Bulletin of the Technical Committee on

Data Engineering

March 1998 Vol. 21 No. 1

IEEE Computer Society

Letters

Special Issue on Mining of Large Datasets

Announcements and Notices

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

Membership in the TC on Data Engineering (http: www. is open to all current members of the IEEE Computer Society who are interested in database systems.

The web page for the Data Engineering Bulletin is http://www.research.microsoft.com/research/db/debull. The web page for the TC on Data Engineering is http://www.ccs.neu.edu/groups/IEEE/tcde/index.html.

Letter from the new TC Chair

When I talked to Rakesh Agrawal, the outgoing TCDE Chair, he began by telling me that his foremost accomplishment as Chair was to persuade David Lomet to edit the Bulletin. David has done an excellent job, developing LaTeX tools to put the Bulletin on line and make its publication a (relatively) straightforward process. He has also done well in choosing associate editors who each pick a topic and invite articles from the best researchers and practitioners in that area to write on that topic. This has made the Data Engineering Bulletin a more readable and more interesting and more important publication than many other, supposedly more prestigious journals. David has agreed to continue as editor for at least one more year. I would like to thank him for his work in this area.

In addition to publishing the Bulletin, the TCDE has developed an association with the VLDB (Very Large Databases) conference. I believe Rakesh would name this as his second most important contribution. Because of his hard work in establishing connections with the VLDB community, the VLDB conference now has "in-cooperation with IEEE" status. I would like to thank Rakesh both for enlisting David as editor and for establishing the VLDB conference connection.

The new TCDE Executive committee will try to build on the accomplishments of the previous one. We will try to maintain the quality of the Bulletin and the association with VLDB. We have begun building ties with the ACM SIGMOD. I have created a new position on the TCDE Executive Committee—SIGMOD liason. Meral Ozsoyoglu, the VP of ACM SIGMOD has agreed to take the position and has been designated "IEEE liason" within SIGMOD. The new SIGMOD executive committee under Rick Snodgrass seems more than amenable to forging new relationships and cooperating on projects.

We have also begun a WWW presence. Our new web page is at http://www/ccs.neu.edu/groups/IEEE/tcde

The new area coordinators, Svein-Olaf Hvasshovd in Europe, Masaru Kitsuregawa in Asia, and Ron Sacks-Davis in Australia have been asked to make area web pages as well. We will post these web page addresses when they are ready. Our new Vice-President, Erich Neuhold, has indicated a special interest in web-based activities. Please feel free to communicate with Erich or with others on the TCDE Executive Committee about suggestions for our web pages.

I would also like to thank Tracy Woods and Ronald Waxman of the IEEE for their patience in these first few months while I have been getting up to speed on TC procedures.

Betty Salzberg
Northeastern University

Letter from the Editor-in-Chief

New Technical Committee Chair and Executive Board

The current issue of the Bulletin includes the results of the recent Technical Committee on Data Engineering election for TC Chair. Betty Salzberg of Northeastern, a prior Associate Editor of the Bulletin, won election and has appointed a new executive committee. Her committee appointments are listed in her letter, which precedes this letter, and are included on the front inside cover of this and future issues of the Bulletin. I want to congratulate Betty on her election to TC Chair and welcome her and her new appointments to the Technical Committee's Executive Committee.

Changing Editorial Staff

This issue also marks on-going changes in the Bulletin's Editorial Board. Joe Hellerstein, who editted the December, 1997 issue on "Data Reduction Techniques" has completed his two year term. Joe has done a fine job both on the December issue and his prior issue on "Query Processing for Non-standard Data". As I have frequently written during such transitions, the Bulletin depends on the hard work and technical expertise of our Associate Editors, and I want to thank Joe for contributing both of those to the Bulletin.

With Joe's departure, I have now appointed Amr El Abbadi as a Bulletin Associate Editor. Amr has been at the University of California, Santa Barbara since 1987 and is currently an Associate Professor in the Computer Science Department. He has a Ph.D. in Computer Science from Cornell University. Amr's main research interests are in developing basic mechanisms for supporting distributed information management systems, including databases, digital libraries, workflow systems, and geographic information systems. Earlier work concentrated on the development of protocols and algorithms that ensure high availability and fault tolerance in such systems. I welcome Amr to the Bulletin and look forward to working with him in the future.

This Issue

As even a cursory check will reveal, data mining has become *the* hot area of database research. After years of focusing on query processing in which the idea is to make a given user query perform well, a good part of the database community has now refocused on how to extract useful information without the presence of a user query.

This new data mining activity has a longer history in the AI community. Two factors make it an interesting area for the database community.

1. Much of the data being mined was at least originally captured in a database system, and indeed, a fair bit of it is mined directly from a database system.

2. Our database community has unique expertise in the area of scalable algorithms. Given the vast amounts of data being mined, controlling the number and parallelism of the data scans is essential to the data mining enterprise.

This issue, brought together by Daniel Barbará, captures some of the excitement, controversy and diversity of this increasingly important subfield. I want to thank Daniel for his efforts, which I know directly have involved considerable difficulties. I'm sure you will find the issue both interesting and useful.

David Lomet
Microsoft Research

Letter from the Special Issue Editor

Data mining has become a booming topic of research in our community. There are many reasons behind this fact. First, the area is extremely fertile with research problems. Secondly, technology has made it so easy to gather data that now companies and organizations find themselves inundated with data, and eager to extract information from it.

Despite the extraordinary number of papers already published in data mining, we can still argue that the field is in its infancy. Much of the efforts of the database community have been devoted to the issue of finding association rules in datasets, and relatively less attention has been given to other classical data analysis methods such as classification.

This issue of the IEEE Data Engineering Bulletin brings a series of articles that focus on several aspects of data mining. The first paper, written by Surajit Chaudhuri, stresses the fact that our role in the database community must be that of finding techniques to make known data analysis techniques scale over large datasets. He offers guidelines to build **SQL-aware** data mining systems: that is systems that integrate well with an SQL backend.

Clustering, an important technique in data mining is the subject of the next two articles. The second paper in the issue, written by G.D. Ramkumar and Arun Swami, focuses in clustering as an important data mining tool. Clustering is usually performed by using distance functions to measure similarity between data points. However, distance functions are not always available in practice., The authors describe a new method for clustering without distance functions. The third paper in the issue, written by Eui-Hong Han, George Karypis, Vipin Kumar and Bamshad Mobasher presents a method to perform clustering in a high dimensional space.

The next paper in the issue, written by Charu Aggarwal and Philip Yu, surveys the itemset method for the discovery of association rules, discusses its strengths and weaknesses and proposes alternative techniques to the traditional generation methods.

The fourth paper, written by Claudio Bettini, X. Sean Wang and Sushil Jajodia, discusses a particular application domain: mining temporal relationships, such as ”which pairs of events occur frequently every week.” They introduce heuristics to find these temporal associations efficiently.

Our last paper, by Usama Fayyad, provides an overview and survey of knowledge discovery methods and problems. The paper also provides useful references to the knowledge discovery literature. Usama's paper should be of interest as it represents the views of a member of the knowledge discovery community, views that are not often heard in the database research community.

Data Mining is a fruitful area of research with an abundant wealth of practical applications. We hope you find the articles in this issue as a good and stimulating sample of the research being done currently in this area.

Daniel Barbará
George Mason University

Data Mining and Database Systems: Where is the Intersection?

Surajit Chaudhuri
Microsoft Research
Email: surajitc@microsoft.com

1 Introduction

The promise of decision support systems is to exploit enterprise data for competitive advantage. The process of deciding what data to collect and how to clean such data raises nontrivial issues. However, even after a data warehouse has been set up, it is often difficult to analyze and assimilate data in a warehouse. OLAP takes an important first step at the problem by allowing us to view data multidimensionally as a giant spreadsheet with sophisticated visual tools to browse and query the data (See [3] for a survey). Data Mining promises a giant leap over OLAP where instead of a power OLAP user navigating data, the mining tools will automatically discover interesting patterns. Such functionality will be very useful in enterprise databases that are characterized by a large schema as well as large number of rows.

Data Mining involves data analysis techniques that have been used by statisticians and machine learning community for quite some time now (generically referred to data analysts in this paper). This raises the question as to what role, if any, database systems research may contribute to area of data mining. In this article, I will try to present my biased view on this issue and argue that (1) we need to focus on *generic* scalability requirements (rather than on features tuned to specific algorithms) wherever possible and (2) we need to try to build data mining systems that are not just scalable, but "SQL-aware".

2 Data Mining Landscape

We can categorize the ongoing work in data mining area as follows:

- *Inventing* new data analysis techniques
- *Scaling* data analysis technique over large data sets

2.1 Inventing New Data Analysis Techniques

In my opinion, discovery of new data analysis technique is to a large extent an expertise that requires insight in statistical and machine learning and related algorithmic areas. Examples of well-known techniques include decision-tree classification, clustering (see [5] for an overview of known techniques). Innovating in this space requires establishing statistical merit of a proposed technique and appears to have little interaction with database system issues. On a more pragmatic note, we seem to have a large number of established techniques in this space.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

2.2 Scaling Data Analysis Techniques

In contrast to inventing new data analysis techniques, the problem of scaling analysis techniques seems far more familiar for us. Although data analysis experts have worked for quite some time on the problems where the number of dimensions (data attributes) is large, they have been much less concerned with the number of data records. In part, this is because it has been traditional in data analysis work to make assumptions about the data distributions that model a data set. Assumptions on data distribution help reduce the size of the necessary data sets. In contrast, in most databases, little a priori inference on data distribution may be assumed. Thus, while most statistical and machine learning schemes assume a single-level store, over large data sets, we recognize the reality of multi-level store. This leads to two possible consequences:

- Develop efficient algorithms that take into account the fact that the data set is large.
- Restrict the scope of analysis objectives.

Scalability requirements lead to algorithms that carefully *stage computation* when data do not fit in memory. This is an area we have been leveraging. Nonetheless, we have to guard against the following dangers when we consider scalable implementations:

- *Restricting choice of data mining tasks:* While there has been an impressive amount of work related to association rules (see [1] for an overview) and their generalizations. Relatively less work has been done in the context of other classical data analysis technique, e.g., clustering, classification.
- *Scaling specific algorithms:* There are literally many variants of classification or clustering algorithms. The specific choice of an algorithm will depend on an application. Therefore, instead of focusing on how to scale a specific algorithm, we should try to identify common *data centric* steps in a broad class of algorithms. For example, all decision tree classifiers are driven by data-centric operations of building *counts* for distinct values of attributes and then *partitioning* the data set. Therefore, any decision tree classifier can be minimally modified so that whenever counting or partitioning steps are needed, the classifier uses an *interface* to invoke a *generic* middleware that optimizes scalable implementations of those operations by leveraging the way most decision tree classifiers grow a tree [4]. As a consequence, we are able to exploit the middleware for the entire set of decision tree classifiers, instead of requiring to build a specific scalable implementations for many variants.
- *Ignoring sampling as a scaling methodology:* Another generic way to scale the data over large data set is to use *sampling*. Whether sampling is appropriate for a class of data analysis technique, and even when appropriate, how much to sample and how, will become increasingly important question for data analysts. Use of sampling directly brings us to the related issue of restricting the scope of analysis objectives.

In designing the scalable implementations, some of the algorithms have made assumptions that ignore the fact that a datawarehouse will service not just data mining, but also traditional query processing. For example, some of the algorithms discuss how the physical design of the database may be tuned for a specific data mining task. However, in many cases, the physical design of a data warehouse is unlikely to be guided solely by the requirement of a single data analysis algorithm. Many of the scalable implementations also do not consider SQL database as the repository of data. In the next section, I will discuss this issue in somewhat more detail.

The problem of *restricting the scope of analysis* objective is motivated by the desire to strike a balance between accuracy and exhaustiveness of analysis with the desire to be efficient. While not a novel concept by any means, of direct interest to us will be techniques to efficiently cut corners that are motivated specifically by the large database (records and schema) scenarios. Restricting the scope of the analysis can take different forms. First, the analysis can be less than exhaustive. For example, support and confidence parameters are used to restrict the set of association rules that are mined. Next, the guarantee of the analysis can be probabilistic. A wide class of sampling based algorithms can take advantage of such an approximate analysis. This is clearly an area that is rich with past work by AI/statistics community. To ensure that we avoid pitfalls in cutting corners that severely affect quality of analysis, a database systems person needs to carefully *work with* a data analysis expert.

3 Motivation for SQL-aware Data Mining Systems

I will discuss two obvious reasons why we need to consider implementation of data mining algorithms that are "SQL-aware". However, from my point of view, *ad-hoc mining* provides the most compelling reason to consider SQL-aware data mining systems.

Data is in the warehouse

Data warehouses are deploying relational database technology for storing and maintaining data. Furthermore, data in datawarehouse will not be exclusively used for data mining, but will be shared also by OLAP and other database utilities. Therefore, for pragmatic reasons, the data mining utilities *must* assume a relational backend.

SQL Systems can be leveraged

Apart from the reality that SQL databases hold enterprise data, it is also true that SQL database management systems provide a rich set of primitives for data retrieval that the mining algorithms can exploit instead of developing all required functionality from scratch. It is surprising that although scalability of mining algorithms has been an active area of work, few significant pieces of work have looked at the issue of data mining algorithms for SQL systems. A nice study of a SQL-aware scalable implementation of association rules appear in [8].

Ad-hoc Mining

Today's data mining algorithms are invoked on a materialized disk-resident data set. If data mining were to succeed, data mining must evolve to *ad-hoc data mining* where the data set which is mined is specified on-the-fly. In other words, mining may be invoked on a data set that has been created on-the-fly by the powerful query tools. This allows us to mine an arbitrary query, not necessarily just base data. Thus, it is possible for a power user to use OLAP tools to specify a subset of the data and then to invoke mining tools on that data (cf. [7]). Likewise, it is possible to exploit the data reduction capabilities of the data mining tools to identify a subset of data that is interesting and then the OLAP tools can explore the subset of the data using query features. As an example, mining tool may be used to reduce the dimensionality of data. For ad-hoc mining, requiring the query to be materialized will result in unacceptable performance in many cases. A far more sensible approach is to cleverly exploit the interaction of the mining operator with the SQL operators. Similar interactions are possible with data visualization tools.

4 Building SQL-aware Data Mining Systems

In this section, I will use the example of *decision-tree classification* as an example of a data analysis technique, to illustrate the issues related to integration. The decision-tree classification process begins with the root node of the tree representing the entire data set. For each data value for each attribute of the data set, the counts of tuples are computed. These counts are used to determine a criteria to either partition the data set into a set of a disjoint partitions based on values of a specific attribute or to conclude that the node (the root in this case) is a leaf node in the decision tree. This "count and split" cycle is repeated until no new partitions are possible.

Recently, we built a scalable classifier [4] at Microsoft Research. Our approach exemplifies one of the several ways in which the problem of building SQL-aware systems may be approached. We started with a classical main-memory implementation of a decision tree classifier and Microsoft SQL Server. We augmented this set-up with a middleware to enhance performance. In particular, we modified the in-memory classifier such that it invokes the middleware whenever it needed to generate counts for each active node. In our first implementation, our goal was to get the SQL backend to do all the work in generating counts. The implementation helped us identify key bottlenecks and led to implementation changes for optimal use of server functionality as well as led to needs for SQL extensions. In the rest of this section, I will briefly discuss the issues related to exploiting the SQL backend as well as issues related to extensions to SQL for data mining. Where appropriate, I will draw examples from the decision-tree classifier and association rule implementations.

4.1 Using SQL Backend

Effectively using a SQL backend for data mining applications is a nontrivial problem since using the SQL backend as much as possible in an obvious way may hurt performance. The problem is analogous to what ROLAP providers faced in building their middleware over SQL engines. In particular, instead of generating a single complex SQL statement against the backend, they often generate multi-statement SQL that may be executed more efficiently. Similar considerations will be needed in the context of data mining.

On the other hand, we need to exploit the functionality in the SQL subsystem that can indeed be leveraged. Physical database design[1] and query processing subsystem including its use of parallelism are examples of functionality that data mining applications can exploit. Although the above seems too obvious to mention, there are few implementations of data mining algorithms today that take advantage of these functionality. The goal of harnessing the above functionality often lead to novel ways of *staging computation*. In [4], we discuss how we can batch servicing multiple active nodes (i.e., nodes that are still being grown) of a scalable decision tree classification algorithm and exploit data structures in the database server.

4.2 SQL Extensions

As we implement mining algorithms that generate SQL efficiently, we also will identify primitives that need to be incorporated in SQL. Once again, we can draw similarities with the OLAP world. Generation of SQL queries against the backend clearly benefits from the CUBE construct [6]. We can identify two goals for studying possible extensions to SQL, extensions that:

1. strongly interact with core SQL primitives and can result in significant performance improvement.
2. encapsulate a set of useful data mining primitives.

We feel that extensions that belong to (1) are extremely useful. An example of an operator which belongs there is the ability to sample a relation and more generally a query. This functionality can be exploited by many data mining algorithms, especially algorithms that provide probabilistic guarantees. However, the operation to sample a query is also a feature that strongly interacts with the query system. In particular, a sampling operator can be pushed down past a selection and interacts with other relational operators. While there has been substantial past research in this area, implementation issues related to processing sampling along with other relational operators continue to be an active area.

In our recent work on building scalable classification algorithms over SQL subsystems [4], we recognized that there is strong performance incentive to do *batch aggregation*. Intuitively, batch aggregation helps fully leverage a single data scan by evaluating multiple aggregation over the same data (or, query). This functionality

[1] It is important to emphasize that data mining algorithms need to *exploit* the physical design, but should not assume that such algorithms will singularly dictate such designs.

is important in classification since while growing a decision-tree classifier, for every active (non-leaf) node, we must collect the count of the number of tuples for every value of every attribute of the data table. This corresponds to a set of single-block aggregation queries where all the queries share the same *From* and *Where* clauses, i.e., differ only on *Group By* and *Select* clauses. Having the ability to do multi-statement optimization of the above set of related queries and to exploit a single data scan to evaluate them greatly speed up the classification algorithms. Such batch aggregation functionality goes beyond the CUBE operator [6]. Both sampling and batch aggregation strongly interact with core SQL primitives and thus with the SQL relational engine implementations.

We distinguish the above set of new operators with those that do not strongly interact with core SQL but presents a set of useful encapsulated procedures, perhaps supported via a mining extender/cartridge/blade or simply via extended system-provided stored procedures (depending on the database vendor). The purpose of such a set of operators is to make it easy to develop new data mining applications. However, for the set of primitives in this class to be useful, it is important that the operations be *unbundled* so that they may be shared. This issue is best illustrated through a recent SQL extension that has been proposed for association rules [2]. In that proposal, an extension is proposed to generate association rules. However, note that an alternative would have been to consider specifying *frequent itemsets* instead. An association rule can be derived easily with frequent itemsets as the primitive. Furthermore, the construct for frequent itemset may be exploited more generally for different variants of association rules. Thus, building extenders that directly map one-on-one to individual data mining algorithms may not be ideal.

5 Conclusion

In this article, I have reviewed various facets of data mining. There is an opportunity to work closely with data analysts to develop approximations of classical data analysis that are scalable. There is a need to look for generic scalability extensions for each class of data mining algorithms, rather than for specific scalable algorithms in each class. Another important direction is to consider scalable implementations over SQL systems. Such an effort will lead not only to changes in scalable algorithms, but also lead to new extensions for SQL that will make it better suited to support a variety of data mining utilities. Finally, it is well understood that core data mining algorithms by themselves are not sufficient, but needs to be integrated with other database tools. In particular, it is necessary to augment them with visualization support, as has been done in OLAP.

Acknowledgement

We thank Umesh Dayal, Usama Fayyad, Goetz Graefe, and Jim Gray for many fruitful discussions.

References

[1] Agrawal R. et. al. “Fast Dicovery of Association Rules,” pp. 307-328 in [5].

[2] Meo R., P. Giuseppe, Ceri S., “A new SQL-like Operator for Mining Association Rules,” in *Proc. of VLDB96*, pp. 122-133, Mumbai, India.

[3] Chaudhuri S., Dayal U. “An Overview of Datawarehousing and OLAP Technology,” in *Sigmod Record*, March 1997.

[4] Chaudhuri S., Fayyad U., Bernhardt J. “Scalable Classifier over SQL Databases,” in preparation.

[5] Fayyad U. et. al. *Advances in Knowledge Discovery and Data Mining*, MIT Press, 1996.

[6] Gray et al. “Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub Totals,”in *Data Mining and Knowledge Discovery*, 1(1), pp. 29-53, 1997.

[7] Han J. “Towards On-Line Analytical Mining in Large Databases,”to appear.

[8] Sarawagi S., Thomas S., Agrawal R. “Integrating Mining with Relational Database Systems: Alternatives and Implications,” in *Proc. of ACM Sigmod 98*, To appear.

Clustering Data Without Distance Functions

G.D. Ramkumar
Information Technology Lab
Hitachi America

Arun Swami
Neta Corporation

Abstract

Data mining is being applied with profit in many applications. Clustering or segmentation of data is an important data mining application. One of the problems with traditional clustering methods is that they require the analyst to define distance functions that are not always available. In this paper, we describe a new method for clustering without distance functions.

1 Introduction

Mining for information from databases has several important applications [11]. Three of the most common methods to mine data are association rules [1, 2], classification [7], and clustering [8, 3]. Association rules derive patterns from grouped data attributes that co-occur with high frequency. Classification methods produce hierarchical decision models for input data that is sub-divided into classes. Finally, clustering methods group together the records of a data set into disjoint sets that are similar in some respect. Clustering also attempts to place dissimilar records in different partitions.

Clustering arises naturally in several applications. For example, in the context of super market data, clustering of sale items to perform effective shelf-space organization is a common application. Another application is in clustering medical patient care data so that services performed for similar treatments are grouped together.

There is a lot of work on clustering particularly in the field of statistics. Traditional clustering algorithms [6, 4, 9, 10] use distance functions to measure similarity. The distance between any two elsements to be clustered is used in the decision on whether to put them in the same cluster or in disjoint clusters. For many applications, there may not be natural distance measures. For example, if the elements to be clustered have many categorical attributes, defining useful distance measures is problematic.

In this paper, we explore methods of clustering without distance functions. The methods are based on two basic principles. The first principle is that elements that share many attribute values are similar. One method will favor clustering such elements. The second principle is to group elements that share a common value for a distinguished attribute or label. We present a method to cluster records using a combination of these two methods: the exact combination depends on an input parameter specified by the user.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

2 Parametrized Clustering

We use the term *co-occurrence maximization* to describe the clustering method that tries to group together records that have frequently co-occurring items. We use the term *label minimization* to describe the clustering method that tries to cluster records of the same label. We propose that these two methods can co-exist in a hierarchical partitioning of data. At a node of the partition, one of co-occurrence maximization and label minimization is performed; choice of the type of partition depends on a) quality measures of co-occurrence maximization and label minimization, and b) a user specified numeric parameter that allows continuous variation between co-occurrence maximization and label minimization.

2.1 Co-occurrence Maximization

Let the input consists of a set of data records R. Each record $r \epsilon R$ defines a set of *(attribute, value)* pairs. We define the set of items I from R as the set of *(attribute, value)* pairs occurring in R. Items and combinations of items are interesting if they occur frequently enough in the data. The threshold frequency is given by a parameter *support* s. Pairs of items present in greater than a fraction s of the records are referred to in this paper as *frequent* item sets, $F \subseteq I^2$.

Recent work of Vipin Kumar, et al [3] studied clustering using frequent item set computation. Consider the set of items to be vertices of a graph. Frequent item sets can be viewed as edges of this graph (more generally, the authors consider frequent item sets of cardinality higher than 2, which induce hyper edges). The vertices I of the graph are partitioned into I_A and I_B so that the weight of edges across the partition is minimized. We refer to such an item partition as p_{AB}. Item partition p_{AB} can be used to induce partitions on the records of R, as described in [3]. Let us denote the induced partition as q_{AB} which divides the records of R into subsets R_A and R_B.

We implement co-occurrence maximization as follows:

1. Compute frequent item sets F of cardinality 2 from the data records, using any association mining algorithm, e.g., the Apriori algorithm [1].

2. Construct a graph G with vertices I as items and item sets J as the edges. The weight on a edge corresponds to the support for the item set.

3. Partition I into subsets I_A and I_B so that a) the weight of edges across the partition is minimized, and b) the number of vertices in the larger partition is close to $n/2$. We achieve such a partitioning using the randomized MinCut algorithm [5]; we terminate partitioning when condition b) is reached.

4. Partition the records of R into R_A and R_B. A record r of R goes into partition R_A (respectively, partition R_B) if its overlap with I_A is greater than its overlap with R_B. In calculating the overlap, the effect of single item frequency is taken into account by weighting each item's contribution with the inverse of its support.

2.1.1 Quality function Q_c

We measure the quality of a partitioning p_{AB} as a numeric between 0 and 1 as follows. Let $r \subseteq R$ be a record in the data set. If r falls entirely in I_A (or respectively, I_B) then the partition quality is maximum, namely 1. On the other hand, if r overlaps I_A and I_B equally, then the quality is minimum, namely 0. We are now ready to formally define the quality of a partitioning. Let the weighted cardinality of an item set $w(I)$ be defined as $\sum_{i \epsilon I} 1/s(i)$ where $s(i)$ is the support of the item in the data.

Definition: Quality $Q_c(r)$ of record r over partition p_{AB} is defined as

$$Q_c(r) = \frac{\textbf{abs}(s(r \cap I_A) - s(r \cap I_B))}{|I_A| + |I_B|}$$

Definition: Quality $Q_c(R)$ of a data set R over partition p_{AB} is

$$Q_c(R) = \frac{\sum_{r \epsilon R} Q_c(r, p_{AB})}{|R|}$$

We observe that the quality functions defined above have the desired properties mentioned earlier, in the way that they measure the partitions of items and records.

2.2 Label Minimization

The goal of label minimization based clustering of records is to increase the uniformity of class labels in each cluster; traditionally, this is measured using information gain at the clusters, as derived from an entropy function. We use a classification method similar to C4.5 [7] to partition the data records R based on their class labels. Let C denote the set of class labels of R. Partitioning of records is performed so that entropy of the data is reduced; in other words, the number of labels in each partition is reduced. The ultimate goal of partitioning is to make each cluster label uniform. Partitioning is performed using candidate tests derived from *(attribute, value)* pairs.

We look at how candidate tests are determined. Attributes of data records are of two types: categorical and numerical. For categorical attributes, the candidate tests are equality tests to each value of the attribute. For numerical attributes, candidate tests are range comparisons to check if the value of the attribute is less than or equal to the test value. Conceptually, for each test t, we divide the data records into R_A and R_B. We measure the gain in information attained by the division of R into R_A and R_B.

Let us define the information content of a set of records R along the lines of [7]. For every class label $c \epsilon C$, let $n(c, R)$ denote the frequency of occurrence of c in R. The information content $i(R)$ is defined as:

$$i(R) = \frac{\sum_{c \epsilon C} n(c, R) \times -\log_2(\frac{n(c,R)}{|R})}{|R|}$$

The information content of R_A and R_B combined is:

$$i(R_A, R_B) = \frac{i(R_A)|R_A| + i(R_B)|R_B|}{|R_A| + |R_B|}$$

The information gained by partitioning the data records into R_A and R_B using a test t is $i(R_A, R_B) - i(R)$. We pick the test that maximizes information gain in partitioning the set of data records. We are now ready to define the quality of label minimization as the relative information gain:

$$Q_l(R, t) = i(R_A, R_B) - i(R)$$

As in the case of co-occurrence, the quality measure Q_l is a number in the range $[0, 1]$.

2.3 Parametrization

We have now described two methods of partitioning the data: co-occurrence maximization based partitioning and label minimization based partitioning. Each method has a quality measure associated with it. We need a way to choose between the two partitioning methods at a recursive partitioning step. Such a choice can be facilitated using a user defined parameter w as follows. If $w * Q_c(R) > (1 - w) * Q_l(R)$, then pick co-occurrence maximization. Otherwise, pick label minimization.

Applying this strategy recursively results in a break down of the data into clusters. The parameter w can be used to control the clustering. As w increases from 0 to 1, emphasis goes continuously from co-occurrence maximization to label minimization. At an intermediate value of, say 0.5, emphasis is equally shared between co-occurrence maximization and label minimization. That is, the partitioning that has the best quality among the two is picked.

3 Experiments

We implemented our method for parametric clustering and tested it on some well known data sets. For brevity, we give only two examples in detail. We used data sets from the Machine learning data repository at UC Irvine[1]. We present sample results from applying our method to the following data sets:

- Zoo data set of animals.
- Adult data set of persons with attributes. We ran tests on both the test and the full data sets.

3.1 Results on Zoo Data Set

For the zoo data set, the schema attributes were:

1. Animal name: string.
2. Hair, feathers, eggs, milk, airborne, aquatic, predator, toothed, backbone, breathes, venomous, and fins: all boolean.
3. Legs: numeric (set of values: 0,2,4,5,6,8).
4. Tail, domestic, catsize: all boolean.
5. Type: numeric (integer values in range [1,7]).

The class label is the last attribute, namely type. The type attribute distinguishes mammals, fish, insects, birds, and so on.

When we ran parametric clustering with a parameter $w = 0.35$ and support threshold 0.06 for frequent itemsets, we got an interesting partition tree as shown in Figure 1. The test at the root node of the tree was a label minimization test on the milk attribute. All mammals were classified to the left. Subsequent test was a co-occurrence maximization test which divided the animals into two clusters A and B. Following are samples of animals from clusters A and B:

Partition A: Bear, boar, buffalo, calf, cheetah, deer, elephant, leopard, and others.

Partition B: Dolphin, fruitbat, girl, gorilla, platypus, seal, sealion, and others.

We noticed distinguishing characteristics of the partitions that are not otherwise apparent in the data attributes. Animals in partition A do not fly and partition B does not include any fast moving animals on land. Even though the class labels of the two partitions are identical, co-occurrence maximization was able to partition the data into qualitatively distinguishable groups.

3.2 Results on Adult Data Set

We ran tests on the adult data set as well and saw a few interesting clusters. For the adult data set, the schema attributes were:

1. age: continuous.
2. workclass, education: categorical.
3. education-num: continuous.
4. marital-status, occupation, relationship, race, sex: categorical.

[1]The URL is `http://www.ics.uci.edu/ mlearn/MLRepository.html`

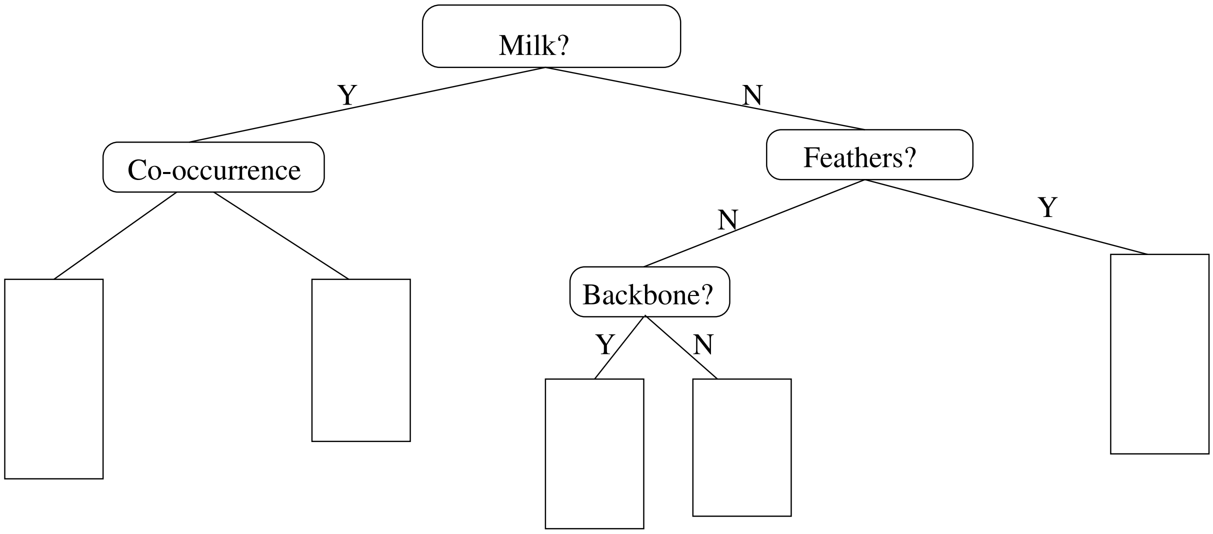


Figure 1: Partition tree obtained by performing parametric clustering on the zoo database, with parameter w set to 0.35 and a support threshold of 0.06.

5. capital-gain, capital-loss, hours-per-week: continuous.
6. native-country: categorical.
7. Class attribute is $> 50K$, $<= 50K$: categorical.

When we ran parametric clustering with a parameter $w = 0.3$ and support threshold 0.05 for frequent itemsets, we got an interesting partition tree. The test at the root node of the tree was a label minimization test on the marital status attribute. A few further partitions of the tree were co-occurrence maximizations. Subsequently, we noticed a label minimization partition testing the continuous attribute education level. The following test was a co-occurrence maximization test which divided the records into two clusters A and B. In Figure 3.2, we show some records sampled from partitions A and B. For clarity we show only a few attributes for each data record (we show missing attribute values by '?').

Note the distinguishing characteristics of partitions A versus partition B. Most records of partition A are white male. Partition B contains mostly private sector employees; also most records of B are female or not white. In this example as well, co-occurrence maximization was able to partition the data into qualitatively distinguishable groups, thereby enhancing the clustering capabilities of label minimization.

4 Conclusion

We presented a method for parametrized clustering of data using a strategy based on the principles of co-occurrence maximization and label minimization. The principles are naturally implemented using well known techniques from association rules and classification. Our method combines the benefits from the two approaches by seeking to optimize a quality function that is a weighted average of the qualities of the co-occurrence maximization and label minimization methods. Based on a user-defined parameter, parametrized clustering can be tuned to favor one of the methods more. The quality function we use depends linearly on the parameter and the quality functions of co-occurrence maximization and label minimization. Other quality functions could be used and are the subject of further research.

Partition A:

Employment	Education	Work class	Relationship	Race	Sex	Class
Self-emp-not-inc	HS-grad	Other-service	Husband	White	Male	$> 50K$
?	HS-grad	Other-service	Husband	White	Male	$<= 50K$
Self-emp-not-inc	10th	Farming-fishing	Husband	White	Male	$<= 50K$
Private	10th	Sales	Husband	White	Male	$<= 50K$

Partition B:

Employment	Education	Work class	Relationship	Race	Sex	Class
Private	10th	Other-service	Wife	White	Female	$<= 50K$
Private	11th	Exec-managerial	Husband	White	Male	$<= 50K$
Private	5th-6th	Machine-op-inspct	Wife	White	Female	$<= 50K$
?	10th	?	Husband	Asian-Pac-Islander	Male	$> 50K$

Figure 2: Partition tree obtained by performing parametric clustering on the adult database, with parameter w set to 0.3 and a support threshold of 0.05.

We tested our method on data sets from the machine learning library at UC Irvine. Further work remains on devising efficient algorithms and testing performance of parametric clustering.

References

[1] R. Agrawal, T. Imilienski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, pages 207–216, May 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference, Santiago, Chile*, 1994.

[3] E. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering based on association rule hypergraphs. In *Proc. Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1997.

[4] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[5] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, 1995.

[6] R. T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, 1994.

[7] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[8] T. Zhang, R. Ramakrishnan, and M. Linvy. Birch: an efficient data clustering method for large databases. In *Proc. of the 1996 ACM SIGMOD Int'l Conf. on Management of Data*, Montreal, Quebec, 1996.

[9] D. Fisher. Improving inference through conceptual clustering. In *1987 AAAI Conference*, pages 461–465, Seattle, Washington, 1987.

[10] D. Fisher. Optimization and simplification of hierarchical clusterings. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 118–123, Montreal, Quebec, 1995.

[11] U.M. Fayyad, G. Piatetski-Shapiro, and P. Smith. From data mining to knowledge discovery: An overview. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI/MIT Press, 1996.

Hypergraph Based Clustering in High-Dimensional Data Sets: A Summary of Results †

Eui-Hong (Sam) Han George Karypis Vipin Kumar
Bamshad Mobasher

Department of Computer Science and Engineering/Army HPC Research Center
University of Minnesota
{han,karypis,kumar,mobasher}@cs.umn.edu

Abstract

Clustering of data in a large dimension space is of a great interest in many data mining applications. In this paper, we propose a method for clustering of data in a high dimensional space based on a hypergraph model. In this method, the relationship present in the original data in high dimensional space are mapped into a hypergraph. A hyperedge represents a relationship (affinity) among subsets of data and the weight of the hyperedge reflects the strength of this affinity. A hypergraph partitioning algorithm is used to find a partitioning of the vertices such that the corresponding data items in each partition are highly related and the weight of the hyperedges cut by the partitioning is minimized. We present results of experiments on two different data sets: S&P500 stock data for the period of 1994-1996 and protein coding data. These experiments demonstrate that our approach is applicable and effective in high dimensional datasets.

1 Introduction

Clustering in data mining is a discovery process that groups a set of data such that the intracluster similarity is maximized and the intercluster similarity is minimized [CHY96]. These discovered clusters are used to explain the characteristics of the data distribution. For example, in many business applications, clustering can be used to characterize different customer groups and allow businesses to offer customized solutions, or to predict customer buying patterns based on the profiles of the cluster to which they belong.

Given a set of n data items with m variables, traditional clustering techniques [CS96, JD88] group the data based on some measure of similarity or distance between data points. Most of these clustering algorithms are

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

†This work was supported by NSF ASC-9634719, by Army Research Office contract DA/DAAH04-95-1-0538, by Army High Performance Computing Research Center cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Additional support was provided by the IBM Partnership Award, and by the IBM SUR equipment grant. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute. See http://www.cs.umn.edu/∼han for other related papers.

able to effectively cluster data when the dimensionality of the space (i.e., the number of variables) is relatively small. However, these schemes fail to produce meaningful clusters, if the number of variables is large.

Clustering of large dimensional data sets is of a great interest in many data mining applications. For example, in market basket analysis, a typical store sells thousands of different items to thousands of different customers. If we can cluster the items sold together, we can then use this knowledge to perform effective shelf-space organization as well as target sales promotions. Clustering of items from the sales transactions requires handling thousands of variables corresponding to customer transactions. Finding clusters of customers based on the sales transactions also presents a similar problem. In this case, the items sold in the store correspond to the variables in the clustering problem.

In this paper, we propose a method for clustering of data in a high dimensional space based on a hypergraph model. In a hypergraph model, each data item is represented as a vertex and related data items are connected with weighted hyperedges. A hyperedge represents a relationship (affinity) among subsets of data and the weight of the hyperedge reflects the strength of this affinity. Now a hypergraph partitioning algorithm is used to find a partitioning of the vertices such that the corresponding data items in each partition are highly related and the weight of the hyperedges cut by the partitioning is minimized.

To test the applicability and robustness of our scheme, we evaluated it on a wide variety of data sets [HKKM97a, MHB$^+$97, HBG$^+$98, HKKM97b]. We present a summary of results on two different data sets: S&P500 stock data for the period of 1994-1996 and protein coding data. These experiments demonstrate that our approach is applicable and effective in a wide range of domains. More specifically, our approach performed much better than traditional schemes for dimensionality reduction [Jac91, BDO95] in terms of quality of clusters and runtime.

The rest of this paper is organized as follows. Section 2 presents our clustering method based on hypergraph models. Section 3 presents the experimental results. Section 4 contains conclusion and directions for future work.

2 Hypergraph-Based Clustering

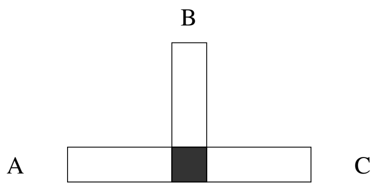
Our algorithm for clustering related items consists of the following two steps. During the first step, a weighted hypergraph H is constructed to represent the relations among different items, and during the second step, a hypergraph partitioning algorithm is used to find k partitions such that the items in each partition are highly related. In our current implementation, we use frequent item sets found by the association rule algorithm [AS94] as hyperedges.

2.1 Hypergraph Modeling

A hypergraph [Ber76] $H = (V, E)$ consists of a set of vertices (V) and a set of hyperedges (E). A hypergraph is an extension of a graph in the sense that each hyperedge can connect more than two vertices. In our model, the set of vertices V corresponds to the set of data items being clustered, and each hyperedge $e \in E$ corresponds to a set of related items. A key problem in modeling of data items as hypergraph is the determination of related items that can be grouped as hyperedges and determining weights of each such hyperedge.

The frequent item sets computed by an association rule algorithm such as Apriori are excellent candidates to find such related items. Note that these algorithms only find frequent item sets that have support greater than a specified threshold. The value of this threshold may have to be determined in a domain specific manner. The frequent item sets capture relationship of items of size greater than or equal to 2. Note that distance based relationships can only capture relationship among pairs of data points whereas the frequent items sets can capture relationship among larger sets of data points. This added modeling power is nicely captured in our hypergraph model.

Assignment of weights to the resulting hyperedges is more tricky. One obvious possibility is to use the support of each frequent item set as the weight of the corresponding hyperedge. Another possibility is to make the weight



support{A} = 0.25	
support{B} = 0.25	
support{C} = 0.25	
support{A,B} = 0.1	P(A|B) = P(B|A) = 0.4
support{B,C} = 0.1	P(B|C) = P(C|B) = 0.4
support{A,C} = 0.1	P(A|C) = P(C|A) = 0.4
support{A,B,C} = 0.1	P(A|BC) = P(B|AC) = P(C|AB) = 1.0

Figure 1: Illustration of the power of confidence in capturing relationships among items.

as a function of the confidence of the underlying association rules. For size two hyperedges, both support and confidence provide similar information. In fact, if two items A and B are present in equal number of transactions (i.e., if the support of item set {A} and item set {B} are the same), then there is a direct correspondence between the support and the confidence of the rules between these two items (i.e., greater the support for {A, B}, more confidence for rules "{A} $\Longrightarrow$ {B}" and "{A} $\Longrightarrow$ {B}"). However, support carries much less meaning for hyperedges of size greater than two, as in general, the support of a large hyperedge will be much smaller than the support of smaller hyperedges. Furthermore, for these larger item sets, confidence of the underlying association rules can capture correlation among data items that is not captured by support. For example, consider three items A, B and C and their relationship as represented in Figure 1. The support of each of the pairs {A, B}, {B, C} and {A, C} is 0.1, and the support of {A, B, C} is also 0.1. However, the three-way relationship among A, B and C is much stronger than those among pairs, as $P(C|AB)$, $P(B|AC)$ and $P(A|BC)$ are all 1.0. These conditional probabilities are captured in the confidence of corresponding association rules. This example also illustrates that relationships among item sets of size greater than 2 cannot always be captured by the pairwise relationships of its subsets irrespective of whether the relationship is modeled by the support or the confidence of the individual rules. Hence, a hypergraph is a more expressive model than a graph for this problem.

Another, more natural, possibility is to define weight as a function of the support and confidence of the rules that are made of a group of items in a frequent item set. Other options include correlation [BMS97], distance or similarity measure.

In our current implementation of the model, each frequent item-set is represented by a hyperedge $e \in E$ whose weight is equal to the average confidence of the association rules, called *essential* rules, that have all the items of the edge and has a singleton right hand side. We call them are *essential* rules, as they capture information unique to the given frequent item set. Any rule that has only a subset of all the items in the rule is already included in the rules of subset of this frequent item set. Furthermore, all the rules that have more than 1 item on the right hand size are also covered by the subset of the frequent item set. For example, if {A,B,C} is a frequent item-set, then the hypergraph contains a hyperedge that connects A, B, and C. Consider a rule {A}$\Longrightarrow${B,C}. Interpreted as an implication rule, this information is captured by {A}$\Longrightarrow${B} and {A}$\Longrightarrow${C}. Consider the following essential rules (with confidences noted on the arrows) for the item set {A,B,C}: {A,B}$\stackrel{0.4}{\Longrightarrow}${C}, {A,C}$\stackrel{0.6}{\Longrightarrow}${B}, and {B,C}$\stackrel{0.8}{\Longrightarrow}${A}. Then we assign weight of 0.6 ($\frac{0.4+0.6+0.8}{3} = 0.6$) to the hyperedge connecting A,B, and C. We will refer to this hypergraph as the association-rule hypergraph.

2.2 Finding Clusters of Items

Note that the frequent items sets already represent relationship among the items of a transaction. But these relationships are "fine grain". For example, consider the following three frequent item sets found from a database of stock transactions:

$$\{\text{Texas Inst}\uparrow, \text{Intel}\uparrow, \text{Micron Tech}\uparrow\}$$

$$\{\text{National Semiconduct}\uparrow\text{, Intel}\uparrow\}$$
$$\{\text{National Semiconduct}\uparrow\text{,Micron Tech}\uparrow\}$$

These item sets indicate that on many different days, stocks of Texas Instrument, Intel and Micron Technology moved up together, and on many days, stocks of Intel and National Semiconductor moved up together, etc. From these, it appears that Texas Instrument, Intel, Micron Technology and National Semiconductor are somehow related. But a frequent item set of these four stocks may have small support and may not be captured by the association rule computation algorithm.

However the hypergraph representation can be used to cluster together relatively large groups of related items by partitioning it into highly connected partitions. One way of achieving this is to use a hypergraph partitioning algorithm that partitions the hypergraph into two parts such that the weight of the hyperedges that are cut by the partitioning is minimized. Note that by minimizing the hyperedge-cut we essentially minimize the relations that are violated by splitting the items into two groups. Now each of these two parts can be further bisected recursively, until each partition is highly connected.

HMETIS [KAKS97, Kar98] is a multi-level hypergraph partitioning algorithm that has been shown to produce high quality bi-sections on a wide range of problems arising in scientific and VLSI applications. HMETIS minimizes the weighted hyperedge cut, and thus tends to create partitions in which the connectivity among the vertices in each partition is high, resulting in good clusters.

Once, the overall hypergraph has been partitioned into k parts, we eliminate bad clusters using the following cluster fitness criterion. Let e be a set of vertices representing a hyperedge and C be a set of vertices representing a partition. The fitness function that measures the goodness of partition C is defined as follow:

$$fitness(C) = \frac{\sum_{e \subseteq C} Weight(e)}{\sum_{|e \cap C| > 0} Weight(e)}$$

The fitness function measures the ratio of weights of edges that are within the partition and weights of edges involving any vertex of this partition. The high fitness value suggests that the partition has more weights for the edges connecting vertices within the partition. The partitions with fitness measure greater than a given threshold value are considered to be good clusters and retained as clusters found by our approach. In our experiments, we set this fitness threshold to 0.1. Note that this fitness criterion can easily be incorporated into a partitioning algorithm such that each partition is further bisected only if the fitness of the partition is below the given threshold value. Then all the partitions found can be considered good partitions.

Once good partitions are found, each good partition is examined to filter out vertices that are not highly connected to the rest of the vertices of the partition. The connectivity function of vertex v in C is defined as follow:

$$connectivity(v, C) = \frac{|\{e | e \subseteq C, v \in e\}|}{|\{e | e \subseteq C\}|}$$

The connectivity measures the percentage of edges that each vertex is associated with. High connectivity value suggests that the vertex has many edges connecting good proportion of the vertices in the partition. The vertices with connectivity measure greater than a give threshold value are considered to belong to the partition, and the remaining vertices are dropped from the partition. In our experiments, we set this connectivity threshold to 0.1.

3 Experimental Results

We tested the ability of our item-clustering algorithm to find groups of related items, on data-sets from many application areas. The results of these experiments are described in the following subsections. In all of our experiments, we used a locally implemented version of *Apriori* algorithm [AS94] to find the association rules and construct the association-rule hypergraph.

Discovered Clusters	Industry Group
APPLIED MATL↓, BAY NETWORK↓, 3 COM↓, CABLETRON SYS↓, CISCO↓, DSC COMM↓, HP↓, INTEL↓, LSI LOGIC↓, MICRON TECH↓, NATL SEMICONDUCT↓, ORACLE↓, SGI↓, SUN↓, TELLABS INC↓, TEXAS INST↓	Technology 1
APPLE COMP↓, AUTODESK↓, ADV MICRO DEVICE↓, ANDREW CORP↓, COMPUTER ASSOC↓, CIRC CITY STORES↓, COMPAQ↓, DEC↓, EMC CORP↓, GEN INSTRUMENT↓, MOTOROLA↓, MICROSOFT↓, SCIENTIFIC ATL↓	Technology 2
FANNIE MAE↓, FED HOME LOAN↓, MBNA CORP↓, MORGAN STANLEY↓	Financial
BAKER HUGHES↑, DRESSER INDS↑, HALLIBURTON HLD↑, LOUISIANA LAND↑, PHILLIPS PETRO↑, SCHLUMBERGER↑, UNOCAL↑	Oil
BARRICK GOLD↑, ECHO BAY MINES↑, HOMESTAKE MINING↑, NEWMONT MINING↑, PLACER DOME INC↑	Gold
ALCAN ALUMINUM↓, ASARCO INC↓, CYPRUS AMAX MIN↓, INLAND STEEL INC↓, INCO LTD↓, NUCOR CORP↓, PRAXAIR INC↓, REYNOLDS METALS↓, STONE CONTAINER↓, USX US STEEL↓	Metal

Table 1: Clustering of S&P 500 Stock Data

3.1 S&P 500 Stock Data

Our first data-set consists of the daily price movement of the stocks that belong to the S&P500 index. It is well known in the financial community, that stocks belonging to the same industry group tend to trade similarly. For example, a group of stocks from a particular industry tend to move up or down together depending on the market's belief about the health of this industry group. For this reason, we used this data set to verify the ability of our clustering algorithm to correctly cluster the various stocks according to their industry group.

The data set consists of a binary table of size 1000×716. Each row of this table corresponds to up or down movement indicator for one of the 500 stocks in S&P500 stocks, and each column corresponds to a trading day from Jan. 1994 to Oct. 1996. An entry of 1 in location (i, j) where i corresponds to up indicator of a stock means that the closing price of this stock on j-th day is significantly higher (2% or 1/2 point or more) than the day before. Similarly, an entry of 1 in location (i, j) where i corresponds to down indicator of a stock means that the closing price of this stock on j-th day is significantly lower (2% or 1/2 point or more) than the day before.

We clustered these stock indicators using our hypergraph-based method. We used a minimum support threshold of 3% which means that all stocks in a frequent item set must have moved together at least on 22 days. This lead to a hypergraph consisting of 440 vertices and 19602 hyperedges. Note that the number of vertices in the hypergraph is considerably smaller than the number of distinct items in the data-set. This is because some of the stocks do not move very frequently, hence the corresponding items do not have sufficient support. This hypergraph was then partitioned into 40 partitions. Out of these 40 partitions, only 20 of them satisfy the fitness function. Out of 20 clusters, 16 clusters were clean clusters as they contain stocks primarily from one industry group. Some of the clean clusters found from this data are shown in Table 1 and the complete list of clusters is available in [HKKM97b]. Looking at these clusters we can see that our item-clustering algorithm was very successful in grouping together stocks that belong to the same industry group. For example, our algorithm was able to find technology-, financial-, oil-, gold-, and metal-related stock-clusters. Also, it is interesting to see that our clustering algorithm partitioned the technology companies into two groups, and the first of them consists mostly of networking and semiconductor companies.

3.2 Protein Coding Database

Our next data set is from a problem from the domain of molecular biology. Molecular biologists study genes within the cells of organisms to determine the biological function of the proteins that those genes code for. Faced with a new protein, biologists have to perform a very laborious and painstaking experimental process to determine the function of the protein. To rapidly determine the function of many previously unknown genes, biologists generate short segments of protein-coding sequences (called expressed sequence tags, or ESTs) and match each

EST against the sequences of known proteins, using similarity matching algorithms [NRS+95]. The result of this matching is a table showing similarities between ESTs and known proteins. If the EST clusters can be found from this table such that ESTs within the same cluster are related to each other functionally, then biologists can match any new EST from new proteins against the EST clusters to find those EST clusters that match the new EST most closely. At this point, the biologists can focus on experimentally verifying the functions of the new EST represented by the matching EST clusters. Hence finding clusters of related ESTs is an important problem. Related work in this area can be found in [HHS92], which reports clustering of the sequence-level building blocks of proteins by finding transitive closure of the pairwise probabilistic similarity judgments.

To assess the utility of hypergraph-based clustering in this domain, we performed experiments with data provided by the authors of [NRS+95]. Our data set consists of a binary table of size 662×11986. Each row of this table corresponds to an EST, and each column corresponds to a protein. An entry of 1 in location (i, j) means that there is a significant match between EST i and protein j. We clustered the ESTs using our hypergraph-based method. In finding the frequent item sets we used a support of 0.02%, which essentially created frequent item sets of EST that are supported by at least three proteins. This led to a hypergraph with 407 vertices and 128,082 hyperedges. We used HMETIS to find 46 partitions, out of which 39 of them satisfied the fitness criteria. The total number of ESTs clustered was 389. These 39 EST clusters were then given to biologist to determine whether or not they are related. Their analysis showed that most of the EST clusters found by our algorithm correspond to ESTs that are related. In fact, 12 of the 39 clusters are very good, as each corresponds to a single protein family. These 12 clusters together contain 113 ESTs. Three clusters are bad (they contained random ESTs), and analysts could not determine the quality of 2 other clusters. Each of the remaining 22 clusters has subclusters corresponding to distinct protein families – 6 clusters have two subclusters, 7 clusters have three subclusters, 3 clusters have four subclusters, and 6 clusters have five subclusters. Furthermore, when we examined the connectivity of the sub-hypergraphs that correspond to some of these 22 clusters, we were able to see that further subdivision would have created single-protein clusters. This is particularly important, since it verified that the association-rule hypergraph is highly effective in modeling the relations between the various ESTs, and we are able to verify the quality of the clusters by looking at their connectivity. Also, our clustering algorithm took under five minutes to find the various clusters which includes the time required to find the association rules and form the hypergraph.

4 Conclusion and Directions for Future Work

In this paper, we have presented a method for clustering data in a high dimensional space based on a hypergraph model. Our experiments indicate that the hypergraph-based clustering holds great promise for clustering data in large dimensional spaces. Traditional clustering schemes such as *Autoclass* [CS96] and K-means [JD88] cannot be directly used in such large dimensionality data sets, as they tend to produce extremely poor results [HKKM97b]. These methods perform much better when the dimensionality of the data is reduced using methods such as Principal Component Analysis [Jac91]. However, as shown by our experiments in [HKKM97b], the hypergraph-based scheme produces clusters that are at least as good or better than those produced by *AutoClass* or K-means algorithm on the reduced dimensionality data sets.

One of the major advantages of our scheme over traditional clustering schemes is that it does not require dimensionality reduction, as it uses the hypergraph model to represent relations among the data items. This model allows us to effectively represent important relations among items in a sparse data structure on which computationally efficient partitioning algorithms can be used to find clusters of related items. Note that the sparsity of the hypergraph can be controlled by using an appropriate support threshold. An additional advantage of this scheme is its ability to control the quality of clusters according to the requirements of users and domains. With different levels of minimum support used in *Apriori* algorithm, the amount of relationship captured in the hypergraph model can be adjusted. Higher support gives better quality clusters containing smaller number of data items, whereas lower support results in clustering of larger number of items in poorer quality clusters. Furthermore, our hypergraph model allows us to correctly determine the quality of the clusters by looking at the internal connectivity of the nodes in each cluster. This fitness criterion in conjunction with the connectivity threshold discussed in Section 2 provide additional control over the quality of each cluster. Computationally, our scheme is linearly scalable with respect to the number of dimensions of data (as measured in terms of the number of binary variables) and items, provided the support threshold used in generating the association rules is sufficiently high.

Like other clustering schemes and dimensionality reduction schemes, our approach also suffers from the fact that right parameters are necessary to find good clusters. The appropriate support level for finding frequent item sets is largely depend on the application domain. Another limitation of our approach is that our scheme does not naturally handle continuous variables as they need to be discretized. However, such a discretization can lead to a distortion in the relations among items, especially in cases in which a higher value indicates a stronger relation. For this type of domains, in which continuous variables with higher values imply greater importance, we have developed a new algorithm called *Min-Apriori* [Han98] that operates directly on these continuous variables without discretizing them.

Our current clustering algorithm relies on the hypergraph partitioning algorithm HMETIS to find a good k-way partitioning. As discussed in Section 2, even though HMETIS produces high quality partitions, it has some limitations. Particularly, the number of partitions must be specified by the users as HMETIS does not know when to stop recursive bisection. However, we are working to incorporate the fitness criteria into the partitioning algorithm such that the partitioning algorithm determines the right number of partitions automatically. Another way of clustering is to do a bottom-up partitioning followed by a cluster refinement. In this approach, instead of using HMETIS to find the partitions in a top-down fashion, we can start growing the partitions bottom-up by repeatedly grouping highly connected vertices together. These bottom-up partitions can then be further refined using a k-way partitioning refinement algorithm as implemented in HMETIS.

Acknowledgments

We would like to thank Elizabeth Shoop and John Carlis for providing the protein coding data and verifying our results.

References

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference*, pages 487–499, Santiago, Chile, 1994.

[BDO95] M.W. Berry, S.T. Dumais, and G.W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1995.

[Ber76] C. Berge. *Graphs and Hypergraphs*. American Elsevier, 1976.

[BMS97] S. Brin, R. Motwani, and C. Silversteim. Beyond market baskets: Generalizing association rules to correlations. In *Proc. of 1997 ACM-SIGMOD Int. Conf. on Management of Data*, Tucson, Arizona, 1997.

[CHY96] M.S. Chen, J. Han, and P.S. Yu. Data mining: An overview from database perspective. *IEEE Transactions on Knowledge and Data Eng.*, 8(6):866–883, December 1996.

[CS96] P. Cheeseman and J. Stutz. Baysian classification (autoclass): Theory and results. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI/MIT Press, 1996.

[Han98] E.H. Han. Knowledge discovery in a large dimensional space using hypergraph models. Technical Report Thesis In Progress, Department of Computer Science, University of Minnesota, M inneapolis, 1998.

[HBG+98] E.H. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Webace: A web agent for document categorization and exploartion. In *Proc. of the 2nd International Conference on Autonomous Agents*, May 1998.

[HHS92] N. Harris, L. Hunter, and D. States. Mega-classification: Discovering motifs in massive datastreams. In *Proceedings of the Tenth International Conference on Artificial Intelligence (AAAI)*, 1992.

[HKKM97a] E.H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering based on association rule hypergraphs (position paper). In *Proc. of the Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 9–13, Tucson, Arizona, 1997.

[HKKM97b] E.H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering in a high-dimensional space using hypergraph models. Technical Report TR-97-063, Department of Computer Science, University of Minnesota, Minneapolis, 1997.

[Jac91] J. E. Jackson. *A User's Guide To Principal Components*. John Wiley & Sons, 1991.

[JD88] A.K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[KAKS97] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in VLSI domain. In *Proceedings ACM/IEEE Design Automation Conference*, 1997.

[Kar98] G. Karypis. hMETIS 1.0.0. *http://www.cs.umn.edu/~karypis/metis/hmetis/main.html*, 1998.

[MHB+97] J. Moore, E. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, and B. Mobasher. Web page categorization and feature selection using association rule and principal component clustering. In *7th Workshop on Information Technologies and Systems*, Dec. 1997.

[NRS+95] T. Newman, E.F. Retzel, E. Shoop, E. Chi, and C. Somerville. Arabidopsis thaliana expressed sequence tags: Generation, analysis and dissemination. In *Plant Genome III: International Conference on the Status of Plant Genome Research*, San Diego, CA, 1995.

Mining Large Itemsets for Association Rules

Charu C. Aggarwal and Philip S. Yu
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

Abstract

This paper provides a survey of the itemset method for association rule generation. The paper discusses past research on the topic and also studies the relevance and importance of the itemset method in generating association rules. We discuss a number of variations of the association rule problem which have been proposed in the literature and their practical applications. Some inherent weaknesses of the large itemset method for association rule generation have been explored. We also discuss some other formulations of associations which can be viable alternatives to the traditional association rule generation method.

1 Introduction

Association rules find the relationships between the different items in a database of sales transactions. Such rules track the buying patterns in consumer behavior eg. finding how the presence of one item in the transaction affects the presence of another and so forth. The problem of association rule generation has recently gained considerable prominence in the data mining community because of the capability of its being used as an important tool for knowledge discovery. Consequently, there has been a spurt of research activity in the recent years surrounding this problem.

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of binary literals called items. Each transaction T is a set of items, such that $T \subseteq I$. This corresponds to the set of items which a consumer may buy in a basket transaction.

An *association rule* is a condition of the form $X \Rightarrow Y$ where $X \subseteq I$ and $Y \subseteq I$ are two sets of items. The idea of an association rule is to develop a systematic method by which a user can figure out how to infer the presence of some sets of items, given the presence of other items in a transaction. Such information is useful in making decisions such as customer targeting, shelving, and sales promotions.

An important approach to the association rule problem was developed by Agrawal et. al. in [4]. This is a two-phase large itemset approach. Certain terminologies were defined in the same work which formalize these notions. In order to discuss the method further, we shall review some important definitions.

The *support* of a rule $X \Rightarrow Y$ is the fraction of transactions which contain both X and Y.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

The *confidence* of a rule $X \Rightarrow Y$ is the fraction of transactions containing X, which also contain Y. Thus, if we say that a rule has 90% confidence then it means that 90% of the tuples containing X also contain Y.

The large itemset approach is as follows. Generate all combinations of items that have fractional transaction support above a certain user-defined threshold called *minsupport*. We call all such combinations *large itemsets*. Given an itemset $S = \{i_1, i_2, \ldots, i_k\}$, we can use it to generate at most k rules of the type $[S - \{i_r\}] \Rightarrow \{i_r\}$ for each $r \in \{1, \ldots, k\}$. Once these rules have been generated, only those rules above a certain user defined threshold called *minconfidence* may be retained.

In order to generate the large itemsets, an iterative approach is used to first generate the set of large 1-itemsets L_1, then the set of large itemsets L_2, and so on until for some value of r the set L_r is empty. At this stage the algorithm can be terminated. During the kth iteration of this procedure a set of candidates C_k is generated by performing a $(k-2)$-join on the large itemsets L_{k-1}. The itemsets in this set C_k are *candidates* for large itemsets, and the final set of large itemsets L_k must be a subset of C_k. Each element of C_k needs to be validated against the transaction database to see if it indeed belongs to L_k. The validation of the candidate itemset C_k against the transaction database seems to be bottleneck operation for the algorithm. This method requires multiple passes over a transaction database which may potentially be quite large. For evaluating itemsets with a specific number of items, one pass is required over the transaction database. Thus, if the large itemset with the maximum number of items has 9 items in it, then the method requires 9 passes over the transaction database. This may result in substantial I/O times for the algorithm.

2 A survey of research on the large itemset method

After the inital algorithms proposed by Agrawal et. al. in [4], the problem has been extensively studied by other researchers and a number of fast variants have been proposed. In a subsequent paper in [5], Agrawal et. al. has discussed how the the algorithm for finding large itemsets may be sped up substantially by introducing a pruning approach which reduces the size of the candidate C_k. This algorithm uses the pruning trick that all subsets of a large itemset must also be large. Thus, if some $(k-1)$-subset of an itemset $I \in C_k$ does not belong to L_{k-1}, then that itemset can be pruned from further consideration. This process of pruning eliminates the need for finding the support of the candidate itemset I. In the same paper [5], an efficient data structure known as the hash-tree was introduced for evaluating the support of an itemset.

Subsequent work on the large itemset method has concentrated on the following aspects:

(1) Improving the I/O costs by reducing the number of passes over the transaction database.

(2) Improving the computational efficiency of the large itemset generation procedure.

(3) Finding efficient parallel algorithms for association rule generation.

(4) Introducing sampling techniques for improving the I/O and computational costs of large itemset generation.

(5) Extensions of the large itemset method to other problems such as quantitative association rules, generalized associations, and cyclic association rules.

(6) Finding methods for online generation of association rules by using the preprocess-once-query-many paradigm of online analytical processing.

2.1 Alternative Enhancements

We shall provide a brief survey of the work done in each of the above categories. A hash-based algorithm for efficiently finding large itemsets was proposed by Park et. al. in [19]. It was observed that most of the time in the was spent in evaluating and finding large 2-itemsets. The algorithm in Park et. al.[19] attempts to improve this approach by providing a hash based algorithm for quickly finding large 2-itemsets.

Brin et. al. proposed a method for large itemset generation which reduces the number of passes over the transaction database by counting some $(k+1)$-itemsets in parallel with counting k-itemsets. In most previously proposed algorithms for finding large itemsets, the support for a $(k+1)$-itemset was counted after k-itemsets have already been generated. In this work, it was proposed that one could start counting a $(k+1)$-itemset as soon as it was suspected that this itemset might be large. Thus, the algorithm could start counting for $(k+1)$-itemsets much earlier than completing the counting of k-itemsets. The total number of passes required by this algorithm is usually much smaller than the maximum size of a large itemset.

A partitioning algorithm was proposed by Savasere et. al. [21] for finding large itemsets by dividing the database into n partitions. The size of each partition is such that the set of transactions can be maintained in main memory. Then, large itemsets are generated separately for each partition. Let LP_i be the set of large itemsets associated with the ith partition. Then, if an itemset is large, then it must be the case that it must belong to at least one of LP_i for $i \in \{1, \ldots, k\}$. Now, the support of the candidates $\cup_{i=1}^{k} LP_i$ can be counted in order to find the large itemsets. This method requires just two passes over the transaction database in order to find the large itemsets.

The approach described above is highly parallelizable, and has been used to generate large itemsets by assigning each partition to a processor. At the end of each iteration of the large itemset method the processors need to communicate with one another in order to find the global counts of the candidate k-itemsets. Often, this communication process may impose a substantial bottleneck on the running time of the algorithm. In other cases, the time taken by the individual processors in order to generate the processor-specific large itemsets may be the bottleneck. Other methods have been proposed such as using a shared hash-tree among the multi-processors in order to generate the large itemsets[28]. More work on the parallelization of the itemset method may be found in [7, 17, 20, 28, 29].

A common feature of most of the algorithms reviewed above and proposed in the literature is that most such research is are variations on the "bottom-up theme" proposed by the *Apriori* algorithm[4, 5]. For databases in which the itemsets may be long, these algorithms may require substantial computational effort. Consider for example a database in which the length of the longest itemset is 40. In this case, there are 2^{40} subsets of this single itemset, each of which would need to be validated against the transaction database. Thus, the success of the above algorithms critically relies on the fact that the length of the frequent patterns in the database are typically short. An interesting algorithm for itemset generation has been proposed very recently by Bayardo [8]. This algorithm uses clever "look-ahead" techniques in order to identify longer patterns earlier on. The subsets of these patterns can then be pruned from further consideration. Initial computational results [8] indicate that the algorithm can lead to substantial performance improvements over the *Apriori* method.

Since the size of the transaction database is typically very large, it may often be desirable to use random sampling in order to generate the large itemsets. The use of random sampling to generate large itemsets may save considerable expense in terms of the I/O costs. A method of random sampling was introduced by Toivonen in [27]. The weakness of using random sampling is that it may often result in inaccuracies because of the presence of *data skew*. Data which are located on the same page may often be highly correlated and may not represent the over all distribution of patterns through the entire database. As a result, it may often be the case that sampling just 5% of the transactions may be as expensive as a pass through the entire database. Anti-skew algorithms for mining association rules have been discussed by Lin and Dunham in [16]. The techniques proposed in this paper reduce the maximum number of scans required to less than 2. The algorithms use a sampling process in order to collect knowledge about the data and reduce the number of passes.

The problems created by data skewness also arise in the context of parallel methods which divide the load among processors by partitioning the transaction data among the different processors. This is because each processor may receive a set of transactions which are not very representative of the entire data set. The problems created by skewness of the data in parallel mining of association rules have been investigated in [25].

2.2 Generalizations of the association rule problem

Initially, the association rule problem was proposed in the context of supermarket data. The motivation was to find how the items bought in a consumer basket related to each other. A number of interesting extensions and applications have been proposed. The problem of mining quantitative association rules in relational tables was proposed in [26]. In such cases association rules are discovered in relational tables which have both categorical and quantitative attributes. Thus, it is possible to find rules which indicate how a given range of quantitative and categorical attributes may affect the values of other attributes in the data. The algorithm for the quantitative association rule problem discretizes the quantitative data into disjoint ranges and then constructs an item corresponding to each such range. Once these pseudo-items have been constructed, a large itemset procedure can be applied in order to find the association rules. Often a large number of rules may be produced by such partitioning methods, many of which may not be interesting. An interest measure was defined and used in [26] in order to generate the association rules. In [14], an algorithm for clustering quantitative asociation rules was proposed. The aim of this algorithm was to generate rules which were more natural in terms of the quantitative clusters with which individual rules were associated. A closely related issue to finding quantitative association rules is the problem of finding profile association rules in which it is desirable to tie together rules which tie together user profiles with buying patterns. An algorithm for finding profile association rules was discussed in [2]. A method for finding optimized quantitative association rules has been discussed in [26]. This paper discusses how to choose the quantitative ranges in an optimal way so as to maximize the strength of the given association rules.

An interesting issue is that of handling taxonomies of items. For example, in a store, there may be several kinds of cereal, and for each individual kind of cereal, there may be multiple brands. Rules which handle such taxonomies are called *generalized associations*. The motivation is to generate rules which are as general as possible and also as general as possible while taking such taxonomies into account. Algorithms for finding such rules were presented in [24]. Savasere et. al.[22] also discuss how to find interesting negative association rules in the context of taxonomies of items. The focus of this work is to find rules which negatively correlate with rules which are discovered at higher levels of the taxonomy. Another useful extension of association rules which has been recently been proposed is the concept of *cyclic association rules*. It may often be the case that when association rules are computed for data which have a time component, periodic seasonal variations may be observed. For example, the monthly sales of goods correlate with each other differently on a seasonal basis. Ozden et. al. [18] have proposed efficient algorithms for finding association rules which display cyclic variation over time.

2.3 Online generation of association rules

Since the size of the transaction database may be very large, the algorithms for finding association rules are both compute-intensive and require substantial I/O. Thus it is difficult to provide quick responses to user queries. Methods for online generation of association rules have been discussed by Aggarwal and Yu[1]. This algorithm uses the preprocess-once-query-many paradigm of OLAP in order to generate association rules quickly by using an adjacency lattice to prestore itemsets. The interesting feature of this work is that the rules which are generated are independent of both the size of the transaction data and the number of itemsets prestored. In fact, the running time of the algorithm is completely proportional to the size of the output. It is also possible to generate queries for rules with specific items in them. In the same work, redundancy in association rule generation has been discussed. A rule is said to be redundant at a given level of support and confidence if its existance is implied by

X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

Rule	**Support**	**Confidence**
$X \Rightarrow Y$	25%	50%
$X \Rightarrow Z$	37.5%	75%

Table 2: (a) The base data (b) Corresponding support and confidence

some other rule in the set. For example, consider the following pair of rules:

$$\{\text{Milk}\} \Rightarrow \{\text{Bread, Butter}\}$$
$$\{\text{Milk, Bread}\} \Rightarrow \{\text{Butter}\}$$

In the above example, the second rule is redundant since its existance is implied by the first. Typically, a single essential rule may imply a large number of rules in the final output. Algorithms were proposed to generate a minimal set of essential rules for a given set of data.

3 A criticism of the large itemset method

The large itemset method has proved as a useful tool for mining association rules in large datasets which are inherently sparse. However, in many cases the method is difficult to generalize to other scenarios for computational reasons. Some of the criticisms associated with the large itemset method are the following:

- **Spuriousness in itemset generation:** We shall explain this issue with the help of an example. Consider a retailer of breakfast cereal which surveys 5000 students on the activities they engage in the morning. The data shows that 3000 students play basketball, 3750 eat cereal, and 2000 students both play basketball and eat cereal. For a minimum support of 40%, and minimum confidence of 60%, we find the following association rule:
  play basketball ⇒ eat cereal
  The association rule is misleading because the overall percentage of students eating cereal is 75% which is even larger than 60%. Thus, playing basketball and eating cereals are negatively associated: being involved in one decreases the chances of being involved in the other. Consider the following association:
  play basketball ⇒ (not) eat cereal
  Although this rule has both lower support and lower confidence than the rule implying positive association, it is far more accurate. Thus, if we set the support and confidence sufficiently low, two contradictory rules would be generated; on the other hand if we set the parameters sufficiently high only the inaccurate rule would be generated. In other words, no combination of support and confidence can generate purely the correct association.

  Another example is illustrated in Table 2(a), in which we have three items X, Y, and Z. The coefficient of correlation between X and Y is 0.577, while that between X and Z is -0.378. Thus, X and Y are positively related to each other, while X and Z are negatively related. The support for the rules $X \Rightarrow Y$ and $X \Rightarrow Z$ are illustrated in Table 2(b). Interestingly, the support and confidence for the rule $X \Rightarrow Z$ strictly dominates the support and confidence for the rule $X \Rightarrow Y$. Thus, it is not possible to find any level of *minsupport* and *minconfidence* at which only the rule $X \Rightarrow Y$ would be generated, without a spurious rule $X \Rightarrow Z$ being generated as well. If we set *minsupport* and *minconfidence* too low, then in many cases (especially when different items have widely varying global density) an unacceptably large number of rules might be generated, (a great majority of which may be spurious) and this defeats the purpose of data mining in the first place.

- **Dealing with dense data sets:** For a k-dimensional database, there are 2^k possibilities for itemsets. Some data sets may be such that a large number of these 2^k possibilities may qualify above the minimum support. For such situations, it may be necessary to set the *minsupport* to an unacceptably high level. This may result in a number of important rules being lost. Often, the value of *minsupport* is decided based on computational constraints or in restricting the number of rules generated to a manageable number. Thus, it is impossible to ascertain when important rules are being lost and when they are not.

  Some important applications in which the data may be dense are the following:

  - **Negative association rules:** Suppose we wish to find negative correlations among items. In other words, we wish to mine association rules in which presence as well as absence of an item may be used. Although the large itemset approach can be directly extended to this problem by treating the absence of an item as a pseudo-item, the sparsity of item presence in real transaction data may result in considerable bias towards rules which are concerned only with finding rules corresponding to absence of items rather than their presence.
  - **Data in which different attributes have widely varying densities:** Suppose that the categorical data corresponding to customer profile information is available along with the sales transaction data and we wish to find association rules concerning the demographic nature of people buying different items. In this case, directly applying the itemset method may be very difficult because of the different densities of the demographic and the sales transaction information. An example of such a demographic attribute is sex, which may take on either value 50% of the time. On the other hand a bit representing an item may take on the value of 1 only 5% of the time.

4 An alternative approach using collective strength

The use of an interest measure has been presented as a solution in order to avoid spurious association rules. The interest level of a rule is the ratio of the actual strength to the expected strength based upon the assumption of statistical independence. Past work has concentrated on using the interest measure as a pruning tool in order to remove the uninteresting rules in the output. However, (as the basketball-cereal example illustrates) as long as support is still the primary determining factor in the initial itemset generation, either the user has to set the initial support parameter low enough so as to not lose any interesting rules in the output or risk losing some important rules. In the former case, computational efficiency may be a problem, while the latter case has the problem of not being able to retain rules which may be interesting from the point of view of a user.

There is some other work which deals with providing alternative methods for viewing itemsets. Correlation rules have been discussed by Brin et. al. in [9]. In [10], the notion of developing implication rules instead of association rules has been discussed. The implication strength of a rule is a number between 0 and ∞. An implication strength of 1 indicates that the strength of the rule is exactly as it would be under the assumption of statistical independence. An implication strength which is larger than 1 indicates greater than expected presence. This measure is preferable to confidence because it deals with greater than expected measures for finding association rules.

A different notion called "collective strength" has been defined in [3]. In this work, it is desired to find interesting association rules by using "greater than expected values".

The collective strength of an itemset is defined to be a number between 0 to ∞. A value of 0 indicates perfect negative correlation, while a value of ∞ indicates perfectly positive correlation. A value of 1 indicates the "break-even point", corresponding to an itemset present at expected value.

An itemset I is said to be in *violation* of a transaction, if some of the items are present in the transaction, and others are not. Thus, the concept of violation denotes how many times a customer may buy at least some of the items in the itemset, but may not buy the rest of the items.

The *violation rate* of an itemset I is denoted by $v(I)$ and is the fraction of violations of the itemset I over all transactions.

The *collective strength* of an itemset I is denoted by $C(I)$ and is defined as follows:

$$C(I) = \frac{1 - v(I)}{1 - E[v(I)]} \cdot \frac{E[v(I)]}{v(I)} \tag{1}$$

The expected value of of $v(I)$ is calculated assuming statistical independence. Let us pause here to understand the meaning of collective strength before proceeding further. We note that the violation of an itemset in a transaction is a "bad event" from the perspective of trying to establish high correlation among the corresponding items. Thus $v(I)$ is the fraction of bad events while $(1 - v(I))$ is the fraction of "good events". The above definition for collective strength can be recast as follows:

$$C(I) = \frac{\text{Good Events}}{\text{E[Good Events]}} \cdot \frac{\text{E[Bad Events]}}{\text{Bad Events}} \tag{2}$$

The following are some of the interesting properties of collective strength.

(1) The notion of collective strength treats the 0 and 1 attributes in a symmetric way. Thus, if we were to apply this to a problem in which the absence as well as the presence of an item is used to find the itemsets, then we can immediately perceive the benefits of this definition.

(2) It is instructive to examine how this definition for collective strength fares for the case of a 2-itemset $I = \{i_1, i_2\}$. If i_1 and i_2 items are perfectly positively correlated, then the collective strength for the corresponding 2-itemset is ∞. This is because in this case, the fraction of occurances of bad events is 0. On the other hand, when the items are perfectly negatively correlated, the fraction of good events is 0, and hence the collective strength for the corresponding itemset pair is 0. For items which are independent from one another, the collective strength is 1. We provide this example in light of the criticisms that we expounded for the use of the interest measure at the beginning of the section.

(3) The collective strength uses the *relative* number of times an itemset is present in the database. The itemsets which have an insignificant presence can always be pruned off at a later stage. The level of presence of an itemset I which constitutes useful information is not exactly the same problem as finding whether or not the items in I are related to each other. Support can still be used in order to isolate those itemsets which have substantial presence in the database.

A *strongly collective itemset* is one which has high collective strength along with all of its subsets.

Definition 1: An itemset I is denoted to be strongly collective at level K, if it satisfies the following properties:

(1) The collective strength $C(I)$ of the itemset I is at least K.

(2) **Closure property:** The collective strength $C(J)$ of every subset J of I is at least K.

It is necessary to force the closure property in order to ensure that unrelated items may not be present in a itemset. Consider, for example, the case when itemset I_1 is $\{$Milk, Bread$\}$ and itemset I_2 is $\{$Diaper, Beer$\}$. If I_1 and I_2 each have high collective strength, then it may often be the case that the itemset $I_1 \cup I_2$ may also have a high collective strength, even though items such as milk and beer may be independent.

Algorithms for finding strongly collective itemsets have been proposed in [3]. This algorithm requires at most 2 passes over the transaction database in order to find the strongly collective baskets.

5 Conclusions and Summary

This paper discusses a survey of the large itemset method and its applications. The amount of research devoted to this problem has been very substantial in the recent years. Increasingly efficient algorithms have been proposed for the large itemset method by using smart modifications on the *Apriori* algorithm proposed by Agrawal et. al. However, the considerable computational requirements of the *Apriori* method may often require methods which are capable of online generation of association rules. Such techniques have been discussed by Aggarwal and Yu in [1]. We also discussed the weaknesses of the large itemset method for association rule generation. An alternative to the previously proposed large itemset technique has been discussed in [3] which uses greater than expected measures in order to generate association rules. This technique may be often preferable to to the large itemset method when the data is either fully or partially dense.

References

[1] Aggarwal C. C., and Yu P. S. "Online Generation of Association Rules." *Proceedings of the International Conference on Data Engineering,* Orlando, Florida, February 1998.

[2] Aggarwal C. C., Sun Z., and Yu P. S. "Generating Profile Association Rules." *IBM Research Report*, RC-21037.

[3] Aggarwal C. C., and Yu P. S. "A new framework for itemset generation." *IBM Research Report*, RC-21064.

[4] Agrawal R., Imielinski T., and Swami A. "Mining association rules between sets of items in very large databases." *Proceedings of the ACM SIGMOD Conference on Management of data,* pages 207-216, 1993.

[5] Agrawal R., and Srikant R. "Fast Algorithms for Mining Association Rules in Large Databases." *Proceedings of the 20th International Conference on Very Large Data Bases,* pages 478-499, 1994.

[6] Agrawal R., and Srikant R. " Mining Sequential Patterns." *Proceedings of the 11th International Conference on Data Engineering,* pages 3-14, March 1995.

[7] Agrawal R. and Shafer J. "Parallel Mining of Association Rules: Design, Implementation, and Experience." *Technical Report RJ10004, IBM Almaden Research Center*, San Jose, CA 95120, Jan. 1996.

[8] Bayardo R. J. "Efficiently Mining Long Patterns from Databases." *Unpublished Research Report*.

[9] Brin S., Motwani R. and Silverstein C. "Beyond Market Baskets: Generalizing Association Rules to Correlations." *Proceedings of the ACM SIGMOD, 1997.* pages 265-276.

[10] Brin S., Motwani R. Ullman J. D. and Tsur S. "Dynamic Itemset Counting and implication rules for Market Basket Data." *Proceedings of the ACM SIGMOD, 1997.* pages 255-264.

[11] Chen M. S., Han J., and Yu P. S. "Data Mining: An Overview from Database Perspective." *IEEE Transactions on Knowledge and Data Engineering,* 8(6): 866-883, December 1996.

[12] Klementtinen M., Mannila H., Ronkainen P., Toivonen H., and Verkamo A. I. "Finding interesting rules from large sets of discovered association rules." *Proceedings of the CIKM* 1994.

[13] Han J., and Fu Y. "Discovery of Multi-level Association Rules From Large Databases." *Proceedings of the International Conference on Very Large Databases*, pages 420-431, Zurich, Switzerland, September 1995.

[14] Lent B., Swami A., and Widom J. "Clustering Association Rules." *Proceedings of the Thirteenth International Conference on Data Engineering.* pages 220-231, Birmingham, UK, April 1997.

[15] Mannila H., Toivonen H., and Verkamo A. I. “Efficient algorithms for discovering association rules.” *AAAI Workshop on Knowledge Discovery in Databases,* 1994, pages 181-192.

[16] Lin J.-L. and Dunham M. H. “Mining Association Rules: Anti-Skew Algorithms.” *Proceedings of the International Conference on Data Engineering*, Orlando, Florida, February 1998.

[17] Mueller A. “Fast sequential and parallel methods for association rule mining: A comparison.” Technical Report CS-TR-3515, Department of Computer Science, University of Maryland, College Park, MD, 1995.

[18] Ozden B., Ramaswamy S, and Silberschatz A. “Cyclic Association Rules.” *Proceedings of the International Conference on Data Engineering*, Orlando, Florida, February 1998.

[19] Park J. S., Chen M. S., and Yu P. S. “An Effective Hash-based Algorithm for Mining Association Rules.” *Proceedings of the ACM-SIGMOD Conference on Management of Data, 1995.* Extended version appears as: “Using a Hash-based Method with Transaction Trimming for Mining Association Rules.” *IEEE Transactions ob Knowledge and Data Engineering*, Volume 9, no 5, September 1997, pages 813-825.

[20] Park J. S., Chen M. S. and Yu P. S. “Efficient Parallel Data Mining of Association Rules.” *Fourth International Conference on Information and Knowledge Management*, Baltimore, Maryland, November 1995, pages 31-36. Technical Report RC20156, IBM T. J. Watson Research Center, August 1995.

[21] Savasere A., Omiecinski E., and Navathe S. B. “An efficient algorithm for mining association rules in large databases.” *Proceedings of the 21st International Conference on Very Large Databases,* 1995.

[22] Savasere A., Omiecinski E., and Navathe S. B. “Mining for Strong Negative Associations in a Large Database of Customer Transactions.” *Proceedings of the International Conference on Data Engineering,* February 1998.

[23] Rastogi R., and Shim K. “Mining Optimized Association Rules with Categorical and Numeric Attributes.” *Proceedings of the International Conference on Data Engineering*, Orlando, Florida, February 1998.

[24] Srikant R., and Agrawal R. “Mining Generalized Association Rules.” *Proceedings of the 21st International Conference on Very Large Data Bases*,1995, pages 407-419.

[25] Xiao Y. and Cheung D. W. “Effects of data Skewness and Workload Balance in Parallel Data Mining.” *Unpublished Research Report.*

[26] Srikant R., and Agrawal R. “Mining quantitative association rules in large relational tables”. *Proceedings of the ACM SIGMOD Conference on Management of Data, 1996.* pages 1-12.

[27] Toivonen H. “Sampling Large Databases for Association Rules”. *Proceedings of the 22nd International Conference on Very Large Databases*, Bombay, India, September 1996.

[28] Zaki M. J., Ogihara M., Parthasarathy S., and Li W. “Parallel Data Mining for Association Rules on Shared-Memory Multi-processors.” *Supercomputing'96*, Pittsburg, PA, pages 17-22, 1996 (also available as URCS Technical Report 618 , May 1996).

[29] Zaki M. J., Parthasarathy S., Li W., “A Localized Algorithm for Parallel Association Mining”, *9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pp 321-330, Newport, Rhode Island, June 22-25, 1997.

Mining Temporal Relationships with Multiple Granularities in Time Sequences*

Claudio Bettini
DSI Department, University of Milano
via Comelico 39
20135 Milano, Italy
`bettini@dsi.unimi.it`

X. Sean Wang and Sushil Jajodia
Department of Information and Software Systems Engineering
George Mason University
Fairfax, VA 22030
`{xywang,jajodia}@isse.gmu.edu`

1 Introduction

Discovering sequential relationships in a time sequence is often important to many application domains, including financial, manufacturing, etc. Indeed, a lot of work has been done in this area (see [1, 6] for an overview). An important aspect for such a discovery process is, however, largely missing from the literature: namely discovering temporal patterns or relationships that involve multiple time granularities. This paper reports the progress in this front. A more detailed study can be found in [4].

In this paper, we focus on algorithms for discovering sequential relationships when a rough pattern of relationships is given. The rough pattern (which we term "event structure") specifies what sort of relationships a user is interested in. For example, a user may be interested in "which pairs of events occur frequently one week after another". The algorithms will find the instances that fit the event structure. They can also be used when more information is filled in the structure, as in in the following example from the stock market domain: "what kind of events frequently follow within 5 trading-days from the occurrence of a rise in IBM stocks, when this is preceded, one week before, by the fall of another hi-tech company stock". These algorithms form a core component for a data mining environment.

We view the actual data mining process as being done interactively through a user interface. Data mining requests (in terms of event structures) are issued through the user interface and processed using the algorithms. Initially, the user will issue a request with a simple structure. Complicated structures may be given by the user

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*This work was supported by the National Science Foundation (NSF) under the grant IRI–9633541. The work of Wang and Jajodia was also supported by the NSF grants IRI–9409769 and INT–9412507, respectively. The work of Bettini was partially carried out while visiting George Mason University.

after s/he explores the data set using simpler ones. That is, event structures "evolve" from simpler ones to more complex ones. Our algorithms are designed, however, to handle complicated as well as simple event structures.

The remainder of this paper is organized as follows. In Section 2, we precisely define what we mean by "mining temporal relationships" via the concepts of event structures and their solutions. In Section 3, we present an algorithm for finding frequent events that fit the given event structures. This algorithm may be too time consuming when implemented directly. In Section 4, we provide a powerful heuristic that dramatically reduces the computation time. The heuristic takes advantage of our theoretical study of the event structures. We conclude with Section 5.

2 The event-mining problem

A number of notions must be introduced to formalize the mining problem. We assume that there is a finite set of *event types*. Examples of event types are "deposit to an account" or "price increase of a specific stock". We use the symbol E, possibly with subscripts, to denote event types. An *event* is a pair $e = (E, t)$, where E is an event type and t is a positive integer, called the *timestamp* of e. An *event sequence* is a finite set of events. Intuitively, each event (E, t) appearing in an event sequence σ represents the occurrence of event type E at time t.

To model the temporal relationships among events in a sequence, we first specify what we mean by a time granularity, following [8, 5], and then introduce the notion of a temporal constraint with granularity.

We assume an underlying time domain $(\mathcal{T}, \leq)$ where $\mathcal{T}$ is an infinite set of time instants and $\leq$ a total order. A *granularity* is a mapping μ from the set of the positive integers (used as index) to subsets of the time domain such that for all positive integers i and j with $i < j$: (1) $\mu(i) \neq \emptyset \wedge \mu(j) \neq \emptyset$ implies that each number in $\mu(i)$ is less than all the numbers in $\mu(j)$, and (2) $\mu(i) = \emptyset$ implies $\mu(j) = \emptyset$.

Each set $\mu(i)$, if non-empty, is called a *granule* of μ. Property (1) says that granules do not overlap and that the order on indexes follows the order on the corresponding granules. Property (2) says that the subset of the index corresponding to the granules forms a set of contiguous integers. Intuitive granularities, e.g., `day`, `month`, `week` and `year`, satisfy the above definition. For example, we can define a special granularity `year` starting from year 1800 as follows: `year`(1) is the subset of the time domain (an interval of reals) corresponding to the year 1800, `year`(2) is the subset corresponding to the year 1801, and so on. The above definition allows to model also more complex granularities like `leap-year`, `business-week`, or `business-day` (in the following abbreviated in `b-day`). In this paper, all timestamps in an event sequence are assumed to be in terms of a fixed granularity, and in particular we assume that granularity to be `second`.

We can now define a temporal constraint with granularity.

Definition 1: Let m and n be non-negative integers with $m \leq n$ and μ be a granularity. A *temporal constraint with granularity* (TCG) $[m, n]\,\mu$ is the binary relation on positive integers defined as follows: For positive integers t_1 and t_2, (t_1, t_2) satisfies $[m, n]\,\mu$ iff (1) $t_1 \leq t_2$, (2) $\lceil t_1 \rceil^\mu$ and $\lceil t_2 \rceil^\mu$ are both defined, and (3) $m \leq (\lceil t_2 \rceil^\mu - \lceil t_1 \rceil^\mu) \leq n$.

In the above, $\lceil k \rceil^\mu$ denotes the index i of the granule of μ which contains the k^{th} second, if it exists, otherwise denotes undefined. Note that a set of these constraints cannot be translated into an equivalent set of constraints in terms of the same granularity [5]. For example, we should *not* translate $[0, 0]$ `day` into $[0, 23]$ `hour` since 23 hours can overlap across two consecutive days.

In the following we say that *a pair of events satisfies a constraint* if the corresponding timestamps do. It is easily seen that the pair of events (e_1, e_2) satisfies TCG $[0, 0]$ `day` if events e_1 and e_2 happen within the same day but e_2 does not happen earlier than e_1. Similarly, (e_1, e_2) satisfies TCG $[0, 2]$ `hours` if e_2 happens either in the same second as e_1 or within two hours after e_1. Finally, (e_1, e_2) satisfies $[1, 1]$ `month` if e_2 occurs in the month immediately after that in which e_1 occurs.

Hence, TCGs are used as temporal binary constraints on events and can be combined in a complex temporal relationship among several events, each being assigned to a different variable.

Definition 2: An *event structure (with granularities)* is a rooted directed acyclic graph (W, A, Γ), where W is a finite set of event variables, $A \subseteq W \times W$ and Γ is a mapping from A to the finite sets of TCGs.

The set of TCGs assigned to an edge is taken as conjunction. Figure 1 shows an event structure.

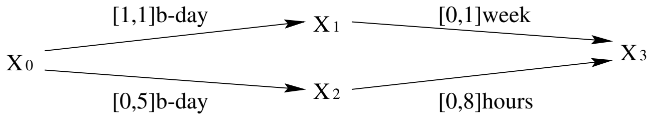


Figure 1: An event structure.

Given an event structure $\mathcal{S} = (W, A, \Gamma)$, we call *complex event type derived from* $\mathcal{S}$ the structure $\mathcal{S}$ with each variable associated with a specific event type, and we call *complex event matching* $\mathcal{S}$ the structure $\mathcal{S}$ with each variable associated with a distinct event such that the event timestamps satisfy the time constraints in Γ.

Example 1: Assume an event sequence that records stock-price fluctuations (rise and fall) every 15 minutes (this sequence can be derived from the sequence of stock prices) as well as the time of the releases of company earnings reports. Consider the event structure depicted in Figure 1. If we assign the event types for X_0, X_1, X_2 and X_3 to be `IBM-rise`, `IBM-earnings-report`, `HP-rise`, and `IBM-fall`, respectively, we have a complex event type. This complex event type describes that the IBM earnings were reported one business day after the IBM stock rose, and in the same or the next week the IBM stock fell; while the HP stock rose within 5 business days after the same rise of the IBM stock and within 8 hours before the same fall of the IBM stock.

If $\mathcal{T}$ is a complex event type derived from an event structure having k variables, and σ' is a subset of an event sequence σ such that $|\sigma'| = k$, then σ' is said to be an *occurrence* of $\mathcal{T}$ if a complex event matching that event structure can be derived by assigning a distinct event in σ' to each variable so that the type of the event is the same as the type assigned to the same variable by $\mathcal{T}$.

We are now ready to formally define the mining problem.

Definition 3: An *event-mining problem* is a quadruple $(\mathcal{S}, \gamma, E_0, \rho)$, where $\mathcal{S}$ is an event structure, γ (the *minimum confidence value*) a real number between 0 and 1 inclusive, E_0 (the *reference type*) an event type, and ρ is a partial mapping which assigns a set of event types to some of the variables (except the root).

An event-mining problem $(\mathcal{S}, \gamma, E_0, \rho)$ is the problem of finding all complex event types $\mathcal{T}$ such that each $\mathcal{T}$ (i) occurs frequently in the input sequence and (ii) is derived from $\mathcal{S}$ by assigning E_0 to the root and a specific event type to each of the other variables. (The assignments in (ii) must respect the restriction stated in ρ.) The frequency is calculated against the number of occurrences of E_0. This is intuitively sound: If we want to say that event type E frequently happens one day after IBM stock falls, then we need to use the events corresponding to falls of IBM stock as a reference to count the frequency of E. We are not interested in an "absolute" frequency, but only in frequency relative to some event type.

The *solution* of an event-mining problem $(\mathcal{S}, \gamma, E_0, \rho)$ on a given event sequence σ is the set of all complex event types derived from $\mathcal{S}$, with the following conditions: (i) E_0 is associated with the root of $\mathcal{S}$ and each event type assigned to a non-root variable X belongs to $\rho(X)$ if $\rho(X)$ is defined, and (ii) each complex event type occurs in σ with a *frequency* greater than γ.[1]

[1] The frequency here is defined as the number of times the complex event type occurs for a different occurrence of E_0 (i.e., all the occurrences using the same occurrence of E_0 for the root are counted as one) divided by the number of times E_0 occurs.

Example 2: $(\mathcal{S}, 0.8, \texttt{IBM-rise}, \rho)$ is a mining problem, where $\mathcal{S}$ is the structure in Figure 1 and ρ assigns X_3 to `IBM-fall` and assigns all other variables to all the possible event types. Intuitively, we want to discover what happens between a rise and fall of IBM stocks, looking at particular windows of time. The complex event type described in Example 1 where X_1 and X_2 are assigned respectively to `IBM-earnings-report` and `HP-rise` will belong to the solution of this problem if it occurs in the input sequence with a frequency greater than 0.8 with respect to the occurrences of `IBM-rise`.

3 Finding frequent complex event types

In [4] we introduced a variation of the timed automaton [3] that we call a *timed automaton with granularities* (TAG). Through the use of TAGs, we can find occurrences of a complex event type in an event sequence. This ability forms the basis for finding a solution to an event-mining problem.

A TAG is essentially an automaton that recognizes words. However, there is a timing information associated with the symbols of the words signifying the time when the symbol arrives at the automaton. When a timed automaton makes a transition, the choice of the next state depends not only on the input symbol read, but also on values in the clocks which are maintained by the automaton and each of which is "ticking" in terms of a specific time granularity. A clock can be set to zero by any transition and, at any instant, the reading of the clock equals the time (in terms of the granularity of the clock) that has elapsed since the last time it was reset. A constraint on the clock values is associated with any transition, so that the transition can occur only if the current values of the clocks satisfy the constraint. It is then possible to constrain, for example, that a transition fires only if the current value of a clock, say in terms of `week`, reveals that the current time is in the next week with respect to the previous value of the clock. We do not report here the formal definition of TAGs and related notions, which can be found in [4]. An example of TAG is reported in Figure 2.

To achieve our goal, we must be able, given a complex event type $\mathcal{T}$, to derive a corresponding TAG. This is guaranteed by the following theorem ([4]).

Theorem 4: Given a complex event type $\mathcal{T}$, there exists a timed automaton with granularities $\text{TAG}_{\mathcal{T}}$ such that $\mathcal{T}$ occurs in an event sequence σ iff $\text{TAG}_{\mathcal{T}}$ has an accepting run over σ. This automaton can be constructed by a polynomial-time algorithm.

The technique we use to derive the TAG corresponding to a complex event type derived from $\mathcal{S}$ is based on a decomposition of $\mathcal{S}$ into chains from the root to terminal nodes. For each chain we build a simple TAG where each transition has as input symbol the variable corresponding to a node in $\mathcal{S}$ (starting from the root), and clock constraints for the same transition correspond to the TCGs associated with the edge leading to that node. Then, we combine the resulting TAGs into a single TAG using a "cross-product" technique and we add transitions to allow the skipping of events. Finally, we change each input symbol X with the corresponding event type. A detailed procedure for TAG generation can be found in [4]. Figure 2 shows the TAG corresponding to the complex event type in Example 1.

We have also found a bound on the time needed to determine whether an event sequence is accepted by a TAG corresponding to a certain complex event type. This time is $O(|\sigma| * (|S| * min(|\sigma|, (|V| * K)^p))^2)$ time, where $|S|$ is the number of states in the TAG, $|\sigma|$ is the number of events in the input sequence, $|V|$ is the number of variables in the longest chain used in the construction of the automata, K is the size of the maximum range appearing in the constraints, and p is the number of chains used in the construction of the automata. Note that, even for reasonably complex event structures, the constant p is very small; hence, $(|V| * K)^p$ is often much smaller than $|\sigma|$.

Since we now have a tool to check occurrences of complex event types in a time sequence, a naive algorithm for discovering frequent complex event types can proceed as follows: Consider all the event types that occur in the given event sequence, and consider all the complex types derived from the given event structure, one from each

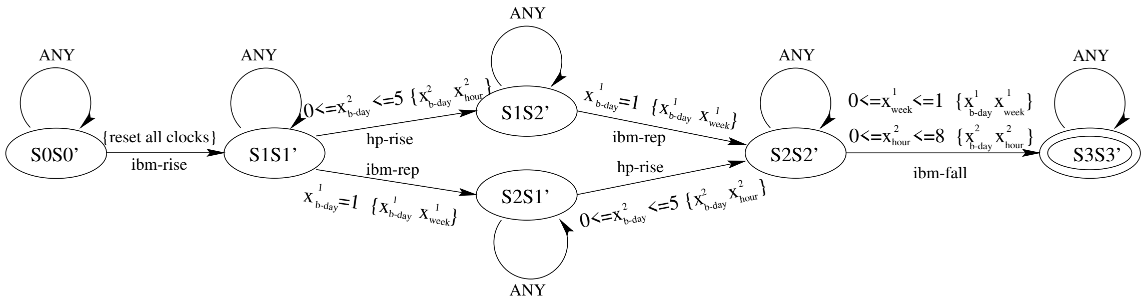


Figure 2: An example of Timed Automaton with Granularities

assignment of these event types to the variables. Each of these complex types is called a *candidate complex type* for the event-mining problem. For each candidate complex type, start the corresponding TAG at every occurrence of E_0. That is, for each occurrence of E_0 in the event sequence, use the rest of the event sequence (starting from the position where E_0 occurs) as the input to one copy of the TAG. By counting the number of TAGs reaching a final state, versus the number of occurrences of E_0, all the solutions of the event-mining problem will be derived.

This naive algorithm, however, can be too costly to implement. Indeed, the number of candidate types to be checked is exponential in the number of event types occurring in the event sequence and in $\rho(X)$ for all X, with the exponent being the number of non-root variables in the event structure. In the worst case, this implies an exponential number of TAGs, each one to be run for each occurrence of the reference event type in the sequence.

4 Achieving effectiveness by temporal reasoning and decomposition

Our strategy for finding the solutions of event-mining problems relies on the many optimization opportunities provided by the temporal constraints of the event structures. These optimizations consist in (1) identifying the possible inconsistencies in the given event structure before starting the process, (2) reducing the length of the sequence, (3) the number of times an automaton has to be started, and (4) the number of different automata to be started. The naive algorithm illustrated earlier is applied only after these steps.

Steps (1–4) exploit the implicit temporal relationships in the given event structure and a *decomposition* strategy, based on the observation that if an event-mining problem has a solution, then part of this solution is a solution also for the corresponding "sub-problem".

To derive implicit relationships, we must be able to convert TCGs from one granularity to another, not necessarily obtaining equivalent constraints, but *logically implied* ones. However, for an arbitrarily given TCG_1 and a granularity μ, it is not always possible to find a TCG_2 in terms of μ such that it is logically implied by TCG_1, i.e., any pair of events satisfying TCG_1 also satisfy TCG_2. For example, $[m, n]$ `b-day` is not implied by $[0, 0]$ `day` no matter what m and n are. The reason is that $[0, 0]$`day` is satisfied by any two events that happen during the same day, whether the day is a business day or a weekend day.

In our framework, we allow a conversion of a TCG in an event structure into another TCG if the resulting constraint is implied by the set of all the TCGs in the event structure.

In [5, 4] the interested reader can find approximate polynomial algorithms for conversion of TCGs and for the derivation of implicit TCGs in the event structure. Since there are possibly infinite implicit TCGs, the algorithm's goal is to approximates the *tightest* ones. Referring to the event structure of Figure 1 the algorithm would derive, among others, TCG $[0, 127]$ `hour` between X_1 and X_3, and TCG $[0, 1]$ `week` between X_0 and X_3. Consider the last one: this is the tightest in terms of `week` between those two variables, since no solution to the temporal constraints can lead to a distance less than 0, nor greater than 1.

When an unsatisfiable TCG is derived from an event structure by the algorithm, we can avoid starting the mining process, since this means that an inconsistent structure specification has been given. Implicit TCGs are also the basis for steps 2-4. In this paper we limit the discussion to step 4 which is probably the most effective of these optimization techniques. We refer to [4] for a detailed description of 2 and 3.

The basic idea of step 4 is as follows: If a complex event type occurs frequently, then any of its sub-type should also occur frequently. (This is similar to [7].) Here by a *sub-type* of a complex type $\mathcal{T}$, we mean a complex event type, induced by a subset of variables, such that each occurrence of the sub-type can be "extended" to an occurrence of $\mathcal{T}$. However, not every subset of variables of a structure can induce a sub-structure. For example, consider the event structure in Figure 1 and let $\mathcal{S}' = (\{X_0, X_3\}, \{(X_0, X_3)\}, \Gamma')$. $\mathcal{S}'$ cannot be an induced sub-structure, since it is not possible for Γ' to capture precisely the four constraints of that structure. This forces us to consider approximated sub-structures.

Let $\mathcal{S} = (W, A, \Gamma)$ be an event structure and M the set of all the granularities appearing in Γ. For each $\mu \in M$, let C_μ be the collection of constraints that we derive at the end of the approximate algorithm for the derivation of implicit constraints. Then, for each subset W' of W, the *induced approximated sub-structure of* W' is (W', A', Γ'), where A' consists of all pairs $(X, Y) \subseteq W' \times W'$ such that there is a path from X to Y in $\mathcal{S}$ and there is at least a constraint (original or derived) on (X, Y). For each $(X, Y) \in A'$, the set $\Gamma'(X, Y)$ contains all the constraints in C_μ on (X, Y) for all $\mu \in M$. For example, $\Gamma'(X_0, X_3)$ in the previous paragraph contains $[0, 1]\,\mathtt{week}$ and $[1, 175]\,\mathtt{hour}$. Note that if a complex event matches $\mathcal{S}$ using events from σ, then there exists a complex event using events from a sub-sequence σ' of σ that matches the sub-structure $\mathcal{S}'$.

By using the notion of an approximated sub-structure, we proceed to reduce candidate event types as follows: Suppose the event-mining problem is $(\mathcal{S}, \gamma, E_0, \rho)$. For each variable X appearing in $\mathcal{S}$, except the root X_0, consider the approximated sub-structure $\mathcal{S}'$ induced from X_0 and X (i.e., two variables). If there is a relationship between X_0 and X (i.e., $\Gamma'(X_0, X) \neq \emptyset$), consider the event-mining problem (called *induced mining problem*) $(\mathcal{S}', \gamma, E_0, \rho')$, where ρ' is a restriction of ρ with respect to the variables in $\mathcal{S}'$. The key observation is ([7]) that if no solution to any of these induced mining problems assigns event type E to X, then there is no need to consider any candidate complex type that assigns E to X. This reduces the number of candidate event types for the original mining problem.

To find the solutions to the induced mining problems is rather straightforward and simple in time complexity. Indeed, the induced sub-structure gives the distance from the root to the variable (in effect, two distances, namely the minimum distance and the maximum distance). For each occurrence of E_0, this distance translates into a window, i.e., a period of time during which the event for X must appear. If the frequency (i.e., the number of windows in which the event occurs divided by the total number of these windows) an event type E occurs is less than or equal to γ, then any candidate complex type with X assigned to E can be "screened out" for further consideration. Consider the mining problem of Example 2 with the simple variation that $\rho = \emptyset$, i.e. all non-root variables are free. $(\mathcal{S}', 0.8, \mathtt{IBM{-}rise}, \emptyset)$ is one of its induced mining problems. $\Gamma'(X_0, X_3)$, through the constraints reported above, identifies a window for X_3 for each occurrence of `IBM-rise`. It is easy to screen out all candidate event types for X_3 that have a frequency of occurrence in these windows less than 0.8.

The above idea can easily be extended to consider induced approximated sub-structures that include more than one non-root variable. For each integer $k = 2, 3, \ldots$, consider all the approximated sub-structures $\mathcal{S}_k$ induced from the root variable and k other variables in $\mathcal{S}$, where these variables (including the root) form a sub-chain in $\mathcal{S}$ (i.e., they are all on a particular path from the root to a particular leaf), and $\mathcal{S}_k$, considering the derived constraints, forms a connected graph. We now find the solutions to the induced event-mining problem $(\mathcal{S}_k, \gamma, E_0, \rho_k)$. Again, if no solution assigns an event type E to a variable X, then any candidate complex type that has this assignment is screened out. To find the solutions to these induced mining problems, the naive algorithm mentioned earlier can be used. Of course, any screened-out candidates from previous induced mining problems should not be considered any further. This means that if in a previous step only k event types have been assigned to variable X as a solution of a mining problem, if the current problem involves variable X, we consider only candidates within those k event types. This process can be extended to event types assigned to combinations of variables. This process results, in practice, in a smaller number of candidate types for induced mining problems.

5 Conclusion

In this paper, we briefly discussed the process of mining sequential relationships that involve multiple granularities. We first precisely formulated the problem and then developed an effective procedure. We have conducted several experiments (see [4]) which show that the heuristics we introduce are especially powerful in reducing the number of candidate event types.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *IEEE Trans. Knowledge and Data Engineering*, 5(5):914–925, 1990.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *International Conference on Database Engineering*, pp. 3–14. IEEE, 1995.

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[4] C. Bettini, X. S. Wang, S. Jajodia, and J. Lin. Discovering Frequent Event Patterns With Multiple Granularities in Time Sequences. *IEEE Trans. Knowledge and Data Engineering*, to appear.

[5] C. Bettini, X. Wang, and S. Jajodia. A General Framework for Time Granularity and its Application to Temporal Reasoning. *Annals of Mathematics and Artificial Intelligence*, Baltzer Science Publishers, 1998.

[6] P. Laird. Identifying and using patterns in sequential data. In *Algorithmic Learning theory, 4th International Workshop*, pp. 1–18. Springer-Verlag, 1993.

[7] H. Mannila, H. Toivonen, and A.I. Verkamo. Discovering frequent episodes in sequences (extended abstract). In *Proc. 1st Conference on Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, pp. 210-215, 1995.

[8] X. Wang, C. Bettini, A. Brodsky, and S. Jajodia. Logical design for temporal databases with multiple granularities. *ACM Transactions of Database Systems*, 22(2):115–170, June 1997.

[9] J. Tsong-Li Wang, G. Chirn, T.G. Marr, B.A. Shapiro, D. Shasha, and K. Zhang. Combinatorial pattern discovery for scientific data: Some preliminary results. In *Proc. of SIGMOD Conference*, pages 115–125. ACM Press, May 1994.

[10] K. Wang and J. Tan. Incremental discovery of sequential patterns. In *Proc. of Workshop on Research Issues on Data Mining and Knowledge Discovery, in cooperation with ACM-SIGMOD'96 and IRIS/Precarn*, Montreal, Canada, June 1996.

Mining Databases: Towards Algorithms for Knowledge Discovery

Usama Fayyad
Microsoft Research
fayyad@microsoft.com
http://research.microsoft.com/˜fayyad

1 Introduction

Data Mining and Knowledge Discovery in Databases (KDD) are rapidly evolving areas of research that are at the intersection of several disciplines, including statistics, databases, pattern recognition/AI, optimization, visualization, and high-performance and parallel computing. The increased recent attention is primarily because many more people now have "databases". This is turn is driven by two factors: the ever decreasing cost of computers and storage media, and the success of database systems in becoming a mainstay of many activities in business, science, and government.

With the widespread use of databases and the explosive growth in their sizes, individuals and organizations are faced with the problem of making use of this data. Traditionally, "use" of data has been limited to querying a reliable store via some well-circumscribed application or canned report-generating entity. While this mode of interaction is satisfactory for a wide class of well-defined processes, it was not designed to support data exploration, decision support applications, and ad hoc querying of the data.

Now that capturing data and storing it has become easy and inexpensive, certain questions begin to naturally arise: Will this data help my business gain an advantage? How can we use historical data to build models of underlying processes that generated such data? Can we predict the behavior of such processes? How can we "understand" the data? These questions become particularly important in the presence of massive data sets. A large database represents a large body of information that is presumed to be valuable since it records vital measurements, be it a business venture, a scientific endeavor, or the operations of a government entity. Yet frequently, this potentially valuable data resource is far from being effectively accessible. The current interfaces between humans and storage systems do not support navigation, exploration, summarization, or modeling of large databases. Providing these types of capabilities and more is the goal of the emerging research area of Data Mining and Knowledge Discovery in Databases.

1.1 From OLTP to Decision Support

While transactional systems provide effective solutions to the problems of reliable logging, book-keeping, availability, and recovery, there has been little emphasis on supporting summarization, aggregation, and ad hoc querying over transactional stores. A recent wave of activity in the database field, called *data warehousing*, has been concerned with turning transactional data into more traditional relational databases that can be queried for summaries and aggregates of transactions. Data warehousing also includes the integration of multiple sources of data

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

along with handling the host of problems associated with such an endeavor. These problems include: dealing with multiple data formats, multiple database management systems (DBMS), distributed databases, unifying data representation, data cleaning, and providing a unified logical view of an underlying collection of non-homogeneous databases.

Data warehousing is the first step in transforming a database system from a system whose primary purpose is *reliable storage* to one whose primary use is *decision support*. A closely related area is called On-Line Analytical Processing (OLAP) [6]. The current emphasis of OLAP systems is on supporting query-driven exploration of the data warehouse. Part of this entails pre-computing aggregates along data "dimensions" in the multi-dimensional data store. Because the number of possible aggregates is exponential in the number of "dimensions", much of the work in OLAP systems is concerned with deciding which aggregates to pre-compute and how to derive other aggregates (or estimate them reliably) from the pre-computed projections. There are also very interesting questions to be answered regarding how much of the cube to materialize, how to represent it and access it, and which dimensions should be chosen for aggregation since it is not possible to build a cube on all available fields.

1.2 Why Data Mining?

In the OLAP framework, the analysis and exploration is driven entirely by the human analyst. Hence, OLAP may be viewed as extending the SQL querying framework to accommodate queries that if executed on a relational DBMS would be computationally prohibitive. Unlike OLAP, data mining techniques allow for the possibility of computer-driven exploration of the data. This opens up the possibility for a new way of interacting with databases: specifying queries at a much more abstract level than SQL permits. It also facilitates data exploration for problems that, due to high-dimensionality, would otherwise be very difficult to explore by humans, regardless of difficulty of use of, or efficiency issues with, SQL.

A problem that has not received much attention in database research is the *query formulation problem*: how can we provide access to data when the user does not know how to describe his goal in terms of a specific query? Examples of this situation are fairly common in decision support situations. For example, in a business setting, say a credit card or telecommunications company would like to query its database of usage data for records representing fraudulent cases. In a science data analysis context, a scientist dealing with a large body of data would like to request a catalog of events of interest appearing in the data. Such patterns, while recognizable by human analysts on a case by case basis are typically very difficult to describe in a SQL query or even as a computer program in a stored procedure. A more natural means of interacting with the database is to state the query by example. In this case, the analyst would label a training set of cases of one class versus another and let the data mining system build a model for distinguishing one class from another. The system can then apply the extracted classifier to search the full database for events of interest. This is typically easier because examples are usually readily available, and humans find it natural to interact at the level of cases.

Another major problem which data mining could help alleviate is that humans find it particularly difficult to visualize and understand a large data set. Data can grow along two dimensions: the number of fields (also called dimensions or attributes) and the number of cases. Human analysis and visualization abilities do not scale to high-dimensions and massive volumes of data. A standard approach to dealing with high-dimensional data is to project it down to a low-dimensional space and attempt to build models in this simplified subspace. As the number of dimensions grow, the number of choice combinations for dimensionality reduction explode. Furthermore, a projection to lower dimensions could easily transform a relatively easy discrimination problem into one that is extremely difficult. In fact, some mining algorithms (e.g. support vector machines [26]) employ a reverse technique where dimensionality is purposefully increased to render the classification problem easy (linear).

However, even if one is to accept that dimensionality reduction is necessary if exploration is to be guided by a human, this still leaves a significant projection selection problem to solve. It is infeasible to explore all of the ways of projecting the dimensions or selecting the right subsamples (reduction along columns and rows). An effective means to visualize data would be to employ data mining algorithms to perform the appropriate reductions. For example, a clustering algorithm could pick out a distinguished subset of the data embedded in a high-dimensional space and proceed to select a few dimensions to distinguish it from the rest of the data or from other clusters. Hence a much more effective visualization mode could be established: one that may enable an analyst to find patterns or models that may otherwise remain hidden in the high-dimensional space.

Another factor that is turning data mining into a necessity is that the rates of growth of data sets exceed by far any rates with which traditional "manual" analysis techniques could cope. Hence, if one is to utilize the data in a timely manner, it would not be possible to achieve this goal in the traditional data analysis regime. Effectively this means that most of the data would remain unused. Such a scenario is not realistic in any competitive environment where those who better utilize data resources will gain a distinct advantage. This sort of pressure is present in a wide variety of organizations, spanning the spectrum from business, to science, to government. It is leading to serious reconsideration of data collection and analysis strategies that are nowadays causing the accumulation of huge "write-only" data stores.

2 KDD and Data Mining

The term *data mining* is often used as a synonym for the process of extracting useful information from databases. In this paper, as in [9], we draw a distinction between the latter, which we call KDD, and "data mining". The term *data mining* has been mostly used by statisticians, data analysts, and the database communities. The earliest uses of the term come from statistics and its usage in most settings was associated with negative connotations of blind exploration of data without a priori hypotheses to be verified. However, notable exceptions can be found. For example, as early as 1978 [16], the term is used in a positive sense in a demonstration of how generalized linear regression can be used to solve problems that are very difficult for humans and the traditional statistical techniques.

The term KDD was coined at the first KDD workshop in 1989 [20] to emphasize that "knowledge" is the end product of a data-driven discovery. In our view, KDD refers to the overall *process* of discovering useful knowledge from data while *data mining* refers to a particular *step* in this process. Data mining is the application of specific algorithms for extracting patterns from data. The additional steps in the KDD process, such as data preparation, data selection, data cleaning, incorporating appropriate prior knowledge, and proper interpretation of the results of mining, are essential to ensure that useful knowledge is derived from the data. Blind application of data mining methods (rightly criticized as "data dredging" in the statistical literature) can be a dangerous activity easily leading to discovery of meaningless patterns.

We give an overview of the KDD process in Figure 1. Note that in the KDD process, one typically iterates many times over previous steps and the process is fairly messy with plenty of experimentation. For example, one may select, sample, clean, and reduce data only to discover after mining that one or several of the previous steps need to be redone. We have omitted arrows illustrating these potential iterations to keep the figure simple.

We adopt the definitions of KDD and Data mining provided in [9] as follows:

Knowledge Discovery in Databases: is *the process of identifying valid, novel, potentially useful, and ultimately understandable structure in data.* Here, data is a set of facts (cases in a database) and structure refers to either patterns or models. A pattern is an expression representing a parsimonious description of a subset of the data. A model is a representation of the source generating the data. The term process implies that KDD is comprised of many steps (Figure 1) which involve data preparation, search for patterns, knowledge evaluation, and refinement, all potentially repeated in multiple iterations.

Data Mining: is a *step in the KDD process that, under acceptable computational efficiency limitations, enumerates structures (patterns or models) over the data.* There are many (potentially infinitely) more patterns/models over a finite data set than there are data records. To be deemed knowledge, the derived structure must pass certain criteria. Notions of utility and validity have classical definitions in decision analysis and statistics. While it is possible to define quantitative measures for *certainty* (e.g., estimated prediction accuracy on new data) or *utility* (e.g. gain, perhaps in dollars saved due to better predictions or speed-up in response time of a system), no-

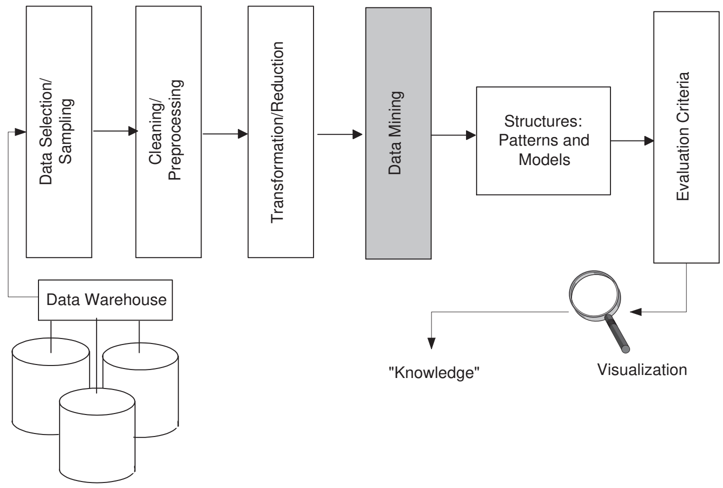


Figure 1: An overview of the steps comprising the KDD process.

tions such as *novelty* and *understandability* are much more subjective and difficult to define. In certain contexts understandability can be estimated by simplicity (e.g., the number of bits to describe a pattern). Sometimes the measures are combined under a single *interestingness* measure (e.g., see [23] and references within). Interestingness functions can be explicitly defined or can be manifested implicitly via an ordering placed by the KDD system on the discovered patterns or models. The term knowledge in KDD is user-oriented, domain-specific, and determined by the interestingness measure; it is not a general (e.g. philosophical) definition.

The data mining component of the KDD process is concerned with the algorithmic means by which patterns are extracted and enumerated from data. The overall KDD process (Figure 1) includes the *evaluation* and possible *interpretation* of the "mined" patterns to determine which patterns may be considered new "knowledge."

3 Data Mining Methods: An Overview

Data mining techniques can be divided into five classes of methods. These methods are listed below. While many of these techniques have been historically defined to work over memory-resident data and not much attention has been given to integrating them with database systems, some of these techniques are beginning to be scaled to operate on large databases. Examples in classification (Section 3) include decision trees [18], in summarization (Section 3) association rules [1], and in clustering [28]

Predictive Modeling

The goal is to predict some field(s) in a database based on other fields. If the field being predicted is a numeric (continuous) variable (such as a physical measurement of e.g. *height*) then the prediction problem is a *regression* problem. If the field is categorical then it is a *classification* problem. There is a wide variety of techniques for classification and regression. The problem in general is to determine the most likely value of the variable being predicted given the other fields (inputs), the training data (in which the target variable is given for each observation), and a set of assumptions representing one's prior knowledge of the problem.

Linear regression combined with non-linear transformation on inputs could be used to solve a wide range of problems. Transformation of the input space is typically the difficult problem requiring knowledge of the problem and quite a bit of "art". In classification problems, this type of transformation is often referred to as "feature extraction".

In classification, the basic goal is to predict the most likely state of a categorical variable (the class). This is fundamentally a density estimation problem. If one can estimate the probability that the class $C = c$, given the other fields $X = x$ for some feature vector x, then one could derive this probability from the joint density on

C and X. However, this joint density is rarely known and very difficult to estimate. Hence, one has to resort to various techniques for estimating. These techniques include:

1. Density estimation, e.g. kernel density estimators [7, 22] or graphical representations of the joint density [14].
2. Metric-space based methods: define a distance measure on data points and guess the class value based on proximity to data points in the training set. An example is the K-nearest-neighbor method [7].
3. Projection into decision regions: divide the attribute space into decision regions and associate a prediction with each. For example, linear discriminant analysis finds linear separators, while decision tree or rule-based classifiers make a piecewise constant approximation of the decision surface. Neural nets find non-linear decision surfaces.

Clustering

Also known as segmentation, clustering does not specify fields to be predicted but targets separating the data items into subsets that are similar to each other. Since, unlike classification, we do not know the number of desired "clusters", clustering algorithms typically employ a two-stage search: An outer loop over possible cluster numbers and an inner loop to fit the best possible clustering for a given number of clusters. Given the number k of clusters, clustering methods can be divided into three classes:

1. Metric-distance based methods: a distance measure is defined and the objective becomes finding the best k-way partition such as cases in each block of the partition are closer to each other (or centroid) than to cases in other clusters.
2. Model-based methods: a model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each cluster. If M_ℓ is the model hypothesized for cluster ℓ, ($\ell \in \{1, \ldots, k\}$), then one way to score the fit of a model to a cluster is via the likelihood:
$$\text{Prob}(M_\ell|D) = \text{Prob}(D|M_\ell)\frac{\text{Prob}(M_\ell)}{\text{Prob}(D)} \tag{1}$$
The prior probability of the data D, $\text{Prob}(D)$ is a constant and hence can be ignored for comparison purposes, while $\text{Prob}(M_\ell)$ is the prior assigned to a model. In maximum likelihood techniques, all models are assumed equally likely and hence this term is ignored. A problem with ignoring this term is that models that are more complex are always preferred and this leads to overfitting the data.
3. Partition-based methods: basically, enumerate various partitions and then score them by some criterion. The above two techniques can be viewed as special cases of this class. Many AI techniques fall into this category and utilize ad hoc scoring functions.

Note again that the problem is fundamentally *statistical* in nature. If one had access to the *joint probability density* function governing the data source, then the problem of clustering becomes the problem of determining the "modes' (maxima) in the density function. Hence, one approach is to estimate the density and then go "bump-hunting" [12]. Unfortunately, in high-dimensional data, neither density estimation (e.g. kernel-density estimation [22] nor bump-hunting are feasible [12]. Hence, methods within the above three classes are necessary. **Data Summarization**

Sometimes the goal is to simply extract compact patterns that describe subsets of the data. There are two classes of methods which represent taking horizontal (cases) or vertical (fields) slices of the data. In the former, one would like to produce summaries of subsets: e.g. sufficient statistics, or logical conditions that hold for subsets. In the latter case, one would like to predict relations between fields. This class of methods is distinguished from the above in that the goal is to find relations between fields. One common method is called association rules [1]. Associations are rules that state that certain combinations of values occur with other combinations of values with a certain frequency and certainty. A common application of this is market basket analysis were one would like to summarize which products are bought with what other products. While there are exponentially many rules, due to data sparseness only few such rules satisfy given support and confidence thresholds. Scalable algorithms find all such rules in linear time (for reasonable threshold settings). While these rules should not be viewed as statements about causal effects in the data, they are useful for modeling purposes if viewed as frequent marginals in a discrete (e.g. multinomial) probability distribution. Of course to do proper inference one needs to know the frequent, infrequent, and all probabilities in between. However, approximate inference can sometimes be useful.

Dependency Modeling

Insight into data is often gained by deriving some causal structure within the data. Models of causality can be probabilistic (as in deriving some statement about the probability distribution governing the data) or they can be deterministic as in deriving functional dependencies between fields in the data [20]. Density estimation methods in general fall under this category, as do methods for explicit causal modeling (e.g. [13] and [14]).

Change and Deviation Detection

These methods account for sequence information, be it time-series or some other ordering (e.g. protein sequencing in genome mapping). The distinguishing feature of this class of methods is that ordering of observations is important. Scalable methods for finding frequent sequences in databases, while in the worst-case exponential in complexity, do appear to execute efficiently given sparseness in real-world transactional databases [17].

4 Discussion: Issues and Problems

The problem of mining data is not an easy one. The classes of data mining methods all include *ill-posed problems*: many solutions exist for a given problem, with no clear means of judging a solution's quality. This is fundamentally different from the type of challenge faced with familiar well-defined problems like: sorting data, matching a query to records, joining two tables, or restoring a database.

Some degree of caution needs to be taken when approaching data mining problems. In this section, we shall consider a few examples of how these problems can manifest themselves in sometimes subtle ways. Hence, one needs to be careful to avoid the *mines*[1] associated with data mining.

The Curse of Dimensionality

Data in higher dimensional spaces can behave in ways that are quite puzzling and counter-intuitive. Many examples can be found in the literature (for a good exposition see [11]). Notions on "neighborhood" that have a certain meaning, typically attached to "physical proximity", in two and three dimensions, completely break down as one moves to higher dimensions. Let us consider a simple example: suppose that one is given a data set, and one knows that data in this data set is drawn from a mixture of two Gaussian distributions. Furthermore, assume that the Gaussians are very simple: the covariance matrix is diagonal, and all diagonal entries are the same, say a value like 2.0[2]. Intuitively, the data distributions look like two "spheres" in whatever dimensions the data are in, with "most" of the data appearing "closer" to the centers of the two spheres.

Now in the familiar one-dimensional case, one can safely say that *the majority of the data* in each Gaussian falls within a "distance" of one standard deviation (1-σ) of the mean of the Gaussian. Many readers are likely to be familiar with the fact that some 68% of the data falls within this interval. But now let us go to higher dimensions while preserving the same distribution. For dimensionality $d = 12$, how much of the data is within a Euclidean distance of 1-σ of the mean (center) of the Gaussian? The answer may surprise many: *less than 1% of the data*. In fact, if we let dimensionality go to $d = 100$, then *there is practically no data "near" the mean* (chance of an item falling anywhere within the sphere at the center with radius = 1-σ is $< 10^{-16}$). This means that essentially *all of the data is "far" from the mean!*. What significance do we attach to being "far" from the mean in this case?

[1] Ones of the explosive variety.

[2] we make this choice to avoid having to explain the *Mahalanobis distance* and simply use Euclidean distance in our example.

Correlation versus Causality

Having access to a large amount of data, one may be tempted to attach special significance to *correlation* of events. For example, assume that one has found that in a set of supermarket transactions, the following associations are high: "buy(hot dogs) $\longrightarrow$ buy(steak sauce)" and "buy(hamburger) $\longrightarrow$ buy(steak sauce)"[3]. These associations might lead a store manager to conclude that hot dogs and hamburgers are causally linked to sales of steak sauce. In fact, the real reason for this correlation might be that generally people who like to eat meat, in all its varieties, also happen to buy more steak sauce than the general population.

Now suppose a sales manager, motivated to increase the sales of steak sauce, which happens to be a high profit margin item, decides to start a promotion designed to drive the sales of steak sauce. Perhaps giving away hot dogs might cause people to buy more steak sauce? Now in this example, basic common sense and the semantics of the items might prevent the manager from embarking on this lossy policy. However, often these correlations can link items in a store in relationships that might seem *mysterious and intriguing*. Situations under which correlation can lead to causality are not straightforward and are the subject of much study [13, 19].

Association rules can be viewed as an efficient and scalable means for finding frequent marginals in the data. However, the use of these marginals in probabilistic inference requires care, as the infrequent marginals can be just as important. For a related discussion, see [24].

Interestingness

Which patterns are likely to be interesting to the user? For example, suppose one is to run association rules over a demographics database containing age, sex, marital status, family conditions, etc. A strong association[4] might take the form of a statement that essentially says: "With confidence better than 99%, in this database, all individuals who are *husbands* also happen to be *males*." While this rule might strike many as trivial, it may actually have interesting side-effects. In this example, an interesting question is raised by the fact that the association does not hold with 100% confidence (due to errors in data cleaning). Furthermore, how should a machine decide that this association is less interesting than one that might be extremely informative, as in the next example.

Lest the reader be misled into thinking that association rules are not really interesting, I'll quote an example from IBM's data mining group. In one application to data from the Australian government healthcare agency, association rules found that there was a strong correlation between two billing codes charged by physicians to the government insurance. In this case, this was a double-billing practice and its detection is expected to save the government tens of millions of dollars. Note that evaluated purely on a syntactic basis, there is little reason to distinguish between this valuable association and the one about husbands being males above.

Model Complexity

In general, it should be pointed out that a model that perfectly fits the data (i.e. an apparently optimal solution) may be less desirable than one that does not fit the data. This is known as *overfit* or *overspecialization*. The problem of trading off the simplicity of a model with how well it fits the training data is a well-studied problem. In statistics this is known as the *bias-variance tradeoff* [11], in Bayesian inference it is known as *penalized likelihood* [2, 14], and in pattern recognition/machine learning it manifests itself as the *minimum message length* (MML) [27] problem. The MML framework, also called minimum description length (MDL) [21] dictates that the best model for a given data set is one that minimizes the coding length of the data and the model combined.

If a model fits the data exactly, the data need not be encoded and the cost is that of coding the model. If the data is not represented by a model, the cost is that of encoding the data. One can show that minimizing the MDL is equivalent to selecting the model that minimizes the Bayes risk assuming cost of errors is uniform, i.e. for a data set D, the MDL prefers the model M for which $\text{Prob}(M|D)$ is maximized. This can be shown by a simple application of Bayes rule (see equation 1) which, after taking the logarithm of each side reduces to

$$-\log(\text{Prob}(M|D)) = -\log(\text{Prob}(D|M)) - \log(\text{Prob}(M)) + log(\text{Prob}(D))$$

[3]Example provided by P. Spirtes at U. Washington/Microsoft Summer Research Inst. on Data Mining (1997) .

[4]This is a real example originally provided by Ronny Kohavi of SGI's Data Mining Group.

Noting that Prob(D) is a constant for all models being compared, and that the minimal cost of encoding an object requires at least logarithm of its probability in bits, we see that MDL is indeed calling for choosing the model with the maximum likelihood given the data. The lesson here is that a penalty must be paid if more complex models are used. The formula above gives the appropriate tradeoff. Of course, in many situations, determining the minimal encoding cost of a hypothesis and the data given the hypothesis may be just as difficult as the original problem of finding the best model.

Enumerating a Plurality of Models

The enumeration of many patterns over data can be a dangerous activity. For example, assume one is given a database that happens to be generated completely randomly: i.e. there are no dependencies between fields and hence no models or patterns should be valid in this data, since by definition there are no regularities in it. However, because there are many more patterns/models possible than data points in a finite database, some patterns/models are likely to fit the data by pure chance. When testing many such models, it can be shown that with probability rapidly approaching 1.0, some pattern/model is likely to fit the data nicely. There are methods for adjusting for such random chance (e.g. *Bonferroni adjustment*) but these are not very effective. One needs to bear it mind when it is time to draw conclusions or summarize "knowledge" from the data.

5 Is There Any Hope?

The previous section could leave the reader in a state of doubt: with a path littered with so many explosive *mines*, is there any hope to data mining endeavors? The answer is a resounding yes. Indeed, many data mining activities have resulted in great results. For examples of applications in science data analysis, see [10] and in industry see [4]. The primary reasons for hope in data mining include:

Database Regularity. Most databases exhibit a good degree of regularity and do not approach the theoretically possible levels of "difficulty". For example, consider data on user supermarket shopping, or equivalently people browsing web pages. While the space of all possible behavior is huge, extending from thousands to millions of dimensions, with as many cases (data points), most data on these processes exhibits a merciful dose of regularity. Real data typically occupies a very low dimensional subspace of the space of possible cases. Not only is the data extremely sparse, but frequent events are fairly clustered. This is evidenced by association rule algorithms that often run in linear time and almost never approach the theoretical worst-case behavior of exponential complexity.

Humans cannot process lots of data, especially in high-dimensional spaces. This renders alternative computational means for extracting a solution (e.g. data mining techniques), even if heuristic and suboptimal, of high value to users. The absence of a viable "manual" alternative makes any solution better than no solution.

Sufficient statistics to the rescue. Many algorithms for classification and clustering do not require direct access to the data, only to sufficient statistics about the data. Hence, even though most implementations of statistical techniques assume data is in memory, the usage of such data is typically limited to deriving the sufficient statistics from it. This observation suggests a nice decomposition between what a server needs to provide (data intensive operations to derive sufficient statistics) versus what a data mining "client" needs to do (consume sufficient statistics to build a model). This is likely to be a good direction towards scaling mining algorithms to effectively work with large databases. See [18, 5, 28, 3] for examples in classification and clustering.

Partitioning methods are suitable to large databases. Most data mining methods are essentially partitioning methods: find local partitions in the data and build a model for each local region. This is a particularly suitable model for large databases. Large databases grow over time, with the underlying data generating source changing properties over time. While most methods in statistics assume the data is governed by some static global model, partitioning based methods do cope well with such data, often better than humans or traditional statistical models (which attempt to fit a single model to the entire data).

Greedy heuristics seem to do just fine. Frequent experience is that greedy heuristic algorithm that examine one dimension at a time, locally ignoring interactions between dimensions, appear to perform nicely over real-world data sets. This surprising fact is responsible for many of the successes of data mining [10, 4].

Sampling can help. It may not be necessary or desirable to work over the entire contents of a large database. If the model being built is simple, then using lots of data may not be called for. Recall that if all one is doing is estimating the average value of a column, then a small sample of a few hundred points is more than sufficient, a few thousand is overkill. The value computed from a few thousand cases is virtually indistinguishable from one computed over millions of cases. Of course, some operations such as clustering may be more effective with larger samples, and sometimes getting the random sample does not buy one any advantages over operating on the entire database.

6 Concluding Remarks

Successful KDD applications continue to appear, driven mainly by a glut in databases that have clearly grown to surpass raw human processing abilities. For examples of success stories in commercial applications see [4] and in science analysis see [10]. More detailed case studies are found in [8]. Scalable methods for decision trees, clustering, nearest neighbor, and density estimation have been developed and can work with very large databases.

The fundamental problems of data analysis and how to mechanize it are fairly difficult. Scaling mining methods to work efficiently on large databases involves considering issues of limited main memory, efficient indexing structures, and effective sampling methods tightly coupled to model construction. Issues of representing metadata (information of data lineage, transformations, and properties) and its use in data cleaning, mining, and visualization and mining distributed stores where data movement between client and server must be minimized are important challenges. Often mining is desirable over non-homogenous data sets (including mixtures of multimedia, video, and text modalities); current methods assume fairly uniform and simple data structure.

While operating in a very large sample size environment is a blessing against overfitting problems, data mining systems need to guard against fitting models to data by chance. This problem becomes significant as an algorithm explores a huge search space over many models for a given data set. Also, traditional statistical methods for assessing significance were not formulated to operate in large sample environments. If one has lots of data, then almost all "events" become statistically significant. For purposes of visualization and reporting, proper tradeoffs between complexity and understandability of models are needed. Finally, an important challenge is to develop theory and techniques to model growth and change in data. Large databases, because they grow over a long time, do not grow as if sampled from a static probability density. The question of *how does the data grow?* needs to be better understood; see articles by P. Huber, by Fayyad and Smyth, and by others in [15]. For a more extensive list of research problems in data mining and KDD, the reader is referred to the Editorials of issues 1:1 and 2:2 of the journal: *Data Mining and Knowledge Discovery*. These are accessible on-line at http://research.microsoft.com/datamine. Interested readers will also find links to many related web sites there.

Data mining and KDD are inherently interdisciplinary fields, and meaningful advances will require using techniques from a variety of fields. An examination of the literature on data mining techniques in different communities quickly reveals many problems arising from the fact that the various constituent communities do not interact closely. Techniques developed in statistics typically pay little to no attention to questions of scale, dealing with database issues, and restricted access to data. Approaches from the database community are often lacking on the statistical side, sometimes with fairly blatant violations of basic statistical principles. Techniques developed in the data visualization community are typically divorced from data mining or database issues. Visualization techniques exist to visualize a few dimensions effectively (sometimes up to 10 dimensions), but little attention is paid to *which* dimensions out of the hundreds/thousands possible, and which of the exponentially many possible combinations of dimensions, are to be chosen for visualization. Techniques from optimization theory and mathematical programming often ignore both database and statistical issues.

It is my hope that the emerging new field of Data Mining and KDD will bring together the proper mix of techniques, to bring us closer to the ultimate goal: enabling the effective use of large stores and allowing us to derive value from our capability to store large amounts of data. It may be instructive to take a global view of where we stand today in terms of our ability to understand, navigate, and exploit the digital information universe we now find ourselves embedded in. If I were to draw on a historical analogy of where we stand today with regards to digital information manipulation, navigation, and exploitation, I find myself thinking of Ancient Egypt. We can build large impressive structures. We have demonstrated abilities at the grandest of scales in being able to capture data and construct huge data warehouses. However, our ability to navigate the digital stores and truly make use of their contents, or to understand how they can be exploited effectively is still fairly primitive. A large data store today, in practice, is not very far from being a grand, write-only, data tomb. It is my hope that the recent flurry in activity in data mining and KDD will advance us a little towards bringing some life into our data pyramids.

References

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and I.C. Verkamo. Fast discovery of association rules. In *Advances in knowledge Discovery and Data Mining*, pages 307 – 328. MIT Press, Cambridge, MA, 1996.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.

[3] P. Bradley, U. Fayyad, and C. Reina. "Scaling Clustering Algorithms to Large Databases". *Microsoft Research Technical Report*, 1998.

[4] R. Brachman, T. Khabaza, W. Kloesgen, G. Piatetsky-Shapiro, and E. Simoudis. Industrial applications of data mining and knowledge discovery. *Communications of ACM*, 39(11), 1996.

[5] S. Chaudhuri, U. Fayyad, J. Bernhardt. "Scalable Classification over SQL Databases". *Microsoft Research Technical Report*, 1998.

[6] E.F. Codd. Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate. Technical report, E.F. Codd and Associates, 1993.

[7] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. New York: John Wiley, 1973.

[8] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. MIT Press, Cambridge, MA, 1996.

[9] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In *Advances in knowledge Discovery and Data Mining*, pages 1 – 36. MIT Press, Cambridge, MA, 1996.

[10] U. Fayyad, D. Haussler, and P. Stolorz. Mining science data. *Communications of ACM*, 39(11), 1996.

[11] J. Friedman. On bias, variance, 0/1 - loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1), 1997.

[12] K. Fukunaga. *Statistical Pattern Recognition*. Academic Press, NY, 1990.

[13] C. Glymour, R. Scheines, and P. Spirtes ABD K. Kelly. *Discovering Causal Structure*. Academic Press, New York, 1987.

[14] D. Heckerman. Bayesian networks for data mining. *Data Mining and Knowledge Discovery*, 1(1), 1997.

[15] J. Kettenring and D. Pregibon, editors. *Statistics and Massive Data Sets, Report to the Committee on Applied and Theoretical Statistics*, Washington, D.C., 1996. National Research Council.

[16] E.E. Leamer. *Specification searches: ad hoc inference with nonexperimental data*. Wiley, New York, 1978.

[17] H. Mannila, H. Toivonen, and A.I. Verkamo. Discovery of frequent episodes in event sequence. *Data Mining and Knowledge Discovery*, 1(3), 1997.

[18] M. Mehta, R. Agrawal, and J. Rissanen. Sliq: a fast scalable classifier for data mining. In *Proceedings of EDBT-96*. Springer Verlag, 1996.

[19] J. Pearl. *Probabilistic Inference in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann, 1988.

[20] G. Piatetsky-Shapiro and W. Frawley, editors. *Knowledge Discovery in Databases*. MIT Press, Cambridge, MA, 1991.

[21] J. Rissanen. Modeling by shortest data description. *Automatica*, pages 465–471, 1978.

[22] D.W. Scott. *Multivariate Density Estimation*. New York: Wiley, 1992

[23] A. Silberschatz and A. Tuzhilin. On subjective measures of interestingness in knowledge discovery. In U. Fayyad and R. Uthurusamy, editors, *Proceedings of KDD-95: First International Conference on Knowledge Discovery and Data Mining*, pages 275–281, Menlo Park, CA, 1995. AAAI Press.

[24] C. Silverstein, S. Brin, and R. Motwani. "Beyond Market Baskets: Generalizing Association Rules to Dependence Rules". *Data Mining and Knowledge Discovery*, 2(1), 1998.

[25] J. Tukey. *Exploratory Data Analysis*. Addison Wesley, 1975.

[26] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[27] C.S. Wallace and J.D. Patrick. Coding decision trees. TR 151, Monash University, Australia, 1991.

[28] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2), 1997.

Preliminary Announcement

International Joint Conference on Work Activities Coordination and Collaboration (WACC '99)

February 22 - 25, 1999, San Francisco, USA

WACC '99 brings together researchers and practitioners from a variety of disciplines who are addressing or facing issues in work activities coordination and collaboration. Various aspects of this topic have been addressed previously under the separate banners of process-orientation, process reengineering and automation, workflow, software process, groupware, and computer-supported cooperative work. WACC '99 provides a multi-perspective forum for the presentation, discussion, demonstration, and integration of key ideas in this important area.

Topics of interest include:

- Coordination models for synchronous and asynchronous collaboration management
- Collaboration and process awareness
- Software architectures
- Case studies and experience
- Human interaction
- Collaborative transactions
- Consistency and inconsistency
- Transactional workflow processes
- Dynamic and flexible process specification and automation
- Exceptions and deviations
- Metrics and measurement
- Process evaluation and reengineering
- Organizational structures
- Dynamic workflow process
- Process evolution
- Web-enabled/web-based coordination and collaboration
- Service-orientation and management
- Virtual organizations
- Mobile-aware processes and human interactions
- System/tool scalability and reliability
- Partial/incomplete/high-level process specification

Important Dates and Information:

Abstracts:	July 17, 1998
Full papers:	July 24, 1998
Notification:	October 01, 1998
Camera ready:	November 17, 1998

Further information and instructions for paper submissions are available at:
http://www.cs.colorado.edu/wacc99/

Sponsored by:
ACM Special Interest Groups in
Computer-Human Interaction (SIGCHI)*
Groupware (SIGGROUP)*
Management of Data (SIGMOD)*
Software Engineering (SIGSOFT)*
*Pending final approval

In-association with IEEE TCDE

Organizing Committee:

General Chair:
Richard N. Taylor, U. of California, Irvine

Program Co-Chairs:
Dimitrios Georgakopoulos, MCC/GTE Labs
Wolfgang Prinz, GMD - FIT
Alexander L. Wolf, U. of Colorado

Industrial Liason:
Ming-Chien Shan, Hewlett Packard Laboratories

Steering Committee:
Alfonso Fuggetta, Politecnico di Milano
Raul Medina-Mora, Action Technologies
Marek Rusinkeiwicz, MCC
Amit Sheth (Chair), U. of Georgia

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398