

Wray Buntine
Marko Grobelnik
Dunja Mladenić
John Shawe-Taylor (Eds.)

LNAI 5782

Machine Learning and Knowledge Discovery in Databases

European Conference, ECML PKDD 2009
Bled, Slovenia, September 2009
Proceedings, Part II

2
Part II

 Springer



Lecture Notes in Artificial Intelligence 5782

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Wray Buntine Marko Grobelnik
Dunja Mladenić John Shawe-Taylor (Eds.)

Machine Learning and Knowledge Discovery in Databases

European Conference, ECML PKDD 2009
Bled, Slovenia, September 7-11, 2009
Proceedings, Part II

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Wray Buntine
NICTA
Locked Bag 8001, Canberra, 2601, Australia
and
Helsinki Institute of IT
Finland
E-mail: wray.buntine@nicta.com.au

Marko Grobelnik
Dunja Mladenić
Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
E-mail: {marko.grobelnik,dunja.mladenic}@ijs.si

John Shawe-Taylor
University College London
Gower St., London, WC1E 6BT, UK
E-mail: jst@cs.ucl.ac.uk

Library of Congress Control Number: 2009933615

CR Subject Classification (1998): I.2, H.2.8, H.3, G.3, H.5, G.2, I.7

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-642-04173-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-04173-0 Springer Berlin Heidelberg New York

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

springer.com

© Springer-Verlag Berlin Heidelberg 2009
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12754656 06/3180 5 4 3 2 1 0

Preface

The year 2008 was the first year that the previously separate European Conferences on Machine Learning (ECML) and the Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD) were merged into a unified event. This is a natural evolution after eight consecutive years of their being collocated after the first joint conference in Freiburg in 2001. The European Conference on Machine Learning (ECML) traces its origins to 1986, when the first European Working Session on Learning was held in Orsay, France followed by the second European Working Session on Learning held in Bled, the location of this year's ECML PKDD 2009 conference. The European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD) was first held in 1997 in Trondheim, Norway. Over the years, the ECML/PKDD series has evolved into one of the largest and most selective international conferences in machine learning and data mining, the only one that provides a common forum for the two closely related fields. In 2009, ECML PKDD conference was held during September 7–11 in Bled, Slovenia.

The conference used a hierarchical reviewing process. We nominated 26 Area Chairs, each of them responsible for one sub-field or several closely related research topics. Suitable areas were selected on the basis of the submission statistics for ECML PKDD 2008 and from last year's International Conference on Machine Learning (ICML 2008) and International Conference on Knowledge Discovery and Data Mining (KDD 2008) to ensure a proper load balance among the Area Chairs. A joint Program Committee (PC) was nominated consisting of some 300 renowned researchers, mostly proposed by the Area Chairs. In order to make best use of the reviewing capabilities we initially only requested that two reviews be sought. However, in the event of an inconsistency between the two assessments a third review was requested. Papers receiving two very positive reviews were considered for inclusion in the two special issues of *Machine Learning* and *Data Mining and Knowledge Discovery* appearing in time for the conference. A further review was also sought for these papers in order to assess their suitability to appear in journal form. Aleksander Kolcz was the Best Papers Chair responsible for overseeing the selection of papers for these special issues.

ECML PKDD 2009 received 679 abstract submissions resulting in a final total of 422 papers that were submitted and not withdrawn during the reviewing process. Based on the reviews, and on discussions among the reviewers, the Area Chairs provided a recommendation for each paper with a ranking of the borderline papers. The three Program Chairs made the final program decisions after merging the opinions of the 26 Area Chairs.

All accepted papers were of equal status with an oral presentation, poster presentation and 16 pages in the proceedings, with the exception of those accepted for the special issues of journals that were only allocated a single page abstract

in the proceedings. We have selected a total of 106 papers of which 14 were equally divided between the two special issues. The acceptance rate for all papers is therefore 25%, in line with the high-quality standards of the conference series. It is inevitable with such a low acceptance rate that some good papers were rejected and we hope that authors of these papers were not discouraged by their disappointment. We are, however, confident that the accepted papers are of a high quality, making a very exciting and stimulating conference. In addition to research papers, 15 demo papers were accepted, each having 4 pages in the proceedings and demo of the system during the poster session. In addition to the paper and poster/demo sessions, ECML PKDD 2009 also featured five invited talks, ten workshops, six tutorials, and the ECML PKDD discovery challenge and industrial track. The selection of Invited Speakers covered a broad range from theoretical to leading application-orientated research. Together they made a very strong addition to the conference program. We are grateful to Shai Ben-David (University of Waterloo, Canada), Nello Cristianini (University of Bristol, UK), Mark Greaves (Vulcan Inc.), Rosie Jones (Yahoo! Research), Ralf Steinberger (European Commission - Joint Research Centre) for their participation in ECML PKDD 2009. The abstracts of their presentations are included in this volume.

This year we continued to promote an Industrial Track chaired by Marko Grobelnik (Jožef Stefan Institute, Slovenia) and Nataša Milić-Frayling (Microsoft Research, Cambridge, UK) consisting of selected talks with a strong industrial component presenting research from the area covered by the ECML PKDD conference. We have also included a Demonstration Track chaired by Alejandro Jaimés Larrarte, providing a venue for exciting exemplars of applications of novel technologies.

As in recent years, the conference proceedings were available on-line to conference participants during the conference. We are grateful to Springer for accommodating this access channel for the proceedings.

As in previous years we will continue with the recently established tradition of videorecording the event, ensuring an enduring record of the event made accessible at <http://videlectures.net/>. Mitja Jermol is the Video Chair overseeing this aspect of the organization.

This year's Discovery Challenge was coordinated by Andreas Hotho together with Folke Eisterlehner and Robert Jäschke. It involved three tasks in the area of tag recommendation.

We are all indebted to the Area Chairs, Program Committee members and external reviewers for their commitment and hard work that resulted in a rich but selective scientific program for ECML PKDD 2009. We are particularly grateful to those reviewers who helped with additional reviews at a very short notice to assist us in a small number of difficult decisions. We further thank the Workshop and Tutorial Chairs Ravid Ghani and Cédric Archambeau for selecting and coordinating the ten workshops and six tutorials that accompany the conference; the workshop organizers, tutorial presenters, and the organizers of the discovery challenge, the Industrial and Demonstration Tracks; the Video Chair;

the Publicity Chair David Hardoon; and Richard van de Stadt and CyberChair-PRO for highly competent and flexible support when confronted by novel features in our handling of the papers. Special thanks are due to the Local Chair, Tina Anžič, for the many hours spent ensuring the success of the conference. Finally, we are grateful to the Steering Committee and the ECML PKDD community that entrusted us with the organization of the ECML PKDD 2009.

Most of all, however, we would like to thank all the authors who trusted us with their submissions, thereby contributing to the main yearly European-focussed international event in the life of our expanding research community.

June 2009

Dunja Mladenić
Wray Buntine
Marko Grobelnik
John Shawe-Taylor

Best Paper Chair

Aleksander Kolcz Microsoft Live Labs, USA

Publicity Chair

David Hardoon University College London, UK

Video Chair

Mitja Jermol Jožef Stefan Institute, Slovenia

Steering Committee

Walter Daelemans	Bart Goethals
Katharina Morik	Johannes Fürnkranz
Joost N. Kok	Stan Matwin
Dunja Mladeníc	Tobias Scheffer
Andrzej Skowron	Myra Spiliopoulou

Area Chairs

Francis Bach	Hendrik Blockeel
Francesco Bonchi	Pavel Brazdil
Toon Calders	Nitesh Chawla
Walter Daelemans	Tijl De Bie
Johannes Fürnkranz	Thomas Gärtner
João Gama	Bart Goethals
Eamonn Keogh	Joost Kok
Alek Kolcz	Jure Leskovec
Stan Matwin	Taneli Mielikainen
Claire Nedellec	Martin Scholz
David Silver	Steffen Staab
Gerd Stumme	Luis Torgo
Michael Witbrock	Stefan Wrobel

Program Committee

Ameen Abu-Hanna
Osman Abul
Lada Adamic
Abdullah Al Mueen
Enrique Alfonseca
Erick Alphonse
Carlos Alzate
Massih-Reza Amini
Gennady Andrienko
Annalisa Appice
Hiroki Arimura
Andrew Arnold
Sitaram Asur
Martin Atzmueller
Nathalie Aussenac-Gilles
Paulo Azevedo
Lars Backstrom
Tony Bagnall
Roberto Basili
Vladimir Batagelj
Ron Bekkerman
Marc Bellemare
Paul Bennett
Bettina Berendt
Tanya Berger-Wolf
Michael Berthold
Sourangshu Bhattacharya
Concha Bielza
Misha Bilenko
Stephan Bloehdorn
Christian Bockermann
Mario Boley
Christian Borgelt
Karsten Borgwardt
Henrik Bostrom
Guillaume Bouchard
Jean-François Boulicaut
Janez Brank
Ulf Brefeld
Bjorn Bringmann
Paul Buitelaar
Rui Camacho
Stephane Canu
Olivier Cappe
Andre Carvalho
Carlos Castillo
Ciro Cattuto
Vineet Chaoji
Sanjay Chawla
David Cieslak
Philipp Cimiano
Lucchese Claudio
Vincent Claveau
Fabrice Colas
Antoine Cornuejols
Christophe Costa Florencio
Fabrizio Costa
Bruno Cremilleux
Padraig Cunningham
Alfredo Cuzzocrea
Florence D'Alche-Buc
Claudia d'Amato
Gautam Das
Kamalika Das
Jesse Davis
Alneu de Andrade Lopes
Jeroen de Bruin
Marco de Gemmis
Jeroen De Knijf
Thomas Degrís-Dard
Jose del Campo-Avila
Krzysztof Dembczynski
Laura Dietz
Carlos Diuk
Kurt Driessens
Pierre Dupont
Jennifer Dy
Saso Dzeroski
Charles Elkan
Tapio Elomaa
Damien Ernst
Floriana Esposito
Fazel Famili
Nicola Fanizzi
Amir-massoud Farahmand
Ad Feelders

Xiaoli Fern
Daan Fierens
Ilias Flaounas
George Forman
Blaz Fortuna
Eibe Frank
Jordan Frank
Mohamed Gaber
Dragan Gamberger
Gemma Garriga
Gilles Gasso
Eric Gaussier
Ricard Gavalda
Floris Geerts
Peter Geibel
Lise Getoor
Olivier Gevaert
Rayid Ghani
Fosca Gianotti
Melanie Gnasa
Henrik Grosskreutz
Amit Gruber
Vincent Guigue
Robert Gwadera
Larry Hall
Zaid Harchaoui
Hannes Heikinheimo
Iris Hendrickx
Mark Herbster
Tom Heskes
Melanie Hilario
Alexander Hinneburg
Susanne Hoche
Frank Höppner
Geoff Holmes
Tamas Horvath
Bettina Hoser
Veronique Hoste
Andreas Hotho
Eyke Hüllermeier
Inaki Inza
Mariya Ishteva
Robert Jaeschke
Longin Jan Latecki
Nathalie Japkowicz

Szymon Jaroszewicz
Thorsten Joachims
Alipio Jorge
Felix Jungermann
Matti Kaariainen
Alexandros Kalousis
Murat Kantarcioglu
Samuel Kaski
Philip Kegelmeyer
Kristian Kersting
Svetlana Kiritchenko
Igor Kononenko
Anna Koop
Walter Kusters
Wojciech Kotlowski
Stefan Kramer
Andreas Krause
Yuval Krymolowski
Miroslav Kubat
Ravi Kumar
James Kwok
Bill Lampos
Niels Landwehr
Mark Last
Nada Lavrac
Lihong Li
Jessica Lin
Charles Ling
Huan Liu
XiaoHui Liu
Eneldo Loza Mencía
Peter Lucas
Elliot Ludvig
Yiming Ma
Sofus Macskassy
Michael Madden
Donato Malerba
Bradley Malin
Lluís Marquez
Michael May
Prem Melville
Rosa Meo
Pauli Miettinen
Roser Morante
Fabian Morchen

Katharina Morik
 Flavia Moser
 Fabien Moutarde
 Klaus-Robert Müller
 Ion Muslea
 Amedeo Napoli
 Olfa Nasraoui
 Vivi Nastase
 James Neufeld
 Alexandru Niculescu-Mizil
 Siegfried Nijssen
 Joakim Nivre
 Blaž Novak
 Ann Nowe
 Alexandros Ntoulas
 Andreas Nuernberger
 Guillaume Obozinski
 Arlindo Oliveira
 Martijn van Otterlo
 Gerhard Paass
 Cosmin Paduraru
 Georgios Paliouras
 Themis Palpanas
 Junfeng Pan
 Rong Pan
 Andrea Passerini
 Mykola Pechenizkiy
 Dmitry Pechyony
 Dino Pedreschi
 Kristiaan Pelckmans
 Jose-Maria Pena
 Ruggero Pensa
 Raffaele Perego
 Bernhard Pfahringer
 Christian Plagemann
 Barnabas Poczos
 Doina Precup
 Philippe Preux
 Kai Puolamaki
 Peter van der Putten
 Sampo Pyysalo
 Predrag Radivojac
 Troy Raeder Davood Rafiei
 Chedy Raissi
 Shyam Rajaram

Alain Rakotomamonjy
 Liva Ralaivola
 Jan Ramon
 Chotirat Ratanamahatana
 Chandan Reddy
 Ehud Reiter
 Elisa Ricci
 Martin Riedmiller
 Celine Robardet
 Marko Robnik-Sikonja
 Pedro Rodrigues
 Teemu Roos
 Fabrice Rossi
 Volker Roth
 Celine Rouveirol
 Ulrich Rueckert
 Stefan Rüping
 Yvan Saeys
 Lorenza Saitta
 Scott Sanner
 Vitor Santos Costa
 Craig Saunders
 Yucel Saygin
 Lars Schmidt-Thieme
 Jouni Seppanen
 Shashi Shekhar
 Jin Shieh
 Stefan Siersdorfer
 Tomi Silander
 Ricardo Silva
 Ozgur Simsek
 Ajit Singh
 Sergej Sizov
 Carlos Soares
 Maarten van Someren
 Yang Song
 Elaine Sousa
 Myra Spiliopoulou
 Karsten Steinhaeuser
 David Stern
 Jan Struyf
 Jiang Su
 Masashi Sugiyama
 Johan Suykens
 Vojtech Svatek

Sandor Szedmak
Nikolaj Tatti
Evimaria Terzi
Gerald Tesauro
Hanghang Tong
Volker Tresp
Koji Tsuda
Ville Tuulos
Rasmus Ulslev Pedersen
Dries Van Dyck
Stijn Vanderlooy
Sergei Vassilvskii
Cor Veenman
Paola Velardi
Shankar Vembu
Celine Vens
Jean-Philippe Vert
Ricardo Vilalta
Michalis Vlachos
Christel Vrain
Jilles Vreeken

Christian Walder
Xiaoyue Wang
Markus Weimer
David Wingate
Michael Wurst
Dragomir Yankov
Lexiang Ye
Jie Yin
François Yvon
Menno van Zaanen
Bianca Zadrozny
Osmar Zaiane
Mikhail Zaslavskiy
Gerson Zaverucha
Filip Zelezny
Justin Zhan
Bin Zhang
Zhi-Hua Zhou
Qiang Zhu
Xiaojin Zhu
Albrecht Zimmermann

Additional Reviewers

Dima Alberg
Anelia Angelova
Mohammad Aziz
Michele Berlingerio
Marenglen Biba
Alexander Binder
Zoran Bosnić
Christos Boutsidis
Fabian Buchwald
Markus Bundschuh
Wray Buntine
Lijuan Cai
Michelangelo Ceci
Weiwei Cheng
Joaquim Costa
Dave DeBarr
Marcos Domingues
Jun Du
Charles Elkan
Daan Fierens
Nuno A. Fonseca

Dmitriy Fradkin
Thomas Gabel
Zeno Gantner
Robby Goetschalckx
Habiba
Niina Haiminen
Katja Hansen
Andreas Hapfelmeier
Jingrui He
Raymond Heatherly
Thibault Helleputte
James Henderson
Yi Huang
Ali Inan
Tsuyoshi Ide
Szymon Jaroszewicz
Takafumi Kanamori
Alexandros Karatzoglou
Hisashi Kashima
Tsuyoshi Kato
Maarten Keijzer

Evan Kirshenbaum
Arto Klami
Thoralf Klein
Marius Kloft
Arne Koopman
Da Kuang
Mayank Lahiri
Tobias Lang
Lieven De Lathauwer
Gayle Leen
Jens Lehmann
Guy Lever
Biao Li
Nan Li
Shuyan Li
Yu-Feng Li
Yuan Li
Grigorios Loukides
Jose A. Lozano
Juan Luo
Hamid Reza Maei
Michael Mampaey
Alain-Pierre Manine
Leandro Marinho
Eneldo Loza Mencia
João Mendes-Moreira
Olana Missura
Matteo Mordacchini
Marianne Mueller
Eileen A. Ni
Martijn van Otterlo
Aline Paes
Indranil Palit
Sang-Hyeun Park
Aritz Perez
Claudia Perlich
Georgios Petasis
Dimitrios Pierrakos

Fabio Pinelli
Cristiano Pitangui
Troy Raeder
Alain Rakotomamonjy
Steffen Rendle
Achim Rettinger
François Riout
Jan Rupnik
Jarkko Salojärvi
Leander Schietgat
Jana Schmidt
Armin Shmilovici
David Silver
Karsten Steinhäuser
Erik Štrumbelj
Jan Struyf
Ilija Subašić
Taiji Suzuki
Takashi Takenouchi
Nicola Tonello
Grigorios Tsoumakas
Mikalai Tsytarau
Stijn Vanderlooy
Guy Van den Broeck
Sicco Verwer
Ricardo Vilalta
Dimitrios Vogiatzis
Petar Vračar
Chris Watkins
Joerg Wicker
Eric Wiewiora
Fuxiao Xin
Xingwei Yan
Xingwei Yang
Monika Zakova
De-Chuan Zhan
Indre Zliobaite

Sponsors

We wish to express our gratitude to the sponsors of ECML PKDD 2009 for their essential contribution to the conference. We wish to thank Jožef Stefan Institute, Slovenia, for providing financial and organizational means for the conference; the European Office of Aerospace Research and Development, a detachment of U.S. Air Force Office of Scientific Research (EOARD) for generous financial support; Pascal European Network of Excellence (PASCAL2) for sponsoring the Invited Speaker program and the videorecording of the conference; Slovenia Research Agency (ARRS); Google for supporting a Poster Reception; Microsoft Research Ltd., Cambridge, UK for supporting the Industrial track; Yahoo! Research, Quintelligence, Hewlett-Packard Labs, and ACTIVE European Integrated project for their financial support; the *Machine Learning* Journal for supporting the Student Best Paper Award; the *Data Mining and Knowledge Discovery* Journal for supporting the Student Best Paper Award; Nokia for sponsoring the Discovery Challenge Awards and the Best Demo Award.



Table of Contents – Part II

Regular Papers

Decomposition Algorithms for Training Large-Scale Semiparametric Support Vector Machines	1
<i>Sangkyun Lee and Stephen J. Wright</i>	
A Convex Method for Locating Regions of Interest with Multi-instance Learning	15
<i>Yu-Feng Li, James T. Kwok, Ivor W. Tsang, and Zhi-Hua Zhou</i>	
Active Learning for Reward Estimation in Inverse Reinforcement Learning	31
<i>Manuel Lopes, Francisco Melo, and Luis Montesano</i>	
Simulated Iterative Classification a New Learning Procedure for Graph Labeling	47
<i>Francis Maes, Stéphane Peters, Ludovic Denoyer, and Patrick Gallinari</i>	
Graph-Based Discrete Differential Geometry for Critical Instance Filtering	63
<i>Elena Marchiori</i>	
Integrating Novel Class Detection with Classification for Concept-Drifting Data Streams	79
<i>Mohammad M. Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham</i>	
Neural Networks for State Evaluation in General Game Playing	95
<i>Daniel Michulke and Michael Thielscher</i>	
Learning to Disambiguate Search Queries from Short Sessions	111
<i>Lilyana Mihalkova and Raymond Mooney</i>	
Dynamic Factor Graphs for Time Series Modeling	128
<i>Piotr Mirowski and Yann LeCun</i>	
On Feature Selection, Bias-Variance, and Bagging	144
<i>M. Arthur Munson and Rich Caruana</i>	
Efficient Pruning Schemes for Distance-Based Outlier Detection	160
<i>Nguyen Hoang Vu and Vivekanand Gopalkrishnan</i>	

The Sensitivity of Latent Dirichlet Allocation for Information Retrieval	176
<i>Laurence A.F. Park and Kotagiri Ramamohanarao</i>	
Efficient Decoding of Ternary Error-Correcting Output Codes for Multiclass Classification	189
<i>Sang-Hyeun Park and Johannes Fürnkranz</i>	
The Model of Most Informative Patterns and Its Application to Knowledge Extraction from Graph Databases	205
<i>Frédéric Pennerath and Amedeo Napoli</i>	
On Discriminative Parameter Learning of Bayesian Network Classifiers	221
<i>Franz Pernkopf and Michael Wohlmayr</i>	
Mining Spatial Co-location Patterns with Dynamic Neighborhood Constraint	238
<i>Feng Qian, Qinming He, and Jiangfeng He</i>	
Classifier Chains for Multi-label Classification	254
<i>Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank</i>	
Dependency Tree Kernels for Relation Extraction from Natural Language Text	270
<i>Frank Reichartz, Hannes Korte, and Gerhard Paass</i>	
Statistical Relational Learning with Formal Ontologies	286
<i>Achim Rettinger, Matthias Nickles, and Volker Tresp</i>	
Boosting Active Learning to Optimality: A Tractable Monte-Carlo, Billiard-Based Algorithm	302
<i>Philippe Rolet, Michèle Sebag, and Olivier Teytaud</i>	
Capacity Control for Partially Ordered Feature Sets	318
<i>Ulrich Rückert</i>	
Reconstructing Data Perturbed by Random Projections When the Mixing Matrix Is Known	334
<i>Yingpeng Sang, Hong Shen, and Hui Tian</i>	
Identifying the Original Contribution of a Document via Language Modeling	350
<i>Benyah Shaparenko and Thorsten Joachims</i>	
Relaxed Transfer of Different Classes via Spectral Partition	366
<i>Xiaoxiao Shi, Wei Fan, Qiang Yang, and Jiangtao Ren</i>	
Mining Databases to Mine Queries Faster	382
<i>Arno Siebes and Diyah Puspitaningrum</i>	

MACs: Multi-Attribute Co-clusters with High Correlation Information	398
<i>Kelvin Sim, Vivekanand Gopalkrishnan, Hon Nian Chua, and See-Kiong Ng</i>	
Bi-directional Joint Inference for Entity Resolution and Segmentation Using Imperatively-Defined Factor Graphs	414
<i>Sameer Singh, Karl Schultz, and Andrew McCallum</i>	
Latent Dirichlet Allocation for Automatic Document Categorization	430
<i>István Bíró and Jácint Szabó</i>	
New Regularized Algorithms for Transductive Learning	442
<i>Partha Pratim Talukdar and Koby Crammer</i>	
Enhancing the Performance of Centroid Classifier by ECOC and Model Refinement	458
<i>Songbo Tan, Gaowei Wu, and Xueqi Cheng</i>	
Optimal Online Learning Procedures for Model-Free Policy Evaluation	473
<i>Tsuyoshi Ueno, Shin-ichi Maeda, Motoaki Kawanabe, and Shin Ishii</i>	
Kernels for Periodic Time Series Arising in Astronomy	489
<i>Gabriel Wachman, Roni Khardon, Pavlos Protopapas, and Charles R. Alcock</i>	
K-Subspace Clustering	506
<i>Dingding Wang, Chris Ding, and Tao Li</i>	
Latent Dirichlet Bayesian Co-Clustering	522
<i>Pu Wang, Carlotta Domeniconi, and Kathryn Blackmond Laskey</i>	
Variational Graph Embedding for Globally and Locally Consistent Feature Extraction	538
<i>Shuang-Hong Yang, Hongyuan Zha, S. Kevin Zhou, and Bao-Gang Hu</i>	
Protein Identification from Tandem Mass Spectra with Probabilistic Language Modeling	554
<i>Yiming Yang, Abhay Harpale, and Subramaniam Ganapathy</i>	
Causality Discovery with Additive Disturbances: An Information-Theoretical Perspective	570
<i>Kun Zhang and Aapo Hyvärinen</i>	
Subspace Regularization: A New Semi-supervised Learning Method	586
<i>Yan-Ming Zhang, Xinwen Hou, Shiming Xiang, and Cheng-Lin Liu</i>	

Heteroscedastic Probabilistic Linear Discriminant Analysis with Semi-supervised Extension	602
<i>Yu Zhang and Dit-Yan Yeung</i>	
Semi-Supervised Multi-Task Regression	617
<i>Yu Zhang and Dit-Yan Yeung</i>	
A Flexible and Efficient Algorithm for Regularized Fisher Discriminant Analysis	632
<i>Zhihua Zhang, Guang Dai, and Michael I. Jordan</i>	
Debt Detection in Social Security by Sequence Classification Using Both Positive and Negative Patterns	648
<i>Yanchang Zhao, Huai Feng Zhang, Shanshan Wu, Jian Pei, Longbing Cao, Chengqi Zhang, and Hans Bohlscheid</i>	
Learning the Difference between Partially Observable Dynamical Systems	664
<i>Sami Zhioua, Doina Precup, François Laviolette, and Josée Desharnais</i>	
Universal Learning over Related Distributions and Adaptive Graph Transduction	678
<i>Erheng Zhong, Wei Fan, Jing Peng, Olivier Verscheure, and Jiangtao Ren</i>	
The Feature Importance Ranking Measure	694
<i>Alexander Zien, Nicole Krämer, Sören Sonnenburg, and Gunnar Rätsch</i>	
Demo Papers	
OTTHO: On the Tip of My THOught	710
<i>Pierpaolo Basile, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro</i>	
Protecting Sensitive Topics in Text Documents with PROTEXTOR	714
<i>Chad Cumby</i>	
Enhanced Web Page Content Visualization with Firefox	718
<i>Lorand Dali, Delia Rusu, and Dunja Mladenić</i>	
ClusTR: Exploring Multivariate Cluster Correlations and Topic Trends	722
<i>Luigi Di Caro and Alejandro Jaimes</i>	
Visual OntoBridge: Semi-automatic Semantic Annotation Software	726
<i>Miha Grcar and Dunja Mladenic</i>	

Semi-automatic Categorization of Videos on VideoLectures.net	730
<i>Miha Grcar, Dunja Mladenic, and Peter Kese</i>	
Discovering Patterns in Flows: A Privacy Preserving Approach with the ACSM Prototype	734
<i>Stéphanie Jacquemont, François Jacquenet, and Marc Sebban</i>	
Using Temporal Language Models for Document Dating	738
<i>Nattiya Kanhabua and Kjetil Nørnvåg</i>	
Omiotis: A Thesaurus-Based Measure of Text Relatedness	742
<i>George Tsatsaronis, Iraklis Varlamis, Michalis Vazirgiannis, and Kjetil Nørnvåg</i>	
Found in Translation	746
<i>Marco Turchi, Ilias Flaounas, Omar Ali, Tijl De Bie, Tristan Snowsill, and Nello Cristianini</i>	
A Community-Based Platform for Machine Learning Experimentation	750
<i>Joaquin Vanschoren and Hendrik Blockeel</i>	
TeleComVis: Exploring Temporal Communities in Telecom Networks . . .	755
<i>Qi Ye, Bin Wu, Lijun Suo, Tian Zhu, Chao Han, and Bai Wang</i>	
Author Index	759

Table of Contents – Part I

Invited Talks (Abstracts)

Theory-Practice Interplay in Machine Learning – Emerging Theoretical Challenges	1
<i>Shai Ben-David</i>	
Are We There Yet?	2
<i>Nello Cristianini</i>	
The Growing Semantic Web	3
<i>Mark Greaves</i>	
Privacy in Web Search Query Log Mining	4
<i>Rosie Jones</i>	
Highly Multilingual News Analysis Applications	5
<i>Ralf Steinberger</i>	

Machine Learning Journal Abstracts

Combining Instance-Based Learning and Logistic Regression for Multilabel Classification	6
<i>Weiwei Cheng and Eyke Hüllermeier</i>	
On Structured Output Training: Hard Cases and an Efficient Alternative	7
<i>Thomas Gärtner and Shankar Vembu</i>	
Sparse Kernel SVMs via Cutting-Plane Training	8
<i>Thorsten Joachims and Chun-Nam John Yu</i>	
Hybrid Least-Squares Algorithms for Approximate Policy Evaluation ...	9
<i>Jeff Johns, Marek Petrik, and Sridhar Mahadevan</i>	
A Self-training Approach to Cost Sensitive Uncertainty Sampling	10
<i>Alexander Liu, Goo Jun, and Joydeep Ghosh</i>	
Learning Multi-linear Representations of Distributions for Efficient Inference	11
<i>Dan Roth and Rajhans Samdani</i>	
Cost-Sensitive Learning Based on Bregman Divergences	12
<i>Raúl Santos-Rodríguez, Alicia Guerrero-Curieses, Rocío Alaiz-Rodríguez, and Jesús Cid-Sueiro</i>	

Data Mining and Knowledge Discovery Journal Abstracts

RTG: A Recursive Realistic Graph Generator Using Random Typing . . . 13
Leman Akoglu and Christos Faloutsos

Taxonomy-Driven Lumping for Sequence Mining 29
Francesco Bonchi, Carlos Castillo, Debora Donato, and Aristides Gionis

On Subgroup Discovery in Numerical Domains 30
Henrik Grosskreutz and Stefan Rüping

Harnessing the Strengths of Anytime Algorithms for Constant Data Streams 31
Philipp Kranen and Thomas Seidl

Identifying the Components 32
Matthijs van Leeuwen, Jilles Vreeken, and Arno Siebes

Two-Way Analysis of High-Dimensional Collinear Data 33
Ilkka Huopaniemi, Tommi Suviavaara, Janne Nikkilä, Matej Orešič, and Samuel Kaski

A Fast Ensemble Pruning Algorithm Based on Pattern Mining Process 34
Qiang-Li Zhao, Yan-Huang Jiang, and Ming Xu

Regular Papers

Evaluation Measures for Multi-class Subgroup Discovery 35
Tarek Abudawood and Peter Flach

Empirical Study of Relational Learning Algorithms in the Phase Transition Framework 51
Erick Alphonse and Aomar Osmani

Topic Significance Ranking of LDA Generative Models 67
Louwah AlSumait, Daniel Barbará, James Gentle, and Carlotta Domeniconi

Communication-Efficient Classification in P2P Networks 83
Hock Hee Ang, Vivekanand Gopalkrishnan, Wee Keong Ng, and Steven Hoi

A Generalization of Forward-Backward Algorithm 99
Ai Azuma and Yuji Matsumoto

Mining Graph Evolution Rules	115
<i>Michele Berlingerio, Francesco Bonchi, Björn Bringmann, and Aristides Gionis</i>	
Parallel Subspace Sampling for Particle Filtering in Dynamic Bayesian Networks	131
<i>Eva Besada-Portas, Sergey M. Plis, Jesus M. de la Cruz, and Terran Lane</i>	
Adaptive XML Tree Classification on Evolving Data Streams	147
<i>Albert Bifet and Ricard Gavaldà</i>	
A Condensed Representation of Itemsets for Analyzing Their Evolution over Time	163
<i>Mirko Boettcher, Martin Spott, and Rudolf Kruse</i>	
Non-redundant Subgroup Discovery Using a Closure System	179
<i>Mario Boley and Henrik Grosskreutz</i>	
PLSI: The True Fisher Kernel and beyond: IID Processes, Information Matrix and Model Identification in PLSI	195
<i>Jean-Cédric Chappelier and Emmanuel Eckard</i>	
Semi-supervised Document Clustering with Simultaneous Text Representation and Categorization	211
<i>Yanhua Chen, Lijun Wang, and Ming Dong</i>	
One Graph Is Worth a Thousand Logs: Uncovering Hidden Structures in Massive System Event Logs	227
<i>Michal Aharon, Gilad Barash, Ira Cohen, and Eli Mordechai</i>	
Conference Mining via Generalized Topic Modeling	244
<i>Ali Daud, Juanzi Li, Lizhu Zhou, and Faqir Muhammad</i>	
Within-Network Classification Using Local Structure Similarity	260
<i>Christian Desrosiers and George Karypis</i>	
Multi-task Feature Selection Using the Multiple Inclusion Criterion (MIC)	276
<i>Paramveer S. Dhillon, Brian Tomasik, Dean Foster, and Lyle Ungar</i>	
Kernel Polytope Faces Pursuit	290
<i>Tom Diethe and Zakria Hussain</i>	
Soft Margin Trees	302
<i>Jorge Díez, Juan José del Coz, Antonio Bahamonde, and Oscar Luaces</i>	
Feature Weighting Using Margin and Radius Based Error Bound Optimization in SVMs	315
<i>Huyen Do, Alexandros Kalousis, and Melanie Hilario</i>	

Margin and Radius Based Multiple Kernel Learning	330
<i>Huyen Do, Alexandros Kalousis, Adam Woznica, and Melanie Hilario</i>	
Inference and Validation of Networks	344
<i>Ilias N. Flaounas, Marco Turchi, Tijl De Bie, and Nello Cristianini</i>	
Binary Decomposition Methods for Multipartite Ranking	359
<i>Johannes Fürnkranz, Eyke Hüllermeier, and Stijn Vanderlooy</i>	
Leveraging Higher Order Dependencies between Features for Text Classification	375
<i>Murat C. Ganiz, Nikita I. Lytkin, and William M. Pottenger</i>	
Syntactic Structural Kernels for Natural Language Interfaces to Databases	391
<i>Alessandra Giordani and Alessandro Moschitti</i>	
Active and Semi-supervised Data Domain Description	407
<i>Nico Görnitz, Marius Kloft, and Ulf Brefeld</i>	
A Matrix Factorization Approach for Integrating Multiple Data Views	423
<i>Derek Greene and Pádraig Cunningham</i>	
Transductive Classification via Dual Regularization	439
<i>Quanquan Gu and Jie Zhou</i>	
Stable and Accurate Feature Selection	455
<i>Gokhan Gulgezen, Zehra Cataltepe, and Lei Yu</i>	
Efficient Sample Reuse in EM-Based Policy Search	469
<i>Hiroataka Hachiya, Jan Peters, and Masashi Sugiyama</i>	
Applying Electromagnetic Field Theory Concepts to Clustering with Constraints	485
<i>Huseyin Hakkoymaz, Georgios Chatzimilioudis, Dimitrios Gunopulos, and Heikki Mannila</i>	
An ℓ_1 Regularization Framework for Optimal Rule Combination	501
<i>Yanjun Han and Jue Wang</i>	
A Generic Approach to Topic Models	517
<i>Gregor Heinrich</i>	
Feature Selection by Transfer Learning with Linear Regularized Models	533
<i>Thibault Helleputte and Pierre Dupont</i>	
Integrating Logical Reasoning and Probabilistic Chain Graphs	548
<i>Arjen Hommersom, Nivea Ferreira, and Peter J.F. Lucas</i>	

Max-Margin Weight Learning for Markov Logic Networks	564
<i>Tuyen N. Huynh and Raymond J. Mooney</i>	
Parameter-Free Hierarchical Co-clustering by n -Ary Splits	580
<i>Dino Ienco, Ruggero G. Pensa, and Rosa Meo</i>	
Mining Peculiar Compositions of Frequent Substrings from Sparse Text Data Using Background Texts	596
<i>Daisuke Ikeda and Einoshin Suzuki</i>	
Minimum Free Energy Principle for Constraint-Based Learning Bayesian Networks	612
<i>Takashi Isozaki and Maomi Ueno</i>	
Kernel-Based Copula Processes	628
<i>Sebastian Jaimungal and Eddie K.H. Ng</i>	
Compositional Models for Reinforcement Learning	644
<i>Nicholas K. Jong and Peter Stone</i>	
Feature Selection for Value Function Approximation Using Bayesian Model Selection	660
<i>Tobias Jung and Peter Stone</i>	
Learning Preferences with Hidden Common Cause Relations	676
<i>Kristian Kersting and Zhao Xu</i>	
Feature Selection for Density Level-Sets	692
<i>Marius Kloft, Shinichi Nakajima, and Ulf Brefeld</i>	
Efficient Multi-start Strategies for Local Search Algorithms	705
<i>Levente Kocsis and András György</i>	
Considering Unseen States as Impossible in Factored Reinforcement Learning	721
<i>Olga Kozlova, Olivier Sigaud, Pierre-Henri Wuillemin, and Christophe Meyer</i>	
Relevance Grounding for Planning in Relational Domains	736
<i>Tobias Lang and Marc Toussaint</i>	
Author Index	753

Mining Databases to Mine Queries Faster

Arno Siebes and Diyah Puspitaningrum

Department Of Information and Computing Sciences
Universiteit Utrecht, Utrecht, The Netherlands
{arno,diyah}@cs.uu.nl

Abstract. Inductive databases are databases in which models and patterns are first class citizens. Having models and patterns in the database raises the question: do the models and patterns that are stored help in computing new models and patterns? For example, let C be a classifier on database DB and let Q be a query. Does knowing C speed up the induction of a new classifier on the result of Q ?

In this paper we answer this problem positively for the code tables induced by our KRIMP algorithm. More in particular, assume we have the code tables for all tables in the database. Then we can approximate the code table induced by KRIMP on the result of a query, using only these global code tables as candidates. That is, we do not have to mine for frequent item sets on the query result.

1 Introduction

The problem investigated in this paper can informally be phrased as follows. Let M_{DB} be the model we induced from database DB and let Q be a query on DB . Does knowing M_{DB} help in inducing a model M_Q on $Q(DB)$, i.e., on the result of Q when applied to DB . For example, if M_{DB} is a classifier and Q selects a subset of DB , does knowing M_{DB} speed-up the induction of a new classifier M_Q on the subset $Q(DB)$?

There are at least two contexts in which this question is relevant. Firstly in the context of inductive databases. Ever since their introduction in the seminal paper by Imielinski and Mannila [11], they have been a constant theme in data mining research. There is no formal definition of an inductive database, in fact, it may be too early for such a definition [14]. However, consensus is that models and patterns should be first class citizens in such a database. That is, e.g., one should be able to query for patterns. Having models and patterns in the database naturally raises the question: do the models and patterns that are stored help in computing new models and patterns?

The second context in which the problem is relevant is in every day data mining practice. In the data mining literature, the usual assumption is that we are given some database that has to be mined. In practice, however, this assumption is usually not met. Rather, the construction of the mining database is often one of the hardest parts of the KDD process. The data often resides in a data warehouse or in multiple databases, and the mining database is constructed from these underlying databases.

From most perspectives, it is not very interesting to know whether one mines a specially constructed database or an original database. For example, if the goal is to build the best possible classifier on that data set, the origins of the database are of no importance whatsoever.

It makes a difference, however, if the underlying databases have already been modelled. Then, like with inductive databases, one would hope that knowing such models would help in modelling the specially constructed ‘mining database. For example, if we have constructed a classifier on a database of customers, one would hope that this would help in developing a classifier for the female customers only.

So, the problem is relevant, but isn’t it trivial? After all, if M_{DB} is a good model on DB , it is almost always also a good model on a random subset of DB ; almost always, because a random subset may be highly untypical. The problem is, however, *not* trivial because queries in general do *not* compute a random subset. Rather, queries construct a very specific result.

For the usual “project-select-join” queries, there is not even a natural way in which the query-result can be seen as subset of the original database. Even if Q is just a “select”-query, the result is usually not random and M_{DB} can even be highly misleading on $Q(DB)$. This is nicely illustrated by the well-known example of *Simpson’s Paradox*, viz., Berkeley’s admission data [2]. Overall, 44% of the male applicants were admitted, while only 35% of the females were admitted. Four of the six departments, however, have a bias that is in favour of female applicants. While the overall model may be adequate for certain purposes, it is woefully inadequate for a query that selects a single department.

Solving the problem for all model classes and algorithms is a rather daunting task. Rather, in this paper we study the problem for one specific class of models, viz., the code tables induced by our KRIMP algorithm [15]. Given all frequent item sets on a table, KRIMP selects a small subset of these frequent item sets. The reason why we focus on KRIMP is threefold. Firstly, because together the selected item sets describe the underlying data distribution of the complete database very well, see, e.g., [16,17]. Secondly, because the code table consists of *local* patterns. Such a local pattern can be seen as a selection query on the database (for the transactions in its support), hence, one would expect KRIMP to do well on selection queries. Thirdly, from earlier research on KRIMP in a multi-relational setting, we noticed as a side-result that KRIMP is probably easily transformed for joins [13]; this is investigated further in this paper.

More in particular, we show that if we know the code tables for all tables in the database, then we can approximate the code table induced by KRIMP on the result of a query, using *only* the item sets in these global code tables as candidates. Since KRIMP is linear in the number of candidates and KRIMP reduces the set of frequent item sets by many orders of magnitude, this means that we can now speed up the induction of code tables on query results by many orders of magnitude.

This speed-up results in a slightly less good code table, but it approximates the optimal solution within a few percent. We will formalise “approximation”

in terms of MDL [10]. Hence, the data miner has a choice: either a quick, good approximation, or the optimal result taking longer time to compute.

2 Problem Statement

This section starts with some preliminaries and assumptions. Then we introduce the problem informally. To formalise it we use MDL, which is briefly discussed.

2.1 Preliminaries and Assumptions

We assume that our data resides in relational databases. In fact, note that the union of two relational databases is, again, a relational database. Hence, we assume, without loss of generality, that our data resides in *one* relational database DB . As query language we will use the standard relational algebra. More precisely, we focus on the usual “select-project-join” queries. That is, on the selection operator σ , the projection operator π , and the (equi-)join operator \bowtie ; see [5]. Note that, as usual in the database literature, we use *bag* semantics. That is, we do allow duplicates tuples in tables and query results.

As mentioned in the introduction, the mining database is constructed from DB using queries. Given the compositionality of the relational algebra, we may assume, again without loss of generality, that the analysis database is constructed using one query Q . That is, the analysis database is $Q(DB)$, for some relational algebra expression Q . Since DB is fixed, we will often simply write Q for $Q(DB)$; that is, we will use Q to denote both the query and its result.

2.2 The Problem Informally

In the introduction we stated that knowing a model on DB should help in inducing a model on Q . To make this more precise, let \mathcal{A} be our data mining algorithm. At this point, \mathcal{A} can be any algorithm, it may, e.g., compute a decision tree, all frequent item sets or a neural network.

Let \mathcal{M}_{DB} denote the model induced by \mathcal{A} from DB , i.e., $\mathcal{M}_{DB} = \mathcal{A}(DB)$. Similarly, let $\mathcal{M}_Q = \mathcal{A}(Q)$. We want to transform \mathcal{A} into an algorithm \mathcal{A}^* that takes at least two inputs, i.e., both Q and \mathcal{M}_{DB} , such that:

1. \mathcal{A}^* gives a reasonable approximation of \mathcal{A} when applied to Q , i.e.,

$$\mathcal{A}^*(Q, \mathcal{M}_{DB}) \approx \mathcal{M}_Q$$

2. $\mathcal{A}^*(Q, \mathcal{M}_{DB})$ is simpler to compute than \mathcal{M}_Q .

The second criterion is easy to formalise: the runtime of \mathcal{A}^* should be shorter than that of \mathcal{A} . The first one is harder. What do we mean that one model is an approximation of another? Moreover, what does it mean that it is a *reasonable* approximation?

2.3 Model Approximation

The answer to the question how to formalise that one model approximates another depends very much on the goal. If \mathcal{A} induces classifiers, approximation should probably be defined in terms of prediction accuracy, e.g., on the Area Under the ROC-curve (AUC).

KRIMP computes code tables. Hence, the quick approximating algorithm we are looking for, KRIMP^* in the notation used above, also has to compute code tables. So, one way to define the notion of approximation is by comparing the resulting code tables. Let CT_{KRIMP} be the code table computed by KRIMP and similarly, let CT_{KRIMP^*} denote the code table computed by KRIMP^* on the same data set. The more similar CT_{KRIMP^*} is to CT_{KRIMP} , the better KRIMP^* approximates KRIMP.

While this is intuitively a good way to proceed, it is far from obvious how to compare two code tables. Fortunately, we do not need to compare code tables directly. KRIMP is based on the Minimum Description Length principle (MDL) [10], and MDL offers another way to compare models, viz., by their *compression-rate*. Note that using MDL to define “approximation” has the advantage that we can formalise our problem for a larger class of algorithms than just KRIMP. It is formalised for all algorithms that are based on MDL. MDL is quickly becoming a popular formalism in data mining research, see, e.g., [8] for an overview of other applications of MDL in data mining.

2.4 Minimum Description Length

MDL embraces the slogan *Induction by Compression*. It can be roughly described as follows.

Given a set of models¹ \mathcal{H} , the best model $H \in \mathcal{H}$ is the one that minimises

$$L(H) + L(D|H)$$

in which

- $L(H)$ is the length, in bits, of the description of H , and
- $L(D|H)$ is the length, in bits, of the description of the data when encoded with H .

One can paraphrase this by: the smaller $L(H) + L(D|H)$, the better H models D .

What we are interested in is comparing two algorithms on the same data set, viz., on $Q(DB)$. Slightly abusing notation, we will write $\mathcal{L}(\mathcal{A}(Q))$ for $L(\mathcal{A}(Q)) + L(Q(DB)|\mathcal{A}(Q))$, similarly, we will write $\mathcal{L}(\mathcal{A}^*(Q, \mathcal{M}_{DB}))$. Then, we are interested in comparing $\mathcal{L}(\mathcal{A}^*(Q, \mathcal{M}_{DB}))$ to $\mathcal{L}(\mathcal{A}(Q))$. The closer the former is to the latter, the better the approximation is.

¹ MDL-theorists tend to talk about *hypothesis* in this context, hence the \mathcal{H} ; see [10] for the details.

Just taking the difference of the two, however, can be quite misleading. Take, e.g., two databases db_1 and db_2 sampled from the same underlying distribution, such that db_1 is far bigger than db_2 . Moreover, fix a model H . Then necessarily $L(db_1|H)$ is bigger than $L(db_2|H)$. In other words, big absolute numbers do not necessarily mean very much. We have to *normalise* the difference to get a feeling for how good the approximation is. Therefore we define the asymmetric dissimilarity measure (ADM) as follows.

Definition 1. Let H_1 and H_2 be two models for a dataset D . The asymmetric dissimilarity measure $ADM(H_1, H_2)$ is defined by:

$$ADM(H_1, H_2) = \frac{|\mathcal{L}(H_1) - \mathcal{L}(H_2)|}{\mathcal{L}(H_2)}$$

Note that this dissimilarity measure is related to the Normalised Compression Distance [4]. The reason why we use this asymmetric version is that we have a “gold standard”. We want to know how far our approximate result $\mathcal{A}^*(Q, \mathcal{M}_{DB})$ deviates from the optimal result $\mathcal{A}(Q)$.

Clearly, $ADM(\mathcal{A}^*(Q, \mathcal{M}_{DB}), \mathcal{A}(Q))$ does not only depend on \mathcal{A}^* and on \mathcal{A} , but also very much on Q . We do not seek a low ADM on one particular Q , rather we want to have a reasonable approximation on all possible queries. Requiring that the ADM is equally small on all possible queries seems to strong a requirement. Some queries might result in a very untypical subset of DB , the ADM is probably higher on the result of such queries than it is on queries that result in more typical subsets. Hence, it is more reasonable to require that the ADM is small most of the time. This is formalised through the notion of an (ϵ, δ) -approximation.

Definition 2. Let DB be a database and let Q be a random query on DB . Moreover, let \mathcal{A}_1 and \mathcal{A}_2 be two data mining algorithms on DB . \mathcal{A}_1 is an (ϵ, δ) -approximation of \mathcal{A}_2 iff

$$P(ADM(\mathcal{A}_1(Q), \mathcal{A}_2(Q)) > \epsilon) < \delta$$

2.5 The Problem

Using the notation introduced above, we formalise the problem as follows.

Problem Statement

For a given data mining algorithm \mathcal{A} , devise an algorithm \mathcal{A}^* , such that for a random database DB :

1. \mathcal{A}^* is an (ϵ, δ) -approximation of \mathcal{A} for reasonable values for ϵ and δ .
2. Computing $\mathcal{A}^*(Q, \mathcal{M}_{DB})$ is faster than computing $\mathcal{A}(Q)$ for a random query Q on DB .

What reasonable values for ϵ and δ are depends very much on the application. While $\epsilon = 0.5$ for $\delta = 0.9$ might be acceptable for one application, these values may be unacceptable for others.

The ultimate solution to the problem as stated here would be an algorithm that transforms any data mining algorithm \mathcal{A} in an algorithm \mathcal{A}^* with the requested properties. This is a rather ambitious, ill-defined (what is the class of all data mining algorithms?), and, probably, not attainable goal. Hence, in this paper we take a more modest approach: we transform one algorithm only, our KRIMP algorithm.

3 Introducing KRIMP

For the convenience of the reader we provide a brief introduction to KRIMP in this section, it was originally introduced in [15] (although not by that name) and the reader is referred to that paper for more details.

Since KRIMP selects a small set of representative item sets from the set of all frequent item sets, we first recall the basic notions of frequent item set mining [1].

3.1 Preliminaries

Let $\mathcal{I} = \{I_1, \dots, I_n\}$ be a set of binary (0/1 valued) attributes. That is, the domain D_i of item I_i is $\{0, 1\}$. A transaction (or tuple) over \mathcal{I} is an element of $\prod_{i \in \{1, \dots, n\}} D_i$. A database DB over \mathcal{I} is a bag of tuples over \mathcal{I} . This bag is indexed in the sense that we can talk about the i -th transaction.

An item set J is, as usual, a subset of \mathcal{I} , i.e., $J \subseteq \mathcal{I}$. The item set J occurs in a transaction $t \in DB$ if $\forall I \in J : \pi_I(t) = 1$. The *support* of item set J in database DB is the number of transactions in DB in which J occurs. That is, $supp_{DB}(J) = |\{t \in DB \mid J \text{ occurs in } t\}|$. An item set is called *frequent* if its support is larger than some user-defined threshold called the *minimal support* or *min-sup*. Given the A Priori property,

$$\forall I, J \in \mathcal{P}(\mathcal{I}) : I \subset J \rightarrow supp_{DB}(J) \leq supp_{DB}(I)$$

frequent item sets can be mined efficiently level wise, see [1] for more details.

Note that while we restrict ourself to binary databases in the description of our problem and algorithms, there is a trivial generalisation to categorical databases. In the experiments, we use such categorical databases.

3.2 KRIMP

The key idea of the KRIMP algorithm is the code table. A code table is a two-column table that has item sets on the left-hand side and a code for each item set on its right-hand side. The item sets in the code table are ordered descending on 1) item set length and 2) support size and 3) lexicographically. The actual codes on the right-hand side are of no importance but their lengths are. To explain how these lengths are computed, the coding algorithm needs to be introduced.

A transaction t is encoded by KRIMP by searching for the first item set I in the code table for which $I \subseteq t$. The code for I becomes part of the encoding of t . If $t \setminus I \neq \emptyset$, the algorithm continues to encode $t \setminus I$. Since it is insisted that

each code table contains at least all singleton item sets, this algorithm gives a unique encoding to each (possible) transaction over \mathcal{I} .

The set of item sets used to encode a transaction is called its *cover*. Note that the coding algorithm implies that a cover consists of non-overlapping item sets.

The length of the code of an item in a code table CT depends on the database we want to compress; the more often a code is used, the shorter it should be. To compute this code length, we encode each transaction in the database DB . The *frequency* of an item set $I \in CT$, denoted by $freq(I)$ is the number of transactions $t \in DB$ which have I in their cover. That is,

$$freq(I) = |\{t \in DB | I \in cover(t)\}|$$

The relative frequency of $I \in CT$ is the probability that I is used to encode an arbitrary $t \in DB$, i.e.

$$P(I|DB) = \frac{freq(I)}{\sum_{J \in CT} freq(J)}$$

For optimal compression of DB , the higher $P(c)$, the shorter its code should be. Given that we also need a prefix code for unambiguous decoding, we use the well-known optimal Shannon code [7]:

$$l(I|CT) = -\log(P(I|DB)) = -\log\left(\frac{freq(I)}{\sum_{J \in CT} freq(J)}\right)$$

The length of the encoding of a transaction is now simply the sum of the code lengths of the item sets in its cover. Therefore the encoded size of a transaction $t \in DB$ compressed using a specified code table CT is calculated as follows:

$$L(t|CT) = \sum_{I \in cover(t, CT)} l(I|CT)$$

The size of the encoded database is the sum of the sizes of the encoded transactions, but can also be computed from the frequencies of each of the elements in the code table:

$$\begin{aligned} L(DB|CT) &= \sum_{t \in DB} L(t|CT) \\ &= - \sum_{I \in CT} freq(I) \log\left(\frac{freq(I)}{\sum_{J \in CT} freq(J)}\right) \end{aligned}$$

To find the optimal code table using MDL, we need to take into account both the compressed database size, as described above, as well as the size of the code table. For the size of the code table, we only count those item sets that have a non-zero frequency. The size of the right-hand side column is obvious; it is simply the sum of all the different code lengths. For the size of the left-hand side column, note that the simplest valid code table consists only of the singleton item sets. This is the *standard encoding (ST)*, of which we use the codes to

compute the size of the item sets in the left-hand side column. Hence, the size of code table CT is given by:

$$L(CT|DB) = \sum_{I \in CT: freq(I) \neq 0} l(I|ST) + l(I|CT)$$

In [15] we defined the optimal set of (frequent) item sets as that one whose associated code table minimises the total compressed size:

$$L(CT, DB) = L(CT|DB) + L(DB|CT)$$

As before, this minimal compressed size of DB is denoted by $\mathcal{L}(DB)$. KRIMP starts with a valid code table (only the collection of singletons) and a sorted list of candidates (frequent item sets). These candidates are assumed to be sorted descending on 1) support size, 2) item set length and 3) lexicographically. Each candidate item set is considered by inserting it at the right position in CT and calculating the new total compressed size. A candidate is only kept in the code table iff the resulting total size is smaller than it was before adding the candidate. If it is kept, all other elements of CT are reconsidered to see if they still positively contribute to compression. The whole process is illustrated in Figure 1; see [15].

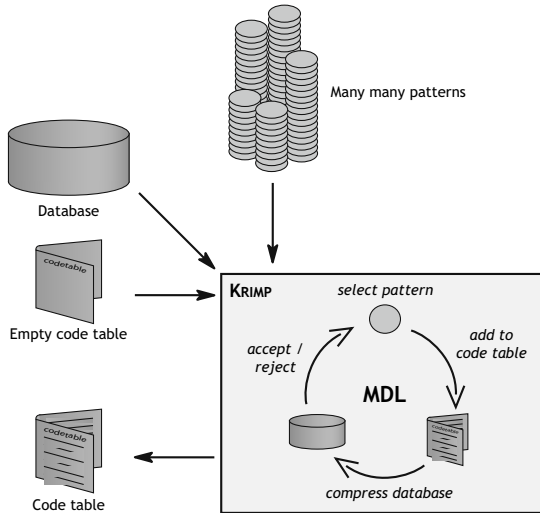


Fig. 1. KRIMP in action

4 The Problem for KRIMP

If we assume a fixed minimum support threshold for a database, KRIMP has only one essential parameter: the database. For, given the database and the

(fixed) minimum support threshold, the candidate list is also specified. Hence, we will simply write CT_{DB} and $\text{KRIMP}(DB)$, to denote the code table induced by KRIMP from DB . Similarly CT_Q and $\text{KRIMP}(Q)$ denote the code table induced by KRIMP from the result of applying query Q to DB .

Given that KRIMP results in a code table, there is only one sensible way in which $\text{KRIMP}(DB)$ can be re-used to compute $\text{KRIMP}(Q)$: provide KRIMP only with the item sets in CT_{DB} as candidates. While we change nothing to the algorithm, we'll use the notation KRIMP^* to indicate that KRIMP got only code table elements as candidates. So, e.g., $\text{KRIMP}^*(Q)$ is the code table that KRIMP induces from $Q(DB)$ using the item sets in CT_{DB} only.

Given our general problem statement, we now have to show that KRIMP^* satisfies our two requirements for a transformed algorithm. That is, we have to show for a random database DB :

- For reasonable values for ϵ and δ , KRIMP^* is an (ϵ, δ) -approximation of KRIMP , i.e, for a random query Q on DB :

$$P(\text{ADM}(\text{KRIMP}^*(Q), \text{KRIMP}(Q)) > \epsilon) < \delta$$

Or in MDL-terminology:

$$P\left(\frac{|\mathcal{L}(\text{KRIMP}^*(Q)) - \mathcal{L}(\text{KRIMP}(Q))|}{\mathcal{L}(\text{KRIMP}(Q))} > \epsilon\right) < \delta$$

- Moreover, we have to show that it is faster to compute $\text{KRIMP}^*(Q)$ than it is to compute $\text{KRIMP}(Q)$.

Neither of these two properties can be formally proven, if only because KRIMP and thus KRIMP^* are both heuristic algorithms. Rather, we report on extensive tests of these two requirements.

5 The Experiments

In this section we describe our experimental set-up. First we briefly describe the data sets we used. Next we discuss the queries used for testing. Finally we describe how the tests were performed.

5.1 The Data Sets

To test our hypothesis that KRIMP^* is a good and fast approximation of KRIMP , we have performed extensive tests mostly on 6 well-known UCI [6] data sets and one data set from the KDDcup 2004.

More in particular, we have used the data sets *connect*, *adult*, *chessBig*, *letRecog*, *PenDigits* and *mushroom* from UCI. These data sets were chosen because they are well suited for KRIMP . Some of the other data sets in the UCI repository are simply too small for KRIMP to perform well. MDL needs a reasonable amount of data to be able to function. Some other data sets are very dense. While

KRIMP performs well on these data sets, choosing them would have turned our extensive testing prohibitively time-consuming.

Since all these data sets are single table data sets, they do not allow testing with queries involving joins. To test such queries, we used tables from the “Hepatitis Medical Analysis”² of the KDDcup 2004. From this relational database we selected the tables *bio* and *hemat*. The former contains biopsy results, while the latter contains results on hematological analysis. The original tables have been converted to item set data and rows with missing data have been removed.

5.2 The Queries

To test our hypothesis, we need to consider randomly generated queries. On first sight this appears a daunting task. Firstly, because the set of all possible queries is very large. How do we determine a representative set of queries? Secondly, many of the generated queries will have no or very few results. If the query has no results, the hypothesis is vacuously true. If the result is very small, MDL (and KRIMP) doesn’t perform very well.

To overcome these problems, we restrict ourselves to queries that are build using selections (σ), projections (π), and joins (\bowtie) only. The rationale for this choice is twofold. Firstly, the well-known “project-select-join” queries are among the most used queries in practice. This is witnessed by the important role they play in benchmarks for DBMSs such as the TPC family of benchmarks. Secondly, simple queries will have, in general, larger results than more complex queries.

5.3 The Experiments

The experiments preformed for each of the queries on each of the data sets were generated as follows.

Projection: The projection queries were generated by randomly choosing a set X of n attributes, for $n \in \{1, 3, 5, 7, 9\}$. The generated query is then $\pi_{\overline{X}}$. That is, the elements of X are projected out of each of the transactions. For example, $\pi_{\overline{\{I_1, I_3\}}}(\{I_1, I_2, I_3\}) = \{I_2\}$. For this case, the code table elements generated on the complete data set were projected in the same way. For each value of n , 10 random sets X were generated on each data set.

As an aside, note that the rationale for limiting X to maximally 9 elements is that for larger values too many result sets became too small for meaningful results.

Selection: The random selection queries were again generated by randomly choosing a set X of n attributes, with $n \in \{1, 2, 3, 4\}$. Next for each random attribute A_i a random value v_i in its domain D_i was chosen. Finally, for each A_i in X a random $\theta_i \in \{=, \neq\}$ was chosen. The generated query is thus $\sigma(\bigwedge_{A_i \in X} A_i \theta_i v_i)$. As in the previous case, we performed 10 random experiments on each of the data sets for each of the values of n .

² <http://lisp.vse.cz/challenge/>

Project-Select: The random project-select queries generated, are essentially combinations of the simple projection and selection queries as explained above. The only difference is that we used $n \in \{1, 3\}$ for the projection and $n \in \{1, 2\}$ for the selections. That is we select on 1 or 2 attributes and we project away either 1 or 3 attributes. The size of the results is, of course, again the rationale for this choice. For each of the four combinations, we performed 100 random experiments on each of the data sets: first we chose randomly the selection (10 times for each selection), for each such selection we performed 10 random projections.

Project-Select-Join: Since we only use one “multi-relational” data set and there is only one possible way to join the *bio* and *hemat* tables, we could not do random tests for the join operator. However, in combination with projections and selections, we can perform random tests. These tests consist of randomly generated project-select queries on the join of *bio* and *hemat*. In this two-table case, KRIMP* got as input all pairs $(\mathcal{I}_1, \mathcal{I}_2)$ in which \mathcal{I}_1 is an item set in the code table of *bio*, and \mathcal{I}_2 is an item set in the code table of *hemat*. Again we select on 1 or 2 attributes and we project away either 1 or 3 attributes. And, again, we performed again 100 random experiments on the database for each of the four combinations; as above.

6 The Results

In this section we give an overview of the results of the experiments described in the previous section. Each test query is briefly discussed in its own subsection.

6.1 Projection Queries

In Figure 2 the results of the random projection queries on the *letRecog* data set are visualised. The marks in the picture denote the averages over the 10 experiments, while the error bars denote the standard deviation. Note that, while not statistically significant, the average ADM grows with the number of attributes projected away. This makes sense, since the more attributes are projected away, the smaller the result set becomes. On the other data sets, KRIMP* performs similarly. Since this is also clear from the project-select query results, we do not provide all details here. This will become clear when we report on the project-select queries.

6.2 Selection Queries

The results of the random selection queries on the *penDigits* data set are visualised in figure 3. For the same reason as above, it makes sense that the average ADM grows with the number of attributes selected on. Note, however, that the ADM averages for selection queries seem much larger than those for projection queries. These numbers are, however, not representative for the results on the other data sets. It turned out that *penDigits* is actually too small and sparse to

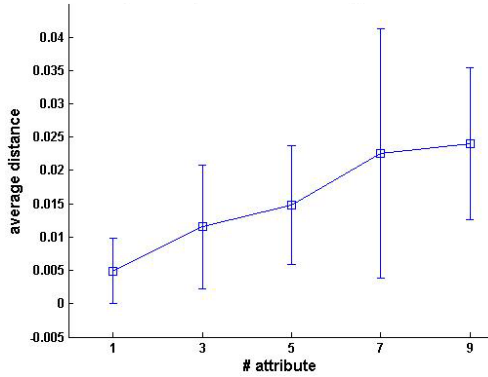


Fig. 2. Projection results on *letRecog*

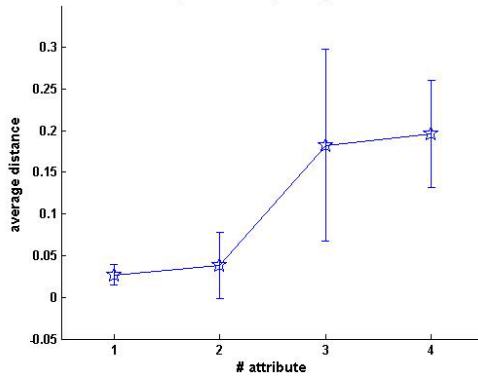


Fig. 3. Selection results on *penDigits*

test KRIMP* seriously. In the remainder of our results section, we do not report further results on *penDigits*. The reason why we report on it here is to illustrate that even on rather small and sparse data sets KRIMP* still performs reasonably well. On all other data sets KRIMP* performs far better, as will become clear when we report on the project-select queries.

6.3 Project-Select Queries

The results of the projection-select queries are given in the table in Figure 4. All numbers are the average ADM score \pm the standard deviation for the 100 random experiments. All the ADM numbers are rather small, only for mushroom do they get above 0.2. Two important observations can be made from this table. Firstly, as for the projection and selection queries reported on above, the ADM scores get only slightly worse when the query results get smaller: “Select 2, Project out 3” has slightly worse ADM scores than “Select 1, Project out 1”. Secondly,

ADM+STD		connect	adult	chessBig	letRecog	mushroom
Select 1	Project out 1	0.1 ± 0.01	0.1 ± 0.01	0.04 ± 0.01	0.1 ± 0.01	0.3 ± 0.02
	Project out 3	0.1 ± 0.02	0.1 ± 0.01	0.04 ± 0.03	0.1 ± 0.01	0.3 ± 0.16
Select 2	Project out 1	0.2 ± 0.01	0.1 ± 0.01	0.1 ± 0.03	0.04 ± 0.01	0.2 ± 0.04
	Project out 3	0.2 ± 0.02	0.1 ± 0.01	0.1 ± 0.03	0.04 ± 0.01	0.2 ± 0.05

Fig. 4. Results of Project-Select Queries

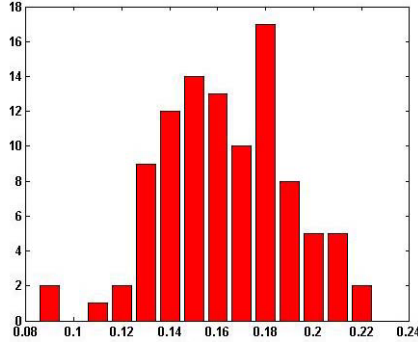


Fig. 5. Histogram of 100 Project-Select Queries on connect

Relative #candidates		connect	adult	chessBig	letRecog	mushroom
Select 1	Project out 1	0.01 ± 0.001	0.01 ± 0.002	0.21 ± 0.012	0.01 ± 0.001	0.01 ± 0.001
	Project out 3	0.01 ± 0.001	0.01 ± 0.004	0.26 ± 0.031	0.02 ± 0.004	0.01 ± 0.001
Select 2	Project out 1	0.01 ± 0.001	0.03 ± 0.003	0.76 ± 0.056	0.02 ± 0.002	0.03 ± 0.002
	Project out 3	0.01 ± 0.002	0.03 ± 0.008	0.96 ± 0.125	0.02 ± 0.004	0.03 ± 0.003

Fig. 6. Relative number of candidates for KRIMP*

even more importantly, combining algebra operators only degrades the ADM scores slightly. This can be seen if we compare the results for “Project out 3” on *letRecog* in Figure 2 with the “Select 1, Project out 3” and “Select 2, Project out 3” queries in Figure 4 on the same data set. These results are very comparable, the combination effect is small and mostly due to the smaller result sets. While not shown here, the same observation holds for the other data sets.

To give insight in the distribution of the ADM scores of the “Select 2, Project out 3” queries on the *connect* data set are given in Figure 5. From this figure we see that if we choose $\epsilon = 0.2$, $\delta = 0.08$. In other words, KRIMP* is a pretty good approximation of KRIMP. Almost always the approximation is less than 20% worse than the optimal result. The remaining question is, of course, how much faster is KRIMP*? This is illustrated in the table in Figure 6 This table gives the average number of candidates KRIMP* has to consider relative to those that the full KRIMP run has to consider. Since, both KRIMP* and KRIMP are linear in the number of candidates, this table shows that the speed-up is considerable; a factor of 100 is often attained; except for *chessBig* were the query results get

small and, thus, have few frequent item sets. The experiments are those that are reported on in Figure 4.

6.4 Select-Project-Join Queries

The results for the select-project-join queries are very much in line with the results reported on above. In fact, they are even better. Since the join leads to rather large results, the ADM score is almost always zero: in only 15 of the 400 experiments the score is non-zero (average of non-zero values is 1%). The speed-up is also in line with the numbers reported above, a factor of 100 is again often attained.

7 Discussion

As noted in the previous section, the speed-up of KRIMP* is easily seen. The number of candidates that KRIMP* has to consider is often a factor 100 smaller than those that the full KRIMP run has to consider. Given that the algorithm is linear in the number of candidates, this means a speed-up by a factor 100. In fact, one should also note that for KRIMP*, we do not have to run a frequent item set miner. In other words, in practice, using KRIMP* is even faster than suggested by the Speed-up scores.

But, how about the other goal: how good is the approximation? That is, how should one interpret ADM scores? Except for some outliers, ADM scores are below 0.2. That is, a full-fledged KRIMP run compresses the data set 20% better than KRIMP*. Is that good?

In a previous paper [17], we took two random samples from data sets, say D_1 and D_2 . Code tables CT_1 and CT_2 were induced from D_1 and D_2 respectively. Next we tested how well CT_i compressed D_j . For the four data sets also used in this paper, *Iris*, *Led7*, *Pima* and, *PageBlocks*, the “other” code table compressed 16% to 18% worse than the “own” code table; the figures for other data sets are in the same ball-park. In other words, an ADM score on these data sets below 0.2 is on the level of “natural variations” of the data distribution. Hence, given that the average ADM scores are often much lower we conclude that the approximation by KRIMP* is good.

In other words, the experiments verify our hypothesis: KRIMP* gives a fast and good approximation of KRIMP. The experiments show this for simple “project-select-join” queries, but as noticed with the results of the “project-select” queries, the effect of combining algebra operators is small. If the result set is large enough, the approximation is good.

8 Related Work

While there are, as far as the authors know, no other papers that study the same problem, the topic of this paper falls in the broad class of data mining with background knowledge. For, the model on the database, \mathcal{M}_{DB} , is used as

background knowledge in computing \mathcal{M}_Q . While a survey of this area is beyond the scope of this paper, we point out some papers that are related to one of the two aspects we are interested in, viz., speed-up and approximation.

A popular area of research in using background knowledge is that of constraints. Rather than trying to speed up the mining, the goal is often to produce models that adhere to the background knowledge. Examples are the use of constraints in frequent pattern mining, e.g. [3], and monotonicity constraints [9]. Note, however, that for frequent pattern mining the computation can be speeded up considerably if the constraints can be pushed into the mining algorithm [3]. So, speed-up is certainly a concern in this area. However, as far as we know approximation plays no role. The goal is still to find all patterns that satisfy the constraints.

Another use of background knowledge is to find unexpected patterns. In [12], e.g., Bayesian Networks of the data are used to estimate how surprising a frequent pattern is. In other words, the (automatically induced) background knowledge is used filter the output. In other words, speed-up is of no concern in this approach. Approximation clearly is, albeit in the opposite direction of ours: the more a pattern deviates from the global model, the more interesting it becomes. Whereas we would like that all patterns in the query result are covered by our approximate answer.

9 Conclusions

In this paper we introduce a new problem: given that we have a model induced from a database DB , does that help us in inducing a model on the result of a query Q on DB ? We formalise the problem for algorithms based on MDL and solve it for a particular algorithm, viz., our KRIMP algorithm. More in particular we introduce KRIMP*. This is actually the same as KRIMP, but it gets a restricted input. The code tables computed by KRIMP on DB are used as input, and thus as background knowledge, for KRIMP* on $Q(DB)$.

Extensive experiments with select-project-join queries show that KRIMP* approximates the results of KRIMP very well while it computes these results upto hundreds of times faster. Hence, the data analyst has a real choice: either get good result fast, or get optimal results slower.

References

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Inkeri Verkamo, A.: Fast discovery of association rules. In: *Advances in Knowledge Discovery and Data Mining*, pp. 307–328. AAAI, Menlo Park (1996)
2. Bickel, P.J., Hammel, E.A., O’Connell, J.W.: Sex bias in graduate admissions: Data from berkeley. *Science* 187(4175), 398–404 (1975)
3. Boulicaut, J.-F., Bykowski, A.: Frequent closures as a concise representation for binary data mining. In: Terano, T., Chen, A.L.P. (eds.) *PAKDD 2000*. LNCS, vol. 1805, pp. 62–73. Springer, Heidelberg (2000)

4. Cilibrasi, R., Vitanyi, P.: Automatic meaning discovery using google. *IEEE Transactions on Knowledge and Data Engineering* 19, 370–383 (2007)
5. Codd, E.F.: A relational model of data for large shared data banks. *Communications of the ACM* 13(6), 377–387 (1970)
6. Coenen, F.: The LUCS-KDD discretised/normalised ARM and CARM data library (2003), http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS_KDD_DN/
7. Cover, T.M., Thomas, J.A.: *Elements of Information Theory*, 2nd edn. John Wiley and Sons, Chichester (2006)
8. Faloutsos, C., Megalooikonomou, V.: On data mining, compression and kolmogorov complexity. In: *Data Mining and Knowledge Discovery*, vol. 15, pp. 3–20. Springer, Heidelberg (2007)
9. Feelders, A.J., van der Gaag, L.C.: Learning bayesian network parameters under order constraints. *Int. J. Approx. Reasoning* 42(1-2), 37–53 (2006)
10. Grünwald, P.D.: Minimum description length tutorial. In: Grünwald, P.D., Myung, I.J. (eds.) *Advances in Minimum Description Length*. MIT Press, Cambridge (2005)
11. Imielinski, T., Mannila, H.: A database perspective on knowledge discovery. *Communications of the ACM* 39(11), 58–64 (1996)
12. Jaroszewicz, S., Simovici, D.A.: Interestingness of frequent itemsets using bayesian networks as background knowledge. In: *Proceedings KDD*, pp. 178–186 (2004)
13. Koopman, A., Siebes, A.: Discovering relational item sets efficiently. In: *Proceedings SDM 2008*, pp. 585–592 (2008)
14. De Raedt, L.: A perspective on inductive databases. *SIGKDD Explorations* 4(2), 69–77 (2000)
15. Siebes, A., Vreeken, J., van Leeuwen, M.: Item sets that compress. In: *Proceedings of the SIAM Conference on Data Mining*, pp. 393–404 (2006)
16. van Leeuwen, M., Vreeken, J., Siebes, A.: Compression picks item sets that matter. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) *PKDD 2006*. LNCS (LNAI), vol. 4213, pp. 585–592. Springer, Heidelberg (2006)
17. Vreeken, J., van Leeuwen, M., Siebes, A.: Preserving privacy through data generation. In: *Proceedings of the IEEE International Conference on Data Mining*, pp. 685–690 (2007)