

INTELLIGENT SYSTEMS REFERENCE LIBRARY
Volume 23

Dawn E. Holmes
Lakhmi C. Jain (Eds.)

Data Mining: Foundations and Intelligent Paradigms

Volume 1: Clustering, Association
and Classification

Dawn E. Holmes and Lakhmi C. Jain (Eds.)

Data Mining: Foundations and Intelligent Paradigms

Intelligent Systems Reference Library, Volume 23

Editors-in-Chief

Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Prof. Lakhmi C. Jain
University of South Australia
Adelaide
Mawson Lakes Campus
South Australia 5095
Australia
E-mail: Lakhmi.jain@unisa.edu.au

Further volumes of this series can be found on our homepage:
springer.com

Vol. 1. Christine L. Mumford and Lakhmi C. Jain (Eds.)
Computational Intelligence: Collaboration, Fusion and Emergence, 2009
ISBN 978-3-642-01798-8

Vol. 2. Yuehui Chen and Ajith Abraham
Tree-Structure Based Hybrid Computational Intelligence, 2009
ISBN 978-3-642-04738-1

Vol. 3. Anthony Finn and Steve Scheding
Developments and Challenges for Autonomous Unmanned Vehicles, 2010
ISBN 978-3-642-10703-0

Vol. 4. Lakhmi C. Jain and Chee Peng Lim (Eds.)
Handbook on Decision Making: Techniques and Applications, 2010
ISBN 978-3-642-13638-2

Vol. 5. George A. Anastassiou
Intelligent Mathematics: Computational Analysis, 2010
ISBN 978-3-642-17097-3

Vol. 6. Ludmila Dymowa
Soft Computing in Economics and Finance, 2011
ISBN 978-3-642-17718-7

Vol. 7. Gerasimos G. Rigatos
Modelling and Control for Intelligent Industrial Systems, 2011
ISBN 978-3-642-17874-0

Vol. 8. Edward H.Y. Lim, James N.K. Liu, and Raymond S.T. Lee
Knowledge Seeker – Ontology Modelling for Information Search and Management, 2011
ISBN 978-3-642-17915-0

Vol. 9. Menahem Friedman and Abraham Kandel
Calculus Light, 2011
ISBN 978-3-642-17847-4

Vol. 10. Andreas Tolk and Lakhmi C. Jain
Intelligence-Based Systems Engineering, 2011
ISBN 978-3-642-17930-3

Vol. 11. Samuli Niiranen and Andre Ribeiro (Eds.)
Information Processing and Biological Systems, 2011
ISBN 978-3-642-19620-1

Vol. 12. Florin Gorunescu
Data Mining, 2011
ISBN 978-3-642-19720-8

Vol. 13. Witold Pedrycz and Shyi-Ming Chen (Eds.)
Granular Computing and Intelligent Systems, 2011
ISBN 978-3-642-19819-9

Vol. 14. George A. Anastassiou and Oktay Duman
Towards Intelligent Modeling: Statistical Approximation Theory, 2011
ISBN 978-3-642-19825-0

Vol. 15. Antonino Freno and Edmondo Trentin
Hybrid Random Fields, 2011
ISBN 978-3-642-20307-7

Vol. 16. Alexiei Dingli
Knowledge Annotation: Making Implicit Knowledge Explicit, 2011
ISBN 978-3-642-20322-0

Vol. 17. Crina Grosan and Ajith Abraham
Intelligent Systems, 2011
ISBN 978-3-642-21003-7

Vol. 18. Achim Zielesny
From Curve Fitting to Machine Learning, 2011
ISBN 978-3-642-21279-6

Vol. 19. George A. Anastassiou
Intelligent Systems: Approximation by Artificial Neural Networks, 2011
ISBN 978-3-642-21430-1

Vol. 20. Lech Polkowski
Approximate Reasoning by Parts, 2011
ISBN 978-3-642-22278-8

Vol. 21. Igor Chikalov
Average Time Complexity of Decision Trees, 2011
ISBN 978-3-642-22660-1
ław Różewski,

Vol. 22. Ruzetta, Ryszard Tadeusiewicz, and Oleg Zaikin
Intelligent Open Learning Systems, 2011
ISBN 978-3-642-22666-3

Vol. 23. Dawn E. Holmes and Lakhmi C. Jain (Eds.)
Data Mining: Foundations and Intelligent Paradigms, 2012
ISBN 978-3-642-23165-0

Dawn E. Holmes and Lakhmi C. Jain (Eds.)

Data Mining: Foundations and Intelligent Paradigms

Volume 1: Clustering, Association and Classification

Prof. Dawn E. Holmes
Department of Statistics and Applied Probability
University of California
Santa Barbara,
CA 93106
USA
E-mail: holmes@pstat.ucsb.edu

Prof. Lakhmi C. Jain
Professor of Knowledge-Based Engineering
University of South Australia
Adelaide
Mawson Lakes, SA 5095
Australia
E-mail: Lakhmi.jain@unisa.edu.au

ISBN 978-3-642-23165-0

e-ISBN 978-3-642-23166-7

DOI 10.1007/978-3-642-23166-7

Intelligent Systems Reference Library

ISSN 1868-4394

Library of Congress Control Number: 2011936705

© 2012 Springer-Verlag Berlin Heidelberg

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, reuse of illustrations, recitation, broadcasting, reproduction on microfilm 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.

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

There are many invaluable books available on data mining theory and applications. However, in compiling a volume titled “DATA MINING: Foundations and Intelligent Paradigms: Volume 1: Clustering, Association and Classification” we wish to introduce some of the latest developments to a broad audience of both specialists and non-specialists in this field.

The term ‘data mining’ was introduced in the 1990’s to describe an emerging field based on classical statistics, artificial intelligence and machine learning. Clustering, a method of unsupervised learning, has applications in many areas. Association rule learning, became widely used following the seminal paper by Agrawal, Imielinski and Swami; “Mining Association Rules Between Sets of Items in Large Databases”, SIGMOD Conference 1993: 207-216. Classification is also an important technique in data mining, particularly when it is known in advance how classes are to be defined.

In compiling this volume we have sought to present innovative research from prestigious contributors in these particular areas of data mining. Each chapter is self-contained and is described briefly in Chapter 1.

This book will prove valuable to theoreticians as well as application scientists/engineers in the area of Data Mining. Postgraduate students will also find this a useful sourcebook since it shows the direction of current research.

We have been fortunate in attracting top class researchers as contributors and wish to offer our thanks for their support in this project. We also acknowledge the expertise and time of the reviewers. We thank Professor Dr. Osmar Zaiane for his visionary Foreword. Finally, we also wish to thank Springer for their support.

Dr. Dawn E. Holmes
University of California
Santa Barbara, USA

Dr. Lakhmi C. Jain
University of South Australia
Adelaide, Australia

Contents

Chapter 1

Data Mining Techniques in Clustering, Association and Classification	1
Dawn E. Holmes, Jeffrey Tweedale, Lakhmi C. Jain	
1 Introduction	1
1.1 Data	1
1.2 Knowledge	2
1.3 Clustering	2
1.4 Association	3
1.5 Classification	3
2 Data Mining	4
2.1 Methods and Algorithms	4
2.2 Applications	4
3 Chapters Included in the Book	5
4 Conclusion	5
References	6

Chapter 2

Clustering Analysis in Large Graphs with Rich Attributes	7
Yang Zhou, Ling Liu	
1 Introduction	8
2 General Issues in Graph Clustering	11
2.1 Graph Partition Techniques	12
2.2 Basic Preparation for Graph Clustering	14
2.3 Graph Clustering with SA-Cluster	15
3 Graph Clustering Based on Structural/Attribute Similarities	16
4 The Incremental Algorithm	19
5 Optimization Techniques	21
5.1 The Storage Cost and Optimization	22
5.2 Matrix Computation Optimization	23
5.3 Parallelism	24
6 Conclusion	24
References	25

Chapter 3

Temporal Data Mining: Similarity-Profiled Association Pattern	29
Jin Soung Yoo	
1 Introduction	29
2 Similarity-Profiled Temporal Association Pattern	32
2.1 Problem Statement	32
2.2 Interest Measure	34
3 Mining Algorithm	35
3.1 Envelope of Support Time Sequence	35
3.2 Lower Bounding Distance	36
3.3 Monotonicity Property of Upper Lower-Bounding Distance	38
3.4 SPAMINE Algorithm	39
4 Experimental Evaluation	41
5 Related Work	43
6 Conclusion	45
References	45

Chapter 4

Bayesian Networks with Imprecise Probabilities: Theory and Application to Classification	49
G. Corani, A. Antonucci, M. Zaffalon	
1 Introduction	49
2 Bayesian Networks	51
3 Credal Sets	52
3.1 Definition	53
3.2 Basic Operations with Credal Sets	53
3.3 Credal Sets from Probability Intervals	55
3.4 Learning Credal Sets from Data	55
4 Credal Networks	56
4.1 Credal Network Definition and Strong Extension	56
4.2 Non-separately Specified Credal Networks	57
5 Computing with Credal Networks	60
5.1 Credal Networks Updating	60
5.2 Algorithms for Credal Networks Updating	61
5.3 Modelling and Updating with Missing Data	62
6 An Application: Assessing Environmental Risk by Credal Networks	64
6.1 Debris Flows	64
6.2 The Credal Network	65
7 Credal Classifiers	70
8 Naive Bayes	71
8.1 Mathematical Derivation	73
9 Naive Credal Classifier (NCC)	74

9.1	Comparing NBC and NCC in Texture Recognition	76
9.2	Treatment of Missing Data	79
10	Metrics for Credal Classifiers.....	80
11	Tree-Augmented Naive Bayes (TAN).....	81
11.1	Variants of the Imprecise Dirichlet Model: Local and Global IDM	82
12	Credal TAN	83
13	Further Credal Classifiers.....	85
13.1	Lazy NCC (LNCC)	85
13.2	Credal Model Averaging (CMA)	86
14	Open Source Software	88
15	Conclusions	88
	References	88

Chapter 5

Hierarchical Clustering for Finding Symmetries and Other Patterns in Massive, High Dimensional Datasets	95
Fionn Murtagh, Pedro Contreras	

1	Introduction: Hierarchy and Other Symmetries in Data Analysis	95
1.1	About This Article	96
1.2	A Brief Introduction to Hierarchical Clustering.....	96
1.3	A Brief Introduction to p-Adic Numbers	97
1.4	Brief Discussion of p-Adic and m-Adic Numbers.....	98
2	Ultrametric Topology	98
2.1	Ultrametric Space for Representing Hierarchy	98
2.2	Some Geometrical Properties of Ultrametric Spaces.....	100
2.3	Ultrametric Matrices and Their Properties.....	100
2.4	Clustering through Matrix Row and Column Permutation	101
2.5	Other Miscellaneous Symmetries	103
3	Generalized Ultrametric	103
3.1	Link with Formal Concept Analysis.....	103
3.2	Applications of Generalized Ultrametrics	104
3.3	Example of Application: Chemical Database Matching	105
4	Hierarchy in a p-Adic Number System	110
4.1	p-Adic Encoding of a Dendrogram	110
4.2	p-Adic Distance on a Dendrogram	113
4.3	Scale-Related Symmetry	114
5	Tree Symmetries through the Wreath Product Group	114
5.1	Wreath Product Group Corresponding to a Hierarchical Clustering	115
5.2	Wreath Product Invariance	115

5.3	Example of Wreath Product Invariance: Haar Wavelet Transform of a Dendrogram	116
6	Remarkable Symmetries in Very High Dimensional Spaces	118
6.1	Application to Very High Frequency Data Analysis: Segmenting a Financial Signal	119
7	Conclusions	126
	References	126

Chapter 6

Randomized Algorithm of Finding the True Number of Clusters Based on Chebychev Polynomial Approximation	131	
R. Avros, O. Granichin, D. Shalymov, Z. Volkovich, G.-W. Weber		
1	Introduction	131
2	Clustering	135
2.1	Clustering Methods	135
2.2	Stability Based Methods	138
2.3	Geometrical Cluster Validation Criteria	141
3	Randomized Algorithm	144
4	Examples	147
5	Conclusion	152
	References	152

Chapter 7

Bregman Bubble Clustering: A Robust Framework for Mining Dense Clusters	157	
Joydeep Ghosh, Gunjan Gupta		
1	Introduction	157
2	Background	161
2.1	Partitional Clustering Using Bregman Divergences	161
2.2	Density-Based and Mode Seeking Approaches to Clustering	162
2.3	Iterative Relocation Algorithms for Finding a Single Dense Region	164
2.4	Clustering a Subset of Data into Multiple Overlapping Clusters	165
3	Bregman Bubble Clustering	165
3.1	Cost Function	165
3.2	Problem Definition	166
3.3	Bregmanian Balls and Bregman Bubbles	166
3.4	BBC-S: Bregman Bubble Clustering with Fixed Clustering Size	167
3.5	BBC-Q: Dual Formulation of Bregman Bubble Clustering with Fixed Cost	169

4	Soft Bregman Bubble Clustering (Soft BBC)	169
4.1	Bregman Soft Clustering	169
4.2	Motivations for Developing Soft BBC	170
4.3	Generative Model	171
4.4	Soft BBC EM Algorithm	171
4.5	Choosing an Appropriate p_0	173
5	Improving Local Search: Pressurization	174
5.1	Bregman Bubble Pressure	174
5.2	Motivation	175
5.3	BBC-Press	176
5.4	Soft BBC-Press	177
5.5	Pressurization vs. Deterministic Annealing	177
6	A Unified Framework	177
6.1	Unifying Soft Bregman Bubble and Bregman Bubble Clustering	177
6.2	Other Unifications	178
7	Example: Bregman Bubble Clustering with Gaussians	180
7.1	σ^2 Is Fixed	180
7.2	σ^2 Is Optimized	181
7.3	“Flavors” of BBC for Gaussians	182
7.4	Mixture-6: An Alternative to BBC Using a Gaussian Background	182
8	Extending BBOCC & BBC to Pearson Distance and Cosine Similarity	183
8.1	Pearson Correlation and Pearson Distance	183
8.2	Extension to Cosine Similarity	185
8.3	Pearson Distance vs. (1-Cosine Similarity) vs. Other Bregman Divergences – Which One to Use Where?	185
9	Seeding BBC and Determining k Using Density Gradient Enumeration (DGRADE)	185
9.1	Background	186
9.2	DGRADE Algorithm	186
9.3	Selecting s_{one} : The Smoothing Parameter for DGRADE	188
10	Experiments	190
10.1	Overview	190
10.2	Datasets	190
10.3	Evaluation Methodology	192
10.4	Results for BBC with Pressurization	194
10.5	Results on BBC with DGRADE	198
11	Concluding Remarks	202
	References	204

Chapter 8

DepMiner: A Method and a System for the Extraction of Significant Dependencies	209
Rosa Meo, Leonardo D'Ambrosi	
1 Introduction	209
2 Related Work	211
3 Estimation of the Referential Probability	213
4 Setting a Threshold for Δ	213
5 Embedding Δ_n in Algorithms	215
6 Determination of the Itemsets Minimum Support Threshold	216
7 System Description	218
8 Experimental Evaluation	220
9 Conclusions	221
References	221

Chapter 9

Integration of Dataset Scans in Processing Sets of Frequent Itemset Queries	223
Marek Wojciechowski, Maciej Zakrzewicz, Paweł Boinski	
1 Introduction	223
2 Frequent Itemset Mining and Apriori Algorithm	225
2.1 Basic Definitions and Problem Statement	225
2.2 Algorithm Apriori	226
3 Frequent Itemset Queries – State of the Art	227
3.1 Frequent Itemset Queries	227
3.2 Constraint-Based Frequent Itemset Mining	229
3.3 Reusing Results of Previous Frequent Itemset Queries	230
4 Optimizing Sets of Frequent Itemset Queries	231
4.1 Basic Definitions	232
4.2 Problem Formulation	233
4.3 Related Work on Multi-query Optimization	234
5 Common Counting	234
5.1 Basic Algorithm	234
5.2 Motivation for Query Set Partitioning	237
5.3 Key Issues Regarding Query Set Partitioning	237
6 Frequent Itemset Query Set Partitioning by Hypergraph Partitioning	238
6.1 Data Sharing Hypergraph	239
6.2 Hypergraph Partitioning Problem Formulation	239
6.3 Computation Complexity of the Problem	241
6.4 Related Work on Hypergraph Partitioning	241
7 Query Set Partitioning Algorithms	241
7.1 CCRecursive	242
7.2 CCFull	243
7.3 CCCoarsening	246

7.4	CCAgglomerative	247
7.5	CCAgglomerativeNoise	248
7.6	CCGreedy	249
7.7	CCSemiGreedy	250
8	Experimental Results	251
8.1	Comparison of Basic Dedicated Algorithms	252
8.2	Comparison of Greedy Approaches with the Best Dedicated Algorithms	257
9	Review of Other Methods of Processing Sets of Frequent Itemset Queries	260
10	Conclusions	261
	References	262

Chapter 10

	Text Clustering with Named Entities: A Model, Experimentation and Realization	267
	Tru H. Cao, Thao M. Tang, Cuong K. Chau	
1	Introduction	267
2	An Entity-Keyword Multi-Vector Space Model	269
3	Measures of Clustering Quality	271
4	Hard Clustering Experiments	273
5	Fuzzy Clustering Experiments	277
6	Text Clustering in VN-KIM Search	282
7	Conclusion	285
	References	286

Chapter 11

	Regional Association Rule Mining and Scoping from Spatial Data	289
	Wei Ding, Christoph F. Eick	
1	Introduction	289
2	Related Work	291
2.1	Hot-Spot Discovery	291
2.2	Spatial Association Rule Mining	292
3	The Framework for Regional Association Rule Mining and Scoping	293
3.1	Region Discovery	293
3.2	Problem Formulation	294
3.3	Measure of Interestingness	295
4	Algorithms	298
4.1	Region Discovery	298
4.2	Generation of Regional Association Rules	301

5	Arsenic Regional Association Rule Mining and Scoping in the Texas Water Supply	302
5.1	Data Collection and Data Preprocessing	302
5.2	Region Discovery for Arsenic Hot/Cold Spots	304
5.3	Regional Association Rule Mining	305
5.4	Region Discovery for Regional Association Rule Scoping	307
6	Summary	310
	References	311

Chapter 12

Learning from Imbalanced Data: Evaluation Matters	315	
Troy Raeder, George Forman, Nitesh V. Chawla		
1	Motivation and Significance	315
2	Prior Work and Limitations	317
3	Experiments	318
3.1	Datasets	321
3.2	Empirical Analysis	321
4	Discussion and Recommendations	325
4.1	Comparisons of Classifiers	325
4.2	Towards Parts-Per-Million	328
4.3	Recommendations	329
5	Summary	329
	References	330
Author Index	333	

Editors



Dr. Dawn E. Holmes serves as Senior Lecturer in the Department of Statistics and Applied Probability and Senior Associate Dean in the Division of Undergraduate Education at UCSB. Her main research area, Bayesian Networks with Maximum Entropy, has resulted in numerous journal articles and conference presentations. Her other research interests include Machine Learning, Data Mining, Foundations of Bayesianism and Intuitionistic Mathematics. Dr. Holmes has co-edited, with Professor Lakhmi C. Jain, volumes 'Innovations in Bayesian Networks' and 'Innovations in Machine Learning'. Dr. Holmes teaches a broad range of courses, including SAS programming, Bayesian Networks and Data Mining. She was awarded the Distinguished Teaching Award by Academic Senate, UCSB in 2008.

As well as being Associate Editor of the International Journal of Knowledge-Based and Intelligent Information Systems, Dr. Holmes reviews extensively and is on the editorial board of several journals, including the Journal of Neurocomputing. She serves as Program Scientific Committee Member for numerous conferences; including the International Conference on Artificial Intelligence and the International Conference on Machine Learning. In 2009 Dr. Holmes accepted an invitation to join Center for Research in Financial Mathematics and Statistics (CRFMS), UCSB. She was made a Senior Member of the IEEE in 2011.



Professor Lakhmi C. Jain is a Director/Founder of the Knowledge-Based Intelligent Engineering Systems (KES) Centre, located in the University of South Australia. He is a fellow of the Institution of Engineers Australia.

His interests focus on the artificial intelligence paradigms and their applications in complex systems, art-science fusion, e-education, e-healthcare, unmanned air vehicles and intelligent agents.

Chapter 1

Data Mining Techniques in Clustering, Association and Classification

Dawn E. Holmes¹, Jeffrey Tweedale², and Lakhmi C. Jain³

¹ Department of Statistics and Applied Probability
University of California Santa Barbara
Santa Barbara
CA 93106-3110
USA

² School of Electrical and Information Engineering
University of South Australia
Adelaide
Mawson Lakes Campus
South Australia SA 5095
Australia

³ School of Electrical and Information Engineering
University of South Australia
Adelaide
Mawson Lakes Campus
South Australia SA 5095
Australia

1 Introduction

The term *Data Mining* grew from the relentless growth of techniques used to interrogate masses of data. As a myriad of databases emanated from disparate industries, management insisted their information officers develop methodology to exploit the knowledge held in their repositories. The process of extracting this knowledge evolved as an interdisciplinary field of computer science within academia. This included study into statistics, database management and Artificial Intelligence (AI). Science and technology provide the stimulus for an extremely rapid transformation from data acquisition to enterprise knowledge management systems.

1.1 Data

Data is the representation of anything that can be meaningfully quantized or represented in digital form, as a number, symbol or even text. We process data into information by initially combining a collection of artefacts that are input into a system which is generally stored, filtered and/or classified prior to being translated into a useful form for dissemination [1]. The processes used to achieve this task have evolved over many years and has been applied to many situations using a magnitude of techniques. Accounting and pay role applications take center place in the evolution of information processing.

Data mining, expert system and knowledge-based system quickly followed. Today we live in an information age where we collect data faster than it can be processed. This book examines many recent advances in digital information processing with paradigms for acquisition, retrieval, aggregation, search, estimation and presentation.

Our ability to acquire data electronically has grown exponentially since the introduction of mainframe computers. We have also improved the methodology used to extract information from data in almost every aspect of life. Our biggest challenge is in identifying targeted information and transforming that into useful knowledge within the growing collection of noise collected in repositories all over the world.

1.2 Knowledge

Information, knowledge and wisdom are labels commonly applied to the way humans aggregate practical experience into an organised collection of facts. Knowledge is considered a collection of facts, truths, or principles resulting from a study or investigation. The concept of knowledge is a collection of facts, principles, and related concepts. Knowledge representation is the key to any communication language and a fundamental issue in AI. The way knowledge is represented and expressed has to be meaningful so that the communicating entities can grasp the concept of the knowledge transmitted among them. This requires a good technique to represent knowledge. In computers symbols (numbers and characters) are used to store and manipulate the knowledge. There are different approaches for storing the knowledge because there are different kinds of knowledge such as facts, rules, relationships, and so on. Some popular approaches for storing knowledge in computers include procedural, relational, and hierarchical representations. Other forms of knowledge representation used include *Predicate Logic*, *Frames*, *Semantic Nets*, *If-Then rules* and *Knowledge Inter-change Format*. The type of knowledge representation to be used depends on the AI application and the domain that Intelligent Agents (IAs) are required to function [2]. Knowledge should be separated from the procedural algorithms in order to simplify knowledge modification and processing. For an IA to be capable of solving problems at different levels of abstraction, knowledge should be presented in the form of frames or semantic nets that can show the *is-a* relationship of objects and concepts. If an IA is required to find the solution from the existing data, Predicate logic using IF-THEN rules, Bayesian or any number of techniques can be used to cluster information [3].

1.3 Clustering

In data mining a cluster is the resulting collection of similar or same items from a volume of acquired facts. Each cluster has distinct characteristics, although each has a similarity, its size is measured from the centre with a distance or separation from the next [4]. Non-hierarchical clusters are generally partitioned by class or clumping methods. Hierarchical clusters produce sets of nested groups that need to be progressively isolated as individual subsets. The methodology used are described as: partitioning, hierarchical agglomeration, Single Link (SLINK), Complete Link (CLINK), group average and text based document methods. Other techniques include [5]:

- A Comparison of Techniques,
- Artificial Neural Networks for Clustering, and
- Clustering Large Data Sets, and
- Evolutionary Approaches for Clustering, and
- Fuzzy Clustering, and
- Hierarchical Clustering Algorithms, and
- Incorporating Domain Constraints in Clustering, and
- Mixture-Resolving and Mode-Seeking Algorithms, and
- Nearest Neighbour Clustering, and
- Partitional Algorithms, and
- Representation of Clusters, and
- Search-Based Approaches.

Where clustering can typically be applied in Image Segmentation, Object/Character Recognition, Information Retrieval and Data Mining.

1.4 Association

Data is merely a collection of facts. To make sense of that collection, a series of rules can be created to sort, select and match a pattern of behavior or association based on specified dependancies or relationships. For instance a collection of sales transaction within a department store can hold a significant volume of information. If a cosmetics manager desired to improve sales, knowledge about existing turnover provides an excellent base-line (this is a form of market analysis). Similarly, using the same data set, the logistics manager could determine inventory levels (this concept is currently associated with trend analysis and prediction). Association rules allow the user to reveal sequences, links and unique manifestations that emerge over time [6]. Typically cross-tabulation can be used where items, words or conjunctions are employed to analyse simple collections that are easily classified, such as age, cost or gender.

1.5 Classification

Data bases provide an arbitrary collection of facts. In order to make sense of the random nature of such collections, any number of methods can be used to map the data into usable or quantifiable categories based on a series of attributes. These subsets improve efficiency by reducing the noise and volume of data during subsequent processing. The goal is to predict the target class for each case. An example would be to measure the risk management of an activity, as either low, high or some category in between. Prior to classification, the target categories must be defined before the process is run [7]. A number of AI techniques are used to classify data. Some include decision-trees, rule-based, Bayesian, rough sets, dependency networks, Support Vector Machines (SVM), Neural Networkss (NNs), genetic algorithms and fuzzy logic.

2 Data Mining

There are many commercial data mining methods, algorithms and applications, with several that have had major impact. Examples include: *SAS*¹, *SPSS*² and *Statistica*³. Other examples are listed in sections 2.1 and 2.2. Any number can be found on-line, and many are free. Examples include: *Environment for DeveLoping KDD-Applications Sup-ported by Index-Structures (ELKI)*⁴, *General Architecture for Text Engineering (GATE)*⁵ and *Waikato Environment for Knowledge Analysis (Weka)*⁶.

2.1 Methods and Algorithms

- Association rule learning,
- Cluster analysis, and
- Constructive induction, and
- Data analysis, and
- Decision trees, and
- Factor analysis, and
- Knowledge discovery, and
- Neural nets, and
- Predictive analytics, and
- Reactive business intelligence, and
- Regression, and
- Statistical data analysis, and
- Text mining.

2.2 Applications

- Customer analytics,
- Data Mining in Agriculture, and
- Data mining in Meteorology, and
- Law-enforcement, and
- National Security Agency, and
- Quantitative structure-activity relationship, and
- Surveillance.

¹ See <http://www.sas.com/>

² See <http://www.spss.com/>

³ See <http://www.statsoft.com/>

⁴ See <http://www.dbs.ifi.lmu.de/research/KDD/ELKI> from Ludwig Maximilian University.

⁵ See gate.ac.uk from the University of Sheffield.

⁶ See <http://www.cs.waikato.ac.nz/~ml/weka/> from the University of Waikato.

3 Chapters Included in the Book

This book includes twelve chapters. Each chapter is described briefly below. Chapter 1 provides an introduction to data mining and presents a brief abstract of each chapter included in the book. Chapter 2 is on clustering analysis in large graphs with rich attributes. The authors state that a key challenge for addressing the problem of clustering large graphs with rich attributes is to achieve a good balance between structural and attribute similarities. Chapter 3 is on temporal data mining. A temporal association mining problem, based on similarity constraint, is presented. Chapter 4 is on Bayesian networks with imprecise probabilities. The authors report extensive experimentation on public benchmark data sets in real-world applications to show that on the instances indeterminately classified by a credal network, the accuracy of its Bayesian counterpart drops.

Chapter 5 is on hierarchical clustering for finding symmetries and other patterns in massive, high dimensional datasets. The authors have illustrated the powerfulness of hierarchical clustering in case studies in chemistry and finance. Chapter 6 is on randomized algorithm of finding the true number of clusters based on Chebychev polynomial approximation. A number of examples are used to validate the proposed algorithm. Chapter 7 is on Bregman bubble clustering. The authors present a broad framework for finding k dense clusters while ignoring rest of the data. The results are validated on various datasets to demonstrate the relevance and effectiveness of the technique.

Chapter 8 is on *DepMiner*. It is a method for implementing a model for the evaluation of item-sets, and in general for the evaluation of the dependencies between the values assumed by a set of variables on a domain of finite values. Chapter 9 is on the integration of dataset scans in processing sets of frequent item-set queries. Chapter 10 is on text clustering with named entities. It is demonstrated that a weighted combination of named entities and keywords are significant to clustering quality. The authors present implementation of the scheme and demonstrate the text clustering with named entities in a semantic search engine.

Chapter 11 is on learning from imbalanced data. Using experimentations, the authors made some recommendations related to the data evaluation methods. Finally Chapter 12 is on regional association rule mining and scoping from spatial data. The authors have investigated the duality between regional association rules and regions where the associations are valid. The design and implementation of a reward-based region discovery framework and its evaluation are presented.

4 Conclusion

This chapter presents a collection of selected contribution of leading subject matter experts in the field of data mining. This book is intended for students, professionals and academics from all disciplines to enable them the opportunity to engage in the state of art developments in:

- Clustering Analysis in Large Graphs with Rich Attributes;
- Temporal Data Mining: Similarity-Profiled Association Pattern;
- Bayesian Networks with Imprecise Probabilities: Theory and Application to Classification;
- Hierarchical Clustering for Finding Symmetries and Other Patterns in Massive, High Dimensional Datasets;
- Randomized Algorithm of Finding the True Number of Clusters Based on Chebychev Polynomial Approximation;
- Bregman Bubble Clustering: A Robust Framework for Mining Dense Clusters;
- DepMiner: A method and a system for the extraction of significant dependencies;
- Integration of Dataset Scans in Processing Sets of Frequent Itemset Queries;
- Text Clustering with Named Entities: A Model, Experimentation and Realization;
- Regional Association Rule Mining and Scoping from Spatial Data; and
- Learning from Imbalanced Data: Evaluation Matters.

Readers are invited to contact individual authors to engage with further discussion or dialog on each topic.

References

1. Moxon, B.: Defining data mining, the hows and whys of data mining, and how it differs from other analytical techniques. *DBMS* 9(9), S11–S14 (1996)
2. Bigus, J.P., Bigus, J.: Constructing Intelligent Agents Using Java. Professional Developer's Guide Series. John Wiley & Sons, Inc., New York (2001)
3. Tweedale, J., Jain, L.C.: Advances in information processing paradigms. In: Watanabe, T. (ed.) *Innovations in Intelligent Machines*, pp. 1–19. Springer, Heidelberg (2011)
4. Bouguettaya, A.: On-line clustering. *IEEE Trans. on Knowl. and Data Eng.* 8, 333–339 (1996)
5. Jain, A., Murty, M., Flynn, P.: Data clustering: A review. *ACM Computing Surveys* 3(3), 264–323 (1999)
6. Hill, T., Lewicki, P.: *Statistics: Methods and Applications*, StatSoft, Tulsa, OK (2007)
7. Classification, clustering, and data mining applications. In: Banks, D., House, L., Morris, F., Arabie, P., Gaul, W. (eds.) *International Federation of Classification Societies (IFCS)*, Illinois Institute of Technology, Chicago, p. 658. Springer, New York (2004)

Chapter 2

Clustering Analysis in Large Graphs with Rich Attributes

Yang Zhou and Ling Liu

DiSL, College of Computing, Georgia Institute of Technology,
Atlanta, Georgia, USA

Abstract. Social networks, communication networks, biological networks and many other information networks can be modeled as a large graph. Graph vertices represent entities and graph edges represent the relationships or interactions among entities. In many large graphs, there is usually one or more attributes associated with every graph vertex to describe its properties. The goal of graph clustering is to partition vertices in a large graph into subgraphs (clusters) based on a set of criteria, such as vertex similarity measures, adjacency-based measures, connectivity-based measures, density measures, or cut-based measures. Although graph clustering has been studied extensively, the problem of clustering analysis of large graphs with rich attributes remains a big challenge in practice. In this chapter we first give an overview of the set of issues and challenges for clustering analysis of large graphs with vertices of rich attributes. Based on the type of measures used for identifying clusters, existing graph clustering methods can be categorized into three classes: structure based clustering, attribute based clustering and structure-attribute based clustering. Structure based clustering mainly focuses on the topological structure of a graph for clustering, but largely ignore the vertex properties which are often heterogenous. Attribute based clustering, in contrast, focuses primarily on attribute-based vertex similarity, but suffers from isolated partitions of the graph as a result of graph clustering. Structure-attribute based clustering is a hybrid approach, which combines structural and attribute similarities through a unified distance measure. We argue that effective clustering analysis of a large graph with rich attributes requires the clustering methods to provide a systematic graph analysis framework that partition the graph based on both structural similarity and attribute similarity. One approach is to model rich attributes of vertices as auxiliary edges among vertices, resulting in a complex attribute augmented graph with multiple edges between some vertices. To show how to best combine structure and attribute similarity in a unified framework, the second part of this chapter will outline a cluster-convergence based iterative edge-weight assignment scheme that assigns different weights to different attributes based on how fast the clusters converge. We use a K-Medoids clustering algorithm to partition a graph into k clusters with both cohesive intra-cluster structures and homogeneous attribute values based on iterative weight updates. At each iteration, a series of matrix multiplication operations is used for calculating the random walk distances between graph

vertices. Optimizations are used to reduce the cost of recalculating the random walk distances upon each iteration of the edge weight update. Finally, we discuss the set of open problems in graph clustering with rich attributes, including storage cost and efficiency, scalable analytics under memory constraints, distributed graph clustering and parallel processing.

1 Introduction

A number of scientific and technical endeavors are generating data that usually consists of a large number of interacting physical, conceptual, and societal components. Such examples include social networks, semantic networks, communication systems, the Internet, ecological networks, transportation networks, database schemas and ontologies, electrical power grids, sensor networks, research coauthor networks, biological networks, and so on. All the above networks share an important common feature: they can be modeled as graphs, i.e., individual objects interact with one another, forming large, interconnected, and sophisticated graphs with vertices of rich attributes. Multi-relational data mining finds the relational patterns in both the entity attributes and relations in the data. Graph mining, as one approach of multi-relational data mining, finds relational patterns in complex graph structures. Mining and analysis of these annotated and probabilistic graph structures is crucial for advancing the state of scientific research, accurate modeling and analysis of existing systems, and engineering of new systems.

Graph clustering is one of the most popular graph mining methodologies. Clustering is a useful and important unsupervised learning technique widely studied in literature [1,2,3,4]. The general goal of clustering is to group similar objects into one cluster while partitioning dissimilar objects into different clusters. Clustering has broad applications in the analysis of business and financial data, biological data, time series data, spatial data, trajectory data and so on. As one important approach of graph mining, graph clustering is an interesting and challenging research problem which has received much attention recently [5,6,7,8]. Clustering on a large graph aims to partition the graph into several densely connected components. This is very useful for understanding and visualizing large graphs. Typical applications of graph clustering include community detection in social networks, reduction of very large transportation networks, identification of functional related protein modules in large protein-protein interaction networks, *etc.* Although many graph clustering techniques have been proposed in literature, the problem of clustering analysis in large graphs with rich attributes remains to be challenging due to the demand on memory and computational resources and the demand on fast access to disk-based storage. Furthermore, with the grand vision of utility-driven and pay-as-you-go cloud computing paradigm shift, there is a growing demand for providing graph-clustering as a service. We witness the emerging interests from science and engineering fields in design and development of efficient and scalable graph analytics for managing and mining large information graphs.

Applications of graph clustering

In almost all information networks, graph clustering is used as a tool for analysis, modeling and prediction of the function, usage and evolution of the network, including business analysis, marketing, and anomaly detection. It is widely recognized by many that the task of graph clustering is highly application specific. In addition, by treating n -dimensional datasets as points in n -dimensional space, one can transform such n -dimensional datasets into graphs with rich attributes and apply graph theory to analyze the datasets. For example, modeling the World Wide Web (the Web) as a graph by representing each web page by a vertex and each hyperlink by an edge enables us to perform graph clustering analysis of hypertext documents and identify interesting artifacts about the Web, and visualize the usage and function of the Web. Furthermore, by representing each user as a vertex and placing (weighted) edges between two users as they communicate over the Internet services such as Skype, Microsoft's Messenger Live, and twitter, one can perform interesting usage statistics for optimizing related software and hardware configurations.

Concretely, in computer networks, clustering can be used to identify relevant substructures, analyze the connectivity for modeling or structural optimization, and perform root cause analysis of network faults [9,10]. In tele-communication systems, savings could be obtained by grouping a dense cluster of users on the same server as it would reduce the inter-server traffic. Similar analysis can help traditional tele-operators offer more attractive service packages or improve call delivery efficiency by identifying “frequent call clusters”, i.e., groups of people that mainly call each other (such as families, coworkers, or groups of teenage friends) and hence better design and target the call service offers and special rates for calling to a limited set of pre-specified phone numbers. Clustering the caller information can also be used for fraud detection by identifying changes (outliers) in the communication pattern, call durations and a geographical embedding, in order to determine the cluster of “normal call destinations” for a specific client and which calls are “out of the ordinary”. For networks with a dynamic topology, with frequent changes in the edge structure, local clustering methods prove useful, as the network nodes can make local decisions on how to modify the clustering to better reflect the current network topology [11]. Imposing a cluster structure on a dynamic network eases the routing task [12]. In bioinformatics, graph clustering analysis can be applied to the classification of gene expression data (e.g., gene-activation dependencies), protein interactions, and epidemic spreading of diseases (e.g., identifying groups of individuals “exposed” to the influence of a certain individual of interest or locating potentially infected people when an infected and contagious individual is encountered). In fact, cluster analysis of a social network also helps to identify the formation of trends or communities (relevant to market studies) and social influence behavior.

Graph Clustering: State of Art and Open Issues

Graph clustering has been studied by both theoreticians and practitioners over the last decade. Theoreticians are interested in investigating cluster properties,

algorithms and quality measures by exploiting underlying mathematical structures formalized in graph theory. Practitioners are investigating graph clustering algorithms by exploiting known characteristics of application-specific datasets. However, there is little effort on bridging the gap between theoretical aspect and practical aspect in graph clustering.

The goal of graph clustering is to partition vertices in a large graph into subgraphs (clusters) based on a set of criteria, such as vertex similarity measures, adjacency-based measures, connectivity-based measures, density measures, or cut-based measures. Based on the type of measures used for identifying clusters, existing graph clustering methods can be categorized into three classes: structure based clustering, attribute based clustering and structure-attribute based clustering. Structure based clustering mainly focuses on the topological structure of a graph for clustering, but largely ignores the rich attributes of vertices. Attribute based clustering, in contrast, focuses primarily on attribute-based vertex similarity, but suffers from isolated partitions of the graph as a result of graph clustering. Structure-attribute clustering is a hybrid approach, which combines structural similarity and attribute similarity through a unified distance measure. Most of the graph clustering techniques proposed to date are mainly focused on the topological structures using various criteria, including normalized cut [5], modularity [6], structural density [7] or flows [8]. The clustering results usually contain densely connected subgraphs within clusters. However, such methods largely ignore vertex attributes in the clustering process. On the other hand, attribute similarity based clustering [13] partitions large graphs by grouping nodes based on user-selected attributes and relationships. Vertices in one group share the same attribute values and relate to vertices in another group through the same type of relationship. This method achieves homogeneous attribute values within clusters, but ignores the intra-cluster topological structures. As shown in our experiments [14,15], the generated partitions tend to have very low connectivity.

Other recent studies on graph clustering include the following. Sun et al. [16] proposed GraphScope which is able to discover communities in large and dynamic graphs, as well as to detect the changing time of communities. Sun et al. [17] proposed an algorithm, RankClus, which integrates clustering with ranking in large-scale information network analysis. The final results contain a set of clusters with a ranking of objects within each cluster. Navlakha et al. [18] proposed a graph summarization method using the MDL principle. Tsai and Chiu [19] developed a feature weight self-adjustment mechanism for K-Means clustering on relational datasets. In that study, finding feature weights is modeled as an optimization problem to simultaneously minimize the separations within clusters and maximize the separations between clusters. The adjustment margin of a feature weight is estimated by the importance of the feature in clustering. [20] proposed an algorithm for mining communities on heterogeneous social networks. A method was designed for learning an optimal linear combination of different relations to meet users' expectation.

The rest of this chapter is organized as follows. Section 2 describes the basic concepts and general issues in graph clustering. Section 3 introduces the preliminary concepts and formulates the clustering problem for attribute augmented graphs and our proposed approach SA-Cluster. Section 4 presents presents our proposed incremental algorithm Inc-Cluster. Section 5 discusses optimization techniques to further improve computational performance. Finally, Section 6 concludes the chapter.

2 General Issues in Graph Clustering

Although graph clustering has been studied extensively, the problem of clustering analysis of large graphs with rich attributes remains to be a big challenge in practice. We argue that effective clustering analysis of a large graph with rich attributes requires a systematic graph clustering analysis framework that partition the graph based on both structural similarity and attribute similarity. One approach is to model rich attributes of vertices as auxiliary edges among vertices, resulting in a complex attribute augmented graph with multiple edges between some vertices.

In this section, we first describe the problem with an example. Then we review the graph clustering techniques and basic steps to take for preparation of clustering. We end this section by introducing the approach to combine structure and attribute similarity in a unified framework, called SA-Cluster. We dedicate Section 3 to present the design of SA-Cluster approach. The main idea is to use a cluster-convergence based iterative edge-weight assignment technique, which assigns different weights to different attributes based on how fast the clusters converge. We use a K-Medoids clustering algorithm to partition a graph into k clusters with both cohesive intra-cluster structures and homogeneous attribute values by applying a series of matrix multiplication operations for calculating the random walk distances between graph vertices. Optimization techniques are developed to reduce the cost of recalculating the random walk distances upon an iteration of the edge weight update.

The general methodology of graph clustering makes the following hypothesis [21]: First, a graph consists of dense subgraphs such that a dense subgraph contains more well-connected internal edges connecting the vertices in the subgraph than cutting edges connecting the vertices across subgraphs. Second, a random walk that visits a subgraph will likely stay in the subgraph until many of its vertices have been visited. Third, among all shortest paths between all pairs of vertices, links between different dense subgraphs are likely to be in many shortest paths. We will briefly review the graph clustering techniques developed based on each of the hypothesis.

The graph clustering framework consists of four components: modeling, measure, algorithm, and evaluation. The modeling component deals with the problem of transforming data into a graph or modeling the real application as a graph. The measurement deals with both distance measure and quality measure, both of which implement an objective function that determines and rates the quality

of a clustering. The algorithm is to exactly or approximately optimize the quality measure of the graph clustering. The evaluation component involves a set of metrics used to evaluate the performance of clustering by comparing with a “ground truth” clustering.

An *attribute-augmented graph* is denoted as $G = (V, E, \Lambda)$, where V is the set of n vertices, E is the set of edges, and $\Lambda = \{a_1, \dots, a_m\}$ is the set of m attributes associated with vertices in V for describing vertex properties. Each vertex $v_i \in V$ is associated with an attribute vector $[a_1(v_i), \dots, a_m(v_i)]$ where $a_j(v_i)$ is the attribute value of vertex v_i on attribute a_j , and is taken from the attribute domain $dom(a_j)$. We denote the set of attribute values by V_a and $V_a = \{a_j(v_i) \in dom(a_j) | i = 1, \dots, n, j = 1, \dots, m\}$. The **graph partition** of G is to partition an attribute-augmented graph G into k disjoint subgraphs, denoted by $G_i = (V_i, E_i, \Lambda)$, where $V = \bigcup_{i=1}^k V_i$ and $V_i \cap V_j = \emptyset$ for any $i \neq j$.

Figure 1 shows an example of a coauthor attributed Graph [15] where a vertex represents an author and an edge represents the coauthor relationship between two authors. In addition to the author ID, each vertex also has two attributes, research topic and age, associated with each author as the vertex properties. As shown in Figure 1, authors r_1-r_7 work on *XML*, authors r_9-r_{11} work on *Skyline* and r_8 works on both. In addition, each author has a range value to describe his/her age.

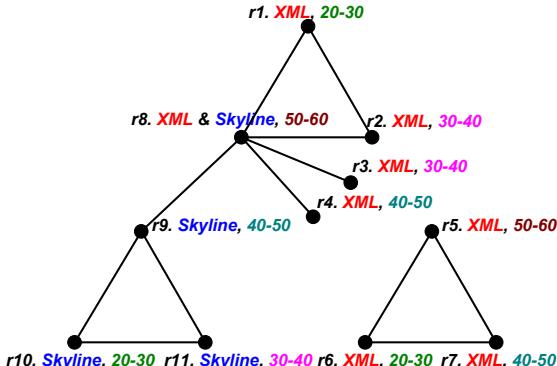


Fig. 1. A Coauthor Network with Two Attributes “Topic” and “Age” [15]

2.1 Graph Partition Techniques

Graph partition techniques refer to methods and algorithms that can partition a graph into densely connected subgraphs which are sparsely connected to each other. As we have discussed previously, there are three kinds of graph partition approaches: structure-similarity based graph clustering, attribute similarity based graph clustering, and structure-attribute combined graph clustering. Structure-based clustering only considers topological structure similarity but ignores the correlation of vertex attribute. Therefore, the clusters generated have a rather random distribution of vertex properties within clusters. On the

other hand, the attribute based clustering follows the grouping of compatible attributes and the clusters generated have good intra-cluster attribute similarity but a rather loose intra-cluster topological structure. A desired clustering of an attribute augmented graph should achieve a good balance between the following two properties: (1) vertices within one cluster are close to each other in terms of structure, while vertices between clusters are distant from each other; and (2) vertices within one cluster have similar attribute values, while vertices between clusters could have quite different attribute values. The structure-attribute combined graph clustering method aims at partitioning a graph with rich attributes into k clusters with cohesive intra-cluster structures and homogeneous attribute values.

Orthogonal to the structure and attribute similarity based classification of graph clustering algorithms, another way to categorize graph clustering algorithms is in terms of top-down or bottom up partitioning. There are two major classes of algorithms: divisive and agglomerative. Divisive clustering follows top-down style and recursively splits a graph into subgraphs. In contrast, agglomerative clustering works bottom-up and iteratively merges singleton sets of vertices into subgraphs. The divisive and agglomerative algorithms are also called hierarchical since they produce multi-level clusterings, i.e., one clustering follows the other by refining (divisive) or coarsening (agglomerative). Most graph clustering algorithms proposed to date are divisive, including cut-based, spectral clustering, random walks, and shortest path.

The cut-based algorithms are associated with max-flow min-cut theorem [22], which states that “the value of the maximum flow is equal to the cost of the minimum cut”. One of the earliest algorithms by Kernighan and Lin [23] splits the graph by performing recursive bisection (split into two parts at a time), aiming to minimize inter-cluster density (cut size). The high complexity of the algorithm ($O(|V|^3)$) makes it less competitive in real applications. An optimization is proposed by Flake et al. [24] to optimize the bicriterion measure and the complexity, resulting in a more practical cut-based algorithm that is proportional to the number of clusters K using a heuristic.

The spectral clustering algorithms are based on spectral graph theory with Laplacian matrix as the mathematical tool. The proposition that the multiplicity k of the eigenvalue 0 of L equals to the number of connected components in the graph is used to establish the connection between clustering and spectrum of Laplacian matrix (L). The main reason for spectral clustering is that it does not make strong assumptions on the form of the clusters and can solve very general problems like intertwined spirals which k-means clustering handles poorly. Unfortunately, spectral clustering could be unstable under different choices of graphs and parameters [25,26]. The running complexity of spectral clustering equals to the complexity of computing the eigenvectors of Laplacian matrix which is ($O(|V|^3)$).

The random walk based algorithms are based on the hypothesis that a random walk is likely to visit many vertices in a cluster before moving to the other cluster. The Markov clustering algorithm (MCL) by Van Dogen [21] is one of

the best in this category. The MCL algorithm iteratively applies two operators (expansion and inflation) by matrix computation until convergence. Expansion operator simulates spreading of random walks and inflation models demotion of inter-cluster walks; the sequence matrix computation results in eliminating inter-cluster interactions and leaving only intra-cluster components. The complexity of MCL is $O(m^2|V|)$, where m is the number of attributes associated to each vertex. A key point of random walk is that it is actually linked to spectral clustering [26], e.g., ncut can be expressed in terms of transition probabilities and optimizing ncut can be achieved by computing the stationary distribution of a random walk in the graph.

The shortest path based graph clustering algorithms are based on the hypothesis that the links between clusters are likely to be in the shortest paths. The use of betweenness and information centrality are two representative approaches in this category. The concept of edge betweenness [27] refers to the number of shortest paths connecting any pair of vertices that pass through the edge. Girvan and Newman [27] proposed an algorithm that iteratively removes one of the edges with the highest betweenness. The complexity of the algorithm is $O(|V||E|^2)$. Instead of betweenness, Fortunato et al. [28] used information centrality for each edge and stated that it performs better than betweenness but with a higher complexity of $O(|V||E|^3)$.

We firmly believe that no algorithm is a panacea for three reasons. First, the “best clustering” depends on applications, data characteristics, and granularity. Second, a clustering algorithm is usually developed to optimize some quality measure as its objective function, therefore, it is unfair to compare one algorithm that favors one measure with another that favors some different measure. Finally, there is no perfect measure that captures all the characteristics of cluster structures for all types of datasets. However, all graph clustering algorithms share some common open issues, such as storage cost, processing cost in terms of memory and computation, and the need for optimizations and distributed graph clustering algorithms for big graph analytics.

2.2 Basic Preparation for Graph Clustering

Graph Storage Structure. There are mainly three types of data structures for the representation of graphs in practice [29]: Adjacency list, Adjacency matrix, and Sparse Matrix. Adjacency list of a vertex keeps, for each vertex in the graph, a list of all other vertices to which it has an edge. Adjacency matrix of a graph G on n vertices is the $n \times n$ matrix where the non-diagonal entry a_{ij} is the number of edges from vertex i to vertex j , and the diagonal entry a_{ii} , depending on the convention, is either once or twice the number of edges (loops) from vertex i to itself. A sparse matrix is an adjacency matrix populated primarily with zeros. In this case, we create vectors to store the indices and values of the non-zero elements. The computational complexity of sparse matrix operation is proportional to the number of non-zero elements in the matrix. Sparse matrix is generally preferred because substantial memory requirement reductions can be realized by storing only the non-zero entries.

Handling A Large Number of Attributes. Large attributed graphs usually contain huge amounts of attributes in real applications. Each attribute may have abundant values. The available main-memory still remains very small compared to the size of large graphs with rich attributes. To make graph clustering approach applicable to a wide range of applications, we need to first handle rich attributes as well as continuous attributes with preprocessing techniques in the following.

First of all, we can perform correlation analysis to detect correlation between attributes and then perform dimensionality reduction to retain a smaller set of orthogonal dimensions. Widely used dimensionality reduction techniques such as principal component analysis (PCA) and multifactor dimensionality reduction (MDR) can be used to create a mapping from the original space to a new space with fewer dimensions. According to the mapping, we can compute the new attribute values of a vertex based on the values of its original attributes. Then we can construct the attribute augmented graph in the new feature space and perform graph clustering.

Discretization for Continuous Attributes. To handle continuous attributes, discretization can be applied to convert them to nominal features. Typically the continuous values are discretized into K partitions of an equal interval (equal width) or K partitions each with the same number of data points (equal frequency). For example, there is an attribute “prolific” for each author in the DBLP bibliographic graph indicating whether the author is prolific. If we use the number of publications to measure the prolific value of an author, then “prolific” is a continuous attribute. According to the distribution of the publication number in DBLP, we discretized the publication number into 3 partitions: authors with < 5 papers are labeled as low prolific, authors with ≥ 5 but < 20 papers are prolific, and the authors with ≥ 20 papers are tagged as highly prolific.

2.3 Graph Clustering with SA-Cluster

In order to demonstrate the advantage and feasibility of graph clustering with both structure similarity and attribute similarity, we describe the SA-Cluster proposed by Zhou et.al [14], a graph clustering algorithm by combining structural and attribute similarities. SA-Cluster uses the random walk distance as the vertex similarity measure and performs clustering by following the K-Medoids framework. As different attributes may have different degrees of importance, a weight self-adjustment method was used to learn the degree of contributions by different attributes in the graph clustering process based on clustering convergence rate. The attribute edge weights $\{\omega_1, \dots, \omega_m\}$ are updated in each iteration of the clustering process. Accordingly, the transition probabilities on the graph are affected iteratively with the attribute weight adjustments. Thus the random walk distance matrix needs to be recalculated in each iteration of

the clustering process. Since the random walk distance calculation involves matrix multiplication, which has a time complexity of $O(n^3)$, the repeated random walk distance calculation causes a non-trivial computational cost in SA-Cluster. Zhou et.al [14] showed through the experiments that the random walk distance computation takes 98% of the total clustering time in SA-Cluster.

The concept of random walk has been widely used to measure vertex distances and similarities. Jeh and Widom [30] designed a measure called SimRank, which defines the similarity between two vertices in a graph by their neighborhood similarity. Pons and Latapy [31] proposed to use short random walks of length l to measure the similarity between two vertices in a graph for community detection. Tong et al. [32] designed an algorithm for fast random walk computation. Other studies which use random walk with restarts include connection subgraph discovery [33] and center-piece subgraph discovery [34]. Liu et al. [35] proposed to use random walk with restart to discover subgraphs that exhibit significant changes in evolving networks.

In the subsequent sections, we describe in detail the SA-Cluster algorithm, especially the weight self-adjustment mechanism in [14] and the possible techniques for cost reduction through efficient computation of random walk distance upon the weight increments via incremental approaching the augmented graph[15]. We also provide a discussion on the set of open issues and research challenges for scaling large graph clustering with rich attributes.

3 Graph Clustering Based on Structural/Attribute Similarities

In this section, we first present the formulation of attribute augmented graph considering both structural and attribute similarities. A unified distance measure based on random walk is proposed to combine these two objectives. We then give an adaptive clustering algorithm SA-Cluster for the attributed graph.

The problem is quite challenging because structural and attribute similarities are two seemingly independent, or even conflicting goals – in our example, authors who collaborate with each other may have different properties, such as research topics, age, as well as other possible attributes like positions held and prolific numbers; while authors who work on the same topics or who are in a similar age may come from different groups with no collaborations. It is not straightforward to balance these two objectives.

To combine both structural and attribute similarities, we first define an *attributed augmented graph*. Figure 2 is an attribute augmented graph on the coauthor network example. Two attribute vertices v_{11} and v_{12} representing the topics “XML” and “Skyline” are added to the attribute graph and form an attribute augmented graph. Authors with the topic ?XML? are connected to v_{11} in dashed lines. Similarly, authors with the topic ?Skyline? are connected to v_{12} . It intentionally omits the attribute vertices and edges corresponding to the age attribute, for the sake of clear presentation. Then the graph has two types of edges: the

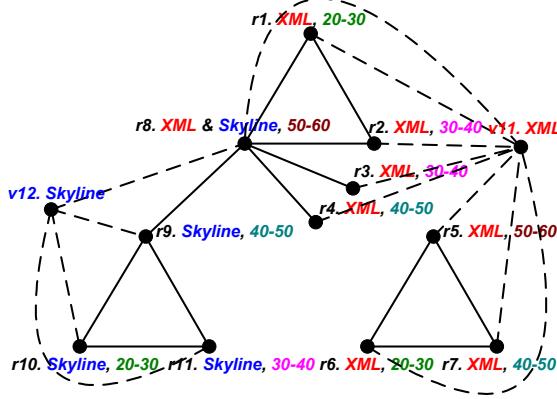


Fig. 2. Attribute Augmented Graph [15]

coauthor edge and the attribute edge. Two authors who have the same research topic are now connected through the attribute vertex.

A unified neighborhood random walk distance measure is designed to measure vertex closeness on an attribute augmented graph. The random walk distance between two vertices $v_i, v_j \in V$ is based on one or more paths consisting of both structure edges and attribute edges. Thus it effectively combines the structural proximity and attribute similarity of two vertices into one unified measure.

We first review the definition of transition probability matrix and random walk matrix. The transition probability matrix P_A is represented as

$$P_A = \begin{bmatrix} P_{V_1} & A_1 \\ B_1 & O \end{bmatrix} \quad (1)$$

where P_{V_1} is a $|V| \times |V|$ matrix representing the transition probabilities between structure vertices; A_1 is a $|V| \times |V_a|$ matrix representing the transition probabilities from structure vertices to attribute vertices; B_1 is a $|V_a| \times |V|$ matrix representing the transition probabilities from attribute vertices to structure vertices; and O is a $|V_a| \times |V_a|$ zero matrix.

The detailed definitions for these four submatrices are shown as follows. The transition probability from vertex v_i to vertex v_j through a structure edge is

$$p_{v_i, v_j} = \begin{cases} \frac{\omega_0}{|N(v_i)| * \omega_0 + \omega_1 + \dots + \omega_m}, & \text{if } (v_i, v_j) \in E \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where $N(v_i)$ represents the set of structure vertices connected to v_i . Similarly, the transition probability from v_i to v_{jk} through an attribute edge is

$$p_{v_i, v_{jk}} = \begin{cases} \frac{\omega_j}{|N(v_i)| * \omega_0 + \omega_1 + \dots + \omega_m}, & \text{if } (v_i, v_{jk}) \in E_a \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The transition probability from v_{ik} to v_j through an attribute edge is

$$p_{v_{ik}, v_j} = \begin{cases} \frac{1}{|N(v_{ik})|}, & \text{if } (v_{ik}, v_j) \in E_a \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

The transition probability between two attribute vertices v_{ip} and v_{jq} is 0 as there is no edge between attribute vertices.

Based on the definition of the transition probability matrix, the unified neighborhood random walk distance matrix R_A can be defined as follow,

$$R_A = \sum_{k=1}^l c(1 - c)^k P_A^k \quad (5)$$

where P_A is the transition probability matrix of an attribute augmented graph G_a . l as the length that a random walk can go and $c \in (0, 1)$ is the random walk restart probability

According to this distance measure, we take a K-Medoids clustering approach to partition the graph into k clusters which have both cohesive intra-cluster structures and homogeneous attribute values. In the preparation phase, we initialize the weight value for each of the m attributes to value of 1, and select k initial centroids with the highest density values.

As different attributes may have different degrees of importance, at each iteration, a weight ω_i , which is initialized to 1.0, is assigned to an attribute a_i . A weight self-adjustment method is designed to learn the degree of contributions of different attributes. The attribute edge weights $\{\omega_1, \dots, \omega_m\}$ are updated in each iteration of the clustering process through quantitatively estimation of the contributions of attribute similarity in the random walk distance measure. Theoretically we can prove that the weights are adjusted towards the direction of clustering convergence.

In the above example, after the first iteration, the weight of research topic will be increased to a larger value while the weight of age will be decreased, as research topic has better clustering tendency than age. Accordingly, the transition probabilities on the graph are affected iteratively with the attribute weight adjustments. Thus the random walk distance matrix needs to be recalculated in next iteration of the clustering process. The algorithm repeats the above four steps until the objective function converges.

One issue with SA-Cluster is the computational complexity. We need to compute N^2 pairs of random walk distances between vertices in V through matrix multiplication. As $W = \{\omega_1, \dots, \omega_n\}$ is updated, the random walk distances need to be recalculated, as shown in SA-Cluster. The cost analysis of SA-Cluster can be expressed as follows.

$$t \cdot (T_{\text{random_walk}} + T_{\text{centroid_update}} + T_{\text{assignment}}) \quad (6)$$

where t is the number of iterations in the clustering process, $T_{\text{random_walk}}$ is the cost of computing the random walk distance matrix R_A , $T_{\text{centroid_update}}$ is the

Algorithm 1. Attributed Graph Clustering SA-Cluster

Input: an attributed graph G , a length limit l of random walk paths, a restart probability c , a parameter σ of influence function, cluster number k .

Output: k clusters V_1, \dots, V_k .

- 1: Initialize $\omega_1 = \dots = \omega_m = 1.0$, fix $\omega_0 = 1.0$;
 - 2: Calculate the unified random walk distance matrix R_A^l ;
 - 3: Select k initial centroids with highest density values;
 - 4: Repeat until the objective function converges:
 - 5: Assign each vertex v_i to a cluster C^* with a centroid c^* where $c^* = \text{argmax}_{c_j} d(v_i, c_j)$;
 - 6: Update the cluster centroids with the most centrally located point in each cluster;
 - 7: Update weights $\omega_1, \dots, \omega_m$;
 - 8: Re-calculate R_A^l ;
 - 9: Return k clusters V_1, \dots, V_k .
-

cost of updating cluster centroids, and $T_{assignment}$ is the cost of assigning all points to cluster centroids.

The time complexity of $T_{centroid_update}$ and $T_{assignment}$ is $O(n)$, since each of these two operations performs a linear scan of the graph vertices. On the other hand, the time complexity of T_{random_walk} is $O(n^3)$ because the random walk distance calculation consists of a series of matrix multiplication and addition. According to the random walk equation, $R_A^l = \sum_{\gamma=1}^l c(1-c)^\gamma P_A^\gamma$ where l is the length limit of a random walk. To compute R_A^l , we have to compute $P_A^2, P_A^3, \dots, P_A^l$, i.e., $(l-1)$ matrix multiplication operations in total. It is clear that T_{random_walk} is the dominant factor in the clustering process. We find in the experiments that the random walk distance computation takes 98% of the total clustering time in SA-Cluster.

To reduce the number of matrix multiplication, full-rank approximation optimization techniques on matrix computation based on Matrix Neumann Series and SVD decomposition are designed to improve efficiency in calculating the random walk distance. It reduces the number of matrix multiplication from $O(l)$ to $O(\log_2 l)$ where l is the length limit of the random walks.

4 The Incremental Algorithm

In this section, we show one way to improve the efficiency and scalability of SA-Cluster by using an efficient incremental computation algorithm to update the random walk distance matrix. The core idea is to compute the full random walk distance matrix only once at the beginning of the clustering process. Then in each following iteration of clustering, given the attribute weight increments $\{\Delta\omega_1, \dots, \Delta\omega_m\}$, we want to update the original random walk distance matrix, instead of re-calculating the matrix from scratch.

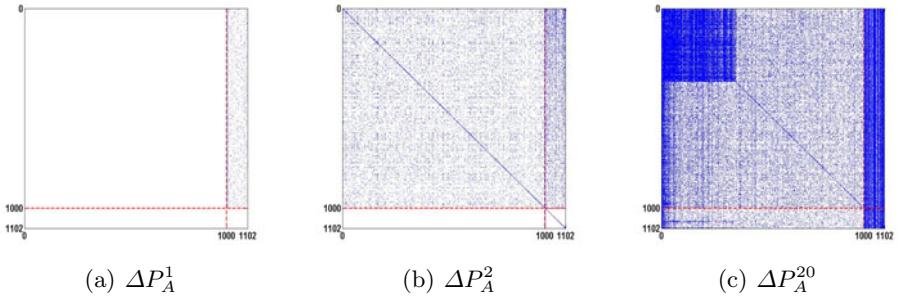


Fig. 3. Matrix Increment Series [15]

Example 1. Each of 1,000 authors has two attributes: “prolific” and “research topic”. The first attribute “prolific” contains two values, and the second one “research topic” has 100 different values. Thus the augmented graph contains 1,000 structure vertices and 102 attribute vertices. The attribute edge weights for “prolific” and “research topic” are ω_1, ω_2 respectively. Figure 3 shows three matrices $\Delta P_A^1, \Delta P_A^2$ and ΔP_A^{20} corresponding to the 1st, 2nd, and 20th order matrix increment, due to the attribute weight increments $\{\Delta\omega_1, \Delta\omega_2\}$. The blue dots represent non-zero elements and the red dashed lines divide each matrix into submatrices according to the block matrix representation. As shown in Figure 3, ΔP_A^k becomes denser when k increases, which demonstrates that the effect of attribute weight increments is propagated to the whole graph through matrix multiplication.

Existing fast random walk [32] or incremental PageRank computation approaches [36,37] can not be directly applied to our problem, as they partition the graph into a changed part and an unchanged part. However, our incremental computation problem is much more challenging than the above problems, because the boundary between the changed part and the unchanged part of the graph is not clear. The attribute weight adjustments will be propagated to the whole graph in l steps. As we can see from Figure 3, although the edge weight increments $\{\Delta\omega_1, \dots, \Delta\omega_m\}$ affect a very small portion of the transition probability matrix P_A , (i.e., see ΔP_A^1), the changes are propagated widely to the whole graph through matrix multiplication (i.e., see ΔP_A^2 and ΔP_A^{20}). It is difficult to partition the graph into a changed part and an unchanged part and focus the computation on the changed part only.

The main idea of the incremental algorithm [15] can be outlined as follows. According to Eq.(5), R_A is the weighted sum of a series of matrices P_A^k , where P_A^k is the k -th power of the transition probability matrix P_A , $k = 1, \dots, l$. Hence the problem of computing ΔR_A can be decomposed into the subproblems of computing ΔP_A^k for different k values. Therefore, our target is, given the original matrix P_A^k and the edge weight increments $\{\Delta\omega_1, \dots, \Delta\omega_m\}$, compute the increment ΔP_A^k .

The k th-order matrix increment ΔP_A^k can be calculated based on: (1) the original transition probability matrix P_A and increment matrix ΔA_1 , (2) the $(k-1)$ -th order matrix increment ΔP_A^{k-1} , and (3) the original k th order sub-matrices A_k and C_k . The key is that, if ΔA_1 and ΔP_A^{k-1} contain many zero elements, we can apply sparse matrix representation to speed up the matrix multiplication.

In summary, the incremental algorithm for calculating the new random walk distance matrix $R_{N,A}$, given the original R_A and the weight increments $\{\Delta\omega_1, \dots, \Delta\omega_m\}$ iteratively computes the increments ΔP_A^k for $k = 1, \dots, l$, and accumulates them into the increment matrix ΔR_A according to Eq.(5). Finally the new random walk distance matrix $R_{N,A} = R_A + \Delta R_A$ is returned.

The total runtime cost of the clustering process with Inc-Cluster can be expressed as

$$T_{\text{random-walk}} + (t - 1) \cdot T_{\text{inc}} + t \cdot (T_{\text{centroid-update}} + T_{\text{assignment}})$$

where T_{inc} is the time for incremental computation and $T_{\text{random-walk}}$ is the time for computing the random walk distance matrix at the beginning of clustering. The speedup ratio r between SA-Cluster and Inc-Cluster is

$$\frac{t(T_{\text{random-walk}} + T_{\text{centroid-update}} + T_{\text{assignment}})}{T_{\text{random-walk}} + (t - 1)T_{\text{inc}} + t(T_{\text{centroid-update}} + T_{\text{assignment}})}$$

Since $T_{\text{inc}}, T_{\text{centroid-update}}, T_{\text{assignment}} \ll T_{\text{random-walk}}$, the speedup ratio is approximately

$$r \approx \frac{t \cdot T_{\text{random-walk}}}{T_{\text{random-walk}}} = t \quad (7)$$

Therefore, Inc-Cluster can improve the runtime cost of SA-Cluster by approximately t times, where t is the number of iterations in clustering.

Readers may refer to [14,15] for detailed experimental evaluation of the SA-Cluster and its incremental algorithm with structure-similarity based approach and attribute-similarity based approach in terms of runtime complexity and graph density and entropy measures.

5 Optimization Techniques

The iterative calculation of random walk distance matrix with attribute weight refinement is a very useful model for analyzing the closeness between two vertices in large graph data. Unfortunately, it has one significant drawback: memory performance is terrible. With large graph data, the results can be disastrous, leading to huge memory use, poor performance, and in the extreme case the termination of the clustering process by the operating system. The complexity of matrix multiplication, if carried out naively, is $O(n^3)$ so that calculating large numbers of matrix multiplications can be very time-consuming [29]. In this section, we will first analyze the storage cost of the incremental algorithm Inc-Cluster. Then we will discuss some techniques to further improve computational performance as well as save memory consumption.

5.1 The Storage Cost and Optimization

According to the incremental algorithm [15], we need to store a series of submatrices, as listed in the following.

The original transition probability matrix P_A . Based on the computational equations of ΔP_{V_k} , ΔA_k , ΔB_k and ΔC_k , we have to store P_{V_1} , B_1 , ΔA_1 and $A_{N,1}$. According to the equation of ΔA_1 , $\Delta A_1 = [\Delta\omega_1 \cdot A_{a_1}, \dots, \Delta\omega_m \cdot A_{a_m}]$ where $A_1 = [A_{a_1}, A_{a_2}, \dots, A_{a_m}]$. In addition $A_{N,1} = A_1 + \Delta A_1$. Therefore, we only need to store A_1 so as to derive ΔA_1 and $A_{N,1}$ with some simple computation. In summary, we need to store the original transition probability matrix P_A .

The $(k-1)$ th order matrix increment ΔP_A^{k-1} . To calculate the k th order matrix increment ΔP_A^k , based on the equations of ΔP_{V_k} , ΔA_k , ΔB_k and ΔC_k , we need to use $\Delta P_{V_{k-1}}$, ΔA_{k-1} , ΔB_{k-1} and ΔC_{k-1} . Therefore, we need to store the $(k-1)$ th order matrix increment ΔP_A^{k-1} . After ΔP_A^k is computed, we can throw away ΔP_A^{k-1} and save ΔP_A^k in turn for the computation of ΔP_A^{k+1} in the next iteration.

A series of A_k and C_k for $k = 2, \dots, l$. In the equation of ΔA_k , we have derived $P_{V_{k-1}} \Delta A_1 = [\Delta\omega_1 \cdot A_{k,a_1}, \dots, \Delta\omega_m \cdot A_{k,a_m}]$. We have mentioned that, the scalar multiplication $\Delta\omega_i \cdot A_{k,a_i}$ is cheaper than the matrix multiplication $P_{V_{k-1}} \Delta A_1$. In addition, this is more space efficient because we only need to store A_k , but not $P_{V_{k-1}}$. The advantage is that the size of A_k is $|V| \times |V_a| = n \times \sum_{i=1}^m n_i$, which is much smaller than the size of $P_{V_{k-1}}$ as $|V| \times |V| = n^2$. The above conclusion holds because $\sum_{i=1}^m n_i \ll n$. For example, in our experiments, we cluster a network of 10,000 authors (i.e., $n = 10,000$) with 103 attribute vertices (i.e., $\sum_{i=1}^m n_i = 103$). The size of $P_{V_{k-1}}$ is $10,000^2$ while the size of A_k is $10,000 \times 103$. Thus $P_{V_{k-1}}$ is about 100 times larger than A_k .

Similarly, in the equation of ΔC_k , we have $B_{k-1} \Delta A_1 = [\Delta\omega_1 \cdot C_{k,a_1}, \dots, \Delta\omega_m \cdot C_{k,a_m}]$. In this case we only need to store C_k , but not B_{k-1} . The advantage is that the size of C_k is $|V_a| \times |V_a| = (\sum_{i=1}^m n_i)^2$, which is much smaller than the size of B_{k-1} as $|V_a| \times |V| = \sum_{i=1}^m n_i \times n$. For example, in our experiments, the size of B_{k-1} is $103 \times 10,000$ while the size of C_k is 103^2 . Thus B_{k-1} is about 100 times larger than C_k .

In summary, to calculate ΔP_A^k for $k = 2, \dots, l$, we have to store A_k and C_k for different k values.

Total storage cost. By adding up the sizes of matrices P_A , ΔP_A^{k-1} , ΔP_A^k , R_A , A_k and C_k for different k , the total storage cost of Inc-Cluster is

$$\begin{aligned} T_{total} &= \text{size}(R_A) + \text{size}(P_A) + \text{size}(\Delta P_A^{k-1}) + \text{size}(\Delta P_A^k) \\ &\quad + \sum_{k=2}^l \text{size}(A_k) + \sum_{k=2}^l \text{size}(C_k) \\ &= |V|^2 + 3(|V| + |V_a|)^2 + (l - 1)(|V| \times |V_a| + |V_a|^2) \\ &= n^2 + 3(n + \sum_{i=1}^m n_i)^2 + (l - 1)(n \cdot \sum_{i=1}^m n_i + (\sum_{i=1}^m n_i)^2) \end{aligned}$$

On the other hand, the non-incremental clustering algorithm SA-Cluster has to store four matrices in memory including P_A , P_A^{k-1} , P_A^k and R_A . Therefore, the extra space used by Inc-Cluster compared with SA-Cluster is

$$T_{extra} = (l - 1)(n \cdot \sum_{i=1}^m n_i + (\sum_{i=1}^m n_i)^2) \quad (8)$$

which is linear of n . Therefore, Inc-Cluster uses a small amount of extra space compared with SA-Cluster.

There are a number of research projects dedicated to graph databases. One objective of such development is to optimize the storage and access of large graphs with billions of vertices and millions of relationships. The Resource Description Framework (RDF) is a popular schema-free data model that is suitable for storing and representing graph data in a compact and efficient storage and access structure. Each RDF statement is in the form of subject-predicate-object expressions. A set of RDF statements intrinsically represents a labeled, directed graph. Therefore, widely used RDF storage techniques such as RDF-3X [40] and Bitmat [41] can be used to create a compact organization for large graphs.

5.2 Matrix Computation Optimization

There are three kinds of optimization techniques for matrix multiplication: block algorithms, sampling techniques and group-theoretic approach. All these techniques can speed up the performance of matrix computation.

Recent work by numerical analysts has shown that the most important computations for dense matrices are blockable [42]. The blocking optimization works well if the blocks fit in the small main memory. It is quite flexible and applicable to adjust block sizes and strategies in terms of the size of main memory as well as the characteristics of the input matrices.

Sampling Techniques are primarily meant to reduce the number of non-zero entries in the matrix and hence save memory. It either samples non-zero entries in terms of some probability distribution [43] or prunes those entries below the threshold based on the average values within a row or column [8]. When we use sparse matrix or other compression representations, it can dramatically improve the scalability and capability of matrix computation.

Recently, a fast matrix multiplication algorithm based on group-theoretic approach was proposed in [44]. It selects a finite group G satisfying the *triple product property* that allows $n \times n$ matrix multiplication to be reduced to multiplication of elements of the group algebra $\mathcal{C}[G]$. A Fourier transform is performed to decompose a large matrix multiplication into several smaller matrix multiplications, whose sizes are the character degrees of G . This gives rise to a complexity at least as great as $O(n^{2.41})$.

As mentioned previously, all experiments [15] on DBLP 84, 170 dataset needed a high-memory configuration (128GB main memory). To improve the applicability of clustering algorithms, Zhou et.al [45] showed their experimental results

about scaling large graph clustering with block multiplication optimization under memory constrained server. Experiments were done on a low-memory environment: two compute servers with 4GB and 8GB main memory, respectively. The runtime curve appears an interesting “U” curve. That is, the runtime is relatively long when the number of blocks is set to be extremely small or large but it keeps quite stable between the two ends of the spectrum. Although block optimization technique can dramatically decrease the memory consumption required, SA-Cluster with block multiplication optimization is usually 8 and 4.5 times slower than non-block SA-Cluster for 4GB and 8GB main memory environments respectively. One obvious approach to scaling large graph clustering is to develop distributed graph clustering methods. In short, we argue that how to improve the scalability and applicability of graph clustering algorithms continues to be an important open issue for wide deployment of graph clustering to big data analytics.

5.3 Parallelism

Although data storage techniques are growing at a much faster speed, the available main-memory still remains very small compared to the accessible disk-space. This creates the main bottleneck while executing large scale matrix computations. Parallel Computing has been employed for many years. Parallel approaches in a distributed memory multicomputer environment can efficiently improve the scalability and applicability of matrix multiplications [46]. The distributed system is viewed as a ring of processors and independent disk algorithms parallelized on block level.

MapReduce [47] is an excellent distributed computing model for processing large data sets on clusters of computers. For example, one can specify mappers that are responsible for distributing the block data to the reducers, with the help of a carefully chosen intermediate key structure, key comparator and partitioning functions. Reducers can be used to do the block multiplications and merge intermediate blocks to produce the final results.

6 Conclusion

In this chapter, we have given an overview of some of the essential techniques of graph clustering based through a classification of graph clustering methods into three classes: structure-based clustering, attribute-based clustering and structure-attribute combined clustering. We argue that a key challenge for addressing the problem of clustering large graphs with rich attributes is to achieve a good balance between structural and attribute similarities. Such balance can be driven by the objective function defined by domain-specific graph clustering analysis. We have described the use of the attribute-augmented graph to tackle graph clustering with rich attributes with two novel developments. First, we described a hybrid structure and attribute based neighborhood random walk distance measure, which is designed to measure vertex closeness on an attribute

augmented graph. Second, we describe the use of a learning algorithm to adjust the degree of contributions of different attributes in the random walk model as we iteratively refine the clusters, and prove that the weights are adjusted towards the direction of clustering convergence. By using a K-Medoids clustering approach, we show that the iterative weight assignment method is effective for partitioning a graph into k clusters with both cohesive intra-cluster structures and homogeneous attribute values. Finally, we presented a set of open issues and challenges in clustering analysis of large graphs with rich attributes, including processing and storage optimizations as well as distributed and parallel analytic models.

Acknowledgement. The authors are partially funded by research grants under NSF NetSE program, NSF CyberTrust program, an IBM SUR grant, an IBM faculty award, and a grant from Intel research council.

References

1. Ng, R., Han, J.: Efficient and effective clustering method for spatial data mining. In: Proc. 1994 Int. Conf. Very Large Data Bases (VLDB 1994), Santiago, Chile, pp. 144–155 (September 1994)
2. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases. In: Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining (KDD 1996), pp. 226–231 (1996)
3. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: Proc. 1998 ACM SIGMOD Int. Conf. Management of Data (SIGMOD 1998), Seattle, WA, pp. 94–105 (June 1998)
4. Gibson, D., Kleinberg, J., Raghavan, P.: Inferring web communities from link topology. In: Proc. 9th ACM Conf. Hypertext and Hypermedia, Pittsburgh, PA, pp. 225–234 (June 1998)
5. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
6. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E69, 026113 (2004)
7. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.J.: Scan: a structural clustering algorithm for networks. In: Proc. 2007 Int. Conf. Knowledge Discovery and Data Mining (KDD 2007), San Jose, CA, pp. 824–833 (August 2007)
8. Satuluri, V., Parthasarathy, S.: Scalable graph clustering using stochastic flows: Applications to community discovery. In: Proc. 2009 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD 2009), Paris, France (June 2009)
9. Wang, T., Srivatsa, M., Agrawal, D., Liu, L.: Learning indexing and diagnosing network faults. In: Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Paris, France, June 28-July 1, pp. 857–866 (2009)
10. Wang, T., Srivatsa, M., Agrawal, D., Liu, L.: Spatio-temporal patterns in network events. In: Proceedings of the 6th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2010), NY, USA, November 30-December 3 (2010)

11. Ramaswamy, L., Gedik, B., Liu, L.: A distributed approach to node clustering in decentralized peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 16, 1–16 (2005)
12. Wang, Y., Liu, L., Pu, C., Zhang, G.: An utility-driven routing scheme for scaling multicast applications. In: Proceedings of 2009 International Conference on Collaborative Computing (CollaborateCom 2009), October 9–12. IEEE Press, Chicago (2009)
13. Tian, Y., Hankins, R.A., Patel, J.M.: Efficient aggregation for graph summarization. In: Proc. 2008 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD 2008), Vancouver, Canada, pp. 567–580 (June 2008)
14. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. In: Proc. 2009 Int. Conf. on Very Large Data Base (VLDB 2009), Lyon, France (August 2009)
15. Zhou, Y., Cheng, H., Yu, J.X.: Clustering large attributed graphs: An efficient incremental approach. In: Proc. 2010 Int. Conf. on Data Mining (ICDM 2010), Sydney, Australia (December 2010)
16. Sun, J., Faloutsos, C., Papadimitriou, S., Yu, P.S.: Graphscope: parameter-free mining of large time-evolving graphs. In: Proc. 2007 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD 2007), San Jose, CA, pp. 687–696 (2007)
17. Sun, Y., Han, J., Zhao, P., Yin, Z., Cheng, H., Wu, T.: Rankclus: Integrating clustering with ranking for heterogenous information network analysis. In: Proc. 2009 Int. Conf. Extending Database Technology (EDBT 2009), Saint Petersburg, Russia (March 2009)
18. Navlakha, S., Rastogi, R., Shrivastava, N.: Graph summarization with bounded error. In: Proc. 2008 ACM-SIGMOD Int. Conf. Management of Data (SIGMO 2008), Vancouver, Canada, pp. 419–432 (June 2008)
19. Tsai, C.-Y., Chui, C.-C.: Developing a feature weight self-adjustment mechanism for a k-means clustering algorithm. *Computational Statistics and Data Analysis* 52, 4658–4672 (2008)
20. Cai, D., Shao, Z., He, X., Yan, X., Han, J.: Mining hidden community in heterogeneous social networks. In: Proc. Workshop on Link Discovery: Issues, Approaches and Applications (LinkKDD 2005), Chicago, IL, pp. 58–65 (August 2005)
21. van Dongen, S.: Graph clustering by flow simulation. Ph.D. dissertation, University of Utrecht (2000)
22. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. *Canadian Journal of Mathematics* 8, 399–404 (1956)
23. Kernighan, B.W., Lin, S.: An efficient heuristic procedur for partitioning graphs. *Bell Syst. Techn. J.* 49, 291–307 (1970)
24. Flake, G.W., Tarjan, R.E., Tsiotsiouliklis, K.: Graph clustering and minimum cut trees. *Internet Mathematics* 1 (2003)
25. Luxburg, U., Bousquet, O., Belkin, M.: Limits of spectral clustering. MIT Press, Cambridge (2005)
26. Luxburg, U.: A tutorial on spectral clustering. Technical Report 149 (2006)
27. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. *Proc. Natl. Acad. Sci., USA* 99, 7821–7826 (2003)
28. Fortunato, S., Latora, V., Marchiori, M.: A method to find community structures based on information centrality. *Phys. Rev. E* 70, 056104 (2004)
29. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*, 2nd edn. MIT Press and McGraw-Hill (2001)

30. Jeh, G., Widom, J.: SimRank: a measure of structural-context similarity. In: Proc. 2002 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2002), Edmonton, Canada, pp. 538–543 (July 2002)
31. Pons, P., Latapy, M.: Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications* 10(2), 191–218 (2006)
32. Tong, H., Faloutsos, C., Pan, J.-Y.: Fast random walk with restart and its applications. In: Proc. 2006 Int. Conf. on Data Mining (ICDM 2006), Hong Kong, pp. 613–622 (December 2006)
33. Faloutsos, C., McCurley, K., Tomkins, A.: Fast discovery of connection subgraphs. In: Proc. 2004 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD 2004), Seattle, WA, pp. 118–127 (August 2004)
34. Tong, H., Faloutsos, C.: Center-piece subgraphs: problem definition and fast solutions. In: Proc. 2006 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD 2006), Philadelphia, PA, pp. 404–413 (2006)
35. Liu, Z., Yu, J.X., Ke, Y., Lin, X., Chen, L.: Spotting significant changing subgraphs in evolving graphs. In: Proc. 2008 Int. Conf. Data Mining (ICDM 2008), Pisa, Italy (December 2008)
36. Desikan, P., Pathak, N., Srivastava, J., Kumar, V.: Incremental page rank computation on evolving graphs. In: Proc. 2005 Int. World Wide Web Conf. (WWW 2005), Chiba, Japan, pp. 1094–1095 (May 2005)
37. Wu, Y., Raschid, L.: Approxrank: Estimating rank for a subgraph. In: Proc. 2009 Int. Conf. Data Engineering (ICDE 2009), Shanghai, China, pp. 54–65 (March 2009)
38. Strang, G.: Linear Algebra and its Applications. Brooks Cole (2005)
39. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
40. Neumann, T., Weikum, G.: Rdf-3x: a risc-style engine for rdf. In: Proceedings of the VLDB Endowment (PVLDB), vol. 1(1), pp. 647–659 (2008)
41. Atre, M., Chaoji, V., Zaki, M.J., Hendler, J.A.: Matrix "bit" loaded: a scalable lightweight join query processor for rdf data. In: Proceedings of the 19th International Conference on World Wide Web (WWW 2010), New York, NY, USA, pp. 41–50 (April 2010)
42. Press, W.H., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical Recipes: The Art of Scientific Computing, 3rd edn. Cambridge University Press, Cambridge (2007)
43. Cohen, E., Lewis, D.: Approximating matrix multiplication for pattern recognition tasks. In: Proceedings of the 8th Symposium on Discrete Algorithms (SODA 1997), New Orleans, Louisiana (Junuary 1997)
44. Cohn, H., Kleinberg, R., Szegedy, B., Umans, C.: Group-theoretic algorithms for matrix multiplication. In: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), Pittsburgh, Pennsylvania, USA (October 2005)
45. Zhou, Y., Liu, L.: Rapid: Resource-aware approach to large scale graph clustering. GT Tech Report, Atlanta, GA (April 2011)
46. Dackland, K., Elmroth, E.: Design and evaluation of parallel block algorithms: Lu factorization on an ibm 3090 vf/600j. In: Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing, Philadelphia, PA, USA (1992)
47. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: Proceedings of the Sixth Symposium on Operating System Design and Implementation (OSDI 2004), San Francisco, CA (December 2004)

Chapter 3

Temporal Data Mining: Similarity-Profiled Association Pattern

Jin Soung Yoo

Department of Computer Science, Indiana University-Purdue University,
Fort Wayne, Indiana, USA
yooj@ipfw.edu

Abstract. Temporal data are of increasing importance in a variety of fields, such as financial data forecasting, Internet site usage monitoring, biomedicine, geographical data processing and scientific observation. Temporal data mining deals with the discovery of useful information from a large amount of temporal data. Over the last decade many interesting techniques of temporal data mining were proposed and shown to be useful in many applications. In this article, we present a temporal association mining problem based on a similarity constraint. Given a temporal transaction database and a user-defined reference sequence of interest over time, similarity-profiled temporal association mining is to discover all associated itemsets whose prevalence variations over time are similar to the reference sequence. The temporal association patterns can reveal interesting association relationships of data items which co-occur with a particular event over time. Most works in temporal association mining have focused on capturing special temporal regulation patterns such as a cyclic pattern and a calendar scheme-based pattern. However, the similarity-based temporal model is flexible in representing interesting temporal association patterns using a user-defined reference sequence. This article presents the problem formulation of similarity-profiled temporal association mining, the design concept of the mining algorithm, and the experimental result.

1 Introduction

Recent advances in data collection and storage technology have made it possible to collect vast amounts of data every day in many areas of business and science. *Data mining* is concerned with analyzing large volumes of data to automatically discover interesting regularities or relationships which in turn lead to better understanding of the underlying processes. The field of *temporal data mining* deals with such analysis in the case of ordered data with temporal interdependences [33,25]. Over the last decades many interesting techniques of temporal data analysis were proposed and shown to be useful in many applications. For example, the most common type of temporal data is time series data, which consist of real values sampled at regular time intervals. Time series analysis has quite a

long history. Weather forecasting, financial or stock market prediction and automatic process control have been of the oldest and most studied applications of time series analysis [12]. Temporal data mining is of a more recent origin with somewhat different objectives. One of major differences between temporal data mining and classical time series analysis lies in the kind of information what we want to estimate or unearth from the data. The scope of temporal data mining extends beyond the standard forecast applications of time series analysis. Very often, in data mining applications, one may be interested in knowing which variables in the data are expected to exhibit any correlations or causal relationships over time. For example, a timestamped list of items bought by customers lends itself to data mining analysis that could reveal which combinations of items tend to be frequently consumed together, and whether they tend to show particular behaviors over time.

Temporal data mining tasks can be grouped as follows: (i) prediction, (ii) classification, (iii) clustering, (iv) search & retrieval and (v) pattern discovery [33]. Of the five categories listed above, algorithms for pattern discovery in large temporal databases, however, are of more recent origin and are mostly discussed in data mining literature. Temporal pattern mining deals with the discovery of *temporal patterns of interest* in temporal data, where the interest is determined by the domain and the application. The diversity of applications has led to the development of many temporal pattern models. The three popular frameworks of temporal pattern discovery are *sequence mining* (or *frequent sequence pattern discovery*), *frequent episode discovery* and *temporal association rule discovery* [33]. Association rule mining is concerned with the discovery of inter-relationships among various data items in transactional data [5]. An example of association rule is

$$\text{wine} \implies \text{cheese} \text{ (support=10\%, confidence =80\%).}$$

This rule says that 10% of customers buy *wine* and *cheese* together, and those who buy *wine* also buy *cheese* 80% of the time. The process of association rule mining is to find all *frequent* itemsets that exceed a user-specified support threshold, and then generate association rules from the frequent itemsets which satisfies a user-given confidence threshold. Following the work of [5], the discovery of association rules has been extensively studied in [23,22,36,38]. In particular, [35,28,37] have paid attention to temporal information which is implicitly related to transaction data, and proposed periodic temporal association mining problems.

Temporal association pattern mining is an important extension of association pattern mining as it can be used to mine the behavior aspects of data over time rather than just states at a point in time. Ozden et al. [35] introduced the idea of cyclic association rules which can discover periodicity time information of data. An association rule is said to be cyclic if it holds with a fixed periodicity along the entire length of the sequence of time intervals. For example, a periodic association pattern may state that wine and cheese are sold together primarily in the weekends. A temporal association pattern can be explained with a binary sequence where the time axis is broken down into equally spaced user-defined

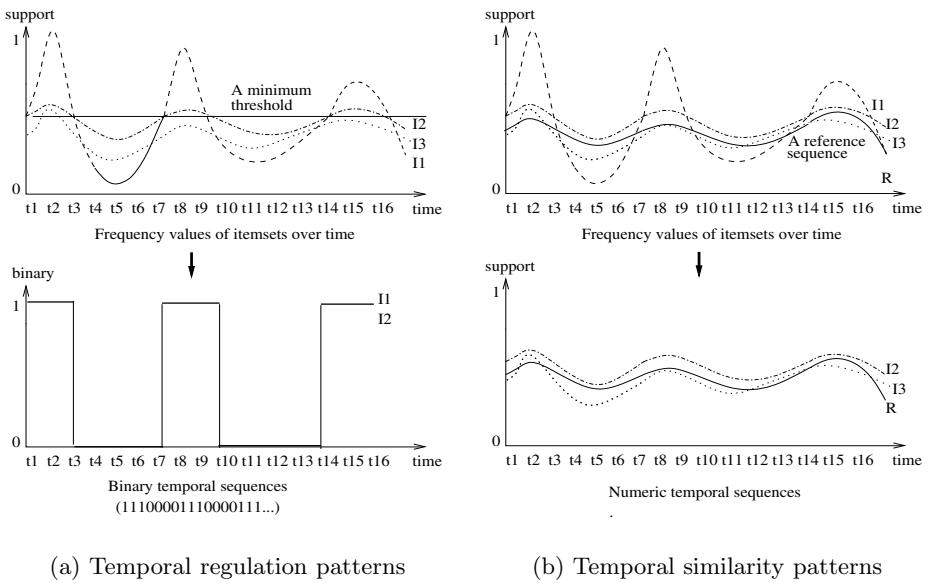


Fig. 1. A comparison of temporal association patterns

time intervals and association rules that hold for the transactions in each of these time intervals are considered. In the binary sequence, 1's correspond to the time intervals in which the association pattern is valid (i.e., the rule satisfies minimum support threshold and confidence threshold.), and the 0's correspond to the time intervals in which it is not valid. For instance, when a defined time interval is day, ‘10000001000000...’ represents the binary sequence of a repetitive temporal pattern on Monday. Fig 1 (a) illustrates an example of periodic temporal association patterns. It shows the support values of three itemsets I_1 , I_2 and I_3 over time, and the binary temporal sequences of I_1 and I_2 under a fixed frequent threshold (e.g., support threshold=0.5).

There are many ways of defining what constitutes a pattern. Fig 1 (a) says that I_2 shows the same temporal pattern with I_1 (i.e., a periodic pattern with the binary sequence, 111000111000...), even if their actual measure strengths are quite different. In contrast, Fig 1 (b) illustrates temporal association patterns based on similarity. It shows that I_2 and I_3 have very similar behaviors over time. As a guidance to find similar itemsets, this model uses a reference sequence. The reference sequence can represent the change of interest measure values of a specific event (item) (e.g., a specific product in market basket data, a stock exchange in the stock market, a climate variable such as temperature or precipitation, and a scientific phenomenon), or a user guided sequence pattern showing special shapes (e.g., a seasonal, emerging or diminishing pattern over time). Current periodic temporal association mining methods cannot reveal

this kind of temporal association pattern. In this article, we present a temporal association pattern mining problem based on a similarity constraint [42].

The remainder of the article is organized as follows. Section 2 presents the problem statement of similarity-profiled temporal association mining. The design concept of the mining algorithm is described in Section 3. Section 4 shows the experimental result. The related work is described in Section 5. Section 6 discusses some future direction of the work.

2 Similarity-Profiled Temporal Association Pattern

Given a timestamped transaction database and a user-defined reference sequence of interest over time, *similarity-profiled temporal association mining* is to discover all associated itemsets whose prevalence(frequency) variations over time are similar to the reference sequence under a threshold. Similarity-profiled temporal association mining can reveal interesting relationships of data items which co-occur with a particular event over time. For example, weather-to-sales relationship is a very interesting problem in retail analysts [2,4]. Wal-Mart discovered a surprising customer buying pattern during hurricane season in a region. Not only did survival kits (e.g., flashlights, generators, tarps) show similar selling patterns with bottled water (which is one of important items in emergency), but so did the sales of Strawberry Pop-Tarts which is an unexpected snack item [4]. The similarity-profiled temporal association mining may help finding such item sets whose sales similarly change to that of a specific item(event) for a period of time. The mining results can improve supply chain planning, and retail decision-making for maximizing the visibility of items most likely to be in high demand during a special time period. As another example in business domain, consider an online web site. Weather.com offers weather-related lifestyle information including travel, driving, home & garden and sporting events as well as weather itself information [3]. According to the web site's report, almost 40% of weather.com visitors shop home improvement products with increase of temperature [3]. The web site may attract more advertisers if it can analyze the relationships of visited web sites through weather.com with changes of weather. Consider a scientific application domain. Earth scientists have been interested in the behavior of climates in a region which are often influenced with the El Niño phenomenon, an abnormal warming in the eastern tropical Pacific Ocean [34]. If we consider the El Niño related index values over last 10 years, e.g., the Southern Oscillation Index(SOI) [34], as a reference sequence, one example of similarity-profiled temporal association might be a climate event pattern of low precipitation and low atmospheric carbon dioxide in Australia whose co-occurrence over time is similar to the fluctuation of the El Niño index sequence.

2.1 Problem Statement

The formal problem statement of similarity-profiled temporal association mining follows below. Fig. 2 shows a simple illustration of the pattern mining.

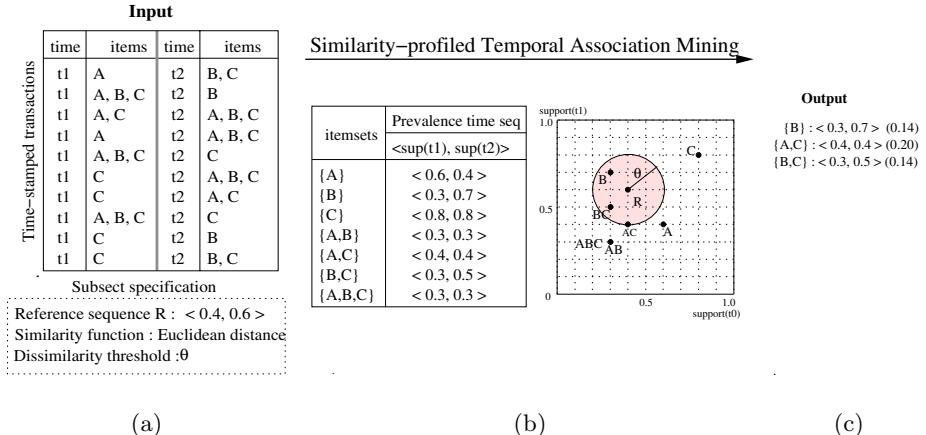


Fig. 2. An example of similarity-profiled temporal association mining (a) Input data (b) Generated support time sequences, and sequence search (c) Output itemsets

Given

- 1) A finite set of items \mathcal{I}
- 2) An interest time period $\mathcal{T}=t_1 \cup \dots \cup t_n$, where t_i is a time slot by a time granularity, $t_i \cap t_j = \emptyset, i \neq j$
- 3) A timestamped transaction database $\mathcal{D}=\mathcal{D}_1 \cup \dots \cup \mathcal{D}_n$, $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset, i \neq j$. Each transaction $d \in \mathcal{D}$ is a tuple $\langle \text{timestamp}, \text{itemset} \rangle$ where timestamp is a time $\in \mathcal{T}$ that the transaction is executed, and $\text{itemset} \subseteq \mathcal{I}$. \mathcal{D}_i is a set of transactions included in time slot t_i .
- 4) A subset specification
 - 4a) A reference sequence $\mathbf{R} = \langle r_1, \dots, r_n \rangle$ over time slots t_1, \dots, t_n
 - 4b) A similarity function $f_{\text{Similarity}}(\mathbf{X}, \mathbf{Y}) \mapsto \mathbb{R}^n$, where \mathbf{X} and \mathbf{Y} are numeric sequences.
 - 4c) A dissimilarity threshold θ

Find A set of itemsets $I \subseteq \mathcal{I}$ which satisfy the given subset specification, i.e., $f_{\text{Similarity}}(S_I, \mathbf{R}) \leq \theta$, where $S_I = \langle s_1, \dots, s_n \rangle$ is the sequence of support values of an itemset I over time slots t_1, \dots, t_n .

Items: We use the standard notion of *items* in traditional association rule mining [5]. Items can be supermarket items purchased by a customer during a shopping visit, product pages viewed in a web session, climate events at a location, stocks exchanged within a hour, etc. Items can be grouped to form an *itemset*. An itemset with k distinct items is referred to as a k *itemset*. The size of the itemset space is $2^{|\mathcal{I}|} - 1$, where $|\mathcal{I}|$ is the number of items.

Time period: A time period can be a particular year or any arbitrary period of time. We model time as discrete, and thus, a total time period can be viewed as a sequence of time slots by a certain time granularity [10]. For example, one

year period can be divided into monthly unit time slots. The i^{th} time slot is denoted with t_i .

Transaction database: The database \mathcal{D} is a set of timestamped transactions. Each transaction is a set of items over a finite item domain, and has a time point when the transaction is executed. The time point associated with a transaction is called its *timestamp*. The transaction dataset can be partitioned to disjoint groups of transactions by a time granularity. \mathcal{D}_i represents a part of transactions of \mathcal{D} executed in time slot t_i .

Subset specification: A subset specification can be used to represent a set of conditions that itemsets have to satisfy to become interesting patterns. Our subset specification consists of three components: a reference sequence, a similarity function, and a dissimilarity threshold. First, we assume that an arbitrary temporal pattern of interest can be defined as a reference sequence by the user. A reference time sequence is a sequence of interesting values over time slots t_1, \dots, t_n . In the example of Fig. 2, $<0.4, 0.6>$ is given as the reference sequence \mathbf{R} . Second, we use a distance-based similarity function, a \mathcal{L}_p norm ($p = 1, 2, \dots, \infty$). Many similarity measures have been discussed in time series database literature [20,18,8,27,16]. A \mathcal{L}_p norm is the most popularly used distance measure in similar time sequence search [41,18,6,26], and can be used as basic building blocks for more complex similarity models as in [7]. When $p=1$, the \mathcal{L}_1 norm is known as a *city-block* or *Manhattan*. When $p=2$, the \mathcal{L}_2 norm is called a *Euclidean distance*, and defined as $\mathcal{L}_2(\mathbf{X}, \mathbf{Y}) = (\sum_{t=1}^n |x_t - y_t|^2)^{\frac{1}{2}}$. We also consider a *normalized Euclidean distance*, $\text{Normalized-L}_2(\mathbf{X}, \mathbf{Y}) = (\frac{1}{n})^{\frac{1}{2}} * \mathcal{L}_2(\mathbf{X}, \mathbf{Y}) = (\frac{\sum_{t=1}^n |x_t - y_t|^2}{n})^{\frac{1}{2}}$, where $\mathbf{X} = < x_1, \dots, x_n >$ and $\mathbf{Y} = < y_1, \dots, y_n >$ are time sequences, and n is the number of time slots. Euclidean distance is used for figure examples in this article. The last component of the subset specification is a dissimilarity threshold. It indicates a maximum discrepancy to allow for similarity-profiled temporal association patterns.

2.2 Interest Measure

The similarity-profiled temporal association pattern uses a composite interest measure which describes a discrepancy degree between the reference sequence and the sequence of frequency values of an itemset over time. A support time sequence is used to represent temporally changed frequency values of an itemset.

Definition 1. Let $\mathcal{D} = \mathcal{D}_1 \cup \dots \cup \mathcal{D}_n$ be a disjoint timestamped transaction dataset. The **support time sequence** of an itemset I is defined as

$$\mathbf{S}_I = < \text{support}(I, \mathcal{D}_1), \dots, \text{support}(I, \mathcal{D}_n) >,$$

where $\text{support}(I, \mathcal{D}_t)$ is the support value of itemset I in a transaction set \mathcal{D}_t , which is the fraction of transactions that contain the itemset I in \mathcal{D}_t such that $\text{support}(I, \mathcal{D}_t) = |\{d \in \mathcal{D}_t | I \subseteq d\}| / |\mathcal{D}_t|$, $1 \leq t \leq n$.

We assume the reference sequence values are in the same scale with the support measure, or can be transformed to the same scale. The interest measure of the similarity-profiled temporal association is defined as following.

Definition 2. Let I be an itemset and $\mathbf{S}_I = \langle s_1, \dots, s_n \rangle$ be the support time sequence of I . Given a reference sequence $\mathbf{R} = \langle r_1, \dots, r_n \rangle$, an interest measure for the similarity-profiled temporal association pattern is defined as $D(\mathbf{R}, \mathbf{S}_I)$ which is a \mathcal{L}_p norm ($p = 1, 2, \dots, \infty$) based dissimilarity distance between \mathbf{R} and \mathbf{S}_I .

An itemset I is called a *similar itemset* if $D(\mathbf{R}, \mathbf{S}_I) \leq \theta$ where θ is a dissimilarity threshold. In Fig. 2, the pattern mining output is $\{\text{B}\}$, $\{\text{A}, \text{C}\}$ and $\{\text{B}, \text{C}\}$ since their interest measure values do not exceed the dissimilarity threshold, 0.2.

3 Mining Algorithm

Similarity-profiled temporal association mining presents challenges in computation. The straight-forward approach is to divide the mining process into two separate phrases. The first phrase computes the support values of all possible itemsets at each time point, and generates their support sequences. The second phrase compares the generated support time sequences with a given reference sequence, and finds similar itemsets. In this step, a multi-dimensional access method such as an R-tree family can be used for a fast sequence search [21,18,26,14]. However, the computational costs of first generating the support time sequences of all combinatorial candidate itemsets and then doing the similarity search become prohibitively expensive with increase of items. Thus it is crucial to devise schemes to reduce the itemset search space effectively for efficient computation. We first present the design concept of similarity-profiled temporal association mining algorithm.

3.1 Envelope of Support Time Sequence

It is a core operation to generate the support time sequences of itemsets in a similarity-profiled association mining algorithm. The operation, however, is very data intensive, and sometimes can produce the sequences of all combinations of items. We explore a way for estimating support time sequences without examining the transaction data. Calders in [13] proposed a set of rules for deducing best bounds on the support of an itemset if the supports of all subsets of it are known.

Theorem 1. Let \mathcal{D} be a transaction dataset, and I be an itemset,

$$\text{support}(I, \mathcal{D}) \in [L(I, \mathcal{D}), U(I, \mathcal{D})]$$

with

$$L(I, \mathcal{D}) = \max\{\sigma_I(J, \mathcal{D}), 0 \mid J \subset I \text{ and } |J| \text{ is even }\},$$

$$U(I, \mathcal{D}) = \min\{\sigma_I(J, \mathcal{D}) \mid J \subset I \text{ and } |J| \text{ is odd }\}$$

$$\text{where } \sigma_I(J, \mathcal{D}) = \sum_{J \subseteq J' \subset I} (-1)^{|I-J'|+1} \cdot \text{support}(J', \mathcal{D}).$$

Timestamped transaction dataset			
D1 time slot t1		D2 time slot t2	
time	items	time	items
.1.	A	.11.	B, C
.2.	A, B, C	.12.	B
.3.	A, C	.13.	A, B, C
.4.	A	.14.	A, B, C
.5.	A, B, C	.15.	C
.6.	C	.16.	A, B, C
.7.	C	.17.	A, C
.8.	A, B, C	.18.	C
.9.	C	.19.	B
10	C	20	B, C

support(AB, D1) <= support(A, D1) = 0.6

- support(AB, D1) <= support(B, D1) = 0.3

- support(AB, D1) <= min(0.6, 03)=0.3

support(AB, D2) <= support(A, D2) = 0.4

- support(AB, D2) <= support(B, D2) = 0.7

- support(AB, D2) <= min(0.4, 07)=0.4

* The upper bound of support sequence of AB : <0.3, 0.4>

support(AB, D1) >= - support((), D1)+ support(A, D1) + support(B, D1)

= -1 +0.6+0.3=-0.1

- support(AB, D1) >= 0

- support(AB, D1) >= max(-0.1, 0)=0

support(AB, D2) >= - support((), D2)+ support(A, D2) + support(B, D2)

= -1 +0.7+0.4=0.1

- support(AB, D2) >= 0

- support(AB, D1) >= max(0.1, 0)=0.1

* The lower bound of support sequence of AB : <0 , 0.1>

(a) An example data

(b) Bounds of supports

Fig. 3. An example of upper and lower bounds of support sequence of itemset AB

$L(I, \mathcal{D})$ means a lower bound of $support(I, \mathcal{D})$, and $U(I, \mathcal{D})$ means an upper bound of $support(I, \mathcal{D})$. The proof of the tight bounds is described in [13]. We adopt this set of rules to derive the tight upper bound and lower bound of support time sequence of an itemset.

Definition 3. Let $\mathcal{D} = \mathcal{D}_1 \cup \dots \cup \mathcal{D}_n$ be a timestamped transaction dataset. The lower bound support time sequence of an itemset I , L_I , and the upper bound support time sequence of I , U_I are defined as following.

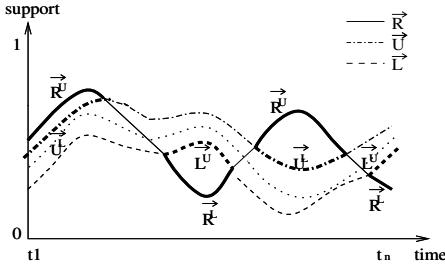
$$L_I = \langle l_1, \dots, l_n \rangle = \langle L(I, D_1), \dots, L(I, D_n) \rangle$$

$$U_I = \langle u_1, \dots, u_n \rangle = \langle U(I, D_1), \dots, U(I, D_n) \rangle$$

Fig. 3 shows the computation of the lower and upper bound support sequences of an itemset $I = \{A, B\}$.

3.2 Lower Bounding Distance

A lower bounding distance concept is used to find itemsets whose support sequences could not possibly match with a reference sequence under a given threshold. If the lower bounding distance of an itemset does not satisfy the dissimilarity threshold, its true distance also does not satisfy the threshold. Thus the lower bounding distance can be used to prune the itemset without computing its true distance. Our lower bounding distance is defined with upper and lower bound support time sequences. It consists of two parts, *upper lower-bounding distance* and *lower lower-bounding distance*.



(a) Subsequences for a lower bounding distance

Itemssets (sequence)	Support seqs $\langle \sup(t_1), \sup(t_2) \rangle$	* similarity function: Euclidean			
		ulb dist	llb dist	lb dist	true dist
A	$\langle 0.6, 0.4 \rangle$	0.20	x	x	0.28
B	$\langle 0.3, 0.7 \rangle$	0.10	x	x	0.14
C	$\langle 0.8, 0.8 \rangle$	0	x	x	0.45
A B; A B_upper	$\langle 0.0, 0.1 \rangle$	0.22	x	0	0.22
A B_lower	$\langle 0.3, 0.3 \rangle$	x	0	?	

Reference : <0.4, 0.6> x : don't care

(b) Lower bounding distance of AB

Fig. 4. An example of lower bounding distances

Definition 4. For a reference sequence \mathbf{R} and the upper bound support sequence \mathbf{U} of an itemset, let $\mathbf{R}^{\mathbf{U}} = \langle r_1, \dots, r_k \rangle$ be a subsequence of \mathbf{R} , and $\mathbf{U}^{\mathbf{L}} = \langle u_1, \dots, u_k \rangle$ be a subsequence of \mathbf{U} where $r_t > u_t$, $1 \leq t \leq k$. The **upper lower-bounding distance** between \mathbf{R} and \mathbf{U} , $D_{Ul}(R, U)$, is defined as $D(\mathbf{R}^{\mathbf{U}}, \mathbf{U}^{\mathbf{L}})$.

The upper lower-bounding distance between \mathbf{R} and \mathbf{U} is a dissimilarity distance between a subsequence of \mathbf{R} , $\mathbf{R}^{\mathbf{U}}$, and a subsequence of \mathbf{U} , $\mathbf{U}^{\mathbf{L}}$, in which each element value r_t in $\mathbf{R}^{\mathbf{U}}$ is greater than the corresponding element value u_t of $\mathbf{U}^{\mathbf{L}}$. For example, when Euclidean distance is the similarity function, $D_{Ul}(\mathbf{R}, \mathbf{U}) = D(\mathbf{R}^{\mathbf{U}}, \mathbf{U}^{\mathbf{L}}) = (\sum_{t=1}^n f(r_t, u_t))^{\frac{1}{2}}$, where if $r_t > u_t$, $f(r_t, u_t) = |r_t - u_t|^2$; otherwise, $f(r_t, u_t) = 0$.

Definition 5. For a reference sequence \mathbf{R} and the lower bound support time sequence \mathbf{L} of an itemset, let $\mathbf{R}^{\mathbf{L}} = \langle r_1, \dots, r_k \rangle$ be a subsequence of \mathbf{R} , and $\mathbf{L}^{\mathbf{U}} = \langle l_1, \dots, l_k \rangle$ be a subsequence of \mathbf{L} where $r_t < l_t$, $1 \leq t \leq k$. The **lower lower-bounding distance** between \mathbf{R} and \mathbf{L} , $D_{Llb}(\mathbf{R}, \mathbf{L})$, is defined as $D(\mathbf{R}^{\mathbf{L}}, \mathbf{L}^{\mathbf{U}})$.

The lower lower-bounding distance between a reference sequence \mathbf{R} and a lower bound support sequence \mathbf{L} is a dissimilarity distance between $\mathbf{R}^{\mathbf{L}}$ and $\mathbf{L}^{\mathbf{U}}$, in which each element value r_t in $\mathbf{R}^{\mathbf{L}}$ are less than the corresponding element value l_t of $\mathbf{L}^{\mathbf{U}}$.

Definition 6. For a reference sequence \mathbf{R} , and the upper bound support time sequence \mathbf{U} and lower bound support time sequence \mathbf{L} of an itemset, the **lower bounding distance**, $D_{lb}(\mathbf{R}, \mathbf{U}, \mathbf{L})$ is defined as $D_{Ul}(\mathbf{R}, \mathbf{U}) + D_{Llb}(\mathbf{R}, \mathbf{L})$.

Fig. 4 (a) gives an example of subsequences, $\mathbf{R}^{\mathbf{U}}$, $\mathbf{U}^{\mathbf{L}}$, $\mathbf{R}^{\mathbf{L}}$ and $\mathbf{L}^{\mathbf{U}}$. As shown, the subsequences do not need to be a continuous sequence. Fig. 4 (b) shows an example of lower bounding distances of {A, B} computed from the upper bound support sequence and lower bound support sequence of {A, B}.

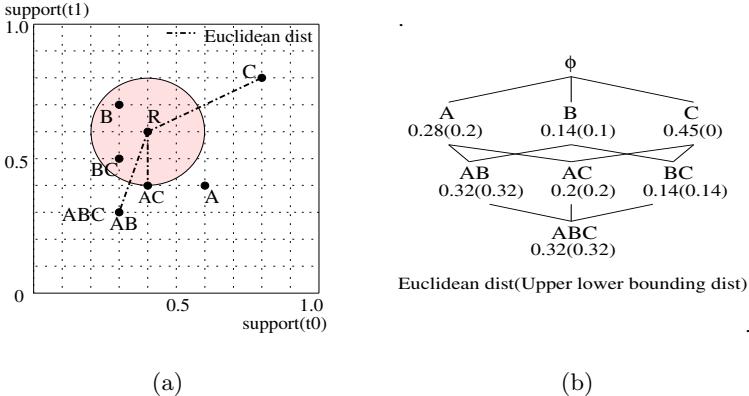


Fig. 5. (a) Non-monotonicity of the Euclidean distance (b) Monotonically non-decreasing property of the upper lower-bounding distance

Lemma 1. For the upper bound support time sequence $\mathbf{U} = < u_1, \dots, u_n >$, lower bound support time sequence $\mathbf{L} = < l_1, \dots, l_n >$ and support time sequence $\mathbf{S} = < s_1, \dots, s_n >$ of an itemset I , and a reference sequence $\mathbf{R} = < r_1, \dots, r_n >$, the lower bounding distance $D_{lb}(\mathbf{R}, \mathbf{U}, \mathbf{L})$ and the true distance $D(\mathbf{R}, \mathbf{S})$ hold the following inequality: $D_{lb}(\mathbf{R}, \mathbf{U}, \mathbf{L}) \leq D(\mathbf{R}, \mathbf{S})$.

For the proof, refer to [42]. Thus, if $D_{lb}(\mathbf{R}, \mathbf{U}, \mathbf{L})$ of an itemset is greater than a given threshold, the true distance $D(\mathbf{R}, \mathbf{S})$ should not satisfy the threshold. Therefore, we know that the itemset will not be in the mining result.

3.3 Monotonicity Property of Upper Lower-Bounding Distance

Next, we explore a scheme to further reduce the itemset search space. The most popular technique to reduce itemset search space in association pattern mining is to use the monotonicity property of support measure [5]. The support values of all supersets of a given itemset are not greater than the support value of that itemset. Thus, if an itemset does not satisfy the support threshold, all supersets of the itemset can be pruned. Unfortunately, our \mathcal{L}_p norms-based interest measure does not show any monotonicity with the size of the itemset. For example, Fig. 5 (a) shows the Euclidean distances between the support time sequences of $\{\{C\}\}$, $\{\{A,C\}\}$ and $\{\{A,B,C\}\}$, \mathbf{S}_C , \mathbf{S}_{AC} and \mathbf{S}_{ABC} , and a reference sequence \mathbf{R} . As can be seen, $D(\mathbf{S}_C, \mathbf{R})=0.45$, $D(\mathbf{S}_{AC}, \mathbf{R})=0.2$ and $D(\mathbf{S}_{ABC}, \mathbf{R}) = 0.32$. Thus, $D(\mathbf{S}_{ABC}, \mathbf{R}) > D(\mathbf{S}_{AC}, \mathbf{R})$ but $D(\mathbf{S}_{AC}, \mathbf{R}) < D(\mathbf{S}_C, \mathbf{R})$. However, we found an interesting property related to our upper lower-bounding distance.

Lemma 2. The upper lower-bounding distance between the (upper bound) support time sequence of an itemset and a reference time sequence is **monotonically non-decreasing** with the size of itemset.

For the proof, refer to [42]. For example, in Fig. 5 (b), $D_{Ulb}(\mathbf{S}_A, \mathbf{R})=0.2$, $D_{Ulb}(\mathbf{S}_B, \mathbf{R})=0.1$ and $D_{Ulb}(\mathbf{S}_{AB}, \mathbf{R})=0.32$. Thus $D_{Ulb}(\mathbf{S}_A, \mathbf{R}) \leq D_{Ulb}(\mathbf{S}_{AB}, \mathbf{R})$ and $D_{Ulb}(\mathbf{S}_B, \mathbf{R}) \leq D_{Ulb}(\mathbf{S}_{AB}, \mathbf{R})$. We can also see that $D_{Ulb}(\mathbf{S}_{AB}, \mathbf{R}) \leq D_{Ulb}(\mathbf{S}_{ABC}, \mathbf{R})$, $D_{Ulb}(\mathbf{S}_{AC}, \mathbf{R}) \leq D_{Ulb}(\mathbf{S}_{ABC}, \mathbf{R})$, and $D_{Ulb}(\mathbf{S}_{BC}, \mathbf{R}) \leq D_{Ulb}(\mathbf{S}_{ABC}, \mathbf{R})$.

3.4 SPAMINE Algorithm

The Similarity-Profiled temporal Association MINing mEthod(SPAMINE) is developed based on the previous algorithm design concept. Algorithm 1 shows the pseudocode of the SPAMINE. Fig. 6 provides an illustration of trace of a SPAMINE execution with the example data in Fig. 2 (a).

Generate the support time sequences of single items and find similar items (Steps 1 - 3): All singletons ($k = 1$) become candidate items(C_1). With reading the entire transaction data, the supports of singletons are computed per each time slot and their support time sequences(S_1) are generated. If the interest measure value of an item (i.e., distance between its support time sequence and a reference sequence) does not exceed a given threshold, the item is added to a result set(R_1). On the fly, if the upper lower-bounding distance of an item satisfies

Inputs:

E : A set of single items.

TD : A time-stamped transaction database

R : A reference sequence

D : A similarity function

θ : A dissimilarity threshold

Output: All itemsets whose support sequences are similar to R under D and θ

Variables :

k : Itemset size

C_k : A set of size k candidate itemsets

U_k : A set of upper bound support sequences of size k itemsets

L_k : A set of lower bound support sequences of size k itemsets

S_k : A set of support sequences of size k itemsets

S : A set of support sequences of all subsets of itemsets

B_k : A set of size k itemsets whose upper lower-bounding distance $\leq \theta$

A_k : A result set of size k itemsets whose true distance $\leq \theta$

Main:

- 1) $C_1 = E$;
- 2) $S_1 = \text{generate_support_sequences}(C_1, TD)$;
- 3) $(A_1, B_1) = \text{find_similar_itemsets}(C_1, S_1, R, D, \theta)$;
- 4) $k = 2$;
- 5) **while** (not empty B_{k-1}) **do**
- 6) $(C_k, U_k, L_k) = \text{generate_candidate_itemsets}(B_{k-1}, S)$;
- 7) $C_k = \text{prune_candidate_itemsets_by_lbd}(C_k, U_k, L_k, R, D, \theta)$;
- 8) $S_k = \text{generate_support_sequences}(C_k, TD)$;
- 9) $(A_k, B_k) = \text{find_similar_itemsets}(C_k, S_k, R, D, \theta)$;
- 10) $S = S \cup S_k$; $k = k + 1$;
- 11) **end**
- 12) **return** $\bigcup(A_1, \dots, A_k)$;

Algorithm 1. SPAMINE algorithm

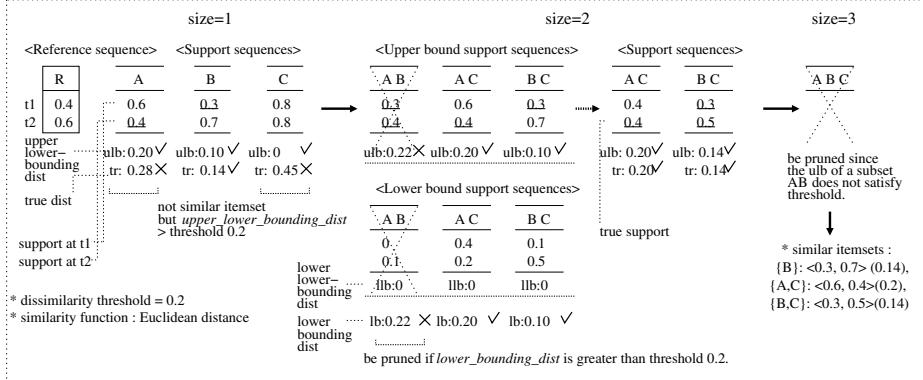


Fig. 6. An illustration of SPAMINE algorithm trace

the dissimilarity threshold, the item is kept to B_1 for generating the next size candidate itemsets. In Fig. 6, only item B is a similar item but items A and C are also kept for generating the next size candidate itemsets.

Generate candidate itemsets and their upper and lower bound support sequences (Step 6): All size k ($k > 1$) candidate itemsets(C_k) are generated using size $k - 1$ itemsets(B_{k-1}) whose upper lower-bounding distances satisfy the dissimilarity threshold. If any subset of size $k - 1$ of the generated itemset is not in the B_{k-1} , the candidate itemset is eliminated according to Lemma 2. The upper and lower bound support sequences of candidate itemsets are generated using Definition 3.

Prune candidate itemsets using their lower bounding distances (Step 7): If the lower bounding distance of the upper and lower bound support sequences of a candidate itemset exceeds the dissimilarity threshold, the candidate itemset is eliminated according to Lemma 1. For example, in Fig. 6, the lower bounding distance of itemset $\{A, B\}$ is 0.22. Because the value is greater than the threshold 0.2, the candidate itemset is pruned.

Scan the transaction data and generate the support time sequences (Step 8): The supports of survived candidates are computed during the scan of the transaction data from time slot t_1 to t_n , and their support time sequences(S_k) are generated.

Find similar itemsets (Step 9): The true distance between the support time sequence of an itemset and the reference sequence is computed. If the value satisfies the threshold, the itemset is included in the result set(R_k). On the fly, if the upper lower-bounding distances of candidate itemsets satisfy the threshold, the itemsets are added to B_k for generating the next size candidate itemsets. The size of examined itemsets is increased to $k = k + 1$. The above procedures(Steps 6-10) are repeated until no itemset in B_k remains.

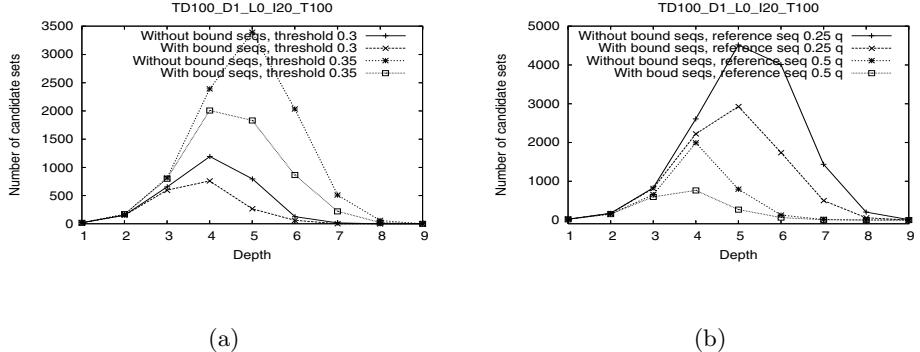


Fig. 7. Effect of lower bounding distance pruning of bounds of support sequence

4 Experimental Evaluation

The proposed algorithm (SPAMINE) was evaluated using synthetic and real datasets. Synthetic datasets were generated with modifying a transaction data generator [5]. In the rest of paper, we use the following parameters to characterize the synthetic datasets we used. TD is the total number of transactions($\times 1,000$), D is the number of transactions per time slot($\times 1,000$), I is the number of distinct items, L is the average size of transactions, and T is the number of time slots. A reference time sequence was generated by choosing randomly a support sequence of an itemset or by selecting a support value near a quartile e.g., 0.25, 0.5, 0.75, of the sorted supports of single items at each time slot. The default reference time sequence was chosen near the 0.5 quartile. For the experiment with real data, we used a Earth climate dataset which includes monthly measurements of various climate variables(e.g., temperature and precipitation) and other related variables(e.g., Net Primary Production). This dataset is non public and was obtained from an Earth science project [1]. Throughout the experiment, we used the normalized Euclidean distance as a similarity function. For the experimental comparison, an alternative algorithm, a sequential method [42] is used. All experiments were performed on a workstation with Intel Xeon 2.8 GHz with 2 Gbytes of memory running the Linux operating system. The following presents the experimental results.

1) *Effect of pruning by bounds of support sequences:* In this experiment, we examined the pruning effect by low bounding distance of the upper and lower bound support sequences. Two versions of the SPAMINE algorithm are used. one uses the bounds of support sequences and the other does not. A synthetic dataset, TD100-D1-L10-I20-T100, is used and the dissimilarity threshold is set to 0.2. Fig. 7 (a) shows the number of candidates which need the database scan to compute their support values. As can be seen, the algorithm using the pruning

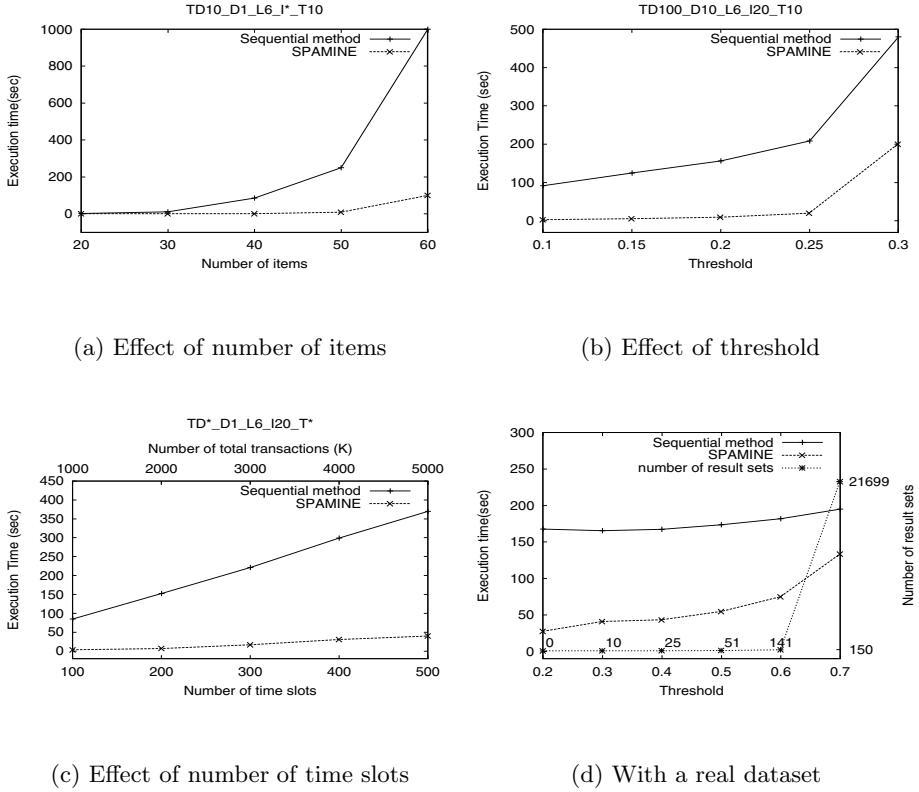


Fig. 8. Experimental Result

scheme based on the bounds of support sequences and the lower bound distance produces fewer candidate itemsets. Fig. 7 (b) shows the results with different reference sequence types.

2) *Effect of number of items:* In this experiment, synthetic datasets of different number of items, TD10_D1_L6_I*_T10, are used. As seen in Fig. 8 (a), SPAMINE showed a similar execution time and received a less effect with the increase of small numbers of items. In contrast, the execution time of the sequential method dramatically increased.

3) *Effect of similarity threshold:* The performance of the two algorithms are examined with different threshold values. The TD10_D1_L10_I20_S10 dataset was used for that. As can be seen in Fig. 8 (b), SPAMINE had overall less execution time than the sequential method.

4) *Effect of number of time slots:* This experiment examined the effect of number of time slots using synthetic datasets, TD*_D1_L6_I20_T*. The number of transactions per time slot was fixed but the total dataset size increased with

the number of time slots. Reference sequences were chosen near the 0.5 quartiles in each dataset and the threshold was 0.1. As seen in Fig. 8 (c), SPAMINE’s execution time increases slowly with increase of number of time slots. In contrast, the sequential method’s execution time rapidly increases.

5) Evaluation with real data: For this experiment, an Earth climate dataset was used. The dataset consists of global snapshots of measurement values for a number of variables (e.g., temperature, precipitation, NPP, CO₂ and Solar). The data was measured at points (grid cells) on latitude-longitude spherical grids of 0.5 degree × 0.5 degree. For our analysis, we used measurement values in the Australia region since the climate phenomena in Australia has been known to be linked to El Niño, the anomalous warming of the eastern tropical region of the Pacific [39]. The total number of grids in the Australia data, i.e., the number of transactions per time slot, was 2827. First, we removed seasonal variation from the time series measurement data using a monthly Z score, i.e., by subtracting off the mean and dividing by the standard deviation. We defined event items based on 4 percentiles from the time series data of each variable(e.g., PRECI-LL, PRECI-L, PRECI-H, PRECI-HH). The total number of items for our analysis was 50. The dataset is available at monthly intervals from 1982 to 1999. We used 214 months of data, i.e., the number of time slots was 214. The total number of transactions was 604,978. For the reference sequence, the sequence of Southern Oscillation Index(SOI) was used, which is one of the indexes related to the El Niño phenomenon. Since the SOI index range is different from the support range, the index values were normalized to the range of 0 to 1 using a min-max normalization method. The transformation of a raw value x was calculated as $(x - x_{min})(1 - 0)/(x_{max} - x_{min}) + 0$, where x_{min} is the minimum value of raw value x , and x_{max} is the maximum value of x . Fig. 8 (d) shows the execution time and number of result sets by different thresholds. The pattern result shows that the prevalence variations of PRECI-L(0.20), CO₂-L(0.21), Solar-H(0.25), NPP-L(0.26), PRECI-L & CO₂-L(0.21), NPP-L & CO₂-L(0.23), PRECI-L & NPP-L(0.27), PRECI-L & NPP-L & CO₂-L(0.29), etc. were very related with the El Niño index sequence, with dissimilarity values of around 0.2.

5 Related Work

Although much work has been done on finding association patterns and similar time series, little attention has been paid to temporal association patterns that can discover similar variance groups over time. The closest related efforts have attempted to capture special temporal regulations of frequent association patterns such as cyclic association rule mining and calendar-based association rule mining [35,37,28,29] in temporal association mining. Özden et al.[35] examined cyclic association rule mining, which detects periodically repetitive patterns of frequent itemsets over time. Cyclic associations can be considered as itemsets that occur in every cycle with no exception. The work of [35] was extended in [37] for relaxed match. The cyclic association rules may not hold on all but most of the time points defined by the temporal patterns. Li et al.[28] explored the

problem of finding frequent itemsets along with calendar-based patterns. The calendar-based patterns are defined with a calendar schema, e.g., (year, month, day). For example, (*,10,31) represents the set of time points each corresponding to the 31st day of October. However, real-life patterns are usually imperfect and may not demonstrate any regular periodicity. In the work of [29], a temporal pattern defines the set of time points where the user expects a discovered itemset to be frequent. However, our temporal patterns are searched with a user defined numeric reference sequence, and consider the prevalence similarity of all possible itemsets not only frequent itemsets.

In temporal pattern mining, sequential patterns mining [9] considers the relative order of the transactions of one customer. In [9], a frequent sequence is defined to consist of frequent itemsets taking place in separate consecutive transactions of the same user. The notion of episodes was introduced in [32]. An episode uses the data model of a sequence of elements, where the inter-element causality happens within a window of a given size. The frequency of an episode is the number of windows that contain the episode. In [11], frequent event patterns are found from time sequences which satisfy a user-specified skeleton. The user-specified skeleton is defined with a reference event and temporal constraints with time granularities. For example, the work can find events which frequently happen within two business days after a reference event, e.g., a rise of the IBM stock price. Recent work has applied mining techniques in a data streaming context. The temporal frequency counting problem for a data stream environment was proposed by [40]. The work is based on the model of inter-transaction association rules [31] for searching associations between itemsets that belong to different transactions performed by the same user in a given time span.

Other studies in temporal data mining have discussed the change of found association rules. Dong et al. [17] presented the problem of mining emerging patterns, which are the itemsets whose supports increase significantly from one dataset to another. The concept of emerging patterns can capture useful contrasts between classes. Ganti et al.[19] presented a framework for measuring difference in two sets of association rules from two datasets. Liu et al.[30] studied the change of fundamental association rules between two time periods using support and confidence. When new transactions are added to the original dataset, the maintenance of discovered association rules with an incremental updating technique was proposed in [15]. In contrast, [9] addressed the problem of monitoring the support and confidence of association rules. First, all frequent rules satisfying a minimum threshold from different time periods are mined and collected into a rule base. Then interesting rules can be queried by specifying shape operators(e.g., ups and downs) in support or confidence over time. On the other hand, online association rule mining was proposed by [24] to give the user the freedom to change the support threshold during the first scan of the transaction sequence.

6 Conclusion

We presented the problem of mining similarity-profiled temporal association patterns. Current similarity model using a \mathcal{L}_p norm-based similarity function is a little rigid in finding similar temporal patterns. It may be interesting to consider a relaxed similarity model to catch temporal patterns which show similar trends but phase shifts in time. For example, the sale of items for clean-up such as chain saws and mops would increase after a storm rather than on the way of the storm. The current framework considers whole-sequence matching for the similar temporal patterns. However, a pattern's similarity may not persist for entire length of the sequence and so may manifest only in some segments. These relaxations present many new challenges for the automatic discovery of all partial temporal patterns based on the similarity constraint. The field of temporal data mining is relatively young and one expects to see many new developments in the near future.

References

1. Discovery of Changes from the Global Carbon Cycle and Climate System Using Data Mining, <http://www.cs.umn.edu/old-ahpcrc/nasa-umn/>
2. NOAAEconomics, <http://www.ncdc.noaa.gov/oa/esb/?goal=climate&file=users/business/>
3. Weather.com, <http://www.weather.com/aboutus/adsales/research.html>
4. After Katrina: Crisis Management, The Only Lifeline Was the Wal-Mart. *FOR-TUNE Magazine*, October 3 (2005)
5. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proc. of the International Conference on Very Large Databases, VLDB (1994)
6. Agrawal, R., Faloutsos, C., Swami, A.: Efficient Similarity Search in Sequence Databases. In: Proc. of the International Conference on Foundations of Data Organization, FODO (1993)
7. Agrawal, R., Lin, K.I., Sawhney, H.S., Shim, K.: Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-series Database. In: Proc. of the International Conference on Very Large Databases (VLDB) Conference (1995)
8. Agrawal, R., Lin, K.I., Sawhney, H.S., Shim, K.: Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-series Database. In: Proc. of the International Conference on Very Large Databases (VLDB) Conference (1995)
9. Agrawal, R., Srikant, R.: Mining Sequential Patterns. In: Proc. of the IEEE International Conference on Data Engineering, ICDE (1995)
10. Bettini, C., Jajodia, S., Wang, X.S.: Time Granularities in Databases, Data Mining and Temporal Reasoning. Springer, Heidelberg (2000)
11. Bettini, C., Wang, X.S., Jajodia, S., Lin, J.: Discovering Frequent Event Patterns with Multiple Granularities in Time Sequences. *IEEE Transactions on Knowledge and Data Engineering* 10(2) (1998)
12. Box, G., Jenkins, G., Reinsel, G.: Time Series Analysis: Forecasting and Control. Prentice-Hall, Englewood Cliffs (1994)
13. Calders, T.: Deducing Bounds on the Frequency of Itemsets. In: Proc. of EDBT Workshop DTDM Database Techniques in Data Mining (2002)
14. Chan, F.K., Fu, A.W.: Efficient Time Series Matching by Wavelets. In: Proc. of International Conference on Data Mining (1999)

15. Cheung, W., Han, J., Ng, V.T., Wong, C.Y.: Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. In: Proc. of the IEEE International Conference on Data Engineering, ICDE (1996)
16. Das, G., Gunopulos, D., Mannila, H.: Finding Similar Time Series. In: Proc. of Principles of Data Mining and Knowledge Discovery, European Symposium (1997)
17. Dong, G., Li, J.: Efficient Mining of Emerging Patterns: Discovering Trends and Differences. In: Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (1999)
18. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast Subsequence Matching in Time-series Database. In: Proc. of the ACM SIGMOD International Conference on Management of Data (1994)
19. Ganti, V., Gehrke, J., Ramakrishnan, R.: A Framework for Measuring Changes in Data Characteristics. In: Proc. of the ACM PODS Conference (1999)
20. Gunopulos, D., Das, G.: Time Series Similarity Measures. Tutorial Notes of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2000)
21. Gunopulos, D., Das, G.: Time Series Similarity Measures and Time Series Indexing. SIGMOD Record 30(2) (2001)
22. Han, J., Fu, Y.: Discovery of Multi-level Association Rules From Large Databases. In: Proc. of the International Conference on Very Large Databases, VLDB (1995)
23. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proc. of the ACM SIGMOD International Conference on Management of Data (2000)
24. Hidber, C.: Online Association Rule Mining. In: Proc. of the ACM SIGMOD International Conference on Management of Data (1998)
25. Hsu, W., Lee, M., Wang, J.: Temporal and Spatio-temporal Data Mini. IGI Publishing (1997)
26. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. Journal of Knowledge and Information Systems 3(3) (2001)
27. Keogh, E., Ratanamahatana, C.A.: Exact Indexing of Dynamic Time Warping. Knowledge and Information Systems 17(3), 358–386 (2005)
28. Li, Y., Ning, P., Wang, X.S., Jajodia, S.: Discovering Calendar-Based Temporal Association Rules. Journal of Data and Knowledge Engineering 15(2) (2003)
29. Li, Y., Zhu, S., Wang, X.S., Jajodia, S.: Looking into the Seeds of Time: Discovering Temporal Patterns in Large Transaction Sets. Journal of Information Sciences 176(8) (2006)
30. Liu, B., Hsu, W., Ma, Y.: Discovering the Set of Fundamental Rule Change. In: Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2001)
31. Lu, H., Han, J., Feng, L.: Stock Movement Prediction and N-Dimensional Inter-Transaction Association Rules. In: Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (1998)
32. Mannila, H., Toivonen, H., Verkamo, A.: Discovering Frequent Episodes in Sequences. In: Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (1995)
33. Mitsa, T.: Temporal Data Mining. Chapman and Hall/CRC (2010)
34. NOAA. El Nino Page, <http://www.elnino.noaa.gov/>
35. Ozden, B., Ramaswamy, S., Silberschatz, A.: Cyclic Association Rules. In: Proc. of the IEEE International Conference on Data Engineering, ICDE (1998)

36. Park, J.S., Chen, M., Yu, P.: An Effective Hashing-based Algorithm for Mining Association Rules. In: Proc. of the ACM SIGMOD International Conference on Management of Data (1995)
37. Ramaswamy, S., Mahajan, S., Silberschatz, A.: On the Discovery of Interesting Patterns in Association Rules. In: Proc. of the International Conference on Very Large Database (VLDB) (1998)
38. Srikant, R., Agrawal, R.: Mining Generalized Association Rules. In: Proc. of the International Conference on Very Large Databases, VLDB (1995)
39. Taylor, G.H.: Impacts of El Nino on Southern Oscillation on the Pacific Northwest, http://www.ocs.orst.edu/reports/enso_pnw.html
40. Teng, W., Chen, M., Yu, P.: A Regression-Based Temporal Pattern Mining Scheme for Data Streams. In: Proc. of the International Conference on Very Large Databases, VLDB (2003)
41. Yi, B.K., Faloutsos, C.: Fast Time Sequence Indexing for Arbitrary \mathcal{L}_p norms. In: Proc. of the International Conference on Very Large Data Bases, VLDB (2000)
42. Yoo, J.S., Shekhar, S.: Similarity-profiled Temporal Association Mining. IEEE Transactions on Knowledge and Data Engineering 21(5), 1147–1161 (2009)

Chapter 4

Bayesian Networks with Imprecise Probabilities: Theory and Application to Classification

G. Corani, A. Antonucci, and M. Zaffalon

IDSIA, Manno, Switzerland

{giorgio,alessandro,zaffalon}@idsia.ch

www.idsia.ch

Abstract. Bayesian networks are powerful probabilistic graphical models for modelling uncertainty. Among others, classification represents an important application: some of the most used classifiers are based on Bayesian networks. Bayesian networks are precise models: exact numeric values should be provided for quantification. This requirement is sometimes too narrow. Sets instead of single distributions can provide a more realistic description in these cases. Bayesian networks can be generalized to cope with sets of distributions. This leads to a novel class of imprecise probabilistic graphical models, called *credal networks*. In particular, classifiers based on Bayesian networks are generalized to so-called *credal classifiers*. Unlike Bayesian classifiers, which always detect a single class as the one maximizing the posterior class probability, a credal classifier may eventually be unable to discriminate a single class. In other words, if the available information is not sufficient, credal classifiers allow for indecision between two or more classes, thus providing a less informative but more robust conclusion than Bayesian classifiers.

Keywords: Credal sets, credal networks, Bayesian networks, classification, credal classifiers, naive Bayes classifier, naive credal classifier, tree-augmented naive Bayes classifier, tree-augmented naive credal classifier.

1 Introduction

Bayesian networks [63] are powerful and widespread tools for modelling uncertainty about a domain. These probabilistic graphical models provide a compact and intuitive quantification of uncertain knowledge. After its specification, a Bayesian network can be queried by appropriate inference algorithms in order to extract probabilistic information about the variables of interest. Among others, *classification* represents an important application of Bayesian networks. Some of the most used classifiers proposed within the Bayesian theory of probability, like the *naive Bayes classifier* (Section 8) and the *tree-augmented naive Bayes classifier* (Section 11) can be regarded as learning/inference algorithms for Bayesian networks with particular topologies.

Bayesian networks are *precise* models, in the sense that exact numeric values should be provided as probabilities needed for the model parameters. This requirement is sometimes too narrow. In fact, there are situations where a single probability distribution cannot properly describe the uncertainty about the state of a variable.¹ In these cases, sets instead of single distributions provide an alternative and more realistic description. E.g., in some cases we may prefer to model our knowledge by interval-valued probabilistic assessments, these corresponding to the specification of the set of distributions compatible with these assessments. Sets of this kind, which are generally required to be closed and convex by some rationality criteria, are called *credal sets* [57]. Approaches where probabilities are quantified in this way are said to be *imprecise* [75].

Bayesian networks can be generalized in order to cope with credal sets. This leads to a novel class of *imprecise* probabilistic graphical models, generalizing Bayesian networks, and called *credal networks* [35]. Expert knowledge is mostly qualitative and it can be therefore naturally described by credal sets instead of single distributions: this makes knowledge-based (or expert) systems represent one of the more natural application of credal networks (e.g., [2,4,5]). But even when the focus is on learning probabilities from data, a credal set may offer a more reliable model of the uncertainty, especially when coping with small or incomplete data sets. Thus, classifiers based on Bayesian networks can be profitably extended to become *credal classifiers* based on credal networks. Unlike Bayesian classifiers, which always detect a single class as the one maximizing the posterior class probability², a credal classifier works with sets of distributions and may eventually be unable to discriminate a single class as that with highest probability. In other words, if the available information is not sufficient to identify a single class, credal classifiers allow for *indecision* between two or more classes, this representing a less informative but more robust conclusion than Bayesian classifiers.

This chapter describes the main tools of the theory of credal networks in Sections 2–6; it starts by reviewing the general theory of Bayesian networks (Section 2) and the fundamental concept of credal set (Section 3), to then illustrate the design and the quantification of the network (Section 4), the query through specific inference algorithms (Section 5), and an environmental application (Section 6). In the second part of the chapter (Sections 7–14) we show how credal networks can be used for classification. In particular, we show how the naive Bayes classifier and the Tree-Augmented Naive (TAN) have been extended to deal with imprecise probabilities, yielding respectively the Naive Credal Classifier (NCC) and the credal TAN (Sections 8–12); this part includes experimental results in texture recognition (Section 9.1) and a discussion of the metrics to evaluate credal classifiers empirically (Section 10). Finally, we review some further credal classifiers (Section 13) and the available software (Section 14).

¹ As an example, a condition of *ignorance* about the state of a variable is generally modelled by a uniform distribution, while a more robust model of this ignorance is the whole set of distributions we can specify over this variable.

² In the Bayesian framework, the only exception to that is when a condition of *indifference* among two or more classes appears. This corresponds to the situation where the classifier assigns the highest probability to more than a class.

2 Bayesian Networks

We deal with multivariate probabilistic models defined over a collection of variables³ $\mathbf{X} := \{X_0, X_1, \dots, X_k\}$. In particular, we consider *graphical* probabilistic models, in the sense that we assume a one-to-one correspondence between the elements of \mathbf{X} and the nodes of a *directed acyclic graph* (DAG) \mathcal{G} .⁴ Accordingly, in the following we use the terms *node* and *variable* interchangeably. The *parents* of a variable are the variables corresponding to its immediate predecessors according to \mathcal{G} . Notation Π_i is used to denote the parents of X_i , for each $X_i \in \mathbf{X}$. Similarly, we define the *children* and, by iterating this relation, the *descendants* of any variable. The graph \mathcal{G} should be intended as a compact description of the conditional independence relations occurring among the variables in \mathbf{X} . This is achieved by means of the *Markov condition* for directed graphs: *every variable is independent of its non-descendant non-parents conditional on its parents*. These conditional independence relations can be used to specify a probabilistic model over the whole set of variables \mathbf{X} by means of *local* probabilistic models, involving only smaller subsets of \mathbf{X} (namely, a variable together with its parents, for each submodel). This feature, characterizing directed probabilistic graphical models, will be shared by both the Bayesian networks reviewed here and the *credal networks* introduced in Section 4.

For each $X_i \in \mathbf{X}$, the set of its possible values is denoted as Ω_{X_i} . Here we focus on the case of categorical variables, i.e., we assume $|\Omega_{X_i}| < +\infty$ for each $X_i \in \mathbf{X}$. Similarly, notation Ω_{Π_i} is used for the set of possible values of the joint variable Π_i , corresponding to the parents of X_i . We denote by $P(X_i)$ a probability mass function over X_i , and by $P(x_i)$ the probability that $X_i = x_i$, where x_i is a generic element of Ω_{X_i} . Finally, in the special case of a binary variable X_i , we set $\Omega_{X_i} := \{x_i, \neg x_i\}$, while the (vertical) array notation is used to enumerate the values of a probability mass functions, i.e., $P(X_i) = [\dots, P(x_i), \dots]^T$. This formalism is sufficient to introduce the definition of Bayesian network, which is reviewed here below. For a deeper analysis of this topic, we point the reader to Pearl's classical textbook [63].

Definition 1. A Bayesian network over \mathbf{X} is a pair $\langle \mathcal{G}, \mathbb{P} \rangle$ such that \mathbb{P} is a set of conditional mass functions $P(X_i | \pi_i)$, one for each $X_i \in \mathbf{X}$ and $\pi_i \in \Omega_{\Pi_i}$.

As noted in the previous section, we assume the *Markov condition* to make \mathcal{G} represent probabilistic independence relations between the variables in \mathbf{X} . Hence, the conditional probability mass functions associated to the specification of the Bayesian network can be employed to specify a joint mass function $P(\mathbf{X})$ by means of the following factorization formula:

$$P(\mathbf{x}) = \prod_{i=0}^k P(x_i | \pi_i), \quad (1)$$

³ In the sections about classification, the first variable in this collection will be identified with the class and the remaining with the attributes. Notation $\mathbf{X} := (C, A_1, \dots, A_k)$ will be therefore preferred.

⁴ A directed graph is *acyclic* if it does not contains any directed loop.

for each $\mathbf{x} \in \Omega_{\mathbf{X}}$, where for each $i = 0, 1, \dots, k$ the values (x_i, π_i) are those consistent with \mathbf{x} .

Bayesian networks provide therefore a specification of a joint probability mass function, describing the probabilistic relations among the whole set of variables. The specification is compact in the sense that only conditional probability mass functions for the variables conditional on (any possible value of) the parents should be assessed. Once a Bayesian network has been specified, a typical task we might consider consists in querying the model to gather probabilistic information about the state of a variable given evidence about the states of some others. This inferential task is called *updating* and it corresponds to the computation of the posterior beliefs about a queried variable X_q , given the available evidence $X_E = x_E$.⁵

$$P(x_q | x_E) = \frac{\sum_{x_M \in \Omega_{X_M}} \prod_{i=0}^k P(x_i | \pi_i)}{\sum_{x_M \in \Omega_{X_M}, x_q \in \Omega_{X_q}} \prod_{i=0}^k P(x_i | \pi_i)}, \quad (2)$$

where $X_M := \mathbf{X} \setminus (\{X_q\} \cup X_E)$ and the values of x_i and π_i are those consistent with $\mathbf{x} = (x_q, x_M, x_E)$. The variables in X_M are marginalized out of Equation (2) because their values are not available or, in other words, they are *missing*, and this missingness is independent of the actual values of the variables. This represents a special case of the *missing at random* assumption for missing data, which will be discussed in Section 5.3 and Section 9.2.

The evaluation of Equation (2) is an NP-hard task [23], but in the special case of *polytrees*, Pearl's local propagation scheme allows for efficient updating [63]. A polytree is a Bayesian network based on a *singly connected* directed acyclic graph, which is a graph that does not contain any undirected loop.

Bayesian networks are powerful means to model uncertain knowledge in many situations. Yet, the specification of a model of this kind requires the *precise* assessments of the conditional probabilities associated to every variable for any possible value of the parents. Some authors claim this requirement is too strong [75]: an *imprecise* probabilistic evaluation corresponding for instance to an interval and in general to a set of possible estimates would represent a more realistic model of the uncertainty. Thus, we consider a generalization of Bayesian networks in which closed convex sets of probability mass functions instead of single mass functions are provided.

3 Credal Sets

Walley's behavioral theory of *imprecise probabilities* [75] provides a complete probabilistic theory, based on *coherent lower previsions*, that generalizes to imprecision de Finetti's classical theory [43]. A coherent lower revision can be equivalently expressed by (the lower envelope of) a closed convex set of linear previsions, which are expectations with respect to a finitely additive probability, and hence in one-to-one relationship with mass functions in the case of finite supports. Accordingly, we formalize our imprecise probabilistic approaches in terms of closed convex sets of probability mass functions as stated in the following section.

⁵ A notation with uppercase subscripts (like X_E) is employed to denote vectors (and sets) of variables in \mathbf{X} .

3.1 Definition

Following Levi [57], we call *credal set* a closed convex set of probability mass functions. A credal set for a random variable X is denoted by $K(X)$. We follow Cozman [35] in considering only *finitely generated* credal sets, i.e., obtained as the convex hull of a finite number of mass functions for a certain variable. Geometrically, a credal set of this kind is a *polytope*. Such credal set contains an infinite number of mass functions, but only a finite number of *extreme mass functions*: those corresponding to the *vertices* of the polytope, which are in general a subset of the generating mass functions. In the following, the set of vertices of $K(X)$ is denoted as $\text{ext}[K(X)]$. Enumerating the elements of $\text{ext}[K(X)]$ is then a way to describe a credal set. It is easy to verify that credal sets over binary variables cannot have more than two vertices, while no bounds characterize the possible number of vertices of credal sets over variables with three or more states.

Given a non-empty subset $\Omega_X^* \subseteq \Omega_X$, an important credal set for our purposes is the *vacuous credal set* relative to Ω_X^* , i.e., the set of all the mass functions for X assigning probability one to Ω_X^* . We denote this set by $K_{\Omega_X^*}(X)$. The vertices of $K_{\Omega_X^*}(X)$ are the $|\Omega_X^*|$ degenerate mass functions assigning probability one to the single elements of Ω_X^* .

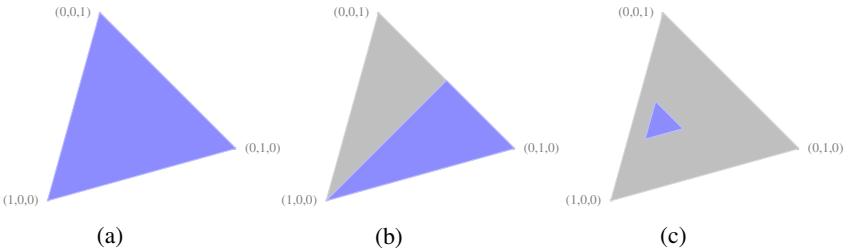


Fig. 1. Geometric representation of credal sets over a ternary variable X (i.e., $\Omega_X = \{x', x'', x'''\}$). The representation is in a three-dimensional space with coordinates $[P(x'), P(x''), P(x''')]^T$. The blue polytopes represent respectively: (a) the vacuous credal set $K_{\Omega_X^*}(X)$; (b) the credal set defined by constraint $P(x''') > P(x'')$; (c) a credal set $K(X)$ such that $\text{ext}[K(X)] = \{[.1,.3,.6]^T, [.3,.3,.4]^T, [.1,.5,.4]^T\}$.

3.2 Basic Operations with Credal Sets

Given $\tilde{x} \in \Omega_X$, the lower probability for \tilde{x} according to credal set $K(X)$ is

$$\underline{P}^K(\tilde{x}) := \min_{P(X) \in K(X)} P(\tilde{x}). \quad (3)$$

If there are no ambiguities about the credal set considered in Equation (3), the superscript K is removed and the corresponding lower probability is simply denoted as $\underline{P}(\tilde{x})$. Walley shows that inferences based on a credal set are equivalent to those based only on its vertices [75]. This makes optimization in Equation (3) a combinatorial task. As an example, for the credal set in Figure 1(c), we have $\underline{P}(x') = .1$, $\underline{P}(x'') = .3$ and $\underline{P}(x''') = .4$.

By simply replacing the minimum with the maximum in Equation (3), we can define the upper probability \bar{P} . Lower/upper probabilities for any event (including conditional events) in Ω_X can be similarly considered. The *conjugacy*⁶ $\bar{P}(\tilde{x}) = 1 - \underline{P}(\Omega_X \setminus \{\tilde{x}\})$ holds, and makes it possible to focus our attention on lower probabilities. Lower/upper expectations can be also considered when coping with generic functions of variable X .

Let us also describe how the basic operations of *marginalization* and *conditioning* can be extended from probability mass functions to credal sets. Given a joint credal set $K(X, Y)$, its marginal over X is denoted by $K(X)$ and is obtained by the convex hull of the collection of mass functions $P(X)$, obtained marginalizing out Y from $P(X, Y)$, for each $P(X, Y) \in K(X, Y)$. In practical situations, instead of considering all the joint probability mass functions of $K(X, Y)$, marginalization can be obtained by considering only the vertices, and then taking the convex hull, i.e.,

$$K(X) = \text{CH} \left\{ P(X) : P(x) = \sum_{y \in \Omega_Y} P(x, y), \forall x \in \Omega_X, \forall P(X, Y) \in \text{ext}[K(X, Y)] \right\}, \quad (4)$$

where CH denotes the convex hull operator. Concerning *conditioning* with credal sets, we simply perform elements-wise application of Bayes' rule. The conditional credal set is the union of all the conditional mass functions. As in the case of marginalization, the practical computation of a conditional credal set from a joint can be obtained by considering only the vertices of the joint and then taking the convex hull. An expression analogous to that in Equation (4) can be written to compute the conditional credal set $K(X|Y = y)$ from $K(X, Y)$. Note that, in order to apply Bayes' rule, we should assume non-zero probability for the conditioning event ($Y = y$). This corresponds to having $P(y) > 0$ for each $P(Y) \in K(X)$ (or equivalently for each $P(Y) \in \text{ext}[K(Y)]$), and hence $\underline{P}(y) > 0$. When this condition is not satisfied, other conditioning techniques can be considered. We point the reader to [75, App. J] for a discussion on this issue.

Finally, let us discuss how independence can be intended when knowledge is described by credal sets. In fact, the standard notion of independence (or *stochastic independence*) among two variables X and Y , as adopted within the Bayesian framework, states that X and Y are independent if their joint probability mass function $P(X, Y)$ factorizes, i.e., $P(x, y) = P(x) \cdot P(y)$, for each $x \in \Omega_X$ and $y \in \Omega_Y$. But what should we assume if the knowledge about the two variables is described by a set $K(X, Y)$ instead of a single joint mass function $P(X, Y)$? A possible answer is provided by the notion of *strong independence*: X and Y are strongly independent if they are stochastically independent for each $P(X, Y) \in \text{ext}[K(X, Y)]$. Conditional independence is similarly defined. In the above definition we replace $P(X, Y)$ with $P(X, Y|z)$ and $K(X, Y)$ with $K(X, Y|z)$, and then, if the relation is satisfied for each $z \in \Omega_Z$, we say that X and Y are strongly independent given Z . Strong independence is not the only concept of independence proposed for credal sets. We point the reader to [32] for an overview and [22] for recent developments about other notions of independence in the imprecise-probabilistic framework.

⁶ We use the same notation for the subsets of the possibility space and the corresponding indicator functions. Accordingly, we can regard set $\Omega_X \setminus \{\tilde{x}\}$ even as function of X returning one when $X \neq \tilde{x}$ and zero otherwise.

3.3 Credal Sets from Probability Intervals

According to the discussion in Section 3.1, a credal set can be specified by an explicit enumeration of its (extreme) probability mass functions. Alternatively, we can consider a set of *probability intervals* over Ω_X :

$$\mathbb{I}_X = \{\mathbb{I}_x : \mathbb{I}_x = [l_x, u_x], 0 \leq l_x \leq u_x \leq 1, x \in \Omega_X\}, \quad (5)$$

The set of intervals can be then used as a set of (linear) constraints to specify the following credal set:

$$K(X) = \left\{ P(X) : P(x) \in \mathbb{I}_x, x \in \Omega_X, \sum_{x \in \Omega_X} P(x) = 1 \right\}. \quad (6)$$

Not all the credal sets can be obtained from a set of probability intervals as in Equation (6), but intervals are often a convenient tool to adopt. \mathbb{I}_X is said to *avoid sure loss* if the corresponding credal set is not empty and to be *coherent* (or *reachable*) if $u_{x'} + \sum_{x \in \Omega_X, x \neq x'} l_x \leq 1 \leq l_{x'} + \sum_{x \in \Omega_X, x \neq x'} u_x$, for all $x \in \Omega_X$. \mathbb{I}_X is coherent if and only if the intervals are tight, i.e., for each lower or upper bound in \mathbb{I}_X there is a mass function in the credal set at which the bound is attained [75,14]. Note that for reachable sets of probability intervals, $P(x) = l_x$ and $\bar{P}(x) = u_x$, for each $x \in \Omega_X$. As an example, the credal set in Figure 1(c) is the one corresponding to the reachable set of probability intervals with $\mathbb{I}_{x'} = [.1, .3]$, $\mathbb{I}_{x''} = [.3, .5]$ and $\mathbb{I}_{x'''} = [.4, .6]$. Standard algorithms can compute the vertices of a credal set for which a probability interval has been provided [9]. However, the resulting number of vertices is exponential in the size of the possibility space [71].

3.4 Learning Credal Sets from Data

Probability intervals, and hence credal sets, can be inferred from data by the *imprecise Dirichlet model*, a generalization of Bayesian learning from i.i.d. multinomial data based on imprecise-probability modeling of prior ignorance. The bounds for the probability that $X = x$ are given by

$$\mathbb{I}_x = \left[\frac{n(x)}{s + \sum_{x \in \Omega_X} n(x)}, \frac{s + n(x)}{s + \sum_{x \in \Omega_X} n(x)} \right], \quad (7)$$

where $n(x)$ counts the number of instances in the data set in which $X = x$, and s is a hyperparameter that expresses the degree of caution of inferences, usually chosen in the interval $[1, 2]$ (see [76] for details and [10] for a discussion on this choice). To support this interpretation of s , note that if $s = 0$, the credal set associated through Equation (6) to the probability intervals in Equation (7) collapses to a “precise” credal set made of a single extreme point, corresponding to the *maximum likelihood estimator*. On the other side, if $s \rightarrow \infty$, the corresponding credal set tends to the vacuous credal set $K_{\Omega_X}(X)$. The probability intervals as in Equation (7) are always reachable. As an example, the credal set in Figure 1(c) can be learned through Equation (7) from a complete dataset about X , with counts $n(x') = 1$, $n(x'') = 3$, $n(x''') = 4$ and $s = 2$. Unlike this example,

there are reachable sets of probability intervals that cannot be regarded as the output of Equation (7) (no matter which are the counts in the data set).

Although in this chapter we only consider the imprecise Dirichlet model, other methods have been also proposed in the literature for learning credal sets from multinomial data (see for instance [20] for an alternative approach and a comparison).

4 Credal Networks

In the previous section we presented credal sets as a more general and expressive model of uncertainty with respect to single probability mass functions. This makes it possible to generalize Bayesian networks to imprecise probabilities. Here we report the basics of the theory for this class of models. We point the reader to [35] for an overview of these models, and to [64] for a tutorial on this topic.

4.1 Credal Network Definition and Strong Extension

The extension of Bayesian networks to deal with imprecision in probability is achieved by means of the notion of credal set. The idea is simple: to replace each conditional probability mass function in Definition 1 with a conditional credal set. This leads to the following definition.

Definition 2. *A credal network over \mathbf{X} is a pair $\langle \mathcal{G}, \mathbb{K} \rangle$, where \mathbb{K} is a set of conditional credal sets $K(X_i|\pi_i)$, one for each $X_i \in \mathbf{X}$ and $\pi_i \in \Omega_{\Pi_i}$.*

In the same way as Bayesian networks specify a (joint) probability mass function over their whole set of variables, credal networks, as introduced in Definition 2, can be used to specify a (joint) credal set over the whole set of variables. According to [35], this corresponds to the *strong extension* $K(\mathbf{X})$ of a credal network, which is defined as the convex hull of the joint mass functions $P(\mathbf{X})$, with, for each $\mathbf{x} \in \Omega_{\mathbf{X}}$:

$$P(\mathbf{x}) = \prod_{i=0}^k P(x_i|\pi_i), \quad \begin{aligned} &P(X_i|\pi_i) \in K(X_i|\pi_i), \\ &\text{for each } X_i \in \mathbf{X}, \pi_i \in \Pi_i. \end{aligned} \quad (8)$$

Here $K(X_i|\pi_i)$ can be equivalently replaced by $\text{ext}[K(X_i|\pi_i)]$ according to the following proposition [8].

Proposition 1. *Let $\{P_j(\mathbf{X})\}_{j=1}^v = \text{ext}[K(\mathbf{X})]$, where $K(\mathbf{X})$ is the strong extension of a credal network $\langle \mathcal{G}, \mathbb{K} \rangle$ are joint mass functions obtained by the product of vertices of the conditional credal sets, i.e., for each $\mathbf{x} \in \Omega_{\mathbf{X}}$:*

$$P_j(\mathbf{x}) = \prod_{i=0}^k P_j(x_i|\pi_i), \quad (9)$$

for each $j=1, \dots, v$, where, for each $i=0, \dots, k$ and $\pi_i \in \Omega_{\Pi_i}$, $P_j(X_i|\pi_i) \in \text{ext}[K(X_i|\pi_i)]$.

According to Proposition 1, we have that the vertices of the strong extension of a credal network can be obtained by combining the vertices of the conditional credal sets involved in the definition of credal network. Note that this makes the number of vertices of the strong extension exponential in the input size.

Example 1 (A simple credal network). Consider a credal network associated to the graph in Figure 2. According to Definition 2, the specification requires the assessment of the (unconditional) credal set $K(X_0)$, and two conditional credal sets (one for each value of parent X_0) for X_1 and X_2 . Note also that, according to Proposition 1, the vertices of the strong extension $K(X_0, X_1, X_2)$ cannot be more than 2⁵.

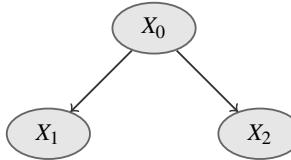


Fig. 2. A credal network over three binary variables. Concerning quantification, we set $\text{ext}[K(X_0)] = \{[.2, .8]^T, [.5, .5]^T\}$, $\text{ext}[K(X_1|x_0)] = \{[.3, .7]^T, [.4, .6]^T\}$, $\text{ext}[K(X_1|\neg x_0)] = \{[.1, .9]^T, [.2, .8]^T\}$, $\text{ext}[K(X_2|x_0)] = \{[.5, .5]^T, [.6, .4]^T\}$, $\text{ext}[K(X_2|\neg x_0)] = \{[.7, .3]^T, [.8, .2]^T\}$.

The key for the decomposition, as in Equation (1), of the joint probability mass function associated to a Bayesian network are the *stochastic* conditional independence relations outlined by the graph underlying the network according to the Markov condition. Similarly, the decomposition characterizing the strong extension of a credal network follows from the *strong* conditional independence relations associated to the graph. Other joint credal sets, alternative to the strong extension, might correspond to different notions of independence adopted in the semantic of the Markov condition. We point the reader to [21], for an example of credal networks based on a different notion of independence.

4.2 Non-separately Specified Credal Networks

In the definition of strong extension as reported in Equation (8), each conditional probability mass function is free to vary in its conditional credal set independently of the others. In order to emphasize this feature, credal networks of this kind are said to be with *separately specified credal sets*, or simply separately specified credal networks.

Separately specified credal networks are the most commonly used type of credal network, but it is possible to consider credal networks whose strong extension cannot be formulated as in Equation (8). This corresponds to having relationships between the different specifications of the conditional credal sets, which means that the possible values for a given conditional mass function can be affected by the values assigned to some other conditional mass functions. A credal network of this kind is called *non-separately specified*.

Some authors considered so-called *extensive* specifications of credal networks [66], where instead of a separate specification for each conditional mass function associated to X_i , the *probability table* $P(X_i|\Pi_i)$, i.e., a function of both X_i and Π_i , is defined to belong to a finite set of tables. This corresponds to assume constraint between the specification of the conditional credal sets $K(X_i|\pi_i)$ for the different values of $\pi_i \in \Omega_{\Pi_i}$. The

strong extension of an extensive credal network is obtained as in Equation (8), by simply replacing the separate requirements for each single conditional mass function with extensive requirements about the tables which take values in the corresponding finite set (and then taking the convex hull).

Example 2 (Extensive specification of a credal network). Consider the credal network defined in Example 1 over the graph in Figure 2. Keep the same specification of the conditional credal sets, but this time use the following (extensive) constraints: when the first vertex of $K(X_1|x_0)$ is chosen, the first vertex of $K(X_1|\neg x_0)$ has to be chosen too; similarly for the second vertex of $K(X_1|x_0)$ and for variable X_2 . This corresponds to assume the following possible values for the conditional probability tables:

$$P(X_1|X_0) \in \left\{ \begin{bmatrix} .3 & .1 \\ .7 & .9 \end{bmatrix}, \begin{bmatrix} .4 & .2 \\ .6 & .8 \end{bmatrix} \right\} \quad P(X_2|X_0) \in \left\{ \begin{bmatrix} .5 & .7 \\ .5 & .3 \end{bmatrix}, \begin{bmatrix} .6 & .8 \\ .4 & .2 \end{bmatrix} \right\}. \quad (10)$$

Extensive specifications are not the only kind of non-separate specification we can consider for credal networks. In fact, we can also consider constraints between the specification of conditional credal sets corresponding to different variables. This is a typical situation when the quantification of the conditional credal sets in a credal network is obtained from a data set. A simple example is illustrated below.

Example 3 (Learning from incomplete data). Given three binary variables X_0 , X_1 and X_2 associated to the graph in Figure 3, we want to learn the model probabilities from the incomplete data set in Table 1, assuming no information about the process making the observation of X_1 missing in the last instance of the data set. A possible approach is to learn two distinct probabilities from the two complete data sets corresponding to the possible values of the missing observation,⁷ and use them to specify the vertices of the conditional credal sets of a credal network.



Fig. 3. The graph considered in Example 3.

Table 1. A data set about three binary variables; “*” denotes a missing observation

X_0	X_1	X_2
x_0	x_1	x_2
$\neg x_0$	$\neg x_1$	x_2
x_0	x_1	$\neg x_2$
x_0	*	x_2

⁷ The rationale of considering alternative complete data sets in order to conservatively deal with missing data will be better detailed in Section 5.3.

To make things simple we compute the probabilities for the joint states by means of the relative frequencies in the complete data sets. Let $P_1(X_0, X_1, X_2)$ and $P_2(X_0, X_1, X_2)$ be the joint mass functions obtained in this way, which define the same conditional mass functions for

$$\begin{aligned} P_1(x_0) &= P_2(x_0) = \frac{3}{4} \\ P_1(x_1 | \neg x_0) &= P_2(x_1 | \neg x_0) = 0 \\ P_1(x_2 | \neg x_1) &= P_2(x_2 | \neg x_1) = 1; \end{aligned}$$

and different conditional mass functions for

$$\begin{aligned} P_1(x_1 | x_0) &= 1 & P_2(x_1 | x_0) &= \frac{2}{3} \\ P_1(x_2 | x_1) &= \frac{2}{3} & P_2(x_2 | x_1) &= \frac{1}{2}. \end{aligned} \tag{11}$$

We have therefore obtained two, partially distinct, Bayesian network specifications over the graph in Figure 3. The conditional probability mass functions of these networks are the vertices of the conditional credal sets for the credal network we consider. Such a credal network is non-separately specified. To see that, just note that if the credal network would be separately specified the values $P(x_1 | x_0) = 1$ and $P(x_2 | x_1) = \frac{1}{2}$ could be regarded as a possible instantiation of the conditional probabilities, despite the fact that there are no complete data sets leading to this combination of values.

Although their importance in modelling different problems, non-separate credal networks have received relatively little attention in the literature. Most of the algorithms for credal networks inference are in fact designed for separately specified credal networks. However, two important exceptions are two credal classifiers which we present later: the naive credal classifier (Section 9) and the credal TAN (Section 12).

Furthermore, in a recent work [8] it has been shown that non-separate credal networks can be equivalently described as separate credal networks augmented by a number of auxiliary parents nodes enumerating only the possible combinations for the constrained specifications of the conditional credal sets. This can be described by means of the two following examples.

Example 4 (Separate specification of non-separate credal networks). Consider the extensive credal network in Example 2. Nodes X_1 and X_2 are characterized by an extensive specification. Thus we add to the model two auxiliary variables X_3 and X_4 , that become parents of X_1 and X_2 respectively. The resulting graph is that in Figure 4(a). Each auxiliary node should index the tables in the specification of its children. In Equation (10) we have two tables for each node. Thus, we assume nodes X_3 and X_4 to be binary, and we redefine the following quantification for nodes X_1 and X_2 : $P(X_1 | X_0, x_3) = P_1(X_1 | X_3)$ and $P(X_1 | X_0, \neg x_3) = P_2(X_1 | X_3)$, where P_1 and P_2 are the two tables in the specification. We similarly proceed for X_2 . Finally, regarding nodes X_2 and X_3 , we set a vacuous specification, i.e., $K(X_2) := K_{\Omega_{X_2}}(X_2)$ and similarly for X_3 . Note that this credal network is separately specified. Let $K(X_0, X_1, X_2, X_3, X_4)$ denote the strong extension of this network, and $K(X_0, X_1, X_2)$ the joint credal set obtained by marginalizing out X_3 and X_4 . The result in [8] states that $K(X_0, X_1, X_2)$ coincides with the strong extension of the extensive credal network of Example 2.

We similarly proceed for the credal network in Example 3. The constraints between P_1 and P_2 in Equation (11) correspond to a non-separate specification of the values

of the conditional probabilities of X_1 and X_2 . As in the previous case, we add to the graph in Figure 3 an auxiliary node X_3 , which is a parent of both X_1 and X_2 , and for the quantification we proceed as in the previous example. This leads to the credal network in Figure 4 (b).

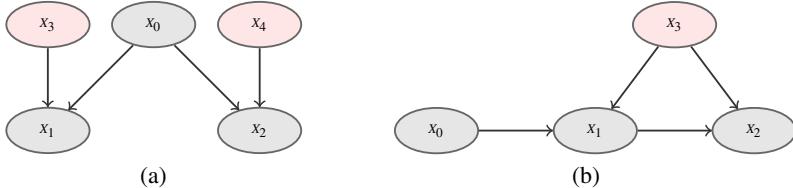


Fig. 4. Modelling non-separately specific conditional credal sets with control nodes (in pink)

This procedure can be easily applied to any non-separate specification of a credal network. We point the reader to [8] for details.

5 Computing with Credal Networks

5.1 Credal Networks Updating

By perfect analogy with what we have done for Bayesian networks in Section 2, we can query a credal network in order to gather probabilistic information about a variable given evidence about some other variables. This task is still called *updating* and consists in the computation of the posterior probability $\underline{P}(x_q|x_E)$ with respect to the network strong extension $K(\mathbf{X})$. Equation (2) generalizes as follows:

$$\underline{P}(x_q|x_E) = \min_{j=1,\dots,v} \frac{\sum_{x_M} \prod_{i=0}^k P_j(x_i|\pi_i)}{\sum_{x_M,x_q} \prod_{i=0}^k P_j(x_i|\pi_i)}, \quad (12)$$

where $\{P_j(\mathbf{X})\}_{j=1}^v$ are the vertices of the strong extension. A similar expression with a maximum replacing the minimum defines upper probabilities $\overline{P}(x_q|x_E)$. Note that, according to Proposition 1, for separately specified credal networks, the number v of vertices of the strong extension is exponential in the input size. Thus, Equation (12) cannot be solved by exhaustive iteration of updating algorithms for Bayesian networks. In fact, exact updating displays higher complexity than Bayesian networks: credal networks updating is NP-complete for polytrees⁸, and NP^{PP}-complete for general credal networks [37]. Nevertheless, a number of exact and approximate algorithm for credal networks updating has been developed. A summary about the state of the art in this field is reported in Section 5.2.

⁸ We extend to credal networks the notion of polytree introduced for Bayesian networks in Section 2.

Algorithms of this kind can be used to compute, given the available evidence x_E , the lower and upper probabilities for the different outcomes of the queried variable X_q , i.e., the set of probability intervals $\{[\underline{P}(x_q|x_E), \bar{P}(x_q|x_E)]\}_{x_q \in \Omega_{X_q}}$. In order to identify the most probable outcome for X_q , a simple *interval dominance* criterion can be adopted. The idea is to *reject* a value of X_q if its upper probability is smaller than the lower probability of some other outcome. Clearly, this criterion is not always intended to return a single outcome as the most probable for X_q . In general, after updating, the posterior knowledge about the state of X_q is described by the set $\Omega_{X_q}^* \subseteq \Omega_{X_q}$, defined as follows:

$$\Omega_{X_q}^* := \{x_q \in \Omega_{X_q} : \nexists x'_q \in \Omega_{X_q} \text{ s.t. } \bar{P}(x_q|x_E) < \underline{P}(x'_q|x_E)\}. \quad (13)$$

Criteria other than interval dominance have been proposed in the literature and formalized in the more general framework of decision making with imprecise probabilities [72]. Most of these criteria require the availability of the *posterior credal set*:

$$K(X_q|x_E) = \text{CH} \left\{ P_j(X_q|x_E) \right\}_{j=1}^v. \quad (14)$$

As an example, the set of non-dominated outcomes $\Omega_{X_q}^{**}$ according to the *maximality* criterion [75] is obtained by rejecting the outcomes whose probabilities are dominated by those of some other outcome, for any distribution in the posterior credal set in Equation (14), i.e.,

$$\Omega_{X_q}^{**} := \{x_q \in \Omega_{X_q} : \nexists x'_q \in \Omega_{X_q} \text{ s.t. } P(x_q|x_E) < P(x'_q|x_E) \forall P(X_q|x_E) \in \text{ext}[K(X_q|x_E)]\}. \quad (15)$$

Maximality is more informative than interval dominance, i.e., $\Omega_{X_q}^{**} \subseteq \Omega_{X_q}^*$. Yet, most of the algorithms for credal networks only returns the posterior probabilities as in Equation (12), while the posterior credal set as in Equation (14) is needed by maximality. Notable exceptions are the models considered in Section 9 and Section 12, for which the computation of the set as in Equation (15) can be performed without explicit evaluation of the posterior credal set. In other cases, a procedure to obtain an (outer) approximation of the credal set in Equation (14) can be used [3].

5.2 Algorithms for Credal Networks Updating

Despite the hardness of the problem, a number of algorithms for exact updating of credal networks have been proposed. Most of these methods generalize existing techniques for Bayesian networks. Regarding Pearl's algorithm for efficient updating on polytree-shaped Bayesian networks [63], a direct extension to credal networks is not possible. Pearl's propagation scheme computes the joint probabilities $P(x_q, x_E)$ for each $x_q \in \Omega_{X_q}$; the conditional probabilities associated to $P(X_q|x_E)$ are then obtained using the normalization of this mass function. Such approach cannot be easily extended to credal networks, because $\underline{P}(X_q|x_E)$ and $\bar{P}(X_q|x_E)$ are not normalized in general. A remarkable exception is the case of binary credal networks, i.e., models for which all the variables are binary. The reason is that a credal set for a binary variable has at most two vertices and can therefore be identified with an interval. This enables an efficient

extension of Pearl’s propagation scheme. The result is an exact algorithm for polytree-shaped binary separately specified credal networks, called *2-Updating* (or simply 2U), whose computational complexity is linear in the input size.

Another approach to exact inference is based on a generalization of the *variable elimination* techniques for Bayesian networks. In the credal case, this corresponds to a *symbolic* variable elimination, where each elimination step defines a *multilinear* constraint among the different conditional probabilities where the variable to be eliminated appears. Overall, this corresponds to a mapping between credal networks updating and *multilinear programming* [14]. Similarly, a mapping with an integer linear programming problem can be achieved [13]. Other exact inference algorithms examine potential vertices of the strong extension according to different strategies in order to produce the required lower/upper values [15,35,66,67].

Concerning approximate inference, *loopy propagation* is a popular technique that applies Pearl’s propagation to multiply connected Bayesian networks [61]: propagation is iterated until probabilities converge or for a fixed number of iterations. In [53], Ide and Cozman extend these ideas to belief updating on credal networks, by developing a loopy variant of 2U that makes the algorithm usable for multiply connected binary credal networks. This idea has further exploited by the *generalized loopy 2U*, which transforms a generic credal network into an equivalent binary credal network, which is indeed updated by the loopy version of 2U [6]. Other approximate inference algorithms can produce either outer or inner approximations: the former produce intervals that enclose the correct probability interval between lower and upper probabilities [18,68,49,71], while the latter produce intervals that are enclosed by the correct probability interval [15,34]. Some of these algorithms emphasize enumeration of vertices, while others resort to optimization techniques (as computation of lower/upper values for $P(x_q|x_E)$ is equivalent to minimization/maximization of a fraction containing polynomials in probability values). Overviews of inference algorithms for imprecise probabilities have been published by Cano and Moral (e.g., [17]).

5.3 Modelling and Updating with Missing Data

In the updating problem described in Equation (12), the evidence x_E is assumed to report the actual values of the variables in X_E . This implicitly requires the possibility of making *perfectly reliable* observations. Clearly, this is not always realistic. An example is the case of *missing data*: we perform an observation but the outcome of the observation is not available. The most popular approach to missing data in the literature and in the statistical practice is based on the so-called *missing at random* assumption (MAR, [58]). This allows missing data to be neglected, thus turning the incomplete data problem into one of complete data. In particular, MAR implies that the probability of a certain value to be missing does not depend on the value itself, neither on other non-observed values. For instance, the temporary breakdown of a sensor produces MAR missing data, because the probability of missing is one, regardless of the actual value⁹. As a further example, consider a medical center where test B is performed only if test

⁹ In this case, the data are *missing completely at random* (MCAR), which is a special case of MAR [54].

A is positive; the missingness of B is MAR because its probability to be missing only depends on the observed value of A. Yet, MAR is not realistic in many cases. Consider for instance an exit poll performed during elections, where the voters of the right-wing party sometimes refuse to answer; in this case, the probability of an answer to be missing depends on its value and thus the missingness is non-MAR. Ignoring missing data that are non-MAR can lead to unreliable conclusions; in the above example, it would underestimate the proportion of right-wing voters. However, it is usually not possible to test MAR on the incomplete observations; if MAR does not appear tenable, more conservative approaches than simply ignoring missing data are necessary in order to avoid misleading conclusions.

De Cooman and Zaffalon have developed an inference rule based on much weaker assumptions than MAR, which deals with near-ignorance about the missingness process [39]. This result has been extended by Zaffalon and Miranda [82] to the case of mixed knowledge about the missingness process: for some variables the process is assumed to be nearly unknown, while it is assumed to be MAR for the others. The resulting updating rule is called *conservative inference rule* (CIR).

To show how CIR-based updating works, we partition the variables in \mathbf{X} in four classes: (i) the queried variable X_q , (ii) the observed variables X_E , (iii) the unobserved MAR variables X_M , and (iv) the variables X_I made missing by a process that we basically ignore. CIR leads to the following credal set as our updated beliefs about the queried variable:¹⁰

$$K(X_q ||^{X_I} x_E) := \text{CH} \left\{ P_j(X_q | x_E, x_I) \right\}_{x_I \in \Omega_{X_I}, j=1, \dots, v}, \quad (16)$$

where the superscript on the double conditioning bar is used to denote beliefs updated with CIR and to specify the set of missing variables X_I assumed to be non-MAR, and $P_j(X_q | x_E, x_I) = \sum_{x_M} P_j(X_q, x_M | x_E, x_I)$. The insight there is that, as we do not know the actual values of the variables in X_I and we cannot ignore them, we consider all their possible explanation (and then we take the convex hull).

When coping only with the missing-at-random variables (i.e., if X_I is empty), Equation (16) becomes a standard updating task to be solved by some of the algorithms in Section 5.2. Although these algorithms cannot be applied to solve Equation (16) if X_I is not empty, a procedure to map a conservative inference task as in Equation (16) into a standard updating task as in Equation (12) over a credal network defined over a wider domain has been developed [7]. The transformation is particularly simple and consists in the augmentation of the original credal network with an auxiliary child for each non-missing-at-random variable, with an extensive quantification. This procedure is described by the following example.

Example 5 (CIR-based updating by standard updating algorithms). Consider the credal network in Example 1. Assume that you want to update your beliefs about X_0 , after the observation of both X_1 and X_2 . The observation of X_1 is x_1 , while the outcome of the

¹⁰ This updating rule can be applied also to the case of *incomplete* observations, where the outcome of the observation of X_I is missing according to a non-missing-at-random process, but after the observation some of the possible outcomes can be excluded. If $\Omega'_{X_I} \subset \Omega_{X_I}$ is the set of the remaining outcomes, we simply rewrite Equation (16), with Ω'_{X_I} instead of Ω_{X_I} .

observation of X_2 is missing, and the MAR assumption seems not tenable. Accordingly we update the model by means of conservative inference rule as in Equation (16) to compute $K(X_0||X_2x_1)$. In order to map this CIR-based updating task into a standard updating, let us perform the following transformation. As described in Figure 5, we first augment the network with an auxiliary binary variable X_3 , which is a child of X_2 . Then we extensively quantify the relation between these two nodes as:

$$P(X_3|X_2) \in \left\{ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right\}, \quad (17)$$

which can be indeed formulated as a separate specification by augmenting the network with a binary node X_4 , which is a parent of X_3 , according to the procedure described in Example 4. The result in [7] states that $K(X_0||X_2x_1) = K(X_0|x_1, x_3)$, where x_3 is the state corresponding to the first row of the tables in Equation (17). The lower and upper probabilities associated to the posterior credal set can be therefore computed by standard updating.

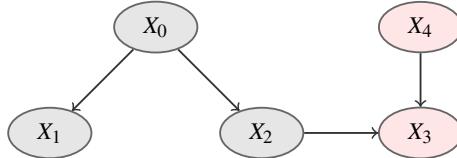


Fig. 5. Modelling non-missing-at-random observation of X_2 in credal network

6 An Application: Assessing Environmental Risk by Credal Networks

In the previous sections we gave the reader a number of theoretical tools for both modelling and interacting with credal networks. In this section, we want to present a real-world application of these methods consisting in a specific risk analysis task.¹¹ The credal network merges into a single coherent framework different kinds of domain knowledge: deterministic equations, human expertise, and historical data are used to quantify the network in its different parts. After the model specification, risk analysis can be automatically performed by means of some of the updating algorithms in Section 5.2.

6.1 Debris Flows

Debris flows are among the most dangerous and destructive natural hazards that affect human life, buildings, and infrastructures (see Figure 6). They are gravity-induced mass

¹¹ We point the reader to [64] for a gentle introduction to the practical implementation of a credal network in knowledge-based expert systems.

movements intermediate between landslides and water floods. The flow is composed of a mixture of water and sediment with a characteristic mechanical behavior varying with water and soil content. According to [31], prerequisite conditions for most debris flows include an abundant source of unconsolidated fine-grained rock and soil debris, steep slopes, a large but intermittent source of moisture (rainfall or snow-melt), and sparse vegetation. As mentioned in [48], several hypotheses have been formulated to explain mobilization of debris flows and this aspect still represents a research field. According to the model proposed by [70], the mechanism to disperse the materials in flow depends on the properties of the materials (like *granulometry* and the internal friction angle), channel slope, flow rate and water depth, particle concentration, etc., and, consequently, the behavior of flow is also various. Unfortunately, not all the triggering factors considered by this model can be directly observed, and their causal relations with other observable quantities can be shaped only by probabilistic relations. In fact, the analysis of historical data and the role of human expertise are still fundamental for hazard identification as many aspects of the whole process are still poorly understood. For these reasons, a credal network seems to be a particularly suitable model for approaching a problem of this kind.



Fig. 6. Debris flows examples

6.2 The Credal Network

In order to implement a credal network able to estimate the level of risk of a debris flow happening in a particular area, we first define the *Movable Debris Thickness* as the

depth of debris likely to be transported downstream during a flood event. Such variable represents an integral indicator of the hazard level. Then we identify a number of triggering factors which may affect the value of this thickness. Once we have identified the factors, the specification of the directed acyclic graph associated to these variables can be achieved assuming a *causal* interpretation to the arcs of the graph.¹² Figure 7 depicts the resulting graph. We point the reader to [5] for a detailed description of the different variables in this model. Here let us only report the information we need to understand the key features of both the modelling and the inference with a model of this kind.

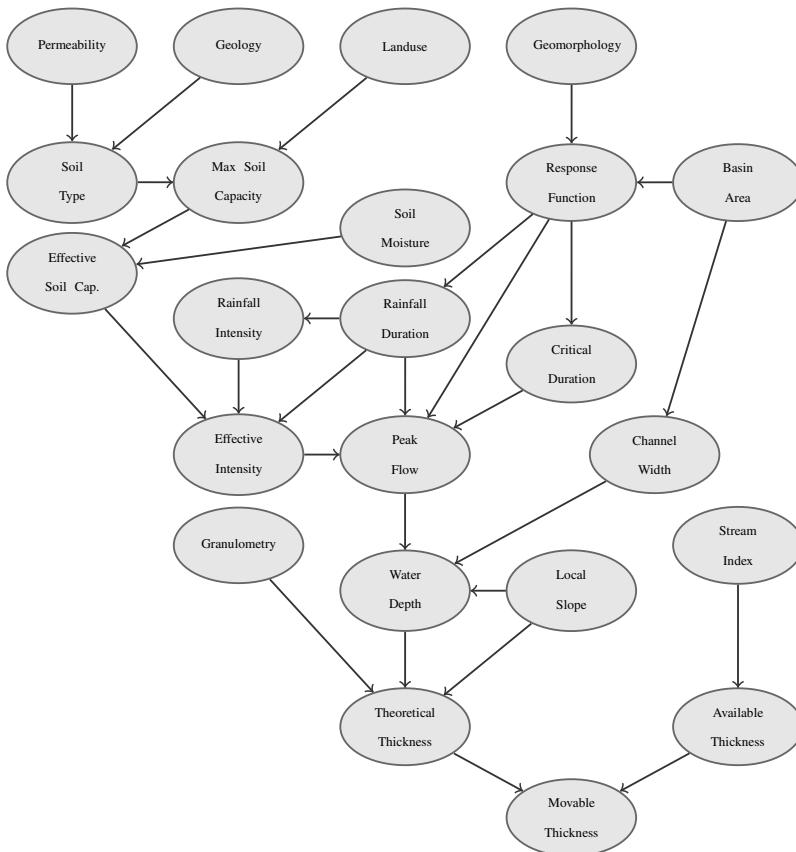


Fig. 7. A credal network for environmental risk analysis

¹² Remember that, according to the Markov condition, the directed graph is a model of conditional independence relations. The causal interpretation is therefore not always justified.

Credal networks have been defined only for categorical variables.¹³ Some of the variables in the network are natively categorical. This is for instance the case of variable *Land Use*, whose six possible values are: *Forest, Pasture, Rivers and water bodies, Improductive vegetation, Bare soils and rocks, Edificated surfaces*. Some other variables, like for example the *Movable Debris Thickness* are numerical and continuous. A discretization like that in Table 2 is therefore required.

Table 2. Discretization of variable *Movable Debris Thickness* and corresponding interpretation in terms of actual level of risk. The same discretization has been used also for variables *Theoretical Thickness* and *Available Thickness*.

Range	Risk Level	Symbol
< 10 cm	low risk	<
10 – 50 cm	medium risk	=
> 50 cm	high risk	>

Regarding the probabilistic quantification of the conditional values of the variables given the parents, as we have based our modelling on a geomorphological model of the triggering of the debris flow, we have deterministic equations for most of the variables in the network. Given an equation returning the numerical value of a child given the values of the parents, we can naturally induce the quantification of the corresponding conditional probabilities. A quantification of this kind is clearly precise (i.e., described by a single conditional probability table) and in particular deterministic (i.e., the columns corresponding to the different conditional mass functions assign all the mass to a single outcome and zero to the others). As an example, the *Movable Debris Thickness* is the minimum between the *Theoretical Thickness* and the *Available Thickness* and this corresponds to the following conditional probability table (where the symbols in Table 2 are used to denote the states of the variables):

		< = >			< = >			< = >		
		<	<	<	=	=	=	>	>	>
		Movable	Theoretical	Available	Movable	Theoretical	Available	Movable	Theoretical	Available
<	1	1	1	1	0	0	0	1	0	0
=	0	0	0	0	1	1	0	1	0	0
>	0	0	0	0	0	0	0	0	0	1

For some other variables, including also all the root (i.e., parentless) nodes, we have not relations of this kind. In this cases, we used, when available, historical dataset, from which we obtained conditional credal sets by means of the imprecise Dirichlet model as in Equation (7). Note that, especially in the conditional case, the amount of data can be relatively small, and the difference between the credal sets we learn by means of

¹³ In a certain sense, the work in [11] can be implicitly regarded as an exception of this statement. Yet, research on credal network with continuous variable is still in its early stage.

the imprecise Dirichlet model and the precise is not trivial. This is a further justification for our choice of modelling the problem by means of a credal instead of a Bayesian network.

Finally, the counterpart of the role of human expertise in the evaluation is the fact that some of the credal set we quantify in our model cannot be obtained from data neither from deterministic relations. In these cases, we ask an expert to report his knowledge. Notably, the possibility of expressing his beliefs by intervals of probability instead of single values makes the description much more realistic.¹⁴ Overall, we achieve in this way the quantification of a credal network over the directed acyclic graph in Figure 7.

The practical application of a model of this kind consists in the computation of the posterior probability intervals for the three different level of the risk given the observed values for some of the other variables in the network for the particular scenario under consideration. The updating has been provided by means of the algorithm in [14]. The histograms in Figure 10 report the posterior intervals obtained for three different scenarios. Note that, according to the interval-dominance criterion in the scenario (a) we can reject the second and the third histogram, and conclude that a level of high risk occurs. Regarding (b), the high risk dominates the low risk, which is therefore rejected, but there is an indecision between high and medium risk, while in the scenario (c) no dominance is present and we are in a situation of complete indecision between the three different levels of risk. This kind of analysis can be automatically performed by the credal network on extensive areas, this providing an important support to the experts for this problem. Figure 9 reports an extensive analysis for the basin in Figure 8.

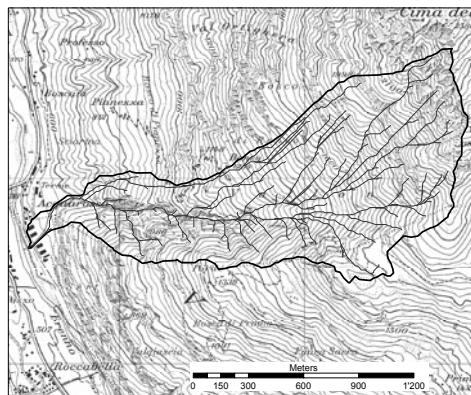


Fig. 8. Acquarossa Creek Basin (area 1.6Km^2 , length 3.1Km)

¹⁴ We thanks Dott. Andrea Salvetti, the environmental expert who was involved in this quantification task and in many other aspects of this project.

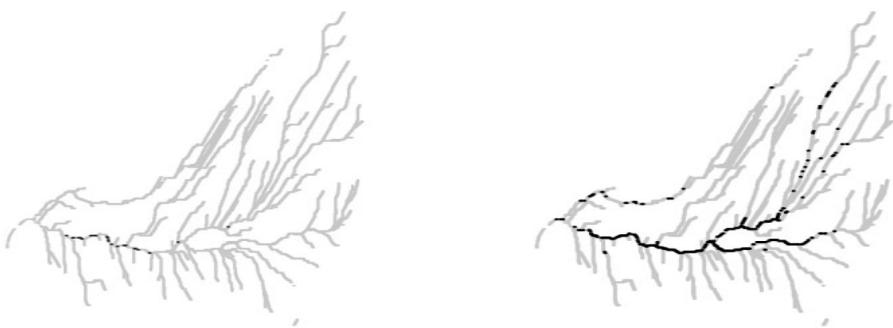


Fig. 9. Spatially distributed identifications for the basin in Figure 8 and rainfall return periods of 10 (left) and 100 (right) years. The points for which the credal network predicts the lower class of risk are depicted in gray, while black refers to points where higher levels of risk cannot be excluded.

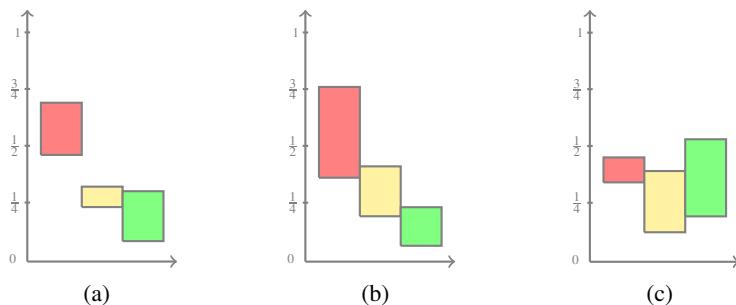


Fig. 10. Posterior probability intervals for the three level of risk (colors red, yellow and green correspond respectively to high, medium and low risk)

7 Credal Classifiers

In the rest of this chapter we show how credal networks can be used to deal with a classical field of data mining, namely *classification*. Classification is the problem of predicting the *class* of a given object, on the basis of some attributes (*features*) of it. A historical example is the iris problem designed by Fisher in 1936: the goal is to predict the species of Iris (among three possible categories) on the basis of four features, namely the length and the width of the sepal and the petal.

Training a probabilistic classifier corresponds to estimating from data the *joint* distribution $P(C, \mathbf{A})$, where C denotes the class variable and $\mathbf{A} = \{A_1, \dots, A_k\}$ the set of k features. In the Bayesian framework, the estimation of the joint distribution starts by initializing it to an initial value (the *prior*), which represents the beliefs of the investigator *before* analyzing the data; the prior thus enables to model domain knowledge. Then the *likelihood* function is computed from the data, modelling the evidence coming from the observations. Prior and likelihood are multiplied, leading to a *posterior* joint distribution. As described in Section 2, when dealing with Bayesian networks, one does not need to specify the full joint; it is enough to specify the local conditional distributions, and the network automatically represents the joint. A trained classifier is assessed by checking its accuracy at classifying instances. To classify an instance characterized by the assignment $\mathbf{a} = \{a_1, \dots, a_k\}$ of the features, the conditional distribution $P(C|\mathbf{a})$ is computed from the posterior joint.

A traditional criticism of Bayesian methods is the need for specifying a prior distribution. In fact, prior information is generally difficult to quantify; moreover one often prefers to let the data speak by themselves, without introducing possibly subjective prior beliefs. As for classification in particular, Bayesian classifiers might happen to return *prior-dependent* classifications, i.e., the most probable class varies under different priors. As the choice of any single prior entails some arbitrariness, prior-dependent classifications are typically unreliable: in fact, they translate the arbitrariness of the choice of the prior into arbitrariness of the conclusions. Prior-dependent classifications are more frequent on small data sets; on large data sets, classifications are less sensitive on the choice of the prior. Nevertheless, as shown in Section 9.1, unreliable prior-dependent classifications can be present also in large data sets. Most often, one deals with the choice of the prior by setting a *uniform* prior, because it looks non-informative; yet, such an approach has important drawbacks, as shown in the following example, inspired to [78].

Let us consider a bag containing blue marbles and red marbles; no drawings have been made from the urn. Which is the probability of getting a red (or blue) marble in the next draw? Using the uniform prior, one should assign the same probability 0.5 to both colors. This underlies the (very strong) assumption that the urn contains an equal number of red and blue marbles; in the subjective interpretation of probability, this means that one is equally available to bet an amount of money 0.5 on either red or blue, in a gamble with reward 1 and 0 for respectively a correct and a wrong prediction. In fact, the uniform prior is a model of *prior indifference*. However, we are ignorant about the content of the urn rather than indifferent between the two colors; in this condition, the only reliable statement is that the proportion of red (or blue) marbles is comprised between 0 and 1. Walley's theory of imprecise probability [77] states that such *prior*

ignorance should be represented by a *set* of prior distributions rather than by a single prior. The adoption of a set of priors (letting the proportion of blue and red vary between 0 and 1) prevents betting on any of the two colors, which is more sensible, under ignorance, than being equally available to bet on both.

Credal classifiers extend Bayesian classifiers to imprecise probabilities; they represent *prior-ignorance*¹⁵ by specifying a (credal) set of priors, often using the IDM [78]. The credal set of the IDM is then turned into a set of posterior by element-wise application of Bayes' rule: in fact, training a credal classifier corresponds to update the set of priors with the likelihood, yielding a *set* of posteriors. Credal classifiers detect prior-dependent instances by checking whether the most probable class is consistent or not across the set of posteriors. If the instance is prior-dependent, a credal classifier returns a set of classes, drawing a less informative but more robust conclusion than a Bayesian classifier.

However, besides prior-ignorance, there is another kind of ignorance involved in the process of learning from data, i.e., ignorance about the missingness process (MP). Usually, classifiers ignore missing data, assuming missing data to be MAR. In general there is no way to verify the MAR assumption on the incomplete data; furthermore assuming MAR when it does not hold can cause to a large decrease of accuracy [65]. However, credal classifiers have been also extended to conservatively deal with non-MAR missing data [29], relying on CIR, namely by considering all the data sets consistent with the observed incomplete data set.

Other classifiers, besides the credal ones, suspend the judgment on doubtful instances. For instance, a *rejection rule* can be set on any classifier, refusing to classify the instances (and hence returning the whole set of classes), where the probability of the most probable class is below a certain threshold. A more sophisticated approach has been developed by del Coz et al. [33]: their algorithm determines which set of classes to return (possibly a single class), on the basis of the posterior distribution computed by the classifier; this algorithm can be therefore applied to any probabilistic classifier. The returned set of classes is identified in order to maximize the F-measure of the issued classifications.

However, both the rejection rule and the algorithm of [33] work on a single probability distribution; instead, credal classifiers deal with a *set* of posterior distributions. The practical difference can be appreciated by considering a classifier trained on a very small learning set. The credal classifier will be very likely to suspend the judgment on any new instance, as most instances are going to be prior-dependent. On the contrary, a traditional classifier equipped with the rejection rule or with the algorithm of [33] blindly trusts the computed posterior distribution, without considering that for instance on small data sets it might be largely sensitive on the choice of the prior.

8 Naive Bayes

The naive Bayes classifier (NBC) [40] “naively” assumes the features A_1, \dots, A_k to be independent given the class C . According to the Markov condition introduced in

¹⁵ More precisely, prior *near-ignorance*; full ignorance is not compatible with learning, as shown in Section 7.3.7 of Walley [77].

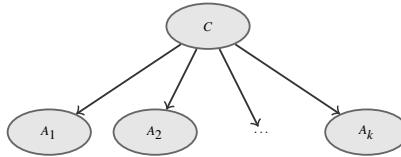


Fig. 11. The naive Bayes classifier

Section 2, these conditional independence relations can be graphically depicted by the directed graph in Figure 11. These assumptions introduce a severe bias in the estimate of probabilities, as the real data generation mechanism does *not* generally satisfy such condition. As a consequence of such unrealistic assumption, NBC is often overconfident in its predictions, assigning a very high probability to the most probable class [50]; this phenomenon is emphasized if redundant features are present.

Despite the simplistic naive assumption, NBC performs surprisingly well under 0-1 loss¹⁶[40,50]. A first reason is that the bias of the probability estimates may not matter under 0-1 loss: given two classes c' and c'' (c' being the correct one), even a severe bias in the estimate of $P(c')$ will not matter, provided that $P(c') > P(c'')$ [46]. The good performance of NBC can be further explained by decomposing the misclassification error into bias and variance [46]: NBC has indeed high bias, but this problem is often successfully remediated by low variance. Especially on small data sets, low variance is more important than low bias; in this way, NBC can outperform more complex classifiers. Instead, more parameterized classifiers tend to outperform NBC on large data sets. The low variance of NBC is due to the low number of parameters, which is a consequence of the naive assumption, which prevents modelling correlations between features. For instance, in comparison with C4.5 (which has lower bias but higher variance), NBC is generally more accurate on smaller sample sizes, but generally outperformed on larger data sets [55].

A further factor which contributes to the good performance of NBC is feature selection, which typically removes the most correlated features and thus makes the naive assumption more realistic. In fact, NBC can be even very competitive, when trained with a carefully designed feature set. For instance, the CoIL challenge [74] was won by a NBC entry [41], which outperformed more complicated models such as SVMs or neural networks. The data set of the competition was characterized by several correlated features and noisy data; a later analysis of the contest [74] showed that variance was a much bigger problem than bias for this data set. However, key factors for the success of NBC were feature selection and the introduction of a particular feature, obtained by taking the Cartesian product of two important features, which enabled NBC to account for the interactions between such two features. In [44], a criterion aimed at removing irrelevant or redundant features on the basis of conditional mutual information is designed; under this feature selection, NBC is competitive with SVMs and boosting in several problems.

¹⁶ This loss function is also known as *misclassification error*: if the most probable class is the correct one the loss is zero and one otherwise.

Further strengths of NBC are computational speed and easy handling of missing data, at least under the MAR assumption. However, if MAR is not assumed, the computation becomes quite complicated; see for instance the algorithms designed in [65].

NBC has been recognized as one of the ten most influential data mining algorithms [80] and in fact there have been also countless NBC variants designed to improve its performance; comprehensive references can be found for instance in [50] and [51].

8.1 Mathematical Derivation

Let us denote by C the classification variable (taking values in Ω_C) and as A_1, \dots, A_k the k feature variables (taking values from the finite sets $\Omega_{A_1}, \dots, \Omega_{A_k}$).

We denote by $\theta_{c,a}$ the chance (i.e., the unknown probability about which we want to make inference) that $(C, A_1, \dots, A_k) = (c, \mathbf{a})$, by $\theta_{a_i|c}$ the chance that $A_i = a_i$ given that $C = c$, by $\theta_{\mathbf{a}|c}$ the chance that $A_1, \dots, A_k = (a_1, \dots, a_k)$ conditional on c .

The naive assumption of independence of the features given the class can be expressed as:

$$\theta_{\mathbf{a}|c} = \prod_{i=1}^k \theta_{a_i|c}. \quad (18)$$

We denote by $n(c)$ and $n(a_i, c)$ the observed counts of $C = c$ and of $(A_i, C) = (a_i, c)$; by \mathbf{n} the vector of all such counts. We assume for the moment the data set to be complete. The *likelihood* function can be expressed as a product of powers of the theta-parameters:

$$L(\theta|\mathbf{n}) \propto \prod_{c \in \Omega_C} \left[\theta_c^{n(c)} \prod_{i=1}^k \prod_{a_i \in \Omega_{A_i}} \theta_{a_i|c}^{n(a_i, c)} \right]. \quad (19)$$

Observe that for all $c \in \Omega_C$ and $i = 1, \dots, k$, the counts satisfy the *structural constraints* $0 \leq n(a_i, c) \leq n(c)$, $\sum_{c \in \Omega_C} n(c) = n$ and $\sum_{a_i \in \Omega_{A_i}} n(a_i, c) = n(c)$, with n total number of instances.

The prior is usually expressed as a product of Dirichlet distributions. Under this choice, the prior is analogous to the likelihood, but the counts $n(\cdot)$ are replaced everywhere by $st(\cdot) - 1$, where $s > 0$ is the *equivalent sample size*, which can be interpreted as the number of hidden instances. The parameters $t(\cdot)$ can be interpreted as the proportion of units of the given type; for instance, $t(c')$ is the proportion of hidden instances for which $C = c'$, while $t(a_i, c')$ is the proportion of hidden instances for which $C = c'$ and $A = a_i$. This is a non-standard parameterization of the Dirichlet distribution, introduced in [77] because of its convenience when dealing with the IDM; the usual parameterization is instead $\alpha(\cdot) = st(\cdot)$.

We consider in particular the Perks prior, as in [81, Section 5.2]:

$$t(c) = \frac{1}{|\Omega_C|}; \quad t(a_i, c) = \frac{1}{|\Omega_C||\Omega_{A_i}|}. \quad (20)$$

However, in some cases the uniform prior is modeled adopting the Laplace estimator [79, Chapter 4.2], which is different from Equation (20): it sets $\alpha(c) = st(c) = 1 \forall c$ and $\alpha(a_i, c) = st(a_i, c) = 1 \forall c, i$, which corresponds to initialize all counts $n(c)$ and $n(a_i, c)$

to 1 before analyzing the data. For instance, the WEKA implementation [79] of NBC is done in this way. However, there are slightly different versions also for the Laplace estimator; see for instance [56].

By multiplying the prior density and the likelihood function, we obtain a posterior density for $\theta_{c,\mathbf{a}}$, which is again a product of independent Dirichlet densities:

$$P(\theta_{c,\mathbf{a}}|\mathbf{n}, \mathbf{t}) \propto \prod_{c \in \Omega_C} \left[\theta_c^{n(c)+st(c)-1} \prod_{i=1}^k \prod_{a_i \in \Omega_{A_i}} \theta_{a_i|c}^{n(a_i,c)+st(a_i,c)-1} \right]. \quad (21)$$

compared to the likelihood (19), the parameters $n(\cdot)$ are replaced by $n(\cdot) + st(\cdot)$. The joint probability of c and \mathbf{a} can be computed by taking expectation from the posterior :

$$P(c, \mathbf{a}|\mathbf{n}, \mathbf{t}) = E(c, \mathbf{a}|\mathbf{n}, \mathbf{t}) = P(c|\mathbf{n}, \mathbf{t}) \prod_{i=1}^k P(a_i|c, \mathbf{n}, \mathbf{t}) \quad (22)$$

where

$$P(c|\mathbf{n}, \mathbf{t}) = E[\theta_c|\mathbf{n}, \mathbf{t}] = \frac{n(c)+st(c)}{n+s}, \quad (23)$$

$$P(a_i|c, \mathbf{n}, \mathbf{t}) = E[\theta_{a_i|c}|\mathbf{n}, \mathbf{t}] = \frac{n(a_i,c)+st(a_i,c)}{n(c)+st(c)}. \quad (24)$$

A problem of NBC, and more in general of any Bayesian classifier, is that sometimes the classifications is *prior-dependent*, namely the most probable class varies with the parameters $t(\cdot)$. Most often, one chooses the uniform prior trying to be non-informative; yet, we have already argued that prior-dependent classifications are fragile and that the uniform prior does not satisfactorily model prior ignorance. To address this issue, NCC is based on a set of priors rather than on a single prior.

9 Naive Credal Classifier (NCC)

NCC extends NBC to imprecise probabilities by considering a (credal) set of prior densities, instead of a unique prior. This prior credal set is modeled through the Imprecise Dirichlet Model (IDM) [77] and expresses prior *near-ignorance* [77, Section 4.6.9];¹⁷ it is then turned into a set of posteriors (posterior credal set) by element-wise application of Bayes' rule.

NCC specifies a *joint* credal set using the IDM; this is obtained by allowing each parameter of type $t(\cdot)$ to range within an interval, rather than being fixed to a single value. In particular the IDM contains all the densities for which \mathbf{t} varies within the polytope T , defined as follows:

$$T = \begin{cases} \sum_{c \in \Omega_C} t(c) = 1 \\ t(c) > 0 & \forall c \in \Omega_C \\ \sum_{a \in \Omega_A} t(a,c) = t(c) & \forall c \in \Omega_C \\ t(a,c) > 0 & \forall a \in \Omega_A, c \in \Omega_C. \end{cases} \quad (25)$$

¹⁷ Indeed, full ignorance is not compatible with learning; see Section 7.3.7 of [77] and [86].

Such constraints are analogous to the structural constraints which characterize the counts $n(\cdot)$. The third constraint introduces a link between the credal set $K(C)$ and the credal sets $K(A_i|c)$, with $c \in \Omega_C$, so that the corresponding credal network is *not* separately specified. Since the $t(\cdot)$ vary within an interval, also the posterior probability of class c lies within an interval. For instance, the upper and lower probability of c and \mathbf{a} are:¹⁸

$$\underline{P}(c, \mathbf{a}|\mathbf{n}, \mathbf{t}) := \inf_{\mathbf{t} \in T} P(c, \mathbf{a}|\mathbf{n}, \mathbf{t})$$

$$\overline{P}(c, \mathbf{a}|\mathbf{n}, \mathbf{t}) := \sup_{\mathbf{t} \in T} P(c, \mathbf{a}|\mathbf{n}, \mathbf{t}).$$

While a traditional classifier returns the class with the highest posterior probability, credal classifiers return the classes which are *non-dominated*. The two criteria introduced in Section 5.1 can be considered to assess whether class c' dominates class c'' . According to *interval-dominance* c' dominates c'' if $\underline{P}(c', \mathbf{a}|\mathbf{n}, \mathbf{t}) > \overline{P}(c'', \mathbf{a}|\mathbf{n}, \mathbf{t})$. Instead, according to *maximality*, c' dominates c'' if $P(c', \mathbf{a}|\mathbf{n}, \mathbf{t}) > P(c'', \mathbf{a}|\mathbf{n}, \mathbf{t})$ for all the values of $\mathbf{t} \in T$. Maximality is more powerful than interval-dominance, because it sometimes detect dominances which cannot be spotted using interval-dominance. Once the dominance criterion is chosen, the set of non-dominated classes is identified through repeated pairwise comparisons, as shown in the following pseudo-code:

```
// Since NCC is based on maximality , we denote the set
// of non-dominated classes as  $\Omega_C^{**}$ .
// With interval-dominance , it should be denoted as  $\Omega_C^*$ .
```

```
 $\Omega_C^{**} := \Omega_C;$ 
for  $c' \in \Omega_C\{$ 
    for  $c'' \in \Omega_C, c'' \neq c'\{$ 
        if ( $c'$  dominates  $c''$ ){
            remove  $c''$  from  $\Omega_C^{**}$ ;
        }
    }
}
return  $\Omega_C^{**};$ 
```

In the following we sketch the test of dominance for NCC under maximality, designed in [81]. According to maximality, c' dominates c'' iff:

$$\inf_{\mathbf{t} \in T} \frac{P(c', \mathbf{a}|\mathbf{n}, \mathbf{t})}{P(c'', \mathbf{a}|\mathbf{n}, \mathbf{t})} > 1, \quad (26)$$

¹⁸ Unlike Equation (3), these optimizations are over the open polytope T . For this reason, infima and suprema are considered instead of minima and maxima.

and assuming $P(c_2, \mathbf{a}|\mathbf{n}, \mathbf{t}) > 0$. Problem (26) can be re-written [81] considering Equations (23–24) as:

$$\inf_{\mathbf{t} \in T} \left\{ \left[\frac{n(c'') + st(c'')}{n(c') + st(c')} \right]^{k-1} \prod_{i=1}^k \frac{n(a_i, c') + st(a_i, c')}{n(a_i, c'') + st(a_i, c'')} \right\}. \quad (27)$$

As proved in [81], the infimum of problem (27) is obtained by letting $t(a_i, c') \rightarrow 0$ and $t(a_i, c'') \rightarrow t(c'')$. The values of these parameters at the optimum are extreme, as they touch the boundary of the IDM. The remaining parameters $t(c')$ and $t(c'')$ are optimized by noting that the infimum is achieved when $t(c') + t(c'') = 1$, which allows to express $t(c'')$ as $1 - t(c'')$. The final step to solve problem (27) involves a convex optimization over the single parameter $t(c')$; see [81] for more details.

The classification is *determinate* or *indeterminate* if there are respectively one or more non-dominated classes. The set of non-dominated classes returned by NCC always contains the most probable class identified by NBC, as the uniform prior is included in the IDM;¹⁹ this also means that NCC, when determinate, returns the same class of NBC. On the other hand, if there are more non-dominated classes, the classification issued by NBC is prior-dependent. The non-dominated classes are *incomparable* and thus cannot be further ranked.

NCC has been originally introduced by [81]; applications of NCC to real-world case studies include diagnosis of dementia [87] and prediction of presence of parasites in crops [83]; it has been then extended with a sophisticated treatment of missing data in [29].

In the following, we show a comparison of NBC and NCC in texture recognition; the data set is complete and thus indeterminate classifications are only due to prior-dependent instances.

9.1 Comparing NBC and NCC in Texture Recognition

The goal of texture classification is to assign an unknown image to the correct texture class; this requires an effective description of the image (i.e., obtaining good features) and a reliable classifier. Texture classification is used in many fields, among which industrial applications, remote sensing and biomedical engineering. We compare NBC and NCC on the public OUTEX [62] data set of textures; the results presented in this section are taken from [26], where more details and experiments are described. The data set contains 4500 images from 24 classes of textures, including different kinds of canvas, carpets, woods etc.; some samples are shown in Fig. 12. We use the standard Local Binary Patterns (LBP) [62] as descriptors; they are computed by assigning each pixel to a category comprised between 1 and 18, on the basis of a comparison between its gray level and the gray level of the neighboring pixels. The features of an image are constituted by the percentage of pixels assigned to the different categories; therefore, 18 features are created for each image. There are no missing data.

¹⁹ This is guaranteed with the Perks prior of Equation (20), but not with the Laplace estimator, which is not included into the IDM; yet, empirically this is most often the case also with the Laplace estimator.

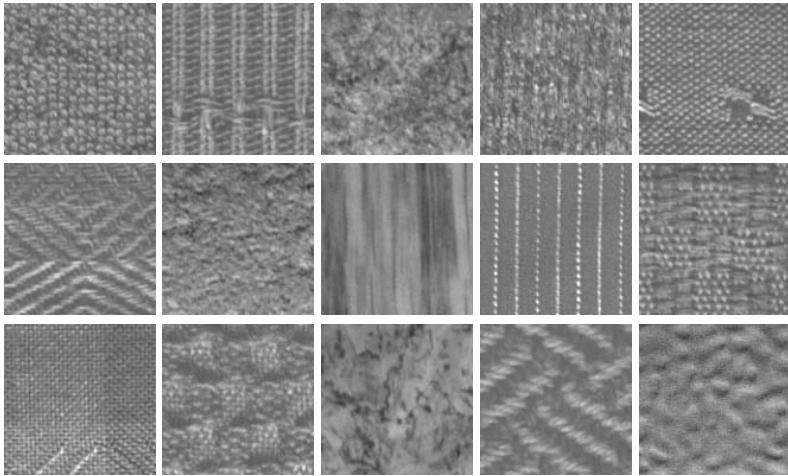


Fig. 12. Examples of some textures: each image refers to a different class

We evaluate the classifiers through cross-validation, discretizing the features via supervised discretization [42]; the feature are discretized on average in some 10 bins. As our aim is to compare NBC and NCC rather than finely tuning the classifiers for maximum performance, we do not perform feature selection.

NBC achieves 92% accuracy, which can be considered satisfactory: for instance, SVMs are only slightly better, achieving 92.5%. However, NBC is unreliable on the prior-dependent instances, which amount to about 5% of the total. On these instances, NBC achieves only 56% accuracy, while NCC returns on average 2.5 classes, achieving 85% accuracy. On the non-prior dependent instances, both NBC and NCC achieve 94% accuracy.

Prior-dependent instances are present even on this large data set because each conditional probability distribution of type $P(A_j|C)$ requires to estimate some 240 parameters (24 classes \times 10 states of the feature after discretization); since some combinations of the value of the class and of the feature rarely appear in the data set, the estimate of their probability is sensitive on the chosen prior.

Experiments at varying size of the training set are presented in Fig.13. At each size of the training set, NBC is characterized by a mix of good accuracy on the instances which are not prior-dependent, and bad accuracy on the prior-dependent ones; see Fig.13(a). Thanks to indeterminate classifications, NCC is instead much more accurate than NBC on the prior-dependent instances: see Fig.13(b). With increasing size of the training set, NCC becomes more determinate, steadily reducing both the percentage of indeterminate classification and the average number of classes returned when indeterminate; see Fig. 13(c).

Summing up, NBC is generally little accurate on the instances indeterminately classified by NCC; NCC preserves its reliability on prior-dependent instances thanks to indeterminate classifications; the determinacy of NCC increases with the size of the training set; indeterminate classifications can convey valuable information, when a small subset

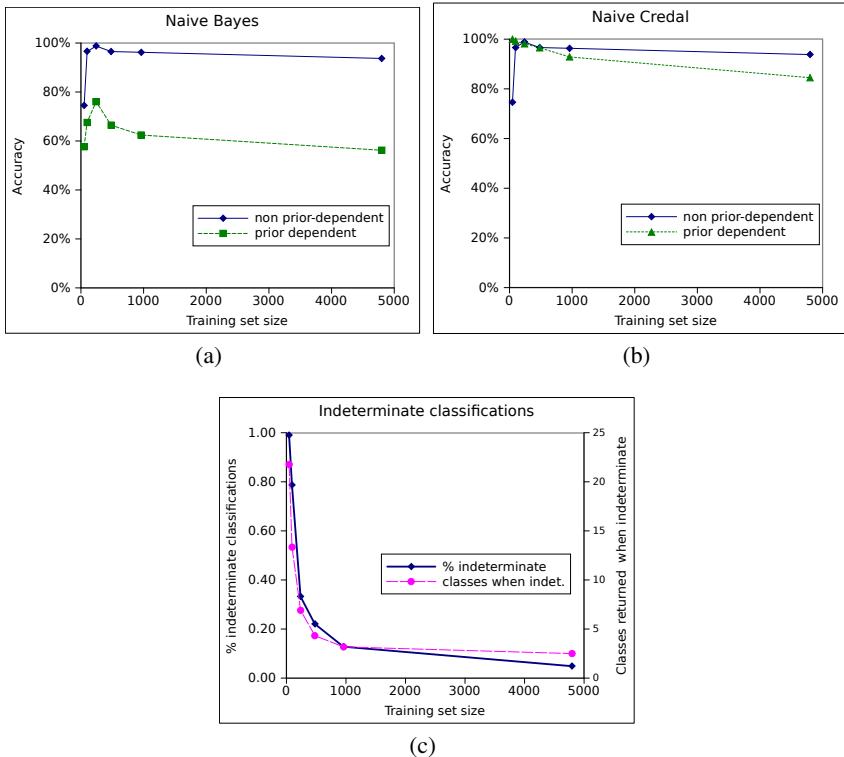


Fig. 13. Experiments with varying sizes of the training set. Plots (a) and (b) show the accuracy of NBC and NCC on instances which are prior dependent (dashed) and non prior-dependent (solid); plot (c) shows the percentage of indeterminate classifications (solid) and the average number of classes returned by NCC when indeterminate (dashed).

of classes is returned out of many possible ones. Such results are consistent with those obtained [29] on the classical UCI data sets.

However, NBC provides no way of understanding whether a certain instance is prior-dependent. One could try to mimic the behavior of NCC by setting a *rejection rule* on NBC, outputting more classes if the probability of the most probable class does not exceed a certain threshold. Yet, a rejection rule is likely to be little effective with NBC, which generally returns high probability for the most probable class. In the texture application, NCC detects about half of the prior-dependent instances among those which are classified by NBC with probability higher than 95%. As discussed in [29, Section 4.4], an instance is less likely to be prior-dependent as the probability computed by NBC for the most probable class increases, but such correlation is not deterministic: there are prior dependent instances classified by NBC with high probability, and non-prior dependent ones classified by NBC with a relatively small margin. Overall, the prior-dependency analysis performed by NCC is much more sophisticated than any rejection rule.

9.2 Treatment of Missing Data

Very often, real data sets are *incomplete*, because some values of the feature variables are not present.²⁰ Dealing with incomplete data sets rests on the assumptions done about the process responsible for the missingness. This process can be regarded as one that takes in input a set of complete data, which is generally not accessible for learning, and that outputs an incomplete data set, obtained by turning some values into missing. Learning about the missingness process' behavior is usually not possible by only using the incomplete data. This fundamental limitation explains why the assumptions about the missingness process play such an important role in affecting classifiers' predictions. Moreover, a missingness process may also be such that the empirical analysis of classifiers is doomed to provide misleading evidence about their actual predictive performance, and hence, indirectly, about the quality of the assumptions done about the missingness process. This point in particular has been discussed in [29, Section 4.6] and [86, Section 5.3.2]. For these reasons, assumptions about the missingness process should be stated with some care.

In the vast majority of cases, common classifiers deal with missing values (sometimes implicitly) assuming that the values are MAR [69]. However, assuming MAR when it does not hold can decrease the classification accuracy.

The NCC has been one of the first classifiers [81, Section 3], together with Ramoni and Sebastiani's *robust Bayes classifier* [65] (a robust variant of NBC to deal with missing data), to provide a way to conservatively deal with non-MAR missing data in the training set. Both approaches are based on very weak, and hence tenable, assumptions about the missingness process; in fact, they regard as possible any realization of the training set, which is consistent with the incomplete training set; this way of dealing with missing data has been pioneered in statistics by Manski [60].

In particular, according to the conservative inference rule, introduced in Section 5.3, conservative treatment of missing data requires to compute many likelihoods, one per each complete data sets consistent with the incomplete training set. In particular, the approach of [81] is equivalent to inferring many NCCs: one per each complete data sets consistent with the incomplete training set; the classification is given by the union of the set of non-dominated classes produced by all the NCCs.²¹ In [81] specific procedures are designed, which perform the computation exactly and in linear time w.r.t. the amount of missing data, avoiding to enumerate the (exponentially many) complete data sets and to infer many NCCs. The imprecision introduced by missing data leads to an increase in the indeterminacy of the NCC, which is related to the amount of missingness. In other words, the NCC copes with the weak knowledge about the missingness process by weakening the answers to maintain reliability.

In [29] the treatment of missing data has further improved, allowing NCC to deal with non-MAR missing data also in the instance to classify and not only in the training set, and to deal with a mix of MAR and non-MAR features (treating the first group according to MAR and the second in a conservative way), using CIR. The resulting classifier is called NCC2. Distinguishing variables that are subject to the two types of processes is important because treating MAR variables in a conservative way leads to

²⁰ We do not consider here the case of missing values of the class variable.

²¹ Strictly speaking, this straightforward explanation is valid for the case of two classes.

an excess of indeterminacy in the output that is not justified. In fact, the experimental results of NCC2 [29] show that the indeterminacy originated from missing data is compatible with informative conclusions provided that the variables treated in a conservative way are kept to a reasonable number (feature selection can help in this respect, too). Moreover, they show that the classifiers that assume MAR for all the variables are often substantially unreliable when NCC2 is indeterminate.

Formal justifications of the rule NCC2 uses to deal with missing values can be found in [29]. This work discusses also, more generally, the problem of incompleteness for uncertain reasoning.

10 Metrics for Credal Classifiers

Before introducing further credal classifiers, it is useful to review the metrics which can be used to compare them. The overall performance of a credal classifier can be fully characterized by four indicators [29]:

- *determinacy*, i.e., the percentage of instances determinately classified;
- *single-accuracy*, i.e., the accuracy on the *determinately* classified instances;
- *set-accuracy*, i.e., the accuracy on the *indeterminately* classified instances;
- *indeterminate output size*: the average number of classes returned on the *indeterminately* classified instances.

However, set-accuracy and indeterminate output size are meaningful only if the data set has more than two classes.

These metrics completely characterize the performance of a credal classifier, but do not allow to readily compare two credal classifiers. Two metrics suitable to compare credal classifiers have been designed in [30]. The first one, borrowed from multi-label classification,²² is the *discounted-accuracy*:

$$\text{d-acc} = \frac{1}{n_{te}} \sum_{i=1}^{n_{te}} \frac{(\text{accurate})_i}{|Z_i|},$$

where $(\text{accurate})_i$ is a 0-1 variable, showing whether the classifier is accurate or not on the i -th instance; $|Z_i|$ is the number of classes returned on the i -th instance and n_{te} is the number of instances of the test set. However, discounting *linearly* the accuracy on the output size is arbitrary. For example, one could instead discount on $|Z_i|^2$.

The non-parametric *rank test* overcomes this problem. On each instance, it ranks two classifiers CL_1 and CL_2 as follows:

- if CL_1 is accurate and CL_2 inaccurate: CL_1 wins;
- if both classifiers are accurate but CL_1 returns less classes: CL_1 wins;
- if both classifiers are wrong: tie;
- if both classifiers are accurate with the same output size: tie.

²² The metric is referred to as *precision* in [73].

The wins, ties and losses are mapped into ranks and then analyzed via the Friedman test. The rank test is more robust than d-acc, as it does not encode an arbitrary function for the discounting; yet, it uses less pieces of information and can therefore be less sensitive. Overall, a cross-check of the both indicators is recommended.

Instead, an open problem is how to compare a credal classifier with a classifier based on traditional probability. So far, this comparison has been addressed by comparing the accuracy achieved by the Bayesian classifier on the instances determinately and indeterminately classified by the credal classifier, thus assessing how good the credal classifier is at isolating instances which cannot be safely classified with a single class. This produces a statistics of type: on the prior-dependent instances, the Bayesian classifier achieves 60% accuracy returning a single class, while the credal classifier achieves 90% accuracy, returning two classes. But which one is better? Moreover, returning a very similar probability for the most probable and the second most probable class (for the Bayesian) should be considered equivalent to returning two classes (for the credal). A metric able to rigorously compare credal and Bayesian classifier could be very important to allow credal classifiers to become widespread.

11 Tree-Augmented Naive Bayes (TAN)

In [47], NBC and Bayesian Networks (BNs) whose topology has been learned from data, have been compared in classification; surprisingly BNs, despite their much higher flexibility, did not outperform NBC. This can be explained through the bias-variance decomposition of the misclassification error, which we already mentioned in Section 8: BNs have much lower bias than NBC, but this effect is often not felt, because of their high variance. However, these results were the inspiration for developing an effective compromise between BNs and NBC, yielding the so-called *tree-augmented naive Bayes* (TAN) [47], which is defined as follows (see also Figure 14):

- each feature has the class as a parent;
- each feature can also have an additional second parent, constituted by another feature.

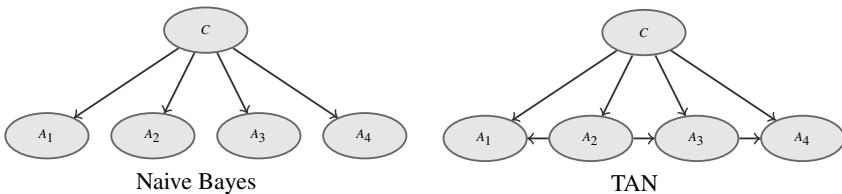


Fig. 14. TAN can model dependencies between features, unlike naive Bayes

In [47], TAN has been shown to be generally more accurate than both NBC and BNs. More recent results [59] point out a flaw regarding the usage of BNs in [47]; however, even after fixing this problem, the results confirm that TAN is generally more accurate

than both NBC and BNs (although the advantage of TAN over BNs is less marked than previously reported, and moreover BNs are now shown to be indeed more accurate than NBC).

This justifies the interest for designing a credal TAN. Before reviewing its development it is however necessary to discuss the different variants of the IDM which can be used for classification.

11.1 Variants of the Imprecise Dirichlet Model: Local and Global IDM

Given a credal network, three kinds of IDM can be used: the *global*, the *local* and the recently introduced Extreme Dirichlet Model (EDM) [16]. In the following, we show the differences between these approaches, using the example network $C \rightarrow A$.

Let us focus on the class node. The constraints which define the set of Dirichlet distributions for the IDM (both local and global) are:

$$T_C = \begin{cases} \sum_{c \in \Omega_C} t(c) = 1 \\ t(c) > 0 \end{cases} \quad \forall c \in \Omega_C. \quad (28)$$

As in Equation (7), the credal set $K(C)$ contains the mass functions of type $P(C)$, which allows the probability of class c to vary within the interval:

$$P(c) \in \left[\frac{n(c)}{s + \sum_{c \in \Omega_C} n(c)}, \frac{s + n(c)}{s + \sum_{c \in \Omega_C} n(c)} \right]. \quad (29)$$

The EDM restricts the set of priors defined by Eq.(28) to its most extreme elements, i.e., each $t(c)$ can be only zero or one. Consequently, the probability of class c corresponds either to the upper or to the lower bound of the interval in Equation (29).

Let us now move to conditional probabilities. The local IDM defines the polytope similarly to Equation (28):

$$T_{A|C} = \begin{cases} \sum_{a \in \Omega_A} t(a, c) = 1 & \forall c \in \Omega_C \\ t(a, c) > 0 & \forall a \in \Omega_A, \forall c \in \Omega_C. \end{cases} \quad (30)$$

Note that there is no relation between the $t(a, c)$ and the $t(c)$ previously used for the class node. For each $c \in \Omega_C$, the credal set $K(A|c)$ contains the mass functions of type $P(A|c)$, which let its probabilities vary as follows:

$$P(a|c) \in \left[\frac{n(a, c)}{s + n(c)}, \frac{s + n(a, c)}{s + n(c)} \right]. \quad (31)$$

The credal sets $\{K(A|c)\}_{c \in \Omega_C}$ and $K(C)$ are thus specified one independently of the others. Following the terminology of Section 4.2, the model is a *separately specified* credal network.

Instead, to understand the estimate of the conditional probabilities under the *global* IDM, we should recall that it is based on a set of *joint* Dirichlet distributions, defined by the constraints (already given in Section 8):

$$T_{A,C} = \begin{cases} \sum_{c \in \Omega_C} t(c) = 1 \\ t(c) > 0 & \forall c \in \Omega_C \\ \sum_a t(a,c) = t(c) & \forall c \in \Omega_C \\ t(a,c) > 0 & \forall a \in \Omega_A, \forall c \in \Omega_C. \end{cases} \quad (32)$$

In particular, the third constraint introduces a link between $t(a,c)$ and $t(c)$, which is missing in the local IDM; therefore, the network is *not* separately specified. Given the value of $t(c)$, the credal set $K(A|c)$ contains the mass functions $P(A|c)$ such that:

$$P(a|c) \in \left[\frac{n(c,a)}{st(c) + n(c)}, \frac{st(c) + n(c,a)}{st(c) + n(c)} \right]. \quad (33)$$

The global IDM estimates narrower intervals than the local, as can be seen by comparing Equation (33) and Equation (31)²³: this implies less indeterminacy in classification. Yet, the global IDM poses challenging computational problems; so far, exact computation with the global IDM has been possible only with NCC. Instead, the local IDM can be computed for any network and is in fact the common choice for general credal networks; yet, it returns wider intervals.

The EDM restricts the *global* IDM to its extreme distributions; it therefore allows $t(a,c)$ to be either 0 or $t(c)$, keeping the constraint $\forall c \in \Omega_C : \sum_{a \in \Omega_A} t(a,c) = t(c)$ inherited from the global IDM. The extreme points of the EDM corresponds in this case to the bounds of the interval in Equation (33); but in general, they are a inner approximation of the extremes of the global IDM [16]. From a different viewpoint, the EDM can be interpreted as treating the s hidden instances as s rows of non-MAR missing data, but with the additional assumption that such rows are all *identical* to each other; ignorance is due to the fact that it is unknown which values they contain.

The approximation provided by the EDM has been experimentally validated [25] by comparing the classification produced by NCC under the global IDM and the EDM; NCC produces almost identical results in the two settings, and thus the EDM appears as a reliable approximation of the global IDM, with the advantage of a simplified computation.

12 Credal TAN

As already discussed, the computational problems posed by the global IDM are quite challenging and imply a large computational overload for non-naive topologies. Thus, over years alternative solutions have been investigated.

A credal TAN was firstly proposed in [84], using the local IDM. The classifier was indeed reliable and very accurate when returning a single class but it was excessively cautious because of the local IDM. We refer to this algorithm as TANC*.

In [25], a credal TAN has been designed using the EDM; we refer this algorithm as TANC. As shown in Fig.15(a), TANC is more determinate than TANC*, because the EDM is an inner approximation of the global IDM, which in turn computes narrower

²³ Recall that $\sum_c t(c) = 1$ and that $t(c) > 0 \forall c \in \Omega_C$.

intervals than the local IDM. More important, TANC consistently achieves higher discounted accuracy than TANC*, as shown in Fig.15(b); therefore, it realizes a better trade-off between informativeness and reliability. However, the two classifiers have the same performance on the kr-kp data set, which contains few thousands of instances and only binary features; in this case, the model of prior ignorance has little importance.

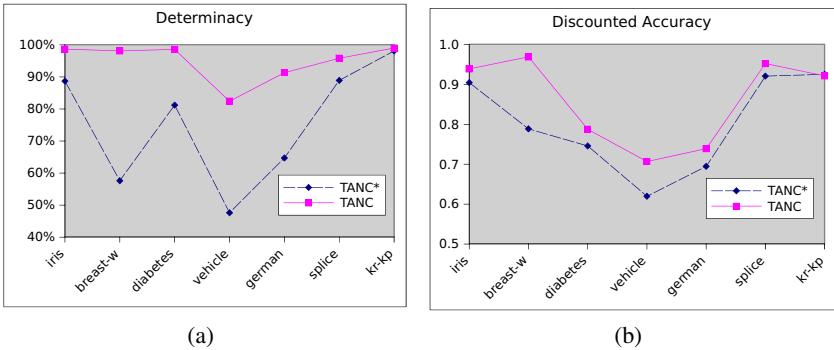


Fig. 15. Comparison of TANC* and TANC. Plot (a) shows the determinacy (% of determinate classifications) of the classifiers on different data sets, while plot (b) shows their discounted accuracy.

TANC is moreover good at spotting instances over which the Bayesian TAN becomes unreliable, similarly to how NCC does with NBC. In [25], experiments over some 40 UCI data sets show an average drop of 30 points of accuracy for the Bayesian TAN between the instances determinately and indeterminately classified by TANC. Instead, TANC preserves reliability also on the prior-dependent instances,²⁴ thanks to indeterminate classifications.

Although TANC is consistently more determinate than TANC*, it becomes sometimes largely indeterminate, especially on small data sets characterized by many classes and/or categorical values of the features. In fact, the TAN architecture (learned using a MDL criterion implemented within WEKA [79]), sometimes assigns the second parent to a feature, even though the resulting contingency table contains many counts which are numerically small. When parsed by TANC, they generate prior-dependent classifications and thus indeterminacy.

This also causes TANC to be slightly outperformed by NCC, as shown by the scatterplot of the discounted accuracy of Fig.16; therefore, TANC loses the advantage which the Bayesian TAN has over NBC. In [25], it is hypothesized that an algorithm for learning the structure more suitable for TANC should return less parameterized structures and could allow a significant performance improvement. Such algorithm should be able to return even a naive structure, if for instance modelling further dependencies makes

²⁴ Note that different credal classifiers, encoding a different probabilistic assumptions, might judge the same instance as prior-dependent or not. Thus, an instance is *not* prior-dependent per se, but according to the judgment of a certain credal classifier.

the joint distribution too sensitive on the prior. Previous attempts for structure learning based on imprecise probability can be found in [85]; yet this field has not been extensively explored and constitutes an interesting area for future research.

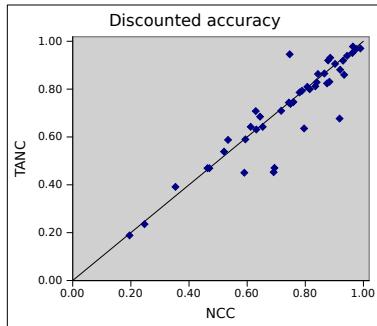


Fig. 16. Discounted accuracy of TANC and NCC

TANC is moreover able to conservatively deal with non-MAR missing data [25]. However, the treatment of missing data is at an earlier stage compared to that of NCC, as all missing data are currently treated as non-MAR (it is currently no possible to deal with a mix of MAR and non-MAR features) and moreover the current algorithms are not yet developed to deal with non-MAR missing data in the instance to classify. Preliminary results [25] show that, when faced with incomplete training sets, TANC is much more indeterminate than NCC but achieves a similar discounted accuracy.

13 Further Credal Classifiers

13.1 Lazy NCC (LNCC)

Besides the TAN approach, a further possibility of reducing the bias due to the naive assumption is to combine NCC and *lazy learning*; this has been explored in [30].

Lazy learning defers the training, until it has to classify an instance (*query*). In order to classify an instance, a lazy algorithm:

1. ranks the instances of the training set according to the distance from the query;
2. trains a local classifier on the k instances nearest to the query and returns the classification using the local classifier;
3. discards the locally trained classifier and keeps the training set in memory in order to answer new queries.

Lazy classifiers are *local*, as they get trained on the subset of instances which are nearest to the query. The parameter k (*bandwidth*) controls the bias-variance trade-off for lazy learning. In particular, a smaller bandwidth implies a smaller bias (even a simple model can fit a complex function on a small subset of data) at a cost of a larger variance

(as there are less data for estimating the parameters). Therefore, learning locally NBC (or NCC) can be a winning strategy as it allows reducing the bias; moreover, it also reduces the chance of encountering strong dependencies between features [45]. In fact, a successful example of lazy NBC is given in [45].

However, an important problem dealing with lazy learning is how to select the bandwidth k . The simplest approach is to empirically choose k (for instance, by cross-validation on the training set) and to then use the same k to answer all queries. However, the performance of lazy learning can significantly improve if the bandwidth is adapted query-by-query, as shown in [12] in the case of regression.

LNCC tunes the bandwidth query-by-query using a criterion based on imprecise probability. After having ranked the instances according to their distance from the query, a local NCC is induced on the k_{min} closest instances (for instance, $k_{min} = 25$) and classifies the instance. The classification is accepted if determinate; otherwise, the local NCC is updated by adding a set of further k_{upd} instances (we set $k_{upd} = 20$) to its training set. The procedure continues until either the classification is determinate or all instances have been added to the training of the local NCC. Therefore, the bandwidth is increased until the locally collected data smooth the effect of the choice of the prior. The naive architecture makes it especially easy updating LNCC with the k_{upd} instances; it only requires to update the counts $n(\cdot)$ that are internally stored by LNCC.

By design LNCC is thus generally more determinate than NCC; this also helps addressing the excessive determinacy which sometimes characterizes also NCC [24]. In [30] that generally LNCC outperforms NCC, both according to the discounted accuracy and the rank test.

13.2 Credal Model Averaging (CMA)

Model uncertainty is the problem of having multiple models which provide a good explanation of the data, but lead to different answers when used to make inference. In this case, selecting a single model underestimates uncertainty, as the uncertainty about model selection is ignored. *Bayesian model averaging* (BMA) [52] addressed model uncertainty by averaging over a set of candidate models rather than selecting a single candidate; each model is given a weight corresponding to its posterior probability.

In case of NBC, given k features, there are 2^k possible NBCs, each characterized by a different subset of features; we denote by \mathcal{M} the set of such models and by m a generic model of the set. Using BMA, the posterior probability $P(c, \mathbf{a}|\mathbf{n}, \mathbf{t})$ is computed by averaging over *all* the 2^k different NBCs, namely by marginalizing m out:

$$P(c, \mathbf{a}|\mathbf{n}, \mathbf{t}) \propto \sum_{m \in \mathcal{M}} P(c, \mathbf{a}|\mathbf{n}, \mathbf{t}, m) P(\mathbf{n}|m) P(m), \quad (34)$$

where $P(m)$ and $P(\mathbf{n}|m) = \int P(\mathbf{n}|m, \mathbf{t}) P(\mathbf{t}|m) d\mathbf{t}$ are respectively the prior probability and the marginal likelihood of model m ; the posterior probability of model m is $P(m|\mathbf{n}, \mathbf{t}) \propto P(\mathbf{n}|m, \mathbf{t}) P(m|\mathbf{t})$.

BMA implies two main challenges [19]: the computation of the exhaustive sum of Eq.(34) and the choice of the prior distribution over the models.

The computation of BMA is difficult, because the sum of Eq. (34) is often intractable; in fact, BMA is often computed via algorithms which are both approximated and time-consuming. However, Dash and Cooper [36] provide an exact and efficient algorithm to compute BMA over 2^k NBCs.

As for the choice of the prior, a common choice is to assign equal probability to all models; however, this is criticized from different standpoints even in the literature of BMA (see the rejoinder of [52]). Moreover, as already discussed, the specification of any single prior implies arbitrariness and entails the risk of issuing prior-dependent classifications. Our view is that this problem should be addressed by using a credal set rather than a single prior. However, in the following it is understood that by BMA we mean BMA learned with the uniform prior over the models.

Credal set. Credal model averaging (CMA) [27] extends to imprecise probabilities the BMA over NBCs of [36], substituting the single prior over the models by a credal set. The prior probability of model m is expressed by Dash and Cooper [36] as:

$$P(g) = \prod_{i \in m} P_i \prod_{i \notin m} (1 - P_i), \quad (35)$$

where P_i is the probability of feature i to be relevant for the problem, while $i \in g$ and $i \notin g$ index respectively the features included and excluded from model g . By setting $P_i := 0.5$ for all i , all models are given the same prior probability.

CMA is aimed at modelling a condition close to *prior ignorance* about the relative credibility of the 2^k NBCs, which also implies ignorance about whether each feature is relevant or not; the credal set $K(M)$ of prior over the models is given by all the mass function obtained by letting vary each P_i within the interval $\varepsilon < P_i < 1 - \varepsilon$ (the introduction of the $\varepsilon > 0$ is necessary to enable learning from the data).

Denoting as $P(M)$ a generic mass function over the graphs, the test of credal-dominance test of CMA is:

$$\inf_{P(M) \in K(M)} \frac{\sum_{g \in \mathcal{G}} P(c_1|g, \mathbf{n}) P(\mathbf{n}|g) P(g)}{\sum_{g \in \mathcal{G}} P(c_2|g, \mathbf{n}) P(\mathbf{n}|g) P(g)} > 1. \quad (36)$$

The computation of the dominance test is accomplished by extending to imprecise probability the BMA algorithm by [36]; see [27] for more details.

Since $K(M)$ contains the uniform prior over the models, the set of non-dominated classes of CMA always contains the most probable class identified by BMA; for the same reasons CMA, when determinate, returns the same class of BMA.

The experiments of [27] shows that the accuracy of BMA sharply drops on the instances where CMA gets indeterminate. The finding that a Bayesian classifier is little accurate on the instances indeterminately classified by its counterpart based on imprecise probabilities is indeed consistent across the various credal classifiers which we have developed.

A possible research direction is the development of CMA for NCC, namely imprecise averaging over credal classifiers. Yet, attempts in this direction seem to involve quite difficult and time-consuming computations.

14 Open Source Software

JNCC2 [28] is the Java implementation of NCC; it is available from www.idsia.ch/~giorgio/jncc2.html and has a command-line interface. This software has been around since some years and is stable.

A second open-source software is a plug-in for the WEKA [79] environment; it implements NCC, LNCC, CMA and the credal version of classification trees [1]. Thanks to the WEKA environment, all the operations with credal classifiers can be performed graphically and moreover many powerful tools (e.g., feature selection) become available to be readily used with credal classifiers. This software is available from <http://decsai.ugr.es/~andrew/weka-ip.html>; it is very recent and thus should be seen as more experimental.

15 Conclusions

Credal networks generalize Bayesian networks, providing a more robust probabilistic representation; in some cases, a single probability distribution cannot robustly describe uncertainty. Being able to work with a set of distributions rather than with a single distribution, credal networks can for instance robustly deal with the specification of the prior and with non-MAR missing data. Credal networks are naturally suited to model expert knowledge, as often the experts feel more confident in assigning to an event an interval of probability rather than a point-wise probability; in fact, knowledge-based systems are a natural application of credal networks. However, credal networks have been also thoroughly developed for classification. The main feature of credal classifiers is that they suspend the judgment returning a set classes; this happens for instance when the instance is prior-dependent or when too much uncertainty arises from missing data, when MAR cannot be assumed. Extensive experiments, performed both on public benchmark data sets and in real-world applications show that on the instances indeterminately classified by a credal network, the accuracy of its Bayesian counterpart (namely, a BN with the same graph, learned with the uniform distribution) drops. Directions for future research include the development of a more rigorous metric to compare credal and traditional probabilistic classifier and algorithms for structure learning especially tailored for credal networks.

Acknowledgements. The research has been partially supported by the Swiss NSF grants n. 200020_134759 / 1, 200020-121785 / 1, 200020-132252 and by the Hasler foundation grant n. 10030.

References

1. Abellán, J., Moral, S.: Upper entropy of credal sets. Applications to credal classification. *International Journal of Approximate Reasoning* 39(2-3), 235–255 (2005)
2. Antonucci, A., Brühlmann, R., Piatti, A., Zaffalon, M.: Credal networks for military identification problems. *International Journal of Approximate Reasoning* 50(4), 666–679 (2009)

3. Antonucci, A., Cuzzolin, F.: Credal sets approximation by lower probabilities: Application to credal networks. In: Hüllermeier, E., Kruse, R., Hoffmann, F. (eds.) IPMU 2010. LNCS, vol. 6178, pp. 716–725. Springer, Heidelberg (2010)
4. Antonucci, A., Piatti, A., Zaffalon, M.: Credal networks for operational risk measurement and management. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part II. LNCS (LNAI), vol. 4693, pp. 604–611. Springer, Heidelberg (2007)
5. Antonucci, A., Salvetti, A., Zaffalon, M.: Credal networks for hazard assessment of debris flows. In: Kropp, J., Scheffran, J. (eds.) Advanced Methods for Decision Making and Risk Management in Sustainability Science. Nova Science Publishers, New York (2007)
6. Antonucci, A., Sun, Y., de Campos, C., Zaffalon, M.: Generalized loopy 2U: a new algorithm for approximate inference in credal networks. International Journal of Approximate Reasoning 51(5), 474–484 (2010)
7. Antonucci, A., Zaffalon, M.: Equivalence between Bayesian and credal nets on an updating problem. In: Lawry, J., Miranda, E., Bugarin, A., Li, S., Gil, M.A., Grzegorzewski, P., Hryniwicz, O. (eds.) Proceedings of Third International Conference on Soft Methods in Probability and Statistics (SMPS 2006), pp. 223–230. Springer, Heidelberg (2006)
8. Antonucci, A., Zaffalon, M.: Decision-theoretic specification of credal networks: A unified language for uncertain modeling with sets of bayesian networks. International Journal of Approximate Reasoning 49(2), 345–361 (2008)
9. Avis, D., Fukuda, K.: Reverse search for enumeration. Discrete Applied Mathematics 65, 21–46 (1996)
10. Benavoli, A., de Campos, C.P.: Inference from multinomial data based on a MLE-dominance criterion. In: Proc. on European Conf. on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (Ecsqaru), Verona, pp. 22–33 (2009)
11. Benavoli, A., Zaffalon, M., Miranda, E.: Reliable hidden Markov model filtering through coherent lower previsions. In: Proc. 12th Int. Conf. Information Fusion, Seattle (USA), pp. 1743–1750 (2009)
12. Bontempi, G., Birattari, M., Bersini, H.: Lazy learning for local modelling and control design. International Journal of Control 72(7), 643–658 (1999)
13. de Campos, C.P., Cozman, F.G.: Inference in credal networks through integer programming. In: Proceedings of the Fifth International Symposium on Imprecise Probability: Theories and Applications. Action M Agency, Prague (2007)
14. Campos, L., Huete, J., Moral, S.: Probability intervals: a tool for uncertain reasoning. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 2(2), 167–196 (1994)
15. Cano, A., Cano, J., Moral, S.: Convex sets of probabilities propagation by simulated annealing on a tree of cliques. In: Bouchon-Meunier, B., Yager, R.R., Zadeh, L.A. (eds.) IPMU 1994. LNCS, vol. 945, pp. 4–8. Springer, Heidelberg (1995)
16. Cano, A., Gómez-Olmedo, M., Moral, S.: Credal nets with probabilities estimated with an extreme imprecise Dirichlet model. In: de Cooman, G., Vejnarová, I., Zaffalon, M. (eds.) Proceedings of the Fifth International Symposium on Imprecise Probability: Theories and Applications (ISIPTA 2007), pp. 57–66. Action M Agency, Prague (2007)
17. Cano, A., Moral, S.: A review of propagation algorithms for imprecise probabilities. In: [38], pp. 51–60 (1999)
18. Cano, A., Moral, S.: Using probability trees to compute marginals with imprecise probabilities. International Journal of Approximate Reasoning 29(1), 1–46 (2002)

19. Clyde, M., George, E.: Model uncertainty. *Statistical Science*, pp. 81–94 (2004)
20. Coolen, F.P.A., Augustin, T.: Learning from multinomial data: a nonparametric predictive alternative to the Imprecise Dirichlet Model. In: ISIPTA 2005: Proceedings of the Fourth International Symposium on Imprecise Probabilities and their Applications, pp. 125–134 (2005)
21. de Cooman, G., Hermans, F., Antonucci, A., Zaffalon, M.: Epistemic irrelevance in credal networks: the case of imprecise markov trees. *International Journal of Approximate Reasoning* (accepted for publication)
22. de Cooman, G., Miranda, E., Zaffalon, M.: Independent natural extension. In: Hüllermeier, E., Kruse, R., Hoffmann, F. (eds.) IJCAI 2010. LNCS, vol. 6178, pp. 737–746. Springer, Heidelberg (2010)
23. Cooper, G.F.: The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence* 42, 393–405 (1990)
24. Corani, G., Benavoli, A.: Restricting the IDM for classification. In: Hüllermeier, E., Kruse, R., Hoffmann, F. (eds.) IJCAI 2010. CCIS, vol. 80, pp. 328–337. Springer, Heidelberg (2010)
25. Corani, G., de Campos, C.P.: A tree-augmented classifier based on Extreme Imprecise Dirichlet Model. *International Journal of Approximate Reasoning* (accepted for publication)
26. Corani, G., Giusti, A., Migliore, D.: Robust texture recognition using imprecise classification. Under Review
27. Corani, G., Zaffalon, M.: Credal model averaging: An extension of bayesian model averaging to imprecise probabilities. In: Proc. of the 2008 European Conf. on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2008), pp. 257–271. Springer, Heidelberg (2008)
28. Corani, G., Zaffalon, M.: JNCC2: The Java implementation of naive credal classifier 2. *Journal of Machine Learning Research* 9, 2695–2698 (2008)
29. Corani, G., Zaffalon, M.: Learning reliable classifiers from small or incomplete data sets: the naive credal classifier 2. *Journal of Machine Learning Research* 9, 581–621 (2008)
30. Corani, G., Zaffalon, M.: Lazy naive credal classifier. In: Proc. of the 1st ACM SIGKDD Workshop on Knowledge Discovery from Uncertain Data, pp. 30–37. ACM, New York (2009)
31. Costa, J.E., Fleisher, P.J. (eds.): Physical geomorphology of debris flows, ch. 9, pp. 268–317. Springer, Berlin (1984)
32. Couso, I., Moral, S., Walley, P.: Examples of independence for imprecise probabilities. In: ISIPTA, pp. 121–130 (1999)
33. del Coz, J., Diez, J., Bahamonde, A.: Learning Nondeterministic Classifiers. *Journal of Machine Learning Research* 10, 2273–2293 (2009)
34. Cozman, F.G.: Robustness analysis of Bayesian networks with finitely generated convex-sets of distributions. Tech. Rep. CMU-RI-TR 96-41, Robotics Institute, Carnegie Mellon University (1996)
35. Cozman, F.G.: Credal networks. *Artificial Intelligence* 120, 199–233 (2000)
36. Dash, D., Cooper, G.: Model averaging for prediction with discrete Bayesian networks. *Journal of Machine Learning Research* 5, 1177–1203 (2004)
37. de Campos, C.P., Cozman, F.G.: The inferential complexity of Bayesian and credal networks. In: Proceedings of the International Joint Conference on Artificial Intelligence, Edinburgh, pp. 1313–1318 (2005)
38. de Cooman, G., Cozman, F.G., Moral, S., Walley, P.: ISIPTA 1999: Proceedings of the First International Symposium on Imprecise Probabilities and Their Applications. The Imprecise Probability Project, Universiteit Gent, Belgium (1999)

39. de Cooman, G., Zaffalon, M.: Updating beliefs with incomplete observations. *Artificial Intelligence* 159, 75–125 (2004)
40. Domingos, P., Pazzani, M.: On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning* 29(2/3), 103–130 (1997)
41. Elkan, C.: Magical thinking in data mining: lessons from CoIL challenge 2000. In: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 426–431. ACM, New York (2001)
42. Fayyad, U.M., Irani, K.B.: Multi-interval Discretization of Continuous-valued Attributes for Classification Learning. In: *Proc. of the 13th International Joint Conference on Artificial Intelligence*, pp. 1022–1027. Morgan Kaufmann, San Francisco (1993)
43. de Finetti, B.: Theory of Probability. Wiley, New York (1974); Two volumes translated from Teoria Delle probabilità, published 1970. The second volume appeared under the same title in 1975
44. Fleuret, F.: Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research* 5, 1531–1555 (2004)
45. Frank, E., Hall, M., Pfahringer, B.: Locally weighted naive Bayes. In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 249–256 (2003)
46. Friedman, J.: On bias, variance, 0/1 - loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery* 1, 55–77 (1997)
47. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. *Machine Learning* 29(2), 131–163 (1997)
48. Griffiths, P.G., Webb, R.H., Melis, T.S.: Frequency and initiation of debris flows in grand canyon, arizona. *Journal of Geophysical Research* 109, 4002–4015 (2004)
49. Ha, V., Doan, A., Vu, V., Haddawy, P.: Geometric foundations for interval-based probabilities. *Annals of Mathematics and Artificial Intelligence* 24(1-4), 1–21 (1998)
50. Hand, D., Yu, K.: Idiot's Bayes-Not So Stupid After All? *International Statistical Review* 69(3), 385–398 (2001)
51. Hoare, Z.: Landscapes of naive Bayes classifiers. *Pattern Analysis & Applications* 11(1), 59–72 (2008)
52. Hoeting, J., Madigan, D., Raftery, A., Volinsky, C.: Bayesian model averaging: A tutorial. *Statistical Science* 14(4), 382–401 (1999)
53. Ide, J.S., Cozman, F.G.: IPE and L2U: Approximate algorithms for credal networks. In: *Proceedings of the Second Starting AI Researcher Symposium*, pp. 118–127. IOS Press, Amsterdam (2004)
54. Jaeger, M.: Ignorability for categorical data. *Annals of Statistics*, 1964–1981 (2005)
55. Kohavi, R.: Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 202–207. AAAI Press, Menlo Park (1996)
56. Kohavi, R., Becker, B., Sommerfield, D.: Improving simple Bayes. In: van Someren, M., Widmer, G. (eds.) *ECML 1997. LNCS*, vol. 1224, pp. 78–87. Springer, Heidelberg (1997)
57. Levi, I.: *The Enterprise of Knowledge*. MIT Press, London (1980)
58. Little, R.J.A., Rubin, D.B.: *Statistical Analysis with Missing Data*. Wiley, New York (1987)
59. Madden, M.: On the classification performance of TAN and general Bayesian networks. *Knowledge-Based Systems* 22(7), 489–495 (2009)
60. Manski, C.F.: *Partial Identification of Probability Distributions*. Springer, New York (2003)
61. Murphy, K., Weiss, Y., Jordan, M.: Loopy belief propagation for te inference: An empirical study. In: *Conference on Uncertainty in Artificial Intelligence*, pp. 467–475. Morgan Kaufmann, San Francisco (1999)
62. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 701–706 (2002)

63. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo (1988)
64. Piatti, A., Antonucci, A., Zaffalon, M.: Building knowledge-based systems by credal networks: a tutorial. In: Baswell, A.R. (ed.) Advances in Mathematics Research, vol. 11, Nova Science Publishers, New York (2010)
65. Ramoni, M., Sebastiani, P.: Robust learning with missing data. *Machine Learning* 45(2), 147–170 (2001)
66. Ferreira da Rocha, J.C., Cozman, F.G.: Inference with separately specified sets of probabilities in credal networks. In: Darwiche, A., Friedman, N. (eds.) Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI 2002), pp. 430–437. Morgan Kaufmann, San Francisco (2002)
67. Ferreira da Rocha, J.C., Cozman, F.G.: Inference in credal networks with branch-and-bound algorithms. In: Bernard, J.M., Seidenfeld, T., Zaffalon, M. (eds.) ISIPTA Proceedings in Informatics, vol. 18, pp. 480–493. Carleton Scientific (2003)
68. da Rocha, J.C., Cozman, F.G., de Campos, C.P.: Inference in polytrees with sets of probabilities. In: Conference on Uncertainty in Artificial Intelligence, pp. 217–224. Acapulco (2003)
69. Rubin, D.B.: Inference and missing data. *Biometrika* 63(3), 581–592 (1976)
70. Takahashi, T.: Debris Flow. iAHR Monograph. A.A. Balkema, Rotterdam (1991)
71. Tessem, B.: Interval probability propagation. *International Journal of Approximate Reasoning* 7(3), 95–120 (1992)
72. Troffaes, M.: Decision making with imprecise probabilities: A short review. In: Cozman, F.G. (ed.) SIPTA Newsletter. Society for Imprecise Probability Theory and Applications, Manno, Switzerland, pp. 4–7 (December 2004)
73. Tsoumakas, G., Vlahavas, I.: Random k-Labelsets: An Ensemble Method for Multilabel Classification. In: Proceedings of the 18th European Conference on Machine Learning, pp. 406–417. Springer, Heidelberg (2007)
74. Van Der Putten, P., Van Someren, M.: A bias-variance analysis of a real world learning problem: The CoIL challenge 2000. *Machine Learning* 57(1), 177–195 (2004)
75. Walley, P.: Statistical Reasoning with Imprecise Probabilities. Chapman and Hall, New York (1991)
76. Walley, P.: Inferences from multinomial data: learning about a bag of marbles. *J. R. Statist. Soc. B* 58(1), 3–57 (1996)
77. Walley, P.: Statistical Reasoning with Imprecise Probabilities. Monographs on Statistics and Applied Probability, vol. 42. Chapman and Hall, London (1991)
78. Walley, P.: Inferences from multinomial data: Learning about a bag of marbles. *Journal of the Royal Statistical Society SeriesB* 58(1), 3–34 (1996)
79. Witten, I., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco (2005)
80. Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., et al.: Top 10 algorithms in data mining. *Knowledge and Information Systems* 14(1), 1–37 (2008)
81. Zaffalon, M.: Statistical inference of the naive credal classifier. In: de Cooman, G., Fine, T.L., Seidenfeld, T. (eds.) ISIPTA 2001: Proceedings of the Second International Symposium on Imprecise Probabilities and Their Applications, pp. 384–393. Shaker, The Netherlands (2001)
82. Zaffalon, M.: Conservative rules for predictive inference with incomplete data. In: Cozman, F.G., Nau, R., Seidenfeld, T. (eds.) Proceedings of the Fourth International Symposium on Imprecise Probabilities and Their Applications (ISIPTA 2005), pp. 406–415. SIPTA (2005)

83. Zaffalon, M.: Credible classification for environmental problems. *Environmental Modelling & Software* 20(8), 1003–1012 (2005)
84. Zaffalon, M., Fagioli, E.: Tree-based credal networks for classification. *Reliable Computing* 9(6), 487–509 (2003)
85. Zaffalon, M., Hutter, M.: Robust inference of trees. *Annals of Mathematics and Artificial Intelligence* 45(1), 215–239 (2005)
86. Zaffalon, M., Miranda, E.: Conservative Inference Rule for Uncertain Reasoning under Incompleteness. *Journal of Artificial Intelligence Research* 34, 757–821 (2009)
87. Zaffalon, M., Wesnes, K., Petrini, O.: Reliable diagnoses of dementia by the naive credal classifier inferred from incomplete cognitive data. *Artificial Intelligence in Medicine* 29(1-2), 61–79 (2003)

Chapter 5

Hierarchical Clustering for Finding Symmetries and Other Patterns in Massive, High Dimensional Datasets

Fionn Murtagh^{1,2} and Pedro Contreras²

¹ Science Foundation Ireland, Wilton Park House,
Wilton Place, Dublin 2, Ireland

² Department of Computer Science
Royal Holloway, University of London
Egham TW20 0EX, UK
fmurtagh@acm.org

Abstract. Data analysis and data mining are concerned with unsupervised pattern finding and structure determination in data sets. “Structure” can be understood as symmetry and a range of symmetries are expressed by hierarchy. Such symmetries directly point to invariants, that pinpoint intrinsic properties of the data and of the background empirical domain of interest. We review many aspects of hierarchy here, including ultrametric topology, generalized ultrametric, linkages with lattices and other discrete algebraic structures and with p-adic number representations. By focusing on symmetries in data we have a powerful means of structuring and analyzing massive, high dimensional data stores. We illustrate the powerfulness of hierarchical clustering in case studies in chemistry and finance, and we provide pointers to other published case studies.

Keywords: Data analytics, multivariate data analysis, pattern recognition, information storage and retrieval, clustering, hierarchy, p-adic, ultrametric topology, complexity.

1 Introduction: Hierarchy and Other Symmetries in Data Analysis

Herbert A. Simon, Nobel Laureate in Economics, originator of “bounded rationality” and of “satisficing”, believed in hierarchy at the basis of the human and social sciences, as the following quotation shows: “... my central theme is that complexity frequently takes the form of hierarchy and that hierarchic systems have some common properties independent of their specific content. Hierarchy, I shall argue, is one of the central structural schemes that the architect of complexity uses.” ([74], p. 184.)

Partitioning a set of observations [75, 76, 49] leads to some very simple symmetries. This is one approach to clustering and data mining. But such approaches, often based on optimization, are not of direct interest to us here. Instead we

will pursue the theme pointed to by Simon, namely that the notion of hierarchy is fundamental for interpreting data and the complex reality which the data expresses. Our work is very different too from the marvelous view of the development of mathematical group theory – but viewed in its own right as a complex, evolving system – presented by Foote [19].

Weyl [80] makes the case for the fundamental importance of symmetry in science, engineering, architecture, art and other areas. As a “guiding principle”, “Whenever you have to do with a structure-endowed entity ... try to determine its group of automorphisms, the group of those element-wise transformations which leave all structural relations undisturbed. You can expect to gain a deep insight in the constitution of [the structure-endowed entity] in this way. After that you may start to investigate symmetric configurations of elements, i.e. configurations which are invariant under a certain subgroup of the group of all automorphisms; ...” ([80], p. 144).

1.1 About This Article

In section 2, we describe ultrametric topology as an expression of hierarchy. This provides comprehensive background on the commonly used quadratic computational time (i.e., $O(n^2)$, where n is the number of observations) agglomerative hierarchical clustering algorithms.

In section 3, we look at the generalized ultrametric context. This is closely linked to analysis based on lattices. We use a case study from chemical database matching to illustrate algorithms in this area.

In section 4, p -adic encoding, providing a number theory vantage point on ultrametric topology, gives rise to additional symmetries and ways to capture invariants in data.

Section 5 deals with symmetries that are part and parcel of a tree, representing a partial order on data, or equally a set of subsets of the data, some of which are embedded. An application of such symmetry targets from a dendrogram expressing a hierarchical embedding is provided through the Haar wavelet transform of a dendrogram and wavelet filtering based on the transform.

Section 6 deals with new and recent results relating to the remarkable symmetries of massive, and especially high dimensional data sets. An example is discussed of segmenting a financial forex (foreign exchange) trading signal.

1.2 A Brief Introduction to Hierarchical Clustering

For the reader new to analysis of data a very short introduction is now provided on hierarchical clustering. Along with other families of algorithm, the objective is automatic classification, for the purposes of data mining, or knowledge discovery. Classification, after all, is fundamental in human thinking, and machine-based decision making. But we draw attention to the fact that our objective is *unsupervised*, as opposed to *supervised* classification, also known as discriminant analysis or (in a general way) machine learning. So here we are *not* concerned with generalizing the decision making capability of training data, nor are we concerned

with fitting statistical models to data so that these models can play a role in generalizing and predicting. Instead we are concerned with having “data speak for themselves”. That this unsupervised objective of classifying data (observations, objects, events, phenomena, etc.) is a huge task in our society is unquestionably true. One may think of situations when precedents are very limited, for instance.

Among families of clustering, or unsupervised classification, algorithms, we can distinguish the following: (i) array permuting and other visualization approaches; (ii) partitioning to form (discrete or overlapping) clusters through optimization, including graph-based approaches; and – of interest to us in this article – (iii) embedded clusters interrelated in a tree-based way.

For the last-mentioned family of algorithm, agglomerative building of the hierarchy from consideration of object pairwise distances has been the most common approach adopted. As comprehensive background texts, see [48, 30, 81, 31].

1.3 A Brief Introduction to p-Adic Numbers

The real number system, and a p-adic number system for given prime, p , are potentially equally useful alternatives. p-Adic numbers were introduced by Kurt Hensel in 1898.

Whether we deal with Euclidean or with non-Euclidean geometry, we are (nearly) always dealing with reals. But the reals start with the natural numbers, and from associating observational facts and details with such numbers we begin the process of measurement. From the natural numbers, we proceed to the rationals, allowing fractions to be taken into consideration.

The following view of how we do science or carry out other quantitative study was proposed by Volovich in 1987 [78, 79]. See also the surveys in [15, 22]. We can always use rationals to make measurements. But they will be approximate, in general. It is better therefore to allow for observables being “continuous, i.e. endow them with a topology”. Therefore we need a completion of the field \mathbb{Q} of rationals. To complete the field \mathbb{Q} of rationals, we need Cauchy sequences and this requires a norm on \mathbb{Q} (because the Cauchy sequence must converge, and a norm is the tool used to show this). There is the Archimedean norm such that: for any $x, y \in \mathbb{Q}$, with $|x| < |y|$, then there exists an integer N such that $|Nx| > |y|$. For convenience here, we write: $|x|_\infty$ for this norm. So if this completion is Archimedean, then we have $\mathbb{R} = \mathbb{Q}_\infty$, the reals. That is fine if space is taken as commutative and Euclidean.

What of alternatives? Remarkably all norms are known. Besides the \mathbb{Q}_∞ norm, we have an infinity of norms, $|x|_p$, labeled by primes, p . By Ostrowski’s theorem [65] these are all the possible norms on \mathbb{Q} . So we have an unambiguous labeling, via p , of the infinite set of non-Archimedean completions of \mathbb{Q} to a field endowed with a topology.

In all cases, we obtain locally compact completions, \mathbb{Q}_p , of \mathbb{Q} . They are the fields of p-adic numbers. All these \mathbb{Q}_p are continua. Being locally compact, they have additive and multiplicative Haar measures. As such we can integrate over them, such as for the reals.

1.4 Brief Discussion of p-Adic and m-Adic Numbers

We will use p to denote a prime, and m to denote a non-zero positive integer. A p -adic number is such that any set of p integers which are in distinct residue classes modulo p may be used as p -adic digits. (Cf. remark below, at the end of section 4.1, quoting from [25]. It makes the point that this opens up a range of alternative notation options in practice.) Recall that a ring does not allow division, while a field does. m -Adic numbers form a ring; but p -adic numbers form a field. So a priori, 10 -adic numbers form a ring. This provides us with a reason for preferring p -adic over m -adic numbers.

We can consider various p -adic expansions:

1. $\sum_{i=0}^n a_i p^i$, which defines positive integers. For a p -adic number, we require $a_i \in 0, 1, \dots, p - 1$. (In practice: just write the integer in binary form.)
2. $\sum_{i=-\infty}^n a_i p^i$ defines rationals.
3. $\sum_{i=k}^{\infty} a_i p^i$ where k is an integer, not necessarily positive, defines the field \mathbb{Q}_p of p -adic numbers.

\mathbb{Q}_p , the field of p -adic numbers, is (as seen in these definitions) the field of p -adic expansions.

The choice of p is a practical issue. Indeed, adelic numbers use all possible values of p (see [6] for extensive use and discussion of the adelic number framework). Consider [14, 37]. DNA (deoxyribonucleic acid) is encoded using four nucleotides: A, adenine; G, guanine; C, cytosine; and T, thymine. In RNA (ribonucleic acid) T is replaced by U, uracil. In [14] a 5-adic encoding is used, since 5 is a prime and thereby offers uniqueness. In [37] a 4-adic encoding is used, and a 2-adic encoding, with the latter based on 2-digit boolean expressions for the four nucleotides (00, 01, 10, 11). A default norm is used, based on a longest common prefix – with p -adic digits from the start or left of the sequence (see section 4.2 below where this longest common prefix norm or distance is used and, before that, section 3.3 where an example is discussed in detail).

2 Ultrametric Topology

In this section we mainly explore symmetries related to: geometric shape; matrix structure; and lattice structures.

2.1 Ultrametric Space for Representing Hierarchy

Consider Figures 1 and 2, illustrating the ultrametric distance and its role in defining a hierarchy. An early, influential paper is Johnson [35] and an important survey is that of Rammal et al. [67]. Discussion of how a hierarchy expresses the semantics of change and distinction can be found in [61].

The ultrametric topology was introduced by Marc Krasner [40], the ultrametric inequality having been formulated by Hausdorff in 1934. Essential motivation for the study of this area is provided by [70] as follows. Real and complex fields

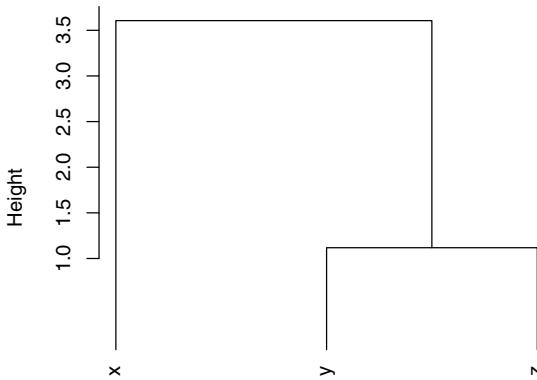


Fig. 1. The strong triangular inequality defines an ultrametric: every triplet of points satisfies the relationship: $d(x, z) \leq \max\{d(x, y), d(y, z)\}$ for distance d . Cf. by reading off the hierarchy, how this is verified for all x, y, z : $d(x, z) = 3.5$; $d(x, y) = 3.5$; $d(y, z) = 1.0$. In addition the symmetry and positive definiteness conditions hold for any pair of points.

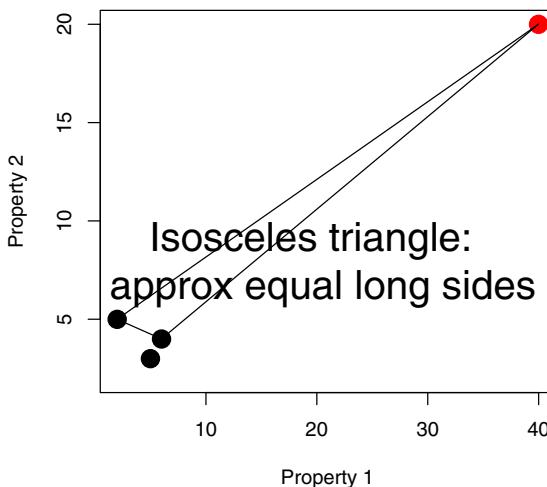


Fig. 2. How metric data can approximate an ultrametric, or can be made to approximate an ultrametric in the case of a stepwise, agglomerative algorithm. A “query” is on the far right. While we can easily determine the closest target (among the three objects represented by the dots on the left), is the closest really that much different from the alternatives? This question motivates an ultrametric view of the metric relationships shown.

gave rise to the idea of studying any field K with a complete valuation $| \cdot |$ comparable to the absolute value function. Such fields satisfy the “strong triangle inequality” $|x + y| \leq \max(|x|, |y|)$. Given a valued field, defining a totally ordered Abelian (i.e. commutative) group, an ultrametric space is induced through $|x - y| = d(x, y)$. Various terms are used interchangeably for analysis in and over such fields such as p-adic, ultrametric, non-Archimedean, and isosceles. The natural geometric ordering of metric valuations is on the real line, whereas in the ultrametric case the natural ordering is a hierarchical tree.

2.2 Some Geometrical Properties of Ultrametric Spaces

We see from the following, based on [41] (chapter 0, part IV), that an ultrametric space is quite different from a metric one. In an ultrametric space everything “lives” on a tree.

In an ultrametric space, all triangles are either isosceles with small base, or equilateral. We have here very clear symmetries of shape in an ultrametric topology. These symmetry “patterns” can be used to fingerprint data sets and time series: see [55, 57] for many examples of this.

Some further properties that are studied in [41] are: (i) Every point of a circle in an ultrametric space is a center of the circle. (ii) In an ultrametric topology, every ball is both open and closed (termed clopen). (iii) An ultrametric space is 0-dimensional (see [7, 69]). It is clear that an ultrametric topology is very different from our intuitive, or Euclidean, notions. The most important point to keep in mind is that in an ultrametric space everything “lives” in a hierarchy expressed by a tree.

2.3 Ultrametric Matrices and Their Properties

For an $n \times n$ matrix of positive reals, symmetric with respect to the principal diagonal, to be a matrix of distances associated with an ultrametric distance on X , a sufficient and necessary condition is that a permutation of rows and columns satisfies the following form of the matrix:

1. Above the diagonal term, equal to 0, the elements of the same row are non-decreasing.
2. For every index k , if

$$d(k, k+1) = d(k, k+2) = \cdots = d(k, k+\ell+1)$$

then

$$d(k+1, j) \leq d(k, j) \text{ for } k+1 < j \leq k+\ell+1$$

and

$$d(k+1, j) = d(k, j) \text{ for } j > k+\ell+1$$

Under these circumstances, $\ell \geq 0$ is the length of the section beginning, beyond the principal diagonal, the interval of columns of equal terms in row k .

Table 1. Input data: 8 iris flowers characterized by sepal and petal widths and lengths. From Fisher's iris data [17].

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
iris1	5.1	3.5	1.4	0.2
iris2	4.9	3.0	1.4	0.2
iris3	4.7	3.2	1.3	0.2
iris4	4.6	3.1	1.5	0.2
iris5	5.0	3.6	1.4	0.2
iris6	5.4	3.9	1.7	0.4
iris7	4.6	3.4	1.4	0.3

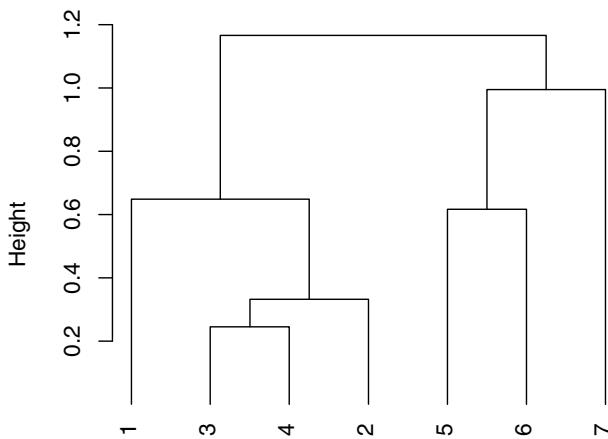


Fig. 3. Hierarchical clustering of 7 iris flowers using data from Table 1. No data normalization was used. The agglomerative clustering criterion was the minimum variance or Ward one.

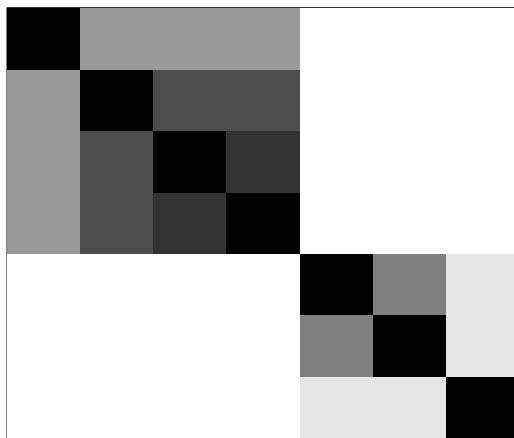
To illustrate the ultrametric matrix format, consider the small data set shown in Table 1. A dendrogram produced from this is in Figure 3. The ultrametric matrix that can be read off this dendrogram is shown in Table 2. Finally a visualization of this matrix, illustrating the ultrametric matrix properties discussed above, is in Figure 4.

2.4 Clustering through Matrix Row and Column Permutation

Figure 4 shows how an ultrametric distance allows a certain structure to be visible (quite possibly, in practice, subject to an appropriate row and column permuting), in a matrix defined from the set of all distances. For set X , then, this matrix expresses the distance mapping of the Cartesian product, $d : X \times X \rightarrow \mathbb{R}^+$. \mathbb{R}^+ denotes the non-negative reals. A priori the rows and columns of the

Table 2. Ultrametric matrix derived from the dendrogram in Figure 3

	iris1	iris2	iris3	iris4	iris5	iris6	iris7
iris1	0	0.6480741	0.6480741	0.6480741	1.1661904	1.1661904	1.1661904
iris2	0.6480741	0	0.3316625	0.3316625	1.1661904	1.1661904	1.1661904
iris3	0.6480741	0.3316625	0	0.2449490	1.1661904	1.1661904	1.1661904
iris4	0.6480741	0.3316625	0.2449490	0	1.1661904	1.1661904	1.1661904
iris5	1.1661904	1.1661904	1.1661904	1.1661904	0	0.6164414	0.9949874
iris6	1.1661904	1.1661904	1.1661904	1.1661904	0.6164414	0	0.9949874
iris7	1.1661904	1.1661904	1.1661904	1.1661904	0.9949874	0.9949874	0

**Fig. 4.** A visualization of the ultrametric matrix of Table 2, where bright or white = highest value, and black = lowest value

function of the Cartesian product set X with itself could be in any order. The ultrametric matrix properties establish what is possible when the distance is an ultrametric one. Because the matrix (a 2-way data object) involves one *mode* (due to set X being crossed with itself; as opposed to the 2-mode case where an observation set is crossed by an attribute set) it is clear that both rows and columns can be permuted to yield the *same* order on X . A property of the form of the matrix is that small values are at or near the principal diagonal.

A generalization opens up for this sort of clustering by visualization scheme. Firstly, we can directly apply row and column permuting to 2-mode data, i.e. to the rows and columns of a matrix crossing indices I by attributes J , $a : I \times J \rightarrow \mathbb{R}$. A matrix of values, $a(i, j)$, is furnished by the function a acting on the sets I and J . Here, each such term is real-valued. We can also generalize the principle of permuting such that small values are on or near the principal diagonal to instead allow similar values to be near one another, and thereby to facilitate visualization. An optimized way to do this was pursued in [45, 44]. Comprehensive surveys of clustering algorithms in this area, including objective functions,

visualization schemes, optimization approaches, presence of constraints, and applications, can be found in [46, 43]. See too [12, 53].

For all these approaches, underpinning them are row and column permutations, that can be expressed in terms of the permutation group, S_n , on n elements.

2.5 Other Miscellaneous Symmetries

As examples of various other local symmetries worthy of consideration in data sets consider subsets of data comprising clusters, and reciprocal nearest neighbor pairs.

Given an observation set, X , we define dissimilarities as the mapping $d : X \times X \longrightarrow \mathbb{R}^+$. A dissimilarity is a positive, definite, symmetric measure (i.e., $d(x, y) \geq 0$; $d(x, y) = 0$ if $x = y$; $d(x, y) = d(y, x)$). If in addition the triangular inequality is satisfied (i.e., $d(x, y) \leq d(x, z) + d(z, y), \forall x, y, z \in X$) then the dissimilarity is a distance.

If X is endowed with a metric, then this metric is mapped onto an ultrametric. In practice, there is no need for X to be endowed with a metric. Instead a dissimilarity is satisfactory.

A hierarchy, H , is defined as a binary, rooted, node-ranked tree, also termed a dendrogram [3, 35, 41, 53]. A hierarchy defines a set of embedded subsets of a given set of objects X , indexed by the set I . That is to say, object i in the object set X is denoted x_i , and $i \in I$. These subsets are *totally ordered* by an index function ν , which is a stronger condition than the *partial order* required by the subset relation. The index function ν is represented by the ordinate in Figure 3 (the “height” or “level”). A bijection exists between a hierarchy and an ultrametric space.

Often in this article we will refer interchangeably to the object set, X , and the associated set of indices, I .

Usually a constructive approach is used to induce H on a set I . The most efficient algorithms are based on nearest neighbor chains, which by definition end in a pair of agglomerable reciprocal nearest neighbors. Further information can be found in [50, 51, 53, 54].

3 Generalized Ultrametric

In this subsection, we consider an ultrametric defined on the power set or join semilattice. Comprehensive background on ordered sets and lattices can be found in [10]. A review of generalized distances and ultrametrics can be found in [72].

3.1 Link with Formal Concept Analysis

Typically hierarchical clustering is based on a distance (which can be relaxed often to a dissimilarity, not respecting the triangular inequality, and *mutatis mutandis* to a similarity), defined on all pairs of the object set: $d : X \times X \rightarrow \mathbb{R}^+$.

I.e., a distance is a positive real value. Usually we require that a distance cannot be 0-valued unless the objects are identical. That is the traditional approach.

A different form of ultrametrization is achieved from a dissimilarity defined on the power set of attributes characterizing the observations (objects, individuals, etc.) X . Here we have: $d : X \times X \rightarrow 2^J$, where J indexes the attribute (variables, characteristics, properties, etc.) set.

This gives rise to a different notion of distance, that maps pairs of objects onto elements of a join semilattice. The latter can represent all subsets of the attribute set, J . That is to say, it can represent the power set, commonly denoted 2^J , of J .

As an example, consider, say, $n = 5$ objects characterized by 3 boolean (presence/absence) attributes, shown in Figure 5 (top). Define dissimilarity between a pair of objects in this table as a *set* of 3 components, corresponding to the 3 attributes, such that if both components are 0, we have 1; if either component is 1 and the other 0, we have 1; and if both components are 1 we get 0. This is the simple matching coefficient [33]. We could use, e.g., Euclidean distance for each of the values sought; but we prefer to treat 0 values in both components as signaling a 1 contribution. We get then $d(a, b) = 1, 1, 0$ which we will call $\mathbf{d1}, \mathbf{d2}$. Then, $d(a, c) = 0, 1, 0$ which we will call $\mathbf{d2}$. Etc. With the latter we create lattice nodes as shown in the middle part of Figure 5.

In Formal Concept Analysis [10, 24], it is the lattice itself which is of primary interest. In [33] there is discussion of, and a range of examples on, the close relationship between the traditional hierarchical cluster analysis based on $d : I \times I \rightarrow \mathbb{R}^+$, and hierarchical cluster analysis “based on abstract posets” (a poset is a partially ordered set), based on $d : I \times I \rightarrow 2^J$. The latter, leading to clustering based on dissimilarities, was developed initially in [32].

3.2 Applications of Generalized Ultrametrics

As noted in the previous subsection, the usual ultrametric is an ultrametric distance, i.e. for a set I , $d : I \times I \rightarrow \mathbb{R}^+$. The generalized ultrametric is also consistent with this definition, where the range is a subset of the power set: $d : I \times I \rightarrow \Gamma$, where Γ is a partially ordered set. In other words, the *generalized* ultrametric distance is a set. Some areas of application of generalized ultrametrics will now be discussed.

In the theory of reasoning, a monotonic operator is rigorous application of a succession of conditionals (sometimes called consequence relations). However negation or multiple valued logic (i.e. encompassing intermediate truth and falsehood) require support for non-monotonic reasoning.

Thus [28]: “Once one introduces negation ... then certain of the important operators are not monotonic (and therefore not continuous), and in consequence the Knaster-Tarski theorem [i.e. for fixed points; see [10]] is no longer applicable to them. Various ways have been proposed to overcome this problem. One such [approach is to use] syntactic conditions on programs ... Another is to consider different operators ... The third main solution is to introduce techniques

	v_1	v_2	v_3
a	1	0	1
b	0	1	1
c	1	0	1
e	1	0	0
f	0	0	1

Potential lattice vertices	Lattice vertices found	Level
d1, d2, d3	d1, d2, d3	3
d1, d2 d2, d3 d1, d3	d1, d2 d2, d3	2
d1 d2 d3	d2	1

```

graph TD
    d1 --- d2 --- d3
    d1 --- d2
    d2 --- d3
    d1 --- d2 --- d3
  
```

The set d1,d2,d3 corresponds to: $d(b, e)$ and $d(e, f)$

The subset d1,d2 corresponds to: $d(a, b), d(a, f), d(b, c), d(b, f)$, and $d(c, f)$

The subset d2,d3 corresponds to: $d(a, e)$ and $d(c, e)$

The subset d2 corresponds to: $d(a, c)$

Clusters defined by all pairwise linkage at level ≤ 2 :

a, b, c, f

a, c, e

Clusters defined by all pairwise linkage at level ≤ 3 :

a, b, c, e, f

Fig. 5. Top: example data set consisting of 5 objects, characterized by 3 boolean attributes. Then: lattice corresponding to this data and its interpretation.

from topology and analysis to augment arguments based on order ... [the latter include:] methods based on metrics ... on quasi-metrics ... and finally ... on ultrametric spaces.”

The convergence to fixed points that are based on a generalized ultrametric system is precisely the study of spherically complete systems and expansive automorphisms discussed in section 4.3 below. As expansive automorphisms we see here again an example of symmetry at work.

3.3 Example of Application: Chemical Database Matching

In the 1990s, the Ward minimum variance hierarchical clustering method became the method of choice in the chemoinformatics community due to its hierarchical nature and the quality of the clusters produced. Unfortunately the method

reached its limits once the pharmaceutical companies tried processing datasets of more than 500,000 compounds due to: the $O(n^2)$ processing requirements of the reciprocal nearest neighbor algorithm; the requirement to hold all chemical structure “fingerprints” in memory to enable random access; and the requirement that parallel implementation use a shared-memory architecture. Let us look at an alternative hierarchical clustering algorithm that bypasses these computational difficulties.

A direct application of generalized ultrametrics to data mining is the following. The potentially huge advantage of the generalized ultrametric is that it allows a hierarchy to be read directly off the $I \times J$ input data, and bypasses the $O(n^2)$ consideration of all pairwise distances in agglomerative hierarchical clustering. In [62] we study application to chemoinformatics. Proximity and best match finding is an essential operation in this field. Typically we have one million chemicals upwards, characterized by an approximate 1000-valued attribute encoding.

Consider first our need to normalize the data. We divide each boolean (presence/absence) value by its corresponding column sum.

We can consider the hierarchical cluster analysis from abstract posets as based on $d : I \times I \rightarrow \mathbb{R}^{|J|}$. In [33], the median of the $|J|$ distance values is used, as input to a traditional hierarchical clustering, with alternative schemes discussed. See also [32] for an early elaboration of this approach.

Let us now proceed to take a particular approach to this, which has very convincing computational benefits.

3.3.1 Ultrametrization through Baire Space Embedding: Notation

A Baire space [42] consists of countably infinite sequences with a metric defined in terms of the longest common prefix: the longer the common prefix, the closer a pair of sequences. The Baire metric, and simultaneously ultrametric, will be defined in definition 1 in the next subsection. What is of interest to us here is this longest common prefix metric, which additionally is an ultrametric. The longest common prefixes at issue here are those of precision of any value (i.e., x_{ij} , for chemical compound i , and chemical structure code j). Consider two such values, x_{ij} and y_{ij} , which, when the context easily allows it, we will call x and y . Each are of some precision, and we take the integer $|K|$ to be the maximum precision. We pad a value with 0s if necessary, so that all values are of the same precision. Finally, we will assume for convenience that each value $\in [0, 1]$ and this can be arranged by normalization.

3.3.2 The Case of One Attribute

Thus we consider ordered sets x_k and y_k for $k \in K$. In line with our notation, we can write x_K and y_K for these numbers, with the set K now ordered. (So, $k = 1$ is the first decimal place of precision; $k = 2$ is the second decimal place; ... ; $k = |K|$ is the $|K|$ th decimal place.) The cardinality of the set K is the precision with which a number, x_K , is measured. Without loss of generality, through normalization, we will take all $x_K, y_K \leq 1$. We will also consider decimal

numbers, only, in this article (hence $x_k \in \{0, 1, 2, \dots, 9\}$ for all numbers x , and for all digits k), again with no loss of generality to non-decimal number representations.

Consider as examples $x_K = 0.478$; and $y_K = 0.472$. In these cases, $|K| = 3$. For $k = 1$, we find $x_k = y_k = 4$. For $k = 2$, $x_k = y_k$. But for $k = 3$, $x_k \neq y_k$.

We now introduce the following distance:

$$d_B(x_K, y_K) = \begin{cases} 1 & \text{if } x_1 \neq y_1 \\ \inf 2^{-n} & x_n = y_n \quad 1 \leq n \leq |K| \end{cases} \quad (1)$$

So for $x_K = 0.478$ and $y_K = 0.472$ we have $d_B(x_K, y_K) = 2^{-2} = 0.25$.

The Baire distance is used in denotational semantics where one considers x_K and y_K as words (of equal length, in the finite case), and then this distance is defined from a common n -length prefix, or left substring, in the two words. For a set of words, a prefix tree can be built to expedite word matching, and the Baire distance derived from this tree.

We have $1 \geq d_B(x_K, y_K) \geq 2^{-|K|}$. Identical x_K and y_K have Baire distance equal to $2^{-|K|}$. The Baire distance is a 1-bounded ultrametric.

The Baire ultrametric defines a hierarchy, which can be expressed as a multi-way tree, on a set of numbers, x_{IK} . So the number x_{iK} , indexed by i , $i \in I$, is of precision $|K|$. It is actually simple to determine this hierarchy. The partition at level $k = 1$ has clusters defined as all those numbers indexed by i that share the same 1st digit. The partition at level $k = 2$ has clusters defined as all those numbers indexed by i that share the same 2nd digit; and so on, until we reach $k = |K|$. A strictly finer, or identical, partition is to be found at each successive level (since once a pair of numbers becomes dissimilar, $d_B > 0$, this non-zero distance cannot be reversed). Identical numbers at level $k = 1$ have distance $\leq 2^{-1} = 0.5$. Identical numbers at level $k = 2$ have distance $\leq 2^{-2} = 0.25$. Identical numbers at level $k = 3$ have distance $\leq 2^{-3} = 0.125$; and so on, to level $k = |K|$, when distance $= 2^{-|K|}$.

3.3.3 Analysis: Baire Ultrametrization from Numerical Precision

In this section we use (i) a random projection of vectors into a 1-dimensional space (so each chemical structure is mapped onto a scalar value, by design ≥ 0 and ≤ 1) followed by (ii) implicit use of a prefix tree constructed on the digits of the set of scalar values. First we will look at this procedure. Then we will return to discuss its properties.

We seek all i, i' such that:

1. for all $j \in J$,
2. $x_{ijK} = x_{i'jK}$
3. to fixed precision K

Recall that K is an ordered set. We impose a user specified upper limit on precision, $|K|$.

Table 3. Results for the three different data sets, each consisting of 7500 chemicals, are shown in immediate succession. The number of significant decimal digits is 4 (more precise, and hence more different clusters found), 3, 2, and 1 (lowest precision in terms of significant digits).

Sig. dig. c	No. clusters
4	6591
4	6507
4	5735
3	6481
3	6402
3	5360
2	2519
2	2576
2	2135
1	138
1	148
1	167

Now rather than $|J|$ separate tests for equality (point 1 above), a *sufficient condition* is that $\sum_j w_j x_{ijK} = \sum_j w_j x_{i'jK}$ for a set of weights w_j . What helps in making this sufficient condition for equality work well in practice is that many of the x_{iJK} values are 0: cf. the approximate 8% matrix occupancy rate that holds here. We experimented with such possibilities as $w_j = j$ (i.e., $\{1, 2, \dots, |J|\}$) and $w_j = |J| + 1 - j$ (i.e., $\{|J|, |J| - 1, \dots, 3, 2, 1\}$). A first principal component would allow for the definition of the least squares optimal linear fit of the projections. The best choice of w_j values we found for uniformly distributed values in $(0, 1)$: for each j , $w_j \sim U(0, 1)$.

Table 3 shows, in immediate succession, results for three data sets. The normalizing column sums were calculated and applied independently to each of the three data sets. Insofar as x_J is directly proportional, whether calculated on 7500 chemical structures or 1.2 million, leads to a constant of proportionality, only, between the two cases. As noted, a random projection was used. Finally, identical projected values were read off, to determine clusters.

3.3.4 Discussion: Random Projection and Hashing

Random projection is the finding of a low dimensional embedding of a point set – dimension equals 1, or a line or axis, in this work – such that the distortion

of any pair of points is bounded by a function of the lower dimensionality [77]. There is a burgeoning literature in this area, e.g. [16]. While random projection *per se* will not guarantee a bijection of best match in original and in lower dimensional spaces, our use of projection here is effectively a hashing method ([47] uses MD5 for nearest neighbor search), in order to deliberately find hash collisions – thereby providing a *sufficient* condition for the mapped vectors to be identical.

Collision of identically valued vectors is guaranteed, but what of collision of non-identically valued vectors, which we want to avoid?

To prove such a result may require an assumption of what distribution our original data follow. A general class is referred to as a stable distribution [29]: this is a distribution such that a limited number of weighted sums of the variables is also itself of the same distribution. Examples include both Gaussian and long-tailed or power law distributions.

Interestingly, however, very high dimensional (or equivalently, very low sample size or low n) data sets, by virtue of high relative dimensionality alone, have points mostly lying at the vertices of a regular simplex or polygon [55, 27]. This intriguing aspect is one reason, perhaps, why we have found random projection to work well. Another reason is the following: if we work on normalized data, then the values on any two attributes j will be small. Hence x_j and x'_j are small. Now if the random weight for this attribute is w_j , then the random projections are, respectively, $\sum_j w_j x_j$ and $\sum_j w_j x'_j$. But these terms are dominated by the random weights. We can expect near equal x_j and x'_j terms, for all j , to be mapped onto fairly close resultant scalar values.

Further work is required to confirm these hypotheses, viz., that high dimensional data may be highly “regular” or “structured” in such a way; and that, as a consequence, hashing is particularly well-behaved in the sense of non-identical vectors being nearly always collision-free. There is further discussion in [8].

We remark that a prefix tree, or trie, is well-known in the searching and sorting literature [26], and is used to expedite the finding of longest common prefixes. At level one, nodes are associated with the first digit. At level two, nodes are associated with the second digit, and so on through deeper levels of the tree.

3.3.5 Simple Clustering Hierarchy from the Baire Space Embedding

The Baire ultrametrization induces a (fairly flat) multiway tree on the given data set.

Consider a partition yielded by identity (over all the attribute set) at a given precision level. Then for precision levels k_1, k_2, k_3, \dots we have, at each, a partition, such that all member clusters are ordered by reverse embedding (or set inclusion): $q_{(1)} \supseteq q_{(2)} \supseteq q_{(3)} \supseteq \dots$. Call each such sequence of embeddings a chain. The entire data set is covered by a set of such chains. This sequence of partitions is ordered by set inclusion.

The computational time complexity is as follows. Let the number of chemicals be denoted $n = |I|$; the number of attributes is $|J|$; and the total number of digits precision is $|K|$. Consider a particular number of digits precision, k_0 , where $1 \leq k_0 \leq |K|$. Then the random projection takes $n \cdot k_0 \cdot |J|$ operations. A sort follows, requiring $O(n \log n)$ operations. Then clusters are read off with $O(n)$ operations. Overall, the computational effort is bounded by $c_1 \cdot |I| \cdot |J| \cdot |K| + c_2 \cdot |I| \cdot \log |I| + c_3 \cdot |I|$ (where c_1, c_2, c_3 are constants), which is equal to $O(|I| \log |I|)$ or $O(n \log n)$.

Further evaluation and a number of further case studies are covered in [8].

4 Hierarchy in a p-Adic Number System

A dendrogram is widely used in hierarchical, agglomerative clustering, and is induced from observed data. In this article, one of our important goals is to show how it lays bare many diverse symmetries in the observed phenomenon represented by the data. By expressing a dendrogram in p-adic terms, we open up a wide range of possibilities for seeing symmetries and attendant invariants.

4.1 p-Adic Encoding of a Dendrogram

We will introduce now the one-to-one mapping of clusters (including singletons) in a dendrogram H into a set of p-adically expressed integers (a fortiori, rationals, or \mathbb{Q}_p). The field of p-adic numbers is the most important example of ultrametric spaces. Addition and multiplication of p-adic integers, \mathbb{Z}_p (cf. expression in subsection 1.4), are well-defined. Inverses exist and no zero-divisors exist.

A terminal-to-root traversal in a dendrogram or binary rooted tree is defined as follows. We use the path $x \subset q \subset q' \subset q'' \subset \dots q_{n-1}$, where x is a given object specifying a given terminal, and q, q', q'', \dots are the embedded classes along this path, specifying nodes in the dendrogram. The root node is specified by the class q_{n-1} comprising all objects.

A terminal-to-root traversal is the shortest path between the given terminal node and the root node, assuming we preclude repeated traversal (backtrack) of the same path between any two nodes.

By means of terminal-to-root traversals, we define the following p-adic encoding of terminal nodes, and hence objects, in Figure 6.

$$\begin{aligned}
x_1 &: +1 \cdot p^1 + 1 \cdot p^2 + 1 \cdot p^5 + 1 \cdot p^7 \\
x_2 &: -1 \cdot p^1 + 1 \cdot p^2 + 1 \cdot p^5 + 1 \cdot p^7 \\
x_3 &: -1 \cdot p^2 + 1 \cdot p^5 + 1 \cdot p^7 \\
x_4 &: +1 \cdot p^3 + 1 \cdot p^4 - 1 \cdot p^5 + 1 \cdot p^7 \\
x_5 &: -1 \cdot p^3 + 1 \cdot p^4 - 1 \cdot p^5 + 1 \cdot p^7 \\
x_6 &: -1 \cdot p^4 - 1 \cdot p^5 + 1 \cdot p^7 \\
x_7 &: +1 \cdot p^6 - 1 \cdot p^7 \\
x_8 &: -1 \cdot p^6 - 1 \cdot p^7
\end{aligned} \tag{2}$$

If we choose $p = 2$ the resulting decimal equivalents could be the same: cf. contributions based on $+1 \cdot p^1$ and $-1 \cdot p^1 + 1 \cdot p^2$. Given that the coefficients of the p^j terms ($1 \leq j \leq 7$) are in the set $\{-1, 0, +1\}$ (implying for x_1 the additional terms: $+0 \cdot p^3 + 0 \cdot p^4 + 0 \cdot p^6$), the coding based on $p = 3$ is required to avoid ambiguity among decimal equivalents.

A few general remarks on this encoding follow. For the labeled ranked binary trees that we are considering (for discussion of combinatorial properties based on labeled, ranked and binary trees, see [52]), we require the labels $+1$ and -1 for the two branches at any node. Of course we could interchange these labels, and have these $+1$ and -1 labels reversed at any node. By doing so we will have different p-adic codes for the objects, x_i .

The following properties hold: (i) *Unique encoding*: the decimal codes for each x_i (lexicographically ordered) are unique for $p \geq 3$; and (ii) *Reversibility*: the dendrogram can be uniquely reconstructed from any such set of unique codes.

The p-adic encoding defined for any object set can be expressed as follows for any object x associated with a terminal node:

$$x = \sum_{j=1}^{n-1} c_j p^j \text{ where } c_j \in \{-1, 0, +1\} \quad (3)$$

In greater detail we have:

$$x_i = \sum_{j=1}^{n-1} c_{ij} p^j \text{ where } c_{ij} \in \{-1, 0, +1\} \quad (4)$$

Here j is the level or rank (root: $n - 1$; terminal: 1), and i is an object index.

In our example we have used: $c_j = +1$ for a left branch (in the sense of Figure 6), $= -1$ for a right branch, and $= 0$ when the node is not on the path from that particular terminal to the root.

A matrix form of this encoding is as follows, where $\{\cdot\}^t$ denotes the transpose of the vector.

Let \mathbf{x} be the column vector $\{x_1 \ x_2 \ \dots \ x_n\}^t$.

Let \mathbf{p} be the column vector $\{p^1 \ p^2 \ \dots \ p^{n-1}\}^t$.

Define a characteristic matrix C of the branching codes, $+1$ and -1 , and an absent or non-existent branching given by 0, as a set of values c_{ij} where $i \in I$, the indices of the object set; and $j \in \{1, 2, \dots, n-1\}$, the indices of the dendrogram levels or nodes ordered increasingly. For Figure 6 we therefore have:

$$C = \{c_{ij}\} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & -1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{pmatrix} \quad (5)$$

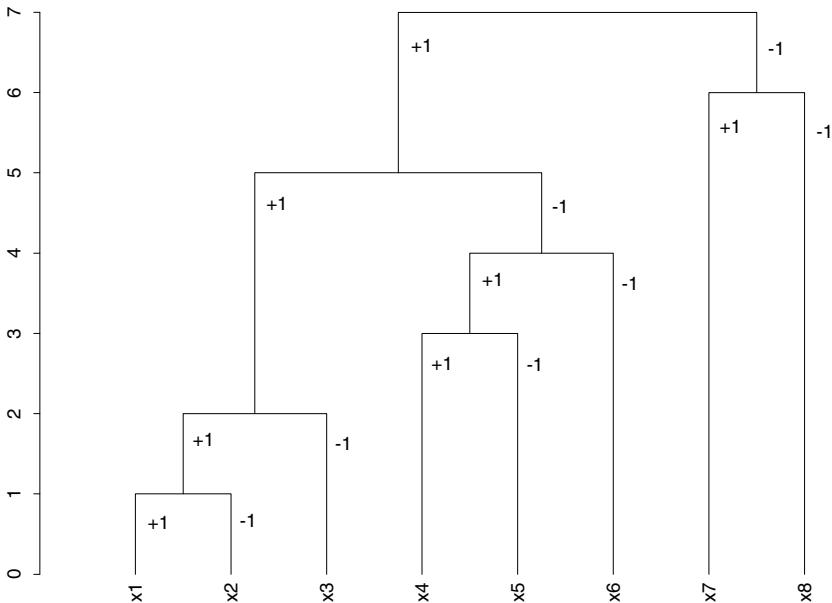


Fig. 6. Labeled, ranked dendrogram on 8 terminal nodes, x_1, x_2, \dots, x_8 . Branches are labeled +1 and -1. Clusters are: $q_1 = \{x_1, x_2\}$, $q_2 = \{x_1, x_2, x_3\}$, $q_3 = \{x_4, x_5\}$, $q_4 = \{x_4, x_5, x_6\}$, $q_5 = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $q_6 = \{x_7, x_8\}$, $q_7 = \{x_1, x_2, \dots, x_7, x_8\}$.

For given level j , $\forall i$, the absolute values $|c_{ij}|$ give the membership function either by node, j , which is therefore read off columnwise; or by object index, i , which is therefore read off rowwise.

The matrix form of the p-adic encoding used in equations (3) or (4) is:

$$\mathbf{x} = C\mathbf{p} \quad (6)$$

Here, \mathbf{x} is the decimal encoding, C is the matrix with dendrogram branching codes (cf. example shown in expression (5)), and \mathbf{p} is the vector of powers of a fixed integer (usually, more restrictively, fixed prime) p .

The tree encoding exemplified in Figure 6, and defined with coefficients in equations (3) or (4), (5) or (6), with labels +1 and -1 was required (as opposed to the choice of 0 and 1, which might have been our first thought) to fully cater for the ranked nodes (i.e. the total order, as opposed to a partial order, on the nodes).

We can consider the objects that we are dealing with to have equivalent integer values. To show that, all we must do is work out decimal equivalents of the p-adic expressions used above for x_1, x_2, \dots . As noted in [25], we have equivalence between: a p-adic number; a p-adic expansion; and an element of \mathbb{Z}_p (the p-adic integers). The coefficients used to specify a p-adic number, [25] notes (p. 69), “must be taken in a set of representatives of the class modulo p . The numbers

between 0 and $p - 1$ are only the most obvious choice for these representatives. There are situations, however, where other choices are expedient.”

We note that the matrix C is used in [9]. A somewhat trivial view of how “hierarchical trees can be perfectly scaled in one dimension” (the title and theme of [9]) is that p-adic numbering is feasible, and hence a one dimensional representation of terminal nodes is easily arranged through expressing each p-adic number with a real number equivalent.

4.2 p-Adic Distance on a Dendrogram

We will now induce a metric topology on the p-adically encoded dendrogram, H . It leads to various symmetries relative to identical norms, for instance, or identical tree distances.

We use the following longest common subsequence, starting at the root: we look for the term p^r in the p-adic codes of the two objects, where r is the lowest level such that the values of the coefficients of p^r are equal.

Let us look at the set of p-adic codes for x_1, x_2, \dots above (Figure 6 and relations 3), to give some examples of this.

For x_1 and x_2 , we find the term we are looking for to be p^1 , and so $r = 1$.

For x_1 and x_5 , we find the term we are looking for to be p^5 , and so $r = 5$.

For x_5 and x_8 , we find the term we are looking for to be p^7 , and so $r = 7$.

Having found the value r , the distance is defined as p^{-r} [3, 25].

This longest common prefix metric is also known as the Baire distance, and has been discussed in section 3.3. In topology the Baire metric is defined on infinite strings [42]. It is more than just a distance: it is an ultrametric bounded from above by 1, and its *infimum* is 0 which is relevant for very long sequences, or in the limit for infinite-length sequences. The use of this Baire metric is pursued in [62] based on random projections [77], and providing computational benefits over the classical $O(n^2)$ hierarchical clustering based on all pairwise distances.

The longest common prefix metric leads directly to a *p-adic hierarchical classification* (cf. [5]). This is a special case of the “fast” hierarchical clustering discussed in section 3.2.

Compared to the longest common prefix metric, there are other related forms of metric, and simultaneously ultrametric. In [23], the metric is defined via the integer part of a real number. In [3], for integers x, y we have: $d(x, y) = 2^{-\text{order}_p(x-y)}$ where p is prime, and $\text{order}_p(i)$ is the exponent (non-negative integer) of p in the prime decomposition of an integer. Furthermore let $S(x)$ be a series: $S(x) = \sum_{i \in \mathbb{N}} a_i x^i$. (\mathbb{N} are the natural numbers.) The order of $S(i)$ is the rank of its first non-zero term: $\text{order}(S) = \inf\{i : i \in \mathbb{N}; a_i \neq 0\}$. (The series that is all zero is of order infinity.) Then the ultrametric similarity between series is: $d(S, S') = 2^{-\text{order}(S-S')}$.

4.3 Scale-Related Symmetry

Scale-related symmetry is very important in practice. In this subsection we introduce an operator that provides this symmetry. We also term it a dilation operator, because of its role in the wavelet transform on trees (see section 5.3 below, and [58] for discussion and examples). This operator is p-adic multiplication by $1/p$.

Consider the set of objects $\{x_i | i \in I\}$ with its p-adic coding considered above. Take $p = 2$. (Non-uniqueness of corresponding decimal codes is not of concern to us now, and taking this value for p is without any loss of generality.) Multiplication of $x_1 = +1 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^5 + 1 \cdot 2^7$ by $1/p = 1/2$ gives: $+1 \cdot 2^1 + 1 \cdot 2^4 + 1 \cdot 2^6$. Each level has decreased by one, and the lowest level has been lost. Subject to the lowest level of the tree being lost, the form of the tree remains the same. By carrying out the multiplication-by- $1/p$ operation on all objects, it is seen that the effect is to rise in the hierarchy by one level.

Let us call product with $1/p$ the operator A . The effect of losing the bottom level of the dendrogram means that either (i) each cluster (possibly singleton) remains the same; or (ii) two clusters are merged. Therefore the application of A to all q implies a subset relationship between the set of clusters $\{q\}$ and the result of applying A , $\{Aq\}$.

Repeated application of the operator A gives Aq, A^2q, A^3q, \dots . Starting with any singleton, $i \in I$, this gives a path from the terminal to the root node in the tree. Each such path ends with the null element, which we define to be the p-adic encoding corresponding to the root node of the tree. Therefore the intersection of the paths equals the null element.

Benedetto and Benedetto [1, 2] discuss A as an expansive automorphism of I , i.e. form-preserving, and locally expansive. Some implications [1] of the expansive automorphism follow. For any q , let us take q, Aq, A^2q, \dots as a sequence of open subgroups of I , with $q \subset Aq \subset A^2q \subset \dots$, and $I = \bigcup\{q, Aq, A^2q, \dots\}$. This is termed an inductive sequence of I , and I itself is the inductive limit ([68], p. 131).

Each path defined by application of the expansive automorphism defines a spherically complete system [70, 23, 69], which is a formalization of well-defined subset embeddedness. Such a methodological framework finds application in multi-valued and non-monotonic reasoning, as noted in section 3.2.

5 Tree Symmetries through the Wreath Product Group

In this section the wreath product group, used up to now in the literature as a framework for tree structuring of image or other signal data, is here used on a 2-way tree or dendrogram data structure. An example of wreath product invariance is provided by the wavelet transform of such a tree.

5.1 Wreath Product Group Corresponding to a Hierarchical Clustering

A dendrogram like that shown in Figure 6 is invariant as a representation or structuring of a data set relative to rotation (alternatively, here: permutation) of left and right child nodes. These rotation (or permutation) symmetries are defined by the wreath product group (see [20, 21, 18] for an introduction and applications in signal and image processing), and can be used with any m-ary tree, although we will treat the binary or 2-way case here.

For the group actions, with respect to which we will seek invariance, we consider independent cyclic shifts of the subnodes of a given node (hence, at each level). Equivalently these actions are adjacency preserving permutations of subnodes of a given node (i.e., for given q , with $q = q' \cup q''$, the permutations of $\{q', q''\}$). We have therefore cyclic group actions at each node, where the cyclic group is of order 2.

The symmetries of H are given by structured permutations of the terminals. The terminals will be denoted here by Term H . The full group of symmetries is summarized by the following generative algorithm:

1. For level $l = n - 1$ down to 1 do:
2. Selected node, $\nu \leftarrow$ node at level l .
3. And permute subnodes of ν .

Subnode ν is the root of subtree H_ν . We denote H_{n-1} simply by H . For a subnode ν' undergoing a relocation action in step 3, the internal structure of subtree $H_{\nu'}$ is not altered.

The algorithm described defines the automorphism group which is a wreath product of the symmetric group. Denote the permutation at level ν by P_ν . Then the automorphism group is given by:

$$G = P_{n-1} \text{ wr } P_{n-2} \text{ wr } \dots \text{ wr } P_2 \text{ wr } P_1$$

where wr denotes the wreath product.

5.2 Wreath Product Invariance

Call Term H_ν the terminals that descend from the node at level ν . So these are the terminals of the subtree H_ν with its root node at level ν . We can alternatively call Term H_ν the cluster associated with level ν .

We will now look at shift invariance under the group action. This amounts to the requirement for a constant function defined on Term $H_\nu, \forall \nu$. A convenient way to do this is to define such a function on the set Term H_ν via the root node alone, ν . By definition then we have a constant function on the set Term H_ν .

Let us call V_ν a space of functions that are constant on Term H_ν . That is to say, the functions are constant in clusters that are defined by the subset of n objects. Possibilities for V_ν that were considered in [58] are:

1. Basis vector with $|\text{Term}H_{n-1}|$ components, with 0 values except for value 1 for component i .
2. Set (of cardinality $n = |\text{Term}H_{n-1}|$) of m -dimensional observation vectors.

Consider the resolution scheme arising from moving from Term $H_{\nu'}$, Term $H_{\nu''}\}$ to Term H_ν . From the hierarchical clustering point of view it is clear what this represents, simply, an agglomeration of two clusters called Term $H_{\nu'}$ and Term $H_{\nu''}$, replacing them with a new cluster, Term H_ν .

Let the spaces of functions that are constant on subsets corresponding to the two cluster agglomerands be denoted $V_{\nu'}$ and $V_{\nu''}$. These two clusters are disjoint initially, which motivates us taking the two spaces as a couple: $(V_{\nu'}, V_{\nu''})$.

5.3 Example of Wreath Product Invariance: Haar Wavelet Transform of a Dendrogram

Let us exemplify a case that satisfies all that has been defined in the context of the wreath product invariance that we are targeting. It is the algorithm discussed in depth in [58]. Take the constant function from $V_{\nu'}$ to be $f_{\nu'}$. Take the constant function from $V_{\nu''}$ to be $f_{\nu''}$. Then define the constant function, the *scaling function*, in V_ν to be $(f_{\nu'} + f_{\nu''})/2$. Next define the zero mean function, $(w_{\nu'} + w_{\nu''})/2 = 0$, the *wavelet function*, as follows:

$$w_{\nu'} = (f_{\nu'} + f_{\nu''})/2 - f_{\nu'}$$

in the support interval of $V_{\nu'}$, i.e. Term $H_{\nu'}$, and

$$w_{\nu''} = (f_{\nu'} + f_{\nu''})/2 - f_{\nu''}$$

in the support interval of $V_{\nu''}$, i.e. Term $H_{\nu''}$.

Since $w_{\nu'} = -w_{\nu''}$ we have the zero mean requirement.

We now illustrate the Haar wavelet transform of a dendrogram with a case study.

The discrete wavelet transform is a decomposition of data into spatial and frequency components. In terms of a dendrogram these components are with respect to, respectively, within and between clusters of successive partitions. We show how this works taking the data of Table 4.

Table 4. First 8 observations of Fisher's iris data. L and W refer to length and width.

	Sepal.L	Sepal.W	Petal.L	Petal.W
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2
6	5.4	3.9	1.7	0.4
7	4.6	3.4	1.4	0.3
8	5.0	3.4	1.5	0.2

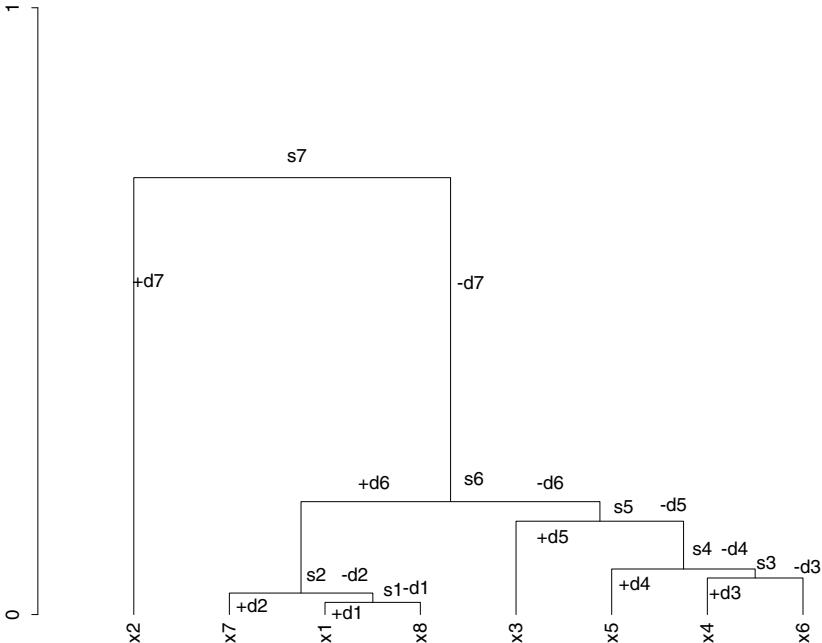


Fig. 7. Dendrogram on 8 terminal nodes constructed from first 8 values of Fisher iris data. (Median agglomerative method used in this case.) Detail or wavelet coefficients are denoted by d , and data smooths are denoted by s . The observation vectors are denoted by x and are associated with the terminal nodes. Each *signal smooth*, s , is a vector. The (positive or negative) *detail signals*, d , are also vectors. All these vectors are of the same dimensionality.

Table 5. The hierarchical Haar wavelet transform resulting from use of the first 8 observations of Fisher's iris data shown in Table 4. Wavelet coefficient levels are denoted d1 through d7, and the continuum or smooth component is denoted s7.

	s7	d7	d6	d5	d4	d3	d2	d1
Sepal.L	5.146875	0.253125	0.13125	0.1375	-0.025	0.05	-0.025	0.05
Sepal.W	3.603125	0.296875	0.16875	-0.1375	0.125	0.05	-0.075	-0.05
Petal.L	1.562500	0.137500	0.02500	0.0000	0.000	-0.10	0.050	0.00
Petal.W	0.306250	0.093750	-0.01250	-0.0250	0.050	0.00	0.000	0.00

The hierarchy built on the 8 observations of Table 4 is shown in Figure 7. Here we denote irises 1 through 8 as, respectively: $x_1, x_3, x_4, x_6, x_8, x_2, x_5, x_7$.

Something more is shown in Figure 7, namely the detail signals (denoted $\pm d$) and overall smooth (denoted s), which are determined in carrying out the wavelet transform, the so-called forward transform.

The inverse transform is then determined from Figure 7 in the following way. Consider the observation vector x_2 . Then this vector is reconstructed exactly by reading the tree from the root: $s_7 + d_7 = x_2$. Similarly a path from root to terminal is used to reconstruct any other observation. If x_2 is a vector of dimensionality m , then so also are s_7 and d_7 , as well as all other detail signals.

This procedure is the same as the Haar wavelet transform, only applied to the dendrogram and using the input data.

This wavelet transform for the data in Table 4, based on the “key” or intermediary hierarchy of Figure 7, is shown in Table 5.

Wavelet regression entails setting small and hence unimportant detail coefficients to 0 before applying the inverse wavelet transform. More discussion can be found in [58].

Early work on p-adic and ultrametric wavelets can be found in Kozyrev [38, 39]. While we have treated the case of the wavelet transform on a particular graph, a tree, recent applications of wavelets to general graphs are in [34] and, by representing the graph as a matrix, in [63].

6 Remarkable Symmetries in Very High Dimensional Spaces

In the work of [66, 67] it was shown how as ambient dimensionality increased distances became more and more ultrametric. That is to say, a hierarchical embedding becomes more and more immediate and direct as dimensionality increases. A better way of quantifying this phenomenon was developed in [55]. What this means is that there is inherent hierarchical structure in high dimensional data spaces.

It was shown experimentally in [66, 67, 55] how points in high dimensional spaces become increasingly equidistant with increase in dimensionality. Both [27] and [13] study Gaussian clouds in very high dimensions. The latter finds that “not only are the points [of a Gaussian cloud in very high dimensional space] on the convex hull, but all reasonable-sized subsets span faces of the convex hull. This is wildly different than the behavior that would be expected by traditional low-dimensional thinking”.

That very simple structures come about in very high dimensions is not as trivial as it might appear at first sight. Firstly, even very simple structures (hence with many symmetries) can be used to support fast and perhaps even constant time worst case proximity search [55]. Secondly, as shown in the machine learning framework by [27], there are important implications ensuing from the simple high dimensional structures. Thirdly, [59] shows that very high dimensional clustered data contain symmetries that in fact can be exploited to “read off” the clusters in a computationally efficient way. Fourthly, following [11], what we might want to look for in contexts of considerable symmetry are the “impurities” or small irregularities that detract from the overall dominant picture.

See Table 6 exemplifying the change of topological properties as ambient dimensionality increases. It behoves us to exploit the symmetries that arise when we have to process very high dimensional data.

Table 6. Typical results, based on 300 sampled triangles from triplets of points. For uniform, the data are generated on $[0, 1]^m$; hypercube vertices are in $\{0, 1\}^m$, and for Gaussian on each dimension, the data are of mean 0, and variance 1. Dimen. is the ambient dimensionality. Isosc. is the number of isosceles triangles with small base, as a proportion of all triangles sampled. Equil. is the number of equilateral triangles as a proportion of triangles sampled. UM is the proportion of ultrametricity-respecting triangles (= 1 for all ultrametric).

No. points	Dimen.	Isosc.	Equil.	UM
Uniform				
100	20	0.10	0.03	0.13
100	200	0.16	0.20	0.36
100	2000	0.01	0.83	0.84
100	20000	0	0.94	0.94
Hypercube				
100	20	0.14	0.02	0.16
100	200	0.16	0.21	0.36
100	2000	0.01	0.86	0.87
100	20000	0	0.96	0.96
Gaussian				
100	20	0.12	0.01	0.13
100	200	0.23	0.14	0.36
100	2000	0.04	0.77	0.80
100	20000	0	0.98	0.98

6.1 Application to Very High Frequency Data Analysis: Segmenting a Financial Signal

We use financial futures, circa March 2007, denominated in euros from the DAX exchange. Our data stream is at the millisecond rate, and comprises about 382,860 records. Each record includes: 5 bid and 5 asking prices, together with bid and asking sizes in all cases, and action. We extracted one symbol (commodity) with 95,011 single bid values, on which we now report results. See Figure 8.

Embeddings were defined as follows.

- Windows of 100 successive values, starting at time steps: 1, 1000, 2000, 3000, 4000, ..., 94000.
- Windows of 1000 successive values, starting at time steps: 1, 1000, 2000, 3000, 4000, ..., 94000.
- Windows of 10000 successive values, starting at time steps: 1, 1000, 2000, 3000, 4000, ..., 85000.

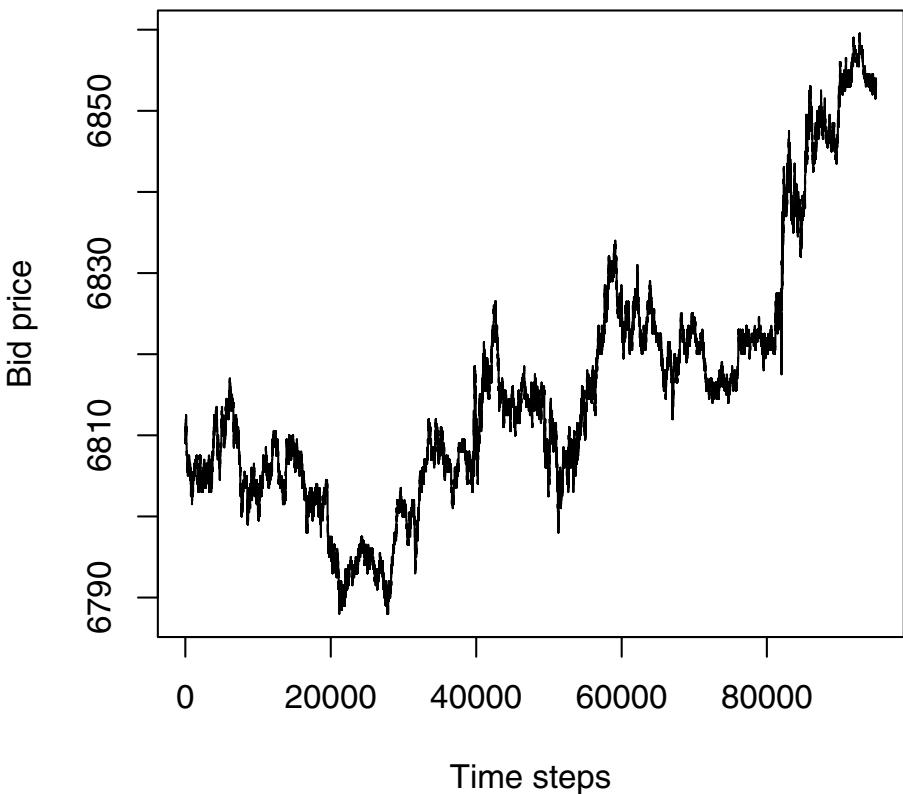


Fig. 8. The signal used: a commodity future, with millisecond time sampling

The histograms of distances between these windows, or embeddings, in respectively spaces of dimension 100, 1000 and 10000, are shown in Figure 9.

Note how the 10000-length window case results in points that are strongly overlapping. In fact, we can say that 90% of the values in each window are overlapping with the next window. Notwithstanding this major overlapping in regard to clusters involved in the pairwise distances, if we can still find clusters in the data then we have a very versatile way of tackling the clustering objective. Because of the greater cluster concentration that we expect (cf. Table 6) from a greater embedding dimension, we use the 86 points in 10000-dimensional space, notwithstanding the fact that these points are from overlapping clusters.

We make the following supposition based on Figure 8: the clusters will consist of successive values, and hence will be justifiably termed segments.

From the distances histogram in Figure 9, bottom, we will carry out Gaussian mixture modeling followed by use of the Bayesian information criterion (BIC, [71]) as an approximate Bayes factor, to determine the best number of clusters (effectively, histogram peaks).

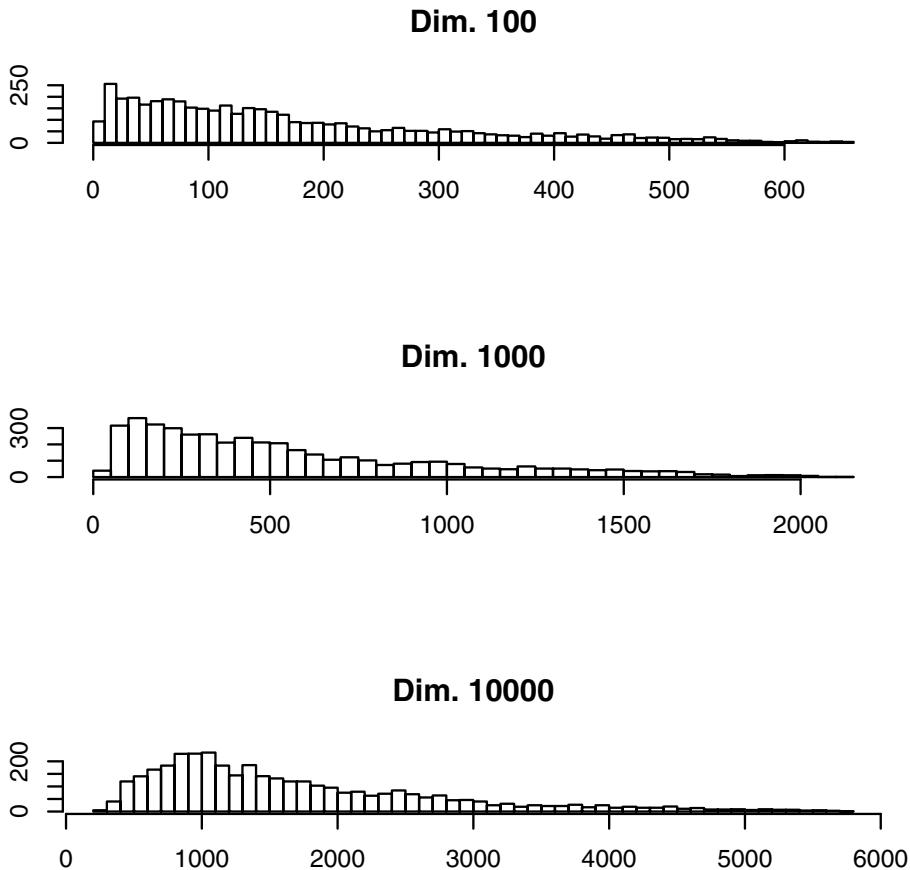


Fig. 9. Histograms of pairwise distances between embeddings in dimensionalities 100, 1000, 10000. Respectively the numbers of embeddings are: 95, 95 and 86.

We fit a Gaussian mixture model to the data shown in the bottom histogram of Figure 9. To derive the appropriate number of histogram peaks we fit Gaussians and use the Bayesian information criterion (BIC) as an approximate Bayes factor for model selection [36, 64]. Figure 10 shows the succession of outcomes, and indicates as best a 5-Gaussian fit. For this result, we find the means of the Gaussians to be as follows: 517, 885, 1374, 2273 and 3908. The corresponding standard deviations are: 84, 133, 212, 410 and 663. The respective cardinalities of the 5 histogram peaks are: 358, 1010, 1026, 911 and 350. Note that this relates so far only to the histogram of pairwise distances. We now want to determine the corresponding clusters in the input data.

While we have the segmentation of the distance histogram, we need the segmentation of the original financial signal. If we had 2 clusters in the original financial signal, then we could expect up to 3 peaks in the distances histogram

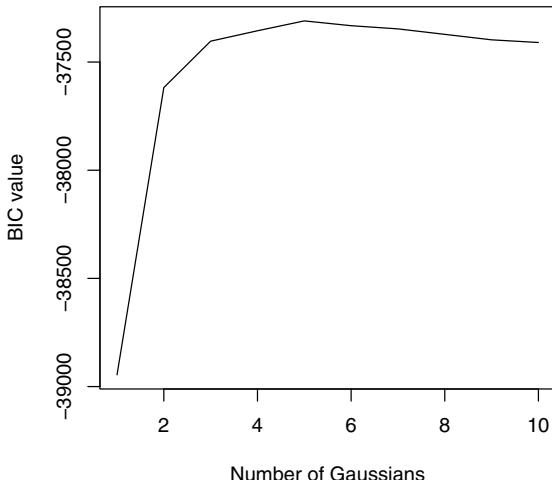


Fig. 10. BIC (Bayesian information criterion) values for the succession of results. The 5-cluster solution has the highest value for BIC and is therefore the best Gaussian mixture fit.

(viz., 2 intra-cluster peaks, and 1 inter-cluster peak). If we had 3 clusters in the original financial signal, then we could expect up to 6 peaks in the distances histogram (viz., 3 intra-cluster peaks, and 3 inter-cluster peaks). This information is consistent with asserting that the evidence from Figure 10 points to two of these histogram peaks being approximately co-located (alternatively: the distances are approximately the same). We conclude that 3 clusters in the original financial signal is the most consistent number of clusters. We will now determine these.

One possibility is to use principal coordinates analysis (Torgerson's, Gower's metric multidimensional scaling) of the pairwise distances. In fact, a 2-dimensional mapping furnishes a very similar pairwise distance histogram to that seen using the full, 10000, dimensionality. The first axis in Figure 11 accounts for 88.4% of the variance, and the second for 5.8%. Note therefore how the scales of the planar representation in Figure 11 point to it being very linear.

Benzécri ([4], chapter 7, section 3.1) discusses the Guttman effect, or Guttman scale, where factors that are not mutually correlated, are nonetheless functionally related. When there is a “fundamentally unidimensional underlying phenomenon” (there are multiple such cases here) factors are functions of Legendre polynomials. We can view Figure 11 as consisting of multiple horseshoe shapes. A simple explanation for such shapes is in terms of the constraints imposed by lots of equal distances when the data vectors are ordered linearly (see [56], pp. 46-47).

Another view of how embedded (hence clustered) data are capable of being well mapped into a unidimensional curve is Critchley and Heiser [9]. Critchley and Heiser show one approach to mapping an ultrametric into a linearly or

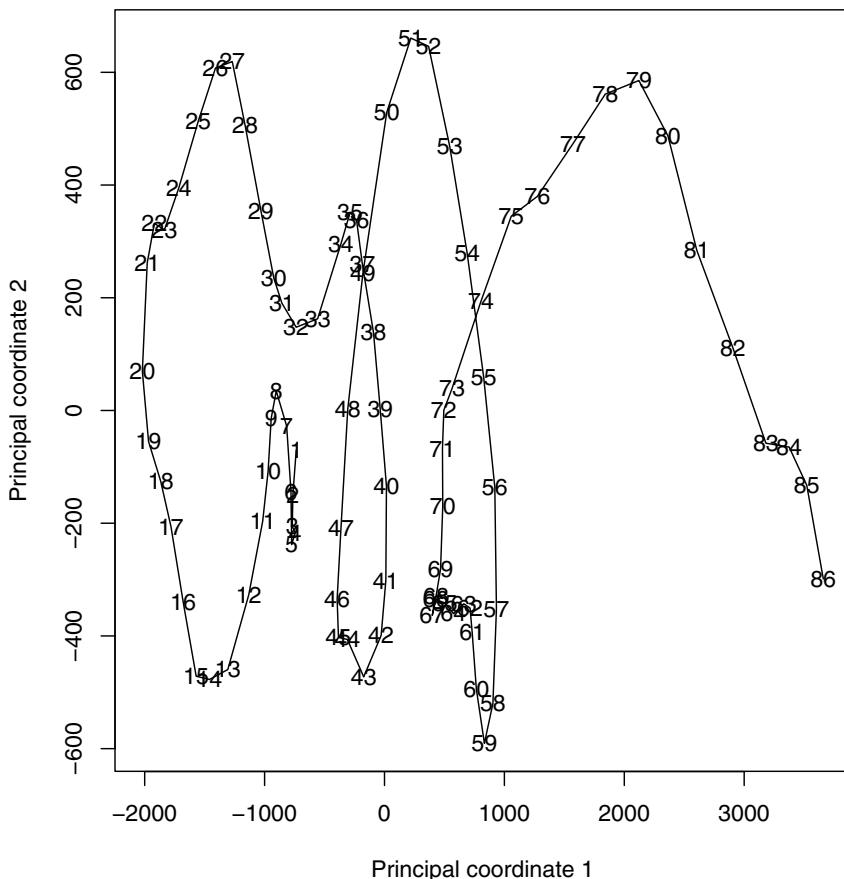


Fig. 11. An interesting representation – a type of “return map” – found using a principal coordinates analysis of the 86 successive 10000-dimensional points. Again a demonstration that very high dimensional structures can be of very simple structure. The planar projection seen here represents most of the information content of the data: the first axis accounts for 88.4% of the variance, while the second accounts for 5.8%.

totally ordered metric. We have asserted and then established how hierarchy in some form is relevant for high dimensional data spaces; and then we find a very linear projection in Figure 11. As a consequence we note that the Critchley and Heiser result is especially relevant for high dimensional data analysis.

Knowing that 3 clusters in the original signal are wanted, we could use Figure 11. There are various ways to do so.

We will use an adjacency-constrained agglomerative hierarchical clustering algorithm to find the clusters: see Figure 12. The contiguity-constrained complete link criterion is our only choice here if we are to be sure that no inversions can come about in the hierarchy, as explained in [53]. As input, we use the coordinates

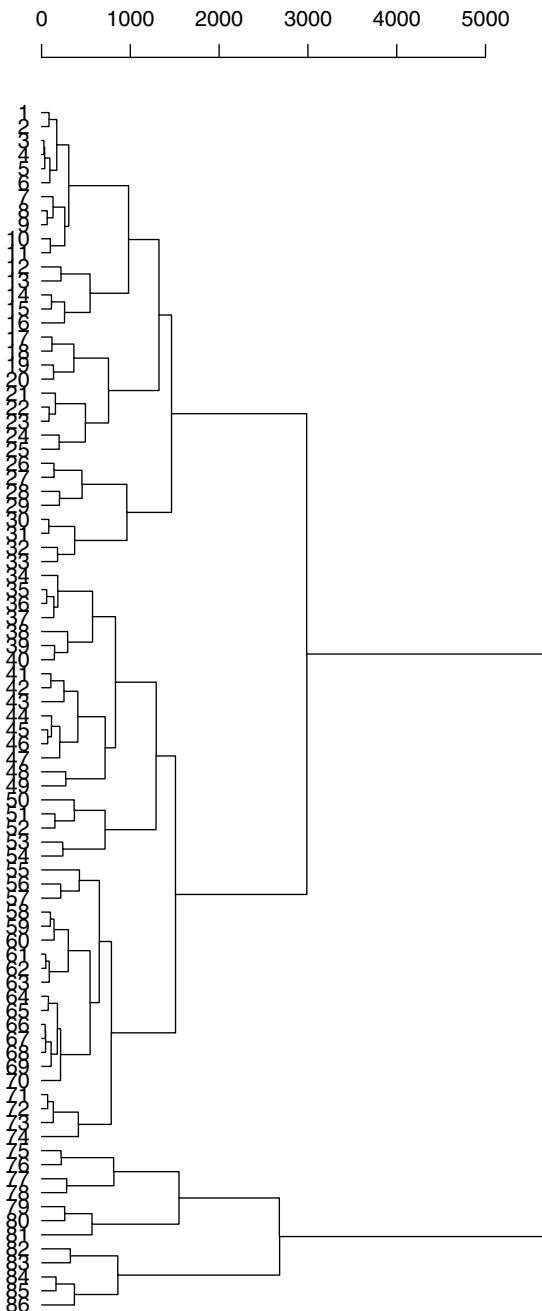


Fig. 12. Hierarchical clustering of the 86 points. Sequence is respected. The agglomerative criterion is the contiguity-constrained complete link method. See [53] for details including proof that there can be no inversion in this dendrogram.

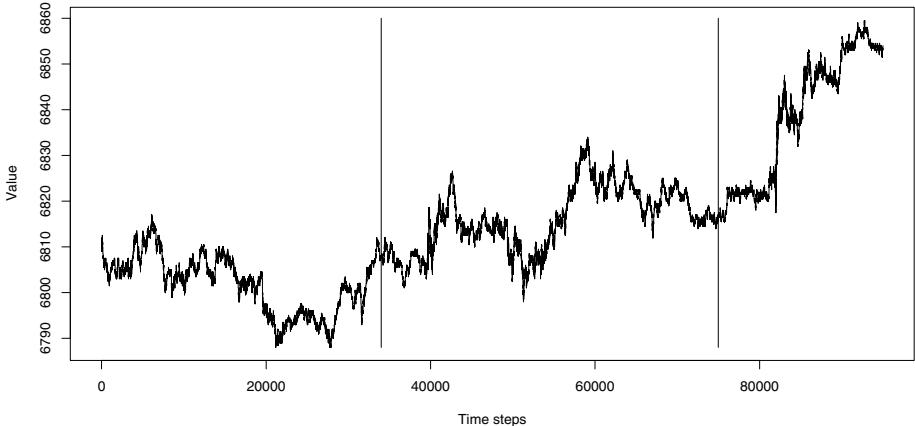


Fig. 13. Boundaries found for 3 segments

in Figure 11. The 2-dimensional Figure 11 representation relates to over 94% of the variance. The most complete basis was of dimensionality 85. We checked the results of the 85-dimensionality embedding which, as noted below, gave very similar results.

Reading off the 3-cluster memberships from Figure 12 gives for the signal actually used (with a very initial segment and a very final segment deleted): cluster 1 corresponds to signal values 1000 to 33999 (points 1 to 33 in Figure 12); cluster 2 corresponds to signal values 34000 to 74999 (points 34 to 74 in Figure 12); and cluster 3 corresponds to signal values 75000 to 86999 (points 75 to 86 in Figure 12). This allows us to segment the original time series: see Figure 13. (The clustering of the 85-dimensionality embedding differs minimally. Segments are: points 1 to 32; 33 to 73; and 74 to 86.)

To summarize what has been done:

1. the segmentation is initially guided by the peak-finding in the histogram of distances
2. with high dimensionality we expect simple structure in a low dimensional mapping provided by principal coordinates analysis
3. either the original high dimensional data or the principal coordinates analysis embedding are used as input to a sequence-constrained clustering method in order to determine the clusters
4. which can then be displayed on the original data.

In this case, the clusters are defined using a complete link criterion, implying that these three clusters are determined by minimizing their maximum internal pairwise distance. This provides a strong measure of signal volatility as an explanation for the clusters, in addition to their average value.

7 Conclusions

Among themes not covered in this article are data stream clustering. To provide background and motivation, in [60], we discuss permutation representations of a data stream. Since hierarchies can also be represented as permutations, there is a ready way to associate data streams with hierarchies. In fact, early computational work on hierarchical clustering used permutation representation to great effect (cf. [73]). To analyze data streams in this way, in [57] we develop an approach to ultrametric embedding of time-varying signals, including biomedical, meteorological, financial and other. This work has been pursued in physics by Khrennikov.

Let us now wrap up on the exciting perspectives opened up by our work on the theme of symmetry-finding through hierarchy in very large data collections.

“My thesis has been that one path to the construction of a nontrivial theory of complex systems is by way of a theory of hierarchy.” Thus Simon ([74], p. 216). We have noted symmetry in many guises in the representations used, in the transformations applied, and in the transformed outputs. These symmetries are non-trivial too, in a way that would not be the case were we simply to look at classes of a partition and claim that cluster members were mutually similar in some way. We have seen how the p-adic or ultrametric framework provides significant focus and commonality of viewpoint.

Furthermore we have highlighted the computational scaling properties of our algorithms. They are fully capable of addressing the data and information deluge that we face, and providing us with the best interpretative and decision-making tools. The full elaboration of this last point is to be sought in each and every application domain, and face to face with old and new problems.

In seeking (in a general way) and in determining (in a focused way) structure and regularity in massive data stores, we see that, in line with the insights and achievements of Klein, Weyl and Wigner, in data mining and data analysis we seek and determine symmetries in the data that express observed and measured reality.

References

- [1] Benedetto, J.J., Benedetto, R.L.: A wavelet theory for local fields and related groups. *The Journal of Geometric Analysis* 14, 423–456 (2004)
- [2] Benedetto, R.L.: Examples of wavelets for local fields. In: Larson, D., Heil, C., Jorgensen, P. (eds.) *Wavelets, Frames, and Operator Theory, Contemporary Mathematics*, vol. 345, pp. 27–47 (2004)
- [3] Benzécri, J.-P.: *L’Analyse des Données. Tome I. Taxinomie*, 2nd edn., Dunod, Paris (1979)
- [4] Benzécri, J.-P.: *L’Analyse des Données. Tome II, Correspondances*, 2nd edn., Dunod, Paris (1979)
- [5] Bradley, P.E.: Mumford dendrograms. *Computer Journal* 53, 393–404 (2010)
- [6] Brekke, L., Freund, P.G.O.: p-Adic numbers in physics. *Physics Reports* 233, 1–66 (1993)

- [7] Chakraborty, P.: Looking through newly to the amazing irrationals. Technical report, arXiv: math.HO/0502049v1 (2005)
- [8] Contreras, P.: Search and Retrieval in Massive Data Collections. PhD thesis, Royal Holloway, University of London (2010)
- [9] Critchley, F., Heiser, W.: Hierarchical trees can be perfectly scaled in one dimension. *Journal of Classification* 5, 5–20 (1988)
- [10] Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order, 2nd edn. Cambridge University Press, Cambridge (2002)
- [11] Delon, F.: Espaces ultramétriques. *Journal of Symbolic Logic* 49, 405–502 (1984)
- [12] Deutsch, S.B., Martin, J.J.: An ordering algorithm for analysis of data arrays. *Operations Research* 19, 1350–1362 (1971)
- [13] Donoho, D.L., Tanner, J.: Neighborliness of randomly-projected simplices in high dimensions. *Proceedings of the National Academy of Sciences* 102, 9452–9457 (2005)
- [14] Dragovich, B., Dragovich, A.: p-Adic modelling of the genome and the genetic code. *Computer Journal* 53, 432–442 (2010)
- [15] Dragovich, B., Khrennikov, A.Y., Kozyrev, S.V., Volovich, I.V.: On p-adic mathematical physics. *P-Adic Numbers, Ultrametric Analysis, and Applications* 1, 1–17 (2009)
- [16] Dutta, D., Guha, R., Jurs, P., Chen, T.: Scalable partitioning and exploration of chemical spaces using geometric hashing. *Journal of Chemical Information and Modeling* 46, 321–333 (2006)
- [17] Fisher, R.A.: The use of multiple measurements in taxonomic problems. *The Annals of Eugenics*, 179–188 (1936)
- [18] Foote, R.: An algebraic approach to multiresolution analysis. *Transactions of the American Mathematical Society* 357, 5031–5050 (2005)
- [19] Foote, R.: Mathematics and complex systems. *Science* 318, 410–412 (2007)
- [20] Foote, R., Mirchandani, G., Rockmore, D., Healy, D., Olson, T.: A wreath product group approach to signal and image processing: Part I — multiresolution analysis. *IEEE Transactions on Signal Processing* 48, 102–132 (2000)
- [21] Foote, R., Mirchandani, G., Rockmore, D., Healy, D., Olson, T.: A wreath product group approach to signal and image processing: Part II — convolution, correlations and applications. *IEEE Transactions on Signal Processing* 48, 749–767 (2000)
- [22] Freund, P.G.O.: p-Adic strings and their applications. In: Rakic, Z., Dragovich, B., Khrennikov, A., Volovich, I. (eds.) *Proc. 2nd International Conference on p-Adic Mathematical Physics*, pp. 65–73. American Institute of Physics (2006)
- [23] Gajić, L.: On ultrametric space. *Novi Sad Journal of Mathematics* 31, 69–71 (2001)
- [24] Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999); Formale Begriffsanalyse. Mathematische Grundlagen. Springer, Heidelberg (1996)
- [25] Gouvea, F.Q.: p-Adic Numbers: An Introduction. Springer, Heidelberg (2003)
- [26] Gusfield, D.: Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, Cambridge (1997)
- [27] Hall, P., Marron, J.S., Neeman, A.: Geometric representation of high dimensional, low sample size data. *Journal of the Royal Statistical Society B* 67, 427–444 (2005)
- [28] Hitzler, P., Seda, A.K.: The fixed-point theorems of Priess-Crampe and Ribenboim in logic programming. *Fields Institute Communications* 32, 219–235 (2002)
- [29] Indyk, P., Andoni, A., Datar, M., Immorlica, N., Mirrokni, V.: Locally-sensitive hashing using stable distributions. In: Darrell, T., Indyk, P., Shakhnarovich, G. (eds.) *Nearest Neighbor Methods in Learning and Vision: Theory and Practice*, pp. 61–72. MIT Press, Cambridge (2006)

- [30] Jain, A.K., Dubes, R.C.: Algorithms For Clustering Data. Prentice-Hall, Englewood Cliffs (1988)
- [31] Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys 31, 264–323 (1999)
- [32] Janowitz, M.F.: An order theoretic model for cluster analysis. SIAM Journal on Applied Mathematics 34, 55–72 (1978)
- [33] Janowitz, M.F.: Cluster analysis based on abstract posets. Technical report (2005–2006), <http://dimax.rutgers.edu/~melj>
- [34] Jansen, M., Nason, G.P., Silverman, B.W.: Multiscale methods for dataon graphs and irregular multidimensional situations. Journal of the Royal Statistical Society B 71, 97–126 (2009)
- [35] Johnson, S.C.: Hierarchical clustering schemes. Psychometrika 32, 241–254 (1967)
- [36] Kass, R.E., Raftery, A.E.: Bayes factors and model uncertainty. Journal of the American Statistical Association 90, 773–795 (1995)
- [37] Khrennikov, A.Y.: Gene expression from polynomial dynamics in the 2-adic information space. Technical report, arXiv:q-bio/06110682v2 (2006)
- [38] Kozyrev, S.V.: Wavelet theory as p-adic spectral analysis. Izvestiya: Mathematics 66, 367–376 (2002)
- [39] Kozyrev, S.V.: Wavelets and spectral analysis of ultrametric pseudodifferential operators. Sbornik: Mathematics 198, 97–116 (2007)
- [40] Krasner, M.: Nombres semi-réels et espaces ultramétriques. Comptes-Rendus de l’Académie des Sciences, Tome II 219, 433 (1944)
- [41] Lerman, I.C.: Classification et Analyse Ordinale des Données, Dunod, Paris (1981)
- [42] Levy, A.: Basic Set Theory. Dover, Mineola (1979); Springer, Heidlberg (1979)
- [43] Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: a survey. IEEE/ACM Transactions on Computational Biology and Bioinformatics 1, 24–45 (2004)
- [44] March, S.T.: Techniques for structuring database records. Computing Surveys 15, 45–79 (1983)
- [45] McCormick, W.T., Schweitzer, P.J., White, T.J.: Problem decomposition and data reorganization by a clustering technique. Operations Research 20, 993–1009 (1982)
- [46] Van Mechelen, I., Bock, H.-H., De Boeck, P.: Two-mode clustering methods: a structured overview. Statistical Methods in Medical Research 13, 363–394 (2004)
- [47] Miller, M.L., Rodriguez, M.A., Cox, I.J.: Audio fingerprinting: nearest neighbor search in high dimensional binary spaces. Journal of VLSI Signal Processing 41, 285–291 (2005)
- [48] Mirkin, B.: Mathematical Classification and Clustering. Kluwer, Dordrecht (1996)
- [49] Mirkin, B.: Clustering for Data Mining. Chapman and Hall/CRC, Boca Raton, FL (2005)
- [50] Murtagh, F.: A survey of recent advances in hierarchical clustering algorithms. Computer Journal 26, 354–359 (1983)
- [51] Murtagh, F.: Complexities of hierachic clustering algorithms: state of the art. Computational Statistics Quarterly 1, 101–113 (1984)
- [52] Murtagh, F.: Counting dendograms: a survey. Discrete Applied Mathematics 7, 191–199 (1984)
- [53] Murtagh, F.: Multidimensional Clustering Algorithms. Physica-Verlag, Heidelberg (1985)
- [54] Murtagh, F.: Comments on: Parallel algorithms for hierarchical clustering and cluster validity. IEEE Transactions on Pattern Analysis and Machine Intelligence 14, 1056–1057 (1992)

- [55] Murtagh, F.: On ultrametricity, data coding, and computation. *Journal of Classification* 21, 167–184 (2004)
- [56] Murtagh, F.: Correspondence Analysis and Data Coding with R and Java. Chapman and Hall/CRC Press (2005)
- [57] Murtagh, F.: Identifying the ultrametricity of time series. *European Physical Journal B* 43, 573–579 (2005)
- [58] Murtagh, F.: The Haar wavelet transform of a dendrogram. *Journal of Classification* 24, 3–32 (2007)
- [59] Murtagh, F.: The remarkable simplicity of very high dimensional data:application to model-based clustering. *Journal of Classification* 26, 249–277 (2009)
- [60] Murtagh, F.: Symmetry in data mining and analysis: a unifying view based on hierarchy. In: *Proceedings of Steklov Institute of Mathematics*, vol. 265, pp. 177–198 (2009)
- [61] Murtagh, F.: The correspondence analysis platform for uncovering deep structure in data and information (sixth Annual Boole Lecture). *Computer Journal* 53, 304–315 (2010)
- [62] Murtagh, F., Downs, G., Contreras, P.: Hierarchical clustering of massive, high dimensional data sets by exploiting ultrametric embedding. *SIAM Journal on Scientific Computing* 30, 707–730 (2008)
- [63] Murtagh, F., Starck, J.-L., Berry, M.: Overcoming the curse of dimensionality in clustering by means of the wavelet transform. *Computer Journal* 43, 107–120 (2000)
- [64] Murtagh, F., Starck, J.L.: Quantization from Bayes factors with application to multilevel thresholding. *Pattern Recognition Letters* 24, 2001–2007 (2003)
- [65] Ostrowski, A.: Über einige Lösungen der Funktionalgleichung $\phi(x)\phi(y) = \phi(xy)$. *Acta Mathematica* 41, 271–284 (1918)
- [66] Rammal, R., Angles d'Auriac, J.C., Doucot, B.: On the degree of ultrametricity. *Le Journal de Physique - Lettres* 46, 945–952 (1985)
- [67] Rammal, R., Toulouse, G., Virasoro, M.A.: Ultrametricity for physicists. *Reviews of Modern Physics* 58, 765–788 (1986)
- [68] Reiter, H., Stegeman, J.D.: Classical Harmonic Analysis and Locally Compact Groups, 2nd edn. Oxford University Press, Oxford (2000)
- [69] Van Rooij, A.C.M.: Non-Archimedean Functional Analysis. Marcel Dekker, New York (1978)
- [70] Schikhof, W.H.: Ultrametric Calculus. Cambridge University Press, Cambridge (1984); (Chapters 18, 19, 20, 21)
- [71] Schwarz, G.: Estimating the dimension of a model. *Annals of Statistics* 6, 461–464 (1978)
- [72] Seda, A.K., Hitzler, P.: Generalized distance functions in the theory of computation. *Computer Journal* 53, 443–464 (2010)
- [73] Sibson, R.: Slink: an optimally efficient algorithm for the single-link cluster method. *Computer Journal* 16, 30–34 (1980)
- [74] Simon, H.A.: The Sciences of the Artificial. MIT Press, Cambridge (1996)
- [75] Steinley, D.: K-means clustering: a half-century synthesis. *British Journal of Mathematical and Statistical Psychology* 59, 1–3 (2006)

- [76] Steinley, D., Brusco, M.J.: Initializing K-means batch clustering: a critical evaluation of several techniques. *Journal of Classification* 24, 99–121 (2007)
- [77] Vempala, S.S.: The Random Projection Method. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 65. American Mathematical Society (2004)
- [78] Volovich, I.V.: Number theory as the ultimate physical theory, Technical report (1987); Preprint No. TH 4781/87, CERN, Geneva
- [79] Volovich, I.V.: p-Adic string. *Classical Quantum Gravity* 4, L83–L87 (1987)
- [80] Weyl, H.: Symmetry. Princeton University Press, Princeton (1983)
- [81] Xu, R., Wunsch, D.: Survey of clustering algorithms. *IEEE Transactions on Neural Networks* 16, 645–678 (2005)

Chapter 6

Randomized Algorithm of Finding the True Number of Clusters Based on Chebychev Polynomial Approximation

R. Avros¹, O. Granichin², D. Shalymov²,
Z. Volkovich¹, and G.-W. Weber³

¹ Ort Braude College of Engineering, Karmiel 21982, Israel

r_avros@braude.ac.il, v1volkov@braude.ac.il

² Saint Petersburg State University, Russia

Oleg_granichin@mail.ru, shalydim@mail.ru

³ Institute of Applied Mathematics, Middle East Technical University,
06531 Ankara, Turkey University of Siegen (Germany),

University of Aveiro (Portugal), Universiti Teknologi Malaysia, Skudai
gweber@metu.edu.tr

Abstract. One of the important problems arising in cluster analysis is the estimation of the appropriate number of clusters. In the case when the expected number of clusters is sufficiently large, the majority of the existing methods involve high complexity computations. This difficulty can be avoided by using a suitable confidence interval to estimate the number of clusters. Such a method is proposed in the current chapter.

The main idea is to allocate the jump position of the within-cluster dispersion function using Chebyshev polynomial approximations. The confidence interval for the true number of clusters can be obtained in this way by means of a comparatively small number of the distortion calculations. a significant computational complexity decreasing is proven. Several examples are given to demonstrate the high ability of the proposed methodology.

Keywords: Cluster analysis, Clustering, Cluster stability, Randomized algorithms.

1 Introduction

Cluster analysis methods can be roughly divided into two categories: clustering and validation approaches. In the latter methods, which are intended to estimate the optimal (“true”) number of clusters, the obtained partitions are evaluated according to a given rule, and the number of clusters is selected on the basis of the optimal rule value. This crucial problem, known as an “ill posed” problem [23,30], may have several solutions. For example, the answer may depend on the data measurement units. The selection of the particular clustering algorithm used here is another major difficulty since the partitions constructed are

intended more or less to reflect the inner hidden data structure. Solutions given by different algorithms can be essentially different and lead to quite different conclusions about the stable cluster configurations. To illustrate this phenomenon, we consider partitions into two clusters created by two algorithms for a dataset simulated on the real line as a mix of three Gaussian components. The partition obtained using the randomly initialized standard k -means algorithm is presented in Figure 1, while Figure 2 demonstrates the result obtained using the *ClassificationEM* algorithm, a version the *EM* approach introduced in [8]. It can be seen that the k -means algorithm reveals “false” stable two-cluster construction, even as a more flexible *CEM* method leads to an unreliable two-cluster structure. Figure 2 demonstrates the result obtained by the *CEM* algorithm.

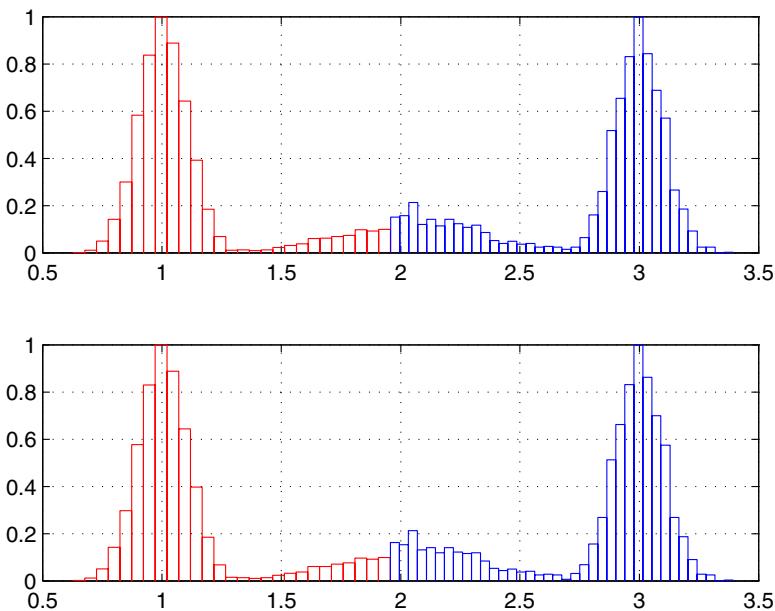


Fig. 1. Outcomes of repeated clusterings by means of the standard k -means algorithm

The current manuscript discusses a new approach to the determination of the true number of clusters. Although a great number of methods have been proposed to tackle this problem, none of them has yet been accepted as a superior one. We review here several approaches because they can be incorporated into the proposed in this paper methodology.

Geometrical approaches were employed by Dunn [17], Hubert and Schultz [29] (C-index), Calinski-Harabasz [7], Hartigan [26], Krzanowski-Lai [35], Sugar-James [45], Gordon [22], Milligan and Cooper [40], and Tibshirani, Walter and Hastie [47] (the Gap Statistic method). Stability models compare the pairs of

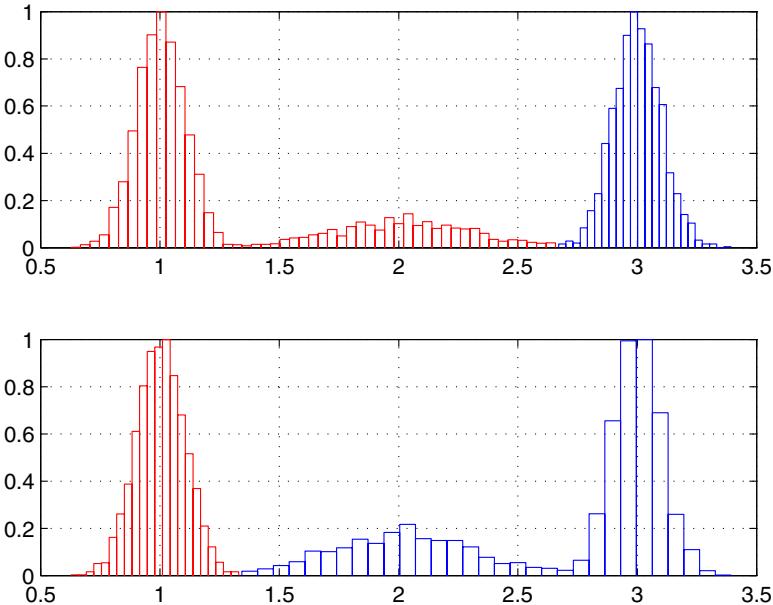


Fig. 2. Outcomes of repeated clusterings by means of the *CEM*

clustered samples obtained by application of a clustering algorithm in which the partition consistency is interpreted as the partition reliability [9], and thus the “true” number of clusters corresponds to the maximal stability score. In the framework of this methodology, Levine and Domany [38], Ben-Hur, Elisseeff and Guyon [3], and Ben-Hur and Guyon [4] represented the stability criteria through the fractions of times that pairs of elements provide the same membership within the clustering algorithm reiterations. Bel Mufti, Bertrand, and El Moubaraki [41] used the Loevinger’s isolation measure to determine the stability function.

Another group of methods utilizes external correlation indexes as a stability degree. For example, such a method was implemented in the known Clest approach of Dudoit and Fridlyand [16]. A general prediction resampling procedure was proposed by Roth, Lange, Braun and Buhmann [36,37]. Tibshirani and Walther [46] described a process of comparable forecast strength process. Jain and Moreau [31] considered the dispersions of empirical distributions as a stability measure. Nonparametric density estimation methodology relates the clusters to the probability density function peaks. The clustering assigns each item to a “domain of attraction” of the density modes. Evidently, Wishart [51] appears to be the first to propose looking for modes in order to reveal the cluster structure. Apparently, this idea was formulated by Hartigan ([27], Section 11, and [28]) to introduce the notion of high density clusters. The number of clusters is given here as the number of regions where the density is higher than a certain specified level. Thus, clusters are viewed as isolated islands of “high”

density in the ocean of “low” densities (see, e.g., [10,11,44]). The goodness-of-fit-test procedures should be also mentioned. Pelleg and Moore [42] developed an X -means algorithm, where the best score of the so-called Bayesian Information Criterion [32] is found in order to determine the true number of clusters. Hamerly and Elkan [25] applied another scoring criteria to the G -means algorithm, namely, the null hypothesis about the clusters drawn from a Gaussian population was tested by means of a statistical projection procedure. Furthermore, Feng and Hamerly [18] reported the PG -means (PG stands for projected Gaussian) algorithm, which also operates with projections onto the clustering model. The PG -means employs the Gaussian mixture model incorporated into the Expectation-Maximization algorithm. Other applications of the goodness-of-fit test were suggested by Volkovich, Barzily and Morozensky [50], Barzily, Volkovich, Akteke-Ozturk and Weber [2] and Volkovich and Barzily [48]. Here the models of clusters were created based on the model of well-mixed samples within the clusters. Volkovich, Barzily, Avros and Toledano-Kitai [49] used the binomial model of the K -Nearest Neighbors belonging to the own point’s sample. Another model suggested by Volkovich, Barzily and Morozensky [50] and Barzily, Volkovich, Akteke-Ozturk and Weber [2] considers the probability distances between clustered samples taken from the cluster cores and the whole population, correspondingly.

The cluster validation procedures mentioned above usually check the cluster quality for all possible numbers of clusters in a given area. The strategy could be in the geometrical approach to generate the distortion curve for the input data by running a standard clustering algorithm, such as the k -means, for all values of k between 1 and k_{\max} and computing the resulting clustering distortions. In the case when the suggested number of clusters is sufficiently large, such a methodology leads to high complexity of the computations.

This difficulty can be avoided by a preliminary estimation of the appropriate number of clusters by means of a suitable confidence interval. Such a method is proposed in the present article. Generally speaking, the presented methodology is based on the employment of the “elbow criterion”. This rule recommends selecting of a number of clusters in order that further clusters splitting does not provide more relevant information. It is expressed as a sharp jump point on the graph of the explained by the clusters variance fraction as a function of the number of clusters. The main idea is to compute a small amount of differential distortion function values and to allocate the jump position relying on its approximations by fixed set of Chebyshev polynomials with uniformly bounded coefficients. A confidence interval for the true number of clusters can be obtained by comparatively small amount of the distortion calculations.

The rest of the paper is organized in the following way. Subsection 2.1 is devoted to description of the known clustering algorithm such k -means and PAM. Several stability based cluster validation methods are presented in Subsection 2.2. Geometrical approaches are discussed in Subsection 2.3. A randomized algorithm for estimation of the true number of clusters is explained in Section 3. Examples of the algorithm application are presented in Section 4.

Notations

- \mathbf{X} is a finite subset of the Euclidean space \mathbb{R}^d to be clustered. The elements of \mathbf{X} are represented in the form $\mathbf{x} = (x_1, \dots, x_d)$;
- $N_{\mathbf{X}} = |\mathbf{X}|$ is the size of the set \mathbf{X} ;
- $\mathbb{R}_+ = [0, +\infty)$;
- $\langle \cdot, \cdot \rangle$ denotes the inner product of two elements of \mathbb{R}^d ;
- $tr(A)$ denotes the trace of a matrix A ;
- k denotes the number of clusters being considered; $Cl(\mathbf{X}, k)$ is a clustering algorithm dividing the set \mathbf{X} into k non-overlapping clusters;
- k^* denotes the true number of clusters.

2 Clustering

2.1 Clustering Methods

Partitioning and hierarchical methods are frequently used in the clustering process. According to hierarchical approaches, a collection of nested partitions is built based on point clusters which include just one data element. On the other hand, the whole set of points, which is, actually, the universal cluster, is found at the end of the process. The traditional representation of the cluster hierarchy is a two-dimensional diagram tree. The true number of clusters is not usually specified; instead, an appropriate solution is obtained by cutting the dendrogram tree at a certain level. Hierarchical procedures are divided into agglomerative (“bottom-up”) or divisive (“top-down”). Divisive (“top-down”) algorithms start from the whole set and successively separate the items into improved partitions. Agglomerative (“bottom-up”) methods produce series of fusions of the data elements into groups.

Partitioning approaches are based on an optimization procedure applied to an objective function which governs the partition quality and can produce a tighter cluster structure than in the case of the hierarchical methods. Additionally, such procedures have lower complexity since a large number of variables are clustered into a small number of groups. However, these approaches often provide non-globular clusters and, consequently, different methods may generate different results. Moreover, a partition may be constructed for almost any number of clusters without its verification.

Let us consider a general clustering model. A partition of the set \mathbf{X} is defined as a collection of non-empty disjoint subsets

$$\boldsymbol{\Pi}_k(\mathbf{X}) = \{\pi_i(\mathbf{X}), i = 1, \dots, k\},$$

such that

$$\mathbf{X} = \bigcup_{i=1}^k \pi_i(\mathbf{X}).$$

The elements $\pi_i(\mathbf{X})$ of the partition are called clusters. The partition quality, intended to be minimized in the partitioning approaches, is defined for a given real-valued function q of X subsets as

$$Q(\Pi_k(\mathbf{X})) = \sum_{i=1}^k q(\pi_i(\mathbf{X})). \quad (1)$$

Hence, clustering can be interpreted as a particular case of a global optimization problem where the partition $\Pi^{(0)}$, which optimizes the objective function Q , is to be found. Function q can be constructed using a distance-like function $d(x, y)$ and a predefined set of k cluster centroids (medoids) $\mathbf{C}=(\mathbf{c}_1, \dots, \mathbf{c}_k)$. The partition of \mathbf{X} is built as follows:

$$\pi_i(\mathbf{X}) = \{\mathbf{x} \in \mathbf{X} : d(\mathbf{c}_i, \mathbf{x}) \leq d(\mathbf{c}_j, \mathbf{x}), \text{ for } j \neq i\}, \quad i = 1, \dots, k,$$

(Ties are broken arbitrarily). Alternatively, the centroid is given by a partition in the form:

$$c(\pi_j) = \arg \min_c \left\{ \sum_{x \in \pi_j} d(\mathbf{c}, \mathbf{x}) \right\}.$$

Thus

$$q(\pi_j(\mathbf{X})) = \sum_{x \in \pi_j} d(c(\pi_j), \mathbf{x})$$

and the mentioned optimization problem is reduced to finding a set of centroids as the solution of the problem being considered:

$$\mathbf{C} = \arg \min_{\mathbf{c} \in \mathbf{C}} \left\{ \sum_{j=1}^k \sum_{x \in \pi_j(\mathbf{X})} d(\mathbf{c}_j, \mathbf{x}) \right\}. \quad (2)$$

In the case when $d(\cdot, \cdot)$ is the squared standard Euclidean distance, the objective function is represented as

$$\min_{\mathbf{C}} R(\mathbf{C}) = \sum_{j=1}^k \sum_{\mathbf{x} \in \mathbf{X}} \min_{c_j} \|\mathbf{x} - c_j\|^2. \quad (3)$$

The well-known k -means algorithm was proposed in [19] to provide an approximate solution to this optimization task.

Input: \mathbf{X} is the set to be clustered; k is the number of clusters.

k -means Algorithm:

1. Randomly place k items as initial cluster centers (centroids) into the space represented by \mathbf{X} ;
2. Assign each point $\mathbf{x} \in \mathbf{X}$ to the nearest cluster center;
3. Recompute the new centroids as the group mean values once all elements have been assigned;

4. Repeat Steps 2 and 3 until the convergence criterion is met (e.g. the assignment is no longer changed or the centroids do not move any longer).

The algorithm has the computational complexity $O(kN_X)$ and frequently constructs the so-called “non-optimal stable partitions”. To overcome this problem the incremental k -means algorithm can be used (see, e.g. [15,12,13,14,34]). The k -means approach can be viewed as a version of the famous Expectation Maximization (EM) approach which suggests the Gaussian Mixture Model (GMM) of data in the clustering context. (see, e.g. [1,8,21]):

$$f(\mathbf{x}) = \sum_{j=1}^k p_j G(\mathbf{x}|\mu_j, \Gamma_j), \quad (4)$$

where $f(\mathbf{x})$ is the underlying data density; $G(\mathbf{x}|\mu, \Gamma)$ is the Gaussian density (with the mean value μ and the covariance matrix Γ).

The *EM*-method maximizes the log likelihood function:

$$L = \sum_{\mathbf{x} \in \mathbf{X}} \log \left(\sum_{j=1}^k p_j G(\mathbf{x}|\mu_j, \Gamma_j) \right).$$

Celeux and Govaert [8] demonstrated that the k -means approach appears in the case when the cluster proportions are equal to each other:

$$p_1 = p_2 = \dots = p_k \quad (5)$$

and the covariance matrix has the form

$$\Gamma_j = \sigma^2 I, \quad j = 1, \dots, k,$$

where I is the identity matrix and σ^2 is an unknown parameter.

The Partitioning Around Medoids (PAM) clustering procedure ([33], Chapter 2) is the most common realization of k -medoid approach. In contrary to the k -means algorithm, the k -medoid method seeks for the data elements, named medoids, as the cluster centers. The corresponding objective function, similar to (2), has the form:

$$\min_{C \in \mathbf{X}} R(C) = \sum_{j=1}^k \sum_{\mathbf{x} \in \mathbf{X}} \min_{c_j \in \mathbf{X}} d(c_j, \mathbf{x}). \quad (6)$$

The PAM algorithm which gives an approximate solution of this problem consists of two phases:

- **BUILD** - constructing initial clustering;
- **SWAP** - refining the clustering.

Input: \mathbf{Dis} is a prepared beforehand $N_X * N_X$ dissimilarity matrix between the items to be clustered; k is the number of clusters.

PAM Algorithm:

1. **BUILD Phase:** Build the set \mathbf{C} of medoids which minimizes the objective function (6);
2. **SWAP Phase:** Until no change, do:
 3. Assign each point $\mathbf{x} \in \mathbf{X}$ to the nearest cluster center (medoid);
 4. For each $\mathbf{c} \in \mathbf{C}$ and for each $\mathbf{x} \in \mathbf{X} \setminus \mathbf{C}$:
 - (a) Compute the total cost S of swapping medoid \mathbf{c} with \mathbf{x} ;
 - (b) If $S < 0$, swap \mathbf{c} with \mathbf{x} to create a new set of medoids;
 5. end loop until.

The PAM algorithm is more robust than the k -means algorithm, especially in the case of noisy data with outliers; however, it has higher computational complexity of $O(k(N_X - k)^2)$ for each iteration. An important property of the PAM algorithm is its ability to construct clusterings based on any distances. Additionally, medoids provide more robust cluster centers as compared to k -means centroids.

2.2 Stability Based Methods

We have already mentioned above that stability-based determination of the true number of clusters is a very common cluster analysis tool. Several approaches of this kind are discussed below.

2.2.1 External Indexes

External indexes are often used in cluster stability approaches. These nominal measures of associations are based on the so-called cross-tabulation or contingency tables. Let us suppose that Π_r and Π_c are two partitions of the same dataset of size n into r and c clusters, respectively. Denote by n_{ij} the number of elements belonging to cluster i of Π_r and to cluster j of Π_c ($i = 1, \dots, r$, $j = 1, \dots, c$). The Cramer's V statistic measure of the strength of association between two (nominal) categorical variables is defined in the following way:

$$V = \sqrt{\frac{\chi^2}{N_X * \min(r - 1, c - 1)}},$$

where

$$\chi^2 = \sum_{i=1}^r \sum_{j=1}^c \frac{(n_{ij} - e_{ij})^2}{e_{ij}}, \quad e_{ij} = \frac{n_i^{(r)} * n_j^{(c)}}{N_X}$$

is the chi-square statistic and

$$\begin{aligned} n_i^{(r)} &= \sum_{j=1}^c n_{ij}, \quad i = 1, \dots, r, \\ n_j^{(c)} &= \sum_{i=1}^r n_{ij}, \quad j = 1, \dots, c. \end{aligned}$$

Denote

$$Z = \sum_{j=1}^c \sum_{i=1}^r n_{ij}^2.$$

The index of partition association introduced by Rand [43] is

$$R = 1 + \left(\frac{Z - 0.5 * \left(\sum_{j=1}^c \left(n_j^{(c)} \right)^2 + \sum_{i=1}^r \left(n_i^{(r)} \right)^2 \right)}{\binom{n}{2}} \right).$$

Another index was proposed by Jain and Dubes [30]:

$$JD = \frac{(Z - N_X)}{\left(\sum_{j=1}^c \left(n_j^{(c)} \right)^2 + \sum_{i=1}^r \left(n_i^{(r)} \right)^2 - Z - N_X \right)},$$

while Fowlkes and Mallows [20] suggested the coefficient

$$FM = \frac{(Z - N_X)}{2\sqrt{\sum_{j=1}^c \binom{n_j^{(c)}}{2} \sum_{i=1}^r \binom{n_i^{(r)}}{2}}}.$$

Apparently, the indexes R and FM are linear functions of each other because they are linear functions of Z . The adjusted R index is the corrected-for-chance version of the Rand index, standardized in such a way that its expected value is 0 if the partitions are random and 1 if they correspond to each other perfectly. The standardization is performed as follows:

$$Ind' = \frac{(Ind - E(ind))}{(Ind_{max} - E(ind))},$$

where Ind_{max} is the maximal value of the index Ind . The common null hypothesis suggests that the contingency table is built on the assumption of the generalized hyper-geometric distribution and that partitions Π_r and Π_c are mutually independent. In this case, the adjusted R index equals zero. These indexes were used in the Clest method [16] as a measure of clustering stability.

2.2.2 Clest Algorithm

The Clest algorithm splits the clustered data \mathbf{X} into two non-overlapping halves \mathbf{L}_b and \mathbf{T}_b , called a learning and a test sets, respectively. The main idea of the method, proposed, apparently, by Breckenridge [5], is constructing two partitions on \mathbf{T}_b in such a way that the first partition is obtained by applying the clustering procedure directly, while the second one is obtained as an extension of the \mathbf{L}_b partition to \mathbf{T}_b . The two partitions are compared using one of the described-above external indices. The true number of clusters corresponds to the largest

significant evidence against the null hypothesis about the absence of the cluster structure. The algorithm can be presented in the following form:

For each tested number of clusters k , $2 \leq k \leq k_{\max}$, do 1-4.

1. Repeat B times:

- (a) Split the original dataset into two non-overlapping sets \mathbf{L}_b and \mathbf{T}_b ;
- (b) Construct $\Pi(\mathbf{L}_b) = Cl(\mathbf{L}_b, k)$;
- (c) Construct a classifier $C(\mathbf{L}_b)$ based on $\Pi(\mathbf{L}_b)$;
- (d) Apply the classifier $C(\mathbf{L}_b)$ to the test set \mathbf{T}_b and get $\Pi_1(\mathbf{T}_b)$;
- (e) Construct $\Pi_2(\mathbf{T}_b) = Cl(\mathbf{T}_b, k)$;
- (f) Calculate the external index $I_{k,b}$ comparing $\Pi_1(\mathbf{T}_b)$ and $\Pi_2(\mathbf{T}_b)$;

2. Consider the observed median value of the external index

$$t_k = \text{median}(I_{k,1}, \dots, I_{k,B}).$$

3. Produce B_0 datasets under an appropriate null hypothesis of the absence of the cluster structure and repeat the above steps 1 and 2 until both of them getting B_0 statistics $t_{k,1}, \dots, t_{k,B_0}$.

4. Consider the average of the above B_0 statistics:

$$t_k^{(0)} = \frac{1}{B_0} \sum_{b=1}^{B_0} t_k,$$

and denote by p_k the proportion of those $t_{k,b}$, $1 \leq b \leq B_0$, that are at least as large as the observed statistic t_k , i.e., the p -value for t_k . Let

$$d_k = t_k - t_k^{(0)}$$

denote the difference between the observed similarity statistic and its estimated expected value under the null hypothesis.

5. Introduce the set A as

$$A = \{2 \leq k \leq k_{\max} : p_k \leq p_{\max}, d_k \geq d_{\min}\},$$

where p_{\max} and d_{\min} are predefined parameters. This set is empty if no cluster structure has been found. Otherwise, the number of clusters k corresponds to the largest significant difference statistic d_k :

$$k = \arg \max_{k \in K} d_k.$$

The authors used the PAM algorithm described in Section 2.1, the naive Bayes classifier, the FM index (see, Section 2.2.1), $B = B_0 = 20$, and $p_{\max} = d_{\min} = 0.05$.

2.3 Geometrical Cluster Validation Criteria

The majority of geometrical cluster validation criteria are based on the total dispersion, or total scatter, matrices T_k as well as on between and within k -cluster sums of squares B_k and W_k defined for a given partition $\Pi_k(\mathbf{X})$, $k \geq 2$, as (see, e.g. [39]):

$$T_k = \sum_{j=1}^k \sum_{z \in \pi_j} (\mathbf{x} - \bar{\mu})(\mathbf{x} - \bar{\mu})^t, \quad (7)$$

$$B_k = \sum_{j=1}^k |\pi_j| (\bar{\mu}_j - \bar{\mu})(\bar{\mu}_j - \bar{\mu})^t, \quad W_k = \sum_{j=1}^k \sum_{\mathbf{x} \in \pi_j(\mathbf{X})} (\mathbf{x} - \bar{\mu}_j)(\mathbf{z} - \bar{\mu}_j)^t, \quad (8)$$

where $\bar{\mu}$ is the mean point of the set \mathbf{X} , and $\bar{\mu}_j$ are the arithmetic means of $\pi_j(\mathbf{X})$, $j = 1, \dots, k$. It should be noted that $T_k = W_k + B_k$. The first method proposed for the evaluation of the true number of clusters appears to be the, so called, “elbow criterion”, which employs the graph of the within-cluster dispersion W_k as a function of the number of clusters k . As a rule, this characteristic decreases with the increase of the number of clusters. An attempt to divide a cluster into subgroups further decreases the criterion value when well-separated clusters are considered. In this case, the W_k graph has a sharp decline. The number of clusters is determined at this point and this is where the method got its name from. However, the appropriate “elbow” cannot always be explicitly recognized. Attempts to develop approaches for detecting the “elbow” were made in many studies [7,27,40,47], etc.

The “elbow” phenomenon is illustrated in Figure 4, where the graphs of the functions $\log(W_k)$ (observed), marked in blue, and $\log(W_k^*)$ (reference), marked in red, employed in calculating the Gap statistic [47] are presented for a four-component dataset described in Figure 3. The reference function values are found on the basis of an appropriate null data distribution which appears to be the least favorable from the standpoint of clustering. The uniform distribution is usually used for this purpose.

The inner indexes based on the “elbow” methodology are often employed in the procedures of partitioning, the stopping rules being applied to determine the number of clusters. The stopping-rule (index) value is found, in this case, for a set of cluster solutions and the extreme value, which depends on the particular stopping rule, indicates the most appropriate solutions.

1. The Calinski-Harabasz index (pseudo- F index) [7] is defined as

$$CH_k = \frac{tr(B_k)/(k-1)}{tr(W_k)/(N_X - k)}.$$

The estimated true number of clusters is determined as the value of k that corresponds to the maximum of CH_k . The Calinski-Harabasz index was the best of the 30 indices tried on synthetic data by Milligan and Cooper [40].

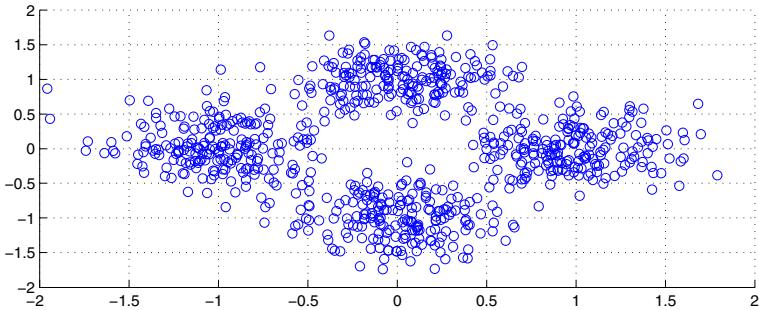


Fig. 3. Scatter plot of a four-component dataset

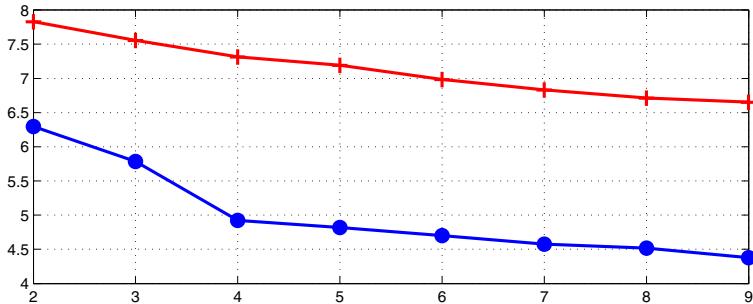


Fig. 4. The graphs of the logarithms of the Gap statistic values

2. The Krzanowski-Lai index [35] is defined by the following relationships:

$$\text{diff}_k = (k-1)^{2/d} \text{tr}(W_{k-1}) - k^{2/d} \text{tr}(W_k),$$

and

$$KL_k = |\text{diff}_k| / |\text{diff}_{k+1}|.$$

The estimated number of clusters corresponds to the maximal value of the index KL_k .

3. The Hartigan index [26] is defined as

$$h_k = \left(\frac{\text{tr}(W_k)}{\text{tr}(W_{k+1})} - 1 \right) (N_X - k - 1).$$

The estimated number of clusters is the smallest value of $k \geq 1$ for $h_k \leq 10$.

4. The above-mentioned Gap method [47] also deals with the values of $\text{tr}(W_k)$, $k \geq 1$, such that B reference datasets are created under the null hypothesis. Then the reference datasets are clustered and the values of

$tr(W_k^1), \dots, tr(W_k^B)$ are computed. The estimated value of the Gap statistic is found as

$$gap_k = \frac{1}{B} \sum_b \log(tr(W_k^b)) - \log(tr(W_k)).$$

Let sd_k be the standard deviation of $\log(tr(W_k^b))$, $1 \leq b \leq B$, and

$$sd_k = sd_k \sqrt{1 + \frac{1}{B}}.$$

The estimated true number of clusters corresponds to the smallest value of $k \geq 1$ that satisfies the inequality

$$gap_k \geq gap_{k^*} - sd_{k^*},$$

where $k^* = \operatorname{argmax}_{k \geq 1} (gap_k)$.

5. A modification of the above approach was proposed by Sugar and James in the framework of the rate distortion theory [45]. In this version of a “jump” method, a distortion curve is computed for d -dimensional data. The latter is assumed to have an underling distribution composed of G components with the common covariance matrix Γ . The distortion value has the form

$$D_k = \frac{1}{d} \min_{c_1, \dots, c_k} E[(\mathbf{X} - c_x)^t \Gamma (\mathbf{X} - c_x)],$$

where c_1, \dots, c_k is a set of k cluster centers obtained by running a standard clustering procedure such as the k -means algorithm; c_x is the center nearest to a given sample of \mathbf{X} . Actually, this version of W_k is the average Mahalanobis distance per dimension between the datum and the family of the cluster centers. Next step, a “jumping differential” curve is constructed according to the following rule:

$$J_k = (D_k^{-\lambda} - D_{k-1}^{-\lambda}),$$

where λ is the transformation power. Its preferred option

$$\lambda = (d/2)$$

is obtained from the asymptotic results of the rate distortion theory. Moreover, for sufficiently high values of d , the differential distortion J_k is approximately zero; if the number of clusters is less than the number of components, then the value jumps and increases linearly. Summarizing the above results, we see that, for sufficiently high values of d , the transformed distortion is approximately zero for $k < G$, then jumps abruptly and increases linearly for $k \geq G$. The jump algorithm makes use of this behavior to identify the most likely value of k as the true number of clusters. The estimated number of clusters corresponds to the maximal value of the index J_k . An example of the distortion curve obtained for the data presented in Figure 3 is given in Figure 5.

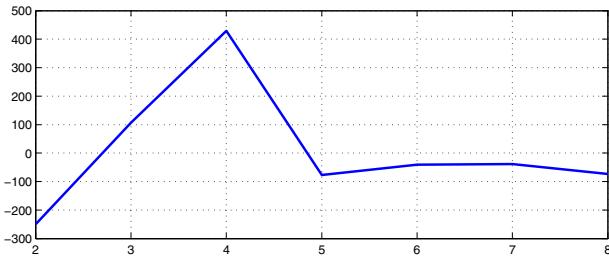


Fig. 5. Graph of the distortion curve

3 Randomized Algorithm

Based on the rate distortion criteria proposed by Sugar and James (see Section 2.3), the task of determining the true number of clusters can be theoretically interpreted as a particular case of a more general problem, namely, the problem of locating the discontinuity points of an implicitly defined function. Let us consider the function of J_k transformed “distortions” mapped into the interval $[0, 1]$ as the index function $I(k)$. This function behaves in a semi-linear way before and after the jump. To determine such jump point, a randomized approach can be used (see [24]). Generally, the problem can be formulated in the following way. Let us take a real-valued function f on the interval $[0, 1]$ having not more than one jump: point of discontinuity $x^* \in [0, 1]$. The considered in [24] problem is: To define the confidence interval for the x^* if the function satisfies conditions:

1. The function $f(\cdot)$ is Lipschitz continuous with a Lipschitz constant C on the intervals $[0, x^*)$ and $(x^*, 1]$;
2. If jump discontinuity exists, then the jump size at this point is above a certain constant value $B > 0$.

The first constant C represents the “smoothness” of the index function on the part of the interval where the function is continuous. The second constant B characterizes a possible “jump” of the index function at the point x^* which corresponds, in our context, to the true number of clusters. Let k_{\max} be the maximal number of clusters tested. Obviously, the case $B \gg C$ appears to be the most interesting because the behavior of the index function scaled by k_{\max} near point the x^* should be essentially different from its behavior at other points.

The scenario optimization method discussed in [6] is an effective technique for solving convex optimization problems with large amount of constraints in a probabilistic setting. For any given sufficiently small positive values ϵ and β , the number of random trials N is *a priori* defined for a given sufficiently small positive confidence parameters ϵ and $(1 - \beta)$. Thus, the solution obtained for merely N constraints satisfies all the others with the probability of $(1 - \beta)$ except for a set whose probability does not exceed ϵ .

To implement the above methodology in the framework of the clustering concept, consider the transformed “distortions” I proposed by Sugar and James [45]. For the sake of generality, we assume that

$$I(0) = I(1)$$

and introduce a continuous piecewise linear function f in the form:

$$f_I\left(\frac{k}{k_{\max}}\right) = I(k),$$

$$f_I(x) = I(k) + \left(x - \frac{k}{k_{\max}}\right)(I(k+1) - I(k))$$

for

$$\frac{k}{k_{\max}} \leq x \leq \frac{k+1}{k_{\max}}, k = 0, \dots, k^* - 2, k^*, \dots, k_{\max} - 1,$$

$$f_I(x) = I(k^* - 1)$$

for

$$\frac{k^* - 1}{k_{\max}} \leq x \leq \frac{k^*}{k_{\max}}.$$

In this case, the analogs of the above restrictions 1 and 2 are the following:

1. $C \geq \max_{j=2, \dots, k^*-1, k^*+1, \dots, k_{\max}} |I(j) - I(j-1)| * k_{\max}$,
2. $B \leq |I(k^*) - I(k^* - 1)|$.

Thus, it follows that

1. The function f_I has no more than one jump discontinuity $x^* \in [0, 1]$.
2. The function $f_I(\cdot)$ is Lipschitz continuous with the Lipschitz constant C on the intervals $[0, x^*]$ and $(x^*, 1]$.
3. If there exists jump discontinuity, then the function jump size at the jump point is above some constant value $B > 0$.

An algorithm which implements the approach under consideration can be described as follows:

1. Choose the reliability parameters $\beta \in (0, 1)$.
2. Choose the parameter M presenting the highest power in the approximation of the function f_I by means of Chebyshev polynomials:

$$p_m(x) = \cos(m \arccos x), m = 0, 1, 2, \dots, M. \quad (9)$$

3. Choose a number $N \geq M$ and set the number of a group points $T > 1$:

$$T = \left\lceil \frac{4C}{\beta BN} - \frac{1}{N} \right\rceil. \quad (10)$$

4. Choose randomly T sets of points having size N in the interval $(0, 1)$:

$$Z_t = \{z_{tn}, n = 1, \dots, N\}, t = 1, \dots, T$$

and denote

$$Z = \bigcup_t Z_t.$$

In the proof of Theorem 1, it will be demonstrated that the largest distance between two sequential points belonging to Z does not exceed $B/4C$ with the probability of $(1 - \beta)$.

5. For each one of the groups $Z_t, t = 1, \dots, T$ construct the uniform approximation for $f_I(x)$:

$$g_t(x) = \sum_{m=0}^M d_{tm} p_m(x), t = 1, \dots, T \quad (11)$$

minimizing the lost

$$\gamma_t = \max_{x \in Z_t} |g_t(x) - f_I(x)|$$

subject to

$$|d_{tm}| \leq D, m = 0, \dots, M, t = 1, \dots, T,$$

where D is a constant.

Here a convex optimization MathLaB's TOOLBOX (YALMIP, SeDuMi or cvx) can be applied.

If one of the approximation problems is not resolved then return to Step 2 with another parameters M, N, K, D .

6. Define the functions

$$\chi(x) = \max_{t=1, \dots, T} g_t(x) - \min_{t=1, \dots, T} g_t(x), x \in (0, 1) \quad (12)$$

and

$$h(x) = \max_{z \in [z_l(x), z_r(x)]} \max_{t=1, \dots, T} |g'_t(z)|, \quad (13)$$

where

$$z_l(x) = \arg \max \{z \in Z : z \leq x\}, x \in (0, 1)$$

and

$$z_r(x) = \arg \min \{z \in Z : z > x\}, x \in (0, 1).$$

7. Calculate

$$\gamma = \max_t \gamma_t \quad (14)$$

and introduce the high line (the level of decision-making)

$$L(x) = \frac{3B}{4} - \frac{B}{4C} h(x) - 2\gamma.$$

The interval

$$\Delta = \{\tilde{x} = xk_{max} : x \in (0, 1), \chi(x) > L(x)\} \quad (15)$$

is not empty with the probability of

$$P = (1 - \beta)$$

and the true number of clusters is located in Δ .

Theorem 1. *If conditions 1 and 2 formulated above hold, then, with the probability of $p = (1 - \beta)$ the set Δ is not empty and contains the point $x^* k_{\max}$ equal to the true number of clusters.*

Sketch of Proof. By virtue of Markov's inequality, it follows from Condition 10 that there exist two points z_{i_l} and z_{j_r} in Z :

$$z_{i_l} < x^*,$$

$$z_{j_r} >= x^*,$$

and

$$|z_{j_r} - z_{i_l}| \leq \frac{B}{4C}$$

with probability of $(1 - \beta)$. Consider the corresponding functions g_i or g_j . It follows from Definition (14) that

$$|f_I(z_{i_l}) - g_i(z_{i_l})| + |f_I(z_{j_r}) - g_j(z_{j_r})| \leq 2\gamma.$$

Consider the intervals $\bar{\Delta}_l = [z_{i_l}, x^*]$ and $\bar{\Delta}_r = [x^*, z_{j_r}]$. The following relationships can be subsequently derived from the above formulas and conditions of the algorithm:

$$\begin{aligned} \chi(x^*) &\geq |g_j(x^*) - g_i(x^*)| \geq |g_j(z_{j_r}) - g_i(z_{i_l})| - (|\bar{\Delta}_l| + |\bar{\Delta}_r|)H \geq \\ &\geq |f_I(z_{j_r}) - f_I(z_{i_l})| - 2\gamma - (|\bar{\Delta}_l| + |\bar{\Delta}_r|)H \geq B - 2\gamma - (|\bar{\Delta}_l| + |\bar{\Delta}_r|)(H + C) \geq \\ &\geq B - 2\gamma - \frac{B}{4C}(H + C), \end{aligned}$$

where H is the maximal derivation $g_i(\cdot)'$ on the interval $[z_{i_l}, z_{j_r}]$.

Finally, taking into account the definition (13) we obtain

$$\chi(x^*) \geq \frac{3B}{4} - \frac{B}{4C}h(x^*) - 2\gamma.$$

4 Examples

Example 1. The first dataset is available at <http://archive.ics.uci.edu/ml/datasets/Libras+Movement>. This datum contains 15 equal sized clusters of 24 instances. Each set refers to one type of hand movements in the official Brazilian signal language LIBRAS.

In the video pre-processing, time normalization is carried out by selecting 45 frames from each video according to a uniform distribution. In each frame, the

centroid pixels of the segmented objects (the hand) are found, which compose the discrete version of curve F comprising 45 points. All curves are normalized in the unitary space. In order to make the hand movements suitable for algorithm analysis, a mapping procedure has been carried out, in which each curve F is mapped in a representation with 90 features, with representing the coordinates of movement.

- Number of Instances - 360;
- Number of Attributes - 91.

We consider the interval $[1, 30]$ which, supposedly, contains the true number of clusters. For each point, the transformed Sugar and James distortion function $I(k)$ is calculated using the Partition Around Medoids approach. The curve obtained in this way is presented in Figure 6.

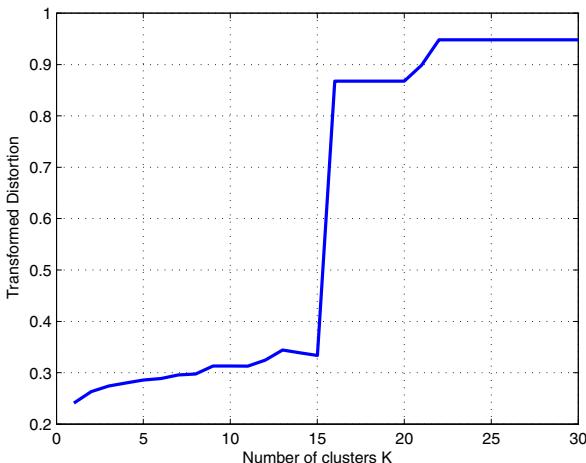


Fig. 6. Sugar and James distortion function $I(k)$ calculated for the official Brazilian signal language LIBRAS dataset

The curve has a “jump” at the point $x^* = 15$ with $B = 0.534$ and

$$c = \max_{k \neq 15} (I(k) - I(k-1)) = 0.049.$$

Thus the true number of clusters is found. The Lipschitz constant of the function f_I is $0.049 * 30 = 1.482$. The total number of the computations of the function $I(k)$ is 30 in this case.

In the framework of the new method proposed here, the number of the points at which the index function values are computed, can be reduced. For example, if we choose $\beta = 0.9$, $M = 4$, $N = 5$, $T = 3$, $D = 0.9$, the function values are

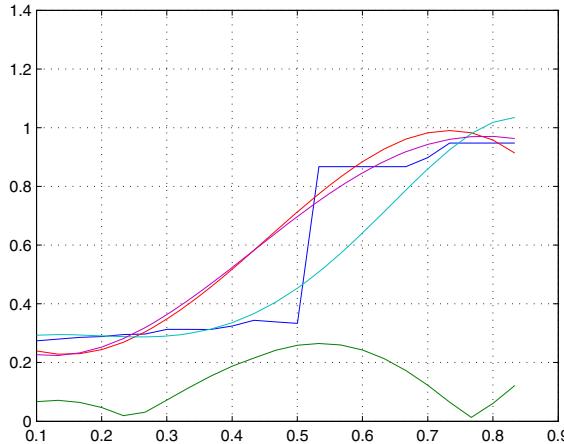


Fig. 7. Approximation curves $g_t(\cdot)$ presented and the resulting function $\chi(\cdot)$

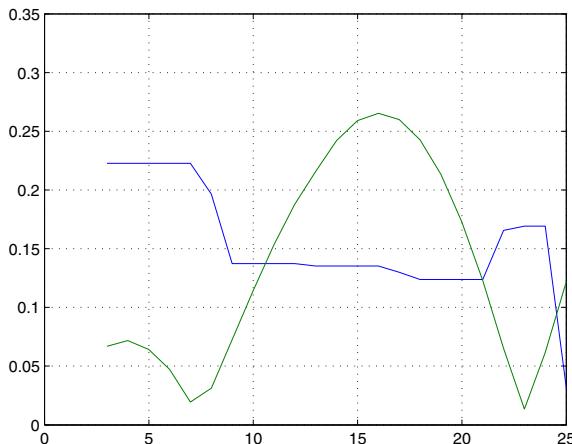


Fig. 8. Level of decision making and the resulting function $\chi(\cdot)$

calculated only at 15 points. As a result, three values of $\{\{0.034, 0.022, 0.021\}$ of γ_t are obtained, which correspond to the three approximation curves $g_t(\cdot)$ presented in Figure 7 together with the resulting function $\chi(\cdot)$.

We do not know the real values of B and C before the calculation of all values. Thus we need to make some “a priory” assumptions about it. If we suggest that $B > 0.5$ and $k_{\max}B/C \approx 10$ then we get the level of decision making which is exposed on Figure 8 together with the resulting function $\chi(\cdot)$. It can be seen that the curve has a peak located near the point $x^* = 15$.

If we choose the confidential interval $[11, 21]$, then 10 additional calculations of the index function $I(x)$ are required to obtain the final solution of the original problem. As the parameter B values increase, the corresponding confidential interval decreases.

Example 2. To check whether the algorithm proposed here can be applied to a large number of clusters, a synthetic dataset was generated. It contained 1024 clusters, each composed of 8 - 16 instances. Instances in each cluster were generated according to a uniform distribution based on a circle with the radius from 10 to 30 (a random value for each cluster).

- Number of Instances - 11245;
- Number of Attributes - 2.

The scatter plot of the synthetic dataset is presented in Figure 9.

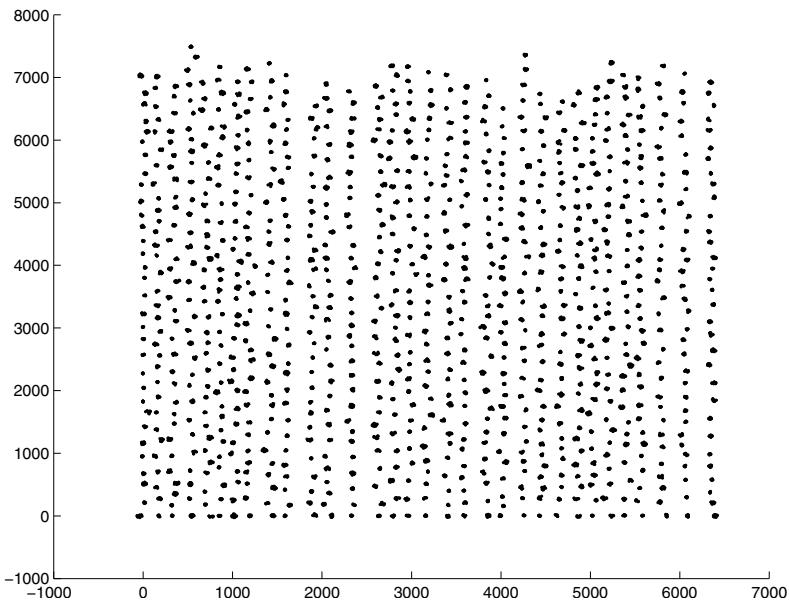


Fig. 9. Synthetic dataset with 1024 clusters

We consider the interval $[1, 3100]$ which contains the real number of clusters. For each point the transformed distortion function $I(k)$ is calculated using the algorithm of Sugar and James. The results are presented in Figure 10.

The scenario approach described above allows us to reduce significantly the number of clustering algorithm rerunning. If we choose $\beta = 0.95$, $M = 8$, $N = 10$, $T = 3$ and $D = 0.7$, then we have to calculate only 30 values of $I(k)$ instead of 3100. Three approximation curves $g_t(\cdot)$ are shown in Figure 11, together with the resulting function $\chi(\cdot)$.

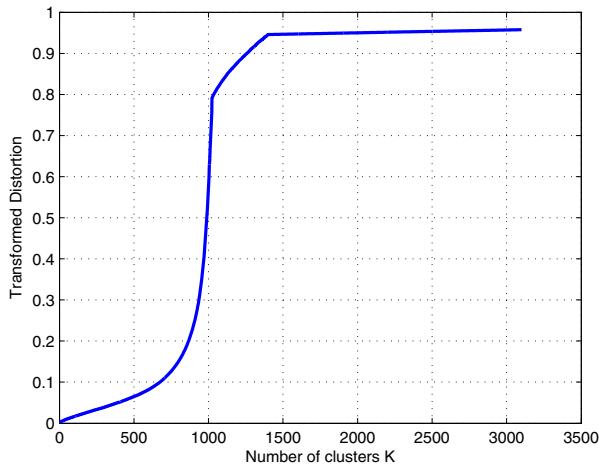


Fig. 10. Sugar and James distortion function $I(k)$

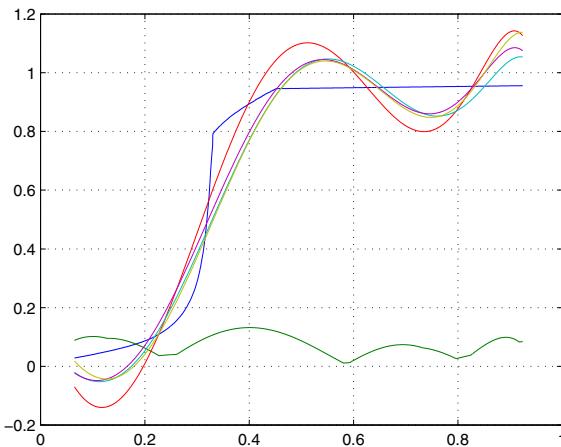


Fig. 11. Level of decision making and the resulting function $\chi(\cdot)$

With the assumption $B > 0.7$ and $k_{\max}B/C \approx 10$ we obtain the level of decision making which is shown in Figure 12 together with the resulting function $\chi(\cdot)$.

A peak near the point $x^* = 1024$ can be observed.

If we choose the confidential interval $[950, 1358]$, then, in order to resolve the original problem, 408 additional calculations of the index function $I(x)$ must be performed. The total number of the computations to be made is 438, which is significantly less than the above number of 3100.

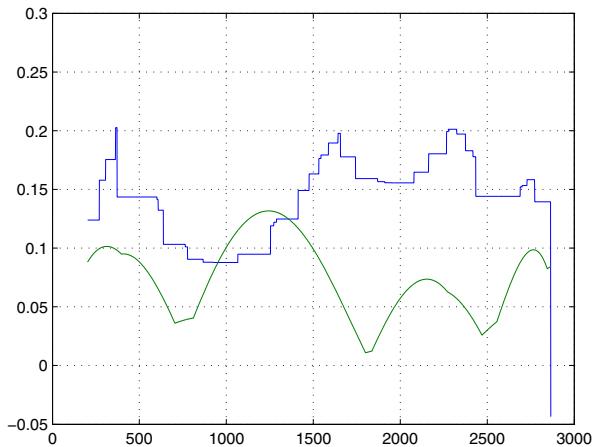


Fig. 12. Level of decision making and the resulting function $\chi(\cdot)$

5 Conclusion

We propose a novel method for the cluster stability assessment based on the ideas of randomized learning theory in the spirit of the well-known “elbow criterion”. The main idea is to compute a small amount of differential distortion function values and to allocate the jump position (an elbow) relying on its approximations by a fixed set of Chebyshev polynomials with uniformly bounded coefficients. A confidence interval for the true number of clusters can be obtained by comparatively small amount of the distortion calculations. As a result one can get sufficiently small confidence interval. The significant decreasing of computations is proved under very general conditions.

References

1. Banfield, J.D., Raftery, A.E.: Model-based gaussian and non-gaussian clustering. *Biometrics* 49, 803–821 (1993)
2. Barzily, Z., Volkovich, Z., Akteke-Ozturk, B., Weber, G.-W.: On a minimal spanning tree approach in the cluster validation problem. *Informatica* 20(2), 187–202 (2009)
3. Ben-Hur, A., Elisseeff, A., Guyon, I.: A stability based method for discovering structure in clustered data. In: Pacific Symposium on Biocomputing, pp. 6–17 (2002)
4. Ben-Hur, A., Guyon, I.: Detecting stable clusters using principal component analysis. In: Brownstein, M.J., Khodursky, A. (eds.) *Methods in Molecular Biology*, pp. 159–182. Humana press (2003)

5. Breckenridge, J.: Replicating cluster analysis: Method, consistency and validity. *Multivariate Behavioral Research* 24, 147–161 (1989)
6. Calafiore, G., Campi, M.C.: The scenario approach to robust control design. *IEEE Trans. Automat. Control* 51(5), 742–753 (2006)
7. Calinski, R., Harabasz, J.: A dendrite method for cluster analysis. *Communications in Statistics* 3(1), 1–27 (1974)
8. Celeux, G., Govaert, G.: A classification *em* algorithm and two stochastic versions. *Computational Statistics and Data Analysis* 14, 315–332 (1992)
9. Cheng, R., Milligan, G.W.: Measuring the influence of individual data points in a cluster analysis. *Journal of Classification* 13, 315–335 (1996)
10. Cuevas, A., Febrero, M., Fraiman, R.: Estimating the number of clusters. *The Canadian Journal of Statistics* 28(2), 367–382 (2000)
11. Cuevas, A., Febrero, M., Fraiman, R.: Cluster analysis: A further approach based on density estimation. *Computational Statistics and Data Analysis* 28, 441–459 (2001)
12. Dhillon, I.S., Kogan, J., Guan, Y.: Refining clusters in high-dimensional text data. In: Dhillon, I.S., Kogan, J. (eds.) *Proceedings of the Workshop on Clustering High Dimensional Data and its Applications at the Second SIAM International Conference on Data Mining*, pp. 71–82. SIAM, Philadelphia (2002)
13. Dhillon, I.S., Kogan, J., Nicholas, C.: Feature selection and document clustering. In: Berry, M.W. (ed.) *A Comprehensive Survey of Text Mining*, pp. 73–100. Springer, Heidelberg (2003)
14. Dhillon, I.S., Mallela, S., Kumar, R.: Enhanced word clustering for hierarchical text classification. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2002)*, pp. 191–200 (2002)
15. Duda, R.O., Hart, P.E., Stork, D.G.: *Pattern Classification*, 2nd edn. John Wiley and Sons, Chichester (2000)
16. Dudoit, S., Fridly, J.: A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biol.* 3(7) (2002)
17. Dunn, J.C.: Well Separated Clusters and Optimal Fuzzy Partitions. *Journal Cybern.* 4, 95–104 (1974)
18. Feng, Y., Hamerly, G.: *pg*-means: learning the number of clusters in data. In: *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems (NIPS)* (December 2006)
19. Forgy, E.W.: Cluster analysis of multivariate data - efficiency vs interpretability of classifications. *Biometrics* 21(3), 768 (1965)
20. Fowlkes, E.W., Mallows, C.L.: A method for comparing two hierarchical clusterings. *J. Am. Stat. Assoc.* 78, 553–584 (1983)
21. Fraley, C., Raftery, A.E.: How many clusters? which clustering method? answers via model-based cluster analysis. *The Computer Journal* 41(8), 578–588 (1998)
22. Gordon, A.D.: Identifying genuine clusters in a classification. *Computational Statistics and Data Analysis* 18, 561–581 (1994)
23. Gordon, A.D.: *Classification*. Chapman and Hall, CRC, Boca Raton, FL (1999)
24. Granichin, O.N., Khalidov, V.I.: Randomized approach to the detection of discontinuity of a function. *Stochastic Optimization in Informatics* 1(1), 73–80 (2005)
25. Hamerly, G., Elkan, C.: Learning the k in k -means. In: *Proceedings of the seventeenth annual conference on neural information processing systems (NIPS)*, December 2003, pp. 281–288 (2003)

26. Hartigan, J.: Statistical theory in clustering. *Journal Classification* 2, 63–76 (1985)
27. Hartigan, J.A.: *Clustering Algorithms*. John Wiley, New York (1975)
28. Hartigan, J.A.: Consistency of single linkage for high-density clusters. *Journal of the American Statistical Association* 76, 388–394 (1981)
29. Hubert, L., Schultz, J.: Quadratic assignment as a general data-analysis strategy. *Br. J. Math. Statist. Psychol.* 76, 190–241 (1974)
30. Jain, A., Dubes, R.: *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs (1988)
31. Jain, A.K., Moreau, J.V.: Bootstrap technique in cluster analysis. *Pattern Recognition* 20(5), 547–568 (1987)
32. Kass, R.E.: A reference bayesian test for nested hypotheses and its relationship to the schwarz criterion. *The Journal of the American Statistical Association* 90(431), 928–934 (1995)
33. Kaufman, L., Rousseeuw, P.J.: *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley and Sons, New York (1990)
34. Kogan, J., Teboulle, M., Nicholas, C.: The entropic geometric means algorithm: an approach for building small clusters for large text datasets. In: Boley, D., et al.(eds.) *Proceedings of the Workshop on Clustering Large Data Sets* (held in conjunction with the Third IEEE International Conference on Data Mining), pp. 63–71 (2003)
35. Krzanowski, W., Lai, Y.: A criterion for determining the number of groups in a dataset using sum of squares clustering. *Biometrics* 44, 23–34 (1985)
36. Lange, T., Braun, M., Roth, V., Buhmann, J.M.: Stability-based model selection (2003)
37. Lange, T., Roth, V., Braun, M.L., Buhmann, J.M.: Stability-based validation of clustering solutions. *Neural Computation* 16(6), 1299–1323 (2004)
38. Levine, E., Domany, E.: Resampling method for unsupervised estimation of cluster validity. *Neural Computation* 13, 2573–2593 (2001)
39. Mardia, J., Kent, K., Bibby, J.: *Multivariate Analysis*. Academic Press, San Diego (1979)
40. Milligan, G., Cooper, M.: An examination of procedures for determining the number of clusters in a data set. *Psychometrika* 50, 159–179 (1985)
41. Mufti, G.B., Bertrand, P., El Moubarki, L.: Determining the number of groups from measures of cluster validity. In: In Proceedigns of ASMDA 2005, pp. 404–414 (2005)
42. Pelleg, D., Moore, A.: X-means: Extending k-means with efficient estimation of the number of clusters. In: Proceedings of the 17th International Conf. on Machine Learning, pp. 727–734. Morgan Kaufmann, San Francisco (2000)
43. Rand, W.: Objective criteria for the evaluation of clustering methods. *Journal Am. Stat. Assoc.* 66, 846–850 (1971)
44. Stuetzle, W.: Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *J. Classification* 20(5), 25–47 (2003)
45. Sugar, C.A., James, G.M.: Finding the number of clusters in a dataset: An information-theoretic approach. *J. of the American Statistical Association* 98(463), 750–763 (2003)
46. Tibshirani, R., Walther, G.: Cluster validation by prediction strength. *Journal of Computational & Graphical Statistics* 14(3), 511–528 (2005)
47. Tibshirani, R., Walther, G., Hastie, T.: Estimating the number of clusters via the gap statistic. *J. Royal Statist. Soc. B* 63(2), 411–423 (2001)

48. Volkovich, Z., Barzily, Z.: On application of probability metrics in the cluster stability problem. In: 1st European Conference on Data Mining (ECDM 2007), Lisbon, Portugal, July 2007, pp. 5–7 (2007)
49. Volkovich, Z., Barzily, Z., Avros, R., Toledano-Kitay, D.: On application of the k -nearest neighbors approach for cluster validation. In: Proceeding of the XIII International Conference Applied Stochastic Models and Data Analysis (ASMDA 2009), Vilnius (2009)
50. Volkovich, Z., Barzily, Z., Morozensky, L.: A statistical model of cluster stability. Pattern Recognition 41(7), 2174–2188 (2008)
51. Wishart, D.: Mode analysis: A generalisation of nearest neighbour which reduces chaining effects. In: Numerical Taxonomy, pp. 282–311 (1969)

Chapter 7

Bregman Bubble Clustering: A Robust Framework for Mining Dense Clusters

Joydeep Ghosh¹ and Gunjan Gupta²

¹ Department of Electrical & Computer Engineering,
The University of Texas at Austin, Austin, TX 78712, USA
ghosh@ece.utexas.edu

² Microsoft, One Microsoft Way, Redmond, WA 98052, USA
gunjang@microsoft.com

Abstract. In classical clustering, each data point is assigned to at least one cluster. However, in many applications only a small subset of the available data is relevant for the problem and the rest needs to be ignored in order to obtain good clusters. Certain non-parametric density-based clustering methods find the most relevant data as multiple dense regions, but such methods are generally limited to low-dimensional data and do not scale well to large, high-dimensional datasets. Also, they use a specific notion of “distance”, typically Euclidean or Mahalanobis distance, which further limits their applicability. On the other hand, the recent One Class Information Bottleneck (OC-IB) method is fast and works on a large class of distortion measures known as Bregman Divergences, but can only find a *single* dense region. This paper presents a broad framework for finding k dense clusters while ignoring the rest of the data. It includes a seeding algorithm that can automatically determine a suitable value for k . When k is forced to 1, our method gives rise to an improved version of OC-IB with optimality guarantees. We provide a generative model that yields the proposed iterative algorithm for finding k dense regions as a special case. Our analysis reveals an interesting and novel connection between the problem of finding dense regions and exponential mixture models; a hard model corresponding to k exponential mixtures with a uniform background results in a set of k dense clusters. The proposed method describes a highly scalable algorithm for finding multiple dense regions that works with any Bregman Divergence, thus extending density based clustering to a variety of non-euclidean problems not addressable by earlier methods. We present empirical results on three artificial, two microarray and one text dataset to show the relevance and effectiveness of our methods.

1 Introduction

Clustering, which involves dividing data into groups of similar objects, is an important unsupervised learning problem that has been extensively applied in

various domains [36], and a variety of hierarchical [39, 38, 29, 32], partitional [3, 17, 51, 48, 8, 41], graphical [34, 50, 61, 32, 25, 41, 42] and overlapping [4, 6, 33, 2] clustering algorithms have been proposed and have found applications in a wide variety of domains such as identifying customer and product groups using market-basket data [71, 63, 31, 32, 25], document/text categorization [17, 18, 4, 72], and identifying functional groupings of genes and proteins in bioinformatics [38, 34, 29, 61, 6, 33].

In classical clustering, each data point either fully belongs to one cluster or is softly assigned to multiple clusters. However, in certain real-world problems, natural groupings are found among only a small subset of the data, while the rest of the data shows little or no clustering tendencies. In such situations it is often more important to cluster a small subset of the data very well, rather than optimizing a clustering criterion over all the data points, particularly in application scenarios where a large amount of noisy data is encountered.

For example, consider a large, high-dimensional transactional market-basket data that consists of product purchase records of a large number of customers of a retail chain, gathered over a considerable period of time. For a multitude of reasons, including rapid growth of the customer base and product offerings, rapid evolution of the product catalog, and customer churn/inactivity, such data can be very sparse, with the majority of the customers buying only a very small set of products from a catalog of thousands of items [49, 30]. Such a dataset can be used for clustering either products (with customers as features) or customers (with products as features). Clustering on such data is useful in many applications including product recommendations, customer and product segmentation, and identifying various customer and market trends. However, typically only a small subset of customers show statistically significant coherent buying behavior and that too when one focuses only a small subset of products [64, 14, 69]. Therefore, a clustering algorithm for such datasets should have the ability to prune out (potentially large) sparse and noisy portions of the data to uncover the highly coherent clusters. There are also non-algorithmic reasons for desiring such a capability, e.g. the marketing department of a retailer might want to target only a fraction of the customers and ignore the rest so as to maximize the ROI of their usually limited budgets. Another reason for desiring higher accuracy in such models at the cost of less coverage could be to minimize the chance targeting of a customer with the wrong product, which can negatively impact a customer's shopping experience.

As a second example, consider microarray datasets that record the relative expression levels of a few thousand genes across multiple experimental conditions. The conditions typically cover only a specific "theme" such as stress-response, and therefore only a few genes that are related to the conditions show good clustering. Biologists are interested in identifying small groups of genes that show strongly correlated expression patterns, as they indicate common participation in biological processes that are involved in the specific context. For example, for the Gasch dataset [21], which consists of only stress response experiments and is a popular benchmark for clustering microarray data, according to the authors, over 5,500 genes out of 6,151 genes are not directly involved in stress response.

These genes show insignificant change in expression level with respect to the control sample, and should be pruned in order to better identify and characterize the genes that are actually involved in specific types of stress responses. Similar characteristics have been observed and exploited in other microarray as well as protein mass spectroscopy and phylogenetic profile datasets [33, 37, 16, 53], where available features are often focused towards a few important contexts that are suitable for resolving only a small number of well-defined genetic pathways.

Finally, consider the grouping of documents according to their relevance to certain search queries. Certain documents are not relevant for any of the queries of interest. Moreover, the user is often interested in finding the top few matches for a broad-topic query rather than all possible matches, so that a system that returns a small number of highly relevant documents might be preferable over the one that returns hundreds of somewhat relevant documents, i.e., precision is more important than recall. By pruning out irrelevant or less relevant documents, precision can be improved without compromising much on recall [12].

One way to handle such scenarios is to prune the large fraction of “don’t care” data as a preprocessing or a post-processing step , and use existing “exhaustive” clustering methods. However, optimal preprocessing requires the knowledge of what subset would cluster well, which can only be defined well in the context of the clustering step. Post-processing is also not ideal since the optimization in exhaustive clustering is over the full dataset, and not on the relevant subset. A more natural approach would involve finding the multiple dense regions and the “don’t care” set simultaneously. Specifically, one desires clustering algorithms that are (1) scalable, (2) can cluster only a specifiable fraction of the whole dataset, (3) find multiple clusters, and (4) can work with a wide variety of data types. Existing density-based methods such as DBSCAN [20] naturally cluster only a subset of the data but are not suitable for many such situations because of implicit metric assumptions, and are not scalable to very large problems since they either require an in-memory $O(n^2)$ distance matrix, or an efficient index [7, 44]¹. In contrast, the One Class Information Bottleneck (OC-IB) [12] provides a local search based approach for finding a single dense region in the data that is fast and scalable to very high-dimensional datasets, and works with a large family of distortion measures known as *Bregman Divergences* (Section 2.1). However OC-IB can only find a single dense region, whereas in many problems dense regions can form multiple natural clusters. Furthermore, OC-IB can get stuck into a bad local minimum and does not allow control over the size of the cluster returned, which can vary greatly depending upon the quality of the local minimum. A subsequent technique called BBOCC enhanced the capabilities of the OC-IB type approach by providing the ability to control the size of the resultant cluster, as well as to use Pearson Correlation and Cosine similarity, in addition to Bregman Divergences [28]. This expanded the applicability of

¹ DBSCAN finds small dense regions of points by connecting nearest neighbor dense points. DBSCAN (and its derivatives) requires an efficient database index to be scalable to large data sets, since it uses the indexes to find the nearest neighbors. However, such indexes are usually efficient for only low-dimensional datasets.

BBOCC to many types of biological and textual clustering problems; however the limitation of identifying only a single cluster remained.

This paper substantially generalizes the single-cluster approach of BBOCC while retaining its key desirable properties, resulting in a robust and scalable framework for finding multiple dense clusters. Our main contributions are as follows:

1. We present a generalization of BBOCC called Bregman Bubble Clustering (BBC) that can simultaneously find k dense clusters. BBC inherits $O(nd)$ time and space complexity of BBOCC for each iteration and is scalable to much larger and higher-dimensional datasets than existing density-based methods. It also goes beyond Euclidean distance centric density-based clustering, and is applicable to all Bregman Divergences. This extension allows the method to be relevant for a very wide class of data properties (and corresponding loss functions) while retaining the simplicity of the squared-loss solution.
2. We develop a generative (soft) model consisting of a mixture of k exponentials and a uniform “background” distribution that leads to several insights into the problem of finding dense clusters using Bregman Divergences. BBC and many existing clustering algorithms are shown to be special cases of this model. Our main contribution here was to show how the seemingly distinct problem of finding dense clusters could be viewed as arising out of a generalization of the well-known mixture of exponential distributions model. This relationship also shows (1) how the problem setup of BBC is not just a convenient heuristic, but arises as a special (hard) case of a much more fundamental generative model, and (2) how BBC relates to the partitional clustering problem (that involves *all* data points) at a fundamental level.
3. We introduce a mechanism called *Pressurization* that substantially improves the quality of the local search in BBC and overcomes the problem of local minima, while keeping the time and space complexity at $O(nd)$. This is especially important for very large problems (e.g. clustering millions of customers in a market-basket dataset) where the deterministic seeding approach is too slow to apply against the full dataset. In empirical evaluations, Pressurization gives results that are robust to initialization and have very small variations in quality over multiple trials.
4. For medium-sized problems, we describe a deterministic seeding algorithm for BBC called Density Gradient Enumeration (DGRADE). At the cost of somewhat increased time and space complexity, DGRADE gives good empirical results when seeding BBC, and can determine k automatically. DGRADE uses a novel “density gradient estimation” at all the data points to identify all the distinct dense regions in the data, which then allows it to automatically estimate the best k , and the corresponding k cluster seeds. For many problems such as clustering gene-expression datasets where the number of relevant clusters in a dataset are often unknown initially and vary greatly, the more expensive time complexity of the seeding method as compared to Pressurization provides a useful trade-off; it provides a meaningful seeding

algorithm for BBC in a completely unsupervised setting. It also makes the BBC results deterministic, a desirable property for discovering deterministic albeit unknown biochemical pathways in an organism. Moreover, DGRADE can be used in conjunction with Pressurization for further improving clustering quality while also determining k .

5. We performed evaluations on a variety of datasets showing the effectiveness of our framework on low, medium and very high-dimensional problems, as compared to Bregman Clustering, Single Link Agglomerative and DBSCAN. We performed two types of experiments: (a) three artificial Gaussian datasets of 2, 10 and 40 dimensions were used to show the stability of observed results to the increasing dimensionality of the data, keeping the number and type of clusters relatively similar, and (b) pertinence to real-life applications was demonstrated using three different types of problems: using microarray data to cluster genes (medium size, high dimensional), clustering of conditions/experiments from microarray data (small, very high dimensional), and text clustering (large, very high dimensional).

A brief word on notation: bold faced variables, e.g. \mathbf{x} , represent vectors whose i^{th} element are accessed as either x_i or $x(i)$. Sets are represented by calligraphic upper-case alphabets such as \mathcal{X} and are enumerated as $\{\mathbf{x}_i\}_{i=1}^n$ where \mathbf{x}_i are the individual elements. $|\mathcal{X}|$ represents the size of set \mathcal{X} . Capital letters such as X are random variables. \mathbb{R} and \mathbb{R}^d represent the domain of real numbers and a d -dimensional vector space respectively. Bold-faced capital letters such as \mathbf{M}_D represent a two-dimensional matrix.

2 Background

We now describe some key concepts and related work that will be important in describing our methods.

2.1 Partitional Clustering Using Bregman Divergences

Bregman Divergences: *Bregman Divergences* form a family of distance measures, defined as follows: Let $\phi : S \mapsto \mathbb{R}$ be a strictly convex function defined on a convex set $S \subseteq \mathbb{R}^d$, such that ϕ is differentiable on $\text{int}(S)$, the interior of S . The Bregman Divergence $D_\phi : S \times \text{int}(S) \mapsto [0, \inf)$ is defined as:

$$D_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - (\mathbf{x} - \mathbf{y}, \nabla \phi(\mathbf{y})), \quad (1)$$

where $\nabla \phi$ is the gradient of ϕ .

For example, for $\phi(\mathbf{x}) = \|\mathbf{x}\|^2$, $D_\phi(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2$, which is the Squared Euclidean Distance. Similarly, other forms of ϕ lead to other popular divergences such as Logistic Loss, Itakura-Saito Distance, Hinge Loss, Mahalanobis Distance and KL Divergence [56, 3].

Bregman Information: An important property of all Bregman Divergences is as follows:

Theorem 2.1. [3]: Let X be a random variable taking values in $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset C \subseteq \mathbb{R}^d$ (C is convex) following a probability measure ν^2 , and let $E[\cdot]$ denote the expectation operator. Given a Bregman Divergence $D_\phi : C \times \text{int}(C) \mapsto [0, \infty)$, the problem

$$\min_{\mathbf{c} \in C} E_\nu[D_\phi(X, \mathbf{c})]$$

has a unique minimizer given by $\mathbf{c}^* = \mu = E_\nu[X]$.

[3] refer to the corresponding minimum $E_\nu[D_\phi(X, \mathbf{c}^*)]$ as the *Bregman Information* of X . Both variance and mutual information are special cases of Bregman Information. Theorem 2.1 essentially states that given any set of data points, the mean vector (or more generally, the *expectation* given a probability measure defined over the points) is the best single representative of the set in the sense of minimizing the average loss when each point gets replaced by a common representative. This result is well known for squared loss, but as per this Theorem it holds true for all Bregman Divergences. An immediate implication is that the k-means type algorithm will have the same guarantee of convergence to a local minima of the cost function for any Bregman Divergence. This result is also used in the BBC algorithm formulated later in Section 3.

Bregman Hard Clustering: [3] describe a partitional clustering algorithm called *Bregman Hard Clustering* that exploits Theorem 2.1. Starting with a random initialization of k centers Bregman Hard Clustering repeats the following until convergence to a local minimum: (1) assign each point to the closest center, as measured by the particular choice of D_ϕ , and (2) update the centers as the mean of points within each cluster. When D_ϕ is Squared Euclidean distance, Bregman Hard Clustering reduces to the K-Means algorithm, so one could view K-Means as a special case of Bregman Hard Clustering. An important result from [3] was to prove the bijection that a K-Means type algorithm exists for any Bregman Divergence, and only for Bregman Divergences. However, a subtle but perhaps more consequential property of Bregman Hard Clustering is that different choices of D_ϕ result in clustering algorithms that are appropriate for very different types of datasets and problems; many of the special forms had been proposed, proved and applied as independent algorithms, such as the celebrated Linde-Buzo-Gray algorithm [48,8], before Bregman Hard Clustering was formulated.

2.2 Density-Based and Mode Seeking Approaches to Clustering

A variety of non-parametric density-based methods have been developed that use different notions of “local” density to cluster only a part of the data and to prune the rest. The classic work in this area is Wishart’s mode analysis [70], which is closely related to the more recent DBSCAN algorithm [20]. Other notable works include the application of mean-shift algorithm to clustering [11,22].

² Unless stated explicitly otherwise, we assume all points to have the same weight, i.e., ν is a uniform measure.

The mean-shift algorithm performs (adaptive) gradient ascent on the estimated density of the data, as obtained by convolving a suitable localized kernel function with the raw data, to find modes or local peaks of the density. If only modes that are sufficiently dominant are selected, then points attracted to less important modes could be discarded. By varying the widths of the kernels, one can investigate clustering behavior at different scales [10]. DBSCAN has a slightly different flavor: given a point that has at least $MinPts$ points enclosed by a hypersphere of radius ϵ centered at the point, all points within the ϵ sphere are assigned to the same cluster. DBSCAN has the ability to find arbitrary shaped clusters, which is useful in certain problems. However, different choices for ϵ and $MinPts$ can give dramatically different clusterings. OPTICS [1] proposed a visualization to make it easier to select these two parameters. Like other mode-seeking algorithms, DBSCAN is computationally efficient only for low-d spatial data where efficient indexing schemes are available, and is therefore popular in the database community for indexing 2-d and 3-d images.

DHC [38] is perhaps the first published work on applying density-based clustering to biological data. It proposes a density-based hierarchical clustering algorithm for time-series data. DHC provides a hierarchical grouping of time-series data that can be used to visually browse similar genes. The cluster hierarchy built by DHC uses the heuristic of *attraction* that assumes the data is uniformly distributed in a d -dimensional space. However, points in many real-life high dimensional datasets tend to reside in much lower dimensional manifolds [67] within the embedded space.

We have recently proposed a non-parametric approach, Auto-HDS [29] for detecting a few dense clusters in data. Inspired by Wishart's work but much more computationally efficient, Auto-HDS simultaneously detects clusters at multiple resolutions and provides a powerful visualization mechanism for cluster exploration. It also provides much superior results as compared to DBSCAN. Since this paper focuses on parametric approaches, we do not discuss Auto-HDS further, but point the interested reader to Chapter 11 of [26], which provides a detailed theoretical as well as empirical comparison of Auto-HDS with BBC. In summary, BBC is more scalable than Auto-HDS and other non-parametric approaches; and works better when one has a fairly good generative model of the data. However, if one has little idea about the nature of the data, or if the data has very odd-shaped dense regions at different resolutions, the additional flexibility of a non-parametric approach is helpful.

A parametric approach wherein a mixture of Gaussians plus a uniform background component is fitted to data in order to detect peaks was recently presented in [57]. The current approach and accompanying software is specifically for detecting peaks in one-dimensional data, with additional constraints such as no other peak allowed within a certain distance to the left or right of a given peak. This constrained one-dimensional setting is designed for specific applications such as detecting transcription start sites from gene annotation data. If one properly generalizes this approach to multivariate data and to all exponential

family mixture models, one will obtain the soft BBC model, which the new method proposed in this paper compares favorably against (see Section 10).

2.3 Iterative Relocation Algorithms for Finding a Single Dense Region

Traditional density-based clustering algorithms (Section 2.2) were aimed at low-dimensional, spatial datasets. However, they have two major shortcomings for broader clustering applications: (1) they typically rely on a Euclidean distance type metric to determine “distance”, even though such measures are not suitable for many datasets and (2) they do not scale well to large, higher-dimensional datasets. A recently proposed algorithm for finding a *single* dense region³ called One Class Information Bottleneck (OC-IB) [12] breaks these two barriers by proposing an iterative relocation based approach that is also generalizable to all Bregman Divergences, and whose scaling properties are akin to K-Means even though K-Means itself is not designed for finding dense clusters.

OC-IB uses the notion of a Bregmanian ball to find a single, locally dense region. Earlier approaches to One Class Clustering [66, 58, 59, 13] used convex cost functions for finding large-scale structures, or correspondingly, for finding a small number of outliers. However, [12] showed that such methods are not appropriate when we want to find distinct dense regions covering only a small fraction of the data. For example, suppose the data is generated by two low-variance Gaussians embedded within a relatively uniform background. Previously proposed convex One Class methods end up finding a solution centered in-between the two Gaussians. In contrast, OC-IB is able to find one of the two Gaussians, and could be applied sequentially to recover both. More discussion and evidence on this important conceptual difference between the two One Class approaches and why the local approach used by OC-IB is more relevant for finding dense regions can be found in [12].

In an earlier paper [27], we described an algorithm called Batch Ball One Class Clustering (BBOCC) that provides several improvements over OC-IB, including the ability to control the size (number of data points) of the dense cluster, improved quality of local search, optimality guarantee using seeding⁴ and extension to Pearson Correlation. However, BBOCC can also only find a single dense region. An obvious solution that comes to mind is to apply OC-IB or BBOCC sequentially: removing points belonging to the first dense cluster and then running the One Class algorithm again on the reduced data. Unfortunately, unless a correspondence problem is solved, this could result in a “cookie-cutter” clustering where additional clusters found are comprised of left-over dense points

³ A problem that is also referred to as One Class Classification or Clustering.

⁴ Guarantees that the solution is within two times of lowest possible cost (as given by Equation 2, only applicable for $k = 1$, since k is always 1 for BBOCC), when the Bregman Divergence was Squared Euclidean, and is constant times optimal for other Bregman Divergences. See [28] for more details.

surrounding the hole created by the removal of the first dense cluster discovered. This limitation was also hinted upon in the conclusion section of [12]⁵.

This paper addresses the problem of simultaneously finding k dense regions in the data while ignoring a specified fraction of the data-points. While the approach taken is not a straightforward generalization of BBOCC and entails several new concepts, familiarity with BBOCC [27] will provide an enriched understanding of this paper.

2.4 Clustering a Subset of Data into Multiple Overlapping Clusters

In the context of clustering microarray data, discovering overlapping gene clusters is popular since many genes participate in multiple biological processes. Gene Shaving [33] uses PCA to find a small subset of genes that show strong expression change compared to the control sample, and allows them to be in multiple clusters. As we mentioned earlier, since only a small fraction of the genes are relevant for clustering in a given dataset, the ability of Gene Shaving to prune a large fraction of genes is particularly attractive. However, Gene Shaving greedily extracts one cluster at a time, and is computationally very expensive ($\Omega(n^3)$). Other greedy methods such as Plaid [46] treat the original data \mathcal{X} as a matrix, and decompose it into a set of sub-matrices that when added together reconstruct \mathcal{X} . This allows Plaid to also find overlapping clusters. However, matrix approximation methods for gene-expression datasets have only had partial success, in large part due to the highly unbalanced nature of the matrix; there are typically to the order of 10^2 (biological experiment) conditions while there are to the order of 10^4 genes. A critical issue therefore continues to be the ability to select a small number of highly relevant genes for clustering, while selecting all the conditions as relevant.

3 Bregman Bubble Clustering

In this section we first generalize the notion of a single dense *Bregmanian Ball* (used by One Class algorithms OC-IB and BBOCC) to the idea of multiple dense regions called *Bregman Bubbles*. We then present an algorithm called *Bregman Bubble Clustering* or BBC, that can find k dense Bregman bubbles using a local search approach.

3.1 Cost Function

Let $\mathcal{X} = \{\mathbf{x}\}_{i=1}^n \subset C \subseteq \mathbb{R}^d$ (where C is convex) be the set of data points. Let $\mathcal{G} \subset \mathcal{X}$ represent a non-exhaustive clustering consisting of k clusters $\{\mathcal{C}_j\}_{j=1}^k$ with $\mathcal{X} \setminus \mathcal{G}$ points that are “don’t care”, i.e., they do not belong to any cluster. For a given Bregman Divergence $D_\phi(\mathbf{x}, \mathbf{y}) \mapsto [0, \infty)$, and a set of k cluster

⁵ Interestingly, the seeding algorithm DGRADE presented in this paper in Section 9, solves exactly this correspondence problem by identifying all the distinct “basins of attraction” corresponding to the densest Bregmanian balls in the data.

representatives $\{\mathbf{c}_j\}_{j=1}^k \in \mathbb{R}^d$ for the k clusters in clustering $\mathcal{G} = \{\mathcal{C}_j\}_{j=1}^k$, we define the cost Q_b as the average distance of all points in \mathcal{G} from their assigned cluster representative:

$$Q_b(\mathcal{G}, \{\mathbf{c}_j\}_{j=1}^k) = \frac{1}{|\mathcal{G}|} \sum_{j=1}^k \sum_{i: \mathbf{x}_i \in \mathcal{C}_j}^{|\mathcal{C}_j|} D_\phi(\mathbf{x}_i, \mathbf{c}_j), \quad (2)$$

3.2 Problem Definition

Given s , k and D_ϕ as inputs, where s out of n points from \mathcal{X} are to be clustered into a clustering $\mathcal{G} \subseteq \mathcal{X}$ consisting of k clusters, where $1 \leq k < n$ and $k \leq s \leq n$, we define the clustering problem as:

Definition 1: Find the clustering \mathcal{G} with smallest cost Q_b such that $|\mathcal{G}| = s$.

Definition 1 builds upon the cost formulation stated in 3.1, where the cost contributed by each point in each cluster is proportional to the distance of the member points from their cluster centroid, with an additional constraint that exactly s points are clustered. For $k = 1$, this problem definition reduces to one used in one of the two form of BBOCC for finding a single dense cluster⁶.

3.3 Bregmanian Balls and Bregman Bubbles

A *Bregmanian ball* [12] $B_\phi(r, \mathbf{c})$ with radius r and centroid \mathbf{c} defines a volume in \mathbb{R}^d such that all points \mathbf{x} where $D_\phi(\mathbf{x}, \mathbf{c}) \leq r$ are enclosed by the ball. Given a set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ of n points in \mathbb{R}^d , the cost of the ball is defined as the average $D_\phi(\mathbf{x}, \mathbf{c})$ of all points enclosed by it.

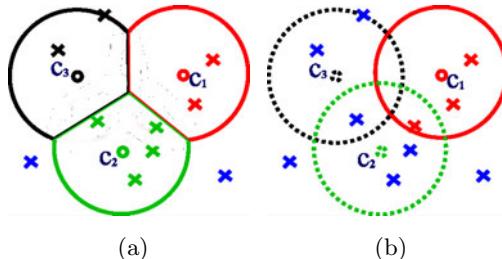


Fig. 1. An illustration showing (a) three Bregman bubbles, and (b) a Bregmanian ball (solid line), and two other possible balls (dotted lines). The union of the points enclosed in (b) is the same as the set of points enclosed by the three bubbles.

For a specified set of k cluster representatives, and a fixed s , it can be shown using Theorem 2.1⁷ that the clustering that minimizes Q consists of: (1) the

⁶ Section 6 discusses the connection with BBOCC in more detail.

⁷ A more formal proof is presented after Proposition 3.1.

assignment phase, where each point is assigned to the nearest cluster representative, and (2) picking points closest to their representatives first until s points are picked. Let r_{max} represent the distance of the last (s^{th}) picked point from its cluster representative.

These clusters can be viewed as k *Bregman bubbles* such that they are either (1) pure Bregmanian balls of radius $r \leq r_{max}$, or (2) *touching* bubbles that form when two or more Bregmanian balls, each of radius r_{max} overlap. Two Bregmanian balls $B_\phi(\mathbf{c1}, r_1)$ and $B_\phi(\mathbf{c2}, r_2)$ are said to overlap when $\exists \mathbf{x} : (D_\phi(\mathbf{x}, \mathbf{c1}) < r_1) \wedge (D_\phi(\mathbf{x}, \mathbf{c2}) < r_2)$. At the point of contact, the touching bubbles form linear boundaries⁸ that result from assigning points to the closest cluster representative. For the part of its boundary where a bubble does not touch any other bubble, it traces the contour of a Bregmanian ball of radius r_{max} . Therefore, bubbles arise naturally as the optimum solution for Q_b for a given s , k and D_ϕ .

Figure 1 illustrates a 2-D example of Bregman bubbles vs. balls. Unlike Bregmanian balls, the boundary of the Bregman bubbles can only be defined in the context of other bubbles touching it. It is important to note that the volume of the convex hull of points in one bubble could be smaller than that of the adjacent touching bubble, and the bubbles could also have different number of points assigned to them.

3.4 BBC-S: Bregman Bubble Clustering with Fixed Clustering Size

For most real life problems, even for a small s , finding the globally optimal solution for problem definition 1 would be too slow. However, a fast iterative relocation algorithm that guarantees a local minimum exists. *Bregman Bubble Clustering-S* (BBC-S, Algorithm 1) starts with k centers and a size s as input. Conceptually, it consists of three stages: (1) the assignment phase, where each point is assigned to the nearest cluster representative, (2) the selection phase, where points are selected in ascending order of their distance from their corresponding cluster representative, until s points are picked, and (3) the update step, where cluster “means” are re-estimated and updated. It is interesting to note that stages 1 and 3 of BBC-S are identical to the Assignment Step and the Re-estimation step of the Bregman Hard Clustering (Section 2.1), properties that lead to the unification described in Section 6. Stages 1, 2 and 3 are repeated until there is no change in assignment between two iterations - i.e. the algorithm converges. Algorithm 1 describes a more detailed implementation of BBC-S where line number 11 represents Stage 1, lines 16 to 20 map to Stage 2, while lines 26-28 represent Stage 3. We randomly pick k data points from \mathcal{X} as the starting cluster representatives, but alternative initialization schemes could be implemented.

Proposition 3.1. *Algorithm 1 terminates in a finite number of steps at a locally optimal solution, i.e., the cost function Q_b cannot be decreased by (a)the assignment step, (b) the data selection step or (c)changing the means of any existing clusters.*

⁸ This can be shown to be true for all Bregman Divergences [3].

Algorithm 1. BBC-S

Input: Set $\mathcal{X} = \{\mathbf{x}\}_{i=1}^n \subset C \subseteq \mathbb{R}^d$, Bregman Divergence D_ϕ , no. of clusters k , desired clustering size s , k seed centroids for the k clusters (optional).

Output: Partitioning \mathcal{G}^* containing k clusters $\{\mathcal{C}_j\}_{j=1}^k$, and the corresponding k cluster representatives $\{\mathbf{c}_j^*\}_{j=1}^k$.

Method:

```

if  $\{\mathbf{c}_j\}_{j=1}^k = \emptyset$  then
    5:   Initialize cluster representatives  $\{\mathbf{c}_j\}_{j=1}^k$  with seed centroids if available, else
         randomly.
    end if
     $\mathcal{G}^l = \emptyset; \mathcal{G} = \emptyset; q = \infty; q_p = \infty;$ 
    repeat
        /*Assign each point to the closest of the  $k$  centroids*/
    10:  for  $i = 1$  to  $n$  do
             $[d_i^{min}, lab_i] = \min_{j=1}^k (D_\phi(\mathbf{x}_i, \mathbf{c}_j))$ 
        end for
        /* Find the  $s$  points closest to their centroids and form the cluster*/
        [val, idx] = sort( $\mathbf{d}^{min}$ )
    15:   $q^{tmp} = 0; s^c = 0; \{\mathcal{C}_j\}_{j=1}^k = \emptyset$ 
        while ( $s^c < s$ ) do
             $s^c = s^c + 1;$ 
             $q^{tmp} = q^{tmp} + val(s^c)$ 
            Add  $\mathbf{x}_{idx(s^c)}$  to cluster  $\mathcal{C}_{lab(idx(s^c))}$ 
    20: end while
        /* Save previous centroids, clustering costs, cluster memberships and update to
            new ones */
         $\{\mathbf{c}_j^p\}_{j=1}^k = \{\mathbf{c}_j\}_{j=1}^k$ 
         $q^{pp} = q; q^p = q; q = q^{tmp}/s$ 
         $\mathcal{G}^l = \mathcal{G}; \mathcal{G} = \{\mathcal{C}_j\}_{j=1}^k$ 
    25: /* Recompute cluster centroids based on the new cluster memberships */
        for  $j = 1$  to  $k$  do
             $\mathbf{c}_j = \frac{1}{|\mathcal{C}_j|} \sum_{i: \mathbf{x}_i \in \mathcal{C}_j} \mathbf{x}_i$ 
        end for
        until ( $\mathcal{G}^l == \mathcal{G}$ )  $\wedge$   $q_{pp} == q$  /* Convergence */
    30: Return  $\{\mathbf{c}_j^*\}_{j=1}^k = \{\mathbf{c}_j\}_{j=1}^k; \mathcal{G}^* = \mathcal{G}$ 

```

Proof: The local optimality of steps (a) and (c) has been established in [3] Prop. 3, and can be summarized as follows: local optimality of step (a) can be readily shown by the contradiction that if a point is not assigned to the nearest centroid, then the total cost Q_b can be decreased by assigning it to a closer centroid. Theorem 2.1 guarantees that step (c) is locally optimal; a representative other than the mean would lead to a higher cost for a given cluster. Local optimality of step (b) can also be shown by contradiction - if a point x_p that is not among the first s points (in the sorted order at line 14 of the algorithm) was part of the optimal solution, then the cost Q_b could be decreased by replacing this point with a point within the first s points that is not picked. Thus no such x_p can be

part of the best solution at step (b). So the algorithm monotonically decreases the objective function value, while the number of distinct clusterings is finite, thus assuring convergence in a finite number of steps.

If heap-sort is used at line 14 of Algorithm 1, then each iteration of BBC-S takes

$O(\max(nkd, s \log(n)))$ time, making it quite fast.

3.5 BBC-Q: Dual Formulation of Bregman Bubble Clustering with Fixed Cost

An alternative “dual” formulation of the Bregman Bubble Clustering called BBC-Q is possible where a threshold cost q_{\max} is specified as input rather than the size s . Given q_{\max} , k and D_ϕ as inputs:

Definition 2: Find the largest \mathcal{G} with cost $Q_b \leq q_{\max}$.

We can show that this definition also results in Bregman bubbles as the optimal solution for a set of k cluster representatives. Definitions 1 and 2 are equivalent, since for a given q_{\max} there exists a largest s for k bubbles, and for the same s , the same solution has the same smallest possible cost q_{\max} . Algorithm 1 can be easily modified to work with q_{\max} by modifying Stage (2) to stop adding points when the cost is more than q_{\max} . The proof of convergence for BBC-Q follows along similar lines as that for Proposition 3.1.

The seemingly minor difference between BBC-S and BBC-Q results in two very different algorithms. For a fixed s as input (BBC-S), for iterations in sparse regions the bubbles expand until s points are covered. As the bubbles move into denser regions, their radii shrink. BBC-Q does not have this property and generally gives worse performance when the bubbles are small [28]. Unless stated explicitly otherwise, the discussion on Bregman Bubble Clustering in the rest of the paper is restricted to BBC-S, i.e. BBC with a fixed s as input.

4 Soft Bregman Bubble Clustering (Soft BBC)

4.1 Bregman Soft Clustering

In hard clustering, each point is assigned to one cluster. In *soft clustering*, each point can be a “partial” member of all of the clusters. If the sum of the assignment weights of a given point to all clusters is normalized to 1, we can interpret the soft assignments as probabilities. One popular way to model such probabilistic assignments is to assume that the set of observed points come from a mixture of k distributions whose parameters are estimated based on the observed data. Once the parameters are estimated, the probabilistic membership of each point to each of the clusters can be computed. [3] proposed a soft clustering algorithm called *Bregman Soft Clustering* as a mixture model consisting of k distributions, taken from the family of *regular exponential distributions*, and showed that there

is a *bijection* between this family and regular Bregman Divergences. This bijection is expressed by:

$$p_{(\psi, \theta)}(\mathbf{x}_s) = \exp(-\beta D_\phi(\mathbf{x}_s, \mu)) f_\phi(\mathbf{x}_s) \quad (3)$$

where ϕ is a convex function, and the conjugate function of ψ , D_ϕ is the corresponding Bregman Divergence, $p_{(\psi, \theta)}$ is the corresponding regular exponential distribution with cumulant ψ , f_ϕ is a uniquely determined normalizing function that depends on the choice of ϕ , β is a scaling factor, μ is the expectation parameter, θ are the natural parameters of p^ϕ , and \mathbf{x}_s is the sufficient statistics vector corresponding to \mathbf{x} .

Well-known examples of regular Bregman Divergences (and the corresponding exponential distribution) include squared Euclidean Distance (Gaussian distribution), KL-divergence (multinomial distribution) and Itakura-Saito distance [48, 8].

[3] not only showed a formal unification of the various hard partitional clustering methods as special cases of Bregman hard clustering, and the corresponding exponential distribution soft clustering models as special cases of Bregman soft clustering, but went on to show that for all regular Bregman divergences, Bregman Hard Clustering falls out as a special case of Bregman Soft Clustering. For example, for the Squared Euclidean distance as D_ϕ , Bregman Hard Clustering maps to the standard K-Means algorithm, and the corresponding Bregman Soft Clustering maps to a mixture of spherical Gaussians with a fixed variance σ^2 , popularly known as *soft K-Means*, and μ maps to Gaussian mean \mathbf{a} , $f_\phi(\mathbf{x}_s) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}}$, $\beta = \frac{1}{2\sigma^2}$, $D_\phi(\mathbf{x}_s, \mu) = \beta \|\mathbf{x} - \mathbf{a}\|^2$, $\theta = \frac{\mathbf{a}}{\sigma^2}$, and $\psi(\theta) = \frac{\sigma^2}{2} \|\theta\|^2$. The soft K-Means model reduces to K-Means when the variance σ^2 of the k Gaussians is set to 0^+ that corresponds to $\beta \rightarrow \infty$ in Equation 3.

4.2 Motivations for Developing Soft BBC

Bregman Bubble Clustering can be thought of as a non-exhaustive hard clustering where points can belong to either one of the k clusters or to a “don’t care” group, while there is no such “don’t care” grouping in Bregman Hard Clustering. The generative model for Bregman Soft Clustering consists of a mixture of k regular exponential distributions of the form p^ϕ corresponding to the k clusters. Correspondingly, *Soft Bregman Bubble Clustering* (Soft BBC) can be formulated as modeling the data as a mixture of k distributions from the exponential family and an additional “background” distribution corresponding to the “don’t care” points. Since we are trying to find k dense clusters, for a good solution the “don’t care” group should be the least dense. One way to model this low density background is with a uniform distribution. The goal of building such a Soft BBC model is to give us deeper insights into the implicit modeling assumptions behind BBC.

4.3 Generative Model

Let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ be the dataset consisting of n i.i.d. points and k be the desired number of clusters. We propose Soft BBC as a generative model containing k mixture components corresponding to k dense clusters labeled 1 to k and one uniform background distribution labeled 0, where each data point is assumed to be generated by a unique but *unknown* component. Let $\mathcal{Y} = \{Y_i\}_{i=1}^n$ be the hidden random variables corresponding to the mixture components associated with the data points, where Y_i can take one of $k + 1$ possible values from 0 to k . In the absence of any other information, the distribution of \mathcal{Y} only depends upon the priors. Hence the model probability of the data points is given by:

$$p(\mathbf{x}_i) = \sum_{j=1}^k \alpha_j p_{(\psi, \theta)}(\mathbf{x}_i | \theta_j) + \alpha_0 p_0, [i]_1^n \quad (4)$$

where $\{\alpha_j\}_{j=1}^k$ and $\{p_{(\psi, \theta)}(\cdot | \theta_j)\}_{j=1}^k$ denote the priors and the conditional distributions of the k clusters, while α_0 and p_0 denotes the prior probability and the probability density of the uniform distribution. Since the data points are assumed to be i.i.d., the log-likelihood of the observed data (or the incomplete log-likelihood) is given by:

$$L(\Theta | \mathcal{X}) = \sum_{i=1}^n \log \left(\sum_{j=1}^k \alpha_j p_{(\psi, \theta)}(\mathbf{x}_i | \theta_j) + \alpha_0 p_0 \right) \quad (5)$$

where Θ denotes all the parameters (priors and mixture component parameters). Maximizing the above data likelihood is a natural approach for fitting this generative model to the data. However, it is non-trivial to directly optimize the likelihood function due to the presence of mixture components.

4.4 Soft BBC EM Algorithm

Since p_0 is a uniform distribution by definition, $1/p_0$ defines the volume of its domain. This domain should include the convex hull of \mathcal{X} , which yields an upper bound for p_0 . In Equation 5, keeping all other parameters constant, a lower value of p_0 will always result in a lower likelihood. For now, we only consider the case where p_0 is set to a fixed value. Therefore, the only parameters we can optimize over are the priors $\{\alpha_j\}_{j=0}^k$ and the exponential mixture parameters $\{\theta_j\}_{j=1}^k$. We consider two slightly different scenarios: (A) where α_0 is a variable parameter, and (B) where α_0 is a fixed value ≤ 1 . To maximize the log-likelihood function, we adopt a standard EM-based approach and first construct the negative free energy function [55]:

$$F(\tilde{P}, \Theta) = \sum_{i=1}^n E_{\tilde{p}(Y_i, \mathbf{x}_i)} [\log p(\mathbf{x}_i, Y_i | \Theta)] - \sum_{i=1}^n E_{\tilde{p}(Y_i, \mathbf{x}_i)} [\log p(Y_i | \mathbf{x}_i)]$$

where $\tilde{P} = \{\{\tilde{p}(Y_i = j | \mathbf{x}_i)\}_{i=1}^n\}_{j=1}^k$ are the current estimates of \mathcal{Y} . It can be shown that the EM procedure with the **E** and **M** steps alternately optimizing

Algorithm 2. Soft BBC

Input: Set $\mathcal{X} = \{\mathbf{x}\}_{i=1}^n \subset C \subseteq \mathbb{R}^d$, Bregman Divergence D_ϕ , no. of clusters k , p_0 , specifying the background distribution, α_0 for Case B.

Output: Θ^* , local maximizer of $L(\Theta|\mathcal{X})$ (Equation 5) where $\Theta = \{\{\theta_j, \alpha_j\}_{j=1}^k, \alpha_0\}$ for Case A and $\{\theta_j, \alpha_j\}_{j=1}^k$ for Case B, soft partitioning $\{\{p(Y_i = j|\mathbf{x}_i)\}_{j=0}^k\}_{i=1}^n$.

Method:

Initialize p_0 , $\{\theta_j, \alpha_j\}_{j=1}^k$ with some $0 \leq p_0 < 1$, $\theta_j \in C$, $\alpha_j \geq 0$, such that $\sum_{j=0}^k \alpha_j = 1$.

repeat

{The E Step}

for $i = 1$ to n **do**

for $j = 0$ to k **do**

$p(Y_i = j|\mathbf{x}_i)$ is computed from equations 6 and 7, where $p_{(\psi, \theta)}(\mathbf{x}_i|\theta_j)$ is defined by equation 3.

end for

end for

{The M Step}

for $j = 0$ to k **do**

Update α_j using equation 8 for Case A and 11 for Case B.

Update θ_j using equation 10.

end for

until convergence

$F(\tilde{P}, \Theta)$ over \tilde{P} and Θ is guaranteed to converge to a local maximum \tilde{P}^* and Θ^* . Furthermore, it can be shown that a local maximum of $F(\tilde{P}, \Theta)$ leads to a local maximum on the original likelihood given by Equation 5. Hence we will now focus on obtaining the updates involved in the **E** and **M** steps for the two cases.

Case A: α_0 is not fixed

E-Step: In this step we optimize $F(\tilde{P}, \Theta)$ (Equation 6) over \tilde{P} under the constraints that the $\sum_{j=0}^k \tilde{p}(Y_i = j|\mathbf{x}_i) = 1$, $[i]_1^n$, and $\tilde{p}(Y_i = j|\mathbf{x}_i) \geq 0, \forall i, j$. Using Lagrange multipliers for the n equality constraints, taking derivatives w.r.t. $\tilde{p}(Y_i = j|\mathbf{x}_i)$, and then eliminating the Lagrange multipliers, we obtain:

$$\tilde{p}(Y_i = j|\mathbf{x}_i)^* = \frac{\alpha_j p_{(\psi, \theta)}(\mathbf{x}_i|\theta_j)}{\sum_{j=1}^k \alpha_j p_{(\psi, \theta)}(\mathbf{x}_i|\theta_j) + \alpha_0 p_0}, 1 \leq j \leq k \quad (6)$$

$$= \frac{\alpha_0 p_0}{\sum_{j=1}^k \alpha_j p_{(\psi, \theta)}(\mathbf{x}_i|\theta_j) + \alpha_0 p_0}, j = 0 \quad (7)$$

M-Step: In this step we optimize $F(\tilde{P}, \Theta)$ over Θ under constraints $\sum_{j=0}^k \alpha_j = 1$ and $\alpha_j \geq 0, \forall j$. It can be shown that the inequality constraints are not binding. On applying the standard Lagrange procedure, one obtains:

$$\alpha_j^* = \frac{\sum_{i=1}^n \tilde{p}(Y_i = j|\mathbf{x}_i)}{n}, [j]_0^k \quad (8)$$

Note that the update equation for the background distribution prior, α_0 , turns out to be the same as that for the exponential mixture distributions α_1 to α_k . The optimal mixture component parameter estimation can be obtained by setting derivatives over $\{\theta_j\}_{j=1}^n$ to 0 as follows:

$$\sum_{i=1}^n \tilde{p}(Y_i = j | \mathbf{x}_i) \nabla_{\theta_j} p_{(\psi, \theta)}(\mathbf{x}_i | \theta_j) = 0 \quad (9)$$

This results in the update equation for the exponential distribution mixtures $\{\theta\}_{j=1}^k$ as the weighted average of \mathbf{x} [3]:

$$\theta_j = \frac{\sum_{i=1}^n p(Y_i = j | \mathbf{x}_i) \mathbf{x}_i}{\sum_{i=1}^n p(Y_i = j | \mathbf{x}_i)} \quad (10)$$

An example of re-estimation of mixture component parameters for Gaussians is described in more detail in Section 7.

Case B: α_0 is fixed

E-Step: Since keeping α_0 fixed does not result in any additional constraints, this step is identical to that of Case A.

M-Step: Keeping α_0 constant modifies the constraints on the priors so that we now require $\sum_{j=1}^k \alpha_j = 1 - \alpha_0$ and $\alpha_j \geq 0, \forall j$. As before, the inequality constraints are not binding and by using a Lagrange multiplier and taking derivatives, we arrive at:

$$\alpha_j^* = (1 - \alpha_0) \frac{\sum_{i=1}^n \tilde{p}(Y_i = j | \mathbf{x}_i)}{\sum_{j=1}^k \sum_{i=1}^n \tilde{p}(Y_i = j | \mathbf{x}_i)} \quad (11)$$

The optimal mixture component parameters are obtained exactly as in Case A.

4.5 Choosing an Appropriate p_0

For Case A of the Soft BBC algorithm, one can argue that the parameter α_0 is essentially a function of p_0 given by the relation (from the M step):

$$\alpha_0 = \frac{1}{n} \sum_{i=1}^n \frac{\alpha_0 p_0}{\sum_{j=1}^k \alpha_j p_{(\psi, \theta)}(\mathbf{x}_i | \theta_j) + \alpha_0 p_0} \quad (12)$$

Using this relation, for a given α_0 and a set of mixture component parameters, it is possible to solve for p_0 . But one cannot do this in the EM framework since the best value for p_0 is always the highest possible one. However this relationship allows us to calculate the value of p_0 for the initial seed parameters. For a given value of α_0 , one approach would be to rewrite Equation 12 as an optimization problem and solve for the best value of p_0 :

$$f(p_0) = \alpha_0 - \frac{1}{n} \sum_{i=1}^n \frac{\alpha_0 p_0}{\sum_{j=1}^k \alpha_j p_{(\psi, \theta)}(\mathbf{x}_i | \theta_j) + \alpha_0 p_0} = 0 \quad (13)$$

Written in this form, one could now start with a seed value between 0 and 1, and then search for the value of p_0 that brings $f(p_0)$ closest to 0. An optimization routine such as Matlab *fsolve* (<http://www.mathworks.com>) could be used for this process. However, a faster approximation of p_0 can be obtained as follows:

1. Perform the first E step (equations 6 and 7).
2. Compute the $p_{max}^i = \max_{j=0}^k (p(Y_i = j | \mathbf{x}_i))$ for each \mathbf{x}_i .
3. Pick p_0 as the s^{th} largest value in $p_{max}^i[i]_1^n$ where $s = \lceil \alpha_0 n \rceil$.

The above formulation works well because of the following reason: the final soft BBC results are probabilistic, with each point having a probability of belonging to either one of the k clusters, or the background. The probabilities need to be converted into hard assignment to obtain clustering needed in many applications. Later in Section 6, we show that the natural hard assignment corresponds to assigning each point to the mixture with the maximum posterior probability (which could be either of the k clusters or the uniform background). In other words, selecting label j such that $p(Y_i = j | \mathbf{x}_i) = p_{max}^i$ using Step 2 above. For the hard assignment case, since the fraction of data points clustered is s , and since $s = \lceil \alpha_0 n \rceil$, picking p_0 equal to (or in theory, slightly larger than) the s^{th} largest value in $p_{max}^i[i]_1^n$ would result in very close to s points getting assigned to the clusters, while the remaining, having a $p_{max}^i \leq p_0$ get assigned to the background cluster. In practice, the value of p_0 obtained using this approach corresponds closely with the value computed using the more expensive optimization type approach.

The following enhancement works even better in practice: an initial estimate of p_0 is computed using the approach described above using the seed cluster parameters to Soft BBC ($\{\theta_j\}_{j=1}^k$ and $\{\alpha_j\}_{j=1}^k$). This initial value of p_0 is then used to run Soft BBC to convergence. The clustering parameters obtained at convergence are then used to compute p_0 again, and Soft BBC is then run to convergence a second time using the new p_0 . This second p_0 estimate is better since it is based on parameters that are closer to the convergence values.

5 Improving Local Search: Pressurization

5.1 Bregman Bubble Pressure

We first introduce a concept called *Bregman bubble pressure* that has properties analogous to that of pressure around air bubbles rising from a seabed; as air bubbles rise in a column of water, the hydrostatic pressure outside drops, and the bubbles expand (see [40], Chapter 10, Section 10.4.4 for a description of this phenomena).

In the case of the generative or Soft BBC model, we can think of this external pressure as being driven by the relative weight of the background distribution, α_0 in Equation 4. Bregman bubble pressure can be seen as being proportional to the ratio of the background distribution weight vs. the weight of all the k bubbles combined, i.e. $\alpha_0 / \sum_{j=1}^k \alpha_j$, which is equal to $\frac{\alpha_0}{1-\alpha_0}$, since $\sum_{j=0}^k \alpha_j = 1$.

As α_0 tends to 1, the Bregman bubble pressure, being proportional to $\frac{\alpha_0}{1-\alpha_0}$, tends to infinity, causing the extent of the regions around the k exponential distribution centroids, where the corresponding distributions have higher weight than the background distribution in Equation 4, to shrink towards 0 (since the weights of the exponential distributions all get forced to 0). For the hard BBC, the increasing weight of α_0 corresponds to the number of points clustered, s , tending towards 0. The Bregman bubble pressure can also be thought of as being proportional to $(n - s)/s$, where s is the number of data points clustered, and is an input to the hard BBC algorithm (Algorithm 1).

Conversely, when α_0 tends to 0, the background pressure (proportional to $\frac{\alpha_0}{1-\alpha_0}$) tends to 0, and the Soft BBC model gives rise to Bregman Soft Clustering, where there is no background distribution. This also corresponds to the hard BBC model reducing to Bregman Clustering, where $s = n$ and all the data points are clustered.

Note that the behavior of the Bregman Bubble for the two extreme cases (0 and infinite pressure) is also analogous to the phenomena of water bubbles reducing to very small sizes under extreme external hydrostatic pressure, and expanding to unlimited extent when all pressure is removed (such as for free moving air molecules in a perfect vacuum).

5.2 Motivation

BBC-S is able to find locally dense regions because of its ability to explicitly ignore large amounts of data by considering only points close to the cluster representatives for cluster membership. If the bubbles are initialized in sparse regions, they have to expand in order to enclose s points. Then, during each iteration, the bubble representatives move to lower cost nearby locations, and the bubbles shrink in their extent⁹. These mechanisms ensure that the results are less sensitive to initialization. However, when threshold s is small, only a few close neighbors get assigned, thereby decreasing the mobility of the representatives at each iteration. This makes it difficult for BBC-S to find small, dense regions far from initial seed locations. Addressing this issue by starting with a large s would be contrary to the goal of finding small dense regions. This problem is even more severe with BBC-Q, since the bubbles cannot expand automatically in sparser regions. Is there a way to improve upon the ability of BBC-S to “expand” in a sparse region, while still optimizing clustering over small, dense regions?

The pressure mechanism described in the previous section indicates a way of ameliorating this problem. The essential idea is to start BBC with a very small pressure, allowing it to reach out to all the data points, and then slowly increasing the pressure, which would correspond to increasing α_0 or s for Soft and Hard BBC respectively. This causes the bubbles to be “squeezed” by the increasing external pressure into denser regions. Moreover, bubbles that move to a denser

⁹ A toy example demonstrating this phenomena can be seen in our power-point slides at <http://www.ideal.ece.utexas.edu/~gunjan/bbc/bbcicdm.ppt.gz>, slides 16 through 20.

region retain proportionately more points at the expense of bubbles in less dense regions. This is because when points compete for being assigned to one of the k clusters (Stage 2 of Algorithm 1), the ones nearest to their respective centroids get assigned first, while $n - s$ points farthest from their cluster centroids get dropped. Both of these trends help to improve solution quality. A demo of the BBC-Press algorithm in action on the Gauss-2 dataset illustrating this “squeezing” phenomena can be seen at <http://www.ideal.ece.utexas.edu/~gunjan/bbc/bbcicdm.ppt.gz>, slides 31 to 36.

5.3 BBC-Press

Based on the ideas described in the last two sections, we propose an algorithmic enhancement to BBC-S that we call *Pressurization* that is designed to improve upon the quality of the local minimum discovered. We start the first iteration of BBC-S with a small enough pressure to cause all points to be assigned to some cluster, and slowly increase the pressure after each iteration. An additional parameter $\gamma \in [0, 1)$ that controls the rate of pressure increase is used as an exponential decay parameter¹⁰, and $s_j = s + \lfloor (n - s)\gamma^{j-1} \rfloor$ is used instead of s for the j^{th} iteration. Convergence is tested only after $(n - s)\gamma^{j-1} < 1$. A slower but more robust alternative involves running BBC-S to full convergence after each recomputation of s and using the resultant centroids to seed the next iteration, and in practice yields slightly better results. Algorithm 3 describes the steps for the full-convergence version of BBC-Press in more detail. Pressurization can similarly be implemented for the alternate formulation BBC-Q, by varying the fixed cost q_{\max} .

Algorithm 3. Hard BBC-Press

Input: Set $\mathcal{X} = \{\mathbf{x}\}_{i=1}^n \subset C \subseteq \mathbb{R}^d$, Bregman Divergence D_ϕ , no. of clusters k , desired clustering size s , k seed centroids for the k clusters (optional).

Set γ to a value between 0 and 1, $j = 1$, and $s_{in} = n$.

Run Algorithm 1 until convergence with random starting centroids, unless seed centroids are available, and with $s = s_{in}$.

Set $j = 2$, $s_{in} = s + \lfloor (n - s)\gamma^1 \rfloor$

while $(n - s)\gamma^{j-1} >= 1$ **do**

$s_{in} = s + \lfloor (n - s)\gamma^{j-1} \rfloor$. /* Reduce s_{in} by pressurization rate */

Run Algorithm 1 until convergence, using the k centroids from last run to seed the centroids for this run, and $s = s_{in}$.

end while

Return resultant clustering from the final run of Algorithm 1.

¹⁰ A smaller value gives better results, but runs slower. A value between 0.01 and 0.05 seems to work well for most real-world scenarios.

5.4 Soft BBC-Press

The Pressurization scheme can also be extended to Soft BBC for Case B when α_0 is not updated. When α_0 and p_0 are large (close to 1), only a small amount of data is “explained” by the k exponential mixtures. This may lead to bad local minima problems similar to (although less severe than) the one faced in BBC. Therefore, we propose a soft version of Pressurization that takes a decay parameter $\tau \in [0, 1)$ and runs Soft BBC (Case B) multiple times as follows: (1) start with some initial model parameters $\{\theta_j^1\}_{j=1}^k$ and run Soft BBC to convergence, (2) at trial r set α_0 to $\alpha_r = \alpha_0(1 - \tau^{r-1})$, and for $r > 1$ set current model parameters to the output of last trial: $\{\theta_j^r\}_{j=1}^k = \{\theta_j^{r-1}\}_{j=1}^k$. Repeat step (2) until $\alpha_r - \alpha_0$ is smaller than ϵ (a small positive value close to 0, e.g. 0.001), and then perform a final run with $\alpha_r = \alpha_0$.

5.5 Pressurization vs. Deterministic Annealing

Although our concept of Pressurization conceptually resembles the approach of *Deterministic Annealing* [68], they are not the same. For example, deterministic annealing in a Gaussian mixture modeling setting would involve gradually reducing the variance term σ^2 (Equation 17), whereas Soft Pressurization involves gradually increasing the probability mass α_0 (Equation 4) of the uniform background distribution. A notable property of Pressurization is that it works on both Hard and Soft BBC, whereas Deterministic Annealing is only applicable in a soft setting. This is significant for large, high dimensional datasets; a Deterministic Annealing approach for improving local search would require us to use Soft BBC, which contains exponential mixtures, and exponential mixtures are generally hard to compute on high-dimensional datasets because of rounding errors [9].

6 A Unified Framework

6.1 Unifying Soft Bregman Bubble and Bregman Bubble Clustering

We are now ready to look at how the generative model Soft BBC relates to the BBC problem, specifically the formulation where the number of points classified into the k real clusters (excluding the “don’t-care” cluster) is fixed (**Definition 1**, Section 3.2), and show the following:

Proposition 6.1. *Maximizing $L_2(\Theta|\mathcal{X})$ is identical to minimizing the BBC objective function Q_b (Equation 2).*

Proof. Let us consider the cost function:

$$L_2(\Theta|\mathcal{X}) = \sum_{i=1}^n E_{p^\dagger(Y_i=j|\mathbf{x}_i, \Theta)} [\log p(\mathbf{x}_i, Y_i = j|\theta_j)] \quad (14)$$

where $p^\dagger(Y_i = j|\mathbf{x}_i, \Theta) = 1$ for $j = \operatorname{argmax}_{0 \leq j \leq k} p(\mathbf{x}_i, Y_i = j|\theta_j)$ and 0 otherwise, which is essentially equivalent to the posterior class probabilities based on the hard assignments used in BBC. It can be shown [43] that for a fixed set of mixture parameters $\Theta = \{\theta\}_{j=1}^k$, and $L(\Theta|\mathcal{X})$ being the log-likelihood objective of Soft BBC (Equation 5):

$$L_2(\Theta|\mathcal{X}) \leq L(\Theta|\mathcal{X}) \quad (15)$$

This result is independent of the choice of priors $\{\alpha_j\}_{j=0}^k$. Note that while $L(\cdot)$ depends upon the priors, $L_2(\cdot)$ does not. For our choice of mixture components, based on Equations 3 and 15, one can readily obtain the following form for $L_2(\cdot)$:

$$\begin{aligned} L_2(\Theta|\mathcal{X}) &= \sum_{j=1}^k \sum_{\forall Y_i=j} \log p^\phi(\mathbf{x}_i) - \\ &\quad \beta D_\phi(\mathbf{x}_i, \theta_j) + \sum_{\forall Y_i=0} \log(p_0)[i]_{i=1}^n \end{aligned} \quad (16)$$

If the number of points assigned to the uniform distribution is fixed to $n - s$, s points are assigned to the k exponential distributions, and p_0 and β are fixed, we can see from Equation 16 that maximizing $L_2(\Theta|\mathcal{X})$ is identical to minimizing the BBC objective function Q_b (Equation 2).

Proposition 6.2. *BBC with a fixed s as input (Definition 1, Section 3.2) is a special case of Soft BBC with fixed α_0 .*

Proof. Let us consider an extreme case when $\beta \rightarrow \infty$ for Soft BBC (see equations 5 and 3). Then the class posterior probabilities in Soft BBC converge to hard assignment (BBC) ensuring that $L(\Theta|\mathcal{X}) = L_2(\Theta|\mathcal{X})$ in Equation 16. Since BBC is equivalent to optimizing $L_2(\Theta|\mathcal{X})$ (Proposition 6.1), we can also view BBC with fixed s (**Definition 1**) as input as a special case of Soft BBC with fixed α_0 .

6.2 Other Unifications

The following other interesting unifications can also be shown easily for our framework:

1. BBC is a special case of BBC-Press when $\gamma = 0$.
2. Bregman Bubble Clustering becomes BBOCC when $k=1$.
3. Soft BBC¹¹ reduces to Bregman Soft Clustering when $p_0 = 0$.
4. Bregman Bubble Clustering reduces to Bregman Hard Clustering (which is a special case of Bregman Soft Clustering) when $q_{max} = \infty$ (for BBC-Q) or when $s = n$ (for BBC-S).

¹¹ For both cases A and B.

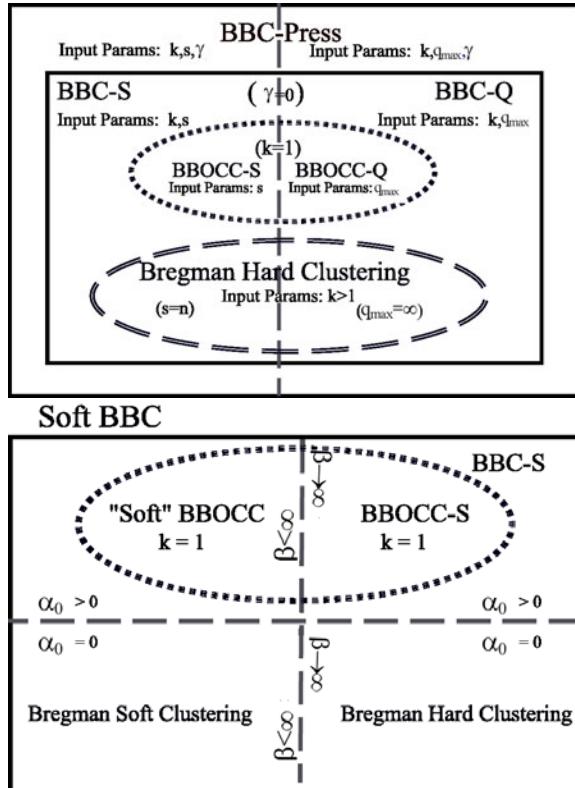


Fig. 2. Unification of various algorithms for a given Bregman Divergence D_ϕ : (top) BBC, BBOCC and Bregman Hard Clustering are special cases of BBC-Press. (bottom) Bregman Hard and Soft Clustering, BBC-S, BBOCC-S and a "soft" BBOCC (consisting of one exponential and a uniform background mixture) are special cases of Soft BBC obtained as specific combinations of (i) whether $\beta \rightarrow \infty$, (ii) whether α_0 is 0 (equation 4), and (iii) whether k is 1. Bregman Clustering (both hard and soft) for $k = 1$ does not result in a useful algorithm. BBOCC-S and BBOCC-Q represent BBOCC with fixed s or q_{max} as inputs respectively.

Figure 2 summarizes the hierarchy of algorithms descending from BBC-Press and Soft BBC. We could think of BBC as a search under "constant pressure", and for Bregman Hard Clustering this pressure is zero. Note that for $k = 1$, BBC gives rise to BBOCC. In the context of finding dense regions in the data, BBC can be thought of as a conceptual bridge between the problems of one class clustering and exhaustive k class clustering. However, the defining characteristic of BBC is its ability to find small, dense regions by modeling a small subset of the data. BBC combines the salient characteristics of both Bregman Hard Clustering and BBOCC resulting in an algorithm more powerful than either, and that works across all Bregman Divergences. BBC-S is a natural extension of BBOCC-S following directly from a common underlying generative model, and

is not just a heuristic; the difference in the generative model is only in having a single vs. multiple exponential distributions mixed with a uniform background.

7 Example: Bregman Bubble Clustering with Gaussians

Now that we have developed the theoretical framework for Soft BBC to work with all regular exponential distributions, we describe a concrete example with the Gaussian distribution, which is popularly used for many real-life applications. Let us consider spherical d -dimensional Gaussian distributions of the form:

$$N(\mathbf{x}|\mathbf{a}, \sigma) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{a}\|^2}{2\sigma^2}\right) \quad (17)$$

where $\mathbf{a} \in \mathbb{R}^d$ is the mean, and $\sigma^2 \in \mathbb{R}$ is the variance that is the same across all the d dimensions. There are two major variations of the Soft BBC algorithm depending upon how we treat the variance σ^2 :

7.1 σ^2 Is Fixed

Soft BBC: The parameters $\{\theta_j\}_{j=1}^k$ in Equation 4 correspond to the parameters of the k exponential distributions of the mixture model that are updated in the M step of Soft BBC (Algorithm 2). For the Gaussian example, if we fix the values of $\{\sigma_j^2\}_{j=1}^k$ for the k spherical Gaussians mixtures then the only parameters that can be updated are the k Gaussian means $\{\mathbf{a}_j\}_{j=1}^k$. For the Soft BBC algorithm this corresponds to $\theta = a$ and the sufficient statistics \mathbf{x}_s is simply \mathbf{x} . $\frac{1}{2\sigma^2}$ is the scaling parameter β (Equation 3) for the exponential function p^ϕ , $D_\phi(\mathbf{x}, \mathbf{a}) = \|\mathbf{x} - \mathbf{a}\|^2$ corresponds to the Squared Euclidean distance of \mathbf{x} from Gaussian mean \mathbf{a} , and $f_\phi(\mathbf{x}) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}}$.

Therefore, the E step of Algorithm 2 involves computing p^ϕ as $N(\mathbf{x}|\mathbf{a}, \sigma)$ using the current Gaussian means $\{\mathbf{a}_j\}_{j=1}^k$ and the fixed variances $\{\sigma_j^2\}_{j=1}^k$ and then performing the rescaling given by equations 6 and 7 to get $p(Y_i = j|\mathbf{x}_i)$ for all the n points. For the M step, the new priors $\alpha_{j=1}^k$ can now be re-estimated using either Equation 8 or 11 depending upon whether we keep α_0 fixed or not (Case A vs. B). The θ_j for j^{th} exponential component represented by the Gaussian mean \mathbf{a}_j can then be re-estimated as the weighted average of \mathbf{x} as described by Equation 10.

BBC-S: The proof for Proposition 6.2 tells us that the BBC-S algorithm will fall out from the Soft BBC when $\beta \rightarrow \infty$. For our Gaussian model with fixed variance, since $\beta = \frac{1}{2\sigma^2}$, this corresponds to setting the variances $\{\sigma_j^2\}_{j=1}^k \rightarrow 0$. This results in BBC-S (Definition 1, Section 3.2) using Squared Euclidean distance as the Bregman divergences. Furthermore, when we set $s = n$, this version of BBC-S also gives us the classical K-Means algorithm, or Bregman Hard Clustering with Squared Euclidean distance as D_ϕ . An example of output from such a BBC-S variant is shown in Figure 3 (d).

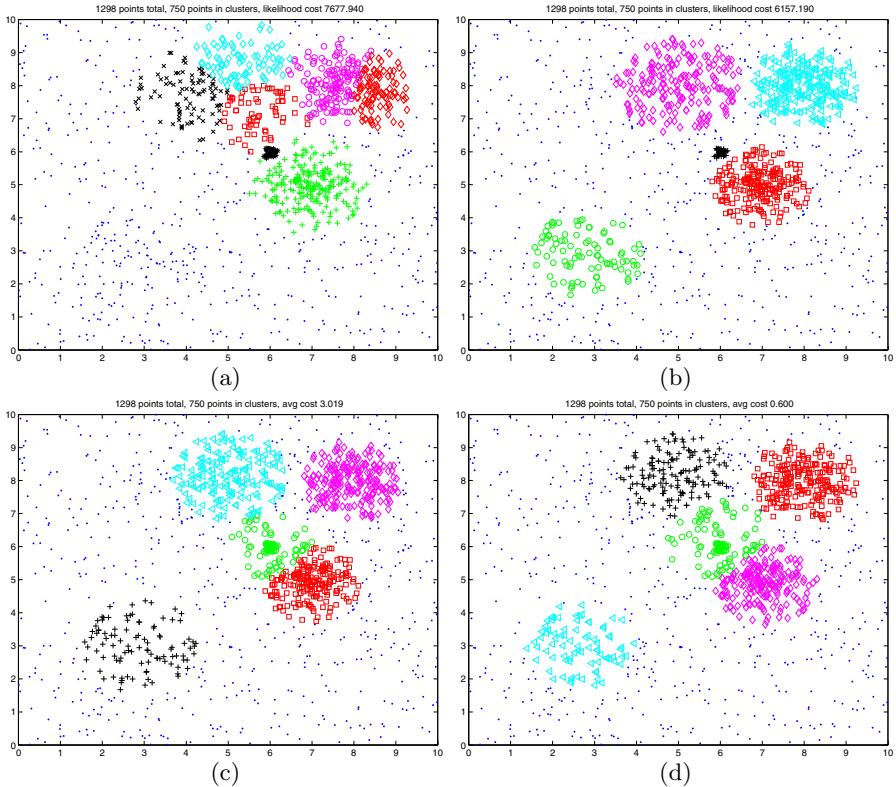


Fig. 3. Comparison of bubbles generated using a variant of Soft BBC and two variants of BBC-S on the simulated 2-D dataset: Soft BBC with updated σ^2 for (a) $k = 7$ and (b) $k = 5$. k was intentionally kept large in (a) to illustrate the non-linear boundaries produced by touching bubbles. BBC-S resulting from Soft BBC model where (c) σ^2 is updatable and (d) σ^2 is fixed. For Soft BBC, each point was assigned at convergence to the cluster to which it had the largest soft assignment value.

7.2 σ^2 Is Optimized

Soft BBC: If the variances $\{\sigma_j^2\}_{j=1}^k$ are also updated as a part of the EM, then both $\{\mathbf{a}_j\}_{j=1}^k$ and $\{\sigma_j^2\}_{j=1}^k$ get updated in the **M** step of Soft BBC (Algorithm 2) and the sufficient statistics \mathbf{x}_s becomes $[\mathbf{x}, \mathbf{x}^2]^T$. The **E** step of Algorithm 2 still involves computing p^ϕ as $N(\mathbf{x}|\mathbf{a}, \sigma)$ using the current Gaussian means $\{\mathbf{a}_j\}_{j=1}^k$ and the current variances $\{\sigma_j^2\}_{j=1}^k$ and then performing the rescaling given by equations 6 and 7 to get $p(Y_i = j|\mathbf{x}_i)$ for all the n points. For the **M** step, the new priors can also be re-estimated as before using either equations 8 or 11 depending upon whether we keep α_0 fixed or not. However, the θ_j for j^{th} exponential component, now a function of both the Gaussian mean \mathbf{a}_j and the variance σ_j^2 , needs to be re-estimated as the weighted average over the sufficient statistics. It can be shown that this maps to: (1) re-estimating the mean \mathbf{a}_j as the

average of \mathbf{x} over the n points weighted by $p(Y_i = j|\mathbf{x}_i)$, and (2) re-estimating the variance as a weighted average of $(\mathbf{x} - \mathbf{a}_j)^2$ over the n points also weighted by $p(Y_i = j|\mathbf{x}_i)$. An example of output from such a Soft BBC variant is shown in Figure 3 (b).

BBC-S: Unlike for the fixed variance case, the scaling parameter β cannot be thought of as a function of variance since σ^2 is a part of the updatable parameters. The corresponding D_ϕ , (which is not the Squared Euclidean distance) can be derived from the relationship defined by Equation 3 and corresponds to Mahalanobis distance in the original space of \mathbf{x} . A corresponding BBC-S algorithm obtained when $\beta \rightarrow \infty$ is different from the BBC-S algorithm described for the scenario where σ^2 was fixed. One property of such a generative model is that in the original space of \mathbf{x} , bubbles of varying diameters can be discovered by both the Soft BBC and the corresponding BBC-S algorithm, which could be suitable for domains where the natural clusters have very different diameters. It can be shown that in each iteration of such an implementation, the estimated distances to clusters need to be rescaled in proportion of the variances of the respective clusters in that iteration. An example of output from such a BBC-S variant is shown in Figure 3 (c).

7.3 “Flavors” of BBC for Gaussians

For Soft BBC built using spherical Gaussians, there are eight possible flavors (Table 1) depending upon whether (1) α_0 is updated (Case A vs. B, Section 4.4), (2) the Gaussian mixture variances are updated (Section 7.1 vs. 7.2), or (3) all cluster variances are forced to be equal. For the cases where variance could be updated, forcing them to be equal requires computing a weighted average of the variances of the k Gaussians after updating the variances in the M step as described in Section 7.2, and then assigning this weighted average to the variances of all the k Gaussians. Corresponding BBC-S for these eight flavors could also be derived. Figure 3 shows a comparison of bubbles generated using some of these variants for the simulated 2-D dataset (Gauss-2 dataset, Table 2, Section 10) that has points generated from 5 Gaussians of variances varying from small to large and a uniform background.

7.4 Mixture-6: An Alternative to BBC Using a Gaussian Background

For the Gaussian case where σ^2 is optimized, we could also define an alternative mixture model where the uniform background distribution is replaced by a background Gaussian distribution with a large variance¹². This results in a mixture of Gaussians model with $k + 1$ Gaussians where the 0th Gaussian has a fixed large variance and only its mean is updated, while for all other Gaussians both the mean and the variance are updated. Such a model can be viewed as a

¹² Much larger than the cluster variances.

Table 1. 8 flavors of Soft BBC for spherical Gaussians arise depending upon the choice of Θ

Flavor	Update σ	$\{\sigma_j\}_{j=1}^k = \sigma_1$	Fixed α_0
1	No	No	No
2	No	No	Yes
3	No	Yes	No
4	No	Yes	Yes
5	Yes	No	No
6	Yes	No	Yes
7	Yes	Yes	No
8	Yes	Yes	Yes

“hybrid” of the models in sections 7.1 and 7.2, and update steps using EM can be readily derived.

We call this model *Mixture-6* since it is analogous to the flavor 6 of Soft BBC (Table 1). Unlike in the Soft BBC where the background mass (and the corresponding fraction of data assigned to the background after converting to hard assignment at convergence) is easy to control and predict (Section 4.5), using a large variance background does not result in a stable background; the final background mass varies substantially depending upon where the center of the background Gaussian lies at convergence. Mixture-6 serves as another baseline for empirically evaluating Soft BBC.

8 Extending BBOCC & BBC to Pearson Distance and Cosine Similarity

8.1 Pearson Correlation and Pearson Distance

In biological organisms, genes involved in the same biological processes are often correlated in an additive or multiplicative manner, or both (Figure 4). *Pearson*

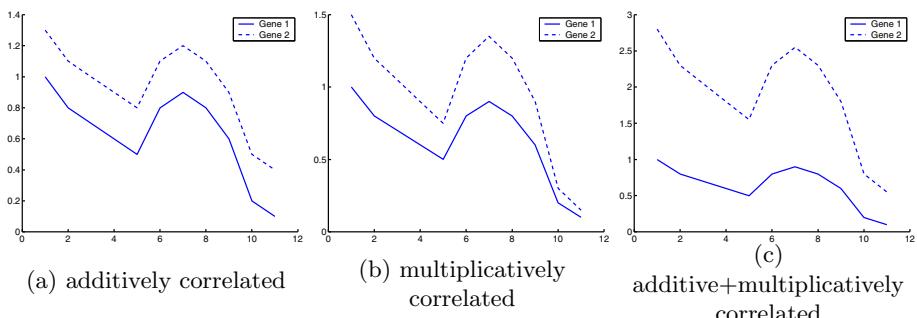


Fig. 4. Three common types of correlations observed between expression levels of genes. The x-axis represents distinct measurements at different time points/across conditions, while the y-axis represents the expression level of the gene.

Correlation captures the similarity between two variables in \mathbb{R}^d that is invariant to linear scaling, such as a multiplicative and/or additive offset, and is therefore a popular similarity measure for clustering gene-expression and other biological data [61, 52].

For two data points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, the Pearson Correlation P can be computed as $P(\mathbf{x}, \mathbf{y}) = \frac{zscore(\mathbf{x}) \cdot zscore(\mathbf{y})}{d-1}$, where $zscore(\mathbf{x}) = \frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}$ represents the vector-based z-scoring of the data point vector \mathbf{x} , $\mu(\mathbf{x})$ is the mean of the elements of the vector \mathbf{x} , and $\sigma(\mathbf{x})$ is the standard deviation. Note that we z-score each of the data points separately across features values¹³. We then define the *Pearson Distance* as $D_P = 1 - P$. Since $P \mapsto [-1, 1]$, therefore $D_P \mapsto [0, 2]$. It can be shown that Pearson Distance is equal to the Squared Euclidean distance between z-scored points normalized by $2(d-1)$:

$$D_P(\mathbf{x}, \mathbf{y}) = \frac{\| zscore(\mathbf{x}) - zscore(\mathbf{y}) \|_2^2}{2(d-1)} \quad (18)$$

D_P can also be viewed as the Squared Euclidean distance between points that have been first rotated by subtracting the mean, and then, by variance normalization, projected onto a hypersphere of radius 1 (radius $\frac{1}{\sqrt{2}}$ in Euclidean space) centered at the origin. When D_ϕ is replaced by D_P in Equation 2, we refer to Q_b as *Average Pearson Distance(APD)*.

Proposition 8.1. *For any cluster \mathcal{C}_j in \mathcal{G} , the cluster representative \mathbf{c}_j^* that minimizes contribution to APD by that cluster is equal to the mean vector of the points in \mathcal{C}_j projected onto a sphere of unit radius, i.e. $\mathbf{c}_j^* = argmin_{\mathbf{c}_j} (APD(\mathcal{C}_j, \mathbf{c}_j)) = \frac{\mathcal{C}_j^m}{\|\mathcal{C}_j^m\|}$, where $\mathcal{C}_j^m = \frac{1}{|\mathcal{C}_j|} \sum_{i: \mathbf{x}_i \in \mathcal{C}_j} zscore(\mathbf{x}_i)$.*

The proof for the above proposition follows directly from the result used by [19] for updating the center for their *Spherical K-Means* algorithm, which is a K-Means type of algorithm that uses Cosine Similarity as the similarity measure. Pearson Distance and Cosine Similarity are closely related; if we estimate the Squared Euclidean distance between points normalized by their L2-norm ($\sqrt{\sum_{i=1}^d \mathbf{x}_i^2}$) instead of between z-scored points, we obtain $2 \times (1 - \text{Cosine Similarity})$. Note that $(1 - \text{Cosine Similarity})$ of z-scored points is the same as Pearson Distance. Because of Proposition 8.1, for $D = D_P$ the optimum representative computation in BBC involves the averaging of the z-scored points rather than the original points, and then re-projecting of the mean onto the sphere. This minor modification to BBC allows it to work with D_P and ensures convergence to a local minimum¹⁴. Since BBOCC is a special case of BBC, the same modification works for BBOCC too for the problem of One Class Clustering.

¹³ This is different from the way z-scoring is often used in statistics, where it is performed for each column/dimension. We are performing it across rows of a data matrix, if the rows were to represent the data points.

¹⁴ And the corresponding local maximum for Average Pearson Correlation.

8.2 Extension to Cosine Similarity

Because of the relationship between Cosine Similarity and Pearson Distance described above, BBC will also work with Cosine Similarity, which is a popular similarity measure for clustering textual data [19]. Note that for $s = n$, BBC with Cosine Similarity degenerates to Spherical K-Means. For running BBC with Pearson Distance, since the z-scored data points have zero mean across the d dimensions, the mean of the z-scored points \mathcal{C}_i^m used for center update only needs to be normalized by its L2-norm to obtain $zscore(\mathcal{C}_i^m)$. For this reason, if we z-score the individual data points in advance and run BBC using Cosine Similarity, it produces the same results as running BBC with Pearson Distance.

8.3 Pearson Distance vs. (1-Cosine Similarity) vs. Other Bregman Divergences – Which One to Use Where?

It is important to note that the effect of not subtracting the mean gives rise to different distance measures (Pearson Distance vs. (1–Cosine Similarity)) and can result in very different clusterings; points with additive offsets will not necessarily be close when using only the Cosine Similarity. This difference could be important depending upon the application. For example, Pearson Distance/Correlation is more suitable for gene-expression data (Figure 4), while Cosine Similarity works better for document clustering. In Section 10, we present results based on Pearson Distance for the biological datasets Lee and Gasch (see Table 2), while for the 20 Newsgroup data, where the features consist of words, (1-Cosine Similarity) is used.

A similar distinction should be kept in mind about Bregman Divergences in general; although BBC works with all Bregman Divergences, it can produce quite different results depending upon the choice of the divergence; the particular problem domain and the underlying exponential distribution (Equation 3 and Proposition 6.2) should guide the selection of the appropriate Bregman Divergence for BBC.

9 Seeding BBC and Determining k Using Density Gradient Enumeration (DGRADE)

We now present an alternative to Pressurization for alleviating the problem of local minima in our local search. For medium-sized datasets¹⁵, the seeding framework described in this section is computationally feasible on machines with modest resources, and provides two key advantages over Pressurization: (1) deterministic results, and (2) the ability to automatically determine the number of distinct dense regions in the data (k), the location of these dense regions, and their representative centroids in \mathcal{X} . These k representatives are then used to seed BBC.

¹⁵ Such as gene clustering datasets, where the number of genes is usually $O(10^4)$.

9.1 Background

In [27], we presented a deterministic enumeration based algorithm called Hyper-sphere One Class Clustering (HOCC) that finds an approximate, restricted solution to the problem of finding a single dense cluster in the data. The centroid location determined by HOCC can be used to seed BBOCC. HOCC is based on the observation that if \mathbf{c} is restricted to one of the sample data points \mathcal{X} , then the number of distinct solutions is only $n(n - 1)$, and can be enumerated efficiently.

The problem of finding a good seeding method for local search for BBC for $k > 1$ is a harder problem than for the One Class case, since the search space for possible initializations gets much larger. For a given k , a simple extension of the strategy used in HOCC that involves restricting the search for cluster representatives to the given data points, and then enumerating all the $\binom{n}{k}$ combinations of centroids, is prohibitively expensive. On the other hand, picking the best solution over multiple trials of BBC-S or BBC-Press seems to give quite high quality solutions in practice, but also requires k as an input. Is there a fast, HOCC-type algorithm that can be used for seeding BBC and also indicates a suitable value of k ? This is answered positively via the DGRADE algorithm described next.

9.2 DGRADE Algorithm

The key idea behind the DGRADE seeding algorithm is to (i) limit the search for seeds to the actual data points and (ii) do the search in a computationally efficient manner. For each point, we consider the cost of a Bregmanian Ball that is centered at that location, and encompasses $s = s_{one}$ points. This cost can be viewed as the “potential” at the corresponding point. If the cost at a neighboring point is lower, then that point is (locally) more preferable. This leads to a chain of local preferences viewed as a potential gradient. Points that follow a chain of preferences to the same final point (lowest local potential) can be viewed as belonging to the same partition, the physical analogy being that they are all part of the same “basin of attraction”. Thus the data points can be quickly grouped, with the number of groups indicating the corresponding value of k . Note that this k is dependent on s_{one} , which acts as a scale or smoothing parameter.

Density Gradient Enumeration (DGRADE), described in detail as Algorithm 4, has the following key steps:

1. Input integers s_{one} , and the number of dense points s to be classified into clusters.
2. Sort each row of the distance matrix and save the corresponding sorted s_{one} nearest neighbors indices into **radM**, **idxM** (just like in HOCC).
3. Compute cost Q_{one} for each of n points as cost of a Bregmanian ball of size s_{one} centered on the point.
4. Sort the n points by increasing cost and save the cost of the first s points.
5. Set $k = 1$, labels of all points to 0. Create a pointer corresponding to each of the n points.

Algorithm 4. DGRADE

Input: Distance matrix \mathbf{M} containing distance between all points, Bregmanian ball size s_{one} , number of points to be clustered $s \leq n$.

Output: k , Partitioning \mathcal{G}^* containing k clusters $\{\mathbf{C}_j\}_{j=1}^k$, and the corresponding k cluster centroids $\{\mathbf{c}_j^*\}_{j=1}^k$.

```

[radM, idxM] = sortrows(M)
for i = 1 to n do
  qlist(i) = Qone({idxM(i, j)}_{j=1}^{s_{one}}, x_i)
end for
[val, idx] = sort(qlist)
k = 1; {lab}_{i=1}^n = 0; {head}_{i=1}^n = 0;
head_{idx(1)} = ∅; lab_{idx(1)} = 1
10: for i = 2 to s - 1 do
    [hCost, hIdxIdx] = min(val({idxM(i, j)}_{j=1}^{s_{one}}))
    hIdx = idxM(i, hIdxIdx)
    if hIdx == idx(i) then
      k = k + 1; lab(hIdx) = k; head(hIdx) = ∅;
    15: else
      lab(hIdx) = lab(idx(i)); head(hIdx) = idx(i);
    end if
  end for
  Return k, {c_j^*}_{j=1}^k as the set of  $k$  points whose corresponding head pointers are ∅, the set of clusters formed by non-zero values in lab as  $\mathcal{G}^*$ .

```

6. Assign cluster label 1 to the lowest cost point¹⁶. Set its pointer to null.
7. Now pick the next $s - 1$ points in the order of increasing cost and perform the following: for each point \mathbf{x} find the lowest cost point \mathbf{y} among the closest s_{one} neighbors of \mathbf{x} (including itself). If $\mathbf{y} = \mathbf{x}$, set $k = k + 1$ and set label of the point to k and set pointer of \mathbf{x} to null. Else assign the label of \mathbf{y} to \mathbf{x} and set pointer of \mathbf{x} to point to \mathbf{y} .
8. Return the clustering \mathcal{G} consisting of the s densest points, and the k cluster centroids as the k points with pointers set to null.

At the end of the process, we get a set $\mathcal{G} \subseteq \mathcal{X}$ consisting of k clusters $\{\mathcal{C}_j\}_{j=1}^k$ formed by a subset of s points from \mathcal{X} with the lowest cost. We also get a pointer from each of the s points leading to a point of lower cost Q_{one} . There are exactly k points from \mathcal{G} that form k centroids, one for each cluster \mathcal{C}_j , and the centroid is the point in \mathcal{C}_j with the lowest cost. The pointers from each of the members of a cluster \mathcal{C}_j form a path of lower cost leading eventually to the centroid of the cluster. Figure 5 shows the output of DGRADE on the Gauss-2 dataset when (b) the assignments of all the points is performed, i.e. when $s = n$, vs. (a) when only $s = 750$ points are assigned. DGRADE consists of two distinct phases:

¹⁶ For the restricted One Class case ($k = 1$), returning this (lowest cost) point as a solution corresponds to the One Class seeding algorithm described in our earlier paper [27], and results in strong optimality guarantees for One Class that are described in more detail in [27, 28].

(1) sorting of points by Bregmanian ball cost to get the absolute measurement of cost in various regions of the data, and (2) pointing each data point to the direction of maximum decline in cost within the same Bregmanian ball to get an estimate of the direction of maximum cost decline. The second phase generates a global map of the distinct valleys that ultimately converge to the locally dense (restricted) centroids. Essentially, the algorithm performs a global search for local density cost estimate *and* the gradient direction, and unlike density-based clustering methods such as DBSCAN, is compatible with asymmetric Bregman Divergences¹⁷. Although DGRADE can be used as a algorithm to find dense regions, or as an exhaustive clustering algorithm (for $s = n$), our main goal in designing it was to obtain a seeding solution for BBC-S. Therefore, we simply use the centroids discovered by DGRADE to seed BBC-S.

A detailed time and space complexity of DGRADE is given in [28]. To a first approximation, time complexity is quadratic in data size, and space requirements are only $O(n)$.

9.3 Selecting s_{one} : The Smoothing Parameter for DGRADE

s_{one} , which is a parameter for DGRADE that needs to be input by the user, acts like a smoothing parameter; a larger value typically results in a smaller k . When s_{one} is increased, the number of clusters found drops rapidly (Figure 5(c)), and beyond a certain value of s_{one} a consecutive set of values result in the same k . This characteristic enables several alternatives for selecting s_{one} automatically. We now present three common scenarios and the corresponding solutions for them:

1. If k is known, we can find the smallest $s_{one} \geq 2$ that results in k clusters and a binary search could be performed in $O(n^2 \log(n))$ time for finding the best centroids of the k clusters using DGRADE. The clustering using this approach on the Gauss-2 data is shown in Figure 5 (a) and (b).
2. If k is not known and somewhat “over-split” clusters are preferred, a user can specify a maximum *stability* integer value of m and a linear search could be performed for up to a certain maximum value of s_{one} to find the value of s_{one} after which $s_{one} + 1$ to $s_{one} + m - 1$ values all result in k clusters. This value of s_{one} and the corresponding solution of DGRADE is then returned. This technique is more appropriate for many biological datasets where the signal-to-noise ratio is often very low, and the clustering present is extremely weak.
3. For datasets with well-defined or prominent clusters, we can simply select k with the largest stability for s_{one} ranging from 1 to the smallest value that returns $k = 1$. Then, we select the smallest s_{one} that gives k clusters. This selection method is shown in Figure 5 (c), where $k = 4$ is obtained for $62 \leq s_{one} \leq 214$, corresponding to the largest stable interval. The smallest

¹⁷ Bregman Divergences are generally not symmetric; Squared Euclidean is a notable exception.

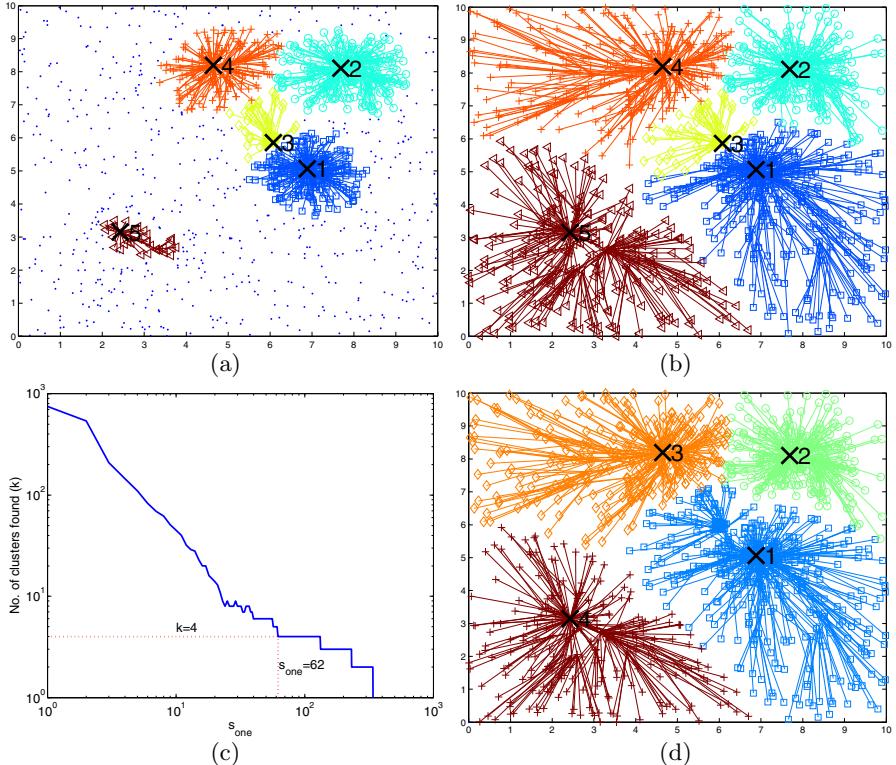


Fig. 5. Clustering of Gauss-2 data using DGRADE for various scenarios. For (a), (b) and (d), lines show the path of the simulated local search converging at the locally lowest cost/densest point, which also form the centroid of the corresponding cluster, and are marked as “x” and numbered 1 to k . For (a) and (b), k is known and set to 5, and s_{one} was automatically determined as 57. (a) shows clustering for $s = 750$, while (b) for $s = n$. For unknown k , (c) shows the relationship between s_{one} and k when $s = n$ and how it can be used for choosing s_{one} and determining k automatically. The most stable k and the corresponding smallest s_{one} is shown using the dotted lines. (d) shows the four clusters discovered by DGRADE using the automatic model selection described in (c).

s_{one} for $k = 4$ is 62, which is the value used for the final clustering output (for $s = n$) shown in Figure 5 (d). It is interesting to note that the densest cluster (numbered 3 in Figure 5 (b)) gets merged with a nearby larger cluster (numbered 1 in Figure 5 (b)) resulting in $k = 4$, which is quite good for this completely parameterless, unsupervised setting.

10 Experiments

10.1 Overview

Results in Section 10.4 show the effectiveness of BBC with Pressurization in finding high-quality, robust results as compared with three other methods, against three real and three synthetic datasets. Section 10.5 then presents the corresponding results when using DGRADE to seed BBC. Although on some datasets DGRADE gave good results by itself, more consistently, seeding BBC with the centers found by DGRADE gave results that were significantly better than using either BBC or DGRADE separately, and generally better than even BBC with Pressurization. Results suggest that it is also possible to combine all three: BBC, Pressurization and DGRADE to achieve the best quality of clustering, and confirm that in practice, DGRADE can estimate k automatically, and the deterministic, high quality results generated are a good alternative to Pressurization for biological datasets.

10.2 Datasets

We tested our algorithms on multiple real and synthetic datasets that are summarized in Table 2.

Table 2. A summary of the datasets used. D is the distance function used for clustering while \mathcal{C} represents the number of classes for labeled datasets only. k represents the number of clusters specified to the clustering methods that require it as an input, and is only needed for BBC when testing without DGRADE. When seeding with DGRADE, k output by DGRADE was used for all methods that required it as an input.

Dataset	Source	n	d	D	k	\mathcal{C}
Lee	Microarray	5,612	591	D_P	9	NA
Gasch Array	Microarray	173	6,151	D_P	12	12
Cleaned 20-NG Web documents	Web documents	19,975	4,292	(1-Cosine Sim.)	6	6
Gauss-2	Synthetic	1,298	2	Sq. Euclidean	5	5
Gauss-10	Synthetic	2,600	10	Sq. Euclidean	5	5
Gauss-40	Synthetic	1,298	40	Sq. Euclidean	5	5

Real Data

A. Microarray Datasets: A microarray dataset can be represented by a matrix that shows (suitably normalized) expression levels of genes (rows) across different experiments/conditions (columns). Researchers are interested in clustering either the rows, the columns, or simultaneously clustering both rows and columns, in order to find similar genes, similar conditions, or subsets of genes with related expressions across a subset of conditions, respectively [62, 60, 4]. For this paper, we report results on clustering the rows of the Lee dataset, which was obtained from [47], and consists of 591 gene-expression conditions on yeast obtained from

the Stanford Microarray database [24] (<http://genome-www5.stanford.edu/>), and also contains a *Gold* standard based on Gene Ontology (GO) annotations (<http://www.geneontology.org>). The Gold standard contains 121,406 pairwise links (out of a total of 15,744,466 gene pairs) between 5,612 genes in the Lee data that are known to be functionally related. The Gold standard was generated using Gene Ontology biological process from level 6 through 10. The Gasch dataset [21] consists of 6,151 genes of yeast *Saccharomyces cerevisiae* responding to diverse environmental conditions over 173 microarray experiments. These experiments were designed to measure the response of the yeast strain over various forms of stress such as temperature shock, osmotic shock, starvation and exposure to various toxins. Each experiment is labeled with one of the 12 different categories of experiments. For Gasch, we clustered the columns instead, for three reasons: (1) we have good labels for experiments from [21], (2) the Gasch Array dataset viewing the conditions as the objects to be clustered, provides a high-dimensional biological testbed (6,151 dimensions), and (3) the 173 Gasch Array experiments are already incorporated in the 591 conditions contained in the Lee dataset, which we use for clustering genes.

B. Text Data: The 20-Newsgroup (20-NG) dataset is a popular dataset for text classification [45], and is widely available on the web including the KDD UCI repository (<http://kdd.ics.uci.edu/>). It consists of 20,000 Usenet articles taken from 20 different newsgroup, 1,000 from each newsgroup. The 20 groups can also be categorized into 6 high-level categories shown in Table 3, which are then used as the class labels for evaluating the clustering algorithms. The 20-NG data contains over 50,000 distinct words after removing punctuations, common words such as articles, and stemming. Since we use the 20-NG data to evaluate (unsupervised) clustering algorithms, we used an unsupervised approach for selecting features; we selected 4,292 most frequent words (all words occurring ≥ 100 times over the 20,000 documents) as features.

Table 3. The 6 top-level classes (\mathcal{C}) in the 20-Newsgroup data

\mathcal{C}	$ \mathcal{C} $	Member newsgroups
Computers	4,959	<i>comp.graphics, comp.os.ms-windows.misc,</i> <i>comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x</i>
Recreation	3,984	<i>rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey</i>
Science	3,989	<i>sci.crypt, sci.electronics, sci.med, sci.space</i>
Miscellaneous	988	<i>misc.forsale</i>
Talk	2,994	<i>talk.politics.misc, talk.politics.guns, talk.politics.mideast</i>
Religion	2,994	<i>talk.religion.misc, alt.atheism, soc.religion.christian</i>

Synthetic Data: These datasets are useful for verifying algorithms since the true labels are known exactly. The Gauss-2 dataset was generated using 4 2-D Gaussians of different variances (Figure 3) and a uniform distribution. Similar datasets were generated with 5 Gaussians in 10-D and 40-D to produce Gauss-10 and Gauss-40 datasets.

10.3 Evaluation Methodology

Having shown the value of the local search approach and seeding for the One Class problem in [27], this paper presents results for the general case of finding multiple dense clusters. Additional results for the One Class case can be found in [28].

Evaluation Criteria: Evaluating clustering is a challenging problem even when labeled data is available [65]. Depending upon the type of the labeled data, we performed the following three different types of evaluations:

1. *Adjusted Rand Index:* Given a set of class labels \mathcal{U} , and a set of cluster labels \mathcal{V} for a set of points \mathcal{X} , *Rand Index (RI)* is computed as:

$$RI = \frac{a_{uv} + a_u}{a_{uv} + a_u + a_v + d_{uv}} \quad (19)$$

where: a_{uv} represents the number of pairs of points (in \mathcal{X}) that have the same label in \mathcal{U} and \mathcal{V} , a_u represents the pairs of points that have the same label in \mathcal{U} but not in \mathcal{V} , a_v represents the pairs of points that have the same label in \mathcal{V} but not in \mathcal{U} , and d_{uv} represents the pairs of points that have different labels both in \mathcal{U} and \mathcal{V} .

A problem with Rand Index is that the expected value for Rand Index of two random partitions does not go to 0, but depends on a multitude of factors, including the number of distinct labels in \mathcal{U} and \mathcal{V} , and the size of the sets, (i.e. the number of data points being clustered). Adjusted Rand Index was proposed by [35] as a normalized version of Rand Index that takes care of this problem, and returns 1 for a perfect agreement between the class label set \mathcal{U} and the clustering label set \mathcal{V} , and 0 when the clustering is as bad as random assignments. The adjustment uses the following formula:

$$ARI = \frac{RI - (ExpectedRI)}{(MaximumRI) - (ExpectedRI)} \quad (20)$$

Using a generalized hypergeometric model, [35] showed that the ARI computation can be reduced to the following form:

$$ARI = \frac{\sum_{i,j} \binom{n_{i,j}}{2} - [\sum_i \binom{n_{i..}}{2} \sum_j \binom{n_{..j}}{2}] / \binom{n}{2}}{\frac{1}{2}[\sum_i \binom{n_{i..}}{2} + \sum_j \binom{n_{..j}}{2}] - [\sum_i \binom{n_{i..}}{2} \sum_j \binom{n_{..j}}{2}] / \binom{n}{2}} \quad (21)$$

where $n_{i,j}$ represents the number of data points that are in class i and cluster j , $n_{i..}$ represents the number of data points in class i , $n_{..j}$ represents the number of data points in cluster j .

ARI can be used on the Gasch Array, 20-NG and the synthetic datasets since the true class-labels are available.

2. *p-value:* We use p-value to evaluate individual clusters of Yeast genes found using BBC for the Lee dataset. *Funspec* (<http://funspec.med.utoronto.ca/>) is a popular Yeast database query interface on the Web that computes cluster p-values for individual clusters using the hypergeometric distribution,

representing the probability that the intersection of a given list of genes with any given functional category occurs by random chance. *p-value* is a commonly used measure of individual cluster quality used by bioinformatics researchers.

3. *Overlap Lift*: For evaluating the overall clustering quality, it is not possible to use ARI to evaluate against the links in the Lee Gold standard. In general, measuring overall clustering quality for genes is quite difficult since only an incomplete and partially verified ground truth is known, such as the links in the Lee Gold standard. We propose *Overlap Lift* as a measure of the statistical significance of our clustering results against the gold standard as follows: a cluster containing w genes creates $w(w - 1)/2$ links between genes, since every point within the cluster is linked to every other point. Therefore, k clusters of size $\{w_j\}_{j=1}^k$ would result in a total of $l_c = \sum_{j=1}^k w_j(w_j - 1)/2$ links. The fraction of pairs in the Gold standard that are linked f_{linked} is known (for example for Lee dataset $f_{\text{linked}} = 121,406/15,744,466 = 0.007711$). If we construct a null hypothesis as randomly picking l_c pairs out of $n(n-1)/2$ pairs in the Gold standard, we can expect $l_{\text{null}} = f_{\text{linked}}l_c$ pairs to be correctly linked. A good clustering should result in (a lot) more correctly linked pairs than l_{null} . If l_{true} is the number of correct links observed (which will always be $\leq l_c$) in our clustering, then the Overlap Lift is computed as the ratio $\frac{l_{\text{true}}}{l_{\text{null}}}$, which represents how many times more correct links are observed as compared to random chance. A larger ratio implies better clustering.

Handling “don’t care” points in evaluations: All the evaluations are performed across a range of coverage of the data; a coverage of $s/n = 0.4$ implies 40% of the points are in clusters while the remaining 60% are in the “don’t care” or the background cluster. The points in the background or the “don’t care” clusters are excluded from all evaluations. To keep the comparisons fair, all methods are compared against each other only across the same coverage.

Evaluating Soft BBC: We tested Soft BBC using Gaussians as the exponential mixture components. There are eight possible flavors of Soft BBC (Table 1) depending upon the choice of the updatable parameters. We present results on the Soft BBC implementation, flavor 6, i.e. with updatable, unequal variances with a fixed α_0 . We also compared Soft BBC for Gaussians with the alternative soft model called Mixture-6 (Section 7.4).

Hard Assignments for Soft BBC: To compare Soft BBC against BBC and other hard assignment methods, on convergence, the points are assigned to the mixture with the largest probability, i.e. to $j = \underset{j=0 \rightarrow k}{\operatorname{argmax}} p(Y_i = j|\mathbf{x}_i)$. The esti-

mation of p_0 described in Section 4.5 results in approximately $(n \times \alpha_0)$ points getting assigned to the background. In order to ensure that exactly $(n - s)/n$ points are assigned to the “don’t care” set, a post-processing is performed where we: (1) compute $p_{\max}^i = \max_{j=0}^k (p(Y_i = j|\mathbf{x}_i))$ for each \mathbf{x}_i , (2) set p_0^\dagger to the s^{th} largest value in $p_{\max}^i[i]_{i=1}^n$, (3) put all points below p_0^\dagger into the “don’t care”

cluster, and assign rest to cluster $j = \operatorname{argmax}_{j=1 \rightarrow k} p(Y_i = j | \mathbf{x}_i)$. A similar conversion was required for evaluating Mixture-6.

Comparison against other methods: We also compared our method with Bregman Hard Clustering, Single Link Agglomerative clustering and DBSCAN. Bregman Hard Clustering¹⁸ assigns every data point into a cluster. To be able to compare it meaningfully with BBC, we picked s points closest to their respective cluster representatives. This procedure was also used for Single Link Agglomerative clustering. For the two DBSCAN parameters, we set *MinPts* to 4 as recommended by [20], while we searched for *Eps* that resulted in s points in clusters. k is automatically estimated by DBSCAN while for all the other methods and datasets, when evaluating BBC with Pressurization, k was set to $|\mathcal{C}|$ (Table 2), except for the Lee dataset (where $|\mathcal{C}|$ is not known) where we set k to 9.

All five methods use the same and appropriate distance measure that corresponds to the D listed for each of the datasets in Table 2; Sq. Euclidean for the synthetic Gaussian datasets, Pearson Distance for the gene-expression datasets, and (1-Cosine Similarity) for the 20-Newsgroup data.

10.4 Results for BBC with Pressurization

For the lower dimensional datasets, both Soft and Hard BBC with Pressurization perform extremely well, giving near-perfect results ($\text{ARI} \approx 1$) for up to 40% coverage on Gauss-10 data and an ARI between 0.8 and 0.9 for up to 40% coverage on Gauss-2 data. As expected, both BBC and Soft BBC without Pressurization tend to be a lot more sensitive to initialization, thus exhibiting noticeable error-bars¹⁹. For Gauss-40 and all the real datasets, results are only shown for Hard BBC with Pressurization (BBC-Press). This is because exponential mixture models in general, including Bregman Soft Clustering, Mixture-6 and Soft BBC, all suffer from an inherent problem that makes them impractical for high dimensional datasets: there are rounding errors while estimating the mixture membership probabilities, and these rounding errors worsen exponentially with the dimensionality of the data d (e.g., for Gaussians, when L.H.S. of equation 17 is substituted for $p_{(\psi, \theta)}$ in equation 6), so much so that the models often do not work well beyond $d = 10$. However, the main purpose of designing Soft BBC was to show that a fundamental generative model lies behind BBC (Section 6).

On the Gauss-40 dataset, BBC-Press continues to give an $\text{ARI} \approx 1$ for up to 40% coverage (Figure 6(c)). These results are impressive given that the ARI was obtained as averages of multiple runs with random seeding. The improvement against labeled data using BBC Press as compared to BBC is also quite remarkable for the two micro-array datasets Gasch Array and Lee, and a similar

¹⁸ Bregman Hard Clustering reduces to K-Means when D_ϕ is Sq. Euclidean distance, which is the distance measure used for the Gaussian datasets (Table 2).

¹⁹ Note that the error bars were plotted on all the local search algorithms, but are often too small to be visible.

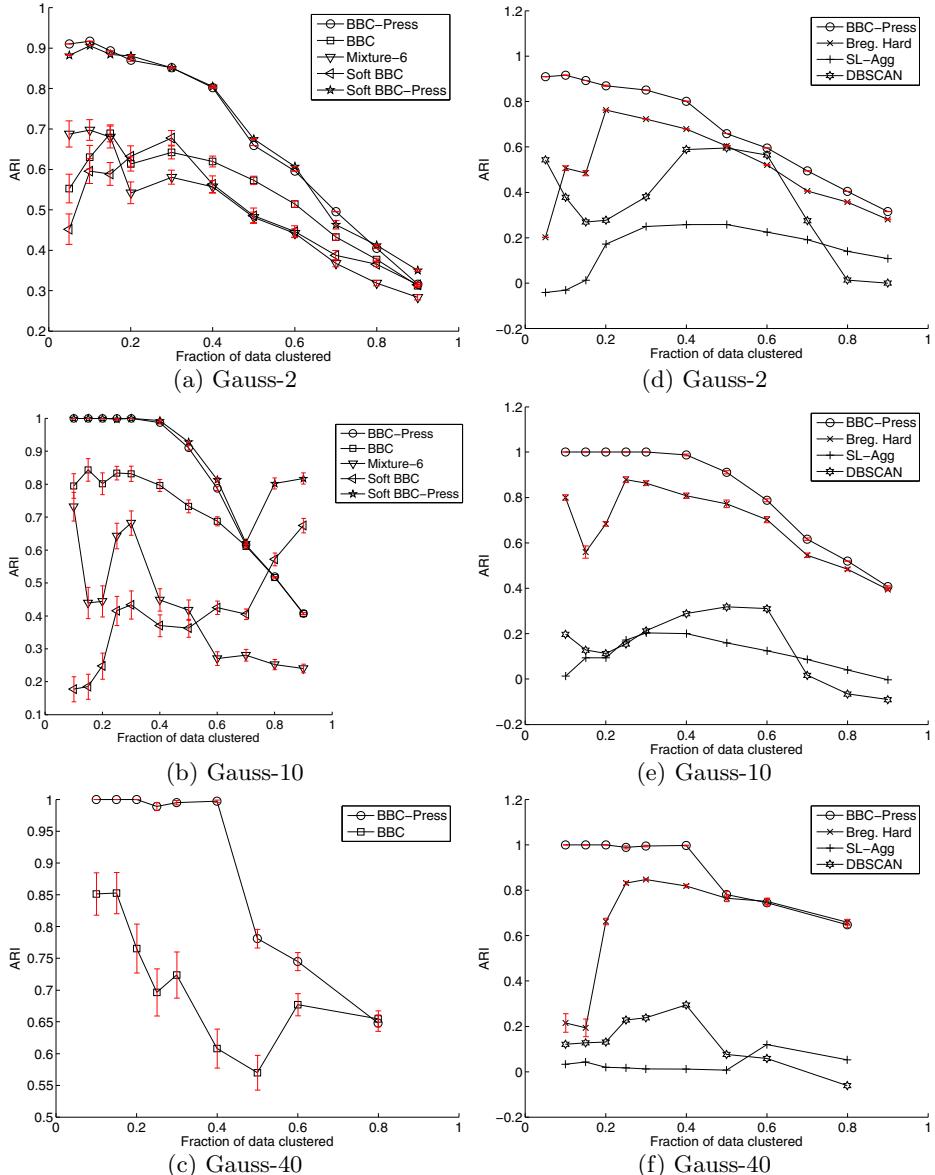


Fig. 6. Evaluation on synthetic Gaussian data of increasing dimensionality using ARI: (a), (b) and (c) demonstrate the effectiveness of Pressurization. (d), (e) and (f) show effectiveness of BBC-Press as compared to three other methods: Bregman Hard Clustering, Single Link Agglomerative and DBSCAN. Error bars of one std. deviation are shown (but are sometimes too small to be visible) for non-deterministic methods (i.e. excluding DBSCAN and Agglomerative) for which ARI is plotted as the average over 100 trials with random initialization.

trend is seen for 20-Newsgroup as well. This indicates that Pressurization works well on a variety of real datasets; from very high dimensional gene experiments (Figure 7(a)) where most of the data is relevant, discovering small number of relevant gene clusters on high-dimensional microarray data (7 (e)), to clustering large and high-dimensional documents (7 (c)).

On both artificial and real datasets (figures 6, 7), DBSCAN, Single Link Agglomerative and Bregman Hard Clustering all perform much worse than BBC-Press in general, and especially when clustering a part of the data. Note that these results are based on labels that were *not* used for clustering; using ARI on Gaussians, Gasch Array and the 20-Newsgroup data, and using Overlap Lift on Lee, and are therefore independent of the clustering methodology. Figure 7(f) shows that (1) BBC-Press not only beats other methods by a wide margin but also shows high enrichment of links for low coverages (over 6 times for 5 % coverage), and (2) Single Link Agglomerative clustering does not work well for clustering genes and gives results not much better than random. On all datasets, Single Link tends to perform the worst; one explanation might be its inability to handle noisy data. For 20-Newsgroup, the ARI of Single Link is not clearly visible although it has been plotted because it hovers close to 0 for all coverages. In fact, for some situations (Figure 6(d) to (f)), DBSCAN and Single Link Agglomerative give slightly worse than random performance resulting in ARI values that are slightly below 0. The performance difference between our method (BBC-Press) and the three other methods is quite significant on all the six datasets, given the small error bars. Additionally, if we were to pick the minimum-cost solution out of multiple trials for the local search methods, the differences in the performance between BBC-Press vs. DBSCAN and Single Link become even more substantial.

Selecting size and number of dense clusters in the absence of DGRADE seeding: In BBC-Press, s controls the number of data points in dense clusters. The dense clusters were invariably very pure when using BBC-Press, with near-perfect clusters on the Gaussian data for s of up to 40% of n , while on the Gasch Array dataset the performance peaks at a coverage of around 0.3 but shows a general decline after that. The rapid increase in cluster quality with decreasing s is more pronounced in BBC-Press than in the other methods, and shows that on these datasets, dense regions are indeed highly correlated with the class labels. In practice, selecting dense clusters with BBC-Press requires choosing an appropriate s and k . If a small amount of labeled data is available, the best k can be estimated for a fixed s using an approach such as PAC-MDL [5], while a reasonable s can be picked by applying BBC-Press on a range of s and picking the “knee” (e.g. Figures 6(a),(b),(c) and 7(b) show a sudden decline in ARI near $s = 0.4 \times n$). Alternatively, in many problems k can be an input, while s simply has to be a small threshold (e.g. for finding a small number of relevant web documents or a small number of relevant genes).

Evaluating individual clusters: Although the results based on ARI and Overlap Lift show the effectiveness of our method, visual verification serves as another

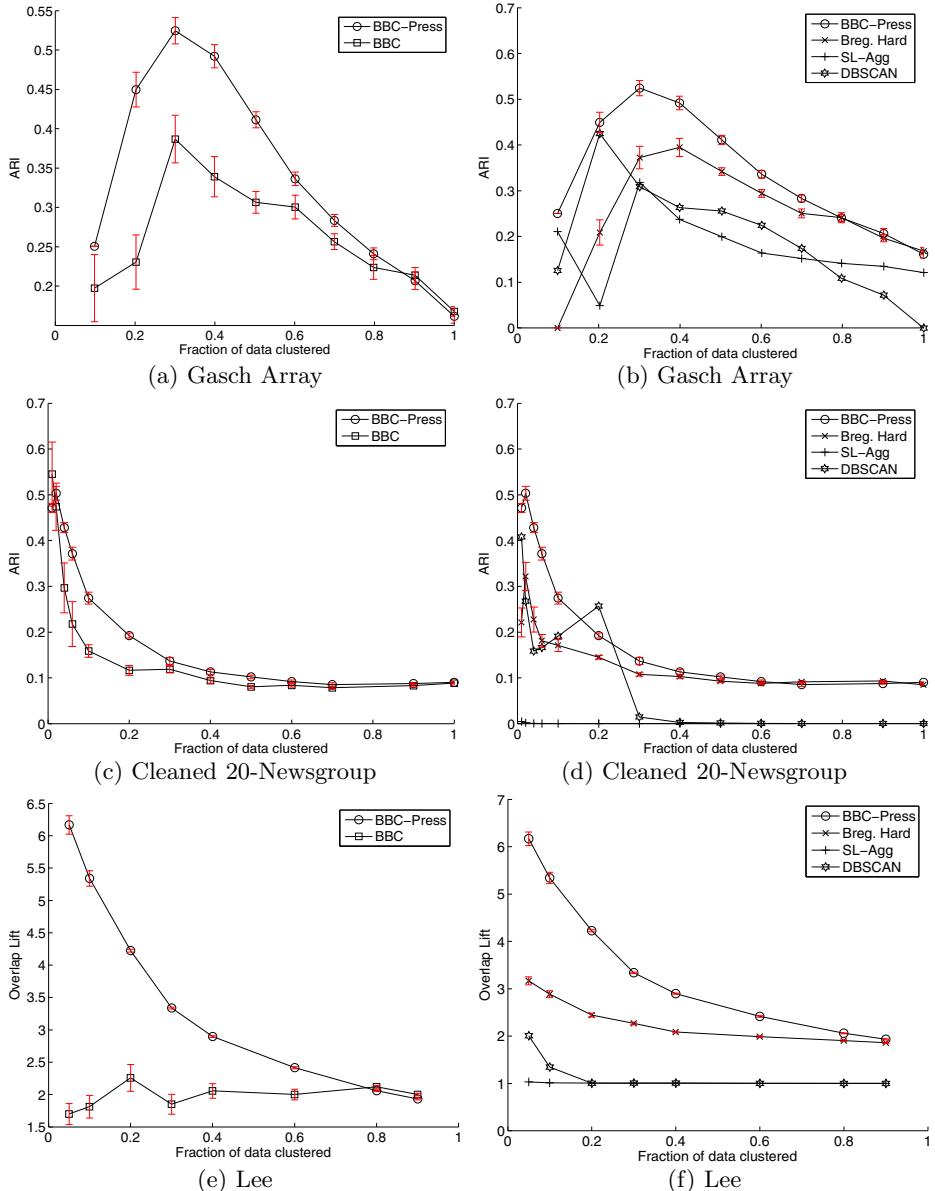


Fig. 7. Evaluation of BBC-Press on real data using ARI for Gasch Array and Cleaned 20-NG and Overlap Lift for Lee, as compared to BBC, Bregman Hard Clustering, Single Link Agglomerative, and DBSCAN. Local search (i.e. excluding DBSCAN and Agglomerative) results were averaged over 20 trials for Gasch Array and Cleaned 20-NG, and over 10 trials for Lee. The corresponding one std. dev. error-bars are plotted, but are sometimes too small to be visible.

independent validation that the clusters are not only statistically significant but also useful in practice. For the Gauss-2 dataset, it is easy to verify the quality of the clusters visually (Figure 3(b)). For the Gasch Array clustering, most clusters were generally very pure using BBC-Press for lower coverages. For example, when only 70 out of 173 experiments are clustered by repeating BBC-Press 20 times and picking the lowest cost solution, the average ARI is around 0.6 over 12 classes. Some clusters are even purer, for example, one of the clusters contained 12 out of 13 points belonging to the class “YPD”²⁰. Similarly, for the Lee dataset, when clustering only 600 genes into 30 clusters and pruning the rest, we verified a high purity cluster using FunSpec; 10 out of 14 genes in one of the clusters belonged to the functional category “cytoplasmic and nuclear degradation” (p -value of $< 10^{-14}$). Many other gene clusters on the Lee dataset also had low p -values for some of the categories recovered by FunSpec.

10.5 Results on BBC with DGRADE

In an alternative setting where DGRADE is used to seed BBC, we compared both DGRADE and BBC seeded with DGRADE with Bregman Hard Clustering, Single Link Agglomerative clustering and DBSCAN. The experimental setup and evaluation was similar to when comparing BBC with Pressurization against the three other methods, except that for a given coverage, k was automatically determined by DGRADE, and this k was then used as input for methods that require it: BBC, BBC-Press, Bregman Hard and Single Link. For DBSCAN, which determines k internally, as before, we set $MinPts$ to 4 as recommended by [20], while we searched for Eps that resulted in s points in clusters. The input parameter s_{one} required by DGRADE was determined automatically by using the approach described in Section 9. DGRADE then uses the same s_{one} for smaller coverages, which can result in a smaller k (Figure 9) as the lesser dense clusters become “don’t care” points, and the corresponding k is then used as input to all the algorithms requiring it. When seeding (Hard/Soft/Pressurized) BBC with the output of DGRADE, the cluster centroids output by DGRADE were used as the seed/initial centroids. We now present results on DGRADE when it is used for seeding BBC, on two real and three synthetic datasets.

Ability to estimate s_{one} automatically: In the table in Figure 8(a), the first column shows the values of s_{one} determined automatically using one of the three methods described in Section 9, the second column shows the number of clusters discovered when clustering all of the data ($s = n$), while the third column shows the user input, if any, required by DGRADE. For all the datasets except Lee, k was known and was used to determine s_{one} automatically. For the Gauss-2 dataset, when both k and s_{one} were determined automatically using the maximum cluster stability criteria, we obtained $k = 4$ (Figure 5(c) and (d)). For the Lee dataset, by using a cluster stability threshold > 1 we obtained $k = 9$ and $s_{one} = 10$. Figure 8 (b) shows the process of automatically determining both k and s_{one} for the Lee dataset; interestingly, $k = 9$ also had the maximum stability

²⁰ Which represents one of the class of experiments set up by [21].

Dataset	s_{one}	k , when $s = n$	User input
Gasch Array	4	11 ^a	$k = 12$
Lee	10	9	$stability > 1$ ^b
Gauss-2	62	4	None ^c
Gauss-2	57	5	$k = 5$ ^d
Gauss-10	104	5	$k = 5$
Gauss-40	57	5	$k = 5$

^a Closest possible k to 12 found when $s_{one} = 3$. For $s_{one} = 3$, k increases to 14.

^b See plot (b) in this figure.

^c Figure 5(c) and (d)

^d Figure 5(b)

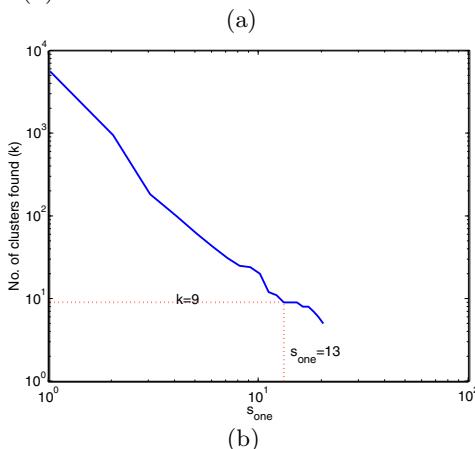
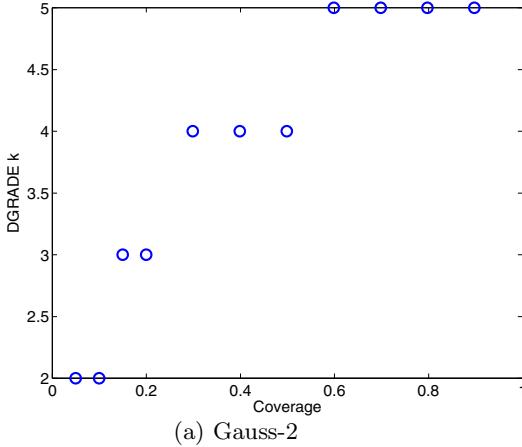


Fig. 8. (a) Automatically determined s_{one} for DGRADE on various datasets, using the approach described in Section 9. (b) For the Lee data, selecting the largest k (smallest s_{one}) with $stability > 1$ gives a $stability = 3$, $s_{one} = 13$ and $k = 9$.

of 3 in the range $2 \leq s_{one} \leq 20$, for which k ranged from 948 (for $s_{one} = 2$) to 5 (for $s_{one} = 20$).

Using cluster centroids from DGRADE for seeding and selecting k , for variable coverages: For a constant s_{one} , the results of DGRADE are not only deterministic for varying values of s , but also have another useful property that follows directly from Algorithm 4; using a smaller s gives clusters that are guaranteed to be subsets of the DGRADE clustering with a larger s . This effect can also be seen in Figure 5(b) vs. 5(a), when s was reduced from 1298 to 750. Eventually, some of the less dense clusters disappear completely resulting in a decline in k returned by DGRADE. However, the remaining centroids output by DGRADE remain unchanged. The decline in k with s can be seen for the 5 different datasets in Figure 9, and provides us with a meaningful method for selecting centroids and k for seeding BBC (for varying fraction of data clustered), as compared with other seeding methods that require k as an input. The k



Dataset	cov.(s/n) vs. k			
	Min.		Max.	
	s/n	k	s/n	k
Gasch Array	0.1	3	1.0	11
Lee	0.05	6	1.0	9
Gauss-10	0.1	3	1.0	5
Gauss-40	0.1	4	1.0	5

(b) Other datasets

Fig. 9. (a) On Gauss-2 data, the number of clusters k found by DGRADE declines asymptotically with the fraction of densest data clustered (s/n). (b) A summary of a similar trend on the other datasets. The maximum k corresponds to $s = n$, and the minimum k corresponds to the smallest coverage. s_{one} was held constant for all coverages, and corresponded to the automatically determined values of 4, 10, 57, 104, and 57 for the Gasch, Lee, Gauss-2, Gauss-10 and Gauss-40 datasets respectively.

corresponding to those shown in Figure 9 for various coverages were used as inputs to Bregman Hard and Single Link Agglomerative Clustering. Also, the corresponding centroids output by DGRADE for various coverages were used as inputs for seeding any of the forms of BBC (Hard/Soft/Pressurized).

DGRADE seeding works well with BBC: Figure 10 and 11 show results for BBC seeded with DGRADE as compared to the other alternatives; BBC with Pressurization and the other three benchmark algorithms. For the Gauss-2 dataset, when DGRADE was used to seed Soft BBC, the results were comparable to that of Soft BBC with Pressurization (Figure 10(b)), while the results of DGRADE as a clustering algorithm by itself were quite good (Figure 10(a)). On the Gauss-40 dataset (figures 10(e) and (f)), although DGRADE does not perform well by itself, when used for seeding BBC-Press, the combination gives results that are superior to all other algorithms, and beats even BBC-Press; an ARI close to 1 is observed for coverages as high as 0.6 as compared to only until 0.4 for BBC-Press. Similar trends were seen on the Gauss-10 (figures 10(c) and (d)) dataset. One possibility is that the global search bias behind DGRADE provides additional advantages to a robust local search algorithm such as BBC-Press, and especially on higher-dimensional datasets. This agrees with the intuition that the local search problems should become more severe with increasing dimensionality of the data (Gauss-40 vs. Gauss-2), and is similar to the boost in performance observed when DGRADE-style seeding is combined with local search for the One Class scenario [27, 28]. This phenomenon of DGRADE and BBC-Press improving results as a combination is also observed on the really

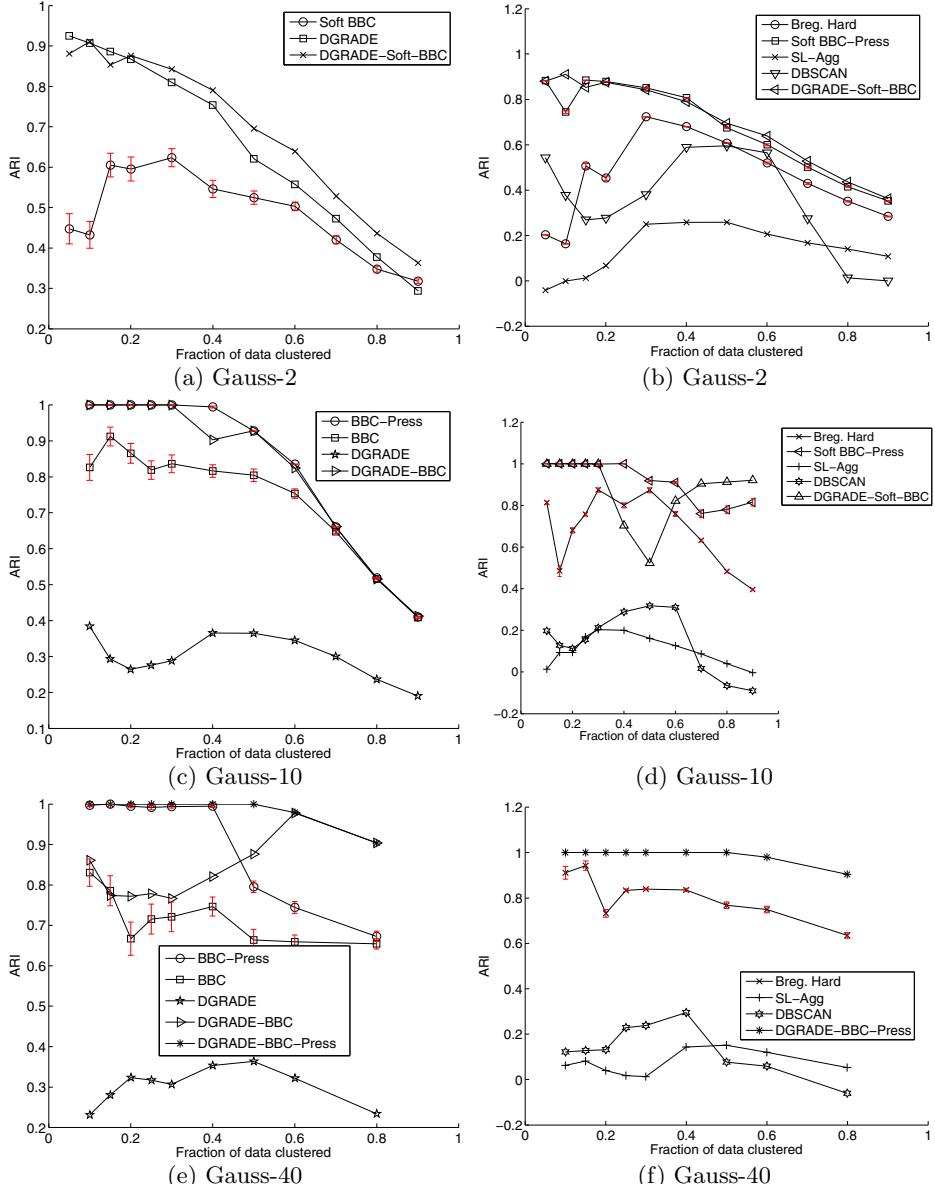


Fig. 10. Evaluation of BBC seeded with DGRADE using the synthetic Gaussian datasets: as compared to BBC with seeding, and against three other methods. Error bars of one std. deviation are shown (but are sometimes too small to be visible) for non-deterministic methods (i.e. all except DBSCAN and Agglomerative) for which ARI is plotted as the average over 100 trials with random initialization. Note: These experiments were setup differently from those without DGRADE; the k output by DGRADE was used as input to all algorithms except DBSCAN.

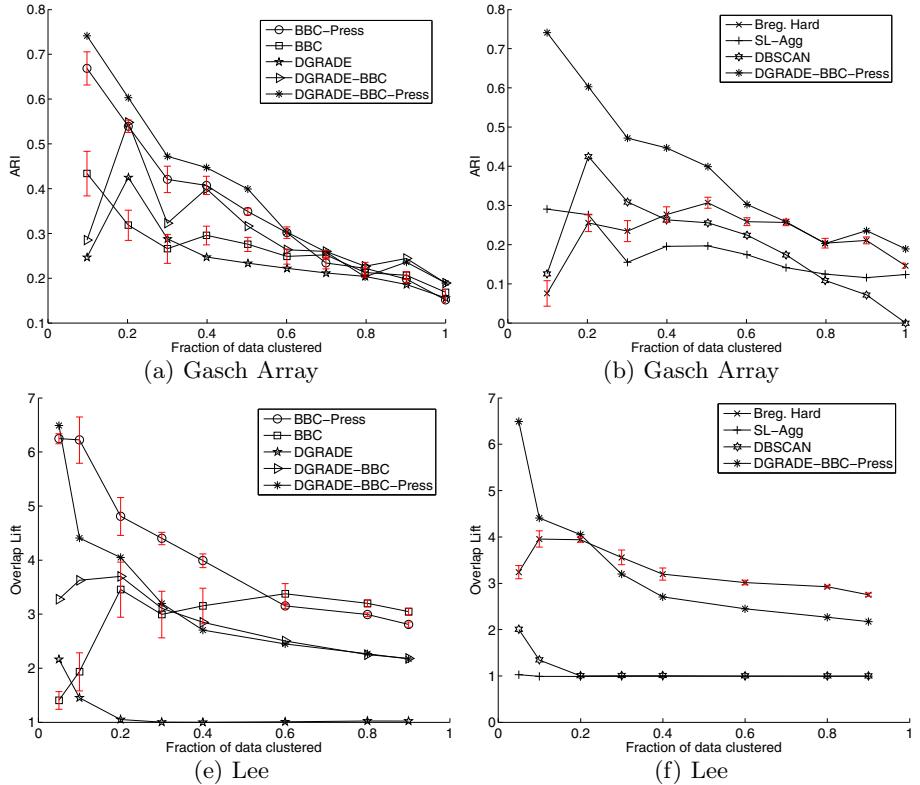


Fig. 11. Evaluation of BBC seeded with DGRADE on two real datasets as compared to BBC without seeding, and against the three other methods. Performance was measured using ARI for Gasch Array and using Overlap Lift for the Lee dataset. Results for Bregman Hard clustering and BBC without seeding were averaged over 20 and 10 trials for Gasch and Lee respectively, and the corresponding one std. dev. error-bars are plotted (but are sometimes too small to be visible). Note: These experiments were setup differently from those without DGRADE; the k output by DGRADE was used as input to all algorithms except DBSCAN.

high-dimensional real datasets- the Gasch Array (figures 11(a) and (b)). However, this relationship is only seen for the lowest coverage of 0.05 on the Lee dataset (Figure 11(e)), perhaps because the fraction of genes that are usually dense when clustering genes is usually very small.

11 Concluding Remarks

Bregman Bubble Clustering extends the notion of “density-based clustering” to a large class of divergence measures, and is perhaps the first that uses a local search/parametric approach in such settings. The availability of appropriate

Bregman Divergences²¹ for a variety of problem domains where finding dense clusters is important, opens density-based clustering to many new domains. Moreover, the extension of BBC to Pearson Correlation (Pearson Distance) is particularly helpful for analyzing several biological datasets. Bregman Bubble Clustering can also be thought of as a conceptual bridge between partitional clustering algorithms and the problem of One Class Clustering. The Soft BBC model shows that BBC arises out of a more fundamental model involving a mixture of exponentials and a uniform background.

Empirical results show that BBC-Press gives good results on a variety of problems involving both low and high-dimensional feature spaces, often outperforming other alternatives by large margins. DGRADE provides effective seeding for BBC and BBC-Press, and provides an additional mechanism for indicating the appropriate number of dense clusters that one should seek. Overall, the combination of the three components; BBC, Pressurization, and DGRADE provides a robust framework for finding dense clusters with several key properties: deterministic results, reasonable scalability to large, high-dimensional datasets, and applicability to a wide variety of problem domains.

There are several properties of the Bregman Bubble Clustering framework that can be used to further expand the robustness and quality of clustering. We conclude this chapter by outlining two promising extensions:

Bregman Bubble Co-clustering: A generalized framework called Bregman Co-clustering was proposed by [54] that unifies many different *checkerboard* co-clustering algorithms, where the clusters are defined not just on the rows (data objects) but also simultaneously on the columns (features) when the dataset is viewed as a matrix. If one rearranges the rows and columns of this matrix such that rows/cols belonging to the same cluster are contiguous, then the permuted matrix shows checkerboard pattern of non-overlapping co-clusters. However, often one desires to co-cluster only parts of the data, and also to allow for co-clusters that overlap. For example, genes' interactions in gene-expression data are often overlapping. Such data also are typically highly skewed - the number of experiments is typically a few dozens or hundreds, while the number of genes is at least an order of magnitude more. For such data, it is better to prune a large number of genes before applying co-clustering. Since Bregman Bubble Clustering is a generalization of Bregman Clustering, and since Bregman Clustering is a component/step in Bregman Co-clustering, it is possible to modify Bregman Co-clustering to use Bregman Bubble Clustering, and to obtain a co-clustering algorithm that can find dense co-clusters. Such an algorithm also makes it possible to find overlapping dense co-clusters. This approach has been successfully applied to gene-expression data with results superior to a host of alternatives [15, 23].

²¹ E.g. Itakura-Saito for voice identification, Mahalanobis distance for digital Mammography, and KL-divergence for identifying most relevant documents.

Online Bregman Bubbles for detecting non-stationary dense clusters: OC-IB (Section 2.3) can be considered as a randomized equivalent of BBC-Q for $k = 1$ (Section 3.5). It is also possible to derive a k -class randomized version of the algorithm for BBC-Q or BBC-S (Section 3.4 and 3.5). This could be useful in online settings, e.g. for modeling dense clusters in scenarios where there is substantial *concept drift*, i.e. the distribution that the data is coming from is not stationary, but drifting gradually. With an online version of BBC, the dense cluster centers could move as the modes in the data shift. One application could be for modeling highly coherent sets of customers in market-basket data. Such customer groupings tend to shift slowly over time [30]. Another application could be to track rare but recurring non-stationary anomalies; as old anomalies stop appearing or shift, the online Bregman Bubble model could adapt quickly because of its localized nature. One area where such anomalous data tends to shift rather than randomly appear is in online fraud, where lots of very similar fraudulent content originates from a single individual, and the fraudsters tend to modify keywords only slightly to try to evade the detection system. Using previous and newly identified fraudulent data points as cluster centers, such similar but constantly drifting patterns could perhaps be better tracked using online BBC. Yet another application could be to follow visual movements of dense patterns In low-dimensional data: such as “blobs” in infra-red images of objects in a military, wildlife, or an oceanic setting, or as a part of an airport security system at an airport.

Acknowledgments. This research was supported by NSF grants IIS-0713142 and IIS-1017614. We are also grateful to Srujana Merugu and Arindam Banerjee for some useful discussions.

References

1. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering points to identify the clustering structure. In: Proc. ACM SIGMOD, pp. 49–60 (1999)
2. Arabie, P., Carroll, J.D., DeSarbo, W., Wind, J.: Overlapping clustering: A new method for product positioning. Journal of Marketing Research 18(3), 317–319 (1981)
3. Banerjee, A., Merugu, S., Dhillon, I., Ghosh, J.: Clustering with Bregman divergences. JMLR 6, 1705–1749 (2005)
4. Banerjee, A., Krumpelman, C., Ghosh, J., Basu, S., Mooney, R.J.: Model-based overlapping clustering. In: Proc. KDD 2005, Chicago, Illinois, USA, pp. 532–537 (2005)
5. Banerjee, A., Langford, J.: An objective evaluation criterion for clustering. In: KDD 2004, Seattle, Washington, USA (August 2004)
6. Battle, A., Segal, E., Koller, D.: Probabilistic discovery of overlapping cellular processes and their regulation. In: Eighth Annual International Conference on Research in Computational Molecular Biology (RECOMB 2004) (April 2004)
7. Berchtold, S., Keim, D.A., Kriegel, H.P.: The X-Tree: An index structure for high-dimensional data. In: Proceedings of the 22nd International Conference on Very Large Databases, pp. 28–39. Morgan Kaufmann Publishers, San Francisco (1996)

8. Buzo, A., Gray, A.H., Gray, R.M., Markel, J.D.: Speech coding based on vector quantization. *IEEE Transactions on Acoustics, Speech and Signal Processing* 28(5), 562–574 (1980)
9. Casella, G., Robert, C.P., Wells, M.T.: Mixture models, latent variables and partitioned importance sampling. Technical Report, RePEc:fth:inseep:2000-03, Institut National de la Statistique et des Etudes Economiques (2003),
<http://ideas.repec.org/p/fth/inseep/2000-03.html>
10. Chakaravathy, S.V., Ghosh, J.: Scale based clustering using a radial basis function network. *IEEE Transactions on Neural Networks* 2(5), 1250–1261 (1996)
11. Cheng, Y.: Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* 17(8), 790–799 (1995)
12. Crammer, K., Chechik, G.: A needle in a haystack: Local one-class optimization. In: ICML 2004, Banff, Alberta, Canada (2004)
13. Crammer, K., Singer, Y.: Learning algorithms for enclosing points in Bregmanian spheres. In: COLT 2003, pp. 388–402 (2003)
14. Deodhar, M., Ghosh, J.: Simultaneous co-clustering and modeling of market data. In: Workshop for Data Mining in Marketing (DMM 2007). IEEE Computer Society Press, Leipzig (2007)
15. Deodhar, M., Ghosh, J., Gupta, G., Cho, H., Dhillon, I.: A scalable framework for discovering coherent co-clusters in noisy data. In: Bottou, L., Littman, M. (eds.) Proceedings of the 26th International Conference on Machine Learning, Omnipress, Montreal, pp. 241–248 (June 2009)
16. Dettling, M., Bühlmann, P.: Supervised clustering of genes. *Genome Biol.* 3(12) (2002)
17. Dhillon, I., Mallela, S., Kumar, R.: A divisive information-theoretic feature clustering algorithm for text classification. *JMLR* 3, 1265–1287 (2003)
18. Dhillon, I.S., Guan, Y., Kogan, J.: Refining clusters in high-dimensional text data. In: 2nd SIAM International Conference on Data Mining (Workshop on Clustering High-Dimensional Data and its Applications) (April 2002)
19. Dhillon, I.S., Modha, D.S.: Concept decompositions for large sparse text data using clustering. *Machine Learning* 42(1-2), 143–175 (2001)
20. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. KDD 1996, pp. 226–231 (1996)
21. Gasch, A.P., et al.: Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Bio. of the Cell* 11(3), 4241–4257 (2000)
22. Georgescu, B., Shimshoni, I., Meer, P.: Mean shift based clustering in high dimensions: A texture classification example. In: ICCV 2003: Proceedings of the Ninth IEEE International Conference on Computer Vision. pp. 456–463. IEEE Computer Society, Washington, DC, USA (2003)
23. Ghosh, J., Deodhar, M., Gupta, G.: Detection of Dense Co-clusters in Large, Noisy Datasets. In: Wang, P.S.P. (ed.) *Pattern Recognition and Machine Vision (in memory of Professor King-Sun Fu)*, pp. 3–18. River Publishers, Aalborg (2010)
24. Gollub, J., et al.: The Stanford Microarray Database: data access and quality assessment tools. *Nucleic Acids Res.* 31, 94–96 (2003)
25. Guha, S., Rastogi, R., Shim, K.: Rock: A robust clustering algorithm for categorical attributes. In: 15th International Conference on Data Engineering, Sydney, Australia, p. 512 (1999)
26. Gupta, G.: Robust methods for locating multiple dense regions in complex datasets. PhD Thesis, University of Texas at Austin (December 2006)

27. Gupta, G., Ghosh, J.: Robust one-class clustering using hybrid global and local search. In: Proc. ICML 2005, Bonn, Germany, pp. 273–280 (August 2005)
28. Gupta, G., Ghosh, J.: Bregman Bubble Clustering: A robust framework for mining dense clusters. Tech Report, Dept. of Elec. & Comp. Engineering, University of Texas at Austin. IDEAL-TR04 (September 2006),
<http://www.lans.ece.utexas.edu/techreps.html>
29. Gupta, G., Liu, A., Ghosh, J.: Automated Hierarchical Density Shaving: A robust, automated clustering and visualization framework for large biological datasets. IEEE Trans. On Comp. Bio. and Bioinformatics (TCBB) 7(2), 223–237 (2010)
30. Gupta, G.K.: Modeling Customer Dynamics Using Motion Estimation in A Value Based Cluster Space for Large Retail Data-sets. Master's thesis, University of Texas at Austin (August 2000)
31. Gupta, G.K., Ghosh, J.: Detecting seasonal trends and cluster motion visualization for very high dimensional transactional data. In: Society for Industrial and Applied Mathematics (First International SIAM Conference on Data Mining (SDM 2001)) (April 2001)
32. Gupta, G.K., Ghosh, J.: Value Balanced Agglomerative Connectivity Clustering. In: SPIE conference on Data Mining and Knowledge Discovery III, Orlando, Florida. SPIE Proc. vol. 4384, pp. 6–15 (April 2001)
33. Hastie, T., et al.: Gene shaving as a method for identifying distinct sets of genes with similar expression patterns. Genome Biology 1, 1–21 (2000)
34. Hendrickson, B., Leland, R.: An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM Journal on Scientific Computing 16(2), 452–469 (1995)
35. Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification, 193–218 (1985)
36. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs (1988)
37. Jiang, D., Pei, J., Ramanathan, M., Tang, C., Zhang, A.: Mining coherent gene clusters from gene-sample-time microarray data. In: KDD 2004, Seattle, WA, USA, pp. 430–439 (2004)
38. Jiang, D., Pei, J., Zhang, A.: DHC: A density-based hierarchical clustering method for time series gene expression data. In: BIBE 2003, p. 393. IEEE Comp. Soc., Washington, DC, USA (2003)
39. Johnson, S.C.: Hierarchical clustering schemes. Psychometrika 32(3), 241–254 (1967)
40. Judd, A., Hovland, M.: Seabed Fluid Flow: The Impact of Geology, Biology and the Marine Environment. Cambridge University Press, Cambridge (2007)
41. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal of Scientific Computing 20(1), 359–392 (1998)
42. Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S.: Multilevel hypergraph partitioning: Applications in VLSI domain. In: Design and Automation Conference (1997)
43. Kearns, M., Mansour, Y., Ng, A.Y.: An information-theoretic analysis of hard and soft assignment methods for clustering. In: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, pp. 282–293. AAAI, Menlo Park (1997)
44. Kriegel, H.P., Pfeifle, M., Pötke, M., Seidl, T.: The paradigm of relational indexing: A survey. In: BTW. LNI, vol. 26. GI (2003)
45. Lang, K.: Newsweeder: Learning to filter netnews. In: Proceedings of the Twelfth International Conference on Machine Learning, pp. 331–339 (1995)

46. Lazzeroni, L., Owen, A.B.: Plaid models for gene expression data. *Statistica Sinica* 12(1), 61–86 (2002)
47. Lee, I., Date, S.V., Adai, A.T., Marcotte, E.M.: A probabilistic functional network of yeast genes. *Science* 306, 1555–1558 (2004)
48. Linde, Y., Buzo, A., Gray, R.M.: An algorithm for vector quantizer design. *IEEE Transactions on Communications* 28(1), 84–95 (1980)
49. Linden, G., Smith, B., York, J.: Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7(1), 76–80 (2003)
50. Long, B., Zhang, Z.M., Wu, X., Yu, P.S.: Relational clustering by symmetric convex coding. In: ICML 2007, pp. 569–576. ACM, New York (2007)
51. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
52. Mansson, R., Tsapogas, P., Akerlund, M., et al.: Pearson correlation analysis of microarray data allows for the identification of genetic targets for early b-cell factor. *J. Biol. Chem.* 279(17), 17905–17913 (2004)
53. McGuire, A.M., Church, G.M.: Predicting regulons and their cis-regulatory motifs by comparative genomics. *Nucleic Acids Research* 28(22), 4523–4530 (2000)
54. Merugu, S.: Privacy-preserving distributed learning using generative models. PhD Thesis, The University of Texas at Austin (August 2006)
55. Neal, R., Hinton, G.: A view of the EM algorithm that justifies incremental, sparse, and other variants. In: Jordan, M.I. (ed.) *Learning in Graphical Models*, Kluwer, Dordrecht (1998),
<http://www.cs.toronto.edu/char126radfordem.abstract.html>
56. Pietra, S.D., Pietra, V.D., Lafferty, J.: Duality and auxiliary functions for Bregman distances. Technical Report CMU-CS-01-109, School of Computer Science. Carnegie Mellon University (2001)
57. Schmid, C., Sengstag, T., Bucher, P., Delorenzi, M.: MADAP, a flexible clustering tool for the interpretation of one-dimensional genome annotation data. *Nucleic Acids Res.*, W201–W205 (2007)
58. Schölkopf, B., Burges, C., Vapnik, V.: Extracting support data for a given task. In: KDD. AAAI Press, Menlo Park (1995)
59. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.S., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. *Neural Computation* 13(7), 1443–1471 (2001)
60. Segal, E., Battle, A., Koller, D.: Decomposing gene expression into cellular processes. In: 8th Pacific Symposium on Biocomputing (PSB), Kaua'i (January 2003)
61. Sharan, R., Shamir, R.: Click: A clustering algorithm with applications to gene expression analysis. In: Proc. 8th ISMB, pp. 307–316 (2000)
62. Slonim, N., Atwal, G.S., Tkacik, G., Bialek, W.: Information-based clustering. *PNAS* 102(51), 18297–18302 (2005)
63. Strehl, A., Ghosh, J.: Value-based customer grouping from large retail data-sets. In: SPIE Conference on Data Mining and Knowledge Discovery: Theory, Tools, and Technology II, Orlando, Florida, USA, April 24–25, vol. 4057, pp. 33–42. SPIE (2000)
64. Strehl, A., Ghosh, J.: Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing* 15(2), 208–230 (2003)
65. Strehl, A., Ghosh, J., Mooney, R.J.: Impact of similarity measures on web-page clustering. In: AAAI Workshop on AI for Web Search (AAAI 2000), pp. 58–64. AAAI/MIT Press (July 2000)

66. Tax, D., Duin, R.: Data domain description using support vectors. In: Proceedings of the ESANN 1999, pp. 251–256 (1999)
67. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. *Science* 290, 2319–2323 (2000)
68. Ueda, N., Nakano, R.: Deterministic annealing EM algorithm. *Neural Networks* 11(2), 271–282 (1998)
69. Wedel, M., Steenkamp, J.: A clusterwise regression method for simultaneous fuzzy market structuring and benefit segmentation. *Journal of Marketing Research*, 385–396 (1991)
70. Wishart, D.: Mode analysis: A generalization of nearest neighbour which reduces chaining effects. In: Proceedings of the Colloquium in Numerical Taxonomy, pp. 282–308. Academic Press, University of St. Andrews, Fife, Scotland (1968)
71. Yun, C.H., Chuang, K.T., Chen, M.S.: An efficient clustering algorithm for market basket data based on small large ratios. In: Computer Software and Applications Conference 2001, pp. 505–510 (2001)
72. Zhong, S.: Efficient streaming text clustering. Special issue IJCNN 2005: Neural Networks 18(5-6), 790–798 (2005)

Chapter 8

DepMiner: A Method and a System for the Extraction of Significant Dependencies

Rosa Meo¹ and Leonardo D'Ambrosi²

¹ University of Torino, Italy

² Regional Agency for Health Care Services - A.Re.S.S. Piemonte, Italy

Abstract. We propose *DepMiner*, a method implementing a simple but effective model for the evaluation of itemsets, and in general for the evaluation of the dependencies between the values assumed by a set of variables on a domain of finite values. This method is based on Δ , the departure of the probability of an observed event from a referential probability of the same event. The observed probability is the probability that the variables assume in the database given values; the referential probability, is the probability of the same event estimated in the condition of maximum entropy.

DepMiner is able to distinguish between dependencies among the variables intrinsic to the itemset and dependencies “inherited” from the subsets: thus it is suitable to evaluate the utility of an itemset w.r.t. its subsets. The method is powerful: at the same time it detects significant positive dependencies as well as negative ones suitable to identify rare itemsets. Since Δ is anti-monotonic it can be embedded efficiently in algorithms. The system returns itemsets ranked by Δ and presents the histogram of Δ distribution. Parameters that govern the method, such as minimum support for itemsets and thresholds of Δ are automatically determined by the system. The system uses the thresholds for Δ to identify the statistically significant itemsets. Thus it succeeds to reduce the volume of results more than competitive methods.

1 Introduction

In statistics, machine learning and data mining the problem of the determination of set of variables whose values are correlated represents an important knowledge for the user in many fields such as in feature selection, database design and schema reverse engineering, market basket analysis, information retrieval, machine translation, biology, etc. Often in the scientific literature, the study of the dependence between variables is limited to pairs [9,19]. Much previous research focused on finding correlated pairs but finding correlations among more than two variables is essential for problems in many commercial and sociological studies (e.g., for collaborations and interaction networks), medical and biological (e.g., interaction among drugs and proteins) and scientific domains. Thus, instead of correlated items we should find correlated itemsets in which all items are correlated with each other.

In practical cases, often it happens that the set of returned itemsets is large and much of the information is redundant because many itemsets are returned together with many

of their subsets. The attempt to reduce the redundancy in the result set answers to two major challenges in frequent-pattern mining: the first is to reduce the often overwhelming size of the mining results and the other is to eliminate redundancy in the information content and the overlapping between the itemsets.

Deciding which itemsets are redundant is not easy and straightforward. It might depend on the applications. For instance, the inclusion in the set of more itemsets with some common items could be acceptable because the itemsets might have different meaning. Instead, the inclusion in the result of both subsets and their supersets is not acceptable if the supersets do not add new information to the information carried by the subsets. In literature, redundant itemsets are detected in many different and sometimes opposite ways. For instance, [25] considers the correlation among the items as strong only when all the items are considered together and when removing any items greatly reduces the correlation. Therefore, the subsets of these itemsets must have instead a weak correlation. On the opposite side, [5] considers interesting an itemset if all its subsets are closely related to all other subsets, and no irrelevant items can be removed. This kind of approach is often adopted in feature selection by step-wise, forward methods [6,12].

In data mining there exist computationally efficient methods to discover significant dependencies and correlations among the items of a frequent pattern [1,3,16]. In order to determine the dependencies in k -itemsets with $k > 2$, either they make the multi-way independence assumption or they evaluate the contribution to the overall itemset of each variable separately [7,23,25]. The difficulty stems from the fact that there is not an easy way to determine a referential probability of an itemset I that represents a condition of independence among the subsets if we do not suppose independence among all the single variables in I . But the multi-way independence condition gives a problem: according to this definition of independence, if a dependence already exists in a subset of I , this dependence is “inherited” from the subset to I and to all the supersets of I [3]. Thus we do not have a way to distinguish if an intrinsic dependence exists in an itemset I in addition to the dependencies inherited from its subsets.

We can solve the problem in terms of quantity of information that an itemset provides: we are interested only in itemsets that add any information to their subsets. If instead, an itemset can be foreseen given the observation of its subsets, it does not carry any new information in addition to the subsets; therefore it can be considered as redundant and it is not interesting. We proposed in [13] a solution based on the maximum entropy. The entropy of an itemset I is computed by an estimation of the probability of I computed on the basis of the probability of its subsets. The probability of I at which the entropy is maximum (denoted by $P_E(I)$) corresponds to the probability that the itemset I would have in the condition in which it carries the maximum amount of information in addition to its subsets. The interest measure that we proposed for an itemset I is the departure of the probability of I w.r.t. the referential value computed at maximum entropy: $\Delta(I) = P(I) - P_E(I)$. The more the departure between the two probabilities, the less the itemset can be correctly foreseen from the observation of its subsets. This departure identifies a dependence between the items and tells us that this dependence is not due to the subsets only. As a consequence the itemset represents a non redundant itemset that must be included in the result. In Section 3 we summarize how Δ is computed.

$\Delta(I)$ decreases with the increase in the cardinality of itemsets. As a consequence, Δ is not a suitable measure to compare itemsets of different cardinality. For this purpose in this chapter we propose Δ_n , a version of Δ normalized w.r.t. the probability of the itemset:

$$\Delta_n(I) = \frac{P(I) - P_E(I)}{P(I)}$$

$\Delta_n(I)$ takes both positive and negative values, in the range from $[-\infty, 1]$. Specifically, if the value is positive, it means a positive dependence, i.e., an itemset that is more frequent than expected; if the value is negative it means a negative dependence, i.e., an itemset that occurs rarer than expected.

In this chapter we present a method for the computation of the interesting and non redundant itemsets based on the above observations. $\Delta_n(I)$ is used as a score function to rank the itemsets. In Section 4 we show how we succeeded to determine the significance level of $\Delta_n(I)$ and to point out to the user the significant itemsets.

Δ computation occurs from an initial, intermediate result composed of frequent itemsets. Another contribution of this chapter is to show how the minimum frequency threshold can be set without the explicit intervention of the user. In fact, the determination of the frequency threshold is well-known to be difficult for the user.

The rest of the chapter is organized as follows. In Section 2 we review the related works. In Section 3 we summarize how Δ is computed. Section 4 shows how to determine the significance level of $\Delta_n(I)$. Though $\Delta_n(I)$ does not satisfy the anti-monotonicity property, in Section 5 we prove an alternative property that guarantees that $\Delta_n(I)$ can be computed efficiently in algorithms as it were anti-monotone. In Section 6 we show how the system determines the minimum support threshold. Section 7 describes the system implementation and the computation flow. Section 8 presents an empirical evaluation study on the results of the system on some common datasets. In this Section DepMiner is compared with other methods for the evaluation of the itemsets, such as [4,7]. The obtained results show that DepMiner identifies the dependencies overcoming the restrictive assumptions of multi-way independence and that the identified significant itemsets are a little portion of the results returned by the other methods. This means that DepMiner is able to compress much without discarding any significant itemsets. Finally, Section 9 draws the conclusions.

2 Related Work

In statistics the problem of the determination of dependent variables is a classical, traditional problem and it has been solved in many ways. The most common approaches are derived by the statistical hypothesis tests such as the tests based on the χ^2 statistics, the Fisher's exact tests [22] and Likelihood-ratio tests [5].

[9] is a deep study on the association between categorical variables and proposes a survey on the measures of association between variables. In machine learning the discovery of dependencies in a multivariate problem (structure learning) is solved by application of neural networks or Bayesian learning methods like MCMC [2] which are NP-hard problems in the number of variables.

In data mining there exist methods to discover significant dependencies and correlations between the items of a frequent pattern with computationally efficient algorithms. [3] is one of the first attempts to discover significant association rules (called dependence rules) by means of the χ^2 test. They point out that the satisfaction of the dependence condition is down-ward closed, i.e., it is monotonic. In other words, once that the dependence is raised in an itemset it will be raised in all the supersets of the itemset. Thus, if the test on the existence of dependencies is checked by a test based on χ^2 , the test is not sensible to the addition of further items to the initial dependent set, either the items are independent or not. Thus, χ^2 does not appear to be a suitable measure to determine the effective contribution of an item to the dependence.

[1] proposes collective strength as an itemset interest measure. Collective strength makes use of the ratio between the probability that the itemset appears w.r.t. the expected one under the hypothesis of multiway independence among the items. In turn this ratio is compared with the analogous ratio computed between the probability that at least one item violates the itemset and the expected probability of the same event under the condition of independence. There is violation of the itemset when there is at least one of its items that appears separately w.r.t. the other items in the set.

[16] proposes other measures based on the minimum and maximum confidence that would be obtained by the association rules generated from the itemset. Furthermore, it proposes bond as a further, more restrictive measure defined as the ratio of the number of occurrences of the itemset and of any of its items. Notice, that, in order to determine the dependencies in k -itemsets with $k > 2$, the most of these approaches make the multi-way independence assumption or they evaluate the contribution to the overall itemset of each variable separately.

[7] ranks the frequent itemsets according to the unlikelihood that they appear under the hypothesis that all the items in the itemset are independent. [23] has the aim to rank the most relevant itemsets (by a suitable measure application dependent) and selects the significant portion of the ranking based on measures of lack of redundancy. According to this study, significance and relevance are strictly connected and are combined in a unique measure.

[4] is one of the most well-known studies that allows the reduction of the number of the itemsets in the result; it allows a lossless compression of the result because all the information on the frequent itemsets lacking from the result can be restored from the result set. A similar aim as regards to the reduction in the number of returned itemsets is proposed by [18] with a criterion based on minimum description length. Each itemset is used to represent the portion of the database in which it occurs and therefore it compresses it. Then, the resulting set of itemsets is interesting if it yields a good and lossless compression of the database. Another work on pattern summarization is [24] in which the authors define a proximity metric between the patterns according to the overlapping between the portions of database in which they occur.

[25] proposes the use of multi-information, an extension of mutual information to more than two variables. It considers an itemsets as correlated only if multi-information is higher than a given threshold and if any proper subset has instead a weak amount of multi-information. In multi-information the contribution of each single item to the itemset is considered. A similar approach is proposed for feature selection in classification

by [6,12]. On the opposite side, [11] proposes to use entropy and the quantity of information lead by a set of features for the identification of the set of k features that are maximally independent. Maximum entropy is seen as a guarantee of lack of redundancy in the set. The aim of this kind of itemset is at optimising the independence of items within the set. According to this approach, any single feature is added to the set only if it provides an additional distinctive power.

On the contrary, [5] proposes the following criterion of “fully-correlation” for the determination of the interest of an itemset: an itemset is fully-correlated if all its subsets are closely related to all other subsets, and no irrelevant items can be removed from the set. As regards to the selection criterion, it selects only the maximal fully-correlated itemsets in the following way: if there is no other item that can be added to the itemset to generate a new fully-correlated itemset, then the itemset is maximal fully-correlated.

[10] deals with diversity measures used as heuristic measures of interest for ranking summaries generated from a single dataset. Summaries are composed by a set of attribute-value pairs where attributes can be generalized to many levels of granularity according to taxonomic hierarchies.

A similar approach to the method proposed here and based on Δ , is proposed in [20] in which the analysis is based on K-L divergence.

3 Estimation of the Referential Probability

Suppose itemset $I = \{i_1, i_2, \dots, i_k\}$.

Entropy $H(I) = -\sum P(i_1^*, i_2^*, \dots, i_k^*) \log[P(i_1^*, i_2^*, \dots, i_k^*)]$ where we denote by i_j^* the item i_j taken affirmed or negated. Summation ranges over the probabilities of all the combinations of the k items taken affirmed or negated. $H(I)$ is not computed by assumption that singletons are independent but taking in consideration the actual probability of occurrence of each subset of I , as observed from the database. The exclusion-inclusion principle [4] is adopted to compute the entropy of I starting from the probability of the subsets of I . Thus, if the dependence in an itemset I is intrinsic, due to the synergy between all its items, then its probability departs with respect to its estimate given only on the basis of the observed probabilities of its subsets. As a result, thanks to $\Delta_n(I)$ we make emerge the intrinsic, actual dependencies, existing among all the items in I .

4 Setting a Threshold for Δ

Another problem that we have to solve is how large must be Δ_n such that an itemset is deemed significant. We use a null model in which there are not dependencies between the variables. The null model is generated empirically via a randomization of the original dataset. Randomization is generally accepted as a way to allow a statistical test on significance of results [8]. Randomization occurs by independently shuffling the variable values among the examples. As a result, the new dataset will have the same marginal probabilities of the single variables but the dependencies between them are spoiled.

Delta/PO	Delta	Frequenza	Dipendenze (attributo=valore)
0,0188738099	0,0075643925	(3256)	Class=poisonous ring-number=one bruises?=no
0,0138819911	0,0055910727	(3272)	stalk-surface-above-ring=smooth ring-number=one Class=edibility
0,0121980129	0,0045524834	(3032)	stalk-surface-below-ring=smooth ring-number=one Class=edibility
0,0055816795	0,0023195162	(3376)	stalk-surface-above-ring=smooth Class=edibility gill-size=broad
0,0007870802	0,0003115768	(3216)	Class=edibility gill-size=broad odor=None
-0,0006493766	-0,0002500308	(3128)	bruises?=bruises gill-spacing=close stalk-surface-above-ring=smooth
-0,0014753219	-0,0006770064	(3728)	veil-color=white Class=edibility gill-size=broad
-0,0020780051	-0,0008000985	(3128)	veil-color=white ring-type=pendant gill-size=broad
-0,0026961403	-0,0013155465	(3964)	gill-attachment=free stalk-surface-above-ring=smooth stalk-surface-below-ring=smooth
-0,0049992833	-0,002146418	(3488)	gill-attachment=free ring-number=one Class=edibility
-0,005177405	-0,002146418	(3368)	gill-attachment=free ring-number=one ring-type=pendant
-0,006102868	-0,0022656635	(3016)	stalk-surface-above-ring=smooth ring-type=pendant gill-size=broad

Fig. 1. Screen-shot: itemsets ranking with significant itemsets in green (*Mushroom*)

Broadly speaking, as a successive step (without discussing the optimizations that will be described in Section 5), we compute itemsets both from the real and the randomized data. (Figure 4 shows the computation flow described in detail in Section 7). Next, we compute the minimum negative value of Δ_n and the maximum positive value of Δ_n in randomized data. Then, we will use the minimum value of Δ_n in randomized data as an upper bound (denoted by UB) for rarer itemsets in real data and use the maximum value of Δ_n in randomized data as a lower bound (denoted by LB) for the more frequent itemsets in real data. This is a sort of statistical test on Δ_n and accept as dependent an itemset if its Δ_n is higher (resp. lower) than the maximum (resp. minimum) Δ_n of the itemsets extracted from the randomized data.

An intuition behind the statistical test is the following. Given the high number of itemsets extracted from randomized data (usually, in the number of hundreds), if none

of the itemsets has reached a so high (resp. low) value of Δ_n it means that it is unlikely that the observed value (and the correspondent itemset) occurs for chance in real data - as it happens indeed in randomized data. Otherwise, if the itemset had occurred for chance also in real data, then the observed values of Δ_n , in real and randomized data, would have been much more similar! If instead a value of Δ_n occurred in real data so departed from the observed values in random data, this constitutes an evidence of the fact that the itemset did not occur for chance. Therefore, an intrinsic dependence exists in that itemset and makes it emerge above the others. Thus the maximum value of Δ_n observed in randomized data constitutes a lower bound of accepted values in real data. Similarly, for the minimum (negative) value.

Consider the dataset *Mushroom*. After randomization, we observed the maximum value of $\Delta_n = 0.04$ while the minimum value is $\Delta_n = -0.03$. In real data, the maximum is $\Delta_n = 0.85$ and the minimum is $\Delta_n = -0.45$. Thus it is evident that in *Mushroom* the positive dependencies are more abundant and more marked while the negative dependencies are few and less evident. In Figure 1 we show one screen shot of our system prototype, *DepMiner*, with the ranking of itemsets extracted from *Mushroom*. In green the significant itemsets are shown (i.e., the itemsets with a value of Δ_n exceeding the observed range of values in randomized data). In yellow, instead, the other itemsets. Notice that both rarer and more frequent itemsets are interesting. Thus, *DepMiner* is not biased toward frequent or rare itemsets as it happens for many other systems of pattern mining.

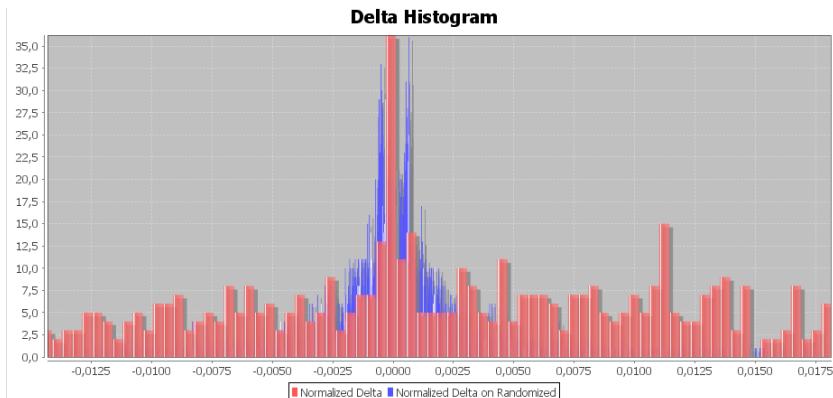


Fig. 2. Histograms of Delta on *Mushroom* (in red) and on its randomization (in blue)

5 Embedding Δ_n in Algorithms

A third problem is how to embed Δ_n in the algorithms. In fact Δ_n does not satisfy an anti monotonicity property that is useful to make efficient the exploration and pruning of the search space of the itemsets. We solved the problem by discovery of the following property.

Theorem 1. Let minsup be the minimum frequency threshold set in the FIMI (Frequent Itemset Mining) algorithm. Let Δ_{nu} denote the upper bound of Δ_n from the randomized data and Δ_{nl} the lower bound.

$LB = \Delta_{nu} \cdot \text{minsup}$ is the minimum threshold for positive dependencies while $UB = \Delta_{nl} \cdot \text{minsup}$ is the maximum threshold for negative dependencies.

While traversing deeper the item-trie containing candidate itemsets it is sufficient to compute Δ_n for an itemset I if:

$\Delta(I) > LB$ or $\Delta(I) < UB$. Otherwise, we can prune I and its children from the item-trie.

Proof. Let be C a child of itemset I . From [4,17] it results that $\Delta(I) \geq \Delta(C)$ since Δ is contained in the range of values of probabilities of any non derivable itemset (if it was derivable it had $\Delta = 0$).

$\Delta(I) < LB$ can be rewritten as $\Delta(I) = \Delta_n(I) \cdot P(I) < LB = \Delta_{nu} \cdot \text{minsup}$. We obtain $\Delta_n(I) < \Delta_{nu} \cdot \frac{\text{minsup}}{P(I)}$. Since $P(I) \geq \text{minsup}$ (otherwise we would have pruned earlier I from the item-trie) it results that $\Delta_n(I) < \Delta_{nu}$. As a consequence, I can be pruned. In addition, since $\Delta(C) \leq \Delta(I) < LB$ we can prune C too. Similar reasoning applies to the other bound.

6 Determination of the Itemsets Minimum Support Threshold

The new model for the determination of the itemsets minimum support allows the replacement of the minimum support threshold imposed as a requirement by the user. As we know, that the minimum support threshold presents some drawbacks due to the fact that a fixed and unique value of support threshold for all the sets (that does not depend on the cardinality of the sets or on their probability density function) is not realistic. Another problem is the fact that the support threshold is given by the user who may not know how to set it. On the contrary, he/she may know how greater is the accuracy of the measure (or the error he/she wants to allow in the inference of the probability of the itemset from the sample database). Therefore, in DepMiner we allow the user to set a different minimum threshold of probability for each itemset: this decision is taken on the basis of the estimated probability of occurrence of the itemset (from the principle of maximum likelihood) and of an error tolerance in this estimation given by the user.

We judge the relevance of the information obtained from the database with a criterion based on the Bayes' Theorem. This one allows us to make an estimate of the probability distribution function of an itemset (a priori probability) starting from the set of observations obtained by the sample database (a posteriori probability). Our criteria is very simple. We consider the probability of an itemset a random variable and make an estimate of this probability on the basis of the observations (obtained a posteriori) from the database. If the most likely value of probability of occurrence of an itemset in the database gives values whose confidence interval width is comparable with the error allowed by the user, we can conclude that the probability estimation is not reliable.

In [15] we proved that this reliability property is, as the itemset support, a property that is anti-monotone, and allows us to stop the lattice traversal in depth, in practice in an equivalent manner as the itemset support.

We apply the theory for the inference of the proportion $\frac{K}{N}$ that gives the more likely value of the probability of an itemset. Since in data mining the sample size is always big, we can approximate the binomial distribution, that is the probability distribution function for the proportion estimation, with a normal distribution with mean $\frac{K}{N}$ and variance $\frac{K}{N}(1 - \frac{K}{N})$. The theory of the confidence interval for the proportion gives the following formula:

$$p_o - Z\sqrt{\frac{p_o(1 - p_o)}{N}} \leq p \leq p_o + Z\sqrt{\frac{p_o(1 - p_o)}{N}} \quad (1)$$

where we denote by p the real probability of the itemset, by p_o the observed proportion (a posteriori observation) that constitutes the estimation on the sample of this probability, by N the sample size and by Z the critical value in the normal distribution corresponding to the confidence level imposed by the user (usually denoted by α). The usual values of Z are 1.96 or 2.58 corresponding to a probability of making an erroneous inference on the proportion with a dataset composed by a collection of random and independent samples equal respectively to 0.05 and 0.01. From the theory, we have that W , the width of the confidence interval, is the maximum error in the estimation of the proportion, and is given by:

$$W = 2Z\sqrt{\frac{p_o(1 - p_o)}{N}} \quad (2)$$

Since we allow the user to set both Z and the maximum relative error in the probability estimation that she wants to allow, the relative error is given by $e_r = \frac{W}{p_o}$ and results equal to:

$$e_r = \frac{2Z}{\sqrt{N}}\sqrt{\frac{(1 - p_o)}{p_o}} \quad (3)$$

Range of observable probabilities with a relative error. e_r is a monotonic decreasing function with the observable probabilities p_o . It means that we can set a certain value of e_r that is the error tolerance the user wishes to allow and we can set the confidence level in the inference of the proportion, that determines a critical Z value (usually set to $Z = 2.58$ corresponding to $\alpha = 0.01$). Given N , the number of samples that is fixed by the given database, the probabilities that are observable from the database within these constraints are higher than a threshold value, given by the diagram in Figure 3. It plots the lower bound to the observable probabilities in datasets of given size N , in correspondence to different values of e_r . It is evident that this methodology of setting the minimum support threshold is statistically reliable and provides the user a guide to set this minimum value that usually is difficult to set. Above this support limit the observable probabilities are statistically reliable: it means that the probability of making an error greater than the error tolerance in this inference is controlled and it is lower than the confidence level. Furthermore, the estimation error is within the user established error tolerance. On the contrary, under this support limit, the estimation of itemsets probabilities is too risky and subjected to a too higher error (outside of the error tolerance).

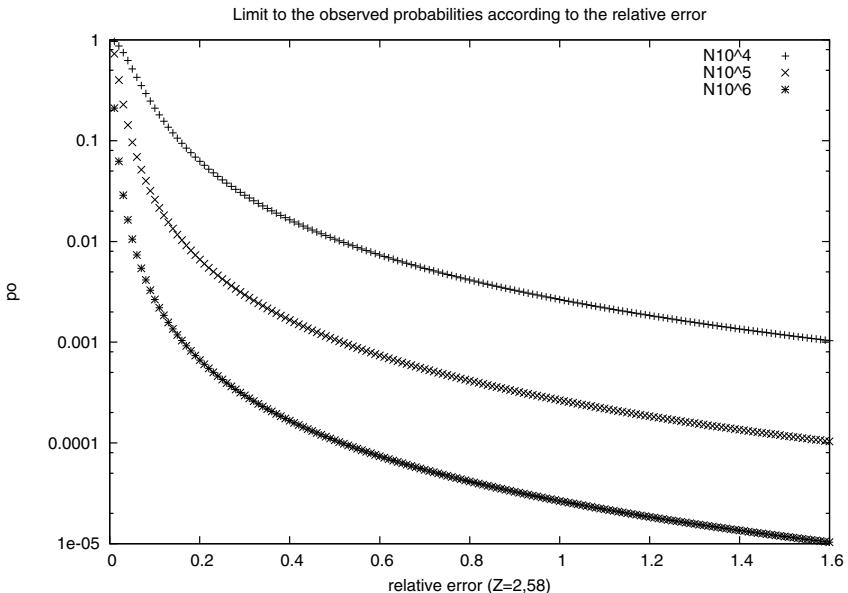


Fig. 3. Observable probabilities by relative error

7 System Description

DepMiner is implemented in java (1.6.0.12) and runs on a laptop. It uses Apache POI HSSF library for I/O. The core of the algorithm for frequent itemsets extraction is LCM FIMI algorithm (4.0) [21], the winner of the FIMI'04 competition. This algorithm is treated as a black-box and could be substituted by any other algorithm supporting the same I/O format. Figure 4 shows the computation flow of the system. Since, FIMI algorithm is treated as a black box, for this reason it is represented as a grey box in the flow. The algorithm for the computation of Δ_n performs the following tasks.

1. sets the *minsup* threshold for FIMI algorithm according to the explanation of Section 6;
2. randomizes the database;
3. runs the FIMI algorithm on the randomized database and reads its result;
4. builds the item-trie from the result of FIMI algorithm;
5. explores the item-trie in a level-wise fashion and computes for each itemset Δ . Δ computation is implemented in java by the algorithm described in [14]);
6. for each itemset, it computes Δ_n and stores the lower and upper bound for Δ_n found (LB and UB);
7. repeats steps 3-5 for the real database; for step 5 it performs pruning of itemsets by enforcing anti-monotonicity of Δ (it prunes if Δ is in the range (LB,UB));
8. enforces the property of Δ_n seen in Section 5 and computes Δ_n only if allowed (if Δ is outside of (LB,UB));
9. produces the itemsets ranking on the basis of Δ_n .

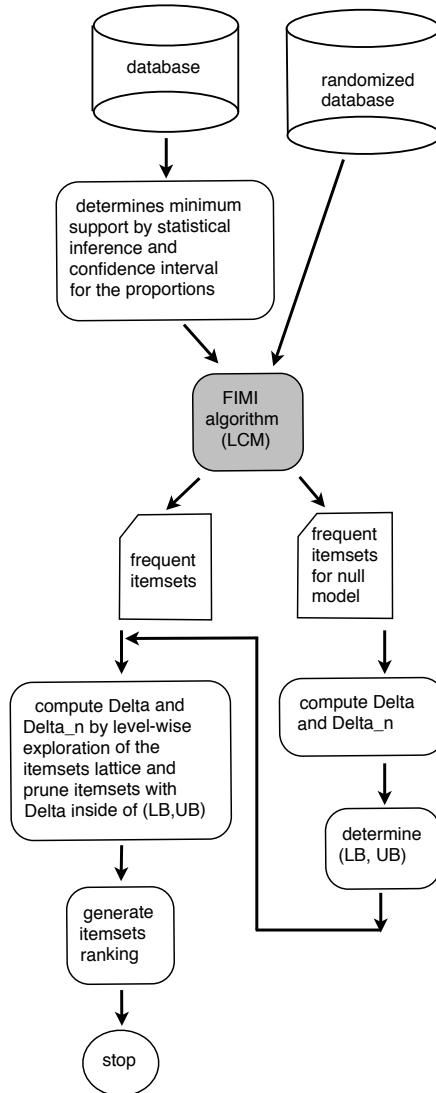


Fig. 4. The computation flow

The output of the itemsets ranking is implemented as a web page in HTML. GUI is implemented on JFreeChart, an open source library in java for the rendering of graphics and diagrams. It allows also to order the list of the itemsets by different criteria in an interactive way, such as by items, or by ascending/descending values of Δ_n : this is useful for the user to explore the results, according to her/his desire to observe the rarer itemsets (with a negative Δ_n) or the more frequent ones (with a positive Δ_n).

In Figure 2 we present another screen-shot of *DepMiner*. It shows in red color the histogram of Δ_n on real data and in blue the histogram of Δ_n on randomized data.

The user can zoom on specific areas of the histogram and observe in more detail the characterization of the dependencies existing in the dataset by the distribution of Δ_n . It is instructive to observe the different distributions obtained in sparse and dense data. Usually dense data have higher values of Δ_n , while in sparse data Δ_n values are lower and more scattered.

8 Experimental Evaluation

We run a set of experiments on 5 real datasets (from FIMI and UCI Machine Learning repositories) and on 2 real datasets coming from NASDAQ stock exchange index (from January 2001 to May 2009) and from the Italian lottery (with data on the numbers drawn from 1939). The lottery dataset is important in order to check the behavior of Δ_n on complete random data where even the marginals were uniform. Experiments were run on a CPU Intel Core 2 Duo T8100, 3GB RAM, SO Win Vista Business (SP1).

In Table 1 we include the total number of examples, *minsup* threshold adopted by LCM, the total number of itemsets generated (N), execution times to compute Δ_n (in seconds).

We performed two experiments: the first one on the compression capability and the second one on the capability of DepMiner to determine the dependencies in contrast to methods that assume the multi-way independence condition.

1. In this experiment, to be further conservative, we compare DepMiner results at many levels. We denote as itemsets clearly non independent, the itemset whose $\Delta_n \neq 0$. Thus we include in the results both these latter ones and the itemsets whose Δ_n is acceptable by the significance test on the lower and upper bounds obtained in randomized data. We include in Table 1 three ratios: the ratio between the number of itemsets with $\Delta_n \neq 0$ and N (denoted by Dep/N), the ratio between the significant dependencies and N (denoted by SDep/N) and the ratio between non derivable itemsets (NDI) obtained by the competitor method [4] and N (denoted by NDI/N).

These ratios quantify the volume of found dependencies in data. They clearly demonstrate the increased ability of DepMiner to reduce redundancies in the result than NDI.

2. In the second experiment we compare the results of DepMiner with MINI [7], the second competitor that we adopted in order to determine the difference between

Table 1. Experimental results

dataset	minsup	itemsets (N)	Dep/N	SDep/N	NDI/N	time(s)	γ (DM,MINI)	γ (RDM,RMINI)
Accidents	35%	65,500	13.5%	0.1%	22.93%	4294	-0.84	0.16
Chess	75%	20,582	1.2%	0.34%	2.11%	135	-0.95	-0.91
Nasdaq	0.14%	242	95.8%	46.69%	100%	107	-0.05	0.27
Kosarak	1.01%	21,934	82.5%	10.39%	95.55%	2221	-0.56	0.28
Mushroom	22.15%	14,189	1.48%	1.03%	5.84%	115	-0.94	-0.29
Retail	4.53%	22,524	79.7%	5.9%	99.56%	1322	0.02	0.55
Lottery	0.006%	91,499	99.1%	0%	100%	5804	0.81	0.77

DepMiner and another method of ranking based on the multi-way independence assumption.

The last columns of Table 1 report the result of a comparison between DepMiner ranking (denoted by DM) and MINI.

As said, our method considers only intrinsic dependencies in the itemsets and makes an estimate of independence of the items by means of the maximum entropy: it corresponds to a condition of minimum amount of information on the items given the knowledge about the other items in the itemset. The adopted referential probability for itemset I coincides with the hypothesis of independence of the singletons only if I has cardinality 2.

In order to measure the correlation between our ranking (DepMiner) and MINI's we adopted an objective measure, known as γ [9]: $\gamma = \frac{n_c - n_d}{n_c + n_d}$. n_c denotes the number of itemsets pairs on which the methods agree (are ranked in the same way by both of them) while n_d is the total number of pairs for which the methods disagree. γ ranges in $[-1, +1]$ and is 0 if there is independence.

Since the methods differ in the referential probability estimate, γ will quantify the impact of this difference. The difference is in the fact that MINI tends to observe an increased amount of dependencies in itemsets due to the fact that it considers also dependencies inherited by a subset to all its supersets.

In Table 1 we also compared the two rankings computed on randomized data (denoted by RDM and RMINI). We can notice that all the values reported by γ denote disagreement and generally have low values. The amount of discrepancies decreases (γ increases) if we move from real data to randomized data (since the high-order dependencies are spoiled during randomization). Furthermore, on complete random data (*Lottery*) the two methods agree ($\gamma = 0.8$) since DepMiner agrees on the hypothesis of independence among the singletons! In addition, we do not observe any significant change in γ if we randomize *Lottery*.

9 Conclusions

We have presented *DepMiner*, a method for the extraction of significant dependencies between the values assumed by database variables. We quantify the volume of these dependencies by the histogram of *Delta*. DepMiner gave good results by comparison of the rankings with [7] by γ and of its capability to compress results with NDI [4].

In DepMiner the user can set the parameter values guided by the system, explore the results in an interactive way, change the itemsets ranking criteria and zoom details in the statistics reports on the dependencies.

DepMiner web site is: <http://www.leodambrosi.it/depminer/>. From the site it is possible to download a presentation video.

References

1. Aggarwal, C.C., Yu, P.S.: A new framework for itemset generation. In: Proc. PODS (1998)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer Science (2006)

3. Brin, S., Motwani, R., Silverstein, C.: Beyond market baskets: Generalizing association rules to correlations. In: Proc. SIGMOD (1997)
4. Calders, T., Goethals, B.: Non-derivable itemset mining. Data Min. Knowl. Discov. 14(1) (2007)
5. Duan, L., Street, W.N.: Finding maximal fully-correlated itemsets in large databases. In: IEEE International Conference on Data Mining, pp. 770–775 (2009)
6. Fleuret, F.: Fast binary feature selection with conditional mutual information. Journal of Machine Learning Research 5, 1531–1555 (2004)
7. Gallo, A., Bie, T.D., Cristianini, N.: Mini: Mining informative non-redundant itemsets. In: PKDD (2007)
8. Gionis, A., Mannila, H., Miilikäinen, T., Tsaparas, P.: Assessing data mining results via swap randomization. In: Proc. KDD (2006)
9. Goodman, Kruskal: Measures of association for cross classifications. J. Amer. Stat. Ass. 49(268) (1954)
10. Hilderman, R.J., Hamilton, H.J.: Measuring the interestingness of discovered knowledge: A principled approach. Intell. Data Anal. 7, 347–382 (2003)
11. Knobbe, A.J., Ho, E.K.Y.: Maximally informative k-itemsets and their efficient discovery. In: KDD, pp. 237–244 (2006)
12. Liu, Z.Z.H.: Searching for interacting features. In: The 20th International Joint Conference on AI, IJCAI 2007 (2007)
13. Meo, R.: Theory of dependence values. TODS 45(3) (2000)
14. Meo, R.: Maximum independence and mutual information. TOIT 48(1) (2002)
15. Meo, R., Ienco, D.: Replacing support in association rule mining. In: Sing, Y., Rountree, N. (eds.) Rare Association Rule Mining and Knowledge Discovery: Technologies for Infrequent and Critical Event Detection. IGI Global publisher (2008)
16. Omiecinski, E.: Alternative interest measures for mining associations in databases. TKDE 15(1) (2003)
17. Savinov, A.: Mining dependence rules by finding largest support quota. In: Proc. SAC (2004)
18. Siebes, A., Vreeken, J., van Leeuwen, M.: Item sets that compress. In: SDM (2006)
19. Tan, P.-N., Kumar, V., Srivastava, J.: Selecting the right interestingness measure for association patterns. In: Proc. KDD (2002)
20. Tatti, N.: Maximum entropy based significance of itemsets. In: Proc. ICDM (2007)
21. Uno, T., Asai, T., Uchida, Y., Arimura, H.: Lcm v2. In: FIMI 2004,
22. Webb, G.I.: Discovering significant rules. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 434–443 (2006)
23. Xin, D., Cheng, H., Yan, X., Han, J.: Extracting redundancy-aware top-k patterns. In: KDD (2006)
24. Xin, D., Han, J., Yan, X., Cheng, H.: Mining compressed frequent-pattern sets. In: In VLDB, pp. 709–720 (2005)
25. Zhang, X., Pan, F., Wang, W., Nobel, A.B.: Mining non-redundant high order correlations in binary data. PVLDB 1(1) (2008)

Chapter 9

Integration of Dataset Scans in Processing Sets of Frequent Itemset Queries

Marek Wojciechowski, Maciej Zakrzewicz, and Paweł Boinski

Institute of Computing Science,
Poznan University of Technology,
ul. Piotrowo 2, 60-965 Poznan, Poland

{Marek.Wojciechowski,Maciej.Zakrzewicz,
Pawel.Boinski}@cs.put.poznan.pl

Abstract. Frequent itemset mining is often regarded as advanced querying where a user specifies the source dataset and pattern constraints using a given constraint model. In this chapter we address the problem of processing sets of frequent itemset queries, which brings the ideas of multiple-query optimization to the domain of data mining. The most attractive method of solving the problem with respect to possible practical applications is Common Counting which consists in concurrent execution of the queries using Apriori with the integration of scans of the parts of the database shared among the queries. The major advantage of Common Counting over its alternatives is its applicability to arbitrarily large batches of queries. If the memory structures of all the queries to be processed by Common Counting do not fit together in main memory, the set of queries has to be partitioned into subsets processed in several phases. We formalize the problem of dividing the set of queries for Common Counting as a specific case of hypergraph partitioning and provide a comprehensive overview of query set partitioning algorithms proposed so far.

1 Introduction

Frequent itemset discovery [1] is a very important data mining problem with numerous practical applications including market-basket analysis, medicine, telecommunications, and web usage analysis. Its goal is to discover the most frequently occurring subsets, called itemsets, in a database of sets of items, called transactions. Discovered frequent itemsets are often used to generate association rules. However, since generation of rules from itemsets is a rather straightforward, computationally inexpensive task, the focus of researchers has been mostly on optimizing the frequent itemset discovery process.

Many frequent itemset mining algorithms have been developed over the last two decades. The two most prominent classes of algorithms are determined by a strategy of traversing the pattern search space. Level-wise algorithms, represented by the classic Apriori algorithm [3], follow the breadth-first strategy, whereas pattern-growth methods, among which FP-growth [20] is the best known, perform the depth-first search.

Despite significant advances in frequent itemset mining, Apriori still remains the most widely implemented and used in practice frequent itemset mining algorithm due to its simplicity and satisfactory performance in real-world scenarios. Apriori starts with the discovery of 1-element frequent itemsets (i.e., frequent items), and then iteratively generates candidates (i.e., potentially frequent itemsets) from previously found smaller frequent itemsets and counts their occurrences in the database. To improve the efficiency of testing which candidates are contained in a transaction read from the database, the candidates are stored in a hash tree in main memory.

Frequent itemset mining is often regarded as advanced database querying where a user specifies the source dataset, the minimum support threshold, and optionally pattern constraints within a given constraint model [23]. A significant amount of research on efficient processing of frequent itemset queries has been done in recent years, focusing mainly on constraint handling [42][44][45][48][52] and reusing results of previous queries [6][12][35][39]. In terms of query optimization, the former can be regarded as optimizing individual queries separately, and the latter as optimizing sequences of queries [36].

This chapter is devoted to a relatively new problem of optimizing processing of sets of frequent itemset queries [51] that brings the concept of multiple-query optimization to the domain of frequent itemset mining. The idea is to process the queries concurrently rather than sequentially and take advantage of overlaps between queries' source datasets, although some of the proposed algorithms target also integration of in-memory data structures and computations. Sets of frequent itemset queries available for concurrent processing may arise in data mining systems operating in a batch mode or be collected within a given time window in multi-user interactive data mining environments. A motivating example from the domain of market basket analysis could be a set of queries discovering frequent itemsets from the overlapping parts of a database table containing customer transaction data from overlapping time periods.

Over the last decade we have proposed several approaches and algorithms to tackle the above problem. Here we focus on the most fundamental method called Common Counting [53], which basically consists in concurrent execution of the queries using Apriori with the integration of scans of parts of the database shared among the queries. Common Counting can be regarded as a fundamental solution of the problem of processing sets of frequent itemset queries for the two following reasons:

- methods proposed later for the Apriori algorithm were designed by extending Common Counting with further possibilities of computation sharing
- the first method of processing sets of frequent itemset queries dedicated to a newer FP-growth algorithm is a direct adaptation of Common Counting.

In general, we claim that Common Counting and its extensions have more practical importance than the methods for FP-growth since:

- despite positive evaluation of FP-growth and its variants in scientific literature, the majority of data mining systems available on the market still implement Apriori (often with some performance-oriented extensions and/or modifications with respect to the original formulation by Agrawal and Srikant).

- concurrent processing of frequent itemset queries requires storing the memory structures of several queries in main memory at the same time (with proposed space-preserving optimizations in some of the methods), which is problematic in case of FP-growth which basically stores a compressed version of the whole source dataset in main memory for each query.

On the other hand, despite the fact that its Apriori-oriented successors have been shown to outperform Common Counting if the queries' datasets significantly overlap, the original Common Counting method might still be preferable in practical implementations because:

- it has very little overhead and thus is the most predictable of all the proposed methods in a sense that it is beneficial for virtually any, even very small, overlapping among the queries' datasets
- it does not conflict with constraint-handling extensions proposed for Apriori
- contrary to its successors it has virtually unbounded scalability with respect to the number of queries.

While focusing on Common Counting, in this chapter we will particularly concentrate on the last of its properties listed above. Basic formulation of Common Counting [53] assumes that memory structures (i.e., hash trees) of all frequent itemset queries (i.e., concurrent Apriori executions) fit together in memory, which may not always be the case, at least for initial Apriori iterations. If the memory can hold only a subset of all queries, then it is necessary to partition the queries into subsets called phases and scan the database once for each phase. This observation leads to an interesting optimization problem with the goal of selecting from the set of all feasible partitionings the one (or one of) resulting in the minimal I/O cost of database scans [54].

In this chapter we summarize our research on query set partitioning algorithms for Common Counting by: (1) providing a formulation of the problem as a particular case of the hypergraph partitioning problem followed by a discussion regarding its computational complexity and obtaining the input information required by the partitioning algorithms; (2) reviewing all the proposed partitioning algorithms: CCRecursive [54], CCFull [57], CCCoarsening [58], CCAgglomerative [56], CCAgglomerativeNoise [8], CCGreedy, and CCSemiGreedy [10]; (3) presenting the extensive results of experiments aimed at evaluating the performance and accuracy of the algorithms.

2 Frequent Itemset Mining and Apriori Algorithm

In this section we review basic definitions concerning frequent itemset mining problem formulation as well as the classic Apriori algorithm, which is regarded as the basic algorithm for the Common Counting technique.

2.1 Basic Definitions and Problem Statement

Definition 1. Let I be a set of literals, called items. An *itemset* X is a set of items from I ($X \subseteq I$). The *size* of an itemset is the number of items in it. An itemset of size k is

called a k -itemset. A *transaction* over I is a couple $T = \langle tid, X \rangle$, where tid is a transaction identifier and X is an itemset. A database D over I is a set of transactions over I such that each transaction has a unique identifier.

Definition 2. A transaction $T = \langle tid, X \rangle$ supports an itemset Y if $Y \subseteq X$. The support of an itemset Y in D is the number (or percentage) of transactions in D that support Y .

Definition 3. An itemset is called frequent in D if its support is no less than a given minimum support threshold.

Problem 1. Given a database D and a minimum support threshold $minsup$, the problem of *frequent itemset mining* consists in discovering all frequent itemsets in D together with their supports.

2.2 Algorithm Apriori

The Apriori algorithm for frequent itemset discovery is formally presented in Fig. 1. In the formulation of the algorithm, F_k denotes the set of all frequent k -itemsets, and C_k denotes a set of potentially frequent k -itemsets, called candidates.

```

Input:  $D, minsup$ 
(1)  $F_1 =$  frequent 1-itemsets
(2) for  $(k=2; F_{k-1} \neq \emptyset; k++)$  do begin
(3)    $C_k = apriori\_gen(F_{k-1})$ 
(4)   forall transactions  $t \in D$  do begin
(5)      $C_t = subset(C_k, t)$ 
(6)     forall candidates  $c \in C_t$  do
(7)        $c.counter++$ 
(8)     end
(9)    $F_k = \{c \in C_k \mid c.counter \geq minsup\}$ 
(10) end
(11) Answer =  $\bigcup_k F_k$ 
```

Fig. 1. Apriori

Apriori starts with the discovery of frequent 1-itemsets, i.e., frequent items (line 1). For this task, the first scan of the database is performed. Before making the k -th pass (for $k > 1$), the algorithm generates the set of candidates C_k using F_{k-1} (line 3). The candidate generation procedure, denoted as *apriori_gen()*, provides efficient pruning of the search space, and will be described later. In the k -th database pass (lines 4–8), Apriori counts the supports of all the itemsets in C_k . (In practice, the database pass is performed only if the set of generated candidates is not empty.) The key step of this phase of the algorithm is determining which candidates from C_k are contained in a transaction t retrieved from the database. This step is denoted in the algorithm as a call to the *subset()* function that will be described later. At the end of the pass all itemsets in C_k with a support greater than or equal to the minimum support threshold $minsup$ form the set of frequent k -itemsets F_k (line 9). The algorithm finishes work if

there are no frequent itemsets found in a given iteration (condition in line 2) and returns all the frequent itemsets found (line 11).

The candidate generation procedure (*apriori_gen()* function in the algorithm) consists of two steps: the join step and the prune step. In the join step, each pair of frequent $k-1$ -itemsets differing only in the last item (according to the lexicographical order of the items within itemsets) is joined to form a candidate. In the prune step, itemsets having at least one subset that was found infrequent in the previous Apriori iteration are removed from the set of candidates.

The key to the overall efficiency of the Apriori algorithm is efficient checking which candidates are contained in a given transaction (*subset()* function in the algorithm). In order to avoid costly testing of each candidate for inclusion in a transaction retrieved from the database, candidates are stored in a special in-memory data structure, called hash tree. Leaves of a hash tree contain pointers to candidates, while the root node and internal nodes contain hash tables with pointers to their child nodes. In order to check which candidates are stored in a given transaction, all the subsets of the transaction are pushed down the hash tree. Candidates pointed by the leaves that are reached in the tree traversal operation are then verified for actual inclusion in the transaction.

3 Frequent Itemset Queries – State of the Art

In this section we review the most important research directions regarding frequent itemset queries: languages and programming interfaces for data mining, optimizing single queries in the form of constraint-based mining, and optimizing sequences of queries by reusing results of previously executed queries.

3.1 Frequent Itemset Queries

The research on data mining queries¹ was initiated by the pioneering work of Imielinski and Mannila [23] who envisioned the evolution of data mining systems analogous to that of database systems. They claimed that one of the major forces behind the success of database management systems (DBMS) had been the development of query languages, SQL in particular. Firstly, SQL together with a relational database API resulted in decoupling applications from a database backend. Secondly, the ad hoc nature of querying posed a challenge to build general-purpose query optimizers. Based on the above observations, Imielinski and Mannila postulated that formulation of a data mining query language could become a foundation for the development of general purpose next-generation data mining systems, which they called Knowledge and Data Discovery Management Systems (KDDMS). Such systems would allow knowledge discovery from data as well as storing the discovered patterns, rules, models, etc.² in the database for further querying. Imielinski and Mannila introduced the term *inductive database* for a database that apart from data also stores discovered knowledge. This term has been subsequently used by some researchers (e.g., [36]) to

¹ Imielinski and Mannila used the term *KDD query*.

² Imielinski and Mannila used the term *KDD object* to describe results of data mining queries and considered three types of KDD objects: rules, classifiers, and clusterings.

describe the research area devoted to data mining query languages and data mining query optimization, with a particular focus on frequent itemset and association rule mining.

Several data mining query languages were proposed following the statement of direction provided by Imielinski and Mannila. In [11] the authors proposed to extend SQL with the MINERULE operator for extracting association rules from the database and storing them back in a database relation. The proposed extension of SQL supported the following features: selecting the source dataset for mining with a possibility of grouping normalized data into sets, definition of the required structure of discovered rules and constraints regarding them, and specifying support and confidence thresholds.

In [24][25] another extension of SQL, called MSQL, was proposed. The focus of the presented approach was not only on the data mining query language itself but also on the application programming interface through which MSQL queries would be send by applications to the data mining system. As for MSQL syntax, it was oriented on both discovering new rules and querying previously discovered rules stored in the database. In contrast to the approach from [11] which used standard SQL queries to check which data supported or violated a given rule, MSQL offered explicit language constructs for that purpose.

In [19] another data mining query language, called DMQL, was introduced as a basic interface to the DBMiner data mining system [18]. One striking difference between DMQL and the two languages mentioned earlier was a much broader scope of DMQL, which supported several other data mining techniques apart from association rule discovery, i.e., mining characteristic, classification, and discriminant rules. Another advantage of DMQL over MINERULE and MSQL was the direct support for incorporating background knowledge in the form of concept hierarchies (taxonomies) in order to discover generalized rules.

A few years after the first proposals of data mining query languages, MineSQL [38][39] was proposed as the first language supporting mining frequent itemsets as a final form of discovered knowledge. Two other types of patterns handled by MineSQL were association rules and sequential patterns. Borrowing the best features of its predecessors, MineSQL integrated SQL queries to select source datasets to be mined, supported creation of taxonomies and using them in the mining process, contained clauses to determine source data supporting or violating discovered rules, and allowed a user to materialize discovered knowledge in the database for further analyses. As for the latter, a novel approach was taken, allowing data mining queries to be defining queries of materialized views, thus introducing the concept of materialized data mining views.

Unfortunately, the aforementioned language proposals by the research community had no or little influence on standards or existing database management systems. The relevant data mining standards that include executing association rule/frequent itemset mining task within their scope are: Java Data Mining [31], OLE DB for Data Mining [41], and SQL/MM Data Mining [26]. All of them treat association rule mining as building a mining model, which can then be browsed/queried, with frequent itemsets regarded as a by-product or an intermediate step. Nevertheless, implicitly a frequent itemset query specifying the source dataset and constraints on frequent itemsets (derived from user-specified association rule constraints) is still executed, which makes

the problem considered in this chapter still relevant. Recently, Oracle provided a strong argument supporting the research on frequent itemset queries by including a PL/SQL package for mining frequent itemsets as one of the standard packages starting from the version 10g of its database server [43]. The package allows frequent itemset queries to be formulated in pure SQL, which is an important signal to the research community. Firstly, it means that data mining queries are present in one of the market-leading DBMSs. Secondly, it clearly indicates that future research on data mining queries should focus on frequent itemset queries.

3.2 Constraint-Based Frequent Itemset Mining

Early research on frequent itemset and association rule query optimization focused on optimizing queries individually in the form of constraint-based mining. The idea was to incorporate user-specified constraints into mining algorithms in order to not only restrict the number of returned itemsets/rules (which can be done by post-processing) but also to reduce the execution time. The first approach to constraint-based frequent itemset mining was presented in [48]. Considered constraints had the form of a disjunctive normal form (DNF) with each disjunct stated that a certain item must or must not be present. Three Apriori-based algorithms were proposed, each of which applied a different modification to the candidate generation procedure.

In [42] Lakshmanan et al. considered more sophisticated constraints on frequent itemsets by allowing the constraints to refer to item attributes. The authors provided the first classification of constraints, identifying two important constraint properties: anti-monotonicity and succinctness. For all the considered constraint types a method of handling them within the Apriori framework was proposed, and formalized in the CAP algorithm.

Pei and Han [44][45] added monotonicity, previously considered in the context of correlated sets, to the two frequent itemset constraint properties identified by Lakshmanan et al. and then identified a broad class of constraints that do not exhibit any of the three properties but become monotone or anti-monotone if a certain order over the item domain is assumed. The new class of constraints was called convertible constraints. After completing the classification of constraints for frequent itemset mining, the authors showed that pattern-growth paradigm (represented by the FP-growth algorithm) is more suitable for constraint-based mining than the Apriori-based approach by providing guidelines on efficient handling of all four types of constraints within FP-growth.

In [52] a completely different method of handling constraints in frequent itemset mining, called dataset filtering, was presented. Instead of integrating the constraint-handling techniques into mining algorithms, the authors showed that certain constraints allow the query to be transformed into a query operating on the subset of the original query's input dataset that is equivalent in terms of the results. Although this approach is applicable to a small subset of constraints considered in frequent itemset mining, it has two important advantages. Firstly, it is independent of any particular frequent itemset mining algorithm. Secondly, it does not conflict with the constraint-handling techniques integrated into Apriori or FP-growth.

Recent works oriented on supporting frequent pattern mining in a query-oriented fashion suggest that, contrary to previous beliefs, pushing constraints down into the

mining process in order to optimize processing of an individual query is not a good approach in terms of the overall system performance [15][22]. The key observation was that if pattern constraints are handled in a post-processing phase, then the system may materialize all the frequent patterns, not just those forming the final result of the query. Such an approach maximizes the chances of reusing materialized patterns by subsequent queries which typically is the most efficient way of answering a frequent pattern query.

3.3 Reusing Results of Previous Frequent Itemset Queries

The fact that mining results are often materialized in the database for further analyses and browsing raised a natural question whether and under what circumstances results of previous frequent itemset queries can be reused to improve the execution time of a new query. To answer the question several techniques were proposed that can be generally described as optimizing sequences of frequent itemset queries, in the sense that when processing a given query it is assumed that the results of previous queries are available.

The research on reusing results of previous queries actually started under the name of “incremental mining” before the term “data mining query” came into use. Cheung et al. considered the scenario when new data is added to the previously mined database, thus potentially making some of the previously frequent itemsets infrequent and vice versa [12]. They proposed an Apriori-based algorithm, called FUP, that used information about the support of previously frequent itemset to prune candidates in the process of mining the incremented database. In their subsequent work [13], the authors generalized their technique in the form of the FUP2 algorithm that was able to efficiently handle not only insertions but also deletions of data³.

Both FUP and FUP2 require iterative scans of the whole input dataset in the same manner as basic Apriori. With the aim of reducing the cost of I/O activity in the process of incremental mining, Thomas et al. [49] proposed an incremental frequent itemset mining algorithm that required at most one scan of the whole input dataset, while still being able to handle both insertions and deletions in the original dataset. To achieve its goal, the algorithm required that not only frequent itemsets from the original dataset had to be available but also negative border of the set of frequent itemsets, i.e., itemsets that were Apriori candidates but turned out to be infrequent. The algorithm’s I/O activity was concentrated on inserted/deleted data, and one scan of the whole dataset was needed only if the information about the supports of itemsets from the original mining result and its negative border together with the information obtained from inserted/deleted data was not sufficient to determine the support of all itemsets potentially frequent in the modified database.

Nag et al. [40] considered an environment where a number of users concurrently issue association rule queries. In order to improve the overall performance of such a system, they suggested caching itemsets (frequent ones and those forming the negative border) generated as a by-product of previous association rule queries and use them in the frequent itemset mining stages of upcoming association rule queries.

³ In fact, FUP2 also handled updates of the input data, treating them as combinations of insertions and deletions.

Obviously, the approach taken actually resulted in optimizing sequences of frequent itemset queries. To facilitate itemset caching, the authors introduced the concept of a knowledge cache and proposed several algorithms for maintaining the cache as well as using its contents within the Apriori framework. As for differences among the queries in the context of which using the cache was beneficial, only differences in the support threshold were considered.

In [6] incremental refinement of association rule queries was considered in the context of the MINERULE operator from [11] by Baralis and Psaila. The authors postulated that a user is likely to refine their query a couple of times before obtaining the expected results. This observation was the motivation for studying syntactic differences between the queries that allow one query to be efficiently answered using the results of another query. Three relationships which occur between two association rule queries were identified: equivalence, inclusion, and dominance. Although, the approach concerned association rules, not frequent itemsets, it inspired subsequent works devoted to frequent itemsets as well.

Meo [35] continued the work of Baralis and Psaila on refinement of association rule queries in the context of the MINERULE operator by providing the ground for a query optimizer supporting the equivalence, inclusion, and dominance relationships. The author identified unique constraints and functional dependencies as elements of database management system functionality that could support such an optimizer and proposed extra intermediate data structures, called mining indices.

In [59] reusing results of previous frequent itemset queries stored in the form of materialized data mining views from [39]. Syntactic differences between MineSQL queries were analyzed and six scenarios of reusing the results of one query by another query were identified. These scenarios covered the techniques from [12], [40], and two of the three relationships from [6] adapted to frequent itemset queries.

The aforementioned methods of optimizing sequences of data mining queries can be regarded as preparing the ground for optimizing sets of data mining queries. The difference between all the above approaches and the problem studied in this chapter is the fact that they all deal with a sequence of queries arriving to the system and processed in a pre-defined order, while we are given a batch of queries at once. Obviously, applying arbitrary order on a set of queries turns it into a sequence, which means that all methods designed to deal with sequences of frequent itemset queries are automatically applicable to sets of frequent itemset queries as well. In fact, as we have shown in [37], it is possible to maximize the chance of one query reusing the results of another query by choosing appropriate ordering of the queries and/or introducing additional queries into the sequence. However, such an approach can be successfully applied just to a small fraction of cases that can be handled by a method designed with sets of queries in mind, like Common Counting to which this chapter is devoted.

4 Optimizing Sets of Frequent Itemset Queries

This section contains our formal, generic model of frequent itemset queries and presents the problem of optimizing sets of frequent queries, followed by a discussion on possible solutions and related work.

4.1 Basic Definitions

Definition 4. A *frequent itemset query* is a tuple $dmq = (R, a, \Sigma, \Phi, \beta)$, where R is a relation, a is a set-valued attribute of R , Σ is a condition involving the attributes of R (called a *data selection predicate*), Φ is a condition involving discovered itemsets (called a *pattern constraint*), and β is the minimum support threshold. The result of dmq is a set of itemsets discovered in $\pi_a \sigma_{\Sigma} R$, satisfying Φ , and having support $\geq \beta$ (π and σ denote projection and selection).

It should be noted that the assumption of using set-valued attributes to store transactions in a database relation does not influence the generality of the proposed query model. If input dataset is stored in the classic first normal form (1NF) where either each item occupies a separate row (so called transactional data format) or is represented by a binary flag in a dedicated column (so called relational format), it will be converted to the form of collection of sets in the process of reading the data from the database.

Our query model is general in the sense that we pose no restrictions on the form of data selection predicates, which we assume will be specified in pure SQL, and pattern constraints whose form and nature is irrelevant for our Common Counting method, which will be discussed later.

Example 1. Given the database relation $R_1(a_1, a_2)$, where a_2 is a set-valued attribute and a_1 is an attribute of integer type. The frequent itemset query $dmq_1 = (R_1, a_2, "a_1 > 5", "itemsetl < 4", 3\%)$ describes the problem of discovering frequent itemsets in the set-valued attribute a_2 of the relation R_1 . The frequent itemsets with support of at least 3% and size less than 4 items are discovered in the collection of records having $a_1 > 5$.

Definition 5. The set of *elementary data selection predicates* for a set of frequent itemset queries $DMQ = \{dmq_1, dmq_2, \dots, dmq_n\}$ is the smallest set $S = \{s_1, s_2, \dots, s_k\}$ of data selection predicates over the relation R such that for each u, v ($u \neq v$) we have $\sigma_{su} R \cap \sigma_{sv} R = \emptyset$ and for each dmq_i there exist integers a, b, \dots, m such that $\sigma_{\Sigma_i} R = \sigma_{sa} R \cup \sigma_{sb} R \cup \dots \cup \sigma_{sm} R$.

The set of elementary⁴ data selection predicates represents the partitioning of the database determined by overlapping of queries' datasets. Each partition contains transactions shared by exactly the same subset of queries and for each partition this subset of queries is different. Thus, the database partitions corresponding to elementary data selection predicates will be units of data subject to the optimization of the disk I/O cost.

⁴ It should be noted that we use the term “elementary” to describe the property that data selection predicates of all the queries from a batch can be expressed as a disjunction of some of the elementary data selection predicates and at the same time this set cannot be reduced by combining predicates with disjunction so that this property still holds. Obviously, in general such elementary data selection predicates can be syntactically complex, i.e., having a form of simpler predicates combined using logical operators.

The set of elementary data selection predicates for a given set of frequent itemset queries $DMQ = \{dmq_1, dmq_2, \dots, dmq_n\}$ can be generated using the following procedure:

- 1) generate S as the set of all possible conjunctions of predicates from the set $\{\Sigma_1, \neg\Sigma_1, \Sigma_2, \neg\Sigma_2, \dots, \Sigma_n, \neg\Sigma_n\}$ such that for each pair of predicates $\Sigma_i, \neg\Sigma_i$, exactly one of them is present in the conjunction
- 2) remove from S the conjunction $\neg\Sigma_1 \wedge \neg\Sigma_2 \wedge \dots \wedge \neg\Sigma_n$
- 3) remove from S all predicates s such that $\sigma_s R = \emptyset$

The first step of the above procedure generates a formula for each subset of the set of queries that selects all the data shared only by this subset of queries. Obviously, the number of such formulas is 2^n . In the second step, the formula representing the empty subset of the set of queries (i.e., selecting data that do not belong to any query) is discarded. Finally, the formulas that select no data, i.e., do not correspond to any actual partition of data implied by the overlapping of queries' datasets are removed.

The last step of the procedure is the most challenging one. Some of the formulas will be discarded after syntactic analysis (i.e., self-contradictory formulas), while others will be identified as selecting no data only after the first execution of SQL queries corresponding to them. In the latter case, the execution of a set of frequent itemset queries will start with the superset of the actual set of elementary data selection predicates that will be tuned at early stages of the mining process. It should be noted that in scenarios that we believe are the most typical, i.e., involving all the queries selecting data according to the same attribute (e.g., queries mining data from different periods of time) syntactic analysis should be sufficient to eliminate redundant formulas from the set of elementary data selection predicates. Even if that is not the case, unnecessary SQL queries could be avoided thanks to statistics collected by the system. Nevertheless, we claim that generation of the set of elementary data selection predicates based on the queries' data selection predicates is a task for a SQL query optimizer.

Example 2. Given the relation $R_1(a_1, a_2)$ and three data mining queries: $dmq_1 = (R_1, a_2, "5 \leq a_1 < 20", \emptyset, 3\%)$, $dmq_2 = (R_1, a_2, "10 \leq a_1 < 30", \emptyset, 5\%)$, $dmq_3 = (R_1, a_2, "15 \leq a_1 < 40", \emptyset, 4\%)$. The set of elementary data selection predicates is then $S = \{s_1 = "5 \leq a_1 < 10", s_2 = "10 \leq a_1 < 15", s_3 = "15 \leq a_1 < 20", s_4 = "20 \leq a_1 < 30", s_5 = "30 \leq a_1 < 40"\}$.

4.2 Problem Formulation

The problem of efficient processing of sets of frequent itemset queries could be formalized as the following optimization problem:

Problem 2. Given a set of frequent itemset queries $DMQ = \{dmq_1, dmq_2, \dots, dmq_n\}$, the problem of *multiple-query optimization* of DMQ consists in generating an algorithm to execute DMQ that minimizes the overall processing time.

The trouble with the above problem formulation is that a hypothetical multi-query optimizer for frequent itemset queries would need formulas for estimating the costs of

various execution plans. Obviously, before such cost formulas for multi-query execution strategies could be developed, they had to exist for single queries⁵, which unfortunately is not the case yet.

We claim that, taking the above observation into account, the present research on efficient processing sets of frequent itemset queries should focus on proposing algorithms that offer performance improvement over sequential execution, at least in typical scenarios, by sharing computations among the queries. Common Counting, presented in detail in the next section, is such a method.

4.3 Related Work on Multi-query Optimization

Multiple-query optimization has been extensively studied in the context of database systems (see [47] for an overview). The idea was to identify common subexpressions (selections, projections, joins, etc.) and construct a global execution plan minimizing the overall processing time by executing the common subexpressions only once for the set of queries [5] [28]. Many heuristic algorithms for multiple-query optimization in database systems were proposed (e.g., [46]). Data mining queries could also benefit from the general strategy of identifying and sharing common computations. However, due to their different nature they require novel multiple-query processing methods.

To the best of our knowledge, apart from the problem considered in this paper, multiple-query optimization for frequent pattern queries has been considered only in the context of frequent pattern mining on multiple datasets [30]. The idea was to reduce the common computations appearing in different complex queries, each of which compared the support of patterns in several disjoint datasets. This is fundamentally different from our problem, where each query refers to only one dataset and the queries' datasets overlap.

Earlier, the need for multiple-query optimization has been postulated in the somewhat related research area of inductive logic programming, where a technique based on similar ideas as Common Counting has been proposed, consisting in combining similar queries into query packs [7].

5 Common Counting

In this section we present the Common Counting method for efficient processing of sets of frequent itemset queries. We start with a basic algorithm that does not take into account the limit of the available memory, and then discuss the way of extending it to deal with such a practical restriction.

5.1 Basic Algorithm

The motivation for Common Counting is the observation that for a set of frequent itemset queries whose input datasets overlap, the most visible common operation in their Apriori-based execution is reading the shared parts of the database in the process of counting the candidates. Common Counting reduces the I/O costs with respect to sequential processing by concurrent execution of a set of frequent itemset queries

⁵ For example, in order to estimate the cost of sequential execution of a set of queries.

using Apriori and integration of scans of the shared parts of the database. The pseudo-code of Common Counting is presented in Fig. 2.

```

Input:  $DMQ = \{dmq_1, dmq_2, \dots, dmq_n\}$ , where  $dmq_i = (R, a, \Sigma_i, \Phi_i, \text{minsup}_i)$ 
(1)  $S$  = set of elementary data selection predicates for  $DMQ$ 
(2) for ( $i=1; i \leq n; i++$ ) do
(3)    $C_{1,i}$  = all possible 1-itemsets
(4)   for ( $k=1; C_{k,1} \cup C_{k,2} \cup \dots \cup C_{k,n} \neq \emptyset; k++$ ) do
(5)     begin
(6)       for each  $s_j \in S$  do
(7)         begin
(8)            $CC = \{C_{k,i} : \sigma_{sj}R \subseteq \sigma_{\Sigma}R\}$ 
(9)           if  $CC \neq \emptyset$  then  $\text{count}(CC, \sigma_{sj}R)$ 
(10)        end
(11)        for ( $i=1; i \leq n; i++$ ) do
(12)          begin
(13)             $F_{k,i} = \{c \in C_{k,i} : c.\text{counter} \geq \text{minsup}_i\}$ 
(14)             $C_{k+1,i} = \text{apriori\_gen}(F_{k,i})$ 
(15)          end
(16)        end
(17)      for ( $i=1; i \leq n; i++$ ) do
(18)         $\text{Answer}_i = \sigma_{\Phi_i} \bigcup_k F_{k,i}$ 
```

Fig. 2. Common Counting

The initial step of Common Counting is the generation of the set of elementary data selection predicates for the set of queries as discussed in Sect. 4.1 (line 1). After that, Common Counting iteratively generates and counts candidates for all frequent itemset queries. In the first iteration, for all the queries, the set of candidates is the set of all possible items (lines 2-3). The candidates of the size k ($k > 1$) are generated from frequent itemsets of size $k-1$, separately for each query (lines 11-15). Generation of candidates (represented in the pseudo-code by the *apriori_gen()* function) is performed exactly as in the original Apriori algorithm. The candidates generated for each query are stored in a separate hash tree. The iterative process of candidate generation and counting ends when for all the queries no further candidates can be generated (the condition in line 4).

Occurrences of candidates for all the queries are counted during one integrated database scan in the following manner. For each elementary data selection predicate, the transactions from its corresponding database partition are read one by one. For each transaction the candidates of the queries referring to the database partition being read are considered, and the counters of candidates contained in the transaction are incremented (lines 6-10). The inclusion test is performed by confronting the transaction with hash trees of all the queries referring to the database partition containing the transaction. Candidate counting is represented in the pseudo-code as the *count()* function. It should be noted that if a given elementary data selection predicate is shared by several queries, its corresponding database partition is read only once during each candidate counting phase.

The formulation of Common Counting from Fig. 2 does not incorporate pattern constraints into the actual mining process, leaving them for post-processing (line 18). The reason for this is the fact that the optimization applied by Common Counting concerns only database access. Nevertheless, it should be noted that the Common Counting scheme does not interfere in any way with constraint-handling techniques described in sect. 3.2, meaning that these techniques could be incorporated into Common Counting in the same way they are incorporated into pure Apriori.

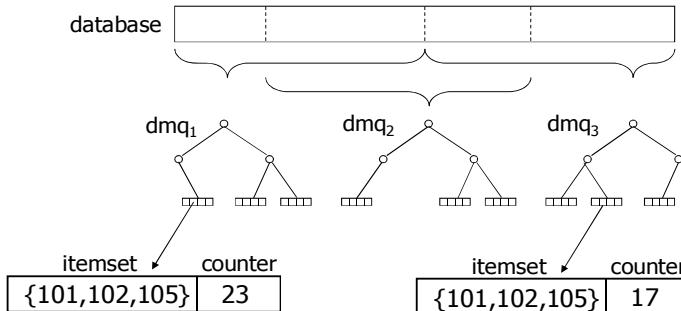


Fig. 3. Illustration of Common Counting and its memory structures

The idea of Common Counting and its memory structures are illustrated in Fig. 3 for the set of three queries. Each query creates its own hash tree to store its candidates. If a given itemset is generated as a candidate by more than one query, it appears in more than one hash tree. Clearly, there are more possibilities of computation sharing among the queries beyond just integrating scans of input data. However, such a tighter integration comes at a certain price, which will be briefly described in Sect. 9.

Common Counting was designed with the assumption that partitions of the database corresponding to elementary data selection predicates can be efficiently retrieved, for example using indexes. However, there is no guarantee that for any data selection predicate specified in a user's query an appropriate index will be present in the database. Interestingly, Common Counting can be even more beneficial if full scans of the database relation containing input datasets of the queries are necessary to retrieve the data partitions. However, a change of the way Common Counting reads data partitions is required to adapt it to the absence of more efficient data access paths than full table scan. Instead of reading partitions one by one, one scan of the whole relation should be performed and for each transaction the check to which queries it belongs should be performed. Such an approach is possible thanks to the fact that Common Counting actually does not require that partitions are read as a whole and can switch from partition to partition during the scan of the database relation. Obviously, after the above modification Common Counting will have to read more data than it would have to if an appropriate index was available. Nevertheless, if full table scans are the only option, sequential execution of the queries will have its performance relatively more degraded than Common Counting, since for each query a full scan will be needed.

5.2 Motivation for Query Set Partitioning

The basic Common Counting assumes that in each of its iterations candidate hash trees of all the frequent itemset queries forming a batch can reside in the main memory at the same time, and thus only one database scan is needed to count current candidates of all the queries. Obviously, in practice memory is going to be limited, so that simple strategy may not always be applicable.

Actual memory requirements of the Common Counting method depend on the number of queries, their support thresholds, and characteristics of the database. Nevertheless, in order to make Common Counting applicable in practice for arbitrarily large batches of queries, regardless of their predicates and the nature of the database, a solution enabling counting the candidates stored in hash trees whose total size exceeds the memory limit has to be provided.

To address the above issue, we propose to partition the set of queries into subsets so that the hash trees of the queries from each subset fit into memory. After the partitioning, counting of the candidates will be performed in several phases, with each of the resulting subsets of queries having their candidates counted during one database scan.

Clearly, the I/O cost of a Common Counting iteration divided into phases will be greater than it would be if query partitioning was not necessary. However, this cost will still be smaller than in case of sequential execution of the queries because the data sharing among the queries assigned to the same phase will still be taken advantage of. It should be noted that in general for a given set of queries many different partitioning will be possible, resulting in potentially different I/O costs. This observation leads to an interesting optimization problem of choosing the partitioning with minimal resulting I/O costs. Before we formalize the problem and present algorithm to solve it, we will discuss a few key issues that query partitioning algorithms for Common Counting have to take into account.

5.3 Key Issues Regarding Query Set Partitioning

In order to be able to verify if a given assignment is feasible and compare feasible assignments in terms of resulting I/O costs, the query set partitioning algorithm has to be provided with the sizes of hash trees and the sizes of database partitions corresponding to the elementary data selecting predicates representing data sharing among the queries.

Since the sizes of candidate hash-trees change between Apriori iterations, the assignment of queries to Common Counting phases has to be performed at the beginning of every Apriori iteration. A partitioning algorithm requires that sizes of candidate hash-trees are known in advance. Therefore, in each iteration of Common Counting, we first generate all the candidate hash trees, measure their sizes, save them to disk, partition the data mining queries into phases, and then load the hash trees from disk when they are needed during Common Counting phases. We have also considered estimating hash tree sizes in order to avoid the costs of migrating pre-created hash trees between main memory and disk [9]. The estimation formula was designed with a tendency to overestimate the size of a tree so as to minimize the chance that some of the phases actually do not fit into memory. The negative side of the approach

taken was that the partitionings based on estimates were significantly worse in terms of I/O costs than those based on actual computed sizes. Therefore, we consider generating all the candidate hash trees in advance and swapping some of them to disk if there is not enough memory to keep them all together as a primary option.

As for the sizes of partitions of the database determined by overlapping of the queries' datasets, it is true that they are not known before the first iteration of Common Counting. Fortunately, the sizes of the database partitions will not be required until the second Common Counting iteration, when the hash trees are starting to be constructed. In the first iteration candidates for all the queries are all single items from the database and storing their counters for a sensible number of queries should not be a problem. Therefore, only one scan of the database will be needed in the first Common Counting iteration with no assignment of queries to phases, i.e., with one phase including all the queries. During that scan the actual sizes of database partitions can be calculated, so we can assume that they are available for subsequent Common Counting iterations.

The exhaustive search for an optimal assignment of queries to Common Counting phases is inapplicable for large batches of queries due to the size of the search space (expressed by a Bell number). Moreover, as we will show in the next section, the problem is NP-hard. Therefore, in practice heuristic algorithms have to be used.

Finally, let us consider the effect of available access paths to data partitions on the problem of assigning queries to Common Counting phase. The above discussion and the problem formulation from the next section are valid if the selective access to partitions of the relation with input data is possible, which we regard as the primary scenario. If the only access path to the data is full table scan, then Common Counting will be reading the whole relation in each Common Counting phase. Thus, the data sharing among the queries assigned to the same will be irrelevant, and only the number of phases will be important. Consequently, the problem will become a classic bin packing problem, where a number of objects (in our case – hash trees) are to be packed to bins of a given size (in our case – memory limit) so that the number of bins (in our case – phases) is minimal. Although bin packing is an NP-hard problem, numerous efficient heuristics are known for solving it. Therefore, we will not analyze this scenario any further.

6 Frequent Itemset Query Set Partitioning by Hypergraph Partitioning

In this section we introduce the concept of data sharing hypergraph as a model of data sharing between frequent itemset queries. Then, in the context of data sharing hypergraph we formulate our problem of partitioning the set of frequent itemset queries as a particular case of hypergraph partitioning. Finally, we discuss computational complexity of the problem and review related work on hypergraph partitioning techniques.

6.1 Data Sharing Hypergraph

A set of frequent itemset queries $DMQ = \{dmq_1, dmq_2, \dots, dmq_n\}$ can be modeled as a weighted hypergraph whose vertices represent queries and hyperedges represent elementary data selection predicates. A hyperedge in the hypergraph corresponds to a database partition and connects the queries whose source datasets *share* that partition. Below we formally define a data sharing hypergraph in the context of elementary data selection predicates.

Definition 6. A *data sharing hypergraph* for the set of data mining queries $DMQ = \{dmq_1, dmq_2, \dots, dmq_n\}$ and its corresponding set of elementary data selection predicates $S = \{s_1, s_2, \dots, s_k\}$ is a hypergraph $DSH = (V, E)$, where $V = DMQ$, $E = S$, and a vertex $dmq_i \in DMQ$ is incident to an hyperedge $s_j \in S$ iff $\sigma_{s_j} R \subseteq \sigma_{dmq_i} R$. Each vertex dmq_i has an associated weight $w(dmq_i)$ representing the amount of memory consumed by data structures of the query dmq_i . Each hyperedge s_j has an associated weight $w(s_j)$ representing the size of the database partition returned by the elementary data selection predicate s_j .

Note that the above definition of a data sharing hypergraph allows hyperedges incident to only one vertex in order to represent database partitions read by only one query. These hyperedges are necessary for a data sharing hypergraph to provide complete information required by the main Common Counting scheme, and will also be used to evaluate the partitioning objective in our hypergraph partitioning problem.

Example 3. Given three frequent itemset queries operating on the relation $R_1 = (a_1, a_2)$: $dmq_1 = (R_1, "a_2", "5 \leq a_1 < 10", \emptyset, 3\%)$, $dmq_2 = (R_1, "a_2", "10 \leq a_1 < 20", \emptyset, 5\%)$, $dmq_3 = (R_1, "a_2", "15 \leq a_1 < 40", \emptyset, 4\%)$. The set of elementary data selection predicates for the set of frequent itemset queries $DMQ = \{dmq_1, dmq_2, dmq_3\}$ is $S = \{"5 \leq a_1 < 10", "10 \leq a_1 < 15", "15 \leq a_1 < 20", "20 \leq a_1 < 30", "30 \leq a_1 < 40"\}$. The data sharing hypergraph for DMQ is shown in Fig. 4.

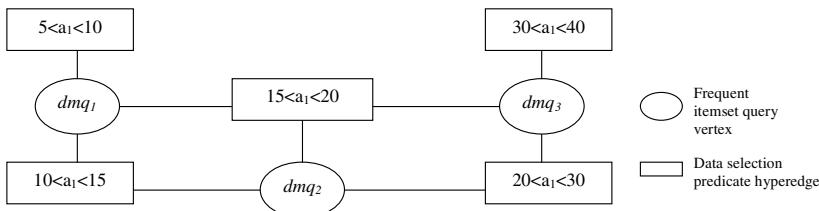


Fig. 4. Example data sharing hypergraph

6.2 Hypergraph Partitioning Problem Formulation

The goal of query set partitioning for Common Counting is assigning queries to phases fitting into main memory in a way minimizing the overall I/O cost. Each of the phases returned by the partitioning algorithm is a set of frequent itemset queries for which a data sharing hypergraph can be constructed. Thus, query partitioning for Common Counting can be interpreted as a particular case of hypergraph partitioning.

After partitioning, elementary data selection predicates corresponding to database partitions shared by queries that have been assigned to different phases will be represented as hyperedges in more than one resulting hypergraph. In other words, a hyperedge that is cut by the partitioning will be partitioned into a number of hyperedges connecting subsets of vertices previously connected by the original hyperedge. This is crucial for our problem because we need to preserve the information about data partitions to be scanned in each of the Common Counting phases.

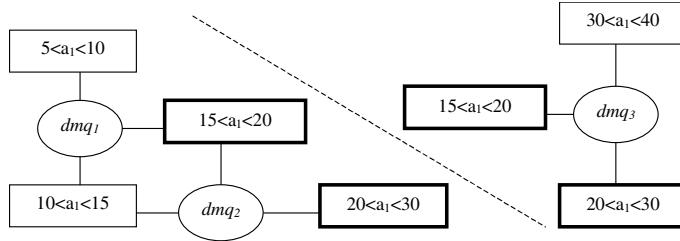


Fig. 5. Example partitioning of the data sharing hypergraph from Fig. 4

One of the possible partitionings of the data sharing hypergraph from Fig. 4, representing partitioning the set of queries into two phases is shown in Fig. 5. Hyperedges that have been cut (partitioned) are presented in bold.

In terms of hypergraph partitioning, the goal of query partitioning for Common Counting can be stated as follows:

Problem 3. Given a data sharing hypergraph for the set of frequent itemsets queries $DSH = (V, E)$ and the amount of available main memory $MEMSIZE$, the goal is to partition the vertices of the hypergraph into k disjoint subsets V_1, V_2, \dots, V_k , and their corresponding data sharing hypergraphs $DSH_1 = (V_1, E_1), DSH_2 = (V_2, E_2), \dots, DSH_k = (V_k, E_k)$ such that

$$\forall_{x=1..k} \sum_{dmq_i \in V_x} w(dmq_i) \leq MEMSIZE$$

minimizing

$$\sum_{x=1..k} \sum_{s_j \in E_x} w(s_j).$$

In the above formulation, the partitioning constraint has the form of an upper bound on the sum of weights of vertices in each partition, reflecting the amount of available memory, while the partitioning objective is to minimize the total sum of weights of hyperedges across all the partitions, representing the overall I/O cost of the Common Counting iteration. According to the classification from [32], the partitioning objective in our problem formulation is equivalent to minimizing the $k-1$ metric, where the goal is to minimize the size of the hyperedge cut to which each cut hyperedge

contributes $k-1$ times its weight (in the definition of the $k-1$ metric k denotes the number of partitions across which a cut hyperedge spans, not the total number of resulting partitions).

It should be noted that the number of resulting partitions (i.e., Common Counting phases) is not known a priori, and there is no lower bound on the sum of weights of vertices in each partition. Informally, the latter means that we do not require that the resulting partitions are of similar sizes.

6.3 Computation Complexity of the Problem

Our hypergraph partitioning problem is NP-hard since if we consider only hypergraphs with hyperedges connecting exactly two vertices, its decision version will restrict itself to the classical graph partitioning problem formulation from [14] (proof of NP-completeness by restriction). Taking that into account, for large number of vertices (frequent itemset queries) heuristic approaches have to be applied to solve the problem, resulting in possibly suboptimal solutions.

6.4 Related Work on Hypergraph Partitioning

Hypergraph partitioning has been extensively studied particularly in the domain of VLSI design [4]. In data mining context it has been proposed as a clustering technique in [34]. Many formulations of the hypergraph partitioning problem have been considered, differing in partitioning constraints and objectives (see e.g. [4] or [32]). Our formulation differs from typical approaches because:

- we do not have any balance constraint on the sizes of resulting partitions, only a strict upper bound on the sum of weights of vertices in a partition, reflecting the memory limit
- we do not specify the desired number of partitions in advance; in fact, the resulting number of partitions (phases) is irrelevant, only the partitioning objective matters
- for a hyperedge that is cut by the partitioning, we take into account the number of partitions to which the vertices connected by the cut hyperedge belong⁶.

7 Query Set Partitioning Algorithms

This section presents heuristic algorithms that we proposed to solve the hypergraph partitioning problem formulated in the previous section. We have taken two different approaches to designing query set partitioning algorithms for Common Counting. The first was to invent new methods, dedicated to our particular problem. The algorithms designed this way are CCRecursive, CCFull, CCCoarsening, CCAgglomerative, and CCAgglomerativeNoise. An alternative approach was to apply some of the

⁶ In other application domains it is more common just to check whether a hyperedge is cut or not. Nevertheless, the same approach as ours has also been considered in the VLSI domain.

well-known metaheuristics. Motivated by the reported success of application of the greedy approach to a related problem of k-way graph partitioning [27], we implemented a greedy and semi-greedy strategies in the context of our hypergraph partitioning problem. The resulting algorithms have been named CCGreedy and CCSemi-Greedy respectively.

7.1 CCRecursive

The CCRecursive algorithm directly utilizes the information contained in the data sharing hypergraph. Obviously, in order to minimize the partitioning criterion, the queries sharing an elementary data selection predicate should be assigned to the same phase. Since, in general it may not be possible for all the predicates, CCRecursive gives preference to predicates corresponding to larger data partitions.

```

Phases = {Ø}
sort S = <s1, s2, ..., sk> in descending order with respect to cost(si)
call CCRecursive(S, DMQ, Phases)

CCRecursive(S, DMQ, Phases):
begin
ignore in S those predicates that are used by less than two dmqs
for each si in S do begin
    tmpDMQ = {dmqj | dmqj = (R, a, Σj, Φj, βj), si ⊆ Σj, dmqj ∈ DMQ}
    commonPhases = {p ∈ Phases | p ∩ tmpDMQ ≠ Ø}
    if commonPhases = Ø then
        newPhase = tmpDMQ
    else
        newPhase = tmpDMQ ∪ Up | p ∈ commonPhases
    end if;
    if treesize(newPhase) ≤ MEMSIZE then
        Phases = Phases \ commonPhases
        Phases = Phases ∪ newPhase
    else
        Phases = CCRecursive(<si+1, ..., sk>, newPhase, Phases)
    end if
end
add phase for each unassigned query
compress Phases containing queries from DMQ
return Phases
end

```

Fig. 6. CCRecursive

The detailed structure of the CCRecursive algorithm is given in Fig. 6. The algorithm iterates over all the elementary data selection predicates, sorted in descending order with respect to their I/O costs. For each elementary data selection predicate we identify all the data mining queries that include the predicate. If none of the identified queries has been already assigned, then we create a new phase (partition) and we put all the queries into the new phase. Otherwise, we merge the phases to which the assigned queries belonged and we assign the other queries to this new phase. If the size of the newly created phase exceeds the memory limit, then the phase is split into smaller ones by recursive execution of the algorithm with the list of data selection predicates reduced to only those following the predicate that led to exceeding the memory limit. (The auxiliary function $\text{treesize}(Q)$, where Q is a set of data mining queries, represents total memory size required to hold candidate hash trees for all the queries in Q .)

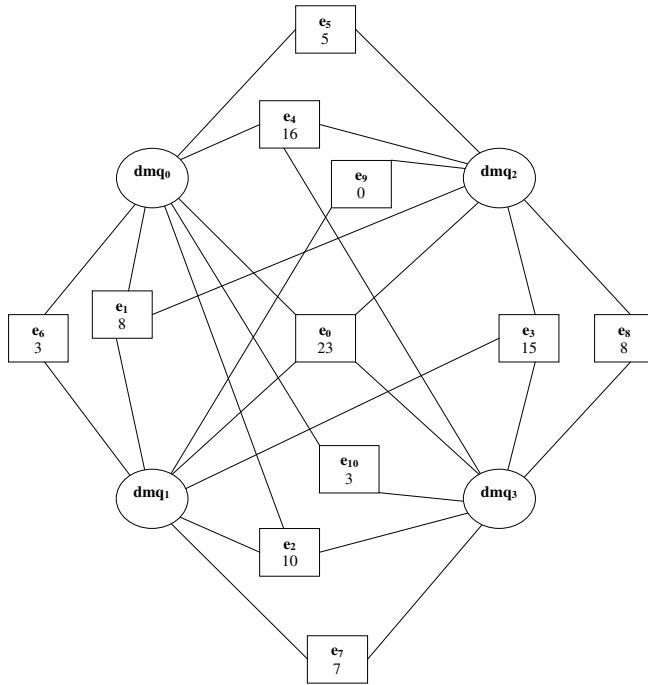
At the end of the algorithm, we perform *phase compression*, which consists in reducing the number of phases by merging the phases so that the resulting phases still do not exceed the memory limit. This in fact leads to a bin packing problem which itself is an NP-hard problem. If the number of phases to compress prevents the exhaustive search for better packing, one of the heuristics proposed for the bin packing problem can be applied.

Although in general, in the context of our problem merging phases that do not share a database partition makes no sense in terms of minimizing the partitioning criterion, this step is required in CCRecursive since the phases may have been sharing a database partition corresponding to a data selection predicate removed in the recursive call of the algorithm.

7.2 CCFull

The problem with CCRecursive is that it does not take into account the fact that the queries may share more than one database partition. As a consequence, CCRecursive may not actually give precedence to query grouping/phase merging resulting in bigger gains with respect to the partitioning criterion. CCFull addresses this problem by considering the actual gains thanks to assigning a given subset of queries to the same phase. Another advantage of CCFull over CCRecursive is that CCFull does not require recursive calls, which makes its number of operations more predictable.

The first step of CCFull is generation of a *gain hypergraph* for the set of data mining queries. The gain hypergraph is a full (i.e., complete) hypergraph, in which vertices represent the data mining queries while hyperedges are labeled with weights which represent the amount of I/O cost reduction to be achieved if data mining queries connected with the hyperedge were executed together (in the same phase). If common execution of given data mining queries results in no reduction of I/O cost, the weight of the connecting hyperedge is zero. A sample gain hypergraph is shown in Fig. 7. For example, it can be noticed that common execution of the data mining queries dmq_0 , dmq_2 , and dmq_3 would reduce the total I/O cost by 16 units (the weight of the connecting hyperedge) compared with the sequential execution. Using the same example, it can also be noticed, that common execution of only the data mining queries dmq_1 and dmq_2 provides no cost reduction (the weight of the connecting hyperedge is zero).

**Fig. 7.** Sample gain hypergraph

The gain hypergraph can be generated using the algorithm *GenerateGainHypergraph* shown in Fig. 8. The algorithm takes two arguments: the set of all elementary data selection predicates and the set of all data mining queries. First, the algorithm builds a full hypergraph whose nodes are the data mining queries. Each hyperedge receives the initial weight of zero. Then, for each hyperedge e , we create a set P of distinct data selection formulas involved in all data mining queries connected with the hyperedge e . I/O costs for executing the distinct data selection formulas from P are then summarized and the result is assigned to the hyperedge e weight.

```
GenerateGainHypergraph(S, DMQ):
begin
  generate a full hypergraph  $G = \{V, E\}$ ,  $V = DMQ$ 
  for each  $e \in E$  do begin
     $e.gain = 0$ 
     $P = \{s_i \in S \mid \exists dmq_j \in e, dmq_j = (R, a, \Sigma_j, \Phi_j, \beta_j), s_i \subseteq \Sigma_j\}$ 
    for each  $s \in P$  do begin
       $e.gain += cost(s) * (|\{dmq_j \in e, dmq_j = (R, a, \Sigma_j, \Phi_j, \beta_j), s_i \subseteq \Sigma_j\}| - 1)$ 
    end
  end
  return  $G$ 
end
```

Fig. 8. Gain hypergraph generation algorithm

After having created the gain hypergraph, CCFull performs the following steps. All hyperedges are sorted in descending order according to their weights. Next, CCFull iterates over the hyperedges and checks if data mining queries connected with the current hyperedge have been already assigned to phases (partitions). If none of the data mining queries has been assigned so far, and if their hash trees fit in memory, then a new phase is generated and the data mining queries are assigned to it. Otherwise, if only some of the data mining queries have been already assigned to different phases, then CCFull tries to combine all those phases together with the unassigned data mining queries. If such combined phase does not fit in memory, then the current hyperedge is ignored and CCFull continues with the next one. The algorithm ends when all hyperedges are processed. The algorithm CCFull is shown in Fig. 9.

```

CCFull(G=(V,E)):
begin
  Phases = { $\emptyset$ }
  sort E =  $\langle e_1, e_2, \dots, e_k \rangle$  in desc. order w.r.t.  $e_i.gain$ , ignore edges with zero gains
  for each  $e_i$  in E do begin
    tmpV = { $v \in V \mid v \in e_i$ }
    if ( $|\{p \in Phases \mid p \cap tmpV \neq \emptyset\}| = 0$ ) then
      commonPhases =  $\emptyset$ 
      newPhase = tmpV
    else
      commonPhases =  $\{p \in Phases \mid p \cap tmpV \neq \emptyset\}$ 
      newPhase = tmpV  $\cup \bigcup_{p \in commonPhases} p$ 
    end if
    if (treesize(newPhase)  $\leq$  MEMSIZE) then
      Phases = Phases - commonPhases
      Phases = Phases  $\cup$  newPhase
    end if
  end
  add phase for each unassigned query
  return Phases
end

```

Fig. 9. CCFull

The detailed steps of the CCFull algorithm are the following. First we initialize the set of phases – we start with the empty set. In the next step we sort the list E of hyperedges from the gain graph. Hyperedges with weights equal to zero are removed from the list. Then a loop starts, which iterates over the list of hyperedges. In the first step of the loop we select all data mining queries which are connected with the current hyperedge ($tmpV$). Next we test if any of the selected data mining queries belongs to any of the phases created so far. If not, then we create a new candidate phase containing all the data mining queries from $tmpV$. Otherwise, we create a new candidate phase containing all the data mining queries from $tmpV$ and data mining queries from earlier created phases, to which any of the $tmpV$ data mining queries was also

assigned. After the process of building the new candidate phase is completed, we check if hash trees of all the data mining queries from it fit together in memory (*MEMSIZE* is the available memory size). If this condition is satisfied, then we append the new candidate phase to the current set of created phases *Phases*, possibly replacing some of the existing phases (when multiple phases are combined). Finally, when the loop is finished, for each data mining query which has not been assigned we create a new phase.

7.3 CCCoarsening

Earlier we have stated that hypergraph partitioning algorithms from other domains are not applicable to our problem due to its specifics. Nevertheless, the existing hypergraph partitioning algorithms can provide inspiration for the development of new methods. This is exactly the case with CCCoarsening which borrows ideas from the heavy edge matching method of graph coarsening in multi-level graph partitioning [33].

The CCCoarsening algorithm starts with transformation of the data sharing hypergraph into a *gain graph*, which contains (1) vertices being the original data mining queries and (2) two-vertex edges whose weights describe gains that can be reached by executing the connected queries in the same phase. The idea is to avoid the problem of an exponential number of hyperedges (with respect to the number of vertices) suffered by CCFull, which uses a gain hypergraph. The price for the above simplification is the loss of precise information on actual gains due to assigning any given subset of queries to the same phase. As a consequence, only pairs of queries/phases will be considered for merging in each step of the iterative process.

A sample gain graph for the set of three data mining queries is shown in Fig. 10. For example, putting the data mining queries dmq_1 and dmq_2 in the same phase would allow us to save 90 I/O cost units (e.g., disk blocks).

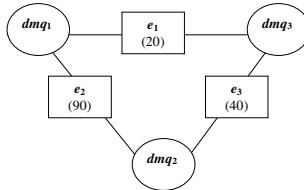


Fig. 10. Sample gain graph

The heavy edge matching method [33], of which CCCoarsening is an adaptation, was designed as a method of coarsening large graphs by collapsing strongly connected vertices. The original heavy-edge matching algorithm iteratively reduces the graph by minimizing the number of its vertices. In each iteration, the algorithm looks for the maximal set of edges (called *matching*) such that it contains no pair of edges incident with the same vertex. In order to generate a matching, a vertex currently not matched is randomly selected. Then, from the set of edges incident to the selected vertex an edge of the maximal weight from the edges leading to other so far unmatched vertices is chosen. The chosen edge results in selection of its two incident vertices for matching and labeling them as „matched”. When the matching is complete, the edges forming it are removed from the graph and each pair of vertices

connected by an edge that is to be removed are merged into one vertex, whose weight is the sum of weights of the two merged vertices. After merging vertices, some of the remaining edges may also merge as only single edge between any pair of vertices is allowed. If a set of edges is being merged into one edge, its resulting weight is the sum of the weights of the replaced edges.

Our adaptation of the heavy-edge matching algorithm for the purpose of query set partitioning for Common Counting focuses on the modification of the vertex reduction step. When an unmatched vertex is randomly selected, we will sort the edges connecting it with other unmatched vertices in the decreasing order of weights. Next, we choose the edge with the highest weight and check if the weight of a new vertex that would be created by collapsing the selected edge does not exceed the limit on the vertex weight (representing the amount of available memory to store hash trees). If not, the pair of vertices connected by the selected edge is merged. Otherwise, the edge next in order is chosen from the sorted list of edges incident to the randomly selected vertex. If none of edges incident to the vertex leads to a feasible merging, the vertex is not merged but still it is marked as matched so it will not be considered again in the current iteration of the graph coarsening process. The algorithm finishes work if no feasible matching of vertices can be found in a new coarsening iteration. The vertices of the resulting reduced graph represent partitions corresponding to phases of our original problem.

7.4 CCAgglomerative

Similarly to CCCoarsening, the CCAgglomerative algorithm first transforms the data sharing graph into a gain graph, but then uses a different method of grouping queries into phases. In CCCoarsening a matched vertex is not considered for subsequent merges until the matching in the present coarsening iteration is completed. CCAgglomerative does not pose such a restriction.

```

CCAgglomerative( $G=(V,E)$ , E contains 2-node edges only):
begin
  Phases =  $\emptyset$ 
  for each  $v$  in  $V$  do Phases = Phases  $\cup$  { $\{v\}$ }
  sort E = { $e_1, e_2, \dots, e_k$ } in descending order w.r.t.  $e_i.gain$ , ignore edges with zero gains
  for each  $e_i = (v_1, v_2)$  in  $E$  do begin
    phase1 = p  $\in$  Phases such that  $v_1 \in p$ 
    phase2 = p  $\in$  Phases such that  $v_2 \in p$ 
    if treesize(phase1  $\cup$  phase2)  $\leq$  MEMSIZE then
      Phases = Phases - {phase1}
      Phases = Phases - {phase2}
      Phases = Phases  $\cup$  {phase1  $\cup$  phase2}
    end if
  end
  return Phases
end

```

Fig. 11. CCAgglomerative

CCAgglomerative starts with an initial partitioning created by putting each data mining query into a separate phase. Next, the algorithm processes the edges sorted with respect to the decreasing weights. For each edge, the algorithm tries to combine

phases containing the connected data mining queries into one phase. If the total size of all the data mining queries in such phase does not exceed the memory size, the original phases are replaced with the new one. Otherwise the algorithm simply ignores the edge and continues. The CCAgglomerative algorithm is shown on Fig. 11.

7.5 CCAgglomerativeNoise

Algorithm CCAgglomerative is a heuristics that suffers from the same problem as classical greedy algorithms. (We will present a greedy algorithm for our problem in Sect. 7.6 and then improve it in Sect. 7.7.) Merging phases connected by the heaviest edge in each iteration may not always lead to the optimal assignment of queries to phases. Let us consider an example gain graph representing a batch of queries shown in Fig. 12.

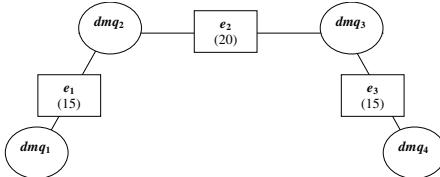


Fig. 12. Example gain graph for which CCAgglomerative misses the optimal solution

Assume that in a certain iteration of Common Counting the sizes of candidate hash-trees are 20 KB for all four queries, and the amount of available memory is 40KB, which means that no more than two queries can be processed in one phase. In such a case, CCAgglomerative would start with assigning dmq_2 and dmq_3 to the same phase, and then dmq_1 and dmq_4 would be assigned to different phases. The reduction in number of disk blocks read, compared to sequential execution, would be 20 blocks. Obviously, the optimal solution is to execute dmq_1 and dmq_2 in one phase and dmq_3 and dmq_4 in another, leading to the gain of 30 blocks.

To give the partitioning algorithm a chance of finding an optimal assignment, we propose to randomize the graph by randomly modifying weights of graph edges within a user-specified window (expressed in percents, e.g., $\pm 10\%$), and then execute the unmodified CCAgglomerative algorithm on a modified gain graph⁷. The procedure of randomizing the graph and partitioning should be repeated a user-specified number of times, each time starting with the original gain graph. We call the extended partitioning algorithm CCAgglomerativeNoise as it introduces some “noise” into the graph model of the batch of queries, before performing actual partitioning. For the noise of $x\%$, in a randomized gain graph the modified weight $e.gain'$ of each edge e will be a random number from the range $<e.gain-x\%*e.gain, e.gain+x\%*e.gain>$, where $e.gain$ is the original weight of the edge e .

⁷ Iterative execution of a partitioning heuristics over a randomized data sharing model could also be considered for the partitioning algorithms presented in previous sections. We implemented this idea in the context of CCAgglomerative as at the time it was the most efficient of the algorithms proposed so far.

To illustrate a potential usefulness of CCAgglomerativeNoise let us go back to the example gain graph from Fig. 12. For the noise of 20%, in each iteration of CCAgglomerativeNoise modified values of edge weights would be from the following ranges: $e_1.gain' \in <12,18>$, $e_2.gain' \in <16,24>$, and $e_3.gain' \in <12,18>$. So, it is possible that in some iteration of CCAgglomerativeNoise we would have $e_1.gain' > e_2.gain'$ or $e_3.gain' > e_2.gain'$ (e.g., $e_1.gain' = 18$, $e_2.gain' = 16$, and $e_3.gain' = 13$), in which case the basic CCAgglomerative partitioning procedure would find the optimal assignment of queries to Common Counting phases.

We should note that the CCAgglomerativeNoise method should be treated as a means of improving the results of pure CCAgglomerative. In other words, the initial iteration of CCAgglomerativeNoise should always be on the original gain graph. This way it can be guaranteed that CCAgglomerativeNoise will never generate worse partitionings than CCAgglomerative.

7.6 CCGreedy

The general greedy strategy can be applied to solve the hypergraph partitioning problem representing query set partitioning for Common Counting by starting with each query in a separate partition and then iteratively merging pairs of partitions, greedily choosing the two partitions whose merging results in greater improvement of the partitioning objective and at the same time does not violate the partitioning constraint. This leads to the CCGreedy algorithm presented in Fig. 13.

```

CCGreedy(GG=(V,E)):
begin
while (true) begin
    sort E in descending order w.r.t.  $e_i.gain$ , ignore edges with zero gains
     $newPartition = \emptyset$ 
    for each  $e_i = \{v_x, v_y\}$  in E do
        if ( $treesize(e_i) \leq MEMSIZE$ ) then
             $newPartition = v_x \cup v_y$ 
             $V = V \setminus \{v_x, v_y\}$ 
             $V = V \cup \{newPartition\}$ 
             $E = E \setminus e_i$ 
            for each  $v$  in  $V$  do begin
                 $newEdge = \{v, newPartition\}$ ; compute  $newEdge.gain$ 
                 $E = E \cup \{newEdge\}$ 
            end
            break
        end if
    end
    if  $newPartition = \emptyset$  then break end if
end
return  $V$ 
end

```

Fig. 13. CCGreedy

To represent the gain in the partitioning objective for all pairs of partitions the algorithm maintains a *gain graph* $GG=(V, E)$, which is a fully connected graph whose nodes represent partitions and each edge weight represents the gain thanks to merging a pair of partitions connected by the edge. The gain is computed as the difference between the values of partitioning objectives after and before merging a given pair of queries.

It should be noted that while CCGoarsening and CCAgglomerative also use the same gain graph structure as CCGreedy, the advantage of CCGreedy is that it updates the gain graph after merging partitions so that the graph always reflects possible gains due to partition merging.

7.7 CCSemiGreedy

An obvious problem with greedy algorithms like CCGreedy is that the locally optimal choice in each operation may not lead to the globally optimal solution. To increase the chances of finding the optimal partitioning we modify CCGreedy by applying a semi-greedy strategy [21] to it. The result is the CCSemiGreedy algorithm depicted in Fig.14.

```

CCSemiGreedy(GG=(V,E), RCLLen):
begin
  while (true) begin
    sort E in desc. order w.r.t.  $e_i.gain$ ,
    ignore edges with zero gains
    newPartition =  $\emptyset$ 
    RCL = genRCL(GG, RCLLen)
    if length(RCL) = 0 then break end if
    randomly choose  $e_i = \{v_x, v_y\}$  from RCL
    newPartition =  $v_x \cup v_y$ 
    V =  $V \setminus \{v_x, v_y\}$ 
    V =  $V \cup \{newPartition\}$ 
    E =  $E \setminus e_i$ 
    for each v in V do begin
      newEdge = {v, newPartition}
      compute newEdge.gain
      E =  $E \cup \{newEdge\}$ 
    end
  end
  return V
end

function  $genRCL(GG=(V,E), RCLLen):$ 
begin
  RCL = nil
  for each  $e_i = \{v_x, v_y\}$  in E do
    if (treesize( $e_i$ )  $\leq$  MEMSIZE) then
      RCL = append(RCL,  $e_i$ )
      if length(RCL) = RCLLen then
        break
      end if
    end if
  end
  return RCL
end

```

Fig. 14. CCSemiGreedy

CCSemiGreedy differs from CCGreedy in the step of choosing the partitions to merge. CCSemiGreedy uses restricted candidate list (RCL) which is returned by the function $genRCL$. This procedure iterates over the gain graph and checks if hash trees of all the queries from a given pair of partitions fit together in memory. If this condition is satisfied, the current edge is added to the RCL . Generation of the RCL is stopped when the list reaches the length of $RCLLen$ (set by a user). In CCSemiGreedy

we check the length of the *RCL*. If it is zero, there is no possible merge, otherwise an edge (for partition merging) is chosen randomly from the *RCL*. Other steps of the CCSemiGreedy algorithm are the same as those described for CCGreedy algorithm.

In practice, CCSemiGreedy should be applied to query partitioning in the following way. Firstly, an initial partitioning should be generated with CCGreedy. Then, CCSemiGreedy should be executed a user-defined number of times. In the end, the best of the generated partitionings should be used for Common Counting.

8 Experimental Results

This section contains the results of experiments that we conducted to compare the proposed query partitioning algorithms for Common Counting. Due to the large number of the algorithms to compare, the experiments were divided into a couple of stages. The first stage was devoted to comparison of basic versions of the partitioning algorithms dedicated to our particular problem and developed with the goal of solving it in mind, i.e. CCRecursive, CCFull, CCCoarsening, and CCAgglomerative. The second stage contained experiments aimed at comparing the best algorithm selected in the first stage, which turned out to be CCAgglomerative, with CCGreedy and CCSemiGreedy, which are adaptations of the universal greedy and semi-greedy metaheuristics. Also at that stage CCAgglomerativeNoise, an extension of CCAgglomerative, somewhat analogous to CCSemiGreedy with respect to CCGreedy, was included in the tests.

In both stages, the experiments were performed on a synthetic dataset generated with GEN [2]. The dataset had the following characteristics: number of transactions = 500000, average number of items in a transaction = 4, number of different items = 10000, number of patterns = 1000. The size of the dataset in a textual form was about 16MB. We stored the dataset in a local PostgreSQL database, where it consumed about 85MB of disk space + extra 43 MB used for a B-tree index.

Batches of frequent itemset were generated with our own random generator parameterized with a desired average percentage of dataset overlapping between pairs of queries from a batch. The minimum support (frequency) threshold of all the queries was always set to 0.75%.

For the above support threshold the hash trees usually had the size of tens of kilobytes. In order to introduce the need for query set partitioning, we deliberately limited the amount of memory available for Common Counting executions to 120kB (different values were also used in one of the experiments)⁸. Obviously, typically for sparse datasets, as those generated with GEN, the hash trees were biggest in the second and third Apriori iteration, and getting smaller with each subsequent iteration. As a consequence, the need for query set partitioning was observed only for some of the Common Counting iterations, which is typical for real-life scenarios with datasets whose characteristics the GEN generator tries to mimic.

⁸ Alternatively, we could have changed the parameters of the GEN generator and/or decrease the minimum support threshold of the queries. The advantage of our approach was that it kept the execution times reasonable without the loss of generality.

8.1 Comparison of Basic Dedicated Algorithms

In this stage of experiments we tested the four dedicated algorithms: CCRecursive, CCFull, CCCoarsening, and CCAgglomerative, comparing them to two extra algorithms provided as reference points: an exact algorithm (denoted as “Exact” in the charts) enumerating all feasible partitionings by performing a brute-force search and an algorithm randomly assigning queries to partitions so that the partitioning constraint is satisfied denoted as “Random” in the charts). These two extra algorithms were ultimate choices to provide the reference points for judging usefulness of our proposed algorithms as:

- a heuristic algorithm should generate solutions as close to those returned by an exact algorithm,
- an algorithm generating solutions worse than generated randomly is obviously useless.

During the experiments we measured total execution time, time consumed by a partitioning algorithm, the number of partitions in partitionings, and the number of disk blocks read in a Common Counting iteration for a generated query set partitioning. The experiments were conducted on a PC with AMD Athlon 1400+ processor and 384 MB of RAM running Windows XP. The data resided in a local PostgreSQL database, the algorithms were implemented in C#.

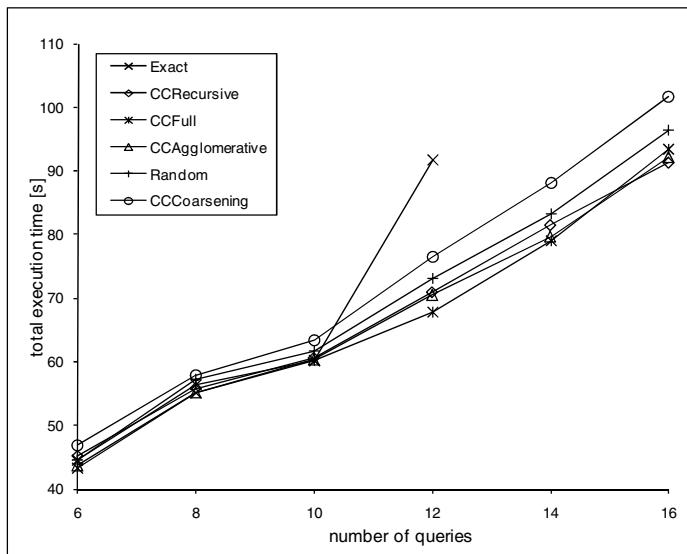


Fig. 15. Total execution time of a batch of queries (40% overlapping, 120kB of memory)

Figure 15 presents total execution times of a batch of randomly generated queries using the Common Counting method equipped with different query set partitioning algorithms. The queries were generated so that the average overlapping between their datasets was 40%. The memory available was limited to 120kB. The number of queries

varied from 6 to 16. The exact algorithm finished in reasonable time only for up to 12 queries (it did not complete in 900s for the case of 14 queries), which is a practical confirmation of our theoretical analysis suggesting that in order to support large batches of queries heuristic partitioning algorithms are required. The two best algorithms in terms of overall processing time of Common Counting are CCAgglomerative and CCFull, with CCRecursive not far behind. However, while CCAgglomerative has polynomial computational complexity with respect to the number of queries, the complexity of CCFull is exponential. This is due to the fact that CCAgglomerative iterates over edges in a connected graph, while CCFull does the same for hyperedges in a hypergraph. The result is observed deterioration of execution times with the increasing number of queries when CCFull was applied, compared to the characteristic of CCAgglomerative. The worst of the four proposed algorithms is CCCoarsening, which performed worse than the random approach.

Obviously, the total execution times are what matters in the end. Nevertheless, to provide an insight into reasons of the overall performance of the tested algorithms, we also analyzed separately the two factors that contributed to differences in the total execution times: the time consumed by partitioning algorithms themselves and the quality of generated partitionings, measured as the number of disk blocks read in a Common Counting iteration due to a generated partitioning.

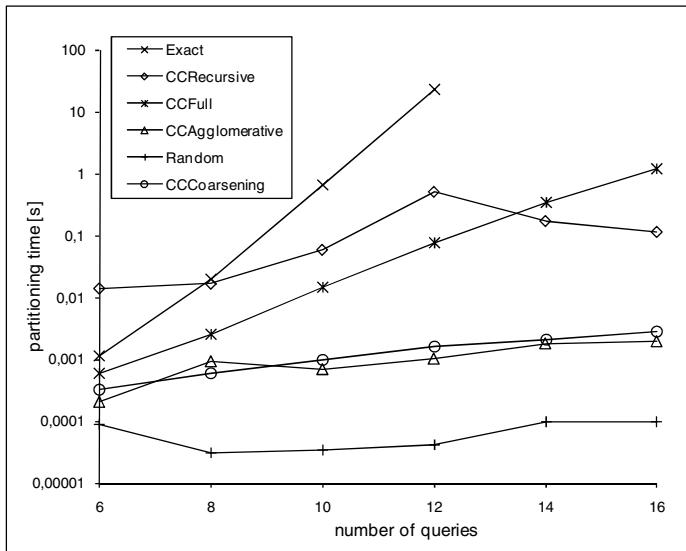


Fig. 16. Partitioning execution time (40% overlapping, 120kB of memory)

The execution times of partitioning algorithms measured in the experiment are depicted in Fig. 16. Due to large disproportions between the algorithms, the chart uses logarithmic scale. The chart confirms that the execution times of the exact algorithm and CCFull grow exponentially with the number of queries, which prohibits application of both to large batches of queries. Still, as far as the partitioning time in a concern, CCFull is an improvement over the exact algorithm as it completed within a

second even for 16 queries, which is acceptable as almost negligible compared to the overall execution time of Common Counting. The fastest algorithms (apart from the random approach) are CCAgglomerative, and CCCoarsening. CCRecursive is the least predictive of the algorithms and for small batches of queries is even slower than CCFull.

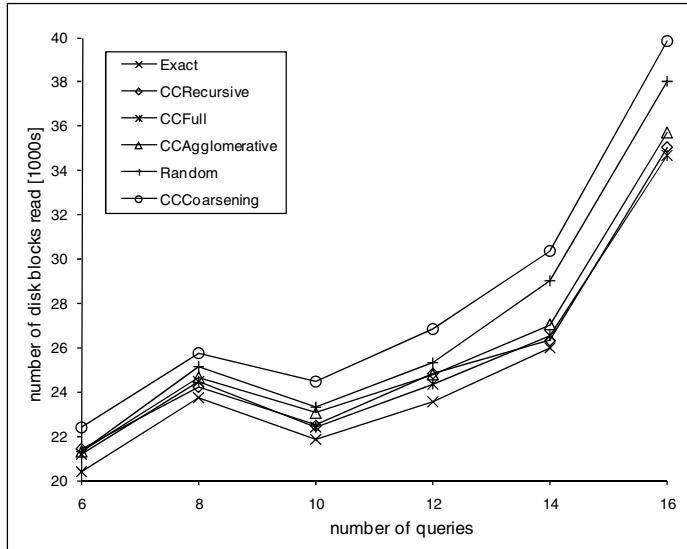


Fig. 17. Number of disk blocks read for a generated partitioning (40% overlapping, 120kB of memory)

Figure 17 shows the numbers of disk blocks read in a Common Counting iteration due to a partitioning generated with a given algorithm, which reflects the accuracy of the partitioning algorithms. The results are generally consistent with the total execution times of Common Counting as the time spend on partitioning was a small fraction of the time spent on database scans. CCFull, CCAgglomerative, and CCRecursive generate partitioning of similar quality, within 5% of those generated by the exact algorithm, while CCCoarsening is worse than the random algorithm. As for the influence of the number of queries on the quality of partitioning, the most important observation is that the relative accuracy of CCFull, CCAgglomerative, CCRecursive, and CCCoarsening with respect to exact and random algorithms was roughly the same for all sizes of query batches. The absolute number of disk blocks read does not necessarily rise with the increase of the number of queries as the complexity of the data sharing hypergraph does not have to be greater for a larger number of randomly generated queries. Nevertheless, still such a tendency can be observed despite the aberration for the case of 8 queries.

Figure 18 provides an explanation of poor accuracy of CCCoarsening. The average number of partitions in partitionings generated by the tested algorithms is presented. Evidently, CCCoarsening is worse than the rest of the algorithms by 0.5 to 1 iteration on average. The reason for this is the origin of the concept underlying the CCCoarsening algorithm. When the actual goal is coarsening the graph, having partitions of similar size is desired to preserve the general structure of a graph. However, in the case of our problem, the fact that the algorithm was focusing on growing several partitions at the same pace reduces the possibility of fully exploiting the memory available as merging partitions into larger ones becomes impossible quicker than in case of other algorithms. Thus, the possibilities of minimizing our partitioning criterion are reduced in the case of CCCoarsening for the sake of balancing the sizes of partitions, which is irrelevant for our problem.

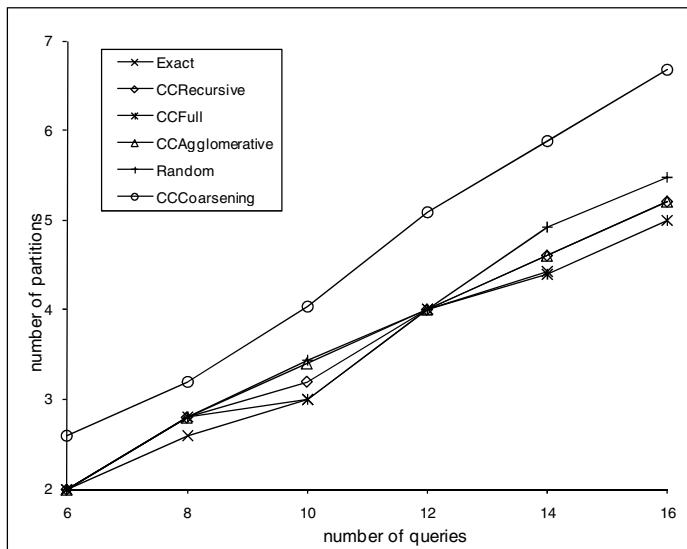


Fig. 18. Average number of partitions in a partitioning (40% overlapping, 120kB of memory)

The last goal of this stage of experiments was testing the impact of the level of dataset overlapping between the queries within a batch on the performance of Common Counting with various partitioning algorithms. Figures 19 and 20 show the total execution times for four different average levels of query dataset overlapping with 10 and 40 queries in a batch respectively (for 40 queries the exact algorithm and CCFull could not complete within reasonable time and therefore are not included in the results).

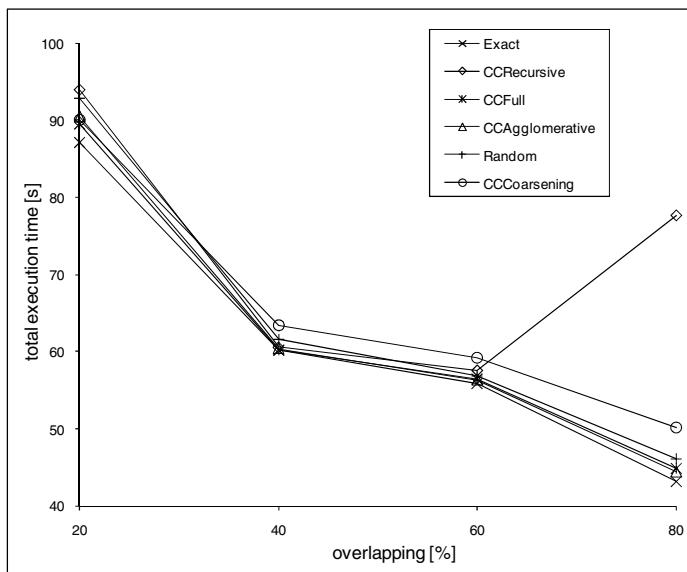


Fig. 19. Total execution time of a batch of queries (10 queries, 120kB of memory)

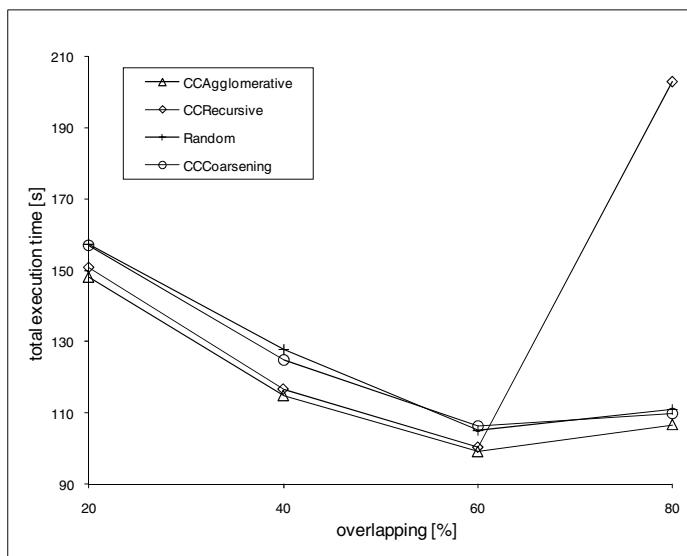


Fig. 20. Total execution time of a batch of queries (40 queries, 120kB of memory)

In general, the results are consistent with expectations as the more significant the overlapping the better chances of reducing the I/O cost of database scanning. A negative surprise is a significant performance degradation of CCRecursive for the average overlapping of 80% (the rest of algorithms exhibit slight performance degradation only for the case of 40 queries as compared with the overlapping of 60%). We believe that such a behavior can be explained by the recursive nature of CCRecursive. When the queries almost completely overlap, there is a large number of elementary data selection predicates corresponding to small dataset partitions, share by a large number of queries. As a result, CCRecursive violates the partitioning constraint while there are still many elementary data selection predicates to process (recall that CCRecursive is the only of the algorithms working with the predicates one by one). This leads to a lot of recursive calls due to attempts of creating partitions exceeding the size limit and consequently deteriorates CCRecursive's performance.

The overall conclusion from this stage of experiments is that the best out of four proposed algorithms dedicated to our query set partitioning problem is CCAgglomerative as one of the two algorithms tied for the first place both in terms of partitioning time and quality.

8.2 Comparison of Greedy Approaches with the Best Dedicated Algorithms

The goal of the second stage of experiments was comparison of the best of dedicated partitioning algorithms, which turned out to be CCAgglomerative, with an implementation of a greedy strategy – CCGreedy. Also included in the tests and evaluated were their extensions, namely: CCAgglomerativeNoise and CCSemiGreedy. Before the actual tests of the partitioning speed and accuracy of the compared algorithms, the optimal values of parameters responsible for the chance of improving the initial solution of the two basic compared algorithms had to be determined. All the tested algorithms were implemented C#. The experiments were conducted on a PC with Intel Pentium IV 2.53GHz processor and 512MB of RAM running Windows XP.

We started the experiments with simulations, performed to determine influence of CCSemiGreedy parameters (RCL length and number of attempts) on its effectiveness. We simulated batches of data mining queries by randomly generating the database predicate and size of the candidate tree for each query. Size of available memory was randomly generated in such way that at least every single query could fit into memory. Series of simulations consisted of 500 iterations to get average values and were applied to batches of queries ranging from 3 to 50 queries per batch.

Figure 21 presents the influence of chosen RCL length on the number of disk blocks read by CCSemiGreedy. The experiments indicate that the length of the RCL should be very small but greater than 2 items. Best results were obtained for 3 to 6 items. For further experiments we have chosen the length of RCL equal to 3.

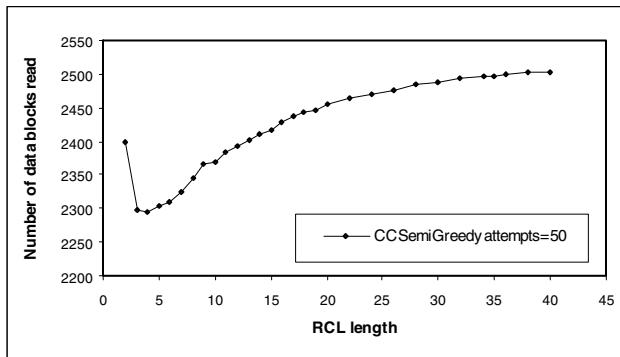


Fig. 21. Influence of the RCL length on the overall accuracy of CCSemiGreedy

Figure 22 presents influence of the second parameter of CCSemiGreedy, which is the number of attempts to generate a partitioning. It is obvious that more attempts generally will result in better partitionings but at the expense of increasing the partitioning time. Results indicate that after more than fifty attempts there is no significant improvement in the quality of the partitioning.

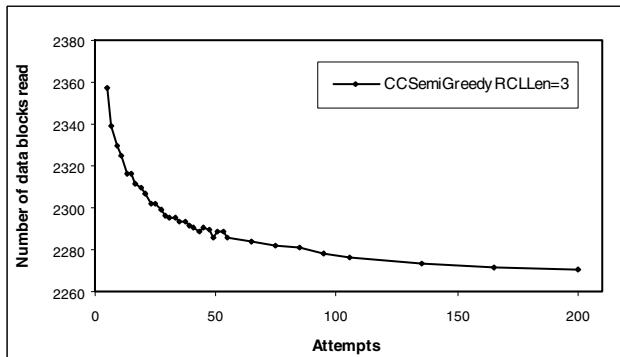


Fig. 22. Influence of the number of attempts on the overall accuracy of CCSemiGreedy

CCAgglomerativeNoise iteratively tries to improve the partitioning generated by CCAgglomerative in a similar way as CCSemiGreedy extends CCGreedy and is also parameterized by the number of iterations. We set the number of attempts to 150 for CCAgglomerativeNoise because this value resulted in CCSemiGreedy and CCAgglomerativeNoise consuming roughly equal time to generate the partitioning for the average size of batches used in the planned experiments. For that number of iterations we determined the optimal value of the noise parameter of CCAgglomerativeNoise in similar simulations to those carried for RCL length of CCSemiGreedy. The influence of the noise parameter on CCAgglomerativeNoise was analogous to that of RCL

length on CCSemiGreedy. The optimal value of noise turned out to be 3%, which is relatively small.

Knowing the optimal values of parameters of CCSemiGreedy and CCAgglomerativeNoise, we used these values in the subsequent experiments in which we compared CCGreedy and CCSemiGreedy algorithms with CCAgglomerative and CCAgglomerativeNoise in terms of effectiveness (quality of generated partitionings) and efficiency (partitioning times). In these experiments we randomly generated batches of 5 to 30 queries, operating on subsets of the test database.

Figure 23 presents how the accuracy of the partitioning algorithms changes with the number of queries. To improve readability of the chart, we present relative amount of data blocks read as result of partitionings generated by CCGreedy, CCSemiGreedy and CCAgglomerativeNoise with respect to CCAgglomerative⁹. Experiments were performed for three values of the main memory limit (90, 120, and 150kB) and for four levels of the average overlapping of datasets read by queries in the set (20%, 40%, 60%, and 80%). The results presented are averages taken over all the conducted experiments. Results show that the most effective partitionings are generated by CCSemiGreedy and are about 5% better than those generated by CCAgglomerative. For CCAgglomerativeNoise and CCGreedy the measured improvement over CCAgglomerative was 2% and 1% respectively.

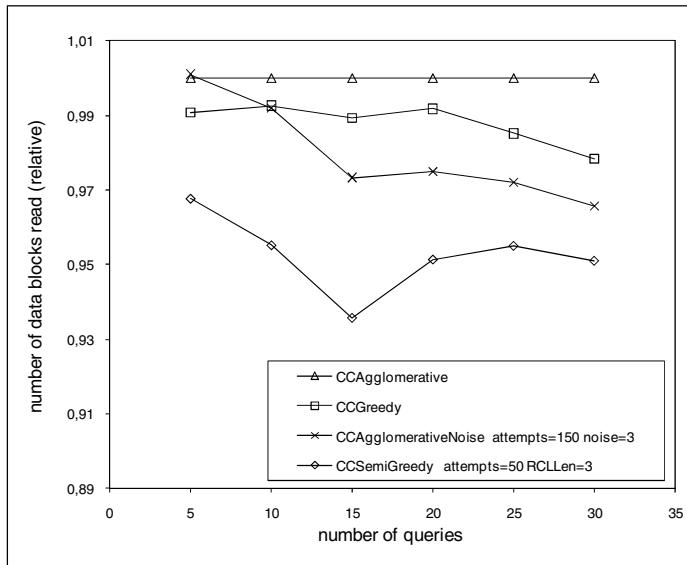


Fig. 23. Amounts of data read by different partitionings

⁹ For this stage of experiments we do not present the total execution times of Common Counting because the difference between the algorithms would be difficult to notice from the chart since the difference in the execution times among the algorithms was two orders of magnitude smaller than the difference between execution times for the smallest and largest of query batches.

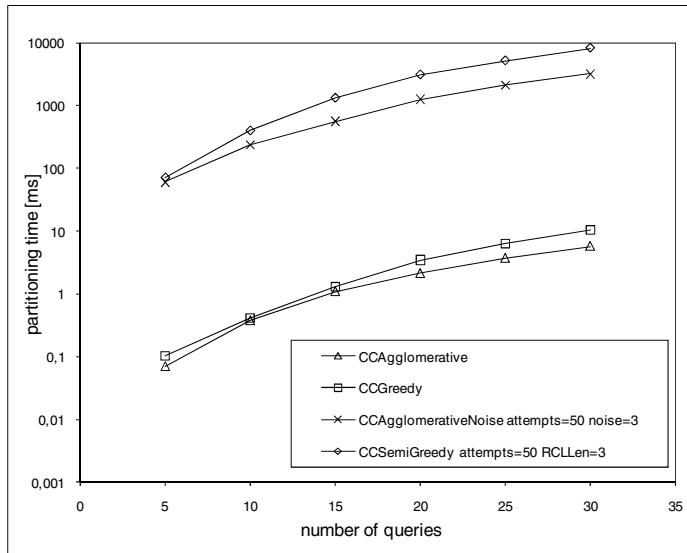


Fig. 24. Partitioning times (logarithmic scale)

Figure 24 presents execution times of the considered partitioning algorithms. This time for CCSemiGreedy and CCAgglomerativeNoise numbers of attempts were fixed at the same level (50). Execution times of CCAgglomerative and CCGreedy are negligible, with CCGreedy requiring at most twice as much time as CCAgglomerative. Execution times of CCSemiGreedy are up to three times longer than those of CCAgglomerativeNoise and the gap increases with the number of queries.

The results of our experiments show that CCGreedy is more effective than CCAgglomerative, and properly parameterized CCSemiGreedy generates better partitions than CCAgglomerativeNoise, which makes it the best partitioning algorithm for Common Counting. The execution times of the new algorithms are longer but in typical situations the increase in partitioning time will be dominated by the reduction of the time spent on disk operations thanks to better partitions.

9 Review of Other Methods of Processing Sets of Frequent Itemset Queries

While Common Counting is the most fundamental, and at the same time predictable in terms of offered performance gains with respect to sequential execution, method of processing sets of frequent itemset queries, it is not the only possible solution of the considered problem. Two general approaches have been taken to design methods of processing batches of frequent itemset queries: (1) providing methods independent from a particular frequent pattern mining algorithm, and (2) tailoring dedicated methods for the two most prominent frequent pattern mining algorithms, i.e., Apriori and FP-growth. Obviously, Common Counting is a representative of the second approach.

The first method independent of the mining algorithm was Mine Merge [55] which transformed the original batch of queries into a set of intermediate queries operating on non-overlapping parts of the database. The results of these intermediate queries were used to answer the original queries during a verifying pass over the database. Mine Merge was shown to scale poorly with the number of queries due to exponential growth of the number of resulting intermediate queries. Moreover, it requires significant overlapping among the queries' datasets in order to compensate the extra database pass. The latter problem has been solved by a modified version of Mine Merge, called Partition Mine Merge Improved [17], which generated intermediate queries whose source datasets could be cached in memory and thus required exactly two scans of the database to process the batch of queries. The price for the reduction of I/O was the increase in the number of intermediate queries, resulting in the increased amount of in-memory computations.

Following Common Counting, two methods offering tighter integration of processing among the concurrently executed queries were proposed for Apriori. Common Candidate Tree [16] replaced individual hash trees with one integrated data structure shared by all the queries, thus reducing the memory consumption and optimizing the candidate counting step of Apriori. Later, Common Candidates [29] integrated also the candidate generation step of Apriori, while preserving all the optimizations proposed earlier in Common Counting and Common Candidate Tree. Unfortunately, the successors of Common Counting do not preserve its capability of handling large batches of queries by partitioning them into phases, thus being restricted by the limit of memory available for the integrated in-memory data structure. Furthermore, Common Candidates limits the possibilities of constraint handling due to replacing the original candidate generation procedure of Apriori which serves as the basis for most of the constraint-handling techniques within the Apriori framework.

As for FP-growth, the methods of processing sets of frequent itemset queries using this algorithm evolved analogously to the ones for Apriori [50]. The initial proposal was Common Building, a direct adaptation of Common Counting, which integrated database scans performed by the queries in order to build their FP-tree structures in main memory. The method was immediately extended by introducing a variation of FP-tree that could be shared by the batch of queries, resulting in the Common FP-tree method.

10 Conclusions

In this chapter we considered the problem of processing sets of frequent itemset queries, which brings the ideas of multiple-query optimization to the domain of data mining as a natural consequence of previous research on optimizing individual frequent itemset queries and sequences of frequent itemset queries.

We provided a general model of frequent itemset queries, which was then used as a basis for the formulation of our multi-query optimization problem. From the algorithms dedicated to solve the problem that we had proposed over the last decade, here we focused on the most fundamental and predictable method, called Common Counting, which consists in concurrent execution of the queries using Apriori with the integration of scans of the parts of the database shared among the queries.

The major advantage of Common Counting over its alternatives is its applicability to arbitrarily large batches of queries. In order to achieve that feature, Common Counting had to be accompanied with a method of dealing with situations when hash trees of all the queries do not fit together in main memory. The problem of limited memory was addressed by partitioning the set of queries into subsets processed in several phases. This approach led to an interesting optimization problem that we formalized as a specific case of hypergraph partitioning. Since the problem is NP-hard, it has to be solved by heuristic algorithms in case of large batches of queries.

For the identified, specific hypergraph partitioning problem, we provided a comprehensive overview of query set partitioning algorithms proposed by us so far: CCRecursive, CCFull, CCCoarsening, CCAgglomerative, CCAgglomerativeNoise, CCGreedy, and CCSemiGreedy.

Finally, we presented extensive results of experiments aimed at evaluating the performance and accuracy of the algorithms. The results indicate that the implementation of the universal greedy metaheuristics (CCGreedy) and its semi-greedy extension (CCSemiGreedy) generate better partitionings in terms of resulting I/O costs of Common Counting than those generated by the best algorithms dedicated to our query set partitioning problem, while offering satisfactory (but not shortest) execution times.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Databases. In: Buneman, P., Jajodia, S. (eds.) Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207–216. ACM Press, New York (1993)
2. Agrawal, R., Mehta, M., Shafer, J., Srikant, R., Arning, A., Bollinger, T.: The Quest Data Mining System. In: Simoudis, E., Han, J., Fayyad, U. (eds.) Proceedings of the Second International Conference on Knowledge Discovery in Databases and Data Mining, pp. 244–249. AAAI Press, Menlo Park (1996)
3. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann, San Francisco (1994)
4. Alpert, C.J., Kahng, A.B.: Recent Directions in Netlist Partitioning: A Survey. *Integration: The VLSI Journal* 19, 1–81 (1995)
5. Alsabagh, J.R., Raghavan, V.V.: Analysis of common subexpression exploitation models in multiple-query processing. In: Rusinkiewicz, M. (ed.) Proceedings of the 10th International Conference on Data Engineering, pp. 488–497. IEEE Computer Society, Los Alamitos (1994)
6. Baralis, E., Psaila, G.: Incremental Refinement of Mining Queries. In: Mohania, M., Tjoa, A.M. (eds.) DaWaK 1999. LNCS, vol. 1676, pp. 173–182. Springer, Heidelberg (1999)
7. Blockeel, H., Dehaspe, L., Demoen, B., Janssens, G., Ramon, J., Vandecasteele, H.: Improving the Efficiency of Inductive Logic Programming Through the Use of Query Packs. *Journal of Artificial Intelligence Research* 16, 135–166 (2002)
8. Boinski, P., Jozwiak, K., Wojciechowski, M., Zakrzewicz, M.: Improving Quality of Agglomerative Scheduling in Concurrent Processing of Frequent Itemset Queries. In: Klopotek, M.A., Wierzchon, S.T., Trojanowski, K. (eds.) Proceedings of the International IIS: IIPWM 2006 Conference, pp. 233–242. Springer, Heidelberg (2006)

9. Boinski, P., Jozwiak, K., Wojciechowski, M., Zakrzewicz, M.: Estimating Hash-Tree Sizes in Concurrent Processing of Frequent Itemset Queries. *International Journal of Information Technology and Intelligent Computing* 1, 405–417 (2006)
10. Boinski, P., Wojciechowski, M., Zakrzewicz, M.: A Greedy Approach to Concurrent Processing of Frequent Itemset Queries. In: Tjoa, A.M., Trujillo, J. (eds.) *DaWaK 2006*. LNCS, vol. 4081, pp. 292–301. Springer, Heidelberg (2006)
11. Ceri, S., Meo, R., Psaila, G.: A New SQL-like Operator for Mining Association Rules. In: Vijayaraman, T.M., Buchmann, A.P., Mohan, C., Sarda, N.L. (eds.) *Proceedings of the 22th International Conference on Very Large Data Bases*, pp. 122–133. Morgan Kaufmann, San Francisco (1996)
12. Cheung, D.W., Han, J., Ng, V.T., Wong, C.Y.: Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. In: Su, S.Y.W. (ed.) *Proceedings of the 12th International Conference on Data Engineering*, pp. 106–114. IEEE Computer Society, Los Alamitos (1996)
13. Cheung, D.W., Lee, S.D., Kao, B.: A General Incremental Technique for Maintaining Discovered Association Rules. In: Topor, R.W., Tanaka, K. (eds.) *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications*, pp. 185–194. World Scientific, Singapore (1997)
14. Garey, M.R., Johnson, D.S.: *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco (1979)
15. Goethals, B., Van den Bussche, J.: On supporting interactive association rule mining. In: Kambayashi, Y., Mohania, M., Tjoa, A.M. (eds.) *DaWaK 2000*. LNCS, vol. 1874, pp. 307–316. Springer, Heidelberg (2000)
16. Grudzinski, P., Wojciechowski, M.: Integration of candidate hash trees in concurrent processing of frequent itemset queries using Apriori. *Control and Cybernetics* 38, 47–65 (2009)
17. Grudzinski, P., Wojciechowski, M., Zakrzewicz, M.: Partition-Based Approach to Processing Batches of Frequent Itemset Queries. In: Larsen, H.L., Pasi, G., Ortiz-Arroyo, D., Andreasen, T., Christiansen, H. (eds.) *FQAS 2006*. LNCS (LNAI), vol. 4027, pp. 479–488. Springer, Heidelberg (2006)
18. Han, J., Fu, Y., Wang, W., Chiang, J., Gong, W., Koperski, K., Li, D., Lu, Y., Rajan, A., Stefanovic, N., Xia, B., Zaiane, O.: DBMiner: A System for Mining Knowledge in Large Relational Databases. In: Simoudis, E., Han, J., Fayyad, U.M. (eds.) *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 250–255. AAAI Press, Menlo Park (1996)
19. Han, J., Fu, Y., Wang, W., Koperski, K., Zaiane, O.: DMQL: A data mining query language for relational databases. In: Jagadish, H.V., Mumick, I.S. (eds.) *Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pp. 27–33. ACM Press, New York (1996)
20. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Chen, W., Naughton, J.F., Bernstein, P.A. (eds.) *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 1–12. ACM Press, New York (2000)
21. Hart, J.P., Shogan, A.W.: Semi-greedy Heuristics: An Empirical Study. *Operations Research Letters* 6, 107–114 (1987)
22. Hipp, J., Guntzer, U.: Is pushing constraints deeply into the mining algorithms really what we want? - An alternative approach for association rule mining. *ACM SIGKDD Explorations Newsletter* 4, 50–55 (2002)
23. Imielinski, T., Mannila, H.: A Database Perspective on Knowledge Discovery. *Communications of the ACM* 39, 58–64 (1996)

24. Imielinski, T., Virmani, A.: MSQL: A Query Language for Database Mining. *Data Mining and Knowledge Discovery* 3, 373–408 (1999)
25. Imielinski, T., Virmani, A., Abdulghani, A.: Discovery board application programming interface and query language for database mining. In: Simoudis, E., Han, J., Fayyad, U. (eds.) *Proceedings of the Second International Conference on Knowledge Discovery in Databases and Data Mining*, pp. 20–26. AAAI Press, Menlo Park (1996)
26. ISO: Information technology – Database languages – SQL multimedia and application packages – Part 6: Data mining. ISO/IEC 13249-6 (2006)
27. Jain, S., Swamy, C., Balaji, K.: Greedy Algorithms for k-way Graph Partitioning. In: Sinha, P.K., Das, C.R. (eds.) *Proceedings of the 6th International Conference on Advanced Computing.*, Tata McGraw Hill, New York (1998)
28. Jarke, M.: Common subexpression isolation in multiple query optimization. In: Kim, W., Reiner, D.S. (eds.) *Query Processing in Database Systems*, pp. 191–205. Springer, New York (1985)
29. Jedrzejczak, P., Wojciechowski, M.: Integrated Candidate Generation in Processing Batches of Frequent Itemset Queries Using Apriori. In: Fred, A., Filipe, J. (eds.) *Proceedings of the 2nd International Conference on Knowledge Discovery and Information Retrieval*, pp. 487–490. SciTePress (2010)
30. Jin, R., Sinha, K., Agrawal, G.: Simultaneous Optimization of Complex Mining Tasks with a Knowledgeable Cache. In: Grossman, R., Bayardo, R.J., Bennett, K.P. (eds.) *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 600–605. ACM Press, New York (2005)
31. JSR-73 Expert Group: Java Specification Request 73: Java Data Mining, JDM (2005)
32. Karypis, G.: Multilevel Hypergraph Partitioning. In: Cong, J., Shinnerl, J. (eds.) *Multilevel Optimization Methods for VLSI*. Kluwer Academic Publishers, Boston (2002)
33. Karypis, G., Kumar, V.: Multilevel Graph Partitioning Schemes. In: Banerjee, P., Boca, P. (eds.) *Proceedings of the 24th International Conference on Parallel Processing*, pp. 113–122. CRC Press, Boca Raton (1995)
34. Karypis, G., Han, E., Kumar, V.: Chameleon: A Hierarchical Clustering Algorithm Using Dynamic Modeling. *IEEE Computer* 32, 68–75 (1999)
35. Meo, R.: Optimization of a Language for Data Mining. In: *Proceedings of the 2003 ACM Symposium on Applied Computing*, pp. 437–444. ACM, New York (2003)
36. Meo, R.: Inductive Databases: Towards a New Generation of Databases for Knowledge Discovery. In: *Proceedings of the First International Workshop on Integrating Data Mining, Database and Information Retrieval*, pp. 1003–1007. IEEE Computer Society, Los Alamitos (2005)
37. Morzy, M., Wojciechowski, M., Zakrzewicz, M.: Optimizing a Sequence of Frequent Pattern Queries. In: Tjoa, A.M., Trujillo, J. (eds.) *DaWaK 2005. LNCS*, vol. 3589, pp. 448–457. Springer, Heidelberg (2005)
38. Morzy, T., Wojciechowski, M., Zakrzewicz, M.: Data Mining Support in Database Management Systems. In: Kambayashi, Y., Mohania, M., Tjoa, A.M. (eds.) *DaWaK 2000. LNCS*, vol. 1874, pp. 382–392. Springer, Heidelberg (2000)
39. Morzy, T., Wojciechowski, M., Zakrzewicz, M.: Materialized Data Mining Views. In: Zighed, D.A., Komorowski, J., Źytkow, J.M. (eds.) *PKDD 2000. LNCS (LNAI)*, vol. 1910, pp. 65–74. Springer, Heidelberg (2000)
40. Nag, B., Deshpande, P.M., DeWitt, D.J.: Using a Knowledge Cache for Interactive Discovery of Association Rules. In: Han, J. (ed.) *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 244–253. ACM Press, New York (1999)

41. Netz, A., Chaudhuri, S., Fayyad, U., Bernhardt, J.: Integrating data mining with SQL databases: OLE DB for data mining. In: Proceedings of the 17th International Conference on Data Engineering, pp. 379–387. IEEE Computer Society, Los Alamitos (2001)
42. Ng, R., Lakshmanan, L.V.S., Han, J., Pang, A.: Exploratory mining and pruning optimizations of constrained association rules. In: Tiwary, A., Haas, L.M. (eds.) Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, pp. 13–24. ACM Press, New York (1998)
43. Oracle Corporation: PL/SQL Packages and Types Reference, 10g Release 1 (10.1) (2003)
44. Pei, J., Han J.: Can We Push More Constraints into Frequent Pattern Mining? In: Ramakrishnan, R., Stolfo, S., Bayardo, R., Parsa, I. (eds.) Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 350–354. ACM Press, New York (2000)
45. Pei, J., Han, J., Lakshmanan, L.V.S.: Pushing Convertible Constraints in Frequent Itemset Mining. *Data Mining and Knowledge Discovery* 8, 227–252 (2004)
46. Roy, P., Seshadri, S., Sundarshan, S., Bhobe, S.: Efficient and Extensible Algorithms for Multi Query Optimization. In: Chen, W., Naughton, J.F., Bernstein, P.A. (eds.) Proceedings of 2000 ACM SIGMOD International Conference on Management of Data, pp. 249–260. ACM Press, New York (2000)
47. Sellis, T.K.: Multiple Query Optimization. *ACM Transactions on Database Systems* 13, 23–52 (1988)
48. Srikant, R., Vu, Q., Agrawal, R.: Mining association rules with item constraints. In: Heckerman, D., Mannila, H., Pregibon, D. (eds.) Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pp. 67–73. AAAI Press, Menlo Park (1997)
49. Thomas, S., Bodagala, S., Alsabti, K., Ranka, S.: An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases. In: Heckerman, D., Mannila, H., Pregibon, D. (eds.) Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pp. 263–266. AAAI Press, Menlo Park (1997)
50. Wojciechowski, M., Galecki, K., Gawronek, K.: Three Strategies for Concurrent Processing of Frequent Itemset Queries Using FP-growth. In: Džeroski, S., Struyf, J. (eds.) KDID 2006. LNCS, vol. 4747, pp. 240–258. Springer, Heidelberg (2007)
51. Wojciechowski, M., Zakrzewicz, M.: Methods for Batch Processing of Data Mining Queries. In: Haav, H.-M., Kalja, A. (eds.) Proceedings of the 5th International Baltic Conference on Databases and Information Systems, Tallinn Technical University, pp. 225–236 (2002)
52. Wojciechowski, M., Zakrzewicz, M.: Dataset Filtering Techniques in Constraint-Based Frequent Pattern Mining. In: Hand, D.J., Adams, N.M., Bolton, R.J. (eds.) Pattern Detection and Discovery. LNCS (LNAI), vol. 2447, pp. 77–91. Springer, Heidelberg (2002)
53. Wojciechowski, M., Zakrzewicz, M.: Evaluation of Common Counting Method for Concurrent Data Mining Queries. In: Kalinichenko, L.A., Manthey, R., Thalheim, B., Wloka, U. (eds.) ADBIS 2003. LNCS, vol. 2798, pp. 76–87. Springer, Heidelberg (2003)
54. Wojciechowski, M., Zakrzewicz, M.: Data Mining Query Scheduling for Apriori Common Counting. In: Barzdins, J. (ed.) Proceedings of the 6th International Baltic Conference on Databases and Information Systems, University of Latvia, pp. 270–281 (2004)
55. Wojciechowski, M., Zakrzewicz, M.: Evaluation of the Mine Merge Method for Data Mining Query Processing. In: Benczur, A., Demetrovics, J., Gottlob, G. (eds.) Proceedings of the 8th East European Conference on Advances in Databases and Information Systems, Computer and Automation Research Institute, Hungarian Academy of Sciences, pp. 78–88 (2004)

56. Wojciechowski, M., Zakrzewicz, M.: On Multiple Query Optimization in Data Mining. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 696–701. Springer, Heidelberg (2005)
57. Wojciechowski, M., Zakrzewicz, M.: Heuristic Scheduling of Concurrent Data Mining Queries. In: Li, X., Wang, S., Dong, Z.Y. (eds.) ADMA 2005. LNCS (LNAI), vol. 3584, pp. 315–322. Springer, Heidelberg (2005)
58. Wojciechowski, M., Zakrzewicz, M.: Partycjonowanie grafow a optymalizacja wykonania zbioru zapytan eksploracyjnych. In: Morzy, T., Rybinski, H. (eds.) Proceedings of I Krajowa Konferencja Naukowa Technologie Przetwarzania Danych, pp. 62–71. Wydawnictwo Politechniki Poznanskiej (2005)
59. Zakrzewicz, M., Morzy, M., Wojciechowski, M.: A Study on Answering a Data Mining Query Using a Materialized View. In: Aykanat, C., Dayar, T., Körpeoglu, İ. (eds.) ISCIS 2004. LNCS, vol. 3280, pp. 493–502. Springer, Heidelberg (2004)

Chapter 10

Text Clustering with Named Entities: A Model, Experimentation and Realization

Tru H. Cao, Thao M. Tang, and Cuong K. Chau

Ho Chi Minh City University of Technology and
John von Neumann Institute VNU-HCM Vietnam
tru@cse.hcmut.edu.vn

Abstract. Named entities often occur in web pages, in particular news articles, and are important to what the web pages are about. They have ontological features, namely, their aliases, types, and identifiers, which are hidden from their textual appearance. In this chapter, for text searching and clustering, we propose an extended Vector Space Model with multiple vectors defined over spaces of entity names, types, name-type pairs, identifiers, and keywords. Both hard and fuzzy text clustering experiments of the proposed model on selected data subsets of Reuters-21578 are conducted and evaluated. The results prove that a weighted combination of named entities and keywords are significant to clustering quality. Implementation and demonstration of text clustering with named entities in a semantic search engine are also presented.

1 Introduction

Clustering, which is to partition and group data points of similar properties together, is not only an important technique for data mining and knowledge discovery, but also a useful technique for information processing in other application areas [21, 24]. Traditional text clustering is only based on keywords (KW) occurring in texts. Words include those that represent named entities (NE), which are referred to by names such as people, organizations, and locations [23]. In particular, news articles usually contain such named entities, which are important for the news contents. Indeed, in the top 10 search terms by YahooSearch¹ and GoogleSearch² in 2008, there are respectively 10 and 9 ones that are named entities. Besides, textual corpora, such as web pages and blogs, often contain named entities.

However, named entities in a document cover under their textual forms (i.e., names) ontological features that are significant to the semantics of the text. Firstly, it is the type of a named entity in the ontology of discourse, for which documents containing “*Ha Noi*”, “*Paris*”, and “*Tokyo*” may be grouped together as those about capital cities in the world. Clustering purely based on keywords fails to do that because it does not use the common latent type information of such named entities.

¹ <http://buzz.yahoo.com/yearinreview2008/top10/>

² <http://www.google.com/intl/en/press/zeitgeist2008/>

Secondly, it is the identifier of a named entity, for which documents about “U.S.”, “USA”, “United States”, and “America” may be grouped together as those about the same country *United States of America*. Keyword-based clustering also fails because it does not use the fact that an entity may exist under different aliases. These are among the ontological features of named entities.

The ontology-based text clustering methods in [14] and [28] actually relied on an ontology of common concepts like WordNet rather than on named entities. In [25], the most significant entity name in a document was used as its label, based on an enhanced version of the *tf.idf* measure. Then the documents with labeling named entities of the same type were grouped together. As such, it was simply classification of texts by the types of their representative entity names, rather than clustering. Consequently, it could not produce a partition each cluster of which was a group of documents having close semantics regarding various named entities occurring in them.

Closely related to our work were [9] and [18]. In [9], a linear combination of one vector on proper names with their types and one vector on common words was used to represent a document. However, only proper names of the person, organization and location types were considered. In [18], each document was represented by three different vectors on named entities of each of the person, organization and location types. While [9] suggested that text clustering on only named entities was not good, [18] reported it was for multilingual news clustering.

Meanwhile, for text searching, in [6] the authors adapted the traditional Vector Space Model (VSM) with vectors over the space of NE identifiers in the knowledge base of discourse and equally linear combination of its NE-identifier-based vector and keyword-based vector. The latent semantics model proposed in [10] used both keywords and named entities as terms for a single vector space, but only entity names were taken into account. In contrast, [3] introduced a multi-vector space model on all of the NE features, then explored and evaluated the information retrieval performance of various combinations of keywords and named entities.

This paper contributes to text clustering using named entities in three aspects:

1. Our document representation model takes into account all types and all combined features of named entities.
2. Both hard clustering and fuzzy clustering are experimented. The results show that, for good clustering quality, the weights of the named entity and keyword components in the model depend on the actual contents of the documents to be clustered.
3. The model is realized and demonstrated in the semantic search engine called VN-KIM Search, for hierarchical clustering of resulting documents by keywords as well as different named entity features.

Section 2 summarizes the basic notions and formulation of our proposed multi-vector space model combining named entities and keywords. Section 3 recalls key measures of hard and fuzzy clustering quality. Sections 4 and 5 respectively present our experiments and evaluation on hard and fuzzy text clustering. Section 6 introduces VN-KIM Search with text clustering using named entities on search results. Finally, Section 7 draws concluding remarks and further work to be investigated.

2 An Entity-Keyword Multi-Vector Space Model

Despite having known disadvantages, VSM is still a popular model and a basis to develop other models for document representation and processing, because it is simple, fast, and its similarity measure is in general either better or almost as good as a large variety of alternatives (cf. [1, 16]). We recall that, in the keyword-based VSM, each document is represented by a vector over the space of keywords of discourse. Conventionally, the weight corresponding to a term dimension of the vector is a function of the occurrence frequency of that term in the document, called *tf*, and the inverse occurrence frequency of the term across all the existing documents, called *idf*. The similarity degree between two documents is then defined as the cosine of their representing vectors.

We represent each named entity by a triple (*name/type/identifier*) where *name*, *type*, and *identifier* are respectively the name, type, and identifier of that named entity. Let N , T , and I be respectively the sets of names, types, and identifiers of named entities in the ontology of discourse. Then:

1. Each document d is modelled as a subset of $(N \cup \{*\}) \times (T \cup \{*\}) \times (I \cup \{*\})$, where $*$ denotes an unspecified name, type, or identifier of a named entity in d , and
2. d is represented by the quadruple $(\vec{d}_N, \vec{d}_T, \vec{d}_{NT}, \vec{d}_I)$, where \vec{d}_N , \vec{d}_T , \vec{d}_{NT} , and \vec{d}_I are respectively vectors over N , T , $N \times T$, and I .

For example, following is a text and its set of named entity features:

“U.N. team survey of public opinion in North Borneo and Sarawak on the question of joining the federation of Malaysia”.

$\{(U.N./*/*), (North\ Borneo/Province/*), (Sarawak/Location/*), (Malaysia/Country/Country_T.MY)\}$

Here, *Country_T.MY* is the identifier of the country *Malaysia* in the knowledge base of discourse. Meanwhile, the type of *U.N.* is presumably unrecognized, and *North Borneo* and *Sarawak* are only recognized as of the types *Province* and *Location*, respectively.

A feature of a named entity could be unspecified due to the incomplete information about that named entity in a document, or the inability of an employed NE recognition engine to fully recognize it. Each of the four component vectors introduced above for a document can be defined as a vector in the traditional *tf.idf* model on the corresponding space of entity names, types, name-type pairs, or identifiers, instead of keywords. However, there are two following important differences with those ontological features of named entities in calculation of their vector weights:

1. The frequency of a name also counts identical entity aliases. That is, if a document contains an entity having an alias identical to that name, then it is assumed as if the name occurred in the document. For example, if a document refers to the country *Georgia*, then each occurrence of that entity in the document is counted as one occurrence of the name *Gruzia*, because it is an alias of *Georgia*. Named entity aliases are specified in a knowledge base of discourse.

2. The frequency of a type also counts occurrences of its subtypes. That is, if a document contains an entity whose type is a subtype of that type, then it is assumed as if the type occurred in the document. For example, if a document refers to *Washington DC*, then each occurrence of that entity in the document is counted as one occurrence of the type *Location*, because *City* is a subtype of *Location*. The type subsumption is defined by the type hierarchy of an ontology of discourse.

We then define the similarity degree of a document d and a document q , with respect to the named entity features, as follows:

$$w_N \cdot \text{cosine}(\vec{d}_N, \vec{q}_N) + w_T \cdot \text{cosine}(\vec{d}_T, \vec{q}_T) + w_{NT} \cdot \text{cosine}(\vec{d}_{NT}, \vec{q}_{NT}) + w_I \cdot \text{cosine}(\vec{d}_I, \vec{q}_I) \quad (\text{Eq. 1})$$

where $w_N + w_T + w_{NT} + w_I = 1$.

We deliberately leave the weights in the sum unspecified, to be flexibly adjusted in applications, depending on developer-defined relative significances of the four ontological features. We note that the join of \vec{d}_N and \vec{d}_T cannot replace \vec{d}_{NT} because the latter is concerned with entities of certain name-type pairs (e.g. the co-occurrence of an entity named *Georgia* and another country mention in a document does not necessarily refer to the country *Georgia*). Meanwhile, \vec{d}_{NT} cannot replace \vec{d}_I because there may be different entities of the same name and type (e.g. there are different cities named *Moscow* in the world). Also, since names and types of an entity are derivable from its identifier, products of I with N or C are not included.

Clearly, named entities alone are not adequate to represent a document. For instance, in the example text above, *opinion*, *joining*, and *federation* are keywords to be taken into account. Therefore, we propose to represent a document by one vector on keywords and four vectors on named entity features. Let \vec{d}_{KW} and \vec{q}_{KW} be respectively the vectors representing the keyword features of two documents d and q , as in the traditional VSM. The similarity degree of d and q is then defined as follows:

$$\begin{aligned} sim(\vec{d}, \vec{q}) = & \alpha \cdot [w_N \cdot \text{cosine}(\vec{d}_N, \vec{q}_N) + w_T \cdot \text{cosine}(\vec{d}_T, \vec{q}_T) + \\ & w_{NT} \cdot \text{cosine}(\vec{d}_{NT}, \vec{q}_{NT}) + w_I \cdot \text{cosine}(\vec{d}_I, \vec{q}_I)] + (1 - \alpha) \cdot \text{cosine}(\vec{d}_{KW}, \vec{q}_{KW}) \end{aligned} \quad (\text{Eq. 2})$$

where $w_N + w_T + w_{NT} + w_I = 1$ and $\alpha \in [0, 1]$. The coefficient α weighs relative importance of the NE and KW components in document representation.

The proposed multi-vector space model can be used for clustering documents into a hierarchy via top-down phases each of which uses one of the four NE-based vectors presented above. For example, given a set of geographical documents, one can first cluster them into groups of documents about rivers and mountains, i.e., clustering with respect to entity types. Then, the documents in the river group can be clustered further into subgroups each of which is about a particular river, i.e., clustering with respect to entity identifiers.

Meanwhile, the KW-based vector is complementary to the NE-based vectors in representing the salient points in the content of a document. For instance, documents about tourist attraction places should contain both keywords related to tourist attraction and named entities being places. As shown in the experiments next, optimal weighting of the NE component and the KW component depends on the contents of the texts to be clustered. However, the point is that relying on keywords alone as in traditional techniques may not be satisfactory in practice.

There are still possible variations of the proposed model that are worth exploring, depending on whether entity names in a document are counted as keywords in constructing its KW-based vector or not. For instance, in the example text above, *U.N*, *North Borneo*, *Sarawak*, and *Malaysia* could also be treated as keywords as usual. In other words, the entity name set and the keyword set of a text may or may not be considered as overlapping. We call these two alternative models NEKW_OVL and NEKW_NOVL, respectively.

3 Measures of Clustering Quality

Traditionally, clustering quality is evaluated using two complementary measures: (1) *internal measure* that reflects the average semantic distance between data points within each cluster; the smaller the better for the cluster purity; and (2) *external measure* that reflects the average semantic distance between the clusters themselves; the larger the better for the cluster separation. In [13], for hard clustering, *cluster entropy* and the *class entropy* are defined as the internal and external measures, respectively, and the Overall Entropy (OE) as their linear combination. The smaller the overall entropy is, the better clustering quality is.

Formally, suppose $C = C_1 \cup C_2 \cup \dots \cup C_k$ is a partition on the set of N data points taking labels in the set $\{l_1, l_2, \dots, l_{k^*}\}$. Let n_j be the total number of data points of label l_j in the dataset, and n_{ij} be the number of data points labeled l_j in cluster C_i . Then, the cluster entropy E_c , the class entropy E_l , and the overall entropy are defined as follows:

$$\begin{aligned} E_c(C) &= -\sum_{i=1}^k \sum_{j=1}^{k^*} \frac{n_{ij}}{N} \log \frac{n_{ij}}{|C_i|} \\ E_l(C) &= -\sum_{j=1}^{k^*} \sum_{i=1}^k \frac{n_{ij}}{N} \log \frac{n_{ij}}{n_j} \\ E(C) &= \beta \cdot E_c(C) + (1 - \beta) \cdot E_l(C) \end{aligned} \quad (\text{Eqs. 3})$$

where $\beta \in [0, 1]$ is empirically determined. The smaller $E(C)$ is, the better clustering quality is. Ideally, all data points in each cluster have the same label, i.e., $E_c = 0$, and all data points of the same label reside in the same cluster, i.e., $E_l = 0$.

Meanwhile, for the Variation of Information (VI) measure [17], assume $C^* = C_1^* \cup C_2^* \cup \dots \cup C_{k^*}^*$ is the pre-constructed correct partition of the dataset of discourse. The information variation between C and C^* is defined by:

$$\begin{aligned}
VI(C, C^*) &= H(C \mid C^*) + H(C^* \mid C) \\
&= H(C) + H(C^*) - 2I(C, C^*) \\
I(C, C^*) &= \sum_{i=1}^k \sum_{j=1}^{k^*} \frac{|C_i \cap C_j^*|}{N} \log \frac{|C_i \cap C_j^*| / N}{(|C_i| / N) \cdot (|C_j^*| / N)} \\
H(C) &= -\sum_{i=1}^k \frac{|C_i|}{N} \log \frac{|C_i|}{N} \\
H(C^*) &= -\sum_{j=1}^{k^*} \frac{|C_j^*|}{N} \log \frac{|C_j^*|}{N}
\end{aligned} \tag{Eqs. 4}$$

Here $H(C \mid C^*)$ is referred to as *clustering conditional entropy* of C given C^* , $I(C, C^*)$ is called *clustering mutual information* between C and C^* , and $H(C)$ and $H(C^*)$ are respectively *clustering entropies* of C and C^* . Significantly, the following theorem states the equivalence of VI and OE, if the data point labels are as given by C^* and the cluster and the class entropies in OE have the same weight. The proof was presented in [8].

Theorem 1. Assume that $C^* = C_1^* \cup C_2^* \cup \dots \cup C_{k^*}^*$ is a partition on a set of data points and the label l_i of each cluster C_i^* is also the label of all the data points in it. Let $C = C_1 \cup C_2 \cup \dots \cup C_k$ be an arbitrary partition on that same data point set. Then $VI(C, C^*) = 2E(C)$ if the cluster entropy and the class entropy in the computation of $E(C)$ have the same weight 0.5.

Since taking the equal weights for the cluster and the class entropies in the OE measure is natural and reasonable, the significance of the property proved above is that one can use either OE or VI for measuring clustering quality when all data points have pre-defined labels. Nevertheless, in practice, data point labels may not be pre-defined but generated as part of a clustering technique, which also affect the clustering quality in terms of OE. That is the case when VI is useful for testing a partition generated by that technique with respect to a subjectively constructed partition on the same dataset.

For fuzzy clustering, Xie-Beni index [24, 26] is among the most popularly used ones, measuring the overall average purity and separation of a fuzzy partition by:

$$S = \frac{\sum_{i=1}^c \sum_{k=1}^n [\mu_i(x_k)]^m \|x_k - v_i\|^2}{n * \min_{i,j} \|v_i - v_j\|^2} \tag{Eq. 5}$$

where n is the number of data points x_k 's, m is the *fuzziness index*, v_i is the centroid of the i -th cluster, $\mu_i(x_k)$ is the membership value of x_k into the i -th cluster, and $\|x_k - v_i\|$ represents the distance between the data point x_k and the i -th cluster, which is usually calculated by Euclidian Distance. The smaller the value of the index, the better the fuzzy partition is. This index could be considered as a fuzzy counterpart of the OE measure.

Measures like OE for hard clustering and XB for fuzzy clustering are based on the purity and separation of the resulting partition itself. We view them as *objective measures*, for which a clustering result is not tested against a pre-constructed gold-standard one. In contrast, the VI measure quantifies how different two partitions are. We view it as a *subjective measure*, which allows one to evaluate the clustering quality of a technique by comparing a partition generated by that technique with a corresponding partition manually constructed by humans. We apply both of these objective and subjective measures in this work.

4 Hard Clustering Experiments

In the scope of this paper, for experiments we focus on the type feature of named entities, because many named entities in various documents may have the same type. That hidden ontological feature is ignored in the traditional keyword-based information processing, which affects clustering quality. That is, our experiments are performed on vectors of the form $\alpha \cdot \text{cosine}(\vec{d}_T, \vec{q}_T) + (1 - \alpha) \cdot \text{cosine}(\vec{d}_{KW}, \vec{q}_{KW})$. The value of α is varied in the experiments to find how significant the NE and KW components are to clustering quality; $\alpha = 0$ means purely keyword-based clustering, while $\alpha = 1$ means purely named entity-based clustering.

For testing clustering quality with respect to the VI measure, we use the Reuters-21578 dataset, which contains 21,578 documents. In this dataset, the header of each document, besides its body text, has the topic tag TOPICS containing the main keywords representing the topic of the document, and the named entity tags PEOPLE, ORGS, PLACES, and EXCHANGES respectively containing the main people, organizations, places, and stock exchange agencies that the document is presumably about. Figure 1 is an example of the header of a document in this dataset. It specifies that the document is about the topics *grain* and *wheat*, the places *USA* and *Australia*, and the people *Lyng* and *Yeutter*.

```

<REUTERS TOPICS="YES" LEWISPLIT="TRAIN" CGISPLIT="TRAINING-
SET" OLDID="12925" NEWID="742">
<DATE> 2-MAR-1987 15:46:40.19</DATE>
<TOPICS><D>grain</D><D>wheat</D></TOPICS>
<PLACES><D>usa</D><D>australia</D></PLACES>
<PEOPLE><D>lyng</D><D>yeutter</D></PEOPLE>
<ORGs></ORGs>
<EXCHANGES></EXCHANGES>
<TEXT>
<TITLE>U.S. WHEAT GROUPS CALL FOR GLOBAL ACTION</TITLE>
<DATELINE>WASHINGTON, March 2 - </DATELINE>
<BODY>....</BODY>
</TEXT>
</REUTERS>
```

Fig. 1. An example header of a document in Reuters-21578

From this dataset, we select a sub-set of 500 typical documents for hard clustering experiments, such that the content of each of them is clearly about named entities of a particular type. Such a size of a testing dataset is common in clustering experiments (cf. [20]). At first, approximately 7,000 documents each of which has only one named entity tag are automatically filtered. Next, we manually select 500 documents each of which is clearly about an entity type. Some tagging errors in the original dataset are also fixed during this document selection process.

Further, the selected documents are automatically annotated using the NE recognition engine of KIM [15], KIM PROTON ontology, and KIM World KB. The ontology consists of about 300 types and 100 relations, and the knowledge base contains over 77,000 named entities. The average precision and recall of the NE recognition engine are about 90% and 86%, respectively³.

Then we obtain a testing dataset, denoted by D_h , for hard clustering with 4 clusters based on the named entity tags. The distribution of the 500 documents across the four NE tags is as follows:

PLACES: 195 documents
 PEOPLE: 105 documents
 ORGS: 129 documents
 EXCHANGES: 71 documents

Here we employ the most popular algorithm k -means [12] for hard clustering. Basically, the k -means algorithm keeps relocating data points into k clusters until the following objective function stops decreasing:

$$f = \sum_{i=1}^k \sum_{x_j \in c_i} |x_j - \bar{c}_i| \quad (\text{Eq. 6})$$

where c_i is the i -th cluster and \bar{c}_i is the average value of its data points x_j 's, called the centroid. In practice, for obtaining the best clustering quality, the optimal value of k is determined by experiments.

First, we run k -means on the constructed 500-document dataset with $k = 4$ and α varying from 0 to 1 on 0.1 incremental steps. Figure 2 illustrates the clustering quality of the NEKW_OVL and NEKW_NOVL models with respect to the OE and VI measures. For the OE measure, we take the equal weight for the cluster entropy and the class entropy, i.e., $\beta = 0.5$ for Equations 3. The corresponding data are presented in Table 1. In accordance to Theorem 1, the corresponding OE and VI curves actually have the same shape. Second, we vary k from 2 to 10, take the best case for each value of k , and plot their OE and VI values as in Figure 3, from the obtained data in Table 2. As expected, $k = 4$ is the optimal value for the testing dataset with 4 pre-defined clusters.

³ It is reported at <http://www.ontotext.com/kim/performance.html>.

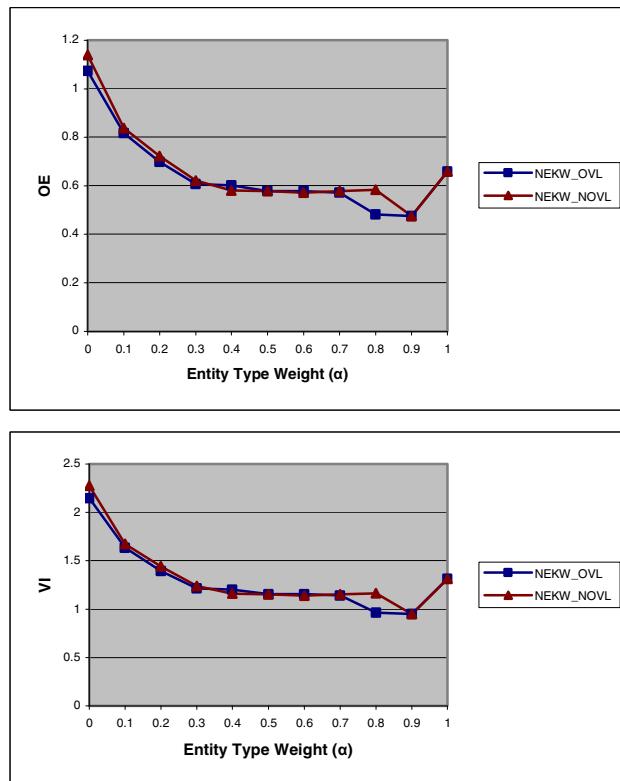
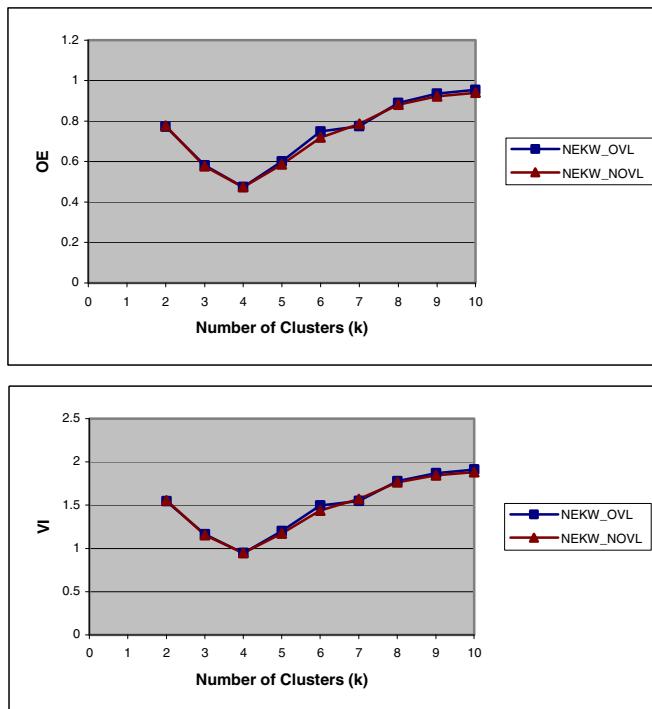


Fig. 2. OE and VI diagrams for hard clustering with $k = 4$ and varied α

Table 1. OE and VI measures for hard clustering with $k = 4$ and varied α

OE	$\alpha=0$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
NEKW_OVL	1.07	0.82	0.7	0.61	0.6	0.58	0.58	0.57	0.48	0.47	0.66
NEKW_NOVL	1.14	0.84	0.72	0.62	0.58	0.58	0.57	0.58	0.58	0.47	0.66
VI	$\alpha=0$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
NEKW_OVL	2.15	1.63	1.39	1.21	1.2	1.15	1.15	1.14	0.96	0.95	1.31
NEKW_NOVL	2.28	1.67	1.44	1.24	1.16	1.15	1.14	1.15	1.16	0.95	1.31

**Fig. 3.** OE and VI diagrams for hard clustering with varied k **Table 2.** OE and VI measures for hard clustering with varied k

OE	$k=2$	3	4	5	6	7	8	9	10
NEKW_OVL	0.77	0.58	0.47	0.6	0.75	0.78	0.89	0.94	0.96
NEKW_NOVL	0.78	0.58	0.47	0.59	0.72	0.79	0.88	0.92	0.94
VI	$k=2$	3	4	5	6	7	8	9	10
NEKW_OVL	1.55	1.16	0.95	1.2	1.5	1.55	1.78	1.87	1.91
NEKW_NOVL	1.56	1.15	0.95	1.17	1.44	1.57	1.76	1.85	1.88

The experimental results show that:

1. The NEKW_OVL and NEKW_NOVL models perform nearly the same for hard clustering. That is, counting or not counting entity names for KW-based vectors make little difference. It means that entity names themselves, i.e., only their textual forms, are not significant to assignment of named entity tags to documents in the Reuters-21578 dataset.

2. The clustering quality is improved by more than 100% with $\alpha = 0.9$ as compared with $\alpha = 0$ ($OE = 0.47$ vs. 1.07 for NEKW_OVL). We note that the NEKW_OVL model with $\alpha = 0$ is actually the traditional purely keyword-based VSM. So, the latent ontological features (e.g. entity types in these experiments) are important to the clustering results.
3. The best clustering quality is obtained when $k = 4$, which is the same as the number of clusters of the pre-constructed testing dataset. It implies that our proposed models represent well the contents of documents like those of the Reuters-21578 dataset for the clustering task.

5 Fuzzy Clustering Experiments

The fuzzy counterpart of k -means is fuzzy c -means. We recall that, basically the fuzzy c -means algorithm keeps relocating data points into c clusters until the following objective function stops decreasing (cf. Equation 5):

$$J_m(P) = \sum_{k=1}^n \sum_{i=1}^c [\mu_i(x_k)]^m \|x_k - v_i\|^2 \quad (\text{Eq. 7})$$

where $P = \{\mu_1, \mu_2, \dots, \mu_c\}$ is a fuzzy c -partition. The centroid of each cluster is computed by the following formula:

$$v_i = \frac{\sum_{k=1}^n [\mu_i(x_k)]^m x_k}{\sum_{k=1}^n [\mu_i(x_k)]^m} \quad (\text{Eq. 8})$$

At each iteration of the algorithm, after the cluster centroids are re-calculated, membership values $\mu_i(x_k)$'s are updated based on the data point x_k 's and the cluster centroids:

$$\mu_i(x_k) = \frac{1}{\sum_{j=1}^c \left(\frac{\|x_k - v_j\|}{\|x_k - v_i\|} \right)^{\frac{2}{m-1}}} \quad (\text{Eq. 9})$$

The process is stopped when the maximum change of membership values between two consecutive iterations is less than a pre-defined threshold value.

We also use the Reuters-21578 dataset for fuzzy clustering experiments, constructing two testing datasets. For fuzzy clustering, the documents are selected so that some of them are about more than one named entity type or more than one document topic. One testing dataset consists of documents of only NE tags, while the other has documents with both NE and topic tags.

The first dataset, denoted by D_{fl} , comprises 500 documents with one or more of the four tags PLACES, PEOPLE, ORGS, and EXCHANGES, in which:

200 documents contain only one NE tag each
 238 documents contain two NE tags each
 57 documents contain three NE tags each
 5 documents contain four NE tags each.

The distribution of the 500 documents across the four NE tags is as follows:

PLACES: 300 documents
 PEOPLE: 200 documents
 ORGS: 281 documents
 EXCHANGES: 86 documents

Details of the numbers of documents containing certain tags are given in Table 3.

The second dataset, denoted by D_{f2} , comprises 350 documents containing both NE tags, namely PLACES and PEOPLE, and topic tags, namely INTEREST and MONEY-FX, with the following distributions:

PLACES: 336 documents
 PEOPLE: 136 documents
 INTEREST: 148 documents
 MONEY-FX: 174 documents

Details of the numbers of documents containing certain tags are given in Table 4.

Table 3. Document-tag distribution in the dataset D_{f1}

PLACES (300)	PEOPLE (200)	ORGs (281)	EXCHANGES (86)	Number of Documents
X				60
	X			29
		X		41
			X	70
X	X			57
X		X		120
X			X	2
	X	X		51
	X		X	1
		X	X	7
X	X	X		56
X	X		X	0
X		X	X	0
	X	X	X	1
X	X	X	X	5
Total				500

We run fuzzy c -means on the two constructed datasets D_{f1} and D_{f2} , using both the NEKW_OVL and NEKW_NOVL models, and evaluating clustering quality with respect to the XB measure. The fuzzy index m is set to 2 and the threshold value to stop the iterative process is set to 0.01. In the experiments, we vary c from 2 to 10 and, for each value of c , vary α from 0 to 1 on 0.1 incremental steps. Since fuzzy c -means relies on the initial membership degrees of the documents to the projected clusters, which are initialized randomly, for each c and α we run the algorithm 10 times and take the average of the results for the XB measure.

Table 4. Document-tag distribution in the dataset D_{f2}

PLACES (300)	PEOPLE (200)	INTEREST (281)	MONEY- FX (86)	Number of Documents
X				50
	X			0
		X		6
			X	2
X	X			50
X		X		50
X			X	50
	X	X		0
	X		X	0
		X	X	6
X	X	X		20
X	X		X	50
X		X	X	50
	X	X	X	0
X	X	X	X	16
Total				350

Table 5 and Table 6 present the XB values with varied c and α on the dataset D_{f1} for the models NEKW_OVL and NEKW_NOVL, respectively. For each value of c , there is an optimal value of α such that the XB measure is minimal, i.e., giving the best clustering quality. In order to evaluate the effect of α in average on clustering quality, we compute the average of the XB values for each common optimal value of α given certain values of c , as shown in the last rows of the two tables. It shows that the best values of α in average for NEKW_OVL and NEKW_NOVL on D_{f1} are respectively 0.9 and 0.7.

Table 5. The XB measure with varied c and α on the dataset D_{fl} for the model NEKW_OVL

XB x1,000	$\alpha = 0$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
c = 2	61040	8309	7747	566	42.3	76.6	44	181	61.8	13	11.1
3	398.8	35838	105	1464	768	237	168	58	93.5	68	14.3
4	265.5	181	6514	3037	726	334	234	99	36	28	25.8
5	296.2	166.9	76.6	52.9	803	457	127	92	75.8	32	18.2
6	314.5	169.6	93.4	38.3	243	311	11	90	128	37	20.5
7	240.2	168.5	82.9	35.1	23.1	15.2	116	41	367	22	23.2
8	228.7	146.7	88.4	43.9	17.5	18.1	10	42	28.1	2.8	22.8
9	206	146.8	82.7	47.5	20.6	12.9	6	4.5	23.4	11	18.1
10	214.7	139.1	58.5	38.9	23.9	17.5	6.8	5	3.44	2.2	12.2
Best Average						15.2	11	4.5		2.5	17.3

Table 6. The XB measure with varied c and α on the dataset D_{fl} for the model NEKW_NOVL

XB x1,000	$\alpha = 0$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
c = 2	29538	400.1	624	1605	917	518	117	212	108	14	22
3	382.3	232	3567	3224	1438	838	390	248	32.2	31	11
4	342.7	223.8	104	1495	1244	1089	448	157	58.3	97	19
5	271.8	205	105	55.7	523	346	222	147	81.8	86	25
6	258.4	188.2	80.5	856	1275	286	158	113	73	42	17
7	215.7	171	69.5	41.8	17.7	11.1	54	33	101	36	50
8	228.8	179.1	86.4	41.5	18.5	13	8.6	67	38.4	59	27
9	219.7	170.3	76.7	38.5	22.3	12.7	70	37	4.2	68	16
10	184.3	156.2	69.1	46.7	21.3	13.2	7.3	4	11.1	9.6	5.1
Best Average						11.1	8.6	4	4.2	14	18

The fact that the best value of α for NEKW_OVL is higher than that for NEKW_NOVL can be explained as follows. In the NEKW_OVL model, entity names are counted as keywords and may cause noises for fuzzy clustering with respect to NE tags. So, the weight for the KW component, i.e., $1 - \alpha$, should be decreased to reduce that noise effect. However, hard clustering as experimented above might not be effected by such noises.

One may also have another observation on the experimental results. That is, for each value of α , let us take the average XB measure on different values of c . Figure 4 plots that average XB measure with varied α on the dataset D_{fl} . It shows that, when α is big enough, e.g. from about 0.3 in this test, the performances of NEKW_OVL and NEKW_NOVL are almost the same. Probably, for that threshold α , counting entity names in the KW component of a document makes nearly no difference.

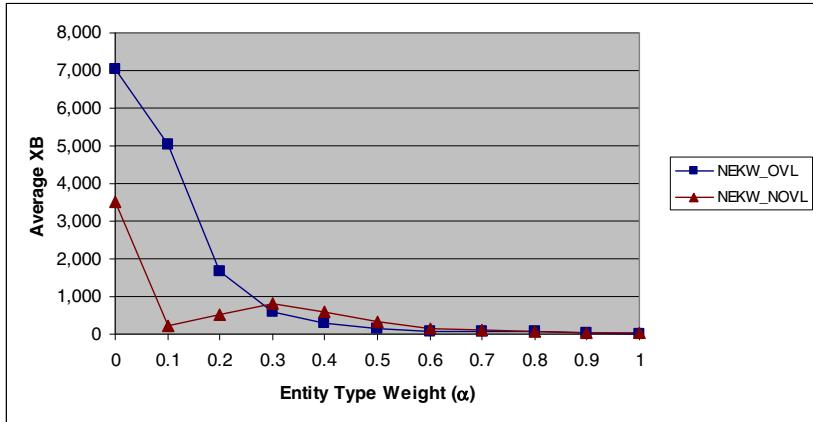


Fig. 4. Average XB with varied α on the dataset D_{f1}

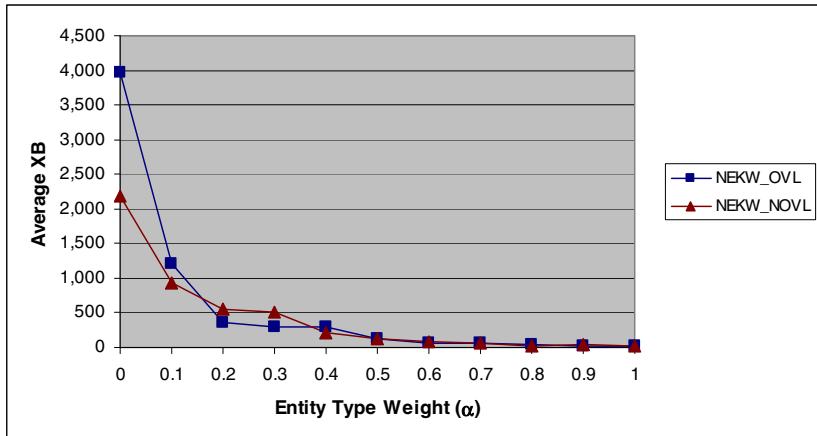
Meanwhile, on the dataset D_{f2} , Table 7 and Table 8 show that the best values of α in average for NEKW_OVL and NEKW_NOVL are respectively 0.6 and 0.5. The lower best values of α as compared to those on D_{f1} are due to the documents containing not only NE tags but also topic tags, which rely on keywords. Figure 5. shows that, as for D_{f1} , NEKW_OVL and NEKW_NOVL perform nearly the same in terms of the average XB measure from a certain threshold α . Also, on both D_{f1} and D_{f2} , as for hard clustering, the fuzzy clustering quality is drastically improved when taking into account the latent named entity types, i.e., with $\alpha > 0$.

Table 7. The XB measure with varied c and α on the dataset D_{f2} for the model NEKW_OVL

XB x1,000	$\alpha = 0$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
c = 2	391.5	4258	996	227	797	170	46	48	10.1	24	24
3	22888	1624	925	650	698	235	82	44	169	17	45
4	6777	4820	825	711	343	235	74	178	26.5	15	21
5	1805	48.6	25.4	172	173	236	55	46	20.2	23	38
6	44.2	37.5	24.5	845	303	230	76	88	20.6	11	19
7	50.1	25.9	28.7	12.6	284	98	112	49	29.1	17	9.5
8	3613	30.8	397	20.8	153	9.3	64	52	31.3	17	13
9	51.2	38.2	30.5	18.4	7.5	6.4	5.4	50	33	37	8.7
10	41.5	39.9	24.4	10.4	8.5	6.5	28	77	9.3	33	11
Best Average						7.9	5.4		15.1	14.3	9.5

Table 8. The XB measure with varied c and α on the dataset D_{f2} for the model NEKW_NOVL

XB $\times 1,000$	$\alpha = 0$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
c = 2	90.3	5490	669	279	202	167	225	83.3	41	16	21
3	638.9	1187	1757	2143	566	321	61	62.4	20	18	68
4	3125	1486	826	549	465	126	115	35.4	37	21	19
5	59.8	67.9	654	1325	336	257	99	163	17.4	54	29
6	15563	37	1023	251	83	195	74	73.5	29	150	16.9
7	54.8	70.1	37	22	218	70	129	72	19	30	14
8	60.6	48	32.7	18.3	15	9.1	36	36.8	45	18	14.1
9	48.6	47.4	27.6	20	8.9	6.4	21	31.4	35	22	9
10	54.3	47.4	35.1	10.8	9.4	7.4	4	53	2.7	21	9.6
Best Average						7.7			10	17	16.6

**Fig. 5.** Average XB with varied α on the dataset D_{f2}

6 Text Clustering in VN-KIM Search

Following KIM [15], we have developed a platform for Vietnamese Semantic Web called VN-KIM. It is firstly a knowledge-based system of popular named entities in Vietnam and the world. Currently VN-KIM ontology consists of 370 types and 115 relations. The knowledge base contains more than 210,000 selected named entities. It can automatically extract the type of a named entity in a web page written in Vietnamese and annotate that information in the web page, using the NE recognition engine for Vietnamese developed in [19].

For managing annotated web pages based on the combined entity-keyword VSM presented in Section 2, we have employed and modified Lucene [11], a general open source for storing, indexing and searching documents. In Lucene, a term is a character string and term occurrence frequency is computed by exact string matching. Here are our modifications for what we call S-Lucene:

1. Indexing documents over the four NE feature spaces corresponding to N , T , $N \times T$, and I , besides the ordinary keyword space, to support the new model.
2. Modifying Lucene codes to compute dimensional weights for the vectors representing a document or a query, in accordance to the new model.
3. Modifying Lucene codes to compute the similarity degree between a document and a query, in accordance to the new model.

On the VN-KIM platform, we have implemented a semantic search engine called VN-KIM Search for text searching and clustering using named entities. The engine works on annotated Vietnamese web pages with the following essential features:

1. Its query syntax is designed to be similar to, and as expressive as, the Google's one.
2. However, being more powerful than a purely keyword-based search engine, its terms include both keywords and phrases representing named entities.
3. Moreover, it accepts named entity phrases that are not only simple entity names, but also complex constraints identifying named entities of user interest.
4. Besides, resulting web pages can be clustered with respect to the keywords and named entities that they contain.

VN-KIM Search has been then adapted for English and demonstrated using KIM ontology and NE recognition engine. As realized in real-world application systems like Clusty [7] and Carrot2 [5], clustering is used in VN-KIM Search to overcome the deficiencies of the query-list approach to showing search results by grouping returned documents into a hierarchy of meaningful thematic categories, providing better data views to users than sequential listings (cf. [22, 27]). However, it is ontology-based clustering as presented above instead of simply keyword-based clustering.

Figure 6 shows a screen interface of VN-KIM Search with the query “*peace (country of Asia)*” for searching documents tentatively about peace with countries in Asia. A phrase put in the parentheses is not a normal sequence of keywords, but represents named entities, which in this example are countries in Asia like *Israel* or *China*. Actually, such a query is first mapped to a conceptual graph to look up satisfying named entities in the knowledge base of discourse, using the processing method in [4]. Then the search engine retrieves documents containing those named entities.

The right window displays some top answer documents with queried named entities and keywords highlighted, e.g. *Israel* and *peace* for this example query. The left window displays hierarchical clusters of the answer documents. In this demonstration, the documents are clustered by two levels. The outer level is clustering by entity types and the inner level by entity names, combined with keywords. For instance, as indicated by the cluster labels, it shows that the dominant entities in the documents of the third outer cluster are of the type *Location*. Meanwhile, the four sub-clusters inside this cluster are more about *Israel*, *Cyprus*, *Pakistan*, or *Vietnam*, for instances. Figure 7 is a search result with highlighted named entities that are related to the query topic.

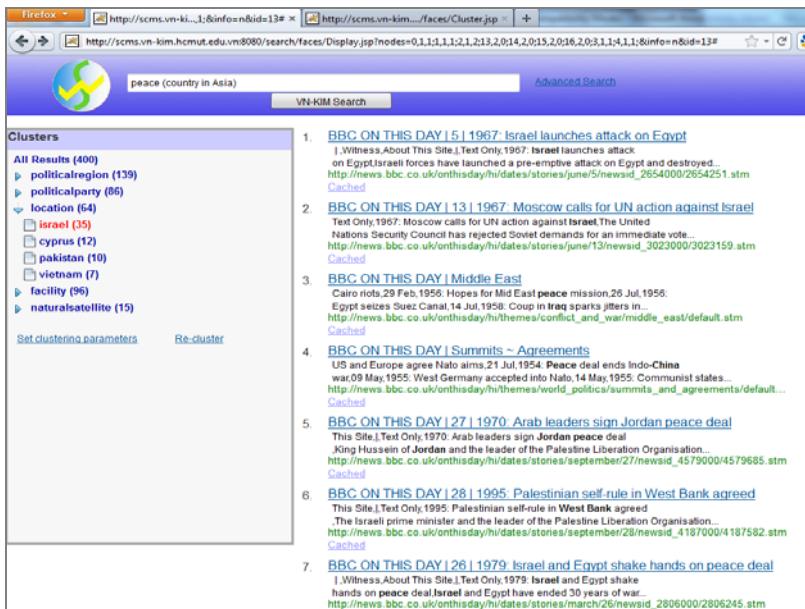


Fig. 6. Ontology-based searching and clustering in VN-KIM Search

The screenshot shows a BBC News web page for 'ON THIS DAY 1967'. The main headline is '5 June 1967 BBC NEWS'. Below it, there's a video player with the text 'Watch/Listen' and 'Report from the Gaza Strip'. To the left, there's a section for '1967: Israel launches attack on Egypt' with a summary and a quote from Levi Eshkol. To the right, there's a sidebar with 'About This Site' (Text Only), 'Stories From 5 Jun', 'Witness' (with a quote from a memoir), 'In Context' (about the Middle East conflict), and 'BBC News' links for 'In Depth' and 'Middle East Crisis'. The bottom of the page has a search bar and various navigation buttons.

Fig. 7. A resulting web page with highlighted named entities in VN-KIM Search

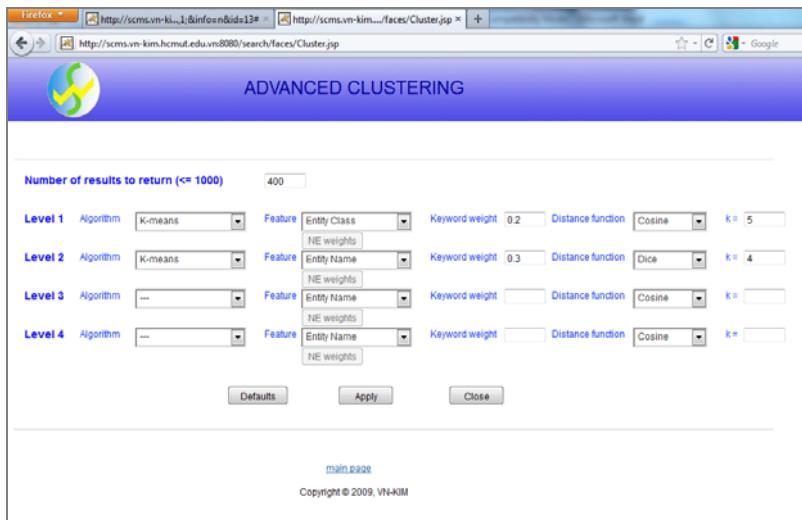


Fig. 8. Setting clustering parameters in VN-KIM Search

Figure 8 shows the interface to set the clustering parameters in VN-KIM Search. Answer documents could be clustered up to four levels. For each level, the user can choose a clustering algorithm (k -means or c -means), named entity features and their weights, a weight for the keyword component as expressed in Equation 2, a distance function (Cosine, Dice, Manhattan, or Euclidean), and a number of clusters. The current setting is for the clustering results in Figure 6.

7 Conclusion

We have presented a multi-vector space model for document representation, searching, and clustering. It is an extension of the VSM that represents a document as a linear combination of a vector on keywords and vectors on features of named entities occurring in the document. Our experimental results using the proposed model for text clustering on the well-known Reuters-21578 dataset are two-fold. First, they show that the latent ontological features of named entities in a document are important to define its contents. In particular, taking into account named entity types, which are covered under their textual forms, drastically improves clustering quality as compared to the purely keyword-based VSM, for both hard and fuzzy clustering on the testing datasets. Second, they show that our model is suitable for representing the subjects of documents involving named entities like Reuters-21578 ones.

One can also observe from the experimental results that optimal weighting of the NE and KW components for clustering depends on document contents. For a dataset whose documents have only NE tags, e.g. D_{f1} in the experiments, the best value of α in average is close to 1, meaning that the NE component plays a major role. For a dataset whose documents have both NE and KW tags, e.g. D_{f2} , the best value of α in average is smaller. Besides, the overlapping and non-overlapping variations of the

proposed model have little difference in performance when the NE component weight is big enough.

The model also supports hierarchical clustering for which each layer uses a certain clustering objective corresponding to a NE feature. We have demonstrated that in the semantic search engine VN-KIM Search. For future work, since named entities are pervasive and play an important role in news articles, we are investigating the proposed model and method for knowledge discovery and integration on the Web.

References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, Reading (1999)
2. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
3. Cao, T.H., Le, K.C., Ngo, V.M.: Exploring Combinations of Ontological Features and Keywords for Text Retrieval. In: Ho, T.-B., Zhou, Z.-H. (eds.) PRICAI 2008. LNCS (LNAI), vol. 5351, pp. 603–613. Springer, Heidelberg (2008)
4. Cao, T.H., Mai, A.H.: Ontology-Based Understanding of Natural Language Queries Using Nested Conceptual Graphs. In: Croitoru, M., Ferré, S., Lukose, D. (eds.) ICCS 2010. LNCS, vol. 6208, pp. 70–83. Springer, Heidelberg (2010)
5. Carrot2: Open Source Search Results Clustering Engine,
<http://project.carrot2.org/architecture.html>
6. Castells, P., Fernández, M., Vallet, D.: An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval. IEEE Transactions on Knowledge and Data Engineering 19, 261–272 (2006)
7. Clusty Search: Clustering Search Engine, <http://clusty.com>
8. Duong, V.T.T., Cao, T.H., Chau, C.K., Quan, T.T.: Latent Ontological Feature Discovery for Text Clustering. In: Proceedings of the 7th IEEE International Conference on Research, Innovation and Vision for the Future - in Computing and Communication Technologies, pp. 264–271 (2009)
9. Friberger, N., Maurel, D., Giacometti, A.: Textual Similarity Based on Proper Names. In: Proceedings of the Workshop on Mathematical/Formal Methods in Information Retrieval at the 25th ACM SIGIR Conference, pp. 155–167 (2002)
10. Gonçalves, A., Zhu, J., Song, D., Uren, V., Pacheco, R.: LRD: Latent Relation Discovery for Vector Space Expansion and Information Retrieval. In: Proceedings of the 7th International Conference on Web-Age Information Management (2006)
11. Gospodnetic, O.: Parsing, Indexing, and Searching XML with Digester and Lucene. Journal of IBM DeveloperWorks (2003)
12. Hartigan, J., Wong, M.: Algorithm AS136: A K-Means Clustering Algorithm. Applied Statistics 28, 100–108 (1979)
13. He, J., Tan, A.H., Tan, C.L., Sung, S.Y.: On Quantitative Evaluation of Clustering Algorithms. In: Wu, et al. (eds.) Clustering and Information Retrieval, pp. 105–133. Kluwer Academic, Dordrecht (2003)
14. Hotho, A., Maedche, A., Maedche, E., Staab, S.: Ontology-based Text Document Clustering. KI 16, 48–54 (2002)
15. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic Annotation, Indexing, and Retrieval. Journal of Web Semantics 2 (2005)

16. Manning, C.D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press, Cambridge (2008)
17. Meilă, M.: Compare Clusterings – an Information Based Distance. *Journal of Multivariate Analysis*, 873–895 (2007)
18. Montalvo, S., Martínez, R., Casillas, A., Fresno, V.: Bilingual News Clustering Using Named Entities and Fuzzy Similarity. In: Matoušek, V., Mautner, P. (eds.) *TSD 2007. LNCS (LNAI)*, vol. 4629, pp. 107–114. Springer, Heidelberg (2007)
19. Nguyen, V.T.T., Cao, T.H.: VN-KIM IE: Automatic Extraction of Vietnamese Named-Entities on the Web. *Journal of New Generation Computing* 25, 277–292 (2007)
20. Niu, Z.-Y., Ji, D.-H., Tan, C.-L.: Using Cluster Validation Criterion to Identify Optimal Feature Subset and Cluster Number for Document Clustering. *Information Processing and Management* 43, 730–739 (2007)
21. Oliveira, J.V., Pedrycz, W. (eds.): *Advances in Fuzzy Clustering and its Applications*. John Wiley & Sons, Chichester (2007)
22. Osinski, S.: Improving Quality of Search Results Clustering with Approximate Matrix Factorisations. In: Lalmas, M., MacFarlane, A., Rüger, S.M., Tombros, A., Tsikrika, T., Yavlinsky, A. (eds.) *ECIR 2006. LNCS*, vol. 3936, pp. 167–178. Springer, Heidelberg (2006)
23. Sekine, S.: Named Entity: History and Future. *Proteus Project Report* (2004)
24. Theodoridis, S., Koutroumbas, K.: *Patern Recognition*. Academic Press, London (2008)
25. Toda, H., Kataoka, R.: A Search Result Clustering Method Using Informatively Named Entities. In: *Proceedings of the 7th ACM International Workshop on Web Information and Data Management*, pp. 81–86 (2005)
26. Xie, X.L., Beni, G.: A Validity Measure for Fuzzy Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 841–847 (1991)
27. Zhang, D., Dong, Y.: Semantic, Hierarchical, Online Clustering of Web Search Results. In: Yu, J.X., Lin, X., Lu, H., Zhang, Y. (eds.) *APWeb 2004. LNCS*, vol. 3007, pp. 69–78. Springer, Heidelberg (2004)
28. Zhang, X., Jing, L., Hu, X., Ng, M., Zhou, X.: A Comparative Study of Ontology Based Term Similarity Measures on PubMed Document Clustering. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) *DASFAA 2007. LNCS*, vol. 4443, pp. 115–126. Springer, Heidelberg (2007)

Chapter 11

Regional Association Rule Mining and Scoping from Spatial Data

Wei Ding¹ and Christoph F. Eick²

¹ Department of Computer Science
University of Massachusetts-Boston
Boston, MA 02125-3393

ding@cs.umb.edu

² Department of Computer Science
University of Houston
Houston, TX 77004

ceick@uh.edu

Abstract. Spatial datasets intrinsically exhibit geographical regional patterns while traditional global statistics seldom provide useful local insights. In this work, we are interested in regional association rule mining and scoping. We investigate the duality between regional association rules and regions where the associations are valid: interesting regions are identified to seek novel regional patterns, and a regional pattern has a scope of a set of regions in which the pattern is valid. We design and implement a reward-based region discovery framework that employs a divisive grid-based supervised clustering for region discovery. We evaluate our approach in a real-world case study to identify spatial risk patterns from arsenic in the Texas water supply. Our experimental results confirm and validate research results in the study of arsenic contamination.

Keywords: Regional Association Rule Mining and Scoping, Region Discovery, Clustering, Spatial Data Mining.

1 Introduction

Enormous amount of spatial data is available due to the rapid advances in database and data acquisition technologies in the last decade. Spatial data mining aims at automatically find novel and useful patterns from large-scale spatial datasets [41,16,30,36,38,22,8,13]. Of particular interests to scientists is to find scientifically meaningful locations and their associated patterns, for example, identification of earthquake hot spots, revealing high-risk zones that particular diseases associated with environmental pollutions, and the detection of emerging crime zones.

The motivation for regional association rule mining and scoping is driven by the facts that global statistics seldom provide useful insight and that most relationships in spatial datasets are geographically regional, rather than global. It

has been pointed out in the literature [20,34,40] that “*whole map statistics are seldom useful,*” that “*most relationships in spatial data sets are geographically regional, rather than global*” and that “*there is no average place on the Earth’s surface*”—a county is not a representative of a state, and a state is not a representative of a country. Therefore, it is not surprising that domain experts are most interested in discovering hidden patterns at a regional scale rather than a global scale [20,32,33].

Unfortunately, traditional association rule mining frequently fails to discover regional patterns due to insufficient global confidence and/or support. A common approach to alleviate the problem is to use a small support threshold. However, this approach usually suffers from a combinatorial explosion in the number of rules generated. Furthermore, for a given dataset, the number of regions as well as the regions themselves are not known *a priori*. This raises two questions: how to measure the interestingness of a set of regions and how to search for interesting regions. One popular approach is to select regions to be mined based on a previously given structure, such as a uniform grid structure using longitude and latitude, or based on political/demographical boundaries, such as counties within a state. But the boundaries of the so-constructed regions fail to consider the natural boundaries of the interesting patterns. Mining local patterns from those regions inevitably leads to spurious patterns.

Another unique phenomenon is that regional association rules, by definition, only hold in a subspace but not in the global space; therefore, regional association rules may only be discovered in a particular subspace of the global space. In this work, we systematically study this problem and address the special challenges for regional association mining and scoping: (1) region discovery: how to identify interesting regions from which novel and useful regional association rules can be extracted; (2) regional association rule scoping: how to determine the scope of regional association rules. Our preliminary work on regional association rule mining was published in [11] and on regional association rule scoping was published in [12]. In this paper, we integrate two originally separated procedures and investigate the duality between regional association rules and regions in which the associations are valid. Interesting regions are identified to seek novel regional patterns, and a regional pattern has a scope that is the set of regions in which the pattern is valid. We design and implement a reward-based framework, utilizing plug-in fitness functions to accomplish two complementary objectives: seeking regions to discover regional association rules, and then identifying regions in which regional association rules are valid. Such regions provide a quantitative measure of how significant a regional association rule is in the global space.

Figure 1 illustrates the procedure of our approach with a real example from our case study. Interesting regions are identified using a grid-based supervised clustering algorithm and a fitness function designed for the identification of arsenic hot spots. An interesting association rule a , *Wells with nitrate concentration lower than 0.085mg/l have dangerous arsenic concentration*, is discovered from an arsenic hot spot area in the South Texas with 100% confidence. The scope of the association rule a is further identified using another fitness function

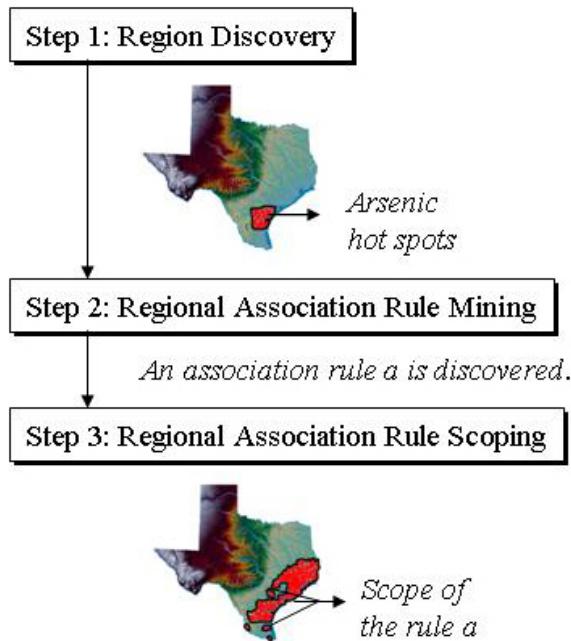


Fig. 1. An example for regional association rule mining and scoping

designed for regional association rule scoping. The scope of the associate rule is a larger area that aligns with the Texas Gulf Coast. Further study shows that this regional association rule a cannot be discovered at the Texas state level due to its insufficient confidence (less than 50%) on a global scale.

2 Related Work

The areas most relevant to our work are on hot-spot discovery and spatial association rule mining.

2.1 Hot-Spot Discovery

Hot spots are traditionally defined as the clusters of “more than usual interest, activity, or popularity” with respect to spatial coordinates [29]. Hot-spot discovery has been investigated in spatial statistics and data mining research.

In spatial statistics, detection of hot spots using a variable resolution approach [7] is investigated to minimize the effects of spatial superposition. In [44], a region-growing method for hot-spot discovery is described, which selects seed points first and then grows clusters from these seed points by adding neighbor points as long as a density threshold is satisfied. The definition of hot spots

is extended in [25] using circular zones for multiple variables. Getis and Ord propose a popular method to find hot spots in spatial datasets relying on the G* Statistic [19,35]. G* Statistic detects local pockets of spatial association, and the value of G* depends on an *a priori* given scale of the packets and is calculated for each object individually. Visualizing the results of G* calculations graphically reveals hot spots (aggregates of objects with values of G* higher than expected) and cold spots (aggregates of objects with values of G* lower than expected). Note that such aggregates are not formally defined clusters since the G*-based method has no built-in clustering capabilities. Instead, hot spots are inferred from visualization and manual selection.

An alternative approach for hot-spot discovery relies on clustering in data mining. Wang *et al.* [47] introduce a “region-oriented” clustering algorithm to select hot spots to satisfy certain conditions such as density. Their approach uses statistical information, for example, means and standard deviations, instead of a fitness function to evaluate a cluster. Eick *et al.* [17,16] propose Supervised Clustering to maximize cluster purity while keeping the number of clusters low. This paper applies Supervised Clustering to a new problem to find interesting regions (hot spots) that maximize a given fitness function. In this paper, we define two plug-in fitness functions for hot-spot discovery with respect to a class attribute and for identifying the scope of a regional association rule, respectively.

2.2 Spatial Association Rule Mining

Spatial association rule mining [24,5,28] applies association rule mining [1] to spatial datasets. Extended from the definition of traditional association rule mining, a spatial association rule takes the form of

$$P_1 \wedge P_2 \wedge \dots \wedge P_m \rightarrow Q_1 \wedge Q_2 \wedge \dots \wedge Q_n (\text{sup}\%, \text{con}\%).$$

It denotes an association relation among a set of predicates P_i ($i = 1, \dots, m$) and Q_j ($j = 1, \dots, n$), containing at least one spatial predicate. Spatial predicates may represent topological relations among spatial objects (e.g., intersecting, containing), or indicate a spatial orientation (e.g., north, left). The support of the rule ($\text{sup}\%$) measures the percentage of transactions containing both the antecedent and consequent of the rule. The confidence of the rule ($\text{con}\%$) indicates that $\text{con}\%$ of transactions satisfy both the antecedent and the consequent of the rule. A rule $P_1 \wedge P_2 \wedge \dots \wedge P_m \rightarrow Q_1 \wedge Q_2 \wedge \dots \wedge Q_n$ is *strong* if $\text{sup}\%$ and $\text{con}\%$ satisfy the minimum support and minimum confidence thresholds.

A common strategy used in spatial association rule mining is to divide the problem into three subtasks:

1. **Item representation and transaction definition:** define “items” and “transactions” for spatial datasets.
2. **Frequent itemset generation:** find all the itemsets that satisfy the minimum support threshold.
3. **Rule generation:** construct rules from the frequent itemsets that satisfy the minimum confidence threshold.

Apriori-style [1] association mining algorithms are often used in Subtasks 2 and 3. These type of algorithms require objects to be described by categorical attributes. Therefore, continuous attributes have to be discretized in Subtask 1, the step of data preprocessing. A transaction is not naturally defined in spatial space. If spatial association rule discovery is restricted to a reference feature (such as cities or wells), then transactions can be defined using the instances of this reference feature, as discussed in [24]. Our work adopts the same transaction model.

A daunting problem of spatial association rule mining, especially in real-world applications, is the huge number of generated patterns. Many associations are either already known geographic dependencies or explicitly represented in geographic databases. For example, that gas stations usually locate at road intersections is a well-known and uninteresting association. In order to extract nontrivial and interesting patterns, Borgorny and Sharma et al. [6,4,5,2,39] proposed a set of algorithms to discard previously known and uninteresting associations, using domain knowledge. In particular, the geographic dependencies between the target feature type and a relevant feature type are eliminated to reduce the input space for the frequent itemset generation and previously known and non-interesting geographic dependencies are further removed at the step of frequent itemset generation. To reduce the number of uninteresting patterns, we introduce the concept of Supervised Association Rules (Section 3.1, Definition 1) and seek associations containing the target feature type.

3 The Framework for Regional Association Rule Mining and Scoping

The framework of regional association rule mining and scoping consists of three steps:

Step 1 Region Discovery: identifying interesting regions for regional association rules.

Step 2 Regional Association Rule Mining: mining regional association rules among discovered regions.

Step 3 Regional Association Rule Scoping: determining the scope of regional association rules.

In the remaining part of the section, we will first discuss our reward-based method for region discovery which is closely involved with Steps 1 and 3, and we will formally define the goal of our framework and formulate the measure of interestingness.

3.1 Region Discovery

Our region discovery method employs a reward-based evaluation schema that evaluates the quality of the discovered regions. Given a set of regions $R = \{r_1, \dots, r_k\}$, identified from a spatial dataset $O = \{o_1, \dots, o_n\}$, the fitness of

R , $q(R)$, is defined as the sum of the rewards obtained from each region r_j ($j = 1 \dots k$):

$$q(R) = \sum_{j=1}^k (i(r_j) \times \text{size}(r_j)^\beta) \quad (1)$$

where $i(r_j)$ is the interestingness measure of a region r_j , a quantity based on domain interest to reflect the degree to which the region is newsworthy. Our reward-based method seeks a set of regions R such that the sum of rewards over all of its constituent regions is maximized. $\text{size}(r_j)^\beta$ ($\beta > 1$) in $q(R)$ increases the value of the fitness nonlinearly with respect to the number of objects in O belonging to the region r_j . A region reward is proportional to its interestingness, but given two regions with the same value of interestingness, a larger region receives a higher reward to reflect a preference given to larger regions.

We employ clustering algorithms for region discovery. A region is a contiguous subspace that contains a set of spatial objects such that for each pair of objects belonging to the same region, there always exists a path within this region that connects them. We search for regions r_1, \dots, r_k such that:

1. $r_i \cap r_j = \emptyset, i \neq j$, that is, the regions are disjoint.
2. $R = \{r_1, \dots, r_k\}$ maximizes $q(R)$.
3. $r_1 \cup \dots \cup r_k \subseteq O$. The generated regions are not required to be exhaustive with respect to the spatial dataset O . It is possible that some objects do not belong to any identified regions; these objects are discarded as outliers due to the lack of interestingness.
4. r_1, \dots, r_k are ranked based on their reward values. The higher rewards a region receives, the more interesting the region is, with respect to the fitness function q .

3.2 Problem Formulation

Let O be a spatial dataset, $S = \{s_1, s_2, \dots, s_l\}$ be a set of spatial attributes, $A = \{a_1, a_2, \dots, a_m\}$ a set of non-spatial attributes, and $CL = \{cl_1, cl_2, \dots, cl_n\}$ a set of class labels. Let

$$\begin{aligned} I &= S \cup A \cup CL \\ &= \{s_1, s_2, \dots, s_l; a_1, a_2, \dots, a_m; cl_1, cl_2, \dots, cl_n\} \end{aligned}$$

be the set of all the items in O , and let $T = \{t_1, t_2, \dots, t_N\}$ be the set of all the transactions. T can be represented as a relational table, which contains N tuples conforming to the schema I (I contains $l + m + n$ items). An item $i \in I$ is a binary variable whose value is 1 if the item is presented in t_i ($i = 1, \dots, N$) or 0, otherwise. Consequently, the set of transactions T is classified based on the given class structure CL .

Our framework leads to a class-guided generation of association rules that sheds more light on the patterns related to the given class structure. We define such rules as supervised association rules.

Definition 1 (Supervised Association Rule). A *supervised association rule* a is of the form $P \rightarrow Q$, where $P \subseteq I$, $Q \subseteq I$, $P \cap Q = \emptyset$, and $(P \cup Q) \cap CL \neq \emptyset$.

The rule a holds in the O with the confidence $conf$ and the support sup :

$$sup(P \rightarrow Q) = \frac{|P \cup Q|}{N}$$

$$conf(P \rightarrow Q) = \frac{|P \cup Q|}{|P|}$$

where $| |$ denotes the number of elements in a set. A supervised association rule is *strong* if it satisfies user-specified minimum support (min_sup) and minimum confidence (min_conf) thresholds: $sup(P \rightarrow Q) \geq min_sup$ and $conf(P \rightarrow Q) \geq min_conf$.

The goal of regional association rule scoping is to compute a set of regions where a given association rule is valid. The scope of a regional association rule represents the spatial impact of this regional pattern. We give formal definition of the scope of an association rule below.

Definition 2 (Scope of an Association Rule). The *scope* of an association rule a is a set of regions in which the association rule a satisfies the min_sup and min_conf thresholds.

Given these definitions and nomenclature, the problem of regional association rule mining and scoping can be formulated as follows.

Find: interesting regions, supervised association rules from the discovered regions, and the scope of regional association rules.

Given: an itemset I , a classified transaction set T , a set of fitness functions for different measure of interestingness.

3.3 Measure of Interestingness

The reward-based framework is designed to support many plug-in interestingness functions, corresponding to various domain interests. The framework utilizes the duality between regions and regional association rules. The framework first identifies “hot” regions using the interestingness function $i_{hotpot_coldspot}$. After strong regional association rules are identified, the scope of those rules are then calculated, using another interestingness function i_{scope} . Although the same clustering algorithm and the same dataset are used in two different steps, different sets of regions are returned in two steps due to the different measure of interestingness defined in the fitness functions.

In function $i_{hotpot_coldspot}$, the measure of interestingness is based on a set of class labels CL . It rewards regions whose probability distribution of CL significantly deviates from its priori probability. A region is a *hot spot/cold spot* if its probability distribution of CL is significantly higher / lower than an expected probability. The interestingness function $i_{hotpot_coldspot}$ is calculated based on

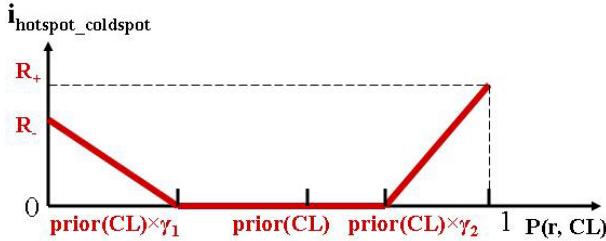


Fig. 2. The interestingness function $i_{\text{hotpot_coldspot}}$ using $\eta = 1$

$P(r, CL)$ and $priori(CL)$, with the following parameters: η , γ_1 , γ_2 , R_+ , R_- , where $\eta > 0$, $\gamma_1 \leq 1 \leq \gamma_2$, $0 \leq R_+, R_- \leq 1$. $P(r, CL)$ is the probability of objects in a region r belonging to CL ; $priori(CL)$ is the probability of objects in the global dataset O belonging to CL ; R_+ and R_- are the maximum rewards for hot spots and cold spots, respectively.

$$i_{\text{hotpot_coldspot}} = \begin{cases} \left[\frac{priori(CL) \times \gamma_1 - P(r, CL)}{priori(CL) \times \gamma_1} \times R_- \right]^\eta & \text{if } P(r, CL) < priori(CL) \times \gamma_1 \\ \left[\frac{P(r, CL) - priori(CL) \times \gamma_2}{1 - priori(CL) \times \gamma_2} \times R_+ \right]^\eta & \text{if } P(r, CL) > priori(CL) \times \gamma_2 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The parameter η determines how quickly the value of interestingness grows to the maximum value (either R_+ or R_-). If η is set to 1, the interestingness function changes linearly, as shown in Figure 2. In general, the larger the value for η is, the higher rewards for purer clusters are. $priori(CL) \times \gamma_1$ and $priori(CL) \times \gamma_2$ determine the thresholds based on which a reward is given to a cluster.

The following example explains how to calculate the fitness of a clustering schema X of an example dataset using Equations 1 and 2.

Example: Let us assume a clustering schema R is evaluated with respect to the class of interest *dangerous* (high-level arsenic) concentration with $priori(\text{dangerous}) = 0.2$ and a dataset that contains 1000 examples. Suppose that the dataset is partitioned into 4 clusters denoted as

$X = \{x_{11}, x_{12}, x_{13}, x_{14}\}$, and $|x_{11}| = 50$, $|x_{12}| = 200$, $|x_{13}| = 400$, $|x_{14}| = 350$. Assume that there are 20, 100, 80, and 0 objects labeled “dangerous” in the 4 clusters, respectively. $P(x_{11}, \text{dangerous}) = \frac{20}{50} = 0.4$,

$P(x_{12}, \text{dangerous}) = \frac{100}{200} = 0.5$, $P(x_{13}, \text{dangerous}) = \frac{80}{400} = 0.2$,

$P(x_{14}, \text{dangerous}) = \frac{0}{350} = 0$. The parameters used in the fitness function are as follows: $\gamma_1 = 0.5$, $\gamma_2 = 1.5$, $R_+ = 1$, $R_- = 1$. Hence, $priori(CL) \times \gamma_1 = 0.2 \times 0.5 = 0.1$, and $priori(CL) \times \gamma_2 = 0.2 \times 1.5 = 0.3$. With this setting, a cluster does not receive any reward if its probability of class “dangerous” is not significantly higher or lower than the expected probability, that is, the value is between $priori(CL) \times \gamma_1 = 0.1$ and $priori(CL) \times \gamma_2 = 0.3$. Therefore, x_{13} receives no reward. The interestingness for the other clusters using $\eta = 1$ is

$$i_{hotpot_coldspot}(x_{11}) = \left(\frac{0.4 - 0.3}{1 - 0.3}\right)^1 = \frac{1}{7},$$

$$i_{hotpot_coldspot}(x_{12}) = \left(\frac{0.5 - 0.3}{1 - 0.3}\right)^1 = \frac{2}{7},$$

$$i_{hotpot_coldspot}(x_{14}) = \left(\frac{0.1 - 0}{0.1}\right)^1 = 1.$$

The fitness value of the clustering schema X calculated using Equation 1 with $\beta = 1.1$ is

$$\begin{aligned} q(X) &= \frac{1}{7} \times \left(\frac{50}{1000}\right)^{1.1} + \frac{2}{7} \times \left(\frac{200}{1000}\right)^{1.1} + \\ &\quad 0 \times \left(\frac{400}{1000}\right)^{1.1} + 1 \times \left(\frac{350}{1000}\right)^{1.1} \\ &= 0.369 \end{aligned}$$

Function i_{scope} evaluates the interestingness of a region for a given association rule. Let a be an association rule, $conf(a, r)$ the confidence of a in a region r , and $sup(a, r)$ the support of a in r , we define the interestingness $i_{scope}(r)$ of region r with respect to the given association rule a as follows:

$$i_{scope}(r) = \begin{cases} 0 & \text{if } sup(a, r) < min_sup \times \delta_1 \text{ or} \\ & \quad conf(a, r) < min_conf \times \delta_2, \\ \left(\frac{sup(a,r)}{min_sup}\right)^{\eta_1} \left(\frac{conf(a,r)-min_conf \times \delta_2}{1-min_conf \times \delta_2}\right)^{\eta_2} & \text{otherwise.} \end{cases} \quad (3)$$

In regional association rule scoping, a region's reward is proportional to its interestingness, which is determined based on the confidence and support of association rule a in region r . In Equation 3, the thresholds $min_sup \times \delta_1$ and $min_conf \times \delta_2$ are introduced to weed out regions in which the association a barely holds. The minimum support and confidence thresholds prevent the clustering solution from containing large clusters with low interestingness. Values of parameters η_1 and η_2 ($\eta_1, \eta_2 > 0$) determine the weight to the increment of the support and confidence, respectively.

The measure of interestingness defined in i_{scope} uses "soft" instead of "hard" thresholds to avoid a harsh crisp effect [3]. For example, with $\delta_1 = \delta_2 = 0.9$, the function $i_{scope}(r)$ rewards regions as long as their confidence or support thresholds are within 90% of the hard thresholds min_conf and min_sup . For example, let's assume that $min_sup = 10\%$, $min_conf = 80\%$, and that the association rule under consideration has support = 9% and confidence = 100% in a region r' . In this case, instead of assigning zero reward to region r' , we argue to reward the region because the confidence of the rule in region r' is significantly above the min_conf threshold and its support is just a little bit lower (1%) than the min_sup threshold. Our approach uses a quantitative evaluation method that assigns a higher degree of interestingness and consequently a higher reward to regions whose support and confidence are high with respect to an association rule of interest. Once an association rule a is discovered from a particular region

r in the first place, we know that region r from which the association rule a originates, receives a positive reward due to the fact that a satisfies the support and confidence thresholds in r . Consequently, region r will always be contained in the set of regions that define the scope of association rule a .

4 Algorithms

4.1 Region Discovery

We formulate region discovery as a clustering problem to search for clusters that maximize domain-specific metrics as described in detail in previous section. Different measure of interestingness may lead to different sets of identified regions. Consequently, clustering algorithms embedded in the framework should allow for plug-in fitness functions. However, the use of fitness functions is quite uncommon in clustering methods, although a few exceptions exist, for example, the hierarchical clustering algorithm CHAMELEON [23] uses fitness functions to evaluate inter-connectivity and proximity between two clusters. Furthermore, our region discovery method is different from traditional clustering methods as it is geared toward finding interesting places with respect to a given measure of interestingness. Clusters are ranked based on reward values, and clusters receive low rewards are discarded as outliers and will not be identified as interesting regions.

We have designed and implemented a new Supervised Clustering algorithm using Multi-Resolution Grids (SCMRG). SCMRG is a hierarchical, grid-based method that utilizes a top-down search. The spatial space of the dataset is partitioned into grid cells. Each grid cell at a higher level is partitioned further into smaller cells at the lower level, and this process continues as long as the sum of the rewards of the lower level cells $q(R)$ is not decreased. The regions returned by SCMRG are the combination of grid cells obtained at different levels of resolution. The number of clusters, k , is calculated by the algorithm itself.

Algorithm 1 gives the pseudo-code of SCMRG. A queue data structure is used to store all the cells that need to be processed. The algorithm starts at a user-defined level of resolution and considers the following three cases when processing a cell c :

Case 1: if the cell c receives a reward, and its reward is greater than the sum of the rewards of its children ($succ(c)$) and also greater than the sum of rewards of its grandchildren, this cell is returned as a cluster by the algorithm (steps 15-17).

Case 2: if the cell c does not receive a reward and its children and grandchildren do not receive a reward, neither the cell nor any of its descendants will be labeled clusters (steps 23-29).

Case 3: otherwise, put all the children of the cell c ($succ(c)$) into a queue for further processing (steps 18-21, steps 24-28).

The algorithm traverses through the hierarchical structure and examines those cells in the queue from the higher level. It uses a user-defined cell size as a depth

Algorithm 1. The Algorithm of Supervised Clustering using Multi-Resolution Grids (SCMRG)

Input:

- A fitness function.
- A level of resolution l for the initial grid structure.
- The minimum cell size. A cell will not be divided further if it approaches the minimum cell size.

Output:

- Discovered regions $R = \{r_1, \dots, r_k\}$.

SCMRG (min_cell_size)

1. Determine a level of resolution l to start with.
2. Assign spatial objects to grid cells.
3. **for** each cell c at the current level l **do**
4. enqueue(c , cellQueue).
5. **end for**
6. **while** *NOT empty*(cellQueue) **do**
7. $c = \text{dequeue}(\text{cellQueue})$.
8. $r = \text{reward}(c)$. {Calculate reward for the cell.}
9. **for** each $c_{\text{child}} \in \text{succ}(c)$ **do**
10. $r_{\text{children}} = r_{\text{children}} + \text{reward}(c_{\text{child}})$.
11. **end for** {Calculate reward for its children.}
12. **for** each $c_{\text{grandchild}} \in \text{succ}(\text{succ}(c))$ **do**
13. $r_{\text{grandchildren}} = r_{\text{grandchildren}} + \text{reward}(c_{\text{grandchild}})$.
14. **end for** {Calculate reward for its grandchildren.}
15. **if** $r > 0$ {The cell receives a reward.}
16. **if** $r > r_{\text{children}}$ AND $r > r_{\text{grandchildren}}$
17. label the cell a cluster.
18. **else** {The cell should be divided further.}
19. **if** (the size of each $c_{\text{child}} \in \text{succ}(c) > \text{min_cell_size}$)
20. enqueue($\text{succ}(c)$, cellQueue).
21. **end if**
22. **end if**
23. **else if** $r = 0$ {The cell does not receive a reward.}
24. **if** NOT ($r_{\text{children}} = 0$ AND $r_{\text{grandchildren}} = 0$)
25. **if** (the size of each $c_{\text{child}} \in \text{succ}(c) > \text{min_cell_size}$)
26. enqueue($\text{succ}(c)$, cellQueue).
27. **end if**
28. **end if** {The cell should be divided further.}
29. **end if**
30. **end while**
31. Collect all the cluster-labeled cells from different levels.
32. Obtain regions by merging neighbor clusters if it improves the fitness.
33. Return the obtained regions.

boundary. Cells smaller than this cell size will not be split any further (step 19, step 25). Finally, SCMRG collects all the cells that have been identified in Case 1 from different levels, and merges neighbor clusters if the overall fitness can be improved. The obtained regions are returned as the result of the SCMRG clustering algorithm (steps 31-33).

This hierarchical grid-based approach captures clustering information associated with spatial cells without recourse to the individual objects because we do not drill down a cell if it does not look so promising (Case 2). SCMRG avoids time-consuming distance calculation because it uses the grid structure to define the neighborhood of objects. The computational complexity of SCMRG is thus linear in the number of grid cells processed, which is usually much less than the number of objects. Thus, the algorithm is capable of processing large datasets efficiently. The SCMRG algorithm has some similarity with the STING clustering algorithm [47]. The difference is that the SCMRG algorithm focuses on finding interesting cells (those receive high rewards) instead of cells that contain answers to a given query. In addition, SCMRG only computes cell statistics when needed and not in advance as STING does, thus saving storage space as well.

The complexity of the SCMRG algorithm is controlled by two factors: the number of the candidate cells in the queue and the calculation of the fitness. The algorithm calculates the fitness of all objects inside a cell and a cell will not be further divided if drilling down cannot improve the current reward. The number of cells of a layer is less than one-fourth of the number of the layer one level lower. The total number of cells to be processed in the worst case is less than $1.33N_c$, where N_c is the number of the cells at the bottom layer¹. The actual number of cells is usually less than $1.33N_c$ due to the reward-based pruning. It is also reasonable to assume that each cell at the bottom layer likely contains many objects because the reward function is designed to favor larger cell (a.k.a. larger clusters). In our empirical study, the average cell size is above 400 objects. In general, the total number of cells is much less than the total number of objects. Let the cost of fitness calculation is $O(q)$. Thus the complexity of the algorithm in average is usually much better than $O(N) \times O(q)$, where N is the total number of objects in the dataset.

The example in Figure 3 explains the procedure of the SCMRG algorithm using a sample dataset. The first decomposition results into four cells $c_{11}, c_{12}, c_{13}, c_{14}$ at Level 1. If the reward of c_{11} is greater than the sum of the rewards of its children, and if it is also greater than the sum of rewards of its grandchildren, c_{11} is then labeled a cluster according to Case 1. Cell c_{14} does not receive any rewards, if neither its children nor grandchildren receive any rewards. According to Case 2, c_{14} is not labeled a cluster, and its successors are not saved in the queue. Although Cell c_{13} receives no reward, assume its children receive rewards, all the children of c_{13} are saved in the queue to be further processed (Case 3). The cells at Level 1 are then divided into Levels 2 and 3, and the same procedure is applied to all the cells in the queue. Each cell is labeled accordingly. The

¹ Total number of cells = $N_c \times (1 + \sum_{n \rightarrow \infty} \frac{1}{4^n})$ and $\sum_{n \rightarrow \infty} \frac{1}{4^n} = \frac{1}{3}$.

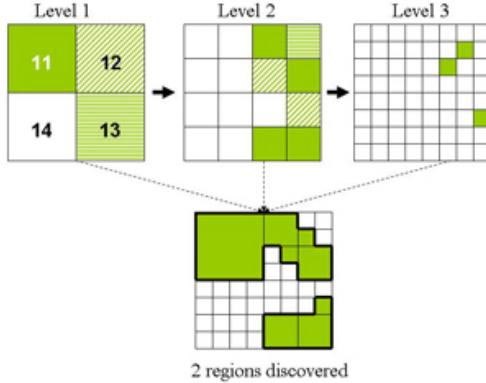


Fig. 3. Runing the SCMRG algorithm on a sample dataset

intermediate results are shown at Levels 2 and 3 in Figure 3. Neighbor clusters are merged if this improves the fitness. In this example, two regions are identified.

4.2 Generation of Regional Association Rules

Once regions are identified, we construct frequent itemsets for each region. Our Supervised_Apriori_Gen algorithm (Pseudo code is provided in our previous work [13]) extends the *Apriori* algorithm [1] by utilizing a given class structure.

The *Apriori* algorithm first makes a single pass over the dataset to determine the support of each single item, which generates all frequent 1-itemsets F_1 . Next, the algorithm iteratively generates candidate k-itemsets using the frequent (k-1)-itemsets found in the previous iteration. A k-itemset is an itemset that has k attributes. A candidate itemset is pruned if it is not frequent. The algorithm terminates when there are no new frequent itemsets generated, for example, $F_k = \emptyset$. Supervised_Apriori_Gen algorithm uses a different approach: the given class structure is incorporated by enforcing that each candidate k-itemset include at least one class label; otherwise it is pruned even if it is frequent. The Supervised-Apriori-Gen uses the $F_{k-1} \times F_{k-1}$ method [43] to merge a pair of frequent (k-1)-itemsets. Basically, let $A = \{a_1, a_2, \dots, a_{k-1}\}$ and $B = \{b_1, b_2, \dots, b_{k-1}\}$ be a pair of frequent (k-1)-itemsets. A and B are merged to form a k-itemset $\{a_1, a_2, \dots, a_{k-1}, b_{k-1}\}$ if they satisfy the following conditions:

$$a_i = b_i \quad (for i = 1, 2, \dots, k-2) \text{ and } a_{k-1} \neq b_{k-1}.$$

After frequent itemsets are generated, we use the same approach proposed by the *Apriori* algorithm to generate strong supervised association rules using the min_conf threshold.

5 Arsenic Regional Association Rule Mining and Scoping in the Texas Water Supply

In this section, we describe the experimental procedures of applying the framework of regional association rule mining and scoping to a real world case study that identifies arsenic spatial risk patterns in the Texas water supply. We then discuss the experimental results and evaluate the performance of the proposed framework.

The experiments are conducted in four steps:

1. Data collection and data preprocessing, including cleaning data, transforming continuous attributes into categorical attributes, and constructing transactions using water wells as the reference feature.
2. Identifying arsenic *hot spots* and *cold spots*. A region whose arsenic distribution is significantly higher than the Texas state level is considered an arsenic hot spot; a region whose arsenic distribution is significantly lower the Texas state level is considered an arsenic cold spot.
3. Mining supervised association rules from each identified region and for the complete dataset.
4. Determining scope of strong supervised association rules.

5.1 Data Collection and Data Preprocessing

The datasets used in this study are extracted from the Texas Ground Water Database (GWDB) maintained by the Texas Water Development Board, the state agency in charge of statewide water planning [45]. The Texas Water Development Board has monitored and analyzed arsenic concentration over the last 30 years. Arsenic in very high concentration is poisonous. Long term exposure to arsenic, even though at low level, can still lead to increased risk of cancers [42]. Arsenic is derived from both anthropogenic sources, such as the drainage from mines and mine tailings, pesticides, and biocides, and from natural sources, such as the hydrothermal leaching of arsenic-containing minerals or rocks. The World Health Organization has reported arsenic in drinking water in U.S., Thailand, Mexico, India, Hungary, Ghana, Chile, China, Bangladesh, and Argentina [48], as one of the key parameters for drinking water quality and safety evaluation.

Because data collection and maintenance procedures and standards have changed over the years in GWDB, datasets have to be cleaned to deal with problems such as missing values, inconsistent data, and duplicate entries. The obtained arsenic spatial dataset includes spatial attributes (S), non-spatial attributes (A), and class labels (CL) for each water well. Some of the spatial attributes are directly extracted from the database, such as *river basin*, *zone*, *latitude* and *longitude*. Implicit spatial attributes, such as *distance* between wells and rivers, are estimated using the 9-intersection model [15]. Non-spatial attributes are selected with the assistance of domain experts [21,26,37]; they include *well depth*, and concentration of *fluoride*, *nitrate*, and other chemical metal

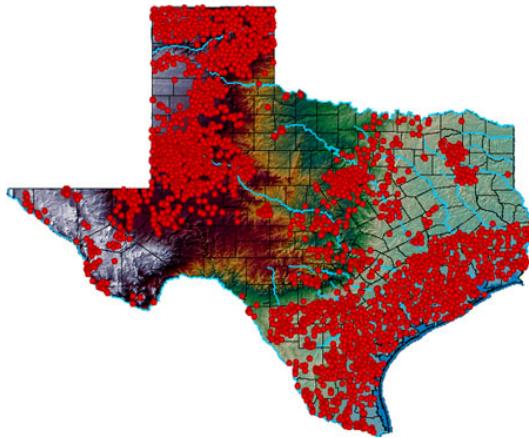


Fig. 4. Arsenic contamination in Texas; background depicts Texas terrain color ramp. Legend: red (or dark gray) dots – dangerous wells.

elements including *vanadium*, *iron*, *molybdenum* and *selenium*. Among those attributes, the attribute *well depth* is used for studies on mobilizing mechanism; the attributes *vanadium* and *molybdenum* have similar geochemical behavior; the attributes *fluoride*, *nitrate*, *iron*, and *selenium* may suggest the ultimate origin of arsenic. The arsenic dataset generated by our research group and the dataset is available on the web at [10].

We classify water wells into two classes: *safe* and *dangerous*. Based on the standard for drinking water defined by the Environment Protection Agency [46], a well is considered dangerous if its arsenic concentration level is above $10\mu\text{g/l}$. To ensure the quality of the association rule generated in the study, we only select lab test results that use honored sampling procedures. This results in 11,922 records selected from GWDB after data preprocessing. Figure 4 illustrates arsenic contamination in Texas, where dangerous wells are in red (or dark gray).

Table 1 describes the 7 non-spatial attributes used in the arsenic dataset. The table lists the mean and the standard deviation of those continuous attributes before discretization. In preparation of the association rule mining, continuous attributes excluding latitude and longitude are first converted into categorical attributes. In general, two different methods are used for discretization of continuous attributes: unsupervised discretization without using class information and supervised discretization using class information [43]. In our experiments, we adopt the supervised method Recursive Minimal Entropy Partitioning introduced in [18]. The supervised entropy-based method uses class labels *dangerous* and *safe* to place the splits in a way that maximizes the purity of arsenic classes in the intervals. This discretization method maximized the support for arsenic class attribute, facilitating the discovery of supervised association rules involving with arsenic. Hence the method can effectively find the supervised association rules related with arsenic classes. The method produces unequal bin sizes and

Table 1. Arsenic dataset

Total # of Wells	11,922		
Non-Spatial Attributes	Mean	STD	Splitting Points
1. well depth (foot)	587.959	654.962	215.5
2. nitrate (mg/l)	11.362	27.499	0.085, 0.455, 16.1, 28.085
3. fluoride (mg/l)	1.161	1.349	0.315, 2.445, 3.375, 4.605
4. vanadium ($\mu g/l$)	8.755	25.827	1.2, 2.05, 2.95, 3.25, 5.945, 11.85,
5. iron ($\mu g/l$)	9.226	15.651	19.95, 20.05, 37.95
6. molybdenum ($\mu g/l$)	259.882	1320.784	9.05, 11.35, 19.95, 20.1, 28.1, 47.2, 51.05
7. selenium ($\mu g/l$)	14.243	34.75	4.995, 5.01, 19.95, 20.05, 34.65,
			14.243, 34.75
			43.05, 52.85, 74.55

has been shown to produce better results in data mining tasks [14]. The splitting points of each continuous attribute are listed in Table 1. For example, the value of nitrate concentration has been discretized into five intervals with respect to the arsenic classes: (0,0.085], (0.085,0.455], (0.455,16.1], (16.1,28.085], and (28.085, ∞) (measurement unit mg/l).

5.2 Region Discovery for Arsenic Hot/Cold Spots

We have re-discovered several interesting risk regions with high arsenic concentration (hot spots), which have been studied by geoscientists before. We have also identified regions with low arsenic concentration (cold spots). The association rules that we constructed from those identified regions can help geoscientists identify the causes of high arsenic concentration in different regions. We now present our results with validation from the published results in the geosciences for both region discovery and association rule mining and scoping.

In region discovery, the SCMRG algorithm is applied to the dataset that consists of longitude and latitude of wells along with arsenic class labels *dangerous* or *safe* using Equation 2. Figure 5 depicts the result of the top four regions that have received the highest reward. Specifically, Regions 1 and 3 have high density of dangerous wells, and Regions 2 and 4 have high density of safe wells. Hot spot Region 1 overlaps with the arsenic risk zone reported in the National Water-Quality Assessment Program [31], and hot spot Region 3 is confirmed as an arsenic risk zone by Parker's work [37].

If we are interested in finding larger regions with lower purity, using a larger value of β results in a bigger size of the regions. Figure 6 shows enlarged regions

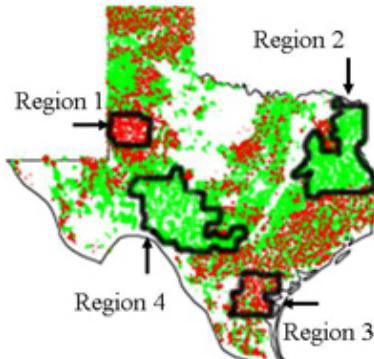


Fig. 5. Interesting regions are identified using $\beta = 1.01$, $\eta = 1$, $\gamma_1 = 0.5$, $\gamma_2 = 1.5$, $R+ = 1$, $R- = 1$. Average region purity = 0.85.

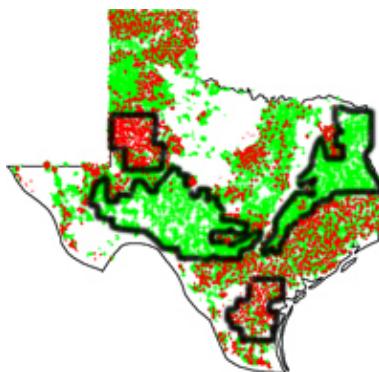


Fig. 6. Interesting regions are identified using $\beta = 1.035$, $\eta = 1$, $\gamma_1 = 0.5$, $\gamma_2 = 1.5$, $R+ = 1$, $R- = 1$. Average region purity = 0.83.

when β is increased from 1.01 to 1.035. In our experiments, we adjusted the granularity of regions by the quality of rules discovered in step 3. We observed that $\beta = 1.01$ and $\eta = 1$ give us the best results in the rules constructed in supervised association rule mining.

5.3 Regional Association Rule Mining

The Supervised_Apriori_Gen algorithm is used to generate frequent itemsets for all the regions identified. We use $min_support = 10\%$ and $min_confidence = 70\%$ thresholds for the experiments. We present the first few rules for the regions investigated, which are all meaningful and important according to the arsenic study literature.

Mining regional rules in arsenic hot spots discovers attributes that are associated with high arsenic concentration; in cold spots it discovers attributes related to low arsenic concentration. For example, in Region 3 of Figure 5, we discover

$$\begin{aligned} & \text{is_a}(X, \text{Well}) \wedge \text{nitrate}(X, 0 - 0.085) \\ \rightarrow & \text{arsenic_level}(X, \text{dangerous}) \quad (100\%). \end{aligned} \quad (1)$$

The rule states, with 100% confidence, that the wells in Region 3 with nitrate concentration lower than 0.085mg/l have dangerous arsenic concentration. The strong association between nitrate and high arsenic concentration is verified by Hudak's work [21] in environmental geology.

In Region 1 of Figure 5, we also discover

$$\begin{aligned} & \text{is_a}(X, \text{Well}) \wedge \text{vanadium}(X, 20.05 - 37.95) \wedge \text{selenium}(X, 74.55 - \infty) \\ \rightarrow & \text{arsenic_level}(X, \text{dangerous}) \quad (100\%). \end{aligned} \quad (2)$$

The rule states with 100% confidence that the wells in Region 1, with vanadium concentration between 20.05 and $37.95\mu\text{g/l}$ and selenium concentration larger than $74.55\mu\text{g/l}$, have dangerous arsenic concentration. Our discovery is confirmed by the work of Lee *et al.* in [26].

Our experimental results also show some novel rules that have not been reported in the literature of arsenic analysis. For example, in Region 1 the following rule is discovered:

$$\begin{aligned} & \text{is_a}(X, \text{Well}) \wedge \text{depth}(X, 0 - 215.5) \wedge \text{iron}(X, 19.65 - 20.05) \\ \rightarrow & \text{arsenic_level}(X, \text{dangerous}) \quad (100\%). \end{aligned} \quad (3)$$

The rule indicates that shallow wells with a certain range of iron concentration are associated with high arsenic concentration. We hope that the results from our study will help domain experts in selecting interesting hypotheses for further scientific exploration.

Furthermore, we are interested to know whether the rules are different in different regions. We compared the sets of rules generated for Region 1 and Region 3 (hot spots), as well as for Region 2 and Region 4 (cold spots). Due to different geographical structure and farm activities of the study area, the spatial risk patterns associated with arsenic are different in each region. For example, comparing the previously studied rule 1 identified in Region 3 with rule 4 extracted from Region 1:

$$\begin{aligned} & \text{is_a}(X, \text{Well}) \wedge \text{nitrate}(X, 28.085 - \infty) \wedge \text{fluoride}(X, 4.605 - \infty) \\ \rightarrow & \text{arsenic_level}(X, \text{dangerous}) \quad (100\%). \end{aligned} \quad (4)$$

Instead of being related to relatively low concentration of nitrate ($< 0.085\text{mg/l}$), the rule says that with 100% confidence, the wells in Region 3, with high nitrate concentration ($> 28.085\text{mg/l}$) and fluoride concentration higher than 4.605mg/l , have dangerous arsenic concentration.

Rules in Regions 2 and 4 (cold spots) shed light on what may prevent high arsenic concentration. For example, we find the following rule, discovered both in Regions 2 and 4, states what is associated with low arsenic concentration.

$$\begin{aligned} & \text{is_a}(X, \text{Well}) \wedge \text{nitrate}(X, 0.455 - 16.1) \wedge \\ & \text{fluoride}(X, 0.095 - 0.315) \wedge \text{vanadium}(X, 3.25 - 5.945) \\ & \rightarrow \text{arsenic_level}(X, \text{safe}) \text{ (100\%)} \end{aligned} \quad (5)$$

For comparative purposes, we also mine supervised association rules in the whole dataset. Using low support values in global datasets to find more interesting association rules has been suggested by [27]. However, even with a rather low support threshold $\text{min_support} = 1\%$, none of the top ranked interesting regional association rules we identified previously are included among over 100,000 resulting rules. On the other hand, up to 300 rules on average are identified per region using our framework with $\text{min_support} = 10\%$ and $\text{min_confidence} = 70\%$ thresholds. Regional association rules identified from those arsenic hot/cold spots tend to be more revealing and interesting. Not surprisingly, a large portion of 100,000 statewide association rules are trivial and general rules, such as

$$\begin{aligned} & \text{is_a}(X, \text{Well}) \wedge \text{water_use}(X, \text{"by humam beings"}) \wedge \text{arsenic_level}(X, \text{safe}) \\ & \rightarrow \text{inside}(X, \text{Basin19}) \text{ (86\%)} \end{aligned} \quad (6)$$

This global association rule claims that wells which are used by human beings and have safe arsenic concentration are very likely (confidence is 86%) located in river basin 19 (in San Antonio area). It is a well-known fact in Texas.

5.4 Region Discovery for Regional Association Rule Scoping

We use the same clustering algorithm SCMRG but a different fitness function i_{scope} (Equation 3) for regional association rule scoping. The following four regional association rules with 100% confidence from Regions 1, 2, 3, and 4 are used as illustration examples in the rest of this section for regional association rule scoping. Association rules 1 and 3 are confirmed in arsenic literature [21,26].

Association Rule 1

$$\text{nitrate}(X, 28.31 - \infty) \wedge \text{arsenic_level}(X, \text{dangerous}) \rightarrow \text{depth}(X, 0 - 251.5)$$

Association Rule 2

$$\text{depth}(X, 0 - 251.5) \wedge \text{fluoride}(X, 0 - 0.085) \rightarrow \text{arsenic_level}(X, \text{safe})$$

Association Rule 3

$$\text{nitrate}(X, 0 - 0.085) \rightarrow \text{arsenic_level}(X, \text{dangerous})$$

Association Rule 4

$$\text{depth}(X, 251.5 - \infty) \wedge \text{nitrate}(X, 0.265 - 16.1) \rightarrow \text{arsenic_level}(X, \text{safe})$$

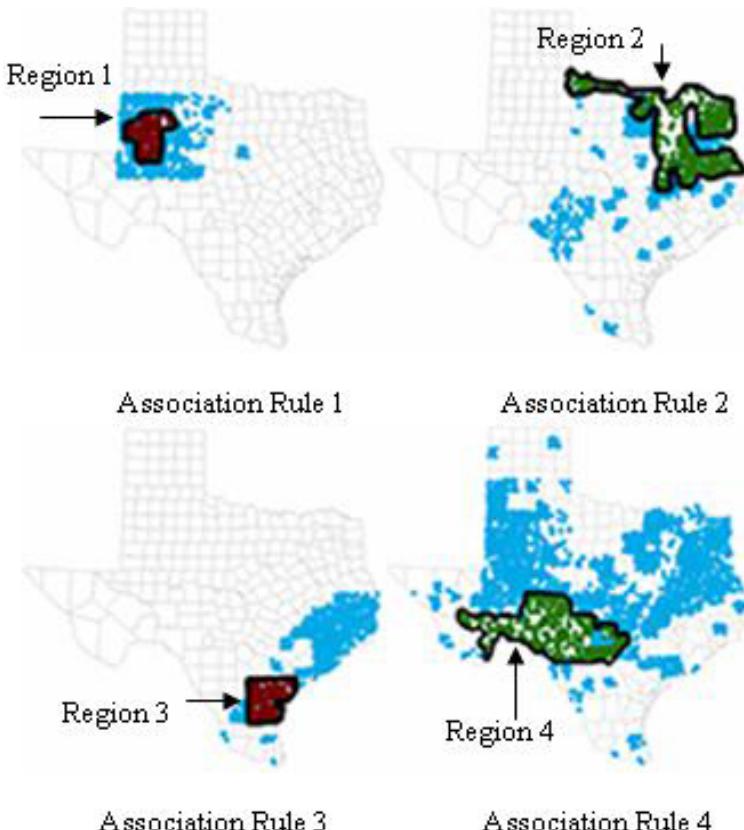


Fig. 7. Region - Regional association rule - Scope using $\beta = 1.01$, $\eta_1 = 1$, $\eta_2 = 1.1$, $\delta_1 = \delta_2 = 0.9$, $min_sup = 10\%$, $min_conf = 80\%$. Legend: regions are highlighted by bold border line; scopes are in color blue (or light grey).

Figure 7 depicts the scope of four association rules above. The scope of an association rule can contain several regions. The scope of Association Rule 1 (top row, left column) overlaps with the Texas High Plains. In this area, shallow depth wells (< 251.5 feet) indicate the aquifer is thin; thus, nitrate comes from surface contamination ($> 28.31 \text{ MG/L}$). Arsenic contamination is of geological origin and is then enhanced by the lack of dilution because the aquifer is thin. The scope of Association Rule 3 (bottom row, left column) is applicable to the whole Texas Gulf Coast because the geology there is similar. The scope of Association Rules 2 and 4 represents the areas where arsenic contamination is low. They are interesting places that domain scientists will explore in the future.

It is also important to point out that the scope of an association rule indicates how global, regional, or local a pattern is. For example, the scope of the association rule 4 in Figure 7 covers a large percentage of the global space ($> 75\%$). We find that the association rule 4 is also valid (holds with 85% confidence) in

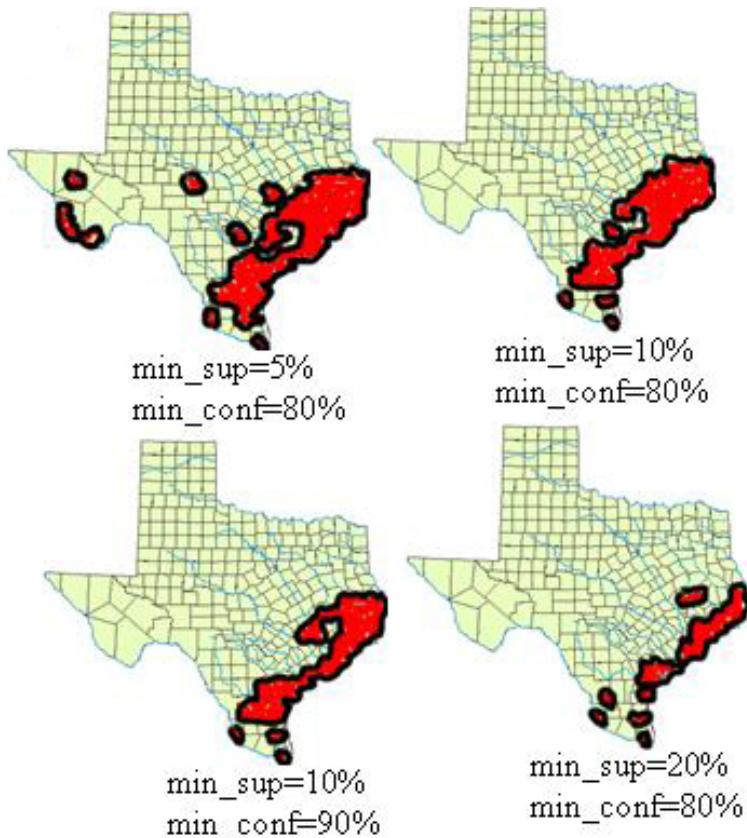


Fig. 8. The scope of a particular rule changes based on the different values of min_sup and min_conf . $\beta = 1.01$, $\eta_1 = 1$, $\eta_2 = 1.1$, $\delta_1 = \delta_2 = 0.9$, $\text{min_sup} = 10\%$, $\text{min_conf} = 80\%$.

the global dataset. Hence, it is indeed a global association rule. However, none of the other three association rules are discovered globally. We can also fine-tune the measure of interestingness for association rule scoping by varying its support and confidence thresholds for a given association rule. Figure 8 shows how the scope of the association rule 3 changes using different confidence and support thresholds. Typically, a lower value of the min_sup results in a larger scope; a higher value of the min_conf results in a smaller scope.

Association rule scoping has many applications that go beyond the proposed framework introduced in this paper. Scoping can be applied to any spatial association rules, including global association rules. For example, a domain expert can check whether an arsenic association, which is valid in Texas, also holds in Bangladesh, a country that has serious arsenic contamination in drinking water. It is also inspirational for domain experts to explore how the scope of an association rule changes, if an association rule is slightly modified, for example, a

condition in its antecedent is dropped. Furthermore, in addition to finding the scope where an association holds, it might be interesting to search for the scope where it does not hold. For example, if we find that high levels of iron associates with high arsenic concentration in one region, but with low arsenic concentration in another region, this case should be further analyzed. Last but not least, the regions obtained using association rule scoping can serve as a source for mining new association rules. For example, if we are interested in the places where high levels of iron associate with high levels of fluoride, $high_iron(X) \rightarrow high_fluoride(X)$. We can then determine the scope of this association rule and use the new obtained regions to mine new interesting association rules that provide further details that contribute to the association between iron and fluoride.

Our SCMRG algorithm is computationally efficient. On average, it takes 3.031 seconds for hot spots/cold spots discovery, and 4.68 seconds for regional association rule scoping. The computer has an Intel(R) Pentium(R) M, a CPU 1.2GHz, and 632 MB of RAM. The algorithm implemented in Java can be accessed on the Web at our open source project *Cougar² Java Library for Machine Learning and Data Mining Algorithms* [9].

6 Summary

One critical requirement for spatial data mining is the capability to analyze datasets at different levels of granularity, as well as analyze the data globally. We face two unique challenges in regional association mining and scoping: (1) how to determine regions from which regional association rules will be extracted, and (2) how to compute the scope of regional association rules. We solve the first issue using a reward-based region discovery algorithm that employs a grid-based supervised approach to identify interesting subregions in spatial datasets. We address the second problem by exploiting the duality between regional patterns and regions: regions are used to discover regional association rules; next the obtained regional association rules are used to determine places in which the association rules are valid. Such regions capture the scopes of regional patterns and provide a quantitative measure of how significant a regional association rule is in the global space.

We evaluate the proposed framework in a real-world case study to identify spatial risk patterns and risk zones of arsenic in the Texas water supply. The goal of the case study is to understand what regional associations exist between high arsenic concentration and other factors. We have identified arsenic hot spots and cold spots, created regional rules from the obtained regions, and evaluated the spatial impact of interesting regional associations. We are not interested in predicting whether a well is safe or dangerous because this information is already known. A classification algorithm would only be helpful if we could drill into the classification model to determine which factors are associated with high arsenic pollution. In general, our work can be viewed as an exploratory data analysis approach that centers on which features are potentially relevant in causing arsenic pollution. Moreover, our approach identified several new relationships between

arsenic and other factors which provide scientists with novel hypotheses for further exploration.

References

1. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Buneman, P., Jajodia, S. (eds.) Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C, pp. 207–216 (26–28, 1993)
2. Appice, A., Ceci, M., Lanza, A., Lisi, F.A., Malerba, D.: Discovery of spatial association rules in geo-referenced census data: A relational mining approach. *Intell. Data Anal.* 7(6), 541–566 (2003)
3. Bistarelli, S., Bonchi, F.: Interestingness is not a dichotomy: Introducing softness in constrained pattern mining. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 22–33. Springer, Heidelberg (2005)
4. Bogorny, V., Camargo, S., Engel, P.M., Alvares, L.O.: Mining frequent geographic patterns with knowledge constraints. In: GIS 2006: Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems, Arlington, Virginia, USA, pp. 139–146 (November 2006)
5. Bogorny, V., Kuijpers, B., Alvares, L.O.: Reducing uninteresting spatial association rules in geographic databases using background knowledge: a summary of results. *Int. J. Geogr. Inf. Sci.* 22(4), 361–386 (2008)
6. Bogorny, V., Valiati, J., Camargo, S., Engel, P., Kuijpers, B., Alvares, L.: Mining maximal generalized frequent geographic patterns with knowledge constraints. In: The 6th International Conference on Data Mining, Hong Kong, pp. 813–817 (December 2006)
7. Brimicombe, A.J.: Cluster detection in point event data having tendency towards spatially repetitive events. In: the 8th Intl. Conf. on GeoComputation (2005)
8. Celepcikay, O.U., Eick, C.F.: Reg: A regional regression framework for geo-referenced datasets. In: 17th ACM SIGSPATIAL International Conference on Advances in GIS (ACM SIGSPATIAL GIS)
9. CougarSquared Data Mining and Machine Learning Framework, Data Mining and Machine Learning Group, University of Houston (2011), <https://cougarsquared.dev.java.net/>
10. Data Mining and Machine Learning Group, University of Houston (2011), <http://www.tlc2.uh.edu/dmmlg/Datasets>
11. Ding, W., Eick, C.F., Wang, J., Yuan, X.: A framework for regional association rule mining in spatial datasets. In: The 6th IEEE International Conference on Data Mining, ICDM, (December 2006)
12. Ding, W., Eick, C.F., Yuan, X., Wang, J., Nicot, J.-P.: On regional association rule scoping. In: The International Workshop on Spatial and Spatio-temporal Data Mining in Cooperation with IEEE ICDM 2007, Omaha, NE, USA (October 2007)
13. Ding, W., Eick, C.F., Yuan, X., Wang, J., Nicot, J.-P.: A framework for regional association rule mining and scoping in spatial datasets. *Geoinformatica* 15(1), 1–28 (2011)
14. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: International Conference on Machine Learning, pp. 194–202 (1995)

15. Egenhofer, M.J., Franzosa, R.D.: Pointset topological spatial relations. *International Journal for Geographical Information Systems* 5(2), 161–174 (1991)
16. Eick, C.F., Vaezian, B., Jiang, D., Wang, J.: Discovery of interesting regions in spatial data sets using supervised clustering. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 127–138. Springer, Heidelberg (2006)
17. Eick, C.F., Zeidat, N., Zhao, Z.: Supervised clustering: Algorithms and application. In: International Conference on Tools with AI, Boca Raton, Florida, pp. 774–776 (2004)
18. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Kaufmann, M. (ed.) Proceedings of the 13th International Joint Conference on Artificial Intelligence, pp. 1022–1027 (1993)
19. Getis, A., Ord, J.K.: The analysis of spatial association by use of distance statistics. *Geographical Analysis* 24, 189–206 (1992)
20. Goodchild, M.F.: The fundamental laws of GIScience. In: Invited talk at University Consortium for Geographic Information Science, University of California, Santa Barbara (2003)
21. Hudak, P.F.: Arsenic, nitrate, chloride and bromide contamination in the gulf coast aquifer, south-central Texas, USA. *Intl. Journal of Environmental Studies* 60, 123–133 (2003)
22. Jiamthaphaksin, R., Eick, C.F., Lee, S.: Gac-geo: A generic agglomerative clustering framework for geo-referenced datasets. In: Knowledge and Information Systems (KAIS), pp. 1–29 (2011)
23. Karypis, G., Han, E.-H.S., Kumar, V.: Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer* 32(8), 68–75 (1999)
24. Koperski, K., Han, J.: Discovery of spatial association rules in geographic information databases. In: Egenhofer, M.J., Herring, J.R. (eds.) Proc. 4th Int. Symp. Advances in Spatial Databases, SSD, vol. 951, pp. 47–66 (6–9, 1995)
25. Kulldorff, M.: Prospective time periodic geographical disease surveillance using a scan statistic. *Journal Of The Royal Statistical Society Series A* 164, 61–72 (2001)
26. Lee, L.M., Herbert, B.: A GIS survey of arsenic and other trace metals in groundwater resources of Texas. In: Natural Arsenic in Groundwater: Science, Regulation, and Health Implications, Posters (2001)
27. Li, W., Han, J., Pei, J.: CMAR: Accurate and efficient classification based on multiple class-association rules. In: International Conference on Data Mining (ICDM 2001), San Jose, CA (November 2001)
28. Mennis, J., Liu, J.: Mining association rules in spatio-temporal data: an analysis of urban socioeconomic and sand cover change *Transactions in GIS* 9, 5–17 (2005)
29. Merriam-Webster Online Dictionary (2011), <http://www.merriam-webster.com>
30. Munro, R., Chawla, S., Sun, P.: Complex spatial relationships. In: The Third IEEE International Conference on Data Mining, ICDM (2003)
31. National Water-Quality Assessment Program, U.S. Department of the Interior and U.S. Geological Survey. Ground-Water Quality of the Southern High Plains Aquifer, Texas and New Mexico, Open-File Report 03-345 (2001)
32. Openshaw, S.: Two exploratory space-time attribute pattern analysers relevant to GIS. In: Fotheringham, S., Rogerson, P. (eds.) *Spatial Analysis and GIS*, London, pp. 83–104. Taylor and Francis, Abingdon (1994)
33. Openshaw, S.: Developing automated and smart spatial pattern exploration tools for geographical information systems applications. *The Statistician* 44(1), 3–16 (1995)

34. Openshaw, S.: Geographical data mining: Key design issues. *GeoComputation* (1999)
35. Ord, J.K., Getis, A.: Local spatial autocorrelation statistics: Distributional issues and an application. *Geographical Analysis* 27(4), 286–306 (1995)
36. Papadimitriou, S., Gionis, A., Tsaparas, P., Väistönen, A., Mannila, H., Faloutsos, C.: Parameter-free spatial data mining using MDL. In: 5th International Conference on Data Mining, ICDM (2005)
37. Parker, R.: Ground water discharge from mid-tertiary rhyolitic ash-rich sediments as the source of elevated arsenic in South Texas surface waters. In: Natural Arsenic in Groundwater: Science, Regulation, and Health Implications (2001)
38. Roddick, J.F., Spiliopoulou, M.: A bibliography of temporal, spatial and spatio-temporal data mining research. In: SIGKDD Explorations, vol. 1, pp. 34–38 (1999)
39. Sharma, L., Tiwary, U., Vyas, O.: An efficient approach to spatial association rule mining. In: Int. Conf. On ISPR IIIT, Allahabad, India, pp. 1–5 (2004)
40. Shekhar, S.: Spatial data mining: Accomplishments and research needs. Keynote Speech at GIScience 2004 (3rd Bi-Annual International Conference on Geographic Information Science) (2004)
41. Shekhar, S., Chawla, S.: Spatial Databases: A Tour. Prentice-Hall, Englewood Cliffs (2003) ISBN 013-017480-7
42. Smith, A., Hopenhayn-Rich, C.: Cancer risks from arsenic in drinking water. *Environmental Health Perspectives* 97, 259–267 (1992)
43. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Reading (2006)
44. Tay, S.C., Hsu, W., Lim, K.H.: Spatial data mining: Clustering of hot spots and pattern recognition. In: IEEE International Geoscience and Remote Sensing Symposium (2003)
45. Texas Water Development Board. (2011),
<http://www.twdb.state.tx.us/home/index.asp>
46. U.S. Environmental Protection Agency (2011), <http://www.epa.gov/>
47. Wang, W., Yang, J., Muntz, R.R.: STING: A statistical information grid approach to spatial data mining. In: Twenty-Third International Conference on Very Large Data Bases, Athens, Greece, pp. 186–195. Morgan Kaufmann, San Francisco (1997)
48. World Health Organization (2011), <http://www.who.int/>

Chapter 12

Learning from Imbalanced Data: Evaluation Matters

Troy Raeder¹, George Forman², and Nitesh V. Chawla¹

¹ University of Notre Dame, Notre Dame, IN, USA

² HP Labs, Palo Alto, CA, USA

`traeder@nd.edu, ghforman@hpl.hp.com, nchawla@nd.edu`

Abstract. Datasets having a highly imbalanced class distribution present a fundamental challenge in machine learning, not only for training a classifier, but also for evaluation. There are also several different evaluation measures used in the class imbalance literature, each with its own bias. Compounded with this, there are different cross-validation strategies. However, the behavior of different evaluation measures and their relative sensitivities—not only to the classifier but also to the sample size and the chosen cross-validation method—is not well understood. Papers generally choose one evaluation measure and show the dominance of one method over another. We posit that this common methodology is myopic, especially for imbalanced data. Another fundamental issue that is not sufficiently considered is the sensitivity of classifiers both to class imbalance as well as to having only a small *number* of samples of the minority class. We consider such questions in this paper.

1 Motivation and Significance

A dataset is imbalanced if the different categories of instances are not approximately equally represented. Recent years have brought increased interest in applying machine learning techniques to difficult “real-world” problems, many of which are characterized by imbalanced data. The imbalance can be an artifact of class distribution and/or different costs of errors or examples. With an increasing influx of applications of data mining, the pervasiveness of the issues of class imbalance is becoming only more profound. These applications include telecommunications management [13], text classification [15, 22], bioinformatics [25], medical data mining [26], direct marketing [11], and detection of oil spills in satellite images [18]. These applications not only present the challenge of high degrees of class imbalance (for instance, some have less than 0.5% positives), but also the problem of small sample sizes. We assume that the positive (more interesting) class is the minority class, and the negative class is the majority class.

Let us consider a couple of cases here to underline the extreme imbalance in real-world applications. The first example is from the public Reuters RCV1

dataset [19]. Figure 1 shows a histogram of the class distribution of 600+ classes identified in the dataset. The y-axis is the number of classes that belong to the histogram bin. The majority of classes occur less than 0.3%, and some of the classes have less than one part-per-ten-thousand in the dataset.

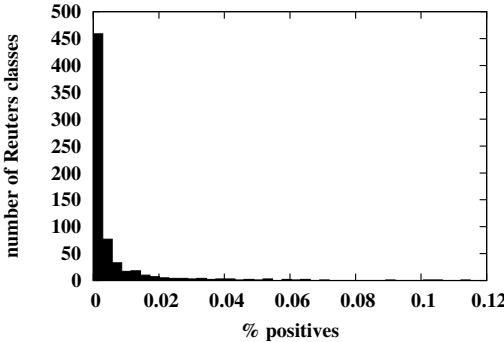


Fig. 1. Class Distribution of Reuters' Dataset

Another example is the detection of adverse drug events in a medical setting. It is extremely important to capture adverse drug events, but such events are often rare. The Institute of Medicine has encouraged incorporation of decision based tools to prevent medication errors. In our prior work, we considered prediction of such adverse drug events in Labor and Delivery [26]. The objective was to generate a classifier to identify ADE in women admitted for Labor and Delivery based on patient risk factors and comorbidities. The sample of 135,000 patients had only 0.34% instances marked as adverse drug events.

In the direct marketing domain, advertisers make money by identifying customers who will make purchases from unsolicited mailings. In this case the interesting class composes less than 1% of the population.

With a growing number of applications that are confounded by the problem of class imbalance, the question of evaluation methodology looms. We demonstrate that the choices in evaluation methodology matter substantially in order to raise the awareness to make these choices deliberately and (ideally) consistently among researchers, and to discuss frontiers of research directions.

Contribution. We address the following questions in this paper:

1. What is the effect of sample size versus class skew on the problems of learning from imbalanced data?
2. What effect does changing the class skew (making more imbalanced) have on the conclusions?
3. What is the sensitivity of validation strategies and evaluation measures to varying degrees of class imbalance?
4. Do different cross-validation strategies (10-fold or 5x2 [10]) affect the conclusions?

5. Do different evaluation measures lead us to different conclusions for the same classifiers on the same data sets?

We address the aforementioned issues by considering three different classifiers — Naive Bayes (NB), C4.5 (J48), and Support Vector Machines (SMO), and multiple datasets from a number of different domains and applications, including public data sets from UCI [3] and LIBSVM [4]. We consider both 10-fold and 5x2 cross-validation (CV) in the paper. Our evaluation methods comprise of AUC, F-measure, Precision @ Top 20, Brier score (quadratic loss on probabilistic predictions, indicative of classifier calibration), Accuracy, and the new H-measure proposed by David Hand [16]. We believe that a uniform comparison and benchmarking strategy can help innovation, achieving not only a theoretical impact but also a broad practical impact on a number of real-world domains.

2 Prior Work and Limitations

The major forefront of research in learning from imbalanced datasets has been the incorporation of sampling strategies with different learning algorithms. See the recent workshops and survey papers for a comprehensive discussion on different methods [2, 5, 23, 17, 21, 29]. Recent research has also focused on new or modified objective functions for SVMs or decision trees [8, 31, 1, 28].

We analyzed a number of papers published in the last few years on the topic of learning from class imbalance and find that researchers are very inconsistent in their choice of metric, cross-validation strategy, and benchmark datasets. Thus published studies are difficult to compare and leave fundamental questions unanswered. What is really the progress of our approach for imbalanced data? What area of the imbalanced problem space are we really addressing? What can we do as a community to ensure more real data is made available for researchers to collaborate and/or benchmark their methods on?

We give here a brief review of these recent papers on class imbalance. There is no agreement on the cross-validation strategies deployed — these range from 5x2 to 5-fold to 10-fold. Each of these can have an impact on the performance measurement, as these result in different numbers of instances in the training and testing sets. This is especially critical when there are few of the minority class instances in the dataset. The mix of performance measures is especially interesting — balanced accuracy, AUC, geometric mean, F-measure, precision, recall, and probabilistic loss measures. Particular methodologies have been shown to perform more optimally on a particular measure. The final straw man in the related work is the use of datasets. Two recent surveys on experimental comparisons of different sampling methods and classifiers (published in 2004 and 2007) have used different validation strategies (10-fold versus 5-fold), evaluation measures, and even disagreed on some of the important conclusions. Hulse et al. [17] had 8 out of 35 datasets between 1.3% and 5% of class skew. Batista et al [2] had even fewer datasets in that range, and its lowest class skew was 2.5%. Recent research on link prediction [20] has provided some insight into class skews

on the order of thousands or tens-of-thousands of negative examples per positive example, but these data sets are relatively rare.

Table 1. Data sets used in this study

No.	Dataset	Examples	Features	# MinClass	% MinClass
1	Boundary (Biology)	3,505	174	140	4%
2	Breast-W (UCI)	569	30	210	37%
3	Calmodoulin (Biology)	18,916	131	945	5%
4	Compustat (Finance)	10,358	20	414	4%
5	Covtype (UCI)	38,500	54	2,747	7.1%
6	E-State (Drug Discovery)	5,322	12	636	12%
7	FourClass (LIBSVM)	862	2	307	35.6%
8	German.Numer (LIBSVM)	1,000	24	300	30%
9	Letter (UCI)	20,000	16	789	3.9%
10	Mammography (Breast Cancer)	11,183	6	223	2.3%
11	Oil (Oil Spills))	937	49	41	4%
12	Page (UCI)	5,473	10	560	10%
13	Pendigits (UCI)	10,992	16	1142	10%
14	Phoneme (Elenna Project)	5,404	5	1584	29%
15	PhosS (Biology)	11,411	479	613	5%
16	Pima (UCI)	768	8	268	35%
17	Satimage (UCI)	6,435	36	625	9.7%
18	Segment (UCI)	2,310	19	330	14%
19	Splice (UCI)	1,000	60	483	48.3%
20	SVMGuide1 (LIBSVM)	3,089	4	1089	35%

While we also encounter a similar problem of limited availability of real-world datasets in this paper, we try to overcome this by artificially reducing the positive class to increase the class imbalance. We also consider a number of different real-world domains to allow for broader generalizations.

3 Experiments

We considered three different classifiers — Naive Bayes (NB), J48 (with Laplace smoothing at the leaves), and SMO using the Platt's calibration (-N 2 -M -V 2 options in WEKA). We used WEKA [27] v3.6 implementations of each to ensure repeatability. Again, our goal was not to research optimal methods of dealing with imbalance, but simply to have a set of common classifiers to illustrate the differences in evaluation methodologies and measures. Each classifier produced scores that were then plugged into a number of different measures. We used 5x2 CV and 10-fold CV. 5x2 CV performs traditional 2-fold cross-validation and repeats it with five different random splits the data; thus, each training and testing set comprises 50% of the original data. 10-fold CV splits the data into ten disjoint folds, with 90% of the data used for training (combination of 9 folds) and

10% of the data used for testing (10th fold). The folds were completely stratified, i.e. nearly the same number of positives appear in each fold; moreover, the same training and testing sets were used for each classifier to avoid any variability arising from different random seeds.

Evaluation Metrics We evaluate each classifier using a variety of measures, as indicated in the Introduction, representing the panoply appearing in recent imbalance papers. We define these measures after introducing our notation.

Assume that we are given a series of instances $x_i \in x$ and their true class labels $y_i \in y$. For two-class problems like the ones we deal with in this paper, $y_i \in \{0, 1\}$. Define the number of instances as n , the number of negative instances in the test set as n_0 , and the number of positive instances as n_1 . When classifying an instance x_i each classifier produces a *score* $f(x_i)$, such that instances with higher scores are deemed more likely to belong to the positive class. Many machine learning packages output scores scaled between 0 and 1, which can then be interpreted as a probability of belonging to the positive class. We assume that the cost of a misclassification error depends only on the class of the example and denote the cost of misclassifying a negative example as c_0 and the cost of misclassifying a positive example as c_1 . On the basis of these scores, we define the metrics used in the paper:

- **Accuracy:** The most basic performance measure, simply the percentage of test instances that the classifier has classified correctly. For the purposes of assigning classifications to instances, we use a threshold of 0.5. That is, instances with $f(x_i) < 0.5$ are classified as negative, and all other instances are classified as positive.
- **AUC:** AUC quantifies the quality of the scores $f(x_i)$ in terms of rank-order. AUC is usually calculated as the empirical probability that a randomly chosen positive instance is ranked above a randomly-chosen negative instance. That is: $AUC = \frac{1}{n_{0n_1}} \sum_{i|y_i=1} \sum_{j|y_j=0} I(f(x_i), f(x_j))$, where $I(x, y)$ takes on the value 1 if $f(x_i) > f(x_j)$, 1/2 if $f(x_i) = f(x_j)$ and 0 otherwise. AUC is often preferred over Accuracy for imbalanced datasets because it does not implicitly assume equal misclassification costs.
- **Brier Score:** The Brier score is the average quadratic loss on each instance in the test set: $S_{brier} = \frac{1}{n} \sum_i (f(x_i) - y_i)^2$. This quantifies the average deviation between predicted probabilities and their outcomes.
- **Precision @ Top 20:** The Precision @ Top 20 is simply the fraction of the top 20 instances (as ranked by $f(x_i)$) that are actually positive. It measures the ability of a classifier to accurately place positive instances in the most important positions, i.e. for information retrieval.
- **F-measure:** F-measure measures a classifier's effectiveness at both precision and recall. The measure we implement is known as the F_1 -Measure, which is simply the harmonic mean of precision and recall. Again, we use a threshold of 0.5 to distinguish between positive and negative instances.
- **H-Measure:** H-Measure [16] is a very recently developed threshold-varying evaluation metric that is designed to overcome an inherent inconsistency in

the AUC metric. H-measure calculates the *expected loss* of the classifier (as a proportion of the maximum possible loss) under a hypothetical probability distribution $u(c)$ of the class-skew ratio $c = \frac{c_0}{c_0+c_1}$. For the purposes of this paper, we use the *beta(2, 2)* distribution suggested by Hand [16] which is given by $u(c) = 6c(1 - c)$.

- **Precision-Recall Break-Even point:** A *precision-recall* (PR) curve [9] plots recall on the x-axis and precision on the y-axis as the classifier's decision threshold varies across all possible values. The precision-recall break-even point is calculated as the intersection point between the PR curve and the line $y = x$. In the event that multiple intersection points exist, the largest value is used.

The appropriateness of many of these measures has been hotly debated in the literature. Accuracy is generally regarded as a poor metric because it implicitly assumes equal misclassification costs, which is rarely true in general and never true for imbalanced problems. Additionally it requires the researcher to choose a decision threshold, often without knowledge of the domain [24]. AUC is very popular in applications involving imbalanced data, both because it does not require the choice of a decision threshold and because it is completely agnostic to class skew.

However, AUC is not without its detractors. Two of the most vocal criticisms of AUC are that it is *misleading* in cases of extreme class skew [9] and that it is an *inconsistent* measure of classification performance. We briefly address these points now, as they lead nicely into important points later in the paper. Both arguments, at their heart, deal with the relationship, or lack thereof, between AUC and actual misclassification cost.

Consider a simple test set with 9 negative examples and 1 positive example (9:1 class skew). If the examples, ranked by $f(x_i)$ have classes $\{0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\}$, then the classifier's AUC is 0.9, the precision at the optimal decision threshold is 0.5, and the misclassification cost at the optimal threshold is c_0 . A similar example can be concocted under 99:1 class skew. If ten negative examples are ranked above the single positive example, the AUC is still 0.9, but the optimal precision is 0.09, and the optimal misclassification cost is $9c_0$. Thus, two classifiers with identical AUC can incur vastly different misclassification costs, depending on the inherent difficulty of the problem under consideration. In other words, *there is no simple way to infer misclassification cost from AUC*.

Hand takes this argument one step further and shows that the actual relationship between AUC and misclassification cost is complicated and is equivalent to assuming a likelihood distribution over the possible cost ratios that depends on the classifiers being compared. Instead, he proposes to estimate misclassification cost by fixing a continuous distribution over the cost ratios and computing expected classification loss.

This is a reasonable approach except that accurate performance estimation then depends on the choice of probability distribution. In his paper, Hand proposes a Beta distribution given by $u(c) = 6c(1 - c)$. There are two potential problems with this choice. First, the distribution $u(c)$ has the greatest mass

near $c = 0.5$, the value that represents equal misclassification costs. Second, it is symmetric about this point, meaning that it actually assigns a likelihood of 0.5 to the possibility that the misclassification of minority class examples is *less* costly than the misclassification of majority class examples. As we will see later, this poses a problem under circumstances of extreme imbalance.

Brier score is unique among the metrics we consider in that it actually takes the magnitude of the score $f(x_i)$ into account. It seems most appropriate for situations (such as investment or betting, perhaps) where the action taken depends on the absolute confidence of the classifier in its prediction. If this information is irrelevant, and only the relative positions of the instances matter then Brier Score is an inappropriate metric, because it has a substantial impact on the rank-ordering of classifiers in our results.

3.1 Datasets

Table 1 summarizes the different datasets from different applications, and public sources such as the UCI [3] and LIBSVM [4]. Data is derived from biology [25], medicine [6, 26], finance [7], and intrusion detection. Some of these datasets were originally multi-class datasets and were converted into two class problems by keeping the smallest class in the data as minority and clumping the rest together as majority class.

The class imbalance varies from 2.3% to 48.3% (balanced). However, in our experiments we also reduced the number of minority class examples in the data, such that the class priors were artificially reduced to half of the original. That is, if the original data had 140 minority class instances, we reduced it by multiples of 5% until we had 70 (50%) minority class instances. This allowed us to consider the effect of sample size and high class skews in the experiments as well. We removed a maximum of 50% to be consistent across all the datasets; while some datasets could support further reduction, it would have severely impacted some of the datasets with few positives, such as Oil, which only has 41 examples to start with.

3.2 Empirical Analysis

We show aggregate results across all the datasets. Please note that the point here is not to compare classifiers or to state which classifier is most appropriate for a given dataset. Rather, the point is to see the sensitivity of classifiers and performance measures (and hence conclusions drawn) to different validation strategies and rates of class imbalance.

Figure 2 shows the different performance measures. The y-axis on the figure is the performance measure averaged over all datasets, and the x-axis is the increasing rate of imbalance. That is, the leftmost point (0) is the original dataset, and as we move along the x-axis, we remove x percent of the minority class. So, 10 represents removing 10% of the minority class examples prior to splitting for cross-validation.

Some interesting trends emerge from these results. Let us first consider Figure 2(a) for AUC. For each of the three classifiers, the AUC consistently drops as the imbalance increases. However, the AUC does not change nearly as much as one might expect (compare the y-axis range with the wide range of almost every other graph). This illustrates a weakness of AUC, which was pointed out by Hand [16]: the measurement of AUC depends on the relative score distributions of the positives and the negatives, which essentially depends on the classifier itself. It is independent of class priors; it is measuring only the quality of rank-order. In the absence of true costs of misclassification, AUC is relying on score distributions, which are not shifting significantly, since the feature distribution $p(x)$ for the classifier is a random subset of the original data. The change in class skew toward high imbalance is not having a significant effect. Furthermore, we see that when using 5x2, NB is the best classifier, whereas this is not observed with 10-fold. Thus, if one were to use 5x2 CV in a paper, NB may emerge as a winner, while another paper using 10-fold may discover a tie between J48 and NB. The question then is, *which one to believe?*

Figure 2(b) shows the performance with H-measure, as proposed by David Hand [16]. Hand argues the limitations of using AUC for comparing classifiers — each classifier is calibrated differently, and thus produces different score distributions. It implies that AUC is evaluating a classifier conditioned on the classifier itself, thereby resulting in different “metrics” for comparing classifiers. To that end, he proposes the H-measure, which is independent of the score distributions. It is not independent of the class priors and is sensitive to the class skew, as one would expect. This is a necessary property as the misclassification costs are related to class priors. As we shift the minority class instances to be more skewed, the class priors are changing and the evaluation measures will shift. The H-measure declines with the increasing class skew and also demonstrates a higher variance than AUC for the same classifier over different rates of class imbalance. It is also more sensitive to the size of training and testing sets, as compared to AUC.

Figure 2(c) shows the result with F-measure. Both 10-fold and 5x2 are indistinguishable in this case. The F-measure is computed by thresholding at 0.5, and then calculating the TP, FP, TN, and FN. It is simply a function of those quantities at a fixed threshold. F-measure is also very sensitive to imbalance and rapidly drops, which is not surprising as both precision and recall will deteriorate. We found F-measure exhibited a greater variance as compared to AUC.

Figure 2(d) shows Precision @ Top 20. Again the performance generally drops across imbalance. As the class imbalance increases, the expectation of a minority class example to be in the Top 20 of the probability scores (ranks) drops. Hence, the relative precision drops as the imbalance increases. There is no thresholding done, and the performance is reflective of ranking, such as one may desire in most information retrieval tasks where high recall is not essential. Furthermore, observe that with 10-fold cross-validation J48 dominated, whereas with 5x2 cross-validation the NB classifier dominated.

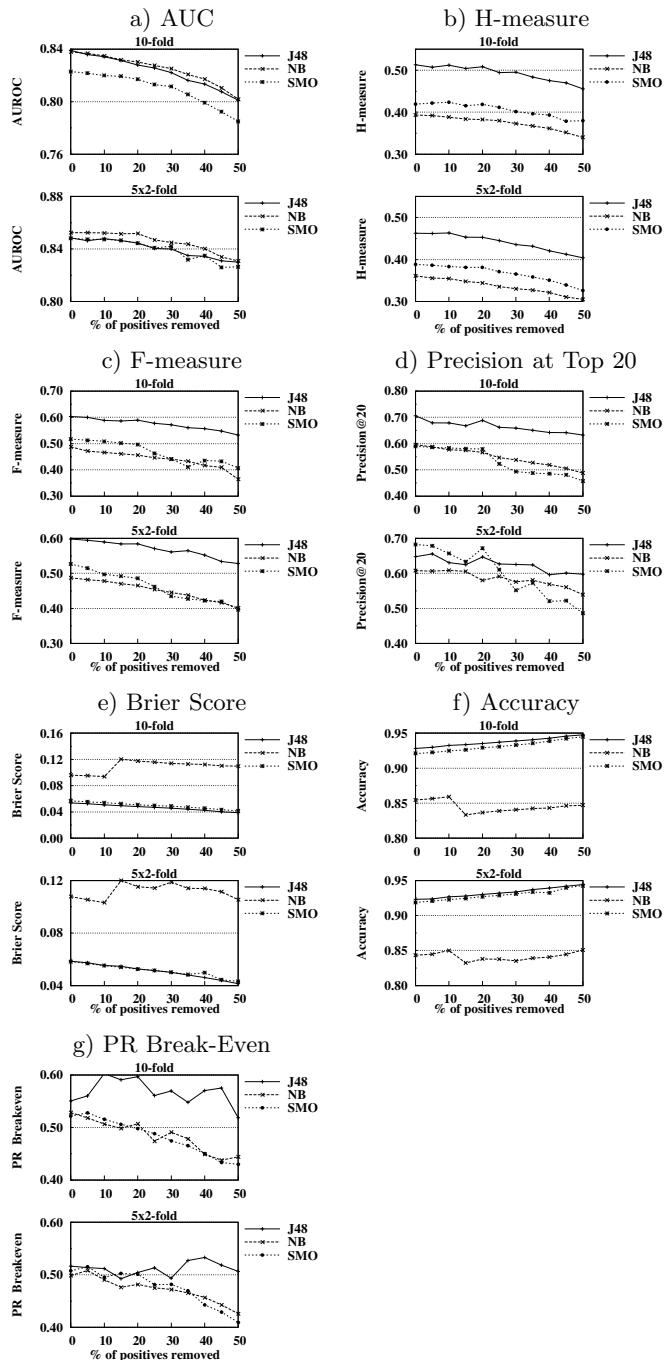


Fig. 2. Performance trends at increasing levels of class imbalance

Figure 2(e) shows the result on Brier score. As a loss measure, lower loss is better. For J48 and SMO, as imbalance increases the loss decreases, which is expected given that fewer of the positive class examples are contributing to the loss function. Since there are more negative class examples, the model is calibrated better towards predicting the negative class. NB is different from the two classifiers. The blip in NB performance at 40% appears to be a random event: the high imbalance caused performance to degrade severely on the compustat dataset, which captures the rating of companies based on their financial parameters for three different years. However, the general trend of Naive Bayes corroborates the previous observations of Domingos & Pazzani [12] and Zadrozny & Elkan [30]. They have noted that Naive Bayes gives inaccurate probability estimates (but can still give good rank-ordering). Naive Bayes tends to give more extreme values, and with the shrinking minority class examples, the classifiers are becoming worse in their calibration. Since this does not affect its ability for rank-ordering, this phenomenon was not observed with AUC.

For completeness, we also included the accuracy Figure 2(f), even though it is accepted to be a weak metric for imbalanced datasets. As expected, accuracy increases with imbalance — a classifier becomes increasingly confident on the majority class. Hence, accuracy is not a useful metric for class imbalance research.

Finally, Figure 2(g) shows results for the break-even point of precision and recall. The most striking aspect of this graph is the instability of the metric under increasing imbalance for the J48 classifier. While NB and SMO generally decline in performance as the difficulty of the classification task increases, J48’s performance is tremendously erratic, especially under 10-fold cross-validation. This variability serves to illustrate an important point: while performance under two-fold cross-validation may suffer from a lack of positive *training* examples, the lack of positive *test* examples in CV folds can make estimation under extreme imbalance problematic. J48 is unique in that it generally provides very coarse-grained probability estimates (based on class membership at the leaves). One result of this is that large blocks of test examples can be given the same probability estimate. As a result, small changes in classifier probability estimates can result in very large changes in the rank-ordering of positive examples. If there are few test examples, this will have a profound effect on the final performance estimate.

Summary. The results generally show that (1) greater class imbalance leads to a decay of the evaluation measure (except for accuracy), and more importantly, (2) the choice of evaluation methodology can have a substantial effect on which classifier methods are considered best. The three classifiers were ranked differently by the different evaluation measures. For example, Naive Bayes performed terribly for Brier score, and yet its rankings with respect to AUC were the best. This result underscores the importance of choosing a metric which is appropriate for the final application of the classifier. Moreover, in some cases the cross-validation strategy also has a large effect on the conclusions, especially in the case of Precision @ Top 20. With more classifiers being evaluated in a real study, the inconsistent results would multiply.

In our results, we observe several differences evaluation metrics and cross-validation methods. F-measure was more favorable to J48 versus NB or SMO. On the other hand, AUC generally found J48 and NB competitive, with a slight bias towards NB under 5x2 cross-validation. The H-measure strongly favors J48 and not so SMO and NB. Precision @ Top 20 yields a clear winner with no ties, but that winner depends on which form of cross-validation is used (J48 for 10-fold and NB for 5x2-fold). If we compare based on Brier score, NB emerged as the weakest classifier, with no clear distinction between J48 and SMO, which is not surprising given the poor calibration of NB. Finally, if we look at Precision @ Top 20, NB again was the weakest classifier, with no significant differences between SMO and J48.

These results are clear evidence that different validation methods and performance measures can result in potentially different conclusions. These variations in classifier ranking show that it is important for the community to evaluate classifiers in the light of different metrics and to be very careful when stating conclusions that may not deserve much generalization.

4 Discussion and Recommendations

We conclude with some general recommendations in the light of the results, related research, and make a call to the community for research directions, problems and questions as we strive to handle greater degrees of class imbalance.

4.1 Comparisons of Classifiers

It is evident from Figure 2 that, depending on the measure and/or the mode of validation, one can arrive at a fundamentally different conclusion about the tested classifiers. The scenario of selecting a single ‘best’ classifier that performs well on one chosen measure makes complete sense for more focused application settings where an optimal performance objective has been determined. But it becomes myopic or misleading for general research papers comparing methods.

Comparing the different measures sheds an interesting light. As an example, let us consider SMO at 10-fold. While it has a competitive performance in AUC, its Precision @ Top 20 suffers at a high class imbalance. This also demonstrates a potential weakness of AUC as it is looking at the entire curve. The classifier is not able to achieve a relatively higher precision in the beginning of the curve, but potentially recovers the performance along the curve, leading to a higher AUC. Now a practitioner may only be interested in the power of a classifier in ranking correct positive class predictions over the negative class, without an explicit threshold. A high AUC in this case can be misleading. A similar comparison can be drawn between Precision @ 20 versus H-measure. H-measure puts NB as the worst classifier for 5x2 but Precision @ 20 puts it as the best classifier. Such differences in the ranking of the classifiers bring out a compelling point — different classifiers have different optimal operating regions in the trade-off between the two types of errors for imbalanced data. Looking at a single metric

without attention to how the classifier may be used or even the property of the data (degree of class imbalance, sample size, etc) may bring one to incorrect conclusions.

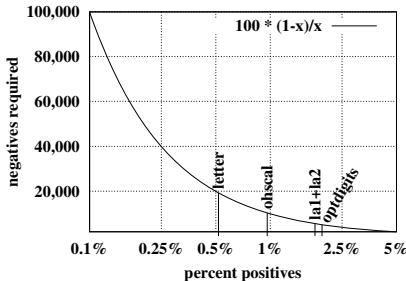


Fig. 3. The minimum number of negative cases required in a dataset in order to do research with $x\%$ positives, with a minimum of 100 positives

Effect of Sample Size. As we observed in the previous section, the limited sample size of the positive class mitigates careful experimentation and generalized conclusions. As the research community studies greater degrees of imbalance, we will need larger public benchmark datasets. How large should the datasets be? Clearly there needs to be some minimum number of positive cases in the dataset, which we discuss further in the next section. Suppose one decides that 100 positive examples are sufficient for some learning task and that they would like to perform imbalance research up to, say, 0.25% positives. Then 39,900 negative examples will be needed. Even our largest text and UCI datasets do not have anywhere near this number of negatives. Figure 3 shows the number of negatives needed for a variety of imbalance goals down to one part-per-thousand, assuming a minimum of 100 positives, which is probably a bare minimum. The figure also marks for each of our larger datasets the greatest imbalance that it can support. Keep in mind that this curve represents a lower bound. The minimum requirement on positives may need to be increased—with a proportional increase in the demand for negatives.

Sample Size and Evaluation. Consider a data set with fewer than 50 positive examples. If we do a 10-fold CV, then the number of positive training items in each fold will be no more than 45, and the testing positives will be less than or equal to 5. While this gives a reasonable (relative) sample for training, the testing set is very small, which could lead us to arrive at potentially unreliable performance estimates. The extreme scenario for 10-fold cross-validation is that there are some folds that have no positive class instances. If we do a 5x2 fold, then it would give us about 25 positive examples in training and testing. This is a much smaller size for training and will now actually effect the model calibration. By using just 50% of the dataset for training, we are indirectly preferring classifier models that can learn well from smaller samples — a perhaps unintended

consequence of a methodology choice that may have little bearing for research with more balanced class distributions.

The small sample size issue is clearly confounded by the need for **internal cross-validation** to allow learning methods that perform some sort of self-calibration or parameter tuning, such as the well known Platt scaling post-processing phase for SVM, or the selection of its complexity parameter C via internal cross-validation. This internal validation becomes tricky and questionable, as the number of instances per fold are even smaller. Can the parameters then be trusted?

Null Hypothesis. If we have very few positives, not only may we be unable to determine the best method, in addition there is the possibility that we may mistake *worthless* methods for good ones. One might not think this would be a concern, but it happened in the thrombin task of the 2001 KDD Cup. The score of the winning entry achieved 0.68 AUC, and with 634 test cases, people generally believed that the test set was big enough to yield valid results. But it turned out that if each of the 117 contestants were to have submitted completely random classifiers, the *expected value* for the highest score would be slightly higher [14].

But the lesson holds especially for researchers of class imbalance. If we have small a number of positives, the possibility of getting large performance scores under the null hypothesis is remarkably high. For example, supposing we have 50 positives and 1000 negatives, the AUC critical value that must be exceeded is 0.654 in order to limit the probability to $p=0.01$ that our best method's score could be due only to chance—alarmingly high [15].

Test Variance. Even supposing that our methods perform well above the critical value for random classifiers, just having fewer positives in the test set leads to higher variance for most performance measurements, except accuracy or error rate. Greater variance in our test results makes it more difficult to draw research conclusions that pass traditional significance tests, such as the paired t-test or Wilcoxon rank tests.

We illustrate this point with AUC, since its known insensitivity to the testing class distribution is sometimes incorrectly taken to mean that it is acceptable to measure AUC with very few positives. We simulated a fixed classifier on various test sets, varying the number of positives and negatives. As expected, the mean AUC averaged over millions of trials was always the same, regardless of the test set. But the variance tells another story. For example, a fixed classifier that achieved mean 0.95 AUC on all test sets had the following standard deviation: 0.010 for 100:5000 positives to negatives, 0.011 for 100:500, and 0.032 for 10:500. To interpret this, the standard deviation changed little (+9%) for a shift in the class distribution from 100:5000 to 100:500, but changed a lot (+320%) when the class distribution was preserved but the number of test items was decreased in size from 100:5000 to 10:500. Furthermore, when we reduce only the positives for a large test set of 10:5000, we still get high variance (+314% of that of 100:5000). The upshot of this demonstration is that we need to have

large numbers of positives in our test sets, in order to keep the variance of our test measurement low.

4.2 Towards Parts-Per-Million

Suppose ambitious researchers extend their goal to one part-per-million, 0.0001%: then 9,999,900 negatives would theoretically be needed to balance 100 positives. Such demands for labeled data are unreasonable. Not only does the effort to label by random sampling grow linearly with the total size of the dataset, it would likely also suffer from class noise that well exceeds one part-per-million. And once the price is paid to obtain all these labeled negatives, what is to be done with them? One of the most successful techniques for dealing with class imbalance is simply to discard many negatives from the training set, whether by random sampling or more involved methods. Paying a large cost to obtain a huge dataset and then throwing away a large fraction of it is somewhat nonsensical.

Thus, it appears that research under very high class imbalance cannot expect to receive randomly sampled datasets. Instead, the datasets will consist of a small number of selected positives, and a mostly unverified supply of background cases, for which the prevalence of positives is expected to be low, but non-zero. This approach has been adopted by the Information Retrieval community since the 1970's. To cope with the problem, they have developed the idea of pooled judging: all cases predicted to be positive by any of the competing methods are pooled into a union set, and then domain experts laboriously check each one to determine its ground-truth label. Once this judging is completed, one can finally score the individual methods based on their *true positives* and *false positives*, yielding F-measure, Precision, Recall, Precision@20, etc. Since no judgment is made on the majority of cases that were not retrieved by any of the methods, one cannot know the true recall, accuracy or AUC (such measures are sometimes reported by making the bold and unjustified assumption that there are no other positives).

Further complicating matters, the training positives are unlikely to be a random sample from the source distribution of positives. There is a ubiquitous assumption in machine learning that the training set is a random sample from the test distribution. But for very high class imbalance, it is unreasonable to expect a person to identify the requisite number of positives by random sampling. Instead, they will probably find positive examples via search. If they search with a single keyword query and already obtain the desired minimum number of positives, they will have appeared to meet the needs of building a labeled dataset for study. But such positives would have very low diversity. In fact, the learning problem is then simply reduced to trying to figure out the keyword query that was used (likely just a few words). Not what is intended. But such cases have occurred in practice.

4.3 Recommendations

Based on our observations and preceding discussions, we make some recommendations for the research community and practitioners who are focused on high class imbalance. While our experiments do not exhaust all possible classifiers, datasets and performance measures, they do shed light on the trends of classifiers' performances under different scenarios. We characterize the evaluation of classifiers under high imbalance as follows. Foremost, if a specific classification threshold or misclassification costs are known, then naturally one should rely on the domain expertise and target the study towards the tuned costs. Otherwise, when working in a domain confounded by high imbalance and small sample size (the parts-per-million conundrum), then we argue that of all the measures we studied, it is most appropriate to use a Precision @ top N measure. The specific operating region at very conservative decision thresholds then becomes critical.

For H-Measure, we can conclude that it is not heavily influenced by the performance of the classifier at the top of the operating range. Indeed, observe in Figure 2(d) that Naive Bayes has good performance with respect to Precision @ Top 20, and yet in Figure 2(b) we see that Naive Bayes receives the consistently worst H-measure scores (and well as Brier scores). Based on this, we conclude that H-measure, although perhaps appropriate for more balanced situations, is not a good candidate for judging performance at highly conservative thresholds (low false positive rate, high precision, low recall).

It is worth noting that the rank-order induced by H-measure is at all points equivalent to the rank-order induced by Accuracy. Recall that the computation of H-Measure is biased toward the notion that misclassification costs are approximately equal. At high levels of imbalance this is increasingly unlikely to be true, and this result suggests that the standard computation of H-Measure may as inappropriate as Accuracy for performance comparisons on highly imbalanced data sets. A principled study of the effect of the chosen cost-likelihood distribution on final performance estimates would make interesting future work.

One caveat with Precision @ Top N is in selecting the value of N. While in many studies a value of 20 or 100 is sufficient, it would be useful to use a common threshold so that different studies may be compared. That said, if a single N is chosen, it may be inappropriate for a study in which many more than N positives are available; the classifiers may easily fill the first N positions with only positives, and then the Precision @ Top N may not differentiate among classifiers. For instance, consider the case of 10,000 positives out of 1 million instances, which still gives a high skew of 0.01, but top 20 or top 100 may not differentiate classifiers.

5 Summary

To summarize, many of the general lessons of machine learning are amplified in research under high class imbalance. We need imbalanced datasets with very many more cases than is typically available in existing public datasets. As a

community we need to converge on a validation framework with a set of evaluation metrics that is used consistently throughout. In particular, the evaluation metrics chosen need to be suitable for the problems being analyzed. To draw statistically valid conclusions and avoid overfitting, the datasets must not have too few positives, and they need to have very few labeling errors, especially in the negative/majority class. When the number of positives available is limited, the choice of 10-fold or 5x2 cross-validation can substantially affect the training sets available to the classifiers; 5x2 may penalize classifiers that have difficulty training on small samples of positives. And finally, as we move toward parts-per-million, the growing need for randomly sampled data is clearly unworkable, and the nature of the research must deal with such issues as training with positives only, leveraging a large unlabeled background dataset that may contain some positives, and perhaps Information Retrieval methods for measuring performance, where we must postpone scoring a classifier until its positive predictions have been examined by a judge.

Acknowledgements. This research was supported in part by the NSF grant EECS-0926170.

References

- [1] Akbani, R., Kwek, S.S., Japkowicz, N.: Applying support vector machines to imbalanced datasets. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 39–50. Springer, Heidelberg (2004)
- [2] Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explorations 6(1) (2004)
- [3] Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
- [4] Chang, C., Lin, C.: Libsvm data sets,
<http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets>
- [5] Chawla, N.V., Japkowicz, N., Kotcz, A.: Editorial: special issue on learning from imbalanced data sets. ACM SIGKDD Explorations Newsletter 6(1), 1–6 (2004)
- [6] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic Minority Oversampling TEchnique. JAIR 16, 321–357 (2002)
- [7] Chawla, N.V., Cieslak, D., Hall, L.O., Joshi, A.: Automatically Countering Imbalance and Its Empirical Relationship to Cost. In: DMKD (2009)
- [8] Cieslak, D.A., Chawla, N.V.: Learning decision trees on unbalanced data. In: ECML (2008)
- [9] Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: Proceedings of the 23rd International Conference on Machine learning, p. 240. ACM, New York (2006)
- [10] Demesar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. JMLR 7, 1–30 (2006)
- [11] Direct Marketing Association. The dmef data set library,
<http://www.directworks.org/Educators/Default.aspx?id=632>
- [12] Domingos, P., Pazzani, M.J.: Beyond independence: Conditions for the optimality of the simple bayesian classifier. In: ICML (1996)

- [13] Ezawa, K.J., Singh, M., Norton, S.W.: Learning Goal Oriented Bayesian Networks for Risk Management. In: ICML, pp. 139–147 (1996)
- [14] Forman, G.: A method for discovering the insignificance of one's best classifier and the unlearnability of a classification task. In: Data Mining Lessons Learned Workshop, ICML (2002)
- [15] Forman, G., Cohen, I.: Beware the null hypothesis: Critical value tables for evaluating classifiers. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720, pp. 133–145. Springer, Heidelberg (2005)
- [16] Hand, D.J.: Measuring classifier performance: a coherent alternative to the area under the ROC curve. Machine Learning 77(1), 103–123 (2009)
- [17] Hulse, J.V., Khoshgoftaar, T.M., Napolitano, A.: Experimental perspectives on learning from imbalanced data. In: Ghahramani, Z. (ed.) ICML, pp. 935–942. ACM, New York (2007)
- [18] Kubat, M., Holte, R., Matwin, S.: Machine Learning for the Detection of Oil Spills in Satellite Radar Images. Machine Learning 30, 195–215 (1998)
- [19] Lewis, D.D., Yang, Y., Rose, T., Li, F.: RCV1: A new benchmark collection for text categorization research. Journal of Machine Learning Research 5, 361–397 (2004)
- [20] Lichtenwalter, R., Lussier, J., Chawla, N.: New Perspectives and Methods in Link Prediction. In: Proceedings of KDD
- [21] Mease, D., Wyner, A.J., Buja, A.: Boosted classification trees and class probability/quantile estimation. Journal of Machine Learning Research 8(3), 557–562 (2007)
- [22] Mladenić, D., Grobelnik, M.: Feature Selection for Unbalanced Class Distribution and Naive Bayes. In: Proceedings of the 16th International Conference on Machine Learning, pp. 258–267 (1999)
- [23] Chawla, N.V., Japkowicz, N., Kolcz, A.: Proceedings of the ICML Workshop on Learning from Imbalanced Data Sets (August 2003)
- [24] Provost, F., Fawcett, T., Kohavi, R.: The case against accuracy estimation for comparing induction algorithms. In: Proceedings of the Fifteenth International Conference on Machine Learning, Citeseer, vol. 445 (1998)
- [25] Radivojac, P., Chawla, N.V., Dunker, K., Obradovic, Z.: Classification and Knowledge Discovery in Protein Databases. JBI 37(4), 224–239 (2004)
- [26] Tafts, L.M., et al.: Countering imbalanced datasets to improve adverse drug event predictive models in labor and delivery. JBI (2009)
- [27] Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
- [28] Wu, G., Chang, E.Y.: Kba: Kernel boundary alignment considering imbalanced data distribution. IEEE TKDE 17(6), 786–795 (2005)
- [29] Wu, J., Xiong, H., Wu, P., Chen, J.: Local Decomposition for Rare Class Analysis. In: Proceedings of KDD, pp. 814–823 (2007)
- [30] Zadrozny, B., Elkan, C.: Learning and making decisions when costs and probabilities are both unknown. In: Proceedings KDD (2001)
- [31] Zhou, Z., Liu, X.: Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE TKDE 18(1), 63–77 (2006)

Author Index

- Antonucci, A. 49
Avros, R. 131
Boinski, Pawel 223
Cao, Tru H. 267
Chau, Cuong K. 267
Chawla, Nitesh V. 315
Contreras, Pedro 95
Corani, G. 49
D'Ambrosi, Leonardo 209
Ding, Wei 289
Eick, Christoph F. 289
Forman, George 315
Ghosh, Joydeep 157
Granichin, O. 131
Gupta, Gunjan 157
Holmes, Dawn E. 1
Jain, Lakhmi C. 1
Liu, Ling 7
Meo, Rosa 209
Murtagh, Fionn 95
Raeder, Troy 315
Shalymov, D. 131
Tang, Thao M. 267
Tweedale, Jeffrey 1
Volkovich, Z. 131
Weber, G.-W. 131
Wojciechowski, Marek 223
Yoo, Jin Soung 29
Zaffalon, M. 49
Zakrzewicz, Maciej 223
Zhou, Yang 7