Francesco Bonchi
Jean-François Boulicaut (Eds.)

LNCS 3933

Knowledge Discovery in Inductive Databases

4th International Workshop, KDID 2005
Porto, Portugal, October 2005
Revised Selected and Invited Papers

Springer

Knowledge Discovery in Inductive Databases

4th International Workshop, KDID 2005
Porto, Portugal, October 3, 2005
Revised Selected and Invited Papers

Volume Editors

Francesco Bonchi
Pisa KDD Laboratory, ISTI - C.N.R.
Area della Ricerca di Pisa
Via Giuseppe Moruzzi, 1 - 56124 Pisa, Italy
E-mail: francesco.bonchi@isti.cnr.it

Jean-François Boulicaut
INSA Lyon, LIRIS CNRS UMR 5205
Bâtiment Blaise Pascal, 69621 Villeurbanne Cedex, France
E-mail: jean-francois.boulicaut@insa-lyon.fr

Library of Congress Control Number: 2006922625

CR Subject Classification (1998): H.2, I.2

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

ISSN 0302-9743
ISBN-10 3-540-33292-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-33292-3 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11733492 06/3142 5 4 3 2 1 0

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Preface

The 4th International Workshop on Knowledge Discovery in Inductive Databases (KDID 2005) was held in Porto, Portugal, on October 3, 2005 in conjunction with the 16th European Conference on Machine Learning and the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases.

Ever since the start of the field of data mining, it has been realized that the integration of the database technology into knowledge discovery processes was a crucial issue. This vision has been formalized into the inductive database perspective introduced by T. Imielinski and H. Mannila (CACM 1996, 39(11)). The main idea is to consider knowledge discovery as an extended querying process for which relevant query languages are to be specified. Therefore, inductive databases might contain not only the usual data but also inductive generalizations (e.g., patterns, models) holding within the data. Despite many recent developments, there is still a pressing need to understand the central issues in inductive databases. Constraint-based mining has been identified as a core technology for inductive querying, and promising results have been obtained for rather simple types of patterns (e.g., itemsets, sequential patterns). However, constraint-based mining of models remains a quite open issue. Also, coupling schemes between the available database technology and inductive querying proposals are not yet well understood. Finally, the definition of a general purpose inductive query language is still an on-going quest.

This workshop aimed to bring together database, machine learning and data mining researchers/practitioners who were interested in the numerous scientific and technological challenges that inductive databases offers. The workshop followed the previous three successful workshops organized in conjunction with ECML/PKDD: KDID 2002 held in Helsinki, Finland, KDID 2003 held in Cavtat-Dubrovnik, Croatia, and KDID 2004 held in Pisa, Italy. Its scientific program included seven regular presentations and four short communications, an invited talk by Carlo Zaniolo, and an invited "workshop-closing talk" by Arno Siebes. During the workshop, only informal proceedings were distributed. Most of the papers within this volume have been revised by the authors based on the comments from the initial referring stage and the discussion during the workshop. A few are invited chapters.

We wish to thank the invited speakers, all the authors of submitted papers, the Program Committee members and the ECML/PKDD 2005 Organization Committee. KDID 2005 was supported by the European project IQ "Inductive Queries for Mining Patterns and Models" (IST FET FP6-516169, 2005-2008).

December 2005

Francesco Bonchi
Jean-François Boulicaut

Organization

Program Chairs

Francesco Bonchi
Pisa KDD Laboratory
ISTI - C.N.R.
Italy
http://www-kdd.isti.cnr.it/~bonchi/

Jean-François Boulicaut
INSA Lyon
France
http://liris.cnrs.fr/~jboulica/

Program Committee

Hendrik Blockeel, *K.U. Leuven, Belgium*
Toon Calders, *University of Antwerp, Belgium*
Sašo Džeroski, *Jozef Stefan Institute, Slovenia*
Minos N. Garofalakis, *Bell Labs, USA*
Fosca Giannotti, *ISTI-C.N.R., Italy*
Bart Goethals, *University of Antwerp, Belgium*
Dominique Laurent, *LICP, Université de Cergy-Pontoise, France*
Giuseppe Manco, *ICAR-C.N.R., Italy*
Heikki Mannila, *University of Helsinki, Finland*
Rosa Meo, *University of Turin, Italy*
Taneli Mielikäinen, *University of Helsinki, Finland*
Katharina Morik, *University of Dortmund, Germany*
Céline Robardet, *INSA de Lyon, France*
Sunita Sarawagi, *KR School of Information Technology, IIT Bombay, India*
Arno Siebes, *University of Utrecht, The Netherlands*
Mohammed Zaki, *Rensselaer Polytechnic Institute, USA*
Carlo Zaniolo, *UCLA, USA*

Lecture Notes in Computer Science

For information about Vols. 1–3824

please contact your bookseller or Springer

Vol. 3933: F. Bonchi, J.-F. Boulicaut (Eds.), Knowledge Discovery in Inductive Databases. VIII, 251 pages. 2006.

Vol. 3928: J. Domingo-Ferrer, J. Posegga, D. Schreckling (Eds.), Smart Card Research and Advanced Applications. XI, 359 pages. 2006.

Vol. 3927: J. Hespanha, A. Tiwari (Eds.), Hybrid Systems: Computation and Control. XII, 584 pages. 2006.

Vol. 3925: A. Valmari (Ed.), Model Checking Software. X, 307 pages. 2006.

Vol. 3924: P. Sestoft (Ed.), Programming Languages and Systems. XII, 343 pages. 2006.

Vol. 3923: A. Mycroft, A. Zeller (Eds.), Compiler Construction. XIII, 277 pages. 2006.

Vol. 3922: L. Baresi, R. Heckel (Eds.), Fundamental Approaches to Software Engineering. XIII, 427 pages. 2006.

Vol. 3921: L. Aceto, A. Ingólfsdóttir (Eds.), Foundations of Software Science and Computation Structures. XV, 447 pages. 2006.

Vol. 3920: H. Hermanns, J. Palsberg (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. XIV, 506 pages. 2006.

Vol. 3916: J. Li, Q. Yang, A.-H. Tan (Eds.), Data Mining for Biomedical Applications. VIII, 155 pages. 2006. (Sublibrary LNBI).

Vol. 3915: R. Nayak, M.J. Zaki (Eds.), Knowledge Discovery from XML Documents. VIII, 105 pages. 2006.

Vol. 3909: A. Apostolico, C. Guerra, S. Istrail, P.A. Pevzner (Eds.), Research in Computational Molecular Biology. XVII, 612 pages. 2006. (Sublibrary LNBI).

Vol. 3907: F. Rothlauf, J. Branke, S. Cagnoni, E. Costa, C. Cotta, R. Drechsler, E. Lutton, P. Machado, J.H. Moore, J. Romero, G.D. Smith, G. Squillero, H. Takagi (Eds.), Applications of Evolutionary Computing. XXIV, 813 pages. 2006.

Vol. 3906: J. Gottlieb, G.R. Raidl (Eds.), Evolutionary Computation in Combinatorial Optimization. XI, 293 pages. 2006.

Vol. 3905: P. Collet, M. Tomassini, M. Ebner, S. Gustafson, A. Ekárt (Eds.), Genetic Programming. XI, 361 pages. 2006.

Vol. 3904: M. Baldoni, U. Endriss, A. Omicini, P. Torroni (Eds.), Declarative Agent Languages and Technologies III. XII, 245 pages. 2006. (Sublibrary LNAI).

Vol. 3903: K. Chen, R. Deng, X. Lai, J. Zhou (Eds.), Information Security Practice and Experience. XIV, 392 pages. 2006.

Vol. 3901: P.M. Hill (Ed.), Logic Based Program Synthesis and Transformation. X, 179 pages. 2006.

Vol. 3899: S. Frintrop, VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search. XIV, 216 pages. 2006. (Sublibrary LNAI).

Vol. 3897: B. Preneel, S. Tavares (Eds.), Selected Areas in Cryptography. XI, 371 pages. 2006.

Vol. 3896: Y. Ioannidis, M.H. Scholl, J.W. Schmidt, F. Matthes, M. Hatzopoulos, K. Boehm, A. Kemper, T. Grust, C. Boehm (Eds.), Advances in Database Technology - EDBT 2006. XIV, 1208 pages. 2006.

Vol. 3895: O. Goldreich, A.L. Rosenberg, A.L. Selman (Eds.), Theoretical Computer Science. XII, 399 pages. 2006.

Vol. 3894: W. Grass, B. Sick, K. Waldschmidt (Eds.), Architecture of Computing Systems - ARCS 2006. XII, 496 pages. 2006.

Vol. 3890: S.G. Thompson, R. Ghanea-Hercock (Eds.), Defence Applications of Multi-Agent Systems. XII, 141 pages. 2006. (Sublibrary LNAI).

Vol. 3889: J. Rosca, D. Erdogmus, J.C. Príncipe, S. Haykin (Eds.), Independent Component Analysis and Blind Signal Separation. XXI, 980 pages. 2006.

Vol. 3888: D. Draheim, G. Weber (Eds.), Trends in Enterprise Application Architecture. IX, 145 pages. 2006.

Vol. 3887: J.R. Correa, A. Hevia, M. Kiwi (Eds.), LATIN 2006: Theoretical Informatics. XVI, 814 pages. 2006.

Vol. 3886: E.G. Bremer, J. Hakenberg, E.-H.(S.) Han, D. Berrar, W. Dubitzky (Eds.), Knowledge Discovery in Life Science Literature. XIV, 147 pages. 2006. (Sublibrary LNBI).

Vol. 3885: V. Torra, Y. Narukawa, A. Valls, J. Domingo-Ferrer (Eds.), Modeling Decisions for Artificial Intelligence. XII, 374 pages. 2006. (Sublibrary LNAI).

Vol. 3884: B. Durand, W. Thomas (Eds.), STACS 2006. XIV, 714 pages. 2006.

Vol. 3881: S. Gibet, N. Courty, J.-F. Kamp (Eds.), Gesture in Human-Computer Interaction and Simulation. XIII, 344 pages. 2006. (Sublibrary LNAI).

Vol. 3880: A. Rashid, M. Aksit (Eds.), Transactions on Aspect-Oriented Software Development I. IX, 335 pages. 2006.

Vol. 3879: T. Erlebach, G. Persinao (Eds.), Approximation and Online Algorithms. X, 349 pages. 2006.

Vol. 3878: A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing. XVII, 589 pages. 2006.

Vol. 3877: M. Detyniecki, J.M. Jose, A. Nürnberger, C. J. '. van Rijsbergen (Eds.), Adaptive Multimedia Retrieval: User, Context, and Feedback. XI, 279 pages. 2006.

Vol. 3876: S. Halevi, T. Rabin (Eds.), Theory of Cryptography. XI, 617 pages. 2006.

Vol. 3875: S. Ur, E. Bin, Y. Wolfsthal (Eds.), Hardware and Software, Verification and Testing. X, 265 pages. 2006.

Vol. 3874: R. Missaoui, J. Schmidt (Eds.), Formal Concept Analysis. X, 309 pages. 2006. (Sublibrary LNAI).

Vol. 3873: L. Maicher, J. Park (Eds.), Charting the Topic Maps Research and Applications Landscape. VIII, 281 pages. 2006. (Sublibrary LNAI).

Vol. 3872: H. Bunke, A. L. Spitz (Eds.), Document Analysis Systems VII. XIII, 630 pages. 2006.

Vol. 3870: S. Spaccapietra, P. Atzeni, W.W. Chu, T. Catarci, K.P. Sycara (Eds.), Journal on Data Semantics V. XIII, 237 pages. 2006.

Vol. 3869: S. Renals, S. Bengio (Eds.), Machine Learning for Multimodal Interaction. XIII, 490 pages. 2006.

Vol. 3868: K. Römer, H. Karl, F. Mattern (Eds.), Wireless Sensor Networks. XI, 342 pages. 2006.

Vol. 3866: T. Dimitrakos, F. Martinelli, P.Y.A. Ryan, S. Schneider (Eds.), Formal Aspects in Security and Trust. X, 259 pages. 2006.

Vol. 3865: W. Shen, K.-M. Chao, Z. Lin, J.-P.A. Barthès, A. James (Eds.), Computer Supported Cooperative Work in Design II. XII, 659 pages. 2006.

Vol. 3863: M. Kohlhase (Ed.), Mathematical Knowledge Management. XI, 405 pages. 2006. (Sublibrary LNAI).

Vol. 3862: R.H. Bordini, M. Dastani, J. Dix, A.E.F. Seghrouchni (Eds.), Programming Multi-Agent Systems. XIV, 267 pages. 2006. (Sublibrary LNAI).

Vol. 3861: J. Dix, S.J. Hegner (Eds.), Foundations of Information and Knowledge Systems. X, 331 pages. 2006.

Vol. 3860: D. Pointcheval (Ed.), Topics in Cryptology – CT-RSA 2006. XI, 365 pages. 2006.

Vol. 3858: A. Valdes, D. Zamboni (Eds.), Recent Advances in Intrusion Detection. X, 351 pages. 2006.

Vol. 3857: M.P.C. Fossorier, H. Imai, S. Lin, A. Poli (Eds.), Applied Algebra, Algebraic Algorithms and Error-Correcting Codes. XI, 350 pages. 2006.

Vol. 3855: E. A. Emerson, K.S. Namjoshi (Eds.), Verification, Model Checking, and Abstract Interpretation. XI, 443 pages. 2005.

Vol. 3854: I. Stavrakakis, M. Smirnov (Eds.), Autonomic Communication. XIII, 303 pages. 2006.

Vol. 3853: A.J. Ijspeert, T. Masuzawa, S. Kusumoto (Eds.), Biologically Inspired Approaches to Advanced Information Technology. XIV, 388 pages. 2006.

Vol. 3852: P.J. Narayanan, S.K. Nayar, H.-Y. Shum (Eds.), Computer Vision – ACCV 2006, Part II. XXXI, 977 pages. 2006.

Vol. 3851: P.J. Narayanan, S.K. Nayar, H.-Y. Shum (Eds.), Computer Vision – ACCV 2006, Part I. XXXI, 973 pages. 2006.

Vol. 3850: R. Freund, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing. IX, 371 pages. 2006.

Vol. 3849: I. Bloch, A. Petrosino, A.G.B. Tettamanzi (Eds.), Fuzzy Logic and Applications. XIV, 438 pages. 2006. (Sublibrary LNAI).

Vol. 3848: J.-F. Boulicaut, L. De Raedt, H. Mannila (Eds.), Constraint-Based Mining and Inductive Databases. X, 401 pages. 2006. (Sublibrary LNAI).

Vol. 3847: K.P. Jantke, A. Lunzer, N. Spyratos, Y. Tanaka (Eds.), Federation over the Web. X, 215 pages. 2006. (Sublibrary LNAI).

Vol. 3846: H. J. van den Herik, Y. Björnsson, N.S. Netanyahu (Eds.), Computers and Games. XIV, 333 pages. 2006.

Vol. 3845: J. Farré, I. Litovsky, S. Schmitz (Eds.), Implementation and Application of Automata. XIII, 360 pages. 2006.

Vol. 3844: J.-M. Bruel (Ed.), Satellite Events at the MoDELS 2005 Conference. XIII, 360 pages. 2006.

Vol. 3843: P. Healy, N.S. Nikolov (Eds.), Graph Drawing. XVII, 536 pages. 2006.

Vol. 3842: H.T. Shen, J. Li, M. Li, J. Ni, W. Wang (Eds.), Advanced Web and Network Technologies, and Applications. XXVII, 1057 pages. 2006.

Vol. 3841: X. Zhou, J. Li, H.T. Shen, M. Kitsuregawa, Y. Zhang (Eds.), Frontiers of WWW Research and Development - APWeb 2006. XXIV, 1223 pages. 2006.

Vol. 3840: M. Li, B. Boehm, L.J. Osterweil (Eds.), Unifying the Software Process Spectrum. XVI, 522 pages. 2006.

Vol. 3839: J.-C. Filliâtre, C. Paulin-Mohring, B. Werner (Eds.), Types for Proofs and Programs. VIII, 275 pages. 2006.

Vol. 3838: A. Middeldorp, V. van Oostrom, F. van Raamsdonk, R. de Vrijer (Eds.), Processes, Terms and Cycles: Steps on the Road to Infinity. XVIII, 639 pages. 2005.

Vol. 3837: K. Cho, P. Jacquet (Eds.), Technologies for Advanced Heterogeneous Networks. IX, 307 pages. 2005.

Vol. 3836: J.-M. Pierson (Ed.), Data Management in Grids. X, 143 pages. 2006.

Vol. 3835: G. Sutcliffe, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. XIV, 744 pages. 2005. (Sublibrary LNAI).

Vol. 3834: D.G. Feitelson, E. Frachtenberg, L. Rudolph, U. Schwiegelshohn (Eds.), Job Scheduling Strategies for Parallel Processing. VIII, 283 pages. 2005.

Vol. 3833: K.-J. Li, C. Vangenot (Eds.), Web and Wireless Geographical Information Systems. XI, 309 pages. 2005.

Vol. 3832: D. Zhang, A.K. Jain (Eds.), Advances in Biometrics. XX, 796 pages. 2005.

Vol. 3831: J. Wiedermann, G. Tel, J. Pokorný, M. Bieliková, J. Štuller (Eds.), SOFSEM 2006: Theory and Practice of Computer Science. XV, 576 pages. 2006.

Vol. 3830: D. Weyns, H. V.D. Parunak, F. Michel (Eds.), Environments for Multi-Agent Systems II. VIII, 291 pages. 2006. (Sublibrary LNAI).

Vol. 3829: P. Pettersson, W. Yi (Eds.), Formal Modeling and Analysis of Timed Systems. IX, 305 pages. 2005.

Vol. 3828: X. Deng, Y. Ye (Eds.), Internet and Network Economics. XVII, 1106 pages. 2005.

Vol. 3827: X. Deng, D.-Z. Du (Eds.), Algorithms and Computation. XX, 1190 pages. 2005.

Vol. 3826: B. Benatallah, F. Casati, P. Traverso (Eds.), Service-Oriented Computing - ICSOC 2005. XVIII, 597 pages. 2005.

Table of Contents

Invited Papers

Contributed Papers

Data Mining in Inductive Databases

Arno Siebes

Universiteit Utrecht,
Department of Computer Science,
Padualaan 14, 3584CH Utrecht, The Netherlands
arno@cs.uu.nl

Abstract. Ever since the seminal paper by Imielinski and Mannila [11], inductive databases have been a constant theme in the data mining literature. Operationally, such an inductive database is a database in which models and patterns are first class citizens.

In the extensive literature on inductive databases there is at least one consequence of this operational definition that is conspicuously missing. That is the question: if we have models and patterns in our inductive database, how does this help to discover other models and patterns? This question is the topic of this paper.

1 Introduction

Ever since the start of research in data mining, it has been clear that data mining, and more general the KDD process, should be merged into DBMSs. Since the seminal paper by Imielinski and Mannila [11], the so-called *inductive databases* have been a constant theme in data mining research, with its own series of workshops.

Perhaps surprisingly, there is no formal definition of what an inductive database actually is. In [30] it is stated that it might be too early for such a definition, given the issues I raise in this paper, I tend to agree with this opinion. Still, we need some sort of shared concept of an inductive database.

Mostly, people think of inductive databases in analogy with *deductive databases;* an analogy that is not without its weaknesses as we will see later. I take a slightly different angle, viz., an inductive database is a database in which the discovered models and patterns are first class citizens. That is, we should be able to treat models and patterns as any other data object. This very operational definition of an inductive database is our guiding principle in this paper.

Research in inductive databases is mainly focused on two aspects:

1. The integration of data mining and DBMSs, which itself encompasses two, not necessarily disjunct, main topics,
   (a) database support for data mining, or, the integration of data mining algorithms into a DBMS and
   (b) integrating data mining into standard query languages like SQL.
2. Querying models and patterns.

F. Bonchi and J.-F. Boulicaut (Eds.): KDID 2005, LNCS 3933, pp. 1–23, 2006.

These are clearly important aspects of an inductive database and surprisingly hard to do well to boot. However, they are not all there is for an inductive database. This alone doesn't make models and patterns first class cizitizens. In fact, the most important aspect of an inductive database is missing: the data mining!

This might seem a strange statement since both main topics are deeply concerned with data mining. The first one is all about making data mining no different from other, more standard, queries in, e.g., SQL. The second one is about storing the models and patterns that result from mining queries in the database and querying those results with constraints.

This is very much in line what would would expect for inductive databases, especially if one compares with deductive databases [23]. For, except for the architectural issues of integration, these topics can be nicely formalised in first order logic [30]. Moreover, pushing the query constraints into the mining algorithm is a natural extension of standard relational query optimisation.

So, the analogy of inductive databases and deductive databases is certainly a fruitful one. However, this analogy doesn't tell the whole story.

> *In deductive databases, the Intentional Database (the rules) is a static component. Queries result in new facts, not in new rules.*

In data mining, however, we are not interested in new facts, we want to discover new models and patterns. If we already have models and patterns in our database, a natural question is: does this help? So, a central question for data mining in inductive databases that is not covered by the analogy with deductive databases is:

> *How do the models and patterns we have already discovered help us in discovering other models and patterns?*

This question is the topic of this paper. Given that it is an invited paper, I feel free to raise more questions than I answer. The goal is to point to new research questions, not to answer them.

I discuss three aspects of this question in this paper:

Relational Algebra: Models and patterns are tightly connected to the data tables they have been induced from. In a DBMS we can construct new tables from existing ones using relational algebra. It would be useful if these algebraic manipulations could be *lifted* to the models and patterns. It would give us models and patterns for free.

Models for Models: If we have already induced models and/or patterns from a data table, does this help us in the induction of other models and/or patterns from that same table?

Models on Models: If models and patterns are first class citizens in our database, we should be able to mine a collection of models or a collection of patterns. How can we do this and, perhaps more importantly, does this make sense?

The questions are discussed both from a pattern and from a model perspective. The patterns used are mostly frequent item sets, the models mostly Bayesian networks.

This paper is not meant to be a survey paper, i.e., it is in no way complete. For some if not all off the (sub-)questions the paper discusses there is far more published literature than is mentioned or discussed. The choices made are mostly based on what I thought would nicely illustrate my point. Moreover, there is a clear bias to papers I have been involved in[1].

The road map of this paper is as follows. In Section 2, some preliminaries are introduced. The next three sections discuss the three sub-questions introduced above. Finally, in the final section concludes the paper by formulating a couple of research topics I feel are important for inductive databases.

2 Preliminaries

In order to keep our discussion simple, we assume a binary database. In fact, in general we assume the database contains one binary table. Only in the case of relational algebra we assume multiple tables when the operators require more than one input table.

In the case of frequent item sets, we use the standard terminology of items and transactions. In the Bayesian networks case, we will mostly call them variables and tuples. Given the simple relationship between a binary table and a set of transactions, this should not confuse the reader. We discuss both contexts briefly.

2.1 Models and Patterns

In the introduction we already used both the terms *model* and *patterns*. Both terms are probably familiar to all data miner, although I wouldn't know a formal definition of either. The goal of this subsection is not to present such a definition, but to point out the most important difference between the two.

Models describe the whole database, they are *global*. Patterns describe *local* phenomena in the database. In [8], a pattern is defined by:

$$data = model + patterns + random$$

In [22] this definition extended with three characteristics, viz.,

- Local patterns cover small parts of the data space.
- Local patterns deviate from the distribution of which they are part.
- Local patterns show some internal structure.

In other words, while a model tries to capture the whole distribution, patterns describe small sub-spaces were this distribution differs markedly from the global picture.

Given this distinction, it seems obvious that models that have been discovered offer more aid in the discovery of other models and patterns than discovered patterns can. While most of the examples in this paper agree with this observation, this is not true for all of them.

[1] In other words, this paper is blatant self promotion!

2.2 Frequent Item Sets

The problem of frequent item set mining [1] can be described as follows. The basis is a set of items $\mathcal{I}$, e.g., the items for sale in a store; $|\mathcal{I}| = n$. A transaction $t \in \mathcal{P}(\mathcal{I})$ is a set of items, e.g., representing the set of items a client bought at that store. A database over $\mathcal{I}$ is simply a set of transactions, e.g., the different sale transactions in the store on a given day. An item set $I \subset \mathcal{I}$ occurs in a transaction $t \in db$ iff $I \subseteq t$. The *support* of I in db, denoted by $supp_{db}(I)$ is the number of transactions in the database in which t occurs. The problem of frequent item set mining is: given a threshold *min-sup*, determine all item sets I such that $supp_{db}(I) \geq$ *min-sup*. These *frequent item sets* represent, e.g., sets of items customers buy together often enough.

Association Rules are generated from these frequent item sets. If X is a frequent item set and $Y \subset X$,

$$X \setminus Y \rightarrow Y$$

is an association rule. Its *confidence* is defined as $\frac{supp_{db}(X)}{supp_{db}(X \setminus Y)}$. For association rule mining, one has the *min-sup* threshold for support and a *min-conf* threshold for the confidence of a rule. The problem is to find all rules that satisfy both minimal thresholds.

Often there are other interestingness measures used to reduce the number of discovered association rules. The one that is most often used is the *lift*. The lift of a rule $X \rightarrow Y$ is defined as $\frac{conf_{db}(X \rightarrow Y)}{supp_{db}(Y)}$.

If the database consists of a set of *sequences of events*, we can define analogous concepts [20]. An episode is simply a sequence of events. An episode E occurs in a sequence S if deleting events from S yields E; note that an episode E may occur multiple times in S. The support of an episode is the number of times an episode occurs in the database. With a minimal support threshold, the problem is: find all frequent episodes.

2.3 Bayesian Networks

Bayesian networks by now are widely accepted as powerful tools for representing and reasoning with uncertainty in decision-support systems. A Bayesian network is a concise model of a joint probability distribution over a set of stochastic variables [29]; it consists of a directed acyclic graph that captures the qualitative dependence structure of the distribution and a numerical part that specifies conditional probability distributions for each variable given its parents in the graph. Since a Bayesian network defines a unique distribution, it provides for computing any probability of interest over its variables.

A *Bayesian network* is a concise representation of a joint probability distribution over a set of stochastic variables $\mathbf{X} = (X_1, \ldots, X_n)$. The network consists of a directed acyclic graph in which each node corresponds with a variable and the arcs capture the qualitative dependence structure of the distribution. The network further includes a number of conditional probabilities, or *parameters*, $p(X_i \mid \mathbf{X}_{\pi(i)})$ for each variable X_i given its parents $\mathbf{X}_{\pi(i)}$ in the graph.

The graphical structure and associated probabilities with each other represent a unique joint probability distribution $\Pr(\mathbf{X})$ over the variables involved, which is factorised according to

$$\Pr(\mathbf{X}) = \prod_{i=1}^{n} p(X_i \mid \mathbf{X}_{\pi(i)})$$

There are numerous algorithms that induce Bayesian networks from data, see, e.g., [24].

3 Lifting Relational Algebra

The question is: can we extend the relational operators to models and patterns? By focusing on the relational algebra, we have already a syntax. How about the semantics? For example, what is the join of two models? Obviously there are many ways in which this can be defined and the choice for a particular semantics is perhaps the most important factor for our practical view on inductive databases. Our choice is to *lift* the standard operators to models. Lifting means that we want our new operator to construct a new model or a new collection of patterns from the input models or patterns only. That is, without consulting the database.

Note, we use the *bag* semantics for relational algebra rather than the *set* semantics that are more standard in database theory. The reason is that the databases we want to mine adhere to the bag semantics since this is the underlying principle of each available DBMS.

3.1 Select

The relational algebra operator σ (select) is a mapping:

$$\sigma : \mathcal{B}(D) \rightarrow \mathcal{B}(D)$$

in which $\mathcal{B}(D)$ denotes all possible bags over domain D.

Lifting means that we are looking for an operator $\sigma_{(D,\mathcal{A})}$ that makes the diagram in figure 1 commute: Such diagrams are well-known in , e.g., category theory [3] and the standard interpretation is:

$$\mathcal{A} \circ \sigma = \sigma_{(D,\mathcal{A})} \circ \mathcal{A}$$

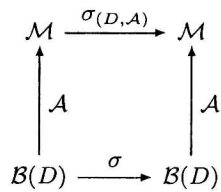

Fig. 1. Lifting the selection operator

In other words, first inducing the model using algorithm $\mathcal{A}$ followed by the application of the *lifted* selection operator $\sigma_{(D,\mathcal{A})}$ yields the same result as first applying the *standard* selection operator σ followed by induction with algorithm $\mathcal{A}$.

For algorithms that do compute the optimal result, such a strict interpretation of the diagram seems reasonable. However, many algorithms rely on heuristic search. In such cases, it doesn't seem reasonable at all to require this strict reading of the diagram. Rather we settle for a *reasonably good approximation.* That is, the lifted selection operator doesn't have to result in a locally optimal model, but it should be close to one[2]. If not explicitly stated otherwise, we will use commutation in this loose sense.

Frequent Item Sets. The three basic selections are $\sigma_{I=0}$, $\sigma_{I=1}$, and $\sigma_{I_1=I_2}$. More complicated selections can be made by conjunctions of these basic comparisons. We look at the different basic selections in turn.

First consider $\sigma_{I=0}$. If it is applied to the database, all transactions in which I occurs are removed from the database. Hence, all item sets that contain I get a frequency of zero in the resulting database. For those item sets in which I doesn't occur, we have to compute which part of their support consists of transactions in which I does occur and subtract that number. Hence, we have:

$$freq_{\sigma_{I=0}(db)}(J) = \begin{cases} 0 & \text{if } I \in J, \\ freq_{db}(J) - freq_{db}(J \cup \{I\}) & \text{else.} \end{cases}$$

If we apply $\sigma_{I=1}$ to the database, all transactions in which I doesn't occur are removed from the database. In other words, the frequency of item sets that contain I doesn't change. For those item sets that do not contain, the frequency is given by those transactions that also contained I. Hence, we have:

$$freq_{\sigma_{I=1}(db)}(J) = \begin{cases} freq_{db}(J) & \text{if } I \in J, \\ freq_{db}(J \cup \{I\}) & \text{else.} \end{cases}$$

If we apply $\sigma_{I_1=I_2}$ to the database, the only transactions that remain are those that either contain both I_1 and I_2 or neither. In other words, for frequent item sets that contain both the frequency remains the same. For all others, the frequency changes. For those item sets J that contain just one of the I_i the frequency will be the frequency of $J \cup \{I_1, I_2\}$. For those that contain neither of the I_i, we have to correct for those transactions that contain one of the I_i in their support. If we denote this by $freq_{db}(J\neg I_1 \neg I_2)$ (a frequency that can be easily computed) We have:

$$freq_{\sigma_{I_1=I_2}(db)}(J) = \begin{cases} freq_{db}(J \cup \{I_1, I_2\}) & \text{if } \{I_1, I_2\} \cap J \neq \emptyset, \\ freq_{db}(J\neg I_1 \neg I_2) & \text{else.} \end{cases}$$

Clearly, we can also "lift" conjunctions of the basic selections, simply process one at the time. So, in principle, we can lift all selections for frequent item sets.

[2] Given the nature of this paper, I am not going to attempt to formalise this notion. I hope the reader has some idea of what I mean.

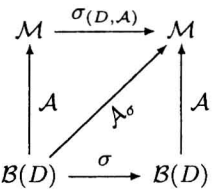

Fig. 2. Lifting selections for succinct constraints

But only in principle, because we need the frequency of item sets that are *not necessarily frequent.* Frequent item sets are a lossy model (not all aspects of the data distribution are modelled) and that can have its repercussions: in general the lifting will *not* be commutative. In our loose sense of "commutativity", the situation is slightly better. For, we can give bounds for the resulting frequencies.

We haven't mentioned constraints [25] so far. Constraints in frequent item set mining are the pre-dominant way to select a subset of the frequent item sets. That is exactly, why we haven't mentioned them so far. In general the constraints studied do not correspond to selections on the database. The exception is the class of *succinct anti-monotone constraints* introduced in [26]. For these constraints there is such a selection (that is what succinct means) and the constraint can be pushed into the algorithm. This means we get the commutative diagram in figure 2. Note that in this case we know that the diagonal arrow makes the bottom right triangle commute in the strict sense of the word. For the upper left triangle, as well as the square, our previous analysis remains true.

Bayesian Networks. The selections $\sigma_{A=0}$ and $\sigma_{A=1}$ in Bayesian networks are a simple example of *partial knowledge*: if we know that variable A has value 1, what can we infer about the values of the other attributes? There are standard inference algorithms [24] for this problem that allow us to propagate this partial knowledge After that, we can remove the variables that are now fixed, such as A. For example:

$$B \leftarrow A \rightarrow C \text{ transforms to } B \quad C$$

That is, in this example B and C become independent after the selection. In the case of induced dependencies, we have to be careful to add the necessary induced arcs, such as:

$$B \rightarrow A \leftarrow C \text{ transforms to } B \rightarrow C$$

Note that for this simple case, the inference algorithms are polynomial.

The selection $\sigma_{A=B}$ is slightly more complicated. There are three cases we need to consider.

Firstly, if A and B are in disconnected components of the graph, we can simply add an arc from A to B[3]. Furthermore, we have to update the (conditional) probability table of B such that it gives probability zero to those cases were

[3] Or from B to A, without a causal interpretation this doesn't matter.