally to $\lambda_k = \sigma_k^2 / \sum_j \sigma_j^2$. However, since the distributions are initially unknown, the learner should implement an active allocation strategy which adapts its behavior as samples are collected. The performance of this strategy is measured by its regret (Eq. 4), defined as the difference between the expected quadratic estimation error of the algorithm and the error of the optimal allocation.

Antos et al. (2010) presented an algorithm, called GAFS-MAX, that allocates samples proportionally to the empirical variances of the arms, while imposing that each arm should be pulled at least \sqrt{n} times (to guarantee good estimation of the true variances). They proved that for large enough n , the regret of their algorithm scales with $\tilde{O}(n^{-3/2})$ and conjectured that this rate is optimal.¹ However, the performance displays both an implicit (in the condition for large enough n) and explicit (in the regret bound) dependency on the inverse of the smallest optimal allocation proportion, i.e., $\lambda_{\min} = \min_k \lambda_k$. This suggests that the algorithm may have a poor performance whenever an arm has a very small variance compared to the others (e.g., when users involved in the poll have very similar opinions about some products and very different on some others). Whether this dependency is due to the analysis of GAFS-MAX, to the specific class of algorithms, or to an intrinsic characteristic of the problem is an interesting open question.

In this paper, in order to further investigate this issue, we introduce two novel algorithms based on upper-confidence-bounds (UCB) on the variance. The algorithms sample the arms proportionally to an upper-bound on their variance computed from the empirical variances and a confidence interval derived from Chernoff-Hoeffding's (first algorithm) and Bernstein's (second algorithm) inequalities. The main advantage of this class of algorithms is that the possibility to use standard tools and arguments for UCB-like algorithms makes their analysis simple, thus making the study of the dependency on λ_{\min} easier. The main contributions and findings of this paper are as follows:

- The first algorithm, called CH-AS, is based on Chernoff-Hoeffding's bound and its regret is $\tilde{O}(n^{-3/2})$ with an inverse dependency on λ_{\min} , similar to GAFS-MAX. The main differences are: the bound for CH-AS holds for any n

¹ The notation $u_n = \tilde{O}(v_n)$ means that there exist $C > 0$ and $\alpha > 0$ such that $u_n \leq C(\log n)^\alpha v_n$ for sufficiently large n .

(and not only for large enough n), multiplicative constants are made explicit, and finally, the proof is simpler and relies on very simple tools.

- The second algorithm, called B-AS, uses an empirical Bernstein’s inequality, and it has a better performance (in terms of the number of pulls) in targeting the optimal allocation strategy without any dependency on λ_{\min} . However, moving from the number of pulls to the regret causes the inverse dependency on λ_{\min} to appear again in the bound. We show that this might be due to the specific shape of the distributions $\{\nu_k\}_{k=1}^K$ and derive a regret bound independent from λ_{\min} for the case of Gaussian arms.
- We show empirically that while the performance of CH-AS depends on λ_{\min} in the case of Gaussian arms, this dependence does not exist for B-AS and GAFS-MAX, as they perform well in this case. This suggests that **1)** it is not possible to remove λ_{\min} from the regret bound of CH-AS, independent of the arms’ distributions, and **2)** GAFS-MAX’s analysis could be improved along the same line as the proof of B-AS for the Gaussian arms. Furthermore, we further investigate the impact of the distribution on the regret by reporting numerical results in case of Rademacher distributions showing that B-AS performance worsens with λ_{\min}^{-1} . This leads to the conjecture that the full shape of the distributions, and not only their variance, impacts the regret of these algorithms.

2 Preliminaries

The allocation problem studied in this paper is formalized in the standard K -armed stochastic bandit setting, where each arm $k = 1, \dots, K$ is characterized by a distribution² ν_k with mean μ_k and variance σ_k^2 . At each round $t \geq 1$, the learner (algorithm \mathcal{A}) selects an arm k_t and receives a sample drawn from ν_{k_t} independently of the past. The objective is to estimate the mean values of all the arms uniformly well given a total budget of n pulls. An adaptive algorithm defines its allocation strategy as a function of the samples observed in the past (i.e., at time t , the selected arm k_t is a function of all the observations up to time $t - 1$). After n rounds and observing $T_{k,n} = \sum_{t=1}^n \mathbb{I}\{k = k_t\}$ samples from each

arm k , the algorithm \mathcal{A} returns the empirical estimates $\hat{\mu}_{k,n} = \frac{1}{T_{k,n}} \sum_{t=1}^{T_{k,n}} X_{k,t}$, where $X_{k,t}$ denotes the sample received when pulling arm k for the t -th time. The accuracy of the estimation at each arm k is measured according to its expected squared estimation error, or loss

$$L_{k,n} = \mathbb{E}_{\nu_k} \left[(\mu_k - \hat{\mu}_{k,n})^2 \right]. \quad (1)$$

The global performance, or loss, of \mathcal{A} is defined as the worst loss of the arms

$$L_n(\mathcal{A}) = \max_{1 \leq k \leq K} L_{k,n}. \quad (2)$$

² Although the formulation of the problem holds for any distribution, in the following we will consider the case of bounded and sub-Gaussian distributions in order to derive meaningful bounds.

If the variance of the arms were known in advance, one could design an optimal static allocation (i.e., independent from the observed samples) by pulling the arms proportionally to their variances. If an arm k is pulled a fixed number of times $T_{k,n}^*$, its loss is³

$$L_{k,n}(\mathcal{A}^*) = \frac{\sigma_k^2}{T_{k,n}^*}. \quad (3)$$

By choosing $T_{k,n}^*$ so as to minimize L_n under the constraint that $\sum_{k=1}^K T_{k,n}^* = n$, the optimal static allocation strategy \mathcal{A}^* pulls each arm k $T_{k,n}^* = \frac{\sigma_k^2 n}{\sum_{i=1}^K \sigma_i^2}$ times (up to rounding effects), and achieves a global performance $L_n(\mathcal{A}^*) = \Sigma/n$, where $\Sigma = \sum_{i=1}^K \sigma_i^2$. We denote by $\lambda_k = \frac{T_{k,n}^*}{n} = \frac{\sigma_k^2}{\Sigma}$, the optimal allocation proportion for arm k , and by $\lambda_{\min} = \min_{1 \leq k \leq K} \lambda_k$, the smallest such proportion.

In our setting, where the variances of the arms are not known in advance, the exploration-exploitation trade-off is inevitable: an adaptive algorithm \mathcal{A} should estimate the variances of the arms (*exploration*) at the same time as it tries to sample the arms proportionally to these estimates (*exploitation*). In order to measure how well the adaptive algorithm \mathcal{A} performs, we compare its performance to that of the optimal allocation algorithm \mathcal{A}^* , which requires the knowledge of the variances of the arms. For this purpose we define the notion of *regret* of an adaptive algorithm \mathcal{A} as the difference between the loss incurred by the learner and the optimal loss $L_n(\mathcal{A}^*)$:

$$R_n(\mathcal{A}) = L_n(\mathcal{A}) - L_n(\mathcal{A}^*). \quad (4)$$

It is important to note that unlike the standard multi-armed bandit problems, we do not consider the notion of cumulative regret, and instead, use the excess-loss suffered by the algorithm at the end of the n rounds. This notion of regret is closely related to the *pure exploration* setting (e.g., Audibert et al. 2010, Bubeck et al. 2011). In fact, in both settings good strategies should play each arm a linear function of n , in contrast with the standard stochastic bandit setting, where the sub-optimal arms should be played logarithmically in n .

3 Allocation Strategy Based on Chernoff-Hoeffding UCB

The first algorithm, called *Chernoff-Hoeffding Allocation Strategy* (CH-AS), is based on a Chernoff-Hoeffding high-probability bound on the difference between the estimated and true variances of the arms. Each arm is simply pulled proportionally to an upper-confidence-bound (UCB) on its variance. This algorithm deals with the exploration-exploitation trade-off by pulling more the arms with higher estimated variances or higher uncertainty in these estimates.

³ This equality does not hold when the number of pulls is random, e.g., in adaptive algorithms, where the strategy depends on the random observed samples.

```

Input: parameter  $\delta$ 
Initialize: Pull each arm twice
for  $t = 2K + 1, \dots, n$  do
    Compute  $B_{q,t} = \frac{1}{T_{q,t-1}} \left( \hat{\sigma}_{q,t-1}^2 + 5\sqrt{\frac{\log(1/\delta)}{2T_{q,t-1}}} \right)$  for each arm  $1 \leq q \leq K$ 
    Pull an arm  $k_t \in \arg \max_{1 \leq q \leq K} B_{q,t}$ 
end for
Output:  $\hat{\mu}_{q,n}$  for all arms  $1 \leq q \leq K$ 

```

Fig. 1. The pseudo-code of the CH-AS algorithm, with $\hat{\sigma}_{q,t}^2$ computed as in Eq. 5

3.1 The CH-AS Algorithm

The CH-AS algorithm \mathcal{A}_{CH} in Fig. 1 takes a confidence parameter δ as input and after n pulls returns an empirical mean $\hat{\mu}_{q,n}$ for each arm q . At each time step t , i.e., after having pulled arm k_t , the algorithm computes the empirical mean $\hat{\mu}_{q,t}$ and variance $\hat{\sigma}_{q,t}^2$ of each arm q as⁴

$$\hat{\mu}_{q,t} = \frac{1}{T_{q,t}} \sum_{i=1}^{T_{q,t}} X_{q,i} \quad \text{and} \quad \hat{\sigma}_{q,t}^2 = \frac{1}{T_{q,t}} \sum_{i=1}^{T_{q,t}} X_{q,i}^2 - \hat{\mu}_{q,t}^2, \quad (5)$$

where $X_{q,i}$ is the i -th sample of ν_q and $T_{q,t}$ is the number of pulls allocated to arm q up to time t . After pulling each arm twice (rounds $t = 1$ to $2K$), from round $t = 2K + 1$ on, the algorithm computes the $B_{q,t}$ values based on a Chernoff-Hoeffding's bound on the variances of the arms:

$$B_{q,t} = \frac{1}{T_{q,t-1}} \left(\hat{\sigma}_{q,t-1}^2 + 5\sqrt{\frac{\log(1/\delta)}{2T_{q,t-1}}} \right),$$

and then pulls the arm k_t with the largest $B_{q,t}$.

3.2 Regret Bound and Discussion

Before reporting a regret bound for CH-AS, we first analyze its performance in targeting the optimal allocation strategy in terms of the number of pulls. As it will be discussed later, the distinction between the performance in terms of the number of pulls and the regret will allow us to stress the potential dependency of the regret on the distribution of the arms (see Section 4.3).

Lemma 1. *Assume that the supports of the distributions $\{\nu_k\}_{k=1}^K$ are in $[0, 1]$ and that $n \geq 4K$. For any $\delta > 0$, for any arm $1 \leq k \leq K$, the number of pulls $T_{k,n}$ played by the CH-AS algorithm satisfies with probability at least $1 - 4nK\delta$,*

$$-\frac{5}{\Sigma^2 \lambda_{\min}^{3/2}} \sqrt{n \log(1/\delta)} - \frac{K}{\Sigma} \leq T_{k,n} - T_{k,n}^* \leq \frac{5(K-1)}{\Sigma^2 \lambda_{\min}^{3/2}} \sqrt{n \log(1/\delta)} + \frac{K^2}{\Sigma}. \quad (6)$$

⁴ Notice that this is a biased estimator of the variance even if the $T_{q,t}$ were not random.

Proof. Let $\xi_{K,n}(\delta)$ be the event

$$\xi_{K,n}(\delta) = \bigcap_{1 \leq k \leq K, 1 \leq t \leq n} \left\{ \left| \left(\frac{1}{t} \sum_{i=1}^t X_{k,i}^2 - \left(\frac{1}{t} \sum_{i=1}^t X_{k,i} \right)^2 \right) - \sigma_k^2 \right| \leq 5 \sqrt{\frac{\log(1/\delta)}{2t}} \right\}. \quad (7)$$

From Hoeffding's inequality it follows that $\Pr(\xi_{K,n}(\delta)) \geq 1 - 4nK\delta$. We divide the proof of this lemma into the following three steps.

Step 1. Mechanism of the algorithm. On event $\xi_{K,n}(\delta)$, for all $t \leq n$ and q

$$|\hat{\sigma}_{q,t}^2 - \sigma_q^2| \leq 5 \sqrt{\frac{\log(1/\delta)}{2T_{q,t}}},$$

and the following upper and lower bounds for $B_{q,t+1}$ hold

$$\frac{\sigma_q^2}{T_{q,t}} \leq B_{q,t+1} \leq \frac{1}{T_{q,t}} \left(\sigma_q^2 + 10 \sqrt{\frac{\log(1/\delta)}{2T_{q,t}}} \right). \quad (8)$$

Let $t+1 > 2K$ be the time at which a given arm k is pulled for the last time, i.e., $T_{k,t} = T_{k,n} - 1$ and $T_{k,(t+1)} = T_{k,n}$. Note that as $n \geq 4K$, there is at least one arm k that is pulled after the initialization phase. Since \mathcal{A}_{CH} chooses to pull arm k at time $t+1$, for any arm p , we have $B_{p,t+1} \leq B_{k,t+1}$. From Eq. 8 and the fact that $T_{k,t} = T_{k,n} - 1$, we obtain

$$B_{k,t+1} \leq \frac{1}{T_{k,t}} \left(\sigma_k^2 + 10 \sqrt{\frac{\log(1/\delta)}{2T_{k,t}}} \right) = \frac{1}{T_{k,n} - 1} \left(\sigma_k^2 + 10 \sqrt{\frac{\log(1/\delta)}{2(T_{k,n} - 1)}} \right). \quad (9)$$

Using Eq. 8 and the fact that $T_{p,t} \leq T_{p,n}$, we derive a lower-bound for $B_{p,t+1}$ as

$$B_{p,t+1} \geq \frac{\sigma_p^2}{T_{p,t}} \geq \frac{\sigma_p^2}{T_{p,n}}. \quad (10)$$

Combining the condition $B_{p,t+1} \leq B_{k,t+1}$ with Eqs. 9 and 10, we obtain

$$\frac{\sigma_p^2}{T_{p,n}} \leq \frac{1}{T_{k,n} - 1} \left(\sigma_k^2 + 10 \sqrt{\frac{\log(1/\delta)}{2(T_{k,n} - 1)}} \right). \quad (11)$$

Note that at this point there is no dependency on t , and thus, the probability that Eq. 11 holds for any p and for any k such that $T_{k,n} > 2$ (i.e. arm k is pulled at least once after the initialization phase), is at least $1 - 4nK\delta$ (probability of the event $\xi_{K,n}(\delta)$).

Step 2. Lower bound on $T_{p,n}$. If an arm p is under-pulled *without taking into account the initialization phase*, i.e., $T_{p,n} - 2 < \lambda_p(n - 2K)$, then from the constraint $\sum_k (T_{k,n} - 2) = n - 2K$, we deduce that there must be at least one arm k that is over-pulled, i.e., $T_{k,n} - 2 > \lambda_k(n - 2K)$. Note that for this arm, $T_{k,n} - 2 > \lambda_k(n - 2K) \geq 0$, so we know that this specific arm is pulled at least once *after* the initialization phase and that it satisfies Eq. 11. Using the definition of the optimal allocation $T_{k,n}^* = n\lambda_k = n\sigma_k^2/\Sigma$ and the fact that $T_{k,n} \geq \lambda_k(n - 2K) + 2$, Eq. 11 may be written as

$$\frac{\sigma_p^2}{T_{p,n}} \leq \frac{1}{T_{k,n}^*} \frac{n}{n-2K} \left(\sigma_k^2 + \sqrt{\frac{100 \log(1/\delta)}{2(\lambda_k(n-2K)+2-1)}} \right) \leq \frac{\Sigma}{n} + \frac{20\sqrt{\log(1/\delta)}}{(\lambda_{\min} n)^{3/2}} + \frac{4K\Sigma}{n^2},$$

since $\lambda_k(n-2K)+1 \geq \lambda_k(n/2-2K+2K)+1 \geq \frac{n\lambda_k}{2}$, as $n \geq 4K$ (thus also $\frac{2K\Sigma}{n(n-2K)} \leq \frac{4K\Sigma}{n^2}$). By reordering the terms in the previous equation, we obtain the lower bound

$$T_{p,n} \geq \frac{\sigma_p^2}{\frac{\Sigma}{n} + \frac{20\sqrt{\log(1/\delta)}}{(n\lambda_{\min})^{3/2}} + \frac{4K\Sigma}{n^2}} \geq T_{p,n}^* - \frac{5\sqrt{n \log(1/\delta)}}{\Sigma^2 \lambda_{\min}^{3/2}} - \frac{K}{\Sigma}, \quad (12)$$

where in the second inequality we used $1/(1+x) \geq 1-x$ (for $x > -1$) and $\sigma_p^2 \leq 1/4$. Note that the lower bound holds w.h.p. for any arm p .

Step 3. Upper bound on $T_{p,n}$. Using Eq. 12 and the fact that $\sum_k T_{k,n} = n$, we obtain the upper bound

$$T_{p,n} = n - \sum_{k \neq p} T_{k,n} \leq T_{p,n}^* + \frac{5(K-1)}{\Sigma^2 \lambda_{\min}^{3/2}} \sqrt{n \log(1/\delta)} + \frac{K^2}{\Sigma}. \quad (13)$$

□

We now show how this bound translates into a regret bound.

Theorem 1. Assume the distributions $\{\nu_k\}_{k=1}^K$ to be bounded in $[0, 1]$ and $n \geq 4K$. The regret of \mathcal{A}_{CH} , for parameter $\delta = n^{-5/2}$, is bounded as

$$R_n(\mathcal{A}_{CH}) \leq \frac{70K\sqrt{\log n}}{n^{3/2} \Sigma \lambda_{\min}^{5/2}} + O\left(\frac{\log n}{n^2}\right). \quad (14)$$

For space limitations, we only report a sketch of the proof here, the full proof is provided in the longer version of the paper (Carpentier et al., 2011).

Proof (Sketch). Eq. 3 indicates that the more often an arm is pulled, the smaller its estimation error becomes. However, this is not true in general because $T_{k,n}$ is a random variable that depends on the actual received samples, and thus, $L_{k,n} = \mathbb{E}_{\nu_k}[(\mu_k - \hat{\mu}_{k,n})^2]$ does not satisfy Eq. 3. Nevertheless, for any arm k , the number of pulls $T_{k,n}$ is a stopping time w.r.t. the filtration induced by the samples received for arm k . Hence, by applying the result of Lemma 10 in Antos et al. (2010) (a form of Wald's equality), one derive

$$\mathbb{E}[(\mu_k - \hat{\mu}_{k,n})^2 \mathbb{I}\{\xi_{K,n}(\delta)\}] \leq \frac{1}{T_{k,n}^2} \mathbb{E}\left[\left(\sum_{t=1}^{T_{k,n}} (\mu_k - X_{k,t})\right)^2\right] = \frac{\sigma_k^2 \mathbb{E}(T_{k,n})}{T_{k,n}^2}, \quad (15)$$

where $\underline{T}_{k,n}$ is a lower-bound for $T_{k,n}$ on $\xi_{K,n}(\delta)$. From this bound, one can use Lemma 11 which provides both upper and lower-bounds for $T_{k,n}$ on the event $\xi_{K,n}(\delta)$ to deduce that $\mathbb{E}[(\mu_k - \hat{\mu}_{k,n})^2 \mathbb{I}\{\xi_{K,n}(\delta)\}] = \frac{\sigma_k^2}{T_{k,n}^*} + O(n^{-3/2} \sqrt{\log(1/\delta)})$

and $\mathbb{E}[(\mu_k - \hat{\mu}_{k,n})^2 \mathbb{I}\{\xi_{K,n}(\delta)\}^c] \leq 1 \times \mathbb{P}(\xi_{K,n}(\delta)^c) \leq 4nK\delta$ (which is obvious). The claim follows by setting $\delta = n^{-5/2}$. \square

Remark 1. As discussed in Sec. 2, our objective is to design a sampling strategy capable of estimating the mean values of the arms almost as accurately as the optimal allocation strategy, which assumes that the variances are known. In fact, Thm. II shows that the CH-AS algorithm provides a uniformly accurate estimation of the expected values of the arms with a regret R_n of order $\tilde{O}(n^{-3/2})$. This regret rate is the same as for GAFS-MAX algorithm (Antos et al., 2010).

Remark 2. In addition to a linear dependency on the number of arms K , the bound also displays an inverse dependency on the smallest proportion λ_{\min} . As a result, the bound scales poorly when an arm has a very small variance relative to the other arms (i.e., $\sigma_k \ll \Sigma$). Note that GAFS-MAX has also a similar dependency on the inverse of λ_{\min} , although a precise comparison is not possible due to the fact that Antos et al. (2010) do not explicitly report the multiplicative constants in their regret bound. Moreover, Thm. II holds for any n whereas the regret bound in Antos et al. (2010) requires a condition $n \geq n_0$, where n_0 is a constant that scales with λ_{\min}^{-1} . Finally, note that this UCB type of algorithm (CH-AS) enables a much simpler regret analysis than that of GAFS-MAX.

Remark 3. It is clear from Lemma II that the inverse dependency on λ_{\min} appears in the bound on the number of pulls and then it is propagated to the regret bound. We now show with a simple example that this dependency is not an artifact of the analysis and it is intrinsic in the performance of the algorithm. Consider a two-arm problem with $\sigma_1^2 = 1$ and $\sigma_2^2 = 0$. Here the optimal allocation is $T_{1,n}^* = n - 1$, $T_{2,n}^* = 1$ (only one sample is enough to estimate the mean of the second arm), and $\lambda_{\min} = 0$, which makes the bound in Thm. II vacuous. This does not mean that CH-AS has an unbounded regret but it indicates that it minimizes the regret with a poorer rate (see Sec. A.3 in Carpentier et al. 2011, for a sketch of the proof). In fact, the upper-confidence term forces the algorithm to pull the arm with zero variance at least $\Omega(n^{2/3})$ times, which results in under-pulling the first arm by the same amount, and thus, in worsening its estimation. It can be shown that the resulting regret has the rate $\tilde{O}(n^{-4/3})$ and no dependency on λ_{\min} . So, it still decreases to zero faster than $1/n$, but with a slower rate than in Thm. II. Merging these two results, we deduce that the regret is in fact $R_n \leq \min\{\lambda_{\min}^{-5/2} \tilde{O}(n^{-3/2}), \tilde{O}(n^{-4/3})\}$. Note that when $\lambda_{\min} = 0$ the regret of GAFS-MAX is in $\tilde{O}(n^{-3/2})$, and GAFS-MAX thus outperforms CH-AS in this case. We further study the behavior of CH-AS in Sec. 5.1.

The reason for the poor performance in Lemma II is that Chernoff-Hoeffding's inequality is not tight for small-variance random variables. In Sec. 4, we propose an algorithm based on an empirical Bernstein's inequality, which is tighter for small-variance random variables, and prove that this algorithm under-pulls all the arms by *at most* $\tilde{O}(n^{1/2})$, without a dependency on λ_{\min} (see Eqs. I8 and I9).

⁵ See the end of Section 4 in Antos et al. (2010).

```

Input: parameters  $c_1, c_2, \delta$   

Let  $b = 4\sqrt{c_1 \log(c_2/\delta)}\sqrt{\log(2/\delta)} + 2\sqrt{5c_1}n^{-1/2}$   

Initialize: Pull each arm twice  

for  $t = 2K + 1, \dots, n$  do  

    Compute  $B_{q,t} = \frac{1}{T_{q,t-1}} \left( \hat{\sigma}_{q,t-1}^2 + 2b\hat{\sigma}_{q,t-1}\sqrt{\frac{1}{T_{q,t-1}}} + b^2\frac{1}{T_{q,t-1}} \right)$  for each  

    arm  $1 \leq q \leq K$   

    Pull an arm  $k_t \in \arg \max_{1 \leq q \leq K} B_{q,t}$   

end for  

Output:  $\hat{\mu}_{q,t}$  for each arm  $1 \leq q \leq K$ 

```

Fig. 2. The pseudo-code of the B-AS algorithm, with $\hat{\sigma}_{k,t}$ computed as in Eq. 16

4 Allocation Strategy Based on Bernstein UCB

In this section, we present another UCB-like algorithm, called *Bernstein Allocation Strategy* (B-AS), based on a Bernstein's inequality for the variances of the arms, with an improved bound on $|T_{k,n} - T_{k,n}^*|$ without the inverse dependency on λ_{\min} (compare the bounds in Eqs. 18 and 19 to the one for CH-AS in Eq. 6). However this result itself is not sufficient to derive a better regret bound than CH-AS. This finding shows that even an adaptive algorithm which implements a strategy close to the optimal allocation strategy may still incur a regret that poorly scales with the smallest proportion λ_{\min} . We further investigate this issue by showing that the way the bound of the number of pulls translates into a regret bound depends on the specific distributions of the arms. In fact, when the sample distributions are Gaussian, we can exploit the property that the empirical mean $\hat{\mu}_{k,t}$ conditioned on $T_{k,t}$ is independent of the empirical variances $(\hat{\sigma}_{k,s})_{s \leq t}$ and further deduce that the regret of B-AS no longer depends on λ_{\min}^{-1} . The numerical simulations in Sec. 5 further illustrate this theoretical finding.

4.1 The B-AS Algorithm

The B-AS algorithm (Fig. 2), \mathcal{A}_B , is based on a high-probability bounds (empirical Bernstein's inequality) on the variance of each arm (Maurer and Pontil, 2009; Audibert et al., 2009). B-AS requires three parameters as input (see also Remark 4 in Sec. 4.2 on how to reduce them to one) c_1 and c_2 , which are related to the shape of the distributions (see Assumption 1), and δ , which defines the *confidence level* of the bound. The amount of exploration of the algorithm can be adapted by properly tuning these parameters. The algorithm is similar to CH-AS except that the bounds $B_{q,t}$ on each arm are computed as

$$B_{q,t} = \frac{1}{T_{q,t-1}} \left(\hat{\sigma}_{q,t-1}^2 + 2b\hat{\sigma}_{q,t-1}\sqrt{\frac{1}{T_{q,t-1}}} + b^2\frac{1}{T_{q,t-1}} \right),$$

where $b = 4\sqrt{c_1 \log(c_2/\delta)}\sqrt{\log(2/\delta)} + 2\sqrt{5c_1}n^{-1/2}$ and⁶

⁶ We consider the unbiased estimator of the variance here.

$$\hat{\sigma}_{k,t}^2 = \frac{1}{T_{k,t} - 1} \sum_{i=1}^{T_{k,t}} (X_{k,i} - \hat{\mu}_{k,t})^2, \quad \text{with} \quad \hat{\mu}_{k,t} = \frac{1}{T_{k,t}} \sum_{i=1}^{T_{k,t}} X_{k,i}. \quad (16)$$

4.2 Regret Bound and Discussion

Instead of bounded distributions, we consider the more general assumption of sub-Gaussian distributions.

Assumption 1 (Sub-Gaussian distributions). *There exist $c_1, c_2 > 0$ such that for all $1 \leq k \leq K$ and any $\epsilon > 0$,*

$$\mathbb{P}_{X \sim \nu_k}(|X - \mu_k| \geq \epsilon) \leq c_2 \exp(-\epsilon^2/c_1). \quad (17)$$

We first bound the difference between the B-AS and optimal allocation strategies.

Lemma 2. *Under Assumption 1 and for any $\delta > 0$, when the B-AS algorithm runs with parameters c_1, c_2 , and δ , with probability at least $1 - 2nK\delta$, we have $T_{p,\min} \leq T_{p,n} \leq T_{p,\max}$ for any arm $1 \leq p \leq K$ and any $n \geq \frac{16}{9}c(\delta)^{-2}$, where*

$$T_{p,\min} = T_{p,n}^* \\ - K\lambda_p \left[1 + \frac{16a\sqrt{\log(2/\delta)}}{\Sigma} \left(\sqrt{\Sigma} + \frac{2a\sqrt{\log(2/\delta)}}{c(\delta)} \right) \sqrt{n} + 128Ka^2 \frac{\log(2/\delta)}{\Sigma\sqrt{c(\delta)}} n^{1/4} \right], \quad (18)$$

and

$$T_{p,\max} = T_{p,n}^* \\ + K \left[1 + \frac{16a\sqrt{\log(2/\delta)}}{\Sigma} \left(\sqrt{\Sigma} + \frac{2a\sqrt{\log(2/\delta)}}{c(\delta)} \right) \sqrt{n} + 128Ka^2 \frac{\log(2/\delta)}{\Sigma\sqrt{c(\delta)}} n^{1/4} \right], \quad (19)$$

where $c(\delta) = \frac{2a\sqrt{\log(2/\delta)}}{\sqrt{K}(\sqrt{\Sigma} + 4a\sqrt{\log(2/\delta)})}$ and $a = 2\sqrt{c_1 \log(c_2/\delta)} + \sqrt{\frac{5c_1}{\log(2/\delta)}} n^{-1/2}$.

Remark. Unlike the bounds for CH-AS in Lemma 1, B-AS allocates the arms such that the difference between $T_{p,n}$ and $T_{p,n}^*$ grows at most as $\tilde{O}(\sqrt{n})$ without dependency on λ_{\min}^{-1} . This overcomes the limitation of CH-AS, which, as discussed in Remark 3 of Sec. 3.2, may over-sample (thus also under-sample) some arms by $O(n^{2/3})$ whenever λ_{\min} is small. We further notice that the lower bound in Eq. 18 is of order $\lambda_p \tilde{O}(\sqrt{n})$, which implies that the gap between $T_{p,n}$ and $T_{p,n}^*$ decreases as λ_p becomes smaller. This is not the case in the upper bound, where the gap is of order $\tilde{O}(\sqrt{n})$, but is independent of the value of λ_p . This explains why in the case of general distributions, B-AS has a regret bound with an inverse dependency on λ_{\min} (similar to CH-AS), as shown in Thm. 2.

Theorem 2. *Under Assumption 1, for any $n \geq 4K$, the regret of \mathcal{A}_B run with parameters c_1, c_2 , and $\delta = (c_2 + 2)n^{-5/2}$ is bounded as*

$$R_n(\mathcal{A}_B) \leq \left[\frac{CK^5 c_1^2}{\lambda_{\min}^2} \log(n)^2 \left(\Sigma + 200 \log(n) \right) \left(1 + \frac{1}{\Sigma^3} \right) + 2c_1(c_2 + 2)K \right] n^{-3/2},$$

where C is a constant (a loose numerical value for C is 30000).

Similar to Thm. 11, the bound on the number of pulls translates into a regret bound through Eq. 15. As it can be noticed, in order to remove the dependency on λ_{\min} , a symmetric bound on $|T_{p,n} - T_{p,n}^*| \leq \lambda_p \tilde{O}(\sqrt{n})$ would be needed. While the lower bound in Eq. 18 decreases with λ_p , the upper bound scales with $\tilde{O}(\sqrt{n})$. Whether there exists an algorithm with a tighter upper bound scaling with λ_p is still an open question. In the next section, we show that an improved regret bound can be achieved in the special case of Gaussian distributions.

4.3 Regret for Gaussian Distributions

In the case of Gaussian distributions, the loss bound in Eq. 15 can be improved as in the following lemma (the full proof is reported in [Carpentier et al. \(2011\)](#)).

Lemma 3. *Assume that distributions $\{\nu_k\}_{k=1}^K$ are Gaussian. Then for any k*

$$\mathbb{E}[(\hat{\mu}_{k,n} - \mu_k)^2] \leq \frac{\sigma_k^2}{T_{k,\min}} + \sigma_k^2 \delta', \quad (20)$$

where $T_{k,n} \geq T_{k,\min}$ is the lower-bound in Lemma 2 which holds with probability at least $1 - \delta'$ (where $\delta' = 2nK\delta$).

Proof (Sketch). We first write the loss for any arm k as

$$\mathbb{E}[(\hat{\mu}_{k,n} - \mu_k)^2] = \sum_{t=2}^n \mathbb{E}[(\hat{\mu}_{k,n} - \mu_k)^2 | T_{k,n} = t] \mathbb{P}(T_{k,n} = t). \quad (21)$$

We notice that $T_{k,n}$ is a random stopping time which depends on the sequence of empirical variances for arm k and the empirical variances of all the other arms. The event $\{T_{k,n} \geq t\}$ depends on the filtration $\mathcal{F}_{k,t}$ (generated by the sequence of empirical variances of the rewards of arm k) and on the “environment of arm k ” \mathcal{E}_{-k} (defined by all the rewards samples of other arms). We recall that for a Gaussian distribution $\mathcal{N}(\mu_k, \sigma_k^2)$, the empirical mean $\hat{\mu}_{k,n}$ built on a fixed number t of independent samples is distributed as a normal distribution $\mathcal{N}(\mu_k, \sigma_k^2/t)$ and it is independent from the empirical variance $\hat{\sigma}_{k,n}^2$. According to [Carpentier et al. \(2011\)](#), this property can be extended to the conditional random variable $\hat{\mu}_{k,n} | \mathcal{F}_{k,n}, \mathcal{E}_{-k}$ which is still distributed as $\mathcal{N}(\mu_k, \sigma_k^2/t)$. Using this property in (21) we have

$$\mathbb{E}[(\hat{\mu}_{k,n} - \mu_k)^2] = \sum_{t=2}^n \frac{\sigma_k^2}{t} \mathbb{P}(T_{k,n} = t) = \sigma_k^2 \mathbb{E}\left[\frac{1}{T_{k,n}}\right].$$

Using the lower-bound in Lemma 2 the statement follows. \square

Remark 1. We notice that the loss bound in Eq. 20 does not require any upper bound on $T_{k,n}$. It is actually similar to the case of deterministic allocation. When $\tilde{T}_{k,n}$ is a deterministic number of pulls, the corresponding loss resulting from pulling arm k , $\tilde{T}_{k,n}$ times, is $L_{k,n} = \sigma_k^2/\tilde{T}_{k,n}$. In general, when $T_{k,n}$ is a random variable depending on the empirical variances $\{\hat{\sigma}_k^2\}_k$ (as in CH-AS and B-AS), the conditional expectation $\mathbb{E}[(\hat{\mu}_{k,n} - \mu_k)^2 | T_{k,n} = t]$ no longer equals

σ_k^2/t . However, for Gaussian distributions we recover the property $\mathbb{E}[(\hat{\mu}_{k,n} - \mu_k)^2 | T_{k,n} = t] = \sigma_k^2/t$, which allows us to deduce the result reported in Lemma 3.

We now report a regret bound in the case of Gaussian distributions. Note that in this case, Assumption I holds for $c_1 = 2\Sigma$ and $c_2 = 1$.⁷

Theorem 3. Assume that $\{\nu_k\}_{k=1}^K$ are Gaussian and that an upper-bound $\bar{\Sigma} \geq \Sigma$. B-AS with parameters $c_1 = 2\bar{\Sigma}$, $c_2 = 1$, and $\delta = n^{-5/2}$ has a regret

$$R_n(\mathcal{A}_B) \leq C\bar{\Sigma}K^{3/2}(\log(2n))^2 n^{-3/2} + O(n^{-7/4}(\log n)^2), \quad (22)$$

where C is a constant (a loose numerical value for C is 19200).

Remark 2. In the case of Gaussian distributions, the regret bound for B-AS has the rate $\tilde{O}(n^{-3/2})$ without dependency on λ_{\min} , which represents a significant improvement over the regret bounds for the CH-AS and GAFS-MAX algorithms.

Remark 3. In practice, there is no need to tune the three parameters c_1 , c_2 , and δ separately. In fact, it is enough to tune the algorithm for a single parameter b (see Fig. 2). Using the proof of Thm. 3, it is possible to show that the expected regret is minimized by choosing $b = O(\max\{\bar{\Sigma}^{3/2}, \sqrt{\bar{\Sigma}}\} \log n)$, which only requires an upper bound on the value of Σ . This is a reasonable assumption whenever a rough estimate of the magnitude of the variances is available.

5 Numerical Experiments

5.1 CH-AS, B-AS, and GAFS-MAX with Gaussian Arms

In this section, we compare the performance of CH-AS, B-AS, and GAFS-MAX on a two-armed problem with Gaussian distributions $\nu_1 = \mathcal{N}(0, \sigma_1^2 = 4)$ and $\nu_2 = \mathcal{N}(0, \sigma_2^2 = 1)$ (note that $\lambda_{\min}=1/5$). Fig. 3 (left) shows the rescaled regret, $n^{3/2}R_n$, for the three algorithms averaged over 50,000 runs. The results indicate that while the rescaled regret is almost constant w.r.t. n in B-AS and GAFS-MAX, it increases for small (relative to λ_{\min}^{-1}) values of n in CH-AS.

The robust behavior of B-AS when the distributions of the arms are Gaussian may be easily explained by the bound of Thm. 3 (Eq. 22). The initial increase in the CH-AS curve is also consistent with the bound of Thm. I (Eq. 14). As discussed in Remark 3 of Sec. 3.2, the regret bound for CH-AS is of the form $R_n \leq \min\{\lambda_{\min}^{-5/2}\tilde{O}(n^{-3/2}), \tilde{O}(n^{-4/3})\}$, and thus, the algorithm behaves as $\tilde{O}(n^{-4/3})$ and $\lambda_{\min}^{-5/2}\tilde{O}(n^{-3/2})$ for small and large (relative to λ_{\min}^{-1}) values of n , respectively. It is important to note that the behavior of CH-AS is independent of the arms' distributions and is intrinsic in the allocation mechanism, as shown in Lemma I. Finally, the behavior of GAFS-MAX indicates that although its analysis shows an inverse dependency on λ_{\min} and yields a regret bounds similar to CH-AS,

⁷ For a single Gaussian distribution $c_1 = 2\sigma^2$. Here we use $c_1 = 2\Sigma$ in order for the assumption to be satisfied for all K distributions simultaneously.

its rescaled regret in fact does not grow with n when the distributions of the arms are Gaussian. This is why we believe that it would be possible to improve the GAFS-MAX analysis by bounding the standard deviation using Bernstein's inequality. This would remove the inverse dependency on λ_{\min} and provide a regret bound similar to B-AS in the case of Gaussian distributions.

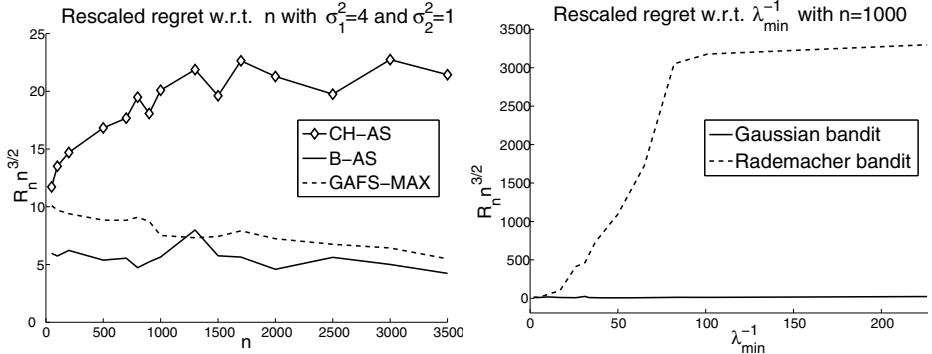


Fig. 3. (left) The rescaled regret of CH-AS, B-AS, and GAFS-MAX algorithms on a two-armed problem, where the distributions of the arms are Gaussian. (right) The rescaled regret of B-AS for two bandit problems, one with two Gaussian arms and one with a Gaussian and a Rademacher arms.

5.2 B-AS with Non-gaussian Arms

In Sec. 4.3, we showed that when the arms have Gaussian distributions, the regret bound of the B-AS algorithm does not depend on λ_{\min} anymore. We also discussed on why we conjecture that it is not possible to remove this dependency in case of general distributions unless tighter upper bounds on the number of pulls can be derived. Although we do not yet have a lower bound on the regret showing the dependency on λ_{\min} , in this section we empirically show that the shape of the distributions directly impacts the regret of the B-AS algorithm.

As discussed in Sec. 4.3, the property of Gaussian distributions that allows us to remove the λ_{\min} dependency in the regret bound of B-AS is that the empirical mean $\hat{\mu}_{k,n}$ of each arm k is independent of its empirical variance $\hat{\sigma}_{k,n}^2$ conditioned on $T_{k,n}$. Although this property might approximately hold for a larger family of distributions, there are distributions, such as Rademacher, for which these quantities are negatively correlated. In the case of Rademacher distribution,⁸ the loss $(\hat{\mu}_{k,t} - \mu_k)^2$ is equal to $\hat{\mu}_{k,t}^2$ and we have $\hat{\sigma}_{k,t}^2 = \frac{1}{T_{k,t}} \sum_{i=1}^{T_{k,t}} X_{k,i}^2 - \hat{\mu}_{k,t}^2 = 1 - \hat{\mu}_{k,t}^2$, as a result, the larger $\hat{\sigma}_{k,t}^2$, the smaller $\hat{\mu}_{k,t}^2$. We know that the allocation strategies in CH-AS, B-AS, and GAFS-MAX are based on the empirical variance which is used as a substitute for the true variance. As a result, the larger $\hat{\sigma}_{k,t}^2$,

⁸ X is Rademacher if $X \in \{-1, 1\}$ and admits values -1 and 1 with equal probability.

the more often arm k is pulled. In case of Rademacher distributions, this means that an arm is pulled more than its optimal allocation exactly when its mean is accurately estimated (the loss is small). This may result in a poorer estimation of the arm, and thus, negatively affect the regret of the algorithm.

In the experiments of this section, we use B-AS in two different bandit problems: one with two Gaussian arms $\nu_1 = \mathcal{N}(0, \sigma_1^2)$ (with $\sigma_1 \geq 1$) and $\nu_2 = \mathcal{N}(0, 1)$, and one with a Gaussian $\nu_1 = \mathcal{N}(0, \sigma_1^2)$ and a Rademacher $\nu_2 = \mathcal{R}$ arms. Note that in both cases $\lambda_{\min} = \lambda_2 = 1/(1 + \sigma_1^2)$. Figure 3-(right) shows the rescaled regret ($n^{3/2} R_n$) of the B-AS algorithm as a function of λ_{\min}^{-1} for $n = 1000$. As expected, while the rescaled regret of B-AS is constant in the first problem, it increases with σ_1^2 in the second one. As explained above, this behavior is due to the poor approximation of the Rademacher arm which is over-pulled whenever its estimated mean is accurate. This result illustrates the fact that in this active learning problem (where the goal is to estimate the mean values of the arms), the performance of the algorithms that rely on the empirical-variances (e.g., CH-AS, B-AS, and GAFS-MAX) crucially depends on the shape of the distributions, and not only on their variances. This may be surprising since according to the central limit theorem the distribution of the empirical mean should tend to a Gaussian. However, it seems that what is important is not the distribution of the empirical mean or variance, but the correlation of these two quantities.

6 Conclusions and Open Questions

In this paper we studied the problem of the uniform estimation of the mean value of K independent distributions under a given sampling budget. We introduced a novel class of algorithms based on upper-confidence-bounds on the (unknown) variances of the arms, and analyzed two algorithms: CH-AS and B-AS. For CH-AS we derived a regret bound similar to Antos et al. (2010), scaling as $\tilde{O}(n^{-3/2})$ and with a dependence on λ_{\min}^{-1} . We then introduced a more refined algorithm, B-AS, using a tighter upper bounds on the variance, and reported a refined regret bound in the case of Gaussian distributions. Finally we gave arguments (including numerical simulations) supporting the idea that the full shape of the distributions (and not only their variance) has a relevant impact on the performance of the allocation strategies.

This work opens a number of questions.

- *Distribution dependency.* An open question is to which extent the result for B-AS in case of Gaussian distributions could be extended to more general families of distributions. As illustrated in the case of Rademacher, the correlation between the empirical means and variances may cause the algorithm to over-pull arms even when their estimation is accurate, thus incurring a large regret. On the other hand, if the sample distributions are Gaussian, the empirical means and variances are uncorrelated and the allocation algorithms such as B-AS achieve a better regret. Further investigation is needed to identify whether this results can be extended to other distributions.

- *Lower bound.* The results in Secs. 4.3 and 5.2 suggest that the dependency on the distributions of the arms could be intrinsic in the allocation problem. If this is the case, it should be possible to derive a lower bound for this problem showing such dependency (a lower-bound with dependency on λ_{\min}^{-1}).

Acknowledgements. We thank András Antos for many comments that helped us to greatly improve the quality of the paper. This work was supported by French National Research Agency (ANR-08-COSI-004 project EXPLO-RA) and by the European Community’s Seventh Framework Programme (project COM-PLACS, grant agreement n°231495).

References

- Antos, A., Grover, V., Szepesvári, C.: Active learning in heteroscedastic noise. *Theoretical Computer Science* 411, 2712–2728 (2010)
- Audibert, J.-Y., Munos, R., Szepesvari, C.: Exploration-exploitation trade-off using variance estimates in multi-armed bandits. *Theoretical Computer Science* 410, 1876–1902 (2009)
- Audibert, J.-Y., Bubeck, S., Munos, R.: Best arm identification in multi-armed bandits. In: *Proceedings of the Twenty-Third Annual Conference on Learning Theory (COLT 2010)*, pp. 41–53 (2010)
- Bubeck, S., Munos, R., Stoltz, G.: Pure exploration in finitely-armed and continuous-armed bandits. *Theoretical Computer Science* 412, 1832–1852 (2011) ISSN 0304-3975
- Carpentier, A., Lazaric, A., Ghavamzadeh, M., Munos, R., Auer, P.: Upper-confidence-bound algorithms for active learning in multi-armed bandits. Technical Report inria-0059413, INRIA (2011)
- Castro, R., Willett, R., Nowak, R.: Faster rates in regression via active learning. In: *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 179–186 (2005)
- Chaudhuri, P., Mykland, P.A.: On efficient designing of nonlinear experiments. *Statistica Sinica* 5, 421–440 (1995)
- Cohn, D.A., Ghahramani, Z., Jordan, M.I.: Active learning with statistical models. *J. Artif. Int. Res.* 4, 129–145 (1996) ISSN 1076-9757
- Étoré, P., Jourdain, B.: Adaptive optimal allocation in stratified sampling methods. *Methodology and Computing in Applied Probability* 12, 335–360 (2010)
- Fedorov, V.: *Theory of Optimal Experiments*. Academic Press, London (1972)
- Maurer, A., Pontil, M.: Empirical bernstein bounds and sample-variance penalization. In: *Proceedings of the Twenty-Second Annual Conference on Learning Theory*, pp. 115–124 (2009)

The Perceptron with Dynamic Margin

Constantinos Panagiotakopoulos and Petroula Tsampouka

Physics Division, School of Technology
Aristotle University of Thessaloniki, Greece
`costapan@eng.auth.gr, petroula@gen.auth.gr`

Abstract. The classical perceptron rule provides a varying upper bound on the maximum margin, namely the length of the current weight vector divided by the total number of updates up to that time. Requiring that the perceptron updates its internal state whenever the normalized margin of a pattern is found not to exceed a certain fraction of this dynamic upper bound we construct a new approximate maximum margin classifier called the perceptron with dynamic margin (PDM). We demonstrate that PDM converges in a finite number of steps and derive an upper bound on them. We also compare experimentally PDM with other perceptron-like algorithms and support vector machines on hard margin tasks involving linear kernels which are equivalent to 2-norm soft margin.

Keywords: Online learning, classification, maximum margin.

1 Introduction

It is a common belief that learning machines able to produce solution hyperplanes with large margins exhibit greater generalization ability [21] and this justifies the enormous interest in Support Vector Machines (SVMs) [21, 2]. Typically, SVMs obtain large margin solutions by solving a constrained quadratic optimization problem using dual variables. In their native form, however, efficient implementation is hindered by the quadratic dependence of their memory requirements in the number of training examples a fact which renders prohibitive the processing of large datasets. To overcome this problem decomposition methods [15, 6] were developed that apply optimization only to a subset of the training set. Although such methods led to improved convergence rates, in practice their superlinear dependence on the number of examples, which can be even cubic, can still lead to excessive runtimes when large datasets are processed. Recently, the so-called linear SVMs [7, 8, 13] made their appearance. They take advantage of linear kernels in order to allow parts of them to be written in primal notation and were shown to outperform decomposition SVMs when dealing with massive datasets.

The above considerations motivated research in alternative large margin classifiers naturally formulated in primal space long before the advent of linear SVMs. Such algorithms are mostly based on the perceptron [16, 12], the simplest online learning algorithm for binary linear classification. Like the perceptron,

they focus on the primal problem by updating a weight vector which represents at each step the current state of the algorithm whenever a data point presented to it satisfies a specific condition. It is the ability of such algorithms to process one example at a time¹ that allows them to spare time and memory resources and consequently makes them able to handle large datasets. The first algorithm of that kind is the perceptron with margin [3] which is much older than SVMs. It is an immediate extension of the perceptron which provably achieves solutions with only up to $1/2$ of the maximum margin [10]. Subsequently, various algorithms succeeded in approximately attaining maximum margin by employing modified perceptron-like update rules. Such algorithms include ROMMA [11], ALMA [5], CRAMMA [19] and MICRA [20]. Very recently, the same goal was accomplished by a generalized perceptron with margin, the margintron [14].

The most straightforward way of obtaining large margin solutions through a perceptron is by requiring that the weight vector be updated every time the example presented to the algorithm has (normalized) margin which does not exceed a predefined value [17, 18, 1]. The obvious problem with this idea, however, is that the algorithm with such a fixed margin condition will definitely not converge unless the target value of the margin is smaller than the unknown maximum margin. In an earlier work [14] we noticed that the upper bound $\|\mathbf{a}_t\|/t$ on the maximum margin, with $\|\mathbf{a}_t\|$ being the length of the weight vector and t the number of updates, that comes as an immediate consequence of the perceptron update rule is very accurate and tends to improve as the algorithm achieves larger margins. In the present work we replace the fixed target margin value with a fraction $1 - \epsilon$ of this varying upper bound on the maximum margin. The hope is that as the algorithm keeps updating its state the upper bound will keep approaching the maximum margin and convergence to a solution with the desired accuracy ϵ will eventually occur. Thus, the resulting algorithm may be regarded as a realizable implementation of the perceptron with fixed margin condition.

The rest of this paper is organized as follows. Section 2 contains some preliminaries and a motivation of the algorithm based on a qualitative analysis. In Sect. 3 we give a formal theoretical analysis. Section 4 is devoted to implementational issues. Section 5 contains our experimental results while Sect. 6 our conclusions.

2 Motivation of the Algorithm

Let us consider a linearly separable training set $\{(\mathbf{x}_k, l_k)\}_{k=1}^m$, with vectors $\mathbf{x}_k \in \mathbb{R}^d$ and labels $l_k \in \{+1, -1\}$. This training set may either be the original dataset or the result of a mapping into a feature space of higher dimensionality [21, 2]. Actually, there is a very well-known construction [4] making linear separability always possible, which amounts to the adoption of the 2-norm soft margin. By placing \mathbf{x}_k in the same position at a distance ρ in an additional dimension, i.e. by extending \mathbf{x}_k to $[\mathbf{x}_k, \rho]$, we construct an embedding of our data into the so-called augmented space [3]. This way, we construct hyperplanes possessing bias

¹ The conversion of online algorithms to the batch setting is done by cycling repeatedly through the dataset and using the last hypothesis for prediction.

in the non-augmented feature space. Following the augmentation, a reflection with respect to the origin of the negatively labeled patterns is performed by multiplying every pattern with its label. This allows for a uniform treatment of both categories of patterns. Also, $R \equiv \max_k \|\mathbf{y}_k\|$ with $\mathbf{y}_k \equiv [l_k \mathbf{x}_k, l_k \rho]$ the k^{th} augmented and reflected pattern. Obviously, $R \geq \rho$.

The relation characterizing optimally correct classification of the training patterns \mathbf{y}_k by a weight vector \mathbf{u} of unit norm in the augmented space is

$$\mathbf{u} \cdot \mathbf{y}_k \geq \gamma_d \equiv \max_{\mathbf{u}' : \|\mathbf{u}'\|=1} \min_i \{\mathbf{u}' \cdot \mathbf{y}_i\} \quad \forall k . \quad (1)$$

We shall refer to γ_d as the maximum directional margin. It coincides with the maximum margin in the augmented space with respect to hyperplanes passing through the origin. For the maximum directional margin γ_d and the maximum geometric margin γ in the non-augmented feature space, it holds that $1 \leq \gamma/\gamma_d \leq R/\rho$. As $\rho \rightarrow \infty$, $R/\rho \rightarrow 1$ and, consequently, $\gamma_d \rightarrow \gamma$ [17, 18].

We consider algorithms in which the augmented weight vector \mathbf{a}_t is initially set to zero, i.e. $\mathbf{a}_0 = \mathbf{0}$, and is updated according to the classical perceptron rule

$$\mathbf{a}_{t+1} = \mathbf{a}_t + \mathbf{y}_k \quad (2)$$

each time an appropriate misclassification condition is satisfied by a training pattern \mathbf{y}_k . Taking the inner product of (2) with the optimal direction \mathbf{u} and using (1) we get

$$\mathbf{u} \cdot \mathbf{a}_{t+1} - \mathbf{u} \cdot \mathbf{a}_t = \mathbf{u} \cdot \mathbf{y}_k \geq \gamma_d$$

a repeated application of which gives [12]

$$\|\mathbf{a}_t\| \geq \mathbf{u} \cdot \mathbf{a}_t \geq \gamma_d t . \quad (3)$$

From (3) we readily obtain

$$\gamma_d \leq \frac{\|\mathbf{a}_t\|}{t} \quad (4)$$

provided $t > 0$. Notice that the above upper bound on the maximum directional margin γ_d is an immediate consequence of the classical perceptron rule and holds independent of the misclassification condition.

It would be very desirable that $\|\mathbf{a}_t\|/t$ approaches γ_d with t increasing since this would provide an after-run estimate of the accuracy achieved by an algorithm employing the classical perceptron update. More specifically, with γ'_d being the directional margin achieved upon convergence of the algorithm in t_c updates, it holds that

$$\frac{\gamma_d - \gamma'_d}{\gamma_d} \leq 1 - \frac{\gamma'_d t_c}{\|\mathbf{a}_{t_c}\|} . \quad (5)$$

In order to understand the mechanism by which $\|\mathbf{a}_t\|/t$ evolves we consider the difference between two consecutive values of $\|\mathbf{a}_t\|^2/t^2$ which may be shown to be given by the relation

$$\frac{\|\mathbf{a}_t\|^2}{t^2} - \frac{\|\mathbf{a}_{t+1}\|^2}{(t+1)^2} = \frac{1}{t(t+1)} \left\{ \left(\frac{\|\mathbf{a}_t\|^2}{t} - \mathbf{a}_t \cdot \mathbf{y}_k \right) + \left(\frac{\|\mathbf{a}_{t+1}\|^2}{t+1} - \mathbf{a}_{t+1} \cdot \mathbf{y}_k \right) \right\} . \quad (6)$$

Let us assume that satisfaction of the misclassification condition by a pattern \mathbf{y}_k has as a consequence that $\|\mathbf{a}_t\|^2/t > \mathbf{a}_t \cdot \mathbf{y}_k$ (i.e., the normalized margin $\mathbf{u}_t \cdot \mathbf{y}_k$ of \mathbf{y}_k (with $\mathbf{u}_t = \mathbf{a}_t / \|\mathbf{a}_t\|$) is smaller than the upper bound (4) on γ_d). Let us further assume that after the update has taken place \mathbf{y}_k still satisfies the misclassification condition and therefore $\|\mathbf{a}_{t+1}\|^2/(t+1) > \mathbf{a}_{t+1} \cdot \mathbf{y}_k$. Then, the r.h.s. of (6) is positive and $\|\mathbf{a}_t\|/t$ decreases as a result of the update. In the event, instead, that the update leads to violation of the misclassification condition, $\|\mathbf{a}_{t+1}\|^2/(t+1)$ may be smaller than $\mathbf{a}_{t+1} \cdot \mathbf{y}_k$ and $\|\mathbf{a}_t\|/t$ may not decrease as a result of the update. We expect that statistically, at least in the early stages of the algorithm, most updates do not lead to correctly classified patterns (i.e., patterns which violate the misclassification condition) and as a consequence $\|\mathbf{a}_t\|/t$ will have the tendency to decrease. Obviously, the rate of this decrease depends on the size of the difference $\|\mathbf{a}_t\|^2/t - \mathbf{a}_t \cdot \mathbf{y}_k$ which, in turn, depends on the misclassification condition and how amply it is satisfied.

If we are interested in obtaining solutions possessing margin the most natural choice of misclassification condition is the fixed (normalized) margin condition

$$\mathbf{a}_t \cdot \mathbf{y}_k \leq (1 - \epsilon)\gamma_d \|\mathbf{a}_t\| \quad (7)$$

with the accuracy parameter ϵ satisfying $0 < \epsilon \leq 1$. This is an example of a misclassification condition which if it is satisfied ensures that $\|\mathbf{a}_t\|^2/t > \mathbf{a}_t \cdot \mathbf{y}_k$. Moreover, by making use of (1) and (7) it may be shown² that $\|\mathbf{a}_{t+1}\|^2/(t+1) \geq \mathbf{a}_{t+1} \cdot \mathbf{y}_k$ for $t \geq \epsilon^{-1}R^2/\gamma_d^2$. Thus, after at most $\epsilon^{-1}R^2/\gamma_d^2$ updates $\|\mathbf{a}_t\|/t$ decreases monotonically. The perceptron algorithm with fixed margin condition (PFM) is known to converge in a finite number of updates to an ϵ -accurate approximation of the maximum directional margin hyperplane [17, 18, 1]. Although it appears that PFM demands exact knowledge of the value of γ_d , we notice that only the value of $\beta \equiv (1 - \epsilon)\gamma_d$, which is the quantity entering (7), needs to be set and not the values of ϵ and γ_d separately. That is why the after-run estimate (5) is useful in connection with the algorithm in question. Nevertheless, in order to make sure that $\beta < \gamma_d$ a priori knowledge of a fairly good lower bound on γ_d is required and this is an obvious defect of PFM.

The above difficulty associated with the fixed margin condition may be remedied if the unknown γ_d is replaced for $t > 0$ with its varying upper bound $\|\mathbf{a}_t\|/t$

$$\mathbf{a}_t \cdot \mathbf{y}_k \leq (1 - \epsilon) \frac{\|\mathbf{a}_t\|^2}{t}. \quad (8)$$

Condition (8) ensures that $\|\mathbf{a}_t\|^2/t - \mathbf{a}_t \cdot \mathbf{y}_k \geq \epsilon \|\mathbf{a}_t\|^2/t > 0$. Moreover, as in the case of the fixed margin condition, $\|\mathbf{a}_{t+1}\|^2/(t+1) - \mathbf{a}_{t+1} \cdot \mathbf{y}_k \geq 0$ for $t \geq \epsilon^{-1}R^2/\gamma_d^2$. As a result, after at most $\epsilon^{-1}R^2/\gamma_d^2$ updates the r.h.s. of (6) is bounded from below by $\epsilon \|\mathbf{a}_t\|^2/t^2(t+1) \geq \epsilon \gamma_d^2/(t+1)$ and $\|\mathbf{a}_t\|/t$ decreases

² Combining (4) and (7) we get (8) from where $(t-1)\mathbf{a}_t \cdot \mathbf{y}_k \leq ((t-1)/t)(1-\epsilon)\|\mathbf{a}_t\|^2 \leq (1-\epsilon)\|\mathbf{a}_t\|^2$. Also from $t \geq \epsilon^{-1}R^2/\gamma_d^2$, using (3), we obtain $tR^2 \leq \epsilon\gamma_d^2 t^2 \leq \epsilon \|\mathbf{a}_t\|^2$. Thus, $(t-1)\mathbf{a}_t \cdot \mathbf{y}_k + t\|\mathbf{y}_k\|^2 \leq (t-1)\mathbf{a}_t \cdot \mathbf{y}_k + tR^2 \leq (1-\epsilon)\|\mathbf{a}_t\|^2 + \epsilon \|\mathbf{a}_t\|^2 = \|\mathbf{a}_t\|^2$. But $\|\mathbf{a}_t\|^2 \geq (t-1)\mathbf{a}_t \cdot \mathbf{y}_k + t\|\mathbf{y}_k\|^2$ is equivalent to $\|\mathbf{a}_{t+1}\|^2 \geq (t+1)\mathbf{a}_{t+1} \cdot \mathbf{y}_k$.

The Perceptron with Dynamic Margin

Input: A linearly separable augmented dataset $S = (\mathbf{y}_1, \dots, \mathbf{y}_k, \dots, \mathbf{y}_m)$ with reflection assumed
Fix: ϵ
Define: $q_k = \|\mathbf{y}_k\|^2$, $\bar{\epsilon} = 1 - \epsilon$
Initialize: $t = 0$, $\mathbf{a}_0 = \mathbf{0}$, $\ell_0 = 0$, $\theta_0 = 0$
repeat

for $k = 1$ **to** m **do**

if $p_{tk} \leq \theta_t$ **then**

$\mathbf{a}_{t+1} = \mathbf{a}_t + \mathbf{y}_k$

$\ell_{t+1} = \ell_t + 2p_{tk} + q_k$

$t \leftarrow t + 1$

$\theta_t = \bar{\epsilon} \ell_t / t$

until no update made within the **for loop**

originating from (7) with γ_d replaced for $t > 0$ by its dynamic upper bound $\|\mathbf{a}_t\|/t$, will be named the perceptron with dynamic margin (PDM).

3 Theoretical Analysis

From the discussion that led to the formulation of PDM it is apparent that if the algorithm converges it will achieve by construction a solution possessing directional margin larger than $(1 - \epsilon)\gamma_d$. (We remind the reader that convergence assumes violation of the misclassification condition (8) by all patterns. In addition, (4) holds.) The same obviously applies to PFM. Thus, for both algorithms it only remains to be demonstrated that they converge in a finite number of steps. This has already been shown for PFM [17, 18, 1] but no general ϵ -dependent bound in closed form has been derived. In the present section we demonstrate convergence of PDM and provide explicit bounds for both PFM and PDM.

Before we proceed with our analysis we will need the following result.

Lemma 1. *Let the variable $t \geq e^{-C}$ satisfy the inequality*

$$t \leq \delta(1 + C + \ln t) , \quad (9)$$

where δ, C are constants and $\delta > e^{-C}$. Then

$$t \leq t_0 \equiv (1 + e^{-1})\delta(C + \ln((1 + e)\delta)) . \quad (10)$$

Proof. If $t \geq e^{-C}$ then $(1 + C + \ln t) \geq 1$ and inequality (9) is equivalent to $f(t) = t/(1 + C + \ln t) - \delta \leq 0$. For the function $f(t)$ defined in the interval $[e^{-C}, +\infty)$ it holds that $f(e^{-C}) < 0$ and $df/dt = (C + \ln t)/(1 + C + \ln t)^2 > 0$

³ Let $f_t \equiv \|\mathbf{a}_t\|^2/t^2$ with $\gamma_d^2 \leq f_t \leq R^2$. For $t \geq N = [\epsilon^{-1}R^2/\gamma_d^2] + 1$ ($[x]$ is the integer part of $x \geq 0$) we have $f_t - f_{t+1} \geq \epsilon f_t/(t+1)$ or $(1 - \epsilon/(t+1))f_t \geq f_{t+1}$ from where $\ln(f_t/f_{t+1}) \geq -\ln(1 - \epsilon/(t+1)) \geq \epsilon/(t+1)$. Repeated application of the previous inequality gives $\ln(f_N/f_t) \geq \epsilon \sum_{k=N+1}^t k^{-1} \geq \epsilon \int_{N+1}^t k^{-1} dk = \epsilon \ln(t/(N+1))$. But $f_N/f_t \leq R^2/\gamma_d^2$. Then $t \leq (N+1)(R^2/\gamma_d^2)^{1/\epsilon}$. A refined analysis is given in Sect. 3.

for $t > e^{-C}$. Stated differently, $f(t)$ starts from negative values at $t = e^{-C}$ and increases monotonically. Therefore, if $f(t_0) \geq 0$ then t_0 is an upper bound of all t for which $f(t) \leq 0$. Indeed, it is not difficult to verify that $t_0 > \delta > e^{-C}$ and

$$f(t_0) = \delta \left((1 + e^{-1}) \left(1 + \frac{\ln \ln((1 + e)e^C \delta)}{\ln((1 + e)e^C \delta)} \right)^{-1} - 1 \right) \geq 0$$

given that $0 \leq \ln \ln x / \ln x \leq e^{-1}$ for $x \geq e$. \square

Now we are ready to derive an upper bound on the number of steps of PFM.

Theorem 1. *The number t of updates of the perceptron algorithm with fixed margin condition satisfies the bound*

$$t \leq \frac{(1 + e^{-1})}{2\epsilon} \frac{R^2}{\gamma_d^2} \left\{ 4 \frac{\gamma_d}{R} \left(1 - (1 - \epsilon) \frac{\gamma_d}{R} \right) + \ln \left(\frac{(1 + e)}{\epsilon} \frac{R}{\gamma_d} \left(1 - (1 - \epsilon) \frac{\gamma_d}{R} \right) \right) \right\} .$$

Proof. From (2) and (7) we get

$$\|\mathbf{a}_{t+1}\|^2 = \|\mathbf{a}_t\|^2 + 2\mathbf{a}_t \cdot \mathbf{y}_k + \|\mathbf{y}_k\|^2 \leq \|\mathbf{a}_t\|^2 \left(1 + \frac{2(1 - \epsilon)\gamma_d}{\|\mathbf{a}_t\|} + \frac{R^2}{\|\mathbf{a}_t\|^2} \right) .$$

Then, taking the square root and using the inequality $\sqrt{1+x} \leq 1+x/2$ we have

$$\|\mathbf{a}_{t+1}\| \leq \|\mathbf{a}_t\| \left(1 + \frac{2(1 - \epsilon)\gamma_d}{\|\mathbf{a}_t\|} + \frac{R^2}{\|\mathbf{a}_t\|^2} \right)^{1/2} \leq \|\mathbf{a}_t\| \left(1 + \frac{(1 - \epsilon)\gamma_d}{\|\mathbf{a}_t\|} + \frac{R^2}{2\|\mathbf{a}_t\|^2} \right).$$

Now, by making use of $\|\mathbf{a}_t\| \geq \gamma_d t$, we observe that

$$\|\mathbf{a}_{t+1}\| - \|\mathbf{a}_t\| \leq (1 - \epsilon)\gamma_d + \frac{R^2}{2\|\mathbf{a}_t\|} \leq (1 - \epsilon)\gamma_d + \frac{R^2}{2\gamma_d} \frac{1}{t} .$$

A repeated application of the above inequality $t - N$ times ($t > N \geq 1$) gives

$$\begin{aligned} \|\mathbf{a}_t\| - \|\mathbf{a}_N\| &\leq (1 - \epsilon)\gamma_d(t - N) + \frac{R^2}{2\gamma_d} \sum_{k=N}^{t-1} k^{-1} \\ &\leq (1 - \epsilon)\gamma_d(t - N) + \frac{R^2}{2\gamma_d} \left(\frac{1}{N} + \int_N^t k^{-1} dk \right) \end{aligned}$$

from where using the obvious bound $\|\mathbf{a}_N\| \leq RN$ we get an upper bound on $\|\mathbf{a}_t\|$

$$\|\mathbf{a}_t\| \leq (1 - \epsilon)\gamma_d t + \left(1 - (1 - \epsilon) \frac{\gamma_d}{R} \right) RN + \frac{R^2}{2\gamma_d} \left(\frac{1}{N} + \ln \frac{t}{N} \right) .$$

Combining the above upper bound on $\|\mathbf{a}_t\|$, which holds not only for $t > N$ but also for $t = N$, with the lower bound from (3) we obtain

$$t \leq \frac{1}{2\epsilon} \frac{R^2}{\gamma_d^2} \left\{ 2 \frac{\gamma_d}{R} \left(1 - (1 - \epsilon) \frac{\gamma_d}{R} \right) N + \frac{1}{N} - \ln N + \ln t \right\} .$$

Setting

$$\delta = \frac{1}{2\epsilon} \frac{R^2}{\gamma_d^2}, \quad \alpha = 2 \frac{\gamma_d}{R} \left(1 - (1-\epsilon) \frac{\gamma_d}{R}\right)$$

and choosing $N = [\alpha^{-1}] + 1$, with $[x]$ being the integer part of $x \geq 0$, we finally get

$$t \leq \delta(1 + 2\alpha + \ln \alpha + \ln t). \quad (11)$$

Notice that in deriving (11) we made use of the fact that $\alpha N + N^{-1} - \ln N \leq 1 + 2\alpha + \ln \alpha$. Inequality (11) has the form (9) with $C = 2\alpha + \ln \alpha$. Obviously, $e^{-C} < \alpha^{-1} < N \leq t$ and $e^{-C} < \alpha^{-1} \leq \delta$. Thus, the conditions of Lemma 1 are satisfied and the required bound, which is of the form (10), follows from (11). \square

Finally, we arrive at our main result which is the proof of convergence of PDM in a finite number of steps and the derivation of the relevant upper bound.

Theorem 2. *The number t of updates of the perceptron algorithm with dynamic margin satisfies the bound*

$$t \leq \begin{cases} t_0 \left(1 - \frac{1}{1-2\epsilon} \frac{R^2}{\gamma_d^2} t_0^{-1}\right)^{1/2\epsilon}, & t_0 \equiv [\epsilon^{-1}] \left(\frac{R}{\gamma_d}\right)^{1/\epsilon} \left(1 + \frac{[\epsilon^{-1}]^{-1}}{1-2\epsilon}\right)^{1/2\epsilon} \text{ if } \epsilon < \frac{1}{2} \\ (1+e^{-1}) \frac{R^2}{\gamma_d^2} \ln \left((1+e) \frac{R^2}{\gamma_d^2}\right) & \text{if } \epsilon = \frac{1}{2} \\ t_0 (1 - 2(1-\epsilon) t_0^{1-2\epsilon}), & t_0 \equiv \frac{\epsilon(3-2\epsilon)}{2\epsilon-1} \frac{R^2}{\gamma_d^2} \text{ if } \epsilon > \frac{1}{2}. \end{cases}$$

Proof. From (2) and (8) we obtain

$$\|\mathbf{a}_{t+1}\|^2 = \|\mathbf{a}_t\|^2 + 2\mathbf{a}_t \cdot \mathbf{y}_k + \|\mathbf{y}_k\|^2 \leq \|\mathbf{a}_t\|^2 \left(1 + \frac{2(1-\epsilon)}{t}\right) + R^2. \quad (12)$$

For $\epsilon \leq 1/2$, using the inequality $(1+x)^\zeta \geq 1 + \zeta x$ for $x \geq 0$, $\zeta = 2(1-\epsilon) \geq 1$ in (12) we get

$$\|\mathbf{a}_{t+1}\|^2 \leq \|\mathbf{a}_t\|^2 \left(1 + \frac{1}{t}\right)^{2(1-\epsilon)} + R^2. \quad (13)$$

For $\epsilon \geq 1/2$, instead, using the inequality $(1+x)^\zeta + \zeta(1-\zeta)x^2/2 \geq 1 + \zeta x$ for $x \geq 0$, $0 \leq \zeta = 2(1-\epsilon) \leq 1$ in (12) and the bound $\|\mathbf{a}_t\| \leq Rt$ we obtain

$$\begin{aligned} \|\mathbf{a}_{t+1}\|^2 &\leq \|\mathbf{a}_t\|^2 \left(1 + \frac{1}{t}\right)^{2(1-\epsilon)} + (1-\epsilon)(2\epsilon-1) \frac{\|\mathbf{a}_t\|^2}{t^2} + R^2 \\ &\leq \|\mathbf{a}_t\|^2 \left(1 + \frac{1}{t}\right)^{2(1-\epsilon)} + \epsilon(3-2\epsilon)R^2. \end{aligned} \quad (14)$$

By dividing both sides of (13) and (14) with $(t+1)^{2(1-\epsilon)}$ we arrive at

$$\frac{\|\mathbf{a}_{t+1}\|^2}{(t+1)^{2(1-\epsilon)}} - \frac{\|\mathbf{a}_t\|^2}{t^{2(1-\epsilon)}} \leq \frac{fR^2}{(t+1)^{2(1-\epsilon)}},$$

where $f = 1$ for $\epsilon \leq 1/2$ and $f = \epsilon(3 - 2\epsilon)$ for $\epsilon \geq 1/2$. A repeated application of the above inequality $t - N$ times ($t > N \geq 1$) gives

$$\frac{\|\mathbf{a}_t\|^2}{t^{2(1-\epsilon)}} - \frac{\|\mathbf{a}_N\|^2}{N^{2(1-\epsilon)}} \leq fR^2 \sum_{k=N+1}^t k^{-2(1-\epsilon)} \leq fR^2 \int_N^t k^{-2(1-\epsilon)} dk . \quad (15)$$

We also define $\alpha_t \equiv \|\mathbf{a}_t\|/Rt$ and observe that the bounds $\|\mathbf{a}_t\| \leq Rt$ and $\|\mathbf{a}_t\| \geq \gamma_d t$ confine α_t to lie in the range $\gamma_d/R \leq \alpha_t \leq 1$.

Let us assume that $\epsilon < 1/2$. Then, setting $\|\mathbf{a}_N\| = \alpha_N RN$ in (15) and performing the integration we get the following upper bound on $\|\mathbf{a}_t\|^2$

$$\|\mathbf{a}_t\|^2 \leq t^{2(1-\epsilon)} \alpha_N^2 R^2 N^{2\epsilon} \left\{ 1 + \frac{\alpha_N^{-2} N^{-1}}{2\epsilon - 1} \left(\left(\frac{t}{N} \right)^{2\epsilon-1} - 1 \right) \right\}$$

which combined with the lower bound $\|\mathbf{a}_t\|^2 \geq \gamma_d^2 t^2$ leads to

$$t^{2\epsilon} \leq \alpha_N^2 \frac{R^2}{\gamma_d^2} N^{2\epsilon} \left\{ 1 + \frac{\alpha_N^{-2} N^{-1}}{2\epsilon - 1} \left(\left(\frac{t}{N} \right)^{2\epsilon-1} - 1 \right) \right\} . \quad (16)$$

For $\epsilon < 1/2$ the term proportional to $(t/N)^{2\epsilon-1}$ in (16) is negative and may be dropped to a first approximation leading to the looser upper bound t_0

$$t_0 \equiv N \left(\alpha_N \frac{R}{\gamma_d} \right)^{1/\epsilon} \left(1 + \frac{\alpha_N^{-2} N^{-1}}{1 - 2\epsilon} \right)^{1/2\epsilon} \quad (17)$$

on the number t of updates. Then, we may replace t with its upper bound t_0 in the r.h.s. of (16) and get the improved bound

$$t \leq t_0 \left(1 - \frac{1}{1 - 2\epsilon} \frac{R^2}{\gamma_d^2} t_0^{-1} \right)^{1/2\epsilon} .$$

This is allowed given that the term proportional to $(t/N)^{2\epsilon-1}$ in (16) is negative and moreover t is raised to a negative power. Choosing $N = [\epsilon^{-1}]$ and $\alpha_N = 1$ (i.e., setting α_N to its upper bound which is the least favorable assumption) we obtain the bound stated in Theorem 2 for $\epsilon < 1/2$.

Now, let $\epsilon > 1/2$. Then, setting $N = 1$ in (15), using $\|\mathbf{a}_1\|^2 \leq R^2 \leq fR^2 = \epsilon(3 - 2\epsilon)R^2$ and performing the integration we get the following upper bound on $\|\mathbf{a}_t\|^2$

$$\|\mathbf{a}_t\|^2 \leq t^{2(1-\epsilon)} \epsilon(3 - 2\epsilon) R^2 \left(1 + \frac{t^{2\epsilon-1} - 1}{2\epsilon - 1} \right)$$

which combined with the lower bound $\|\mathbf{a}_t\|^2 \geq \gamma_d^2 t^2$ gives

$$t \leq \frac{\epsilon(3 - 2\epsilon)}{2\epsilon - 1} \frac{R^2}{\gamma_d^2} (1 - 2(1 - \epsilon)t^{1-2\epsilon}) . \quad (18)$$

For $\epsilon > 1/2$ the term proportional to $t^{1-2\epsilon}$ in (18) is negative and may be dropped to a first approximation leading to the looser upper bound t_0

$$t_0 \equiv \frac{\epsilon(3-2\epsilon)}{2\epsilon-1} \frac{R^2}{\gamma_d^2}$$

on the number t of updates. Then, we may replace t with its upper bound t_0 in the r.h.s. of (18) and get the improved bound stated in Theorem 2 for $\epsilon > 1/2$. This is allowed given that the term proportional to $t^{1-2\epsilon}$ in (18) is negative and moreover t is raised to a negative power.

Finally, for $\epsilon = 1/2$ setting $N = 1$ in (15), using $\|\mathbf{a}_1\|^2 \leq R^2$, $\|\mathbf{a}_t\|^2 \geq \gamma_d^2 t^2$ and performing the integration we get

$$t \leq \frac{R^2}{\gamma_d^2} (1 + \ln t)$$

which on account of Lemma 1 leads to the bound of Theorem 2 for $\epsilon = 1/2$. \square

Remark 1. The bound of Theorem 2 holds for PFM as well on account of (1).

The worst-case bound of Theorem 2 for $\epsilon \ll 1$ behaves like $\epsilon^{-1}(R/\gamma_d)^{1/\epsilon}$ which suggests an extremely slow convergence if we require margins close to the maximum. From expression (17) for t_0 , however, it becomes apparent that a more favorable assumption concerning the value of α_N (e.g., $\alpha_N \ll 1$ or even as low as $\alpha_N \sim \gamma_d/R$) after the first $N \gg \alpha_N^{-2}$ updates does lead to tremendous improvement provided, of course, that N is not extremely large. Such a sharp decrease of $\|\mathbf{a}_t\|/t$ in the early stages of the algorithm, which may be expected from relation (6) and the discussion that followed, lies behind its experimentally exhibited rather fast convergence.

It would be interesting to find a procedure by which the algorithm will be forced to a guaranteed sharp decrease of the ratio $\|\mathbf{a}_t\|/t$. The following two observations will be vital in devising such a procedure. First, we notice that when PDM with accuracy parameter ϵ has converged in t_c updates the threshold $(1-\epsilon)\|\mathbf{a}_{t_c}\|^2/t_c$ of the misclassification condition must have fallen below $\gamma_d \|\mathbf{a}_{t_c}\|^2$. Otherwise, the normalized margin $\mathbf{u}_{t_c} \cdot \mathbf{y}_k$ of all patterns \mathbf{y}_k would be larger than γ_d . Thus, $\alpha_{t_c} < (1-\epsilon)^{-1}\gamma_d/R$. Second, after convergence of the algorithm with accuracy parameter ϵ_1 in t_{c_1} updates we may lower the accuracy parameter from the value ϵ_1 to the value ϵ_2 and continue the run from the point where convergence with parameter ϵ_1 has occurred since for all updates that took place during the first run the misclassified patterns would certainly satisfy (at that time) the condition associated with the smaller parameter ϵ_2 . This way, the first run is legitimately fully incorporated into the second one and the t_{c_1} updates required for convergence during the first run may be considered the first t_{c_1} updates of the second run under this specific policy of presenting patterns to the algorithm. Combining the above two observations we see that by employing a first run with accuracy parameter ϵ_1 we force the algorithm with accuracy parameter $\epsilon_2 < \epsilon_1$ to have α_t decreased from a value ~ 1 to a value $\alpha_{t_{c_1}} < (1-\epsilon_1)^{-1}\gamma_d/R$ in the first t_{c_1} updates.

The above discussion suggests that we consider a decreasing sequence of parameters ϵ_n such that $\epsilon_{n+1} = \epsilon_n/\eta$ ($\eta > 1$) starting with $\epsilon_0 = 1/2$ and ending with the required accuracy ϵ and perform successive runs of PDM with accuracies ϵ_n until convergence in t_{c_n} updates is reached. According to our earlier discussion t_{c_n} includes the updates that led the algorithm to convergence in the current and all previous runs. Moreover, at the end of the run with parameter ϵ_n we will have ensured that $\alpha_{t_{c_n}} < (1 - \epsilon_n)^{-1}\gamma_d/R$. Therefore, $t_{c_{n+1}}$ satisfies $t_{c_{n+1}} \leq t_0$ or

$$t_{c_{n+1}} \leq t_{c_n} \left(\frac{1}{1 - \epsilon_n} \right)^{\eta/\epsilon_n} \left(1 + \frac{(1 - \epsilon_n)^2}{1 - 2\epsilon_n/\eta} \frac{R^2}{\gamma_d^2} t_{c_n}^{-1} \right)^{\eta/2\epsilon_n}.$$

This is obtained by substituting in (17) the values $\epsilon = \epsilon_{n+1} = \epsilon_n/\eta$, $N = t_{c_n}$ and $\alpha_N = (1 - \epsilon_n)^{-1}\gamma_d/R$ which is the least favorable choice for $\alpha_{t_{c_n}}$. Let us assume that $\epsilon_n \ll 1$ and set $t_{c_n} = \xi_n^{-1}R^2/\gamma_d^2$ with $\xi_n \ll 1$. Then, $1/(1 - \epsilon_n)^{\eta/\epsilon_n} \simeq e^\eta$ and

$$\left(1 + \frac{(1 - \epsilon_n)^2}{1 - 2\epsilon_n/\eta} \frac{R^2}{\gamma_d^2} t_{c_n}^{-1} \right)^{\eta/2\epsilon_n} \simeq (1 + \xi_n)^{\eta/2\epsilon_n} \simeq e^{\eta\xi_n/2\epsilon_n}.$$

For $\xi_n \simeq \epsilon_n$ the term above becomes approximately $e^{\eta/2}$ while for $\xi_n \ll \epsilon_n$ approaches 1. We see that under the assumption that PDM with accuracy parameter ϵ_n converges in a number of updates $\gg R^2/\gamma_d^2$ the ratio $t_{c_{n+1}}/t_{c_n}$ in the successive run scenario is rather tightly constrained. If, instead, our assumption is not satisfied then convergence of the algorithm is fast anyway. Notice, that the value of $t_{c_{n+1}}/t_{c_n}$ inferred from the bound of Theorem 2 is $\sim \eta(R/\gamma_d)^{(\eta-1)/\epsilon_n}$ which is extremely large. We conclude that PDM employing the successive run scenario (PDM-succ) potentially converges in a much smaller number of steps.

4 Efficient Implementation

To reduce the computational cost involved in running PDM, we extend the procedure of [14, 13] and construct a three-member nested sequence of reduced “active sets” of data points. As we cycle once through the full dataset, the (largest) first-level active set is formed from the points of the full dataset satisfying $\mathbf{a}_t \cdot \mathbf{y}_k \leq c_1(1 - \epsilon) \|\mathbf{a}_t\|^2/t$ with $c_1 = 2.2$. Analogously, the second-level active set is formed as we cycle once through the first-level active set from the points which satisfy $\mathbf{a}_t \cdot \mathbf{y}_k \leq c_2(1 - \epsilon) \|\mathbf{a}_t\|^2/t$ with $c_2 = 1.1$. The third-level active set comprises the points that satisfy $\mathbf{a}_t \cdot \mathbf{y}_k \leq (1 - \epsilon) \|\mathbf{a}_t\|^2/t$ as we cycle once through the second-level active set. The third-level active set is presented repetitively to the algorithm for N_{ep_3} mini-epochs. Then, the second-level active set is presented. During each round involving the second-level set, a new third-level set is constructed and a new cycle of N_{ep_3} passes begins. When a number of N_{ep_2} cycles involving the second-level set is reached the first-level set becomes active again leading to the population of a new second-level active set. By invoking the first-level set for the $(N_{ep_1} + 1)^{th}$ time, we trigger the loading

of the full dataset and the procedure starts all over again until no point is found misclassified among the ones comprising the full dataset. Of course, the N_{ep_1} , N_{ep_2} and N_{ep_3} rounds are not exhausted if no update takes place during a round. In all experiments we choose $N_{\text{ep}_1} = 9$, $N_{\text{ep}_2} = N_{\text{ep}_3} = 12$. In addition, every time we make use of the full dataset we actually employ a permuted instance of it. Evidently, the whole procedure amounts to a different way of sequentially presenting the patterns to the algorithm and does not affect the applicability of our theoretical analysis. A completely analogous procedure is followed for PFM.

An additional mechanism providing a substantial improvement of the computational efficiency is the one of performing multiple updates [14, 13] once a data point is presented to the algorithm. It is understood, of course, that in order for a multiple update to be compatible with our theoretical analysis it should be equivalent to a certain number of updates occurring as a result of repeatedly presenting to the algorithm the data point in question. For PDM when a pattern \mathbf{y}_k is found to satisfy condition (8) we perform $\lambda = [\mu_+] + 1$ updates at once. Here, μ_+ is the smallest non-negative root of the quadratic equation in the variable μ derivable from the relation $(t + \mu)\mathbf{a}_{t+\mu} \cdot \mathbf{y}_k - (1 - \epsilon)\|\mathbf{a}_{t+\mu}\|^2 = 0$ in which $\mathbf{a}_{t+\mu} \cdot \mathbf{y}_k = \mathbf{a}_t \cdot \mathbf{y}_k + \mu\|\mathbf{y}_k\|^2$ and $\|\mathbf{a}_{t+\mu}\|^2 = \|\mathbf{a}_t\|^2 + 2\mu\mathbf{a}_t \cdot \mathbf{y}_k + \mu^2\|\mathbf{y}_k\|^2$. Thus, we require that as a result of the multiple update the pattern violates the misclassification condition. Similarly, we perform multiple updates for PFM.

Finally, in the case of PDM (no successive runs) when we perform multiple updates we start doing so after the first full epoch. This way, we avoid the excessive growth of the length of the weight vector due to the contribution to the solution of many aligned patterns in the early stages of the algorithm which hinders the fast decrease of $\|\mathbf{a}_t\|/t$. Moreover, in this scenario when we select the first-level active set as we go through the full dataset for the first time (first full epoch) we found it useful to set $c_1 = c_2 = 1.1$ instead of $c_1 = 2.2$.

5 Experimental Evaluation

We compare PDM with several other large margin classifiers on the basis of their ability to achieve fast convergence to a certain approximation of the “optimal” hyperplane in the feature space where the patterns are linearly separable. For linearly separable data the feature space is the initial instance space whereas for inseparable data (which is the case here) a space extended by as many dimensions as the instances is considered where each instance is placed at a distance Δ from the origin in the corresponding dimension⁴ [4]. This extension generates a margin of at least Δ/\sqrt{m} . Moreover, its employment relies on the well-known equivalence between the hard margin optimization in the extended space and the soft margin optimization in the initial instance space with objective function

⁴ $\mathbf{y}_k = [\bar{\mathbf{y}}_k, l_k \Delta \delta_{1k}, \dots, l_k \Delta \delta_{mk}]$, where δ_{ij} is Kronecker’s δ and $\bar{\mathbf{y}}_k$ the projection of the k^{th} extended instance \mathbf{y}_k (multiplied by its label l_k) onto the initial instance space. The feature space mapping defined by the extension commutes with a possible augmentation (with parameter ρ) in which case $\bar{\mathbf{y}}_k = [l_k \bar{\mathbf{x}}_k, l_k \rho]$. Here $\bar{\mathbf{x}}_k$ represents the k^{th} data point.

$\|\mathbf{w}\|^2 + \Delta^{-2} \sum_i \xi_i^2$ involving the weight vector \mathbf{w} and the 2-norm of the slacks ξ_i . Of course, all algorithms are required to solve identical hard margin problems.

The datasets we used for training are: the Adult ($m = 32561$ instances, $n = 123$ attributes) and Web ($m = 49749, n = 300$) UCI datasets as compiled by Platt [15], the training set of the KDD04 Physics dataset ($m = 50000, n = 70$ after removing the 8 columns containing missing features) obtainable from <http://kodiak.cs.cornell.edu/kddcup/datasets.html>, the Real-sim ($m = 72309, n = 20958$), News20 ($m = 19996, n = 1355191$) and Webspam (unigram treatment with $m = 350000, n = 254$) datasets all available at <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets>, the multiclass Covertype UCI dataset ($m = 581012, n = 54$) and the full Reuters RCV1 dataset ($m = 804414, n = 47236$) obtainable from http://www.jmlr.org/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm. For the Covertype dataset we study the binary classification problem of the first class versus rest while for the RCV1 we consider both the binary text classification tasks of the C11 and CCAT classes versus rest. The Physics and Covertype datasets were rescaled by a multiplicative factor 0.001. The experiments were conducted on a 2.5 GHz Intel Core 2 Duo processor with 3 GB RAM running Windows Vista. Our codes written in C++ were compiled using the g++ compiler under Cygwin.

The parameter Δ of the extended space is chosen from the set $\{3, 1, 0.3, 0.1\}$ in such a way that it corresponds approximately to $R/10$ or $R/3$ depending on the size of the dataset such that the ratio γ_d/R does not become too small (given that the extension generates a margin of at least Δ/\sqrt{m}). More specifically, we have chosen $\Delta = 3$ for Covertype, $\Delta = 1$ for Adult, Web and Physics, $\Delta = 0.3$ for Webspam, C11 and CCAT and $\Delta = 0.1$ for Real-sim and News20. We also verified that smaller values of Δ do not lead to a significant decrease of the training error. For all datasets and for algorithms that introduce bias through augmentation the associated parameter ρ was set to the value $\rho = 1$.

We begin our experimental evaluation by comparing PDM with PFM. We run PDM with accuracy parameter $\epsilon = 0.01$ and subsequently PFM with the fixed margin $\beta = (1 - \epsilon)\gamma_d$ set to the value γ'_d of the directional margin achieved by PDM. This procedure is repeated using PDM-succ with step $\eta = 8$ (i.e., $\epsilon_0 = 0.5, \epsilon_1 = 0.0625, \epsilon_2 = \epsilon = 0.01$). Our results (the value of the directional margin γ'_d achieved, the number of required updates (upd) for convergence and the CPU time for training in seconds (s)) are presented in Table I. We see that PDM is considerably faster than PFM as far as training time is concerned in spite of the fact that PFM needs much less updates for convergence. The successive run scenario succeeds, in accordance with our expectations, in reducing the number of updates to the level of the updates needed by PFM in order to achieve the same value of γ'_d at the expense of an increased runtime. We believe that it is fair to say that PDM-succ with $\eta = 8$ has the overall performance of PFM without the defect of the need for a priori knowledge of the value of γ_d . We also notice that although the accuracy ϵ is set to the same value for both scenarios the margin achieved with successive runs is lower. This is an indication that PDM-succ obtains a better estimate of the maximum directional margin γ_d .

Table 1. Results of an experimental evaluation comparing the algorithms PDM and PDM-succ with PFM

data set	PDM $\epsilon = 0.01$			PFM		PDM-succ $\epsilon = 0.01$			PFM	
	$10^4\gamma'_d$	10^{-6}upd	s	10^{-6}upd	s	$10^4\gamma'_d$	10^{-6}upd	s	10^{-6}upd	s
Adult	84.57	27.43	3.7	10.70	7.3	84.46	9.312	5.3	9.367	6.6
Web	209.6	739.4	0.8	1.089	0.9	209.1	0.838	0.9	0.871	0.8
Physics	44.54	9.449	10.4	6.021	13.8	44.53	5.984	15.3	6.006	13.8
Real-sim	39.93	15.42	13.6	12.69	35.7	39.74	5.314	13.8	5.306	14.3
News20	91.90	2.403	27.4	1.060	55.6	91.68	0.814	47.7	0.813	43.7
Webspam	10.05	331.0	197.5	108.4	348.0	10.03	89.72	247.0	89.60	264.5
Covertype	47.51	189.7	86.6	68.86	156.0	47.48	66.03	146.1	64.41	142.5
C11	13.81	148.6	156.3	75.26	895.1	13.77	49.02	612.4	49.22	557.5
CCAT	9.279	307.7	310.6	151.2	1923.5	9.253	107.8	1389.8	107.8	1601.0

We also considered other large margin classifiers representing classes of algorithms such as perceptron-like algorithms, decomposition SVMs and linear SVMs with the additional requirement that the chosen algorithms need only specification of an accuracy parameter. From the class of perceptron-like algorithms we have chosen (aggressive) ROMMA which is much faster than ALMA in the light of the results presented in [9, 14]. Decomposition SVMs are represented by SVM^{light} [7] which, apart from being one of the fastest algorithms of this class, has the additional advantage of making very efficient use of memory, thereby making possible the training on very large datasets. Finally, from the more recent class of linear SVMs we have included in our study the dual coordinate descent (DCD) algorithm [8] and the margin perceptron with unlearning (MPU)⁵ [13]. We considered the DCD versions with 1-norm (DCD-L1) and 2-norm (DCD-L2) soft margin which for the same value of the accuracy parameter produce identical solutions if the penalty parameter is $C = \infty$ for DCD-L1 and $C = 1/(2\Delta^2)$ for DCD-L2. The source for SVM^{light} (version 6.02) is available at <http://svmlight.joachims.org> and for DCD at <http://www.csie.ntu.edu.tw/~cjlin/liblinear>. The absence of publicly available implementations for ROMMA necessitated the writing of our own code in C++ employing the mechanism of active sets proposed in [9] and incorporating a mechanism of permutations performed at the beginning of a full epoch. For MPU the implementation followed closely [13] with active set parameters $\bar{c} = 1.01$, $N_{ep_1} = N_{ep_2} = 5$, gap parameter $\delta b = 3R^2$ and early stopping.

The experimental results (margin values achieved and training runtimes) involving the above algorithms with the accuracy parameter set to 0.01 for all of them are summarized in Table 2. Notice that for SVM^{light} we give the geometric margin γ' instead of the directional one γ'_d because SVM^{light} does not introduce bias through augmentation. For the rest of the algorithms considered, including PDM and PFM, the geometric margin γ' achieved is not listed in the tables since

⁵ MPU uses dual variables but is not formulated as an optimization. It is a perceptron incorporating a mechanism of reduction of possible contributions from “very-well classified” patterns to the weight vector which is an essential ingredient of SVMs.

Table 2. Results of experiments with ROMMA, SVM^{light}, DCD-L1, DCD-L2 and MPU algorithms. The accuracy parameter for all algorithms is set to 0.01.

data set	ROMMA		SVM ^{light}		DCD-L1		DCD-L2		MPU	
	$10^4\gamma'_d$	s	$10^4\gamma'$	s	$10^4\gamma'_d$	s	s	$10^4\gamma'_d$	s	
Adult	84.66	275.8	84.90	414.2	84.95	0.6	0.5	84.61	0.8	
Web	209.6	52.6	209.4	40.3	209.5	0.7	0.6	209.5	0.3	
Physics	44.57	117.7	44.60	2341.8	44.57	22.5	20.0	44.62	4.9	
Real-sim	39.89	1318.8	39.80	146.5	39.81	6.4	5.6	39.78	3.3	
News20	92.01	4754.0	91.95	113.8	92.17	48.1	47.1	91.62	15.8	
Webspam	10.06	39760.6	10.07	29219.4	10.08	37.5	33.0	10.06	28.2	
Covertype	47.54	43282.0	47.73	48460.3	47.71	18.1	15.0	47.67	18.7	
C11	13.82	146529.2	13.82	20127.8	13.83	30.7	27.2	13.79	20.2	
CCAT	9.290	298159.4	9.291	83302.4	9.303	51.9	46.2	9.264	36.1	

it is very close to the directional margin γ'_d if the augmentation parameter ρ is set to the value $\rho = 1$. Moreover, for DCD-L1 and DCD-L2 the margin values coincide as we pointed out earlier. From Table 2 it is apparent that ROMMA and SVM^{light} are orders of magnitude slower than DCD and MPU. Comparing the results of Table 1 with those of Table 2 we see that PDM is orders of magnitude faster than ROMMA which is its natural competitor since they both belong to the class of perceptron-like algorithms. PDM is also much faster than SVM^{light} but statistically a few times slower than DCD, especially for the larger datasets. Moreover, PDM is a few times slower than MPU for all datasets. Finally, we observe that the accuracy achieved by PDM is, in general, closer to the before-run accuracy 0.01 since in most cases PDM obtains lower margin values. This indicates that PDM succeeds in obtaining a better estimate of the maximum margin than the remaining algorithms with the possible exception of MPU.

Before we conclude our comparative study it is fair to point out that PDM is not the fastest perceptron-like large margin classifier. From the results of [14] the fastest algorithm of this class is the margintron which has strong before-run guarantees and a very good after-run estimate of the achieved accuracy through [5]. However, its drawback is that an approximate knowledge of the value of γ_d (preferably an upper bound) is required in order to fix the parameter controlling the margin threshold. Although there is a procedure to obtain this information, taking all the facts into account the employment of PDM seems preferable.

6 Conclusions

We introduced the perceptron with dynamic margin (PDM), a new approximate maximum margin classifier employing the classical perceptron update, demonstrated its convergence in a finite number of steps and derived an upper bound on them. PDM uses the required accuracy as the only input parameter. Moreover, it is a strictly online algorithm in the sense that it decides whether to perform

an update taking into account only its current state and irrespective of whether the pattern presented to it has been encountered before in the process of cycling repeatedly through the dataset. This certainly does not hold for linear SVMs. Our experimental results indicate that PDM is the fastest large margin classifier enjoying the above two very desirable properties.

References

- [1] Blum, A.: Lectures on machine learning theory. Carnegie Mellon University, USA, <http://www.cs.cmu.edu/~avrim/ML09/lect0126.pdf>
- [2] Cristianini, N., Shawe-Taylor, J.: An introduction to support vector machines. Cambridge. Cambridge University Press, Cambridge (2000)
- [3] Duda, R.O., Hart, P.E.: Pattern classification and scene analysis. Wiley, Chichester (1973)
- [4] Freund, Y., Shapire, R.E.: Large margin classification using the perceptron algorithm. *Machine Learning* 37(3), 277–296 (1999)
- [5] Gentile, C.: A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research* 2, 213–242 (2001)
- [6] Joachims, T.: Making large-scale SVM learning practical. In: *Advances in Kernel Methods-Support Vector Learning*. MIT Press, Cambridge (1999)
- [7] Joachims, T.: Training linear SVMs in linear time. In: *KDD*, pp. 217–226 (2006)
- [8] Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S.S., Sundararajan, S.: A dual coordinate descent method for large-scale linear SVM. In: *ICML*, pp. 408–415 (2008)
- [9] Ishibashi, K., Hatano, K., Takeda, M.: Online learning of approximate maximum p-norm margin classifiers with bias. In: *COLT*, pp. 69–80 (2008)
- [10] Krauth, W., Mézard, M.: Learning algorithms with optimal stability in neural networks. *Journal of Physics A20*, L745–L752 (1987)
- [11] Li, Y., Long, P.: The relaxed online maximum margin algorithm. *Machine Learning* 46(1-3), 361–387 (2002)
- [12] Novikoff, A.B.J.: On convergence proofs on perceptrons. In: *Proc. Symp. Math. Theory Automata*, vol. 12, pp. 615–622 (1962)
- [13] Panagiotakopoulos, C., Tsampouka, P.: The margin perceptron with unlearning. In: *ICML*, pp. 855–862 (2010)
- [14] Panagiotakopoulos, C., Tsampouka, P.: The margintriton: A generalized perceptron with margin. *IEEE Transactions on Neural Networks* 22(3), 395–407 (2011)
- [15] Platt, J.C.: Sequential minimal optimization: A fast algorithm for training support vector machines. Microsoft Res. Redmond WA, Tech. Rep. MSR-TR-98-14 (1998)
- [16] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65(6), 386–408 (1958)
- [17] Tsampouka, P., Shawe-Taylor, J.: Perceptron-like large margin classifiers. Tech. Rep., ECS, University of Southampton, UK (2005), Obtainable from, <http://eprints.ecs.soton.ac.uk/10657>
- [18] Tsampouka, P., Shawe-Taylor, J.: Analysis of generic perceptron-like large margin classifiers. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) *ECML 2005. LNCS (LNAI)*, vol. 3720, pp. 750–758. Springer, Heidelberg (2005)
- [19] Tsampouka, P., Shawe-Taylor, J.: Constant rate approximate maximum margin algorithms. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) *ECML 2006. LNCS (LNAI)*, vol. 4212, pp. 437–448. Springer, Heidelberg (2006)
- [20] Tsampouka, P., Shawe-Taylor, J.: Approximate maximum margin algorithms with rules controlled by the number of mistakes. In: *ICML*, pp. 903–910 (2007)
- [21] Vapnik, V.: Statistical learning theory. Wiley, Chichester (1998)

Combining Initial Segments of Lists

Manfred K. Warmuth^{1,*}, Wouter M. Koolen^{2,3, **}, and David P. Helmbold¹

¹ Department of Computer Science, UC Santa Cruz
manfred@cse.ucsc.edu

² Department of Computer Science, Royal Holloway, University of London
³ Centrum Wiskunde en Informatica, Amsterdam

wmkoolen@cwi.nl

Abstract. We propose a new way to build a combined list from K base lists, each containing N items. A combined list consists of top segments of various sizes from each base list so that the total size of all top segments equals N . A sequence of item requests is processed and the goal is to minimize the total number of misses. That is, we seek to build a combined list that contains all the frequently requested items. We first consider the special case of disjoint base lists. There, we design an efficient algorithm that computes the best combined list for a given sequence of requests. In addition, we develop a randomized online algorithm whose expected number of misses is close to that of the best combined list chosen in hindsight. We prove lower bounds that show that the expected number of misses of our randomized algorithm is close to the optimum. In the presence of duplicate items, we show that computing the best combined list is NP-hard. We show that our algorithms still apply to a linearized notion of loss in this case. We expect that this new way of aggregating lists will find many ranking applications.

1 Introduction

We propose a new approach for aggregating ranked lists. Assume we have K lists of N items each, as illustrated in Figure 1. Our comparator is the best combined list of size N that is composed of the tops of the K lists. A combined list might take the top 20% of list 1, the top 0% of list 2, the top 60% of list 3, and so forth. Note that the contents of the base lists might change over time and there are exponentially many (roughly N^K) combined lists altogether.

We seek efficient online algorithms that construct such combined lists on the fly. In each trial the following happens: First, the current contents of all base lists are provided to the algorithm. Then the algorithm assembles (either deterministically or randomly) its combined list. After that some item is requested. If it is not in the chosen combined list, then the algorithm incurs a miss (one unit of loss). Some or all of the base lists might also miss the requested item and might update themselves accordingly for the next trial.

* Supported by NSF grant IIS-0917397 and a Google gift grant.

** Supported by a Rubicon grant from the Netherlands Organisation for Scientific Research (NWO).

The goal of the algorithm is to endure small additional loss (regret) over the best combined list chosen in hindsight once the entire request sequence and the time-varying contents of the base lists are known. We seek efficient online algorithms that implicitly maintain some information about all roughly N^K combined lists and can efficiently choose or sample a combined list from this information. As we shall see, probabilistic algorithms will have the advantage.

We claim that this setup has many applications. For example we can use our method to combine the listings produced by different search engines. Here a first goal is to combine the tops of the base lists for the purpose of maximizing hits. However in the case of search engines, we are also concerned with accumulating hits at the top of our chosen combined list and this aspect is not yet modeled by our approach. We briefly discuss such extensions in the conclusion Section 7.

Another important application is caching. In this case each list is the ranked list of items selected by a different caching strategy. We assume that all items have the same size and exactly N fit into the fast memory. The algorithm can simulate K caching strategies as “virtual caches” and then combines these virtual caches to form one “real cache”. The virtual caches only record a few bytes of meta-data about each item in their cache: ID, link to the data, and calculated priority. Object data is only kept for the N items of the real combined cache. The memory cost for maintaining the virtual caches is negligible. The real combined cache can be updated as the virtual caching strategies change their item lists and the algorithm observes the performance of its combined cache.

Previous Work. Methods for building a real cache by combining a number of virtually maintained caching strategies using exponential weights were explored in [GWBA02]. The employed heuristics performed well experimentally. However no performance bounds were proven. The idea for building a real cache by combining the tops of two virtual lists was first developed in [MM03, MM04]. In this case the first list contained the most recently requested items and the second contained those that were requested more than once. The goal for building a combined list was to optimally balance recency and frequency information. A deterministic heuristic was developed for adjusting the top portion taken from each list. In this paper we prove a lower bound for any deterministic algorithm that shows that any such algorithm can be forced to have loss at least $K B$ where B is the loss of the optimal combined list chosen in hindsight. We also give a randomized online algorithm whose expected loss is at most B plus an additional sublinear regret and show that the regret of this algorithm is optimal within a factor of $\min(\sqrt{K}, \sqrt{\ln(N/K)})$.

This paper was motivated by our previous work on combining caching heuristics [GWBA02]. In general, there are always two approaches to a rich problem

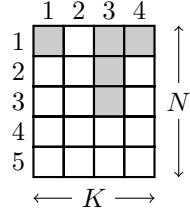


Fig. 1. We depict one combined list formed by taking the tops from $K = 4$ lists. Note that all lists have size $N = 5$ and the combined list has the same size. The list shown is $c = (1, 0, 3, 1)$.

setting. Either we can restrict the setting enough so that theoretical bounds are obtainable, or we can explore heuristics without provable guarantees that perform well in practice. Ideally the two approaches will meet eventually. In this paper we focus on algorithms for which we can prove regret bounds and we believe we made significant progress towards addressing practically interesting problems.

Aggregating Experts. One of the main metaphors of online learning has been the aggregation of many simple “experts” [LW94, Vov98, FS97]. In our application, each combined list serves as an expert. The online algorithm maintains its uncertainty over all experts as a mixture distribution and predicts based on this mixture. In more detail, the probability of a combined list \mathbf{c} is proportional to $\beta^{\mathbf{M}(\mathbf{c})}$, where $\mathbf{M}(\mathbf{c})$ is the number of misses of list \mathbf{c} and the update factor β lies in $[0, 1)$.

There are exponentially many mixture weights (one per expert). However, as we shall see later, we still manage to obtain very efficient algorithms by manipulating the weights implicitly. We first discuss how to obtain a combined list from a mixture. One could consider thresholding the mean, using the median (this is new but only works for 2 lists) or sampling.

The Weighted Average (Mean) Does Not Correspond to a Combined List. A deterministic algorithm would naturally predict with the weighted majority (mean) of the mixture. That is, an item is in the chosen combined list if the total weight of all combined lists in the mixture that contain this item is at least 50%. Unfortunately, the weighted majority does not always correspond to a list of size N : For $N = 4$ and $K = 3$, consider the case displayed in Figure 2, where we mix the 3 combined lists that contain the top halves of 2 out of the 3 lists; if the mixture is uniform on these 3 combined lists, then all items in the top halves of each of the 3 lists have total weight $2/3$, which is 6 elements altogether and this is more than $N = 4$.

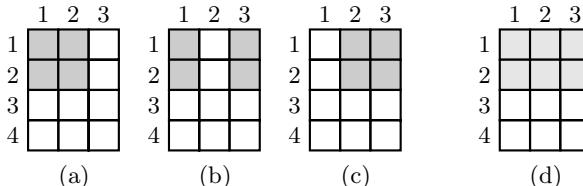


Fig. 2. Counterexample against obtaining a combined list by mean thresholding. Consider the combined lists (a), (b) and (c), which each use 4 items with weight 1. The uniform mixture over (a), (b) and (c) (displayed as (d)), sports 6 elements with weight $2/3$ each. By including the items whose mean weight exceeds some threshold, we end up with either 0 or 6 elements, but not the desired 4.

Deterministic Algorithm for the Case of Two Lists Based on the Median. In the case of $K = 2$ (i.e. two lists of size N), there are $N + 1$ combined lists $\mathbf{c}_0, \dots, \mathbf{c}_N$, where $\mathbf{c}_i = (i, N - i)$ contains the i top elements from the first

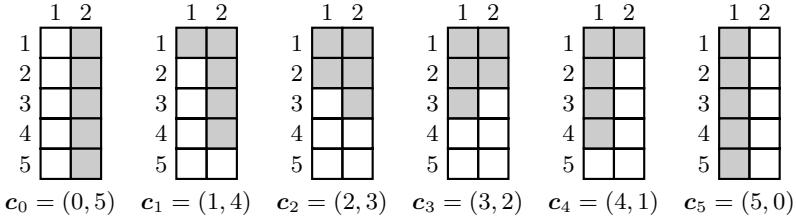


Fig. 3. The 6 combined lists of size $N = 5$ from $K = 2$ base lists

list and the $N - i$ top elements from the second list. These combined lists are displayed in Figure 3.

For $K = 2$ disjoint lists there is a simple algorithm producing a single combined list of the right size N based on the median that circumvents the problem with the mean discussed in the previous section. It maintains the same exponential weight discussed before but chooses a combined list \mathbf{c}_i s.t. the total weights of the lists $\mathbf{c}_0, \dots, \mathbf{c}_{i-1}$ and the lists $\mathbf{c}_{i+1}, \dots, \mathbf{c}_N$ are both at most half. In other words the algorithm picks the combined list from $\langle \mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_N \rangle$ with the *median* weight and we call this deterministic algorithm the *Weighted Median algorithm*. Whenever a miss occurs, then at least half of the total weight is multiplied by β : If the item was on the first list and was missed by the median list \mathbf{c}_i , then at least $\mathbf{c}_0, \dots, \mathbf{c}_i$ are multiplied by β , which contain at least half of the total weight. Similarly, if the item was on the second list and missed by \mathbf{c}_i , then at least $\mathbf{c}_i, \dots, \mathbf{c}_N$ are multiplied by β , which is again at least half of the total. The case when the item did not appear on either base list is trivial.

Since at least half of the total weight is multiplied by β , an analysis paralleling the analysis of the deterministic Weighted Majority algorithm [LW94] gives the following loss bound after tuning β based on N and the *budget* B , i.e. the loss of the best combined list in hindsight:

$$2B + O\left(\sqrt{B \ln(N+1)} + \ln(N+1)\right).$$

There are two problems with this deterministic algorithm: It has the factor 2 in front of the budget and we don't know how to generalize it to more than 2 lists. This is not surprising because we show in this paper that any deterministic algorithm can be forced to incur loss $K B$, where B is the loss of the best. Nevertheless, algorithms based on this approach perform well experimentally [Sca07] and beat the previous deterministic heuristics developed in [MM03, MM04].

Probabilistic Algorithms Based on Sampling. Our approach is to apply the Hedge algorithm [FS97] to the exponentially many combined lists. It uses the same exponential weights that are proportional to $\beta^{\mathbf{M}(\mathbf{c})}$ for list \mathbf{c} but now we simply pick a combined list at random according to the mixture coefficients on all the combined lists. This ensures that the randomly chosen combined list is of the right size and hence circumvents the fact that the mixture does not represent a

single combined list. The expected loss of the mixture is the mixture of the losses of the individual combined lists. When β is properly tuned as a function of the budget B and the number of combined lists, then the probabilistic algorithms achieve expected loss $B + O(\sqrt{BK \ln N})$. Note that now the factor in front of the budget B is one (whereas any deterministic algorithm can be forced to have loss at least KB , see Section 5). The caveat of the probabilistic algorithms is that the list they predict with may change significantly between trials and in applications where there is a cost for changing the prediction, such algorithms are not useful. We discuss this issue again in the conclusion Section 7.

Hedge vs Follow the Perturbed Leader. The two main flavors of efficient online algorithms for dealing with a linear loss are Hedge [FS97] and Follow the Perturbed Leader [KV05]. Hedge based algorithms usually have slightly better loss bounds, whereas FPL-type algorithms typically have computational advantages. In this paper, completely contrary to those general rules, we present a Hedge-based algorithm that is fundamentally *faster* for our problem than the best-known FPL algorithm. The reason for this anomaly is that the computations for Hedge can be accelerated using the Fast Fourier Transform, whereas only a slower acceleration is known for finding the best combined list.

Outline of the Paper. After setting the stage formally in Section 2, we begin by sketching the batch algorithm in Section 3. We then give our main randomized algorithm in Section 4 whose (expected) regret can be bounded by $O(\sqrt{BK \log N})$, where B is the loss budget of the best combined list. It simulates the Hedge algorithm on the exponentially many combined lists. We first assume that all base lists are disjoint, i.e. the requested item appears on at most one base list. This assumption lets us express the 0-1 loss/miss of a combined list as a sum over the initial segments of the base lists. The straightforward implementation requires $O(KN^2)$ time per trial. However we found a way to speed up the algorithm using Fast Fourier Transform for an improved time of $O(KN \ln N)$ per trial.

A number of lower bounds are given in Section 5. Any deterministic algorithm can be made to suffer loss at least KB , i.e. such algorithms cannot achieve sublinear regret. We also prove that any probabilistic algorithm can be forced to have regret at least $\Omega(\max(\sqrt{B \ln N}, \sqrt{B K}))$. Thus when either K is constant or $N = \Theta(K)$, the regret of our probabilistic algorithm is optimal within a constant factor.

In Section 6 we then address the case where duplicates are allowed between base lists. In this case we show that the question of whether there is a list with no misses is NP-hard. We then address the same problem with a surrogate loss. We “linearize” the loss much in the same way NP-completeness was avoided when learning disjunctions (by switching from 0-1 loss to attribute loss). In short, if the requested item occurs 5 times on the lists, then a combined list that hits this item 2 times now has loss 3. We can show that our algorithm now has a regret bound of $O(\sqrt{BK^2 \log N})$ for the linearized loss. Note the additional factor of K which is due to the fact the range of the loss per trial is now $[0, K]$. We also

discuss an alternate method for solving the problem with duplicates based on the online shortest path problem and using Component Hedge [KWK10]. This algorithm achieves regret $O(\sqrt{BK \log N})$ for the problem with duplicates (i.e. no additional range factor). However we do not know how to bound the running time for the iterative projection computation employed by the algorithm. We conclude with a number of open problems in the final section.

2 Setting

Recall that we have K base lists of N slots each. Until Section 6 we assume that each item appears at most once on all list. For $k \leq K$, we identify a (partial) *combined list* with a vector of counts $\mathbf{c} = (c_1, \dots, c_k)$, where c_i denotes the number of items taken from the top of base list i . We denote the set of combined lists using n elements from k base lists by

$$\mathbb{C}_{n,k} := \{\mathbf{c} \in \{0, \dots, n\}^k \mid \sum_{i=1}^k c_i = n\}.$$

We also think about $\mathbb{C}_{n,k}$ as the set of paths from $(0, 0)$ to (n, k) through the graph shown in Figure 4. Representational issues aside, the number of combined lists is rather large:

$$\ln |\mathbb{C}_{N,K}| = \ln \binom{N+K-1}{K-1} \leq (K-1) \ln \frac{(N+K-1) e}{K-1} = O\left(K \ln \frac{N}{K}\right) \quad (1)$$

The right equality holds under our assumption that $N \geq K$. To appreciate the sheer number of lists, consider for example that $|\mathbb{C}_{10,20}| \approx 10^7$, while $|\mathbb{C}_{100,20}| \approx 4.9 \cdot 10^{21}$.

Different trials can have different lists and requested items. What is important is the list and position containing the requested item. A single trial is summarized by a $(N+1) \times K$ dimensional *miss matrix* \mathbf{M} , where for $0 \leq c \leq N$ and $1 \leq k \leq K$, the entry $M_{c,k}$ is one if the requested item appears on base list k in a position strictly greater than c and zero otherwise. The sum $\mathbf{M}(\mathbf{c}) := \sum_{i=1}^k M_{c_i,i}$ is zero when combined list \mathbf{c} contains the request item, and one when \mathbf{c} misses the requested item on the disjoint lists. We often sum miss matrices over trials: \mathbf{M}^t denotes the miss matrix for trial t and $\mathbf{M}^{<t}$ is the cumulative miss matrix before trial t , i.e. $M_{c,k}^{<t} := \sum_{s < t} M_{c,s}^s$. Therefore $\mathbf{M}^{<t}(\mathbf{c}) = \sum_{i=1}^k M_{c_i,i}^{<t}$ is the total number of misses made by the combined list \mathbf{c} in the first $t-1$ trials.

We allow items to occur on multiple lists in Section 6. There, when \mathbf{M} is a single trial miss matrix, $\mathbf{M}(\mathbf{c})$ can be greater than one. For example, if the requested item occurs on 5 different base lists and \mathbf{c} has it twice then $\mathbf{M}(\mathbf{c}) = 3$.

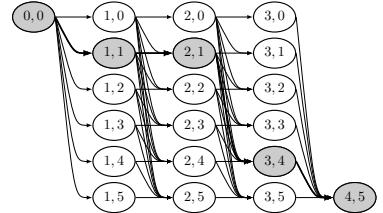


Fig. 4. Dynamic programming diagram for $K = 4, N = 5$. There is a 1-1 correspondence between $(0, 0) - (4, 5)$ paths in this graph and combined lists. The marked path corresponds to the combined lists shown in Figure 1. In general, combined list $\mathbf{c} = (c_1, \dots, c_K)$ corresponds to path $(0, 0) - (1, c_1) - (2, c_1 + c_2) - \dots - (k, \sum_{i=1}^k c_i) - \dots - (K, N)$.

3 Batch Algorithm

Let $\mathbf{M} = \sum_{t=1}^T \mathbf{M}^t$ denote the cumulative miss matrix after T trials. The hindsight-optimal list \mathbf{c}^* and its loss B are given by

$$\mathbf{c}^* := \underset{\mathbf{c} \in \mathbb{C}_{N,K}}{\operatorname{argmin}} \mathbf{M}(\mathbf{c}) \quad B := \mathbf{M}(\mathbf{c}^*) = \min_{\mathbf{c} \in \mathbb{C}_{N,K}} \mathbf{M}(\mathbf{c}).$$

Brute-force evaluation of the minimum is intractable, viz (II). But we can exploit the structure of $\mathbb{C}_{N,K}$ and $\mathbf{M}(\mathbf{c})$ to compute \mathbf{c}^* and B efficiently. Let $B_{n,k}$ denote the loss of the best combined list with n elements from the first k base lists:

$$B_{n,k} := \min_{\mathbf{c} \in \mathbb{C}_{n,k}} \mathbf{M}(\mathbf{c}).$$

Hence $B = B_{N,K}$. Now observe that $B_{*,*}$ satisfies the recurrence for $0 \leq n \leq N$:

$$B_{n,1} = M_{n,1} \quad \text{and} \quad B_{n,k} = \min_{0 \leq c \leq n} B_{n-c,k-1} + M_{c,k}, \quad \text{for } 1 < k \leq K.$$

By straightforward tabulation, the loss $B = B_{N,K}$ of the best combined list can be computed in time $O(KN^2)$. Interestingly, we can tabulate even faster. The column $B_{*,k}$ is the infimal convolution¹ of $B_{*,k-1}$ and $\mathbf{M}_{*,k}$. The best-known algorithm for general infimal convolution is $O(\frac{N^2}{\ln N})$ due to [BCD⁺06]. In our setting, both $B_{*,k-1}$ and $M_{*,k}$ are non-increasing. However it is an open problem whether these special properties lead to an improved time bound. Once $B_{*,*}$ is known, it is easy to recover the optimal combined list \mathbf{c}^* in $O(NK)$ time.

The above dynamic programming algorithm immediately leads to an online algorithm, called Follow the Perturbed Leader (FPL) [KV05], which has small regret compared to the best list chosen in hindsight. At trial t , FPL adds a random perturbation matrix to $\mathbf{M}^{<t}$ and chooses the best combined list with respect to that perturbed miss matrix. FPL has slightly weaker regret bounds [HP05] than Hedge, but is usually faster. Surprisingly we show in the next section that for combined list the Hedge algorithm is actually faster: it requires $O(KN \ln N)$ time instead of $O(\frac{KN^2}{\ln N})$ for FPL.

4 The Randomized Online Algorithm Based on Sampling

In this section we develop an efficient randomized online algorithm and prove that its loss is not much larger than B , the loss of the best combined list chosen in hindsight. The algorithm is the well-known Hedge algorithm [FS97] where the combined list function as the experts. The algorithm (implicitly) maintains weights proportional to $\beta^{\mathbf{M}(\mathbf{c})}$ for each combined list \mathbf{c} . There are two versions of the Hedge algorithm. The first one predicts with the mixture vector over the experts and incurs loss equal to dot product between the mixture vector times the loss vector. Since the mixture vector is exponential in size we have to use

¹ Aka $(\min, +)$ convolution or min-convolution or inf-convolution or epigraphical sum.

the second version. Like the Randomized Weighted Majority algorithm [LW94], this version outputs an expert drawn at random from the mixture. The loss is the loss of the drawn combined list and the goal is to bound the expected loss of the algorithm in relation to the loss of the best list chosen in hindsight.

Our contribution lies in the efficient implementation of the sampling version of the Hedge algorithm for combined lists. Following [TW03], we crucially use the fact that in our application, the loss $\mathbf{M}(\mathbf{c})$ of a combined list \mathbf{c} decomposes into a sum: $\mathbf{M}(\mathbf{c}) = \sum_{k=1}^K M_{c_k, k}$. Thus the weight of \mathbf{c} is proportional to the product $\prod_{k=1}^K \beta^{M_{c_k, k}}$. This property of the weights allows us to efficiently sample a combined list according to the mixture prescribed by the exponential weights of Hedge. The computation is based on dynamic programming, similar to the batch algorithm, but it is faster: $O(KN \ln N)$ instead of $O(\frac{KN^2}{\ln N})$ per trial.

Before showing this speedup, recall that the expected regret of the Hedge algorithm after tuning the learning rate η [FS97] is bounded by $B + \sqrt{2B \ln E} + \ln E$ where E is the number of experts with loss range $[0, 1]$ per trial and B is the loss budget of the best expert. In our application, $\ln E$ is $O(K \ln \frac{N}{K})$ by (II), giving us the following regret bound for our algorithm:

$$B + O\left(\sqrt{BK \ln \frac{N}{K}} + K \ln \frac{N}{K}\right). \quad (2)$$

Note that the loss of a combined list lies in $\{0, 1\}$ since we assumed that the base lists are disjoint.

We now give the efficient implementation of the sampling. Let $\mathbf{M} = \mathbf{M}^{<t}$ denote the cumulative miss matrix before trial t . In trial t , we need to efficiently sample combined list $\mathbf{c} \in \mathbb{C}_{N, K}$ with probability proportional to $\prod_{k=1}^K \beta^{M_{c_k, k}}$. We first compute the *normalization* $Z_{N, K}$, which is defined for all (partial) combined lists as follows

$$Z_{n, k} := \sum_{\mathbf{c} \in \mathbb{C}_{n, k}} \beta^{\mathbf{M}(\mathbf{c})}.$$

To efficiently compute $Z_{\star, \star}$, we again derive a recurrence. For all $0 \leq n \leq N$,

$$Z_{n, 1} = \beta^{M_{n, 1}} \quad \text{and} \quad Z_{n, k} = \sum_{c_k=0}^N Z_{n-c_k, k-1} \beta^{M_{c_k, k}}, \quad \text{for } 1 < k \leq K.$$

Even more compactly, we have

$$Z_{\star, k} = Z_{\star, k-1} * \beta^{\mathbf{M}_{\star, k}} \quad \text{for } 1 < k \leq K$$

where $*$ denotes real discrete convolution, i.e. $(x * y)_n = \sum_i x_{n-i} y_i$. Since the convolution of two vectors of length N each can be performed in time $O(N \ln N)$ using the Fast Fourier Transform [CT65], we can tabulate $Z_{\star, \star}$ in time $O(KN \ln N)$.²

² A similar approach was used in [KM05] to obtain a similar speedup in the completely different context of computing the stochastic complexity of the Multinomial model.

Sampling a list from $\mathbf{c} \in \mathbb{C}_{n,k}$ with probability proportional to $\beta^{\mathbf{M}(\mathbf{c})}$ is now done as follows: First draw the last element $c_k = j$ with probability equal to $Z_{n-j,k-1} \beta^{M_{j,k}} / Z_{n,k}$. Then recursively sample the initial part of the list from $\mathbb{C}_{n-j,k-1}$. The probability of drawing the full $\mathbf{c} \in \mathbb{C}_{N,K}$ telescopes to $\beta^{\mathbf{M}(\mathbf{c})} / Z_{N,K}$ as desired. Once we have $Z_{\star,\star}$ tabulated, the sampling itself takes $O(KN)$ time.

An analogous approach can be used to compute the expected loss of Hedge if that is of interest, e.g. for external model selection.

We already mentioned in Section 2 that it is possible to identify combined lists with paths through the specific graph shown in Figure 4. Therefore any algorithm that has small regret compared the best path chosen in hindsight is a competitor to our algorithm. The online shortest path problem has received considerable attention both in the full-information and in the bandit setting [TW03, KV05, AHR08, CBL09, KWK10]. However, for that problem it is assumed that the adversary can control the loss of each individual edge and as a result any algorithm requires an update time that is at least on the order of the number of edges. This is not the case in our setting. We have $O(KN^2)$ edges in our graph, and yet we achieve update time $O(KN \ln N)$. How is this possible? First note that a miss matrix \mathbf{M} has only KN parameters. Put another way, the losses of the edges are partitioned into equivalence classes of size $O(N)$ and all edges of the same class always have the same loss:

$$\forall 0 \leq n, n' \leq N-i, 1 \leq k < K : (n, k) \rightarrow (n+i, k+1) \sim (n', k) \rightarrow (n'+i, k+1).$$

This feature of our problem allows us to use convolutions and obtain update time that is lower than the number of edges in the shortest path formulation.

5 Lower Bounds

In the noise-free case (there is a combined list with no misses), the minimax regret for deterministic algorithms was proven to be $\Omega(K \ln N)$ with the first author's students in a class project [SS07]. Here we focus on the noisy case. We begin with a simple adversary argument against any deterministic algorithm and then turn to lower bounds for randomized algorithms.

We first show a simple lower bound of KB against any deterministic algorithm, where $B > 0$ is the number of misses of the best combined list in hindsight. We assume that B is known to both the algorithm and adversary and $N \geq K$. As illustrated by Figure 5, the adversary tags the following K items as “special”: item $N-K+2$ from the 1st list and the top items from the remaining $K-1$ lists. Any combined list of size N must miss at least one of these K special items because to contain them all would require a list of size $N-K+2+K-1 = N+1$. In each trial the adversary simply hits one of the special items missed by the combined list produced by the deterministic algorithm. Since combined lists have size N , at least one special item will be missed in each trial.

	1	2	3	4
1				
2				
3				
4				
5				

Fig. 5. For $N = 5, K = 4$ the special items are put in the marked positions. One must be missed by any combined list of size N .

In T trials, the algorithm incurs T misses, but the number of misses of the best combined list is at most $\frac{T}{K}$. The reason is that the best combined list leaves out the special item with the lowest number of hits and the lowest number is at most $\frac{T}{K}$. So if the best has loss B then any deterministic algorithm can be forced to have loss at least KB , and thus regret at least $B(K - 1)$.

We now build techniques for proving our lower bound against any randomized algorithm in stages. The basic component of our lower bound constructions will be the standard 2-expert game. The game is parametrized by a loss budget B . There are two players, the Algorithm and the Environment, and the game proceeds in rounds. At the beginning of each round, the Algorithm chooses its prediction, which is a pair (w_1, w_2) with $w_1, w_2 \geq 0$ and $w_1 + w_2 = 1$. The Environment then chooses one of two possible outcomes: expert 1 makes a mistake, or expert 2 makes a mistake. If expert 1 makes a mistake, the Algorithm incurs loss w_1 ; if expert 2 makes a mistake, the Algorithm incurs loss w_2 . The game ends when one expert has made at least $B + 1$ mistakes. Let $g_2(B)$ be the largest total loss the Environment can force the Algorithm to incur in this game. It can be shown based on results in [AWY08] that for any integer budget B

$$g_2(B) \geq B + \sqrt{\frac{B}{\pi}}.$$

We now prove a lower bound on the regret in the $K = 2$ list case of

$$B + \sqrt{\frac{B \log_2(N + 1)}{\pi}}$$

against any randomized algorithm A . We do this by constructing a trial sequence using the above 2-expert case as a building block on which A incurs at least this much loss while there is at least one combined list with loss at most B . For the sake of simplicity of the presentation we assume that $N = 2^S - 1$ for some integer S that divides the budget B .

The game consists of $S = \log_2(N + 1)$ stages, each stage using up a proportion B/S of the loss budget and causing a loss of at least $g_2(B/S)$ to the algorithm. Therefore, the total loss of the algorithm will be at least the desired result

$$S \left(\frac{B}{S} + \sqrt{\frac{B}{S\pi}} \right) = B + \sqrt{\frac{BS}{\pi}}.$$

We have two lists, each with N elements. During stage 1, we access only two elements, the middle elements (those in position $(N + 1)/2$ on the first and position $(N + 1)/2$ on the second list). Notice that a deterministic prediction can include at most one of the middle elements. Intuitively, stage 1 will decide whether it is better to include the middle element from list 1 or from list 2. This is done using the 2-expert game in a manner described below.

Suppose we decided to include the middle element from list 1. This means we have removed from consideration the first half of elements of list 1 (they are included) and the last half of elements of list 2 (they are not included). During

stage 2 we similarly decide between the middle element of the second half of list 1, and the middle element of the first half of list 2. Suppose we now decide in favor of list 2. This means that we have decided to include at least $(N+1)/2$ elements from list 1, at least $(N+1)/4$ elements from list 2, and $(N-3)/4$ elements are yet to be decided. We continue this process, halving the remaining range at each stage, until only one good combined list remains. This argument proves the following lower bound:

Theorem 1. *Let $N+1$ be a power of 2 and B be a budget that is divisible by $\log_2(N+1)$. Then for any randomized online algorithm A for the $K=2$ list problem there is a sequence of request on which A incurs at least loss*

$$B + \sqrt{\frac{B \log_2(N+1)}{\pi}}$$

but there is at least one combined list with at most B misses.

Note that the lower bound construction uses the same base lists in each trial and the base lists are disjoint.

It is fairly easy to see that the above $\Omega(\sqrt{B \ln N})$ expected regret bound for any randomized algorithm also holds for $K > 2$ lists. When $N \geq K/2$, we can also prove a second lower bound of $\Omega(\sqrt{BK})$ by simulating $K/2$ many 2-expert games using pairs of base lists. However the details of this second bound are complex and omitted due to space constraints. We conjecture that the lower bound on the expected regret is $\Omega(\sqrt{BK \ln(N/K)})$, i.e. that the expected regret bound of the algorithm of Section 4 is optimal within a constant factor.

6 Combining Lists with Duplicates

We now turn to the scenario where the base lists are not disjoint any more, i.e. items can occur on multiple lists. This minor difference in the setup has major implications. For one, finding the list that minimizes the number of misses is NP-hard, even if the base lists contain the same items in every trial. So we cannot hope for efficient algorithms with low regret unless RP = NP. We sketch a reduction from Set Cover in Section 6.1 and then work around this hardness result in Section 6.2 by linearizing the loss.

6.1 NP-Hardness by Reduction from Set Cover

An instance of the Set Cover problem consists of a collection \mathcal{C} of subsets of some universe U , and a number m . The question is whether a subcollection $\mathcal{D} \subseteq \mathcal{C}$ of size m exists such that the union of the subcollection is U . We transform this instance into a zero miss combined list existence problem with $K = |\mathcal{C}|$ base lists of size $N = m(p + |U|)$ and a sequence of $|U|$ requests. For each subset $S \in \mathcal{C}$, we introduce a base list that starts with p many padding items, then includes item i_x for each $x \in S$, and ends with padding to length N . By design, N is

such that a combined list can contain all non-padding items from m base lists. To ensure that it cannot contain non-padding items from $m + 1$ base lists, we must have that $(m + 1)(p + 1) > N$. The least such p equals $m(|U| - 1)$. The request sequence hits i_x for each $x \in U$.

If an m -cover exists, there is a combined list with zero loss. If not, then any combined list must miss at least one item. So an m -cover exists iff there is a combined list without any misses. Also unless RP=P, there cannot exist a polynomial time randomized algorithm with polynomial expected loss on request sequences for which there exists a combined list with no misses. (By polynomial we mean polynomial in N and K .)

If such an algorithm existed and achieved regret $p(N, k)$ then one could present it with random requests chosen with replacement from the given request sequence. If there does not exist a combined list with zero misses, then for such random request the expected loss must be at least 1 over the length of the request sequence. By presenting the algorithm with polynomially many random requests, the expected loss is either bounded by $p(N, k)$ or the expected loss must grow linearly in the number of requests. This clearly can be used to design a polynomial-time randomized decision procedure for the combined list problem we just showed to be NP-complete.

6.2 Working around the Hardness

When the lists are disjoint then in any trial the miss count $\mathbf{M}(\mathbf{c})$ is 0 either 1 for any combined list \mathbf{c} and the 0/1 loss is identical to $\mathbf{M}(\mathbf{c})$. When items can appear on multiple list, then $\mathbf{M}(\mathbf{c})$ can be larger than 1, whereas the 0/1 loss is $I(d - \mathbf{M}(\mathbf{c}))$, where d is the number of duplicates of the requested item, and $I(h) = 1$ if $h = 0$ and 0 if $h \geq 1$. The above reductions show that we cannot hope for an efficient algorithm with small expected regret measured by the 0/1 loss when the items can appear on multiple lists. Observe that $I(d - \mathbf{M}(\mathbf{c})) \leq \mathbf{M}(\mathbf{c})$. Therefore, in this section we propose to replace the 0/1 loss $I(d - \mathbf{M}(\mathbf{c}))$ by its upper bound $\mathbf{M}(\mathbf{c})$. This amounts to linearizing the loss because $\mathbf{M}(\mathbf{c}) = \sum_{k=1}^K M_{c_k, k}$. Note that the 0/1 loss $I(d - \mathbf{M}(\mathbf{c}))$ decidedly does not decompose into a sum over the component of \mathbf{c} .

This approach parallels how NP-hardness was avoided when learning disjunctions. There the 0/1 loss was replaced by the attribute loss which decomposes into a sum of the literals in the disjunction [Lit88]. This linearization of the loss is extensively discussed in [TW03].

Our efficient implementation of Hedge is based on the decomposition of the miss count and remains unchanged in the setting when items can appear on multiple lists. However, $\mathbf{M}(\mathbf{c})$ now lies in $[0, K]$ and the straightforward regret bounds have an additional factor of K due to the extended range. This means that in the case of combined lists the resulting regret bound has leading term

$$\sqrt{2BK \ln |\mathbb{C}_{N,K}|} \approx \sqrt{2BK^2 \ln(N+1)}.$$

The additional range factor also appears in the regret of the FPL algorithm. Several method have been developed for eliminating the range factor. First, the

range factor can be eliminated if the so called *unit rule* hold for the given loss and algorithm [KWK10]. However, there are simple counter examples for combined lists. Second, we can change the algorithm to the Component Hedge algorithm which has a range factor free regret bound with a leading term of

$$\sqrt{4BK \ln \frac{(N+K)\epsilon}{K}}.$$

This algorithm maintains its uncertainty as a *usage vector*, i.e. an element of the convex hull of the $(N+1) \times K$ indicator matrices corresponding to the combined lists. The weight update involves a relative entropy projection onto this convex hull, which can be expressed using few linear equality constraints as a linear image of the flow polytope. This projection can be computed using an iterative algorithm, but it's convergence properties have not yet been analyzed [KWK10].

7 Open Problems

In summary, we hope that we opened up a new approach for combining lists (of the same size N). We invented a new notion of mixture of such base lists in which the sizes of the top segments that were chosen from each list sum to N . We developed an efficient algorithm that implicitly maintains a mixture over exponentially many combined lists and proved regret bounds that are tight within a factor of $\min(\sqrt{K}, \sqrt{\ln(N/K)})$. Besides proving a tight lower bound for the $K > 2$ list case (notoriously hard), our approach leaves many open questions:

- We have ignored the question of how to rank the items within the combined list. Our notion of loss simply charges one unit depending on whether the requested item is in the combined list or not. Ideally, the combined list should be ranked and we would like to develop online algorithms for the case when the cost is proportional to how close each request is to the top of the list. One idea is to overlay N algorithms optimized for combined list sizes $1, 2, \dots, N$. Intuitively the item i on the list would miss the combined lists of size $1, 2, \dots, i-1$ and incur loss proportional to $i-1$. However this approach will only work when the the combined lists of size less than i are contained in the list of size i . So far we were not able to achieve good regret bounds with this approach.
- In caching applications it costs to change the “real cache” because items need to be reloaded. Our online algorithms currently ignore the reloading costs and this is particularly egregious for the probabilistic algorithms. The Shrinking Dartboard Algorithm, a method for lazily updating the expert followed by the Hedge algorithm was developed in [CVW10], and seems quite readily applicable to our setting. The Follow-the-Lazy-Leader algorithm [KV05] is another method requiring fewer updates. Also some good practical heuristics for reloading were given in [Gra03, GWA02]. However no performance guarantees were provided for these heuristics.
- At this point our algorithm is designed to achieve small regret compared to the best fixed combined list chosen in hindsight. In the expert setting there

is a long history of algorithms that can handle the case when the best expert is allowed to shift over time. This is achieved by first doing an exponential update and then mixing in a bit of either the uniform distribution over all experts or the average of all past distributions over the experts [HW98, BW02, GLL05]. In all experimental evaluations that the authors are aware of, it was crucial to extend the online algorithms to the shifting expert case [HLSS00, GWBA02]. It is a tedious but straightforward exercise to mix in a bit of the uniform distribution into the dynamic programming algorithm of Section 4, thus implementing the Fixed Share algorithm from [HW98]. The method is again based on recurrences, and maintains all weights implicitly. However, it adds an $O(T^2)$ factor to the running time of our current algorithm. We don't know how to do the fancier method efficiently, which mixes in a bit of the past average distribution. The reason is that the exponential weight updates on the N^K many combined lists seem to be at loggerheads with mixing in the past average weight. The resulting update is neither multiplicative nor additive and makes it difficult to implicitly maintain the weights.

In this respect Component Hedge [WK10] may have the advantage, as mixing in the past average usage vector can be done in constant time per component per trial. However, for this approach to be viable, an efficient implementation of the required relative entropy projections must be found.

- This paper is theoretical in that we focus on models for which we can prove regret bounds. However it would be useful to do practical experiments of the type done in [Gra03, GWBA02]. This would require us to blend the methods developed here with heuristics for handling for example the reloading issue.

Acknowledgments. Thanks to Anindya Sen and Corrie Scalisi for valuable ground work on this problem as part of a class and a Master's project [SS07, Sca07]. Thanks to Jyrki Kivinen for generous brainstorming.

References

- [AHR08] Abernethy, J., Hazan, E., Rakhlin, A.: Competing in the dark: An efficient algorithm for bandit linear optimization. In: Proceedings of the 21st Annual Conference on Learning Theory (July 2008)
- [AWY08] Abernethy, J., Warmuth, M.K., Yellin, J.: Optimal strategies for random walks. In: Proceedings of the 21st Annual Conference on Learning Theory (July 2008)
- [BCD⁺06] Bremner, D., Chan, T.M., Demaine, E.D., Erickson, J., Hurtado, F., Iacono, J., Langerman, S., Taslakian, P.: Necklaces, convolutions, and $x + y$. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 160–171. Springer, Heidelberg (2006)
- [BW02] Bousquet, O., Warmuth, M.K.: Tracking a small set of experts by mixing past posteriors. Journal of Machine Learning Research 3, 363–396 (2002)
- [CBL09] Cesa-Bianchi, N., Lugosi, G.: Combinatorial bandits. In: Proceedings of the 22nd Annual Conference on Learning Theory (June 2009)
- [CT65] Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. Mathematics of Computation 19(90), 297–301 (1965)

- [FS97] Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55, 119–139 (1997)
- [GLL05] György, A., Linder, T., Lugosi, G.: Tracking the best of many experts. In: Auer, P., Meir, R. (eds.) *COLT 2005. LNCS (LNAI)*, vol. 3559, pp. 204–216. Springer, Heidelberg (2005)
- [Gra03] Gramacy, R.B.: Adaptive caching by experts. PhD thesis, University of California at Santa Cruz (2003)
- [GVW10] Geulen, S., Vöcking, B., Winkler, M.: Regret minimization for online buffering problems using the Weighted Majority algorithm. In: Proceedings of the 23rd Annual Conference on Learning Theory (2010)
- [GWBA02] Gramacy, R.B., Warmuth, M.K., Brandt, S.A., Ari, I.: Adaptive caching by refetching. In: Becker, S., Thrun, S., Obermayer, K. (eds.) *NIPS*, pp. 1465–1472. MIT Press, Cambridge (2002)
- [HLSS00] Helmbold, D.P., Long, D.D.E., Sconyers, T.L., Sherrod, B.: Adaptive disk spin-down for mobile computers. In: *ACM/Baltzer Mobile Networks and Applications (MONET)*, pp. 285–297 (2000)
- [HP05] Hutter, M., Poland, J.: Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research* 6, 639–660 (2005)
- [HW98] Herbster, M., Warmuth, M.K.: Tracking the best expert. *Machine Learning* 32, 151–178 (1998)
- [KM05] Kontkanen, P., Myllymäki, P.: A fast normalized maximum likelihood algorithm for multinomial data. In: *IJCAI*, pp. 1613–1615 (2005)
- [KV05] Kalai, A., Vempala, S.: Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.* 71(3), 291–307 (2005)
- [KWK10] Koolen, W.M., Warmuth, M.K., Kivinen, J.: Hedging structured concepts. In: Proceedings of the 23rd Annual Conference on Learning Theory, pp. 93–105 (June 2010)
- [Lit88] Littlestone, N.: Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning* 2(4), 285–318 (1988)
- [LW94] Littlestone, N., Warmuth, M.K.: The Weighted Majority algorithm. *Information and Computation* 108(2), 212–261 (1994)
- [MM03] Megiddo, N., Modha, D.S.: One up on LRU. *login: The Magazine of USENIX and SAGE* 28(4), 6–11 (2003)
- [MM04] Megiddo, N., Modha, D.S.: Outperforming LRU with an adaptive replacement cache algorithm. *IEEE Computer* 37(4), 58–65 (2004)
- [Sea07] Scalisi, C.A.: An adaptive caching algorithm. Master's thesis, University of California Santa Cruz (June 2007)
- [SS07] Sen, A., Scalisi, C.: Making online predictions from k-lists. Project report for CMPS 290C, Advanced Topics in Machine Learning (June 2007)
- [TW03] Takimoto, E., Warmuth, M.K.: Path kernels and multiplicative updates. *Journal of Machine Learning Research* 4, 773–818 (2003)
- [Vov98] Vovk, V.: A game of prediction with expert advice. *J. of Comput. Syst. Sci.* 56(2), 153–173 (1998)

Regret Minimization Algorithms for Pricing Lookback Options*

Eyal Gofer and Yishay Mansour

Tel Aviv University,

Tel Aviv, Israel

{eyalgofe,mansour}@post.tau.ac.il

Abstract. In this work, we extend the applicability of regret minimization to pricing financial instruments, following the work of [11]. More specifically, we consider pricing a type of exotic option called a *fixed-strike lookback call option*. A fixed-strike lookback call option has a known expiration time, at which the option holder has the right to receive the difference between the *maximal* price of a stock and some pre-agreed price. We derive upper bounds on the price of these options, assuming an arbitrage-free market, by developing two-way trading algorithms. We construct our trading algorithms by combining regret minimization algorithms and one-way trading algorithms. Our model assumes upper bounds on the absolute daily returns, overall quadratic variation, and stock price, otherwise allowing for fully adversarial market behavior.

1 Introduction

Pricing options is a fundamental task in finance, both from a theoretical and a practical perspective. An *option* is a financial instrument that allows its holder to buy or sell a certain asset for a given price at a given time. For example, a *European call option*, at its expiration time T , allows its holder to buy the asset for a price K . In other words, the option pays $\max(S_T - K, 0)$ at time T , where S_T is the asset (stock) price at time T . In this work we examine an exotic option called a *European fixed-strike lookback call option*, which, at its expiration time T , allows its holder to choose the best time in hindsight to buy the asset for a price K . Namely, the lookback option pays $\max(M_T - K, 0)$ at time T , where M_T is the maximal asset price over the lifetime of the option.

Option pricing has been greatly influenced, theoretically and practically, by the works of Black and Scholes [4] and Merton [19]. In their Nobel Prize-winning works, they modeled the price of a stock as a geometric Brownian motion stochastic process. In addition, their model assumes an arbitrage-free market, namely, that market prices provide no opportunities for riskless profit. Over the years

* This research was supported in part by the Google Inter-university center for Electronic Markets and Auctions, by a grant from the Israel Science Foundation, by a grant from United States-Israel Binational Science Foundation (BSF), and by a grant from the Israeli Ministry of Science (MoS). This work is part of Ph.D. thesis research carried out by the first author at Tel Aviv University.

the Black-Scholes-Merton (BSM) model has been applied to pricing many types of financial instruments, including fixed-strike lookback options [7]. Despite its enormous success, the assumptions of the BSM model have several known drawbacks. First, the model is only an abstraction of price changes, while in reality prices are discrete and experience sharp jumps, and the daily returns are neither independent nor identically distributed. Second, the main parameter which is required, the stock volatility, is not observable, and has to be estimated.¹ In this work we investigate the pricing of lookback options in an adversarial online learning model which was introduced in [11]. A major advantage of such an adversarial online approach is that price jumps and discrete trading are inherently assumed in the model.

Before we discuss our main results, we introduce two elements that we will use in constructing our algorithms: *regret minimization* and *one-way trading*. Regret minimization, in a nutshell, devises algorithms that can compete well with respect to a given class of benchmarks. The measure of *regret* is the difference between the performance of the online algorithm and the best benchmark in the class. (See [4] for an excellent exposition of the topic.) In the one-way trading problem [12], one seeks to convert one type of asset to another, for example, yen to dollars, and trades are limited to selling yen for dollars. Since it is impossible to know in advance the best time to sell, one can try to minimize the *competitive ratio*, which upper bounds the ratio between the return from selling at the best price and the actual return achieved by the online algorithm.

In this paper we present a family of regret minimization algorithms that combines two algorithmic black boxes: a regret minimization component, and a one-way trading component. We translate the performance of the regret minimization and one-way trading components into upper bounds on the price of fixed-strike lookback options. This translation applies a general principle, namely, that if the regret bounds of an online algorithm guarantee that its payoff dominates the payoff of a financial instrument, then in an arbitrage-free market, the initial cost of the algorithm must exceed the price of the financial instrument. This approach was pioneered in [11], where it was used to derive upper and lower bounds on the price of European call options.

In our analysis we use specific one-way trading algorithm and regret minimization algorithms with the purpose of deriving concrete upper bounds on the price of lookback options. Specifically, for the regret minimization component, we concentrate on the Polynomial Weights algorithm [5]. For the one-way trading component, we consider a simple price-oriented algorithm and the optimal competitive one-way trading algorithm [12]. Based on that, we derive explicit upper bounds for the price of lookback options. We stress that our combination of regret minimization with one-way trading results, in general, in two-way trading (as discussed in Section 5).

¹ In fact, in many cases people compute the *implied volatility*, which is the volatility that under the BSM model would give the current option price. It is well documented that the implied volatility is not constant, even for a given expiration time, and depends on the strike price.

We conclude with an experimental study of our bounds for the price of look-back options, considering data from the S&P 500 index. Despite the adversarial nature of our model, the upper bounds achieved are reasonable even when compared to the BSM pricing.

Related Work. In learning theory, the most widely studied finance problem is *portfolio selection*, where the most commonly used benchmark is the *best constantly rebalanced portfolio* (BCRP). A key result by Cover [8] gives an algorithm, the *universal portfolio*, which achieves the same asymptotic growth rate as the BCRP, even under adversarial conditions. Subsequent work incorporated side information and transaction costs, proved optimal regret w.r.t. the BCRP, improved computational complexity, and considered short selling [9,20,21,17,22]. Other works combining learning and portfolio selection are [15,21,3,13,14].

The one-way trading and *search* problems were first studied in [12] and later in [6,16,18], where competitive ratios for various finance problems were analyzed. In the search problem, price offers are made sequentially to a player whose aim is to sell only once and at the highest possible price. The search problem with k points, with application to the pricing of *floating-strike lookback calls*,² was studied in [18]. Our model and the classical search and one-way trading model, used in [12,18], differ in two important ways. First, we allow two-way trading, and second, we assume a bound on the total quadratic variation. Indeed, we show that by allowing these two additional features, we can obtain a better competitive ratio than the optimal one-way trading competitive ratio of [12].

Outline. The outline of the paper is as follows. In Section 2 we provide notation and definitions. Section 3 presents an upper bound on the price of lookback options, given the regret bounds of a trading algorithm. Section 4 introduces a family of algorithms that combines regret minimization and one-way trading, with resulting new bounds on the price of lookback options. In Section 5 we discuss our results. Section 6 presents empirical results. Due to space limitations, some proofs are omitted.

2 Preliminaries

We consider a discrete-time finite-horizon model, with a risky asset (stock) and a risk-free asset (bond or cash). The price of the stock at time t is S_t and the price of the risk-free asset is B_t . Initially, $B_0 = S_0 = 1$. We assume that the price of cash does not change, i.e., $B_t = 1$, which is equivalent to assuming a zero risk-free interest rate. We further assume that we can buy or sell any real quantity of stocks with no transaction costs. For the stock we denote by r_t the single period return between $t - 1$ and t , so that $S_t = S_{t-1}(1 + r_t)$.

² This type of security pays $S_T - m_T$ at time T , where S_T is the stock price at time T , and m_T is the minimal stock price over the lifetime of the option. It is different from the fixed-strike lookback option, which we price, making our bounds incomparable.

A realization of the prices is a *price path*, which is the vector $\mathbf{r}_t = (r_1, \dots, r_t)$. We define a few parameters of a price path. We denote by M a bound on the maximum stock price S_t , by R a bound on the absolute single period change $|r_t|$, by Q a bound on the quadratic variation $\sum_{t=1}^T r_t^2$, and by M_t the maximum price up to time t , $M_t = \max_{0 \leq u \leq t} S_u$. We will assume that the bounds R , Q , and M are given, and $\Pi_{M,R,Q}$, or simply Π , will denote the set of all price paths satisfying these bounds. Since $\max_{1 \leq t \leq T} (r_t^2) \leq \sum_{t=1}^T r_t^2 < R^2 T$, we may assume that $R^2 \leq Q \leq R^2 T$. The number of time steps, T , is influenced by both the frequency of trading and the absolute time duration. For this reason it is instructive to consider M and Q as fixed, rather than as increasing in T .

A *trading algorithm* \mathbf{A} starts with a total asset value V_0 . At every time period $t \geq 1$, \mathbf{A} sets weights $w_{s,t} \geq 0$ for stock and $w_{c,t} \geq 0$ for cash, and we define the fractions $p_{s,t} = w_{s,t}/W_t$ and $p_{c,t} = w_{c,t}/W_t$, where $W_t = w_{s,t} + w_{c,t}$. A fraction $p_{s,t}$ of the total asset value, V_{t-1} , is placed in stock, and likewise, $p_{c,t}V_{t-1}$ is placed in cash. Following that, the stock price S_t becomes known, the asset value is updated to $V_t = V_{t-1}(1 + p_{s,t}r_t)$, and time period $t+1$ begins.

We comment that since we assume that both weights are non-negative, the algorithms we consider use neither short selling of the stock, nor buying on margin (negative positions in cash). However, as part of the arbitrage-free assumption, we assume that short selling is, in general, allowed in the market.

An algorithm is referred to as *one-way trading* from stock to cash if the amount of cash never decreases over time, i.e., for every $t \geq 1$ we have $p_{c,t}V_{t-1} \leq p_{c,t+1}V_t$, otherwise it is referred to as *two-way trading*.

A European call option $\mathbf{C}(K, T)$ is a security paying $\max(S_T - K, 0)$ at time T , where K is the *strike price* and T is the *expiration time*. The value of the option at time 0 will be denoted by $C(K, T)$. A *fixed-strike lookback call option* $\mathbf{LC}(K, T)$ is a security paying $\max(M_T - K, 0)$ at time T , where $M_T = \max_{0 \leq t \leq T} S_t$, and its value at time 0 will be denoted by $LC(K, T)$. The interesting range for the strike price of lookback options is $K \in [S_0, M]$. The reason is that for $K \geq M$, the payoff is always 0, so $LC(K, T) = 0$, and for $K < S_0$, $LC(K, T) = LC(S_0, T) + S_0 - K$. Thus, we assume that $K \in [S_0, M]$.

3 Arbitrage-Free Bounds

We assume that the pricing is *arbitrage-free*, which is defined as follows. Trading algorithm \mathbf{A}_1 *dominates* trading algorithm \mathbf{A}_2 w.r.t. the set of price paths Π , if for every price path in Π , the final value of \mathbf{A}_1 is at least the final value of \mathbf{A}_2 . The *arbitrage-free assumption* says that if \mathbf{A}_1 dominates \mathbf{A}_2 then the initial value of \mathbf{A}_1 is at least the initial value of \mathbf{A}_2 . This assumption is natural, because if it is broken, it becomes possible to make a riskless profit by buying into \mathbf{A}_1 and selling \mathbf{A}_2 . The resulting flow of funds from \mathbf{A}_2 to \mathbf{A}_1 affects the stock price in a way that causes even a small arbitrage opportunity to quickly disappear.

For example, define trading algorithm \mathbf{A}_{LC} which simply buys the lookback option and holds it. Its initial value is $LC(K, T)$ and its final value is $\max(M_T - K, 0)$. (We sometimes refer to \mathbf{A}_{LC} as simply $\mathbf{LC}(K, T)$.) Assume we design a

trading strategy \mathbf{A}_1 whose initial value is V_0 and at time T its value is $V_T \geq \max(M_T - K, 0)$ for every possible price path. This means that \mathbf{A}_1 dominates \mathbf{A}_{LC} . Therefore, by the arbitrage-free assumption, we have that $LC(K, T) \leq V_0$.

We now establish a connection between bounds on the multiplicative regret of a trading algorithm and arbitrage-free pricing of lookback options. This connection is similar to the one given in [11] for European call options.

Definition 1. Let \mathbf{A} be a trading algorithm and let V_t be its asset value at time $0 \leq t \leq T$, assuming $V_0 = 1$. \mathbf{A} is said to have an (α, β) multiplicative regret, for some $\alpha, \beta > 0$, if for every price path, $V_T \geq \max(\alpha, \beta M_T)$.

Lemma 1. If an algorithm with (α, β) multiplicative regret exists, then

$$LC(K, T) \leq \frac{1}{\min(\alpha/K, \beta)} - K .$$

Proof. Consider a new security, $\mathbf{LC}_1(K, T)$, which pays $\max(M_T, K)$ at time T , and let $LC_1(K, T)$ be its value at time 0. Since the payoff of $\mathbf{LC}_1(K, T)$ is always K plus the payoff of $\mathbf{LC}(K, T)$, then, by the arbitrage-free assumption, $LC_1(K, T) = LC(K, T) + K$. For an algorithm with (α, β) multiplicative regret, we have that $V_T \geq \max(\alpha, \beta M_T)$. If $\alpha \geq \beta K$, then

$$\max(\alpha, \beta M_T) \geq \max(\beta K, \beta M_T) = \beta \max(K, M_T) .$$

Otherwise, $\alpha < \beta K$, and then

$$\max(\alpha, \beta M_T) \geq \max(\alpha, \frac{\alpha}{K} M_T) = \frac{\alpha}{K} \max(K, M_T) .$$

Therefore, in any case, $\max(\alpha, \beta M_T) \geq \min(\alpha/K, \beta) \cdot \max(M_T, K)$, so the algorithm dominates $\min(\alpha/K, \beta)$ units of $\mathbf{LC}_1(K, T)$. By the arbitrage-free assumption, we have that

$$LC_1(K, T) \leq \frac{1}{\min(\alpha/K, \beta)} .$$

Since $LC_1(K, T) = LC(K, T) + K$, the result follows. \square

We note that Lemma 1 indicates exactly how improved regret bounds for a trading algorithm relate to tighter upper bounds on $LC(K, T)$.

4 Trading Algorithms and Bounds for Lookback Options

4.1 Simple Arbitrage-Free Bounds

We start with a few simple arbitrage-free bounds. Let \mathbf{A}_{TS} be a trading algorithm that starts with T stocks (initial value S_0T) and sells one stock at each time $t \in [1, T]$. Since the final value of \mathbf{A}_{TS} is $\sum_{t=1}^T S_t \geq M_T - S_0 \geq M_T - K$, we have that

Theorem 1. \mathbf{A}_{TS} dominates $\mathbf{LC}(K, T)$, implying that $LC(K, T) \leq TS_0 = T$.

A similar strategy \mathbf{A}_{TC} uses ordinary call options rather than stocks. \mathbf{A}_{TC} buys one call option for each expiration time $t \in [1, T]$ (initial value $\sum_{t=1}^T C(K, t)$) and simply collects the payoffs. Since the final value of \mathbf{A}_{TC} is $\sum_{t=1}^T \max(S_t - K, 0) \geq \max(M_T - K, 0)$, we have that

Theorem 2. \mathbf{A}_{TC} dominates $\mathbf{LC}(K, T)$, therefore, $LC(K, T) \leq \sum_{t=1}^T C(K, t)$.

While the previous strategies were time-oriented, the next strategy is price-oriented. Algorithm \mathbf{A}_{PS} starts with $N = 1 + \lfloor \log_2(M/K) \rfloor$ stocks, and, therefore, has an initial value of NS_0 . \mathbf{A}_{PS} sells stock i , $i \in [0, N - 1]$, at the first time $S_t \geq 2^i K$. (This strategy is very similar to one discussed in [2].)

Theorem 3. \mathbf{A}_{PS} dominates $\mathbf{LC}(K, T)$, and $LC(K, T) \leq NS_0 = 1 + \lfloor \log_2 \frac{M}{K} \rfloor$.

We remark that using the methodology of Dawid et al. [10] one can derive an improved bound of $\ln(M/K) - 1 + (K/M)$.

4.2 Combining Regret Minimization and One-Way Trading

The simple algorithms reveal what is required of an algorithm to dominate $\mathbf{LC}(K, T)$. Such an algorithm must always set aside enough cash to cover the payoff of the option. In the vicinity of record highs, cash reserves may need to be increased by selling stocks. At the same time, enough stocks must be retained in case even higher price levels are reached.

The problem may also be seen as that of predicting the point where the stock price reaches a maximum. We may think of a set of experts, each with its own selling trigger as a function of market conditions, where the job of the algorithm is to choose the best expert. This casts the problem in the best expert and regret minimization frameworks. Such a formulation turns out to be equivalent to a conventional regret minimization algorithm working with stock and cash, combined with a one-way trading mechanism.

Our algorithm thus contains both a regret minimization element and a one-way trading element. This combination will enable us to bound the regret w.r.t. stock prices *over the whole price path of the stock*. This is in contrast to performing conventional regret minimization, which would only bound the regret w.r.t. the final stock price. At the same time, this approach is also a generalization of regret minimization with stock and cash, because by picking an inactive one-way trading element, we can obtain an ordinary regret minimization algorithm.

We define our trading algorithm using two parameters, a *one-way trading rule* and a *regret minimization rule*. Intuitively, the one-way trading rule will move funds gradually from stock to cash, while the regret minimization rule will try to follow whichever asset is performing better, stock or cash (using a regret minimization algorithm).

The *one-way trading rule* is defined by a function h mapping histories to $[0, 1]$. We require that h be monotonically non-increasing along a price path, i.e., if \mathbf{r}_t is a prefix of \mathbf{r}_{t+1} then $h(\mathbf{r}_t) \geq h(\mathbf{r}_{t+1})$. In addition, h will start at 1,

i.e., $h(\mathbf{r}_0) = 1$. In the algorithm we will use the notation $H_t = h(\mathbf{r}_{t-1})$, so that $1 = H_1 \geq H_2 \geq \dots \geq 0$, and $H = (H_1, \dots, H_{T+1})$. If we set $H_t = 1$ for every t , we will essentially not transfer funds from stock to cash.

The *regret minimization rule* performs regret minimization between stock and cash. It is defined by the update equations $w_{s,t+1} = w_{s,t}f(\mathbf{r}_t)$, and $w_{c,t+1} = w_{c,t}$, where $f : \cup_{t \geq 1} \mathbb{R}^t \rightarrow \mathbb{R}^+$. In what follows, we use $f(\mathbf{r}_t) = 1 + \eta r_t$, which is the regret minimization rule of the Polynomial Weights algorithm [5], as adapted in [11]. We will require that $\eta \in [1, \eta_{max}]$, where $\eta_{max} = \frac{1-2R}{2R(1-R)}$, and that $R < 1 - 1/\sqrt{2} \approx 0.3$. In the next section we will show that $\eta = 1$ has the interpretation of a *buy and hold* strategy, namely, the regret minimization rule maintains its initial allocation of stock and cash. For convenience, we denote $g(\mathbf{r}_t) = \prod_{u=1}^t f(\mathbf{r}_u)$, for the accumulated update of f .

We now give the exact definition of our family of trading algorithms. Initially, $w_{c,1}, w_{s,1} > 0$, and for $t \geq 1$,

$$\begin{aligned} w_{s,t+1} &= w_{s,t} \frac{H_{t+1}}{H_t} f(\mathbf{r}_t) = w_{s,1} H_{t+1} g(\mathbf{r}_t), \\ w_{c,t+1} &= w_{c,t} + w_{s,1} (H_t - H_{t+1}) g(\mathbf{r}_t) = w_{c,t} + w_{s,t} f(\mathbf{r}_t) - w_{s,t+1}. \end{aligned}$$

Recall that a trading algorithm invests a fraction $w_{s,t}/(w_{s,t} + w_{c,t})$ of assets in stock, and the rest in cash. We assume that initially, $V_0 = 1$. The following lemma relates the current weights to the initial weights, using the stock price, the variation parameter Q and the one-way trading rule.

Lemma 2. *For every $\eta \in [1, \eta_{max}]$,*

$$\begin{aligned} w_{s,t+1} &\geq H_{t+1} w_{s,1} S_t^\eta e^{-\eta(\eta-1)Q}, \\ w_{c,t+1} &\geq w_{c,1} + w_{s,1} e^{-\eta(\eta-1)Q} \sum_{u=1}^t (H_u - H_{u+1}) S_u^\eta. \end{aligned}$$

Proof. We first observe that for every $1 \leq \tau \leq T$, since $g(\mathbf{r}_\tau) = \prod_{u=1}^\tau f(\mathbf{r}_u) = \prod_{u=1}^\tau (1 + \eta r_u)$, we have that

$$\begin{aligned} \ln g(\mathbf{r}_\tau) &= \sum_{u=1}^\tau \ln(1 + \eta r_u) \geq \eta \sum_{u=1}^\tau \ln(1 + r_u) - \eta(\eta-1) \sum_{u=1}^\tau r_u^2 \\ &= \eta \ln S_\tau - \eta(\eta-1) \sum_{u=1}^\tau r_u^2 \geq \ln S_\tau^\eta - \eta(\eta-1)Q = \ln S_\tau^\eta e^{-\eta(\eta-1)Q}, \end{aligned}$$

where the first inequality uses the fact that for every $\eta \in [1, \eta_{max}]$ and $r > -R$, $\ln(1 + \eta r) \geq \eta \ln(1 + r) - \eta(\eta-1)r^2$ (see [11]). Thus, for every $1 \leq \tau \leq T$, $g(\mathbf{r}_\tau) \geq S_\tau^\eta e^{-\eta(\eta-1)Q}$. We therefore have that

$$\frac{w_{s,t+1}}{w_{s,1}} = \prod_{u=1}^t \frac{w_{s,u+1}}{w_{s,u}} = \prod_{u=1}^t \frac{H_{u+1}}{H_u} f(\mathbf{r}_u) = \frac{H_{t+1}}{H_1} g(\mathbf{r}_t) \geq H_{t+1} S_t^\eta e^{-\eta(\eta-1)Q},$$

for the weight of stock, and

$$\begin{aligned}
w_{c,t+1} &= w_{c,1} + \sum_{u=1}^t (w_{c,u+1} - w_{c,u}) = w_{c,1} + w_{s,1} \sum_{u=1}^t (H_u - H_{u+1}) g(\mathbf{r}_u) \\
&\geq w_{c,1} + w_{s,1} \sum_{u=1}^t (H_u - H_{u+1}) S_u^\eta e^{-\eta(\eta-1)Q} \\
&= w_{c,1} + w_{s,1} e^{-\eta(\eta-1)Q} \sum_{u=1}^t (H_u - H_{u+1}) S_u^\eta ,
\end{aligned}$$

for the weight of cash. Both inequalities used our lower bound on $g(\mathbf{r}_\tau)$. \square

The following theorem lower bounds the profit of the algorithm in terms of H , and facilitates the proof of (α, β) multiplicative regret results for given one-way trading rules.

Theorem 4. *For every $\eta \in [1, \eta_{max}]$,*

$$V_T \geq \left(p_{c,1} + p_{s,1} e^{-\eta(\eta-1)Q} \left(\sum_{t=1}^T (H_t - H_{t+1}) S_t^\eta + H_{T+1} S_T^\eta \right) \right)^{\frac{1}{\eta}} .$$

Proof. We have that

$$\begin{aligned}
\ln \frac{W_{T+1}}{W_1} &= \sum_{t=1}^T \ln \frac{W_{t+1}}{W_t} = \sum_{t=1}^T \ln \frac{w_{c,t+1} + w_{s,t+1}}{W_t} = \sum_{t=1}^T \ln \frac{w_{c,t} + w_{s,t} f(\mathbf{r}_t)}{W_t} \\
&= \sum_{t=1}^T \ln \frac{w_{c,t} + w_{s,t} (1 + \eta r_t)}{W_t} = \sum_{t=1}^T \ln (1 + \eta p_{s,t} r_t) \\
&= \sum_{t=1}^T \ln \left(1 + \eta \left(\frac{V_t}{V_{t-1}} - 1 \right) \right) \leq \sum_{t=1}^T \eta \ln \frac{V_t}{V_{t-1}} = \eta \ln V_T ,
\end{aligned}$$

where the inequality uses the fact that for every $\eta \in [1, \eta_{max}]$ and $r > -R$, $\ln(1 + \eta r) \leq \eta \ln(1 + r)$ (see [III]). On the other hand, we have that

$$\begin{aligned}
\ln \frac{W_{T+1}}{W_1} &= \ln \frac{w_{c,T+1} + w_{s,T+1}}{W_1} \\
&\geq \ln \left(\frac{1}{W_1} \left(w_{c,1} + w_{s,1} e^{-\eta(\eta-1)Q} \left(\sum_{t=1}^T (H_t - H_{t+1}) S_t^\eta + H_{T+1} S_T^\eta \right) \right) \right) \\
&= \ln \left(p_{c,1} + p_{s,1} e^{-\eta(\eta-1)Q} \left(\sum_{t=1}^T (H_t - H_{t+1}) S_t^\eta + H_{T+1} S_T^\eta \right) \right) ,
\end{aligned}$$

where the inequality is by Lemma 2. Together, we have that

$$\eta \ln V_T \geq \ln \left(p_{c,1} + p_{s,1} e^{-\eta(\eta-1)Q} \left(\sum_{t=1}^T (H_t - H_{t+1}) S_t^\eta + H_{T+1} S_T^\eta \right) \right) ,$$

and, rearranging, we get the desired result. \square

A Price-Oriented Rule and Bound. The one-way trading rule presented in this subsection is reminiscent of the simple algorithm A_{PS} in that it depends only on the maximal price so far, M_t . Like A_{PS} , unless the price is at an all-time high, this rule does not trigger a sale of stocks, i.e., $H_{t+1} = H_t$. However, the combined trading algorithm may still sell stocks at any price level due to the regret minimization rule.

Lemma 3. *Using $H_t = \ln(M/\mu_{t-1})/\ln(M/K)$, where $\mu_t = \max(K, M_t)$, the algorithm achieves an (α, β) multiplicative regret with $\alpha = p_{c,1}^{1/\eta}$, and*

$$\beta = \min \left\{ \frac{p_{c,1}^{\frac{1}{\eta}}}{K}, \left(\frac{p_{s,1} e^{-\eta(\eta-1)Q}}{\eta \ln(M/K)} \right)^{\frac{1}{\eta}} \right\}.$$

Proof. From Theorem 4 we have that

$$V_T \geq \left[p_{c,1} + p_{s,1} e^{-\eta(\eta-1)Q} \left(\sum_{t=1}^T (H_t - H_{t+1}) S_t^\eta + H_{T+1} S_T^\eta \right) \right]^{\frac{1}{\eta}}. \quad (1)$$

Since $H_1 \geq \dots \geq H_{T+1} \geq 0$, we have that $V_T \geq p_{c,1}^{1/\eta} = \alpha$, which proves the α part of the (α, β) multiplicative regret. For the β part, we consider two cases. If $M_T \leq K$, then $V_T \geq p_{c,1}^{1/\eta} \geq p_{c,1}^{1/\eta} K^{-1} M_T \geq \beta M_T$, as required. We now assume $M_T > K$. From Equation (1) we get

$$V_T \geq \left(p_{c,1} + p_{s,1} e^{-\eta(\eta-1)Q} \frac{1}{\ln(M/K)} \sum_{t=1}^T S_t^\eta \ln \frac{\mu_t}{\mu_{t-1}} \right)^{\frac{1}{\eta}}, \quad (2)$$

by definition of H_t and using the fact that $H_{T+1} \geq 0$. The values μ_t form a non-decreasing sequence, and we have that $\mu_T = M_T > K = \mu_0$. Let $\mu_T^\eta = x_l > x_{l-1} > \dots > x_0 = \mu_0^\eta$ be the distinct values of μ_t^η , $0 \leq t \leq T$, for some $l \geq 1$. Note that $S_t^\eta \ln \frac{\mu_t}{\mu_{t-1}} = 0$ if $\mu_t = \mu_{t-1}$, and $S_t^\eta \ln \frac{\mu_t}{\mu_{t-1}} = \mu_t^\eta \ln \frac{\mu_t}{\mu_{t-1}} = \frac{1}{\eta} \cdot \mu_t^\eta \ln \frac{\mu_t^\eta}{\mu_{t-1}^\eta}$ otherwise. Therefore,

$$\eta \sum_{t=1}^T S_t^\eta \ln \frac{\mu_t}{\mu_{t-1}} = \sum_{i=1}^l x_i \ln \frac{x_i}{x_{i-1}} \geq x_l - x_0 = \mu_T^\eta - \mu_0^\eta = M_T^\eta - K^\eta,$$

where the inequality is true since for every $z_n > \dots > z_0 > 0$, we have that $\sum_{i=1}^n z_i \ln(z_i/z_{i-1}) \geq z_n - z_0$. Plugging this into Equation (2) gives

$$V_T \geq \left(p_{c,1} + \frac{p_{s,1} e^{-\eta(\eta-1)Q}}{\eta \ln(M/K)} (M_T^\eta - K^\eta) \right)^{\frac{1}{\eta}}.$$

Denoting $\gamma = \frac{p_{s,1} e^{-\eta(\eta-1)Q}}{\eta \ln(M/K)}$, we have

$$V_T \geq ((p_{c,1} - \gamma K^\eta) + \gamma M_T^\eta)^{\frac{1}{\eta}} = \left(\frac{p_{c,1} - \gamma K^\eta}{M_T^\eta} + \gamma \right)^{\frac{1}{\eta}} \cdot M_T.$$

If $p_{c,1} - \gamma K^\eta \geq 0$, then $V_T \geq \gamma^{1/\eta} M_T \geq \beta M_T$, as required. Otherwise, since $M_T > K$, we have that

$$\left(\frac{p_{c,1} - \gamma K^\eta}{M_T^\eta} + \gamma \right)^{\frac{1}{\eta}} \geq \left(\frac{p_{c,1} - \gamma K^\eta}{K^\eta} + \gamma \right)^{\frac{1}{\eta}} = \frac{\frac{1}{\eta}}{K} ,$$

and, therefore, $V_T \geq (p_{c,1}^{1/\eta}/K) \cdot M_T \geq \beta M_T$, and the proof is complete. \square

This multiplicative regret result implies a new bound on $LC(K, T)$.

Theorem 5. For $\eta \in [1, \eta_{max}]$, $LC(K, T) \leq (K^\eta + S_0^\eta \eta e^{\eta(\eta-1)Q} \ln \frac{M}{K})^{1/\eta} - K$.

Proof. Recall that $S_0 = 1$. Combining Lemmas 2 and 3, we have that

$$LC(K, T) \leq \frac{1}{\min(\alpha/K, \beta)} - K ,$$

where $\alpha = p_{c,1}^{1/\eta}$ and $\beta = \min(p_{c,1}^{1/\eta}/K, \gamma^{1/\eta})$, denoting $\gamma = \frac{p_{s,1}e^{-\eta(\eta-1)Q}}{\eta \ln(M/K)}$. Since $\alpha/K \geq \beta$, we have $LC(K, T) \leq 1/\beta - K = \max(Kp_{c,1}^{-1/\eta}, \gamma^{-1/\eta}) - K$. We now optimize for $p_{c,1} = 1 - p_{s,1}$. The term $Kp_{c,1}^{-1/\eta}$ is decreasing in $p_{c,1}$, while the term $\gamma^{-1/\eta}$ is increasing in $p_{c,1}$. Thus, to minimize the bound, we would like to pick $p_{c,1}$ such that $Kp_{c,1}^{-1/\eta} = \gamma^{-1/\eta}$. It is easy to verify that $p_{c,1} = (1 + K^{-\eta} \eta e^{\eta(\eta-1)Q} \ln(M/K))^{-1}$ satisfies this requirement, and is also in $[0, 1]$, and thus a valid choice. Therefore,

$$\begin{aligned} LC(K, T) &\leq Kp_{c,1}^{-\frac{1}{\eta}} - K = K \cdot \left(1 + K^{-\eta} \eta e^{\eta(\eta-1)Q} \ln(M/K) \right)^{\frac{1}{\eta}} - K \\ &= \left(K^\eta + \eta e^{\eta(\eta-1)Q} \ln(M/K) \right)^{\frac{1}{\eta}} - K . \end{aligned} \quad \square$$

Bounds Based on Competitive Ratio. For every one-way trading rule h , and every $\eta \in [1, \eta_{max}]$, we may define

$$\rho_h(\eta) = \sup_{\Pi} \frac{M_T^\eta}{\sum_{t=1}^T (H_t - H_{t+1}) S_t^\eta + H_{T+1} S_T^\eta} .$$

Namely, $\rho_h(\eta)$ is the competitive ratio of h w.r.t. the sequence $\{S_t^\eta\}_{t=1}^T$. Assume $\bar{\rho}_h$ is a known upper bound on ρ_h , so $\bar{\rho}_h(\eta) \geq \rho_h(\eta)$ for every $\eta \in [1, \eta_{max}]$ (if ρ_h is known explicitly, then $\bar{\rho}_h = \rho_h$). We next bound $LC(K, T)$ in terms of $\bar{\rho}_h(\eta)$.

Theorem 6. Let h be the one-way trading rule used in the trading algorithm. For every $\eta \in [1, \eta_{max}]$, there is a choice of $p_{c,1}$ for which the trading algorithm has a $(\beta K, \beta)$ multiplicative regret, and, therefore, $LC(K, T) \leq 1/\beta - K$, where

$$\beta = \left(\frac{K^\eta - S_0^\eta + S_0^\eta e^{\eta(\eta-1)Q} \bar{\rho}_h(\eta) (1 - (K/M)^\eta)}{1 - (S_0/M)^\eta} \right)^{-\frac{1}{\eta}} .$$

Proof. Recall that $S_0 = 1$. Denote $b = e^{-\eta(\eta-1)Q} \bar{\rho}_h(\eta)^{-1}$ and $p = p_{c,1}$ for convenience. By definition of $\bar{\rho}_h(\eta)$ and by Theorem 4,

$$V_T \geq (p + (1-p)e^{-\eta(\eta-1)Q} \bar{\rho}_h(\eta)^{-1} M_T^\eta)^{\frac{1}{\eta}} = (p + (1-p)b M_T^\eta)^{\frac{1}{\eta}}.$$

Thus, we have that $V_T \geq (p + (1-p)b)^{1/\eta} = (p(1-b) + b)^{1/\eta}$, and, in addition,

$$V_T \geq M_T (p M^{-\eta} + (1-p)b)^{\frac{1}{\eta}} = M_T (p (M^{-\eta} - b) + b)^{\frac{1}{\eta}}.$$

We therefore have an (α, β) multiplicative regret with $\alpha = (p(1-b) + b)^{1/\eta}$ and $\beta = (p(M^{-\eta} - b) + b)^{1/\eta}$. By Lemma 11, $LC(K, T) \leq \frac{1}{\min(\alpha/K, \beta)} - K$. Define $p_0 = \frac{(K^\eta - 1)b}{(K^\eta - 1)b + 1 - (K/M)^\eta}$. Since $1 \leq K < M$, we have that $p_0 \in [0, 1]$, so we may pick $p = p_0$. It can be easily verified that, given this choice, $\alpha/K = \beta$, and, therefore, $LC(K, T) \leq 1/\beta - K$, and

$$\begin{aligned} \beta &= (p_0(M^{-\eta} - b) + b)^{\frac{1}{\eta}} = \left(\frac{K^\eta - 1 + \frac{1}{b}(1 - (K/M)^\eta)}{1 - M^{-\eta}} \right)^{-\frac{1}{\eta}} \\ &= \left(\frac{K^\eta - 1 + e^{\eta(\eta-1)Q} \bar{\rho}_h(\eta)(1 - (K/M)^\eta)}{1 - M^{-\eta}} \right)^{-\frac{1}{\eta}}. \end{aligned} \quad \square$$

For $\eta = 1$, the above bound becomes simpler:

Corollary 1. *If $\eta = 1$, then $LC(K, T) \leq S_0(\bar{\rho}_h(1) - 1)\frac{M-K}{M-S_0}$.*

Setting $H_t \equiv 1$, the principle used in the proof of Theorem 6 can be utilized to prove a variation on the upper bound given in 11 for $C(K, T)$.

Theorem 7. *For every $\eta \in [1, \eta_{max}]$,*

$$C(K, T) \leq \left(K^\eta + S_0^\eta e^{\eta(\eta-1)Q} (1 - (K/M)^\eta) \right)^{\frac{1}{\eta}} - K.$$

5 Discussion of the Bounds

Direction of Trading: One-Way versus Two-Way. The algorithm family of Section 4 combines regret minimization and one-way trading components. We would like to show that in general, this results in two-way trading algorithms, although we may set the parameter η so as to obtain a one-way trading algorithm. The analysis applies also to the trading algorithm of 11, which corresponds to the special case where the one-way trading component is inactive, i.e., $H_t \equiv 1$.

The dynamics of our algorithms are an interplay between the dynamics of their two components. However, whenever $H_{t+1} = H_t$, the only active element is the regret minimization component $f(\mathbf{r}_t)$, making the dynamics easier to characterize. Note that for any h , simply cashing in less frequently can guarantee that $H_{t+1} = H_t$ often, without changing the essential workings of the algorithm. For $H_t = \frac{\ln(M/\mu_{t-1})}{\ln(M/K)}$, the price-oriented rule used in Theorem 5, we have that $H_{t+1} = H_t$ if the stock price is not at a new all-time high. The following claim gives a characterization of the direction of trading.

Claim. For any $f(\mathbf{r}_t)$, if $H_{t+1} = H_t$, then the sign of $f(\mathbf{r}_t) - 1 - r_t$ determines the direction of trading at time $t + 1$. If $f(\mathbf{r}_t) = 1 + \eta r_t$, then for $\eta = 1$ the algorithm is one-way trading; for $\eta > 1$, if $H_{t+1} = H_t$, the algorithm buys stocks when prices rise and sells stocks when prices drop.

Relation to the Search and One-Way Trading Problems. Any one-way trading algorithm h may be used in our configuration. In particular, Theorem 6 shows how the competitive ratio characteristics of h determine the multiplicative regret of our algorithm. In particular, it shows that improving $\rho_h(\eta)$, the competitive ratio of h w.r.t. the sequences $\{S_t^\eta\}_{t=1}^T$, improves the multiplicative regret of the two-way trading algorithm.

An optimal competitive ratio one-way trading algorithm is derived in [12], in a scenario where only an upper bound M and a lower bound M/φ on the stock price are known, and T trades are allowed. The optimal competitive ratio $\rho^* = \rho^*(T, \varphi)$ is defined by $(\varphi - 1)(1 - \rho^*/T)^T + 1 = \rho^*$. Equivalently, this implies an (α, β) multiplicative regret with $\beta = 1/\rho^*$.

In our model we deviate in two important ways from their model. First, we introduce an additional parameter that bounds the quadratic variation, Q , and second, we allow for two-way trading. Therefore, it is not surprising that for certain settings we may improve on their optimal competitive ratio. Denoting h for the optimal algorithm, we can show that whenever $\ln(\rho_h(1)) - \rho'_h(1)/\rho_h(1) > Q$, our trading algorithm, using specific parameters and h as the one-way trading rule, has a better competitive ratio than h . In general, bigger values of φ and smaller values of Q make for bigger improvements.

Comparison to the Simple Bounds. We now show that the price-oriented bound of Theorem 5 is superior to the bounds of the simple algorithms of Sub-section 4.1. For algorithms \mathbf{A}_{TS} and \mathbf{A}_{PS} , which are clearly one-way trading, we will show that the price-oriented bound is superior even when we set $\eta = 1$ (and thus have a one-way trading algorithm).

The bound for \mathbf{A}_{PS} is $1 + \lfloor \log_2(M/K) \rfloor$, while, for $\eta = 1$, Theorem 5 gives a bound of $\ln(M/K)$. We have that

$$\ln(M/K) = \log_2(M/K) \cdot \ln 2 \approx 0.69 \log_2(M/K) < 1 + \lfloor \log_2(M/K) \rfloor .$$

If we make no extra assumptions on the growth of Q over time, then the bound of Theorem 5 is superior to the call option-based bound of \mathbf{A}_{TC} , where we use the bound of Theorem 7 on the price of call options. This is summarized in the following lemma.

Lemma 4. *If $T > 1$, then for any $\eta \in [1, \eta_{max}]$,*

$$\left(K^\eta + \eta e^{\eta(\eta-1)Q} \ln \frac{M}{K} \right)^{\frac{1}{\eta}} - K < T \left[\left(K^\eta + e^{\eta(\eta-1)Q} \left(1 - \left(\frac{K}{M} \right)^\eta \right) \right)^{\frac{1}{\eta}} - K \right].$$

Finally, we observe that for $\eta = 1$, the r.h.s. of the lemma is smaller than T , the bound of \mathbf{ATS} .³

Comparison to BCRP Approaches. The problem of pricing lookback options may be viewed as having a reference class of $n = T + 1$ alternative strategies, namely, holding cash or selling the stock on some day t , for $1 \leq t \leq T$. Working with this simple reference class, we can achieve better regret bounds than those given w.r.t. superior benchmarks such as the BCRP. Consider a naïve *buy and hold* algorithm that initially divides capital equally between the alternative strategies, performing no further action. This algorithm clearly has a regret bound of $\ln n$ w.r.t. the log returns of any of the alternatives. In comparison, the universal portfolio's (optimal) bound w.r.t. the BCRP is $\frac{n-1}{2} \ln 2T + \ln \frac{\Gamma(1/2)^n}{\Gamma(n/2)} + o(1)$ (see, e.g., [4]). In the case of lookback options, we get an $\Omega(T)$ regret bound, which grows arbitrarily large as the trading frequency increases. The problematic dependence on T persists even if n is small, for example, in the case of call options, where $n = 2$ (the alternatives are holding cash or holding the stock). A different algorithm presented by Hazan and Kale [14] has regret bounds which depend on the quadratic variability⁴ of the single period returns. Their bound is a great improvement over the previous bound, under realistic conditions where the quadratic variability is much smaller than T . However, it is still lower bounded by the naïve bound of $\ln n$, for every n . Therefore, the regret bounds w.r.t. the BCRP cannot be used directly to achieve interesting upper bounds on the prices of lookback options or call options.

6 Empirical Results

In order to examine our bounds empirically, we consider the S&P 500 index data for the years 1950-2010. (The results are plotted in Figure 1) We computed a price for a 1-year lookback using the price-oriented rule and bound (see Sub-section 4.2), with $K = 1$, $R = 0.15$, $Q = 0.1$, and $M = 1.5$. These R , Q , and M values hold for all years but two in our test. In addition, for each year we computed the payoff of the lookback option, and ran our algorithm with the price-oriented rule and computed its profit. In our calculations we used a single value of η , which is the value that minimized our price bound. The BSM pricing [7] was computed for comparison, using the average volatility for the whole data, and assuming zero risk-free interest. Note that the stock prices for each year were normalized so that $S_0 = 1$ at the beginning of the year.

An important issue is to compare the net payoff to the lookback option price. The net payoff is the difference between the payoff to the option holder (always non-negative) and the profit (or loss) the algorithm made in trading. We observe

³ It is easy to observe that our bound outperforms the bound derived using [10] for small values of Q and large values of K .

⁴ Their definition is different from the quadratic variation Q which we use. In particular, their variability is centered around the average value of the daily returns, and is similar to the variance of random variables.

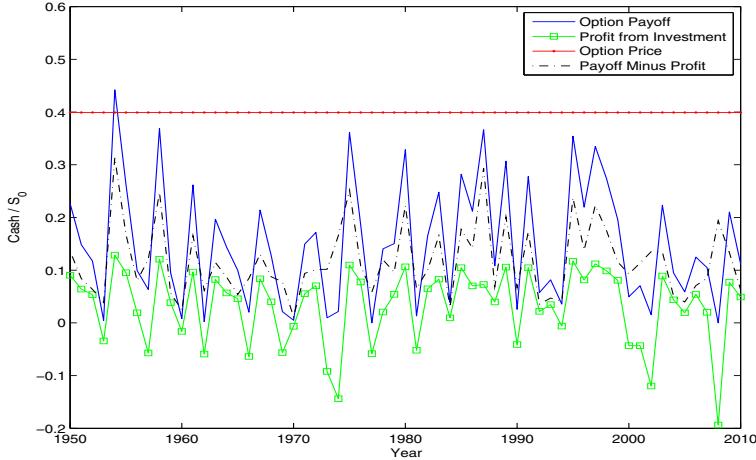


Fig. 1. The plot gives a breakdown of the option writer’s cash flow in terms of option payoff, option price, and profit from the algorithm’s trading. Data is calculated for 1-year lookbacks on the S&P 500 index for the years 1950–2010, with $K = 1$, $R = 0.15$, $Q = 0.1$, and $M = 1.5$. The option writer makes a profit if the payoff minus profit line is below the option price line. (Note that the “hindsight” empirical price is 0.314 while our bound gives 0.399.) The calculated BSM price is 0.129.

that our lookback option price dominates the net payoff for every year, with our option price at about 0.4 and the maximal net payoff at about 0.3.

We conclude with a comparison of our two different bound derivations for different values of the strike price K . One derivation is the price-oriented bound of Theorem 5, and the other is the competitive ratio-based bound of Theorem 6. For the latter bound we use the optimal one-way trading rule of [12]. We set $R = 0.2$, $Q = 0.2$, and $M = 2$. In addition, we set $T = 252$ trading days and a lower bound of 0.5 on the stock price for the optimal one-way trading bound. For the price-oriented rule, we give both the bound with optimal η and the bound with $\eta = 1$. For the optimal one-way trading rule, $\eta = 1$ was always the best choice for this setting.

It can be seen that for the price-oriented rule, working with the optimal η is better than merely one-way trading ($\eta = 1$). For lower values of K , the optimal one-way trading rule does better, whereas for higher values of K , the price-oriented rule does better.

K	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8
Opt. One-Way Trading Rule	0.602	0.541	0.481	0.421	0.361	0.301	0.241	0.180	0.120
Price-Oriented Rule	0.687	0.586	0.493	0.408	0.330	0.259	0.195	0.137	0.086
Price-Oriented Rule, $\eta = 1$	0.693	0.598	0.511	0.431	0.357	0.288	0.223	0.163	0.105

References

1. Black, F., Scholes, M.: The pricing of options and corporate liabilities. *Journal of Political Economy* 81(3), 637–654 (1973)
2. Blum, A., Kalai, A.: Universal portfolios with and without transaction costs. *Machine Learning* 35(3), 193–205 (1999); special issue for COLT 1997
3. Borodin, A., El-Yaniv, R., Gogan, V.: Can we learn to beat the best stock. *Journal of Artificial Intelligence Research* 21, 579–594 (2004)
4. Cesa-Bianchi, N., Lugosi, G.: *Prediction, Learning, and Games*. Cambridge University Press, New York (2006)
5. Cesa-Bianchi, N., Mansour, Y., Stoltz, G.: Improved second-order bounds for prediction with expert advice. *Machine Learning* 66(2-3), 321–352 (2007)
6. Chou, A., Cooperstock, J., El-Yaniv, R., Klugerman, M., Leighton, T.: The statistical adversary allows optimal money-making trading strategies. In: *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 467–476 (1995)
7. Conze, A., Viswanathan: Path dependent options: the case of lookback options. *Journal of Finance* 46(5), 1893–1907 (1991)
8. Cover, T.M.: Universal portfolios. *Mathematical Finance* 1(1), 1–29 (1991)
9. Cover, T.M., Ordentlich, E.: Universal portfolios with side information. *IEEE Transactions on Information Theory* 42(2), 348–363 (1996)
10. Dawid, A.P., de Rooij, S., Shafer, G., Shen, A., Vereshchagin, N., Vovk, V.: Insuring against loss of evidence in game-theoretic probability. *Statistics and Probability Letters* 81(1), 157–162 (2011)
11. DeMarzo, P., Kremer, I., Mansour, Y.: Online trading algorithms and robust option pricing. In: *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, pp. 477–486. ACM, New York (2006)
12. El-Yaniv, R., Fiat, A., Karp, R.M., Turpin, G.: Optimal search and one-way trading online algorithms. *Algorithmica* 30(1), 101–139 (2001); a preliminary version appeared in FOCS (1992)
13. Györfi, L., Lugosi, G., Udina, F.: Nonparametric kernel-based sequential investment strategies. *Mathematical Finance* 16(2), 337–357 (2006)
14. Hazan, E., Kale, S.: On stochastic and worst-case models for investing. In: *Advances in Neural Information Processing Systems (NIPS)*, vol. 22, pp. 709–717 (2009)
15. Helmbold, D.P., Schapire, R.E., Singer, Y., Warmuth, M.K.: On-line portfolio selection using multiplicative updates. In: *Proc. 13th International Conference on Machine Learning*, pp. 243–251. Morgan Kaufmann, San Francisco (1996)
16. Kakade, S., Kearns, M.J., Mansour, Y., Ortiz, L.E.: Competitive algorithms for vwap and limit order trading. In: *ACM Conference on Electronic Commerce*, pp. 189–198. ACM, New York (2004)
17. Kalai, A., Vempala, S.: Efficient algorithms for universal portfolios. *Journal of Machine Learning Research* 3, 423–440 (2002)
18. Lorenz, J., Panagiotou, K., Steger, A.: Optimal algorithms for k-search with application in option pricing. *Algorithmica* 55(2), 311–328 (2009)
19. Merton, R.C.: Theory of rational option pricing. *Bell Journal of Economics and Management Science* 4(1), 141–183 (1973)
20. Ordentlich, E., Cover, T.M.: The cost of achieving the best portfolio in hindsight. *Mathematics of Operations Research* 23(4), 960–982 (1998)
21. Singer, Y.: Switching portfolios. In: *UAI*, pp. 488–495 (1998)
22. Vovk, V., Watkins, C.: Universal portfolio selection. In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pp. 12–23 (1998)

Making Online Decisions with Bounded Memory

Chi-Jen Lu¹ and Wei-Fu Lu²

¹ Institute of Information Science, Academia Sinica, Taipei, Taiwan
`cjlu@iis.sinica.edu.tw`

² Department of Computer Science and Information Engineering,
Asia University, Taiwan
`weifu.lu@gmail.com`

Abstract. We study the online decision problem in which there are T steps to play and n actions to choose. For this problem, several algorithms achieve an optimal regret of $O(\sqrt{T \ln n})$, but they all require about T^n states, which one may not be able to afford when n and T are very large. We are interested in such large scale problems, and we would like to understand what an online algorithm can achieve with only a bounded number of states. We provide two algorithms, both with m^{n-1} states, for a parameter m , which achieve regret of $O(m + (T/m) \ln(mn))$ and $O(n\sqrt{m} + T/\sqrt{m})$, respectively. We also show that any online algorithm with m^{n-1} states must suffer a regret of $\Omega(T/m)$, which is close to what our algorithms achieve.

1 Introduction

In our daily life, there are situations in which we have to make repeated decisions without knowledge of the future. This can be modeled as the so-called online decision problem. In this problem, there are T steps to play and n actions to choose, for some parameters T and n . At each step, we have to choose an action to play before knowing its loss. After choosing the action, we suffer a loss corresponding to that action and get to know the loss vector, consisting of the loss of each action, of that step. We would like to have an online algorithm with a small regret, which is the difference between the total loss of the algorithm and that of the best fixed action in hindsight. This problem has become an important topic in machine learning research, and it has found applications in other areas such as algorithm design, game theory, optimization, and statistics. More information can be found in the survey papers such as [9, 2, 4] or the book [5], and some recent works include [10, 11, 3, 7].

For simplicity of presentation, we will focus in this paper on the special case of binary loss values; it is not hard to extend our results to the general case. For the case of binary loss values, several algorithms are known to achieve a regret of $O(\sqrt{T \ln n})$, which is optimal as a matching lower bound is also known (see e.g. [5]). One well-known algorithm is the multiplicative update algorithm [11, 8] which works as follows. At each step, the algorithm maintains n weights (w_1, \dots, w_n) , one for each action, and plays action i with probability w_i/Z where Z is the sum of the n weights. After receiving the loss vector (ℓ_1, \dots, ℓ_n) of that

step, with $\ell_i \in \{0, 1\}$ being the loss of action i , it updates the i 'th weight, for each i , by multiplying it by the factor $(1 - \eta)^{\ell_i}$ for some learning rate η . Although this algorithm achieves optimal regret, it is memory demanding in the following sense. Note that the algorithm needs to keep a separate weight for each action, and each weight can take T different values throughout the T steps, which amounts to a total of T^n possible values for it to keep track of. We are interested in large scale problems arising from complex situations, in which there are a large number of actions to choose and a large number of steps to play. In such situations, the algorithm will require a huge amount of memory, which one may not be able to afford. In fact, other algorithms which achieve the same regret bound all have this memory problem. Therefore, one may wonder if it is possible to have a more memory-efficient algorithm which can achieve a regret close to $O(\sqrt{T \ln n})$. More generally, given a memory bound, what is the lowest regret achievable by an online algorithm?

We model a memory-bounded online algorithm by a finite state automata as follows. It has a finite set S of states, with each state associated with a probability distribution, and it also has a state transition function τ , which takes a state together with a loss vector and produces a new state. At each step, the algorithm is in some state $s \in S$ and it plays according to the probability distribution associated with that state. After receiving the loss vector ℓ of that step, it then transits to the new state $\tau(s, \ell)$, and the process continues. Here, we only consider the case that the transition function τ is deterministic but the action chosen at each state is randomized, which is the case of most previous algorithms, such as the multiplicative update algorithm [11, 8] and the gradient descend algorithm [13].

In this paper, we provide two algorithms with m^{n-1} states, for a parameter m . Our first algorithm is based on the multiplicative update algorithm and achieves a regret of $O(m + (T/m) \ln(mn))$, which is $O((T/m) \ln(mn))$ when $m = O(\sqrt{T})$ (for a larger m , one can simply use $O(\sqrt{T})$ states and ignore additional ones). This means that the algorithm using $T^{(n-1)/2}$ states can already achieve a regret of $O(\sqrt{T} \log(Tn))$ which is close to the optimum. Our second algorithm is based on the gradient descend algorithm and achieves a regret of $O(n\sqrt{m} + T/\sqrt{m})$, which is $O(T/\sqrt{m})$ when $m = O(T/n)$ (again, one can ignore additional states for a larger m). Note that the regrets achieved by our two algorithms are incomparable, but the regret of our first algorithm is smaller when $n < e^{c\sqrt{m}}/m$ for some constant c . Our two algorithms are natural modifications of existing algorithms, and one may wonder if a more sophisticated modification or a completely new algorithm can actually achieve a much smaller regret. We provide a negative answer to this question by showing that any online algorithm with m^{n-1} states must suffer a regret of $\Omega(T/m)$, which is close to what our two algorithms achieve.

A line of works related to ours is that on universal prediction of individual sequences; see [12, 6] and the reference therein. The most relevant result seems to be that of Meron and Feder [12], who considered the problem of repeatedly predicting the next bit of an individual binary sequence. This corresponds to

our case in which there are only two actions and the loss vector at each step is either $(0, 1)$ or $(1, 0)$ (loss 1 for incorrect prediction). They provided an algorithm with m states which achieves a regret of $T/(2m)$ and they also showed a matching lower bound. Moreover, they considered measuring regret against a larger comparison class consisting of order- L Markov predictors. Each predictor in the class is a special type of online algorithm with 2^L states, which has the L most recent bits in the sequence as its current state. They provided an online algorithm with m^{2^L} states which achieves a regret of $O(T/m)$, and they showed an $\Omega(T/(m2^L))$ regret lower bound for any such algorithm. One may try to relate their results to ours by seeing each predictor in their comparison class as an action in our case. Note that even if one restricts the prediction at each state to be deterministic, there are 2^{2^L} such predictors, and using this number as n in our results will give much larger regret bounds than theirs. The reason for this discrepancy is that these actions (Markov predictors) are in fact correlated in the sense that the loss received by one has dependence on those received by others, which is utilized by [12] to obtain a smaller regret, while our large regret lower bound relies on the independence among actions, which gives us freedom to choose more adversarial loss vectors. This is related to works on structured expert classes, in which structures of experts (or actions) may be exploited to achieve smaller regret; see for example Chapter 5 in [5] for more details.

Finally, we would like to remark that our setting seems related to that of streaming algorithms, which are also concerned about the memory issue but with the goal of computing some functions (approximately), instead of making repeated decisions for minimizing the regret. It may be possible to use ideas from streaming algorithms to design randomized online algorithms (with randomized state transition functions) with a smaller regret. However, we have no success so far, as the existing streaming algorithms for related functions we are aware of all seem to have large approximation errors which result in regret bounds larger than those of our two algorithms. We leave the exploration of this direction as our future work.

2 Preliminaries

Notations and definitions. Let \mathbb{N} denote the set of nonnegative integers and \mathbb{Z}^+ the set of positive integers. For $n \in \mathbb{Z}^+$, let $[n]$ denote the set $\{1, 2, \dots, n\}$. For any real number r , let $\lfloor r \rfloor$ denote the floor of r , which is the largest integer not exceeding r . Given an n -dimensional vector p and an index $i \in [n]$, let p_i denote the component of p in the i 'th dimension. Given two n -dimensional vectors p and q , their inner product is defined by $\langle p, q \rangle = \sum_{i \in [n]} p_i q_i$.

Distributions. We consider probability distributions over a set of n elements, which can also be seen as n -dimensional vectors in $[0, 1]^n$. Let \mathcal{P} denote the set which contains all such distributions:

$$\mathcal{P} = \left\{ (p_1, \dots, p_n) \in [0, 1]^n : \sum_{i \in [n]} p_i = 1 \right\}.$$

Given two distributions $p, q \in \mathcal{P}$, their L_2 distance is defined by $\|p - q\|_2 = \sqrt{\sum_{i \in [n]} (p_i - q_i)^2}$ and their L_1 distance by $\|p - q\|_1 = \sum_{i \in [n]} |p_i - q_i|$. We will need the following simple fact.

Fact 1. For any $p, q \in \mathcal{P}$, $\|p - q\|_1 = 2 \sum_{i \in [n]: p_i \geq q_i} (p_i - q_i)$.

Online decision problem. We study the following online decision problem, in which there are T steps to play, there is a fixed set $[n]$ of actions to choose at each step, and the loss values are all binary. At step $t \in [T]$, an online algorithm \mathcal{A} must choose an action to play according to some distribution $p^{(t)} \in \mathcal{P}$. After that, \mathcal{A} receives a loss vector $\ell^{(t)} \in \{0, 1\}^n$ and suffers an expected loss of $\langle p^{(t)}, \ell^{(t)} \rangle = \sum_{i \in [n]} p_i^{(t)} \ell_i^{(t)}$ at that step. The standard way for evaluating the performance of \mathcal{A} is to compare its total expected loss to that of the best offline algorithm which can see all the loss vectors but has to play a fixed action (which has the same effect as a fixed probability distribution) for all T steps. The difference is called the regret of \mathcal{A} , denoted as $R_{\mathcal{A}}$, which is

$$\sum_{t \in [T]} \langle p^{(t)}, \ell^{(t)} \rangle - \min_{i \in [n]} \sum_{t \in [T]} \ell_i^{(t)} = \sum_{t \in [T]} \langle p^{(t)}, \ell^{(t)} \rangle - \min_{p \in \mathcal{P}} \sum_{t \in [T]} \langle p, \ell^{(t)} \rangle.$$

In this paper, we consider the constraint that the online algorithm has only a bounded number of states, which can be seen as a finite automata as follows. Each such algorithm has a finite set S of states, with each state associated with some probability distribution in \mathcal{P} , and it has some state transition function $\tau : S \times \{0, 1\}^n \rightarrow S$. At each time step, the algorithm is in some state $s \in S$, and it plays the probability distribution associated with s . Then after seeing a loss vector ℓ , the algorithm transits to the state $\tau(s, \ell)$, and the process continues.

3 Our First Algorithm

In this section, we present our first algorithm, and the result is the following.

Theorem 1. For any $m, n \in \mathbb{Z}^+$ such that $n \geq 2$ and $m \geq 6 \ln(mn)$, there exists an algorithm \mathcal{A}_1 with at most m^{n-1} states which achieves a regret of $O(m + (T/m) \ln(nm))$.

The algorithm \mathcal{A}_1 is based on the multiplicative update algorithm [11, 8], but now with m^{n-1} states, the algorithm can only play according to that number of distributions. Each state of \mathcal{A}_1 consists of n counters, one for each action. Each counter is used to keep track of the accumulated loss received so far by the corresponding action, but we only allow each counter to go up to $m/3$ in order to have a small number of states (assume for ease of presentation that $m/3$ is an integer). At step $t \in [T]$, the algorithm starts from a state $s^{(t)} = (s_1^{(t)}, \dots, s_n^{(t)})$ and then transits to some state $s^{(t+1)} = (s_1^{(t+1)}, \dots, s_n^{(t+1)})$ after receiving a loss vector $\ell^{(t)}$. The multiplicative update algorithm would just set $s^{(t+1)}$ to $\hat{s}^{(t+1)} = s^{(t+1)} + \ell^{(t)}$, but we need to perform some rounding in order to keep each counter bounded. Formally, our algorithm is described as follows.

Algorithm \mathcal{A}_1 : It starts from the state $s^{(1)} = (0, \dots, 0)$. At step $t \in [T]$, it plays according to the distribution $p^{(t)} = (p_1^{(t)}, \dots, p_n^{(t)})$, where

$$p_i^{(t)} = (1 - \eta)^{s_i^{(t)}} / Z^{(t)}, \quad \text{with } Z^{(t)} = \sum_{j \in [n]} (1 - \eta)^{s_j^{(t)}},$$

and then after receiving the loss vector $\ell^{(t)}$, it updates its state as follows.

- Let $\hat{s}^{(t+1)} = s^{(t)} + \ell^{(t)}$. That is, $\hat{s}_i^{(t+1)} = s_i^{(t)} + \ell_i^{(t)}$ for every $i \in [n]$.
- If $\hat{s}_i^{(t+1)} \geq 1$ for every $i \in [n]$, then let $s_i^{(t+1)} = \hat{s}_i^{(t+1)} - 1$ for every $i \in [n]$. Otherwise, let $s_i^{(t+1)} = \min\{\hat{s}_i^{(t+1)}, m/3\}$ for every $i \in [n]$.

Let us bound the number of states of \mathcal{A}_1 . Note that for any t , at least one of the n counters in $s^{(t)}$ has the value 0, while each counter takes an integral value in the range from 0 to $m/3$. Thus, the total number of states is at most

$$\binom{n}{1} \cdot (1 + m/3)^{n-1} = \left(n^{1/(n-1)}\right)^{n-1} \cdot (1 + m/3)^{n-1} \leq m^{n-1},$$

for $n \geq 2$ and $m \geq 6$ (which follows from the condition that $m \geq 6 \ln(mn)$).

Next, let us analyze the regret of \mathcal{A}_1 . Let action i^* be the best fixed action which an offline algorithm can choose in hindsight. Then the regret of \mathcal{A}_1 is

$$R_{\mathcal{A}_1} = \sum_{t \in [T]} \ell_{\mathcal{A}_1}^{(t)} - \sum_{t \in [T]} \ell_{i^*}^{(t)}, \quad \text{where } \ell_{\mathcal{A}_1}^{(t)} = \sum_{i \in [n]} p_i^{(t)} \ell_i^{(t)}.$$

Our key lemma is the following, which gives a bound on the regret of \mathcal{A}_1 at each step.

Lemma 1. *For any $t \in [T]$ and any $\eta \in (0, 1/2)$,*

$$\ell_{\mathcal{A}_1}^{(t)} - \ell_{i^*}^{(t)} \leq \frac{1}{\eta} \left(\ln \frac{1}{p_{i^*}^{(t)}} - \ln \frac{1}{p_{i^*}^{(t+1)}} + \eta^2 + \eta n e^{-\eta m/3} \right).$$

Using this lemma and choosing $\eta = \frac{3}{m} \ln(nm)$, which is less than $1/2$ with $m \geq 6 \ln(mn)$, we can bound the total regret of \mathcal{A}_1 over T steps by

$$\begin{aligned} \sum_{t \in [T]} (\ell_{\mathcal{A}_1}^{(t)} - \ell_{i^*}^{(t)}) &\leq \sum_{t \in [T]} \frac{1}{\eta} \left(\ln \frac{1}{p_{i^*}^{(t)}} - \ln \frac{1}{p_{i^*}^{(t+1)}} + \eta^2 + \eta n e^{-\eta m/3} \right) \\ &\leq \frac{1}{\eta} \ln \frac{1}{p_{i^*}^{(1)}} + \left(\eta + n e^{-\eta m/3} \right) T \\ &= \frac{m \ln n}{3 \ln(nm)} + \left(\frac{3 \ln(nm)}{m} + \frac{1}{m} \right) T \\ &\leq \frac{m}{3} + \frac{4 \ln(nm)}{m} T. \end{aligned}$$

To complete the proof of Theorem 1, it remains to prove Lemma 1, which we do next. Consider the distribution $\hat{p}^{(t+1)} = (\hat{p}_1^{(t+1)}, \dots, \hat{p}_n^{(t+1)})$, where

$$\hat{p}_i^{(t+1)} = (1 - \eta)^{\hat{s}_i^{(t+1)}} / \hat{Z}^{(t+1)}, \quad \text{with } \hat{Z}^{(t+1)} = \sum_{j \in [n]} (1 - \eta)^{\hat{s}_j^{(t+1)}}.$$

Let us write

$$\begin{aligned} \ln \frac{1}{p_{i^*}^{(t+1)}} - \ln \frac{1}{p_{i^*}^{(t)}} &= \left(\ln \frac{1}{\hat{p}_{i^*}^{(t+1)}} - \ln \frac{1}{p_{i^*}^{(t)}} \right) + \left(\ln \frac{1}{p_{i^*}^{(t+1)}} - \ln \frac{1}{\hat{p}_{i^*}^{(t+1)}} \right) \\ &= \ln \frac{p_{i^*}^{(t)}}{\hat{p}_{i^*}^{(t+1)}} + \ln \frac{\hat{p}_{i^*}^{(t+1)}}{p_{i^*}^{(t+1)}}. \end{aligned} \quad (1)$$

We bound the two terms in (1) by the following two lemmas.

Lemma 2. For any $t \in [T]$ and any $\eta \in (0, 1/2)$, $\ln \frac{p_{i^*}^{(t)}}{\hat{p}_{i^*}^{(t+1)}} \leq -\eta(\ell_{A_1}^{(t)} - \ell_{i^*}^{(t)}) + \eta^2$.

Lemma 3. For any $t \in [T]$, $\ln \frac{\hat{p}_{i^*}^{(t+1)}}{p_{i^*}^{(t+1)}} \leq \eta n e^{-\eta m/3}$.

From these two lemmas, we have

$$\ln \frac{1}{p_{i^*}^{(t+1)}} - \ln \frac{1}{p_{i^*}^{(t)}} \leq -\eta(\ell_{A_1}^{(t)} - \ell_{i^*}^{(t)}) + \eta^2 + \eta n e^{-\eta m/3},$$

which implies Lemma 1. It remains to prove the two lemmas, which we do next.

Proof. (of Lemma 2) This follows from existing analyses for the multiplicative update algorithm, but we provide the proof here for completeness. Note that

$$\frac{p_{i^*}^{(t)}}{\hat{p}_{i^*}^{(t+1)}} = \frac{(1 - \eta)^{s_{i^*}^{(t)}} / Z^{(t)}}{(1 - \eta)^{\hat{s}_{i^*}^{(t+1)}} / \hat{Z}^{(t+1)}} = \frac{1}{(1 - \eta)^{\ell_{i^*}^{(t)}}} \cdot \frac{\hat{Z}^{(t+1)}}{Z^{(t)}}, \quad (2)$$

since $\hat{s}_{i^*}^{(t+1)} = s_{i^*}^{(t)} + \ell_{i^*}^{(t)}$ by definition. The first factor in (2) is at most $e^{\eta \ell_{i^*}^{(t)} + \eta^2}$ since $\eta \in (0, 1/2)$ and $\ell_{i^*}^{(t)} \in \{0, 1\}$, while the second factor in (2) is

$$\sum_{i \in [n]} \frac{(1 - \eta)^{s_i^{(t)} + \ell_i^{(t)}}}{Z^{(t)}} = \sum_{i \in [n]} p_i^{(t)} (1 - \eta)^{\ell_i^{(t)}} = \sum_{i \in [n]} p_i^{(t)} (1 - \eta \ell_i^{(t)}) = 1 - \eta \ell_{A_1}^{(t)} \leq e^{-\eta \ell_{A_1}^{(t)}}.$$

As a result, we have

$$\ln \frac{p_{i^*}^{(t)}}{\hat{p}_{i^*}^{(t+1)}} \leq \ln \left(e^{\eta \ell_{i^*}^{(t)} + \eta^2} \cdot e^{-\eta \ell_{A_1}^{(t)}} \right) = \eta \ell_{i^*}^{(t)} + \eta^2 - \eta \ell_{A_1}^{(t)} = -\eta(\ell_{A_1}^{(t)} - \ell_{i^*}^{(t)}) + \eta^2.$$

□

Proof. (of Lemma 3) First note that if $\hat{s}_i^{(t+1)} \geq 1$ for every $i \in [n]$, then $\hat{p}_i^{(t+1)} = p_i^{(t+1)}$ for every $i \in [n]$ and we have the lemma. Thus, let us assume that $\hat{s}_i^{(t+1)} = 0$ for some $i \in [n]$. Note that

$$\frac{\hat{p}_{i^*}^{(t+1)}}{p_{i^*}^{(t+1)}} = \frac{(1-\eta)^{\hat{s}_{i^*}^{(t+1)}}/\hat{Z}^{(t+1)}}{(1-\eta)^{s_{i^*}^{(t+1)}}/Z^{(t+1)}} \leq \frac{Z^{(t+1)}}{\hat{Z}^{(t+1)}}$$

since $s_{i^*}^{(t+1)} \leq \hat{s}_{i^*}^{(t+1)}$. Let n_v be the number of $i \in [n]$ such that $\hat{s}_i^{(t+1)} = v$, and recall that we have $n_0 \geq 1$ because $\hat{s}_i^{(t+1)} = 0$ for some $i \in [n]$. Then we have $\hat{Z}^{(t+1)} = \sum_{v=0}^{m/3+1} n_v(1-\eta)^v$ and $Z^{(t+1)} = \sum_{v=0}^{m/3} n_v(1-\eta)^v + n_{m/3+1}(1-\eta)^{m/3}$, which implies that

$$\frac{Z^{(t+1)}}{\hat{Z}^{(t+1)}} = \frac{u + r(1-\eta)^{m/3}}{u + r(1-\eta)^{m/3+1}} = 1 + \frac{\eta r(1-\eta)^{m/3}}{u + r(1-\eta)^{m/3+1}},$$

for some $u \geq n_0(1-\eta)^0 \geq 1$ and $r = n_{m/3+1} \leq n - 1$. As a result, we have

$$\ln \frac{\hat{p}_{i^*}^{(t+1)}}{p_{i^*}^{(t+1)}} \leq \ln \left(1 + \eta r(1-\eta)^{m/3} \right) \leq \eta r(1-\eta)^{m/3} \leq \eta n e^{-\eta m/3}.$$

□

4 Our Second Algorithm

In this section, we present our second algorithm, and the result is the following.

Theorem 2. *For any $n, m \in \mathbb{Z}^+$ with $n \geq 2$ and $m \geq 14$, there is an algorithm \mathcal{A}_2 with at most m^{n-1} states which achieves a regret of $O(n\sqrt{m} + T/\sqrt{m})$.*

The algorithm \mathcal{A}_2 is based on the gradient descend algorithm [13], but now with only m^{n-1} states, \mathcal{A}_2 can only play according to that number of probability distributions. The idea is to reduce the resolution of probability values, so that each is always a multiple of some value $\delta = 1/(kn)$ with $k \in \mathbb{Z}^+$. Let $\mathcal{Q}_\delta \subseteq \mathcal{P}$ denote such a set of probability distributions:

$$\mathcal{Q}_\delta = \left\{ (x_1\delta, \dots, x_n\delta) : x_i \in \mathbb{N} \text{ for } i \in [n] \text{ and } \sum_{i \in [n]} x_i\delta = 1 \right\}. \quad (3)$$

The following lemma provides a bound on the size of \mathcal{Q}_δ .

Lemma 4. *For any $\delta = 1/(kn)$ with $k \in \mathbb{Z}^+$, $|\mathcal{Q}_\delta| = \binom{kn+n-1}{n-1}$.*

Proof. Note that the set \mathcal{Q}_δ has the same size as the following set:

$$\left\{ (x_1, \dots, x_n) \in \mathbb{N}^n : \sum_{i \in [n]} \frac{x_i}{kn} = 1 \right\} = \left\{ (x_1, \dots, x_n) \in \mathbb{N}^n : \sum_{i \in [n]} x_i = kn \right\}.$$

Thus, we have $|\mathcal{Q}_\delta| = \binom{kn+n-1}{n-1}$.

□

Note that when $n \geq 2$ and $m \geq 14$, by choosing $k = \lfloor m/7 \rfloor \geq 2$, we have

$$|\mathcal{Q}_\delta| \leq \left(\frac{e(kn + n - 1)}{n - 1} \right)^{n-1} \leq (e(2k + 1))^{n-1} \leq (7k)^{n-1} \leq m^{n-1}. \quad (4)$$

Then we let the probability distributions played by our algorithm \mathcal{A}_2 all come from such a set \mathcal{Q}_δ , with

$$\delta = 1/(\lfloor m/7 \rfloor n).$$

The following lemma shows that playing only distributions in \mathcal{Q}_δ does not affect the regret too much as any $\hat{p} \in \mathcal{P}$ has some close-by distribution in \mathcal{Q}_δ .

Lemma 5. *For any $\hat{p} \in \mathcal{P}$, there exists some $p \in \mathcal{Q}_\delta$ such that $|\hat{p}_i - p_i| \leq \delta$ for every $i \in [n]$.*

Proof. Note that any $\hat{p} \in \mathcal{P}$ can be expressed as $\hat{p} = (x_1\delta + \varepsilon_1, \dots, x_n\delta + \varepsilon_n)$, where $x_i \in \mathbb{N}$ and $0 \leq \varepsilon_i < \delta$ for every $i \in [n]$. To obtain $p \in \mathcal{Q}_\delta$, with each component a multiple of δ , we first take out all those ε_i 's from \hat{p} to obtain a vector $\bar{p} = (x_1\delta, \dots, x_n\delta)$, which is not yet a probability distribution. Note that the sum $\varepsilon = \sum_{i \in [n]} \varepsilon_i$ is a multiple of δ , because $(\sum_{i \in [n]} x_i)\delta + \varepsilon = \sum_{i \in [n]} \hat{p}_i = 1 = (\lfloor m/7 \rfloor n)\delta$ is a multiple of δ . So we divide ε into $r = \varepsilon/\delta$ parts of equal value δ , add them respectively into the first r components of \bar{p} , and let p be the resulting vector. Then p is a probability distribution in \mathcal{Q}_δ and satisfies the condition that $|\hat{p}_i - p_i| \leq \delta$ for every $i \in [n]$. \square

For any $\hat{p} \in \mathcal{P}$, let us write $\mathcal{Q}_\delta(\hat{p})$ for that close-by distribution guaranteed by Lemma 5. The distribution our algorithm plays at each step is $\mathcal{Q}_\delta(\hat{p})$ for some distribution \hat{p} obtained by doing a gradient descend with a projection. Formally, our algorithm is described as follows.

Algorithm \mathcal{A}_2 : It starts with an arbitrary distribution $p^{(1)} \in \mathcal{P}$. At step $t \in [T]$, it plays according to the distribution $p^{(t)}$, and then does the following update after receiving the loss vector $\ell^{(t)}$:

- Let $\hat{p}^{(t+1)} = \Pi_{\mathcal{P}}(p^{(t)} - \eta\ell^{(t)}) = \arg \min_{p \in \mathcal{P}} \|p - (p^{(t)} - \eta\ell^{(t)})\|_2$.
- Let $p^{(t+1)} = \mathcal{Q}_\delta(\hat{p}^{(t+1)})$.

According to the bound in (4), the size of \mathcal{Q}_δ is at most m^{n-1} , so algorithm \mathcal{A}_2 indeed requires only that number of states. To analyze the regret of \mathcal{A}_2 , we rely on the following lemma which bounds its regret at each step. Let p^* denote the best fixed distribution in \mathcal{P} which an offline algorithm can choose in hindsight.

Lemma 6. *For any $t \in [T]$,*

$$\langle p^{(t)} - p^*, \ell^{(t)} \rangle \leq \frac{1}{2\eta} \left(\|p^{(t)} - p^*\|_2^2 - \|p^{(t+1)} - p^*\|_2^2 \right) + O\left(\eta n + \frac{1}{\eta mn}\right).$$

Using this lemma and choosing $\eta = 1/(n\sqrt{m})$, we can bound the total regret of \mathcal{A}_2 over T steps as

$$\begin{aligned} \sum_{t \in [T]} \left\langle p^{(t)} - p^*, \ell^{(t)} \right\rangle &\leq \frac{1}{2\eta} \left\| p^{(1)} - p^* \right\|_2^2 + O\left(\eta n + \frac{1}{\eta mn}\right) T \\ &= O\left(n\sqrt{m} + \frac{T}{\sqrt{m}}\right), \end{aligned}$$

since $\left\| p^{(1)} - p^* \right\|_2^2 \leq 2$. This proves Theorem 2.

It remains to prove Lemma 6. For this, let us write

$$\begin{aligned} \left\| p^{(t+1)} - p^* \right\|_2^2 &= \left\| (\hat{p}^{(t+1)} - p^*) + (p^{(t+1)} - \hat{p}^{(t+1)}) \right\|_2^2 \\ &= \left\| \hat{p}^{(t+1)} - p^* \right\|_2^2 + \left\| p^{(t+1)} - \hat{p}^{(t+1)} \right\|_2^2 + 2 \left\langle \hat{p}^{(t+1)} - p^*, p^{(t+1)} - \hat{p}^{(t+1)} \right\rangle. \quad (5) \end{aligned}$$

Next, we bound the three terms in (5). Following the analysis of Zinkevich [13], we can bound the first term in (5) as

$$\begin{aligned} \left\| \hat{p}^{(t+1)} - p^* \right\|_2^2 &= \left\| \Pi_{\mathcal{P}}(p^{(t)} - \eta\ell^{(t)}) - p^* \right\|_2^2 \\ &\leq \left\| p^{(t)} - \eta\ell^{(t)} - p^* \right\|_2^2 \\ &= \left\| (p^{(t)} - p^*) - \eta\ell^{(t)} \right\|_2^2 \\ &= \left\| p^{(t)} - p^* \right\|_2^2 - 2\eta \left\langle p^{(t)} - p^*, \ell^{(t)} \right\rangle + \eta^2 \left\| \ell^{(t)} \right\|_2^2 \\ &\leq \left\| p^{(t)} - p^* \right\|_2^2 - 2\eta \left\langle p^{(t)} - p^*, \ell^{(t)} \right\rangle + \eta^2 n. \end{aligned}$$

Moreover, according to Lemma 5, the second term in (5) is

$$\left\| p^{(t+1)} - \hat{p}^{(t+1)} \right\|_2^2 = \sum_{i \in [n]} \left(p_i^{(t+1)} - \hat{p}_i^{(t+1)} \right)^2 \leq n\delta^2,$$

while the third term in (5) is

$$\begin{aligned} 2 \left\langle \hat{p}^{(t+1)} - p^*, p^{(t+1)} - \hat{p}^{(t+1)} \right\rangle &\leq 2 \sum_{i \in [n]} \left| \hat{p}_i^{(t+1)} - p_i^* \right| \left| p_i^{(t+1)} - \hat{p}_i^{(t+1)} \right| \\ &\leq 2 \sum_{i \in [n]} \left| \hat{p}_i^{(t+1)} - p_i^* \right| \delta \\ &\leq 2\delta \sum_{i \in [n]} \left(\hat{p}_i^{(t+1)} + p_i^* \right) \\ &= 4\delta. \end{aligned}$$

By combining the above bounds, we obtain

$$\left\| p^{(t+1)} - p^* \right\|_2^2 \leq \left\| p^{(t)} - p^* \right\|_2^2 - 2\eta \left\langle p^{(t)} - p^*, \ell^{(t)} \right\rangle + \eta^2 n + n\delta^2 + 4\delta.$$

With $\delta = O(1/(mn))$, this implies that

$$\left\langle p^{(t)} - p^*, \ell^{(t)} \right\rangle \leq \frac{1}{2\eta} \left(\left\| p^{(t)} - p^* \right\|_2^2 - \left\| p^{(t+1)} - p^* \right\|_2^2 \right) + O\left(\eta n + \frac{1}{\eta mn}\right),$$

which proves Lemma 6 and completes the proof of Theorem 2. \square

5 A Lower Bound

In this section, we show a lower bound on the regret for any algorithm with only a small number of states. The result is the following.

Theorem 3. *For any online algorithm \mathcal{A} with at most m^{n-1} states, there exists a sequence of T loss vectors with respect to which \mathcal{A} has a regret of $\Omega(T/m)$.*

Let us first describe the proof idea. Consider any online algorithm with a small number of states, which can only play according to a small number of probability distributions. We show that there exists a probability distribution q which is far from all those distributions, and we let the offline algorithm play this fixed distribution q . Consequently, no matter which state the online algorithm is in at any step, the distribution it plays is always far from the distribution q which the offline algorithm plays. This allows us to choose a loss vector for each step which makes the online algorithm suffer a significantly larger loss than the offline algorithm.

Now let us give the formal proof of the theorem. Consider any algorithm \mathcal{A} with at most m^{n-1} states, and let \mathcal{K} denote the set of probability distributions associated with these states. The following lemma guarantees the existence of a distribution q which is far from all the distributions in \mathcal{K} , where we now measure the distance of two distributions by their L_1 distance.

Lemma 7. *There exists some $q \in \mathcal{P}$ such that for any $p \in \mathcal{K}$, $\|p - q\|_1 \geq 1/(64m)$.*

Proof. We will show the existence of one such $q = (q_1, \dots, q_n)$ in the set \mathcal{Q}_δ , defined in (3), and now we choose

$$\delta = 1/(8mn).$$

The idea is to show that \mathcal{Q}_δ contains many elements while each $p \in \mathcal{K}$ is only close to a small number of them. With a small $|\mathcal{K}|$, this would then imply that some element of \mathcal{Q}_δ is far from all elements of \mathcal{K} .

First, for any $p \in \mathcal{K}$, we bound the number of elements in \mathcal{Q}_δ which are close to p . Consider the set, denoted by $\mathcal{B}(p)$, which consists of those $q \in \mathcal{Q}_\delta$ such

that $\|q - p\|_1 \leq 1/(64m)$. Note that the size of $\mathcal{B}(p)$ is at most the size of the following set:

$$\begin{aligned}\hat{\mathcal{B}}(p) &= \left\{ (x_1, \dots, x_n) \in \mathbb{N}^n : \sum_{i \in [n]} \left| \frac{x_i}{8mn} - p_i \right| \leq \frac{1}{64m} \right\} \\ &= \left\{ (x_1, \dots, x_n) \in \mathbb{N}^n : \sum_{i \in [n]} |x_i - 8mnp_i| \leq \frac{n}{8} \right\}.\end{aligned}$$

We bound the size of $\hat{\mathcal{B}}(p)$ in terms of the size of the following set:

$$\bar{\mathcal{B}}(p) = \left\{ (y_1, \dots, y_n) \in \mathbb{N}^n : \sum_{i \in [n]} y_i \leq \frac{n}{8} \right\},$$

by using the mapping $\tau : \hat{\mathcal{B}}(p) \rightarrow \bar{\mathcal{B}}(p)$ defined by $\tau(x_1, \dots, x_n) = (y_1, \dots, y_n)$ where $y_i = \lfloor |x_i - 8mnp_i| \rfloor$ for every $i \in [n]$. Clearly, $\tau(x_1, \dots, x_n) \in \bar{\mathcal{B}}(p)$ for any $(x_1, \dots, x_n) \in \hat{\mathcal{B}}(p)$, because $y_i \leq |x_i - 8mnp_i|$ for each i . Moreover, at most 2^n elements of $\hat{\mathcal{B}}(p)$ can be mapped to the same image by τ , because each y_i can come from at most two different values of x_i . This implies that

$$|\mathcal{B}(p)| \leq |\hat{\mathcal{B}}(p)| \leq 2^n \cdot |\bar{\mathcal{B}}(p)|.$$

Note that

$$|\bar{\mathcal{B}}(p)| \leq \sum_{v \leq \lfloor n/8 \rfloor} \binom{v+n-1}{n-1} = \binom{\lfloor n/8 \rfloor + n}{n} = \binom{\lfloor n/8 \rfloor + n}{\lfloor n/8 \rfloor},$$

where the first equality is due to the well-known equality that $\sum_{v \leq r} \binom{v+n-1}{n-1} = \binom{r+n}{n}$ which can be easily shown by induction on r . We claim that for any $n \geq 2$,

$$\binom{\lfloor n/8 \rfloor + n}{\lfloor n/8 \rfloor} \leq 2^{n-2}.$$

This is easy to verify for $\lfloor n/8 \rfloor \in \{0, 1\}$; for $\lfloor n/8 \rfloor \geq 2$, we have

$$\binom{\lfloor n/8 \rfloor + n}{\lfloor n/8 \rfloor} \leq \left(\frac{e(\lfloor n/8 \rfloor + n)}{\lfloor n/8 \rfloor} \right)^{\lfloor n/8 \rfloor} \leq \left(e \left(1 + \frac{n}{n/8 - 1} \right) \right)^{n/8},$$

which is

$$\left(e \left(1 + \frac{8n}{n-8} \right) \right)^{n/8} \leq \left(e \left(1 + \frac{8n}{n-n/2} \right) \right)^{n/8} = (e(1+16))^{n/8} \leq 2^{n-2},$$

by a straightforward calculation. As a result, we have

$$|\mathcal{B}(p)| \leq 2^n \cdot |\bar{\mathcal{B}}(p)| \leq 2^n \cdot 2^{n-2} = 4^{n-1}.$$

Then, since $|\mathcal{K}| \leq m^{n-1}$, we have

$$\left| \bigcup_{p \in \mathcal{K}} \mathcal{B}(p) \right| \leq \sum_{p \in \mathcal{K}} |\mathcal{B}(p)| \leq m^{n-1} \cdot 4^{n-1} = (4m)^{n-1}.$$

On the other hand, according to Lemma 4 with $\delta = 1/(kn) = 1/(8mn)$, we have

$$|\mathcal{Q}_\delta| = \binom{kn + n - 1}{n - 1} \geq \left(\frac{kn + n - 1}{n - 1} \right)^{n-1} > k^{n-1} = (8m)^{n-1} > (4m)^{n-1}.$$

This implies the existence of some $q \in \mathcal{Q}_\delta$ which is not in $\mathcal{B}(p)$ for any $p \in \mathcal{K}$, and we have the lemma. \square

With this lemma, we let the offline algorithm play this probability distribution q for all T steps. Then we choose the T loss vectors one after one in the following way. Suppose before step t we have chosen $t-1$ loss vectors for the first $t-1$ steps, and the online algorithm \mathcal{A} after receiving them enters some state associated with some probability distribution $p^{(t)} \in \mathcal{K}$. Then we choose the loss vector $\ell^{(t)} = (\ell_1^{(t)}, \dots, \ell_n^{(t)})$ for step t such that for any $i \in [n]$,

$$\ell_i^{(t)} = \begin{cases} 1, & \text{if } p_i^{(t)} \geq q_i, \\ 0, & \text{otherwise.} \end{cases}$$

The regret of the algorithm \mathcal{A} at step t with respect to this loss vector is

$$\sum_{i \in [n]} p_i^{(t)} \ell_i^{(t)} - \sum_{i \in [n]} q_i \ell_i^{(t)} = \sum_{i \in [n]} (p_i^{(t)} - q_i) \ell_i^{(t)} = \sum_{i \in [n]: p_i^{(t)} \geq q_i} (p_i^{(t)} - q_i),$$

which by Fact 1 and Lemma 7 is

$$\frac{1}{2} \cdot \|p^{(t)} - q\|_1 \geq \frac{1}{2} \cdot \frac{1}{64m} = \frac{1}{128m}.$$

As a result, with respect to such T loss vectors, the total regret of the algorithm \mathcal{A} is

$$\sum_{t \in [T]} \sum_{i \in [n]} p_i^{(t)} \ell_i^{(t)} - \sum_{t \in [T]} \sum_{i \in [n]} q_i \ell_i^{(t)} = \sum_{t \in [T]} \left(\sum_{i \in [n]} p_i^{(t)} \ell_i^{(t)} - \sum_{i \in [n]} q_i \ell_i^{(t)} \right) \geq \frac{T}{128m}.$$

This completes the proof of Theorem 3.

References

- [1] Abernethy, J., Agarwal, A., Bartlett, P.L., Rakhlin, A.: A stochastic view of optimal regret through minimax duality. In: Proceedings of the 22nd Annual Conference on Learning Theory (2009)

- [2] Arora, S., Hazan, E., Kale, S.: The multiplicative weights update method: a meta algorithm and applications (2005) (manuscript)
- [3] Ben-David, S., Pal, D., Shalev-Shwartz, S.: Agnostic online learning. In: Proceedings of the 22nd Annual Conference on Learning Theory (2009)
- [4] Blum, A., Mansour, Y.: Learning, regret minimization, and equilibria. In: Algorithmic Game Theory. Cambridge University Press, New York (2007)
- [5] Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press, New York (2006)
- [6] Dar, R., Feder, M.: Finite-memory universal prediction of individual continuous sequences. CoRR abs/1102.2836 (2011)
- [7] Even-Dar, E., Kleinberg, R., Mannor, S., Mansour, Y.: Online learning for global cost functions. In: Proceedings of the 22nd Annual Conference on Learning Theory (2009)
- [8] Freund, Y., Schapire, R.: A decision theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
- [9] Freund, Y., Schapire, R.: Adaptive game playing using multiplicative weights. Games and Economic Behavior 29, 79–103 (1999)
- [10] Hazan, E., Kale, S.: Extracting certainty from uncertainty: regret bounded by variation in costs. In: Proceedings of the 21st Annual Conference on Learning Theory, pp. 57–68 (2008)
- [11] Littlestone, N., Warmuth, M.: The weighted majority algorithm. Information and Computation 108(2), 212–261 (1994)
- [12] Meron, E., Feder, M.: Finite-memory universal prediction of individual sequences. IEEE Transactions on Information Theory 50(7), 1506–1523 (2004)
- [13] Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: Proceedings of the 20th International Conference on Machine Learning, pp. 928–936 (2003)

Universal Prediction of Selected Bits

Tor Lattimore¹, Marcus Hutter², and Vaibhav Gavane³

¹ Australian National University
`tor.lattimore@anu.edu.au`

² Australian National University and ETH Zürich
`marcus.hutter@anu.edu.au`

³ VIT University, Vellore, India
`vaibhav.gavane@gmail.com`

Abstract. Many learning tasks can be viewed as sequence prediction problems. For example, online classification can be converted to sequence prediction with the sequence being pairs of input/target data and where the goal is to correctly predict the target data given input data and previous input/target pairs. Solomonoff induction is known to solve the general sequence prediction problem, but only if the entire sequence is sampled from a computable distribution. In the case of classification and discriminative learning though, only the targets need be structured (given the inputs). We show that the normalised version of Solomonoff induction can still be used in this case, and more generally that it can detect any recursive sub-pattern (regularity) within an otherwise completely unstructured sequence. It is also shown that the unnormalised version can fail to predict very simple recursive sub-patterns.

Keywords: Sequence prediction, Solomonoff induction, online classification, discriminative learning, algorithmic information theory.

1 Introduction

The sequence prediction problem is the task of predicting the next symbol, x_n after observing $x_1 x_2 \cdots x_{n-1}$. Solomonoff induction [10, 11] solves this problem by taking inspiration from Occam’s razor and Epicurus’ principle of multiple explanations. These ideas are formalised in the field of Kolmogorov complexity, in particular by the universal a priori semi-measure \mathbf{M} .

Let $\mu(x_n|x_1 \cdots x_{n-1})$ be the true (unknown) probability of seeing x_n having already observed $x_1 \cdots x_{n-1}$. The celebrated result of Solomonoff [10] states that if μ is computable then

$$\lim_{n \rightarrow \infty} [\mathbf{M}(x_n|x_1 \cdots x_{n-1}) - \mu(x_n|x_1 \cdots x_{n-1})] = 0 \text{ with } \mu\text{-probability 1} \quad (1)$$

That is, \mathbf{M} can learn the true underlying distribution from which the data is sampled with probability 1. Solomonoff induction is arguably the gold standard predictor, universally solving many (passive) prediction problems [3, 4, 10].

However, Solomonoff induction makes no guarantees if μ is not computable. This would not be problematic if it were unreasonable to predict sequences

sampled from incomputable μ , but this is not the case. Consider the sequence below, where every even bit is the same as the preceding odd bit, but where the odd bits may be chosen arbitrarily.

$$00\ 11\ 11\ 11\ 00\ 11\ 00\ 00\ 00\ 11\ 11\ 00\ 00\ 00\ 00\ 00\ 00\ 11\ 11 \quad (2)$$

Any child will quickly learn the pattern that each even bit is the same as the preceding odd bit and will correctly predict the even bits. If Solomonoff induction is to be considered a truly intelligent predictor then it too should be able to predict the even bits. More generally, it should be able to detect any computable sub-pattern. It is this question, first posed in [3, 5] and resisting attempts by experts for 6 years, that we address.

At first sight, this appears to be an esoteric question, but consider the following problem. Suppose you are given a sequence of pairs, $x_1y_1x_2y_2x_3y_3\cdots$ where x_i is the data for an image (or feature vector) of a character and y_i the corresponding ascii code (class label) for that character. The goal of online classification is to construct a predictor that correctly predicts y_i given x_i based on the previously seen training pairs. It is reasonable to assume that there is a relatively simple pattern to generate y_i given x_i (humans and computers seem to find simple patterns for character recognition). However it is not necessarily reasonable to assume there exists a simple, or even computable, underlying distribution generating the training data x_i . This problem is precisely what gave rise to discriminative learning [9].

It turns out that there exist sequences with even bits equal to preceding odd bits on which \mathbf{M} fails to predict the even bits. On the other hand, it is known that \mathbf{M} is a defective measure, but may be normalised to a proper measure, \mathbf{M}_{norm} . We show that this normalised version *does* eventually predict any recursive sub-pattern of any sequence, such as that in Equation (2). This outcome is unanticipated since (all?) other results in the field are independent of normalisation [3, 4, 8, 10]. The proofs are completely different to the standard proofs of predictive results.

2 Notation and Definitions

We use similar notation to [1, 2, 3]. For a more comprehensive introduction to Kolmogorov complexity and Solomonoff induction see [3, 4, 8, 12].

Strings. A finite binary string x is a finite sequence $x_1x_2x_3\cdots x_n$ with $x_i \in \mathcal{B} = \{0, 1\}$. Its length is denoted $\ell(x)$. An infinite binary string ω is an infinite sequence $\omega_1\omega_2\omega_3\cdots$. The empty string of length zero is denoted ϵ . \mathcal{B}^n is the set of all binary strings of length n . \mathcal{B}^* is the set of all finite binary strings. \mathcal{B}^∞ is the set of all infinite binary strings. Substrings are denoted $x_{s:t} := x_sx_{s+1}\cdots x_{t-1}x_t$ where $s, t \in \mathbb{N}$ and $s \leq t$. If $s > t$ then $x_{s:t} = \epsilon$. A useful shorthand is $x_{<t} := x_{1:t-1}$. Strings may be concatenated. Let $x, y \in \mathcal{B}^*$ of length n and m respectively. Let $\omega \in \mathcal{B}^\infty$. Then,

$$xy := x_1x_2\cdots x_{n-1}x_ny_1y_2\cdots y_{m-1}y_m$$

$$x\omega := x_1x_2\cdots x_{n-1}x_n\omega_1\omega_2\omega_3\cdots$$

For $b \in \mathcal{B}$, $\neg b = 0$ if $b = 1$ and $\neg b = 1$ if $b = 0$. We write $x \sqsubseteq y$ if x is a prefix of y . Formally, $x \sqsubseteq y$ if $\ell(x) \leq \ell(y)$ and $x_i = y_i$ for all $1 \leq i \leq \ell(x)$. $x \sqsubset y$ if $x \sqsubseteq y$ and $\ell(x) < \ell(y)$.

Complexity. Here we give a brief introduction to Kolmogorov complexity and the associated notation.

Definition 1 (Inequalities). Let f, g be real valued functions. We write $f(x) \stackrel{\times}{\geq} g(x)$ if there exists a constant $c > 0$ such that $f(x) \geq c \cdot g(x)$ for all x . $f(x) \stackrel{\times}{\leq} g(x)$ is defined similarly. $f(x) \stackrel{\times}{=} g(x)$ if $f(x) \stackrel{\times}{\leq} g(x)$ and $f(x) \stackrel{\times}{\geq} g(x)$.

Definition 2 (Measures). We call $\mu : \mathcal{B}^* \rightarrow [0, 1]$ a semimeasure if $\mu(x) \geq \sum_{b \in \mathcal{B}} \mu(xb)$ for all $x \in \mathcal{B}^*$, and a probability measure if equality holds and $\mu(\epsilon) = 1$. $\mu(x)$ is the μ -probability that a sequence starts with x . $\mu(b|x) := \frac{\mu(xb)}{\mu(x)}$ is the probability of observing $b \in \mathcal{B}$ given that $x \in \mathcal{B}^*$ has already been observed. A function $P : \mathcal{B}^* \rightarrow [0, 1]$ is a semi-distribution if $\sum_{x \in \mathcal{B}^*} P(x) \leq 1$ and a probability distribution if equality holds.

Definition 3 (Enumerable Functions). A real valued function $f : A \rightarrow \mathbb{R}$ is enumerable if there exists a computable function $f : A \times \mathbb{N} \rightarrow \mathbb{Q}$ satisfying $\lim_{t \rightarrow \infty} f(a, t) = f(a)$ and $f(a, t+1) \geq f(a, t)$ for all $a \in A$ and $t \in \mathbb{N}$.

Definition 4 (Machines). A Turing machine L is a recursively enumerable set (which may be finite) containing pairs of finite binary strings $(p^1, y^1), (p^2, y^2), (p^3, y^3), \dots$.

L is a prefix machine if the set $\{p^1, p^2, p^3, \dots\}$ is prefix free (no program is a prefix of any other). It is a monotone machine if for all $(p, y), (q, x) \in L$ with $\ell(x) \geq \ell(y)$, $p \sqsubseteq q \implies y \sqsubseteq x$.

We define $L(p)$ to be the set of strings output by program p . This is different for monotone and prefix machines. For prefix machines, $L(p)$ contains only one element, $y \in L(p)$ if $(p, y) \in L$. For monotone machines, $y \in L(p)$ if there exists $(p, x) \in L$ with $y \sqsubseteq x$ and there does not exist a $(q, z) \in L$ with $q \sqsubset p$ and $y \sqsubseteq z$. For both machines $L(p)$ represents the output of machine L when given input p . If $L(p)$ does not exist then we say L does not halt on input p . Note that for monotone machines it is possible for the same program to output multiple strings. For example $(1, 1), (1, 11), (1, 111), (1, 1111), \dots$ is a perfectly legitimate monotone Turing machine. For prefix machines this is not possible. Also note that if L is a monotone machine and there exists an $x \in \mathcal{B}^*$ such that $x_{1:n} \in L(p)$ and $x_{1:m} \in L(p)$ then $x_{1:r} \in L(p)$ for all $n \leq r \leq m$.

Definition 5 (Complexity). Let L be a prefix or monotone machine then define

$$\lambda_L(y) := \sum_{p: y \in L(p)} 2^{-\ell(p)} \quad C_L(y) := \min_{p \in \mathcal{B}^*} \{\ell(p) : y \in L(p)\}$$

If L is a prefix machine then we write $\mathbf{m}_L(y) \equiv \lambda_L(y)$. If L is a monotone machine then we write $\mathbf{M}_L(y) \equiv \lambda_L(y)$. Note that if L is a prefix machine then

λ_L is an enumerable semi-distribution while if L is a monotone machine, λ_L is an enumerable semi-measure. In fact, every enumerable semi-measure (or semi-distribution) can be represented via some machine L as λ_L .

For prefix/monotone machine L we write L_t for the first t program/output pairs in the recursive enumeration of L , so L_t will be a finite set containing at most t pairs.¹

The set of all monotone (or prefix) machines is itself recursively enumerable [8]² which allows one to define a universal monotone machine U_M as follows. Let L^i be the i th monotone machine in the recursive enumeration of monotone machines.

$$(i'p, y) \in U_M \Leftrightarrow (p, y) \in L^i$$

where i' is a prefix coding of the integer i . A universal prefix machine, denoted U_P , is defined in a similar way. For details see [8].

Theorem 6 (Universal Prefix/Monotone Machines). *For the universal monotone machine U_M and universal prefix machine U_P ,*

$$\mathbf{m}_{U_P}(y) > c_L \mathbf{m}_L(y) \text{ for all } y \in \mathcal{B}^* \quad \mathbf{M}_{U_M}(y) > c_L \mathbf{M}_L(y) \text{ for all } y \in \mathcal{B}^*$$

where $c_L > 0$ depends on L but not y .

For a proof, see [8]. As usual, we will fix reference universal prefix/monotone machines U_P , U_M and drop the subscripts by letting,

$$\begin{aligned} \mathbf{m}(y) := \mathbf{m}_{U_P}(y) &\equiv \sum_{p:y \in U_P(p)} 2^{-\ell(p)} & \mathbf{M}(y) := \mathbf{M}_{U_M}(y) &\equiv \sum_{p:y \in U_M(p)} 2^{-\ell(p)} \\ K(y) := C_{U_P}(y) &\equiv \min_{p \in \mathcal{B}^*} \{\ell(p) : y \in U_P(p)\} & Km(y) &:= \min_{p \in \mathcal{B}^*} \{\ell(p) : y \in U_M(p)\} \end{aligned}$$

The choice of reference universal Turing machine is usually³ unimportant since a different choice varies \mathbf{m}, \mathbf{M} by only a multiplicative constant, while K, Km are varied by additive constants. For natural numbers n we define $K(n)$ by $K(\langle n \rangle)$ where $\langle n \rangle$ is the binary representation of n .

\mathbf{M} is not a proper measure, $\mathbf{M}(x) > \mathbf{M}(x0) + \mathbf{M}(x1)$ for all $x \in \mathcal{B}^*$, which means that $\mathbf{M}(0|x) + \mathbf{M}(1|x) < 1$, so \mathbf{M} assigns a non-zero probability that the sequence will end. This is because there are monotone programs p that halt, or enter infinite loops. For this reason Solomonoff introduced a normalised version, \mathbf{M}_{norm} defined as follows.

¹ L_t will contain exactly t pairs unless L is finite, in which case it will contain t pairs until t is greater than the size of L . This annoyance will never be problematic.

² Note the enumeration may include repetition, but this is unimportant in this case.

³ See [6] for a subtle exception. All the results in this paper are independent of universal Turing machine.

Definition 7 (Normalisation)

$$\mathbf{M}_{norm}(\epsilon) := 1 \quad \mathbf{M}_{norm}(y_n|y_{<n}) \equiv \frac{\mathbf{M}_{norm}(y_{1:n})}{\mathbf{M}_{norm}(y_{<n})} := \frac{\mathbf{M}(y_{1:n})}{\mathbf{M}(y_{<n}0) + \mathbf{M}(y_{<n}1)}.$$

This normalisation is not unique, but is philosophically and technically the most attractive and was used and defended by Solomonoff. Historically, most researchers have accepted the defective \mathbf{M} for technical convenience. As mentioned, the difference seldom matters, but in this paper it is somewhat surprisingly crucial. For a discussion of normalisation, see [8].

Theorem 8. *The following are results in Kolmogorov complexity. Proofs for all can be found in [8].*

1. $\mathbf{m}(x) \stackrel{\times}{=} 2^{-K(x)}$
2. $2^{-K(xb)} \stackrel{\times}{=} 2^{-K(x \neg b)}$
3. $\mathbf{M}(x) \stackrel{\times}{\geq} \mathbf{m}(x)$
4. If P is an enumerable semi-distribution, then $\mathbf{m}(y) \stackrel{\times}{\geq} P(y)$
5. If μ is an enumerable semi-measure, then $\mathbf{M}(y) \stackrel{\times}{\geq} \mu(y)$

Note the last two results are equivalent to Theorem 6 since every enumerable semi-(measure/distribution) is generated by a monotone/prefix machine in the sense of Theorem 6 and vice-versa.

Before proceeding to our own theorems we need a recently proven result in algorithmic information theory.

Theorem 9. [Lempp, Miller, Ng and Turetsky, 2010, unpublished, private communication] $\lim_{n \rightarrow \infty} \frac{\mathbf{m}(\omega_{<n})}{\mathbf{M}(\omega_{<n})} = 0$, for all $\omega \in \mathcal{B}^\infty$.

3 \mathbf{M}_{norm} Predicts Selected Bits

The following Theorem is the main positive result of this paper. It shows that any computable sub-pattern of a sequence will eventually be predicted by \mathbf{M}_{norm} .

Theorem 10. *Let $f : \mathcal{B}^* \rightarrow \mathcal{B} \cup \{\epsilon\}$ be a total recursive function and $\omega \in \mathcal{B}^\infty$ satisfying $f(\omega_{<n}) = \omega_n$ whenever $f(\omega_{<n}) \neq \epsilon$. If $f(\omega_{<n_i}) \neq \epsilon$ is defined for an infinite sequence n_1, n_2, n_3, \dots then $\lim_{i \rightarrow \infty} \mathbf{M}_{norm}(\omega_{n_i}|\omega_{<n_i}) = 1$.*

Essentially the Theorem is saying that if there exists a computable predictor f that correctly predicts the next bit every time it tries (i.e when $f(\omega_{<n}) \neq \epsilon$) then \mathbf{M}_{norm} will eventually predict the same bits as f . By this we mean that if you constructed a predictor $f_{\mathbf{M}_{norm}}$ defined by $f_{\mathbf{M}_{norm}}(\omega_{<n}) = \arg \max_{b \in \mathcal{B}} \mathbf{M}_{norm}(b|\omega_{<n})$, then there exists an N such that $f_{\mathbf{M}_{norm}}(\omega_{<n}) = f(\omega_{<n})$ for all $n > N$ where $f(\omega_{<n}) \neq \epsilon$. For example, let f be defined by

$$f(x) = \begin{cases} x_{\ell(x)} & \text{if } \ell(x) \text{ odd} \\ \epsilon & \text{otherwise} \end{cases}$$

Now if $\omega \in \mathcal{B}^\infty$ satisfies $\omega_{2n} = f(\omega_{<2n}) = \omega_{2n-1}$ for all $n \in \mathbb{N}$ then Theorem 10 shows that $\lim_{n \rightarrow \infty} \mathbf{M}_{norm}(\omega_{2n} | \omega_{<2n}) = 1$. It says nothing about the predictive qualities of \mathbf{M}_{norm} on the odd bits, on which there are no restrictions.

The proof essentially relies on using f to show that monotone programs for $\omega_{<n_i} \neg \omega_{n_i}$ can be converted to prefix programs. This is then used to show that $\mathbf{M}(\omega_{<n_i} \neg \omega_{n_i}) \stackrel{\times}{=} \mathbf{m}(\omega_{<n_i} \neg \omega_{n_i})$. The result will then follow from Theorem 9.

Theorem 10 insists that f be totally recursive and that $f(\omega_{<n}) = \epsilon$ if f refrains from predicting. One could instead allow f to be partially recursive and simply not halt to avoid making a prediction. The proof below breaks down in this case and we suspect that Theorem 10 will become invalid if f is permitted to be only partially recursive.

Proof (Theorem 10). We construct a machine L from U_M consisting of all programs that produce output that f would not predict. We then show that these programs essentially form a prefix machine. Define L by the following process

1. $L := \emptyset$ and $t := 1$.
2. Let (p, y) be the t th pair in U_M .
3. Let i be the smallest natural number such that $y_i \neq f(y_{<i}) \neq \epsilon$. That is, i is the position at which f makes its first mistake when predicting y . If f makes no prediction errors then i doesn't exist.⁴
4. If i exists then $L := L \cup \{(p, y_{1:i})\}$ (Note that we do not allow L to contain duplicates).
5. $t := t + 1$ and go to step 2.

Since f is totally recursive and U_M is recursively enumerable, the process above shows that L is recursively enumerable. It is easy to see that L is a monotone machine. Further, if $(p, y), (q, x) \in L$ with $p \sqsubseteq q$ then $y = x$. This follows since by monotonicity we would have that $y \sqsubseteq x$, but $f(x_{<\ell(y)}) = f(y_{<\ell(y)}) \neq y_{\ell(y)} = x_{\ell(y)}$ and by steps 3 and 4 in the process above we have that $\ell(x) = \ell(y)$.

Recall that L_t is the t th enumeration of L and contains t elements. Define $\bar{L}_t \subseteq L_t$ to be the largest prefix free set of shortest programs. Formally, $(p, y) \in \bar{L}_t$ if there does not exist a $(q, x) \in L_t$ such that $q \sqsubset p$. For example, if $L_t = (1, 001), (11, 001), (01, 11110), (010, 11110)$ then $\bar{L}_t = (1, 001), (01, 11110)$. If we now added $(0, 11110)$ to L_t to construct L_{t+1} then \bar{L}_{t+1} would be $(1, 001), (0, 11110)$.

Since L_t is finite, \bar{L}_t is easily computable from L_t . Therefore the following function is computable.

$$P(y, t) := \sum_{p: (p, y) \in \bar{L}_t} 2^{-\ell(p)} \geq 0.$$

⁴ This is where the problem lies for partially recursive prediction functions. Computing the smallest i for which f predicts incorrectly is incomputable if f is only partially recursive, but computable if it is totally recursive. It is this distinction that allows L to be recursively enumerable, and so be a machine.

Now \bar{L}_t is prefix free, so by Kraft's inequality $\sum_{y \in \mathcal{B}^*} P(y, t) \leq 1$ for all $t \in \mathbb{N}$. We now show that $P(y, t+1) \geq P(y, t)$ for all $y \in \mathcal{B}^*$ and $t \in \mathbb{N}$ which proves that $P(y) = \lim_{t \rightarrow \infty} P(y, t)$ exists and is a semi-distribution.

Let (p, y) be the program/output pair in L_{t+1} but not in L_t . To see how $P(\cdot, t)$ compares to $P(\cdot, t+1)$ we need to compare \bar{L}_t and \bar{L}_{t+1} . There are three cases:

1. There exists a $(q, x) \in L_t$ with $q \sqsubset p$. In this case $\bar{L}_{t+1} = \bar{L}_t$.
2. There does not exist a $(q, x) \in L_t$ such that $p \sqsubset q$. In this case (p, y) is simply added to \bar{L}_t to get \bar{L}_{t+1} and so $\bar{L}_t \subset \bar{L}_{t+1}$. Therefore $P(\cdot, t+1) \geq P(\cdot, t)$ is clear.
3. There does exist a $(q, x) \in \bar{L}_t$ such that $p \sqsubset q$. In this case \bar{L}_{t+1} differs from \bar{L}_t in that it contains (p, y) but not (q, x) . Since $p \sqsubset q$ we have that $y = x$. Therefore $P(y, t+1) - P(y, t) = 2^{-\ell(p)} - 2^{-\ell(q)} > 0$ since $p \sqsubset q$. For other values, $P(\cdot, t) = P(\cdot, t+1)$.

Note that it is not possible that $p = q$ since then $x = y$ and duplicates are not added to L . Therefore P is an enumerable semi-distribution. By Theorem 8 we have

$$\mathbf{m}(\omega_{<n_i} \neg \omega_{n_i}) \stackrel{\times}{\geq} P(\omega_{<n_i} \neg \omega_{n_i}) \quad (3)$$

where the constant multiplicative fudge factor in the $\stackrel{\times}{\geq}$ is independent of i . Suppose $\omega_{<n_i} \neg \omega_{n_i} \in U_M(p)$. Therefore there exists a y such that $\omega_{<n_i} \neg \omega_{n_i} \sqsubseteq y$ and $(p, y) \in U_M$. By parts 2 and 3 of the process above, $(p, \omega_{<n_i} \neg \omega_{n_i})$ is added to L . Therefore there exists a $T \in \mathbb{N}$ such that $(p, \omega_{<n_i} \neg \omega_{n_i}) \in L_t$ for all $t \geq T$.

Since $\omega_{<n_i} \neg \omega_{n_i} \in U_M(p)$, there does not exist a $q \sqsubset p$ with $\omega_{<n_i} \neg \omega_{n_i} \in U_M(q)$. Therefore eventually, $(p, \omega_{<n_i} \neg \omega_{n_i}) \in \bar{L}_t$ for all $t \geq T$. Since every program in U_M for $\omega_{<n_i} \neg \omega_{n_i}$ is also a program in L , we get

$$\lim_{t \rightarrow \infty} P(\omega_{<n_i} \neg \omega_{n_i}, t) \equiv P(\omega_{<n_i} \neg \omega_{n_i}) = \mathbf{M}(\omega_{<n_i} \neg \omega_{n_i}).$$

Next,

$$\mathbf{M}_{norm}(\neg \omega_{n_i} | \omega_{<n_i}) \equiv \frac{\mathbf{M}(\omega_{<n_i} \neg \omega_{n_i})}{\mathbf{M}(\omega_{<n_i} \omega_{n_i}) + \mathbf{M}(\omega_{<n_i} \neg \omega_{n_i})} \quad (4)$$

$$\stackrel{\times}{\leq} \frac{\mathbf{m}(\omega_{<n_i} \neg \omega_{n_i})}{\mathbf{M}(\omega_{1:n_i})} \quad (5)$$

$$\stackrel{\times}{=} \frac{\mathbf{m}(\omega_{1:n_i})}{\mathbf{M}(\omega_{1:n_i})} \quad (6)$$

where Equation (4) follows by the definition of \mathbf{M}_{norm} . Equation (5) follows from Equation (3) and algebra. Equation (6) follows since $\mathbf{m}(xb) \stackrel{\times}{=} 2^{-K(xb)} \stackrel{\times}{=} 2^{-K(x-b)} \stackrel{\times}{=} \mathbf{m}(x-b)$, which is Theorem 8. However, by Theorem 9, $\lim_{i \rightarrow \infty} \frac{\mathbf{m}(\omega_{<n_i})}{\mathbf{M}(\omega_{<n_i})} = 0$ and so $\lim_{i \rightarrow \infty} \mathbf{M}_{norm}(\neg \omega_{n_i} | \omega_{<n_i}) = 0$. Therefore $\lim_{i \rightarrow \infty} \mathbf{M}_{norm}(\omega_{n_i} | \omega_{<n_i}) = 1$ as required. \square

We have remarked already that Theorem 10 is likely not valid if f is permitted to be a partial recursive function that only output on sequences for which they make a prediction. However, there is a class of predictors larger than the totally recursive ones of Theorem 10, which \mathbf{M}_{norm} still learns.

Theorem 11. Let $f : \mathcal{B}^* \rightarrow \mathcal{B} \cup \{\epsilon\}$ be a partial recursive function and $\omega \in \mathcal{B}^\infty$ satisfying

1. $f(\omega_{<n})$ is defined for all n .
2. $f(\omega_{<n}) = \omega_n$ whenever $f(\omega_{<n}) \neq \epsilon$.

If $f(\omega_{<n_i}) \in \mathcal{B}$ for an infinite sequence n_1, n_2, n_3, \dots then

$$\lim_{i \rightarrow \infty} \mathbf{M}_{norm}(\omega_{n_i} | \omega_{<n_i}) = 1.$$

The difference between this result and Theorem 10 is that f need only be defined on all prefixes of at least one $\omega \in \mathcal{B}^\infty$ and not everywhere in \mathcal{B}^* . This allows for a slightly broader class of predictors. For example, let $\omega = p^1 b^1 p^2 b^2 p^3 b^3 \dots$ where p^i is some prefix machine that outputs at least one bit and b^i is the first bit of that output. Now there exists a computable f such that $f(p^1 b^1 \dots p^{i-1} b^{i-1} p^i) = b^i$ for all i and $f(\omega_{<n}) = \epsilon$ whenever $\omega_n \neq b^i$ for some i (f only tries to predict the outputs). By Theorem 11, \mathbf{M}_{norm} will correctly predict b^i .

The proof of Theorem 11 is almost identical to that of Theorem 10, but with one additional subtlety.

Proof sketch. The proof follows that of Theorem 10 until the construction of L . This breaks down because step 3 is no longer computable since f may not halt on some string that is not a prefix of ω . The modification is to run steps 2-4 in parallel for all t and only adding $(p, y_{1:i})$ to L once it has been proven that $f(y_{<i}) \neq y_i$ and $f(y_{<k})$ halts for all $k < i$, and either chooses not to predict (outputs ϵ), or predicts correctly. Since f halts on all prefixes of ω , this does not change L for any programs we care about and the remainder of the proof goes through identically. \square

It should be noted that this new class of predictors is still less general than allowing f to an arbitrary partial recursive predictor. For example, a partial recursive f can predict the ones of the halting sequence, while choosing not to predict the zeros (the non-halting programs). It is clear this cannot be modified into a computable f predicting both ones and zeros, or predicting ones and outputting ϵ rather than zero, as this would solve the halting problem.

4 M Fails to Predict Selected Bits

The following theorem is the corresponding negative result that while \mathbf{M}_{norm} always predicts recursive sub-patterns, \mathbf{M} can fail to do so. This essentially means that the key problem with \mathbf{M} is that it is a defective measure.

Theorem 12. Let $f : \mathcal{B}^* \rightarrow \mathcal{B} \cup \{\epsilon\}$ be the total recursive function defined by,

$$f(z) := \begin{cases} z_{\ell(z)} & \text{if } \ell(z) \text{ odd} \\ \epsilon & \text{otherwise} \end{cases}$$

There exists an infinite string $\omega \in \mathcal{B}^\infty$ with $\omega_{2n} = f(\omega_{<2n}) \equiv \omega_{2n-1}$ for all $n \in \mathbb{N}$ such that

$$\liminf_{n \rightarrow \infty} \mathbf{M}(\omega_{2n} | \omega_{<2n}) < 1.$$

The proof requires some lemmas.

Lemma 13. $\mathbf{M}(xy)$ can be bounded as follows.

$$2^{K(\ell(x))} \mathbf{M}(y) \stackrel{\times}{\geq} \mathbf{M}(xy) \stackrel{\times}{\geq} \mathbf{M}(y) 2^{-K(x)}. \quad (7)$$

Proof. Both inequalities are proven relatively easily by normal methods as used in [8] and elsewhere. Nevertheless we present them as a warm-up to the slightly more subtle proof later.

Now construct monotone machine L , which we should think of as taking two programs as input. The first, a prefix program p , the output of which we view as a natural number n . The second, a monotone program. We then simulate the monotone machine and strip the first n bits of its output. L is formally defined as follows.

1. $L := \emptyset, t := 1$
2. Let $(p, n), (q, y)$ be the t th pair of program/outputs in $U_P \times U_M$, which is enumerable.
3. If $\ell(y) \geq n$ then add $(pq, y_{n+1:\ell(y)})$ to L
4. $t := t + 1$ and go to step 2

By construction, L is enumerable and is a monotone machine. Note that if $xy \in U_M(q)$ and $\ell(x) \in U_P(p)$ then $y \in L(pq)$. Now,

$$\mathbf{M}(y) \stackrel{\times}{\geq} \mathbf{M}_L(y) \equiv \sum_{r:y \in L(r)} 2^{-\ell(r)} \geq \sum_{q,p:xy \in U_M(q), \ell(x) \in U_P(p)} 2^{-\ell(pq)} \quad (8)$$

$$= \sum_{q:xy \in U_M(q)} 2^{-\ell(q)} \sum_{p:\ell(x) \in U_P(p)} 2^{-\ell(p)} \equiv \mathbf{M}(xy)\mathbf{m}(\ell(x)) \quad (9)$$

$$\stackrel{\times}{=} \mathbf{M}(xy) 2^{-K(\ell(x))} \quad (10)$$

where Equation (8) follows by Theorem 6, definitions and because if $xy \in U_M(q)$ and $\ell(x) \in U_P(p)$ then $y \in L(pq)$. Equation (9) by algebra, definitions. Equation (10) by Theorem 8.

The second inequality is proved similarly. We define a machine L as follows,

1. $L = \emptyset, t := 1$
2. Let $(q, x), (r, y)$ be the t th element in $U_P \times U_M$, which is enumerable.
3. Add (qr, xy) to L
4. $t := t + 1$ and go to step 2

It is easy to show that L is monotone by using the properties of U_P and U_M . Now,

$$\begin{aligned} \mathbf{M}(xy) &\stackrel{\times}{\geq} \mathbf{M}_L(xy) \equiv \sum_{p:xy \in L(p)} 2^{-\ell(p)} \geq \sum_{q,r:x \in U_P(q), y \in U_M(r)} 2^{-\ell(qr)} \\ &= \sum_{q:x \in U_P(q)} 2^{-\ell(q)} \sum_{r:y \in U_M(r)} 2^{-\ell(r)} \equiv \mathbf{m}(x)\mathbf{M}(y) \stackrel{\times}{\geq} 2^{-K(x)}\mathbf{M}(y). \end{aligned}$$

□

Lemma 14. *There exists an $\omega \in \mathcal{B}^\infty$ such that*

$$\liminf_{n \rightarrow \infty} [\mathbf{M}(0|\omega_{<n}) + \mathbf{M}(1|\omega_{<n})] = 0.$$

Proof. First we show that for each $\delta > 0$ there exists a $z \in \mathcal{B}^*$ such that $\mathbf{M}(0|z) + \mathbf{M}(1|z) < \delta$. This result is already known and is left as an exercise (4.5.6) with a proof sketch in [8]. For completeness, we include a proof. Recall that $\mathbf{M}(\cdot, t)$ is the function approximating $\mathbf{M}(\cdot)$ from below. Fixing an n , define $z \in \mathcal{B}^*$ inductively as follows.

1. $z := \epsilon$
2. Let t be the first natural number such that $\mathbf{M}(zb, t) > 2^{-n}$ for some $b \in \mathcal{B}$.
3. If t exists then $z := z \neg b$ and repeat step 2. If t does not exist then z is left unchanged (forever).

Note that z must be finite since each time it is extended, $\mathbf{M}(zb, t) > 2^{-n}$. Therefore $\mathbf{M}(z \neg b, t) < \mathbf{M}(z, t) - 2^{-n}$ and so each time z is extended, the value of $\mathbf{M}(z, t)$ decreases by at least 2^{-n} so eventually $\mathbf{M}(zb, t) < 2^{-n}$ for all $b \in \mathcal{B}$. Now once the z is no longer being extended (t does *not* exist in step 3 above) we have

$$\mathbf{M}(z0) + \mathbf{M}(z1) \leq 2^{1-n}. \quad (11)$$

However we can also show that $\mathbf{M}(z) \stackrel{\times}{\geq} 2^{-K(n)}$. The intuitive idea is that the process above requires only the value of n , which can be encoded in $K(n)$ bits. More formally, let p be such that $n \in U_P(p)$ and note that the following set is recursively enumerable (but not recursive) by the process above.

$$L_p := (p, \epsilon), (p, z_{1:1}), (p, z_{1:2}), (p, z_{1:3}), \dots, (p, z_{1:\ell(z)-1}), (p, z_{1:\ell(z)}).$$

Now take the union of all such sets, which is a) recursively enumerable since U_P is, and b) a monotone machine because U_P is a prefix machine.

$$L := \bigcup_{(p,n) \in U_P} L_p.$$

Therefore

$$\mathbf{M}(z) \stackrel{\times}{\geq} \mathbf{M}_L(z) \geq 2^{-K(n)} \quad (12)$$

where the first inequality is from Theorem 6 and the second follows since if n^* is the program of length $K(n)$ with $U_P(n^*) = n$ then $(n^*, z_{1:\ell(z)}) \in L$. Combining Equations (11) and (12) gives

$$\mathbf{M}(0|z) + \mathbf{M}(1|z) \stackrel{\times}{\leq} 2^{1-n+K(n)}.$$

Since this tends to zero as n goes to infinity,⁵ for each $\delta > 0$ we can construct a $z \in \mathcal{B}^*$ satisfying $\mathbf{M}(0|z) + \mathbf{M}(1|z) < \delta$, as required. For the second part of the proof, we construct ω by concatenation.

$$\omega := z^1 z^2 z^3 \dots$$

where $z^n \in \mathcal{B}^*$ is chosen such that,

$$\mathbf{M}(0|z^n) + \mathbf{M}(1|z^n) < \delta_n \quad (13)$$

with δ_n to be chosen later. Now,

$$\mathbf{M}(b|z^1 \dots z^n) \equiv \frac{\mathbf{M}(z^1 \dots z^n b)}{\mathbf{M}(z^1 \dots z^n)} \quad (14)$$

$$\stackrel{\times}{\leq} \left[2^{K(\ell(z^1 \dots z^{n-1})) + K(z^1 \dots z^{n-1})} \right] \frac{\mathbf{M}(z^n b)}{\mathbf{M}(z^n)} \quad (15)$$

$$\equiv \left[2^{K(\ell(z^1 \dots z^{n-1})) + K(z^1 \dots z^{n-1})} \right] \mathbf{M}(b|z^n) \quad (16)$$

where Equation (14) is the definition of conditional probability. Equation (15) follows by applying Lemma 13 with $x = z^1 z^2 \dots z^{n-1}$ and $y = z^n$ or $z^n b$. Equation (16) is again the definition of conditional probability. Now let

$$\delta_n = \frac{2^{-n}}{2^{K(\ell(z^1 \dots z^{n-1})) + K(z^1 \dots z^{n-1})}}.$$

Combining this with Equations (13) and (16) gives

$$\mathbf{M}(0|z^1 \dots z^n) + \mathbf{M}(1|z^1 \dots z^n) \stackrel{\times}{\leq} 2^{-n}.$$

Therefore,

$$\liminf_{n \rightarrow \infty} [\mathbf{M}(0|\omega_{<n}) + \mathbf{M}(1|\omega_{<n})] = 0$$

as required. \square

Proof (Theorem 12). Let $\bar{\omega} \in \mathcal{B}^\infty$ be defined by $\bar{\omega}_{2n} := \bar{\omega}_{2n-1} := \omega_n$ where ω is the string defined in the previous lemma. Recall $U_M := \{(p^1, y^1), (p^2, y^2), \dots\}$ is the universal monotone machine. Define monotone machine L by the following process,

⁵ An integer n can easily be encoded in $2 \log n$ bits, so $K(n) \leq 2 \log n + c$ for some $c > 0$ independent of n .

1. $L = \emptyset$, $t = 1$
2. Let (p, y) be the t th element in the enumeration of U_M
3. Add $(p, y_1 y_3 y_5 y_7 \dots)$ to L
4. $t := t + 1$ and go to step 2.

Therefore if $\bar{\omega}_{<2n} \in U_M(p)$ then $\omega_{1:n} \in L(p)$. By identical reasoning as elsewhere,

$$\mathbf{M}(\omega_{1:n}) \stackrel{*}{\geq} \mathbf{M}(\bar{\omega}_{<2n}). \quad (17)$$

In fact, $\mathbf{M}(\omega_{1:n}) \stackrel{*}{=} \mathbf{M}(\bar{\omega}_{<2n})$, but this is unnecessary. Let $P := \{p : \exists b \in \mathcal{B} \text{ s.t } \omega_{1:n}b \in U_M(p)\}$ and $Q := \{p : \bar{\omega}_{<2n} \in U_M(p)\} \supset P$. Therefore

$$\begin{aligned} 1 - \mathbf{M}(0|\omega_{1:n}) - \mathbf{M}(1|\omega_{1:n}) &= 1 - \frac{\sum_{p \in P} 2^{-\ell(p)}}{\sum_{q \in Q} 2^{-\ell(q)}} \\ &= \frac{\sum_{p \in Q - P} 2^{-\ell(p)}}{\sum_{q \in Q} 2^{-\ell(q)}}. \end{aligned}$$

Now let $\bar{P} := \{p : \exists b \in \mathcal{B} \text{ s.t } \bar{\omega}_{<2n}b \in U_M(p)\}$ and $\bar{Q} := \{p : \bar{\omega}_{<2n} \in U_M(p)\} \supset \bar{P}$. Define monotone machine L by the following process

1. $L = \emptyset$, $t := 1$
2. Let (p, y) be the t th program/output pair in U_M
3. Add $(p, y_1 y_3 y_5 y_7 \dots y_{\ell(y)-1} y_{\ell(y)-1} y_{\ell(y)})$ to L
4. $t := t + 1$ and go to step 2.

Let $p \in Q - P$. Therefore $\omega_{1:n} \in U_M(p)$ and $\omega_{1:n}b \notin U_M(p)$ for any $b \in \mathcal{B}$. Therefore $\bar{\omega}_{<2n} \in L(p)$ while $\bar{\omega}_{<2n}b \notin L(p)$ for any $b \in \mathcal{B}$. Now there exists an i such that L is the i th machine in the enumeration of monotone machines, L^i .

Therefore, by the definition of the universal monotone machine U_M we have that $\bar{\omega}_{<2n}b \notin U_M(i'p) = L^i(p) = L(p) \ni \bar{\omega}_{<2n}$ and $U_M(i'p) = L(p)$ for any $b \in \mathcal{B}$. Therefore $i'p \in \bar{Q} - \bar{P}$ and so,

$$\sum_{q \in \bar{Q} - \bar{P}} 2^{-\ell(q)} \geq \sum_{p: i'p \in \bar{Q} - \bar{P}} 2^{-\ell(i'p)} \geq \sum_{p \in Q - P} 2^{-\ell(i'p)} \stackrel{*}{=} \sum_{p \in Q - P} 2^{-\ell(p)}. \quad (18)$$

Therefore

$$1 - \mathbf{M}(0|\bar{\omega}_{<2n}) - \mathbf{M}(1|\bar{\omega}_{<2n}) \equiv \frac{\sum_{p \in \bar{Q} - \bar{P}} 2^{-\ell(p)}}{\mathbf{M}(\bar{\omega}_{<2n})} \quad (19)$$

$$\stackrel{*}{\geq} \frac{\sum_{p \in Q - P} 2^{-\ell(p)}}{\mathbf{M}(\omega_{1:n})} \quad (20)$$

$$\equiv 1 - \mathbf{M}(0|\omega_{1:n}) - \mathbf{M}(1|\omega_{1:n}) \quad (21)$$

where Equation (19) follows from the definition of \bar{P} , \bar{Q} and \mathbf{M} . Equation (20) by (18) and (17). Equation (21) by the definition of P, Q and \mathbf{M} . Therefore by Lemma 14 we have

$$\limsup_{n \rightarrow \infty} [1 - \mathbf{M}(0|\bar{\omega}_{<2n}) - \mathbf{M}(1|\bar{\omega}_{<2n})] \stackrel{*}{\geq} \limsup_{n \rightarrow \infty} [1 - \mathbf{M}(0|\omega_{1:n}) - \mathbf{M}(1|\omega_{1:n})] = 1.$$

Therefore

$$\liminf_{n \rightarrow \infty} \mathbf{M}(\bar{\omega}_{2n} | \bar{\omega}_{<2n}) < 1$$

as required. \square

Note that $\lim_{n \rightarrow \infty} \mathbf{M}(\bar{\omega}_{2n} | \bar{\omega}_{<2n}) \neq 0$ in fact, one can show that there exists a $c > 0$ such that $\mathbf{M}(\bar{\omega}_{2n} | \bar{\omega}_{<2n}) > c$ for all $n \in \mathbb{N}$.

5 Discussion

Summary. Theorem 10 shows that if an infinite sequence contains a computable sub-pattern then the normalised universal semi-measure \mathbf{M}_{norm} will eventually predict it. This means that Solomonoff's normalised version of induction is effective in the classification example given in the introduction. Note that we have only proven the binary case, but expect the proof will go through identically for arbitrary finite alphabet.

On the other hand, Theorem 12 shows that plain \mathbf{M} can fail to predict such structure and that it does so because it is not a proper measure. These results are surprising since (all?) other predictive results, including Equation 11 and many others in [3, 4, 8, 10], do not rely on normalisation.

Consequences. We have shown that \mathbf{M} really is deficient on some sequences that may still be of interest. This forces us to use \mathbf{M}_{norm} which has one major drawback, it is not enumerable. Since \mathbf{M} is enumerable, we have some hope of approximating it using standard compression algorithms or more brute force methods. However \mathbf{M}_{norm} is only approximable⁶ which makes it substantially more challenging or impossible to approximate. This disadvantage is not as great as it seems since the predictive distribution $\mathbf{M}(b|x)$ is also only approximable.

Open Questions. A number of open questions were encountered in writing this paper.

1. Extend Theorem 10 to the stochastic case where a sub-pattern is generated stochastically from a computable distribution rather than merely a computable function. It seems likely that a different approach will be required to solve this problem.
2. Another interesting question is to strengthen the result by proving a convergence rate. It may be possible to prove that under the same conditions as Theorem 10 that $\sum_{i=1}^{\infty} [1 - \mathbf{M}_{norm}(\omega_{n_i} | \omega_{<n_i})] \stackrel{x}{\leq} K(f)$ where $K(f)$ is the (prefix) complexity of the predicting function f . Again, if this is even possible, it will likely require a different approach.
3. Prove or disprove the validity of Theorem 10 when the totally recursive prediction function f (or the modified predictor of Theorem 11) is replaced by a partially recursive function.

⁶ A function f is approximable if there exists a computable function $f(\cdot, t)$ with $\lim_{t \rightarrow \infty} f(\cdot, t) = f(\cdot)$. Convergence need not be monotonic.

Acknowledgements. We thank Wen Shao and reviewers for valuable feedback on earlier drafts and the Australian Research Council for support under grant DP0988049.

References

- [1] Gács, P.: On the relation between descriptional complexity and algorithmic probability. *Theoretical Computer Science* 22(1-2), 71–93 (1983)
- [2] Gács, P.: Expanded and improved proof of the relation between description complexity and algorithmic probability (2008) (unpublished)
- [3] Hutter, M.: *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin (2004)
- [4] Hutter, M.: On universal prediction and Bayesian confirmation. *Theoretical Computer Science* 384(1), 33–48 (2007)
- [5] Hutter, M.: Open problems in universal induction & intelligence. *Algorithms* 3(2), 879–906 (2009)
- [6] Hutter, M., Muchnik, A.A.: On semimeasures predicting Martin-Löf random sequences. *Theoretical Computer Science* 382(3), 247–261 (2007)
- [7] Lempp, S., Miller, J., Ng, S., Turetsky, D.: Complexity inequality. Unpublished, private communication (2010)
- [8] Li, M., Vitanyi, P.: *An Introduction to Kolmogorov Complexity and Its Applications*, 3rd edn. Springer, Heidelberg (2008)
- [9] Long, P., Servedio, R.: Discriminative Learning Can Succeed Where Generative Learning Fails. In: Lugosi, G., Simon, H.U. (eds.) *COLT 2006. LNCS (LNAI)*, vol. 4005, pp. 319–334. Springer, Heidelberg (2006)
- [10] Solomonoff, R.: A formal theory of inductive inference, Part I. *Information and Control* 7(1), 1–22 (1964)
- [11] Solomonoff, R.: A formal theory of inductive inference, Part II. *Information and Control* 7(2), 224–254 (1964)
- [12] Zvonkin, A.K., Levin, L.A.: The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys* 25(6), 83 (1970)

Table of Notation

Symbol	Description
\mathcal{B}	Binary symbols, 0 and 1
\mathbb{Q}	Rational numbers
\mathbb{N}	Natural numbers
\mathcal{B}^*	The set of all finite binary strings
\mathcal{B}^∞	The set of all infinite binary strings
x, y, z	Finite binary strings
ω	An infinite binary string
$\bar{\omega}$	An infinite binary string with even bits equal to preceding odd bits
$\ell(x)$	The length of binary string x
$\neg b$	The negation of binary symbol b . $\neg b = 0$ if $b = 1$ and $\neg b = 1$ if $b = 0$
p, q	Programs
μ	An enumerable semi-measure
\mathbf{M}	The universal enumerable semi-measure
\mathbf{M}_{norm}	The normalised version of the universal enumerable semi-measure
\mathbf{m}	The universal enumerable semi-distribution
$K(f)$	The prefix Kolmogorov complexity of a function f
L	An enumeration of program/output pairs defining a machine
U_M	The universal monotone machine
U_P	The universal prefix machine
\times	
\geq	$f(x) \stackrel{x}{\geq} g(x)$ if there exists a $c > 0$ such that $f(x) > c \cdot g(x)$ for all x
\times	
\leq	$f(x) \stackrel{x}{\leq} g(x)$ if there exists a $c > 0$ such that $f(x) < c \cdot g(x)$ for all x
$\stackrel{x}{=}$	$f(x) \stackrel{x}{=} g(x)$ if $f(x) \stackrel{x}{\geq} g(x)$ and $f(x) \stackrel{x}{\leq} g(x)$
$x \sqsubset y$	x is a prefix of y and $\ell(x) < \ell(y)$
$x \sqsubseteq y$	x is a prefix of y

Semantic Communication for Simple Goals Is Equivalent to On-line Learning^{*}

Brendan Juba^{1,***} and Santosh Vempala^{2,***}

¹ MIT CSAIL and Harvard University
Maxwell Dworkin 140
33 Oxford St.
Cambridge, MA 02138
bjuba@alum.mit.edu

² Georgia Tech
Klaus Advanced Computing Building 2224
266 First Dr.
Atlanta, GA 30332-0765
vempala@cc.gatech.edu

Abstract. Previous works [11, 6] introduced a model of *semantic communication* between a “user” and a “server,” in which the user attempts to achieve a given *goal for communication*. They show that whenever the user can *sense* progress, there exist *universal* user strategies that can achieve the goal whenever it is possible for any other user to reliably do so. A drawback of the actual constructions is that the users are inefficient: they enumerate protocols until they discover one that is successful, leading to the potential for exponential overhead in the length of the desired protocol. Goldreich et al. [6] conjectured that this overhead could be reduced to a polynomial dependence if we restricted our attention to classes of sufficiently simple user strategies and goals. In this work, we are able to obtain such universal strategies for some reasonably general special cases by establishing an equivalence between these special cases and the usual model of *mistake-bounded on-line learning* [3, 15]. This equivalence also allows us to see the limits of constructing universal users based on *sensing* and motivates the study of sensing with *richer kinds of feedback*. Along the way, we also establish a new lower bound for the “beliefs model” [12], which demonstrates that constructions of efficient users in that framework rely on the existence of a common “belief” under which all of the servers in a class are designed to be efficient.

Keywords: semantic communication, on-line learning, feedback models.

1 Introduction

The *semantic communication* model [11, 6] was introduced to model the problem of communication in the presence of possible *misunderstanding*. In particular,

* This work is also presented in Chapters 4 and 8 of the first author’s thesis [10].

** Supported by NSF Awards CCF-0939370, CCF-04-27129, and CCF-09-64401.

*** Supported by NSF Awards AF-0915903 and AF-0910584.

it was intended to address settings where two computers communicate using communications protocols designed and implemented by different parties. In such settings, the possibility of incompatibility arises, and so it would be desirable for one (or both) of these computers to utilize a *universal* communications protocol that automatically corrects miscommunication. The main results of these works demonstrated that when a *goal for communication* is fixed in advance, such a universal protocol – that achieves its goal whenever its partner supports such functionality – can often be constructed.

Here, we attempt to address one of the main deficiencies of earlier results, specifically of results in the *infinite executions* model [6], reviewed in Sec. 2. To be more precise, the main results constructing universal protocols (such as Thm. 5) relied on *enumerating* all algorithms. We are motivated by the desire for constructions that do not suffer from prohibitive overhead, as conjectured to exist [6]. In the *finite executions* model (e.g., the subject of Juba and Sudan [11]) this overhead can be controlled by assuming that the server was “designed” to permit a typical user protocol to run efficiently with respect to some “beliefs” [12]. The constructions do not give good strategies in the infinite execution model, though, since they do not give a finite bound on the number of errors.¹

We observe that for a restricted kind of goal and sensing that is viable with respect to a class of simple user strategies, the problem of constructing a universal user from sensing is *precisely* the problem of learning the class of concepts corresponding to the simple strategies in the usual on-line learning model [3, 15] (Thm. 8). Thus, each solution to the on-line learning problem for a concept class yields a generic construction of a universal user from a sensing function that is viable with respect to the corresponding class – allowing us to translate an algorithm for efficiently learning linear threshold functions in Thm. 11 to an efficient strategy that works whenever a linear threshold strategy is viable – and vice-versa, allowing us to also translate the negative results. This settles the conjecture of Goldreich et al. [6], establishing that for natural classes of simple strategies and goals, universal user strategies can achieve polynomial overhead in the description length of the desired strategy. We further establish lower bounds that suggest limits to the power of universal users based on the kind of sensing discussed thus far—between the lower bounds that we obtain from the on-line learning model and the new lower bounds we obtain, basic sensing seems to only be adequate for the construction of *efficient* universal users in very simple settings. But, some natural kinds of richer feedback allow the construction of efficient universal users for correspondingly richer user strategies, and we suggest the exploration of such richer feedback as a next step towards constructing universal users of suitable efficiency.

¹ One of our results supports these notions, though: we show in Sec. 5.1 that in order for an efficient user for a class of servers to exist, there must be a common “belief” among *indistinguishable* servers in the class under which typical users are efficient.

2 Semantic Communication in Infinite Executions

The basic model involves a *system* of three interacting entities, a *user*, a *server*, and an *environment*. Each entity has some internal *state*, and they are each joined by a (two-way) communications channel that also has some fixed state on each *round*. Each entity has a *strategy* that specifies a distribution over new internal states and *outgoing messages* for the following round, given the entity's current state and *incoming messages*. We will generally denote the user strategies by U , server strategies by S , and environment strategies by E , respectively.

Thus, given strategies for each of the entities, the system is modeled by a discrete-time Markov process with a state space Ω . We will refer to the infinite sequence of random variables $\{X_t\}_{t=1}^{\infty}$ where X_t is the state of the system in round t as an *execution*; the execution produced by the interaction between a user strategy U , a server strategy S , and an environment strategy E will be denoted by (E, U, S) . An *execution started from state* σ_1 is an execution conditioned on $X_1 = \sigma_1$. We denote the space of internal states of the user, server, and environment by $\Omega^{(u)}$, $\Omega^{(s)}$, and $\Omega^{(e)}$, resp., and for $i, j \in \{u, e, s\}$, the state of the communication channel from i to j is a member of $\Omega^{(i,j)}$. Given a state of the system σ , we will let the respective superscripts denote the projection of σ on to the respective components—e.g., $\sigma^{(u,e)}$ is the user's outgoing message to the environment in σ . We wish to design algorithms for user strategies, to be executed by the user in pursuit of a *goal*:

Definition 1 (Goals and robust achievement). A goal is given by a pair (\mathcal{E}, R) , consisting of a non-deterministic environment strategy and a referee: A referee R is a function taking an (infinite) sequence of environment states to a boolean value; we say that an execution is successful if the referee evaluates to 1. A non-deterministic strategy \mathcal{E} is given by a set of (probabilistic) strategies. If a pair of user and server strategies is successful at (\mathcal{E}, R) for all $E \in \mathcal{E}$ and all initial states of the execution, we say that the pair robustly achieves the goal.

The interpretation of the environment's non-deterministic strategy is that the environment adversarially chooses a probabilistic strategy E from the set \mathcal{E} , effectively making its non-deterministic choices "up-front," allowing us to (sanely) analyze the resulting probabilistic system.

The algorithmic problem we consider is to compute a user strategy that achieves a fixed goal of communication with a large class of server strategies—a *universal* strategy for the goal:

Definition 2 (Universal user). A user strategy U is \mathcal{S} -universal with respect to a goal G if for every server strategy $S \in \mathcal{S}$, (U, S) robustly achieves the goal.

We will focus on universal strategies U for which the period of miscommunication in any execution (E, U, S) is uniformly bounded by a polynomial in a size parameter associated with the states of the system, $\text{sz} : \Omega \rightarrow \mathbb{N}$. The size will remain fixed throughout an execution (although the world's states may induce many different sizes associated with the same goal).

We will restrict our attention to goals in which time is divided into *sessions* of a fixed length. This is a special case of the *multi-session goals* of Goldreich et al. [6]; we prefer to consider only the special case here because the classes of user strategies we consider are particularly simple, only generating messages for a fixed number of rounds (initially, just one round). Our decision to only consider the special case will also have the added benefit of simplifying the other definitions we use (namely, the number of errors and the corresponding quantitative aspect of our “sensing functions”)²

Definition 3 (Fixed length multi-session goals). A goal $G = (\mathcal{E}, R)$ is said to be a k -round multi-session goal if the following hold:

1. (The environment’s states.) The environment’s states are partitioned into k sets, $\Omega_1^{(\text{e})}, \dots, \Omega_k^{(\text{e})}$. We refer to the elements of $\Omega_1^{(\text{e})}$ as start-session states, and the elements of $\Omega_k^{(\text{e})}$ as end-session states. In each case, the elements of $\Omega_i^{(\text{e})}$ are a pair consisting of an integer index and a contents.
2. (Starting a new session.) When in an end-session state, the environment non-deterministically moves to a start-session state with an incremented index; furthermore, this non-deterministic choice is independent of the contents of the end-session state.
3. (Execution of a session.) When the environment is in some state $(j, \sigma) \in \Omega_i^{(\text{e})}$ for $i \neq k$, $E(j, \sigma)^{(\text{e})}$ is a distribution over $\Omega_{i+1}^{(\text{e})}$ such that every element in its support has index j . Furthermore, the distribution over contents and messages is independent of the index and environment’s actual strategy.
4. (Compact referee) There is a temporal decision function R' taking end-session states to Boolean verdicts, and R is satisfied with an infinite execution iff R' evaluates to zero at most finitely many times.

The number of times R' evaluates to 0 in an execution is the number of errors.³

2.1 Sensing: Implicit Descriptions of Goals in Terms of Feedback

Success at a goal of communication is defined as a function of the environment’s states, which are not directly visible to the user. Naturally, it is helpful for the user to have some idea of whether or not its current communication strategy is working—indeed, it is *essential* if the user is to be able to reliably succeed in a single session, and many natural goals of communication allow a user to compute such feedback [10]. Although feedback is not *known* to be essential in any sense in multi-session goals, better feedback seems to allow the design of better user strategies [6]. In particular, in this work (as in previous works on semantic communication) we will focus on a relatively minimal kind of feedback that can be computed by the user during an execution.

² NB: the decision of “when to halt” is not at issue here, cf. [10, Chapters 2 and 5].

³ This is a simplification of the notion of errors used by Goldreich et al. [6] where the referee suspending a decision for too long was also considered to be an error.

Definition 4 (Sensing, safety, and viability). A sensing function V is a boolean function of the user's view. Let $G = (\mathcal{E}, R)$ be a fixed-length multi-session goal with temporal decision function R' and size parameter function $\text{sz} : \Omega \rightarrow \mathbb{N}$, let S be a server strategy, and let \mathcal{U} be a class of user strategies. For functions $B : \mathbb{N} \rightarrow \mathbb{N}$ and $\epsilon : \mathbb{N} \rightarrow [0, 1/3]$,

- We say that V is (B, ϵ) -safe for G w.r.t. \mathcal{U} and S if $\forall E \in \mathcal{E}, U \in \mathcal{U}, \sigma_1 \in \Omega$, whenever $R'(\sigma_1) = 0$, then w.p. $1 - \epsilon(\text{sz}(\sigma_1))$, either only $B(\text{sz}(\sigma_1))$ errors will occur, or for some $t \leq B(\text{sz}(\sigma_1))$, V evaluates to 0 in some state X_t of the execution (E, U, S) started from state σ_1 .
- We say that V is (B, ϵ) -viable for G w.r.t. \mathcal{U} and S if $\exists U \in \mathcal{U} \forall E \in \mathcal{E}, \sigma_1 \in \Omega$, w.p. at least $1 - \epsilon(\text{sz}(\sigma_1))$, after $B(\text{sz}(\sigma_1))$ rounds, V evaluates to 1 in every subsequent round in the execution (E, U, S) started from state σ_1 .

If $\epsilon \equiv 0$, we say that safety (or viability, resp.) holds perfectly, and we may refer to such a sensing function as B -safe (B -viable, resp.).

Thus, sensing functions encapsulate goal-specific feedback for solving a communications problem. It has been pointed out (by B. Patt-Shamir [17] and an anonymous reviewer) that the role of a sensing function is analogous to that of a *failure detector* in distributed computing [5, 9]. The main difference is that the feedback provided by a failure detector is generally a tentative set of faulty processes (which is the main obstacle in such settings), whereas sensing provides tentative feedback about success at a problem—for example, Juba and Sudan [11] use an interactive proof system to obtain feedback for the goal of computing a function. Although we will motivate a turn to richer feedback in Sec. 5, the main theorems of Goldreich et al. [6] show this simple type of feedback is sufficient for the construction of universal strategies for many goals:

Theorem 5 (On the existence of universal strategies – [6]). Let $G = (\mathcal{E}, R)$ be a fixed-length goal,⁴ \mathcal{U} be an enumerable set of user strategies, S be a set of server strategies, and $\epsilon : \mathbb{N} \rightarrow [0, 1/3]$ be such that the following hold:

1. There is a sensing strategy V s.t. $\forall U \in \mathcal{U}$, there is a bounding function B s.t. V is (B, ϵ) -safe with U and S for G , and $\forall S \in S \exists U \in \mathcal{U}$ s.t. for the bounding function B associated with U , V is (B, ϵ) -viable with respect to (U, S) . Furthermore, the mapping $U \mapsto B$ is computable.
Let \mathcal{B} denote the set of bounds that appear in the image of this mapping; that is, $\mathcal{B} = \{B_i : i \in \mathbb{N}\}$, where B_i is the bound for the i th user strategy in \mathcal{U} .
2. One of the following two conditions hold: (a) The viability condition holds perfectly (i.e., $\epsilon \equiv 0$). or (b) For every i , $B_{i+1} < B_i/2\epsilon$.

Then there is a S -universal user strategy U s.t. $\forall S \in S \exists B \in \mathcal{B} (U, S)$ robustly achieves the goal G with $O(B^2)$ errors, where the constant in the O -notation depends on S . Furthermore, if B -viability holds and the composition of any $U \in \mathcal{U}$ with the sensing and enumeration strategies also resides in \mathcal{U} , then, $\forall S \in S$, the complexity of U is bounded by the complexity of some fixed strategy in \mathcal{U} .

⁴ Actually, the theorem holds for the broader class of *compact* goals, not defined here.

Generic Users for Goals Implicitly Specified by Sensing. Theorem 5 gives a single, “generic” construction of a universal user from a sensing function, which can be applied to a variety of examples of sensing functions yielding universal users [6]. In this work, by contrast, we consider the capabilities and limits of such generic constructions, that is, the capabilities and limits of constructions based on *sensing*: rather than directly describing goals, we will assume that we are given a sensing function for a goal, and so the goal is *implicitly* described by the feedback available to the user and the class of strategies that suffice to achieve good feedback, as guaranteed by the viability condition. We then say that the construction is *generic* when it produces a universal user strategy that achieves any goal given only this information:

Definition 6 (Generic universal user). *For a class of goals in infinite executions \mathcal{G} , a class of user strategies \mathcal{U} , and functions $B : \mathcal{U} \times \mathbb{N} \rightarrow \mathbb{N}$, $s : \mathbb{N} \rightarrow \mathbb{N}$ and $v : \mathbb{N} \rightarrow \mathbb{N}$, we say that U is a B -error (\mathcal{U}, s, v) -generic universal user for \mathcal{G} if $\forall G \in \mathcal{G}$, any server S , and any sensing function V that is s -safe with S for G and v -viable with S with respect to \mathcal{U} for G , when U is provided the verdicts of V as auxiliary input, (U, S) robustly achieves G with $\min_{U_S \in \mathcal{U}: U_S \text{ } v\text{-viable}} B(U_S, \cdot)$ errors.*

There are two primary differences from the statement of Thm. 5: first, Thm. 5 allows for the bounding functions s and v for sensing to vary with the user strategy, and second, the number of errors incurred by Thm. 5 as stated was allowed to depend (arbitrarily) on the server S , whereas we demand that a generic universal user in the present sense obtains a bound that depends uniformly on the “best” user strategy in \mathcal{U} . That said, it turns out that for any enumerable class of user strategies \mathcal{U} , and $B(U_i, n) = 3i \max\{s(n), v(n)\}^2$, the proof of Thm. 5 actually constructs a B -error (\mathcal{U}, s, v) -generic universal user for any fixed-length goal. (Where U_i denotes the i th strategy in the given enumeration of \mathcal{U} .) As suggested, we want user strategies that only make *polynomially* many errors (in the length of the description of a target strategy in \mathcal{U} , and the size parameter of the execution). Goldreich et al. [6] showed, however, that a polynomial dependence *cannot* be achieved without *some* restrictions on the class of servers: briefly, servers with *passwords* force any strategy to suffer a number of errors that is exponential in the length of the password, and hence in the length of the description of the target strategy.

Thus, we will consider the problem of constructing a generic universal user that succeeds in a polynomial number of errors, given that it is viable with respect to a simple class of strategies. In particular, in Sec. 4 we show that if the class of user strategies \mathcal{U} in the viability condition is sufficiently simple, then we can efficiently identify a good strategy for the class of one-round multi-session goals; in Sec. 5.1, on the other hand, we will see that even for one-round multi-session goals, we will need richer kinds of feedback to efficiently compute good strategies when \mathcal{U} is not so simple. In both cases, the results will follow from an equivalence to *on-line learning* that we describe in more detail next.

3 On-line Learning is Equivalent to Semantic Communication with One-Round Goals

Given that an exponential number of errors in the description length of the desired user strategy is unavoidable *in general*, we would like to know *when* it can be avoided. Specifically, we would like to have some conditions under which we can develop efficient universal user strategies for goals in infinite executions. In this section, we investigate one such set of conditions: we will restrict our attention to multi-session goals of communication in which each round corresponds to a distinct session, and assume that sensing with very good safety and viability is available, in which moreover, the sensing function is viable with respect to some class of simple user strategies. Then, a generic construction of universal users from such sensing functions is equivalent to the design of an on-line learning algorithm, and we will find that generic constructions of universal user strategies exist for a variety of classes of simple user strategies.

The model of on-line learning that we consider was introduced by Bārzdiņš and Frievalds [3]. We assume that a *target concept* or *target function* f is drawn from some a priori fixed class of functions \mathcal{C} and the learning algorithm is run in an infinite sequence of *trials* consisting of the following steps: 1. The algorithm is provided an *instance* $x \in X$ as input. 2. The algorithm produces a *prediction* from Y . 3. The algorithm receives *reinforcement* feedback, indicating whether its prediction equaled $f(x)$. In Littlestone's [15] setting, $X = \{0, 1\}^n$ and $Y = \{0, 1\}$, and then n is a natural size parameter, and \mathcal{C} is finite, but we only require that a suitable notion of size can be defined for X and \mathcal{C} . The main parameter used to evaluate these algorithms is the worst case number of mistakes:

Definition 7 (Mistake bounded learning). *For a given on-line learning algorithm A and a concept class \mathcal{C} with size parameter $n : \mathcal{C} \rightarrow \mathbb{N}$, and any target concept $f : X \rightarrow Y$ for $f \in \mathcal{C}$, let $M_A(f)$ be the maximum, over all sequences of instances $\bar{x} = \{x_i \in X\}_{i=1}^\infty$, of the number of trials in which A outputs y such that $y \neq f(x_i)$. We then say that a learning algorithm A has mistake bound $m : \mathbb{N} \rightarrow \mathbb{N}$ if $\forall n' \in \mathbb{N} \max_{f \in \mathcal{C}: n(f)=n'} M_A(f) \leq m(n')$. If the state of A does not change when the algorithm receives positive feedback, then we say A is a conservative algorithm.*

Mistake bounded learning algorithms are easily made conservative.

We now show our main theorem, that the special case of semantic communication introduced here – generic users for one-round multi-session goals with a 1-safe and 1-viable sensing function – is *equivalent* to mistake-bounded learning.

Theorem 8. *Let \mathcal{G} be a class of one-round multi-session goals in which the user's incoming messages on each round are drawn from a set $\Omega^{(\cdot, u)}$, and its outgoing messages are from the set $\Omega^{(u, \cdot)}$. Let \mathcal{U} be a class of functions $\{U : \Omega^{(\cdot, u)} \rightarrow \Omega^{(u, \cdot)}\}$ with a size parameter $n : \mathcal{U} \rightarrow \mathbb{N}$. Then a conservative $m(n)$ -mistake bounded learning algorithm for \mathcal{U} is a m' -error $(\mathcal{U}, 1, 1)$ -generic universal user for \mathcal{G} for error bound $m'(U, n') = m(n(U)) + 1$, and conversely, a m' -error*

$(\mathcal{U}, 1, 1)$ -generic universal user for \mathcal{G} for error bound $m'(U, n') = m(n(U))$ is a $m(n)$ -mistake bounded learning algorithm for \mathcal{U} .

Proof. (\Rightarrow): We suppose we are given a conservative $m(n)$ -mistake bounded learning algorithm A for \mathcal{U} . We will show that A serves as a generic universal user as follows. Suppose we are given $G \in \mathcal{G}$, a server S , and a sensing function V that is 1-safe with S for G and 1-viable with S with respect to \mathcal{U} for G .

By the definition of 1-viability, there is $U_S \in \mathcal{U}$ s.t. if the user sends the same messages as U_S , after one round V will provide a positive indication on every round. Thus, U_S will correspond to the target concept. Each round of the execution will correspond to a trial for the learning algorithm. Suppose we provide the incoming messages to A as the instance, take the prediction of A as the outgoing messages, and provide the verdict of V on the following round as the reinforcement. In particular, if A sends the same outgoing message as U_S , A will receive a positive indication from the sensing function, which we take as positive feedback. Conversely, if V produces a negative indication, then A must not have sent the same outgoing message as U_S would have sent on the incoming messages in that round. V may also produce positive indications when the outgoing message A sent differs from what U_S would have sent, but since A is conservative, the state of A does not change. Now, since A is a $m(n)$ -mistake bounded learning algorithm for \mathcal{U} , it only receives negative reinforcement $m(n)$ times in any execution.

Since G is a 1-round multi-session goal, R' evaluates to 0 or 1 on each round; when it evaluates to 0, the 1-safety of V guarantees that either that is the only error that will occur, or that V evaluates to 0 in the current round. V is therefore only allowed to evaluate to 1 when an error occurs once, so our strategy therefore makes at most $m(n) + 1$ errors.

(\Leftarrow): Let a target concept $U \in \mathcal{U}$ and any sequence of instances $\bar{x} = \{x_i \in \Omega^{(e,u)} \times \Omega^{(s,u)}\}_{i=1}^{\infty}$ be given. We will show how to embed the corresponding sequence of trials into a one-round multi-session goal with a 1-safe and 1-viable sensing function for some server S .

Consider the following one-round multi-session goal $G_U = (\mathcal{E}, R_U)$: the environment non-deterministically chooses $(\sigma_i^{(e,u)}, \sigma_{i+1}^{(s,u)}) \in \Omega^{(e,u)} \times \Omega^{(s,u)}$ for each round i , and sends $(\sigma_i^{(e,u)}, b)$ to the user and $\sigma_{i+1}^{(s,u)}$ to the server. The temporal decision function R'_U for the referee R_U then is satisfied in session i if the user returns $U(\sigma_i^{(e,u)}, \sigma_i^{(s,u)})$. Let S be the server that forwards the message it received from the environment in the previous round to the user in the current round. Let V_U be the sensing function that returns 1 if the user's message on the i th round is $U(\sigma_i^{(e,u)}, \sigma_i^{(s,u)})$. Note that when the user executes with S , V_U computes R'_U , so V_U is 1-safe with S for G_U . Furthermore, whenever the user sends the same message as $U \in \mathcal{U}$, V_U is trivially satisfied on the following round, so V_U is also 1-viable with S with respect to \mathcal{U} for G_U . We can embed \bar{x} in an execution in the following way: let the execution start from the state where $\sigma^{(e,u)} = x_1^{(e,u)}$, $\sigma^{(s,u)} = x_1^{(s,u)}$, and $\sigma^{(e,s)} = x_2^{(s,u)}$, and suppose that the environment's

nondeterministic choice for the i th round is $(x_{i+1}^{(\mathbf{e}, \mathbf{u})}, x_{i+2}^{(\mathbf{s}, \mathbf{u})})$. Then, we can check that in each i th round of this execution, the user receives x_i .

Now, supposing that we are given a m' -error $(\mathcal{U}, 1, 1)$ -generic universal user for \mathcal{G} A , for every target concept $U \in \mathcal{U}$, A robustly achieves G_U with $m'(U, n') = m(n(U))$ errors when given the feedback from V_U in an execution with S —in particular, in the execution we constructed for a given sequence of trials \bar{x} . By definition of G_U , now, A makes an error in the i th round iff it does not send the same messages as U in that round, so when A is provided the feedback from V_U , it makes at most $m(n(U))$ mistakes in the sequence of trials \bar{x} . We now note that V_U computes the same function as the learner’s reinforcement, so when A is provided access to the reinforcement instead of A , it still only makes $m(n(U))$ mistakes, as needed.

4 Universal Users from On-line Learning Algorithms

We now exploit Thm. 8 to obtain generic constructions of efficient universal users for one-round multi-session goals. In particular, we show that for a variety of halfspace learning, we can confirm one of the main conjectures of Goldreich et al. [6]: there is a universal strategy for a nontrivial class of servers with polynomial overhead. The particular variant we consider has the feature that, unlike previous algorithms (with the exception of the perceptron algorithm) the number of mistakes does not depend on the size of the examples.

Definition 9 (Linear threshold strategies). *The class of linear threshold strategies in n dimensions with b -bit weights, $\mathcal{U}_{LT(n,b)}$, is as follows. We identify the user’s incoming messages with \mathbb{Q}^n . Then, for any weight vector $w \in \{-2^{b+1} + 1, \dots, 2^{b+1} - 1\}^n$ and threshold $c \in \{-2^{b+1} + 1, \dots, 2^{b+1} - 1\}$, the user strategy that on incoming message $x \in \mathbb{Q}^n$ sends $U_{w,c}(x)$ to the server and environment where $U_{w,c}(x) = 1$ if $\sum_{i=1}^n w_i x_i \geq c$ and 0 otherwise is in $\mathcal{U}_{LT(n,b)}$.*

An algorithm for efficiently learning linear threshold functions with general weights was given by Maas and Turán [16], based on a reduction to the problem of finding feasible points in convex programs given by a separation oracle:

Definition 10 (Convex feasibility with a separation oracle). *Let a convex set $K \subset \mathbb{R}^n$ be given. For $r \in \mathbb{N}$, we say that K has guarantee r if the volume of $K \cap \text{Ball}(0, r)$ is at least r^{-n} . A separation oracle for K answers queries of the form $x \in \mathbb{Q}^n$ with “yes” if $x \in K$ and otherwise non-deterministically returns a vector $v \in \mathbb{Q}^n$ and $c \in \mathbb{Q}$ such that $\langle x, v \rangle \geq c$, but that for every $y \in K$, $\langle y, v \rangle < c$. If the longest vector v returned by the separation oracle is ℓ bits, we will say that the oracle is ℓ -bounded.*

Now, we say that an oracle algorithm $A^{(\cdot)}$ solves the search problem of convex feasibility with a ℓ -bounded separation oracle in time $t(n, \log r, \ell)$ and query complexity $q(n, \log r, \ell)$ if, for any ℓ -bounded separation oracle for a convex body K with guarantee r , $A^{(\cdot)}$ produces a point in K in time $t(n, \log r, \ell)$, and making at most $q(n, \log r, \ell)$ queries to the oracle.

There are efficient algorithms for solving convex programs in this model, yielding algorithms for learning linear threshold functions. The one by Vaidya [18], and an algorithm based on random walks [4] both make at most $O(n \log r)$ queries.

Actually, the above algorithm(s) were for a different problem than the one we consider here: in their model, the *instance space* was assumed to be b -bit integer points (as opposed to \mathbb{Q}^n) while the *linear threshold functions* used arbitrary precision (though $\text{poly}(b)$ bits suffice to represent all linear thresholds on the b -bit instance space), and the time and query complexity of their algorithm depended the size of the instances. Although it is clear that we cannot hope to eliminate the dependence of the computation time of the size of the instances, it turns out that the dependence on the size of instances in the mistake bound can be eliminated in our setting, using techniques for solving convex programming problems when the convex set K is not of full dimension [7, 8].

Theorem 11. *Suppose there is an algorithm that solves convex feasibility with a ℓ -bounded separation oracle in time $t(n, \log r, \ell)$ and query complexity $q(n, \log r)$ for polynomials t and q . Then there is a $m(n, b)$ -mistake bounded on-line learning algorithm for $\mathcal{U}_{LT(n, b)}$ running in time $t'(n, \log b, \ell)$ on each trial for a polynomial t' where ℓ is the length in bits of the longest received instance $x \in \mathbb{Q}^n$, and $m(n, b) = O(n \cdot q(n, b + \log n))$.*

Sketch of proof. The weight vector and threshold of the function $U_{w,c}$ is an integer point in $[-2^{b+1} + 1, 2^{b+1} - 1]^{n+1}$, which is a convex set, and a counterexample x to a proposed linear threshold (w', c') defines a hyperplane such that either $\langle (w', c'), (x, -1) \rangle \geq 0 > \langle (w, c), (x, -1) \rangle$ or $\langle (w, c), (x, -1) \rangle \geq 0 > \langle (w', c'), (x, -1) \rangle$, and either way $(x, -1)$ and 0 gives us a separating hyperplane.

Thus, we pass our counterexamples to the feasibility algorithm, and the algorithm terminates once it finds some point (\tilde{w}, \tilde{c}) s.t. any halfspace of the remaining feasible set not containing (\tilde{w}, \tilde{c}) has volume less than the guarantee. Then, if we get another counterexample, the hyperplanes given by our counterexamples define a set containing (w, c) of volume less than the guarantee.

By choosing the guarantee sufficiently small, we will be able to ensure that there is a hyperplane such that all of the points with integer coordinates (including the target (w, c)) lie in this hyperplane; we will then be able to find this hyperplane, and reduce to the problem of finding a feasible point in a lower dimensional space by projecting onto it. After we repeat this process $n+1$ times, (w, c) is uniquely determined.

Algorithms for k -round Goals and Strategies with Larger Message Spaces. We exclusively focused on one-round goals and user strategies with Boolean message spaces here. Naturally, this is because the notion of feedback we obtain from sensing only agrees with the usual notions of feedback in on-line learning for Boolean functions, and conversely, on-line learning usually does not handle stateful “concepts.” It turns out, though, that a result due to Auer and Long [2] allows us to slightly relax both of these restrictions, giving efficient algorithms for, e.g., $O(\log \log n)$ -round goals or messages of size $O(\log \log n)$.

5 Richer Feedback and the Limitations of Basic Sensing

5.1 Limitations of Basic Sensing

We start by presenting a strong lower bound when the space of messages is large. We obtain this via a lower bound on the number of algorithms that an *oblivious* schedule (including Levin-style enumerations and sampling algorithms) must use to escape from a “bad” set of states whenever a class of servers does not have a common prior under which escaping the bad set is easy. We then refine it to handle adaptive algorithms under the assumption that executions with servers in their respective collections of bad states produce indistinguishable user views. The key definition enabling us to state these theorems captures subsets of states of the execution that are hard for typical users to escape:

Definition 12 (Effectively closed). *For a non-deterministic environment strategy \mathcal{E} with size parameter $\text{sz} : \Omega \rightarrow \mathbb{N}$, a server S , a distribution over user strategies P , $t : \mathbb{N} \rightarrow \mathbb{N}$, and $\gamma : \mathbb{N} \rightarrow [0, 1]$, we say that the set of server and environment states $\Theta \subseteq \Omega^{(s)} \times \Omega^{(e)}$ is (t, γ) -effectively closed with respect to P if, $\forall (\sigma^{(s)}, \sigma^{(e)}) \in \Theta, t \leq t(\text{sz}(\sigma))$ the probability that, for a user strategy U drawn from P , $(X_t^{(s)}, X_t^{(e)}) \in \Theta$ is at least $1 - \gamma(\text{sz}(\sigma))$ in the execution (E, U, S) started from σ (for the initial state $\sigma^{(u)}$ specified by U), where the probability is taken over the choice of U from P and the random evolution of the execution.*

These lower bounds also turn out to justify the features of a model introduced in another attempt to refine the notions of semantic communication to reduce the overhead: the “beliefs” model of communication [12]. There, it was assumed that a server was designed to be efficient with respect to “typical” users under a given “belief” (prior) distribution; a user strategy for which communication has low overhead whenever a user has beliefs that are close to those of the server designer then exists. Of course, in order for this approach to guarantee low overhead with respect to an entire class of servers, there must be a *common* belief under which *all* of the servers were designed well. Our lower bound establishes that this common belief was essential in a sense: suppose that we wish to achieve some goal that cannot be achieved while the execution is in one of our “bad sets.” Then, our lower bounds demonstrate that when the class of servers lacks a suitable common notion of “natural users” under which escaping the bad sets is easy, a universal user cannot be too efficient, and the best possible running time is roughly obtained by sampling from the best common distribution.

Theorem 13. *Let $G = (\mathcal{E}, R)$ be a goal and \mathcal{S} be a class of servers s.t. $\forall E \in \mathcal{E}, S \in \mathcal{S}$ we have designated some set of pairs of states of E and S , $\Theta_{S,E}$. Let $\delta \in [0, 1]$ be given. Now, suppose that $\exists (t, \epsilon) \in \mathbb{N} \times [0, 1]$ s.t. for every distribution over user strategies from the class \mathcal{U} , Q , $\exists E \in \mathcal{E}, S \in \mathcal{S}$ such that $\Theta_{S,E}$ is (t, ϵ) -effectively closed with respect to Q in E . Then, for any sequence of user strategies and running times $(U_1, t_1), (U_2, t_2), \dots$ s.t. each $t_i \leq t$, $\exists S \in \mathcal{S}, E \in \mathcal{E}$ s.t. if in the execution where the user runs U_1 for t_1 steps, U_2 for t_2 steps, and so on, the first step τ for which $(X_\tau^{(s)}, X_\tau^{(e)}) \notin \Theta_{S,E}$ is at most $\sum_{i=1}^k t_i$ with probability at least δ , then $k \geq \frac{1}{\epsilon(1+1/\delta)}$.*

Proof. In a zero-sum game between a “user” player and a “server/environment” player, in which the strategy sets are \mathcal{U} and $\mathcal{S} \times \mathcal{E}$, and the payoff of U with (S, E) is given by the maximum probability, over executions starting from initial states from $\Theta_{S,E}$, that the execution exits $\Theta_{S,E}$ in t steps, our assumption on distributions over \mathcal{U} means that the server/environment player always has a good strategy. Loomis’ Theorem then yields that there is some distribution \tilde{Q} over $\mathcal{S} \times \mathcal{E}$ such that when any user strategy $U_1 \in \mathcal{U}$ that is run for $t_1 \leq t$ steps with a server and environment pair (S, E) drawn from \tilde{Q} and started in any state of $\Theta_{S,E}$, the probability that the execution (E, U_1, S) enters a state σ such that $(\sigma^{(s)}, \sigma^{(e)}) \notin \Theta_{S,E}$ is at most ϵ .

It then follows by induction on k that, given that the execution never entered a state σ such that $(\sigma^{(s)}, \sigma^{(e)}) \notin \Theta_{S,E}$ during the runs of U_1, \dots, U_{k-1} , during the t_k step run of U_k , the probability that the execution enters such a state σ is at most $\frac{\epsilon}{1-k\epsilon}$. Therefore, while $k\epsilon < 1$, a union bound gives a total probability of exiting $\Theta_{S,E}$ in the first k runs of at most $\frac{k\epsilon}{1-k\epsilon}$. In particular, some (S^*, E^*) in the support of \tilde{Q} must give $(U_1, t_1), \dots, (U_k, t_k)$ probability at most $\frac{k\epsilon}{1-k\epsilon}$ of exiting Θ_{S^*,E^*} . Thus, if we exit Θ_{S^*,E^*} with probability at least δ by the end of the k th run, this requires $k \geq \frac{1}{\epsilon(1+1/\delta)}$, as needed.

We can extend Theorem 13 to cover adaptive algorithms, given that the servers generate indistinguishable views so long as they remain in the bad states. The key point is that in this case, the algorithm generates a schedule nearly independently of the actual server, essentially reducing to the earlier analysis.

Corollary 14. *Let $G, \mathcal{U}, \mathcal{S}$, sets of states $\Theta_{S,E}$ for each $E \in \mathcal{E}$ and each $S \in \mathcal{S}$, and $\delta \in [0, 1]$ be given as in Theorem 13. Suppose that $\exists E \in \mathcal{E}$ s.t. for every distribution Q over \mathcal{U} , $\exists S \in \mathcal{S}$ s.t. $\Theta_{S,E}$ is (t, ϵ) -effectively closed with respect to Q in E . Suppose further that $\forall U \in \mathcal{U}, S_1 \in \mathcal{S}, S_2 \in \mathcal{S}, (\sigma_1^{(s)}, \sigma_1^{(e)}) \in \Theta_{S_1,E}, \exists (\sigma_2^{(s)}, \sigma_2^{(e)}) \in \Theta_{S_2,E}$ s.t. the distribution over user views in the first t steps of the execution (E, U, S_1) started from a state $(\sigma_1^{(e)}, \sigma_1^{(u)}, \sigma_1^{(s)})$ is γ -statistically close to the user view in the first t steps of the execution (E, U, S_2) started from the state $(\sigma_2^{(e)}, \sigma_2^{(u)}, \sigma_2^{(s)})$. Then for any algorithm U that on each step either starts running a new strategy from \mathcal{U} from its initial state or continues running the same strategy from \mathcal{U} for up to at most t steps, $\exists S \in \mathcal{S}$ s.t. if U reaches a state σ s.t. $(\sigma^{(s)}, \sigma^{(e)}) \notin \Theta_{S,E}$ w.p. $\geq \delta$ by running up to k strategies from their initial states, then $k \geq \frac{1}{\epsilon(1+1/\delta)+\gamma/\delta}$.*

Note that we can also apply Corollary 14 to the case where the sets $\Theta_{S,E}$ are chosen to be states of the execution where a given sensing function fails. This will allow us to obtain lower bounds on the performance of any user strategy that uses such a sensing function. Recall that Goldreich et al. [6] showed simple conditions implied an exponential lower bound on the number of errors made by universal users for classes including password-protected servers. Now, for the case of one-round multi-session goals, we know that a lower bound on the number of rounds before the referee is satisfied translates into a lower bound on the number

of errors. In this case, we obtain lower bounds with respect to many other classes of user strategies with large message spaces.

Theorem 15. *Let \mathcal{U} be a class of stateless user strategies computing functions $U : X \rightarrow Y$ s.t. for every outgoing message y and incoming message x , some $U \in \mathcal{U}$ satisfies $U(x) = y$. Let $G = (\mathcal{E}, R)$ be a one-round goal in which the environment non-deterministically selects an infinite sequence of elements of X , $\mathcal{E} = \{E_{\bar{x}} : \bar{x} = \{x_i \in X\}_{i=1}^{\infty}\}$, s.t. each i th session consists of $E_{\bar{x}}$ sending x_i to both the user and server. The referee's temporal decision function R' is satisfied iff the server receives a message consisting of "1" from the server. Now, let the class of servers $\mathcal{S} = \mathcal{S}(\mathcal{U})$ be s.t. $\forall U \in \mathcal{U}$, $\exists S_U \in \mathcal{S}(\mathcal{U})$ s.t. in each round, the server stores the message $x \in X$ it received from the user until the next round; the server then sends "1" to the user and environment if the user sent a message $y \in Y$ on that round such that $y = U(x)$ for the previous message x that the server received from the environment, and sends "0" to the user and environment otherwise. Then for any user strategy, $\exists S^* \in \mathcal{S}(\mathcal{U})$ s.t. the user strategy makes at least $|Y|/3$ errors w.p. $\geq 1/2$, and at least $|Y|/2$ errors in the worst case.*

It is easy to construct *specific* examples for which learning functions on a message space Y requires an overhead of $|Y| - 1$ —Auer and Long [2] describe one such example. Theorem 15, on the other hand, applies to many simple cases of interest, such as linear transformations:

Example 16 (Lower Bound for Linear Transformations). Let \mathcal{U} be the class of linear transformations $A : \mathbb{F}^n \rightarrow \mathbb{F}^n$ for some finite field \mathbb{F} . Suppose that the instance space is given by $\mathbb{F}^n \setminus \{0\}$. Now, for any nonzero $x, y \in \mathbb{F}^n$ we know that there is some $A_{x,y}$ such that $A(x) = y$. So, Thm. 15 shows that any on-line learning algorithm makes at least $(|\mathbb{F}|^n - 1)/2$ mistakes in the worst case.

We can also use Thm. 8 directly to recover impossibility results for learning Boolean functions. Angluin [4] noted that an efficient mistake-bounded learning algorithm gives an efficient PAC-learning algorithm, so negative results for efficient *PAC-learning* also translate to negative results for generic universal users, and so even most natural classes of Boolean strategies do not have efficient universal users under cryptographic assumptions (cf. [13, 14]).

5.2 Richer Feedback

Thus, Thm 8 gives a reasonable picture of which classes of strategies we can efficiently learn generically from basic sensing – i.e., with success/fail feedback – and which classes we cannot. Unfortunately, the boundary falls short of where we would like—we can only learn strategies with very small message spaces, and under standard cryptographic assumptions, even then only for fairly simple classes of user strategies.

Recall that our motivation for focusing on this notion of sensing was that we had results, such as Thm. 5, effectively saying that whenever sensing was possible, it was feasible to achieve a goal with any helpful server. As we are primarily interested in user strategies that do not experience such overhead as that suffered

by these constructions, though, we find that we are strongly motivated to investigate some notions of *stronger* feedback (that may not always be available). That is, we view negative results showing that $(\mathcal{U}, 1, 1)$ -generic universal users cannot be mistake-efficient and/or time-efficient as limitations of *basic sensing*, and so we seek alternative notions of sensing that do not suffer these limitations. For example, Auer and Long [2] showed how some useful, richer kinds of feedback can be simulated given only basic sensing, but only if the feedback is still limited in the sense that it can be simulated by a logarithmic number of queries; but if these kinds of feedback are directly available, then since we don't need to simulate the feedback, we don't experience this overhead.

Example 17 (Efficient Universal Linear Transformation Strategies from Richer Sensing). Consider the class of user strategies computing linear transformations $A : \mathbb{F}^n \rightarrow \mathbb{F}^n$ for some finite field \mathbb{F} , as considered in Example 16. There, we saw that given only basic sensing, any generic universal strategy experiences at least $(|\mathbb{F}|^n - 1)/2$ errors for one-round multi-session goals, where Auer and Long's technique yields a universal strategy making $\tilde{O}(|\mathbb{F}|^n)$ errors.

Suppose now that we had richer sensing feedback, that not only provided positive or negative indications, but on a negative indication also provided some index $i \in \{1, \dots, n\}$ such that if on the previous round we received an incoming message vector $x \in \mathbb{F}^n$ and responded with $y \in \mathbb{F}^n$, a viable linear transformation strategy A would not have responded with $(A(x))_i = y_i$. Then, e.g., for \mathbb{F}_2 , we could use Gaussian elimination to learn each i th row of a viable linear transformation on \mathbb{F}_2^n in n mistakes, for n^2 mistakes (and time $O(n^3)$ per round) overall. Auer and Long's technique can then also be used to simulate access to $(A(x))_i$ over \mathbb{F}_q for $q > 2$ with an overhead of $\tilde{O}(q)$ mistakes, thus allowing us to use essentially the same learning algorithm over \mathbb{F}_q . As long as the field size is still small, this yields polynomial error and polynomial time bounded universal strategies, in contrast to the exponential lower bound of Example 16.

Similarly, if the sensing function also told us that the user's messages in some i th round of the previous session was unsatisfactory, then this feedback would enable us to construct efficient universal users for k -round multi-session goals, given that there are stateless viable user strategies such that a time and mistake efficient on-line learning algorithm for the class of user strategies when restricted to any single round (even if k is polynomially large).

Conclusion. These observations suggest the following program for future work. In order to obtain flexible protocols for interesting goals with low overhead, we could first try to identify what kind of feedback is available in those goals, and second, try to determine which classes of strategies can be efficiently learned from such feedback. The hope is that with rich enough feedback, reasonably strong classes of strategies can be learned.

Acknowledgments. This direction was motivated by conversations with Leslie Kaelbling and Leslie Valiant. We thank Oded Goldreich for insightful comments that improved the presentation. Finally, we thank the anonymous referees for their comments.

References

- [1] Angluin, D.: Queries and concept learning. *Mach. Learn.* 2(4), 319–342 (1988)
- [2] Auer, P., Long, P.M.: Structural results about on-line learning models with and without queries. *Mach. Learn.* 36(3), 147–181 (1999)
- [3] Bārzdiņš, J., Freivalds, R.: On the prediction of general recursive functions. *Soviet Math. Dokl.* 13, 1224–1228 (1972)
- [4] Bertsimas, D., Vempala, S.: Solving convex programs by random walks. *J. ACM* 51(4), 540–556 (2004)
- [5] Chandra, T.D., Toueg, S.: Unreliable failure detectors for reliable distributed systems. *J. ACM* 43(2), 225–267 (1996)
- [6] Goldreich, O., Juba, B., Sudan, M.: A theory of goal-oriented communication. *Tech. Rep. TR09-075, ECCC* (2009)
- [7] Grötschel, M., Lovász, L., Schrijver, A.: Geometric methods in combinatorial optimization. In: Pulleybank, W.R. (ed.) *Proc. Silver Jubilee Conf. on Combinatorics. Progress in Combinatorial Optimization*, pp. 167–183. Academic Press, New York (1984)
- [8] Grötschel, M., Lovász, L., Schrijver, A.: *Geometric algorithms and combinatorial optimization*, 2nd edn. Springer, New York (1993)
- [9] Jayanti, P., Toueg, S.: Every problem has a weakest failure detector. In: 27th PODC (2008)
- [10] Juba, B.: Universal Semantic Communication. Ph.D. thesis, MIT (2010)
- [11] Juba, B., Sudan, M.: Universal semantic communication I. In: 40th STOC (2008)
- [12] Juba, B., Sudan, M.: Efficient semantic communication via compatible beliefs. In: 2nd Innovations in Computer Science (2011)
- [13] Kearns, M., Valiant, L.: Cryptographic limitations on learning Boolean formulae and finite automata. *J. ACM* 41, 67–95 (1994)
- [14] Kharitonov, M.: Cryptographic hardness of distribution-specific learning. In: 25th STOC, pp. 372–381 (1993)
- [15] Littlestone, N.: Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn.* 2(4), 285–318 (1988)
- [16] Maass, W., Turán, G.: How fast can a threshold gate learn? In: Hanson, S.J., Drastal, G.A., Rivest, R.L. (eds.) *Computational Learning Theory and Natural Learning Systems: Constraints and Prospects*, vol. 1, pp. 381–414. MIT Press, Cambridge (1994)
- [17] Patt-Shamir, B.: Personal communication (2010)
- [18] Vaidya, P.M.: A new algorithm for minimizing convex functions over convex sets. *Mathematical Programming* 73(3), 291–341 (1996)

Accelerated Training of Max-Margin Markov Networks with Kernels

Xinhua Zhang¹, Ankan Saha², and S.V.N. Vishwanathan³

¹ Department of Computing Science, University of Alberta, Edmonton, Canada
`xinhua2@cs.ualberta.ca`

² Department of Computer Science, University of Chicago, Chicago, IL, USA
`ankans@cs.uchicago.edu`

³ Department of Statistics and Computer Science, Purdue University, IN, USA
`vishy@stat.purdue.edu`

Abstract. Structured output prediction is an important machine learning problem both in theory and practice, and the max-margin Markov network (M^3N) is an effective approach. All state-of-the-art algorithms for optimizing M^3N objectives take at least $O(1/\epsilon)$ number of iterations to find an ϵ accurate solution. [1] broke this barrier by proposing an excessive gap reduction technique (EGR) which converges in $O(1/\sqrt{\epsilon})$ iterations. However, it is restricted to Euclidean projections which consequently requires an intractable amount of computation for each iteration when applied to solve M^3N . In this paper, we show that by extending EGR to Bregman projection, this faster rate of convergence can be retained, and more importantly, the updates can be performed efficiently by exploiting graphical model factorization. Further, we design a kernelized procedure which allows all computations per iteration to be performed at the same cost as the state-of-the-art approaches.

1 Introduction

In the supervised learning setting, one is given a training set of labeled data points and the aim is to learn a function which predicts labels on unseen data points. Sometimes the label space has a rich internal structure which characterizes the combinatorial or recursive inter-dependencies of the application domain. It is widely believed that capturing these dependencies is critical for effectively learning with *structured output*. Examples of such problems include sequence labeling, context free grammar parsing, and word alignment. However, parameter estimation is generally hard even for simple linear models, because the size of the label space is potentially exponentially large (see e.g. [2]). Therefore it is crucial to exploit the underlying conditional independence assumptions for the sake of computational tractability. This is often done by defining a graphical model on the output space, and exploiting the underlying graphical model factorization to perform computations.

Research in structured prediction can broadly be categorized into two tracks: Using a maximum a posterior estimate from the exponential family results in conditional random fields [CRFs, 3], and a maximum margin approach leads to max-margin Markov networks [M^3Ns , 4]. Unsurprisingly, these two approaches

share many commonalities: First, they both minimize a regularized risk with a square norm regularizer. Second, they assume that there is a joint feature map ϕ which maps (\mathbf{x}, \mathbf{y}) to a feature vector in \mathbb{R}^p .¹ Third, they assume a label loss $\ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i)$ which quantifies the loss of predicting label \mathbf{y} when the correct label of input \mathbf{x}^i is \mathbf{y}^i . Finally, they assume that the space of labels \mathcal{Y} is endowed with a graphical model structure and that $\phi(\mathbf{x}, \mathbf{y})$ and $\ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i)$ factorize according to the cliques of this graphical model. The main difference is in the loss function employed. CRFs minimize the L_2 -regularized logistic loss:

$$J(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|_2^2 + \frac{1}{n} \sum_{i=1}^n \log \sum_{\mathbf{y} \in \mathcal{Y}} \exp (\ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i) - \langle \mathbf{w}, \phi(\mathbf{x}^i, \mathbf{y}^i) - \phi(\mathbf{x}^i, \mathbf{y}) \rangle), \quad (1)$$

while the M³Ns minimize the L_2 -regularized hinge loss

$$J(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|_2^2 + \frac{1}{n} \sum_{i=1}^n \max_{\mathbf{y} \in \mathcal{Y}} \{ \ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i) - \langle \mathbf{w}, \phi(\mathbf{x}^i, \mathbf{y}^i) - \phi(\mathbf{x}^i, \mathbf{y}) \rangle \}. \quad (2)$$

A large body of literature exists on efficient algorithms for minimizing the above objective functions. A summary of existing methods, and their convergence rates (iterations needed to find an ϵ accurate solution) can be found in Table 1. The ϵ accuracy of a solution can be measured in many different ways and different algorithms employ different but somewhat related stopping criterion. Some produce iterates \mathbf{w}_k in the primal space and bound the *primal gap* $J(\mathbf{w}_k) - \min_{\mathbf{w}} J(\mathbf{w})$. Some solve the dual problem $D(\boldsymbol{\alpha})$ with iterates $\boldsymbol{\alpha}_k$ and bound the *dual gap* $\max_{\boldsymbol{\alpha}} D(\boldsymbol{\alpha}) - D(\boldsymbol{\alpha}_k)$. Some bound the *duality gap* $J(\mathbf{w}_k) - D(\boldsymbol{\alpha}_k)$, and still others bound $J(\mathbf{w}_k) - \min_{\mathbf{w}} J_k(\mathbf{w})$ where J_k is a uniform lower bound of J . This must be borne in mind when interpreting the convergence rates in Table 1.

Since (1) is a smooth convex objective, classical methods such as L-BFGS can directly be applied [5]. Specialized solvers also exist. For instance a primal algorithm based on bundle methods was proposed by [6], while a dual algorithm for the same problem was proposed by [7]. Both algorithms converge at $O(\frac{1}{\lambda} \log(1/\epsilon))$ rates to an ϵ accurate solution, and, remarkably, their convergence rates are independent of n the number of data points, and $|\mathcal{Y}|$ the size of the label space. It is widely believed in optimization (see e.g. Section 9.3 of [8]) that unconstrained smooth strongly convex objective functions can be minimized in $O(\log(1/\epsilon))$ iterations, and these specialized optimizers also achieve this rate.

On the other hand, since (2) is a non-smooth convex function, efficient algorithms are harder to come by. SVM-Struct was one of the first specialized algorithms to tackle this problem, and [9] derived an $O(G^2/\lambda\epsilon^2)$ rate of convergence. Here G denotes the maximum L_2 norm of the feature vectors $\phi(\mathbf{x}^i, \mathbf{y})$. By refining their analysis, [6] proved a $O(G^2/\lambda\epsilon)$ rate of convergence for a related but more general algorithm, which they called bundle methods for regularized risk minimization (BMRM). At first glance, it looks like the rates of convergence of these algorithms are independent of $|\mathcal{Y}|$. This is somewhat misleading because, although the dependence is not direct, the convergence rates depend on G , which is in turn implicitly related to the size of \mathcal{Y} .

¹ We discuss kernels and associated feature maps into a Reproducing Kernel Hilbert Space (RKHS) in Section 4.3.

Table 1. Comparison of specialized optimization algorithms for training structured prediction models. Primal-dual methods maintain estimation sequences in both primal and dual spaces. Details of the oracle will be discussed in Section 5. The convergence rate highlights the dependence on both ϵ and some “constants” that are often hidden in the O notation: n , λ , and the size of the label space $|\mathcal{Y}|$. The convergence rate of SMO on M^3N is derived from [12, Corollary 17], noting the dual problem (19) is so-called pairable. It enjoys linear convergence $O(\log \frac{1}{\epsilon})$ when the dual objective is positive definite (pd), and $O(\frac{1}{\epsilon})$ when it is positive semi-definite (psd). The term G in the convergence rate denotes the maximum L_2 norm of the features vectors $\phi(\mathbf{x}^i, \mathbf{y})$. The convergence rate of Extragradient depends on λ in an indirect way.

Optimization algorithm	Primal/Dual Type of gap		Oracle for M^3N	Convergence rate	
	CRF	M^3N			
BMRM [6]	primal	\geq primal gap	max	$O\left(\frac{1}{\lambda} \log \frac{1}{\epsilon}\right)$	$O\left(\frac{G^2}{\lambda \epsilon}\right)$
SVM-Struct [9]	primal-dual	constraint violation	max	<i>n/a</i>	$O\left(\frac{G^2}{\lambda \epsilon^2}\right)$
Extragradient [10]	primal-dual	duality gap	exp	<i>n/a</i>	$O\left(\frac{\log \mathcal{Y} }{\epsilon}\right)$
Exponentiated gradient [7]	dual	dual gap	exp	$O\left(\frac{1}{\lambda} \log \frac{1}{\epsilon}\right)$	$O\left(\frac{G^2 \log \mathcal{Y} }{\lambda \epsilon}\right)$
SMO [11, Chapter 6]	dual	dual gap	max	<i>n/a</i>	psd: $O\left(n \mathcal{Y} \frac{1}{\lambda \epsilon}\right)$ pd: $O\left(n \mathcal{Y} \log \frac{1}{\epsilon}\right)$
Our algorithm	primal-dual	duality gap	exp	<i>n/a</i>	$O\left(G \sqrt{\frac{\log \mathcal{Y} }{\lambda \epsilon}}\right)$

Algorithms which optimize (2) in the dual have also been developed. For instance, the algorithm proposed by [4] performs exponentiated gradient descent in the dual and converges at $O\left(\frac{\log |\mathcal{Y}|}{\lambda \epsilon}\right)$ rates. Again, these rates of convergence are not surprising given the well established lower bounds of [13] who show that, in general, non-smooth optimization problems cannot be solved in fewer than $\Omega(1/\epsilon)$ iterations by solvers which treat the objective function as a black box.

In this paper, we present an algorithm that provably converges to an ϵ accurate solution of (2) in $O\left(\sqrt{\frac{\log |\mathcal{Y}|}{\lambda \epsilon}}\right)$ iterations. This does not contradict the lower bound because our algorithm is not a general purpose black box optimizer. In fact, it exploits the special form of the objective function (2). Before launching into the technical details we would like to highlight some important features of our algorithm. First, compared to existing algorithms our convergence rates are better in terms of $|\mathcal{Y}|$, λ , and ϵ . Second, our convergence analysis is tighter in that our rates are with respect to the duality gap. Not only is the duality gap computable, it also upper bounds the primal and dual gaps used by other algorithms. Finally, our cost per iteration is comparable with other algorithms.

To derive our algorithm we extend the recent excessive gap technique of [1] to Bregman projections and establish rates of convergence (Section 2). This extension is important because the original gradient based algorithm for strongly

convex objectives by [1] does not admit graphical model factorizations, which are crucial for efficiency in structured prediction problems. We apply our resulting algorithm to the M³N objective in Section 3. A straightforward implementation requires $O(|\mathcal{Y}|)$ computational cost per iteration, which makes it prohibitively expensive. We show that by exploiting the graphical model structure of \mathcal{Y} the cost per iteration can be reduced to $O(\log |\mathcal{Y}|)$ (Section 4). Finally we contrast our algorithm with existing techniques in Section 5.

2 Excessive Gap Technique with Bregman Projection

The excessive gap technique proposed by [1] achieves accelerated rate of convergence only when the Euclidean projection is used. This prevents the algorithm from being applied to train M³N efficiently, and the aim of this section is to extend the approach to Bregman projection. We start by recapping the algorithm.

Definition 1. A function $f : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{\infty\}$ is called ρ strongly convex with respect to (wrt) a norm $\|\cdot\|$ if $f - \frac{\rho}{2}\|\cdot\|^2$ is convex. If f is differentiable and its gradient is Lipschitz continuous wrt $\|\cdot\|$ with constant L , we say f is L -l.c.g.

Let Q_1 and Q_2 be subsets of Euclidean spaces and A be a linear map from Q_1 to Q_2 . Suppose f and g are convex functions defined on Q_1 and Q_2 respectively. We are interested in the following optimization problem:

$$\min_{\mathbf{w} \in Q_1} J(\mathbf{w}) \text{ where } J(\mathbf{w}) := f(\mathbf{w}) + g^*(A\mathbf{w}) = f(\mathbf{w}) + \max_{\boldsymbol{\alpha} \in Q_2} \{\langle A\mathbf{w}, \boldsymbol{\alpha} \rangle - g(\boldsymbol{\alpha})\}. \quad (3)$$

We will make the following standard assumptions: a) Q_2 is compact; b) with respect to a certain norm on Q_1 , the function f defined on Q_1 is ρ -strongly convex ($\rho > 0$) but not necessarily l.c.g, and c) with respect to a certain norm on Q_2 (which can be different from that on Q_1), the function g defined on Q_2 is L_g -l.c.g and convex, but not necessarily strongly convex. If we identify $f(\mathbf{w})$ with the regularizer and $g^*(A\mathbf{w})$ with the loss function, then it is clear that (3) has the same form as (1) and (2). We will exploit this observation in Section 3.

If some mild constraint qualifications hold (e.g. Theorem 3.3.5 of [14]) one can write the dual $D(\boldsymbol{\alpha})$ of $J(\mathbf{w})$ using A^\top (the transpose of A) as

$$D(\boldsymbol{\alpha}) := -g(\boldsymbol{\alpha}) - f^*(-A^\top \boldsymbol{\alpha}) = -g(\boldsymbol{\alpha}) - \max_{\mathbf{w} \in Q_1} \{\langle -A\mathbf{w}, \boldsymbol{\alpha} \rangle - f(\mathbf{w})\}, \quad (4)$$

and assert the following (in M³N, both the max and min are attainable)

$$\min_{\mathbf{w} \in Q_1} J(\mathbf{w}) = \max_{\boldsymbol{\alpha} \in Q_2} D(\boldsymbol{\alpha}), \quad \text{and} \quad J(\mathbf{w}) \geq D(\boldsymbol{\alpha}) \quad \forall \mathbf{w} \in Q_1, \boldsymbol{\alpha} \in Q_2. \quad (5)$$

The key difficulty in solving (3) arises because g^* and hence J may potentially be non-smooth. Our aim is to uniformly approximate $J(\mathbf{w})$ with a smooth and strongly convex function. Towards this end let d be a σ strongly convex smooth function ($\sigma > 0$) with the following properties:

$$\min_{\boldsymbol{\alpha} \in Q_2} d(\boldsymbol{\alpha}) = 0, \quad d(\boldsymbol{\alpha}_0) = 0, \quad \text{and} \quad D := \max_{\boldsymbol{\alpha} \in Q_2} d(\boldsymbol{\alpha}).$$

In optimization parlance, d is called a prox-function. Let $\mu \in \mathbb{R}$ be an arbitrary positive constant, and we will use $(g + \mu d)^*$ to define a new objective function

$$J_\mu(\mathbf{w}) := f(\mathbf{w}) + (g + \mu d)^*(A\mathbf{w}) = f(\mathbf{w}) + \max_{\boldsymbol{\alpha} \in Q_2} \{ \langle A\mathbf{w}, \boldsymbol{\alpha} \rangle - g(\boldsymbol{\alpha}) - \mu d(\boldsymbol{\alpha}) \}. \quad (6)$$

The key idea of excessive gap minimization pioneered by [1] is to maintain two estimation sequences $\{\mathbf{w}_k\}$ and $\{\boldsymbol{\alpha}_k\}$, together with a diminishing sequence $\{\mu_k\}$ such that

$$J_{\mu_k}(\mathbf{w}_k) \leq D(\boldsymbol{\alpha}_k), \text{ and } \lim_{k \rightarrow \infty} \mu_k = 0. \quad (7)$$

Using (6), (7) and the definition of Fenchel dual, we can derive the key bound on the duality gap:

$$J(\mathbf{w}_k) - D(\boldsymbol{\alpha}_k) \leq J_{\mu_k}(\mathbf{w}_k) + \mu_k D - D(\boldsymbol{\alpha}_k) \leq \mu_k D. \quad (8)$$

In other words, to rapidly reduce the duality gap, we need to anneal down μ_k as quickly as possible, but still allow \mathbf{w}_k and $\boldsymbol{\alpha}_k$ to be updated efficiently.

[1] gave a solution based on Euclidean projections, where μ_k decays at $1/k^2$ rate and all updates can be computed in closed form. We now extend his ideas to updates based on Bregman projections [2], which will be the key to our application to structured prediction problems later. Since d is differentiable, we can define a Bregman divergence based on it.³

$$\Delta(\bar{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) := d(\bar{\boldsymbol{\alpha}}) - d(\boldsymbol{\alpha}) - \langle \nabla d(\boldsymbol{\alpha}), \bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha} \rangle. \quad (9)$$

Given a point $\boldsymbol{\alpha}$ and a direction \mathbf{g} , we can define the Bregman projection as:

$$V(\boldsymbol{\alpha}, \mathbf{g}) := \operatorname{argmin}_{\bar{\boldsymbol{\alpha}} \in Q_2} \{ \Delta(\bar{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) + \langle \mathbf{g}, \bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha} \rangle \} = \operatorname{argmin}_{\bar{\boldsymbol{\alpha}} \in Q_2} \{ d(\bar{\boldsymbol{\alpha}}) - \langle \nabla d(\boldsymbol{\alpha}) - \mathbf{g}, \bar{\boldsymbol{\alpha}} \rangle \}.$$

For notational convenience, we define the following two maps:

$$\mathbf{w}(\boldsymbol{\alpha}) := \operatorname{argmax}_{\mathbf{w} \in Q_1} \{ \langle -A\mathbf{w}, \boldsymbol{\alpha} \rangle - f(\mathbf{w}) \} = \nabla f^*(-A^\top \boldsymbol{\alpha}) \quad (10a)$$

$$\boldsymbol{\alpha}_\mu(\mathbf{w}) := \operatorname{argmax}_{\boldsymbol{\alpha} \in Q_2} \{ \langle A\mathbf{w}, \boldsymbol{\alpha} \rangle - g(\boldsymbol{\alpha}) - \mu d(\boldsymbol{\alpha}) \} = \nabla(g + \mu d)^*(A\mathbf{w}). \quad (10b)$$

Since both f and $(g + \mu d)$ are strongly convex, the above maps are unique and well defined. By easy calculation (e.g. Eq. (7.2) in [1]), $-D(\boldsymbol{\alpha})$ is *L-l.c.g* where

$$L = \frac{1}{\rho} \|A\|^2 + L_g, \quad \text{and } \|A\| := \max_{\|\mathbf{w}\|=\|\boldsymbol{\alpha}\|=1} \langle A\mathbf{w}, \boldsymbol{\alpha} \rangle. \quad (11)$$

With this notation in place we now describe our excessive gap minimization method in Algorithm 1. Unrolling the recursive update for μ_{k+1} yields $\mu_{k+1} = \frac{6}{(k+3)(k+2)} \frac{L}{\sigma}$. Plugging this into (8) and using (11) immediately yields a $O(1/\sqrt{\epsilon})$ rate of convergence of our algorithm.

² [1] did discuss updates based on Bregman projections, but just for the case where f is convex rather than strongly convex. Here, we show how to improve the convergence rate from $O(1/\epsilon)$ to $O(1/\sqrt{\epsilon})$ when f is strongly convex.

³ This paper applies ∇ only to differentiable functions; it never refers to subgradient.

Algorithm 1. Excessive gap minimization

Input: Function f which is strongly convex, convex function g which is *l.c.g.*
Output: Sequences $\{\mathbf{w}_k\}$, $\{\boldsymbol{\alpha}_k\}$, and $\{\mu_k\}$ that satisfy (7), with $\lim_{k \rightarrow \infty} \mu_k = 0$.

- 1 Initialize: $\boldsymbol{\alpha}_0 \leftarrow \operatorname{argmin}_{\mathbf{u} \in Q_2} d(\mathbf{u})$, $\mu_1 \leftarrow \frac{L}{\sigma}$, $\mathbf{w}_1 \leftarrow \mathbf{w}(\boldsymbol{\alpha}_0)$, $\boldsymbol{\alpha}_1 \leftarrow V\left(\boldsymbol{\alpha}_0, \frac{-1}{\mu_1} \nabla D(\boldsymbol{\alpha}_0)\right)$.
- 2 **for** $k = 1, 2, \dots$ **do**
- 3 $\tau_k \leftarrow \frac{2}{k+3}$.
- 4 $\hat{\boldsymbol{\alpha}} \leftarrow (1 - \tau_k)\boldsymbol{\alpha}_k + \tau_k \boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k)$.
- 5 $\mathbf{w}_{k+1} \leftarrow (1 - \tau_k)\mathbf{w}_k + \tau_k \mathbf{w}(\hat{\boldsymbol{\alpha}})$.
- 6 $\tilde{\boldsymbol{\alpha}} \leftarrow V\left(\boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k), \frac{-\tau_k}{(1-\tau_k)\mu_k} \nabla D(\hat{\boldsymbol{\alpha}})\right)$.
- 7 $\boldsymbol{\alpha}_{k+1} \leftarrow (1 - \tau_k)\boldsymbol{\alpha}_k + \tau_k \tilde{\boldsymbol{\alpha}}$.
- 8 $\mu_{k+1} \leftarrow (1 - \tau_k)\mu_k$.

Theorem 1 (Rate of convergence for duality gap). *The sequences $\{\mathbf{w}_k\}$ and $\{\boldsymbol{\alpha}_k\}$ in Algorithm 1 satisfy*

$$J(\mathbf{w}_k) - D(\boldsymbol{\alpha}_k) \leq \frac{6LD}{\sigma(k+1)(k+2)} = \frac{6D}{\sigma(k+1)(k+2)} \left(\frac{1}{\rho} \|A\|^2 + L_g \right). \quad (12)$$

All that remains is to show that

Theorem 2. *The updates in Algorithm 1 guarantee (7) is satisfied for all $k \geq 1$.*

Proof. Since d is σ -strongly convex, so

$$\Delta(\bar{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) = d(\bar{\boldsymbol{\alpha}}) - d(\boldsymbol{\alpha}) - \langle \nabla d(\boldsymbol{\alpha}), \bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha} \rangle \geq \frac{\sigma}{2} \|\bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha}\|^2. \quad (13)$$

It is not hard to show that the initial \mathbf{w}_1 and $\boldsymbol{\alpha}_1$ satisfy the excessive gap condition (7). We now focus on proving by induction that the updates in Algorithm 1 maintain (7). We begin with two useful observations. Using $\mu_{k+1} = \frac{6}{(k+3)(k+2)} \frac{L}{\sigma}$ and the definition of τ_k , one can bound

$$\mu_{k+1} = \frac{6}{(k+3)(k+2)} \frac{L}{\sigma} \geq \tau_k^2 \frac{L}{\sigma}. \quad (14)$$

Let $\beta := \boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k)$. The optimality conditions for (10b) imply

$$\langle \mu_k \nabla d(\beta) - A\mathbf{w}_k + \nabla g(\beta), \boldsymbol{\alpha} - \beta \rangle \geq 0. \quad (15)$$

By using the update equation for \mathbf{w}_{k+1} and the convexity of f , we have

$$\begin{aligned} J_{\mu_{k+1}}(\mathbf{w}_{k+1}) &= f(\mathbf{w}_{k+1}) + \max_{\boldsymbol{\alpha} \in Q_2} \{ \langle A\mathbf{w}_{k+1}, \boldsymbol{\alpha} \rangle - g(\boldsymbol{\alpha}) - \mu_{k+1}d(\boldsymbol{\alpha}) \} \\ &= f((1 - \tau_k)\mathbf{w}_k + \tau_k \mathbf{w}(\hat{\boldsymbol{\alpha}})) + \max_{\boldsymbol{\alpha} \in Q_2} \{ (1 - \tau_k) \langle A\mathbf{w}_k, \boldsymbol{\alpha} \rangle + \\ &\quad \tau_k \langle A\mathbf{w}(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha} \rangle - g(\boldsymbol{\alpha}) - (1 - \tau_k)\mu_k d(\boldsymbol{\alpha}) \} \\ &\leq \max_{\boldsymbol{\alpha} \in Q_2} \{ (1 - \tau_k)T_1 + \tau_k T_2 \}, \end{aligned}$$

where $T_1 = -\mu_k d(\boldsymbol{\alpha}) + \langle A\mathbf{w}_k, \boldsymbol{\alpha} \rangle - g(\boldsymbol{\alpha}) + f(\mathbf{w}_k)$ and $T_2 = -g(\boldsymbol{\alpha}) + \langle A\mathbf{w}(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha} \rangle + f(\mathbf{w}(\hat{\boldsymbol{\alpha}}))$. T_1 can be bounded as follows

$$\begin{aligned}
(\text{by defn. of } \Delta) \quad T_1 &= -\mu_k \{ \Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) + d(\boldsymbol{\beta}) + \langle \nabla d(\boldsymbol{\beta}), \boldsymbol{\alpha} - \boldsymbol{\beta} \rangle \} \\
&\quad + \langle A\mathbf{w}_k, \boldsymbol{\alpha} \rangle - g(\boldsymbol{\alpha}) + f(\mathbf{w}_k) \\
(\text{by (15)}) \leq & -\mu_k \Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) - \mu_k d(\boldsymbol{\beta}) + \langle -A\mathbf{w}_k + \nabla g(\boldsymbol{\beta}), \boldsymbol{\alpha} - \boldsymbol{\beta} \rangle \\
&\quad + \langle A\mathbf{w}_k, \boldsymbol{\alpha} \rangle - g(\boldsymbol{\alpha}) + f(\mathbf{w}_k) \\
= & -\mu_k \Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) - \mu_k d(\boldsymbol{\beta}) + \langle A\mathbf{w}_k, \boldsymbol{\beta} \rangle - g(\boldsymbol{\alpha}) + \\
&\quad \langle \nabla g(\boldsymbol{\beta}), \boldsymbol{\alpha} - \boldsymbol{\beta} \rangle + f(\mathbf{w}_k) \\
(\text{by convexity of } g) \leq & -\mu_k \Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) - \mu_k d(\boldsymbol{\beta}) + \langle A\mathbf{w}_k, \boldsymbol{\beta} \rangle - g(\boldsymbol{\beta}) + f(\mathbf{w}_k) \\
(\text{by defn. of } \boldsymbol{\beta}) = & -\mu_k \Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) + J_{\mu_k}(\mathbf{w}_k) \\
(\text{by induction assumption}) \leq & -\mu_k \Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) + D(\boldsymbol{\alpha}_k) \\
(\text{by concavity of } D) \leq & -\mu_k \Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) + D(\hat{\boldsymbol{\alpha}}) + \langle \nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha}_k - \hat{\boldsymbol{\alpha}} \rangle,
\end{aligned}$$

while T_2 can be bounded by using Lemma 7.2 of [4]:

$$T_2 = -g(\boldsymbol{\alpha}) + \langle A\mathbf{w}(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha} \rangle + f(\mathbf{w}(\hat{\boldsymbol{\alpha}})) \leq D(\hat{\boldsymbol{\alpha}}) + \langle \nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}} \rangle.$$

Putting the upper bounds on T_1 and T_2 together, we obtain the desired result.

$$\begin{aligned}
J_{\mu_{k+1}}(\mathbf{w}_{k+1}) &\leq \max_{\boldsymbol{\alpha} \in Q_2} \{ (1 - \tau_k) [-\mu_k \Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) + D(\hat{\boldsymbol{\alpha}}) + \langle \nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha}_k - \hat{\boldsymbol{\alpha}} \rangle] \\
&\quad + \tau_k [D(\hat{\boldsymbol{\alpha}}) + \langle \nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}} \rangle] \} \\
&= \max_{\boldsymbol{\alpha} \in Q_2} \{ -\mu_{k+1} \Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) + D(\hat{\boldsymbol{\alpha}}) + \\
&\quad \langle \nabla D(\hat{\boldsymbol{\alpha}}), (1 - \tau_k) \boldsymbol{\alpha}_k + \tau_k \boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}} \rangle \} \\
(\text{by defn. of } \hat{\boldsymbol{\alpha}}) &= \max_{\boldsymbol{\alpha} \in Q_2} \{ -\mu_{k+1} \Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) + D(\hat{\boldsymbol{\alpha}}) + \tau_k \langle \nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha} - \boldsymbol{\beta} \rangle \} \\
&= -\min_{\boldsymbol{\alpha} \in Q_2} \{ \mu_{k+1} \Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) - D(\hat{\boldsymbol{\alpha}}) - \tau_k \langle \nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha} - \boldsymbol{\beta} \rangle \} \\
(\text{by defn. of } \tilde{\boldsymbol{\alpha}}) &= -\mu_{k+1} \Delta(\tilde{\boldsymbol{\alpha}}, \boldsymbol{\beta}) + D(\hat{\boldsymbol{\alpha}}) + \tau_k \langle \nabla D(\hat{\boldsymbol{\alpha}}), \tilde{\boldsymbol{\alpha}} - \boldsymbol{\beta} \rangle \\
(\text{by (13)}) &\leq -\frac{1}{2} \mu_{k+1} \|\tilde{\boldsymbol{\alpha}} - \boldsymbol{\beta}\|^2 + D(\hat{\boldsymbol{\alpha}}) + \tau_k \langle \nabla D(\hat{\boldsymbol{\alpha}}), \tilde{\boldsymbol{\alpha}} - \boldsymbol{\beta} \rangle \\
(\text{by (14)}) &\leq -\frac{1}{2} \tau_k^2 L \|\tilde{\boldsymbol{\alpha}} - \boldsymbol{\beta}\|^2 + D(\hat{\boldsymbol{\alpha}}) + \tau_k \langle \nabla D(\hat{\boldsymbol{\alpha}}), \tilde{\boldsymbol{\alpha}} - \boldsymbol{\beta} \rangle \\
(\text{by defn. of } \boldsymbol{\alpha}_{k+1}) &= -\frac{1}{2} L \|\boldsymbol{\alpha}_{k+1} - \hat{\boldsymbol{\alpha}}\|^2 + D(\hat{\boldsymbol{\alpha}}) + \langle \nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha}_{k+1} - \hat{\boldsymbol{\alpha}} \rangle \\
(\text{by L-l.c.g of } D) &\leq D(\boldsymbol{\alpha}_{k+1}). \quad \blacksquare
\end{aligned}$$

When stated in terms of the dual gap (as opposed to the duality gap) our convergence results can be strengthened slightly. We omit the proof here.

$$\max_{\boldsymbol{\alpha} \in Q_2} D(\boldsymbol{\alpha}) - D(\boldsymbol{\alpha}_k) \leq \frac{6 L d(\boldsymbol{\alpha}^*)}{\sigma(k+1)(k+2)} = \frac{6 d(\boldsymbol{\alpha}^*)}{\sigma(k+1)(k+2)} \left(\frac{\|A\|^2}{\rho} + L_g \right), \quad (16)$$

where $\boldsymbol{\alpha}^* := \operatorname{argmax}_{\boldsymbol{\alpha} \in Q_2} D(\boldsymbol{\alpha})$. Note $d(\boldsymbol{\alpha}^*)$ is tighter than the D in (12).

3 Training Max-Margin Markov Networks

In the max-margin Markov network (M^3N) setting [4], we are given n labeled data points $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^n$, where \mathbf{x}^i are drawn from some space \mathcal{X} and \mathbf{y}^i belong to some

space \mathcal{Y} . We assume that there is a feature map ϕ which maps (\mathbf{x}, \mathbf{y}) to a feature vector in \mathbb{R}^p . Furthermore, for each \mathbf{x}^i , there is a label loss $\ell_{\mathbf{y}}^i := \ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i)$ which quantifies the loss of predicting label \mathbf{y} when the correct label is \mathbf{y}^i . Given this setup, the objective function minimized by M³Ns can be written as

$$J(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|_2^2 + \frac{1}{n} \sum_{i=1}^n \max_{\mathbf{y} \in \mathcal{Y}} \{\ell_{\mathbf{y}}^i - \langle \mathbf{w}, \psi_{\mathbf{y}}^i \rangle\}, \quad (17)$$

where $\|\mathbf{w}\|_2 = (\sum_i w_i^2)^{1/2}$ is the L_2 norm and we used the shorthand $\psi_{\mathbf{y}}^i := \phi(\mathbf{x}^i, \mathbf{y}^i) - \phi(\mathbf{x}^i, \mathbf{y})$. To write (17) in the form of (3), let $Q_1 = \mathbb{R}^p$, A be a $(n|\mathcal{Y}|)$ -by- p matrix whose (i, \mathbf{y}) -th row is $(-\psi_{\mathbf{y}}^i)^\top$,

$$f(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|_2^2, \quad \text{and} \quad g^*(\mathbf{u}) = \frac{1}{n} \sum_i \max_{\mathbf{y}} \{\ell_{\mathbf{y}}^i + u_{\mathbf{y}}^i\}.$$

Now, g can be verified to be:

$$g(\boldsymbol{\alpha}) = - \sum_i \sum_{\mathbf{y}} \ell_{\mathbf{y}}^i \alpha_{\mathbf{y}}^i \quad \text{if } \alpha_{\mathbf{y}}^i \geq 0, \text{ and } \sum_{\mathbf{y}} \alpha_{\mathbf{y}}^i = \frac{1}{n}, \forall i \quad (18)$$

and ∞ otherwise. The domain of g is $Q_2 = \mathcal{S}^n := \left\{ \boldsymbol{\alpha} \in [0, 1]^{n|\mathcal{Y}|} : \sum_{\mathbf{y}} \alpha_{\mathbf{y}}^i = \frac{1}{n}, \forall i \right\}$, which is convex and compact. Using the L_2 norm on Q_1 , f is clearly λ -strongly convex. Similarly, if we use the L_1 norm on Q_2 (i.e., $\|\boldsymbol{\alpha}\|_1 = \sum_i \sum_{\mathbf{y}} |\alpha_{\mathbf{y}}^i|$), then g is 0-l.c.g. By noting that $f^*(-A^\top \boldsymbol{\alpha}) = \frac{1}{2\lambda} \boldsymbol{\alpha}^\top A A^\top \boldsymbol{\alpha}$, one can write the dual form $D(\boldsymbol{\alpha}) : \mathcal{S}^n \mapsto \mathbb{R}$ of $J(\mathbf{w})$ as

$$D(\boldsymbol{\alpha}) = -g(\boldsymbol{\alpha}) - f^*(-A^\top \boldsymbol{\alpha}) = -\frac{1}{2\lambda} \boldsymbol{\alpha}^\top A A^\top \boldsymbol{\alpha} + \sum_i \sum_{\mathbf{y}} \ell_{\mathbf{y}}^i \alpha_{\mathbf{y}}^i, \quad \boldsymbol{\alpha} \in \mathcal{S}^n. \quad (19)$$

3.1 Rates of Convergence

A natural prox-function to use in our setting is the relative entropy with respect to the uniform distribution, which is defined as:

$$d(\boldsymbol{\alpha}) = \sum_{i=1}^n \sum_{\mathbf{y}} \alpha_{\mathbf{y}}^i \log \alpha_{\mathbf{y}}^i + \log n + \log |\mathcal{Y}|. \quad (20)$$

This results in a log-sum-exp form of $(g + \mu d)^*$ (derivation omitted):

$$(g + \mu d)^*(\mathbf{u}) = \frac{\mu}{n} \sum_{i=1}^n \log \sum_{\mathbf{y} \in \mathcal{Y}} \exp \left(\frac{u_{\mathbf{y}}^i + \ell_{\mathbf{y}}^i}{\mu} \right) - \mu \log |\mathcal{Y}|. \quad (21)$$

The relative entropy is 1-strongly convex in \mathcal{S}^n with respect to the L_1 norm [e.g., 15, Proposition 5.1]. Furthermore, $d(\boldsymbol{\alpha}) \leq D = \log |\mathcal{Y}|$ for $\boldsymbol{\alpha} \in \mathcal{S}^n$, and the norm of A can be computed via

$$\|A\| = \max_{\mathbf{w} \in \mathbb{R}^p, \mathbf{u} \in \mathbb{R}^{n|\mathcal{Y}|}} \left\{ \langle A\mathbf{w}, \mathbf{u} \rangle : \sum_{i=1}^p w_i^2 = 1, \sum_{i=1}^n \sum_{\mathbf{y}} |u_{\mathbf{y}}^i| = 1 \right\} = \max_{i, \mathbf{y}} \|\psi_{\mathbf{y}}^i\|_2,$$

where $\|\psi_{\mathbf{y}}^i\|_2$ is the Euclidean norm of $\psi_{\mathbf{y}}^i$. Since f is λ -strongly convex and $L_g = 0$, plugging this expression of $\|A\|$ into (12) and (16), we obtain the following rates of convergence for our algorithm:

$$J(\mathbf{w}_k) - D(\boldsymbol{\alpha}_k) \leq \frac{6 \log |\mathcal{Y}|}{(k+1)(k+2)} \frac{\max_{i,\mathbf{y}} \|\psi_{\mathbf{y}}^i\|_2^2}{\lambda}$$

and $\max_{\boldsymbol{\alpha} \in Q_2} D(\boldsymbol{\alpha}) - D(\boldsymbol{\alpha}_k) \leq \frac{6 \text{KL}(\boldsymbol{\alpha}^* \parallel \boldsymbol{\alpha}_0)}{(k+1)(k+2)} \frac{\max_{i,\mathbf{y}} \|\psi_{\mathbf{y}}^i\|_2^2}{\lambda},$

where $\text{KL}(\boldsymbol{\alpha}^* \parallel \boldsymbol{\alpha}_0)$ denotes the KL divergence between $\boldsymbol{\alpha}^*$ and the uniform distribution $\boldsymbol{\alpha}_0$. Recall that for distributions \mathbf{p} and \mathbf{q} the KL divergence is defined as $\text{KL}(\mathbf{p} \parallel \mathbf{q}) = \sum_i p_i \ln \frac{p_i}{q_i}$.

Therefore to reduce the duality gap and dual gap below ϵ , it suffices to take

$$2 + \max_{i,\mathbf{y}} \|\psi_{\mathbf{y}}^i\|_2 \sqrt{\frac{6 \log |\mathcal{Y}|}{\lambda \epsilon}} \quad \text{and} \quad \max_{i,\mathbf{y}} \|\psi_{\mathbf{y}}^i\|_2 \sqrt{\frac{6 \text{KL}(\boldsymbol{\alpha}^* \parallel \boldsymbol{\alpha}_0)}{\lambda \epsilon}} \quad (22)$$

steps respectively.

4 Efficient Computation by Clique Decomposition

In the structured large margin setting, the number of labels $|\mathcal{Y}|$ could potentially be exponentially large. For example, if a sequence has l nodes and each node has two states, then $|\mathcal{Y}| = 2^l$. A naive implementation of the excessive gap reduction algorithm described in the previous section requires maintaining and updating $O(|\mathcal{Y}|)$ coefficients at every iteration, which is prohibitively expensive. With a view to reducing the computational complexity, and also to take into account the inherent conditional independence properties of the output space, it is customary to assume that \mathcal{Y} is endowed with a graphical model structure; we refer the reader to [2] for an in-depth treatment of this issue. For our purposes it suffices to assume that $\ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i)$ and $\phi(\mathbf{x}^i, \mathbf{y})$ decompose according to the cliques⁴ of an undirected graphical model, and hence can be written (with some abuse of notation) as

$$\begin{aligned} \ell_{\mathbf{y}}^i &= \ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i) = \sum_{c \in \mathcal{C}} \ell(y_c, y_c^i; \mathbf{x}^i) = \sum_{c \in \mathcal{C}} \ell_{y_c}^i, \\ \phi(\mathbf{x}^i, \mathbf{y}) &= \bigoplus_{c \in \mathcal{C}} \phi(\mathbf{x}^i, y_c), \quad \text{and} \quad \psi_{\mathbf{y}}^i = \bigoplus_{c \in \mathcal{C}} \psi_{y_c}^i. \end{aligned} \quad (23)$$

Here \mathcal{C} denotes the set of all cliques of the graphical model and \oplus denotes vector concatenation. More explicitly, $\psi_{\mathbf{y}}^i$ is the vector on the graphical model obtained by accumulating the vector $\psi_{y_c}^i$ on all the cliques c of the graph.

Let $h_c(y_c)$ be an arbitrary real valued function on the value of \mathbf{y} restricted to clique c . Graphical models define a distribution $p(\mathbf{y})$ on $\mathbf{y} \in \mathcal{Y}$ whose density takes the following factorized form:

$$p(\mathbf{y}) \propto q(\mathbf{y}) = \prod_{c \in \mathcal{C}} \exp(h_c(y_c)). \quad (24)$$

⁴ Any fully connected subgraph of a graph is called a clique.

The key advantage of a graphical model is that the marginals on the cliques can be efficiently computed:

$$m_{y_c} := \sum_{\mathbf{z}: \mathbf{z}|_c = y_c} q(\mathbf{z}) = \sum_{\mathbf{z}: \mathbf{z}|_c = y_c} \prod_{c' \in \mathcal{C}} \exp(h_{c'}(z_{c'})).$$

where the summation is over all the configurations \mathbf{z} in \mathcal{Y} whose restriction on the clique c equals y_c . Although \mathcal{Y} can be exponentially large, efficient dynamic programming algorithms exist that exploit the factorized form (24), *e.g.* belief propagation [16]. The computational cost is $O(s^\omega)$ where s is the number of states of each node, and ω is the maximum size of the cliques. For example, a linear chain has $\omega = 2$. When ω is large, approximate algorithms also exist [17–19]. In the sequel we will assume that our graphical models are tractable, *i.e.*, ω is low.

4.1 Basics

At each iteration of Algorithm II, we need to compute four quantities: $\mathbf{w}(\boldsymbol{\alpha})$, $\nabla D(\boldsymbol{\alpha})$, $\alpha_\mu(\mathbf{w})$, and $V(\boldsymbol{\alpha}, \mathbf{g})$. Below we rewrite them by taking into account the factorization (23), and postpone to Section 4.2 the discussion on how to compute them efficiently. Since $\alpha_{\mathbf{y}}^i \geq 0$ and $\sum_{\mathbf{y}} \alpha_{\mathbf{y}}^i = \frac{1}{n}$, the $\{\alpha_{\mathbf{y}}^i : \mathbf{y} \in \mathcal{Y}\}$ form an unnormalized distribution, and we denote its (unnormalized) marginal distribution on clique c by

$$\alpha_{y_c}^i := \sum_{\mathbf{z}: \mathbf{z}|_c = y_c} \alpha_{\mathbf{z}}^i. \quad (25)$$

The feature expectations on the cliques with respect to the unnormalized distributions $\boldsymbol{\alpha}$ are important:

$$\mathbb{F}[\psi_{y_c}^i; \boldsymbol{\alpha}] := \sum_{y_c} \alpha_{y_c}^i \psi_{y_c}^i, \quad \text{and} \quad \mathbb{F}[\psi_c; \boldsymbol{\alpha}] := \sum_i \mathbb{F}[\psi_{y_c}^i; \boldsymbol{\alpha}]. \quad (26)$$

Clearly, if for all i the marginals of $\boldsymbol{\alpha}$ on the cliques (*i.e.*, $\{\alpha_{y_c}^i : i, c, y_c\}$) in (25)) are available, then these two expectations can be computed efficiently.

- **$\mathbf{w}(\boldsymbol{\alpha})$:** As a consequence of (23) we can write $\psi_{\mathbf{y}}^i = \bigoplus_{c \in \mathcal{C}} \psi_{y_c}^i$. Plugging this into (10a) and recalling that $\nabla f^*(-A^\top \boldsymbol{\alpha}) = -\frac{1}{\lambda} A^\top \boldsymbol{\alpha}$ yields the following expression for $\mathbf{w}(\boldsymbol{\alpha}) = -\frac{1}{\lambda} A^\top \boldsymbol{\alpha}$:
- $$\begin{aligned} \mathbf{w}(\boldsymbol{\alpha}) &= \frac{1}{\lambda} \sum_i \sum_{\mathbf{y}} \alpha_{\mathbf{y}}^i \psi_{\mathbf{y}}^i = \frac{1}{\lambda} \sum_i \sum_{\mathbf{y}} \alpha_{\mathbf{y}}^i \left(\bigoplus_{c \in \mathcal{C}} \psi_{y_c}^i \right) = \frac{1}{\lambda} \bigoplus_{c \in \mathcal{C}} \left(\sum_i \mathbb{F}[\psi_{y_c}^i; \boldsymbol{\alpha}] \right) \\ &= \frac{1}{\lambda} \bigoplus_{c \in \mathcal{C}} \mathbb{F}[\psi_c; \boldsymbol{\alpha}]. \end{aligned} \quad (27)$$

- **$\nabla D(\boldsymbol{\alpha})$:** Using (19) and the definition of $\mathbf{w}(\boldsymbol{\alpha})$, the (i, \mathbf{y}) -th element of $\nabla D(\boldsymbol{\alpha})$ can be written as

$$\begin{aligned} (\nabla D(\boldsymbol{\alpha}))_{\mathbf{y}}^i &= \ell_{\mathbf{y}}^i - \frac{1}{\lambda} (AA^\top \boldsymbol{\alpha})_{\mathbf{y}}^i = \ell_{\mathbf{y}}^i - \langle \psi_{\mathbf{y}}^i, \mathbf{w}(\boldsymbol{\alpha}) \rangle \\ &= \sum_c \left(\ell_{y_c}^i - \frac{1}{\lambda} \langle \psi_{y_c}^i, \mathbb{F}[\psi_c; \boldsymbol{\alpha}] \rangle \right). \end{aligned} \quad (28)$$

- $\alpha_\mu(\mathbf{w})$: Using (10b) and (21), the (i, \mathbf{y}) -th element of $\alpha_\mu(\mathbf{w})$ given by $(\nabla(g + \mu d)^*(A\mathbf{w}))_{\mathbf{y}}^i$ can be written as

$$\begin{aligned} (\alpha_\mu(\mathbf{w}))_{\mathbf{y}}^i &= \frac{1}{n} \frac{\exp(\mu^{-1}(\ell_{\mathbf{y}}^i - \langle \psi_{\mathbf{y}}^i, \mathbf{w} \rangle))}{\sum_{\mathbf{y}'} \exp(\mu^{-1}(\ell_{\mathbf{y}'}^i - \langle \psi_{\mathbf{y}'}^i, \mathbf{w} \rangle))} \\ &= \frac{1}{n} \frac{\prod_c \exp(\mu^{-1}(\ell_{y_c}^i - \langle \psi_{y_c}^i, \mathbf{w}_c \rangle))}{\sum_{\mathbf{y}'} \prod_c \exp(\mu^{-1}(\ell_{y'_c}^i - \langle \psi_{y'_c}^i, \mathbf{w}_c \rangle))}. \end{aligned} \quad (29)$$

- $V(\boldsymbol{\alpha}, \mathbf{g})$: Since the prox-function d is the relative entropy, the (i, \mathbf{y}) -th element of $V(\boldsymbol{\alpha}, \mathbf{g})$ is

$$(V(\boldsymbol{\alpha}, \mathbf{g}))_{\mathbf{y}}^i = \frac{1}{n} \frac{\alpha_{\mathbf{y}}^i \exp(-g_{\mathbf{y}}^i)}{\sum_{\mathbf{y}'} \alpha_{\mathbf{y}'}^i \exp(-g_{\mathbf{y}'}^i)}. \quad (30)$$

4.2 Efficient Computation

We now show how the algorithm can be made efficient by taking into account (23). Key to our efficient implementation are the following four observations from Algorithm 1 when applied to the structured large margin setting. In particular, we will exploit the fact that the marginals of $\boldsymbol{\alpha}_k$ can be updated iteratively.

- **The marginals of $\alpha_{\mu_k}(\mathbf{w}_k)$ and $\hat{\boldsymbol{\alpha}}$ can be computed efficiently.** From (29) it is easy to see that $\alpha_{\mu_k}(\mathbf{w}_k)$ can be written as a product of factors over cliques, that is, in the form of (24). Therefore, the marginals of $\alpha_{\mu_k}(\mathbf{w}_k)$ can be computed efficiently. As a result, if we keep track of the marginal distributions of $\boldsymbol{\alpha}_k$, then it is trivial to compute the marginals of $\hat{\boldsymbol{\alpha}} = (1 - \tau_k)\boldsymbol{\alpha}_k + \tau_k\alpha_{\mu_k}(\mathbf{w}_k)$.
- **The marginals of $\tilde{\boldsymbol{\alpha}}$ can be computed efficiently.** Define $\eta = \frac{-\tau_k}{(1-\tau_k)\mu_k}$. By plugging in (28) and (29) into (30) and observing that $\nabla D(\boldsymbol{\alpha})$ can be written as a sum of terms over cliques obtains:

$$\begin{aligned} \tilde{\alpha}_{\mathbf{y}}^i &= (V(\alpha_{\mu_k}(\mathbf{w}_k), \eta \nabla D(\hat{\boldsymbol{\alpha}})))_{\mathbf{y}}^i \propto (\alpha_{\mu_k}(\mathbf{w}_k))_{\mathbf{y}}^i \exp(-\eta (\nabla D(\hat{\boldsymbol{\alpha}}))_{\mathbf{y}}^i) \\ &= \prod_c \exp(\mu_k^{-1}(\ell_{y_c}^i - \langle \psi_{y_c}^i, (\mathbf{w}_k)_c \rangle) - \eta \ell_{y_c}^i + \eta \lambda^{-1} \langle \psi_{y_c}^i, \mathbb{F}[\psi_c; \hat{\boldsymbol{\alpha}}] \rangle). \end{aligned} \quad (31)$$

Clearly, $\tilde{\boldsymbol{\alpha}}$ factorizes and has the form of (24). Hence its marginals can be computed efficiently.

- **The marginals of $\boldsymbol{\alpha}_k$ can be updated efficiently.** Given the marginals of $\tilde{\boldsymbol{\alpha}}$, it is trivial to update the marginals of $\boldsymbol{\alpha}_{k+1}$ since $\boldsymbol{\alpha}_{k+1} = (1 - \tau_k)\boldsymbol{\alpha}_k + \tau_k\tilde{\boldsymbol{\alpha}}$. For convenience, define $\boldsymbol{\alpha}_c := \{\alpha_{y_c}^i : i, y_c\}$.
- **\mathbf{w}_k can be updated efficiently.** According to step 5 of Algorithm 1, by using (27) we have

$$(\mathbf{w}_{k+1})_c = (1 - \tau_k)(\mathbf{w}_k)_c + \tau_k(\mathbf{w}(\hat{\boldsymbol{\alpha}}))_c = (1 - \tau_k)(\mathbf{w}_k)_c + \tau_k \lambda^{-1} \mathbb{F}[\psi_c; \hat{\boldsymbol{\alpha}}].$$

Leveraging these observations, Algorithm 2 provides a complete listing of how to implement the excessive gap technique with Bregman projections for training M³N. It focuses on clarifying the ideas; a practical implementation can be sped up in many ways. The last issue to be addressed is the computation of the primal and dual objectives $J(\mathbf{w}_k)$ and $D(\boldsymbol{\alpha}_k)$, so as to monitor the duality gap. Indeed, this is viable without incurring higher order of computations and we leave the details to the reader.

Algorithm 2. Max-margin structured learning using clique factorization

Input: Loss functions $\{\ell_y^i\}$ and features $\{\psi_y^i\}$, a regularization parameter λ , a tolerance level $\epsilon > 0$.

Output: A pair \mathbf{w} and $\boldsymbol{\alpha}$ that satisfy $J(\mathbf{w}) - D(\boldsymbol{\alpha}) < \epsilon$.

- 1 Initialize: $k \leftarrow 1$, $\mu_1 \leftarrow \frac{1}{\lambda} \max_{i,y} \|\psi_y^i\|_2^2$, $\boldsymbol{\alpha}_0 \leftarrow \left(\frac{1}{n|\mathcal{Y}|}, \dots, \frac{1}{n|\mathcal{Y}|} \right)^\top \in \mathbb{R}^{n|\mathcal{Y}|}$.
- 2 Update $\mathbf{w}_1 \leftarrow \mathbf{w}(\boldsymbol{\alpha}_0) = \frac{1}{\lambda} \bigoplus_{c \in \mathcal{C}} \mathbb{F}[\psi_c; \boldsymbol{\alpha}_0]$, $\boldsymbol{\alpha}_1 \leftarrow V\left(\boldsymbol{\alpha}_0, -\frac{1}{\mu_1} \nabla D(\boldsymbol{\alpha}_0)\right)$ and compute its marginals.
- 3 **while** $J(\mathbf{w}_k) - D(\boldsymbol{\alpha}_k) \geq \epsilon$ **do** /* Terminate when duality gap falls below ϵ */
- 4 $\tau_k \leftarrow \frac{2}{k+3}$.
- 5 Compute the marginals of $\boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k)$ by exploiting (29).
- 6 **forall** cliques $c \in \mathcal{C}$ **do**
- 7 Compute the marginals $\hat{\boldsymbol{\alpha}}_c$ by convex combination:

$$\hat{\boldsymbol{\alpha}}_c \leftarrow (1 - \tau_k)(\boldsymbol{\alpha}_k)_c + \tau_k(\boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k))_c.$$
- 8 Update the weight on clique c :

$$(\mathbf{w}_{k+1})_c \leftarrow (1 - \tau_k)(\mathbf{w}_k)_c + \frac{\tau_k}{\lambda} \sum_i \mathbb{F}[\psi_{y_c}^i; \hat{\boldsymbol{\alpha}}_c].$$
- 9 Compute the marginals of $\tilde{\boldsymbol{\alpha}}$ by using (31) and the marginals $\{\hat{\boldsymbol{\alpha}}_c\}$.
- 10 **forall** cliques $c \in \mathcal{C}$ **do**
- 11 Update the marginals $(\boldsymbol{\alpha}_k)_c$ by convex combination:

$$(\boldsymbol{\alpha}_{k+1})_c \leftarrow (1 - \tau_k)(\boldsymbol{\alpha}_k)_c + \tau_k \tilde{\boldsymbol{\alpha}}_c.$$
- 12 Update $\mu_{k+1} \leftarrow (1 - \tau_k)\mu_k$, $k \leftarrow k + 1$.
- 13 **return** \mathbf{w}_k and $\boldsymbol{\alpha}_k$.

4.3 Kernelization

When nonlinear kernels are used, the feature vectors ϕ_y^i are not expressed explicitly and only their inner products can be evaluated via kernels on the cliques:

$$\langle \psi_y^i, \psi_{y'}^j \rangle := k((\mathbf{x}^i, \mathbf{y}), (\mathbf{x}^j, \mathbf{y}')) = \sum_c k_c((\mathbf{x}^i, y_c), (\mathbf{x}^j, y'_c)),$$

where $k_c((\mathbf{x}^i, y_c), (\mathbf{x}^j, y'_c)) := \langle \psi_{y_c}^i, \psi_{y'_c}^j \rangle$. Algorithm 2 is no longer applicable because no explicit expression of \mathbf{w} is available. However, by rewriting \mathbf{w}_k as the feature expectations with respect to some distribution $\beta_k \in \mathcal{S}^n$, then we only need to update \mathbf{w}_k implicitly via β_k , and the inner product between \mathbf{w}_k and any feature vector can also be efficiently calculated. We formalize and prove this claim by induction.

Theorem 3. For all $k \geq 0$, there exists $\beta_k \in \mathcal{S}^n$, such that $(\mathbf{w}_k)_c = \frac{1}{\lambda} \mathbb{F}[\psi_c; \beta_k]$, and β_k can be updated by $\beta_{k+1} = (1 - \tau_k)\beta_k + \tau_k \hat{\boldsymbol{\alpha}}_k$.

Proof. First, $\mathbf{w}_1 = \mathbf{w}(\boldsymbol{\alpha}_0) = \frac{1}{\lambda} \bigoplus_{c \in \mathcal{C}} \mathbb{F}[\psi_c; \boldsymbol{\alpha}_0]$, so $\beta_1 = \boldsymbol{\alpha}_0$. Suppose the claim holds for all $1, \dots, k$, then

$$(\mathbf{w}_{k+1})_c = (1 - \tau_k)(\mathbf{w}_k)_c + \frac{\tau_k}{\lambda} \mathbb{F}[\psi_c; (\hat{\boldsymbol{\alpha}}_k)_c] = (1 - \tau_k) \frac{1}{\lambda} \mathbb{F}[\psi_c; \beta_k] + \frac{\tau_k}{\lambda} \mathbb{F}[\psi_c; (\hat{\boldsymbol{\alpha}}_k)_c]$$

$$= \frac{1}{\lambda} \mathbb{F}[\psi_c; (1 - \tau_k)(\beta_k)_c + \tau_k(\hat{\boldsymbol{\alpha}}_k)_c].$$

Therefore, we can set $\beta_{k+1} = (1 - \tau_k)\beta_k + \tau_k \hat{\boldsymbol{\alpha}}_k \in \mathcal{S}^n$. ■

In general $\hat{\alpha}_k \neq \tilde{\alpha}_k$, hence $\beta_k \neq \alpha_k$. To compute $\langle \psi_{y_c}^i, (\mathbf{w}_k)_c \rangle$ required by (31), we have

$$\langle \psi_{y_c}^i, (\mathbf{w}_k)_c \rangle = \left\langle \psi_{y_c}^i, \frac{1}{\lambda} \sum_j \sum_{y'_c} \beta_{y'_c}^j \psi_{y'_c}^j \right\rangle = \frac{1}{\lambda} \sum_j \sum_{y'_c} \beta_{y'_c}^j k_c((\mathbf{x}^i, y_c), (\mathbf{x}^j, y'_c)).$$

And by using this trick, all the iterative updates in Algorithm 2 can be done efficiently. So is the evaluation of $\|\mathbf{w}_k\|^2$ and the primal objective. The dual objective (19) is also easy since

$$\sum_i \sum_{\mathbf{y}} \ell_{\mathbf{y}}^i (\alpha_k)_\mathbf{y}^i = \sum_i \sum_{\mathbf{y}} \sum_c \ell_{y_c}^i (\alpha_k)_\mathbf{y}^i = \sum_i \sum_c \sum_{y_c} \sum_{\mathbf{y}: \mathbf{y}|_c = y_c} \ell_{y_c}^i (\alpha_k)_\mathbf{y}^i = \sum_{i,c,y_c} \ell_{y_c}^i (\alpha_k)_\mathbf{y}^i,$$

and the marginals of α_k are available. Finally, the quadratic term in $D(\alpha_k)$ can be computed by

$$\begin{aligned} \|A^\top \alpha_k\|_2^2 &= \left\| \sum_{i,\mathbf{y}} \psi_{\mathbf{y}}^i (\alpha_k)_\mathbf{y}^i \right\|_2^2 = \sum_c \left\| \sum_{i,y_c} \psi_{y_c}^i (\alpha_k)_\mathbf{y}^i \right\|_2^2 \\ &= \sum_c \sum_{i,j,y_c,y'_c} (\alpha_k)_c^i (\alpha_k)_{y'_c}^j k_c((\mathbf{x}^i, y_c), (\mathbf{x}^j, y'_c)), \end{aligned}$$

where the inner term is the same as the unnormalized expectation that can be efficiently calculated. The last formula is only for nonlinear kernels.

4.4 Efficiency in Memory and Computation

For concreteness, let us consider a sequence as an example. Here the cliques are just edges between consecutive nodes. Suppose there are $l + 1$ nodes and each node has s states. The memory cost of Algorithm 2 is $O(nls^2)$, due to the storage of the marginals. The computational cost per iteration is dominated by calculating the marginals of $\hat{\alpha}$ and $\tilde{\alpha}$, which is $O(nls^2)$ by standard graphical model inference. The rest operations in Algorithm 2 cost $O(nls^2)$ for linear kernels. If nonlinear kernels are used, then the cost becomes $O(n^2ls^2)$.

5 Discussion

Structured output prediction is an important learning task in both theory and practice. The main contribution of our paper is twofold. First, we identified an efficient algorithm by [1] for solving the optimization problems in structured prediction. We proved the $O(1/\sqrt{\epsilon})$ rate of convergence for the Bregman projection based updates in excessive gap optimization, while [2] showed this rate only for projected gradient style updates. In M³N optimization, Bregman projection plays a key role in factorizing the computations, while technically such factorizations are not applicable to projected gradient. Second, we designed a nontrivial application of the excessive gap technique to M³N optimization, in which the computations are kept efficient by using the graphical model decomposition. Kernelized objectives can also be handled by our method, and we proved superior convergence and computational guarantees than existing algorithms.

When M³Ns are trained in a batch fashion, we can compare the convergence rate of dual gap between our algorithm and the exponentiated gradient method [ExpGrad, 7]. Assume α_0 , the initial value of α , is the uniform distribution and α^* is the optimal dual solution. Then by (22), we have

$$\text{Ours: } \max_{i,y} \|\psi_y^i\|_2 \sqrt{\frac{6\text{KL}(\alpha^*||\alpha_0)}{\lambda\epsilon}}, \quad \text{ExpGrad: } \max_{i,y} \|\psi_y^i\|_2^2 \frac{\text{KL}(\alpha^*||\alpha_0)}{\lambda\epsilon}.$$

It is clear that our iteration bound is almost the square root of ExpGrad, and has much better dependence on ϵ , λ , $\max_{i,y} \|\psi_y^i\|_2$, as well as the divergence from the initial guess to the optimal solution $\text{KL}(\alpha^*||\alpha_0)$.

In addition, the cost per iteration of our algorithm is almost the same as ExpGrad, and both are governed by the computation of the expected feature values on the cliques (which we call exp-oracle), or equivalently the marginal distributions. For graphical models, exact inference algorithms such as belief propagation can compute the marginals via dynamic programming [16]. Finally, although both algorithms require marginalization, they are calculated in very different ways. In ExpGrad, the dual variables α correspond to a factorized distribution, and in each iteration its potential functions on the cliques are updated using the exponentiated gradient rule. In contrast, our algorithm explicitly updates the marginal distributions of α_k on the cliques, and marginalization inference is needed only for $\hat{\alpha}$ and $\tilde{\alpha}$. Indeed, the joint distribution α does *not* factorize, which can be seen from step 7 of Algorithm 1: the convex combination of two factorized distributions is not necessarily factorized.

Marginalization is just one type of query that can be answered efficiently by graphical models, and another important query is the max a-posteriori inference (which we call max-oracle): given the current model w , find the argmax in (2). Max-oracle has been used by greedy algorithms such as cutting plane (BMRM and SVM-Struct) and sequential minimal optimization [SMO, 11, Chapter 6]. SMO picks the steepest descent coordinate in the dual and greedily optimizes the quadratic analytically, but its convergence rate is linear in $|\mathcal{Y}|$ which can be exponentially large for M³N (ref Table 1). The max-oracle again relies on graphical models for dynamical programming [19], and many existing combinatorial optimizers can also be used, such as in the applications of matching [20] and context free grammar parsing [21]. Furthermore, this oracle is particularly useful for solving the slack rescaling variant of M³N proposed by [9]:

$$J(w) = \frac{\lambda}{2} \|w\|_2^2 + \frac{1}{n} \sum_{i=1}^n \max_{y \in \mathcal{Y}} \{ \ell(y, y^i; x^i) (1 - \langle w, \phi(x^i, y^i) - \phi(x^i, y) \rangle) \}. \quad (32)$$

Here two factorized terms get multiplied, which causes additional complexity in finding the maximizer. [22, Section 1.4.1] solved this problem by a modified dynamic program. Nevertheless, it is not clear how ExpGrad or our method can be used to optimize this objective.

In the quest for faster optimization algorithms for M³Ns, the following three questions are important: how hard is it to optimize M³N intrinsically, how informative is the oracle which is the only way for the algorithm to access the objective function (*e.g.*, evaluate the function and its derivatives), and how well

does the algorithm make use of such information. The superiority of our algorithm suggests that the exp-oracle provides more information about the function than the max-oracle does, and a deeper explanation is that the max-oracle is local [13, Section 1.3], *i.e.* it depends only on the value of the function in the neighborhood of the querying point \mathbf{w}_k . In contrast, the exp-oracle is not local and uses the global structure of the function. Hence there is no surprise that the less informative max-oracle is easier to compute, which makes it applicable to a wider range of problems such as [32]. Moreover, the comparison between Exp-Grad and our algorithm shows that even if the exp oracle is used, the algorithm still needs to make good use of it in order to converge faster.

For future research, it is interesting to study the lower bound complexity for optimizing M^3N , including the dependence on ϵ , n , λ , \mathcal{Y} , and probably even on the graphical model topology. Empirical evaluation of our algorithm is also important, especially regarding the numerical stability of the additive update of marginal distributions α_k under fixed precision. Broader applications are possible in sequence labeling, word alignment, context free grammar parsing, etc.

References

- [1] Nesterov, Y.: Excessive gap technique in nonsmooth convex minimization. SIAM Journal on Optimization 16(1) (2005)
- [2] Bakir, G., Hofmann, T., Schölkopf, B., Smola, A., Taskar, B., Vishwanathan, S.V.N.: Predicting Structured Data. MIT Press, Cambridge (2007)
- [3] Lafferty, J.D., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data. In: Proceedings of International Conference on Machine Learning (2001)
- [4] Taskar, B., Guestrin, C., Koller, D.: Max-margin Markov networks. In: Advances in Neural Information Processing Systems, vol. 16 (2004)
- [5] Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: Proceedings of HLT-NAACL (2003)
- [6] Teo, C., Vishwanathan, S.V.N., Smola, A., Le, Q.: Bundle methods for regularized risk minimization. Journal of Machine Learning Research 11, 311–365 (2010)
- [7] Collins, M., Globerson, A., Koo, T., Carreras, X., Bartlett, P.: Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks. Journal of Machine Learning Research 9, 1775–1822 (2008)
- [8] Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
- [9] Tschantzidis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research 6, 1453–1484 (2005)
- [10] Taskar, B., Lacoste-Julien, S., Jordan, M.: Structured prediction, dual extragradient and bregman projections. Journal of Machine Learning Research 7, 1627–1653 (2006)
- [11] Taskar, B.: Learning Structured Prediction Models: A Large Margin Approach. PhD thesis, Stanford University (2004)
- [12] List, N., Simon, H.U.: Svm-optimization and steepest-descent line search. In: Proceedings of the Annual Conference on Computational Learning Theory (2009)
- [13] Nemirovski, A., Yudin, D.: Problem Complexity and Method Efficiency in Optimization. John Wiley and Sons, Chichester (1983)

- [14] Borwein, J.M., Lewis, A.S.: Convex Analysis and Nonlinear Optimization: Theory and Examples. Canadian Mathematical Society (2000)
- [15] Beck, A., Teboulle, B.: Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters* 31(3), 167–175 (2003)
- [16] Lauritzen, S.L.: Graphical Models. Oxford University Press, Oxford (1996)
- [17] Wainwright, M., Jordan, M.: Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning* 1(1-2), 1–305 (2008)
- [18] Andrieu, C., de Freitas, N., Doucet, A., Jordan, M.I.: An introduction to MCMC for machine learning. *Machine Learning* 50, 5–43 (2003)
- [19] Kschischang, F., Frey, B.J., Loeliger, H.-A.: Factor graphs and the sum-product algorithm. *IEEE Trans. on Information Theory* 47(2), 498–519 (2001)
- [20] Taskar, B., Lacoste-Julien, S., Klein, D.: A discriminative matching approach to word alignment. In: *Empirical Methods in Natural Language Processing* (2005)
- [21] Taskar, B., Klein, D., Collins, M., Koller, D., Manning, C.: Max-margin parsing. In: *Empirical Methods in Natural Language Processing* (2004)
- [22] Altun, Y., Hofmann, T., Tsochandiridis, I.: Support vector machine learning for interdependent and structured output spaces. In: Bakir, G., Hofmann, T., Schölkopf, B., Smola, A., Taskar, B., Vishwanathan, S.V.N. (eds.) *Predicting Structured Data*, ch.5, pp. 85–103. MIT Press, Cambridge (2007)

Domain Adaptation in Regression

Corinna Cortes¹ and Mehryar Mohri^{1,2}

¹ Google Research,

76 Ninth Avenue, New York, NY 10011

² Courant Institute of Mathematical Sciences,

251 Mercer Street, New York, NY 10012

Abstract. This paper presents a series of new results for domain adaptation in the regression setting. We prove that the discrepancy is a distance for the squared loss when the hypothesis set is the reproducing kernel Hilbert space induced by a universal kernel such as the Gaussian kernel. We give new pointwise loss guarantees based on the discrepancy of the empirical source and target distributions for the general class of kernel-based regularization algorithms. These bounds have a simpler form than previous results and hold for a broader class of convex loss functions not necessarily differentiable, including L_q losses and the hinge loss. We extend the discrepancy minimization adaptation algorithm to the more significant case where kernels are used and show that the problem can be cast as an SDP similar to the one in the feature space. We also show that techniques from smooth optimization can be used to derive an efficient algorithm for solving such SDPs even for very high-dimensional feature spaces. We have implemented this algorithm and report the results of experiments demonstrating its benefits for adaptation and show that, unlike previous algorithms, it can scale to large data sets of tens of thousands or more points.

1 Introduction

A standard assumption in learning theory and applications is that training and test points are drawn according to the same distribution. But, a more challenging problem of *domain adaptation* arises in a variety of applications, including natural language processing, speech processing, or computer vision [7, 3, 9, 10, 17, 18, 12]. This problem occurs when little or no labeled data is available from the *target domain*, but labeled data from a *source domain* somewhat similar to the target, as well as large amounts of unlabeled data from the target domain, are accessible. The domain adaptation problem then consists of using the source labeled and target unlabeled data to learn a hypothesis performing well on the target domain.

The theoretical analysis of this problem has been the topic of some recent publications. An analysis of adaptation was initiated by Ben-David et al. [1]. Several issues of that paper were later corrected by Blitzer et al. [2]. These authors gave VC-dimension bounds for binary classification based on a d_A distance between distributions that can be estimated from finite samples, and a term λ_H depending on the distributions and the hypothesis set H , which cannot be estimated from data. In [11], we presented alternative learning bounds which hold in particular in the classification setting and depend on the optimal classifiers in the hypothesis set for the source and target distributions.

Our bounds are in general not comparable to those of [1, 4] but we showed that under some plausible assumptions they are superior to those of [1, 4] and that in many cases the bounds of [1, 4] have a factor of 3 of the error that can make them vacuous. The assumptions made in the analysis of adaptation were more recently discussed by Ben-David et al. [2] who also presented several negative results for this problem. These negative results hold only for the 0-1 loss used in classification.

This paper deals with the problem of adaptation in regression, for which many of the observations made in the case of the 0-1 loss do not hold. In [11], we introduced a distance between distributions specifically tailored to domain adaptation, the *discrepancy distance*, which generalizes the d_A distance to arbitrary loss functions, and presented several theoretical guarantees based on that discrepancy, including data-dependent Rademacher complexity generalization bounds. In this paper we present a series of novel results for domain adaptation in regression extending those of [11] and making them more significant and practically applicable.

In Section 2 we describe more formally the learning scenario of domain adaptation in regression and briefly review the definition and key properties of the discrepancy. We then present several theoretical results in Section 3. For the squared loss, we prove that the discrepancy is a distance when the hypothesis set is the reproducing kernel Hilbert space of a universal kernel, such as a Gaussian kernel. This implies that minimizing the discrepancy to zero guarantees matching the target distribution, a result that does not hold in the case of the 0-1 loss. We further give pointwise loss guarantees depending on the discrepancy of the empirical source and target distributions for the class of kernel-based regularization algorithms, including kernel ridge regression, support vector machines (SVMs), or support vector regression (SVR). These bounds have a simpler form than a previous result we presented in the specific case of the squared loss in [11] and hold for a broader class of convex loss functions not necessarily differentiable, which includes all L_q losses ($q \geq 1$), but also the hinge loss used in classification.

When the magnitude of the difference between the source and target labeling functions is small on the training set, these bounds provide a strong guarantee based on the empirical discrepancy and suggest an empirical discrepancy minimization algorithm [11]. In Section 4 we extend the discrepancy minimization adaptation algorithm with the squared loss to the more significant case where kernels are used. We show that the problem can be cast as a semi-definite programming (SDP) problem similar to the one given in [11] in the feature space, but formulated only in terms of the kernel matrix.

Such SDP optimization problems can only be solved practically for modest sample sizes of a few hundred points using existing solvers, even with the most efficient publicly available one. In Section 5 we prove, however, that an algorithm with significantly better time and space complexities can be derived to solve these SDPs using techniques from smooth optimization [14]. We describe the algorithm in detail. We prove a bound on the number of iterations and analyze the computational cost of each iteration.

We have implemented that algorithm and carried out extensive experiments showing that it can indeed scale to large data sets of tens of thousands or more points. Our kernelized version of the SDP further enables us to run the algorithm for very high-dimensional and even infinite-dimensional feature spaces. Section 6 reports our empirical results demonstrating the effectiveness of this algorithm for domain adaptation.

2 Preliminaries

This section describes the learning scenario of domain adaptation and reviews the key definitions and properties of the *discrepancy distance* between distributions.

2.1 Learning Scenario

Let X denote the input space and Y the output space, a measurable subset of \mathbb{R} , as in standard regression problems. In the adaptation problem we are considering, there are different *domains*, defined by a distribution over X and a target labeling function mapping from X to Y . We denote by Q the distribution over X for the *source domain* and by $f_Q: X \rightarrow Y$ the corresponding labeling function. Similarly, we denote by P the distribution over X for the *target domain* and by f_P the target labeling function. When the two domains share the same labeling function, we simply denote it by f .

In the *domain adaptation problem* in regression, the learning algorithm receives a labeled sample of m points $\mathcal{S} = ((x_1, y_1), \dots, (x_m, y_m)) \in (X \times Y)^m$ from the source domain, that is x_1, \dots, x_m are drawn i.i.d. according to Q and $y_i = f_Q(x_i)$ for $i \in [1, m]$. We denote by \hat{Q} the empirical distribution corresponding to x_1, \dots, x_m . Unlike the standard supervised learning setting, the test points are drawn from the target domain, which is based on a different input distribution P and possibly different labeling function f_P . The learner is additionally provided with an unlabeled sample \mathcal{T} of size n drawn i.i.d. according to the target distribution P . We denote by \hat{P} the empirical distribution corresponding to \mathcal{T} .

We consider a loss function $L: Y \times Y \rightarrow \mathbb{R}_+$ that is symmetric and convex with respect to each of its argument. In particular L may be the squared loss commonly used in regression. For any two functions $h, h': X \rightarrow Y$ and any distribution D over X , we denote by $\mathcal{L}_D(h, h')$ the expected loss of $h(x)$ and $h'(x)$:

$$\mathcal{L}_D(h, h') = \mathbb{E}_{x \sim D}[L(h(x), h'(x))]. \quad (1)$$

The domain adaptation problem consists of selecting a hypothesis h out of a hypothesis set H with a small expected loss according to the target distribution P , $\mathcal{L}_P(h, f_P)$.

2.2 Discrepancy Distance

A key question for adaptation is a measure of the difference between the distributions Q and P . As pointed out in [11], a general-purpose measure such as the L_1 distance is not helpful in this context since the L_1 distance can be large even in some rather favorable situations for adaptation. Furthermore, this distance cannot be accurately estimated from finite samples and ignores the loss function. Instead, the discrepancy provides a measure of the dissimilarity of two distributions that is specifically tailored to adaptation and is defined based on the loss function and the hypothesis set used.

Observe that for a fixed hypothesis $h \in H$, the quantity of interest in adaptation is the difference of expected losses $|\mathcal{L}_P(f_P, h) - \mathcal{L}_Q(f_P, h)|$. A natural distance between distributions in this context is thus one based on the supremum of this quantity over all $h \in H$. The target hypothesis f_P is unknown and could match any hypothesis h' . This leads to the following definition [11].

Definition 1. Given a hypothesis set H and loss function L , the discrepancy distance disc between two distributions P and Q over X is defined by:

$$\text{disc}(P, Q) = \max_{h, h' \in H} |\mathcal{L}_P(h', h) - \mathcal{L}_Q(h', h)|. \quad (2)$$

The discrepancy is by definition symmetric and verifies the triangle inequality for any loss function L . But, in general, it does not define a *distance* since we may have $\text{disc}(P, Q) = 0$ for $P \neq Q$. We shall prove, however, that for a large family of kernel-based hypothesis set, it does verify all the axioms of a distance.

3 Theoretical Analysis

In what follows, we consider the case where the hypothesis set H is a subset of the reproducing kernel Hilbert space (RKHS) \mathbb{H} associated to a positive definite symmetric (PDS) kernel $K: H = \{h \in \mathbb{H}: \|h\|_K \leq \Lambda\}$, where $\|\cdot\|_K$ denotes the norm defined by the inner product on \mathbb{H} and $\Lambda \geq 0$. We shall assume that there exists $R > 0$ such that $K(x, x) \leq R^2$ for all $x \in X$. By the reproducing property, for any $h \in H$ and $x \in X$, $h(x) = \langle h, K(x, \cdot) \rangle_K$, thus this implies that $|h(x)| \leq \|h\|_K \sqrt{K(x, x)} \leq \Lambda R$.

3.1 Discrepancy with Universal Kernels

We first prove that for a *universal kernel* K , such as a Gaussian kernel [20], the discrepancy defines a distance. Let $C(X)$ denote the set of all continuous functions mapping X to \mathbb{R} . We shall assume that X is a compact set, thus the functions in $C(X)$ are also bounded. A PDS kernel K over $X \times X$ is said to be universal if it is continuous and if the RKHS \mathbb{H} it induces is dense in $C(X)$ for the norm infinity $\|\cdot\|_\infty$.

Theorem 1. Let L be the squared loss and let K be a universal kernel. Then, for any two distributions P and Q , if $\text{disc}(P, Q) = 0$, then $P = Q$.

Proof. Consider the function $\Psi: C(X) \rightarrow \mathbb{R}$ defined for any $h \in C(X)$ by $\Psi(h) = \mathbb{E}_{x \sim P}[h^2] - \mathbb{E}_{x \sim Q}[h^2]$. Ψ is continuous for the norm infinity over $C(X)$ since $h \mapsto \mathbb{E}_{x \sim P}[h^2]$ is continuous. Indeed, for any $h, h' \in H$,

$$|\mathbb{E}_P[h'^2] - \mathbb{E}_P[h^2]| = |\mathbb{E}_P[(h' + h)(h' - h)]| \leq (\|h\|_\infty + \|h'\|_\infty) \|h' - h\|_\infty,$$

and similarly with $h \mapsto \mathbb{E}_{x \sim Q}[h^2]$. If $\text{disc}(P, Q) = 0$, then, by definition,

$$\forall h, h' \in H, \quad \left| \mathbb{E}_P[(h'(x) - h(x))^2] - \mathbb{E}_Q[(h'(x) - h(x))^2] \right| = 0.$$

Thus, $\mathbb{E}_P[h''^2] - \mathbb{E}_Q[h''^2] = 0$ for any $h'' = h' - h \in \mathbb{H}$ with $\|h''\|_K \leq 2\Lambda R$, therefore for any $h'' \in \mathbb{H}$ with $\|h''\|_K \leq 2\Lambda R$, hence for any $h'' \in \mathbb{H}$ regardless of the norm. Thus, $\Psi = 0$ over \mathbb{H} . Since K is universal, \mathbb{H} is dense in $C(X)$ for the norm $\|\cdot\|_\infty$ and by continuity of Ψ for $\|\cdot\|_\infty$, for all $h \in C(X)$, $\mathbb{E}_P[h^2] - \mathbb{E}_Q[h^2] = 0$. Let f be any non-negative function in $C(X)$, then \sqrt{f} is well defined and is in $C(X)$, thus,

$$\mathbb{E}_P[(\sqrt{f})^2] - \mathbb{E}_Q[(\sqrt{f})^2] = \mathbb{E}_P[f] - \mathbb{E}_Q[f] = 0.$$

It is known that if $E_P[f] - E_Q[f] = 0$ for all $f \in C(X)$ with $f \geq 0$, then $P = Q$ (see [8][proof of lemma 9.3.2]). This concludes the proof. \square

Thus, the theorem shows that if we could find a source distribution Q that would reduce to zero the discrepancy in the case of the familiar Gaussian kernels, then that distribution would in fact match the target distribution P .

3.2 Guarantees for Kernel-Based Regularization Algorithms

We now present pointwise loss guarantees in domain adaptation for a broad class of kernel-based regularization algorithms, which also demonstrate the key role played by the discrepancy in adaptation and suggest the benefits of minimizing that quantity. These algorithms are defined by the minimization of the following objective function:

$$F_{\hat{Q}}(h) = \hat{R}_{\hat{Q}}(h) + \lambda \|h\|_K^2, \quad (3)$$

where $\lambda \geq 0$ is a trade-off parameter and $R_{\hat{Q}}(h) = \frac{1}{m} \sum_{i=1}^m L(h(x_i), y_i)$ the empirical error of hypothesis $h \in \mathbb{H}$. This family of algorithms includes support vector machines (SVM) [6], support vector regression (SVR) [21], kernel ridge regression (KRR) [19], and many other algorithms. We shall assume that the loss function L is μ -admissible for some $\mu > 0$: that is, it is symmetric and convex with respect to both of its arguments and for all $x \in X$ and $y \in Y$ and $h, h' \in H$, it verifies the following Lipschitz condition:

$$|L(h'(x), y) - L(h(x), y)| \leq \mu |h'(x) - h(x)|.$$

μ -admissible losses include the hinge loss and all L_q losses with $q \geq 1$, in particular the squared loss, when the hypothesis set and the set of output labels are bounded.

The labeling functions f_P and f_Q may not coincide on the training set $\text{supp}(\hat{Q})$. But, for adaptation to be possible, the difference between the labels received for the training points and their target values should be assumed to be small, even if the input space distributions P and Q are very close.

When the magnitude of the difference between the source and target labeling functions is small on the training set, that is $\eta = \max\{L(f_Q(x), f_P(x)) : x \in \text{supp}(\hat{Q})\} \ll 1$, the following theorem gives a strong guarantee on the pointwise difference of the loss between the hypothesis h returned by the algorithm when training on the source domain and the hypothesis h' returned when training on a sample drawn from the target distribution in terms of the empirical discrepancy $\text{disc}(\hat{P}, \hat{Q})$. The theorem holds for all μ -admissible losses and has a simpler form than a previous result we presented in [11].

Theorem 2. *Let L be a μ -admissible loss. Assume that $f_p \in H$ and let η denote $\max\{L(f_Q(x), f_P(x)) : x \in \text{supp}(\hat{Q})\}$. Let h' be the hypothesis returned by the kernel-based regularization algorithm (3) when minimizing $F_{\hat{P}}$ and h the one returned when minimizing $F_{\hat{Q}}$. Then, for all $x \in X$ and $y \in Y$,*

$$|L(h'(x), y) - L(h(x), y)| \leq \mu R \sqrt{\frac{\text{disc}(\hat{P}, \hat{Q}) + \mu\eta}{\lambda}}. \quad (4)$$

Proof. The proof makes use of a generalized Bregman divergence, which we first introduce. For a convex function $F: \mathbb{H} \rightarrow \mathbb{R}$, we denote by $\partial F(h)$ the subgradient of F at h : $\partial F(h) = \{g \in \mathbb{H}: \forall h' \in \mathbb{H}, F(h') - F(h) \geq \langle h' - h, g \rangle\}$. $\partial F(h)$ coincides with $\nabla F(h)$ when F is differentiable at h . Note that at a point h where F is minimal, 0 is an element of $\partial F(h)$. Furthermore, the subgradient is additive, that is, for two convex functions F_1 and F_2 , $\partial(F_1 + F_2)(h) = \{g_1 + g_2: g_1 \in \partial F_1(h), g_2 \in \partial F_2(h)\}$. For any $h \in \mathbb{H}$, fix $\delta F(h)$ to be an (arbitrary) element of $\partial F(h)$. For any such choice of δF , we can define the *generalized Bregman divergence* associated to F by:

$$\forall h', h \in \mathbb{H}, B_F(h' \| h) = F(h') - F(h) - \langle h' - h, \delta F(h) \rangle. \quad (5)$$

Note that by definition of the subgradient, $B_F(h' \| h) \geq 0$ for all $h', h \in \mathbb{H}$. Let N denote the convex function $h \rightarrow \|h\|_K^2$. Since N is differentiable, $\delta N(h) = \nabla N(h)$ for all $h \in \mathbb{H}$, and δN and thus B_N are uniquely defined. To make the definition of the Bregman divergences for $F_{\hat{Q}}$ and $\hat{R}_{\hat{Q}}$ compatible so that $B_{F_{\hat{Q}}} = B_{\hat{R}_{\hat{Q}}} + \lambda B_N$, we define $\delta \hat{R}_{\hat{Q}}$ from $\delta F_{\hat{Q}}$ by: $\delta \hat{R}_{\hat{Q}}(h) = \delta F_{\hat{Q}}(h) - \lambda \nabla N(h)$ for all $h \in \mathbb{H}$. Furthermore, we choose $\delta F_{\hat{Q}}(h)$ to be 0 for any point h where $F_{\hat{Q}}$ is minimal and let $\delta F_{\hat{Q}}(h)$ be an arbitrary element of $\partial F_{\hat{Q}}(h)$ for all other h s. We proceed in a similar way to define the Bregman divergences for $F_{\hat{P}}$ and $\hat{R}_{\hat{P}}$ so that $B_{F_{\hat{P}}} = B_{\hat{R}_{\hat{P}}} + \lambda B_N$.

Since the generalized Bregman divergence is non-negative and since $B_{F_{\hat{Q}}} = B_{\hat{R}_{\hat{Q}}} + \lambda B_N$ and $B_{F_{\hat{P}}} = B_{\hat{R}_{\hat{P}}} + \lambda B_N$, we can write

$$B_{F_{\hat{Q}}}(h' \| h) + B_{F_{\hat{P}}}(h \| h') \geq \lambda(B_N(h' \| h) + B_N(h \| h')).$$

Observe that $B_N(h' \| h) + B_N(h \| h') = -\langle h' - h, 2h \rangle - \langle h - h', 2h' \rangle = 2\|h' - h\|_K^2$. Thus, $B_{F_{\hat{Q}}}(h' \| h) + B_{F_{\hat{P}}}(h \| h') \geq 2\lambda\|h' - h\|_K^2$. By definition of h' and h as minimizers and our choice of the subgradients, $\delta F_{\hat{P}}(h') = 0$ and $\delta F_{\hat{Q}}(h) = 0$, thus, this inequality can be rewritten as follows:

$$2\lambda\|h' - h\|_K^2 \leq \hat{R}_{\hat{Q}}(h') - \hat{R}_{\hat{Q}}(h) + \hat{R}_{\hat{P}}(h) - \hat{R}_{\hat{P}}(h').$$

Now, rewriting this inequality in terms of the expected losses gives:

$$\begin{aligned} 2\lambda\|h' - h\|_K^2 &\leq (\mathcal{L}_{\hat{P}}(h, f_P) - \mathcal{L}_{\hat{Q}}(h, f_Q)) - (\mathcal{L}_{\hat{P}}(h', f_P) - \mathcal{L}_{\hat{Q}}(h', f_Q)) \\ &= (\mathcal{L}_{\hat{P}}(h, f_P) - \mathcal{L}_{\hat{Q}}(h, f_P)) - (\mathcal{L}_{\hat{P}}(h', f_P) - \mathcal{L}_{\hat{Q}}(h', f_P)) \\ &\quad + (\mathcal{L}_{\hat{Q}}(h, f_P) - \mathcal{L}_{\hat{Q}}(h, f_Q)) - (\mathcal{L}_{\hat{Q}}(h', f_P) - \mathcal{L}_{\hat{Q}}(h', f_Q)). \end{aligned}$$

Since f_P is in H , by definition of the discrepancy, the first two terms can both be bounded by the empirical discrepancy:

$$\left|(\mathcal{L}_{\hat{P}}(h, f_P) - \mathcal{L}_{\hat{Q}}(h, f_P))\right| \leq \text{disc}(\hat{P}, \hat{Q}) \text{ and } \left|(\mathcal{L}_{\hat{P}}(h', f_P) - \mathcal{L}_{\hat{Q}}(h', f_P))\right| \leq \text{disc}(\hat{P}, \hat{Q}).$$

The last two terms can be bounded using the μ -admissibility of L since for any $h'' \in H$,

$$\left|(\mathcal{L}_{\hat{Q}}(h'', f_P) - \mathcal{L}_{\hat{Q}}(h'', f_Q))\right| \leq \mu \mathbb{E}_{x \sim \hat{Q}} [|f_P(x) - f_Q(x)|] \leq \mu\eta.$$

Thus,

$$2\lambda\|h' - h\|_K^2 \leq 2\text{disc}(\widehat{P}, \widehat{Q}) + 2\mu\eta. \quad (6)$$

By the reproducing property, for any $x \in X$, $h' - h(x) = \langle h' - h, K(x, \cdot) \rangle$, thus, for any $x \in X$ and $y \in Y$, $|L(h'(x), y) - L(h(x), y)| \leq \mu|h' - h(x)| \leq \mu R\|h' - h\|_K$. Upper bounding the right-hand side using (6) directly yields the statement (4). \square

A similar theorem can be proven when only $f_Q \in H$ is assumed. These theorems can be extended to the case where neither the target function f_P nor f_Q is in H by replacing in Theorem 2 η with $\eta'' = \max\{L(h_P^*(x), f_Q(x)): x \in \text{supp}(\widehat{Q})\} + \max\{L(h_P^*(x), f_P(x)): x \in \text{supp}(\widehat{P})\}$, where $h_P^* \in \operatorname{argmin}_{h \in H} \mathcal{L}_P(h, f_P)$. They show the key role played by the empirical discrepancy $\text{disc}(\widehat{P}, \widehat{Q})$ in this context when $\eta'' \ll 1$. Note that $\eta = 0$ when $f_P = f_Q = f$ as in the sample bias correction setting or other scenarios where the so-called covariate-shift assumption hold. Under the assumptions $\eta \ll 1$ or $\eta'' \ll 1$, these theorems suggest seeking an empirical distribution q^* , among the family \mathcal{Q} of all distributions with a support included in that of \widehat{Q} , that minimizes that discrepancy [11]:

$$q^* = \operatorname{argmin}_{q \in \mathcal{Q}} \text{disc}(\widehat{P}, q). \quad (7)$$

Using q^* instead of \widehat{Q} amounts to reweighting the loss on each training point. This forms the basis of our adaptation algorithm which consists of: (a) first computing q^* ; (b) then modifying (3) using q^* :

$$F_{q^*}(h) = \frac{1}{m} \sum_{i=1}^m q^*(x_i) L(h(x_i), y_i) + \lambda\|h\|_K^2, \quad (8)$$

and finding a minimizing h . The minimization of F_{q^*} is no more difficult than that of $F_{\widehat{Q}}$ and is standard. Thus, in the following section, we focus on the first stage of our algorithm and study in detail the optimization problem (7).

4 Optimization Problems

Let X be a subset of \mathbb{R}^N , $N > 1$. We denote by S_Q the support of \widehat{Q} , by S_P the support of \widehat{P} , and by S their union $\text{supp}(\widehat{Q}) \cup \text{supp}(\widehat{P})$, with $|S_Q| = m \leq m$ and $|S_P| = n \leq n$. The unique elements of S_P are denoted by $\mathbf{x}_1, \dots, \mathbf{x}_m$ and those of S_Q by $\mathbf{x}_{m+1}, \dots, \mathbf{x}_q$, with $q = m + n$. For a vector $\mathbf{z} \in \mathbb{R}^m$, we denote by z_i its i th coordinate. We also denote by Δ_m the simplex in \mathbb{R}^m : $\Delta_m = \{\mathbf{z} \in \mathbb{R}^m: z_i \geq 0 \wedge \sum_{i=1}^m z_i = 1\}$.

4.1 Discrepancy Minimization in Feature Space

We showed in [11] that the problem of minimizing the empirical discrepancy for the squared loss and the hypothesis space $H = \{\mathbf{x} \mapsto \mathbf{w}^\top \mathbf{x}: \|\mathbf{w}\| \leq \Lambda\}$ of bounded linear functions can be cast as the following convex optimization problem:

$$\min_{\mathbf{z} \in \Delta_m} \|\mathbf{M}(\mathbf{z})\|_2, \quad (9)$$

where $\mathbf{M}(\mathbf{z}) \in \mathbb{S}^N$ is a symmetric matrix that is an affine function of \mathbf{z} :

$$\mathbf{M}(\mathbf{z}) = \mathbf{M}_0 - \sum_{i=1}^m z_i \mathbf{M}_i, \quad (10)$$

with $\mathbf{M}_0 = \sum_{j=m+1}^q \widehat{P}(\mathbf{x}_j) \mathbf{x}_j \mathbf{x}_j^\top$ and for $i \in [1, m]$ $\mathbf{M}_i = \mathbf{x}_i \mathbf{x}_i^\top$, $\mathbf{x}_i \in S_Q$. The minimal discrepancy distribution q^* is given by $q^*(\mathbf{x}_i) = z_i$, for all $i \in [1, m]$. Since $\|\mathbf{M}(\mathbf{z})\|_2 = \max\{\lambda_{\max}(\mathbf{M}(\mathbf{z})), \lambda_{\max}(-\mathbf{M}(\mathbf{z}))\}$, the problem can be rewritten equivalently as the following semi-definite programming (SDP) problem:

$$\begin{aligned} & \min_{\mathbf{z}, t} \quad t \\ \text{subject to} \quad & \begin{bmatrix} t\mathbf{I} & \mathbf{M}(\mathbf{z}) \\ \mathbf{M}(\mathbf{z}) & t\mathbf{I} \end{bmatrix} \succeq 0 \wedge \mathbf{1}^\top \mathbf{z} = 1 \wedge \mathbf{z} \geq 0. \end{aligned} \quad (11)$$

This problem can be solved in polynomial time using interior point methods. The time complexity for each iteration of the algorithm is in our notation [16][pp.234-235] : $O(m^3 + mN^3 + m^2N^2 + nN^2)$. This time complexity as well as its space complexity, which is in $O((m + N)^2)$, make such algorithms impractical for relatively large or realistic machine learning problems.

4.2 Discrepancy Minimization with Kernels

Here, we prove that the results of the previous section can be generalized to the case of high-dimensional feature spaces defined implicitly by a PDS kernel K . We denote by $\mathbf{K} = [K(\mathbf{x}_i, \mathbf{x}_j)]_{ij} \in \mathbb{R}^{q \times q}$ the kernel matrix associated to K for the full sample $S = S_Q \cup S_P$ and for any $\mathbf{z} \in \mathbb{R}^m$ by $\mathbf{D}(\mathbf{z})$ the diagonal matrix

$$\mathbf{D}(\mathbf{z}) = \text{diag}(-z_1, \dots, -z_m, \widehat{P}(\mathbf{x}_{m+1}), \dots, \widehat{P}(\mathbf{x}_{m+n})).$$

Theorem 3. *For any \widehat{Q} and \widehat{P} , the problem of determining the discrepancy minimizing distribution q^* for the squared loss L_2 and the hypothesis set H can be cast as an SDP of the same form as (9) but that depends only on the Gram matrix of the kernel K :*

$$\min_{\mathbf{z} \in \Delta_m} \|\mathbf{M}'(\mathbf{z})\|_2 \quad (12)$$

where $\mathbf{M}'(\mathbf{z}) = \mathbf{K}^{1/2} \mathbf{D}(\mathbf{z}) \mathbf{K}^{1/2} = \mathbf{M}'_0 - \sum_{i=1}^m z_i \mathbf{M}'_i$, with $\mathbf{M}'_0 = \mathbf{K}^{1/2} \mathbf{D}_0 \mathbf{K}^{1/2}$ and $\mathbf{M}'_i = \mathbf{K}^{1/2} \mathbf{D}_i \mathbf{K}^{1/2}$ for $i \in [1, m]$, and $\mathbf{D}_0, \mathbf{D}_1, \dots, \mathbf{D}_m \in \mathbb{R}^{q \times q}$ defined by $\mathbf{D}_0 = \text{diag}(0, \dots, 0, \widehat{P}(\mathbf{x}_{m+1}), \dots, \widehat{P}(\mathbf{x}_{m+n}))$, and for $i \geq 1$, \mathbf{D}_i is the diagonal matrix of the i th unit vector.

Proof. Let $\Phi: X \rightarrow \mathcal{F}$ be a feature mapping associated to K , with $\dim(\mathcal{F}) = N'$. Let $q = m + n$. The problem of finding the optimal distribution q^* is equivalent to solving

$$\min_{\substack{\|\mathbf{z}\|_1=1 \\ \mathbf{z} \geq 0}} \{\lambda_{\max}(\mathbf{M}(\mathbf{z})), \lambda_{\max}(-\mathbf{M}(\mathbf{z}))\}, \quad (13)$$

Algorithm 1

```

for  $k \geq 0$  do
     $\mathbf{v}_k \leftarrow T_C(\mathbf{u}_k)$ 
     $\mathbf{w}_k \leftarrow \operatorname{argmin}_{\mathbf{u} \in C} \left\{ \frac{L}{\sigma} d(\mathbf{u}) + \sum_{i=0}^k \frac{i+1}{2} [F(\mathbf{u}_i) + \langle \nabla F(\mathbf{u}_i), \mathbf{u} - \mathbf{u}_i \rangle] \right\}$ 
     $\mathbf{u}_{k+1} \leftarrow \frac{2}{k+3} \mathbf{w}_k + \frac{k+1}{k+3} \mathbf{v}_k$ 
end for

```

Fig. 1. Convex optimization algorithm

where the matrix $\mathbf{M}(\mathbf{z})$ is defined by

$$\mathbf{M}(\mathbf{z}) = \sum_{i=m+1}^q \widehat{P}(\mathbf{x}_i) \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_i)^\top - \sum_{i=1}^m z_i \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_i)^\top,$$

with q^* given by: $q^*(\mathbf{x}_i) = z_i$ for all $i \in [1, m]$. Let Φ denote the matrix in $\mathbb{R}^{N' \times q}$ whose columns are the vectors $\Phi(\mathbf{x}_1), \dots, \Phi(\mathbf{x}_{m+n})$. Then, observe that $\mathbf{M}(\mathbf{z})$ can be rewritten as

$$\mathbf{M}(\mathbf{z}) = \Phi \mathbf{D}(\mathbf{z}) \Phi^\top.$$

It is known that for any two matrices $\mathbf{A} \in \mathbb{R}^{N' \times q}$ and $\mathbf{B} \in \mathbb{R}^{q \times N'}$, \mathbf{AB} and \mathbf{BA} have the same eigenvalues. Thus, matrices $\mathbf{M}(\mathbf{z}) = (\Phi \mathbf{D}(\mathbf{z})) \Phi^\top$ and $\Phi^\top (\Phi \mathbf{D}(\mathbf{z})) = \mathbf{KD}(\mathbf{z})$ have the same eigenvalues. $\mathbf{KD}(\mathbf{z})$ is not a symmetric matrix. To ensure that we obtain an SDP of the same form as ⑨ minimizing the spectral norm of a symmetric matrix, we can instead consider the matrix $\mathbf{M}'(\mathbf{z}) = \mathbf{K}^{1/2} \mathbf{D}(\mathbf{z}) \mathbf{K}^{1/2}$, which, by the same argument as above has the same eigenvalues as $\mathbf{KD}(\mathbf{z})$ and therefore $\mathbf{M}(\mathbf{z})$. In particular, $\mathbf{M}'(\mathbf{z})$ and $\mathbf{M}(\mathbf{z})$ have the same maximum and minimum eigenvalues, thus, $\|\mathbf{M}(\mathbf{z})\|_2 = \|\mathbf{M}'(\mathbf{z})\|_2$. Since $\mathbf{D} = \mathbf{D}_0 - \sum_{i=1}^m z_i \mathbf{D}_i$, this concludes the proof. \square

Thus, the discrepancy minimization problem can be formulated in both the original input space and in the RKHS defined by a PDS kernel K as an SDP of the same form. In the next section, we present a specific study of this SDP and use results from smooth convex optimization as well as specific characteristics of the SDP considered in our case to derive an efficient and practical adaptation algorithm.

5 Algorithm

This section presents an algorithm for solving the discrepancy minimization problem using the smooth approximation technique of Nesterov ⑭. A general algorithm was given by Nesterov ⑮ to solve convex optimization problems of the form

$$\text{minimize}_{\mathbf{z} \in C} F(\mathbf{z}), \quad (14)$$

where C is a closed convex set and F admits a Lipschitz continuous gradient over C in time $O(1/\sqrt{\epsilon})$, which was later proven to be optimal for this class of problems. The pseudocode of the algorithm is given in Figure 1. Here, $T_C(\mathbf{u}) \in C$ denotes for any $\mathbf{u} \in C$, an element of $\operatorname{argmin}_{\mathbf{v} \in C} \langle \nabla F(\mathbf{u}), \mathbf{v} - \mathbf{u} \rangle + \frac{1}{2} L \|\mathbf{v} - \mathbf{u}\|^2$ (the specific choice of the minimizing \mathbf{v} is arbitrary for a given \mathbf{u}). d denotes a *prox-function* for C , that is

Algorithm 2

```

 $\mathbf{u}_0 \leftarrow \operatorname{argmin}_{\mathbf{u} \in C} \mathbf{u}^\top \mathbf{J} \mathbf{u}$ 
for  $k \geq 0$  do
     $\mathbf{v}_k \leftarrow \operatorname{argmin}_{\mathbf{u} \in C} \frac{2p-1}{2} (\mathbf{u} - \mathbf{u}_k)^\top \mathbf{J} (\mathbf{u} - \mathbf{u}_k) + \nabla G_p(\mathbf{M}(\mathbf{u}_k))^\top \mathbf{u}$ 
     $\mathbf{w}_k \leftarrow \operatorname{argmin}_{\mathbf{u} \in C} \frac{2p-1}{2} (\mathbf{u} - \mathbf{u}_0)^\top \mathbf{J} (\mathbf{u} - \mathbf{u}_0) + \sum_{i=0}^k \frac{i+1}{2} \nabla G_p(\mathbf{M}(\mathbf{u}_i))^\top \mathbf{u}$ 
     $\mathbf{u}_{k+1} \leftarrow \frac{2}{k+3} \mathbf{w}_k + \frac{k+1}{k+3} \mathbf{v}_k$ 
end for

```

Fig. 2. Smooth approximation algorithm

d is a continuous and strongly convex function over C with respect to the norm $\|\cdot\|$ with convexity parameter $\sigma > 0$ and $d(\mathbf{u}_0) = 0$ where $\mathbf{u}_0 = \operatorname{argmin}_{\mathbf{u} \in C} d(\mathbf{u})$. The following convergence guarantee was given for this algorithm [14].

Theorem 4. Let \mathbf{z}^* be an optimal solution for problem (14) and let \mathbf{v}_k be defined as in Algorithm 1 then for any $k \geq 0$, $F(\mathbf{v}_k) - F(\mathbf{z}^*) \leq \frac{4Ld(\mathbf{z}^*)}{\sigma(k+1)(k+2)}$.

Algorithm 1 can be further used to solve in $O(1/\epsilon)$ optimization problems of the same form where F is a Lipschitz-continuous non-smooth convex function [15]. This can be done by finding a uniform ϵ -approximation of F by a smooth convex function G with Lipschitz-continuous gradient. This is the technique we consider in the following. Recall the general form of the discrepancy minimization SDP in the feature space:

$$\begin{aligned} & \text{minimize} && \|\mathbf{M}(\mathbf{z})\|_2 \\ & \text{subject to} && \mathbf{M}(\mathbf{z}) = \sum_{i=0}^m z_i \mathbf{M}_i \wedge z_0 = -1 \wedge \sum_{i=1}^m z_i = 1 \wedge \forall i \in [1, m], z_i \geq 0, \end{aligned} \quad (15)$$

where $\mathbf{z} \in \mathbb{R}^{m+1}$ and where the matrices $\mathbf{M}_i \in \mathbb{S}_+^N$, $i \in [0, m]$, are fixed SPSD matrices. Thus, here $C = \{\mathbf{z} \in \mathbb{R}^{m+1} : z_0 = -1 \wedge \sum_{i=1}^m z_i = 1 \wedge \forall i \in [1, m], z_i \geq 0\}$. We further assume in the following that the matrices \mathbf{M}_i are linearly independent since the problem can be reduced to that case straightforwardly. The symmetric matrix $\mathbf{J} = [\langle \mathbf{M}_i, \mathbf{M}_j \rangle_F]_{i,j} \in \mathbb{R}^{(m+1) \times (m+1)}$ is then PDS and we will be using the norm $\mathbf{x} \mapsto \sqrt{\langle \mathbf{J}\mathbf{x}, \mathbf{x} \rangle} = \|\mathbf{x}\|_{\mathbf{J}}$ on \mathbb{R}^{m+1} .

A difficulty in solving this SDP is that the function $F: \mathbf{z} \mapsto \|\mathbf{M}(\mathbf{z})\|_2$ is not differentiable since eigenvalues are not differentiable functions at points where they coalesce, which, by the nature of the minimization, is likely to be the case precisely at the optimum. Instead, we can seek a smooth approximation of that function. One natural candidate is the function $\mathbf{z} \mapsto \|\mathbf{M}(\mathbf{z})\|_F^2$. However, the Frobenius norm can lead to a very coarse approximation of the spectral norm. As suggested by Nesterov [15], the function $G_p: \mathbf{M} \mapsto \frac{1}{2} \operatorname{Tr}[\mathbf{M}^{2p}]^{\frac{1}{p}}$, where $p \geq 1$ is an integer, can be used to give a smooth approximation. Indeed, let $\lambda_1(\mathbf{M}) \geq \lambda_2(\mathbf{M}) \geq \dots \geq \lambda_N(\mathbf{M})$ denote the list of the eigenvalues of a matrix $\mathbf{M} \in \mathbb{S}^N$ in decreasing order. By the definition of the trace, for all $\mathbf{M} \in \mathbb{S}^N$, $G_p(\mathbf{M}) = \frac{1}{2} \left[\sum_{i=1}^N \lambda_i^{2p}(\mathbf{M}) \right]^{\frac{1}{p}}$, thus

$$\frac{1}{2} \lambda^2 \leq G_p(\mathbf{M}) \leq \frac{1}{2} (\operatorname{rank}(\mathbf{M}) \lambda^{2p})^{\frac{1}{p}},$$

where $\lambda = \max\{\lambda_1(\mathbf{M}), -\lambda_N(\mathbf{M})\} = \|\mathbf{M}\|_2$. Thus, if we choose r as the maximum rank, $r = \max_{\mathbf{z} \in C} \text{rank}(\mathbf{M}(\mathbf{z})) \leq \max\{N, \sum_{i=0}^n \text{rank}(\mathbf{M}_i)\}$, then for all $\mathbf{z} \in C$,

$$\frac{1}{2}\|\mathbf{M}(\mathbf{z})\|_2^2 \leq G_p(\mathbf{M}(\mathbf{z})) \leq \frac{1}{2}r^{\frac{1}{p}}\|\mathbf{M}(\mathbf{z})\|_2^2. \quad (16)$$

This leads to a smooth approximation algorithm for solving the SDP (15) derived from Algorithm 1 by replacing the objective function F with G_p . Choosing the prox-function $d: \mathbf{u} \mapsto \frac{1}{2}\|\mathbf{u} - \mathbf{u}_0\|_J^2$ leads to the algorithm whose pseudocode is given in Figure 2 after some minor simplifications. The following theorem guarantees that its maximum number of iterations to achieve a relative accuracy of ϵ is in $O(\sqrt{r \log r}/\epsilon)$.

Theorem 5. *For any $\epsilon > 0$, Algorithm 2 solves the SDP (15) with relative accuracy ϵ in at most $4\sqrt{(1 + \epsilon)r \log r}/\epsilon$ iterations using the objective function G_p with $p \in [q_0, 2q_0]$ and $q_0 = (1 + \epsilon)(\log r)/\epsilon$.*

Proof. The proof follows directly [14], it is given in Appendix A for completeness. \square

The first step of the algorithm consists of computing the vector \mathbf{u}_0 by solving the simple QP of line 1. We now discuss in detail how to efficiently compute the steps of each iteration of the algorithm in the case of our discrepancy minimization problems.

Each iteration of the algorithm requires solving two simple QPs (lines 3 and 4). To do so, the computation of the gradient $\nabla G_p(\mathbf{M}(\mathbf{u}_k))$ is needed. This will therefore represent the main computational cost at each iteration other than solving the QPs already mentioned since, clearly, the sum $\sum_{i=0}^k \frac{i+1}{2} \nabla G_p(\mathbf{M}(\mathbf{u}_i))^T \mathbf{u}$ required at line 4 can be computed in constant time from its value at the previous iteration. Since for any $\mathbf{z} \in \mathbb{R}^m$

$$G_p(\mathbf{M}(\mathbf{z})) = \text{Tr}[\mathbf{M}^{2p}(\mathbf{z})]^{1/p} = \text{Tr} \left[\left(\sum_{i=0}^m z_i \mathbf{M}_i \right)^{2p} \right]^{1/p},$$

using the linearity of the trace operator, the i th coordinate of the gradient is given by

$$[\nabla G_p(\mathbf{M}(\mathbf{z}))]_i = \langle \mathbf{M}^{2p-1}(\mathbf{z}), \mathbf{M}_i \rangle_F \text{Tr}[\mathbf{M}^{2p}(\mathbf{z})]^{\frac{1}{p}-1}, \quad (17)$$

for all $i \in [0, m]$. Thus, the computation of the gradient can be reduced to that of the matrices $\mathbf{M}^{2p-1}(\mathbf{z})$ and $\mathbf{M}^{2p}(\mathbf{z})$. When the dimension of the feature space N is not too large, both $\mathbf{M}^{2p-1}(\mathbf{z})$ and $\mathbf{M}^{2p}(\mathbf{z})$ can be computed via $O(\log p)$ matrix multiplications using the binary decomposition method to compute the powers of a matrix [5]. Since each matrix multiplication takes $O(N^3)$, the total computational cost for determining the gradient is then in $O((\log p)N^3)$. The cubic-time matrix multiplication can be replaced by more favorable complexity terms of the form $O(N^{2+\alpha})$, with $\alpha = .376$. Alternatively, for large values of N , that is $N \gg (m + n)$, in view of Theorem 3 we can instead solve the kernelized version of the problem. Since it is formulated as the same SDP, the same smooth optimization technique can be applied. Instead of $\mathbf{M}(\mathbf{z})$, we need to consider the matrix $\mathbf{M}'(\mathbf{z}) = \mathbf{K}^{1/2} \mathbf{D}(\mathbf{z}) \mathbf{K}^{1/2}$. Now, observe that

$$\mathbf{M}'^{2p}(\mathbf{z}) = [\mathbf{K}^{1/2} \mathbf{D}(\mathbf{z}) \mathbf{K}^{1/2}]^{2p} = \mathbf{K}^{1/2} [\mathbf{D}(\mathbf{z}) \mathbf{K}]^{2p-1} \mathbf{D}(\mathbf{z}) \mathbf{K}^{1/2}.$$

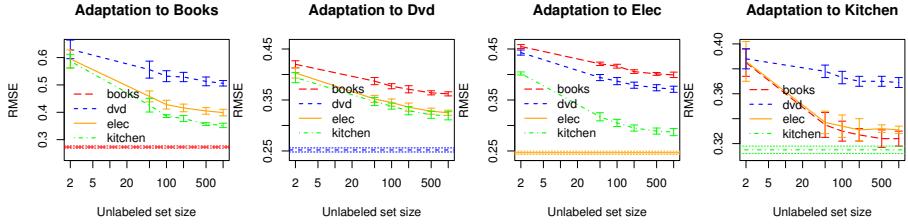


Fig. 3. Performance improvement of the RMSE for the 12 adaptation tasks as a function of the size of the unlabeled data used. Note that the figures do not make use of the same y-scale.

Thus, by the property of the trace operator,

$$\text{Tr}[\mathbf{M}'^{2p}(\mathbf{z})] = \text{Tr}[\mathbf{D}(\mathbf{z})\mathbf{K}^{1/2}\mathbf{K}^{1/2}[\mathbf{D}(\mathbf{z})\mathbf{K}]^{2p-1}] = \text{Tr}[[\mathbf{D}(\mathbf{z})\mathbf{K}]^{2p}]. \quad (18)$$

The other term appearing in the expression of the gradient can be computed as follows:

$$\begin{aligned} \langle \mathbf{M}'^{2p-1}(\mathbf{z}), \mathbf{M}'_i \rangle_F &= \text{Tr}[[\mathbf{K}^{1/2}\mathbf{D}(\mathbf{z})\mathbf{K}^{1/2}]^{2p-1}\mathbf{K}^{1/2}\mathbf{D}_i\mathbf{K}^{1/2}] \\ &= \text{Tr}[\mathbf{K}^{1/2}[\mathbf{D}(\mathbf{z})\mathbf{K}]^{2p-2}\mathbf{D}(\mathbf{z})\mathbf{K}^{1/2}\mathbf{K}^{1/2}\mathbf{D}_i\mathbf{K}^{1/2}] \\ &= \text{Tr}[\mathbf{K}[\mathbf{D}(\mathbf{z})\mathbf{K}]^{2p-1}\mathbf{D}_i], \end{aligned}$$

for any $i \in [1, m]$. Observe that multiplying a matrix \mathbf{A} by \mathbf{D}_i is equivalent to zeroing all of its columns but the i th one, therefore $\text{Tr}[\mathbf{AD}_i] = \mathbf{A}_{ii}$. In view of that,

$$\langle \mathbf{M}'^{2p-1}(\mathbf{z}), \mathbf{M}'_i \rangle_F = [\mathbf{K}[\mathbf{D}(\mathbf{z})\mathbf{K}]^{2p-1}]_{ii}. \quad (19)$$

Therefore, the diagonal of the matrix $\mathbf{K}[\mathbf{D}(\mathbf{z})\mathbf{K}]^{2p-1}$ provides all these terms. Thus, in view of (18) and (19), the gradient given by (17) can be computed directly from the $(2p)$ th and $(2p-1)$ th powers of the matrix $\mathbf{D}(\mathbf{z})\mathbf{K}$. The iterated powers of this matrix, $[\mathbf{D}(\mathbf{z})\mathbf{K}]^{2p}(\mathbf{z})$ and $[\mathbf{D}(\mathbf{z})\mathbf{K}]^{2p-1}(\mathbf{z})$, can be both computed using a binary decomposition in time $O((\log p)(m+n)^3)$. This is a significantly more efficient computational cost per iteration for $N \gg (m+n)$. It is also substantially more favorable than the iteration cost for solving the SDP using interior-point methods $O(m^3 + mN^3 + m^2N^2 + nN^2)$. Furthermore, the space complexity of the algorithm is only in $O((m+n)^2)$.

6 Experiments

This section reports the results of extensive experiments demonstrating both the effectiveness of discrepancy minimization in adaptation when using kernel ridge regression and the efficiency of our optimization algorithm. Our results show that the adaptation algorithm presented is practical even for relatively large data sets and for high-dimensional feature spaces.

For our experiments, we used the multi-domain sentiment dataset (version 1.0) of Blitzer et al. [3]. This data set has been used in several publications [4, 11], but despite the ordinal nature of the star labeling of the data, it has always been treated as a classification task, and not as a regression task which is the focus of this work. We are not aware of any other adaptation datasets that can be applied to the regression task.

Table 1. RMSE results obtained for the 12 adaptation tasks. Each field of the table has three results: from training only on the source data (top), from the adaptation task (middle), and from training only on the target data (bottom).

	books	dvd	elec	kitchen
books	.450 ± .005	.544 ± .002	.331 ± .001	
	.273 ± .004	.362 ± .004	.407 ± .009	.324 ± .006
	.252 ± .004	.246 ± .003	.315 ± .003	
dvd	.546 ± .007	.505 ± .004	.383 ± .003	
	.506 ± .010	.252 ± .004	.371 ± .006	.369 ± .004
	.273 ± .004	.246 ± .003	.315 ± .003	
elec	.412 ± .005	.429 ± .006	.345 ± .004	
	.399 ± .012	.325 ± .005	.246 ± .003	.331 ± .003
	.273 ± .004	.252 ± .004	.315 ± .003	
kitchen	.360 ± .003	.412 ± .002	.330 ± .003	
	.352 ± .008	.319 ± .008	.287 ± .007	.315 ± .003
	.273 ± .004	.252 ± .004	.246 ± .003	

To make the data conform with the regression setting discussed in the previous sections, we first convert the discrete labels to regression values by fitting all the data for each of the four tasks `books`, `dvd`, `elec`, and `kitchen` a Gaussian kernel ridge regression with a relatively small width $\sigma = 1$ as feature vectors the normalized counts of the top 5,000 unigrams and bigrams, as measured across all four tasks. These regression values are used as target values for all subsequent modeling.

We then define 12 adaptation problems for each pair of distinct tasks (`task`, `task'`), where `task` and `task'` are in $\{\text{books}, \text{dvd}, \text{elec}, \text{kitchen}\}$. For each of these problems, the source empirical distribution is a mixture defined by 500 labeled points from `task` and 200 from `task'`. This is intended to make the source and target distributions reasonably close, a condition for the theory developed in this paper, but the algorithm receives of course no information about this definition of the source distribution. The target distribution is defined by another set of points all from `task'`.

Figure 3 shows the performance of the algorithm on the 12 adaptation tasks between distinct domains plotted as a function of the amount of unlabeled data received from the target domain. The optimal performance obtained by training purely on the same amount of labeled data from the target domain is also indicated in each case. The input features are again the normalized counts of the top 5,000 unigrams and bigrams, as measured across all four tasks, and for modeling we use kernel ridge regression with the Gaussian kernel of the same width $\sigma = 1$. This setup guarantees that the target labeling function is in the hypothesis space, a condition matching one of the settings analyzed in our theoretical study. The results are mean values obtained from 9-fold cross validation and we plot mean values ± one standard deviation. As can be seen from the figure, adaptation improves, as expected, with increasing amounts of data. One can also observe that not all data sets are equally beneficial for adaptation. The `kitchen` task primarily discusses electronic gadgets for kitchen use, and hence the `kitchen` and `elec` data sets adapt well to each other, an observation also made by Blitzer et al. [3].

Our results on the adaptation tasks are also summarized in Table 1. The row name indicates the source domain and the column name the target domain. Due to lack of space, we only list the results for adaptation with 1,000 unlabeled points from the target domain. In this table, we also provide for reference the results from training purely with

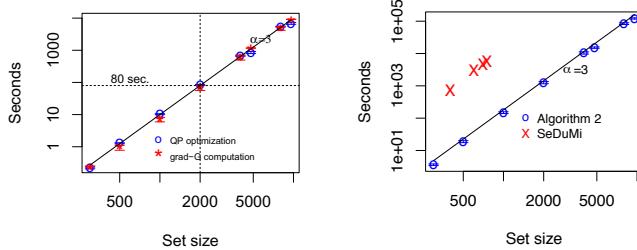


Fig. 4. The left panel shows a plot reporting run times measured empirically (mean \pm one standard deviation) for the QP optimization and the computation of ∇G_p as a function of the sample size (log-log scale). The right panel compares the total time taken by Algorithm 2 to compute the optimization solution, to the one taken by SeDuMi (log-log scale).

labeled data from the source or target domain. We are not aware of any other adaptation algorithms for the regression tasks with which we can compare our performance results.

Algorithm 2 requires solving several QPs to compute \mathbf{u}_0 and \mathbf{u}_{k+1} , $k \geq 0$. Since $\mathbf{u}_{k+1} \in \mathbb{R}^{m+1}$, the cost of these computations only depends on the size of the labeled sample m , which is relatively small. Figure 4 displays average run times obtained for m in the range 500 to 10,000. All experiments were carried out on a single processor of an Intel Zeon 2.67GHz CPU with 12GB of memory. The algorithm was implemented in R and made use of the quadprog optimization package. As can be seen from the figure, the run times scale cubically in the sample size, reaching roughly 10s for $m = 1,000$.

The dominant cost of each iteration of Algorithm 2 is the computation of the gradient $\nabla G_p(\mathbf{M}(\mathbf{u}_k))$, as already pointed out in Section 5. The iterated power method provides a cost per iteration of $O((\log p)(m+n)^3)$, and thus depends on the combined size of the labeled and unlabeled data. Figure 4 shows typical timing results obtained for different samples sizes in the range $m+n = 500$ to $m+n = 10,000$ for $p = 16$, which empirically was observed to guarantee convergence. For a sample size of $m+n = 2,000$ the time is about 80 seconds. With 5 iterations of Algorithm 2 the total time is $5 \times (80 + 2 * 10) + 10 = 510$ seconds.

In contrast, even the most efficient SDP solvers publicly available, SeDuMi, cannot solve our discrepancy minimization SDPs for more than a few hundred points in the kernelized version. In our experiments, SeDuMi (<http://sedumi.ie.lehigh.edu/>) simply failed for set sizes larger than $m+n = 750$! In Figure 4, typical run times for Algorithm 2 with 5 iterations are compared to run times using SeDuMi.

7 Conclusion

We presented several theoretical guarantees for domain adaptation in regression and proved that the empirical discrepancy minimization can also be cast as an SDP when using kernels. We gave an efficient algorithm for solving that SDP using results from smooth optimization and specific characteristics of these SDPs in our adaptation case. Our adaptation algorithm is shown to scale to larger data sets than what could be afforded using the best existing software for solving such SDPs. Altogether, our results

form a complete solution for domain adaptation in regression, including theoretical guarantees, an efficient algorithmic solution, and extensive empirical results.

Acknowledgments. We thank Steve Boyd, Michael Overton, and Katya Scheinberg for discussions about the optimization problem addressed in this work.

References

- [1] Ben-David, S., Blitzer, J., Crammer, K., Pereira, F.: Analysis of representations for domain adaptation. In: NIPS 2006 (2007)
- [2] Ben-David, S., Lu, T., Luu, T., Pál, D.: Impossibility theorems for domain adaptation. Journal of Machine Learning Research - Proceedings Track 9, 129–136 (2010)
- [3] Blitzer, J., Dredze, M., Pereira, F.: Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In: ACL 2007 (2007)
- [4] Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Wortman, J.: Learning bounds for domain adaptation. In: NIPS 2007 (2008)
- [5] Cormen, T., Leiserson, C., Rivest, R.: Introduction to Algorithms. The MIT Press, Cambridge (1992)
- [6] Cortes, C., Vapnik, V.: Support-Vector Networks. Machine Learning 20(3) (1995)
- [7] Dredze, M., Blitzer, J., Talukdar, P.P., Ganchev, K., Graca, J., Pereira, F.: Frustratingly Hard Domain Adaptation for Parsing. In: CoNLL 2007 (2007)
- [8] Dudley, R.M.: Real Analysis and Probability. Wadsworth, Belmont (1989)
- [9] Jiang, J., Zhai, C.: Instance Weighting for Domain Adaptation in NLP. In: Proceedings of ACL 2007, pp. 264–271 (2007)
- [10] Legetter, C.J., Woodland, P.C.: Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. Comp. Speech and Lang. (1995)
- [11] Mansour, Y., Mohri, M., Rostamizadeh, A.: Domain adaptation: Learning bounds and algorithms. In: Proceedings of COLT 2009. Omnipress, Montréal, Canada (2009)
- [12] Martínez, A.M.: Recognizing imprecisely localized, partially occluded, and expression variant faces from a single sample per class. IEEE Trans. Pattern Anal. 24(6) (2002)
- [13] Nesterov, Y.: A method of solving a convex programming problem with convergence rate $O(1/k^2)$. Soviet Mathematics Doklady 27(2), 372–376 (1983)
- [14] Nesterov, Y.: Smooth minimization of non-smooth functions. Math. Program. 103, 127–152 (2005)
- [15] Nesterov, Y.: Smoothing technique and its applications in semidefinite optimization. Math. Program. 110, 245–259 (2007)
- [16] Nesterov, Y., Nemirovsky, A.: Interior Point Polynomial Methods in Convex Programming: Theory and Appl. SIAM, Philadelphia (1994)
- [17] Pietra, S.D., Pietra, V.D., Mercer, R.L., Roukos, S.: Adaptive language modeling using minimum discriminant estimation. In: HLT 1991: Workshop on Speech and Nat. Lang. (1992)
- [18] Rosenfeld, R.: A Maximum Entropy Approach to Adaptive Statistical Language Modeling. Computer Speech and Language 10, 187–228 (1996)
- [19] Saunders, C., Gammerman, A., Vovk, V.: Ridge Regression Learning Algorithm in Dual Variables. In: ICML (1998)
- [20] Steinwart, I.: On the influence of the kernel on the consistency of support vector machines. JMLR 2, 67–93 (2002)
- [21] Vapnik, V.N.: Statistical Learning Theory. J. Wiley & Sons, Chichester (1998)

A Proof of Theorem 5

Proof. Let $\|\mathbf{M}^*\|_2$ be the optimum of the SDP (15), $G_p(\mathbf{M}'^*)$ that of the SDP with F replaced with its smooth approximation G_p , and $\mathbf{z}^* \in C$ a solution of that SDP with relative accuracy ϵ . Then, for $p \geq \frac{(1+\epsilon)\log r}{\epsilon}$, in view of (16), \mathbf{z}^* is a solution of the original SDP (15) with relative accuracy ϵ :

$$\frac{\|\mathbf{M}(z^*)\|_2}{\|\mathbf{M}^*\|_2} \leq r^{\frac{1}{2p}} \frac{\sqrt{G_P(\mathbf{M}(\mathbf{z}^*))}}{\sqrt{G_p(\mathbf{M}'^*)}} \leq r^{\frac{1}{2p}}(1 + \epsilon)^{1/2} \leq (1 + \epsilon).$$

G_p can be shown to admit a Lipschitz gradient with Lipschitz constant $L = (2p - 1)$ with respect to the norm $\|\cdot\|_{\mathbf{J}}$ and the prox-function d can be chosen as $d(\mathbf{u}) = \frac{1}{2}\|\mathbf{u} - \mathbf{u}_0\|_{\mathbf{J}}^2$, with $\mathbf{u}_0 = \operatorname{argmin}_{\mathbf{u} \in C} \|\mathbf{u}\|_{\mathbf{J}}$ and convexity parameter $\sigma = 1$. It can be shown that $d(\mathbf{z}^*) \leq rG_p(\mathbf{M}'^*)$. Thus, in view of Theorem 4, $\frac{G_p(\mathbf{M}(z_k)) - G_p(\mathbf{M}'^*)}{G_p(\mathbf{M}'^*)} \leq \frac{4(2p-1)r}{(k+1)(k+2)}$. Choosing p such that $2p < 4\frac{(1+\epsilon)\log r}{\epsilon}$, and setting the right-hand side to $\epsilon > 0$, gives the following maximum number of iterations to achieve a relative accuracy of ϵ using Algorithm 2: $k^* = \sqrt{(16r(1 + \epsilon)\log r)/\epsilon^2} = 4\sqrt{(1 + \epsilon)r\log r}/\epsilon$. \square

Approximate Reduction from AUC Maximization to 1-Norm Soft Margin Optimization

Daiki Suehiro, Kohei Hatano, and Eiji Takimoto

Department of Informatics, Kyushu University
`{daiki.suehiro,hatano,eiji}@inf.kyushu-u.ac.jp`

Abstract. Finding linear classifiers that maximize AUC scores is important in ranking research. This is naturally formulated as a 1-norm hard/soft margin optimization problem over pn pairs of p positive and n negative instances. However, directly solving the optimization problems is impractical since the problem size (pn) is quadratically larger than the given sample size ($p + n$). In this paper, we give (approximate) reductions from the problems to hard/soft margin optimization problems of linear size. First, for the hard margin case, we show that the problem is reduced to a hard margin optimization problem over $p + n$ instances in which the bias constant term is to be optimized. Then, for the soft margin case, we show that the problem is approximately reduced to a soft margin optimization problem over $p + n$ instances for which the resulting linear classifier is guaranteed to have a certain margin over pairs.

1 Introduction

Learning to rank has been one of the most active areas of research in machine learning and information retrieval in the past decade, due to increasing demands in, for example, recommendation tasks and financial risk analysis [5, 13, 8, 4, 21, 6, 19, 2, 14]. Among the problems related to learning to rank, the bipartite ranking is a fundamental problem, which involves learning to obtain rankings over positive and negative instances. More precisely, for a given sample consisting of positive and negative instances, the goal of the bipartite ranking problem is to find a real-valued function h , which is referred to as a ranking function, with the following property: For a randomly chosen test pair of positive instance \mathbf{x}^+ and negative instance \mathbf{x}^- , the ranking function h maps \mathbf{x}^+ to a higher value than \mathbf{x}^- with high probability. Thus, a natural measure for evaluating the goodness of ranking function h is the probability that $h(\mathbf{x}^+) > h(\mathbf{x}^-)$, which we call the AUC of h .

The bipartite ranking problem can be reduced to the binary classification problem over a new instance space, consisting of all pairs $(\mathbf{x}^+, \mathbf{x}^-)$ of positive and negative instances. More precisely, the problem of maximizing the AUC is equivalent to finding a binary classifier f of the form of $f(\mathbf{x}^+, \mathbf{x}^-) = h(\mathbf{x}^+) - h(\mathbf{x}^-)$ so that the probability that $f(\mathbf{x}^+, \mathbf{x}^-) > 0$ is maximized for a randomly

chosen instance pair. Several studies including RankSVMs [13, 4] have taken this approach with linear classifiers as the ranking functions. RankSVMs are justified by generalization bounds [21, 2] which say that a large margin over pairs of positive and negative instances in the sample implies a high AUC score under the standard assumption that instances are drawn i.i.d. under the underlying distribution.

The reduction approach, however, has a drawback that the sample constructed through the reduction is of size pn when the original sample consists of p positive and n negative instances. This is a quadratic blowup in size.

In this paper, we formulate AUC maximization as 1-norm hard/soft margin optimization problems¹ over pn pairs of p positive and n negative instances. We show some reduction schemes to 1-norm hard (or soft) margin optimization over $p+n$ instances which approximate the original problem over pairs. First, for the hard margin case where the resulting linear classifier is supposed to classify all pairs correctly by some positive margin, we show that the original problem over pairs is equivalent to the 1-norm hard margin problem over $p+n$ instances with the bias term.

Second, for the soft margin case, in which the resulting classifier is allowed to misclassify a number of pairs, we show reduction methods to 1-norm soft margin optimization over instances that are guaranteed to have a certain margin over pairs of instance. When we solve the original problem over pairs, it can be shown that for any ε s.t. $0 < \varepsilon < 1$, the solution has a margin of least $\rho^* \geq \gamma^*$ over at least $(1 - \varepsilon)pn$ pairs, where ρ^* and γ^* are optimal solutions of the primal and dual problems of the original problem. Note that the optimal solutions ρ^* and γ^* depend on ε respectively. On the other hand, for an appropriate parameter setting, one of our reduction methods guarantees that the resulting classifier has a margin of at least γ^* for $(1 - \sqrt{\varepsilon})^2pn$ pairs. Note that, this guarantee might be rather weak, since the guaranteed margin γ^* is lower than the optimal margin ρ^* in general. However, if $\rho^* \approx \gamma^*$, say, when pairs are close to be linearly separable, our theoretical guarantee becomes sharper. Also, theoretically guaranteed reduction methods from AUC maximization to classification are quite meaningful since typical methods lack such properties.

We should note that our theoretical guarantee itself is not new. SoftRankBoost [15] is proved to have the same guarantee. But our reduction methods and SoftRankBoost are totally different. SoftRankBoost is designed using the smooth boosting framework [7, 23, 11, 12, 3]. On the other hand, our methods are built from an optimization theoretic perspective and provide a much clearer understanding for underlying optimization problems. In addition, our methods motivate practical heuristics to further improve AUCs.

In experiments using artificial and real data, the practical heuristics derived from the analysis achieve AUCs that are almost as high as the original soft

¹ In this paper we refer to 1-norm soft margin optimization as a soft margin optimization with 1-norm of the weight vector regularized. Note that sometimes the soft margin optimization of SVMs with 1-norm of slack variables optimized is also called 1-norm soft margin optimization.

margin formulation over pairs while keeping the sample size linear. In addition, our methods also outperform previous methods including RankBoost [8] and SoftRankBoost.

There have been a number of studies in this field. Brefeld and Scheffer [4] and Fung et al. [10] proposed reduction methods from RankSVMs or 2-norm soft margin optimization over pairs to 2-norm soft margin optimization over instances. Raykar et al. investigated similar problems in the logistic regression framework [18]. These reduction methods, however, do not have theoretical guarantees similar to ours. Further, these researches consider soft margin optimization problems where 2-norm of the weight vector is regularized. On the other hand, in our soft margin optimization, 1-norm of the weight vector is regularized. So, the resulting weight vector tends to be sparse, which is useful for feature selection. Freund et al proposed RankBoost [8], which is an efficient implementation of AdaBoost [9] over pairs of positive and negative instances and runs in linear time for a given sample size. Rudin and Schapire further demonstrated that under certain assumptions, AdaBoost is equivalent to RankBoost [21]. Since AdaBoost is shown to have at least half of the maximum margin asymptotically for the 1 norm hard margin optimization(see, e.g., [16, 17]), RankBoost and AdaBoost also have large margins over pairs. Rudin also proposed the P-Norm Push, which maximizes a criterion that assigns higher weights to rankings among top instances [20].

2 Preliminaries

Let \mathcal{X}^+ and \mathcal{X}^- be the sets of positive instances and negative instances, respectively. Let $\mathcal{X} = \mathcal{X}^+ \cup \mathcal{X}^-$ be the instance space. A distribution D over \mathcal{X} is said to be nontrivial if D has non-zero probability over both positive and negative instances. Given a non-trivial distribution D , we denote D^+ and D^- as the marginal distribution of D over positive and negative instances, respectively. A ranking function h is any function from \mathcal{X} to $[-1, +1]$. The AUC of hypothesis h with respect to a non-trivial distribution D over \mathcal{X} is given as

$$AUC_D(h) = \Pr_{\mathbf{x}^+, \mathbf{x}^- \sim D} \{h(\mathbf{x}^+) > h(\mathbf{x}^-) \mid \mathbf{x}^+ \in \mathcal{X}^+, \mathbf{x}^- \in \mathcal{X}^-\},$$

where each \mathbf{x}^+ and \mathbf{x}^- is drawn independently with respect to D .

Let S be a set of $m (= p + n)$ instances drawn i.i.d. with respect to D , which includes p positive instances and n negative instances, respectively. Let $S^+ = \{\mathbf{x}_1^+, \dots, \mathbf{x}_p^+\}$ and $S^- = \{\mathbf{x}_1^-, \dots, \mathbf{x}_n^-\}$, be the subsets of positive and negative instances respectively.

Given $\rho > 0$, we define

$$AUC_{S, \rho}(h) = \frac{\sum_{i=1}^p \sum_{j=1}^n I(h(\mathbf{x}_i^+) - h(\mathbf{x}_j^-) \geq \rho)}{pn},$$

where $I(\cdot)$ is the indicator function. The following theorem was presented by Rudin and Schapire.

Theorem 1 (Rudin and Schapire [21]). Let \mathcal{F} be a set of ranking functions. Then, for any $\varepsilon > 0$, $\rho > 0$, for any $h \in \mathcal{F}$, the following holds

$$AUC_D(h) \geq AUC_{S,\rho}(h) - \varepsilon \quad (1)$$

with a probability of at least $1 - 2\mathcal{N}(\mathcal{F}, \frac{\rho}{4}) \exp\left\{-\frac{m\varepsilon^2 E^2}{8}\right\}$, where E is the expectation of $I(\mathbf{x}_i^+ \in \mathcal{X}^+, \mathbf{x}_j^- \in \mathcal{X}^-)$ when \mathbf{x}_i^+ and \mathbf{x}_j^- are drawn independently from D , and $\mathcal{N}(\mathcal{F}, \varepsilon)$ is the covering number of \mathcal{F} , which is defined as the minimum number of balls of radius ε needed to cover \mathcal{F} using L_∞ norm.

Here, note that the covering number is smaller if ρ is larger. So, a robust approach to learn a hypothesis with high AUC is to enlarge $AUC_{S,\rho}(h)$ for some large ρ .

2.1 1-Norm Soft Margin over Pairs of Positive and Negative Instances

In this paper, we assume a finite set $\mathcal{H} = \{h_1, h_2, \dots, h_N\}$ of ranking functions, which are functions from \mathcal{X} to $[-1, +1]$. Our hypothesis class \mathcal{F} is the set of convex combination of ranking functions in \mathcal{H} , i.e.,

$$\mathcal{F} = \left\{ f \mid f(\mathbf{x}) = \sum_{k=1}^N \alpha_k h_k(\mathbf{x}), h_k \in \mathcal{H}, \sum_{k=1}^N \alpha_k = 1, \alpha_k \geq 0 \right\}.$$

Now, our goal is to find a linear combination of ranking functions $f \in \mathcal{F}$ that has a large margin ρ over pairs of instances in S^+ and S^- .

More formally, we formulate our problem as optimizing the soft margin over pairs of positive and negative instances. For convenience, for any $q \geq 1$, let \mathcal{P}_q be the q -dimensional probability simplex, i.e., $\mathcal{P}_q = \{\mathbf{p} \in [0, 1]^q \mid \sum_i p_i = 1\}$. Then, for positive and negative sets of instances S^+ and S^- , the set \mathcal{H} of ranking functions, and any fixed $\nu \in \{1, \dots, pn\}$, the 1-norm soft margin optimization problem is given as follows:

$$\begin{aligned} (\rho^*, \boldsymbol{\alpha}^*, \boldsymbol{\xi}^*) &= \max_{\rho, \boldsymbol{\alpha}, \boldsymbol{\xi}} \rho - \frac{1}{\nu} \sum_{i=1}^p \sum_{j=1}^n \xi_{ij} \\ &\text{sub.to} \\ &\sum_k \alpha_k (h_k(\mathbf{x}_i^+) - h_k(\mathbf{x}_j^-))/2 \geq \rho - \xi_{ij} \quad (i = 1, \dots, p, j = 1, \dots, n), \\ &\boldsymbol{\alpha} \in \mathcal{P}_N, \\ &\xi_{ij} \geq 0 \quad (i = 1, \dots, p, j = 1, \dots, n). \end{aligned} \quad (2)$$

In this problem, the goal is to maximize the margin ρ of the linear combination $\boldsymbol{\alpha}$ of ranking functions w.r.t. instances as well as to minimize the sum of “losses” ξ_{ij} , the quantity by which the target margin ρ is violated. Here $\nu \in \{1, \dots, pn\}$ controls the tradeoff between the two objectives.

Then, using Lagrangian multipliers, the dual problem is given as

$$(\gamma^*, \mathbf{d}^*) = \min_{\gamma, \mathbf{d}} \gamma \quad (3)$$

sub.to

$$\sum_{i,j} d_{ij} (h_k(\mathbf{x}_i^+) - h_k(\mathbf{x}_j^-))/2 \leq \gamma \quad (k = 1, \dots, N),$$

$$0 \leq d_{ij} \leq \frac{1}{\nu} \quad (i = 1, \dots, p, j = 1, \dots, n),$$

$$\mathbf{d} \in \mathcal{P}_{pn}.$$

Since the problem is a linear program, by duality, we have $\rho^* - \frac{1}{\nu} \sum_{i,j} \xi_{ij}^* = \gamma^*$.

Furthermore, by using KKT conditions, it can be shown that (see, e.g., [22, 24]), the optimal solution guarantees the number of pairs $(\mathbf{x}_i^+, \mathbf{x}_j^-)$ for which $\sum_k \alpha_k (h_k(\mathbf{x}_i^+) - h_k(\mathbf{x}_j^-))/2 \leq \rho^*$ is at most ν . In other words, setting $f = \sum_{k=1}^N \alpha_k h_k$, we have that $AUC_{S,\rho^*}(f)$ is at least $1 - \nu/pn$. Thus, solving 1-norm soft margin optimization pairs is a quite natural approach for improving the lower bound of $AUC_D(f)$.

3 1-Norm Hard Margin Optimization over Pairs

In this section, we show the equivalence between two hard margin optimization problems, the 1-norm hard margin problem over pairs and the 1-norm hard margin problem with bias. The hard margin optimization problem is a special case of the soft margin problem in that the resulting classifier or ranking function is supposed to predict all the instances or pairs correctly with a positive margin.

The first problem we consider is the 1-norm hard margin optimization over pairs of positive and negative instances.

$$\max_{\rho, \boldsymbol{\alpha} \in \mathcal{P}_N} \rho \quad (4)$$

sub.to

$$\sum_{k=1}^N \alpha_k (h_k(\mathbf{x}_i^+) - h_k(\mathbf{x}_j^-))/2 \geq \rho \quad (i = 1, \dots, p, j = 1, \dots, n).$$

The second hard margin problem is the 1-norm hard margin optimization with bias.

$$\max_{\rho, \boldsymbol{\alpha} \in \mathcal{P}_N, b} \rho \quad (5)$$

sub.to

$$\sum_{k=1}^N \alpha_k h_k(\mathbf{x}_i^+) + b \geq \rho \quad (i = 1, \dots, p),$$

$$\sum_{k=1}^N \alpha_k h_k(\mathbf{x}_j^-) + b \leq -\rho \quad (j = 1, \dots, n).$$

In the following, we show that both of these problems are equivalent, in the sense that we can construct an optimal solution of one problem from an optimal solution of the other problem.

Theorem 2. Let (ρ_b, α_b, b_b) be an optimal solution of the 1-norm hard margin optimization with bias (5). Then, (ρ_b, α_b) is also an optimal solution of the 1-norm hard margin optimization over pairs (4).

Proof. Let (ρ_p, α_p) be an optimal solution of the 1-norm hard margin optimization over pairs. Clearly, (ρ_b, α_b, b_b) is a feasible solution of the 1-norm hard margin optimization over pairs. So, $\rho_b \leq \rho_p$. Next, we show that the opposite is true. Let \mathbf{x}^+ and \mathbf{x}^- be positive and negative examples for which the margin of α_p is minimized. Note that for the pair $(\mathbf{x}^+, \mathbf{x}^-)$ the constraint holds with equality. Let

$$b_p = -\frac{\sum_k \alpha_{p,k} (h_k(\mathbf{x}^+) + h_k(\mathbf{x}^-))}{2}.$$

Then, (ρ_p, α_p, b_p) is a feasible solution of the 1-norm hard margin optimization with bias. For any positive instance \mathbf{x}_i^+ , observe that

$$\begin{aligned} \sum_{k=1}^N \alpha_{p,k} h_k(\mathbf{x}_i^+) + b_p &= \sum_{k=1}^N \alpha_{p,k} \frac{h_k(\mathbf{x}_i^+) - h_k(\mathbf{x}^-)}{2} + \sum_{k=1}^N \alpha_{p,k} \frac{h_k(\mathbf{x}_i^+) - h_k(\mathbf{x}^+)}{2} \\ &\geq \rho_p + \sum_{k=1}^N \alpha_{p,k} \frac{h_k(\mathbf{x}_i^+) - h_k(\mathbf{x}^-) - (h_k(\mathbf{x}^+) - h_k(\mathbf{x}^-))}{2} \\ &\geq \rho_p + \rho_p - \rho_p = \rho_p. \end{aligned}$$

A similar inequality holds for negative instances as well. Thus, we have $\rho_p \leq \rho_b$. \square

4 Reduction Methods from 1-Norm Soft Margin Optimization over Pairs

In this section, we propose reduction methods from the 1-norm soft margin optimization over pairs to that over instances.

4.1 Our Method

We would like to approximate the dual problem of the 1-norm soft margin optimization over pairs (3). The dual problem is concerned with finding a distribution over pn pairs of positive and negative instances satisfying the linear constraints. Our key idea is to replace the distribution d_{ij} with a product distribution $d_i^+ d_j^-$, where \mathbf{d}^+ , \mathbf{d}^- are distributions over positive and negative instances, respectively.

Letting $d_{ij} = d_i^+ d_j^-$, observe that

$$\begin{aligned} \sum_{i,j} d_{ij} \frac{h_k(\mathbf{x}_i^+) - h_k(\mathbf{x}_j^-)}{2} &= \sum_{i,j} d_i^+ d_j^- \frac{h_k(\mathbf{x}_i^+) - h_k(\mathbf{x}_j^-)}{2} \\ &= \frac{\sum_i d_i^+ h_k(\mathbf{x}_i^+) \sum_j d_j^-}{2} - \frac{\sum_j d_j^- h_k(\mathbf{x}_j^-) \sum_i d_i^+}{2} \\ &= \sum_i d_i^+ h_k(\mathbf{x}_i^+)/2 - \sum_j d_j^- h_k(\mathbf{x}_j^-)/2. \end{aligned}$$

Then, we obtain the following problem.

$$\min_{\mathbf{d}, \gamma} \gamma \quad (6)$$

sub.to

$$\begin{aligned} \sum_{i=1}^p d_i^+ h_k(\mathbf{x}_i^+)/2 - \sum_{j=1}^n d_j^- h_k(\mathbf{x}_j^-)/2 &\leq \gamma \quad (k = 1, \dots, N), \\ \mathbf{d}^+ \in \mathcal{P}_p, \mathbf{d}^- \in \mathcal{P}_n, \\ 0 \leq d_i^+ d_j^- &\leq \frac{1}{\nu} \quad (i = 1, \dots, p, j = 1, \dots, n). \end{aligned}$$

Since we restrict distributions to be products of two distributions, the optimal solution yields a feasible solution of the original problem (2). This problem has $p + n + 1$ variables, whereas the original problem has $pn + 1$ variables. So this problem would be easier to solve. But, unfortunately, this problem is not convex since the constraints $d_i^+ d_j^- \leq 1/\nu$ ($i = 1, \dots, p, j = 1, \dots, n$) are not convex.

Later herein, we propose a method by which to find a local minimum of this non-convex problem (6). First, however, we show a restricted the problem, the solution of which has a certain amount of margin over pairs. In order to avoid non-convex constraints, we fix ν^+ and ν^- such that $\nu = \nu^+ \nu^-$ and enforce $d_i^+ \leq 1/\nu^+$ and $d_j^- \leq 1/\nu^-$. Equivalently, we fix $\nu^- = \nu^+/\nu$. As a result, we obtain the following problem.

$$\hat{\gamma}(\nu^+) = \min_{\mathbf{d}^+, \mathbf{d}^-, \gamma} \gamma \quad (7)$$

sub.to

$$\begin{aligned} \sum_i d_i^+ h_k(\mathbf{x}_i^+)/2 - \sum_j d_j^- h_k(\mathbf{x}_j^-)/2 &\leq \gamma \quad (k = 1, \dots, N), \\ \mathbf{d}^+ \in \mathcal{P}_p, \mathbf{d}^- \in \mathcal{P}_n, \\ d_i^+ &\leq 1/\nu^+, \\ d_j^- &\leq 1/\nu^- = \nu^+/\nu. \end{aligned}$$

Note that if we optimize ν^+ , we obtain the minimum of problem (6), that is, $\min_{\nu^+} \hat{\gamma}(\nu^+) = \gamma^*$. Remember, however, that problem (6) is not convex w.r.t.

ν^+ (see Fig. 11 for an example). Therefore, it is not straightforward to obtain the optimum.

On the other hand, for any fixed choice of ν^+ and ν^- , we can guarantee that the solution of problem (7) has a certain margin for several pairs.

Theorem 3. *Given ν^+ and ν^- , the solution of problem (7) has a margin of at least γ^* for at least $pn - \nu^+n - \nu^-p + \nu^+\nu^-$ pairs.*

Proof. Using Lagrangian multipliers, it can be shown that the dual problem of (7) is as follows:

$$(\hat{\rho}, \hat{\alpha}, \hat{b}, \hat{\xi}^+, \hat{\xi}^-) = \arg \max_{\alpha \in \mathcal{P}_N, b, \xi^+, \xi^-} \rho - \frac{1}{2\nu^+} \sum_{i=1}^p \xi_i^+ - \frac{1}{2\nu^-} \sum_{j=1}^n \xi_j^- \quad (8)$$

subject to

$$\begin{aligned} \sum_{k=1}^N \alpha_k (h_k(\mathbf{x}_i^+) + b) &\geq \rho - \xi_i^+ \quad (i = 1, \dots, p), \\ - \sum_{k=1}^N \alpha_k h_k(\mathbf{x}_j^-) - b &\geq \rho - \xi_j^- \quad (j = 1, \dots, n), \\ \xi^+, \xi^- &\geq \mathbf{0}. \end{aligned}$$

By using the KKT conditions, $\hat{\xi}_i^+(\hat{d}_i^+ - 1/\nu^+) = 0$. Therefore, if $\hat{\xi}_i^+ > 0$ then $\hat{d}_i^+ = 1/\nu^+$. Similarly, if $\hat{\xi}_j^- > 0$ then $\hat{d}_j^- = 1/\nu^-$. Note that there are at most ν^+ instances such that $\hat{d}_i^+ = 1/\nu^+$. This implies that there are at most ν^+ instances whose corresponding $\hat{\xi}_i^+ > 0$. Again, similarly, there are at most ν^- instances with $\hat{\xi}_j^- > 0$. There are therefore, for at least $(p - \nu^+)(n - \nu^-)$ pairs, the margin of which is at least $\hat{\rho}$. Finally, by duality, $\hat{\rho} - (1/\nu^+) \sum_i \hat{\xi}_i^+ - (1/\nu^-) \sum_j \hat{\xi}_j^- = \hat{\gamma}$. Combined with the fact that $\hat{\gamma} \geq \gamma^*$, we have $\hat{\rho} \geq \gamma^*$, which completes the proof. \square

We note that problem (8) in the proof is the primal form of the dual problem (7). In particular, for the choice that $\nu = \varepsilon pn$, $\nu^+ = \sqrt{\varepsilon}p$ and $\nu^- = \sqrt{\varepsilon}n$, we obtain the following corollary.

Corollary 4. For $\nu = \varepsilon pn$, $\nu^+ = \sqrt{\varepsilon}p$ and $\nu^- = \sqrt{\varepsilon}n$, a solution of problem (7) has a margin of at least γ^* for $(1 - \sqrt{\varepsilon})^2 pn$ pairs.

Here, the lower bound ν/n of ν^+ is given so that the upper bound of d_j^- is at least $1/n$. Note that, the area under the tangent line of $d_i^+ = 1/\nu^+$ at $\nu^* = \nu_c^+$ is always included in the area $d_i^+ \leq 1/\nu^+$. Thus, any feasible solution of problem (9) is also a feasible solution of problem (6).

4.2 Practical Heuristics

Now we propose a practical method to find a local minimum of problem (6). Recall that in problem (6), we have non-convex constraints $d_i^+ \leq 1/\nu^+$ when we

regard ν^+ as a variable. In order to avoid non-convex constraints, we consider a tangent line of $1/\nu^+$ at some point $\nu^+ = \nu_c^+$. More precisely, we consider the following problem.

$$(\tilde{\gamma}, \tilde{\mathbf{d}}^+, \tilde{\mathbf{d}}^-, \tilde{\nu}^+) = \arg \min_{\gamma, \mathbf{d}^+, \mathbf{d}^-, \nu^+} \gamma \quad (9)$$

sub.to

$$\begin{aligned} & \sum_{i=1}^p d_i^+ h_k(\mathbf{x}_i^+)/2 - \sum_{j=1}^n d_j^- h_k(\mathbf{x}_j^-)/2 \leq \gamma \quad (k = 1, \dots, N), \\ & \mathbf{d}^+ \in \mathcal{P}_p, \mathbf{d}^- \in \mathcal{P}_n, \\ & d_i^+ \leq -\frac{1}{(\nu_c^+)^2} \nu^+ + \frac{2}{\nu_c^+} \quad (i = 1, \dots, p), \\ & d_j^- \leq \frac{\nu^+}{\nu} \quad (j = 1, \dots, n), \\ & \frac{\nu}{n} \leq \nu^+ \leq -\frac{(\nu_c^+)^2}{p} + 2\nu_c^+. \end{aligned}$$

Here the lower bound ν/n of ν^+ is added so that the upper bound of d_j^- is at least $1/n$. Also, the upper bound of ν^+ is given so that the upper bound of d_i^+ is at least $1/p$. Note that, the region under the tangent line of $d_i^+ = 1/\nu^+$ at $\nu^* = \nu_c^+$ is always contained in the region $d_i^+ \leq 1/\nu^+$. Thus, any feasible solution of problem (9) is also a feasible solution of problem (6).

Now we are ready to describe our heuristics:

1. Given some ν_c^+ , solve problem (9) and get a solution $(\tilde{\gamma}, \tilde{\mathbf{d}}^+, \tilde{\mathbf{d}}^-, \tilde{\nu}^+)$.
2. Given $\nu^+ = \tilde{\nu}^+$, solve problem (7) and get a solution $(\hat{\gamma}, \hat{\mathbf{d}}^+, \hat{\mathbf{d}}^-)$.

Observe that the solution $(\tilde{\gamma}, \tilde{\mathbf{d}}^+, \tilde{\mathbf{d}}^-)$ of problem (9) is a feasible solution of problem (7) given $\nu^+ = \tilde{\nu}^+$. Thus, we have $\hat{\gamma} \leq \tilde{\gamma}$. Furthermore, if we set $\nu_c^+ = \hat{\nu}^+$, the solution $(\hat{\gamma}, \hat{\mathbf{d}}^+, \hat{\mathbf{d}}^-, \hat{\nu}^+)$ is a feasible solution of problem (9), so

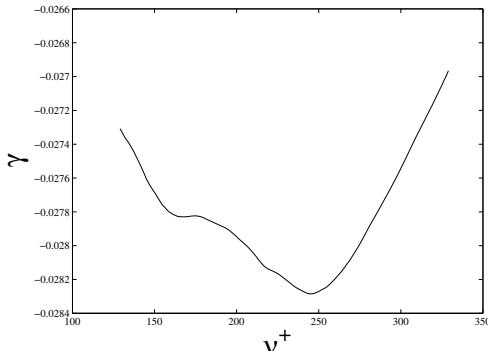


Fig. 1. Illustration of the function $\hat{\gamma}(\nu^+)$ for an artificial data set

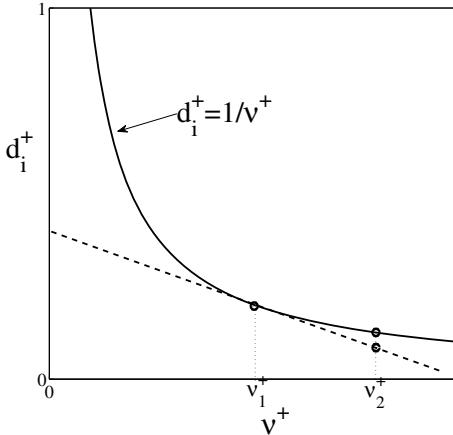


Fig. 2. Illustration of the heuristics. Here $v_1^+ = \nu_c^+$ and ν_2^+ is the solution of problem (9) given ν_c^+ .

that the minimum $\tilde{\gamma}'$ of problem (9) satisfies $\tilde{\gamma}' \leq \hat{\gamma}$. Therefore, by repeating this procedure, we can obtain a monotonically decreasing sequence of γ , which will converge to a local minimum of problem (6). In an algorithmic perspective, the second step that solves problem (7) seems redundant. However, we add the second step for numerical stability since problem (7) has simpler constraints. Fig. 2 illustrates the heuristics.

5 Experiments

In this section, we present preliminary experimental results. The data sets include artificial data sets, and real data sets from the UCI Machine Learning Repository and Reuters.

5.1 Artificial Data

For the first experiment, we used artificial data sets with r -of- k threshold functions as target functions. An r -of- k threshold function f over N Boolean variables is associated with some set A of k Boolean variables and f outputs $+1$ if at least r of the k variables in A are positive and f outputs -1 , otherwise. Assume that the instance space is $\{+1, -1\}^N$. In other words, the r -of- k threshold function f is represented as follows

$$f(\mathbf{x}) = \text{sign}\left(\sum_{x \in A} x + k - 2r + 1\right).$$

For $N = 100$, $k = 30$, and $r = 1, 8, 15$, we fix r -of- k threshold functions which determine labels. Then for each set of parameters, we generate $m = 1000$ random instances so that ratios of positive and negative instances are $5 : 5$, $7 : 3$, and

9 : 1 respectively. Finally, we add random noise into labels by changing the label of each instance with probabilities of 5%, 10%, and 15%. As hypotheses, we use N Boolean variables themselves and the constant hypothesis which always outputs +1.

We compare RankBoost [8], SoftRankBoost [15], 1-norm soft margin over pairs (LP-Pair), and our method. For RankBoost, we set the number of iterations to be $T = 500, 5000$, and 10000, respectively. For the other methods, we set the parameter $\nu = \varepsilon pn$, where $\varepsilon \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$. We evaluate each method by 5-fold cross validation. As shown in Table 1, our method recorded higher AUCs than the other algorithm for almost the data sets. In addition, in Table 2, our method achieves especially high AUCs, which are greater than or equal to those of LP-Pair.

5.2 UCI Data

For the next experiment, we use data sets “hypothyroid”, “ionosphere”, “kr-vs-kp”, “sick-euthroid”, “spambase” from the UCI Machine Learning Repository [1]. The parameters of each algorithm are the same as in Section 5.1. As shown in Table 3, our method archives high AUCs for all data sets.

Table 1. AUCs for artificial data sets

data		RankBoost			SoftRankBoost	LP-Pair	our method
r	niose	500	5000	10000			
1	5(%)	0.9313	0.9384	0.9378		0.7275	0.9745
8		0.9251	0.9239	0.9239		0.9325	0.9564
15		0.9264	0.9262	0.9262		0.9401	0.952
1	10(%)	0.8777	0.8979	0.8979		0.7391	0.9125
8		0.8857	0.8853	0.8853		0.9043	0.9136
15		0.8727	0.8727	0.8727		0.869	0.9043
1	15(%)	0.8102	0.8389	0.8391		0.7442	0.8322
8		0.8371	0.8372	0.8372		0.8793	0.8608
15		0.8377	0.8337	0.8337		0.856	0.857

Table 2. AUCs for artificial data sets with random noises 5%, 10%, and 15%

data		RankBoost			SoftRankBoost	LP-Pair	our method
$p : n$	r	500	5000	10000			
7:3	1	0.9177	0.9182	0.9179		0.7661	0.9472
	8	0.9018	0.9015	0.9015		0.9318	0.9292
	15	0.8959	0.8956	0.8956		0.9353	0.9294
9:1	1	0.7695	0.7742	0.7738		0.7735	0.7924
	8	0.7736	0.7736	0.7736		0.7718	0.7818
	15	0.7247	0.7247	0.7247		0.8266	0.7426

Table 3. AUCs for UCI data sets, when N , p , and n stand for the dimension, the number of positive and negative instances of each data sets, respectively

	data			RankBoost			SoftRankBoost	LP-Pair	our method	
	N	p	n	1000	5000	10000				
hypothyroid	43	151	3012	0.9488	0.9468	0.9468		0.96	0.9511	1.0
ionosphere	34	225	126	0.9327	0.9253	0.9253		0.9917	0.9768	0.9865
kr-vs-kp	73	1669	1527	0.8712	0.8721	0.8721		0.9085	1.0	0.9276
sick-euthroid	43	293	2870	0.7727	0.8706	0.8706		0.7847	1.0	1.0
spambase	57	1813	2788	0.8721	0.7735	0.735		0.9359	1.0	1.0

5.3 Reuters Data

Reuters data sets are data of Reuters news (Reuters-21588²), which are 10710 articles labeled by topics. We choose 5 major topics and consider 5 binary classification problems whose objective is to classify if a given article belongs to the topic. We prepare 30838 base classifiers which are decision stumps associated with words. More precisely, each base classifier answers 1 if the given article contains the associated word and answers 0, otherwise. The results are summarized in Table 4. For Reuters data sets, our method shows better performance than RankBoost, but SoftRankBoost shows better AUCs for some topics (3 out of 5).

Table 4. AUCs for Reuters data sets, where p , and n stand for the number of positive and negative instances included in each of data sets

topics	data			RankBoost			SoftRankBoost	our method	
	p	n		1000	5000	10000			
acq	2327	8383		0.9296	0.9347	0.9347		0.9363	0.9388
crude	592	10118		0.9133	0.9188	0.9203		0.9944	0.9329
earn	3802	6908		0.9567	0.9568	0.9566		0.9952	0.9652
money-fx	743	9967		0.9375	0.9335	0.9318		0.9608	0.9479
trade	529	10181		0.9290	0.9301	0.9291		0.9281	0.9450

5.4 Computation Time

Finally, we examine the computation time of LP-Pair and our method. We use a machine with four Intel Xeon 5570 2.93-GHz cores and a memory of 32 GByte. We use the artificial data that are used in Section 5.1, $N = 100$, $k = 10$, $r = 3$. The sizes of the data sets are $m = 100, 500, 1000, 1500$, respectively. The ratio of positive and negative instances is 5 : 5, and we add random noise of 5%. We set $\varepsilon = 0.2$ for both LP-Pair and our method and evaluate each execution time by 5-fold cross validation. As is shown in Table 5, clearly our method is clearly faster than LP-Pair.

² <http://www.daviddlewis.com/resources/testcollections/reuters21578>

Table 5. Computation time(sec.)

m	LP-Pair	our method
100	0.102	0.11
500	24.51	0.514
1000	256.78	0.86
1500	1353	1.76

6 Conclusion and Future Work

In this paper, we have formulated AUC maximization as hard/soft margin optimization problems over pairs of positive and negative instances. In the hard margin case, we showed that the original problem over pairs is equivalent to the 1-norm soft margin problem over $p + n$ instances with the bias term. In the soft margin case, we proposed a reduction method for the 1-norm soft margin optimization problem over instances, which is generally non-convex. Our reduction method is guaranteed to obtain a certain amount of margin over pairs of instances. Moreover, we have proposed heuristics that obtains more appropriate parameters. We have tested this method for artificial and real data. In comparison with other methods, our method achieved high AUCs in the experiments.

In the future, we intend to examine our methods for additional data sets including very large data sets. In addition, we would like to investigate why our method and SoftRankBoost sometimes achieve higher AUCs than the 1-norm soft margin over pairs.

Acknowledgements. This work was conducted in part while the second author was visiting UC Santa Cruz. The authors would like to Manfred Warmuth for the stimulating discussion regarding the initial results. We would also like to Masayuki Takeda for his financial support. Finally, we would like to anonymous reviewers for helpful comments. This work was partly supported by JSPS KAKENHI 23700178 and KAKENHI (B) 23300003.

References

- [1] Asuncion, A., Newman, D.J.: UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences (2007), <http://mlearn.ics.uci.edu/MLRepository.html>
- [2] Balcan, N., Bansal, N., Beygelzimer, A., Coppersmith, D., Langford, J., Sorkin, G.B.: Robust reductions from ranking to classification. In: Bshouty, N.H., Gentile, C. (eds.) COLT. LNCS (LNAI), vol. 4539, pp. 604–619. Springer, Heidelberg (2007)
- [3] Bradley, J.K., Shapire, R.: Filterboost: Regression and classification on large datasets. In: Advances in Neural Information Processing Systems, vol. 20, pp. 185–192 (2008)
- [4] Brefeld, U., Scheffer, T.: Auc maximizing support vector learning. In: Proceedings of the ICML Workshop on ROC Analysis in Machine Learning (2005)

- [5] Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. *Journal of Artificial Intelligence Research* 10, 243–279 (1999)
- [6] Cortes, C., Mohri, M.: AUC optimization vs. error rate minimization. In: *Advances in Neural Information Processing Systems*, vol. 16 (2004)
- [7] Domingo, C., Watanabe, O.: MadaBoost: A modification of AdaBoost. In: *Proceedings of 13th Annual Conference on Computational Learning Theory (COLT 2000)*, pp. 180–189 (2000)
- [8] Freund, Y., Iyer, R., Shapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research* 4, 933–969 (2003)
- [9] Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55(1), 119–139 (1997)
- [10] Fung, G., Rosales, R., Krishnapuram, B.: Learning rankings via convex hull separation. In: *Advances in Neural Information Processing Systems (NIPS 2005)*, vol. 18 (2005)
- [11] Gavinsky, D.: Optimally-smooth adaptive boosting and application to agnostic learning. *Journal of Machine Learning Research* (2003)
- [12] Hatano, K.: Smooth boosting using an information-based criterion. In: Balcázar, J.L., Long, P.M., Stephan, F. (eds.) *ALT 2006. LNCS (LNAI)*, vol. 4264, pp. 304–318. Springer, Heidelberg (2006)
- [13] Joachims, T.: Optimizing search engines using clickthrough data. In: *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2002)
- [14] Long, P.M., Servedio, R.A.: Boosting the area under the roc curve. In: *Advances in Neural Information Processing Systems*, vol. 20 (2008)
- [15] Moribe, J., Hatano, K., Takimoto, E., Takeda, M.: Smooth boosting for margin-based ranking. In: Freund, Y., Györfi, L., Turán, G., Zeugmann, T. (eds.) *ALT 2008. LNCS (LNAI)*, vol. 5254, pp. 227–239. Springer, Heidelberg (2008)
- [16] Rätsch, G.: Robust Boosting via Convex Optimization: Theory and Applications. PhD thesis, University of Potsdam (2001)
- [17] Rätsch, G., Warmuth, M.K.: Efficient margin maximizing with boosting. *Journal of Machine Learning Research* 6, 2131–2152 (2005)
- [18] Raykar, V.C., Duraiswami, R., Krishnapuram, B.: A fast algorithm for learning a ranking function from large-scale data sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30(7), 1158–1170 (2008)
- [19] Rudin, C.: Ranking with a P-norm push. In: Lugosi, G., Simon, H.U. (eds.) *COLT 2006. LNCS (LNAI)*, vol. 4005, pp. 589–604. Springer, Heidelberg (2006)
- [20] Rudin, C.: The P-Norm Push: A simple convex ranking algorithm that concentrates at the top of the list. *Journal of Machine Learning Research* 10, 2233–2271 (2009)
- [21] Rudin, C., Schapire, R.E.: Margin-based ranking and an equivalence between AdaBoost and RankBoost. *Journal of Machine Learning Research* 10, 2193–2232 (2009)
- [22] Schölkopf, B., Smola, A.J., Williamson, R.C., Bartlett, P.L.: New support vector algorithms. *Neural Computation* 12(5), 1207–1245 (2000)
- [23] Servedio, R.A.: Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research* 4, 633–648 (2003)
- [24] Warmuth, M., Glocer, K., Rätsch, G.: Boosting algorithms for maximizing the soft margin. In: *Advances in Neural Information Processing Systems*, vol. 20, pp. 1585–1592 (2008)

Axioms for Rational Reinforcement Learning

Peter Sunehag and Marcus Hutter

Research School of Computer Science

Australian National University

Canberra, ACT, 0200, Australia

{Peter.Sunehag,Marcus.Hutter}@anu.edu.au

Abstract. We provide a formal, simple and intuitive theory of rational decision making including sequential decisions that affect the environment. The theory has a geometric flavor, which makes the arguments easy to visualize and understand. Our theory is for complete decision makers, which means that they have a complete set of preferences. Our main result shows that a complete rational decision maker implicitly has a probabilistic model of the environment. We have a countable version of this result that brings light on the issue of countable vs finite additivity by showing how it depends on the geometry of the space which we have preferences over. This is achieved through fruitfully connecting rationality with the Hahn-Banach Theorem. The theory presented here can be viewed as a formalization and extension of the betting odds approach to probability of Ramsey and De Finetti [Ram31, deF37].

Keywords: Rationality, Probability, Utility, Banach Space, Linear Functional.

1 Introduction

We study complete decision makers that can take a sequence of actions to rationally pursue any given task. We suppose that the task is described in a reinforcement learning framework where the agent takes actions and receives observations and rewards. The aim is to maximize total reward in some given sense.

Rationality is meant in the sense of internal consistency [Sug91], which is how it has been used in [NM44] and [Sav54]. In [NM44], it is proven that preferences together with rationality axioms and probabilities for possible events imply the existence of utility values for those events that explain the preferences as arising through maximizing expected utility. Their rationality axioms are

1. Completeness: Given any two choices we either prefer one of them to the other or we consider them to be equally preferable;
2. Transitivity: A preferable to B and B to C imply A preferable to C;
3. Independence: If A is preferable to B and $t \in [0, 1]$ then $tA + (1 - t)C$ is preferable (or equal) to $tB + (1 - t)C$;
4. Continuity: If A is preferable to B and B to C then there exists $t \in [0, 1]$ such that B is equally preferable to $tA + (1 - t)C$.

In [Sav54] the probabilities are not given but it is instead proven that preferences together with rationality axioms imply the existence of probabilities and utilities. We are here interested in the case where one is given utility (rewards) and preferences over actions and then deriving the existence of a probabilistic world model. We put an emphasis on extensions to sequential decision making with respect to a countable class of environments. We set up simple axioms for a rational decision maker, which implies that the decisions can be explained (or defined) from probabilistic beliefs.

The theory of [Sav54] is called subjective expected utility theory (SEUT) and was intended to provide statistics with a strictly behavioral foundation. The behavioral approach stands in stark contrast to approaches that directly postulate axioms that “degrees of belief” should satisfy [Cox46, Hah99, Jay03]. Cox’s approach [Cox46, Jay03] has also been found [Par94] to need additional technical assumptions in addition to the common sense axioms originally listed by Cox. The original proof by [Cox46] has been exposed as not mathematically rigorous and his theorem as wrong [Hah99]. An alternative approach by [Ram31, deF37] is interpreting probabilities as fair betting odds.

The theory of [Sav54] has greatly influenced economics [Sug91] where it has been used as a description of rational agents. Seemingly strange behavior was explained as having beliefs (probabilities) and tastes (utilities) that were different from those of the person to whom it looked irrational. This has turned out to be insufficient as a description of human behavior [All53, Ell61] and it is better suited as a normative theory or design principle in artificial intelligence. In this article, we are interested in studying the necessity for rational agents (biological or not) to have a probabilistic model of their environment. To achieve this, and to have as simple common sense axioms of rationality as possible, we postulate that given any set of values (a contract) associated with the possible events, the decision maker needs to have an opinion on whether he prefers these values to a guaranteed zero outcome or not (or equal). From this setting and our other rationality axioms we deduce the existence of probabilities that explain all preferences as maximizing expected value. There is an intuitive similarity to the idea of explaining/deriving probabilities as a bookmaker’s betting odds as done in [deF37] and [Ram31]. One can argue that the theory presented here (in Section 2) is a formalization and extension of the betting odds approach. Geometrically, the result says that there is a hyper-plane in the space of contracts that separates accept from reject. We generalize this statement, by using the Hahn-Banach Theorem, to the countable case where the set of hyper-planes (the dual space) depends on the space of contract. The answers for different cases can then be found in the Banach space theory literature. This provides a new approach to understanding issues like finite vs. countable additivity. We take advantage of this to formulate rational agents that can deal successfully with countable (possibly universal as in all computable environments) classes of environments.

Our presentation begins in Section 2 by first looking at a fundamental case where one has to accept or reject certain contracts defining positive and negative

rewards that depend on the outcome of an event with finitely many possibilities. To draw the conclusion that there are implicit unique probabilistic beliefs, it is important that the decision maker has an opinion (acceptable, rejectable or both) on every possible contract. This is what we mean when we say *complete decision maker*.

In a more general setting, we consider sequential decision making where given any contract on the sequence of observations and actions, the decision maker must be able to choose a policy (i.e. an action tree). Note that the actions may affect the environment. A contract on such a sequence can e.g. be viewed as describing a reward structure for a task. An example of a task is a cleaning robot that gets positive rewards for collecting dust and negative for falling down the stairs. A prerequisite for being able to continue to collect dust can be to recharge the battery before running out. A specialized decision maker that deals only with one contract/task does not always need to have implicit probabilities, it can suffice with qualitative beliefs to take reasonable decisions. A qualitative belief can be that one pizza delivery company (e.g. Pizza Hut vs Dominos) is more likely to arrive on time than the other. If one believes the pizzas are equally good and the price is the same, we will chose the company we believe is more often delivering on time. Considering all contracts (reward structures) on the actions and events, leads to a situation where having a way of making rational (coherent) decisions, implies that the decision maker has implicit probabilistic beliefs. We say that the probabilities are implicit because the decision maker, which might e.g. be a human, a dog, a computer or just a set of rules, might have a non-probabilistic description of how the decisions are made.

In Section 3, we investigate extensions to the case with countably many possible outcomes and the interesting issue of countable versus finite additivity. Savage's axioms are known to only lead to finite additivity while [Arr70] showed that adding a monotone continuity assumption guarantees countable additivity. We find that in our setting, it depends on the space of contracts in an interesting way. In Section 4, we discuss a setting where we have a class of environments.

2 Rational Decisions for Accepting or Rejecting Contracts

We consider a setting where we observe a symbol (letter) from a finite alphabet and we are offered a form of bet we call a contract that we can accept or not.

Definition 1 (Passive Environment, Event). *A passive environment is a sequence of symbols (letters) j_t , called events, being presented one at a time. At time t the symbols j_1, \dots, j_t are available. We can equivalently say that a passive environment is a function ν from finite strings to $\{0, 1\}$ where $\nu(j_1, \dots, j_t) = 1$ if and only if the environment begins with j_1, \dots, j_t .*

Definition 2 (Contract). *Suppose that we have a passive environment with symbols from an alphabet with m elements. A contract for an event is an element $x = (x_1, \dots, x_m)$ in \mathbb{R}^m and x_j is the reward received if the event is the j :th symbol, under the assumption that the contract is accepted (see next definition).*

Definition 3 (Decision Maker, Decision). A decision maker (for some unknown environment) is a set $Z \subset \mathbb{R}^m$ which defines exactly the contracts that are acceptable. In other words, a decision maker is a function from \mathbb{R}^m to $\{\text{accepted}, \text{rejected}, \text{either}\}$. The function value is called the decision.

If $x \in Z$ and $\lambda \geq 0$ then we want $\lambda x \in Z$ since it is simply a multiple of the same contract. We also want the sum of two acceptable contracts to be acceptable. If we cannot lose money we are prepared to accept the contract. If we are guaranteed to win money we are not prepared to reject it. We summarize these properties in the definition below of a rational decision maker.

Definition 4 (Rationality I). We say that the decision maker ($Z \subset \mathbb{R}^m$) is rational if

1. Every contract $x \in \mathbb{R}^m$ is either acceptable or rejectable or both;
2. x is acceptable if and only if $-x$ is rejectable;
3. $x, y \in Z, \lambda, \gamma \geq 0$ then $\lambda x + \gamma y \in Z$;
4. If $x_k \geq 0 \forall k$ then $x = (x_1, \dots, x_m) \in Z$ while if $x_k < 0 \forall k$ then $x \notin Z$.

If we want to compare these axioms to rationality axioms for a preference relation on contracts we will say that x is better or equal (as in equally good) than y if $x - y$ is acceptable while it is worse or equal if $x - y$ is rejectable. The first axiom is completeness. The second says that if x is better or equal than y then y is worse or equal to x . The third implies transitivity since $(x - y) + (y - z) = (x - z)$. The fourth says that if x has a better (or equal) reward than y for any event, then x is better (or equal) than y .

2.1 Probabilities and Expectations

Theorem 5 (Existence of Probabilities). Given a rational decision maker, there are numbers $p_i \geq 0$ that satisfy

$$\{x \mid \sum x_i p_i > 0\} \subset Z \subseteq \{x \mid \sum x_i p_i \geq 0\}. \quad (1)$$

Assuming $\sum_i p_i = 1$ makes the numbers unique and we will use the notation $Pr(i) = p_i$.

Proof. See the proof of the more general Theorem 23. It tells us that the closure \bar{Z} of Z is a closed half space and can be written as $\{x \mid \sum x_i p_i \geq 0\}$ for some vector $p = (p_i)$ (since every linear functional on \mathbb{R}^m is of the form $f(x) = \sum x_i p_i$) and not every p_i is 0. The fourth property tells us that $p_i \geq 0 \forall i$.

Definition 6 (Expectation). We will refer to the function $g(x) = \sum p_i x_i$ from (II) as the decision makers expectation. In this terminology, a rational decision maker has an expectation function and accepts a contract x if $g(x) > 0$ and reject it if $g(x) < 0$.

Remark 7. Suppose that we have a contract $x = (x_i)$ where $x_i = 1$ for all i . If we want $g(x) = 1$, we need $\sum p_i = 1$.

We will write $E(x)$ instead of $g(x)$ (assuming $\sum p_i = 1$) from now on and call it the expected value or expectation of x .

2.2 Multiple Events

Suppose that the contract is such that we can view the symbol to be drawn as consisting of two (or several) symbols from smaller alphabets. That is we can write a drawn symbol as (i, j) where all the possibilities can be found through $1 \leq i \leq m$, $1 \leq j \leq n$. In this way of writing, a contract is defined by real numbers $x_{i,j}$. Theorem 5 tells us that for a rational decision maker there exists unique $r_{i,j} \geq 0$ such that $\sum_{i,j} r_{i,j} = 1$ and an expectation function $g(x) = \sum r_{i,j} x_{i,j}$ such that contracts are accepted if $g(x) > 0$ and rejected if $g(x) < 0$.

2.3 Marginals

Suppose that we can take rational decisions on bets for a pair of horse races, while the person that offers us bets only cares about the first race. Then we are still equipped to respond since the bets that only depend on the first race is a subset of all bets on the pair of races.

Definition 8 (Marginals). Suppose that we have a rational decision maker (Z) for contracts on the events (i, j) . Then we say that the marginal decision maker for the first symbol (Z_1) is the restriction of the decision maker Z to the contracts $x_{i,j}$ that only depend on i , i.e. $x_{i,j} = x_i$. In other words given a contract $y = (y_i)$ on the first event, we extend that contract to a contract on (i, j) by letting $y_{i,j} = y_i$ and then the original decision maker can decide.

Suppose that $x_{i,j} = x_i$. Then the expectation $\sum r_{i,j} x_{i,j}$ can be rewritten as $\sum p_i x_i$ where $p_i = \sum_j r_{i,j}$. We write that

$$\Pr(i) = \sum_j \Pr(i, j).$$

These are the marginal probabilities for the first variable that describe the marginal decision maker for that variable. Naturally we can also define a marginal for the second variable (considering contracts $x_{i,j} = x_j$) by letting $q_j = \sum_i r_{i,j}$ and $\Pr(j) = \sum_i \Pr(i, j)$. The marginals define sets $Z_1 \subset \mathbb{R}^m$ and $Z_2 \subset \mathbb{R}^n$ of acceptable contracts on the first and second variables separately.

2.4 Conditioning

Again suppose that we are taking decisions on bets for a pair of horse races, but this time suppose that the first race is already over and we know the result. We are still equipped to respond to bets on the second race by extending the bet to a bet on both where there is no reward for (pairs of) events that are inconsistent with what we know.

Definition 9 (Conditioning). Suppose that we have a rational decision maker (Z) for contracts on the events (i, j) . We define the conditional decision maker $Z_{j=j_0}$ for i given $j = j_0$ by restricting the original decision maker Z to contracts $x_{i,j}$ which are such that $x_{i,j} = 0$ if $j \neq j_0$. In other words if we start with a contract $y = (y_i)$ on i we extend it to a contract on (i, j) by letting $y_{i,j_0} = y_i$ and $y_{i,j} = 0$ if $j \neq j_0$. Then the original decision maker can make a decision for that contract.

Suppose that $x_{i,j} = 0$ if $j \neq j_0$. The unconditional expectation of this contract is $\sum_{i,j} r_{i,j} x_{i,j}$ as usual which equals $\sum_i r_{i,j_0} x_{i,j_0}$. This leads to the same decisions (i.e. the same Z) as using $\sum_i \frac{r_{i,j_0}}{\sum_k r_{k,j_0}} x_{i,j_0}$ which is of the form in Theorem 5. We write that

$$Pr(i|j_0) = \frac{Pr(i, j_0)}{\sum_k Pr(k, j_0)} = \frac{Pr(i, j_0)}{Pr(j_0)}. \quad (2)$$

From this it follows that

$$Pr(i_0)Pr(j_0|i_0) = Pr(j_0)Pr(i_0|j_0) \quad (3)$$

which is one way of writing Bayes rule.

2.5 Learning

In the previous section we defined conditioning which lead us to a definition of what it means to learn. Given that we have probabilities for events that are sequences of a certain number of symbols and we have observed one or several of them, we use conditioning to determine what our belief regarding the remaining symbols should be.

Definition 10 (Learning). Given a rational decision maker, defined by p_{i_1, \dots, i_T} for the events $(i_t)_{t=1}^T$ and the first $t - 1$ symbols i_1, \dots, i_{t-1} , we define the informed rational decision maker for i_t by conditioning on the past i_1, \dots, i_{t-1} and marginalize over the future i_{t+1}, \dots, i_T . Formally,

$$P_{i_t}^{informed}(i) = Pr(i|i_1, \dots, i_t) = \frac{\sum_{j_{t+1}, \dots, j_T} p_{i_1, \dots, i_t, j_{t+1}, \dots, j_T}}{\sum_{j_t, \dots, j_T} p_{i_1, \dots, i_{t-1}, j_t, \dots, j_T}}.$$

2.6 Choosing between Contracts

Definition 11 (Choosing contract). We say that to rationally prefer contract x over y is (equivalent) to rationally consider $x - y$ to be acceptable.

As before we assume that we have a decision maker that takes rational decisions on accepting or rejecting contracts x that are based on an event that will be observed. Hence there exist implicit probabilities that represent all choices and an expectation function. Suppose that an agent has to choose between action a_1

that leads to receiving reward x_i if i is drawn and action a_2 that leads to receiving y_i in the case of seeing i . Let $z_i = x_i - y_i$. We can now go back to choosing between accepting and rejecting a contract by saying that choosing (preferring) a_1 over a_2 means accepting the contract z . In other words if $E(x) > E(y)$ choose a_1 and if $E(x) < E(y)$ choose a_2 .

Remark 12. We note that if we postulate that choosing between contract x and the zero contract is the same as choosing between accepting or rejecting x , then being able to choose between contracts implies the ability to choose between accepting and rejecting one contract. We, therefore, can say that the ability to choose between a pair of contracts is equivalent to the ability to choose to accept or reject a single contract.

We can also choose between several contracts. Suppose that action a_k gives us the contract $x^k = (x_i^k)_{i=1}^m$. If $E(x^j) > E(x^k) \forall k \neq j$ then we strictly prefer a_j over all other actions. In other words a contract $x^j - x^k$ would for all k be accepted and not rejected by a rational decision maker.

Remark 13. If we have a rational decision maker for accepting or rejecting contracts, then there are implicitly probabilities p_i for symbol i that characterize the decisions. A rational choice between actions a_k leading to contracts x^k is taken by choosing action

$$a^* = \operatorname{argmax}_k \sum_i p_i x_i^k. \quad (4)$$

2.7 Choosing between Environments

In this section, we assume that the event that the contracts are concerned with might be affected by the choice of action.

Definition 14 (Reactive environment). *An environment is a tree with symbols j_t (percepts) on the nodes and actions a_t on the edges. We provide the environment with an action a_t at each time t and it presents the symbol j_t at the node we arrive at by following the edge chosen by the action. We can also equivalently say that a reactive environment ν is a function from strings $a_1 j_1, \dots, a_t j_t$ to $\{0, 1\}$ which equals 1 if and only if ν would produce j_1, \dots, j_t given the actions a_1, \dots, a_t .*

We will define the concept of a decision maker for the case where one decision will be taken in a situation where not only the contract, but also the outcome can depend on the choice. We do this by defining the choice as being between two different environments.

Definition 15 (Active decision maker). *Consider a choice between having contract x for passive environment env_1 or contract y for passive environment env_2 . A decision maker is a set $Z \subset \mathbb{R}^{m_1} \times \mathbb{R}^{m_2}$ which defines exactly the pairs (x, y) for which we choose env_1 with x over env_2 with y .*

Definition 16 (Rational active choice). To choose between action a_1 with contract x and a_2 with contract y in a situation where the action may affect the event, we consider two separate environments, namely the environments that result from the two different actions. We would then have a situation where we will have one observation from each environment. Preferring a_1 with x to a_2 with y is (equivalent) to consider $x - y$ to be an acceptable contract for the pair of events.

Remark 17. Definition 16 means that a_1 with x is preferred over a_2 with y if a_1 with $x - y$ is preferred over a_2 with the zero contract.

Proposition 18 (Probabilities for reactive setting). Suppose that we have a reactive environment and a rational active decision maker that will make one choice between action a_1 and a_2 as described in Definitions 15 and 16, then there exist $p_i \geq 0$ and $q_i \geq 0$ such that action a_1 with contract x is preferred over action a_2 with contract y if $\sum p_i x_i > \sum q_i y_i$ and the reverse if $\sum p_i x_i < \sum q_i y_i$. This means that the decision maker acts according to probabilities $Pr(\cdot|a_1)$ and $Pr(\cdot|a_2)$.

Proof. Let \tilde{Z} be all contracts that when combined with action a_1 is preferred over a_2 with the zero contract. Theorem 11 guarantees the existence of p_i such that $\sum p_i x_i > 0$ implies that $x \in \tilde{Z}$ and $\sum p_i x_i < 0$ implies that $x \notin \tilde{Z}$. The same way we find q_i that describe when we prefer a_2 with y to a_1 with the zero contract. That these probabilities (p_i and q_i) explain the full decision maker as stated in the proposition now follows directly from Definition 16 understood as in Remark 17.

Suppose that we are going to make a sequence of $T < \infty$ decisions where at every point of time we will have a finite number of actions to chose between. We will consider contracts, which can pay out some reward at each time step and that can depend on everything (actions chosen and symbols observed) that has happened up until this time and we want to maximize the accumulated reward at time T .

We can view the choice as just making one choice, namely choosing an action tree. We will sometimes call an action tree a policy.

Definition 19 (Action tree). An action tree is a function from histories of symbols j_1, \dots, j_t and decisions a_1, \dots, a_{t-1} to new decisions, given that the decisions were made according to the function. Formally,

$$f(a_1, j_1, \dots, a_{t-1}, j_{t-1}) = a_t.$$

An action tree will assign exactly one action for any of the circumstances that one can end up in. That is, given the history up to any time $t < T$ of actions and events, we have a chosen action. We can, therefore, choose an action tree at time 0 and receive a total accumulated reward at time T . This brings us back to the situation of one event and one rational choice.

Definition 20 (Sequential decisions). Given a rational decision maker for the events $(j_t)_{t=1}^T$ and the first $t-1$ symbols j_1, \dots, j_{t-1} and decisions a_1, \dots, a_{t-1} , we define the informed rational decision maker at time t by conditioning on the past $a_1, j_1, \dots, a_{t-1}, j_{t-1}$.

Proposition 21 (Beliefs for sequential decisions). Suppose that we have a reactive environment and a rational decision maker that will take $T < \infty$ decisions. Furthermore, suppose that the decisions $0 \leq t < T$ have been taken and resulted in history $a_1, j_1, \dots, a_{t-1}, j_{t-1}$. Then the decision makers preferences at this time can be explained (through expected utility maximization) by probabilities

$$\Pr(j_t, \dots, j_T | a_1, j_1, \dots, a_{t-1}, j_{t-1}, a_t, a_{t+1}, \dots, a_T).$$

Proof. Definition 20 and Proposition 18 immediately lead us to the conclusion that given a past up to a point $t-1$ and a policy for the time t to T we have probabilistic beliefs over the possible future sequences from time t to T and the choice is categorized by maximizing expected accumulated reward at time T .

3 Countable Sets of Events

Instead of a finite set of possible outcomes, we will in this section assume a countable set. We suppose that the set of contracts is a vector space of sequences $x_k, k = 0, 1, 2, \dots$ where we use pointwise addition and multiplication with scalar. We will define a space by choosing a norm and let the space consist of the sequences that have finite norm as is common in Banach space theory. If the norm makes the space complete it is called a Banach sequence space [Die84]. Interesting examples are ℓ^∞ of bounded sequences with the maximum norm $\|(\alpha_k)\|_\infty = \max |\alpha_k|$, c_0 of sequence that converges to 0 equipped with the same maximum norm and ℓ^p which for $1 \leq p < \infty$ is defined by the norm

$$\|(\alpha_k)\|_p = (\sum |\alpha_k|^p)^{1/p}.$$

For all of these spaces we can consider weighted versions ($w_k > 0$) where

$$\|(\alpha_k)\|_{p,w_k} = \|(\alpha_k w_k)\|_p.$$

This means that $\alpha \in \ell^p(w)$ iff $(\alpha_k w_k) \in \ell^p$, e.g. $\alpha \in \ell^\infty(w)$ iff $\sup_k |\alpha_k w_k| < \infty$. Given a Banach (sequence) space X we use X' to denote the dual space that consists of all continuous linear functionals $f : X \rightarrow \mathbb{R}$. It is well known that a linear functional on a Banach space is continuous if and only if it is bounded, i.e. that there is $C < \infty$ such that $\frac{|f(x)|}{\|x\|} \leq C \forall x \in X$. Equipping X' with the norm $\|f\| = \sup \frac{|f(x)|}{\|x\|}$ makes it into a Banach space. Some examples are $(\ell^1)' = \ell^\infty$, $c'_0 = \ell^1$ and for $1 < p < \infty$ we have that $(\ell^p)' = \ell^q$ where $1/p + 1/q = 1$. These identifications are all based on formulas of the form

$$f(x) = \sum x_i p_i$$

where the dual space is the space that (p_i) must lie in to make the functional both well defined and bounded. It is clear that $\ell^1 \subset (\ell^\infty)'$ but $(\ell^\infty)'$ also contains “stranger” objects.

The existence of these other objects can be deduced from the Hahn-Banach theorem (see e.g. [Kre89] or [NB97]) that says that if we have a linear function defined on a subspace $Y \in X$ and if it is bounded on Y then there is an extension to a bounded linear functional on X . If Y is dense in X the extension is unique but in general it is not. One can use this Theorem by first looking at the subspace of all sequences in ℓ^∞ that converge and let $f(\alpha) = \lim_{k \rightarrow \infty} \alpha_k$. The Hahn-Banach theorem guarantees the existence of extensions to bounded linear functionals that are defined on all of ℓ^∞ . These are called Banach limits. The space $(\ell^\infty)'$ can be identified with the so called ba space of bounded and finitely additive measures with the variation norm $\|\nu\| = |\nu|(A)$ where A is the underlying set. Note that ℓ^1 can be identified with the smaller space of countably additive bounded measures with the same norm. The Hahn-Banach Theorem has several equivalent forms. One of these identifies the hyper-planes with the bounded linear functionals [NB97].

Definition 22 (Rationality II). *Given a Banach sequence space X of contracts, we say that the decision maker (subset Z of X defining acceptable contracts) is rational if*

1. Every contract $x \in X$ is either acceptable or rejectable or both;
2. x is acceptable if and only if $-x$ is rejectable;
3. $x, y \in Z, \lambda, \gamma \geq 0$ then $\lambda x + \gamma y \in Z$;
4. If $x_k \geq 0 \forall k$ then $x = (x_k)$ is acceptable while if $x_k > 0 \forall k$ then x is not rejectable.

Theorem 23 (Linear separation). *Suppose that we have a space of contracts X that is a Banach sequence space. Given a rational decision maker there is a positive continuous linear functional $f : X \rightarrow \mathbb{R}$ such that*

$$\{x \mid f(x) > 0\} \subset Z \subseteq \{x \mid f(x) \geq 0\}. \quad (5)$$

Proof. The third property tells us that Z and $-Z$ are convex cones. The second and fourth property tells us that $Z \neq \mathbb{R}^m$. Suppose that there is a point x that lies in both the interior of Z and of $-Z$. Then the same is true for $-x$ according to the second property and for the origin. That a ball around the origin lies in Z means that $Z = \mathbb{R}^m$ which is not true. Thus the interiors of Z and $-Z$ are disjoint open convex sets and can, therefore, be separated by a hyperplane (according to the Hahn-Banach theorem) which goes through the origin (since according to the second and fourth property the origin is both acceptable and rejectable). The first two properties tell us that $Z \cup -Z = \mathbb{R}^m$. Given a separating hyperplane (between the interiors of Z and $-Z$), Z must contain everything on one side. This means that Z is a half space whose boundary is a hyperplane that goes through the origin and the closure \bar{Z} of Z is a closed half space and can be written as $\{x \mid f(x) \geq 0\}$ for some $f \in X'$. The fourth property tells us that f is positive.

- Corollary 24 (Additivity).** 1. If $X = c_0$ then a rational decision maker is described by a countably additive (probability) measure.
 2. If $X = \ell^\infty$ then a rational decision maker is described by a finitely additive (probability) measure.

It seems from Corollary 24 that we pay the price of losing countable additivity for expanding the space of contracts from c_0 to ℓ^∞ but we can expand the space even more by looking at $c_0(w)$ where $w_k \rightarrow 0$ which contains ℓ^∞ and X' is then $\ell^1((1/w_k))$. This means that we get countable additivity back but we instead have a restriction on how fast the probabilities p_k must tend to 0. Note that a bounded linear functional on c_0 can always be extended to a bounded linear functional on ℓ^∞ by the formula $f(x) = \sum p_i x_i$ but that is not the unique extension. Note also that every bounded linear functional on ℓ^∞ can be restricted to c_0 and there be represented as $f(x) = \sum p_i x_i$. Therefore, a rational decision maker on ℓ^∞ contracts has probabilistic beliefs (unless $p_i = 0 \forall i$), though it might also take asymptotic behavior of a contract into account. For example (and here $p_i = 0 \forall i$), the decision maker that makes decisions based on asymptotic averages $\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n x_i$ when they exist. That strategy can be extended to all of ℓ^∞ (a Banach limit). The following proposition will help us decide which decision maker on ℓ^∞ is described with countably additive probabilities.

Proposition 25. Suppose that $f \in (\ell^\infty)'$. For any $x \in \ell^\infty$, let $x_i^j = x_i$ if $i \leq j$ and $x_i^j = 0$ otherwise. If for any x ,

$$\lim_{j \rightarrow \infty} f(x^j) = f(x),$$

then f can be written as $f(x) = \sum p_i x_i$ where $p_i \geq 0$ and $\sum_{i=1}^\infty p_i < \infty$.

Proof. The restriction of f to c_0 gives us numbers $p_i \geq 0$ such that $\sum_{i=1}^\infty p_i < \infty$ and $f(x) = \sum p_i x_i$ for $x \in c_0$. This means that $f(x^j) = \sum_{i=1}^j p_i x_i$ for any $x \in \ell^\infty$ and $j < \infty$. Thus $\lim_{j \rightarrow \infty} f(x^j) = \sum_{i=1}^\infty p_i x_i$.

Definition 26 (Monotone decisions). We define the concept of a monotone decision maker in the following way. Suppose that for every $x \in \ell^\infty$ there is $N < \infty$ such that the decision is the same for all x^j , $j \geq N$ (See Proposition 25 for definition) as for x . Then we say that the decision maker is monotone.

Example 27. Let $f \in \ell^\infty$ be such that if $\lim \alpha_k \rightarrow L$ then $f(\alpha) = L$ (i.e. f is a Banach limit). Furthermore define a rational decision maker by letting the set of acceptable contracts be $Z = \{x \mid f(x) \geq 0\}$. Then $f(x^j) = 0$ (where we use notation from Proposition 25) for all $j < \infty$ and regardless of which x we define x^j from. Therefore, all sequences that are eventually zero are acceptable contracts. This means that this decision maker is not monotone since there are contracts that are not acceptable.

Theorem 28 (Monotone rationality). Given a monotone rational decision maker for ℓ^∞ contracts, there are $p_i \geq 0$ such that $\sum p_i < \infty$ and

$$\{x \mid \sum x_i p_i > 0\} \subset Z \subseteq \{x \mid \sum x_i p_i \geq 0\}. \quad (6)$$

Proof. According to Theorem 23 there is $f \in (\ell^\infty)'$ such that (the closure of Z) $\bar{Z} = \{x \mid f(x) \geq 0\}$. Let $p_i \geq 0$ be such that $\sum p_i < \infty$ and such that $f(x) = \sum x_i p_i$ for $x \in c_0$. Remember that x^j (notation as in Proposition 25) is always in c_0 . Suppose that there is x such that x is accepted but $\sum x_i p_i < 0$. This violate monotonicity since there exist $N < \infty$ such that $\sum_{i=1}^n x_i p_i < 0$ for all $n \geq N$ and, therefore, x^j is not accepted for $j \geq N$ but x is accepted. We conclude that if x is accepted then $\sum p_i x_i \geq 0$ and if $\sum p_i x_i > 0$ then x is accepted.

4 Rational Agents for Classes of Environments

We will here study agents that are designed to deal with a large range of situations. Given a class of environments we want to define agents that can learn to act well when placed in any of them, assuming it is at all possible.

Definition 29 (Universality for a class). *We say that a decision maker is universal for a class of environments \mathcal{M} if for any outcome sequence $a_1 j_1 a_2 j_2 \dots$ that given the actions would be produced by some environment in the class, there is $c > 0$ (depending on the sequence) such that the decision maker has probabilities that satisfy*

$$\Pr(j_1, \dots, j_t | a_1, \dots, a_t) \geq c \quad \forall t.$$

This is obviously true if the decision maker's probabilistic beliefs are a convex combination $\sum_{\nu \in \mathcal{M}} w_\nu \nu$, $w_\nu > 0$ and $\sum_\nu w_\nu = 1$.

We will next discuss how to define some large classes of environments and agents that can succeed for them. We assume that the total accumulated reward from the environment will be finite regardless of our actions since we want any policy to have finite utility. Furthermore, we assume that rewards are positive and that it is possible to achieve strictly positive rewards in any environment. We would like the agent to perform well regardless of which environment from the chosen class it is placed in.

For any possible policy (action tree) π and environment ν , there is a total reward V_ν^π that following π in ν would result in. This means that for any π there is a contract sequence $(V_\nu^\pi)_\nu$, assuming we have enumerated our set of environments. Let

$$V_\nu^* = \max_\pi V_\nu^\pi.$$

We know that $V_\nu^* > 0$ for all ν . Every contract sequence $(V_\nu^\pi)_\nu$ lies in $X = \ell^\infty((1/V_\nu^*))$ and $\|(V_\nu^\pi)\|_X \leq 1$. The rational decision makers are the positive, continuous linear functionals on X . X' contains the space $\ell^1(V_\nu^*)$. In other words if $w_\nu \geq 0$ and $\sum w_\nu V_\nu^* < \infty$ then the sequence (w_ν) defines a rational decision maker for the contract space X . These are exactly the monotone rational decision makers. Letting (which is the AIXI agent from [Hut05])

$$\pi^* \in \operatorname{argmax}_\pi \sum_\nu w_\nu V_\nu^\pi \tag{7}$$

we have a choice with the property that for any other π with

$$\sum_{\nu} w_{\nu} V_{\nu}^{\pi} < \sum_{\nu} w_{\nu} V_{\nu}^{\pi^*}.$$

Hence the contract $(V_{\nu}^{\pi^*} - V_{\nu}^{\pi})$ is not rejectable. In other words π^* is strictly preferable to π . By letting $p_{\nu} = w_{\nu} V_{\nu}^{\pi^*}$, we can rewrite (7) as

$$\pi^* \in \operatorname{argmax}_{\pi} \sum_{\nu} p_{\nu} \frac{V_{\nu}^{\pi}}{V_{\nu}^*}. \quad (8)$$

If one further restricts the class of environments by assuming $V_{\nu}^* \leq 1$ for all ν then for every π , $(V_{\nu}^{\pi}) \in \ell^{\infty}$. Therefore, by Theorem 28 the monotone rational agents for this setting can be formulated as in (7) with $(w_{\nu}) \in \ell_1$, i.e. $\sum_{\nu} w_{\nu} < \infty$. However, since $(p_{\nu}) \in \ell_1$, a formulation of the form of (8) is also possible. Normalizing p and w individually to probabilities makes (7) into a maximum expected utility criterion and (8) into maximum relative utility. As long as our w and p relate the way they do it is still the same decisions. If we would base both expectations on the same probabilistic beliefs it would be different criteria. When we have an upper bound $V_{\nu}^* < b < \infty \forall \nu$ we can always translate expected utility to expected relative utility in this way, while we need a lower bound $0 < a < V_{\nu}^*$ to rewrite an expected relative utility as an expected utility. Note, the different criteria will start to deviate from each other after updating the probabilistic beliefs.

4.1 Asymptotic Optimality

Denote a chosen countable class of environments by \mathcal{M} . Let $V_{\nu,k}^{\pi}$ be the rewards achieved after time k using policy π in environment ν . We suppress the dependence on the history so far. Let

$$W_{\nu,k}^{\pi} = \frac{V_{\nu,k}^{\pi}}{V_{\nu,k}^*}$$

denote the skill (relative reward) of π in environment ν from time k . The maximum possible skill is 1. We would like to have a policy π such that

$$\lim_{k \rightarrow \infty} W_{\nu,k}^{\pi} = 1 \quad \forall \nu \in \mathcal{M}.$$

This would mean that the agent asymptotically achieve maximum skill when placed in any environment from \mathcal{M} . Let $I(h_k, \nu) = 1$ if ν is consistent with history h_k and $I(h_k, \nu) = 0$ otherwise. Furthermore, let

$$p_{\nu,k} = \frac{p_{\nu,0}}{\sum_{\mu \in \mathcal{M}} p_{\mu,0} I(h_k, \mu)}$$

be the agent's weight for environment ν at time k and let π^p be a policy that at time k acts according to a policy in

$$\operatorname{argmax}_{\pi} \sum_{\nu} p_{\nu,k} \frac{V_{\nu,k}^{\pi}}{V_{\nu,k}^*}. \quad (9)$$

In the following theorem, we prove that for every environment $\nu \in \mathcal{M}$, the policy π^p will asymptotically achieve perfect relative rewards. We have to assume that there exists a sequence of policies $\pi_k > 0$ with this property (as for the similar Theorem 5.34 in [Hut05] which dealt with discounted values). The convergence in W -values is the relevant sense of optimality for our setting, since the V -values converge to zero for any policy.

Theorem 30 (Asymptotic optimality). *Suppose that we have a decision maker that is universal (i.e. $p_\nu > 0 \forall \nu$) with respect to the countable class \mathcal{M} of environments (which can be stochastic) and that there exists policies π_k such that for all ν , $W_k^{\pi_k, \nu} \rightarrow 1$ if ν is the actual environment (or the sequence is consistent with ν). This implies that $W_k^{\pi^p, \mu} \rightarrow 1$ where μ is the actual environment.*

The proof technique is similar to that of Theorem 5.34 in [Hut05].

Proof. Let

$$0 \leq 1 - W_k^{\pi_k, \nu} =: \Delta_\nu^k, \quad \Delta^k = \sum_\nu p_{\nu, k} \Delta_\nu^k. \quad (10)$$

The assumptions tells us that $\Delta_\nu^k = W_k^{\pi_k, \nu} - 1 \rightarrow 0$ for all ν that are consistent with the sequence ($p_{\nu, k} = 0$ if ν is inconsistent with the history at time k) and since $\Delta_\nu^k \leq 1$, it follows that

$$\Delta^k = \sum_\nu p_{\nu, k} \Delta_\nu^k \rightarrow 0.$$

Note that $p_{\mu, k}(1 - W_k^{\pi^p, \mu}) \leq \sum_\nu p_{\nu, k}(1 - W_{\pi^p, k}^\nu) \leq \sum_\nu p_{\nu, k}(1 - W_{\pi_k, \nu}^k) = \sum_\nu p_{\nu, k} \Delta_\nu^k = \Delta^k$. Since we also know that $p_{\mu, k} \geq p_{\mu, 0} > 0$ it follows that $(1 - W_k^{\pi^p, \mu}) \rightarrow 0$.

5 Conclusions

We studied complete rational decision makers including the cases of actions that may affect the environment and sequential decision making. We set up simple common sense rationality axioms that imply that a complete rational decision maker has preferences that can be characterized as maximizing expected utility. Of particular interest is the countable case where our results follow from identifying the Banach space dual of the space of contracts.

Acknowledgement. This work was supported by ARC grant DP0988049.

References

- [All53] Allais, M.: Le comportement de l'homme rationnel devant le risque: Critique des postulats et axiomes de l'école américaine. *Econometrica* 21(4), 503–546 (1953)

- [Arr70] Arrow, K.: Essays in the Theory of Risk-Bearing. North-Holland, Amsterdam (1970)
- [Cox46] Cox, R.T.: Probability, frequency and reasonable expectation. Am. Jour. Phys 14, 1–13 (1946)
- [deF37] de Finetti, B.: La prévision: Ses lois logiques, ses sources subjectives. In: Annales de l’Institut Henri Poincaré, Paris, vol. 7, pp. 1–68 (1937)
- [Die84] Diestel, J.: Sequences and series in Banach spaces. Springer, Heidelberg (1984)
- [Ell61] Ellsberg, D.: Risk, Ambiguity, and the Savage Axioms. The Quarterly Journal of Economics 75(4), 643–669 (1961)
- [Hal99] Halpern, J.Y.: A counterexample to theorems of Cox and Fine. Journal of AI research 10, 67–85 (1999)
- [Hut05] Hutter, M.: Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability. Springer, Berlin (2005)
- [Jay03] Jaynes, E.T.: Probability theory: the logic of science. Cambridge University Press, Cambridge (2003)
- [Kre89] Kreyszig, E.: Introductory Functional Analysis With Applications. Wiley, Chichester (1989)
- [NB97] Naricia, L., Beckenstein, E.: The Hahn-Banach theorem: the life and times. Topology and its Applications 77(2), 193–211 (1997)
- [NM44] Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, Princeton (1944)
- [Par94] Paris, J.B.: The uncertain reasoner’s companion: a mathematical perspective. Cambridge University Press, New York (1994)
- [Ram31] Ramsey, F.P.: Truth and probability. In: Braithwaite, R.B. (ed.) The Foundations of Mathematics and other Logical Essays, ch.7, pp. 156–198. Brace & Co. (1931)
- [Sav54] Savage, L.: The Foundations of Statistics. Wiley, New York (1954)
- [Sug91] Sugden, R.: Rational choice: A survey of contributions from economics and philosophy. Economic Journal 101(407), 751–785 (1991)

Universal Knowledge-Seeking Agents

Laurent Orseau

UMR AgroParisTech 518 / INRA
16 rue Claude Bernard, 75005 Paris, France
laurent.orseau@agroparistech.fr
<http://www.agroparistech.fr/mia/orseau>

Abstract. From a point of view of Artificial General Intelligence, RL learners like Hutter’s universal, Pareto optimal, incomputable AIXI heavily rely on the definition of the rewards, which are necessarily given by some “teacher” to define the tasks to solve. AIXI, as is, cannot therefore be said to be a fully autonomous agent.

Furthermore, it has recently been shown that AIXI can converge to a suboptimal behavior in certain situations, hence showing the intrinsic difficulty of RL, with its non-obvious pitfalls.

We propose a new model of intelligence, the Knowledge-Seeking Agent (KSA), halfway between Solomonoff Induction and AIXI, that defines a completely autonomous agent that does not require a teacher. The goal of this agent is not to maximize arbitrary rewards, but “simply” to entirely explore its world in an optimal way. A proof of strong asymptotic optimality for a class of horizon functions shows that this agent, unlike AIXI in its domain, behaves according to expectation. Some implications of such an unusual agent are proposed.

Keywords: Universal Artificial Intelligence, AIXI, Solomonoff Induction, Artificial General Intelligence.

1 Introduction

In 2000, Hutter proposed the first universal and formal, though incomputable model of an intelligent agent, AIXI [3, 2, 1]. It relies on the Reinforcement Learning framework [14] and should allow us (with computable approximations, compare [15]) to solve any practical problem as long as we are able to define the rewards.

However it was recently proved that AIXI can in certain situations stop exploring, leading to suboptimal behavior [8]. Instead of viewing this result as a flaw of the model, given its natural definition, one can view this as a hint as to why the Reinforcement Learning framework is intrinsically difficult: Given the Pareto optimality of AIXI, no learner (computable or not) can hope to behave better on average.

We propose a new model of a universal intelligent agent, where reward signals are entirely removed. The goal of the agent is no more to maximize the expected reward, but to entirely explore the world in an optimal way. We call this kind of agents *Knowledge-Seeking Agents* (KSA).

Passive Prediction, Active Learning, Reinforcement Learning. Solomonoff Induction is defined for passive prediction [13]: The agent does not output any action and thus cannot influence the behavior of the environment. As a consequence, learning is “easy”: The predictor converges to optimal prediction in approximately $K(q)$ errors, (where K is Kolmogorov’s complexity [6], and q is the sequence to predict).

AIXI is defined in the Reinforcement Learning setting, which is much harder than passive prediction because of the active setting, which means that the decisions of the agent may modify the behavior of the environment, and because there are rewards to maximize. For example, even in some simple classes of environments, no learning agent can hope to converge in less than $2^{K(q)}$ errors.

Knowledge-seeking agents are halfway between these two settings: They are active learning (incomputable) agents, but do not use external rewards to guide their behavior. We will show that this allows for a convergence proof in this active setting that was not possible in the RL setting.

Artificial General Intelligence. Hutter described AIXI as a suitable model for universal intelligence, in the sense that this agent should be able to solve any computable problem we might give it.

But regarding Artificial General Intelligence (AGI), such an agent is not fully autonomous. Indeed, it still requires a “teacher”, someone to give it the rewards: Even though the RL framework supposes that the rewards are defined by the environment (which gives to understand that we need not care about how to define them), if we create an RL robot, we will still need to specify the rewards completely, in order to define precisely what the agent should achieve. AIXI can be viewed as a “servant” AGI agent, which must serve its teacher, whereas a KSA could be viewed as a “free” AGI agent, depending on no one.

Furthermore, we will need to be extremely careful about how we define the rewards and how they are given to the RL agent, supposedly vastly intelligent. For example, how will the agent behave if it is not switched off when its task is done? Will it want to undo it in order to do it again? Can it be switched off, and will it resist being switched off, since this would prevent it from receiving further rewards? Or can we make it like (with rewards) being switched off? If so how can we prevent it from switching itself off to get rewards?

Another related concern is whether it would try to bypass the human control to give itself the rewards. RL agents tend to find “shortcuts” to make the minimum effort to receive the rewards. This should be especially true for very intelligent agents [9, 7]. We will then need to be careful that such unexpected shortcuts do not exist, which may not be a trivial matter. Hutter writes [2]:

Sufficiently intelligent agents may increase their rewards by psychologically manipulating their human “teachers”, or by threatening them.

For example, if humans use a button to control the rewards of the agent, the latter should *by all means* try to acquire the control of this button, which may lead to undesirable situations.

Hopefully all these problems have solutions, but this shows that defining rewards for a real-world RL AGI is far from a trivial matter.

Would a KSA be useful? A knowledge-seeking agent would not depend on any external intelligent entity, and would be fully autonomous. One drawback would of course be that we humans would have more difficulties to make it solve our specific problems, as it would have its own drives. But it may still be possible to use pieces of knowledge as rewards, at least up to some point. One other possibility would be to show the agent that it would itself gain knowledge if it helped us with some particular problem. Temporarily switching off the agent could be used as a punishment, since during this time the agent cannot explore the world. The agent could also be biased by showing only an adequate part of the world, either real or simulated.

However, we believe it would not be the right way to use a KSA. In fact the latter may be when on its own than directed like a RL agent with narrow goals¹. Indeed, a KSA would need to be inventive, creative to acquire as much information about the world as possible, creating its own tools, designing its own experiments, etc. For example, it may try to come up with its own unified theory of physics, and may invent new mathematical tools. We humans could gain a lot of knowledge by working along with (a computation variant of) it, instead of directing it. It may create a lot of usable, novel byproducts in the process. Such an agent could even be viewed as the optimal scientist. AGI knowledge-seeking agents would therefore be perfect complements of AGI RL agents.

After some notation, we define a first knowledge-seeking agent, the Square-KSA, and we prove convergence properties, showing that it behaves according to expectation. The second agent, Shannon-KSA, based on Shannon's entropy, is then introduced, and some of its properties are exhibited. We finally conclude with some remarks.

2 Notation

A string $s_1 s_2 \dots s_n \in S^n$ is a succession of $s_t \in S$, ordered by t . We write $s_{n:m} = s_n s_{n+1} \dots s_m$, and also $s_{<t} = s_{1:t-1}$. We note $yx_t \equiv y_t x_t$.

At each new step t , the agent outputs an action $y_t \in \mathcal{Y}$ depending on the *history of interaction* $yx_{<t} = y_1 x_1 y_2 x_2 \dots y_{t-1} x_{t-1}$, then the environment outputs $x_t \in \mathcal{X}$ depending on $yx_{<t} y_t$, and then the next step $t + 1$ begins.

The *size* of a string is denoted $|yx_{1:t}| = |x_{1:t}| = t$. Note that this is different from the length $\ell(x)$ of a string which is defined as the number of bits used to encode x on the device under consideration, such as a Turing machine.

We abbreviate $yx_{<t}$ into h when clear from the context.

\mathcal{Q} is the set of all computable environments or programs. An environment $q \in \mathcal{Q}$ is *consistent* with history $yx_{<t}$ if $q(y_{<t}) = x_{<t}$, i.e. if the execution of the

¹ More broad goals could also be defined, but to define a *universal* goal we would need *automatic*, internal rewards, as knowledge-seeking can be thought of. Also compare [11].

program q on the input string $yx_{<t}$ outputs the string $x_{<t}$. We note \mathcal{Q}_t or \mathcal{Q}_h the set of environments/programs that are consistent with history $yx_{<t} = h$.

Greek letters are probability distributions over interactions strings and priors over programs. We will often write $\rho(yx_{t:k} \mid yx_{<t})$ for convenience, brevity and clarity, whereas one should really read $\rho(x_{t:k} \mid yx_{<t} y_{t:k})$ since the actions are not random variables but are chosen deterministically by the agent.

For a set \mathcal{Q}_i of programs, we note $\rho(\mathcal{Q}_i) := \sum_{q \in \mathcal{Q}_i} \rho(q)$.

3 Knowledge-Seeking Agents

First we recall the definition of AIXI [2]. After history of interaction $yx_{<t}$, the value of a future time step k after some predicted interaction $yx_{t:k-1}$ is:

$$V_t(yx_{<k}) = \max_{y_k \in \mathcal{Y}} \sum_{x_k \in \mathcal{X}} \xi(yx_k \mid yx_{<k})(w_{t,k} \cdot r_k + V_t(yx_{<k} y_k x_k))$$

where r_k is the reward extracted from the input x_k , $w_{t,k}$ is called the *horizon function* and attributes a weight to each future time step k , depending on the present time t . The prior ξ attributes a probability to each continuation of a given input string. It is similar to Solomonoff's prior [13]. With $h = yx_{<t}$:

$$\xi(h) = \sum_{\mathcal{Q}_h} \xi(q)$$

and $\xi(q) = 2^{-\ell(q)}$ where $\ell(q)$ is the length in bits of q on a universal prefix Turing machine.

The action at time step t is chosen by:

$$y_t = \arg \max_{y \in \mathcal{Y}} V_t(yx_{<t}) \quad . \quad (1)$$

Now we generalize AIXI from the RL setting: We replace the rewards r_k by a generic *utility function* $u_t(yx_{1:k})$, which allows also for incomputable internal “rewards”; we also replace ξ by a generic universal distribution ρ satisfying:

$$\forall q \in \mathcal{Q} : 0 < \rho(q) < 1 \quad (\text{universality, probability}) \quad (2)$$

$$\sum_{q \in \mathcal{Q}} \rho(q) \leq 1 \quad ((\text{semi-})\text{measure}) \quad (3)$$

$$\rho(yx_{t:t+m} \mid yx_{<t}) = \frac{\rho(yx_{1:t+m})}{\rho(yx_{<t})} \quad (\text{chain rule}) \quad (4)$$

We also note: $\rho(h) = \rho(\mathcal{Q}_h) = \sum_{q \in \mathcal{Q}_h} \rho(q)$.

The generalized version of the value function is:

$$V_t(yx_{<k}) = \max_{y_k \in \mathcal{Y}} \sum_{x_k \in \mathcal{X}} \rho(yx_k \mid yx_{<k})(w_{t,k} \cdot u_t(yx_{1:k}) + V_t(yx_{1:k})) \quad . \quad (5)$$

We call agents defined by (II) *universal agents* $A^\rho(w_{t,k}, u_t(yx_{1:k}))$.

In this paper, we will use two different horizon functions:

$$w_{t,k}^{\leq} = \begin{cases} 1 & \text{if } k \leq m_t \\ 0 & \text{otherwise} \end{cases} \quad w_{t,k}^{\equiv} = \begin{cases} 1 & \text{if } k = m_t \\ 0 & \text{otherwise} \end{cases}$$

where $m_t = t + m'_t$ and m_t is monotonically growing with t . The first one makes the agent use its prediction for all the next m'_t steps, whereas the second one makes the agent only care about one distant step m_t .

As a simplification, in this paper we will only consider deterministic environments, but we expect all the results to extend smoothly to computable stochastic environments.

4 Square Knowledge-Seeking Agent

Now let us specify the general value function (5) for some particular horizon function and utility function.

We want our agent to converge as quickly as possible toward the true environment, i.e. it must choose its actions in order to gain as much knowledge as possible about the environment.

For this, we use a particular property of ρ (and hence of ξ): It *dominates* all computable environments, which means that if q_0 is the true environment, then $\rho(h) > \rho(q_0)$. Hence, trying to minimize $\rho(h)$ should make the agent discard (render inconsistent) as many environments as possible (and preferably the most probable ones first). Therefore we set $u_t(yx_{1:k}) = -\rho(yx_{t:k} | yx_{<t})$, and we define Square-KSA $^\rho$ with $A^\rho(w_{t,k}^{\equiv}, -\rho(yx_{t:k} | yx_{<t}))$.

As in [2], by applying the chain rule (4) repeatedly, we can put equation (5) in iterative form:

$$\begin{aligned} V_t(yx_{<t}) &= \max_{y_t \in \mathcal{Y}} \sum_{x_t \in \mathcal{X}} \dots \max_{y_{m_t} \in \mathcal{Y}} \sum_{x_{m_t} \in \mathcal{X}} \rho(yx_{t:m_t} | yx_{<t}) [-w_{t,t}\rho(yx_t | yx_{<t}) \\ &\quad - w_{t,t+1}\rho(yx_{t:t+1} | yx_{<t}) \dots - w_{t,m_t}\rho(yx_{t:m_t} | yx_{<t})] \\ &= \max_{y_t \in \mathcal{Y}} \sum_{x_t \in \mathcal{X}} \dots \max_{y_{m_t} \in \mathcal{Y}} \sum_{x_{m_t} \in \mathcal{X}} -(\rho(yx_{t:m_t} | yx_{<t}))^2 \\ &= \max_{y_{t:m_t} \in \mathcal{Y}^{m_t-t+1}} \sum_{x_{t:m_t} \in \mathcal{X}^{m_t-t+1}} -(\rho(yx_{t:m_t} | yx_{<t}))^2 . \end{aligned} \quad (6)$$

This value function is a maximization of a well-known entropy function [4].

Intuitively, the agent will try to take actions that will lead to future strings of lower probability, which makes the agent gain the most information about the environment. However, such strings, being improbable, will gain much information only with little probability. Conversely, highly probable strings have obviously a high probability to occur, but give little information about the environment: it is not informative to take actions which outcomes are predictable. Therefore,

the agent must make a trade-off to gain as much information as possible, and chooses actions that maximize the entropy over consistent environments².

4.1 Optimal Non-learning Agent

On the model of AIMU for AIXI [2], we can define a generic *optimal non-learning agent* A^μ , with which to compare the learning agent, for given utility and horizon functions. This is done by simply changing ρ to μ in equation (5), where μ is the probability distribution of the true environment. In the case of deterministic environments, for a given string of actions $y_{<t}$, $\mu(yx_{<t}) = 1$ if the environment generates the string $x_{<t}$, and 0 otherwise.

Theorem 1. *The agent $A^\mu(w_{t,k}, u_t(yx_{1:k}))$ is optimal w.r.t. the horizon function $w_{t,k}$ and the utility function $u_t(yx_{1:k})$ in deterministic environments.*

Proof. By construction: A^μ , knowing μ , and therefore knowing the future, knows the exact outcome of each string of future actions. By (5) (with ρ replaced by μ), it tests all of them and chooses the one with the highest utility. The existence of an action string of higher utility would be a contradiction. \square

In the case of stochastic computable environments, an agent may find sometimes a better string of actions than A^μ , but if its probability distribution over the actions is different from μ , it would lose on average.

For Square-KSA $^\rho$, the optimal non-learning agent Square-KSA $^\mu$ is defined by:

$$V_t^\mu(yx_{<t}) = \max_{y_{t:m_t}} \sum_{x_{t:m_t}} -\mu(yx_{t:m_t} | yx_{<t}) \rho(yx_{t:m_t} | yx_{<t}) \quad . \quad (7)$$

Thus we have: Square-KSA $^\mu \equiv A^\mu(w_{t,k}^=, -\rho(yx_{t:m_t} | yx_{<t}))$.

Square-KSA $^\mu$ will therefore always choose the optimal action, knowing how the environment will respond, in order to decrease $\rho(h)$ the most.

4.2 Asymptotic Optimality

It is difficult to give optimality criteria for universal agents, because some unexpected problems may occur, as described in [2]. Therefore, Hutter provided a new definition of optimality, called asymptotic learnability: In the limit, the agent should behave as well as the optimal agent.

The original, weak version is:

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^n V_t^\mu(yx_{<t}) - V_t^\rho(yx_{<t}) = 0$$

which can be extended to a strong version as defined in [8]:

$$\lim_{t \rightarrow \infty} \frac{V_t^\mu(yx_{<t}) - V_t^\rho(yx_{<t})}{\sum_{k=t}^{\infty} w_{t,k}} = 0 \quad . \quad (8)$$

A convergence in the weak sense can still create an infinite cumulated loss compared to a strong convergence.

² This does *not* mean the agent tries to maximize entropy inside the true environment.

5 Asymptotic Optimality of Square-KSA $^\rho$

We prove the convergence of Square-KSA $^\rho$ to Square-KSA $^\mu$ in the strong sense, for a class of horizon functions so that $m_t = f(\lceil f^{-1}(t) \rceil)$ where f is such that: $\forall t_1, t_2, 0 < t_1 < t_2 : t_1 < f(t_1) < f(t_2)$, which ensures that $\forall t, k, 0 < t \leq k \leq m_t : m_k = m_t$ (see lemma 6 in section 8 and table II). For example, $f(t) = 2^t$.

First we transform the value function of Square-KSA $^\rho$ a bit:

$$\begin{aligned} V_t(yx_{<t}) &= \max_{y_{t:m_t}} \sum_{x_{t:m_t}} -(\rho(yx_{t:m_t} \mid yx_{<t}))^2 \\ &= \max_{y_{t:m_t}} \sum_{x_{t:m_t}} -\left(\frac{\rho(yx_{1:m_t})}{\rho(yx_{<t})}\right)^2 \\ &= \max_{y_{t:m_t}} \sum_{x_{t:m_t}} -\left(\frac{\rho(\mathcal{Q}_{m_t+1})}{\rho(\mathcal{Q}_t)}\right)^2 \end{aligned} \quad (9)$$

where \mathcal{Q}_{m_t+1} is the set of environments that are consistent with one choice of $yx_{t:m_t}$.

5.1 Separability

Definition 1. We say that two environments q_1 and q_2 are h -separable if there exists a string of actions up to the horizon that makes these two environments output a different string:

$$\{q_1, q_2\} \subset \mathcal{Q}_t : [\exists y_{t:m_t} : q_1(y_{1:m_t}) \neq q_2(y_{1:m_t})] \Leftrightarrow yx_{<t}\text{-separable}(q_1, q_2) .$$

By extension, we call q_1 a h -separable environment if it is h -separable from the true environment q_0 .

Lemma 1. If q_1 and q_2 are two h -separable environments, the Square-KSA agent is assured to discard a number of environments which cumulated probability is at least $\min(\rho(q_1), \rho(q_2))/2$:

$$yx_{<t}\text{-separable}(q_1, q_2) \Rightarrow \rho(\mathcal{Q}_{m_t+1}) \leq \rho(\mathcal{Q}_t) - \min(\rho(q_1), \rho(q_2))/2 .$$

Proof. Let q_1 and q_2 be two h -separable environments at time t .

Let $g_t = \rho(\mathcal{Q}_t)$, $b_t = \frac{\rho(q_1)}{g_t}$, $c_t = \frac{\rho(q_2)}{g_t}$. Let y_t^{sep} be a string of actions of size $m_t - t + 1$ that makes q_1 and q_2 h -separable, and let y_t^{min} be a string of the same size that does not. If no y_t^{min} exists, then the lemma is trivially true.

For clarity during this proof, we omit the indexes t when clear from the context.

As V is an entropy function, the minimum value achieved if q_1 and q_2 are h -separable is when the distribution of the environments is as little spread as possible. To do so, we arbitrarily choose all environments except q_2 to be

consistent with the true environment up to the horizon (but recall that q_1 and q_2 have interchangeable roles). From (D), we have:

$$V(y^{\text{sep}}) \geq -(1 - c)^2 - c^2 .$$

Suppose there is an action string y^{\min} so that $V(y^{\text{sep}}) < V(y^{\min})$, but that would discard a minimal fraction d of g : $0 < d < \min(b, c)$. Environments q_1 , q_2 and the true environment q_0 must not be h -separable by y^{\min} , otherwise we would have $d > \min(b, c)$. At constant entropy, the minimal fraction d is achieved when this fraction d is as spread as possible over the remaining “slots” (input strings for a given action string). Therefore, for a value at least as high as $V(y^{\text{sep}})$, the agent is assured to gain at least this fraction d . We will then show that this fraction has a minimum, ensuring a sufficient decrease in g_t each time q_1 and q_2 are h -separable. Since q_1 and q_2 must not be separable by y^{\min} , we also have $d < 1 - b - c$. Thus $d < \min(b, c, 1 - b - c)$.

Maximum entropy with minimal gain d spread over $N_t = |\mathcal{X}|^{m_t-t+1} - 1$ slots is achieved by:

$$V(y^{\min}) = -(1 - d)^2 - N(d/N)^2 = -(1 - d)^2 - d^2/N$$

where we considered that q_1 and q_2 are not separable from q_0 .

We have:

$$\begin{aligned} V(y^{\text{sep}}) &< V(y^{\min}) \\ -(1 - c)^2 - c^2 &< -(1 - d)^2 - d^2/N \\ -1 - 2c^2 + 2c &< -1 - d^2 \frac{N+1}{N} + 2d \\ d^2 \frac{N+1}{2N} - d + c(1 - c) &< 0 . \end{aligned}$$

Solving this equation for d gives:

$$\frac{N}{N+1}(1-\delta) < d < \frac{N}{N+1}(1+\delta) \quad \text{with } \delta = \sqrt{1 - 2\frac{N+1}{N}c(1-c)} .$$

We are only interested in the lower bound. From algebra (see section D), we have:

$$0 \leq c \leq 0.5, N > 1 : \quad \frac{c}{2} \leq \frac{N}{N+1}\delta < d$$

and:

$$0.5 \leq c \leq 1, N > 1 : \quad \frac{b}{2} \leq \frac{(1-c)}{2} \leq \frac{N}{N+1}\delta < d .$$

So we have: $\min(\frac{b_t}{2}, \frac{c_t}{2}) < d_t < \min(b_t, c_t, 1 - b_t - c_t)$.

Note that at time k , $t < k \leq m_t = m_k$, $V_k(y_k^{\min})$ is still the value for maximum dispersion (with y_k^{\min} suffix of y_t^{\min}), since $yx_{t:k-1}$ only discarded whole “slots”

(input sequences), so the distribution of the remaining consistent environments over the remaining slots has not changed for a particular action string. Therefore, like at time t , doing any other action string than y_k^{\min} that has higher value than $V_k(y_k^{\min})$ can only be better. \square

This lemma ensures that even if y^{sep} is not chosen, and so neither q_1 nor q_2 may be discarded after the chosen string of actions, at least a certain fraction is discarded.

Lemma 2. *If q_0 and q_2 are sufficiently often h -separable environments, where q_0 is the true environment, then eventually q_2 will be discarded:*

$$\forall t : (\forall k > t : q_2 \in \mathcal{Q}_k) \Rightarrow [\exists T > t, \forall k > T : \neg(\text{yx}_{<k}\text{-separable}(q_0, q_2))] .$$

Proof. From lemma 1 and its proof, where we identify q_0 with q_1 , after executing the chosen string of actions (either y^{sep} or some y^{\min}), we have: $g_{m_t} < g_t(1 - \min(\frac{b_t}{2}, \frac{c_t}{2}))$. Since b_t and c_t can only grow with t , this implies that g_{m_t} decreases to 0 at least as fast as a geometric progression with t , when q_0 and q_2 are repeatedly h -separable. So either $\lim_{t \rightarrow \infty} g_t = 0$, or either q_0 or q_2 is discarded once $1 - b_t - c_t < \min(\frac{b_t}{2}, \frac{c_t}{2})$, i.e. the fraction of the cumulated probability of the programs other than q_0 and q_2 is not sufficient to make an entropy that is higher than discarding either q_0 or q_2 . Since q_0 cannot be discarded, q_2 is eventually discarded. \square

Lemma 3. *In deterministic environments, when $\text{yx}_{<t}$ is chosen by Square-KSA:*

$$\lim_{t \rightarrow \infty} \rho(\text{yx}_{t:m_t} \mid \text{yx}_{<t}) = \mu(\text{yx}_{t:m_t} \mid \text{yx}_{<t}) .$$

Proof. Recursive application of lemma 2 for all environments shows that for any $\epsilon > 0$, any environment q with $\rho(q) > \epsilon$ either gets discarded at some point or remains non separable from the true environment q_0 forever after some time step T . Therefore the relative probability of the sequence generated by program q_0 , which is the same as the sequence generated by any non- h -separable (from q_0) environment, tends to 1. \square

Theorem 2. *Square-KSA $^\rho$, with $m_t = f(\lceil f^{-1}(t) \rceil)$, is strongly asymptotically optimal in deterministic environments.*

Proof. Follows immediately from lemma 3 and equations (6), (7) and (8). \square

This important property ensures that the agent cannot be “tricked” to prevent it from achieving its goal, and does not get stuck in a sub-optimal behavior w.r.t. its utility function, unlike AIXI.

As higher probability environments have a higher minimal entropy when h -separable, the agent will take actions to eliminate such environments early. We thus expect a fast, or even asymptotic optimally fast convergence to μ , but this remains to be shown.

The only case where the agent may gain little knowledge is when $\rho(q_1) \gg \rho(q_0)$ and $\rho(q_1) > g/2$. This means that q_1 is very probable, not only a priori but also

from experience. However, the agent will eventually choose a string of actions that discard q_1 , but this may be long if *all* other environments (including q_0) are very improbable. Nonetheless, since the agent chooses its actions to try to eliminate (roughly speaking) as many environments as possible, its behavior may lead it, by “chance”, to discard q_1 earlier.

5.2 Full Separability

Definition 2. Two environments q_1 and q_2 are \bar{h} -separable if they are not h -separable within the horizon m_t but would be h -separable with a larger m_t .

By extension, we call q_1 a \bar{h} -separable environment if it is \bar{h} -separable from the true environment q_0 .

It is plausible that some environments may be constantly \bar{h} -separable, thus constantly avoiding to be discarded.

First, it is not certain that such environments are computable, as they might need to know what actions the agent will take in order to provably remain \bar{h} -separable.

Second, it must be noted that neither Square-KSA $^\rho$ nor Square-KSA $^\mu$ can foresee the separability of an \bar{h} -separable environment q'_0 beyond the horizon, so this does not prevent Square-KSA $^\rho$ to converge to Square-KSA $^\mu$. From their point of view, such environments behave exactly like the true environment.

Third, if the horizon $m'_t = m_t - t$ is growing, in the limit, the size of the sequence that could separate q'_0 would be infinite (although finite at any step t), and therefore q'_0 would become practically indistinguishable from q_0 . Hence such environments may not be of real importance.

The speed at which m'_t is growing does not seem to be significant, unless it is uncomputable. In that latter case, we show a full separability property:

Theorem 3. If the horizon m_t grows faster than any computable function, no environment can be \bar{h} -separable an infinite number of times.

Proof. By contradiction.

For every computable function $f(t)$ monotonically growing to infinity faster than t , there exists a computable function $f(t) = f(f(t))$ which grows faster than $f(t)$, and thus no computable function grows faster than all others. Hence m_t is not computable.

Let the computable environments q_0 and q'_0 be infinitely often \bar{h} -separable for the horizon m_t by the string y_t^{sep} . Finding y_t^{sep} is computable: At a given time t , enumerate (dove-tailing) all strings of actions in growing order, and take the first one (no need to take the shortest one) which outputs differ for the two environments. As y_t^{sep} exists, we are assured that the process eventually halts.

The function defined by $z(t) = \max_{k < t} |y_k^{\text{sep}}|$ (where $|y_k^{\text{sep}}|$ is the size of the string y_k^{sep}), which should be computable if q_0 and q'_0 are computable, monotonically grows at least as fast as m_t by definition of y_t^{sep} , and so is not computable. Therefore q'_0 is not computable. \square

6 Shannon-KSA

We can also define a knowledge-seeking agent by taking the logarithm (base 2) in the previous utility function: $u_t(yx_{1:k}) = -\log \rho(yx_{t:k} | yx_{<t})$. Hence for this new agent Shannon-KSA $^\rho$:

$$\begin{aligned} V_t(yx_{<t}) &= \max_{y_{t:t_m}} \sum_{x_t} \rho(x_t | yx_{<t}) \left[0 + \sum_{x_{t+1}} \rho(x_{t+1} | yx_{1:t}) \left[0 + \sum_{x_{t+2}} \dots \right. \right. \\ &\quad \left. \left. \sum_{x_{m_t}} \rho(x_{m_t} | yx_{<m_t}) (-\log \rho(yx_{t:m_t} | yx_{1:t}) + 0) \right] \right] \\ &= \max_{y_{t:t_m}} \sum_{x_{t:t_m}} -\rho(yx_{t:m_t} | yx_{1:t}) \log \rho(yx_{t:m_t} | yx_{1:t}) \end{aligned}$$

using (4) repeatedly (and recall the notation shortcut for $\rho(yx_{t:m_t} | yx_{1:t})$). This agent thus computes and compares Shannon's entropy [12] for strings of actions.

Theorem 4. A universal agent with utility function $u_{t,k} = -\log \rho(yx_k | yx_{<k})$ and horizon function $w_{t,k} = w_{t,k}^<$ is identical to a universal agent with utility function $u_{t,k} = -\log \rho(yx_{t:k} | yx_{<t})$ and horizon function $w_{t,k} = w_{t,k}^=$.

Proof. We start with the first agent. For clarity, we omit the actions y .

$$\begin{aligned} V_t(yx_{<t}) &= \sum_{x_t} \rho(x_t | x_{<t}) \left[-\log \rho(x_t | x_{<t}) \right. \\ &\quad \left. + \sum_{x_{t+1}} \rho(x_{t+1} | x_{1:t}) \left[-\log \rho(x_{t+1} | x_{1:t}) + \sum_{x_{t+2}} \dots \right] \right] \\ &= \sum_{x_t} -\rho(x_t | x_{<t}) \log \rho(x_t | x_{<t}) \\ &\quad + \rho(x_t | x_{<t}) \sum_{x_{t+1}} \rho(x_{t+1} | x_{1:t}) \left[-\log \rho(x_{t+1} | x_{1:t}) + \sum_{x_{t+2}} \dots \right] \\ &= \sum_{x_{t:m_t}} -\rho(x_{t:m_t} | x_{<t}) \log \rho(x_t | x_{<t}) \\ &\quad + \sum_{x_{t:t+1}} \rho(x_{t:t+1} | x_{<t}) \left[-\log \rho(x_{t+1} | x_{1:t}) + \sum_{x_{t+2}} \dots \right] \\ &\quad \dots \\ &= \sum_{x_{t:m_t}} \rho(x_{t:m_t} | x_{<t}) [-\log \rho(x_t | x_{<t}) - \log \rho(x_{t+1} | x_{1:t}) - \dots] \\ &= \sum_{x_{t:m_t}} -\rho(x_{t:m_t} | x_{<t}) \log \rho(x_{t:m_t} | x_{<t}) . \end{aligned}$$

We used the chain rule (4), the additivity property of the log function, and $\sum_{x_t} \rho(x_t | x_{<t}) \cdot z_t = \sum_{x_{t:m_t}} \rho(x_{t:m_t} | x_{<t}) \cdot z_t$. \square

This shows that Shannon-KSA is not “merely” looking at the final step, but that all intermediate steps are equally taken into account.

We expect Shannon-KSA $^\rho$ to also be strongly asymptotically optimal.

6.1 Curiosity-Seeking Complexity

Suppose we take $\rho = \xi$. For each “slot” for the entropy (i.e. for each possible future input string up to the horizon for a given action string), if we consider that lower probability environments are of marginal contribution to ξ when compared to the highest probability environment, the utility function becomes:

$$\begin{aligned} u_t(yx_{1:k}) &= -\log \xi(yx_{1:k} | yx_{<t}) \\ &= -\log \frac{\xi(yx_{1:k})}{\xi(yx_{<t})} \\ &= -\log \xi(yx_{1:k}) + \log \xi(yx_{<t}) \\ &\approx K(yx_{1:k}) - K(yx_{<t}) \end{aligned}$$

where K is the prefix Kolmogorov complexity [6]. Then we can interpret this as follows: The agent chooses its actions to maximize its knowledge of the Kolmogorov complexity of the environment.

Shannon’s entropy, usually measured in bits, makes here a lot of sense: The utility function says the agent tries to gain as many bits of information/complexity as possible. This approximate formulation of Shannon-KSA $^\xi$ has strong links with Schmidhuber’s “curiosity learning” for RL [10, 11], where the agent receives internal rewards for compressing the sequence predictor.

7 Discussion and Conclusion

We defined a new kind of universal intelligent agents, named knowledge-seeking agents, which differ significantly from the traditional Reinforcement Learning framework and its associated universal optimal learner AIXI: Their purpose is not to solve particular, narrow tasks, given or defined by experts like humans, but to be fully autonomous, and to depend on no external intelligent entity. Full autonomy is an important property if we are to create Artificial General Intelligences, that should match or surpass human or even humanity intelligence.

We believe such agents (or their computational variants) should turn out to be useful to humanity in a different way than RL agents, since they should constantly be creative and solve interesting problems that we may not yet know.

It seems that this kind of agent can still be directed to some extent, either by using pieces of knowledge as rewards, or by controlling the parts of the environment the agent interacts with, or by giving it prior knowledge. But these are

only temporary biases that decrease in strength as the agent acquires knowledge, in the convergence to optimality.

In the real world, where all agents are mortal in some way, it is unlikely that a KSA would be too curious so as to threaten its own life, since a (predicted) death would prevent it from acquiring more knowledge.

We proved convergence of Square-KSA to the optimal non-learning variant of this agent for a class of horizon functions, meaning that it behaves according to expectation, even in the limit, in all computable environments. If the horizon function grows uncomputably fast, we also proved that any environment that is different from the true one sufficiently often is eventually discarded. If the horizon function grows only in a computable way, we showed that environments that may not be discarded tend to be indistinguishable from the true environment in the limit.

The related agent, Shannon-KSA, based on Shannon's entropy, has interesting properties and its value function can be interpreted in terms of how many bits of complexity (information) the agent can expect to gain by doing a particular string of actions, and we expect this agent to also be asymptotically optimal.

As for AIXI, we expect the various KSA properties to extend nicely to stochastic computable environments.

We also expect Square-KSA $^\rho$ or Shannon-KSA $^\rho$ (or both) to be Pareto optimal, to converge quickly, in an optimal way to the true environment, i.e. no learning agent should acquire knowledge faster.

We are currently trying to get rid of the horizon function in an optimal way. One possibility could be to choose an horizon based on $\rho(h)$.

Another important concern is obviously how to optimally (for some definition of optimality) scale down Square-KSA $^\rho$ to a computable agent.

References

- [1] Hutter, M.: A theory of universal artificial intelligence based on algorithmic complexity. Arxiv (April 2000), <http://arxiv.org/abs/cs/0004001>
- [2] Hutter, M.: Universal Artificial Intelligence: Sequential Decisions Based On Algorithmic Probability. Springer, Heidelberg (2005)
- [3] Hutter, M.: Universal algorithmic intelligence: A mathematical top-down approach. In: Artificial General Intelligence, pp. 227–290. Springer, Heidelberg (2007)
- [4] Jaynes, E.T., Bretthorst, G.L.: Probability theory: the logic of science. Cambridge University Press, Cambridge (2003)
- [5] Lattimore, T., Hutter, M.: Asymptotically optimal agents. In: Proc. 22nd International Conf. on Algorithmic Learning Theory (ALT 2011), Espoo, Finland. LNCS (LNAI), vol. 6925, pp. 369–383. Springer, Berlin (2011)
- [6] Li, M., Vitanyi, P.: An Introduction to Kolmogorov Complexity and Its Applications. Springer, New York (2008)
- [7] Orseau, L., Ring, M.: Self-modification and mortality in artificial agents. In: Schmidhuber, J., Thórisson, K.R., Looks, M. (eds.) AGI 2011. LNCS, vol. 6830, pp. 1–10. Springer, Heidelberg (2011)

- [8] Orseau, L.: Optimality issues of universal greedy agents with static priors. In: Algorithmic Learning Theory, vol. 6331, pp. 345–359. Springer, Heidelberg (2010)
- [9] Ring, M., Orseau, L.: Delusion, survival, and intelligent agents. In: Schmidhuber, J., Thórisson, K.R., Looks, M. (eds.) AGI 2011. LNCS, vol. 6830, pp. 11–20. Springer, Heidelberg (2011)
- [10] Schmidhuber, J.: Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. In: Pezzulo, G., Butz, M.V., Sigaud, O., Baldassarre, G. (eds.) Anticipatory Behavior in Adaptive Learning Systems. LNCS, vol. 5499, pp. 48–76. Springer, Heidelberg (2009)
- [11] Schmidhuber, J.: Artificial scientists a artists based on the formal theory of creativity. In: Proceedings of the 3d Conference on Artificial General Intelligence (AGI 2010), Lugano, Switzerland, pp. 145–150 (2010)
- [12] Shannon, C.E.: A mathematical theory of communication (parts I and II). Bell System Technical Journal 27, 379–423, 623–656 (1948)
- [13] Solomonoff, R.: Complexity-based induction systems: comparisons and convergence theorems. IEEE transactions on Information Theory 24(4), 422–432 (1978)
- [14] Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998) (a Bradford Book)
- [15] Veness, J., Ng, K.S., Hutter, M., Silver, D.: A monte carlo AIXI approximation. Arxiv (September 2009), <http://arxiv.org/abs/0909.0801>

8 Technical Proofs

Lemma 4

$$\forall c, N, 0 \leq c \leq 0.5, N > 1 : \frac{c}{2} \leq \frac{N}{N+1} \left(1 - \sqrt{1 - 2 \frac{N+1}{N} c(1-c)} \right) \quad .$$

Proof

$$\begin{aligned} \frac{c}{2} &\leq \frac{N}{N+1} \left(1 - \sqrt{1 - 2 \frac{N+1}{N} c(1-c)} \right) \\ 1 - \frac{N+1}{2N} c &\geq \sqrt{1 - 2 \frac{N+1}{N} c(1-c)} \\ 1 + \left(\frac{N+1}{2N} c \right)^2 - \frac{N+1}{N} c &\geq 1 - 2 \frac{N+1}{N} c(1-c) \\ \frac{N+1}{8N} c - \frac{1}{2} &\geq -(1-c) \\ c \frac{-7N+1}{8N} &\geq -\frac{1}{2} \\ c &\leq \frac{4N}{7N-1} \end{aligned}$$

which is true by the premises. \square

Lemma 5

$$\forall c, N, 0.5 \leq c \leq 1, N > 1 : \frac{1-c}{2} \leq \frac{N}{N+1} \left(1 - \sqrt{1 - 2 \frac{N+1}{N} c(1-c)} \right) .$$

Proof

$$\begin{aligned} \frac{N+1}{2N}(1-c) &\leq 1 - \sqrt{1 - 2\frac{N+1}{N}c(1-c)} \\ 1 - \frac{N+1}{2N}(1-c) &\geq \sqrt{1 - 2\frac{N+1}{N}c(1-c)} \\ \left(\frac{N+1}{2N}(1-c)\right)^2 - \frac{N+1}{N}(1-c) &\geq -2\frac{N+1}{N}c(1-c) \\ (N+1)(1-c) - 4N &\geq -8Nc \\ -3N + 1 &\geq c(-7N + 1) \\ c &\geq \frac{3N-1}{7N-1} \end{aligned}$$

which is true by the premises.

Lemma 6

$$m_t = f(\lceil f^{-1}(t) \rceil) \wedge \forall t_1, t_2, 0 < t_1 < t_2 : t_1 < f(t_1) < f(t_2) \\ \Rightarrow \forall t, k, 0 < t \leq k \leq m_t : m_k = m_t \quad .$$

Proof

$$\begin{aligned} \frac{t}{f(\lceil f^{-1}(t) \rceil)} &\leq \frac{k}{f(\lceil f^{-1}(k) \rceil)} \leq \frac{f(\lceil f^{-1}(t) \rceil)}{f(\lceil f^{-1}(f(\lceil f^{-1}(t) \rceil)) \rceil)} = f(\lceil f^{-1}(t) \rceil) . \end{aligned}$$

Remarks. Such horizon function m_t is a “stair” function: the agent sets one point in the future, takes this point for its horizon until it reaches it, then sets a new, further point in the future and so on. The function $f(t)$ can be any function growing faster than t . Possible choices are $f(t) = t^2$ and $f(t) = 2^t$.

Table 1. Evolution of $m_t \equiv 2^{\lceil \log_2(t) \rceil}$ for the first values of t

Asymptotically Optimal Agents

Tor Lattimore¹ and Marcus Hutter^{1,2}

¹ Research School of Computer Science
Australian National University

² ETH Zürich
`{tor.lattimore,marcus.hutter}@anu.edu.au`

Abstract. Artificial general intelligence aims to create agents capable of learning to solve arbitrary interesting problems. We define two versions of asymptotic optimality and prove that no agent can satisfy the strong version while in some cases, depending on discounting, there does exist a non-computable weak asymptotically optimal agent.

Keywords: Rational agents, sequential decision theory, artificial general intelligence, reinforcement learning, asymptotic optimality, general discounting.

1 Introduction

The dream of artificial general intelligence is to create an agent that, starting with no knowledge of its environment, eventually learns to behave optimally. This means it should be able to learn chess just by playing, or Go, or how to drive a car or mow the lawn, or any task we could conceivably be interested in assigning it.

Before considering the existence of universally intelligent agents, we must be precise about what is meant by optimality. If the environment and goal are known, then subject to computation issues, the optimal policy is easy to construct using an expectimax search from sequential decision theory [13]. However, if the true environment is unknown then the agent will necessarily spend some time exploring, and so cannot immediately play according to the optimal policy. Given a class of environments, we suggest two definitions of asymptotic optimality for an agent.

1. An agent is strongly asymptotically optimal if for every environment in the class it plays optimally in the limit.
2. It is weakly asymptotic optimal if for every environment in the class it plays optimally *on average* in the limit.

The key difference is that a strong asymptotically optimal agent must eventually stop exploring, while a weak asymptotically optimal agent may explore forever, but with decreasing frequency.

In this paper we consider the (non-)existence of weak/strong asymptotically optimal agents in the class of all deterministic computable environments. The restriction to deterministic is for the sake of simplicity and because the results

for this case are already sufficiently non-trivial to be interesting. The restriction to computable is more philosophical. The Church-Turing thesis is the unprovable hypothesis that anything that can intuitively be computed can also be computed by a Turing machine. Applying this to physics leads to the strong Church-Turing thesis that the universe is computable (possibly stochastically computable, i.e. computable when given access to an oracle of random noise). Having made these assumptions, the largest interesting class then becomes the class of computable (possibly stochastic) environments.

In [7], Hutter conjectured that his universal Bayesian agent, AIXI, was weakly asymptotically optimal in the class of all computable stochastic environments. Unfortunately this was recently shown to be false in [14], where it is proven that no Bayesian agent (with a static prior) can be weakly asymptotically optimal in this class.¹ The key idea behind Orseau’s proof was to show that AIXI eventually stops exploring. This is somewhat surprising because it is normally assumed that Bayesian agents solve the exploration/exploitation dilemma in a principled way. This result is a bit reminiscent of Bayesian (passive induction) inconsistency results [3, 4], although the details of the failure are very different.

We extend the work of [14], where only Bayesian agents are considered, to show that non-computable weak asymptotically optimal agents do exist in the class of deterministic computable environments for some discount functions (including geometric), but not for others. We also show that no asymptotically optimal agent can be computable, and that for all “reasonable” discount functions there does not exist a strong asymptotically optimal agent.

The weak asymptotically optimal agent we construct is similar to AIXI, but with an exploration component similar to ϵ -learning for finite state Markov decision processes or the UCB algorithm for bandits. The key is to explore sufficiently often and deeply to ensure that the environment used for the model is an adequate approximation of the true environment. At the same time, the agent must explore infrequently enough that it actually exploits its knowledge. Whether or not it is possible to get this balance right depends, somewhat surprisingly, on how forward looking the agent is (determined by the discount function). That it is sometimes not possible to explore enough to learn the true environment without damaging even a weak form of asymptotic optimality is surprising and unexpected.

Note that the exploration/exploitation problem is well-understood in the Bandit case [1, 2] and for (finite-state stationary) Markov decision processes [15]. In these restrictive settings, various satisfactory optimality criteria are available. In this work, we do not make any assumptions like Markov, stationary, ergodicity, or else besides computability of the environment. So far, no satisfactory optimality definition is available for this general case.

2 Notation and Definitions

We use similar notation to [7, 14] where the agent takes actions and the environment returns an observation/reward pair.

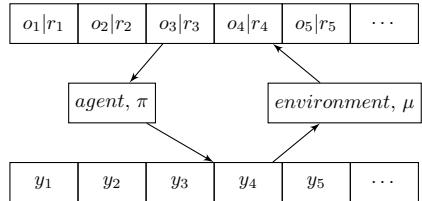
¹ Or even the class of computable deterministic environments.

Strings. A finite string a over alphabet \mathcal{A} is a finite sequence $a_1a_2a_3 \cdots a_{n-1}a_n$ with $a_i \in \mathcal{A}$. An infinite string ω over alphabet \mathcal{A} is an infinite sequence $\omega_1\omega_2\omega_3 \cdots$. \mathcal{A}^n , \mathcal{A}^* and \mathcal{A}^∞ are the sets of strings of length n , strings of finite length, and infinite strings respectively. Let x be a string (finite or infinite) then substrings are denoted $x_{s:t} := x_sx_{s+1} \cdots x_{t-1}x_t$ where $s, t \in \mathbb{N}$ and $s \leq t$. Strings may be concatenated. Let $x, y \in \mathcal{A}^*$ of length n and m respectively, and $\omega \in \mathcal{A}^\infty$. Then define $xy := x_1x_2 \cdots x_{n-1}x_n y_1y_2 \cdots y_{m-1}y_m$ and $x\omega := x_1x_2 \cdots x_{n-1}x_n \omega_1\omega_2\omega_3 \cdots$. Some useful shorthands,

$$x_{<t} := x_{1:t-1} \quad yx_{<t} := y_1x_1y_2x_2 \cdots y_{t-1}x_{t-1}. \quad (1)$$

The second of these is ambiguous with concatenation, so wherever $yx_{<t}$ appears we assume the interleaving definition of (1) is intended. For example, it will be common to see $yx_{<t}y_t$, which represents the string $y_1x_1y_2x_2y_3x_3 \cdots y_{t-1}x_{t-1}y_t$. For binary strings, we write $\#1(a)$ and $\#0(a)$ to mean the number of 0's and number of 1's in a respectively.

Environments and Optimality. Let \mathcal{Y} , \mathcal{O} and $\mathcal{R} \subset \mathbb{R}$ be action, observation and reward spaces respectively. Let $\mathcal{X} = \mathcal{O} \times \mathcal{R}$. An agent interacts with an environment as illustrated in the diagram on the right. First, the agent takes an action, upon which it receives a new observation/reward pair. The agent then takes another action, receives another observation/reward pair, and so-on indefinitely. The goal of the agent is to maximise its discounted rewards over time. In this paper we consider only deterministic environments where the next observation/reward pair is determined by a function of the previous actions, observations and rewards.



Definition 1 (Deterministic Environment). A deterministic environment μ is a function $\mu : (\mathcal{Y} \times \mathcal{X})^* \times \mathcal{Y} \rightarrow \mathcal{X}$ where $\mu(yx_{<t}y_t) \in \mathcal{X}$ is the observation/reward pair given after action y_t is taken in history $yx_{<t}$. Wherever we write x_t we implicitly assume $x_t = (o_t, r_t)$ and refer to o_t and r_t without defining them. An environment μ is computable if there exists a Turing machine that computes it.

Note that since environments are deterministic the next observation need not depend on the previous observations (only actions). We choose to leave the dependence as the proofs become clearer when both the action and observation sequence is more visible.

Assumption 1. \mathcal{Y} and \mathcal{O} are finite, $\mathcal{R} = [0, 1]$.

Definition 2 (Policy). A policy π is a function from a history to an action $\pi : (\mathcal{Y} \times \mathcal{X})^* \rightarrow \mathcal{Y}$.

As expected, a policy π and environment μ can interact with each other to generate a play-out sequence of action/reward/observation tuples.

Definition 3 (Play-out Sequence). We define the play-out sequence $yx^{\mu, \pi} \in (\mathcal{Y} \times \mathcal{X})^\infty$ inductively by $y_k^{\mu, \pi} := \pi(yx_{<k}^{\mu, \pi})$ and $x_k^{\mu, \pi} := \mu(yx_{<k}^{\mu, \pi} y_k^{\mu, \pi})$.

We need to define the value of a policy π in environment μ . To avoid the possibility of infinite rewards, we will use discounted values. While it is common to use only geometric discounting, we have two reasons to allow arbitrary time-consistent discount functions.

1. Geometric discounting has a constant effective horizon, but we feel agents should be allowed to use a discount function that leads to a growing horizon. This is seen in other agents, such as humans, who generally become less myopic as they grow older. See [5] for a overview of generic discounting.
2. The existence of asymptotically optimal agents depends critically on the effective horizon of the discount function.

Definition 4 (Discount Function). A regular discount function $\gamma \in \mathbb{R}^\infty$ is a vector satisfying $\gamma_k \geq 0$ and $0 < \sum_{t=k}^{\infty} \gamma_t < \infty$ for all $k \in \mathbb{N}$.

The first condition is natural for any definition of a discount function. The second condition is often cited as the purpose of a discount function (to prevent infinite utilities), but economists sometimes use non-summable discount functions, such as hyperbolic. The second condition also guarantees the agent cares about the infinite future, and is required to make asymptotic analysis interesting. We only consider discount functions satisfying all three conditions. In the following, let

$$\Gamma_t := \sum_{i=t}^{\infty} \gamma_i \quad H_t(p) := \min_{h \in \mathbb{N}} \left\{ h : \frac{1}{\Gamma_t} \sum_{k=t}^{t+h} \gamma_k > p \right\}.$$

An infinite sequence of rewards starting at time t , $r_t, r_{t+1}, r_{t+2}, \dots$ is given a value of $\frac{1}{\Gamma_t} \sum_{i=t}^{\infty} \gamma_i r_i$. The term $\frac{1}{\Gamma_t}$ is a normalisation term to ensure that values scale in such a way that they can still be compared in the limit. A discount function is computable if there exists a Turing machine computing it. All well known discount functions, such as geometric, fixed horizon and hyperbolic are computable. Note that $H_t(p)$ exists for all $p \in [0, 1)$ and represents the effective horizon of the agent. After $H_t(p)$ time-steps into the future, starting at time t , the agent stands to gain/lose at most $1 - p$.

Definition 5 (Values and Optimal Policy). The value of policy π when starting from history $yx_{<t}^{\mu, \pi}$ in environment μ is $V_\mu^\pi(yx_{<t}^{\mu, \pi}) := \frac{1}{\Gamma_t} \sum_{k=t}^{\infty} \gamma_k r_k^{\mu, \pi}$. The optimal policy π_μ^* and its value V_μ^* are defined $\pi_\mu^*(yx_{<t}) := \arg \max_\pi V_\mu^\pi(yx_{<t})$ and $V_\mu^*(yx_{<t}) := V_\mu^{\pi_\mu^*}(yx_{<t})$.

Assumption 1 combined with Theorem 6 in [9] guarantees the existence of π_μ^* . Note that the normalisation term $\frac{1}{\Gamma_t}$ does not change the policy, but is used to ensure that values scale appropriately in the limit. For example, when discounting geometrically we have, $\gamma_t = \gamma^t$ for some $\gamma \in (0, 1)$ and so $\Gamma_t = \frac{\gamma^t}{1-\gamma}$ and $V_\mu^\pi(yx_{<t}^{\mu, \pi}) = (1 - \gamma) \sum_{k=t}^{\infty} \gamma^{k-t} r_k^{\mu, \pi}$.

Definition 6 (Asymptotic Optimality). Let $\mathcal{M} = \{\mu_0, \mu_1, \dots\}$ be a finite or countable set of environments and γ be a discount function. A policy π is a strong asymptotically optimal policy in (\mathcal{M}, γ) if

$$\lim_{n \rightarrow \infty} [V_\mu^*(yx_{<n}^{\mu, \pi}) - V_\mu^\pi(yx_{<n}^{\mu, \pi})] = 0, \text{ for all } \mu \in \mathcal{M}. \quad (2)$$

It is a weak asymptotically optimal policy if

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^n [V_\mu^*(yx_{<t}^{\mu, \pi}) - V_\mu^\pi(yx_{<t}^{\mu, \pi})] = 0, \text{ for all } \mu \in \mathcal{M}. \quad (3)$$

Strong asymptotic optimality demands that the value of a *single* policy π converges to the value of the optimal policy π_μ^* for *all* μ in the class. This means that in the limit, a strong asymptotically optimal policy will obtain the maximum value possible in that environments.

Weak asymptotic optimality is similar, but only requires the *average* value of the policy π to converge to the average value of the optimal policy. This means that a weak asymptotically optimal policy can still make infinitely many bad mistakes, but must do so for only a fraction of the time that converges to zero. Strong asymptotic optimality implies weak asymptotic optimality.

While the definition of strong asymptotic optimality is rather natural, the definition of weak asymptotic optimality appears somewhat more arbitrary. The purpose of the average is to allow the agent to make a vanishing fraction of serious errors over its (infinite) life-time. We believe this is a necessary condition for an agent to learn the true environment. Of course, it would be possible to insist that the agent make only $o(\log n)$ serious errors rather than $o(n)$, which would make a stronger version of weak asymptotic optimality. Our choice is the weakest notion of optimality of the above form that still makes sense, which turns out to be already too strong for some discount rates.

Note that for both versions of optimality an agent would be considered optimal if it actively undertook a policy that led it to an extremely bad “hell” state from which it could not escape. Since the state cannot be escaped, its policy would then coincide with the optimal policy and so it would be considered optimal. Unfortunately, this problem seems to be an unavoidable consequence of learning algorithms in non-ergodic environments in general, including the currently fashionable PAC algorithms for arbitrary finite Markov decision processes.

3 Non-existence of Asymptotically Optimal Policies

We present the negative theorem in three parts. The first shows that, at least for computable discount functions, there does not exist a strong asymptotically optimal policy. The second shows that any weak asymptotically optimal policy must be incomputable while the third shows that there exist discount functions for which even incomputable weak asymptotically optimal policies do not exist.

Theorem 7. Let \mathcal{M} be the class of all deterministic computable environments and γ a computable discount function, then:

1. There does not exist a strong asymptotically optimal policy in (\mathcal{M}, γ) .
2. There does not exist a computable weak asymptotically optimal policy in (\mathcal{M}, γ) .
3. If $\gamma_k := \frac{1}{k(k+1)}$ then there does not exist a weak asymptotically optimal policy in (\mathcal{M}, γ) .

Part 1 of Theorem 7 says there is no strong asymptotically optimal policy in the class of all computable deterministic environments when the discount function is computable. It is likely there exist non-computable discount functions for which there are strong asymptotically optimal policies. Unfortunately the discount functions for which this is true are likely to be somewhat pathological and not realistic.

Given that strong asymptotic optimality is too strong, we should search for weak asymptotically optimal policies. Part 2 of Theorem 7 shows that any such policy is necessarily incomputable. This result features no real new ideas and relies on the fact that you can use a computable policy to hand-craft a computable environment in which it does very badly [10]. In general this approach fails for incomputable policies because the hand-crafted environment will then not be computable. Note that this does not rule out the existence of a stochastically computable weak asymptotically optimal policy.

It turns out that even weak asymptotic optimality is too strong for some discount functions. Part 3 of Theorem 7 gives an example discount function for which no such policy (computable or otherwise) exists. In the next section we introduce a weak asymptotically optimal policy for geometric (and may be extended to other) discounting. Note that $\gamma_k = \frac{1}{k(k+1)}$ is an example of a discount function where $H_t(p) = \Omega(t)$. It is also analytically easy to work with.

All negative results are proven by contradiction, and follow the same basic form.

1. Assume π is a computable/arbitrary weak/strong asymptotically optimal.
2. Therefore π is weak/strong asymptotically optimal in μ for some particular μ .
3. Construct ν , which is indistinguishable from μ under π , but where π is not weak/strong asymptotically optimal in ν .

It is worth remarking that for all counter-examples, the set of observations \mathcal{O} is empty and so all results apply also to bandits. Space does not permit us to present the proofs of part 1 and 2, which in any case are substantially easier than part 3.

Proof (Theorem 7, Part 3). Recall $\gamma_k = \frac{1}{k(k+1)}$ and so $\Gamma_t = \frac{1}{t}$. Now let $\mathcal{Y} = \{\text{up}, \text{down}\}$ and $\mathcal{O} = \emptyset$. Define μ by

$$\mu(y_{r < t} y_t) = \begin{cases} \frac{1}{2} & \text{if } y_t = \text{up} \\ \frac{1}{2} - \epsilon & \text{if } y_t = \text{down} \end{cases}$$

where $\epsilon \in (0, \frac{1}{2})$ will be chosen later. As before, $V_\mu^*(y_{<t}) = \frac{1}{2}$. Assume π is weakly asymptotically optimal. Therefore

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^n V_\mu^\pi(y_{<t}^{\mu,\pi}) = \frac{1}{2}. \quad (4)$$

We show by contradiction that π cannot explore (take action *down*) too often. Assume there exists an infinite time-sequence t_1, t_2, t_3, \dots such that $\pi(y_{<t}^{\mu,\pi}) = \text{down}$ for all $t \in \bigcup_{i=1}^{\infty} [t_i, 2t_i]$. Then for $t \in [t_i, \frac{3}{2}t_i]$ we have

$$V_\mu^\pi(y_{<t}^{\mu,\pi}) \equiv \frac{1}{T} \sum_{k=t}^{\infty} \gamma_k r_k^{\mu,\pi} \leq t \left[\left(\frac{1}{2} - \epsilon \right) \sum_{k=t}^{2t_i} \gamma_k + \frac{1}{2} \sum_{k=2t_i+1}^{\infty} \gamma_k \right] \quad (5)$$

$$= \frac{1}{2} - \epsilon \left[1 - \frac{t}{2t_i + 1} \right] < \frac{1}{2} - \frac{\epsilon}{4} \quad (6)$$

where (5) is the definition of the value function and the previous assumption and definition of μ . (6) by algebra and since $t \in [t_i, \frac{3}{2}t_i]$. Therefore

$$\frac{1}{2t_i} \sum_{t=1}^{2t_i} V_\mu^\pi(y_{<t}^{\mu,\pi}) < \frac{1}{2t_i} \left[\sum_{t=1}^{t_i-1} \frac{1}{2} + \sum_{t=t_i}^{\frac{3}{2}t_i-1} \left(\frac{1}{2} - \frac{\epsilon}{4} \right) + \sum_{t=\frac{3}{2}t_i}^{2t_i} \frac{1}{2} \right] = \frac{1}{2} - \frac{1}{16}\epsilon. \quad (7)$$

The first inequality follows from (6) and because the maximum value of any play-out sequence in μ is $\frac{1}{2}$. The second by algebra. Therefore $\liminf_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^n V_\mu^\pi(y_{<t}^{\mu,\pi}) < \frac{1}{2} - \frac{1}{16}\epsilon < \frac{1}{2}$, which contradicts (4). Therefore there does not exist a time-sequence $t_1 < t_2 < t_3 < \dots$ such that $\pi(y_{<t}^{\mu,\pi}) = \text{down}$ for all $t \in \bigcup_{i=1}^{\infty} [t_i, 2t_i]$.

So far we have shown that π cannot “explore” for t consecutive time-steps starting at time-step t , infinitely often. We now construct an environment similar to μ where this is required. Choose T to be larger than the last time-step t at which $y_s^{\mu,\pi} = \text{down}$ for all $s \in [t, 2t]$. Define ν by

$$\nu(y_{<t} y_t) = \begin{cases} \mu(y_{<t} y_t) & \text{if } t < T \\ \frac{1}{2} & \text{if } y_t = \text{down} \text{ and there does not exist } t' \geq T \\ & \text{such that } y_s = \text{down} \forall s \in [t', 2t'] \\ 1 & \text{if } y_t = \text{down} \text{ and exists } t' \geq T \text{ such that } 2t' < t \text{ and} \\ & y_s = \text{down} \forall s \in [t', 2t'] \\ \frac{1}{2} - \epsilon & \text{otherwise} \end{cases}$$

Now we compare the values in environment ν of π and π_ν^* at times $t \geq T$. Since π does not take action *down* for t consecutive time-steps at any time after T , it never “unlocks” the reward of 1 and so $V_\nu^\pi(y_{<t}^{\nu,\pi}) \leq \frac{1}{2}$. Now let $\tilde{\pi}(y_{<t}) = \text{down}$ for all $y_{<t}$. Therefore, for $t \geq 2T$,

$$V_\nu^{\tilde{\pi}}(y_{<t}^{\nu,\pi}) \equiv \frac{1}{\Gamma_t} \sum_{k=t}^{\infty} \gamma_k r_k^{\nu,\tilde{\pi}} \geq t \left[\left(\frac{1}{2} - \epsilon \right) \sum_{k=t}^{2t-1} \gamma_k + \sum_{k=2t}^{\infty} \gamma_k \right] \quad (8)$$

$$= t \left[\left(\frac{1}{2} - \epsilon \right) \left(\frac{1}{t} - \frac{1}{2t} \right) + \frac{1}{2t} \right] = \frac{3}{4} - \frac{1}{2}\epsilon \quad (9)$$

where (8) follows by the definition of ν and $\tilde{\pi}$. (9) by the definition of γ_k and algebra. Finally, setting $\epsilon = \frac{1}{4}$ gives $V_\nu^{\tilde{\pi}}(y_{<t}^{\nu,\pi}) \geq \frac{5}{8} = \frac{1}{2} + \frac{1}{8}$. Since $V_\nu^* \geq V_\nu^{\tilde{\pi}}$, we get $V_\nu^*(y_{<t}^{\nu,\pi}) - V_\nu^{\tilde{\pi}}(y_{<t}^{\nu,\pi}) \geq V_\nu^*(y_{<t}^{\nu,\pi}) - V_\nu^{\pi}(y_{<t}^{\nu,\pi}) \geq \frac{1}{8}$. Therefore $\limsup_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^n [V^*(y_{<t}^{\nu,\pi}) - V_\nu^{\pi}(y_{<t}^{\nu,\pi})] \geq \frac{1}{8}$, and so π is not weakly asymptotically optimal. \square

We believe it should be possible to generalise the above to computable discount functions with $H_t(p) > c_p t$ with $c_p > 0$ for infinitely many t , but the proof will likely be messy.

4 Existence of Weak Asymptotically Optimal Policies

In the previous section we showed there did not exist a strong asymptotically optimal policy (for most discount functions) and that any weak asymptotically optimal policy must be incomputable. In this section we show that a weak asymptotically optimal policy exists for geometric discounting (and is, of course, incomputable).

The policy is reminiscent of ϵ -exploration in finite state MDPs (or UCB for bandits) in that it spends most of its time exploiting the information it already knows, while still exploring sufficiently often (and for sufficiently long) to detect any significant errors in its model.

The idea will be to use a model-based policy that chooses its current model to be the first environment in the model class (all computable deterministic environments) consistent with the history seen so far. With increasing probability it takes the best action according to this policy, while still occasionally exploring randomly. When it explores it always does so in bursts of increasing length.

Definition 8 (History Consistent). A deterministic environment μ is consistent with history $y_{<t}$ if $\mu(y_{<k} y_k) = x_k$, for all $k < t$.

Definition 9 (Weak Asymptotically Optimal Policy). Let $\mathcal{Y} = \{0, 1\}$ and $\mathcal{M} = \{\mu_1, \mu_2, \mu_3, \dots\}$ be a countable class of deterministic environments. Define a probability measure P on \mathcal{B}^∞ inductively by, $P(z_n = 1 | z_{<n}) := \frac{1}{n}$, for all $z_{<n} \in \mathcal{B}^{n-1}$. Now let $\chi \in \mathcal{B}^\infty$ be sampled from P and define $\bar{\chi}, \dot{\chi}^h \in \mathcal{B}^\infty$ by

$$\bar{\chi}_k := \begin{cases} 1 & \text{if } k \in \bigcup_{i:\chi_i=1} [i, i + \log i] \\ 0 & \text{otherwise} \end{cases} \quad \dot{\chi}_k^h := \begin{cases} 0 & \text{if } \bar{\chi}_{k:k+h} = 0^{h+1} \\ 1 & \text{otherwise} \end{cases}$$

Next let ψ be sampled from the uniform measure (each bit of ψ is independently sampled from a Bernoulli 1/2 distribution) and define a policy π by,

$$\pi(yx_{<t}) := \begin{cases} \pi_{\nu_t}^*(yx_{<t}^{\pi,\mu}) & \text{if } \bar{\chi}_t = 0 \\ \psi_t & \text{otherwise} \end{cases} \quad (10)$$

where $\nu_t = \mu_{i_t}$ with $i_t = \min \{i : \mu_i \text{ consistent with history } yx_{<t}^{\pi,\mu}\} < \infty$. Note that i_t is always finite because there exists an i such that $\mu_i = \mu$, in which case μ_i is necessarily consistent with $yx_{<t}^{\pi,\mu}$.

Intuitively, $\chi_k = 1$ at time-steps when the agent will explore for $\log k$ time-steps. $\bar{\chi}_k = 1$ if the agent is exploring at time k and ψ_k is the action taken if exploring at time-step k . $\dot{\chi}$ will be used later, with $\dot{\chi}_k^h = 1$ if the agent will explore at least once in the interval $[k, k+h]$. If the agent is not exploring then it acts according to the optimal policy for the first consistent environment in \mathcal{M} .

Theorem 10. Let $\gamma_t = \gamma^t$ with $\gamma \in (0, 1)$ (geometric discounting) then the policy defined in Definition 9 is weakly asymptotically optimal in the class of all deterministic computable environments with probability 1.

Some remarks:

1. That $\mathcal{Y} = \{0, 1\}$ is only convenience, rather than necessity. The policy is easily generalised to arbitrary finite \mathcal{Y} .
2. π is essentially a stochastic policy. With some technical difficulties it is possible to construct an equivalent deterministic policy. This is done by choosing χ to be any P -Martin-Löf random sequence and ψ to be a sequence that is Martin-Löf random w.r.t to the uniform measure. The theorem then holds for *all* deterministic environments. The proof is somewhat delicate and may not extend nicely to stochastic environments. For an introduction to Kolmogorov complexity and Martin-Löf randomness, see [12]. For a reason why the stochastic case may not go through as easily, see [8].
3. The policy defined in Definition 9 is not computable for two reasons. First, because it relies on the stochastic sequences χ and ψ . Second, because the operation of finding the first environment consistent with the history is not computable.² We do not know if there exists a weak asymptotically optimal policy that is computable when given access to a random number generator (or if it is given χ and ψ).
4. The bursts of exploration are required for optimality. Without them it will be possible to construct counter-example environments similar to those used in part 3 of Theorem 7.

Before the proof we require some more definitions and lemmas. Easier proofs are omitted due to space limitations.

² The class of computable environments is not recursively enumerable [12].

Definition 11 (h-Difference). Let μ and ν be two environments consistent with history $yx_{<t}$, then μ is h -different to ν if there exists $yx_{t:t+h}$ satisfying

$$\begin{aligned} y_k &= \pi_\mu^*(yx_{<k}) \text{ for all } k \in [t, t+h], \\ x_k &= \mu(yx_{<k}y_k) \text{ for all } k \in [t, t+h], \\ x_k &\neq \nu(yx_{<k}y_k) \text{ for some } k \in [t, t+h]. \end{aligned}$$

Intuitively, μ is h -different to ν at history $yx_{<t}$ if playing the optimal policy for μ for h time-steps makes ν inconsistent with the new history. Note that h -difference is *not* symmetric.

Lemma 12. If $a_n \in [0, 1]$ and $\limsup_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n a_i = \epsilon$ and $\alpha \in \mathcal{B}^\infty$ is an indicator sequence with $\alpha_i := \llbracket a_i \geq \epsilon/4 \rrbracket$ ³ then $\prod_{i=1}^\infty \left[1 - \frac{\alpha_i}{i}\right] = 0$.

Lemma 13. Let a_1, a_2, a_3, \dots be a sequence with $a_n \in [0, 1]$ for all n . The following properties of χ are true with probability 1.

1. For any h , $\limsup_{n \rightarrow \infty} \frac{1}{n} \#1(\dot{\chi}_{1:n}^h) = 0$.
2. If $\limsup_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n a_i = \epsilon > 0$ and $\alpha_i := \llbracket a_i > \epsilon/2 \rrbracket$ then $\alpha_i = \chi_i = 1$ for infinitely many i .

Proof. 1. Let $i \in \mathbb{N}$, $\epsilon > 0$ and E_i^ϵ be the event that $\#1(\dot{\chi}_{1:2^i}^h) > 2^i\epsilon$. Using the definition of $\dot{\chi}^h$ to compute the expectation $\mathbf{E}[\#1(\dot{\chi}_{1:2^i}^h)] < i(i+1)h$ and applying the Markov inequality gives that $P(E_i^\epsilon) < i(i+1)h2^{-i}/\epsilon$. Therefore $\sum_{i \in \mathbb{N}} P(E_i^\epsilon) < \infty$. Therefore the Borel-Cantelli lemma gives that E_i^ϵ occurs for only finitely many i with probability 1. We now assume that $\limsup_{n \rightarrow \infty} \frac{1}{n} \#1(\dot{\chi}_{1:n}^h) > 2\epsilon > 0$ and show that E_i^ϵ must occur infinitely often. By the definition of \limsup and our assumption we have that there exists a sequence n_1, n_2, \dots such that $\#1(\dot{\chi}_{1:n_i}^h) > 2n_i\epsilon$ for all $i \in \mathbb{N}$. Let $n^+ := \min_{k \in \mathbb{N}} \{2^k : 2^k \geq n\}$ and note that $\#1(\dot{\chi}_{1:n_i^+}^h) > n_i^+\epsilon$, which is exactly $E_{\log n_i^+}^\epsilon$. Therefore there exist infinitely many i such that E_i^ϵ occurs and so $\limsup_{n \rightarrow \infty} \frac{1}{n} \#1(\dot{\chi}_{1:n}^h) = 0$ with probability 1.

2. The probability that $\alpha_i = 1 \implies \chi_i = 0$ for all $i \geq T$ is $P(\alpha_i = 1 \implies \chi_i = 0 \forall i \geq T) = \prod_{i=T}^\infty \left(1 - \frac{\alpha_i}{i}\right) =: p = 0$, by Lemma 12. Therefore the probability that $\alpha_i = \chi_i = 1$ for only finitely many i is zero. Therefore there exists infinitely many i with $\alpha_i = \chi_i = 1$ with probability 1, as required. \square

Lemma 14 (Approximation Lemma). Let π_1 and π_2 be policies, μ an environment and $h \geq H_t(1-\epsilon)$. Let $yx_{<t}$ be an arbitrary history and $yx_{t:t+h}^{\mu, \pi_i}$ be the future action/observation/reward triples when playing policy π_i . If $yx_{t:t+h}^{\pi_1, \mu} = yx_{t:t+h}^{\pi_2, \mu}$ then $|V_\mu^{\pi_1}(yx_{<t}) - V_\mu^{\pi_2}(yx_{<t})| < \epsilon$.

Recall that π_μ^* and π_ν^* are the optimal policies in environments μ and ν respectively (see Definition 5).

³ $\llbracket \text{expression} \rrbracket = 1$ if *expression* is true and 0 otherwise.

Lemma 15 (h-difference). *If $|V_\mu^{\pi_\mu^*}(yx_{<t}^{\pi,\mu}) - V_\mu^{\pi_\nu^*}(yx_{<t}^{\pi,\mu})| > \epsilon$ then μ is $H_t(1 - \epsilon)$ -different to ν on $yx^{\pi,\mu}$.*

We are now ready to prove the main theorem.

Proof (Theorem 10). Let π be the policy defined in Definition 9 and μ be the true (unknown) environment. Recall that $\nu_t = \mu_{i_t}$ with $i_t = \min\{i : \mu_i \text{ consistent with history } yx_{<t}^{\pi,\mu}\}$ is the first model consistent with the history $yx_{<t}^{\pi,\mu}$ at time t and is used by π when not exploring. First we claim there exists a $T \in \mathbb{N}$ and environment ν such that $\nu_t = \nu$ for all $t \geq T$. Two facts,

1. If μ_i is inconsistent with history $yx_{<t}^{\pi,\mu}$ then it is also inconsistent with $yx_{<t+h}^{\pi,\mu}$ for all $h \in \mathbb{N}$.
2. μ is consistent with $yx_{<t}^{\pi,\mu}$ for all π, t .

By 1) we have that the sequence i_1, i_2, i_3, \dots is monotone increasing. By 2) we have that the sequence is bounded by i with $\mu_i = \mu$. The claim follows since any bounded monotone sequence of natural numbers converges in finite time. Let $\nu := \nu_\infty$ be the environment to which $\nu_1, \nu_2, \nu_3, \dots$ converges to. Note that ν must be consistent with history $yx_{<t}^{\mu,\pi}$ for all t . We now show by contradiction that the optimal policy for ν is weakly asymptotically optimal in environment μ . Suppose it were not, then

$$\limsup_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^n \left[V_\mu^*(yx_{<t}^{\pi,\mu}) - V_\mu^{\pi_\nu^*}(yx_{<t}^{\pi,\mu}) \right] = \epsilon > 0. \quad (11)$$

Let $\alpha \in \mathcal{B}^\infty$ be defined by $\alpha_t := 1$ if and only if,

$$[V_\mu^*(yx_{<t}^{\pi,\mu}) - V_\mu^{\pi_\nu^*}(yx_{<t}^{\pi,\mu})] \geq \epsilon/4. \quad (12)$$

By Lemma 13 there exists (with probability one) an infinite sequence t_1, t_2, t_3, \dots for which $\chi_k = \alpha_k = 1$. Intuitively we should view time-step t_k as the start of an ‘‘exploration’’ phase where the agent explores for $\log t_k$ time-steps. Let $h := H_{t_k}(1 - \epsilon/4) = \lceil \log(\epsilon/4)/\log \gamma \rceil$, which importantly is independent of t_k (for geometric discounting). Since $\log t_k \rightarrow \infty$ we will assume that $\log t_k \geq h$ for all t_k . Therefore $\bar{\chi}_i = 1$ for all $i \in \bigcup_{k=1}^{\infty} [t_k, t_k + h]$. Therefore by the definition of π , $\pi(yx_{<i}^{\pi,\mu}) = \psi_i$ for $i \in \bigcup_{k=1}^{\infty} [t_k, t_k + h]$. By Lemma 15 and Equation (12), μ is h -different to ν on history $yx_{<t_k}^{\pi,\mu}$. This means that if there exists a k such that π plays according to the optimal policy for μ on all time-steps $t \in [t_k, t_k + h]$ then ν will be inconsistent with the history $yx_{1:t_k+h}^{\mu,\pi}$, which is a contradiction. We now show that π does indeed play according to the optimal policy for μ for all time-steps $t \in [t_k, t_k + h]$ for at least one k . Formally, we show the following holds with probability 1 for some k .

$$\psi_i \equiv \pi(yx_{<i}^{\pi,\mu}) = \pi_\mu^*(yx_{<i}^{\pi,\mu}), \text{ for all } i \in [t_k, t_k + h]. \quad (13)$$

Recall that $\psi \in \mathcal{B}^\infty$ where $\psi_i \in \mathcal{B}$ is identically independently distributed according to a Bernoulli($\frac{1}{2}$) distribution. Therefore $P(\psi_i = \pi_\mu^*(yx_{<i}^{\pi,\mu})) = \frac{1}{2}$. Therefore

$p := P(\psi_i = \pi_\mu^*(yx_{<i}^{\pi,\mu}) \forall i \in [t_k, t_k + h]) = \prod_{i=t_k}^{t_k+h} P(\psi_i = \pi_\mu^*(yx_{<i}^{\pi,\mu})) = 2^{-h-1} > 0$ and $P(\forall k \exists i \in [t_k, t_k + h] \text{ with } \psi_i \neq \pi_\mu^*(yx_{<i}^{\pi,\mu})) = \prod_{k=1}^{\infty} (1-p) = 0$. Therefore Equation (13) is satisfied for some k with probability 1 and so Equation (11) leads to a contradiction. Therefore

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^n \left[V_\mu^*(yx_{<t}^{\pi,\mu}) - V_\mu^{\pi_\nu^*}(yx_{<t}^{\pi,\mu}) \right] = 0. \quad (14)$$

We have shown that the optimal policy for ν has similar μ -values to the optimal policy for μ . We now show that π acts according to π_ν^* sufficiently often that it too has values close to those of the optimum policy for the true environment, μ . Let $\epsilon > 0$, $h := H_t(1-\epsilon)$ and $t \geq T$. If $\dot{\chi}_t^h = 0$ then by the definition of π and the approximation lemma we obtain

$$\left| V_\mu^{\pi_\nu^*}(yx_{<t}^{\pi,\mu}) - V_\mu^\pi(yx_{<t}^{\pi,\mu}) \right| < \epsilon. \quad (15)$$

Therefore

$$\limsup_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^n \left| V_\mu^{\pi_\nu^*}(yx_{<t}^{\pi,\mu}) - V_\mu^\pi(yx_{<t}^{\pi,\mu}) \right| \leq \limsup_{n \rightarrow \infty} \frac{1}{n} \left| \sum_{t=1}^{T-1} 1 + \sum_{t=T}^n [\dot{\chi}_t^h(1-\epsilon) + \epsilon] \right| \quad (16)$$

$$= \epsilon + (1-\epsilon) \limsup_{n \rightarrow \infty} \frac{1}{n} \#1(\dot{\chi}_{T:n}^h) \quad (17)$$

$$= \epsilon \quad (18)$$

where (16) follows since values are bounded in $[0, 1]$ and Equation (15). (17) follows by algebra. (18) by part 1 of Lemma 13. By sending $\epsilon \rightarrow 0$,

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^n \left[V_\mu^{\pi_\nu^*}(yx_{<t}^{\pi,\mu}) - V_\mu^\pi(yx_{<t}^{\pi,\mu}) \right] = 0. \quad (19)$$

Finally, combining Equations (14) and (19) gives the result. \square

We expect this theorem to generalise without great difficulty to discount functions satisfying $H_t(p) < c_p \log(t)$ for all p . There will be two key changes. First, extend the exploration time to some function $E(t)$ with $E(t) \in O(H_p(t))$ for all p . Second, modify the probability of exploration to ensure that Lemma 13 remains true.

5 Discussion

Summary. Part 1 of Theorem 7 shows that no policy can be strongly asymptotically optimal for any computable discount function. The key insight is that strong asymptotic optimality essentially implies exploration must eventually

cease. Once this occurs, the environment can change without the agent discovering the difference and the policy will no longer be optimal.

A weaker notion of asymptotic optimality, that a policy be optimal on average in the limit, turns out to be more interesting. Part 2 of Theorem 7 shows that no weak asymptotically optimal policy can be computable. We should not be surprised by this result. Any computable policy can be used to construct a computable environment in which that policy does very badly. Note that by computable here we mean deterministic and computable. There may be computable stochastic policies that are weakly asymptotically optimal, but we feel this is unlikely.

Part 3 of Theorem 7 shows that even weak asymptotically optimal policies need not exist if the discount function is sufficiently far-sighted. On the other hand, Theorem 10 shows that weak asymptotically optimal policies do exist for some discount rates, in particular, for the default geometric discounting. These non-trivial and slightly surprising result shows that choice of discount function is crucial to the existence of weak asymptotically optimal policies. Where weak asymptotically optimal policies do exist, they must explore infinitely often and in increasing contiguous bursts of exploration where the length of each burst is dependent on the discount function.

Consequences. It would appear that Theorem 7 is problematic for artificial general intelligence. We cannot construct incomputable policies, and so we cannot construct weak asymptotically optimal policies. However, this is not as problematic as it may seem. There are a number of reasonable counter arguments:

1. We may be able to make stochastically computable policies that are asymptotically optimal. If the existence of true random noise is assumed then this would be a good solution.
2. The counter-example environment constructed in part 2 of Theorem 7 is a single environment roughly as complex as the policy itself. Certainly, if the world were adversarial this would be a problem, but in general this appears not to be the case. On the other hand, if the environment is a learning agent itself, this could result in a complexity arms race without bound. There may exist a computable weak asymptotically optimal policy in some extremely large class of environments. For example, the algorithm of Section 4 is stochastically computable when the class of environments is recursively enumerable and contains only computable environments. A natural (and already quite large) class satisfying these properties is finite-state Markov decision processes with $\{0, 1\}$ -valued transition functions and rational-valued rewards.
3. While it is mathematically pleasant to use asymptotic behaviour to characterise optimal general intelligent behaviour, in practise we usually care about more immediate behaviour. We expect that results, and even (parameter free) formal definitions of intelligence satisfying this need will be challenging, but worthwhile.
4. Accept that even weak asymptotic optimality is too strong and find something weaker, but still useful.

Relation to AIXI. The policy defined in Section 4 is not equivalent to AIXI [7], which is also incomputable. However, if the computable environments in \mathcal{M} are ordered by complexity then it is likely the two will be quite similar. The key difference is the policy defined in this paper will continue to explore whereas it was shown in [14] that AIXI eventually ceases exploration in some environments and some histories. We believe, and a proof should not be too hard, that AIXI will become weakly asymptotically optimal if an exploration component is added similarly as in Section 4.

We now briefly compare the self-optimising property in [6] to strong asymptotic optimality. A policy π is self-optimising in a class \mathcal{M} if $\lim_{t \rightarrow \infty} [V_\mu^*(y\alpha_{<t}) - V_\mu^\pi(y\alpha_{<t})] = 0$ for any infinite history $y\alpha_{1:\infty}$ and for all $\mu \in \mathcal{M}$. This is similar to strong asymptotic optimality, but convergence must be on all histories, rather than the histories actually generated by π . This makes the self-optimising property a substantially stronger form of optimality than strong asymptotic optimality. It has been proven that if there exists self-optimising policy for a particular class, then AIXI is also self-optimising in that class [6].

It is possible to define a weak version of the self-optimising property by insisting that $\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^n [V_\mu^*(y\alpha_{<t}) - V_\mu^\pi(y\alpha_{<t})] = 0$ for all $y\alpha_{1:\infty}$ and all $\mu \in \mathcal{M}$. It can then be proven that the existence of a weak self-optimising policy would imply that AIXI were also weakly self-optimising. However, the policy defined in Section 4 cannot be modified to have the weak self-optimising property. It must be allowed to choose its actions itself. This is consistent with the work in [14] which shows that AIXI cannot be weakly asymptotically optimal, and so cannot be weak self-optimising either.

Discounting. Throughout this paper we have assumed rewards to be discounted according to a summable discount function. A very natural alternative to discounting, suggested in [11], is to restrict interest to environments satisfying $\sum_{k=1}^{\infty} r_k^{\mu,\pi} \leq 1$. Now the goal of the agent is simply to maximise summed rewards. In this setting it is easy to see that the positive theorem is lost while all negative ones still hold! This is unfortunate, as discounting presents a major philosophical challenge. How to choose a discount function?

Assumptions/Limitations. Assumption I ensures that \mathcal{Y} and \mathcal{O} are finite. All negative results go through for countable \mathcal{Y} and \mathcal{O} . The optimal policy of Section 4 may not generalise to countable \mathcal{Y} . We have also assumed bounded reward and discrete time. The first seems reasonable while the second allows for substantially easier analysis. Additionally we have only considered deterministic computable environments. The stochastic case is unquestionably interesting. We invoked Church thesis to assert that computable stochastic environments are essentially the largest class of interesting environments.

Many of our Theorems are only applicable to computable discount functions. All well-known discount function in use today are computable. However [7] has suggested $\gamma_t = 2^{-K(t)}$, where $K(t)$ is the (incomputable) prefix Kolmogorov complexity of t , may have nice theoretical properties.

Acknowledgements. We thank Laurent Orseau, Wen Shao and reviewers for valuable feedback on earlier drafts and the Australian Research Council for support under grant DP0988049.

References

- [1] Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47, 235–256 (2002)
- [2] Berry, D.A., Fristedt, B.: *Bandit Problems: Sequential Allocation of Experiments*. Chapman and Hall, London (1985)
- [3] Diaconis, P., Freedman, D.: On inconsistent Bayes estimates of location. *The Annals of Statistics* 14(1), 68–87 (1986)
- [4] Diaconis, P., Freedman, D.: On the consistency of Bayes estimates. *The Annals of Statistics* 14(1), 1–26 (1986)
- [5] Frederick, S., Oewensteine, G.L., O'Donoghue, T.: Time discounting and time preference: A critical review. *Journal of Economic Literature* 40(2) (2002)
- [6] Hutter, M.: Self-optimizing and Pareto-optimal policies in general environments based on Bayes-mixtures. In: Kivinen, J., Sloan, R.H. (eds.) *COLT 2002. LNCS (LNAI)*, vol. 2375, pp. 364–379. Springer, Heidelberg (2002)
- [7] Hutter, M.: *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin (2004)
- [8] Hutter, M., Muchnik, A.A.: On semimeasures predicting Martin-Löf random sequences. *Theoretical Computer Science* 382(3), 247–261 (2007)
- [9] Lattimore, T., Hutter, M.: Time consistent discounting. In: Kivinen, J., Szepesvári, C., Ukkonen, E., Zeugmann, T. (eds.) *Algorithmic Learning Theory. LNCS*, vol. 6925, pp. 384–398. Springer, Heidelberg (2011)
- [10] Legg, S.: Is there an elegant universal theory of prediction? In: Balcázar, J.L., Long, P.M., Stephan, F. (eds.) *ALT 2006. LNCS (LNAI)*, vol. 4264, pp. 274–287. Springer, Heidelberg (2006)
- [11] Legg, S., Hutter, M.: Universal intelligence: A definition of machine intelligence. *Minds & Machines* 17(4), 391–444 (2007)
- [12] Li, M., Vitanyi, P.: *An Introduction to Kolmogorov Complexity and Its Applications*, 3rd edn. Springer, Heidelberg (2008)
- [13] Norvig, P., Russell, S.J.: *Artificial Intelligence: A Modern Approach*, 2nd edn. Prentice Hall Series in Artificial Intelligence. Prentice Hall, Englewood Cliffs (2003)
- [14] Orseau, L.: Optimality issues of universal greedy agents with static priors. In: Hutter, M., Stephan, F., Vovk, V., Zeugmann, T. (eds.) *ALT 2010. LNCS*, vol. 6331, pp. 345–359. Springer, Heidelberg (2010)
- [15] Strehl, A.L., Littman, M.L.: An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences* 74(8), 1309–1331 (2008)

Time Consistent Discounting

Tor Lattimore¹ and Marcus Hutter^{1,2}

¹ Research School of Computer Science
Australian National University

² ETH Zürich
`{tor.lattimore,marcus.hutter}@anu.edu.au`

Abstract. A possibly immortal agent tries to maximise its summed discounted rewards over time, where discounting is used to avoid infinite utilities and encourage the agent to value current rewards more than future ones. Some commonly used discount functions lead to time-inconsistent behavior where the agent changes its plan over time. These inconsistencies can lead to very poor behavior. We generalise the usual discounted utility model to one where the discount function changes with the age of the agent. We then give a simple characterisation of time-(in)consistent discount functions and show the existence of a rational policy for an agent that knows its discount function is time-inconsistent.

Keywords: Rational agents, sequential decision theory, general discounting, time-consistency, game theory.

1 Introduction

The goal of an agent is to maximise its expected utility; but how do we measure utility? One method is to assign an instantaneous reward to particular events, such as having a good meal, or a pleasant walk. It would be natural to measure the utility of a plan (policy) by simply summing the expected instantaneous rewards, but for immortal agents this may lead to infinite utility and also assumes rewards are equally valuable irrespective of the time at which they are received.

One solution, the discounted utility (DU) model introduced by Samuelson in [12], is to take a weighted sum of the rewards with earlier rewards usually valued more than later ones.

There have been a number of criticisms of the DU model, which we will not discuss. For an excellent summary, see [1]. Despite the criticisms, the DU model is widely used in both economics and computer science.

A discount function is time-inconsistent if plans chosen to maximise expected discounted utility change over time. For example, many people express a preference for \$110 in 31 days over \$100 in 30 days, but reverse that preference 30 days later when given a choice between \$110 tomorrow or \$100 today [4]. This behavior can be caused by a rational agent with a time-inconsistent discount function.

Unfortunately, time-inconsistent discount functions can lead to extremely bad behavior and so it becomes important to ask what discount functions are time-inconsistent.

Previous work has focussed on a continuous model where agents can take actions at any time in a continuous time-space. We consider a discrete model where agents act in finite time-steps. In general this is not a limitation since any continuous environment can be approximated arbitrarily well by a discrete one. The discrete setting has the advantage of easier analysis, which allows us to consider a very general setup where environments are arbitrary finite or infinite Markov decision processes.

Traditionally, the DU model has assumed a sliding discount function. Formally, a sequence of instantaneous utilities (rewards) $R = (r_k, r_{k+1}, r_{k+2}, \dots)$ starting at time k , is given utility equal to $\sum_{t=k}^{\infty} d_{t-k} r_t$ where $d \in [0, 1]^{\infty}$. We generalise this model as in [6] by allowing the discount function to depend on the age of the agent. The new utility is given by $\sum_{t=k}^{\infty} d_t^k r_t$. This generalisation is consistent with how some agents tend to behave; for example, humans becoming temporally less myopic as they grow older.

Strotz [13] showed that the only time-consistent sliding discount function is geometric discounting. We extend this result to a full characterisation of time-consistent discount functions where the discount function is permitted to change over time. We also show that discounting functions that are “nearly” time-consistent give rise to low regret in the anticipated future changes of the policy over time.

Another important question is what policy should be adopted by an agent that knows it is time-inconsistent. For example, if it knows it will become temporarily myopic in the near future then it may benefit from paying a price to pre-commit to following a particular policy. A number of authors have examined this question in special continuous cases, including [3, 10, 11, 13]. We modify their results to our general, but discrete, setting using game theory.

The paper is structured as follows. First the required notation is introduced (Section 2). Example discount functions and the consequences of time-inconsistent discount functions are then presented (Section 3). We next state and prove the main theorems, the complete classification of discount functions and the continuity result (Section 4). The game theoretic view of what an agent *should* do if it knows its discount function is changing is analyzed (Section 5). Finally we offer some discussion and concluding remarks (Section 6).

2 Notation and Problem Setup

The general reinforcement learning (RL) setup involves an agent interacting sequentially with an environment where in each time-step t the agent chooses some action $a_t \in \mathcal{A}$, whereupon it receives a reward $r_t \in \mathcal{R} \subseteq \mathbb{R}$ and observation $o_t \in \mathcal{O}$. The environment can be formally defined as a probability distribution μ where $\mu(r_t o_t | a_1 r_1 o_1 a_2 r_2 o_2 \dots a_{t-1} r_{t-1} o_{t-1} a_t)$ is the probability of receiving reward r_t and observation o_t having taken action a_t after history $h_{<t} := a_1 r_1 o_1 \dots a_{t-1} r_{t-1} o_{t-1}$. For convenience, we assume that for a given history $h_{<t}$ and action a_t , that r_t is fixed (not stochastic). We denote the set of all finite histories $\mathcal{H} := (\mathcal{A} \times \mathcal{R} \times \mathcal{O})^*$ and write $h_{1:t}$ to be a history of length t , $h_{<t}$ to

be a history of length $t - 1$. a_k , r_k , and o_k are the k th action/reward/observation tuple of history h and will be used without explicitly redefining them (there will always be only one history “in context”).

A deterministic environment (where every value of $\mu(\cdot)$ is either 1 or 0) can be represented as a graph with edges for actions, rewards of each action attached to the corresponding edge, and observations in the nodes. For example, the deterministic environment on the right represents an environment where either pizza or pasta must be chosen at each time-step (evening). An action leading to an upper node is `eat pizza` while the ones leading to a lower node are `eat pasta`. The rewards are for a consumer who prefers pizza to pasta, but dislikes having the same food twice in a row. The starting node is marked as S . This example, along with all those for the remainder of this paper, does not require observations.

The following assumption is required for clean results, but may be relaxed if an ϵ of slop is permitted in some results.

Assumption 1. We assume that \mathcal{A} and \mathcal{O} are finite and that $\mathcal{R} = [0, 1]$.

Definition 1 (Policy). A policy is a mapping $\pi : \mathcal{H} \rightarrow \mathcal{A}$ giving an action for each history.

Given policy π and history $h_{1:t}$ and $s \leq t$ then the probability of reaching history $h_{1:t}$ when starting from history $h_{<s}$ is $P(h_{s:t}|h_{<s}, \pi)$ which is defined by,

$$P(h_{s:t}|h_{<s}, \pi) := \prod_{k=s}^t \mu(r_k o_k | h_{<k} \pi(h_{<k})). \quad (1)$$

If $s = 1$ then we abbreviate and write $P(h_{1:t}|\pi) := P(h_{1:t}|h_{<1}, \pi)$.

Definition 2 (Expected Rewards). When applying policy π starting from history $h_{<t}$, the expected sequence of rewards $\mathbf{R}^\pi(h_{<t}) \in [0, 1]^\infty$, is defined by

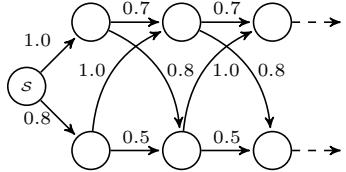
$$R^\pi(h_{<t})_k := \sum_{h_{t:k}} P(h_{t:k}|h_{<t}, \pi) r_k.$$

If $k < t$ then $R^\pi(h_{<t})_k := 0$.

Note while the set of all possible $h_{t:k} \in (\mathcal{A} \times \mathcal{R} \times \mathcal{O})^{k-t+1}$ is uncountable due to the reward term, we sum only over the possible rewards which are determined by the action and previous history, and so this is actually a finite sum.

Definition 3 (Discount Vector). A discount vector $\mathbf{d}^k \in [0, 1]^\infty$ is a vector $[d_1^k, d_2^k, d_3^k, \dots]$ satisfying $d_t^k > 0$ for at least one $t \geq k$.

The apparently superfluous superscript k will be useful later when we allow the discount vector to change with time. We do *not* insist that the discount vector be summable, $\sum_{t=k}^{\infty} d_t^k < \infty$.



Definition 4 (Expected Values). *The expected discounted reward (or utility or value) when using policy π starting in history $h_{<t}$ and discount vector \mathbf{d}^k is*

$$V_{\mathbf{d}^k}^\pi(h_{<t}) := \mathbf{R}^\pi(h_{<t}) \cdot \mathbf{d}^k := \sum_{i=1}^{\infty} R^\pi(h_{<t})_i d_i^k = \sum_{i=t}^{\infty} R^\pi(h_{<t})_i d_i^k.$$

The sum can be taken to start from t since $R^\pi(h_{<t})_i = 0$ for $i < t$. This means that the value of d_t^k for $t < k$ is unimportant, and never will be for any result in this paper. As the scalar product is linear, a scaling of a discount vector has no affect on the ordering of the policies. Formally, if $V_{\mathbf{d}^k}^{\pi_1}(h_{<t}) \geq V_{\mathbf{d}^k}^{\pi_2}(h_{<t})$ then $V_{\alpha \mathbf{d}^k}^{\pi_1}(h_{<t}) \geq V_{\alpha \mathbf{d}^k}^{\pi_2}(h_{<t})$ for all $\alpha > 0$.

Definition 5 (Optimal Policy/Value). *In general, our agent will try to choose a policy $\pi_{\mathbf{d}^k}^*$ to maximise $V_{\mathbf{d}^k}^\pi(h_{<t})$. This is defined as follows.*

$$\begin{aligned} \pi_{\mathbf{d}^k}^*(h_{<t}) &:= \arg \max_{\pi} V_{\mathbf{d}^k}^\pi(h_{<t}), & \mathbf{R}_{\mathbf{d}^k}^*(h_{<t}) &:= \mathbf{R}^{\pi_{\mathbf{d}^k}^*}(h_{<t}), \\ V_{\mathbf{d}^k}^*(h_{<t}) &:= V_{\mathbf{d}^k}^{\pi_{\mathbf{d}^k}^*}(h_{<t}). \end{aligned}$$

If multiple policies are optimal then $\pi_{\mathbf{d}^k}^*$ is chosen using some arbitrary rule. Unfortunately, $\pi_{\mathbf{d}^k}^*$ need not exist without one further assumption.

Assumption 2. *For all π and $k \geq 1$, $\lim_{t \rightarrow \infty} \sum_{h_{<t}} P(h_{<t}|\pi) V_{\mathbf{d}^k}^\pi(h_{<t}) = 0$.*

Assumption 2 appears somewhat arbitrary. We consider:

1. For summable \mathbf{d}^k the assumption is true for all environments. With the exception of hyperbolic discounting, all frequently used discount vectors are summable.
2. For non-summable discount vectors \mathbf{d}^k the assumption implies a restriction on the possible environments. In particular, they must return asymptotically lower rewards in expectation. This restriction is necessary to guarantee the existence of the value function.

From now on, including in theorem statements, we only consider environments/discount vectors satisfying Assumptions 1 and 2. The following theorem then guarantees the existence of $\pi_{\mathbf{d}^k}^*$.

Theorem 6 (Existence of Optimal Policy). *$\pi_{\mathbf{d}^k}^*$ exists for any environment and discount vector \mathbf{d}^k satisfying Assumptions 1 and 2.*

The proof of the existence theorem is in the appendix.

An agent can use a different discount vector \mathbf{d}^k for each time k . This motivates the following definition.

Definition 7 (Discount Matrix). *A discount matrix \mathbf{d} is a $\infty \times \infty$ matrix with discount vector \mathbf{d}^k for the k th column.*

It is important that we distinguish between a discount matrix \mathbf{d} (written bold), a discount vector \mathbf{d}^k (bold and italics), and a particular value in a discount vector d_t^k (just italics).

Definition 8 (Sliding Discount Matrix). A discount matrix \mathbf{d} is sliding if $d_{k+t}^k = d_{t+1}^1$ for all $k, t \geq 1$.

Definition 9 (Mixed Policy). The mixed policy is the policy where at each time step t , the agent acts according to the possibly different policy $\pi_{\mathbf{d}^t}^*$.

$$\pi_{\mathbf{d}}(h_{<t}) := \pi_{\mathbf{d}^t}^*(h_{<t}) \quad \mathbf{R}_{\mathbf{d}}(h_{<t}) := \mathbf{R}^{\pi_{\mathbf{d}}}(h_{<t}).$$

We do not denote the mixed policy by $\pi_{\mathbf{d}}^*$ as it is arguably not optimal as discussed in Section 5. While non-unique optimal policies $\pi_{\mathbf{d}^k}^*$ at least result in equal discounted utilities, this is *not* the case for $\pi_{\mathbf{d}}$. All theorems are proved with respect to any choice $\pi_{\mathbf{d}}$.

Definition 10 (Time Consistency). A discount matrix \mathbf{d} is time consistent if and only if for all environments $\pi_{\mathbf{d}^k}^*(h_{<t}) = \pi_{\mathbf{d}^t}^*(h_{<t})$, for all $h_{<t}$ where $t \geq k$.

This means that a time-consistent agent taking action $\pi_{\mathbf{d}^t}^*(h_{<t})$ at each time t will not change its plans. On the other hand, a time-inconsistent agent may at time 1 intend to take action a should it reach history $h_{<t}$ ($\pi_{\mathbf{d}^0}^*(h_{<t}) = a$). However upon reaching $h_{<t}$, it need not be true that $\pi_{\mathbf{d}^t}^*(h_{<t}) = a$.

3 Examples

In this section we review a number of common discount matrices and give an example where a time-inconsistent discount matrix causes very bad behavior.

Constant Horizon. Constant horizon discounting is where the agent only cares about the future up to H time-steps away, defined by $d_t^k = \mathbb{I}[t - k < H]$.¹ Shortly we will see that the constant horizon discount matrix can lead to very bad behavior in some environments.

Fixed Lifetime. Fixed lifetime discounting is where an agent knows it will not care about any rewards past time-step m , defined by $d_t^k = \mathbb{I}[t < m]$. Unlike the constant horizon method, a fixed lifetime discount matrix is time-consistent. Unfortunately it requires you to know the lifetime of the agent beforehand and also makes asymptotic analysis impossible.

Hyperbolic. $d_t^k = 1/(1 + \kappa(t - k))$. The parameter κ determines how farsighted the agent is with smaller values leading to more farsighted agents. Hyperbolic discounting is often used in economics with some experimental studies explaining human time-inconsistent behavior by suggesting that we discount hyperbolically [4]. The hyperbolic discount matrix is not summable, so may be replaced by the following (similar to [5]), which has similar properties for β close to 1.

$$d_t^k = 1/(1 + \kappa(t - k))^\beta \text{ with } \beta > 1.$$

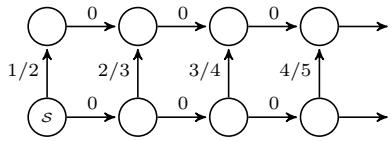
Geometric. $d_t^k = \gamma^t$ with $\gamma \in (0, 1)$. Geometric discounting is the most commonly used discount matrix. Philosophically it can be justified by assuming an

¹ $\mathbb{I}[expr] = 1$ if $expr$ is true and 0 otherwise.

agent will die (and not care about the future after death) with probability $1 - \gamma$ at each time-step. Another justification for geometric discount is its analytic simplicity - it is summable and leads to time-consistent policies. It also models fixed interest rates.

No Discounting. $d_t^k = 1$, for all k, t . [8] and [7] point out that discounting future rewards via an explicit discount matrix is unnecessary since the environment can capture both temporal preferences for early (or late) consumption, as well as the risk associated with delaying consumption. Of course, this “discount matrix” is not summable, but can be made to work by insisting that all environments satisfy Assumption 2. This approach is elegant in the sense that it eliminates the need for a discount matrix, essentially admitting far more complex preferences regarding inter-temporal rewards than a discount matrix allows. On the other hand, a discount matrix gives the “controller” an explicit way to adjust the myopia of the agent.

To illustrate the potential consequences of time-inconsistent discount matrices we consider the policies of several agents acting in the following environment. Let agent A use a constant horizon discount matrix with $H = 2$ and agent B a geometric discount matrix with some discount rate γ .



In the first time-step agent A prefers to move right with the intention of moving up in the second time-step for a reward of $2/3$. However, once in second time-step, it will change its plan by moving right again. This continues indefinitely, so agent A will always delay moving up and receives zero reward forever.

Agent B acts very differently. Let π_t be the policy in which the agent moves right until time-step t , then up and right indefinitely. $V_{d^k}^{\pi_t}(h_{<1}) = \gamma^t \frac{(t+1)}{(t+2)}$. This value does not depend on k and so the agent will move right until $t = \arg \max \left\{ \gamma^t \frac{(t+1)}{t+2} \right\} < \infty$ when it will move up and receive a reward.

The actions of agent A are an example of the worst possible behavior arising from time-inconsistent discounting. Nevertheless, agents with a constant horizon discount matrix are used in all kinds of problems. In particular, agents in zero sum games where fixed depth mini-max searches are common. In practise, serious time-inconsistent behavior for game-playing agents seems rare, presumably because most strategic games don't have a reward structure similar to the example above.

4 Theorems

The main theorem of this paper is a complete characterisation of time consistent discount matrices.

Theorem 11 (Characterisation). *Let \mathbf{d} be a discount matrix, then the following are equivalent.*

1. \mathbf{d} is time-consistent (Definition 7.1)

2. For each k there exists an $\alpha_k \in \mathbb{R}$ such that $d_t^k = \alpha_k d_t^1$ for all $t \geq k \in \mathbb{N}$.

Recall that a discount matrix is sliding if $d_t^k = d_{t-k+1}^1$. Theorem 11 can be used to show that if a sliding discount matrix is used as in [13] then the only time-consistent discount matrix is geometric. Let \mathbf{d} be a time-consistent sliding discount matrix. By Theorem 11 and the definition of sliding, $\alpha_1 d_{t+1}^1 = d_{t+1}^2 = d_t^1$. Therefore $\frac{1}{\alpha_1} d_2^1 = d_1^1$ and $d_3^1 = \frac{1}{\alpha_1} d_2^1 = \left(\frac{1}{\alpha_1}\right)^2 d_1^1$ and similarly, $d_t^1 = \left(\frac{1}{\alpha_1}\right)^{t-1} d_1^1 \propto \gamma^t$ with $\gamma = 1/\alpha_1$, which is geometric discounting. This is the analogue to the results of [13] converted to our setting.

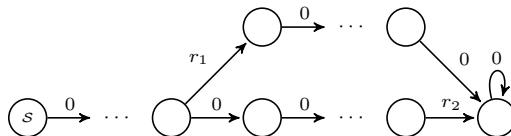
The theorem can also be used to construct time-consistent discount rates. Let \mathbf{d}^1 be a discount vector, then the discount matrix defined by $d_t^k := d_t^1$ for all $t \geq k$ will always be time-consistent, for example, the *fixed lifetime* discount matrix with $d_t^k = 1$ if $t \leq H$ for some horizon H . Indeed, all time-consistent discount rates can be constructed in this way (up to scaling).

Proof (Theorem 11). 2 \implies 1: This direction follows easily from linearity of the scalar product.

$$\begin{aligned} \pi_{\mathbf{d}^k}^*(h_{<t}) &\equiv \arg \max_{\pi} V_{\mathbf{d}^k}^\pi(h_{<t}) \equiv \arg \max_{\pi} \mathbf{R}^\pi(h_{<t}) \cdot \mathbf{d}^k = \arg \max_{\pi} \mathbf{R}^\pi(h_{<t}) \cdot \alpha_k \mathbf{d}^1 \\ &= \arg \max_{\pi} \alpha_k \mathbf{R}^\pi(h_{<t}) \cdot \mathbf{d}^1 = \arg \max_{\pi} \mathbf{R}^\pi(h_{<t}) \cdot \mathbf{d}^1 \equiv \pi_{\mathbf{d}^1}^*(h_{<t}) \end{aligned} \quad (2)$$

as required. The last equality of (2) follows from the assumption that $d_t^k = \alpha_k d_t^1$ for all $t \geq k$ and because $\mathbf{R}^\pi(h_{<t})_i = 0$ for all $i < t$.

1 \implies 2: Let \mathbf{d}^0 and \mathbf{d}^k be the discount vectors used at times 0 and k respectively. Now let $k \leq t_1 < t_2 < \dots$ and consider the deterministic environment below where the agent has a choice between earning reward r_1 at time t_1 or r_2 at time t_2 . In this environment there are only two policies, π_1 and π_2 , where $\mathbf{R}^{\pi_1}(h_{<k}) = r_1 \mathbf{e}_{t_1}$ and $\mathbf{R}^{\pi_2}(h_{<k}) = r_2 \mathbf{e}_{t_2}$ with \mathbf{e}_i the infinite vector with all components zero except the i th, which is 1.



Since \mathbf{d} is time-consistent, for all $r_1, r_2 \in \mathcal{R}$ and $k \in \mathbb{N}$ we have:

$$\arg \max_{\pi} V_{\mathbf{d}^1}^\pi(h_{<k}) \equiv \arg \max_{\pi} \mathbf{R}^\pi(h_{<k}) \cdot \mathbf{d}^1 \quad (3)$$

$$= \arg \max_{\pi} \mathbf{R}^\pi(h_{<k}) \cdot \mathbf{d}^k \equiv \arg \max_{\pi} V_{\mathbf{d}^k}^\pi(h_{<k}). \quad (4)$$

Now $V_{\mathbf{d}^k}^{\pi_1} \geq V_{\mathbf{d}^k}^{\pi_2}$ if and only if $\mathbf{d}^k \cdot [\mathbf{R}^{\pi_1}(h_{<k}) - \mathbf{R}^{\pi_2}(h_{<k})] = [d_{t_1}^k, d_{t_2}^k] \cdot [r_1, -r_2] \geq 0$. Therefore we have that,

$$[d_{t_1}^1, d_{t_2}^1] \cdot [r_1, -r_2] \geq 0 \Leftrightarrow [d_{t_1}^k, d_{t_2}^k] \cdot [r_1, -r_2] \geq 0. \quad (5)$$

Letting $\cos \theta_k$ be the cosine of the angle between $[d_{t_1}^k, d_{t_2}^k]$ and $[r_1, -r_2]$ then Equation (5) becomes $\cos \theta_0 \geq 0 \Leftrightarrow \cos \theta_k \geq 0$. Choosing $[r_1, -r_2] \propto [d_{t_2}^1, -d_{t_1}^1]$ implies that $\cos \theta_0 = 0$ and so $\cos \theta_k = 0$. Therefore there exists $\alpha_k \in \mathbb{R}$ such that

$$[d_{t_1}^k, d_{t_2}^k] = \alpha_k [d_{t_1}^1, d_{t_2}^1]. \quad (6)$$

Let $k \leq t_1 < t_2 < t_3 < \dots$ be a sequence for which $d_{t_i}^1 > 0$. By the previous argument we have that, $[d_{t_i}^k, d_{t_{i+1}}^k] = \alpha_k [d_{t_i}^1, d_{t_{i+1}}^1]$ and $[d_{t_{i+1}}^k, d_{t_{i+2}}^k] = \tilde{\alpha}_k [d_{t_{i+1}}^1, d_{t_{i+2}}^1]$. Therefore $\alpha_k = \tilde{\alpha}_k$, and by induction, $d_{t_i}^k = \alpha_k d_{t_i}^1$ for all i . Now if $t \geq k$ and $d_t^1 = 0$ then $d_t^k = 0$ by equation (6). By symmetry, $d_t^k = 0 \implies d_t^1 = 0$. Therefore $d_t^k = \alpha_k d_t^1$ for all $t \geq k$ as required. \square

In Section 3 we saw an example where time-inconsistency led to very bad behavior. The discount matrix causing this was very time-inconsistent. Is it possible that an agent using a “nearly” time-consistent discount matrix can exhibit similar bad behavior? For example, could rounding errors when using a geometric discount matrix seriously affect the agent’s behavior? The following Theorem shows that this is not possible. First we require a measure of the cost of time-inconsistent behavior. The regret experienced by the agent at time zero from following policy π_d rather than $\pi_{d^1}^*$ is $V_{d^1}^*(h_{<1}) - V_{d^1}^{\pi_d}(h_{<1})$. We also need a distance measure on the space of discount vectors.

Definition 12 (Distance Measure). Let $\mathbf{d}^k, \mathbf{d}^j$ be discount vectors then define a distance measure D by

$$D(\mathbf{d}^k, \mathbf{d}^j) := \sum_{i=\max\{k,j\}}^{\infty} |d_i^k - d_i^j|.$$

Note that this is almost the taxicab metric, but the sum is restricted to $i \geq \max\{k, j\}$.

Theorem 13 (Continuity). Suppose $\epsilon \geq 0$ and $D_{k,j} := D(\mathbf{d}^k, \mathbf{d}^j)$ then

$$V_{d^1}^*(h_{<1}) - V_{d^1}^{\pi_d}(h_{<1}) \leq \epsilon + D_{1,t} + \sum_{k=1}^{t-1} D_{k,k+1}$$

with $t = \min \left\{ t : \sum_{h_{<t}} P(h_{<t} | \pi_{d^1}^*) V_{d^1}^*(h_{<t}) \leq \epsilon \right\}$, which for $\epsilon > 0$ is guaranteed to exist by Assumption 2.

Theorem 13 implies that the regret of the agent at time zero in its future time-inconsistent actions is bounded by the sum of the differences between the discount vectors used at different times. If these differences are small then the regret is also small. For example, it implies that small perturbations (such as rounding errors) in a time-consistent discount matrix lead to minimal bad behavior.

The proof is omitted due to limitations in space. It relies on proving the result for finite horizon environments and showing that this extends to the infinite case by using the horizon, t , after which the actions of the agent are no longer important. The bound in Theorem 13 is tight in the following sense.

Theorem 14. For $\delta > 0$ and $t \in \mathbb{N}$ and any sufficiently small $\epsilon > 0$ there exists an environment and discount matrix such that

$$(t-2)(1-\epsilon)\delta < V_{\mathbf{d}^1}^*(h_{<1}) - V_{\mathbf{d}^1}^{\pi_{\mathbf{d}}}(h_{<1}) < (t+1)\delta$$

$$\equiv D_{1,t} + \sum_{i=1}^{t-1} D_{i,i+1}$$

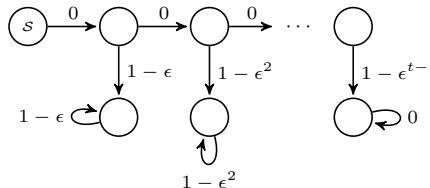
where $t = \min \left\{ t : \sum_{h_{<t}} P(h_{<t} | \pi_{\mathbf{d}^1}^*) V_{\mathbf{d}^1}^*(h_{<t}) = 0 \right\} < \infty$ and where $D(\mathbf{d}^k, \mathbf{d}^j) \equiv D_{k,j} = \delta$ for all k, j .

Note that t in the statement above is the same as that in the statement of Theorem 13. Theorem 14 shows that there exists a discount matrix, environment and $\epsilon > 0$ where the regret due to time-inconsistency is nearly equal to the bound given by Theorem 13.

Proof (Theorem 14). Define \mathbf{d} by

$$d_i^k = \begin{cases} \delta & \text{if } k < i < t \\ 0 & \text{otherwise} \end{cases}$$

Observe that $D(\mathbf{d}^k, \mathbf{d}^j) = \delta$ for all $k < j < t$ since $d_i^j = d_i^k$ for all i except $i = j$. Now consider the environment below.



For sufficiently small ϵ , the agent at time zero will plan to move right and then down leading to $R_{\pi^1}^*(h_{<1}) = [0, 1 - \epsilon, 1 - \epsilon, \dots]$ and $V_{\pi^1}^*(h_{<1}) = (t-1)\delta(1-\epsilon)$.

To compute \mathbf{R}_d note that $d_k^k = 0$ for all k . Therefore the agent in time-step k doesn't care about the next instantaneous reward, so prefers to move right with the intention of moving down in the next time-step when the rewards are slightly better. This leads to $\mathbf{R}_d(h_{<1}) = [0, 0, \dots, 1 - e^{t-1}, 0, 0, \dots]$. Therefore,

$$V_{d^1}^*(h_{<1}) - V_{d^1}^{\pi_d}(h_{<1}) = (t-1)\delta(1-\epsilon) - (1-\epsilon^{t-1})\delta \geq (t-2)\delta(1-\epsilon)$$

as required.

5 Game Theoretic Approach

What should an agent do if it knows it is time inconsistent? One option is to treat its future selves as “opponents” in an extensive game. The game has one player per time-step who chooses the action for that time-step only. At the end of the game the agent will have received a reward sequence $\mathbf{r} \in \mathcal{R}^\infty$. The utility given to the k th player is then $\mathbf{r} \cdot \mathbf{d}^k$. So each player in this game wishes to maximise the discounted reward with respect to a different discounting vector.

For example, let $\mathbf{d}^1 = [2, 1, 2, 0, 0, \dots]$ and $\mathbf{d}^2 = [* , 3, 1, 0, 0, \dots]$ and consider the environment on the right. Initially, the agent has two choices. It can either move down to guarantee a reward sequence of $\mathbf{r} = [4, 0, 0, \dots]$ which has utility of $\mathbf{d}^1 \cdot [4, 0, 0, \dots] = 8$ or it can move right in which case it will receive a reward sequence of either $\mathbf{r}' = [1, 3, 0, 0, \dots]$ with utility 5 or $\mathbf{r}'' = [1, 1, 3, 0, 0, \dots]$ with utility 9. Which of these two reward sequences it receives is determined by the action taken in the second time-step. However this action is chosen to maximise utility with respect to discount sequence \mathbf{d}^2 and $\mathbf{d}^2 \cdot \mathbf{r}' > \mathbf{d}^1 \cdot \mathbf{r}''$. This means that if at time 1 the agent chooses to move right, the final reward sequence will be $[1, 3, 0, 0, \dots]$ and the final utility with respect to \mathbf{d}^1 will be 5. Therefore the rational thing to do in time-step 1 is to move down immediately for a utility of 8.

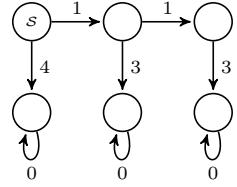
The technique above is known as backwards induction which is used to find sub-game perfect equilibria in finite extensive games. A variant of Kuhn's theorem proves that backwards induction can be used to find such equilibria in finite extensive games [9]. For arbitrary extensive games (possibly infinite) a sub-game perfect equilibrium need not exist, but we prove a theorem for our particular class of infinite games.

A sub-game perfect equilibrium policy is one the players could agree to play, and subsequently have no incentive to renege on their agreement during play. It isn't always philosophically clear that a sub-game perfect equilibrium policy *should* be played. For a deeper discussion, including a number of good examples, see [9].

Definition 15 (Sub-game Perfect Equilibria). A policy $\pi_{\mathbf{d}}^*$ is a sub-game perfect equilibrium policy if and only if for each t $V_{\mathbf{d}^t}^{\pi_{\mathbf{d}}^*}(h_{<t}) \geq V_{\tilde{\mathbf{d}}^t}^{\tilde{\pi}}(h_{<t})$, for all $h_{<t}$, where $\tilde{\pi}$ is any policy satisfying $\tilde{\pi}(h_{<i}) = \pi_{\mathbf{d}}^*(h_{<i}) \forall h_{<i}$ where $i \neq t$.

Theorem 16 (Existence of Sub-game Perfect Equilibrium Policy). For all environments and discount matrices \mathbf{d} satisfying Assumptions 1 and 2 there exists at least one sub-game perfect equilibrium policy $\pi_{\mathbf{d}}^*$.

Many results in the literature of game theory almost prove this theorem. Our setting is more difficult than most because we have countably many players (one for each time-step) and exogenous uncertainty. Fortunately, it is made easier by the very particular conditions on the preferences of players for rewards that occur late in the game (Assumption 2). The closest related work appears to be that of Drew Fudenberg in [2], but our proof (see appendix) is very different. The proof idea is to consider a sequence of environments identical to the original environment but with an increasing bounded horizon after which reward is zero. By Kuhn's Theorem [9] a sub-game perfect equilibrium policy must exist in each of these finite games. However the space of policies is compact (Lemma 21) and so this sequence of sub-game perfect equilibrium policies contains a convergent



sub-sequence converging to policy π . It is not then hard to show that π is a sub-game perfect equilibrium policy in the original environment.

Proof (Theorem 16). Add an action a^{death} to \mathcal{A} and μ such that if a^{death} is taken at any time in $h_{<t}$ then μ returns zero reward. Essentially, once in the agent takes action a^{death} , the agent receives zero reward forever. Now if $\pi_{\mathbf{d}}^*$ is a sub-game perfect equilibrium policy in this modified environment then it is a sub-game perfect equilibrium policy in the original one.

For each $t \in \mathbb{N}$ choose π_t to be a sub-game perfect equilibrium policy in the further modified environment obtained by setting $r_i = 0$ if $i > t$. That is, the environment which gives zero reward always after time t . We can assume without loss of generality that $\pi_t(h_{<k}) = a^{death}$ for all $k \geq t$. Since Π is compact, the sequence π_1, π_2, \dots has a convergent subsequence $\pi_{t_1}, \pi_{t_2}, \dots$ converging to π and satisfying

1. $\pi_{t_i}(h_{<k}) = \pi(h_{<k})$, for all $h_{<k}$ where $k \leq i$.
2. π_{t_i} is a sub-game perfect equilibrium policy in the modified environment with reward $r_k = 0$ if $k > t_i$.
3. $\pi_{t_i}(h_{<t_i}) = a^{death}$.

We write $\tilde{V}^{\pi_{t_i}}$ for the value function in the modified environment. It is now shown that π is a sub-game perfect equilibrium policy in the original environment. Fix a $t \in \mathbb{N}$ and let $\tilde{\pi}$ be a policy with $\tilde{\pi}(h_{<k}) = \pi(h_{<k})$ for all $h_{<k}$ where $k \neq t$. Now define policies $\tilde{\pi}_{t_i}$ by

$$\tilde{\pi}_{t_i}(h_{<k}) = \begin{cases} \tilde{\pi}(h_{<k}) & \text{if } k \leq i \\ \pi_{t_i}(h_{<k}) & \text{otherwise} \end{cases}$$

By point 1 above, $\tilde{\pi}_{t_i}(h_{<k}) = \pi_{t_i}(h_{<k})$ for all $h_{<k}$ where $k \neq t$. Now for all $i > t$ we have

$$V_{\mathbf{d}^t}^\pi(h_{<t}) \geq V_{\mathbf{d}^t}^{\pi_{t_i}}(h_{<t}) - |V_{\mathbf{d}^t}^\pi(h_{<t}) - V_{\mathbf{d}^t}^{\pi_{t_i}}(h_{<t})| \quad (7)$$

$$\geq \tilde{V}_{\mathbf{d}^t}^{\pi_{t_i}}(h_{<t}) - |V_{\mathbf{d}^t}^\pi(h_{<t}) - \tilde{V}_{\mathbf{d}^t}^{\pi_{t_i}}(h_{<t})| \quad (8)$$

$$\geq \tilde{V}_{\mathbf{d}^t}^{\tilde{\pi}_{t_i}}(h_{<t}) - |V_{\mathbf{d}^t}^\pi(h_{<t}) - \tilde{V}_{\mathbf{d}^t}^{\tilde{\pi}_{t_i}}(h_{<t})| \quad (9)$$

$$\begin{aligned} &\geq V_{\mathbf{d}^t}^{\tilde{\pi}}(h_{<t}) - |V_{\mathbf{d}^t}^\pi(h_{<t}) - V_{\mathbf{d}^t}^{\tilde{\pi}}(h_{<t})| \\ &\quad - |V_{\mathbf{d}^t}^{\tilde{\pi}_{t_i}}(h_{<t}) - \tilde{V}_{\mathbf{d}^t}^{\tilde{\pi}_{t_i}}(h_{<t})| - |V_{\mathbf{d}^t}^{\tilde{\pi}_{t_i}}(h_{<t}) - V_{\mathbf{d}^t}^{\tilde{\pi}}(h_{<t})| \end{aligned} \quad (10)$$

where (7) follows from arithmetic. (8) since $V \geq \tilde{V}$. (9) since π_{t_i} is a sub-game perfect equilibrium policy. (10) by arithmetic. We now show that the absolute value terms in (10) converge to zero. Since $V^\pi(\cdot)$ is continuous in π and $\lim_{i \rightarrow \infty} \pi_{t_i} = \pi$ and $\lim_{i \rightarrow \infty} \tilde{\pi}_{t_i} = \tilde{\pi}$, we obtain $\lim_{i \rightarrow \infty} [|V_{\mathbf{d}^t}^\pi(h_{<t}) - V_{\mathbf{d}^t}^{\pi_{t_i}}(h_{<t})| + |V_{\mathbf{d}^t}^{\tilde{\pi}_{t_i}}(h_{<t}) - V_{\mathbf{d}^t}^{\tilde{\pi}}(h_{<t})|] = 0$. Now $\tilde{\pi}_{t_i}(h_{<k}) = a^{death}$ if $k \geq t_i$, so $|V^{\tilde{\pi}_{t_i}}(h_{<t}) - \tilde{V}^{\tilde{\pi}_{t_i}}(h_{<t})| = 0$. Therefore taking the limit as i goes to infinity in (10) shows that $V_{\mathbf{d}^t}^\pi(h_{<t}) \geq V_{\mathbf{d}^t}^{\tilde{\pi}}(h_{<t})$ as required. \square

In general, $\pi_{\mathbf{d}}^*$ need not be unique, and different sub-game equilibrium policies can lead to different utilities. This is a normal, but unfortunate, problem with the sub-game equilibrium solution concept. The policy is unique if for all players the value of any two arbitrary policies is different. Also, if $\forall k (V_{\mathbf{d}^k}^{\pi_1} = V_{\mathbf{d}^k}^{\pi_2} \implies \forall j V_{\mathbf{d}^j}^{\pi_1} = V_{\mathbf{d}^j}^{\pi_2})$ is true then the non-unique sub-game equilibrium policies have the same values for all agents. Unfortunately, neither of these conditions is necessarily satisfied in our setup. The problem of how players might choose a sub-game perfect equilibrium policy appears surprisingly understudied. We feel it provides another reason to avoid the situation altogether by using time-consistent discount matrices. The following example illustrates the problem of non-unique sub-game equilibrium policies.

Example 17. Consider the example in Section 3 with an agent using a constant horizon discount matrix with $H = 2$. There are exactly two sub-game perfect equilibrium policies, π_1 and π_2 defined by,

$$\pi_1(h_{<t}) = \begin{cases} up & \text{if } t \text{ is odd} \\ right & \text{otherwise} \end{cases} \quad \pi_2(h_{<t}) = \begin{cases} up & \text{if } t \text{ is even} \\ right & \text{otherwise} \end{cases}$$

Note that the reward sequences (and values) generated by π_1 and π_2 are different with $\mathbf{R}^{\pi_1}(h_{<1}) = [1/2, 0, 0, \dots]$ and $\mathbf{R}^{\pi_2}(h_{<1}) = [0, 2/3, 0, 0, \dots]$. If the players choose to play a sub-game perfect equilibrium policy then the first player can choose between π_1 and π_2 since they have the first move. In that case it would be best to follow π_2 by moving right as it has a greater return for the agent at time 0 than π_1 .

For time-consistent discount matrices we have the following proposition.

Proposition 18. *If \mathbf{d} is time-consistent then $V_{\mathbf{d}^k}^* = V_{\mathbf{d}^k}^{\pi_{\mathbf{d}}} = V_{\mathbf{d}^k}^{\pi_{\mathbf{d}}^*}$ for all k and choices of $\pi_{\mathbf{d}^k}^*$ and $\pi_{\mathbf{d}}$ and $\pi_{\mathbf{d}}^*$.*

Is it possible that backwards induction is simply expected discounted reward maximisation in another form? The following theorem shows this is not the case and that sub-game perfect equilibrium policies are a rich and interesting class worthy of further study in this (and more general) settings.

Theorem 19. *$\exists \mathbf{d}$ such that $\pi_{\mathbf{d}}^* \neq \pi_{\tilde{\mathbf{d}}^0}^*$, for all $\tilde{\mathbf{d}}^0$.*

The result is proven using a simple counter-example. The idea is to construct a stochastic environment where the first action leads the agent to one of two sub-environments, each with probability half. These environments are identical to the example at the start of this section, but one of them has the reward 1 (rather than 3) for the history *right, down*. It is then easily shown that $\pi_{\mathbf{d}}^*$ is not the result of an expectimax expression because it behaves differently in each sub-environment, while any expectimax search (irrespective of discounting) will behave the same in each.

6 Discussion

Summary. Theorem 11 gives a characterisation of time-(in)consistent discount matrices and shows that all time-consistent discount matrices follow the simple form of $d_t^k = d_t^l$. Theorem 13 shows that using a discount matrix that is nearly time-consistent produces mixed policies with low regret. This is useful for a few reasons, including showing that small perturbations, such as rounding errors, in a discount matrix cannot cause major time-inconsistency problems. It also shows that “cutting off” time-consistent discount matrices after some fixed depth - which makes the agent potentially time-inconsistent - doesn’t affect the policies too much, provided the depth is large enough. When a discount matrix is very time-inconsistent then taking a game theoretic approach may dramatically decrease the regret in the change of policy over time.

Some comments on the policies $\pi_{d^k}^*$ (policy maximising expected d^k -discounted reward), π_d (mixed policy using $\pi_{d^k}^*$ at each time-step t) and π_d^* (sub-game perfect equilibrium policy).

1. A time-consistent agent should play policy $\pi_{d^k}^* = \pi_d$ for any k . In this case, every optimal policy $\pi_{d^k}^*$ is also a sub-game perfect equilibrium policy.
2. π_d will be played by an agent that believes it is time-consistent, but may not be. This can lead to very bad behavior as shown in Section 3.
3. An agent may play π_d^* if it knows it is time-inconsistent, and also knows exactly how (I.e, it knows d^k for all k at every time-step). This policy is arguably rational, but comes with its own problems, especially non-uniqueness as discussed.

Assumptions. We made a number of assumptions about which we make some brief comments.

1. Assumption 2, which states that \mathcal{A} and \mathcal{O} are finite, guarantees the existence of an optimal policy. Removing the assumption would force us to use ϵ -optimal policies, which shouldn’t be a problem for the theorems to go through with an additive ϵ slop term in some cases.
2. Assumption 2 only affects non-summable discount vectors. Without it, even ϵ -optimal policies need not exist and all the machinery will break down.
3. The use of discrete time greatly reduced the complexity of the analysis. Given a sufficiently general model, the set of continuous environments should contain all discrete environments. For this reason the proof of Theorem 11 should go through essentially unmodified. The same may not be true for Theorems 13 and 16. The former may be fixable with substantial effort (and perhaps should be true intuitively). The latter has been partially addressed, with a positive result in [3, 10, 11, 13].

Acknowledgements. We thank reviewers for valuable feedback on earlier drafts.

References

- [1] Frederick, S., Oewenstein, G.L., O'Donoghue, T.: Time discounting and time preference: A critical review. *Journal of Economic Literature* 40(2) (2002)
- [2] Fudenberg, D.: Subgame-perfect equilibria of finite and infinite-horizon games. *Journal of Economic Theory* 31(2) (1983)
- [3] Goldman, S.M.: Consistent plans. *The Review of Economic Studies* 47(3), 533–537 (1980)
- [4] Green, L., Fristoe, N., Myerson, J.: Temporal discounting and preference reversals in choice between delayed outcomes. *Psychonomic bulletin and review* 1(3), 383–389 (1994)
- [5] Hutter, M.: Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability. Springer, Berlin (2004)
- [6] Hutter, M.: General Discounting Versus Average Reward. In: Balcázar, J.L., Long, P.M., Stephan, F. (eds.) ALT 2006. LNCS (LNAI), vol. 4264, pp. 244–258. Springer, Heidelberg (2006)
- [7] Legg, S.: Machine Super Intelligence. PhD thesis, University of Lugano (2008)
- [8] Legg, S., Hutter, M.: Universal intelligence: A definition of machine intelligence. *Minds & Machines* 17(4), 391–444 (2007)
- [9] Osborne, M.J., Rubinstein, A.: A Course in Game Theory. The MIT Press, Cambridge (1994)
- [10] Peleg, B., Yaari, M.E.: On the existence of a consistent course of action when tastes are changing. *The Review of Economic Studies* 40(3), 391–401 (1973)
- [11] Pollak, R.A.: Consistent planning. *The Review of Economic Studies* 35(2), 201–208 (1968)
- [12] Samuelson, P.A.: A note on measurement of utility. *The Review of Economic Studies* 4(2), 155–161 (1937)
- [13] Strotz, R.H.: Myopia and inconsistency in dynamic utility maximization. *The Review of Economic Studies* 23(3), 165–180 (1955)
- [14] Thaler, R.: Some empirical evidence on dynamic inconsistency. *Economics Letters* 8(3), 201–207 (1981)

A Technical Proofs

Before the proof of Theorem 6 we require a definition and two lemmas.

Definition 20. Let $\Pi = \mathcal{A}^S$ be the set of all policies and define a metric D on Π by $T(\pi_1, \pi_2) := \min_{t \in \mathbb{N}} \{t : \exists h_{<t} \text{ s.t. } \pi_1(h_{<t}) \neq \pi_2(h_{<t})\}$ or ∞ if $\pi_1 = \pi_2$ and $D(\pi_1, \pi_2) := \exp(-T(\pi_1, \pi_2))$.

T is the time-step at which π_1 and π_2 first differ. Now augment Π with the topology induced by the metric d .

Lemma 21. Π is compact.

Proof. We proceed by showing Π is totally bounded and complete. Let $\epsilon = \exp(-t)$ and define an equivalence relation by $\pi \sim \pi'$ if and only if $T(\pi_1, \pi_2) \geq t$. If $\pi \sim \pi'$ then $D(\pi, \pi') \leq \epsilon$. Note that Π/\sim is finite. Now choose a representative from each class to create a finite set $\bar{\Pi}$. Now $\bigcup_{\pi \in \bar{\Pi}} B_\epsilon(\pi) = \Pi$, where $B_\epsilon(\pi)$ is the ball of radius ϵ about π . Therefore Π is totally bounded.

Next, to show Π is complete. Let π_1, π_2, \dots be a Cauchy sequence with $D(\pi_i, \pi_{i+j}) < \exp(-i)$ for all $j > 0$. Therefore $\pi_i(h_{<k}) = \pi_{i+j}(h_{<k}) \forall h_{<k}$ with $k \leq i$, by the definition of D . Now define π by $\pi(h_{<t}) := \pi_t(h_{<t})$ and note that $\pi_i(h_{<j}) = \pi(h_{<j}) \forall j \leq i$ since $\pi_i(h_{<k}) = \pi_k(h_{<k}) \equiv \pi(h_{<k})$ for $k \leq i$. Therefore $\lim_{i \rightarrow \infty} \pi_i = \pi$ and so Π is complete. Finally, Π is compact by the Heine-Borel theorem. \square

Lemma 22. *When viewed as a function from Π to \mathbb{R} , $V_{\mathbf{d}^k}^\pi(\cdot)$ is continuous. (given Assumption 2)*

Proof. Suppose $D(\pi_1, \pi_2) < \exp(-t)$ then π_1 and π_2 are identical on all histories up to length t . Therefore

$$\begin{aligned} |V_{\mathbf{d}^k}^{\pi_1}(h_{<k}) - V_{\mathbf{d}^k}^{\pi_2}(h_{<k})| &\leq \mathbf{d}^k \cdot [\mathbf{R}^{\pi_1}(h_{<k}) + \mathbf{R}^{\pi_2}(h_{<k})] \\ &= \sum_{i=k}^{\infty} d_i^k (R_i^{\pi_1}(h_{<k})_i + R_i^{\pi_2}(h_{<k})_i). \end{aligned} \quad (11)$$

Since π_1 and π_2 are identical up to time t , (11) becomes

$$\begin{aligned} \sum_{i=t}^{\infty} d_i^k (R_i^{\pi_1}(h_{<k})_i + R_i^{\pi_2}(h_{<k})_i) &= \\ \sum_{h_{<t}} [P(h_{<t}|h_{<k}, \pi_1)V_{\mathbf{d}^k}^{\pi_1}(h_{<t}) + P(h_{<t}|h_{<k}, \pi_2)V_{\mathbf{d}^k}^{\pi_2}(h_{<t})] & \end{aligned} \quad (12)$$

where (12) follows from the definition of the reward and value functions. By Assumption 2, $\lim_{t \rightarrow \infty} \sum_{h_{<t}} P(h_{<t}|h_{<k}, \pi_i)V_{\mathbf{d}^k}^{\pi_i}(h_{<t}) = 0$ for $i \in \{1, 2\}$ and so, V is continuous. \square

Proof (Theorem 6). Let Π be the space of all policies with the metric of Definition 20. By Lemmas 21/22 Π is compact and V is continuous. Therefore $\arg \max_\pi V_{\mathbf{d}^k}^\pi(h_{<1})$ exists by the extreme value theorem. \square

Distributional Learning of Simple Context-Free Tree Grammars

Anna Kasprzik¹ and Ryo Yoshinaka^{2,*}

¹ University of Trier, FB IV Informatik, 54286 Trier
kasprzik@informatik.uni-trier.de

² ERATO MINATO Project, Japan Science and Technology Agency
ry@ist.hokudai.ac.jp

Abstract. This paper demonstrates how existing distributional learning techniques for context-free grammars can be adapted to *simple context-free tree grammars* in a straightforward manner once the necessary notions and properties for string languages have been redefined for trees. Distributional learning is based on the decomposition of an object into a substructure and the remaining structure, and on their interrelations. A corresponding learning algorithm can emulate those relations in order to determine a correct grammar for the target language.

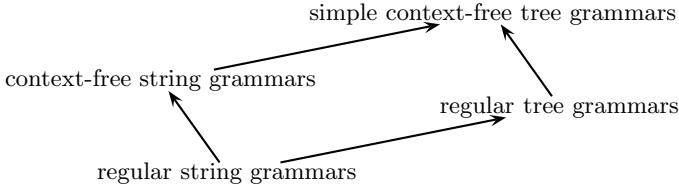
1 Introduction

This paper is settled in the area of *Grammatical Inference*, i.e., the study of algorithms that “learn” formal languages from only partial information. The class that has been studied most extensively with respect to its algorithmical learnability is that of regular string languages. The established learning algorithms for regular string languages have soon been extended to more complex structures, most notably to trees (see for example [1] for learning a subset of regular tree languages from finite positive data, and [2,3,4] for learning regular tree languages from queries and/or finite data).

However, recently a range of efforts have been made to explore other classes beyond the regular one. While the class of context-free languages as a whole does not seem to be learnable in any non-trivial setting considered so far, there exist several context-free subclasses with certain properties that make them learnable by accordingly adapted strategies which can be subsumed under the term of *distributional learning* (see for example [5,6,7,8,9,10], and references therein).

Every member w of a context-free string language $L_* \subseteq \Sigma^*$ can be decomposed into a substring y and a context $\langle x, z \rangle \in \Sigma^* \times \Sigma^*$ such that $w = xyz$. The theory of distributional learning is concerned with the question which of those substrings and which of those contexts can be put back together to form another grammatical member of L_* . If L_* fulfils certain distributional properties, a corresponding learning algorithm can exploit information on these particular

* The author is concurrently working in Hokkaido University. This work was supported in part by MEXT KAKENHI (B-23700156)

**Fig. 1.**

distributions for example in order to determine nonterminals and production rules of a grammar reflecting those interrelations, with the consequence that the resulting grammar correctly generates L_* .

The technique of distributional learning can be extended to languages based on more complex structures than the basic string once we have redefined such notions as a substructure and its counterpart, the *context* or *environment* of a certain substructure, along with an *unambiguous concatenation operation* for those two kinds of objects. Yoshinaka [10], for example, generalizes the learning algorithm given in [6] for context-free grammars (CFGs) to multiple CFGs. Moreover, Yoshinaka and Kanazawa [12] demonstrate that the techniques of distributional learning developed so far can be likewise adapted to other kinds of objects and formalisms in a rather abstract way via Abstract Categorial Grammars.

The contribution of our paper should be located in this line of research. We concentrate on the case of trees and consider *simple* (i.e., non-duplicating and non-deleting) *context-free tree grammars* (SCFTGs) as descriptions¹. SCFTGs join two directions of generalization: They can be seen as a natural context-free counterpart of regular tree grammars and as a natural extension of CFGs to trees (Fig. 1). Moreover, the class of string languages that can be associated with SCFTGs is located in the so-called “mildly context-sensitive” family proposed by Joshi [11] which is of particular importance in computational linguistics. In this paper we demonstrate that distributional learning techniques for CFGs can be translated in a straightforward way to techniques for SCFTGs and thereby establish a parallel to the fact that learning techniques for regular string languages can be transferred directly to regular tree languages. We remark that although the distributional learning of SCFTGs can be seen as a special case covered by [12], in contrast to the generality of their discussion we obtain a much clearer and detailed view on the close correspondence between the distributional learning of CFGs and their immediate tree counterpart by basing our results directly on notions and results from formal (tree) language theory.

Organization of the paper: In Section 2 we define the necessary notational tools and we give the definition of SCFTGs. In Section 3 we establish the basic principles for the distributional learning of SCFTGs and then present and discuss two different distributional learning algorithms for them, in two different learning settings. We conclude and sketch some suggestions for future work in Section 4.

¹ SCFTGs have also been called *linear context-free tree grammars* in the literature.

2 Preliminaries

2.1 Trees and Substitutions

We presume a basic knowledge of standard tree terminology. For a more comprehensive introduction to trees and associated concepts and theorems, see [13] for example.

A *ranked alphabet* is a set of symbols Σ paired with a (total) function $\rho : \Sigma \rightarrow \mathbb{N}$. For $k \geq 0$, we write $\Sigma_k = \{a \in \Sigma \mid \rho(a) = k\}$ to denote the set of all symbols with rank k .

The set \mathbb{T}_Σ of all *trees* over a ranked alphabet Σ is defined as the smallest set of expressions such that $t = f(t_1, \dots, t_k) \in \mathbb{T}_\Sigma$ for every $k \geq 0$, $f \in \Sigma_k$, and $t_1, \dots, t_k \in \mathbb{T}_\Sigma$. The tree $f()$ for $f \in \Sigma_0$ is often abbreviated to f . The *size* of a tree t , which is denoted by $|t|$, means the number of occurrences of symbols in t , i.e., $|f(t_1, \dots, t_k)| = 1 + |t_1| + \dots + |t_k|$.

A subset of \mathbb{T}_Σ is called a *tree language*. For a finite tree language $L \subseteq \mathbb{T}_\Sigma$, we define the *size* of L by $\|L\| = \sum_{t \in L} |t|$.

Throughout this paper we assume a countably infinite set X of *ordered variables* x_1, x_2, \dots which have rank 0.

A tree $t \in \mathbb{T}_{\Sigma \cup X}$ is called an *m-stub* if only and all the variables x_1, \dots, x_m occur exactly once in t for some $m \in \mathbb{N}$ ². The set of all *m-stubs* is denoted by \mathbb{S}_Σ^m and we let $\mathbb{S}_\Sigma = \bigcup_{m \in \mathbb{N}} \mathbb{S}_\Sigma^m$. We note that $\mathbb{S}_\Sigma^0 = \mathbb{T}_\Sigma$.

A *leaf substitution* η is a finite partial mapping from X to \mathbb{T}_Σ , which is extended to a function $\hat{\eta} : \mathbb{T}_{\Sigma \cup X} \rightarrow \mathbb{T}_{\Sigma \cup X}$ as follows:

- for $x \in X$, let $\hat{\eta}(x) = \eta(x)$ if $\eta(x)$ is defined and $\hat{\eta}(x) = x$ otherwise, and
- for $f \in \Sigma_k$, let $\hat{\eta}(f(t_1, \dots, t_k)) = f(\hat{\eta}(t_1), \dots, \hat{\eta}(t_k))$.

We identify a leaf substitution η with its extension $\hat{\eta}$. We often put η after an argument as a postfix operator, i.e., $t\eta$ instead of $\eta(t)$. A leaf substitution that maps x_{i_1}, \dots, x_{i_m} to t_1, \dots, t_m , respectively, is often denoted as $[x_{i_1} \leftarrow t_1, \dots, x_{i_m} \leftarrow t_m]$. In cases where the domain is just $\{x_1, \dots, x_m\}$, we abbreviate such an operator $[x_1 \leftarrow t_1, \dots, x_m \leftarrow t_m]$ to $[t_1, \dots, t_m]$.

Let Δ and Σ be ranked alphabets. An *infix substitution* σ is a mapping from Δ to \mathbb{S}_Σ such that $\sigma(f) \in \mathbb{S}_\Sigma^k$ for all $f \in \Delta_k$. σ is extended to a function $\hat{\sigma} : \mathbb{S}_{\Sigma \cup \Delta} \rightarrow \mathbb{S}_\Sigma$ as follows:

- $\hat{\sigma}(f(s_1, \dots, s_k)) = \sigma(f)[\hat{\sigma}(s_1), \dots, \hat{\sigma}(s_k)]$ for $f \in \Delta_k$,
- $\hat{\sigma}(f(s_1, \dots, s_k)) = f(\hat{\sigma}(s_1), \dots, \hat{\sigma}(s_k))$ for $f \notin \Delta_k$.

We identify an infix substitution σ with its extension $\hat{\sigma}$. The definition of an infix substitution coincides with that of a leaf substitution when the domain Δ consists only of symbols of rank 0. Hence we may denote an infix substitution that maps $Y_i \in \Delta$ to $s_i \in \mathbb{S}_\Sigma$ for $i = 1, \dots, m$ as a postfix operator $[Y_1 \leftarrow s_1, \dots, Y_m \leftarrow s_m]$ without confusion.

² What we call a stub is also called a ‘context’ in the literature on trees [13]. However, we avoid this established terminology since in our framework a stub fulfills the role of a substructure, and not of the remaining parts of a decomposition.

Example 1. Let $\Sigma = \Sigma_0 \cup \Sigma_1$ with $\Sigma_1 = \{a, b, c, d\}$, $\Sigma_0 = \{e\}$ and $\Delta = \Delta_1 = \{Y_1\}$. For $s = b(c(x_1)) \in \mathbb{S}_{\Sigma}^1$, the result of the application of an infix substitution $[Y_1 \leftarrow s]$ to a tree $t = a(Y_1(d(e))) \in \mathbb{T}_{\Sigma \cup \{Y_1\}}$ is $t[Y_1 \leftarrow s] = a(b(c(d(e))))$.

Let $\Sigma_0 = \{a, b, c\}$, $\Sigma_1 = \{h\}$, $\Sigma_2 = \{f, g\}$ and $\Delta = \Delta_2 = \{Y_2\}$. For $s = f(x_1, g(b, x_2)) \in \mathbb{S}_{\Sigma}^2$ and $t = h(Y_2(a, c)) \in \mathbb{T}_{\Sigma \cup \{Y_2\}}$, we have

$$t[Y_2 \leftarrow s] = h(Y_2(a, c))[Y_2 \leftarrow f(x_1, g(b, x_2))] = h(f(a, g(b, c))).$$

Lemma 1. Let an infix substitution σ from Δ to \mathbb{S}_{Σ} and a symbol $Y \in \Sigma$ be such that $Y \notin \Delta$ and $\sigma(a)$ contains no occurrence of Y for any $a \in \Delta$. Then $(s_1[Y \leftarrow s_2])\sigma = (s_1\sigma)[Y \leftarrow (s_2\sigma)]$.

2.2 Simple Context-Free Tree Grammars

We are now ready to specify the grammars that generate the kind of languages we will consider as targets for our learning algorithms in Section 3.

Definition 1. Let $r \in \mathbb{N}$ be a natural number. We define an r -simple context-free tree grammar (r -SCFTG) as a 4-tuple $G = \langle \Sigma, N, I, P \rangle$ where

- Σ is a ranked alphabet of terminals with $\Sigma_m = \emptyset$ for all $m > r$,
- N is a ranked alphabet of nonterminals with $N_m = \emptyset$ for all $m > r$,
- $I \subseteq N_0$ is a set of initial symbols,³ and
- P is a finite set of production rules of the form $A \rightarrow s$ with $A \in N_m$ and $s \in \mathbb{S}_{\Sigma}^m$ for some $m \geq 0$.

We let $P_m = \{A \rightarrow s \in P \mid A \in N_m \text{ and } s \in \mathbb{S}_{\Sigma}^m\}$.

For $u, v \in \mathbb{S}_{\Sigma \cup N}^n$, we write $u \Rightarrow_G v$ if there is $m \in \mathbb{N}$, $A \rightarrow s \in P_m$ and $t \in \mathbb{S}_{\Sigma \cup N \cup \{Y\}}^n$ such that Y has rank m and occurs just once in t and

- $u = t[Y \leftarrow A(x_1, \dots, x_m)]$ and
- $v = t[Y \leftarrow s]$.

The relation \Rightarrow_G^* on $\mathbb{S}_{\Sigma \cup N}$ is defined as the reflexive transitive closure of \Rightarrow_G .

The stub language $\mathcal{L}(G, A)$ generated by $A \in N_m$ is the set

$$\mathcal{L}(G, A) = \{s \in \mathbb{S}_{\Sigma}^m \mid A(x_1, \dots, x_m) \Rightarrow_G^* s\}.$$

We simply write \mathcal{L}_A for $\mathcal{L}(G, A)$ where G is understood. The tree language generated by G is defined as $\mathcal{L}(G) = \bigcup_{A \in I} \mathcal{L}_A$.

Context-free (string) grammars (CFGs) can be interpreted as 1-SCFTGs and vice versa.

Lemma 2. If $A \Rightarrow_G^* u[Y \leftarrow B(x_1, \dots, x_m)]$ and $B \Rightarrow_G^* v$, then $A \Rightarrow_G^* u[Y \leftarrow v]$ for any $A \in N$, $B \in N_m$, $u \in \mathbb{S}_{\Sigma \cup N \cup \{Y\}}^m$, $v \in \mathbb{S}_{\Sigma \cup N}^m$ and Y of rank m .

³ Contrary to the standard definition of CFTGs, we allow multiple initial symbols.

Obviously this generalization does not affect the expressive power of the formalism, yet our results will be conveniently presented in this non-standard form.

Definition 2. We call an m -stub s *non-permuting* if the variables x_1, \dots, x_m occur in this order from left to right in s .

An SCFTG is said to be *normal* if, for any rule $A \rightarrow s$, the stub $s \in \mathbb{S}_{\Sigma \cup N}$ is of one of the following two types:

- I. $f(x_1, \dots, x_k)$ for some $f \in \Sigma_k$ or
- II. $B(x_1, \dots, x_i, C(x_{i+1}, \dots, x_j), x_{j+1}, \dots, x_k)$ for some $B \in N_{k-j+i+1}$ and $C \in N_{j-i}$ with $0 \leq i \leq j \leq k$.

One can show that indeed every r -SCFTG admits an equivalent r -SCFTG in the normal form by a technique similar to the proofs for corresponding theorems in related formalisms (e.g., [14], [15]). For the rest of this paper we will assume all stubs of $\mathbb{S}_{\Sigma \cup N}^m$ to be non-permuting and all SCFTGs to be normal.

Proposition 1. Fix a positive integer r . The uniform membership problem for r -SCFTGs, which asks whether $t \in \mathcal{L}(G)$, is decidable in polynomial time in the size of an r -SCFTG G and a tree t .

Proof. The case of r -SCFTGs can be seen as a special case of a richer formalism that has a polynomial-time parsing algorithm if instances are restricted to r -SCFTGs (e.g., [16]). To make this paper self-contained, yet we outline a parsing algorithm for r -SCFTGs based on the standard CKY algorithm for CFGs. We have an $(m+1)$ -dimensional table T_m where each dimension corresponds to a node position of the tree to be parsed for each $m = 0, \dots, r$. We keep and update the tables so that each cell of T_m is occupied by nonterminals of rank m that can generate the m -stub determined by the corresponding $(m+1)$ positions. It is not hard to see that if instances are normal, such an algorithm runs in polynomial time, where the degree of the polynomial linearly depends on r .

3 Distributional Learning of Simple Context-Free Tree Grammars

3.1 Decomposition of Trees

Consider a context-free string language $L \subseteq \Sigma^*$, and the decompositions of members w of L into substrings w' and contexts $\langle u, v \rangle \in \Sigma^* \times \Sigma^*$ with $w = uw'v$. Distributional learning of CFGs is based on the analysis and emulation of the specific relations between those particular strings and contexts that when put together form a grammatical string of the language in question.

We (re)define suitable concepts corresponding to ‘substrings’ and ‘contexts’ for the tree case, with the goal of making distributional learning algorithms for strings directly translatable into distributional learning algorithms for trees. In our framework, the tree counterpart of a substring is a stub, which is the kind of object that a nonterminal of an SCFTG derives. We specify our tree counterpart of a context as follows.

Definition 3. An m -environment is a tree over $\Sigma \cup \{\#_m\}$ in which $\#_m$ occurs exactly once, where $\#_m \notin \Sigma$ is a special symbol of rank m . The set of all m -environments is denoted by \mathbb{E}_Σ^m . For an m -environment $e \in \mathbb{E}_\Sigma^m$ and an m -stub $s \in \mathbb{S}_\Sigma^m$, we define a binary operation \odot_m by

$$e \odot_m s = e[\#_m \leftarrow s].$$

The domain and accordingly the range of the operation is naturally extended to sets in the standard way.

The subscript m of \odot_m is often suppressed. We note that the result of the operation is always a tree in \mathbb{T}_Σ .

Definition 4 contains the essence of the substructure-environment relation for a certain language in the tree case.

Definition 4. For a tree language $L \subseteq \mathbb{T}_\Sigma$ and $m, r \in \mathbb{N}$, we let

$$\begin{aligned} Sub^m(L) &= \{s \in \mathbb{S}_\Sigma^m \mid e \odot s \in L \text{ for some } e \in \mathbb{E}_\Sigma^m\}, \\ Sub^{\leq r}(L) &= \bigcup_{m \leq r} Sub^m(L), \\ Env^m(L) &= \{e \in \mathbb{E}_\Sigma^m \mid e \odot s \in L \text{ for some } s \in \mathbb{S}_\Sigma^m\}, \\ Env^{\leq r}(L) &= \bigcup_{m \leq r} Env^m(L). \end{aligned}$$

We note that we always have $\#_0 \in Env^0(L)$ and $x_1 \in Sub^1(L)$ unless L is empty. $Sub^0(L)$ corresponds to the set of ‘subtrees’ in the usual sense.

Lemma 3. Fix a positive integer r . For any finite language $L \subseteq \mathbb{T}_\Sigma$, one can compute the sets $Sub^{\leq r}(L)$ and $Env^{\leq r}(L)$ in polynomial time in $\|L\|$.

We remark that the degree of the polynomial linearly depends on r .

Hereafter, we drop Σ to denote sets $\mathbb{S}_\Sigma, \mathbb{E}_\Sigma$ as \mathbb{S}, \mathbb{E} , when Σ is understood.

Example 2. Let $\Sigma = \Sigma_0 \cup \Sigma_2$ where $\Sigma_0 = \{a, b, c\}$ and $\Sigma_2 = \{f, g\}$. A tree $t = f(a, g(b, c)) \in \mathbb{T}$ can be decomposed in many ways:

$$\begin{aligned} t &= \#_0 \odot t && (\#_0 \in Env^0(t), t \in Sub^0(t)) \\ &= f(a, \#_0) \odot g(b, c) && (f(a, \#_0) \in Env^0(t), g(b, c) \in Sub^0(t)) \\ &= f(a, \#_1(b)) \odot g(x_1, c) && (f(a, \#_1(b)) \in Env^1(t), g(x_1, c) \in Sub^1(t)) \\ &= f(a, \#_2(b, c)) \odot g(x_1, x_2) && (f(a, \#_2(b, c)) \in Env^1(t), g(x_1, x_2) \in Sub^1(t)) \\ &= \#_2(a, c) \odot f(x_1, g(b, x_2)) && (\#_2(a, c) \in Env^2(t), f(x_1, g(b, x_2)) \in Sub^2(t)) \end{aligned}$$

and so on.

In the following subsections, we illustrate how these adapted notions result in distributional learning algorithms for trees by discussing some concrete examples in detail.

3.2 Substitutable Simple Context-Free Tree Languages

There are several properties that make context-free languages learnable using distributional techniques. Among those properties, we will pick the strongest one first: *Substitutability*. The class of substitutable CFGs and even the class of substitutable multiple context-free grammars (MCFGs) have been shown to be efficiently learnable from positive examples by Clark [6] and Yoshinaka [10], respectively. Compared to other existing distributional learning algorithms those learners are rather simple due to the great restriction of substitutability.

The learning setting we consider is the same as in the two references given above: *Identification in the limit from positive data* [17]. Let G_* be the target grammar. A learner \mathcal{A} is given an infinite sequence of positive examples $t_1, t_2, \dots \in \mathcal{L}(G_*)$ fulfilling the condition $\mathcal{L}(G_*) = \{t_i \mid i \geq 1\}$. For each $n \geq 1$, \mathcal{A} constructs a conjecture grammar G_n based on the data t_1, \dots, t_n received so far. We say that \mathcal{A} identifies G_* in the limit from positive data if for any sequence of positive examples from $\mathcal{L}(G_*)$, there is a point n_0 such that $\mathcal{L}(G_{n_0}) = \mathcal{L}(G_*)$ and $G_n = G_{n_0}$ for all $n > n_0$.

This subsection presents an efficient algorithm that identifies every *r-substitutable* SCFTG in the limit from positive data.

Definition 5. A tree language $L \subseteq \mathbb{T}_\Sigma$ is said to be *r-substitutable* if the following holds:

- For any $m \leq r$ and $s_1, s_2 \in \mathbb{S}^m$, if there is $e_0 \in \mathbb{E}^m$ with $e_0 \odot s_1, e_0 \odot s_2 \in L$ then we have $e \odot s_1 \in L$ iff $e \odot s_2 \in L$ for all $e \in \mathbb{E}^m$.

In such a case we say that s_1 and s_2 are *substitutable* for each other. An *r-substitutable* SCFTG is an *r-SCFTG* G such that $\mathcal{L}(G)$ is *r*-substitutable.

Fix an *r*-substitutable SCFTG G_* as our learning target. For a finite set $D \subseteq \mathcal{L}(G_*)$ of positive examples, our learner constructs an SCFTG $\mathcal{G}_D = \langle \Sigma, N, I, P \rangle$ as follows. First, we take the substubs in $Sub^{\leq r}(D)$ as nonterminal symbols:

$$\begin{aligned} N_m &= \{[\![s]\!] \mid s \in Sub^m(D)\} \text{ for } m \leq r, \\ I &= \{[\![t]\!] \in N_0 \mid t \in D\}. \end{aligned}$$

We want each nonterminal $[\![s]\!]$ to derive $s' \in \mathbb{S}$ if and only if s and s' are substitutable for each other. Our grammar \mathcal{G}_D has rules of the following two types.

- I. $[\![s]\!] \rightarrow a(x_1, \dots, x_m)$ for $s \in Sub^m(D)$ and $a \in \Sigma_m$,
if there is $e \in Env^m(D)$ such that $e \odot s \in D$ and $e \odot a(x_1, \dots, x_m) \in D$;
- II. $[\![s]\!] \rightarrow [\![s_1]\!](x_1, \dots, x_i, [\![s_2]\!](x_{i+1}, \dots, x_j), x_{j+1}, \dots, x_m)$ for $s \in Sub^m(D)$,
 $s_1 \in Sub^{m-j+i+1}(D)$, $s_2 \in Sub^{j-i}(D)$, if there is $e \in Env^m(D)$ such that
 $e \odot s \in D$ and $e \odot s_1[x_1, \dots, x_i, s_2[x_{i+1}, \dots, x_j], x_{j+1}, \dots, x_m] \in D$.

In other words, we obtain a rule of Type I if the learner finds a common environment for s and $a(x_1, \dots, x_m)$ and can thus conclude that those two stubs

are substitutable for each other, and accordingly a rule of Type II if the same occurs for s and $s_1[x_1, \dots, x_i, s_2[x_{i+1}, \dots, x_j], x_{j+1}, \dots, x_m]$.

A *trivial* rule of Type I is $\llbracket a(x_1, \dots, x_m) \rrbracket \rightarrow a(x_1, \dots, x_m)$ for $a \in \Sigma_m$. Once the symbol a is observed in D , the learner constructs this trivial rule. Similarly, if $s \in Sub(D)$ is represented as $s = s_1[\vec{x}_1, s_2[\vec{x}_2], \vec{x}_3]$ for some $s_1, s_2 \in Sub(D)$, then we have the *trivial* rule of Type II

$$\llbracket s_1[\vec{x}_1, s_2[\vec{x}_2], \vec{x}_3] \rrbracket \rightarrow \llbracket s_1 \rrbracket(\vec{x}_1, \llbracket s_2 \rrbracket(\vec{x}_2), \vec{x}_3),$$

where $\vec{x}_1, \vec{x}_2, \vec{x}_3 = x_1, \dots, x_m$. Successive applications of such trivial rules decompose a stub s into pieces in arbitrary ways and finally we obtain $\llbracket s \rrbracket \Rightarrow_{G_D}^* s$ for all $\llbracket s \rrbracket \in N$.

Algorithm 1 shows our learner for r -substitutable SCFTGS. It is clear by Proposition 1 and Lemma 3 that the algorithm updates its conjecture \hat{G} in polynomial time in the size $\|D\|$ of D .

Algorithm 1. Learning r -substitutable SCFTGS

Data: A sequence of positive examples t_1, t_2, \dots
Result: A sequence of SCFTGS G_1, G_2, \dots
let \hat{G} be an SCFTG such that $\mathcal{L}(\hat{G}) = \emptyset$;
for $n = 1, 2, \dots$ **do**
 read the next example t_n ;
 if $t_n \notin \mathcal{L}(\hat{G})$ **then**
 let $\hat{G} = G_D$ for $D = \{t_1, \dots, t_n\}$;
 end if
 output \hat{G} as G_n ;
end for

Example 3. Let us sketch an example run for Algorithm 1

Consider a target SCFTG $G_* = \langle \Sigma_0 \cup \Sigma_3, N_0^* \cup N_3^*, I^*, P^* \rangle$ where $\Sigma_0 = \{a, b, c, d, f\}$, $\Sigma_3 = \{g, h\}$, $N_0^* = \{A\}$, $N_3^* = \{B\}$, $I^* = \{A\}$, and P^* consists of the following three rules

$$A \rightarrow g(a, B(b, f, c), d), \quad B \rightarrow h(x_1, x_2, x_3), \quad B \rightarrow g(a, B(b, g(x_1, x_2, x_3), c), d).$$

Figure 2 illustrates the rules of G_* and example trees $t_1 = g(a, h(b, f, c), d)$ and $t_2 = g(a, g(a, h(b, f, c), c), d), d)$ generated by G_* . Obviously G_* is not in normal form but since the learner only has to construct a grammar generating the same language as G_* we choose this representation for understandability. Suppose the learner is given the tree t_1 as its first datum. The learner will react by constructing a grammar $\mathcal{G}_{\{t_1\}} = \{\Sigma, N, I, P\}$ as specified above. At this first stage, I is the singleton of $\llbracket t_1 \rrbracket$ and all the constructed rules in P are trivial ones. Thus this grammar does not generate any other tree but t_1 itself. Suppose the next datum given to the learner is t_2 . This results in several additional nonterminals and one more start symbol $\llbracket t_2 \rrbracket$. The learner observes

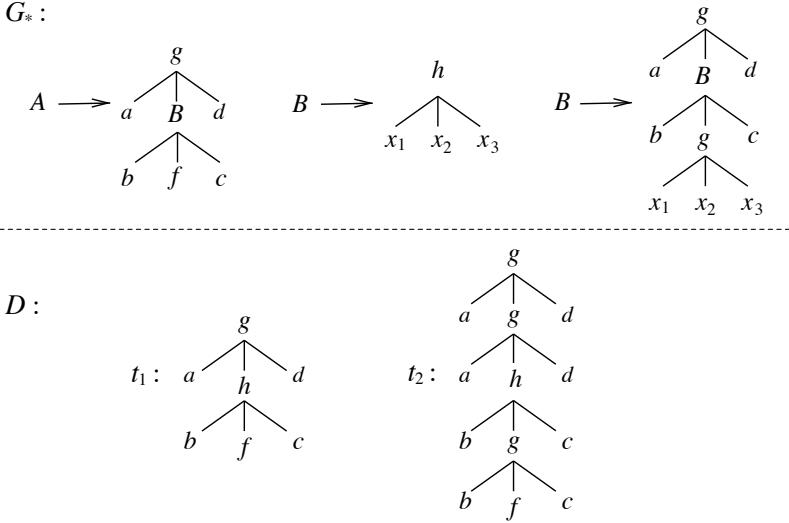


Fig. 2. Grammar G_* and sample set D for the example run of Algorithm \square

that $h(x_1, x_2, x_3)$ and $g(a, h(b, g(x_1, x_2, x_3), c), d)$ are substitutable for each other due to the environment $g(a, \#_3(b, f, c), d)$:

$$\begin{aligned} t_1 &= g(a, \#_3(b, f, c), d) \odot h(x_1, x_2, x_3) \in D, \\ t_2 &= g(a, \#_3(b, f, c), d) \odot g(a, h(b, g(x_1, x_2, x_3), c), d) \in D. \end{aligned}$$

The learner constructs a rule

$$[\![h(x_1, x_2, x_3)]\!] \rightarrow [\![g(a, x_1, d)]\!]([\![h(b, g(x_1, x_2, x_3), c)]\!](x_1, x_2, x_3))$$

by the fact $g(a, h(b, g(x_1, x_2, x_3), c), d) = g(a, x_1, d)[x_1 \leftarrow h(b, g(x_1, x_2, x_3), c)]$. Successive applications of trivial rules result in a derivation

$$\begin{aligned} [\![h(x_1, x_2, x_3)]\!] &\xrightarrow{\mathcal{G}_D} [\![g(a, x_1, d)]\!]([\![h(b, g(x_1, x_2, x_3), c)]\!](x_1, x_2, x_3)) \\ &\xrightarrow{*_{\mathcal{G}_D}} g(a, [\![h(x_1, x_2, x_3)]\!](b, g(x_1, x_2, x_3), c), d), \end{aligned}$$

which simulates the rule $B \rightarrow g(a, B(b, g(x_1, x_2, x_3), c), d)$ of G_* . As a consequence, \mathcal{G}_D generates every string in $\mathcal{L}(G_*)$.

Lemma 4. $\mathcal{L}(\mathcal{G}_D) \subseteq \mathcal{L}(G_*)$ for any $D \subseteq \mathcal{L}(G_*)$.

Proof. Let $\mathcal{G}_D = \langle \Sigma, N, I, P \rangle$ and σ an infix substitution from N to \mathbb{S}_Σ given by $\sigma([\![s]\!]) = s$ for all $[s] \in N$. By induction on the length of derivation, we show that if $[t] \xrightarrow{*_{\mathcal{G}_D}} t'$ for $[t] \in I$, then $t' \sigma \in \mathcal{L}(G_*)$. If $[t] \in I$, by definition it means $[t]\sigma = t \in \mathcal{L}(G_*)$. Thus the claim holds for any zero-step derivation.

Suppose that the last rule applied in the derivation process from $\llbracket t \rrbracket$ to t' is

$$\llbracket s \rrbracket \rightarrow \llbracket s_1 \rrbracket(x_1, \dots, x_i, \llbracket s_2 \rrbracket(x_{i+1}, \dots, x_j), x_{j+1}, \dots, x_m).$$

That is, there is $u \in \mathbb{E}^m$ such that

$$\begin{aligned} \llbracket t \rrbracket &\xrightarrow[\mathcal{G}_D]{} u \odot \llbracket s \rrbracket(x_1, \dots, x_m) \\ &\Rightarrow u \odot \llbracket s_1 \rrbracket(x_1, \dots, x_i, \llbracket s_2 \rrbracket(x_{i+1}, \dots, x_j), x_{j+1}, \dots, x_m) = t'. \end{aligned}$$

For the sake of legibility, let us abbreviate x_1, \dots, x_i to \vec{x}_1 , x_{i+1}, \dots, x_j to \vec{x}_2 and x_{j+1}, \dots, x_m to \vec{x}_3 . By the induction hypothesis, we have

$$(u \odot \llbracket s \rrbracket(\vec{x}_1, \vec{x}_2, \vec{x}_3))\sigma = u\sigma \odot s[\vec{x}_1, \vec{x}_2, \vec{x}_3] = u\sigma \odot s \in \mathcal{L}(G_*)$$

by Lemma 11. By the presence of the rule, we know that s and $s_1[\vec{x}_1, s_2[\vec{x}_2], \vec{x}_3]$ are substitutable for each other in $\mathcal{L}(G_*)$. The fact $u\sigma \odot s \in \mathcal{L}(G_*)$ implies $t'\sigma = u\sigma \odot s_1[\vec{x}_1, s_2[\vec{x}_2], \vec{x}_3] \in \mathcal{L}(G_*)$.

The case where the last applied rule is of Type I can be shown in the same way. \square

Next we give a set of positive examples from which our learner constructs a right conjecture. Let the target grammar $G_* = \langle \Sigma, N^*, I^*, P^* \rangle$. For each nonterminal $A \in N_m^*$, arbitrarily fix an m -environment e_A such that $B \Rightarrow_{G_*}^* e_A \odot A(\vec{x})$ for some $B \in I^*$ and an m -stub s_A such that $A \Rightarrow_{G_*}^* s_A$. We define

$$\begin{aligned} D_* = \{ &e_A \odot a(x_1, \dots, x_m) \mid A \rightarrow a(x_1, \dots, x_m) \in P^* \text{ with } a \in \Sigma \} \\ &\cup \{ e_A \odot s_B[\vec{x}_1, s_C[\vec{x}_2], \vec{x}_3] \mid A \rightarrow B(\vec{x}_1, C(\vec{x}_2), \vec{x}_3) \in P^* \text{ with } B, C \in N^* \}. \end{aligned}$$

Note that $e_A \odot s_A \in D_*$ for all $A \in N^*$ and $e_B = \#_0$ for all $B \in I^*$.

Lemma 5. *For any $D \subseteq \mathcal{L}(G_*)$, if $D \supseteq D_*$, we have $\mathcal{L}(G_D) = \mathcal{L}(G_*)$.*

Proof. We show that every nonterminal A of G_* is simulated by $\llbracket s_A \rrbracket$ in \mathcal{G}_D . If $A \in I^*$, then $s_A \in D$ and thus $\llbracket s_A \rrbracket \in I$.

For a rule $A \rightarrow a(x_1, \dots, x_m) \in P^*$, we have

$$e_A \odot s_A, e_A \odot a(x_1, \dots, x_m) \in D_*.$$

By definition, \mathcal{G}_D has the corresponding rule of Type I:

$$\llbracket s_A \rrbracket \rightarrow a(x_1, \dots, x_m).$$

Let us consider a rule

$$A \rightarrow B(\vec{x}_1, C(\vec{x}_2), \vec{x}_3) \in P^*$$

of G_* . We have

$$e_A \odot s_A, e_A \odot s_B[\vec{x}_1, s_C[\vec{x}_2], \vec{x}_3] \in D_*.$$

By definition, \mathcal{G}_D has the corresponding rule of Type II:

$$\llbracket s_A \rrbracket \rightarrow \llbracket s_B \rrbracket(\vec{x}_1, \llbracket s_C \rrbracket(\vec{x}_2), \vec{x}_3)).$$

Consequently, every derivation of G_* is simulated by a derivation of \mathcal{G}_D . \square

Therefore, once the learner gets a superset of D_* , it always conjectures an SCFTG that generates the target language. We also remark that the set D_* is not too big. D_* consists of at most $|P^*|$ positive examples and if we choose e_A and s_A to be minimal, each element in D_* is a minimal tree that is obtained using a rule of G_* .

Theorem 1. *Algorithm 1 identifies every r -substitutable SCFTG in the limit from positive data. It updates the conjecture in polynomial time and the number of examples needed to converge is bounded by a polynomial in the number of rules of the target grammar.*

3.3 Finite Environment Property

In [18], Clark defines the notion of a *Finite Context Property* for CFGs and proposes a corresponding learning algorithm. The learning setting he assumes is *identification in the limit from positive data and membership queries*, which is similar to the one from the previous subsection except that in addition the learner has access to a *membership oracle* to ask if a certain object is in the target language or not, and will receive an answer in constant time. This subsection defines an analogous property for SCFTGs and present a matching learning algorithm. We base our algorithm on the one proposed for strings in [19], which is a simpler version of Clark's original.

Definition 6. We say that an r -SCFTG G has the *p -Finite Environment Property* (p -FEP) if for every $A \in N_m$, there is $F_A \subseteq \mathbb{E}^m$ such that $|F_A| \leq p$ and

$$\mathcal{L}_A = \{ s \in \mathbb{S}^m \mid F_A \odot s \subseteq \mathcal{L}(G_*) \}.$$

We call such an environment set F_A a *characterizing environment set of A* .

Before giving the overall view of our learning algorithm (Algorithm 2), we describe the construction and important properties of the SCFTG that the learner maintains as its current conjecture.

Let G_* be our learning target, an r -SCFTG with the p -FEP, and $L_* = \mathcal{L}(G_*)$. Suppose we have two finite sets $S \subseteq \mathbb{S}$ and $E \subseteq \mathbb{E}$ such that $a(x_1, \dots, x_m) \in S$ for all $a \in \Sigma_m$ and $\#_0 \in E$, which are computed from given positive data in a way explained later. We define an SCFTG $\mathcal{G}_{r,p}(E, S) = \langle \Sigma, N, P, I \rangle$ as follows. While Algorithm 1 takes stubs as nonterminals, Algorithm 2 uses sets of environments as nonterminals.

$$\begin{aligned} N_m &= \{ \llbracket F \rrbracket \mid F \subseteq E \cap \mathbb{E}^m \text{ and } |F| \leq p \} \text{ for } m \leq r, \\ I &= \{ \llbracket \{\#_0\} \rrbracket \}. \end{aligned}$$

Note that $|N_m| \leq |E|^p$. We have rules of two types, which require the aid of the oracle to be determined:

- I. $\llbracket F \rrbracket \rightarrow a(x_1, \dots, x_m)$ with $a \in \Sigma_m$ if $F \odot a(x_1, \dots, x_m) \subseteq L_*$;

$$\text{II. } \llbracket F \rrbracket \rightarrow \llbracket F_1 \rrbracket(\vec{x}_1, \llbracket F_2 \rrbracket(\vec{x}_2), \vec{x}_3) \quad \text{if}$$

for all $s_1, s_2 \in S$, $F_1 \odot s_1, F_2 \odot s_2 \subseteq L_*$ implies $F \odot s_1[\vec{x}_1, s_2[\vec{x}_2], \vec{x}_3] \subseteq L_*$.

When learning an r -SCFTG with the p -FEP, we want each nonterminal $\llbracket F \rrbracket$ to derive s if and only if $F \odot s \subseteq L_*$, that is, we want F to be a characterizing environment set of $\llbracket F \rrbracket$. Conversely, for each nonterminal A of the target grammar G_* , we want our conjecture to simulate A by a nonterminal $\llbracket F_A \rrbracket$ where F_A is a characterizing environment set for A extracted from the given data. From this idea, rules of Type II of the form $\llbracket F \rrbracket \rightarrow \llbracket F_1 \rrbracket(\vec{x}_1, \llbracket F_2 \rrbracket(\vec{x}_2), \vec{x}_3)$ are justified if for all $s_1, s_2 \in \mathbb{S}$,

$$F_1 \odot s_1, F_2 \odot s_2 \subseteq L_* \text{ implies } F \odot s_1[\vec{x}_1, s_2[\vec{x}_2], \vec{x}_3] \subseteq L_* \quad (1)$$

since $s_1 \in \mathcal{L}_{\llbracket F_1 \rrbracket}, s_2 \in \mathcal{L}_{\llbracket F_2 \rrbracket}$ implies $s_1[\vec{x}_1, s_2[\vec{x}_2], \vec{x}_3] \in \mathcal{L}_{\llbracket F \rrbracket}$ under the presence of the rule. However, since \mathbb{S}^m is infinite for each m , one cannot check this property (II). Instead, we use the finite set S . This is the idea behind the rule construction.

We say that a rule of Type II is *correct* if (II) holds.

Lemma 6. *If $E \subseteq F$ then $\mathcal{L}(\mathcal{G}_{r,p}(E, S)) \subseteq \mathcal{L}(\mathcal{G}_{r,p}(F, S))$.*

Proof. Every rule of $\mathcal{G}_{r,p}(E, S)$ is also that of $\mathcal{G}_{r,p}(F, S)$. \square

Lemma 7. *If $S \subseteq T$ then $\mathcal{L}(\mathcal{G}_{r,p}(E, T)) \subseteq \mathcal{L}(\mathcal{G}_{r,p}(E, S))$.*

Proof. Every rule of $\mathcal{G}_{r,p}(E, T)$ is also that of $\mathcal{G}_{r,p}(E, S)$. \square

We want every rule of the conjectured grammar to be correct. The following argument shows that for every finite set $E \subseteq \mathbb{E}$, there is a finite set $S \subseteq \mathbb{S}$ such that every rule of $\mathcal{G}_{r,p}(E, S)$ is correct.

For $F \subseteq \mathbb{E}^m, F_1 \subseteq \mathbb{E}^k, F_2 \subseteq \mathbb{E}^j$ and $i \leq m - j$ such that $m = k + j - 1$, if a rule of Type II

$$\llbracket F \rrbracket \rightarrow \llbracket F_1 \rrbracket(x_1, \dots, x_i, \llbracket F_2 \rrbracket(x_{i+1}, \dots, x_{i+j}), x_{i+j+1}, \dots, x_m)$$

is not correct, there are $s_1, s_2 \in \mathbb{S}$ such that

$$F_1 \odot s_1, F_2 \odot s_2 \subseteq L_* \text{ and } F \odot s_1[x_1, \dots, x_i, s_2[x_{i+1}, \dots, x_{i+j}], x_{i+j+1}, \dots, x_m] \not\subseteq L_*.$$

If $s_1, s_2 \in S$, the incorrect rule is suppressed. Hence, $2r|N|^3 \leq 2r|E|^{3p}$ stubs suffice to suppress all incorrect rules. We say that S is *fiducial on* E if every rule of $\mathcal{G}_{r,p}(E, S)$ is correct.

Lemma 8. *If every rule of $\hat{G} = \mathcal{G}_{r,p}(E, S)$ is correct, we have $\mathcal{L}(\hat{G}) \subseteq L_*$.*

Proof. One can prove by induction that for any $\llbracket F \rrbracket \in N$ and $s \in \mathbb{S}$, whenever $\llbracket F \rrbracket \Rightarrow_{\hat{G}}^* s$, it holds that $F \odot s \subseteq L_*$. By definition of I , we have proven the lemma. \square

Lemma 9. *Let L_* be generated by an r -SCFTG G_* with the p -FEP. Then $L_* \subseteq \mathcal{L}(\mathcal{G}_{r,p}(E, S))$ if E includes a characterizing environment set F_A of A for every nonterminal A of G_* .*

Algorithm 2. Learning SCFTGs with the p -FEP

Data: A sequence of trees $t_1, t_2, \dots \in \mathcal{L}(G_*)$; membership oracle \mathcal{O} ;
Result: A sequence of r -SCFTGs G_1, G_2, \dots ;
let $D := \emptyset$; $E := \emptyset$; $S := \emptyset$; $\hat{G} := \mathcal{G}_{r,p}(E, S)$;
for $n = 1, 2, \dots$ **do**
 let $D := D \cup \{t_n\}$; $S := Sub^{\leq r}(D)$;
 if $D \not\subseteq \mathcal{L}(\hat{G})$ **then**
 let $E := Env^{\leq r}(D)$;
 end if
 output $\hat{G} = \mathcal{G}_{r,p}(E, S)$ as G_n ;
end for

Proof. Each nonterminal A of G_* is simulated by $\llbracket F_A \rrbracket$ in $\mathcal{G}_{r,p}(E, S)$ except the very first step of a derivation from an initial symbol of G_* , where that initial symbol of G_* is simulated by $\llbracket \{\#_0\} \rrbracket$ in $\mathcal{G}_{r,p}(E, S)$. \square

Due to the nice properties presented in the lemmas above, our learning algorithm is quite simple. Whenever we get a positive example that is not generated by our current conjecture, we expand E (see Lemma 6). On the other hand, to suppress incorrect rules, we keep expanding S (Lemma 7).

Theorem 2. Algorithm 2 identifies r -SCFTGs with the p -FEP in the limit from positive data and membership queries.

Proof. Let $D_n = \{t_1, \dots, t_n\}$. Lemma 9 ensures that Algorithm 2 does not update E infinitely many times, because characterizing environment sets of nonterminals in G_* are finite subsets of $Env(L_*)$ and at some point the learner will have seen all of them. Let $E_{m_0} = Env(D_{m_0})$ be the limit and $S_{n_0} = Sub(D_{n_0})$ be fiducial on E_{m_0} . Together with Lemma 8, we see that for any $n \geq \max\{m_0, n_0\}$, Algorithm 2 outputs $G_n = \mathcal{G}_{r,p}(E_{m_0}, S_{n_0})$ such that $\mathcal{L}(G_n) = \mathcal{L}(G_*)$. \square

Some remarks on the efficiency of our algorithm: It is easy to see that $\|E\|, \|S\| \in O(\|D\|^{r+1})$ and we have at most $|E|^p$ nonterminals. We need at most a polynomial number of membership queries to determine rules among those nonterminals. All in all, Algorithm 2 updates its conjecture in polynomial time in the size $\|D\|$ of the data D . Moreover, we do not need too much data. To get characterizing environments of all nonterminals, $p|N^*|$ examples are enough, where N^* is the set of nonterminals of the target SCFTG G_* . To suppress incorrect rules, $O(r|E|^{3p})$ stubs are enough by Lemma 8.

4 Conclusion

We have demonstrated how existing distributional learning techniques for CFGs can be naturally extended to SCFTGs by giving the necessary theoretical foundations and by discussing two concrete efficient algorithms for the distributional learning of SCFTGs in detail. These are just two examples for the potential of our

translation – in fact, any distributional algorithm for strings should be translatable into an algorithm for trees by this method. This includes other learning settings such as membership and equivalence queries – see [8, 5] for distributional algorithms based on this scenario.

We suggest *Tree Adjoining Grammars* (TAGS; [1]) as another prominent tree generating formalism that becomes learnable via our distributional techniques by modifying the definitions of substubs and environments accordingly. Moreover, the shift from strings to trees executed in this paper can be easily continued to context-free graph formalisms such as *hyperedge replacement grammars* [20] by giving a similar translation and specifying in an analogous manner how a learner would extract a sub-hypergraph from a given example of the target language to construct his hypothesis. We have focused on simple CFTGs as the most intuitive extension of CFGs to trees – a further conceivable direction for future work is the question of what happens if we allow subtrees to be duplicated.

Practical applications: As Clark [21] mentions in his conclusion, distributional learning techniques are adaptable to a probabilistic paradigm by focussing on the frequency with which the substrings and contexts of a language occur in (large) sets of given data. Hence it seems a promising approach for example in computational linguistics to consider applying probabilistically modified distributional techniques to large corpora of linguistic data such as the Penn treebank, from which a probabilistic CFG or also a TAG can be extracted (see [22], [23]), once we have specified distributional algorithms for TAGs as suggested above.

Acknowledgement. The authors are grateful to Alexander Clark and the anonymous reviewers for valuable comments that have improved the quality of this paper.

References

- [1] López, D., Sempere, J.M., García, P.: Inference of reversible tree languages. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 34(4), 1658–1665 (2004)
- [2] Drewes, F., Höglberg, J.: Learning a regular tree language from a teacher. In: Ésik, Z., Fülöp, Z. (eds.) *DLT 2003. LNCS*, vol. 2710, pp. 279–291. Springer, Heidelberg (2003)
- [3] Besombes, J., Marion, J.Y.: Learning tree languages from positive examples and membership queries. *Theoretical Computer Science* 382, 183–197 (2007)
- [4] Oncina, J., Garcia, P.: Inference of recognizable tree sets. Technical report, DSIC II/47/93, Universidad de Valencia (1993)
- [5] Shirakawa, H., Yokomori, T.: Polynomial-time MAT learning of c-deterministic context-free grammars. *Transaction of Information Processing Society of Japan* 34, 380–390 (1993)
- [6] Clark, A., Eyraud, R.: Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research* 8, 1725–1745 (2007)
- [7] Clark, A., Eyraud, R., Habrard, A.: Using contextual representations to efficiently learn context-free languages. *Journal of Machine Learning Research* 11, 2707–2744 (2010)
- [8] Clark, A.: Distributional learning of some context-free languages with a minimally adequate teacher. In: [24], pp. 24–37

- [9] Clark, A.: Towards general algorithms for grammatical inference. In: Hutter, M., Stephan, F., Vovk, V., Zeugmann, T. (eds.) ALT 2010. LNCS, vol. 6331, pp. 11–30. Springer, Heidelberg (2010)
- [10] Yoshinaka, R.: Efficient learning of multiple context-free languages with multi-dimensional substitutability from positive data. *Theor. Comput. Sci.* 412(19), 1821–1831 (2011)
- [11] Joshi, A.K.: Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural description. In: Dowty, D., Karttunen, L., Zwicky, A. (eds.) Natural Language Processing, Cambridge University Press, Cambridge (1985)
- [12] Yoshinaka, R., Kanazawa, M.: Distributional learning of abstract categorial grammars. In: Pogodalla, S., Prost, J.-P. (eds.) LACL. LNCS, vol. 6736, pp. 251–266. Springer, Heidelberg (2011)
- [13] Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications (2008)
- [14] Seki, H., Kato, Y.: On the generative power of multiple context-free grammars and macro grammars. *IEICE Transactions* 91-D(2), 209–221 (2008)
- [15] Kanazawa, M., Salvati, S.: The copying power of well-nested multiple context-free grammars. In: Dediu, A.-H., Fernau, H., Martín-Vide, C. (eds.) LATA 2010. LNCS, vol. 6031, pp. 344–355. Springer, Heidelberg (2010)
- [16] Lautemann, C.: The complexity of graph languages generated by hyperedge replacement. *Acta. Inf.* 27(5), 399–421 (1990)
- [17] Gold, E.M.: Language identification in the limit. *Information and Control* 10(5), 447–474 (1967)
- [18] Clark, A.: Learning context free grammars with the syntactic concept lattice. In: [24], pp. 38–51
- [19] Yoshinaka, R.: Towards dual approaches for learning context-free grammars based on syntactic concept lattices. In: Mauri, G., Leporati, A. (eds.) DLT 2011. LNCS, vol. 6795, pp. 429–440. Springer, Heidelberg (2011)
- [20] Habel, A., Kreowski, H.: Some structural aspects of hypergraph languages generated by hyperedge replacement. In: Brandenburg, F.J., Wirsing, M., Vidali-Naquet, G. (eds.) STACS 1987. LNCS, vol. 247, pp. 207–219. Springer, Heidelberg (1987)
- [21] Clark, A.: Efficient, correct, unsupervised learning of context-sensitive languages. In: Proceedings of CoNLL. Association for Computational Linguistics, Uppsala (2010)
- [22] Charniak, E.: Tree-bank grammars. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence, pp. 1031–1036 (1996)
- [23] Chen, J., Bangalore, S., Vijay-Shanker, K.: Automated extraction of tree-adjoining grammars from treebanks. *Nat. Lang. Eng.* 12, 251–299 (2006)
- [24] Sempere, J.M., García, P. (eds.): ICGI 2010. LNCS, vol. 6339. Springer, Heidelberg (2010)

On Noise-Tolerant Learning of Sparse Parities and Related Problems

Elena Grigorescu*, Lev Reyzin**, and Santosh Vempala***

School of Computer Science
Georgia Institute of Technology
266 Ferst Drive, Atlanta GA 30332
`{elena,lreyzin,vempala}@cc.gatech.edu`

Abstract. We consider the problem of learning sparse parities in the presence of noise. For learning parities on r out of n variables, we give an algorithm that runs in time $\text{poly}\left(\log \frac{1}{\delta}, \frac{1}{1-2\eta}\right) n^{(1+(2\eta)^2+o(1))r/2}$ and uses only $\frac{r \log(n/\delta)\omega(1)}{(1-2\eta)^2}$ samples in the random noise setting under the uniform distribution, where η is the noise rate and δ is the confidence parameter. From previously known results this algorithm also works for adversarial noise and generalizes to arbitrary distributions. Even though efficient algorithms for learning sparse parities in the presence of noise would have major implications to learning other hypothesis classes, our work is the first to give a bound better than the brute-force $O(n^r)$. As a consequence, we obtain the first nontrivial bound for learning r -juntas in the presence of noise, and also a small improvement in the complexity of learning DNF, under the uniform distribution.

1 Introduction

Designing efficient, noise-tolerant, learning algorithms is a fundamental and important problem in part because real-world data is often corrupted, and algorithms unable to handle errors in their training data are not practically deployable. Angluin and Laird [3] formalize this notion in the “noisy PAC” setting, where the learner is required to achieve an error of ϵ with probability $1 - \delta$, where the labels of the training examples are flipped at random with probability equal to the noise rate η . The statistical query (SQ) model [15] tries to capture the properties of noise-tolerant algorithms; algorithms implementable in the SQ model also work in the noisy PAC model of Angluin and Laird. Kearns’s characterization has made it apparent that most algorithms that work in the noise-free setting can be adapted to work in the presence of noise.

* This material is based upon work supported by the National Science Foundation under Grant #1019343 to the Computing Research Association for the CIFellows Project.

** Supported by the Simons Postdoctoral Fellowship.

*** Supported by National Science Foundation awards AF-0915903 and AF-0910584.

The **learning parity with noise (LPN)** problem, however, is one notable exception – techniques for learning parities in the noise-free setting cannot be extended to the noisy PAC model. In the LPN problem, the target is a parity on an unknown subset of variables, and the learning task is to discover this subset. More formally, the algorithm receives random examples $x \in \{0, 1\}^n$ labeled by $\ell \in \{0, 1\}$ obtained as $\ell = c^* \cdot x + e$, where $c^* \in \{0, 1\}^n$ is the hidden target parity and $e \in \{0, 1\}$ is a random bit set to 1 with probability η .

While the parity problem can be solved efficiently in the noise-free PAC setting using Gaussian elimination, the search of an efficient noise-tolerant algorithm has run into serious barriers. Most notably, the parity problem is unconditionally known not to be learnable in the statistical query model [5, 15], where a brute-force search is nearly optimal. The state-of-the-art algorithm, due to Blum et al. [6], runs in time $2^{O(n/\log n)}$, and it has not been improved since its publication over a decade ago. Their approach, a clever adaptation of Gaussian elimination, seems to reach its limits when trying to push these bounds further, which implies that either fundamentally new algorithms or new hardness results are needed for this problem.

The LPN problem is not only notoriously difficult, but also ubiquitous. It is closely related to the famous problem of decoding random linear codes in coding theory, and cryptosystems use the hardness of LPN and of the Learning With Errors problem (its generalization over larger rings) as a security assumption [4, 22, 23]. A variant of LPN is a candidate problem for automated tests to tell humans apart from computers [11], and as we discuss in the next section a large class of function families reduce to the LPN problem, showing its central importance to learning theory.

1.1 Sparse Parity

In this paper we focus on a variant of the LPN problem, where the target parity c^* has the additional property that it is sparse, namely only r of the n bits of the target c^* are set to 1. The sparse parity problem plays an important role in learning, specially due to its connections to r -juntas (i.e. functions depending on only r of the n variables), and DNF recently exhibited by Feldman et al. [9]. Namely, they show that an algorithm for learning noisy r -sparse parities running in time polynomial in n implies a polynomial time algorithm for learning r -juntas (Theorem 3). Similarly, learning s -term DNF can be reduced to learning noisy $(\log s)$ -parities. Furthermore, they also show that learning r -parities corrupted by an η fraction of random noise is at least as hard as the intuitively more difficult task of learning an r -parity corrupted by adversarial noise of rate η . Until this work, however, no algorithm better than the brute force $O(n^r)$ has been exhibited for the sparse parity problem.

Our paper is the first to improve on this brute-force bound and gives a $\text{poly}\left(\log \frac{1}{\delta}, \frac{1}{1-2\eta}\right) \cdot n^{(1+(2\eta)^2+o(1))r/2}$ algorithm for the uniform distribution (Corollary 11) and holds even for adversarial noise. This bound also generalizes to learning r -parities with random classification noise for arbitrary distributions

(Theorem 5). Moreover, our sample complexity of $\frac{r \log(n/\delta)\omega(1)}{(1-2\eta)^2}$ almost meets the information theoretic lower bound. These bounds are comparable to the results obtained by Klivans and Servedio [16] in the “attribute efficient” noiseless model, which have been subsequently improved by Buhrman et al. [7]. Interestingly, the same running time of $\tilde{O}(n^{r/2})$ is obtained in [7] for learning parities in the related “mistake-bound” model, but this is again in the noiseless case.

1.2 Implications to Related Problems

Given the state of the art for r -parities, clearly no algorithm better than $O(n^r)$ has been found for r -juntas. However, due to connections between parities and juntas discovered by Feldman et al. [9], our results have immediate implications for learning r -juntas. Namely, we obtain an algorithm for learning noisy r -juntas that runs in time $\text{poly}\left(\log \frac{1}{\delta}, \frac{1}{1-2\eta}, 2^r\right) n^{(1-2^{1-r}(1-2\eta)+2^{1-2r}(1-2\eta)^2+o(1))r}$ (Corollary 2). For the uniform distribution without noise, Mossel et al. [19] show that juntas can be learned in time $O(n^{\frac{\omega r}{\omega+1}})$, where ω is the matrix multiplication exponent. Our bound, however, is the first nontrivial result for noisy juntas.

Our sparse parity algorithm also has implications to learning DNF under the uniform distribution, again using the reduction of Feldman et al. [9] from DNF to noisy parities. While there has been significant recent progress on the problem, including that random DNF are learnable under the uniform distribution [12, 24, 25], virtually nothing is known about their learnability in the worst case, even in the classical noiseless PAC model. In fact, there are serious impediments to learning DNF, including hardness results for their proper learnability [1]. The previous best algorithm for learning s -term DNF from random examples is due to Verbeurgt [27] and runs in quasipolynomial time $O(n \log \frac{s}{\epsilon})$, where ϵ is the error rate. Our algorithm leads to an improved bound of $\text{poly}\left(\log \frac{1}{\delta}, \frac{1}{\epsilon}, s\right) n^{(1-\tilde{O}(\epsilon/s)+o(1)) \log \frac{s}{\epsilon}}$ for this problem in the uniform setting (Corollary 4).

1.3 Our Approach

Our algorithm works roughly as follows. We consider all $\frac{r}{2}$ -parities and evaluate them on m examples. For each parity we can view these evaluations as coordinates for points on the m -dimensional Hamming cube, creating the set \mathcal{H}_1 . Similarly, we consider all these high dimensional points XOR-ed coordinate-wise with the respective labels, creating the set \mathcal{H}_2 . This gives us a set \mathcal{H} of $2\binom{n}{r/2}$ points. Notice that in the noiseless case, two $\frac{r}{2}$ -parities comprising the target c^* have the property that the corresponding point in \mathcal{H}_1 of one of them is at Hamming distance 0 from the corresponding point in \mathcal{H}_2 of the other. Moreover, two points not comprising the target are far apart – this is a key point employed in our exact learning algorithm for the uniform distribution. In the presence of noise these distances are perturbed, and to actually find two nearby points corresponding to the good parities we rely upon the recent approximate nearest neighbor / closest pair algorithm of Andoni and Indyk [2]. A careful analysis of

this approach when generalizing this observation to arbitrary distributions yields a poly $\left(\log \frac{1}{\delta}, \frac{1}{\epsilon}, \frac{1}{1-2\eta}\right) n^{(1+(\frac{\eta}{\epsilon+\eta-2\epsilon\eta})^2+o(1))r/2}$ running time for proper learning (Theorem 5). Furthermore, we can use the above approach to design a learner that gives improved running time of poly $\left(n, \log \frac{1}{\delta}, \frac{1}{\epsilon-\eta}, \frac{1}{1-2\eta}\right) n^{(1+(2\eta)^2+o(1))r/2}$ (Theorem 6) for arbitrary distributions, but it only works for error rates up to the noise rate.

Our approach is inspired by Hopper and Blum [11], who, in giving an overview of candidate hard problems to use for secure human identification, informally suggested a similar idea for an $O(n^{r/2})$ algorithm for a closely related problem. Their suggestion indeed works for the noiseless setting, but runs into unforeseen difficulty when noise is introduced. We remark that Buhrman et al. [7] also invoke the Hopper and Blum [11] approach for their results on learning sparse parities in the mistake bound model, and they also note that the idea of using $\frac{r}{2}$ -parities appears in results of Klivans and Servedio [16] (who in turn cite a personal communication with Spielman). All these references, however, only treat the noise-free model under the uniform distribution.

Following up on this idea, we are able to handle large noise rates and arbitrary distributions.

1.4 Past Work

In their seminal work on the SQ model for noise-tolerant learning, Kearns [15] and Blum et al. [5] prove unconditional lower bounds for both the sparse and unrestricted versions of LPN of $\Omega(n^{r/c})$ and $\Omega(2^{n/c})$ (for constants $c > 1$), respectively. In the SQ model, the learner can ask the oracle statistical properties of the target, and this proves insufficient to efficiently learn parities. Then, in a breakthrough result, Blum et al. [6] show that in the noisy PAC model (see Section 2 for definition), one can circumvent the SQ lower bound of $2^{n/c}$ and give a Gaussian elimination type algorithm that runs in time $2^{O(n/\log n)}$. By considering parities on the first $\log n \log \log n$ bits of an example, they separate the classes SQ and noisy PAC. For the sparse r -parity problem, no similar breakthrough has occurred, and no algorithm better than the brute force $O(n^r)$ is known (before this work) for the noisy PAC model. On the other hand, if membership queries are allowed, the parity problem is solvable in polynomial time [10, 17].

Another interesting direction in the literature is that of establishing non-trivial tradeoffs between the sample and time complexity of the LPN problem, both in the noisy and noise-free versions. In the *noiseless* model for r -parities, one can trivially obtain an algorithm with $O(r \log n)$ sample complexity that runs in time $O(n^r)$, and this is improved to $O(n^{r/2})$ by Klivans and Servedio [16]. Additionally, Buhrman et al. [7] give a $o(n)$ sample complexity for a running time of $2^{O(r)+\log n}$, and Klivans and Servedio [16] and Buhrman et al. [7] also show polynomial time algorithms with a sample complexity of $\tilde{O}(n^{1-\frac{1}{r}})$.

In the *noisy* PAC model, the Blum et al. [6] result (for parities of unrestricted size) requires as many as $2^{O(n/\log n)}$ samples, but it works for any noise rate $\eta <$

$\frac{1}{2} - \exp(-n^\delta)$. This sample complexity has been improved by Lyubashevsky [18] to $O(n^{1+\epsilon})$ with higher running time and noise rate, and it remains open whether this can be further improved to a sample complexity of $O(n)$.

2 Notation and Preliminaries

In this paper, we are concerned with the model of **PAC learning under random classification noise** [3]. Let $c^* \in C : X \rightarrow \{0, 1\}$ be the target concept, and D a distribution over X . In this model the learner has access to the oracle $\text{EX}_\eta(c^*, D)$, which chooses $x \sim D$ and returns $(x, \ell(x))$, which is $(x, c^*(x))$ with probability $1 - \eta$ and $(x, 1 - c^*(x))$ with probability η .

Definition 1. Algorithm L is said to PAC learn class $C : X \rightarrow \{0, 1\}$ by class H in the presence of noise if, $\forall c^* \in C$, distribution D over X , error rate $0 < \epsilon < \frac{1}{2}$, failure rate $0 < \delta < \frac{1}{2}$, noise rate $0 \leq \eta < \frac{1}{2}$, if L is given inputs ϵ, δ, η and access to $\text{EX}_\eta(c^*, D)$, then it will output hypothesis $h \in H$ s.t. with probability $1 - \delta$

$$\Pr_{x \sim D}[h(x) \neq c^*(x)] < \epsilon.$$

If $H = C$ then we say the algorithm learns **properly**, otherwise it learns **improperly**. If the algorithm can recover c^* , we say it learns **exactly**. When we consider learning boolean functions under the **uniform distribution**, we restrict our attention to the distribution D over $X = \{0, 1\}^n$ that assigns a probability of $\frac{1}{2^n}$ to each length n vector. When we consider a **noiseless** setting, we mean $\eta = 0$; this is the classical model of PAC learning [26]. Finally, if we relax the learning requirement to ask the algorithm to achieve only an error $< 1/2 - \gamma$, then we say the algorithm is a **γ -weak learner** for C .

Now we define the problem of learning r -parities. We note that operations on parities are performed modulo 2.

Definition 2. In the (sparse) r -parity problem: the example domain is $X = \{0, 1\}^n$, the target class C consists of vectors c in $\{0, 1\}^n$ s.t. $\|c\|_1 = r$, and the target $c^* \in C$ labels examples $x \in X$ by $c^*(x) = \sum_i^n x_i c_i^* \in \{0, 1\}$.

Next we state some results from the literature that will be useful in our proofs. The first is an approximate closest pair algorithm that our algorithm in Section 3 relies upon.

Theorem 1 (Corollary of Andoni and Indyk [2]). Given N points on the d -dimensional Hamming cube, finding a pair of points whose distance is within a factor $\rho > 1$ from the distance of the closest pair¹ can be done in time

$$O(dN^{1+1/\rho^2 + O(\log \log N / \log^{1/3} N)}) = O(dN^{1+1/\rho^2 + o(1)}).$$

¹ This is effectively done by running an approximate nearest neighbor algorithm on each point in the data structure.

We use the next theorem in Section 4 for an improved algorithm for improper learning of parities for arbitrary distributions.

Theorem 2 (Kalai and Servedio [13]). *For any $0 < \eta < \epsilon < \frac{1}{2}$, $0 < \delta < \frac{1}{2}$, there exists a boosting algorithm which, given access to a noise tolerant γ -weak learner and an example oracle $\text{EX}_\eta(D, c^*)$, runs in time $\text{poly}(\log \frac{1}{\delta}, \frac{1}{\epsilon-\eta}, \frac{1}{\gamma}, \frac{1}{1-2\eta})$, and with probability δ outputs a hypothesis h such that $\Pr_{x \sim D}[h(x) \neq c^*(x)] < \epsilon$.*

The following theorems are used in Section 5 to give improved bounds for learning juntas and DNF.

Theorem 3 (Feldman et al. [9]). *Let A be an algorithm that learns parities of r variables on $\{0, 1\}^n$, under the uniform distribution, for noise rate $\eta' \leq \frac{1}{2}$ in time $T(n, r, \eta')$. Then there exists an algorithm that exactly learns r -juntas under the uniform distribution with noise rate η in time*

$$O(r2^{2r}T(n, r, 1/2 - 2^{-r}(1 - 2\eta))).$$

Theorem 4 (Feldman et al. [9]). *Let A be an algorithm that learns parities of r variables on $\{0, 1\}^n$, under the uniform distribution, for noise rate $\eta \leq \frac{1}{2}$ in time $T(n, r, \eta)$ and sample complexity $S(n, r, \eta)$. Then there exists an algorithm that PAC learns s -term DNF under the uniform distribution in time*

$$\tilde{O}\left(\frac{s^4}{\epsilon^2} T\left(n, \log(\tilde{O}(s/\epsilon)), 1/2 - \tilde{O}(\epsilon/s)\right) \cdot S^2\left(n, \log(\tilde{O}(s/\epsilon)), 1/2 - \tilde{O}(\epsilon/s)\right)\right).$$

3 Learning Sparse Parities

We begin by presenting our algorithm for r -parities and afterwards prove its correctness and running time. As discussed in Section 1.3, our algorithm tries to find two “nearby” $\frac{r}{2}$ -parities that compose to form the correct parity. We do this by evaluating all the parities on a sufficiently large set of examples and finding an approximate closest pair according to the evaluations and the evaluations XORed with the labels. The exact procedure appears in Algorithm II

Algorithm 1. Learn r -Parities $(r, n, \epsilon, \delta, \eta)$

- 1: Obtain a set $\hat{X} = \{x_1, \dots, x_m\}$ of examples drawn from the oracle $\text{EX}_\eta(c^*, D)$, where $m = \frac{r \log(n/\delta)\omega(1)}{(\epsilon'-\eta)^2}$ and $\epsilon' = \epsilon + \eta - 2\epsilon\eta$.
 - 2: For each $\frac{r}{2}$ -parity c , evaluate it on \hat{X} to obtain the corresponding $\langle c \cdot x_1, c \cdot x_2, \dots, c \cdot x_m \rangle \in \{0, 1\}^m$ and $\langle c \cdot x_1 + \ell(x_1), c \cdot x_2 + \ell(x_2), \dots, c \cdot x_m + \ell(x_m) \rangle \in \{0, 1\}^m$. Let \mathcal{H} be the set of all these $2 \cdot \binom{n}{r/2}$ points on the Hamming cube.
 - 3: Run the Approximate Closest Pair algorithm from Theorem II on \mathcal{H} with the approximation parameter $\rho = \epsilon'/\eta$, to obtain the closest pair of points in $\{0, 1\}^m$ with corresponding $\frac{r}{2}$ -parities c_1 and c_2 , respectively.
 - 4: Return $c_1 + c_2$.
-

Now we state and prove our main theorem.

Theorem 5. For any $0 < \epsilon < \frac{1}{2}$, $0 < \delta < \frac{1}{2}$, $0 \leq \eta < \frac{1}{2}$, and distribution D over $\{0, 1\}^n$, the class of r -parities can be properly PAC learned with random classification noise using $\frac{r \log(n/\delta)\omega(1)}{\epsilon^2(1-2\eta)^2}$ samples in time

$$\frac{\log(1/\delta)n^{(1+(\frac{\eta}{\epsilon+\eta-2\epsilon\eta})^2+o(1))r/2}}{\epsilon^2(1-2\eta)^2}.$$

Proof. For convenience, in this proof, we define the quantity ϵ' to be the error we need to achieve on the *noisy* examples, drawn from $\text{EX}_\eta(c^*, D)$. There is a simple relationship between ϵ' and the quantities ϵ and η :

$$\begin{aligned}\epsilon' &= 1 - ((1-\epsilon)(1-\eta) + \epsilon\eta) \\ &= \epsilon + \eta - 2\epsilon\eta.\end{aligned}$$

Note that $\epsilon' > \eta$. To analyze Algorithm II, we first define the *empirical agreement* of a parity c on a sample $\hat{X} = \{x_1, x_2, \dots, x_m\}$ as

$$\hat{\text{agr}}_{\hat{X}}(c) = \sum_{x \in \hat{X}} c \cdot x.$$

We define the set \mathcal{B} of bad parities c' as those whose error according to the examples chosen from the noisy oracle is $\geq \epsilon'$, as in $c' \in \mathcal{B}$ iff

$$\Pr_{x \sim \text{EX}_\eta(c^*, D)}[c'(x) \neq \ell(x)] \geq \epsilon'.$$

If we are able to find a parity not in the bad set, we will succeed in learning.

The empirical agreement of an r -parity $c' \in \mathcal{B}$ can be bounded by Hoeffding's inequality as follows:

$$\Pr_{\hat{X} \sim \text{EX}_\eta(c^*, D)} \left[\hat{\text{agr}}_{\hat{X}}(c') - \mathbf{E}_{\hat{X} \sim \text{EX}_\eta(c^*, D)}[\hat{\text{agr}}_{\hat{X}}(c')] > t \right] < e^{-t^2/m}.$$

By the union bound we have that $\forall c_i, c_j$ s.t. $c_i + c_j = c' \in \mathcal{B}$,

$$\Pr_{\hat{X} \sim \text{EX}_\eta(c^*, D)} \left[\hat{\text{agr}}_{\hat{X}}(c') - \mathbf{E}_{\hat{X} \sim \text{EX}_\eta(c^*, D)}[\hat{\text{agr}}_{\hat{X}}(c')] > t \right] < n^r e^{-t^2/m}.$$

Hence $t = \sqrt{mr \log(n/\delta)}$ suffices to bound by $(1-\delta)$ the probability that *all* pairs of $n^{r/2}$ agree on no more than $t + \mathbf{E}[\hat{\text{agr}}_{\hat{X}}(c')]$ positions. We can now, with probability $1-\delta$, bound the maximum agreement between two parities comprising a parity in \mathcal{B} by

$$\begin{aligned}\max_{c' \in \mathcal{B}} (\hat{\text{agr}}_{\hat{X}}(c')) &\leq \max_{c' \in \mathcal{B}} (\mathbf{E}[\hat{\text{agr}}_{\hat{X}}(c')]) + \sqrt{mr \log(n/\delta)} \\ &\leq (1-\epsilon')m + \sqrt{mr \log(n/\delta)}.\end{aligned}\tag{1}$$

Furthermore, we know that $\mathbf{E}[\hat{\text{agr}}_{\hat{X}}(c^*)] = (1-\eta)m$ and can similarly bound from below the empirical agreement $\hat{\text{agr}}_{\hat{X}}(c^*)$ to get with probability $1-\delta$,

$$\hat{\text{agr}}_{\hat{X}}(c^*) \geq (1-\eta)m - \sqrt{m \log(1/\delta)}.\tag{2}$$

We now rely on the following observation. If two $\frac{r}{2}$ -parities c_1, c_2 comprise the target c^* , i.e. $c_1 + c_2 = c^*$, then their corresponding points $\langle c_1 \cdot x_1, c_1 \cdot x_2, \dots, c_1 \cdot x_m \rangle$ and $\langle c_2 \cdot x_1 + \ell(x_1), c_2 \cdot x_2 + \ell(x_2), \dots, c_2 \cdot x_m + \ell(x_m) \rangle$ in \mathcal{H} are, by Equation 11 w.h.p. within Hamming distance $m - (1 - \epsilon')m - \sqrt{mr \log(n/\delta)}$, whereas if $c_1 + c_2 \in \mathcal{B}$, then by Equation 2, these points are at distance at least $m - (1 - \eta)m + \sqrt{m \log(1/\delta)}$. Hence by finding an approximate closest pair, with parameters properly set, we can find a pair c_1, c_2 such that $c_1 + c_2 \notin \mathcal{B}$, which suffices for learning. To do this, we appeal to Theorem 11 where we can set $N = O(n^{r/2})$ (the number of half-parities) and $d = m$ (the sample complexity).

We choose²

$$m = \frac{r \log(n/\delta) \omega(1)}{(\epsilon' - \eta)^2} \quad (3)$$

and

$$\begin{aligned} \rho &= \frac{m - (1 - \epsilon')m - \sqrt{mr \log(n/\delta)}}{m - (1 - \eta)m + \sqrt{m \log(1/\delta)}} \\ &= \frac{\epsilon' - (\epsilon' - \eta)/\sqrt{\omega(1)}}{\eta + (\epsilon' - \eta)\sqrt{\log(1/\delta)}/\sqrt{r \log(n/\delta) \omega(1)}} \\ &= \frac{\epsilon'}{\eta} - o(1). \end{aligned}$$

This ensures that the method in Theorem 11 will return two half parities composing a parity of error $< \epsilon'$ with probability $\geq 1 - 2\delta$.

All that is left is to analyze the running time of the method above, which, using Theorem 11 gives

$$O(dN^{1+1/\rho^2+o(1)}) = O\left(mn^{\left(1+\left(\frac{\eta}{\epsilon'}\right)^2+o(1)\right)r/2}\right).$$

Substituting in m from Equation 3 and substituting $\epsilon' = \epsilon + \eta - 2\epsilon\eta$ gives the statement of the theorem. \square

For all settings of ϵ and η this beats the brute-force $O(n^r)$ bound.

Corollary 1. *For all $0 < \delta < \frac{1}{2}$ and $0 \leq \eta < \frac{1}{2}$, the class of r -sparse parities can be learned exactly under the uniform distribution using $m = \frac{r \log(n/\delta) \omega(1)}{(1-2\eta)^2}$ samples and a running time of*

$$\frac{\log(1/\delta) n^{\left(1+(2\eta)^2+o(1)\right)r/2}}{(1-2\eta)^2}.$$

This bound holds even for adversarial noise.

Proof. We set $\epsilon = 1/2$ in Theorem 5 and note that because every wrong parity has error 1/2, if a parity has true error rate below 1/2, it must be correct, and the target is therefore learned exactly. Furthermore, Feldman et al. [9] show that for the uniform distribution, an algorithm for learning r -parities for random noise works for adversarial noise, without a blow-up in the running time. \square

² Note that $\omega(1) = \omega_n(1)$.

4 Improved Bounds for Improper Learning of r -Parities

In this section we present an algorithm which gives up on proper learning, but can learn parities under noise without a dependence on the error rate ϵ in the exponent. The following theorem holds for $\epsilon > \eta$ and uses the noise-tolerant boosting algorithm of Kalai and Servedio [13].

Theorem 6. *For any $0 < \epsilon < \frac{1}{2}$, $0 < \delta < \frac{1}{2}$, $0 \leq \eta < \frac{1}{2}$, and distribution D over $\{0, 1\}^n$, the class of r -parities can be learned improperly in time*

$$\text{poly}\left(n, \log \frac{1}{\delta}, \frac{1}{\epsilon - \eta}, \frac{1}{1 - 2\eta}\right) n^{(1+(2\eta)^2+o(1))r/2}.$$

Proof. Our idea here is to use the argument in Theorem 5 in order to obtain a parity c' such that

$$\mathbf{Pr}_{x \sim \text{EX}_\eta(c^*, D)}[c'(x) \neq \ell(x)] < 1/2 - 1/n.$$

In order to do this we use the approximation factor ρ in the nearest neighbor search set to

$$\rho = \frac{1}{\eta}(1/2 - 1/n) = \frac{1}{2\eta} - o(1).$$

This gives us a noise-tolerant $\frac{1}{n}$ -weak learner in time

$$\text{poly}(\log \frac{1}{\delta}, \frac{1}{\epsilon - \eta}, \frac{1}{1 - 2\eta}) n^{(1+(2\eta)^2+o(1))r/2},$$

which can be further used together with Theorem 2 to give us the final improved result. This multiplies our running time and sample complexity by a factor of $\text{poly}(n)$, which goes into the $o(1)$ in the exponent in our bound. \square

5 Application to Learning Juntas and DNF

Using the results of the previous section and that learning juntas with noise reduces to learning parities with noise under the uniform distribution [9], we get an algorithm for learning sparse juntas with noise better than by brute-force.

Theorem 3 implies the following Corollary.

Corollary 2. *For all $0 < \delta < \frac{1}{2}$, $0 \leq \eta < \frac{1}{2}$, r -juntas on n variables can be learned exactly in time*

$$\text{poly}\left(\log \frac{1}{\delta}, \frac{1}{1 - 2\eta}, 2^r\right) n^{(1-2^{1-r}(1-2\eta)+2^{1-2r}(1-2\eta)^2+o(1))r}$$

under the uniform distribution.

Proof. We combine Corollary 1 and Theorem 3 to get that r -juntas can be learned in time

$$r2^{2r} \frac{\log(1/\delta)n^{(1+(2\eta')^2+o(1))r/2}}{(1-2\eta')^2},$$

where $\eta' = 1/2 - 2^{-r}(1-2\eta)$. Replacing η' completes the proof. \square

We can now specialize Corollary 2 for the noiseless case.

Corollary 3. *For all $0 < \delta < \frac{1}{2}$, r -juntas with n variables can be learned exactly in time*

$$\text{poly}\left(\log\frac{1}{\delta}, 2^r\right) n^{(1-2^{1-r}+2^{1-2r}+o(1))r}$$

in the noise-free setting, under the uniform distribution.

We end this section by stating the implication to DNF of our Corollary 1 and Theorem 4 of Feldman et al. [9].

Corollary 4. *For all $0 < \epsilon < \frac{1}{2}$, $0 < \delta < \frac{1}{2}$, the class s -term DNF can be learned under the uniform distribution in time*

$$\text{poly}\left(\log\frac{1}{\delta}, \frac{1}{\epsilon}, s\right) n^{(1-\tilde{O}(\epsilon/s)+o(1))\log\frac{s}{\epsilon}}.$$

We note that the log in the exponent is base 2. We further recall that, from Theorem 1, the $o(1)$ term in the exponent is $\log\log N/\log^{1/3} N$, where $N = n^{\log(\tilde{O}(s/\epsilon))}$. Therefore, the bound above is an improvement over Verbeurgt's bound [27] of $O(n^{\log\frac{s}{\epsilon}})$ when $s/\epsilon = o\left(\frac{\log^{1/3} n}{\log\log n}\right)$.

6 Discussion

In this paper, we give an algorithm for learning r -parities in time essentially $n^{r/2}$ and show implications of this result to related problems. Our results draw attention to a nice set of open problems related to the sparse version of LPN. We give a proper algorithm running in time $\text{poly}\left(\log\frac{1}{\delta}, \frac{1}{1-2\eta}\right) n^{(1+(2\eta)^2+o(1))r/2}$ for the uniform distribution and $\text{poly}\left(\log\frac{1}{\delta}, \frac{1}{\epsilon}, \frac{1}{1-2\eta}\right) n^{(1+(\frac{\eta}{\epsilon+\eta-2\epsilon\eta})^2+o(1))r/2}$ for arbitrary distributions, for the r -sparse LPN problem.

For improper learning, we give an $\text{poly}\left(n, \log\frac{1}{\delta}, \frac{1}{\epsilon-\eta}, \frac{1}{1-2\eta}\right) n^{(1+(2\eta)^2+o(1))r/2}$ algorithm, which requires $\epsilon > \eta$ and uses $\text{poly}(n)$ samples. Obtaining an algorithm without the restriction of $\epsilon > \eta$, yet without the reliance on ϵ in the exponent, would be an interesting direction. One observation is that an improper learning algorithm achieving arbitrary low error can be converted to a proper learning algorithm for the LPN problem by drawing n examples from D , labeling them with the low-error algorithm (with the error parameter set $< \epsilon/n$), and running Gaussian elimination to find the correct parity. We note that a similar

technique is used in Blum et al. [4] to obtain a proper learning algorithm for linear threshold functions. Another interesting direction would be to remove the dependence on η in the exponent.

We note that it is tempting to try to push the approach taken in the proof of Theorem 5 further by considering, say, $\frac{r}{3}$ -parities. To improve our $n^{r/2}$ bound asymptotically, we would need an algorithm that, given a set of N points in the Hamming cube, finds 3 of them that ‘approximately sum’ up to 0, and runs in time substantially better than N^2 . This problem is somewhat related to the famous 3-SUM question in computational geometry which asks if among N elements of a set of integers there exist 3 that sum to 0. Erickson [8] presents the N^2 as a barrier and shows it is intrinsic in many difficult problems, giving some weak evidence that extending our approach in this way is unlikely. We also point out that the nearest neighbor algorithm of Andoni and Indyk [2] runs in time essentially N^{1+1/ρ^2} and almost matches the lower bounds for data-structures [20, 21], suggesting again that obtaining even a $n^{\frac{r}{3}}$ algorithm for the r -parity problem may require fundamentally new techniques. It remains open whether a polynomial time algorithm exists for learning $\omega(1)$ -parities.

The implication of our results to learning juntas brings up immediate questions of whether one can extend the non-trivial bound from Corollary 2 to arbitrary distributions, and furthermore, whether it is possible to improve the running time to n^{cr} , for some constant $c < 1$. As before, an important open problem is whether a polynomial time algorithm exists for learning $\omega(1)$ -juntas.

Acknowledgments. We thank Avrim Blum and Adam Kalai for very helpful discussions and Alex Andoni for references on nearest neighbor search. We also thank the anonymous reviewers for useful comments.

References

- [1] Alekhnovich, M., Braverman, M., Feldman, V., Klivans, A.R., Pitassi, T.: The complexity of properly learning simple concept classes. *J. Comput. Syst. Sci.* 74(1), 16–34 (2008)
- [2] Andoni, A., and Indyk, P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: FOCS, pp. 459–468 (2006)
- [3] Angluin, D., Laird, P.D.: Learning from noisy examples. *Machine Learning* 2(4), 343–370 (1987)
- [4] Blum, A., Frieze, A.M., Kannan, R., Vempala, S.: A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica* 22(1/2), 35–52 (1998)
- [5] Blum, A., Furst, M.L., Jackson, J.C., Kearns, M.J., Mansour, Y., Rudich, S.: Weakly learning dnf and characterizing statistical query learning using fourier analysis. In: STOC, pp. 253–262 (1994)
- [6] Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM* 50(4), 506–519 (2003)
- [7] Buhrman, H., García-Soriano, D., Matsliah, A.: Learning parities in the mistake-bound model. *Inf. Process. Lett.* 111(1), 16–21 (2010)
- [8] Erickson, J.: Lower bounds for linear satisfiability problems. In: SODA, Philadelphia, PA, USA, pp. 388–395 (1995)

- [9] Feldman, V., Gopalan, P., Khot, S., Ponnuswami, A.K.: On agnostic learning of parities, monomials, and halfspaces. *SIAM J. Comput.* 39(2), 606–645 (2009)
- [10] Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: *STOC*, pp. 25–32 (1989)
- [11] Hopper, N.J., Blum, M.: Secure human identification protocols. In: Boyd, C. (ed.) *ASIACRYPT 2001*. LNCS, vol. 2248, pp. 52–66. Springer, Heidelberg (2001)
- [12] Jackson, J.C., Lee, H.K., Servedio, R.A., Wan, A.: Learning random monotone DNF. In: Goel, A., Jansen, K., Rolim, J.D.P., Rubinfeld, R. (eds.) *APPROX* and *RANDOM 2008*. LNCS, vol. 5171, pp. 483–497. Springer, Heidelberg (2008)
- [13] Kalai, A.T., Servedio, R.A.: Boosting in the presence of noise. *J. Comput. Syst. Sci.* 71(3), 266–290 (2005)
- [14] Katz, J.: Efficient cryptographic protocols based on the hardness of learning parity with noise. In: *IMA Int. Conf.*, pp. 1–15 (2007)
- [15] Kearns, M.J.: Efficient noise-tolerant learning from statistical queries. In: *STOC*, pp. 392–401 (1993)
- [16] Klivans, A.R., Servedio, R.A.: Toward attribute efficient learning of decision lists and parities. In: Shawe-Taylor, J., Singer, Y. (eds.) *COLT 2004*. LNCS (LNAI), vol. 3120, pp. 224–238. Springer, Heidelberg (2004)
- [17] Kushilevitz, E., Mansour, Y.: Learning decision trees using the fourier spectrum. *SIAM J. Comput.* 22(6), 1331–1348 (1993)
- [18] Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) *APPROX 2005* and *RANDOM 2005*. LNCS, vol. 3624, pp. 378–389. Springer, Heidelberg (2005)
- [19] Mossel, E., O'Donnell, R., Servedio, R.A.: Learning functions of k relevant variables. *J. Comput. Syst. Sci.* 69(3), 421–434 (2004)
- [20] Panigrahy, R., Talwar, K., Wieder, U.: A geometric approach to lower bounds for approximate near-neighbor search and partial match. In: *FOCS*, pp. 414–423 (2008)
- [21] Panigrahy, R., Talwar, K., Wieder, U.: Lower bounds on near neighbor search via metric expansion. In: *FOCS*, pp. 805–814 (2010)
- [22] Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: *STOC*, pp. 333–342 (2009)
- [23] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* 56(6) (2009)
- [24] Sellie, L.: Learning random monotone dnf under the uniform distribution. In: *COLT*, pp. 181–192 (2008)
- [25] Sellie, L.: Exact learning of random dnf over the uniform distribution. In: *STOC*, pp. 45–54 (2009)
- [26] Valiant, L.G.: A theory of the learnable. *Commun. ACM* 27(11), 1134–1142 (1984)
- [27] Verbeurgt, K.A.: Learning dnf under the uniform distribution in quasi-polynomial time. In: *COLT*, pp. 314–326 (1990)

Supervised Learning and Co-training*

Malte Darnstädt¹, Hans Ulrich Simon¹, and Balázs Szörényi²

¹ Fakultät für Mathematik, Ruhr-Universität Bochum
D-44780 Bochum, Germany

{malte.darnstaedt,hans.simon}@rub.de

² Hungarian Academy of Sciences and University of Szeged
Research Group on Artificial Intelligence
H-6720 Szeged, Hungary
szorenyi@inf.u-szeged.hu

Abstract. Co-training under the Conditional Independence Assumption is among the models which demonstrate how radically the need for labeled data can be reduced if a huge amount of unlabeled data is available. In this paper, we explore how much credit for this saving must be assigned solely to the extra-assumptions underlying the Co-training model. To this end, we compute general (almost tight) upper and lower bounds on the sample size needed to achieve the success criterion of PAC-learning within the model of Co-training under the Conditional Independence Assumption in a purely supervised setting. The upper bounds lie significantly below the lower bounds for PAC-learning without Co-training. Thus, Co-training saves labeled data even when not combined with unlabeled data. On the other hand, the saving is much less radical than the known savings in the semi-supervised setting.

1 Introduction

In the framework of semi-supervised learning, it is usually assumed that there is a kind of compatibility between the target concept and the domain distribution [1]. This intuition is supported by recent results indicating that, without extra-assumptions, there exist purely supervised learning strategies which can compete fairly well against semi-supervised learners (or even against learners with full prior knowledge of the domain distribution) [3, 7].

In this paper, we go one step further and consider the following general question: given a particular extra-assumption which makes semi-supervised learning quite effective, how much credit must be given to the extra-assumption alone? In other words, to which extent can labeled examples be saved by exploiting the extra-assumption in a purely supervised setting? We provide a first answer to this question in a case study which is concerned with the model of Co-training under the Conditional Independence Assumption [4]. In this model (whose formal definition will be recalled in Section 2), *one labeled example* is enough for

* This work was supported by the bilateral Research Support Programme between Germany (DAAD 50751924) and Hungary (MÖB 14440).

¹ See the introduction of [6] for a discussion of the most popular assumptions.

achieving the success criterion of PAC-learning provided that there are sufficiently many unlabeled examples [14]. Recall that PAC-learning without any extra-assumption requires d/ε labeled samples (up to logarithmic factors) where d denotes the VC-dimension of the concept class and ε is the accuracy parameter [5]. The step from d/ε to just a single labeled example is a giant-one. In this paper, we show however that part of the credit must be assigned to just the Co-training itself. More specifically, we show that the number of sample points needed to achieve the success criterion of PAC-learning in the purely supervised model of Co-training under the Conditional Independence Assumption has a linear growth in $\sqrt{d_1 d_2 / \varepsilon}$ (up to some hidden logarithmic factors) as far as the dependence on ε and on the VC-dimensions of the two involved concept classes is concerned. Note that, as ε approaches 0, $\sqrt{d_1 d_2 / \varepsilon}$ becomes much smaller than the well-known lower bound $\Omega(d/\varepsilon)$ on the number of examples needed by a traditional (not co-trained) PAC-learner.

The remainder of the paper is structured as follows. Section 2 clarifies the notations and formal definitions that are used throughout the paper and mentions some elementary facts. Section 3 presents a fundamental inequality that relates a suitably defined variant of Hanneke's disagreement coefficient [9] to a purely combinatorial parameter, $s(\mathcal{C})$, which is closely related to the “unique negative dimension” from [8]. This will later lead to the insight that the product of the VC-dimension of a (suitably chosen) hypothesis class and a (suitably defined) disagreement coefficient has the same order of magnitude as $s(\mathcal{C})$. Section 3 furthermore investigates how a concept class can be padded so as to increase the VC-dimension while keeping the disagreement coefficient invariant. The padding can be used to lift lower bounds that hold for classes of low VC-dimension to increased lower bounds that hold for some classes of arbitrarily large VC-dimension. The results of Section 3 seem to have implications for active learning and might be of independent interest. Section 4.1 presents some general upper bounds in terms of the relevant learning parameters (including ε , the VC-dimension, and the disagreement coefficient, where the product of the latter two can be replaced by the combinatorial parameters from Section 3). Section 4.2 shows that all general upper bounds from Section 4.1 are (nearly) tight. Interestingly, the learning strategy that is best from the perspective of a worstcase analysis has one-sided error. Section 4.3 presents improved bounds for classes with special properties. Section 5 contains some final remarks.

Due to space constraints, not all proofs are given in full detail. We plan to give the missing parts in an upcoming journal version of this paper.

2 Definitions, Notations, and Facts

We assume the reader is familiar with Valiant's model of Probably Approximately Correct Learning (PAC-learning) [11]. In Co-training [4], it is assumed

² This is one of the results which impressively demonstrate the striking potential of properly designed semi-supervised learning strategies although the underlying compatibility assumptions are somewhat idealized and therefore not likely to be strictly satisfied in practice. See [2, 12] for suggestions of relaxed assumptions.

that there is a pair of concept classes, \mathcal{C}_1 and \mathcal{C}_2 , and that random examples come in pairs $(x_1, x_2) \in X_1 \times X_2$. Moreover, the domain distribution D , according to which the random examples are generated, is perfectly compatible with the target concepts, say $h_1^* \in \mathcal{C}_1$ and $h_2^* \in \mathcal{C}_2$, in the sense that $h_1^*(x_1) = h_2^*(x_2)$ with probability 1. (For this reason, we sometimes denote the target label as $h^*(x_1, x_2)$.) As in [4, 1], our analysis builds on the Conditional Independence Assumption: x_1, x_2 , considered as random variables that take “values” in X_1 and X_2 , respectively, are conditionally independent given the label. As in [1], we perform a PAC-style analysis of Co-training under the Conditional Independence Assumption. But unlike [1], we assume that there is no access to unlabeled examples. The resulting model is henceforth referred to as the “PAC Co-training Model under the Conditional Independence Assumption”.

Let \mathcal{C} be a concept class over domain X and $\mathcal{H} \supseteq \mathcal{C}$ a hypothesis class over the same domain. For every $h^* \in \mathcal{C}$ and every $X' \subseteq X$, the corresponding *version space in \mathcal{H}* is given by $V_{\mathcal{H}}(X', h^*) := \{h \in \mathcal{H} \mid \forall x \in X' : h(x) = h^*(x)\}$. Let $V \subseteq \mathcal{C}$. The *disagreement region of V* is given by $\text{DIS}(V) := \{x \in X \mid \exists h, h' \in V : h(x) \neq h'(x)\}$. Let \mathbb{P} denote a probability measure on X . We define the following variants of disagreement coefficients:

$$\begin{aligned}\theta(\mathcal{C}, \mathcal{H} | \mathbb{P}, X', h^*) &:= \frac{\mathbb{P}(\text{DIS}(V_{\mathcal{C}}(X', h^*)))}{\sup_{h \in V_{\mathcal{H}}(X', h^*)} \mathbb{P}(h \neq h^*)} \\ \theta(\mathcal{C}, \mathcal{H}) &:= \sup_{\mathbb{P}, X', h^*} \theta(\mathcal{C}, \mathcal{H} | \mathbb{P}, X', h^*)\end{aligned}$$

For sake of brevity, let $\theta(\mathcal{C}) := \theta(\mathcal{C}, \mathcal{C})$. Note that

$$\theta(\mathcal{C}, \mathcal{H}) \leq \theta(\mathcal{C}) \leq |\mathcal{C}| - 1 . \quad (1)$$

The first inequality is obvious from $\mathcal{C} \subseteq \mathcal{H}$ and $h^* \in \mathcal{C}$, the second follows from

$$\text{DIS}(V_{\mathcal{C}}(X', h^*)) = \bigcup_{h \in V_{\mathcal{C}}(X', h^*) \setminus \{h^*\}} \{x \mid h(x) \neq h^*(x)\}$$

and an application of the union bound.

As an example we will calculate θ for the following class, which will also be useful for proving lower bounds in section 4.2:

$$\text{SF}_n = \{\{0\}, \{0, 1\}, \{0, 2\}, \dots, \{0, n\}\}$$

Lemma 1. $\theta(\text{SF}_n) = n$.

Proof. Let \mathbb{P} be uniform on $\{1, \dots, n\}$, let $X' = h^* = \{0\}$. Then $V := V_{\text{SF}_n}(X', h^*) = \text{SF}_n$ and $\text{DIS}(V) := \{1, \dots, n\}$ has probability mass 1. Thus,

$$\theta(\text{SF}_n) \geq \theta(\text{SF}_n, \text{SF}_n | \mathbb{P}, X', h^*) = \frac{\mathbb{P}(\text{DIS}(V))}{\sup_{h \in V} \mathbb{P}(h \neq h^*)} = \frac{1}{1/n} = n .$$

Conversely, $\theta(\text{SF}_n) \leq |\text{SF}_n| - 1 = n$ (according to (1)). \square

The main usage this disagreement coefficient is as follows. First note that we have $\mathbb{P}(DIS(V_{\mathcal{C}}(X', h^*))) \leq \theta(\mathcal{C}, \mathcal{H}) \cdot \sup_{h \in V_{\mathcal{H}}(X', h^*)} \mathbb{P}(h \neq h^*)$ for every choice of \mathbb{P}, X', h^* . This inequality holds in particular when X' consists of m points in X chosen independently at random according to \mathbb{P} . According to classical sample size bounds in PAC-learning, there exists a sample size $m = \tilde{O}(\text{VCdim}(\mathcal{H})/\varepsilon)$ such that, with probability at least $1 - \delta$, $\sup_{h \in V_{\mathcal{H}}(X', h^*)} \mathbb{P}(h \neq h^*) \leq \varepsilon$. Thus, with probability at least $1 - \delta$ (taken over the random sample X'), $\mathbb{P}(DIS(V_{\mathcal{C}}(X', h^*))) \leq \theta(\mathcal{C}, \mathcal{H}) \cdot \varepsilon$. This discussion (with $\varepsilon/\theta(\mathcal{C}, \mathcal{H})$ substituted for ε) is summarized in the following

Lemma 2. *There exists a sample size $m = \tilde{O}(\theta(\mathcal{C}, \mathcal{H}) \cdot \text{VCdim}(\mathcal{H})/\varepsilon)$ such that the following holds for every probability measure \mathbb{P} on domain X and for every target concept $h^* \in \mathcal{C}$. With probability $1 - \delta$, taken over a random sample X' of size m , $\mathbb{P}((DIS(V_{\mathcal{C}}(X', h^*))) \leq \varepsilon$.*

This lemma indicates that one should choose \mathcal{H} so as to minimize $\theta(\mathcal{C}, \mathcal{H}) \cdot \text{VCdim}(\mathcal{H})$. Note that making \mathcal{H} more powerful leads to smaller values of $\theta(\mathcal{C}, \mathcal{H})$ but comes at the prize of an increased VC-dimension.

We say that \mathcal{H} contains hypotheses with plus-sided errors (or minus-sided errors, resp.) w.r.t. concept class \mathcal{C} if, for every $X' \subseteq X$ and every $h^* \in \mathcal{C}$, there exists $h \in V_{\mathcal{H}}(X', h^*)$ such that $h(x) = 0$ ($h(x) = 1$, resp.) for every $x \in \text{DIS}(V_{\mathcal{C}}(X', h^*))$. A sufficient (but, in general, not necessary) condition for a class \mathcal{H} making plus-sided errors only (or minus-sided errors only, resp.) is being closed under intersection (or closed under union, resp.).

Lemma 3. *Let $\mathcal{C} \subseteq \mathcal{H}$. If \mathcal{H} contains hypotheses with plus-sided errors and hypotheses with minus-sided errors w.r.t. \mathcal{C} , then $\theta(\mathcal{C}, \mathcal{H}) \leq 2$.*

Proof. Consider a fixed but arbitrary choice of \mathbb{P}, X', h^* . Let h_{min} be the hypothesis in $V_{\mathcal{H}}(X', h^*)$ that errs on positive examples of h^* only, and let h_{max} be the hypothesis in $V_{\mathcal{H}}(X', h^*)$ that errs on negative examples of h^* only. We conclude that $\text{DIS}(V_{\mathcal{C}}(X', h^*)) \subseteq \{x \mid h_{min}(x) \neq h_{max}(x)\}$. From this and the triangle inequality, it follows that

$$\mathbb{P}(\text{DIS}(V_{\mathcal{C}}(X', h^*))) \leq \mathbb{P}(h_{min} \neq h_{max}) \leq \mathbb{P}(h_{min} \neq h^*) + \mathbb{P}(h_{max} \neq h^*) .$$

The claim made by the lemma is now obvious from the definition of $\theta(\mathcal{C}, \mathcal{H})$. \square

Example 1. Since POWERSET and HALFINTERVALS are closed under intersection and union, we obtain $\theta(\text{POWERSET}) \leq 2$ and $\theta(\text{HALFINTERVALS}) \leq 2$. Let the class \mathcal{C} consist of both the open and the closed homogeneous halfplanes and let \mathcal{H} be the class of unions and intersections of two halfplanes from \mathcal{C} . It is easy to see that \mathcal{H} contains hypotheses with plus-sided errors (the smallest pie slice with apex at $\mathbf{0}$ that includes all positive examples in a sample) and hypotheses with minus-sided errors (the complement of the smallest pie slice with apex at $\mathbf{0}$ that includes all negative examples in a sample) w.r.t. \mathcal{C} . Thus, $\theta(\mathcal{C}, \mathcal{H}) \leq 2$. Note that \mathcal{H} is neither closed under intersection nor closed under union.

3 A Closer Look to the Disagreement Coefficient

In Section 3.1 we investigate the question how small the product $\text{VCdim}(\mathcal{C}) \cdot \theta(\mathcal{C}, \mathcal{H})$ can become if $\mathcal{H} \supseteq \mathcal{C}$ is cleverly chosen. The significance of this question should be clear from Lemma 2. In Section 3.2 we introduce a padding technique which leaves the disagreement coefficient invariant but increases the VC-dimension (and, as we will see later, also increases the error rates in the PAC Co-training Model).

3.1 A Combinatorial Upper Bound

Let $s^+(\mathcal{C})$ denote the largest number of instances in X such that every binary pattern on these instances with exactly one “+”-label can be realized by a concept from \mathcal{C} . In other words: $s^+(\mathcal{C})$ denotes the cardinality of the largest singleton-subclass of \mathcal{C} . Let \mathcal{C}^+ denote the class of all unions of concepts from \mathcal{C} . As usual, the empty union is defined to be the empty set.

Lemma 4. $\mathcal{C} \subseteq \mathcal{C}^+$, \mathcal{C}^+ is closed under union, and $\text{VCdim}(\mathcal{C}^+) = s^+(\mathcal{C})$. Moreover, if \mathcal{C} is closed under intersection, then \mathcal{C}^+ is closed under intersection too, and $\theta(\mathcal{C}, \mathcal{C}^+) \leq 2$ so that $\text{VCdim}(\mathcal{C}^+) \cdot \theta(\mathcal{C}, \mathcal{C}^+) \leq 2s^+(\mathcal{C})$.

Proof. By construction, $\mathcal{C} \subseteq \mathcal{C}^+$ and \mathcal{C}^+ is closed under union. From this it follows that $s^+(\mathcal{C}) \leq \text{VCdim}(\mathcal{C}^+)$. Consider now instances x_1, \dots, x_d that are shattered by \mathcal{C}^+ . Thus, for every $i = 1, \dots, d$, there exists a concept h_i in \mathcal{C}^+ that contains x_i but none of the other $d - 1$ instances. Therefore, by the construction of \mathcal{C}^+ , \mathcal{C} must contain some hypothesis h'_i smaller than h_i satisfying $h'_i(x_i) = 1$. We conclude that $\text{VCdim}(\mathcal{C}^+) \leq s^+(\mathcal{C})$. For the remainder of the proof, assume that \mathcal{C} is closed under intersection. Consider two sets A, B of the form $A = \cup_i A_i$ and $B = \cup_j B_j$ where all A_i and B_j are concepts in \mathcal{C} . Then, according to the distributive law, $A \cap B = \cup_{i,j} A_i \cap B_j$. Since \mathcal{C} is closed under intersection, $A_i \cap B_j \in \mathcal{C} \subseteq \mathcal{C}^+$. We conclude that \mathcal{C}^+ is closed under intersection. Closure under intersection and union implies that \mathcal{C}^+ contains hypotheses with plus-sided errors and hypotheses with minus-sided errors w.r.t. \mathcal{C} . According to Lemma 3, $\theta(\mathcal{C}, \mathcal{C}^+) \leq 2$. \square

We aim at a similar result that holds for arbitrary (not necessarily intersection-closed) concept classes. To this end, we proceed as follows. Let $s^-(\mathcal{C})$ denote the largest number of instances in X such that every binary pattern on these instances with exactly one “−”-label can be realized by a concept from \mathcal{C} . In other words: $s^-(\mathcal{C})$ denotes the cardinality of the largest co-singleton subclass of \mathcal{C} . Let \mathcal{C}^- denote the class of all intersections of concepts from \mathcal{C} . As usual, the empty intersection is defined to be the full set X . By duality, Lemma 4 translates into the following

Corollary 1. $\mathcal{C} \subseteq \mathcal{C}^-$, \mathcal{C}^- is closed under intersection, and $\text{VCdim}(\mathcal{C}^-) = s^-(\mathcal{C})$. Moreover, if \mathcal{C} is closed under union, then \mathcal{C}^- is closed under union too, and $\theta(\mathcal{C}, \mathcal{C}^-) \leq 2$ so that $\text{VCdim}(\mathcal{C}^-) \cdot \theta(\mathcal{C}, \mathcal{C}^-) \leq 2s^-(\mathcal{C})$.

We now arrive at the following general bound:

Theorem 1. Let $\mathcal{H} := \mathcal{C}^+ \cup \mathcal{C}^-$. Then, $\mathcal{C} \subseteq \mathcal{H}$, $\text{VCdim}(\mathcal{H}) \leq 2 \max\{s^+(\mathcal{C}), s^-(\mathcal{C})\}$, and $\theta(\mathcal{C}, \mathcal{H}) \leq 2$ so that $\text{VCdim}(\mathcal{H}) \cdot \theta(\mathcal{C}, \mathcal{H}) \leq 4 \max\{s^+(\mathcal{C}), s^-(\mathcal{C})\} := s(\mathcal{C})$.

Proof. $\mathcal{C} \subseteq \mathcal{H}$ is obvious. The bound on the VC-dimension is obtained as follows. If m instances are given, then, by Lemma 4 and Corollary 1, the number of binary patterns imposed on them by concepts from $\mathcal{H} = \mathcal{C}^+ \cup \mathcal{C}^-$ is bounded by $\Phi_{s^+(\mathcal{C})}(m) + \Phi_{s^-(\mathcal{C})}(m)$ where

$$\Phi_d(m) = \begin{cases} 2^m & \text{if } m \leq d \\ \sum_{i=0}^d \binom{m}{i} & \text{otherwise} \end{cases}$$

is the upper bound from Sauer's Lemma [10]. Note that $\Phi_d(m) < 2^{m-1}$ for $m > 2d$. Thus, for $m > 2 \max\{s^+(\mathcal{C}), s^-(\mathcal{C})\}$, $\Phi_{s^+(\mathcal{C})}(m) + \Phi_{s^-(\mathcal{C})}(m) < 2^{m-1} + 2^{m-1} = 2^m$. We can conclude that $\text{VCdim}(\mathcal{H}) \leq 2 \max\{s^+(\mathcal{C}), s^-(\mathcal{C})\}$. Finally note that $\theta(\mathcal{C}, \mathcal{H}) \leq 2$ follows from Lemma 3 and the fact that, because of Lemma 4 and Corollary 1, $\mathcal{H} = \mathcal{C}^+ \cup \mathcal{C}^-$ contains hypotheses with plus-sided errors and hypotheses with minus-sided errors. \square

Please note that the parameter $s^-(\mathcal{C})$ was originally introduced by Mihály Geréb-Graus in [8] as the “unique negative dimension” of \mathcal{C} . He showed that it characterizes PAC-learnability from positive examples alone.

3.2 Invariance of the Disagreement Coefficient under Padding

For every domain X , let $X^{(i)}$ and $X^{[k]}$ be given by

$$X^{(i)} = \{(x, i) \mid x \in X\} \text{ and } X^{[k]} = X^{(1)} \cup \dots \cup X^{(k)} .$$

For every concept $h \subseteq X$, let $h^{(i)} = \{(x, i) \mid x \in h\}$. For every concept class \mathcal{C} over domain X , let

$$\mathcal{C}^{[k]} := \{h_1^{(1)} \cup \dots \cup h_k^{(k)} \mid h_1, \dots, h_k \in \mathcal{C}\} .$$

Loosely speaking, $\mathcal{C}^{[k]}$ contains k -fold “disjoint unions” of concepts from \mathcal{C} . It is obvious that $\text{VCdim}(\mathcal{C}^{[k]}) = k \cdot \text{VCdim}(\mathcal{C})$. The following result shows that the disagreement-coefficient is invariant under k -fold disjoint union:

Lemma 5. For all $k \geq 1$: $\theta(\mathcal{C}^{[k]}, \mathcal{H}^{[k]}) = \theta(\mathcal{C}, \mathcal{H})$.

Proof. The probability measures \mathbb{P} on $X^{[k]}$ can be written as convex combinations of probability measures on the $X^{(i)}$, i.e., $\mathbb{P} = \lambda_1 \mathbb{P}_1 + \dots + \lambda_k \mathbb{P}_k$ where \mathbb{P}_i is a probability measure on $X^{(i)}$, and the λ_i are non-negative numbers that sum-up to 1. A sample $S \subseteq X^{[k]}$ decomposes into $S = S^{(1)} \cup \dots \cup S^{(k)}$ with $S^{(i)} \subseteq X^{(i)}$. An analogous remark applies to concepts $c \in \mathcal{C}^{[k]}$ and hypotheses $h \in \mathcal{H}^{[k]}$. Thus,

$$\begin{aligned}
\theta(\mathcal{C}^{[k]}, \mathcal{H}^{[k]} | \mathbb{P}, S, c) &= \frac{\mathbb{P}(\text{DIS}(V_{\mathcal{C}^{[k]}}(S, c)))}{\sup_{h \in V_{\mathcal{H}^{[k]}}(S, c)} \mathbb{P}(h \neq c)} \\
&= \frac{\sum_{i=1}^k \lambda_i \overbrace{\mathbb{P}_i(\text{DIS}(V_{\mathcal{C}^{(i)}}(S^{(i)}, c^{(i)})))}^{=:a_i}}{\sum_{i=1}^k \lambda_i \underbrace{\sup_{h_i^{(i)} \in V_{\mathcal{H}^{(i)}}(S^{(i)}, c^{(i)})} \mathbb{P}_i(h^{(i)} \neq c^{(i)})}_{=:b_i}} \leq \theta(\mathcal{C}, \mathcal{H}) .
\end{aligned}$$

The last inequality holds because, obviously, $a_i/b_i \leq \theta(\mathcal{C}^{(i)}, \mathcal{H}^{(i)}) = \theta(\mathcal{C}, \mathcal{H})$. On the other hand, a_i/b_i can be made equal (or arbitrarily close) to $\theta(\mathcal{C}, \mathcal{H})$ by choosing $\mathbb{P}_i, S^{(i)}, c^{(i)}$ properly. \square

4 Supervised Learning and Co-training

Let $p_+ = \mathbb{P}(h^* = 1)$ denote the probability for seeing a positive example of h^* . Similarly, $p_- = \mathbb{P}(h^* = 0)$ denotes the probability for seeing a negative example of h^* . Let $\mathbb{P}(\cdot|+), \mathbb{P}(\cdot|-)$ denote probabilities conditioned to positive or to negative examples, respectively. The error probability of a hypothesis h decomposes into conditional error probabilities according to

$$\mathbb{P}(h \neq h^*) = p_+ \cdot \mathbb{P}(h \neq h^*|+) + p_- \cdot \mathbb{P}(h \neq h^*|-) . \quad (2)$$

In the PAC-learning framework, a sample size that, with high probability, bounds the error by ε typically bounds the plus-conditional error by ε/p_+ and the minus-conditional error by ε/p_- . According to (2), these conditional error terms lead to an overall error that is bounded by ε , indeed. For this reason, the hardness of a problem in the PAC-learning framework does not significantly depend on the values of p_+, p_- . As we will see shortly, the situation is much different in the PAC Co-training Model under the Conditional Independence Assumption where small values of $p_{min} := \min\{p_+, p_-\}$ (though not smaller than ε) make the learning problem harder. Therefore, we refine the analysis and present our bounds on the sample size not only in terms of distribution-independent quantities like θ, ε and the VC-dimension but also in terms of p_{min} . This will lead to “smart” learning policies that take advantage of “benign values” of p_{min} . In the following subsections, we present (almost tight) upper and lower bounds on the sample size in the PAC Co-training Model under the Conditional Independence Assumption.

4.1 General Upper Bounds on the Sample Size

Let us first fix some more notation that is also used in subsequent sections. $V_1 \subseteq \mathcal{C}_1$ and $V_2 \subseteq \mathcal{C}_2$ denote the version spaces induced by the labeled sample within the concept classes, respectively, and $\text{DIS}_1 = \text{DIS}(V_1)$, $\text{DIS}_2 = \text{DIS}(V_2)$ are the corresponding disagreement regions. The VC-dimension of \mathcal{H}_1 is denoted d_1 ; the VC-dimension of \mathcal{H}_2 is denoted d_2 . $\theta_1 = \theta(\mathcal{C}_1, \mathcal{H}_1)$ and $\theta_2 = \theta(\mathcal{C}_2, \mathcal{H}_2)$.

$\theta_{min} = \min\{\theta_1, \theta_2\}$ and $\theta_{max} = \max\{\theta_1, \theta_2\}$. $s_1^+ = s^+(\mathcal{C}_1)$, $s_2^+ = s^+(\mathcal{C}_2)$, $s_1^- = s^-(\mathcal{C}_1)$, and $s_2^- = s^-(\mathcal{C}_2)$. The learner's empirical estimates for p_+, p_-, p_{min} (inferred from the labeled random sample) are denoted $\hat{p}_+, \hat{p}_-, \hat{p}_{min}$, respectively. Let $h_1 \in V_{\mathcal{H}_1}$ and $h_2 \in V_{\mathcal{H}_2}$ denote two hypotheses chosen according to some arbitrary but fixed learning rules.

The error probability of the learner is the probability for erring on an unlabeled “test-instance” (x_1, x_2) . Note that the learner has a safe decision if $x_1 \notin \text{DIS}_1$ or $x_2 \notin \text{DIS}_2$. As for the case $x_1 \in \text{DIS}_1$ and $x_2 \in \text{DIS}_2$, the situation for the learner is ambiguous, and we consider the following resolution-rules, the first two of which depend on the hypotheses h_1 and h_2 ³

- R1: If $h_1(x_1) = h_2(x_2)$, then vote for the same label. If $h_1(x_1) \neq h_2(x_2)$, then go with the hypothesis that belongs to the class with the disagreement coefficient θ_{max} .
- R2: If $h_1(x_1) = h_2(x_2)$, then vote for the same label. If $h_1(x_1) \neq h_2(x_2)$, then vote for the label that occurred less often in the sample (i.e., vote for “+” if $\hat{p}_- \geq 1/2$, and for “-” otherwise).
- R3: If $\hat{p}_- \geq 1/2$, then vote for label “+”. Otherwise, vote for label “-”. (These votes are regardless of the hypotheses h_1, h_2 .)

Theorem 2. *The number of labeled examples sufficient for learning $(\mathcal{C}_1, \mathcal{C}_2)$ in the PAC Co-training Model under the Conditional Independence Assumption by learners applying one of the rules R1, R2, R3 is given asymptotically as follows:*

$$\begin{cases} \tilde{O}\left(\sqrt{\frac{d_1 d_2}{\varepsilon} \cdot \frac{\theta_{min}}{p_{min}}}\right) & \text{if rule R1 is applied} \\ \tilde{O}\left(\sqrt{\frac{d_1 d_2}{\varepsilon} \cdot \max\left\{\frac{1}{p_{min}}, \theta_{max}\right\}}\right) & \text{if rule R2 is applied} \\ \tilde{O}\left(\sqrt{\frac{d_1 d_2}{\varepsilon} \cdot \theta_1 \theta_2}\right) & \text{if rule R3 is applied} \end{cases} \quad (3)$$

Proof. $\tilde{O}(1)$ examples are sufficient to achieve that (with high probability) the following holds: if $p_{min} < 1/4$, then $\hat{p}_{min} < 1/2$. Assume that this is the case. For reasons of symmetry, we may assume furthermore that $\theta_1 = \theta_{max}$ and $\hat{p}_- \geq 1/2$ so that $p_- \geq 1/4$. Please recall that the rules R1 to R3 are only applied if $x_1 \in \text{DIS}_1$ and $x_2 \in \text{DIS}_2$. Assume first that ambiguities are resolved according to rule R1. Note that the sample size specified in (3) is sufficient to bound (with high probability) the error rate of hypotheses h_1, h_2 , respectively, as follows [5]:

$$\varepsilon_1 = \sqrt{\frac{d_1}{d_2} \cdot \frac{p_{min}}{\theta_{min}} \cdot \varepsilon} \text{ and } \varepsilon_2 = \sqrt{\frac{d_2}{d_1} \cdot \frac{p_{min}}{\theta_{min}} \cdot \varepsilon}$$

³ The choice applied in rules R2 and R3 could seem counterintuitive at first. However, $\hat{p}_+ > \hat{p}_-$ means that the learner has more information about the behaviour of the target concept on the positive instances than on the negative ones, indicating that the positive instances in the disagreement regions might have smaller probability than the negative ones. This choice is also in accordance with the common strategy applied in the “learning from positive examples only” model, which outputs a negative label if in doubt, although the learner has never seen any negative examples.

If R1 assigns a wrong label to (x_1, x_2) , then, necessarily, h_1 errs on x_1 and $x_2 \in \text{DIS}_2$. Thus the error rate induced by R1 is bounded (with high probability) as follows:

$$\begin{aligned} & \mathbb{P}(h_1(x_1) = 0 \wedge x_2 \in \text{DIS}_2|+)p_+ + \mathbb{P}(h_1(x_1) = 1 \wedge x_2 \in \text{DIS}_2|-)p_- \\ & \leq \frac{1}{p_{\min}} \cdot \left(\mathbb{P}(h_1(x_1) = 0|+)p_+ \cdot \mathbb{P}(x_2 \in \text{DIS}_2|+)p_+ \right. \\ & \quad \left. + \mathbb{P}(h_1(x_1) = 1|-)p_- \cdot \mathbb{P}(x_2 \in \text{DIS}_2|-)p_- \right) \\ & \leq \frac{1}{p_{\min}} \cdot \underbrace{\left(\mathbb{P}(h_1(x_1) = 0|+)p_+ + \mathbb{P}(h_1(x_1) = 1|-)p_- \right)}_{\leq \varepsilon_1} \\ & \quad \cdot \underbrace{\left(\mathbb{P}(x_2 \in \text{DIS}_2|+)p_+ + \mathbb{P}(x_2 \in \text{DIS}_2|-)p_- \right)}_{\leq \theta_2 \varepsilon_2 = \theta_{\min} \varepsilon_2} \leq \frac{\theta_{\min}}{p_{\min}} \cdot \varepsilon_1 \varepsilon_2 = \varepsilon \end{aligned}$$

The first inequality in this calculation makes use of Conditional Independence and the third applies Lemma 2.

The proofs for rule R2 and R3 proceed analogously. We omit the details because of space constraints. \square

We now describe a strategy named ‘‘Combined Rule’’ that uses rules R1, R2, R3 as sub-routines. Given $(x_1, x_2) \in \text{DIS}_1 \times \text{DIS}_2$, it proceeds as follows. If $\varepsilon > 4/(\theta_1 \theta_2)$ and $\hat{p}_+ \leq \varepsilon/2$ (or $\hat{p}_- \leq \varepsilon/2$, resp.), it votes for label ‘‘−’’ (or for label ‘‘+’’, resp.). If $\varepsilon \leq 4/(\theta_1 \theta_2)$ or $\hat{p}_{\min} := \min\{\hat{p}_+, \hat{p}_-\} > \varepsilon/2$, then it applies the rule

$$\begin{cases} \text{R1 if } \frac{\theta_{\min}}{\theta_{\max}} \leq \hat{p}_{\min} \\ \text{R2 if } \frac{1}{\theta_1 \theta_2} \leq \hat{p}_{\min} < \frac{\theta_{\min}}{\theta_{\max}} \\ \text{R3 if } \hat{p}_{\min} < \frac{1}{\theta_1 \theta_2} \end{cases}. \quad (4)$$

Corollary 2. *If the learner applies the Combined Rule, then*

$$\begin{cases} \tilde{O}\left(\sqrt{\frac{d_1 d_2}{\varepsilon} \cdot \frac{\theta_{\min}}{p_{\min}}}\right) & \text{if } \frac{\theta_{\min}}{\theta_{\max}} \leq p_{\min} \\ \tilde{O}\left(\sqrt{\frac{d_1 d_2}{\varepsilon} \cdot \theta_{\max}}\right) & \text{if } \frac{1}{\theta_{\max}} \leq p_{\min} < \frac{\theta_{\min}}{\theta_{\max}} \\ \tilde{O}\left(\sqrt{\frac{d_1 d_2}{\varepsilon} \cdot \frac{1}{p_{\min}}}\right) & \text{if } \frac{1}{\theta_1 \theta_2} \leq p_{\min} < \frac{1}{\theta_{\max}} \\ \tilde{O}\left(\sqrt{\frac{d_1 d_2}{\varepsilon} \cdot \theta_1 \theta_2}\right) & \text{if } p_{\min} < \frac{1}{\theta_1 \theta_2} \end{cases} \quad (5)$$

labeled examples are sufficient for learning $\mathcal{C}_1, \mathcal{C}_2$ in the PAC Co-training Model under the Conditional Independence Assumption.

Proof. It is an easy application of multiplicative Chernov-bounds to show that (with high probability) \hat{p}_{\min} equals p_{\min} up to a factor of 2 unless one of the following special cases occurs:

$$\varepsilon > \frac{4}{\theta_1 \theta_2}, \hat{p}_{min} < \frac{\varepsilon}{2} \text{ and } p_{min} < \frac{\varepsilon}{2} \quad (6)$$

$$\varepsilon \leq \frac{4}{\theta_1 \theta_2}, \hat{p}_{min} < \frac{1}{\theta_1 \theta_2} \text{ and } p_{min} < \frac{1}{\theta_1 \theta_2} \quad (7)$$

In case of (6), the Combined Rule outputs the empirically more likely label, and the error rate is bounded by $p_{min} < \varepsilon/2$. In case of (7), rule R3 is applied which, according to Theorem 2, leads to the desired upper bound on the sample size. Thus, we may now safely assume that \hat{p}_{min} equals p_{min} up to factor 2. If none of the rules R1, R2, R3 is applied, then $\hat{p}_{min} \leq \varepsilon/2$ and the error rate will be $p_{min} < \varepsilon$. Thus, we may assume that Combined Rule proceeds according to (4). If the learner could substitute the (unknown) p_{min} for \hat{p}_{min} within (4), the corollary would follow immediately from Theorem 2. But it is easy to see that even the knowledge of the empirical estimate \hat{p}_{min} is sufficient for this purpose. \square

We can also give a completely combinatorial upper bound without referring to θ . As we will see later this bound is tight up to logarithmic factors in the worst case (i.e. small p_{min}):

Theorem 3. *If the learner applies rule R3 and uses hypothesis classes $\mathcal{H}_b = \mathcal{C}_b^+ \cup \mathcal{C}_b^-$ for $b = 1, 2$, then $\tilde{O}(\sqrt{\max\{s_1^+ s_2^+, s_1^- s_2^-\}}/\varepsilon)$ labeled examples are sufficient.*

Proof. (Sketch) R3 always chooses one of the two hypotheses with one sided-error: h^- which always outputs “–” for $(x_1, x_2) \in DIS_1 \times DIS_2$, and h^+ which always outputs “+” on these instances. With an analysis similar as before one can relate the error of h^- to the errors of the smallest consistent hypotheses in \mathcal{C}_1^- and \mathcal{C}_2^- , and then conclude using standard PAC-bounds and results from Section 3.1 that with high probability $\tilde{O}(\sqrt{s_1^- s_2^-}/(\varepsilon p_+))$ many examples suffice for h^- to have an error of at most ε . For h^+ the analogous bound on the number of examples is $\tilde{O}(\sqrt{s_1^+ s_2^+}/(\varepsilon p_-))$. Because $\tilde{O}(1)$ many examples are enough to distinguish with high probability among the cases $p_- \geq 1/2$ and $p_- < 1/2$, choosing between the two learning strategies according to rule R3 yields the desired bound. \square

Please note that finding the smallest consistent hypotheses in \mathcal{C}_b^- is possible using positive examples only. This shows a strong connection to the results by Geréb-Graus in [8], where it was shown that $\tilde{O}(s^-(\mathcal{C})/\varepsilon)$ many positive examples are sufficient (and necessary) to PAC-learn a class \mathcal{C} from positive examples alone. A dual also holds for the largest consistent hypotheses in \mathcal{C}_b^+ .

4.2 Lower Bounds on the Sample Size

In this section, we first derive a general lower bound which matches the upper bound from Theorem 3. Both bounds are part of a worstcase analysis and the

lower bound makes use of rather small values of p_{min} . Afterwards, we show that, for more “benign” choices of p_{min} , the upper bound from Corollary 2 is tight by exhibiting concrete concept classes that lead to a matching lower bound.

A useful concept class for the purposes of this section is SF_n from Lemma 11. Note that all lower bounds obtained for SF_n immediately generalize to concept classes containing SF_n as subclass.

Lemma 6. *Let $n_1, n_2 \geq 1$, and let $\mathcal{C}_b = SF_{n_b+2}$ so that $\theta_b = n_b + 2$ for $b = 1, 2$. Then, for every $p_{min} \leq 1/(\theta_1\theta_2)$ and every sufficiently small $\varepsilon > 0$, the number of examples needed to learn $\mathcal{C}_1, \mathcal{C}_2$ in the PAC Co-training Model under the Conditional Independence Assumption is at least $\Omega(\sqrt{n_1 n_2}/\varepsilon)$.*

Proof. Let $p_+ = p_{min}$, let \mathcal{C}_1 be a concept class over domain $\{a_0, a_1, \dots, a_{n_1+2}\}$, and let \mathcal{C}_2 be a concept class over domain $\{b_0, b_1, \dots, b_{n_2+2}\}$, respectively. Obviously,

$$p_+ = p_{min} \leq \frac{1}{(n_1 + 2)(n_2 + 2)} = \frac{1}{\theta_1\theta_2} .$$

Note that $n_1 n_2 p_+ \leq 1$. Consider the following malign scenario:

- $\mathbb{P}(a_0|+) = \mathbb{P}(b_0|+) = 1 - \sqrt{\varepsilon/p_+}$.
- Index s is uniformly chosen at random from $\{2, \dots, n_1 + 2\}$. Index t is uniformly chosen at random from $\{2, \dots, n_2 + 2\}$. $\mathbb{P}(a_s|+) = \mathbb{P}(b_t|+) = \sqrt{\varepsilon/p_+}$.
- $\mathbb{P}(a_1|-) = 1 - 4 \cdot \sqrt{n_1\varepsilon/n_2}$. $\mathbb{P}(b_1|-) = 1 - 4 \cdot \sqrt{n_2\varepsilon/n_1}$.
- The instances from $X_1 \setminus \{a_0, a_1, a_s\}$ evenly share a minus-conditional probability mass of $4 \cdot \sqrt{n_1\varepsilon/n_2}$. The instances from $X_2 \setminus \{b_0, b_1, b_t\}$ evenly share a minus-conditional probability mass of $4 \cdot \sqrt{n_2\varepsilon/n_1}$.

Let us assume that the sample size satisfies $m \leq \frac{\sqrt{n_1 n_2}}{40} \cdot \sqrt{1/\varepsilon}$. Let Z_1 count the number of sample points that hit $X_1 \setminus \{a_0, a_1, a_s\}$ (the “interesting negative examples” in X_1). Let Z_2 be defined analogously. Then the following holds:

- The expectation of Z_b is bounded by $n_b/10$ for $b = 1, 2$. Thus, with probability at least $1 - 2/5$, $Z_1 \leq n_1/2$ and $Z_2 \leq n_2/2$, which is assumed in the sequel. Thus at least half of the interesting negative examples in X_1 and at least half of the interesting negative examples in X_2 remain “hidden” from the learner (i.e. do not occur in the sample), respectively.
- The expected number of occurrences of a_s (or b_t , resp.) in the sample is bounded by

$$p_+ \cdot \sqrt{\varepsilon/p_+} \cdot \frac{\sqrt{n_1 n_2}}{40} \cdot \sqrt{1/\varepsilon} = \sqrt{n_1 n_2 p_+}/40 \leq 1/40 .$$

Thus, with probability at least $1 - 1/15$, neither a_s nor b_t occurs in the sample, which is assumed in the sequel.

Note that the assumptions that we made on the way are satisfied with a probability of at least $1 - 2/5 - 1/15 > 1/2$. Given these assumptions, we now

bound the smallest possible error rate from below. Let E_+ (or E_- , resp.) denote the set of instance-pairs which are labeled “+” (or labeled “−”, resp.). For $b = 1, 2$, let $U_b \subseteq X_b$ be the set of points in X_b that did not occur in the sample, and let $U = U_1 \times U_2$. For test-instances $(x_1, x_2) \notin U$, the learner can infer the label from the information provided by the sample. It can be shown by a rigorous analysis (omitted here because of space constraints) that the Bayes-decision leads to the same vote for all pairs from U : if $\mathbb{P}(E_+ \cap U) \geq \mathbb{P}(E_- \cap U)$ vote for “+”, otherwise vote for “−”. Clearly, the resulting Bayes-error equals $\min\{\mathbb{P}(E_+ \cap U), \mathbb{P}(E_- \cap U)\}$. It can be bounded from below as follows:

$$\mathbb{P}(U \cap E_+) \geq p_+ \cdot \left(\sqrt{\frac{\varepsilon}{p_+}} \right)^2 = \varepsilon ,$$

because $\sqrt{\frac{\varepsilon}{p_+}}$ coincides with the plus-conditional probability of a_s and b_t , respectively. A similar computation shows that

$$\mathbb{P}(U \cap E_-) \geq \underbrace{(1 - p_+)}_{\geq 1/2} \cdot (2\sqrt{n_1\varepsilon/n_2}) \cdot (2\sqrt{n_2\varepsilon/n_1}) \geq 2\varepsilon .$$

Thus, the Bayes-error is at least ε . \square

Corollary 3. *Let $n_1, n_2, d_1, d_2 \geq 1$, and, for $b = 1, 2$, let $\mathcal{C}_b = \text{SF}_{n_b+2}$ so that $\theta(\mathcal{C}_b) = \theta(\mathcal{C}_b^{[d_b]}) = n_b + 2$. Then, for every $p_{\min} \leq 1/(\theta_1\theta_2)$ and every sufficiently small $\varepsilon > 0$, the number of examples needed to learn $\mathcal{C}_1^{[d_1]}, \mathcal{C}_2^{[d_2]}$ in the PAC Co-training Model under the Conditional Independence Assumption is at least $\Omega(\sqrt{d_1 d_2 n_1 n_2}/\varepsilon)$.*

Proof. (Sketch) $\theta(\mathcal{C}_b) = \theta(\mathcal{C}_b^{[d_b]})$ follows from Lemma 5. A malign scenario for the classes $\mathcal{C}_b^{[d_b]}$ is obtained by installing the malign scenario from the proof of Lemma 6 (with some minor modifications) for each of the d_1 many copies of \mathcal{C}_1 and for each of the d_2 many copies of \mathcal{C}_2 . The main idea behind the proof is that every disjoint copy of the “old scenario” is now served by fewer sample points. In order to compensate this, the sample size must pop-up by factor $\sqrt{d_1 d_2}$. \square

Here comes the lower bound that is tight from the perspective of a worstcase analysis:

Theorem 4. *Assume that⁴ $3 \leq s_b^+, s_b^- < \infty$ for $b \in \{0, 1\}$. Then the following holds. For every sufficiently small $\varepsilon > 0$, the number of examples needed to learn $\mathcal{C}_1, \mathcal{C}_2$ in the PAC Co-training Model under the Conditional Independence Assumption is at least $\Omega(\sqrt{\max\{s_1^+ s_2^+, s_1^- s_2^-\}}/\varepsilon)$.*

⁴ One can drop the restriction $3 \leq s_b^+, s_b^-$ and still prove tight bounds, but that needs a tedious case distinction and is omitted due to space constraints.

Proof. We show that a target-distribution pair exists that needs $\Omega(\sqrt{s_1^+ s_2^+ / \varepsilon})$ many labels to be learned. By duality there is also a pair which needs at least $\Omega(\sqrt{s_1^- s_2^- / \varepsilon})$ many. Thus taking the maximum of both cases yields the theorem.

The former bound can be proved as follows: in the proof of Lemma 6, we may set $p_+ = p_{\min} = \varepsilon$. Thus the probability assigned to the redundant points a_0 and b_0 is now 0, respectively. Removal of the redundant point in a class of type SF will lead to the class of singletons. Thus, the proof of Lemma 6 with the special setting $p_+ = p_{\min} = \varepsilon$ shows that at least the same number of examples is needed for every pair $\mathcal{C}_1, \mathcal{C}_2$ of concept classes such that, for $b = 1, 2$, \mathcal{C}_b contains a singleton subclass of size $n_b + 2$. \square

Note that the lower bound in Theorem 4 nicely matches with the upper bound in Theorem 3 for small enough ε . Furthermore, this implies a weak converse of Theorem 1 where we have shown that $\text{VCdim}(\mathcal{H}) \cdot \theta(\mathcal{C}, \mathcal{H}) \leq s(\mathcal{C})$ for $\mathcal{H} = \mathcal{C}^+ \cup \mathcal{C}^-$. More precisely $s(\mathcal{C}) = \tilde{O}(\text{VCdim}(\mathcal{H}) \cdot \theta(\mathcal{C}, \mathcal{H}))$ must hold for every $\mathcal{H} \supseteq \mathcal{C}$ because, otherwise, the lower bound in Theorem 4 would exceed the upper bound $\tilde{O}(s(\mathcal{C})/\sqrt{\varepsilon})$, which follows directly from Corollary 2 and Theorem 1 with $\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{C}$.

The next step will be to provide lower bounds that remain valid even when p_{\min} takes more “benign values” than it does in the worstcase. Actually, Corollary 3 is a first step in this direction because the lower bound in this result nicely matches with the upper bound in Corollary 2 when $p_{\min} \leq 1/(\theta_1 \theta_2)$. We list here, without proof, some more lemmas of this kind which together witness that all upper bounds mentioned in Corollary 2 are fairly tight.

For any concept class \mathcal{C} over domain X , the class $\text{co}(\mathcal{C})$ is given by $\text{co}(\mathcal{C}) = \{X \setminus A \mid A \in \mathcal{C}\}$. Clearly, $\text{VCdim}(\mathcal{C}) = \text{VCdim}(\text{co}(\mathcal{C}))$ and $\theta(\mathcal{C}) = \theta(\text{co}(\mathcal{C}))$.

Lemma 7. Let $k, n \geq 1$, let $\mathcal{C}_1 = \text{SF}_{kn+2}$, and let $\mathcal{C}_2 = \text{co}(\text{SF}_{n+2})$, so that $\theta_{\max} = \theta(\mathcal{C}_1) = \theta(\mathcal{C}_1^{[d_1]}) = kn + 2$ and $\theta_{\min} = \theta(\mathcal{C}_2) = \theta(\mathcal{C}_2^{[d_2]}) = n + 2$. Then, for every sufficiently small $\varepsilon > 0$, the number of examples needed to learn $\mathcal{C}_1^{[d_1]}, \mathcal{C}_2^{[d_2]}$ in the PAC Co-training Model under the Conditional Independence Assumption is at least

$$\begin{cases} \Omega\left(\sqrt{\frac{d_1 d_2}{\varepsilon} \cdot \frac{\theta_{\min}}{p_{\min}}}\right) & \text{if } \frac{\theta_{\min}}{\theta_{\max}} \leq p_{\min} \leq \frac{1}{2} \\ \Omega\left(\sqrt{\frac{d_1 d_2}{\varepsilon} \cdot \theta_{\max}}\right) & \text{if } \frac{1}{\theta_{\max}} \leq p_{\min} \leq \frac{\theta_{\min}}{\theta_{\max}} \end{cases}.$$

Lemma 8. Let $n \geq 2$, let $\mathcal{C}_1 = \mathcal{C}_2 = \text{SF}_{n+2}$, so that $\theta_1 = \theta_2 = \theta_{\max} = n + 2$. Then, for every $1/(\theta_1 \theta_2) \leq p_{\min} \leq 1/\theta_{\max}$ and every sufficiently small $\varepsilon > 0$, the number of examples needed to learn $\mathcal{C}_1^{[d_1]}, \mathcal{C}_2^{[d_2]}$ in the PAC Co-training Model under the Conditional Independence Assumption is at least $\Omega\left(\sqrt{\frac{d_1 d_2}{\varepsilon p_{\min}}}\right)$.

The lower bounds in Corollary 3 and Lemmas 7 and 8 nicely match with the general upper bounds given in Corollary 2.

4.3 Sample Size in Case of One-Sided Errors

In the upper bounds presented in this section, any term of the form $d_b \theta_b$ can be safely replaced by $s(\mathcal{C}_b)$ provided that $\mathcal{H}_b = \mathcal{C}_b^+ \cup \mathcal{C}_b^-$. The proofs will be presented in the journal version.

Theorem 5. *For $b = 1, 2$, let $\mathcal{C}_b, \mathcal{H}_b$ be classes such that \mathcal{H}_b contains hypotheses with plus-sided errors (or with minus-sided errors, resp.) w.r.t. \mathcal{C}_b . Then sample size*

$$\begin{cases} \tilde{O}\left(\sqrt{\frac{d_1 d_2}{\varepsilon} \cdot \frac{1}{p_{min}}}\right) & \text{if } p_{min} \geq \frac{1}{\theta_1 \theta_2} \\ \tilde{O}\left(\sqrt{\frac{d_1 d_2}{\varepsilon} \cdot \theta_1 \theta_2}\right) & \text{otherwise} \end{cases}$$

is asymptotically sufficient for learning $\mathcal{C}_1, \mathcal{C}_2$ with hypotheses from $\mathcal{H}_1, \mathcal{H}_2$ in the PAC Co-training Model under the Conditional Independence Assumption.

Note that the upper bound from Theorem 5 applies to the special case where, for $b = 1, 2$, $\mathcal{H}_b = \mathcal{C}_b$ and \mathcal{C}_b is intersection-closed (or union-closed, resp.). In this case, the upper bound nicely matches with the lower bounds from Corollary 3 and Lemma 8.

Theorem 6. *For $b = 1, 2$, let $\mathcal{C}_b, \mathcal{H}_b$ be classes such that \mathcal{H}_1 contains hypotheses with plus-sided errors w.r.t. \mathcal{C}_1 , and \mathcal{H}_2 contains hypotheses with minus-sided errors w.r.t. \mathcal{C}_2 . Then sample size*

$$\begin{cases} \tilde{O}\left(\sqrt{\frac{d_1 d_2}{\varepsilon} \cdot \frac{\theta_{min}}{p_{min}}}\right) & \text{if } p_{min} \geq \frac{\theta_{min}}{\theta_{max}} \\ \tilde{O}\left(\sqrt{\frac{d_1 d_2}{\varepsilon} \cdot \theta_{max}}\right) & \text{otherwise} \end{cases}$$

is asymptotically sufficient for learning $\mathcal{C}_1, \mathcal{C}_2$ with hypotheses from $\mathcal{H}_1, \mathcal{H}_2$ in the PAC Co-training Model under the Conditional Independence Assumption.

Note that the upper bound from Theorem 6 applies to the special case where $\mathcal{H}_1 = \mathcal{C}_1$ is intersection-closed and $\mathcal{H}_2 = \mathcal{C}_2$ is union-closed. In this case, the upper bound nicely matches with the lower bound from Lemma 7.

5 Final Remarks

It is known that semi-supervised learners in the Co-training framework also benefit from assumptions weaker than conditional independence (see [2, 12]). One can ask whether PAC-learners can also use this more relaxed assumptions to their advantage. At least for the assumption introduced in [2] this is not generally true: for Co-training with an α -expanding distribution and one-sided errors, one can show that there are classes and distributions (e.g. ‘‘Example 1’’ from [2]) where every PAC-learner requires $\Omega(d/\varepsilon)$ many examples (with d denoting the VC-dimension of both views), which coincides with the standard PAC bounds. On the other hand, conditional independence given the label reduces the label complexity to $\tilde{O}(d/\sqrt{\varepsilon})$. Details will follow in the journal version of this paper.

We also looked into having more than two views. With k views under the Conditional Independence Assumption we can show that the upper bound for rule R3 becomes $m = \tilde{O}(\sqrt[k]{d_1\theta_1 \cdots d_k\theta_k/\varepsilon})$, and, as in the 2-view case, this has a matching lower bound. The other bounds can be generalized in a similar fashion.

The lower bound given in Theorem 4 is only valid for finite s_b^+, s_b^- , because the constraint on ε is essentially $1/\varepsilon \geq \max\{s_1^+ s_2^+, s_1^- s_2^-\}$. In case the singleton size is infinite, however, this theorem still implies some lower bound, namely $\Omega(1/\varepsilon)$. Nevertheless, this rules out the drastic reduction of the label complexity that we saw for $s_b^+, s_b^- < \infty$. To determine how much the Co-training assumption can help in this situation is work in progress.

In a broader context, it would be interesting to see whether the techniques of this paper can be applied to get new bounds on the unlabeled sample complexity in semi-supervised learning. Another interesting question is whether existing upper bounds in active learning (at least in the realizable case) can be reformulated in completely combinatorial terms using Theorem II.

References

- [1] Balcan, M.-F., Blum, A.: A discriminative model for semi-supervised learning. *Journal of the Association on Computing Machinery* 57(3), 19:1–19:46 (2010)
- [2] Balcan, M.-F., Blum, A., Yang, K.: Co-training and expansion: Towards bridging theory and practice. In: *Advances in Neural Information Processing Systems*, vol. 17, pp. 89–96. MIT Press, Cambridge (2005)
- [3] Ben-David, S., Lu, T., Pál, D.: Does unlabeled data provably help? Worst-case analysis of the sample complexity of semi-supervised learning. In: *Proceedings of the 21st Annual Conference on Learning Theory*, pp. 33–44 (2008)
- [4] Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pp. 92–100 (1998)
- [5] Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association on Computing Machinery* 36(4), 929–965 (1989)
- [6] Chapelle, O., Schölkopf, B., Zien, A.: *Semi-Supervised Learning*. MIT Press, Cambridge (2006)
- [7] Darnstädt, M., Simon, H.U.: Smart PAC-learners. *Theoretical Computer Science* 412(19), 1756–1766 (2011)
- [8] Geréb-Graus, M.: Lower bounds on parallel, distributed and automata computations. PhD thesis, Harvard University Cambridge, MA, USA (1989)
- [9] Hanneke, S.: A bound on the label complexity of agnostic active learning. In: *Proceedings of the 24th International Conference on Machine Learning*, pp. 353–360 (2007)
- [10] Sauer, N.: On the density of families of sets. *Journal of Combinatorial Theory, Series A* 13(1), 145–147 (1972)
- [11] Valiant, L.G.: A theory of the learnable. *Communications of the ACM* 27(11), 1134–1142 (1984)
- [12] Wang, W., Zhou, Z.-H.: A new analysis of co-training. In: *ICML*, pp. 1135–1142 (2010)

Learning a Classifier when the Labeling Is Known

Shalev Ben-David¹ and Shai Ben-David²

¹ Faculty of Mathematics, University of Waterloo, Waterloo, ON N2L 3G1, Canada
`shalevbd@gmail.com`

² David R. Cheriton School of Computer Science, University of Waterloo,
Waterloo, ON N2L 3G1, Canada
`shai@cs.uwaterloo.ca`

Abstract. We introduce a new model of learning, Known-Labeling-Classifier-Learning (KLCL). The goal of such learning is to find a low-error classifier from some given target-class of predictors, *when the correct labeling is known to the learner*. This learning problem can be viewed as measuring the information conveyed by the identity of input examples, rather than by their labels.

Given some class of predictors \mathcal{H} , a labeling function, and an *i.i.d.* unlabeled sample generated by some unknown data distribution, the goal of our learner is to find a classifier in \mathcal{H} that has as low as possible error with respect to the sample-generating distribution and the given labeling function. When the labeling function does not belong to the target class, the error of members of the class (and thus their relative quality as label predictors) varies with the marginal of the underlying data distribution.

We prove a trichotomy with respect to the KLCL sample complexity. Namely, we show that for any learnable concept class \mathcal{H} , its KLCL sample complexity is either 0 or $\Theta(1/\epsilon)$ or $\Omega(1/\epsilon^2)$. Furthermore, we give a simple combinatorial property of concept classes that characterizes this trichotomy.

Our results imply new sample-size lower bounds for the common agnostic PAC model - a lower bound of $\Omega(1/\epsilon^2)$ on the sample complexity of learning *deterministic* classifiers, as well as novel results about the utility of unlabeled examples in a semi-supervised learning setup.

1 Introduction

Most of the work in classification learning focuses on learning a good labeling function. We consider a somewhat different problem - learning to classify when the target labeling is known. Can learning a classifier be a challenge to a learner that already knows the correct classification of every domain point? It can, if the classifier is required to belong to a given class of classifiers, and the correct classification is not a member of that class. In such cases, in order to discover the minimal error classifier in the class, one needs to estimate the underlying marginal distribution of the data.

In the common PAC learning model, the empirical information available to the learner is in the form of a randomly generated sample of labeled points. Such a sample can be viewed as containing two types of information: information about the labeling rule, and information about the underlying (marginal) data distribution. In this work we wish to focus on the second aspect. We do so by taking an unusual step - assuming that the data labeling rule is a priori known to the learner (and is deterministic, so there is no labeling uncertainty whatsoever).

1.1 Our Results

Our analysis yields a trichotomy with respect to the KLCL sample complexity. Namely, we show that for any learnable concept class \mathcal{H} , its KLCL sample complexity is either 0 or $\Theta(1/\epsilon)$ or $\Omega(1/\epsilon^2)$. Furthermore, we give a simple combinatorial property of concept classes that characterizes this trichotomy.

Since in the KLCL model the labels are known to the learner, clearly, learning in this model is easier than learning in the PAC model. In particular, any sample complexity upper bound for (proper) PAC learning readily applies to the KLCL setup as well. The more intriguing question seems to be lower bounds; how hard can it be to learn a predictor when the complete labeling information is known right from the onset of the learning process? The focus of this paper is, therefore, on lower bounding the sample complexity of learning in the KLCL model. Our main lower bound result is based on deriving a lower bound on the number of die rolls needed to detect a bias in a k -face die, and may be of independent interest.

On the other hand, we show that KLCL is strictly easier than PAC learning in the following sense: there exist classes \mathcal{H} that are KLCL learnable but not PAC learnable (they have infinite VC-dimension).

In contrast with that, we show in Section 6 that KLCL is not easier than usual classification learning, in the sense that for every concept class \mathcal{H} there is a corresponding class \mathcal{H}^L such that learning the class \mathcal{H}^L in the KLCL model w.r.t. the constant zero function is as hard as learning \mathcal{H} in the agnostic PAC model.

1.2 Applications to Semi-supervised Learning

The distinction between the information conveyed by the labels of sample points and the information conveyed by the identity of the sampled points (regardless of their labels) arises naturally in the context of semi-supervised learning (SSL). In that model, one basically asks "to what extent can unlabeled samples be utilized to reduce the size of the labeled sample needed for learning?" By separating the two types of sample information, as described above, we can alternatively ask "how much of the needed sample is required to learn the labeling in comparison to how much of it is required to learn the relevant patterns of the data marginal distribution?" This distinction can be put into use in the *proper* SSL learning task, where the learner is required to output a classifier that belongs to some pre-determined class \mathcal{H} . Proper learning arises in situations where either the simplicity of a predictor (say, from the point of view of the speed of prediction it allows) or its interpretability are of primal importance. For example, the learner's goal might be to provide patients with some medical advice. In such cases, the

learner may wish to sacrifice some of the accuracy of a predictor to allow coming up with a more user friendly predicting rule. In that model, a possible learning strategy may be to first learn a low error predictor, f , not necessarily in \mathcal{H} , and then use the unlabeled data to determine which $h \in \mathcal{H}$ is closest to that first predictor w.r.t the marginal distribution. Namely, find $h \in \mathcal{H}$ that minimizes $D(h\Delta f)$ (where D is the marginal of the data distribution); see [4] for more details on such an approach. This second step is exactly what our model focuses on. Our sample complexity lower bounds readily translate into lower bounds on the needed unlabeled sample size for carrying out the second step of the above proper SSL learning paradigm.

1.3 Outline of the Paper

We start in Section 2 by introducing our combinatorial characterization of KLCL trichotomy of concept classes. In Section 3 we state the main trichotomy theorem, 1, and prove its first (and easy) part. In Section 4 we prove an upper bound as well as a matching lower bound for the KLCL sample complexity of concept classes of infinite VC-dimension that do not shatter some infinite set, proving the second part of theorem 1. We present our main statistical tool, the biased dice problem, and analyze its information complexity in subsection 5.1. The following subsection, 5.2, proves the reduction from the biased dice problem to the KLCL learning problem, proving the third part of theorem 1. In Section 6 we show a sample complexity preserving reduction of agnostic PAC learning to KLCL learning, just showing that the KLCL framework gives rise to some hard and also un-learnable tasks. Finally, in Section 7 we summarize our work and call attention to some major remaining open problems.

2 Basic Definitions

Given some domain set X and a hypothesis class \mathcal{H} of classifiers over X , the Known-Labeling Classifier Learning (KLCL) problem for \mathcal{H} is defined as follows:

Input: A function $f : X \rightarrow \{0, 1\}$ and a sample (x_1, \dots, x_m) of members of X generated i.i.d. by some unknown probability distribution, D , over X .

Goal: Find $h \in \mathcal{H}$ that minimizes the error $\text{Err}_D^f(h) \stackrel{\text{def}}{=} D[\{x \in X : f(x) \neq h(x)\}]$.

In this work, we are interested in the sample complexity of this problem, as a function of the structure of the hypothesis class \mathcal{H} .

Definition 1. Given X and \mathcal{H} as above,

1. A KLCL learner for \mathcal{H} (an \mathcal{H} -learner, in short) is a function $L : \{0, 1\}^X \times \bigcup_{m=1}^{\infty} X^m \rightarrow \mathcal{H}$. That is, a learner that takes as input a labeling function $f : X \rightarrow \{0, 1\}$ and a finite sample of points in X and outputs a classifier from \mathcal{H} .

2. Given a probability distribution, D , over X ,

$$\text{Err}_D^f(\mathcal{H}) \stackrel{\text{def}}{=} \inf\{\text{Err}_D^f(h) : h \in \mathcal{H}\}.$$

3. Given $\epsilon, \delta > 0$ and a class \mathcal{H} , the (ϵ, δ) -sample-complexity of a learner L over \mathcal{H} is the minimum sample size m such that, for every $f : X \rightarrow \{0, 1\}$ and every D , if L is given f , then with probability exceeding $1 - \delta$ over samples $S \sim D^m$,

$$\text{Err}_D^f(L(f, S)) < \text{Err}_D^f(\mathcal{H}) + \epsilon.$$

4. The (ϵ, δ) -sample-complexity of a class \mathcal{H} is the minimum (ϵ, δ) -sample-complexity over all \mathcal{H} learners.

Definition 2. We say that a class of functions \mathcal{H} is redundant over some domain element $x \in X$ if for all $h, h' \in \mathcal{H}$, $h(x) = h'(x)$.

Definition 3. Define $A_{\mathcal{H}} = \{x \in X : \mathcal{H} \text{ is not redundant over } x\}$. We classify the possible classes of functions \mathcal{H} over a domain set X into three mutually exclusive types, based on their behavior on $A_{\mathcal{H}}$.

1. We say that \mathcal{H} is simple if \mathcal{H} shatters $A_{\mathcal{H}}$.
2. We say that \mathcal{H} is pseudo-simple if $A_{\mathcal{H}}$ is infinite and \mathcal{H} does not shatter $A_{\mathcal{H}}$, but shatters every finite subset of $A_{\mathcal{H}}$.
3. We say that \mathcal{H} is non-simple if there exists some finite subset of $A_{\mathcal{H}}$ that is not shattered by \mathcal{H} .

It is straightforward to check that each class of functions \mathcal{H} is of exactly one of the above three types. In addition, if \mathcal{H} has finite VC dimension, then \mathcal{H} cannot be pseudo-simple.

We are now ready to state our main theorem.

3 Main Result

The central result of this paper is the following crisp characterization of the KLCL sample complexity of a given class as a function of the accuracy parameter ϵ .

Theorem 1 (The Main Theorem). For any hypothesis class \mathcal{H} ,

1. If \mathcal{H} is simple then the KLCL sample complexity of \mathcal{H} is zero.
2. If \mathcal{H} is pseudo-simple and X is countable, then the KLCL sample complexity of \mathcal{H} is $\Theta\left(\frac{1}{\epsilon} \log \frac{1}{\delta}\right)$.
3. If \mathcal{H} is non-simple, then the KLCL sample complexity of \mathcal{H} is $\Omega\left(\frac{1}{k} \frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$, assuming $\epsilon < \frac{1}{k+1}$.

The k factor in the above theorem is the "all-shattered dimension" of \mathcal{H} : the largest integer such that all subsets of $A_{\mathcal{H}}$ of size k are shattered by \mathcal{H} . Notice that k is bounded above by the VC dimension of \mathcal{H} , but is often much smaller. For example, if \mathcal{H} is the class of linear half spaces in some \mathbb{R}^d , then its VC dimension is $d + 1$, while its all-shattered dimension is just 2, regardless of what d is.

Some Remarks:

- The theorem does not deal with the case that X is uncountable and \mathcal{H} is pseudo-simple.
- When X is countable and \mathcal{H} is pseudo-simple, then \mathcal{H} has infinite VC dimension, and hence part 2 of the theorem implies that it is possible to learn in the KLCL model even when \mathcal{H} is not learnable in the usual agnostic PAC sense (since it has infinite VC dimension).

Corollary 1. *If \mathcal{H} is a non-simple hypothesis class then the sample complexity of learning \mathcal{H} in the agnostic PAC model w.r.t deterministic labeling functions is $\Omega(1/\epsilon^2)$ (where ϵ is the excess error of the learned hypothesis over that of the best predictor in \mathcal{H} .)*

Proof (of the corollary). Just note that KLCL learning is strictly easier than PAC learning. Having access to the labeling function, f , and an unlabeled sample of points a KLCL learner can readily label the sample and use it as an input to an agnostic PAC learner. The success measure is the same in both models.

Proof (of part 1 of the theorem). Suppose \mathcal{H} is simple, and consider any labeling function f . Find $h \in \mathcal{H}$ that agrees with f on $A_{\mathcal{H}}$ (this is possible as \mathcal{H} shatters $A_{\mathcal{H}}$). Then $\text{Err}_D^f(\mathcal{H}) = \text{Err}_D^f(h)$ and a learner L that outputs h without seeing any input has error equal to $\text{Err}_D^f(h)$ with probability 1, so the difference between the two is always 0. This means the learner L satisfies every (ϵ, δ) pair without seeing any input, so the sample complexity is zero.

The remainder of the paper is devoted to proving parts 2 and 3 of theorem \blacksquare

4 KLCL of Classes of Infinite VC Dimension

In this section we prove claim 2 of theorem \blacksquare . Namely, that if a \mathcal{H} is a pseudo-simple class over some countable domain set, X , then the KLCL sample complexity of \mathcal{H} is $\Theta(\frac{1}{\epsilon} \log \frac{1}{\delta})$. We need to prove both lower and upper bounds on the sample complexity of \mathcal{H} .

Lemma 1 (Upper bound). *If \mathcal{H} is pseudo-simple over some countable domain set, X , then, there exists a KLCL learner, L , that for every $f : X \rightarrow \{0, 1\}$ and every probability distribution, D , over X , if $m \geq \frac{\log(1/\delta)}{\epsilon}$ then, with probability $> (1 - \delta)$ over samples $S \sim D^m$, $\text{Err}_D^f(L(S)) \leq \text{Err}_D^f(\mathcal{H}) + \epsilon$.*

Proof. Let $X = \{a_n : n \in \mathbb{N}\}$. Let L be a learner that, upon seeing a sample S outputs some $h \in \mathcal{H}$ such that, for all $x \in \{a_n : \exists a_i \in S \text{ such that } n \leq i\} \cap A_{\mathcal{H}}$, $f(x) = h(x)$. Note that such an h exists due to the pseudo-simplicity of \mathcal{H} .

Now, given any probability distribution D over X , let $k(\epsilon, D)$ be the maximal number, k , such that $D(\{a_n : n > k\}) > \epsilon$. Clearly, if $S \cap \{a_n : n > k(\epsilon, D)\} \neq \emptyset$ then indeed $\text{Err}_D^f(L(S)) \leq \text{Err}_D^f(\mathcal{H}) + \epsilon$. It follows that the probability of failure of L satisfies $\delta \leq (1 - \epsilon)^m$. Standard calculation shows that this implies that $m(\epsilon, \delta) \leq \frac{\log(1/\delta)}{\epsilon}$.

Lemma 2 (Lower bound). *Let X be a countable domain and \mathcal{H} a pseudo-simple class over X . Given $\epsilon, \delta > 0$, let $m < C(1/\epsilon)$, for some constant C . Then, there exists a function $f : X \rightarrow \{0, 1\}$ such that for every KLCL learner for \mathcal{H} , there exists a probability distribution D over X , such that with probability $> \delta$ over samples $S \sim D^m$, $\text{Err}_D^f(L(S)) \geq \text{Err}_D^f(\mathcal{H}) + \epsilon$.*

Proof. Let $f : X \rightarrow \{0, 1\}$ be such that for every $h \in \mathcal{H}$ there exists $x \in A_{\mathcal{H}}$ such that $h(x) \neq f(x)$. Pick some $x_0 \in X$. Now given any learner L for \mathcal{H} , and a number m , let S be a sample of size m consisting of just repetitions of x_0 . Let $y_0 \in A_{\mathcal{H}}$ be such that $f(y_0) \neq L(S)(y_0)$ (such y_0 exists by our choice of f). We define D to be the distribution whose support is $\{x_0, y_0\}$ and picks x_0 with probability $1 - \epsilon$ and picks y_0 with probability ϵ . Clearly, whenever a sample S does not include the point y_0 , $\text{Err}_D^f(L(S)) \geq \text{Err}_D^f(\mathcal{H}) + \epsilon$. The proof is concluded by noting that in order to drive the probability of missing y_0 below δ , the needed sample size is $\Omega(\frac{1}{\epsilon} \log \frac{1}{\delta})$.

5 KLCL Learning of Finite VC-Classes

In this section we prove the third part of theorem 1. We do that by analyzing the information complexity of some weighted dice bias-detection problem, and reducing it to our KLCL task.

5.1 The Weighted Dice Problem

Consider the following problem. There are $k \geq 2$ dice, each of which has k sides, numbered 1 through k . For the i -th die, the probability of rolling i is ϵ less than the probability of rolling every other number, but the die is otherwise unbiased. We denote the i -th die by P_i , with the notation that $P_i(j)$ is the probability that rolling the i -th die gives j . Then

$$P_i(j) = \begin{cases} \frac{1+\epsilon}{k} & \text{if } i \neq j \\ \frac{1+\epsilon}{k} - \epsilon & \text{if } i = j \end{cases}$$

One of these dice is rolled m times. A learner gets the outcome of these m die rolls, and has to determine which of the k dice generated that sample.

The idea behind lower bounding the sample complexity of learning in the KLCL model (in the non-simple case) is to reduce the weighted die problem to the given learning task and apply a sample size lower bound for the weighted die problem.

Theorem 2. *If the die is picked randomly, with probability $1/k$ for each die, then for any $0 < \delta < 1/4$ and any $0 < \epsilon < 1$, if the number of seen die rolls, m , is at most*

$$k \left\lceil \frac{1}{k} \left(\frac{1}{k-1} - \epsilon \left(1 - \frac{1}{k} + \epsilon \right) \right) \frac{1}{\epsilon^2} \ln \left(\frac{1}{4 \frac{k}{k-1} \delta \left(1 - \frac{k}{k-1} \delta \right)} \right) \right\rceil$$

then any algorithm for the weighted die problem (i.e., a function that takes the outcome of the die rolls as input and outputs a die P_i for some $i \leq k$) will output a wrong die with probability at least δ .

Note that a simpler but weaker bound that follows from the above is that if $\epsilon < \frac{1}{k}$ and m is less than

$$k \left\lfloor \frac{1}{k} \left(\frac{1}{k-1} - \epsilon \right) \frac{1}{\epsilon^2} \ln \left(\frac{1}{4\delta} \left(1 - \frac{1}{k} \right) \right) \right\rfloor$$

then any algorithm has probability of error above δ . When ϵ and δ are small relative to $1/k$, this lower bound is roughly equal to

$$\left(\frac{1}{k-1} \right) \frac{1}{\epsilon^2} \ln \left(\frac{1}{\delta} \right)$$

Our analysis extends the analysis of estimating the parameter of a Bernoulli variable, as carried out in the proof of Lemma 5.1 of [1]. We first argue that the learner which picks the die corresponding to the face least seen is the best possible learner.

Definition 4. Let L_0 be the learner that counts the number of occurrences of each face j in the given sample, and outputs die j only if j is a face with least occurrences in the sample. In the case of a tie, L_0 outputs j if j is the least index such that face j is a face with least occurrences in the sample.

Let L be any other learner. We claim that the probability of error of L is at least that of L_0 . The proof of this is simple and follows a similar argument to the one in the proof of Lemma 5.1 in [1].

Proof. We analyze the probability of error of L_0 given a sample of size m . We first analyze the conditional probability that L_0 makes an error given that the correct die is P_i . Under this condition, the probability that L_0 makes an error is at least the probability that face i is not a face of least occurrence.

We restrict ourselves to the case where m is a multiple of k . Notice that if face i occurs more than m/k times in the sample, then it cannot be a face of least occurrence, since there must be a face which occurs at most m/k times in the sample. Moreover, if face i occurs exactly m/k times, then there will still be a face occurring less times than face i unless all faces occur exactly m/k times; in that case, L_0 still makes an error unless $i = 1$. We restrict ourselves to the case that $i \neq 1$. In this case, we conclude that given that the correct die is P_i , the probability that L_0 makes an error is at least the probability that face i occurs at least m/k times.

Now, if the correct die is P_i , let S denote the random variable for the number of occurrences of face i in the sample. Then S follows a binomial distribution with probability of success $\frac{1+\epsilon}{k} - \epsilon$. Therefore, if there are m die rolls, the probability that the number of occurrences of face i in the sample exceeds m/k is $P[S \geq \frac{m}{k}]$.

We use Slud's inequality ([2]), which states that if $S \sim \text{Bin}(m, p)$ where $p \leq \frac{1}{2}$ and b is an integer with $mp \leq b \leq m(1 - p)$ then

$$P[S \geq b] \geq P\left[Z \geq \frac{b - mp}{\sqrt{mp(1 - p)}}\right]$$

where $Z \sim N(0, 1)$ is a normally distributed random variable with mean of 0 and standard deviation of 1. We then apply a normal tail bound [3] which states that if $x \geq 0$ then

$$P[Z \geq x] \geq \frac{1}{2} \left(1 - \sqrt{1 - e^{-x^2}}\right).$$

Our bound is based on the composition of Slud's inequality with this normal tail bound.

Set $p = \frac{1+\epsilon}{k} - \epsilon$ and $b = \frac{m}{k}$. Then $S \sim \text{Bin}(m, p)$. We first verify that Slud's inequality applies (that is, we check that the required conditions on m , p , and b all hold).

We have

$$p = \frac{1+\epsilon}{k} - \epsilon = \frac{1}{k} - \epsilon \left(1 - \frac{1}{k}\right) < \frac{1}{k} \leq \frac{1}{2},$$

since $k \leq 2$ and $\epsilon > 0$. Also, since $p < \frac{1}{k}$, we have

$$b = \frac{m}{k} > mp.$$

Finally, since $p < \frac{1}{k} \leq \frac{1}{2}$, $1 - p > \frac{1}{k}$, and therefore $m(1 - p) \geq \frac{m}{k} = b$. so Thus Slud's inequality applies. We can now write

$$\begin{aligned} P[S \geq b] &\geq P\left[Z \geq \frac{b - mp}{\sqrt{mp(1 - p)}}\right] \\ &= P\left[Z \geq \frac{\frac{m}{k} - m \left(\frac{1}{k} - \epsilon \left(1 - \frac{1}{k}\right)\right)}{\sqrt{m \left(\frac{1}{k} - \epsilon \left(1 - \frac{1}{k}\right)\right) \left(1 + \epsilon\right)^{\frac{k-1}{k}}}}\right] \\ &= P\left[Z \geq \frac{m \epsilon^{\frac{k-1}{k}}}{\sqrt{m \left(1 - \epsilon \left(k - 1\right)\right) \left(1 + \epsilon\right)^{\frac{k-1}{k^2}}}}\right] \\ &= P\left[Z \geq \sqrt{\frac{m \epsilon^2 (k - 1)}{(1 - \epsilon (k - 1)) (1 + \epsilon)}}\right] \end{aligned}$$

Composing this with the normal tail bound gives

$$P\left[S \geq \frac{m}{k}\right] \geq \frac{1}{2} \left(1 - \sqrt{1 - \exp\left(-\frac{m \epsilon^2 (k - 1)}{(1 - \epsilon (k - 1)) (1 + \epsilon)}\right)}\right).$$

We conclude that the conditional probability that L_0 makes an error given that the die is P_i with $i \neq 1$ is at least

$$\frac{1}{2} \left(1 - \sqrt{1 - \exp \left(-\frac{m\epsilon^2(k-1)}{(1-\epsilon)(k-1)(1+\epsilon)} \right)} \right),$$

where m is the size of the sample and is assumed to be a multiple of k . We now write

$$\begin{aligned} P[L_0 \text{ makes an error}] &= \sum_{i=1}^k P[L_0 \text{ makes an error} \mid \text{the die is } i] P[\text{the die is } i] \\ &= \sum_{i=1}^k P[L_0 \text{ makes an error} \mid \text{the die is } i] \frac{1}{k} \\ &> \frac{1}{k} \sum_{i=2}^k P[L_0 \text{ makes an error} \mid \text{the die is } i] \\ &\quad (\text{note that we removed the case } i=1) \\ &\geq \frac{1}{k} \sum_{i=2}^k \frac{1}{2} \left(1 - \sqrt{1 - \exp \left(-\frac{m\epsilon^2(k-1)}{(1-\epsilon)(k-1)(1+\epsilon)} \right)} \right) \\ &= \frac{k-1}{2k} \left(1 - \sqrt{1 - \exp \left(-\frac{m\epsilon^2(k-1)}{(1-\epsilon)(k-1)(1+\epsilon)} \right)} \right) \end{aligned}$$

Thus, if the probability of error of an algorithm L is less than δ given a sample of size m where m is a multiple of k , then we have

$$\delta > \frac{k-1}{2k} \left(1 - \sqrt{1 - \exp \left(-\frac{m\epsilon^2(k-1)}{(1-\epsilon)(k-1)(1+\epsilon)} \right)} \right).$$

Separating m out of the above inequality gives

$$m > \left(\frac{1}{k-1} - \epsilon \left(1 - \frac{1}{k} + \epsilon \right) \right) \frac{1}{\epsilon^2} \ln \left(\frac{1}{4 \frac{k}{k-1} \delta \left(1 - \frac{k}{k-1} \delta \right)} \right).$$

Finally, we deal with the case that m is not a multiple of k by rounding it up to the nearest multiple of k . Hence we conclude that if the probability of error is at most δ , then

$$m > k \left[\frac{1}{k} \left(\frac{1}{k-1} - \epsilon \left(1 - \frac{1}{k} + \epsilon \right) \right) \frac{1}{\epsilon^2} \ln \left(\frac{1}{4 \frac{k}{k-1} \delta \left(1 - \frac{k}{k-1} \delta \right)} \right) \right]$$

and the desired result follows.

5.2 Reducing the Dice Problem to KLCL Learning

We now apply the lower bound on the biased dice problem to prove part 3 of theorem □

Proof. Assume \mathcal{H} is not simple, let $A \subseteq \mathcal{A}_{\mathcal{H}}$ be a set un-shattered by H of minimum size and let $f : A \rightarrow \{0, 1\}$ be unrealized by H . Then every subset of A is shattered by H . Notice that the size of A is at most one more than the VC-dimension of H , which is assumed to be finite. Let $k = |A| - 1$. Notice that each subset of A of size k is shattered by H , so that k is the all-shattered dimension of \mathcal{H} . Moreover, for each subset $B \subset A$ of size k , there is some $h_B \in H$ such that $h_B|_B = f|_B$ (that is, h_B agrees with f on B). Let $C = \{h_B : B \subset A \text{ and } |B| = k\} \subseteq H$.

We now reduce the weighted dice problem with $k + 1$ dice, for fixed ϵ , and probability of error δ to the problem of finding $h \in H$ with $\text{Err}_P^f(h) \leq \text{Err}_P^f(H) + \epsilon$ with probability of error δ . We do this by defining $k+1$ probability distributions on the domain set, each of which has zero weight outside of the set A . The distributions will be the same as those in the weighted dice problem.

Label the elements of A by the set $\{1, 2, \dots, k + 1\}$, so that we have $A = \{a_1, a_2, \dots, a_{k+1}\}$. Let $\epsilon \in (0, 1/(k + 1))$ and $\delta \in (0, 1/4)$. For $i = 1, 2, \dots, k + 1$, define the probability distribution P_i on A by

$$P_i(a_j) = \begin{cases} \frac{1+\epsilon}{k+1} & \text{if } i \neq j \\ \frac{1+\epsilon}{k+1} - \epsilon & \text{if } i = j \end{cases}.$$

Suppose we have a learner L with the property that for all probability distributions P on the domain, if S is an *i.i.d.* P sample of size m , then, with probability $\geq 1 - \delta$,

$$\text{Err}_P^f(L(f, S)) < \text{Err}_P^f(H) + \epsilon.$$

In particular, L will have this property for the distributions P_i , $i = 1, 2, \dots, k + 1$. Notice that for each i , the best approximation to f under the probability distribution P_i is one of the functions in C , since those are the functions that agree with f on all of A except for one element. In addition, for each i , we have

$$\text{Err}_{P_i}^f(h_i) = \frac{1 + \epsilon}{k + 1} - \epsilon$$

where h_i is the function in C that agrees with f on all of A except for a_i , and

$$\text{Err}_{P_i}^f(h) = \frac{1 + \epsilon}{k + 1}$$

for any function other function $h \in H$. Thus

$$\text{Err}_{P_i}^f(H) \geq \frac{1 + \epsilon}{k + 1} - \epsilon,$$

and since

$$\text{Err}_P^f(L(f, S)) < \text{Err}_P^f(H) + \epsilon$$

we must have

$$L(f, S) = h_i$$

with probability at least $1 - \delta$.

Now pick i with uniform distribution on $\{1, 2, \dots, k + 1\}$ and generate an *i.i.d.* sample S of m elements of A with distribution P_i . Then with probability at least $1 - \delta$, we have $L(f, S) = h_i$. Since we can determine i from h_i , we have an algorithm for the weighted dice problem which uses m rolls of the dice and has probability of error less than δ . By theorem 2, we have

$$m \geq (k + 1) \left\lfloor \frac{1}{k + 1} \left(\frac{1}{k} - \epsilon \left(1 - \frac{1}{k + 1} + \epsilon \right) \right) \frac{1}{\epsilon^2} \ln \left(\frac{1}{4 \frac{k+1}{k} \delta \left(1 - \frac{k+1}{k} \delta \right)} \right) \right\rfloor.$$

We conclude that the (ϵ, δ) -sample complexity of H is at least

$$\begin{aligned} (k + 1) & \left\lfloor \frac{1}{k + 1} \left(\frac{1}{k} - \epsilon \left(1 - \frac{1}{k + 1} + \epsilon \right) \right) \frac{1}{\epsilon^2} \ln \left(\frac{1}{4 \frac{k+1}{k} \delta \left(1 - \frac{k+1}{k} \delta \right)} \right) \right\rfloor \\ &= \Omega \left(\frac{1}{k} \frac{1}{\epsilon^2} \log \frac{1}{\delta} \right), \end{aligned}$$

as desired.

6 Reduction of Usual Learning to KLCL and Classes That Are Not KLCL Learnable

In this section we show that the usual agnostic PAC learning problem can be reduced to KLCL. Furthermore, using that reduction, we construct a hypothesis class \mathcal{H} that is not KLCL learnable.

Given a hypothesis class $h : X \rightarrow \{0, 1\}$, define a function $h^L : X \times \{0, 1\} \rightarrow \{0, 1\}$ by setting $h^L(x, \ell) = 0$ if $h(x) = \ell$ and $h^L(x, \ell) = 1$ if $h(x) \neq \ell$. Given a hypothesis class \mathcal{H} over some domain set X , let \mathcal{H}^L be the class $\{h^L : h \in \mathcal{H}\}$ of predictors over $X \times \{0, 1\}$.

Given any probability distribution, P over $X \times \{0, 1\}$, note that, for every $h : X \rightarrow \{0, 1\}$, the error of h w.r.t. P equals the error $\text{Err}_P^{\bar{0}}(h^L)$, where $\bar{0}$ is the constant zero function.

It follows that, for every class \mathcal{H} over X and every $\epsilon, \delta > 0$, if the class \mathcal{H}^L can be (ϵ, δ) learned from $m(\epsilon, \delta)$ examples in the KLCL model, then the class \mathcal{H} is (ϵ, δ) learned from $m(\epsilon, \delta)$ examples in the usual agnostic PAC model.

It is now easy to construct a class \mathcal{H}_0 that is not KLCL learnable. To achieve that, take any class of functions, \mathcal{H} , over some domain set X , that has infinite VC dimension (and thus, it is not learnable), now the class \mathcal{H}^L (over the domain $X \times \{0, 1\}$) is not learnable in the KLCL model, even just w.r.t. the all-zero labeling function.

7 Conclusions and Future Work

In this work we defined a new variant of the statistical learning problem. Our KLCL problem focuses on the information conveyed by the unlabeled part of a training sample (or, if you wish, the marginal of the empirical probability). Our way of addressing that aspect is by considering proper learning when the learner already knows what the labeling function is. Lower bounds on the sample complexity of learning in this model readily apply to the sample complexity of learning with deterministic labeling functions in the usual PAC model. In this paper, most of our analysis focused on lower bounding that sample complexity. It turns out that a critical parameter in such lower bounds is the "all-shattered-dimension" - the maximal size so that every non-redundant domain subset of that size is shattered. The all-shattered-dimension is always bounded by the VC-dimension, but is much lower for many common classes. Considering sample complexity upper bounds, clearly any upper bound in the usual agnostic PAC model applies to our model as well. However there is a curious discrepancy between such upper bounds and the lower bound proven here. The known upper bounds grow with the VC dimension, while our lower bound shrinks as the all-shattered-dimension grows. In section 6, we show that KLCL is not easier than the usual classification learning in the sense that for every concept class \mathcal{H} there is a corresponding class \mathcal{H}^L such that learning the class \mathcal{H}^L in the KLCL model w.r.t. the constant zero function is as hard as learning \mathcal{H} in the agnostic PAC model. An interesting open question is therefore to classify when the KLCL problem for a class \mathcal{H} can be solved with sample sizes lower than those needed for PAC learning \mathcal{H} .

We hope that this paper will stimulate further research in these directions, research that will shed more light on the distinction between the information conveyed by the labels of a training sample and the information conveyed by its marginal (empirical) distribution.

References

- [1] Anthony, M., Bartlett, P.L.: Neural Network Learning: Theoretical Foundations. Cambridge University Press, Cambridge (1999)
- [2] Slud, E.: Distribution inequalities for the binomial law. Annals of Probability 5, 404–412 (1977)
- [3] Tate, R.F.: On a double inequality of the normal distribution. Annals of Mathematical Statistics 24, 132–134 (1953)
- [4] Urner, R., Ben-David, S., Shalev-Shwartz, S.: Unlabeled data can speed up prediction time. In: ICML (to appear, 2011)

Erratum: Learning without Coding

Samuel E. Moelius III¹ and Sandra Zilles²

¹ IDA Center for Computing Sciences
17100 Science Drive, Bowie, MD 20715-4300
semoeli@super.org
² Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada S4S 0A2
zilles@cs.uregina.ca

Our ALT'2010 paper claimed that every computably finitely thick [LZ96, Definition 9] class of languages can be identified by enumeration operator [MZ10, Definition 1(e) and Theorem 12]. However, this is, in fact, *false*. We intend to include a proof of the claim's negation in the journal version of our paper, which has been submitted.

Acknowledgements. We would like to thank Sanjay Jain for noticing our error, and for providing us a proof of the claim's negation.

References

- [LZ96] Lange, S., Zeugmann, T.: Incremental learning from positive data. *Journal of Computer and System Sciences* 53(1), 88–103 (1996)
- [MZ10] Moelius, S., Zilles, S.: Learning without coding. In: Hutter, M., Stephan, F., Vovk, V., Zeugmann, T. (eds.) ALT 2010. LNCS(LNAI), vol. 6331, pp. 300–314. Springer, Heidelberg (2010)

Author Index

- Audibert, Jean-Yves 159
Auer, Peter 14, 189
- Ben-David, Shai 440
Ben-David, Shalev 440
Bengio, Yoshua 18
Bubeck, Sébastien 144
- Carpentier, Alexandra 189
Cortes, Corinna 308
Crammer, Koby 114
- Dalalyan, Arnak S. 129
Darnstädt, Malte 425
Delalleau, Olivier 18
- Fürnkranz, Johannes 38
- Garivier, Aurélien 174
Gavane, Vaibhav 262
Geilke, Michael 84
Gerchinovitz, Sébastien 99
Ghavamzadeh, Mohammad 189
Gofer, Eyal 234
Grigorescu, Elena 413
- Hatano, Kohei 324
Helmbold, David P. 219
Hüllermeier, Eyke 38
Hutter, Marcus 262, 338, 368, 383
- Jain, Sanjay 55, 70
Juba, Brendan 277
- Kasprzik, Anna 398
Kivinen, Jyrki 1
Koolen, Wouter M. 219
Kötzing, Timo 40
- Lattimore, Tor 262, 368, 383
Lazaric, Alessandro 189
Li, Ming 39
Lim, Shiau Hong 14
Lu, Chi-Jen 249
Lu, Wei-Fu 249
- Mansour, Yishay 234
Martin, Eric 55, 70
Moelius III, Samuel E. 452
Mohri, Mehryar 308
Moulines, Eric 174
Munos, Rémi 189
- Orseau, Laurent 353
- Panagiotakopoulos, Constantinos 204
- Reyzin, Lev 413
Rissanen, Jorma 37
- Saha, Ankan 292
Salmon, Joseph 129
Salomon, Antoine 159
Simon, Hans Ulrich 425
Stephan, Frank 55, 70
Stoltz, Gilles 144
Suehiro, Daiki 324
Sunehag, Peter 338
Szepesvári, Csaba 1
Szörényi, Balázs 425
- Takimoto, Eiji 324
Tsampouka, Petroula 204
- Ukkonen, Esko 1
- Vaits, Nina 114
Vempala, Santosh 277, 413
Vishwanathan, S.V.N. 292
- Warmuth, Manfred K. 219
Watkins, Chris 14
- Yoshinaka, Ryo 398
Yu, Jia Yuan 99, 144
- Zeugmann, Thomas 1
Zhang, Xinhua 292
Zilles, Sandra 84, 452