Association Rule Mining: A Survey

Qiankun Zhao
Nanyang Technological University, Singapore
and
Sourav S. Bhowmick
Nanyang Technological University, Singapore

1. DATA MINING OVERVIEW

Data mining [Chen et al. 1996] is the process of extracting interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from large information repositories such as: relational database, data warehouses, XML repository, etc. Also data mining is known as one of the core processes of Knowledge Discovery in Database (KDD).

Many people take data mining as a synonym for another popular term, Knowledge Discovery in Database (KDD). Alternatively other people treat Data Mining as the core process of KDD. The KDD processes are shown in Figure 1 [Han and Kamber 2000]. Usually there are three processes. One is called preprocessing, which is executed before data mining techniques are applied to the right data. The preprocessing includes data cleaning, integration, selection and transformation. The main process of KDD is the data mining process, in this process different algorithms are applied to produce hidden knowledge. After that comes another process called postprocessing, which evaluates the mining result according to users' requirements and domain knowledge. Regarding the evaluation results, the knowledge can be presented if the result is satisfactory, otherwise we have to run some or all of those processes again until we get the satisfactory result. The actually processes work as follows.

First we need to clean and integrate the databases. Since the data source may come from different databases, which may have some inconsistences and duplications, we must clean the data source by removing those noises or make some compromises. Suppose we have two different databases, different words are used to refer the same thing in their schema. When we try to integrate the two sources we can only choose one of them, if we know that they denote the same thing. And also real world data tend to be incomplete and noisy due to the manual input mistakes. The integrated data sources can be stored in a database, data warehouse or other repositories.

As not all the data in the database are related to our mining task, the second process is to select task related data from the integrated resources and transform them into a format that is ready to be mined. Suppose we want to find which items are often purchased together in a supermarket, while the database that records the purchase history may contains *customer ID*, *items bought*, *transaction time*, *prices*, *number of each items* and so on, but for this specific task we only need *items bought*. After selection of relevant data, the database that we are going to apply our data mining techniques to will be much smaller, consequently the whole process will be

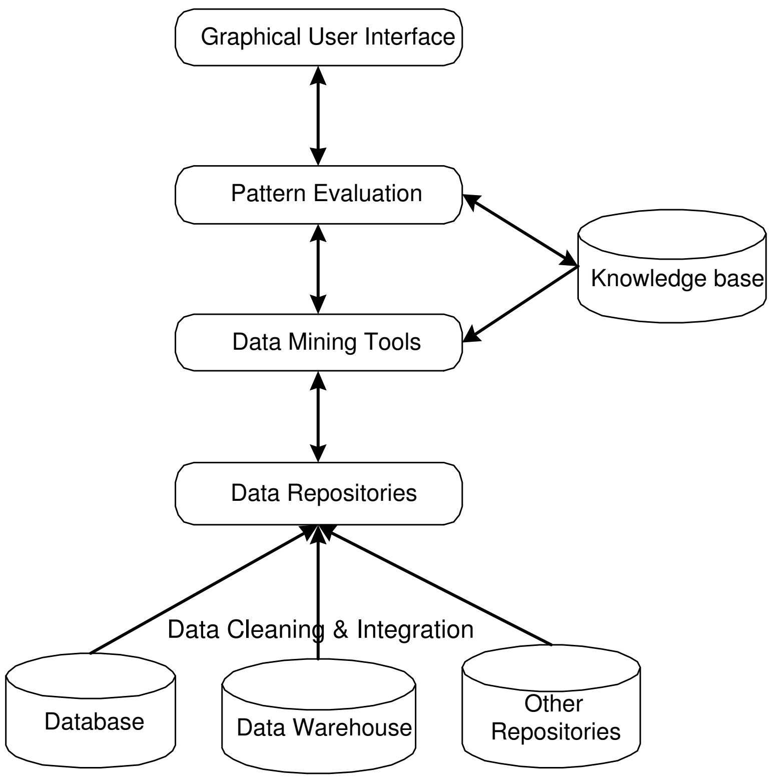

Fig. 1. Knowledge Discovery in Database processes

more efficient.

Various data mining techniques are applied to the data source, different knowledge comes out as the mining result. Those knowledge are evaluated by certain rules, such as the domain knowledge or concepts. After the evaluation, as shown in Figure 1, if the result does not satisfy the requirements or contradicts with the domain knowledge, we have to redo some processes until getting the right results. Depending on the evaluation result we may have to redo the mining or the user may modify his requirements. After we get the knowledge, the final step is to visualize the results. They can be displayed as raw data, tables, decision trees, rules, charts, data cubs or 3D graphics. This process is try to make the data mining results easier to be used and more understandable.

1.1 Types of Mining

Generally speaking, there are two classes of data mining descriptive and prescriptive. Descriptive mining is to summarize or characterize general properties of data in data repository, while prescriptive mining is to perform inference on current data, to make predictions based on the historical data. There are various types of data mining techniques such as association rules, classifications and clustering. Based on those techniques web mining and sequential pattern mining are also well researched. We will review those different types of mining techniques with examples.

Association Rule Association rule mining, one of the most important and well researched techniques of data mining, was first introduced in [Agrawal et al. 1993]. It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories.

EXAMPLE 1.1. *In an online book store there are always some tips after you purchase some books, for instance, once you bought the book Data Mining Concepts and Techniques, a list of related books such as: Database System 40%, Data Warehouse 25%, will be presented to you as recommendation for further purchasing.*

In the above example, the association rules are: when the book *Data Mining Concepts and Techniques* is brought, 40% of the time the book *Database System* is brought together, and 25% of the time the book *Data Warehouse* is brought together. Those rules discovered from the transaction database of the book store can be used to rearrange the way of how to place those related books, which can further make those rules more strong. Those rules can also be used to help the store to make his market strategies such as: by promotion of the book *Data Mining Concepts and Techniques*, it can blows up the sales of the other two books mentioned in the example. Association rules are widely used in various areas such as telecommunication networks, market and risk management, inventory control etc. Various association mining techniques and algorithms will be briefly introduced and compared later in this Chapter.

Classification Classification [Han and Kamber 2000] is to build (automatically) a model that can classify a class of objects so as to predict the classification or missing attribute value of future objects (whose class may not be known). It is a two-step process. In the first process, based on the collection of *training data set*, a model is constructed to describe the characteristics of a set of data classes or concepts. Since data classes or concepts are predefined, this step is also known as *supervised learning*(i.e., which class the training sample belongs to is provided). In the second step, the model is used to predict the classes of future objects or data.

There are handful techniques for classification [Han and Kamber 2000]. Classification by decision tree was well researched and plenty of algorithms have been designed, Murthy did a comprehensive survey on decision tree induction [Murthy 1998]. Bayesian classification is another technique that can be found in Duda and Hart [Duda and Hart 1973]. Nearest neighbor methods are also discussed in many statistical texts on classification, such as Duda and Hart [Duda and Hart 1973] and James [James 1985]. Many other machine learning and neural network techniques are used to help constructing the classification models.

A typical example of decision tree is shown in Figure 2. A decision tree for the class of *buy_laptop*, indicate whether or not a customer is likely to purchase a laptop. Each internal node represents a decision based on the value of corresponding attribute, also each leaf node represents a class(the value of *buy_laptop*=Yes or No). After this model of *buy_laptop* has been built, we can predict the likelihood of buying laptop based on a new customer's attributes such as *age*, *degree* and

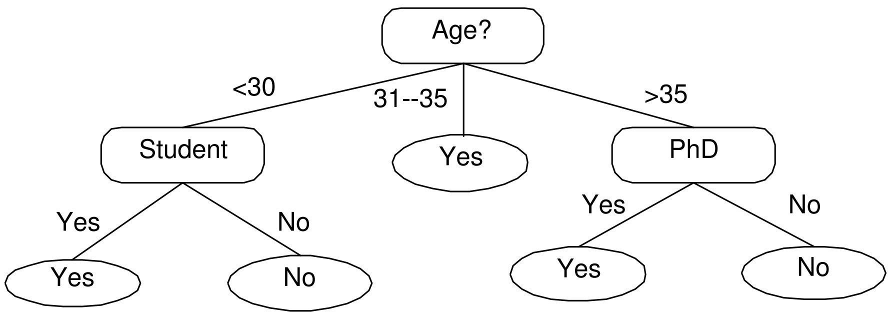

Fig. 2. An example of decision tree

profession. Those information can be used to target customers of certain products or services, especially widely used in insurance and banking.

Clustering As we mentioned before, classification can be taken as supervised learning process, clustering is another mining technique similar to classification. However clustering is a unsupervised learning process. Clustering is the process of grouping a set of physical or abstract objects into classes of similar objects [Han and Kamber 2000], so that objects within the same cluster must be similar to some extend, also they should be dissimilar to those objects in other clusters. In classification which record belongs which class is predefined, while in clustering there is no predefined classes. In clustering, objects are grouped together based on their similarities. Similarity between objects are defined by similarity functions, usually similarities are quantitatively specified as distance or other measures by corresponding domain experts.

Most clustering applications are used in market segmentation. By clustering their customers into different groups, business organizations can provided different personalized services to different group of markets. For example, based on the expense, deposit and draw patterns of the customers, a bank can clustering the market into different groups of people. For different groups of market, the bank can provide different kinds of loans for houses or cars with different budget plans. In this case the bank can provide a better service, and also make sure that all the loans can be reclaimed. A comprehensive survey of current clustering techniques and algorithms is available in [Berkhin 2002].

1.2 Types of Data

Based on the types of data that our mining techniques are applied to, data mining can be classified into different categories.

1.2.1 *Relational database.*

Till now most data are stored in relational database, and relational database is one of the biggest resources of our mining objects. As we know relational database is highly structured data repository, data are described by a set of attributes and stored in tables. With the well developed database query languages, data mining on relational database is not difficult. Data mining on relational database mainly focuses on discovering patterns and trends. For example, in the database of a bank, by using some aggregate operators we can find the high spend customers, with data mining techniques we can further classify those customers based on their profile. Also by analyzing the expenses patterns of customers, we can provide certain information to different business organizations. Suppose we found a customer spends 500 dollars every month on fashion, if permitted we can provide this information to the fashion shops.

1.2.2 *Transactional database.* Transactional database refers to the collection of transaction records, in most cases they are sales records. With the popularity of computer and e-commerce, massive transactional databases are available now. Data mining on transactional database focuses on the mining of association rules, finding the correlation between items in the transaction records. An example of association rule on transactional database is show in **Example2.1**. In the later part of association rule mining, most examples are from transactional database, we will elaborate them in detail later.

1.2.3 *Spatial database.* Spatial databases usually contain not only traditional data but also the location or geographic information about the corresponding data. Spatial association rules describe the relationship between one set of features and another set of features in a spatial database, for example {Most business centers in Singapore are *around* City Hall}, the spatial operations that used to describe the correlation can be *within*, *near*, *next to*, etc.. Definitions of spatial association rules and their parameters [Koperski and Han 1995] are identical to those for regular association rules [Agrawal et al. 1993]. The form of spatial association rules is also X⇒Y, where X, Y are sets of predicates and of which some are spatial predicates, and at least one must be a spatial predicate [Koperski and Han 1995; 1996]. Algorithms for mining spatial association rules are similar to association rule mining except consideration of special data, the predicates generation and rule generation processes are based on Apriori, detail of the algorithm for mining spatial association rules were explained in [Koperski and Han 1996]. A spatial association rule mining application GeoMiner [Han et al. 1997] has been developed to extract useful information from a geographical database.

1.2.4 *Temporal and time-series database.* Differ from traditional transaction data, for each temporal data item the corresponding time related attribute is associated. Temporal association rules can be more useful and informative than basic association rules. For example rather than the basic association rule {diapers}⇒{beer}, mining from the temporal data we can get a more insight rule that the support of {diapers}⇒{beer} jumps to 50% during 7pm to 10pm everyday, obviously retailers can make more efficient promotion strategy by using temporal association rule.

In [Han et al. 1999] and [Mannila et al. 1995] algorithms for mining periodical patterns and episode sequential patterns were introduced respectively. Most of those researches now form a new area of data mining called sequential pattern mining, mining frequent sequential patterns in time series database, which was initiated by Agrawal in [Agrawal and Srikant 1995]. Detail of temporal data mining will be elaborated later in this Chapter.

1.2.5 *World-Wide Web.* As information on the web increases in a phenomena speed and web becomes ubiquitous, most researchers turn to the field of mining web data(Web Mining). Web mining is usually divided into three main categories, web usage mining, web structure mining and web content mining.

Web usage mining concentrates on mining the access patterns of users, so that the structure of the web site can be modified based on the navigation patterns. Different application of mining web logs have been developed to find navigation patterns. Besides improving the web site structure, web usage mining is also valuable for cross marketing strategies, web advertisements and promotion campaigns. Web structure mining focuses in analysis of structures and links in web documents. The basic idea is that those pages that are linked together have some kinds of relationship. With those links, an typical structure mining is to classify those web documents into authoritative pages and hub pages. Authoritative pages are pages that present the original source of information, while hub pages are pages that link to those authoritative pages. Web content includes text, graphic, media etc.. Consequently web content mining includes text mining, multimedia mining and graphic mining.

---

2. ASSOCIATION RULE MINING

In this section we will introduce association rule mining problem in detail. Different issues in Association Rule Mining(ARM) will be elaborated together with classic algorithms and examples.

2.1 Association Rule Problem

With the general example and introduction in last section, the formal statement of association rule mining problem was firstly stated in [Agrawal et al. 1993] by Agrawal. Let I=I_1, I_2, $\cdots$, I_m be a set of m distinct attributes, T be transaction that contains a set of items such that T $\subseteq$ I, D be a database with different transaction records Ts. An ***association rule*** is an implication in the form of X$\Rightarrow$Y, where X, Y$\subset$ I are sets of items called itemsets, and X $\cap$ Y = $\emptyset$. X is called antecedent while Y is called consequent, the rule means X implies Y.

There are two important basic measures for association rules, *support*(s) and *confidence*(c). Since the database is large and users concern about only those frequently purchased items, usually thresholds of support and confidence are predefined by users to drop those rules that are not so interesting or useful. The two thresholds are called *minimal support* and minimal confidence respectively, additional constraints of interesting rules also can be specified by the users. The two basic parameters of Association Rule Mining (ARM) are: support and confidence.

Support(s) of an association rule is defined as the percentage/fraction of records that contain X $\cup$ Y to the total number of records in the database. The count for each item is increased by one every time the item is encountered in different transaction T in database D during the scanning process. It means the support count does not take the quantity of the item into account. For example in a transaction a customer buys three bottles of beers but we only increase the support count number of {beer} by one, in another word if a transaction contains a item then the support count of this item is increased by one. Support(s) is calculated by the following formula:

$$Support(XY) = \frac{Support\ count\ of\ XY}{Total\ number\ of\ transaction\ in\ D}$$

From the definition we can see, support of an item is a statistical significance of an association rule. Suppose the support of an item is 0.1%, it means only 0.1 percent of the transaction contain purchasing of this item. The retailer will not pay much attention to such kind of items that are not bought so frequently, obviously a high support is desired for more interesting association rules. Before the mining process, users can specify the minimum support as a threshold, which means they are only interested in certain association rules that are generated from those itemsets whose supports exceed that threshold. However, sometimes even the itemsets are not so frequent as defined by the threshold, the association rules generated from them are still important. For example in the supermarket some items are very expensive, consequently they are not purchased so often as the threshold required, but association rules between those expensive items are as important as other frequently bought items to the retailer.

Confidence of an association rule is defined as the percentage/fraction of the number of transactions that contain X∪Y to the total number of records that contain X, where if the percentage exceeds the threshold of confidence an interesting association rule X⇒Y can be generated.

$$Confidence(X|Y) = \frac{Support(XY)}{Support(X)}$$

Confidence is a measure of strength of the association rules, suppose the confidence of the association rule X⇒Y is 80%, it means that 80% of the transactions that contain X also contain Y together, similarly to ensure the interestingness of the rules specified minimum confidence is also pre-defined by users.

Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database [Agrawal and Srikant 1994]. The problem is usually decomposed into two subproblems. One is to find those itemsets whose occurrences exceed a predefined threshold in the database, those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with the constraints of minimal confidence. Suppose one of the large itemsets is L_k, $Lk= \{I_1, I_2, \cdots, I_{k-1}, I_k\}$, association rules with this itemsets are generated in the following way: the first rule is $\{I_1, I_2, \cdots, I_{k-1}\} \Rightarrow \{I_k\}$, by checking the confidence this rule can be determined as interesting or not. Then other rule are generated by deleting the last items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine the interestingness of them. Those processes iterated until the antecedent becomes empty. Since the second subproblem is quite straight forward, most of the researches focus on the first subproblem.

2.2 Association Rule Mining Approaches

Association rule mining is a well explored research area, we will only introduce some basic and classic approaches for association rule mining. As stated before, the second subproblem of ARM is straightforward, most of those approaches focus on the

TID	List of items
T100	I_1,I_2, I_5
T200	I_2, I_4
T300	I_2, I_3
T400	I_1, I_2, I_4
T500	I_1, I_3
T600	I_2, I_3
T700	I_1, I_3
T800	I_1, I_2, I_3, I_5
T900	I_1, I_2, I_3
T000	I_1, I_2, I_5, I_6

(a) $OriginalDatabase$

Items	Count number
I_1	7
I_2	8
I_3	6
I_4	2
I_5	3
I_6	1

(b) C_1

Large 1 Items
I_1
I_2
I_3
I_5

(c) L_1

Items	Count number
I_1, I_2	5
I_1, I_5	3
I_2, I_5	3
I_2, I_4	2
I_2, I_3	4
I_1, I_4	1
...	...

(d) C_2

Large 2 Items
I_1, I_2
I_1, I_5
I_2, I_5
I_2, I_3
I_1, I_3

(e) L_2

Items	Count number
I_1, I_2, I_5	3
I_1, I_2, I_4	1
I_1, I_2, I_3	2
I_1, I_2, I_6	1
I_2, I_3, I_5	1
I_1, I_3, I_5	1
...	...

(f) C_3

Table I. AIS Mining Process

first subproblem. The first subproblem can be further divided into two subproblems: *candidate large itemsets* generation process and *frequent itemsets* generation process. We call those itemsets whose support exceed the support threshold as *large* or *frequent itemsets*, those itemsets that are expected or have the hope to be large or frequent are called *candidate itemsets*. Most of the algorithms of mining association rules we surveyed are quite similar, the difference is the extend to which certain improvements have been made, so only some of the milestones of association rule mining algorithms will be introduced.

First we will introduce some naive and basic algorithms for association rule mining, Apriori series approaches. Then another milestone, tree structured approaches will be explained. Finally this section will end with some special issues of association rule mining, including multiple level ARM, multiple dimension ARM, constraint based ARM and incremental ARM.

In order to make it easier for us to compare those algorithms we use the same transaction database, a transaction database from a supermarket, to explain how those algorithms work. This database records the purchasing attributes of its customers. Suppose during the preprocess all those attributes that are not relevant or useful to our mining task are pruned, only those useful attributes are left ready for mining as shown in Table I (a).

2.2.1 *Apriori Series Approaches.*

2.2.2 *AIS Algorithm.* The AIS(Agrawal, Imielinski, Swami) algorithm was the first algorithm proposed for mining association rule in [Agrawal et al. 1993]. It focus on improving the quality of databases together with necessary functionality to process decision support queries. In this algorithm only one item consequent association rules are generated, which means that the consequent of those rules only contain one item, for example we only generate rules like X$\bigcap$Y$\Rightarrow$Z but not those rules as X$\Rightarrow$Y$\bigcap$Z.

The databases were scanned many times to get the frequent itemsets in AIS. During the first pass over the database, the support count of each individual item was accumulated as shown in Table I (b), suppose the *minimal support* threshold is 30%, large one items were generated in Table I(c). According to *minimal support* those items whose support counts are less than 3 (I_4 and I_4) are eliminated from the list of frequent items. With those frequent *1* items, candidate *2*-itemsets are generated by extending those frequent items with other items in the same transaction. To avoid generating the same itemsets repeatedly the items were ordered, candidate itemsets are generated by joining the large items in previous pass and another item in the transaction, which appears later than the last item in the frequent itemsets. For example, based on transaction T100 I_1, I_2, I_5, according to this specific order we generate candidate *2*-itemsets by extending I_1 with only I_2, I_5, similarly I_2 is extended with I_5. The result is shown in Table I(d). During the second pass over the database, the support count of those candidate 2-itemsets are accumulated and checked against the support threshold. Similarly those candidate (*k+1*)-itemsets are generated by extending frequent k-itemsets with items in the same transaction. All those candidate itemsets generation and frequent itemsets generation process iterate until any one of them becomes empty. The result frequent itemsets includes only one large 3-itemsets $\{I_1,\ I_2,\ I_5\ \}$.

To make this algorithm more efficient, an estimation method was introduced to prune those itemsets candidates that have no hope to be large, consequently the unnecessary effort of counting those itemsets can be avoided. Since all the candidate itemsets and frequent itemsets are assumed to be stored in the main memory, memory management is also proposed for AIS when memory is not enough. One approach is to delete candidate itemsets that have never been extended. Another approach is to delete candidate itemsets that have maximal number of items and their siblings, and store this the parent itemsets in the disk as a seed for the next pass. The detail examples are available in [Agrawal et al. 1993].

The main drawback of the AIS algorithm is too many candidate itemsets that finally turned out to be small are generated, which requires more space and wastes much effort that turned out to be useless. At the same time this algorithm requires too many passes over the whole database.

2.2.3 *Apriori Algorithm.* Apriori is a great improvement in the history of association rule mining, Apriori algorithm was first proposed by Agrawal in [Agrawal and Srikant 1994]. The AIS is just a straightforward approach that requires many passes over the database, generating many candidate itemsets and storing counters of each candidate while most of them turn out to be not frequent. Apriori is more efficient during the candidate generation process for two reasons, Apriori employs a different candidates generation method and a new pruning technique.

Items	Count number
I_1	7
I_2	8
I_3	6
I_4	2
I_5	3
I_6	1

(a) C_1

Large 1 Items
I_1
I_2
I_3
I_5

(b) L_1

Items	Count number
I_1, I_2	5
I_1, I_3	4
I_1, I_5	3
I_2, I_3	4
I_2, I_5	3
I_3, I_5	1

(c) C_2

Large 2 Items
I_1, I_2
I_1, I_5
I_2, I_5
I_2, I_3
I_1, I_3

(d) L_2

Items	Count number
I_1, I_2, I_5	3
I_1, I_2, I_3	2

(e) C_3

Table II. Apriori Mining Process

There are two processes to find out all the large itemsets from the database in Apriori algorithm. First the candidate itemsets are generated, then the database is scanned to check the actual support count of the corresponding itemsets. During the first scanning of the database the support count of each item is calculated and the large *1*-itemsets are generated by pruning those itemsets whose supports are below the pre-defined threshold as shown in Table II(a) and (b). In each pass only those candidate itemsets that include the same specified number of items are generated and checked. The candidate k-itemsets are generated after the *(k-1)th* passes over the database by joining the frequent *k-1*-itemsets. All the candidate k-itemsets are pruned by check their sub *(k-1)*-itemsets, if any of its sub *(k-1)*-itemsets is not in the list of frequent *(k-1)*-itemsets, this k-itemsets candidate is pruned out because it has no hope to be frequent according the Apriori property. The Apriori property says that every sub *(k-1)*-itemsets of the frequent k-itemsets must be frequent. Let us take the generation of candidate 3-itemsets as an example. First all the candidate itemsets are generated by joining frequent 2-itemsets, which include (I_1, I_2, I_5), (I_1, I_2, I_3), (I_2, I_3, I_5), (I_1, I_3, I_5). Those itemsets are then checked for their sub itemsets, since (I_3, I_5) is not frequent 2-itemsets, the last two 3-itemsets are eliminated from the list of candidate 3-itemsets as shown in Table II(e). All those processes are executed iteratively to find all frequent itemsets until the candidates itemsets or the frequent itemsets become empty. The result is the same as the AIS algorithm. The algorithm is shown in Figure 3.

In the process of finding frequent itemsets, Apriori avoids the effort wastage of counting the candidate itemsets that are known to be infrequent. The candidates are generated by joining among the frequent itemsets level-wisely, also candidate are pruned according the Apriori property. As a result the number of remaining candidate itemsets ready for further support checking becomes much smaller, which dramatically reduces the computation, I/O cost and memory requirement. Table II shows the process of Apriori algorithm, by compare Table I and Table II we can see the numbers of candidates changed dramatically. Detail of the Apriori-gen and

```
Input:
    database D
    Mini Support $\epsilon$
    Mini Confidence $\xi$
Output:
    $R_t$ All association rules
Method:
01    $L_1$ = large 1-itemsets;
02    for(k=2; $L_{k-1} \neq \emptyset$; k++) do begin
03    $C_k$ =apriori-gen($L_{k-1}$); //generate new candidates from $L_{k-1}$
04    for all transactions T $\in$ D do begin
05    $C_t$=subset($C_k$,T); //candidates contained in T.
06    for all candidates C $\in C_t$ do
07    Count(C)=Count(C)+1; // increase support count of C by 1
08    end
09    $L_k$={C $\in C_t$ | Count(C) $\geq \epsilon \times$ | D | }
10    end
11    $L_f = \bigcup_k L_k$;
12    $R_t$=GenerateRules($L_f$, $\xi$ )
```

Fig. 3. Apriori Algorithm.

GenerateRules functions were elaborated in [Agrawal and Srikant 1994].

Apriori algorithm still inherits the drawback of scanning the whole data bases many times. Based on Apriori algorithm, many new algorithms were designed with some modifications or improvements. Generally there were two approaches: one is to reduce the number of passes over the whole database or replacing the whole database with only part of it based on the current frequent itemsets, another approach is to explore different kinds of pruning techniques to make the number of candidate itemsets much smaller. Apriori-TID and Apriori-Hybrid [Agrawal and Srikant 1994] , DHP [Park et al. 1995], SON [Savesere et al. 1995] are modifications of the Apriori algorithm.

Most of the algorithms introduced above are based on the Apriori algorithm and try to improve the efficiency by making some modifications, such as reducing the number of passes over the database; reducing the size of the database to be scanned in every pass; pruning the candidates by different techniques and using sampling technique. However there are two bottlenecks of the Apriori algorithm. One is the complex candidate generation process that uses most of the time, space and memory. Another bottleneck is the multiple scan of the database.

2.2.4 *FP-Tree(Frequent Pattern Tree) Algorithm.* To break the two bottlenecks of Apriori series algorithms, some works of association rule mining using tree structure have been designed. FP-Tree [Han et al. 2000], frequent pattern mining, is another milestone in the development of association rule mining, which breaks the two bottlenecks of the Apriori. The frequent itemsets are generated with only two passes over the database and without any candidate generation process. FP-Tree was introduced by Han et al in [Han et al. 2000]. By avoiding the candidate generation process and less passes over the database, FP-Tree is an order of magnitude faster than the Apriori algorithm. The frequent patterns generation process includes two sub processes: constructing the FT-Tree, and generating frequent pat-

TID	List of items
T100	I_1,I_2, I_5
T200	I_2, I_4
T300	I_2, I_3
T400	I_1, I_2, I_4
T500	I_1, I_3
T600	I_2, I_3
T700	I_1, I_3
T800	I_1, I_2, I_3, I_5
T900	I_1, I_2, I_3
T000	I_1, I_2, I_5, I_6

(a) $OriginalDatabase$

Large 1 Items	Support
I_1	7
I_2	8
I_3	6
I_5	3

(b) L_1

TID	Ordered Large Items
T100	I_2, I_1, I_5
T200	I_2
T300	I_2, I_3
T400	I_2, I_1
T500	I_1, I_3
T600	I_2, I_3
...	...

(c) Transformed Data

Table III. FPTree data transformation

terns from the FP-Tree.

The process of constructing the FP-Tree is as follows.

(1) The database is scanned for the first time, during this scanning the support count of each items are collected. As a result the frequent *1*-itemsets are generated as shown in Table III(b), this process is the same as in Apriori algorithm. Those frequent itemsets are sorted in a descending order of their supports. Also the head table of ordered frequent *1*-itemsets is created as shown in Figure 4.

(2) Create the root node of the FP-Tree T with a label of *Root*. The database is scanned again to construct the FP-Tree with the head table, for each transaction the order of frequent items is resorted according to the head table. For example, the first transaction (I_1, I_2, I_5) is transformed to (I_2, I_1, I_5), since I_2 occurs more frequently than I_1 in the database. Let the items in the transaction be [p | P], where p is the first frequent item and P is the remaining items list, and call the function *Insert*{[p | P]; T}.

(3) The function *Insert*{[p | P]; T} works as follows. If T has a child N such that N.item-name=p.item-name then the count of N is increased by 1, else a new node N is created and N.item-name=p.item-name with a support count of 1. Its parent link be linked to T and its node link is linked to the node with the same item-name via a sub-link. This function *Insert*{P;T} is called recursively until P becomes empty.

Let's take the insertion of first transaction to the FPTree as an example to illustrate the *insert* function and construction of FPTree we mentioned above. After reorder this transaction is (I_2, I_1, I_5), so p is I_2 in this case, while P is (I_1, I_5). Then we call the function of *insert*, first we search and determine the node I_2 exists in the tree or not, it turns out I_2 is a new node. According to the rules, a new node named I_2 is created with a support count of 1. Since here T is *Root*, node I_2 is linked to Root and call the insert function again. At this time p is I_1, P is I_5, T is I_2. The result of the FPTree of the database is shown in Figure 4.

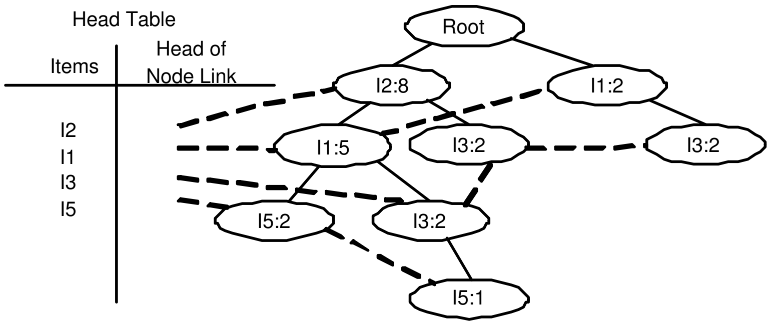

Fig. 4. Result of FPTree

The frequent patterns are generated from the FP-Tree by the procedure named FP-growth [Han and Pei 2000]. Based on the head table and the FP-Tree, frequent patterns can be generated easily. It works as Figure 5. For example, here is the

```
Input:
        the FP-Tree $Tree$
Output:
        $R_t$ Complete set of frequent patterns
Method: Call FP-growth($Tree$ , null).
Procedure FP-growth ($Tree$ , $\alpha$)
{
01          if $Tree$ contains a single path P;
02          then for each combination (denoted as $\beta$) of the nodes in the path P do
03          generate pattern $\beta \bigcup \alpha$ with support= minimum support of nodes in $\beta$;
04          else for each $a_i$ in the header of $Tree$ do {
05          generate pattern $\beta = a_i \bigcup \alpha$ with support= $a_i$. support;
06          construct $\beta$'s conditional pattern base and then $\beta$'s conditional FP-Tree $Tree_\beta$;
07          if $Tree_\beta \neq \emptyset$
08          then call FP-growth ($Tree$, $\beta$)          }
}
```

Fig. 5. FP-growth Algorithm.

whole process of getting all those frequent itemsets concerning I_5. Firstly, follow the head table, we find the pattern base of this node, which are all those paths end with this node. For I_5, its pattern base is: $(I_2, I_1)(2)$ and $(I_2, I_1, I_3)(1)$, the number in the bracket follow the itemsets means the support of this pattern. Then the count of all the items in the pattern base are accumulated, in this case we ge $I_2(3)$, $I_1(3)$ and $I_3(1)$. By checking the support count with the minimal support threshold, the conditional FPTree of I_5 is generated $(I_2, I_1)(3)$. Consequently we generated the frequent itemsets/pattern (I_2, I_1, I_5). The mining result is the same with Apriori series algorithms.

The efficiency of FP-Tree algorithm account for three reasons. First the FP-Tree is a compressed representation of the original database because only those frequent items are used to construct the tree, other irrelevant information are pruned. Also by ordering the items according to their supports the overlapping parts appear only once with different support count. Secondly this algorithm only scans the database twice. The frequent patterns are generated by the FP-growth procedure, constructing the conditional FP-Tree which contain patterns with specified suffix patterns, frequent patterns can be easily generated as shown in above the example. Also the computation cost decreased dramatically. Thirdly, FP-Tree uses a divide and conquer method that considerably reduced the size of the subsequent conditional FP-Tree, longer frequent patterns are generated by adding a suffix to the shorter frequent patterns. In [Han et al. 2000] [Han and Pei 2000] there are examples to illustrate all the detail of this mining process.

Every algorithm has his limitations, for FP-Tree it is difficult to be used in an interactive mining system. During the interactive mining process, users may change the threshold of support according to the rules. However for FP-Tree the changing of support may lead to repetition of the whole mining process. Another limitation is that FP-Tree is that it is not suitable for incremental mining. Since as time goes on databases keep changing, new datasets may be inserted into the database, those insertions may also lead to a repetition of the whole process if we employ FP-Tree algorithm.

2.2.5 *Rapid Association Rule Mining (RARM).* RARM [Das et al. 2001] is another association rule mining method that uses the tree structure to represent the original database and avoids candidate generation process. RARM is claimed to be much faster than FP-Tree algorithm with the experiments result shown in the original paper. By using the SOTrieIT structure RARM can generate large 1-itemsets and 2-itemsets quickly without scanning the database for the second time and candidates generation. Similar to the FP-Tree, every node of the SOTrieIT contains one item and the corresponding support count. The large itemsets generation process is as follows.

Preprocessing, the database is scanned to construct the TrieIT, the process is similar to the process of generation the FP-Tree. For each transaction all the possible itemsets combinations are extracted and for those items that are already in the TrieIT increase their support count by 1, for those that still do not exist in the TrieIT the itemsets are inserted to the TrieIT with the corresponding support count be 1. The difference between FPTree and TrieIT is that TrieIT only increases the support count of the leaf node items while FP-Tree increases all the support counts along the path of the itemsets. Since the TrieIT stores the support counts individually, it requires bigger memory space which may not be satisfied, SOTrieIT (Support Ordered Trie Itemset) is introduced. To construct the SOTrieIT only 1-itemsets and 2-itemsets are extracted from each transaction, the building process is the same as in the construction of TrieIT, after all the itemsets of the same transaction were inserted the tree is ordered in a descending order of support count, the SOTrieIT has only two levels one is for 1-itemsets, another is for 2-itemsets, the result of SOTrieIT is shown in Figure 6. By compare this Figure 6 and Figure 4 we can see that the size of SOTrieIT is much smaller than TrieIT and is possible

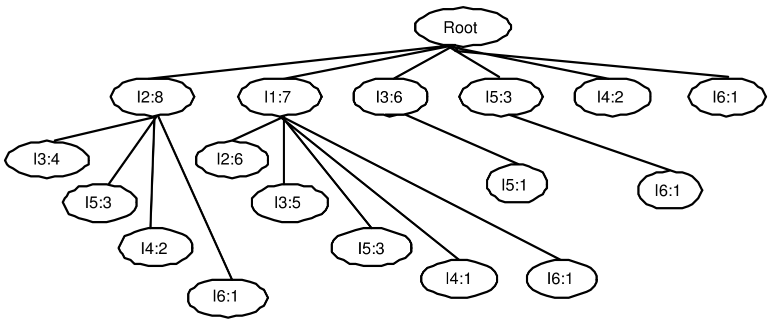

Fig. 6. Result of SOTrieIT

to be stored in the main memory.

The second process is mining of large itemsets, the SOTrieIT tree is scanned in a depth-first search approach, the scanning starts from the leftmost first level node, if the support count of one item in the second level does not satisfy the minimal support threshold the traversal stops and moves to another first level node, if the support count of one item in the first level node does not satisfy the minimal support threshold the whole traversal stops here since all the items in SOTrieIT are in a descending order. After the traversal large 1-itemsets and 2-itemsets are generated, the Apriori algorithm is used to further discovery other large itemsets.

Since generating the large 2-itemsets is the most costly process during the mining process, experiments in [Das et al. 2001] showed that the efficiency of generating large 1-itemsets and 2-itemsets in the SOTrieIT algorithm improves the performance dramatically, SOTrieIT is much faster than FP-Tree, but SOTrieIT also faces the same problem as FP-Tree.

2.2.6 *Special Issues of Association Rule Mining.* Based on those basic algorithms of association rule mining, many special issues of association rule mining have been explored. Multiple concept level association rule mining, multiple dimension association rule mining, constraints based association rule mining and maintenance of association rules will be explained respectively in this section.

2.2.7 *Multiple Concept Level ARM.* In real life, for many applications, it is difficult to find strong association rules between data items at low or primitive level of abstraction due to the sparsity of data in multidimensional space [Han and Kamber 2000]. While strong association rules generated at a higher concept level may be common sense to some users but it also can be novel to other users. Multiple level association rule mining is trying to mine strong association rules among intra and inter different levels of abstraction. For example, besides the association rules between *milk* and *ham*, it can generalize those rules to relation between *drink* and *meat*, at the same time it can also specify relation between certain brand of *milk* and *ham*. Researches have been done in mining association rule at multiple concept levels [Han 1995], [Han and Fu 1995], [Psaila and Lanzi 2000].

Usually data items in database can be classified into different concept levels according to the knowledge of corresponding concept hierarchy, which may be provided in the database or by the domain experts. Three algorithms of mining knowledge at multiple concept levels were firstly introduced in 1995 by Han [Han 1995]. Since association rules at a lower level may be expected to have lower support, it is important to mining association rules at a specific abstraction level with appropriate support threshold. In order to get interesting and useful association rules, in his approach support threshold at different level can be different. An interactive application, in which users can modify the support and confidence thresholds according to the rule results, was also proposed in [Han 1995]. All the methods in [Han 1995] mainly concern about intra concept level association rules, while mining cross level rules, which means that the antecedents and the consequences of the rules belong to different concept levels, was introduced in [Han and Fu 1995].

For multiple level association rule mining, usually there are more than one possible way to classify all the items into a concept hierarchy, different users may prefer different hierarchies. Multiple concept level can provide information according to different requirements in different fields. However, with different concept levels, mining in multiple concept levels may produce a lot of rules that are redundant. Some of the rules may carry the same information or some knowledge contained within a rule maybe also contained in other rules duplicately. A further selection among those result rules is required to provide users useful, concise knowledge.

2.2.8 *Multiple Dimensional ARM.* Most algorithms and techniques discussed above only concern about association rules within single attribute and boolean data, all those rules are about the same attribute and the value can only be *yes/1* or *no/0*. By mining multiple dimensional association rules we can generate the rules such as: age(X,"20—29") $\bigwedge$ occupation(X,"student") $\Longrightarrow$ buys(X,"laptop") rather than rules in a single dimension such as: buys(diaper) $\Longrightarrow$ buys(beer). Multiple dimensional association rule mining is to discovery the correlation between different predicts/attributes. Each attribute/predict is called a dimension, such as: age, occupation and buys in this example. At the same time multiple dimensional association rule mining concerns all types of data such as boolean data, categorical data and numerical data [Srikant and Agrawal 1996]. The mining process is similar to the process of multiple level association rule mining. Firstly the frequent 1-dimensions are generated, then all frequent dimensions are generated based on the Apriori algorithm. The mining process is straightforward, but there are three basic approaches for generating multiple dimensions.

Using Static discretization Method. In this approach, quantitative attributes are partitioned to different ranges according the predefined hierarchies and attributes are replaced by those ranges prior to the mining process. Categorical attributes also can be generalized to higher concept level if necessary. After this process the task relevant data can be stored in the relational database. The relevant data can also be stored in the data cube, which is more suitable for multiple dimensional association rules since data cube itself is multidimensional by definition. The Apriori can be easily modified to get the frequent k-predicts by searching through all the relevant attributes instead of one attribute only.

Using Dynamical Discretization Method. Quantitative attributes are dynamically discretized during the mining process so as to satisfy some mining criteria such as confidence and support. How to mine quantitative association rules was first introduced by Agrawal [Srikant and Agrawal 1996]. Also the method of mining two-dimensional association rules and clustering the result rules were discussed in [Lent et al. 1997].

Using Distance Based Discretization Method. A distance based method is used to mining quantitative attributes instead of the equi-width and equi-depth partition method for each attribute. In this approach values that are close together are grouped into the same interval. There are two steps, firstly intervals or clusters are generated by the clustering techniques, then distance based association rules are generated by searching for groups or clusters that occur frequently together. Some applications of multiple dimensional association rule mining have been implemented in [Lu et al. 1998]. Also weighted association rule mining was proposed in [Wang et al. 1997], in which the frequent itemsets were generated with some parameters, the associated parameters were used as metrics during the rule generation process. Those parameters include density and frequency. In this case rules that satisfy certain density and frequency thresholds are taken as interesting rules.

2.2.9 *Constraints based ARM.* In order to improve the efficiency of existing mining algorithms, constraints were applied during the mining process to generate only those association rules that are interesting to users instead of all the association rules. By doing this lots of costs of mining those rules that are turned out to be not interesting can be saved. Usually constraints are provided by users, it can be knowledge based constraints, data constraints, dimensional constraints, interestingness constraints or rule formation constraints. A handful research literature exists in the study of constraints based association rule mining [Ng et al. 1998], [Pei and Han 2000], [Bayardo et al. 1999], [Srikant et al. 1997], [Garofalakis et al. 1999], [Klemettinen et al. 1994], [Brin et al. 1997], [Smythe and Goodman 1992].

Constraints based association rule mining is to find all rules from a given data-set meeting all the user-specified constraints. Apriori and its variants only employ two basic constraints: *minimal support* and *minimal confidence*. However there are two points, one is some of the generated rules may be usefulness or not informative to individual users; another point is that with the constraints of minimal support and confidence those algorithms may miss some interesting information that may not satisfy them.

In [Ng et al. 1998] and [Srikant et al. 1997], authors proposed some algorithms with faster association rule mining by incorporating item constraints on the process of generating frequent itemsets. Item constraints restrict the items and combination of items that are interesting according to users, association rules are generated from those frequent itemsets. Also some works have been done on measuring association rules with interestingness parameter [Klemettinen et al. 1994] [Brin et al. 1997] [Smythe and Goodman 1992], while item constraints focus on pruning frequent itemsets, the interestingness parameter focuses on pruning the association rules to get more general or informative association rules. In [Bayardo et al. 1999] minimal improvement constraint and consequent constraint are introduced, the consequent constraint is further used to pruning association rules with the minimal confidence, while minimal improvement constraint is used for pruning association rules by measuring each item's contribution to the confidence. Detail explanation and examples can be find in [Bayardo et al. 1999].

To discover only rules of specific patterns using meta-rules was introduced in [Fu and Han 1995], a technique called meta-rule mining was proposed by Han and Fu. The format of interesting rules is specified by a template, the algorithm generates only those rules that can be fitted into this template. For example a meta-rule can be X,Y $\Rightarrow$ Z, where X,Y,Z represents any items in the database, according to this meta-rule only frequent 3-itemsets can produce this kind of rules, which in turn makes the algorithm more efficient. Regular expression constraint was introduced in [Garofalakis et al. 1999], user's specific requirements are stated in regular expression, denoted by C, four algorithms of SPIRIT-Sequential Pattern mIning with Regular expressIon consTraints are discussed. They are SPIRIT Naive, Legal, Valid and Regular, the difference between those four algorithms is the extend to which the constraints are pushed into the candidate generation and pruning processes during the whole mining. From the naive approach, legal approach, valid approach to the regular approach, more and more constraints are pushed into the mining process [Garofalakis et al. 1999].

2.2.10 *Maintaining of Association Rules.* From the definition of data mining, we can see that the object of data mining is data stored in very large repositories. The giant amount of data poses a challenge of maintaining and updating the discovered rules while the data may change from time to time in different ways. Researches have been done for maintaining of discovered association rules [Cheung et al. 1996], [Cheung et al. 1997], [Lee and Cheung 1997], [Thomas et al. 1997].

In [Cheung et al. 1996], the FUP(Fast UPdate) algorithm was introduced to deal with insertion of new transaction data. The problem with incremental updating is to find the large itemsets for a database $D\bigcup db$, where D and db are sets of old and inserted transactions respectively. The main assumption is that the set of large itemsets L for D is already known. FUP is based on the Apriori algorithm. For each iteration, only db is scanned using the known set of large itemsets of size k, L_k, from D as the candidates. This is used to remove the candidates which are no longer large in the larger database, $D\bigcup db$. Simultaneously a set of new candidates is determined. Since the database may change in different ways, FUP can only handle insertion of new transaction data, FUP2 algorithm was proposed in [Cheung et al. 1997]. FUP2 can efficiently update discovered association rules with insertion of some new transactions and deletion of some obsolete transactions. Both FUP and FUP2 have disadvantages, they require space to store the large itemsets and rules of the original database. In FUP2 the deleted transaction must also be retained and FUP2 is efficient only when the database does not change much.

Another approach to maintain association rules is based on the idea of sampling [Lee and Cheung 1997]. The algorithm proposed in this paper, Difference Estimations for Large Itemsets (DELI), uses sampling to estimate the upper bound on the difference between the old and new sets of association rules. Small changes to the association rule set are ignored. Performance studies showed the effectiveness of the DELI approach in saving resources [Lee and Cheung 1997].

A third approach determines the large itemsets of the incremental database and only scans the original database if the negative border of the large itemsets expands from that of the original database [Thomas et al. 1997]. In this situation only one scan over the original database is then required to find all large itemsets.

3. CONCLUSION

In this paper, we surveyed the list of existing association rule mining techniques. This investigation is prepared to our new project titled *mining historical changes to web delta.*

REFERENCES

Agrawal, R., Imielinski, T., and Swami, A. N. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, P. Buneman and S. Jajodia, Eds. Washington, D.C., 207–216.

Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Morgan Kaufmann, 487–499.

Agrawal, R. and Srikant, R. 1995. Mining sequential patterns. In *Eleventh International Conference on Data Engineering*, P. S. Yu and A. S. P. Chen, Eds. IEEE Computer Society Press, Taipei, Taiwan, 3–14.

Bayardo, R., Agrawal, R., and Gunopulos, D. 1999. Constraint-based rule mining in large, dense databases.

Berkhin, P. 2002. Survey of clustering data mining techniques. Tech. rep., Accrue Software, San Jose, CA.

Brin, S., Motwani, R., Ullman, J. D., and Tsur, S. 1997. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, J. Peckham, Ed. ACM Press, 255–264.

Chen, M.-S., Han, J., and Yu, P. S. 1996. Data mining: an overview from a database perspective. *Ieee Trans. On Knowledge And Data Engineering 8*, 866–883.

Cheung, D. W.-L., Han, J., Ng, V., and Wong, C. Y. 1996. Maintenance of discovered association rules in large databases: An incremental updating technique. In *ICDE*. 106–114.

Cheung, D. W.-L., Lee, S. D., and Kao, B. 1997. A general incremental technique for maintaining discovered association rules. In *Database Systems for Advanced Applications*. 185–194.

Das, A., Ng, W.-K., and Woon, Y.-K. 2001. Rapid association rule mining. In *Proceedings of the tenth international conference on Information and knowledge management*. ACM Press, 474–481.

Duda, R. and Hart. 1973. *Pattern Classification and Scene Analysis*. Wiley&Sons, Inc.

Fu, Y. and Han, J. 1995. Meta-rule-guided mining of association rules in relational databases. In *KDOOD/TDOOD*. 39–46.

Garofalakis, M. N., Rastogi, R., and Shim, K. 1999. SPIRIT: Sequential pattern mining with regular expression constraints. In *The VLDB Journal*. 223–234.

Han, J. 1995. Mining knowledge at multiple concept levels. In *CIKM*. 19–24.

Han, J., Dong, G., and Yin, Y. 1999. Efficient mining of partial periodic patterns in time series database. In *Fifteenth International Conference on Data Engineering*. IEEE Computer Society, Sydney, Australia, 106–115.

Han, J. and Fu, Y. 1995. Discovery of multiple-level association rules from large databases. In *Proc. of 1995 Int'l Conf. on Very Large Data Bases (VLDB'95), Zürich, Switzerland, September 1995*. 420–431.

Han, J. and Kamber, M. 2000. *Data Mining Concepts and Techniques*. Morgan Kanufmann.

Han, J., Koperski, K., and Stefanovic, N. 1997. GeoMiner: a system prototype for spatial data mining. 553–556.

Han, J. and Pei, J. 2000. Mining frequent patterns by pattern-growth: methodology and implications. *ACM SIGKDD Explorations Newsletter 2,* 2, 14–20.

HAN, J., PEI, J., AND YIN, Y. 2000. Mining frequent patterns without candidate generation. In *2000 ACM SIGMOD Intl. Conference on Management of Data*, W. Chen, J. Naughton, and P. A. Bernstein, Eds. ACM Press, 1–12.

JAMES, M. 1985. *Classification Algorithms.* Wiley&Sons, Inc.

KLEMETTINEN, M., MANNILA, H., RONKAINEN, P., TOIVONEN, H., AND VERKAMO, A. I. 1994. Finding interesting rules from large sets of discovered association rules. In *Third International Conference on Information and Knowledge Management (CIKM'94)*, N. R. Adam, B. K. Bhargava, and Y. Yesha, Eds. ACM Press, 401–407.

KOPERSKI, K. AND HAN, J. 1995. Discovery of spatial association rules in geographic information databases. In *Proc. 4th Int. Symp. Advances in Spatial Databases, SSD*, M. J. Egenhofer and J. R. Herring, Eds. Vol. 951. Springer-Verlag, 47–66.

KOPERSKI, K. AND HAN, J. 1996. Data mining methods for the analysis of large geographic databases.

LEE, S. D. AND CHEUNG, D. W.-L. 1997. Maintenance of discovered association rules: When to update? In *Research Issues on Data Mining and Knowledge Discovery.* 0–14.

LENT, B., SWAMI, A. N., AND WIDOM, J. 1997. Clustering association rules. In *ICDE.* 220–231.

LU, H., HAN, J., AND FENG, L. 1998. Stock movement prediction and n-dimensional inter-transaction association rules.

MANNILA, H., TOIVONEN, H., AND VERKAMO, A. I. 1995. Discovering frequent episodes in sequences. In *Interantional Conference on Knowledge Discovery and Data Mining.* IEEE Computer Society Press.

MURTHY, S. K. 1998. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery 2,* 4, 345–389.

NG, R. T., LAKSHMANAN, L. V. S., HAN, J., AND PANG, A. 1998. Exploratory mining and pruning optimizations of constrained associations rules. 13–24.

PARK, J. S., CHEN, M.-S., AND YU, P. S. 1995. An effective hash based algorithm for mining association rules. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, M. J. Carey and D. A. Schneider, Eds. San Jose, California, 175–186.

PEI, J. AND HAN, J. 2000. Can we push more constraints into frequent pattern mining? In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM Press, 350–354.

PSAILA, G. AND LANZI, P. L. 2000. Hierarchy-based mining of association rules in data warehouses. In *Proceedings of the 2000 ACM symposium on Applied computing 2000.* ACM Press, 307–312.

SAVESERE, A., OMIECINSKI, E., AND NAVATHE, S. 1995. An efficient algorithm for mining association rules in large databases. In *Proceedings of 20th International Conference on VLDB.*

SMYTHE AND GOODMAN. 1992. An information theoretic approach to rule induction from databases. In *IEEE Transactions on Knowledge and Data Engineering.* IEEE Computer Society Press.

SRIKANT, R. AND AGRAWAL, R. 1996. Mining quantitative association rules in large relational tables. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data.* ACM Press, 1–12.

SRIKANT, R., VU, Q., AND AGRAWAL, R. 1997. Mining association rules with item constraints. In *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining, KDD*, D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, Eds. AAAI Press, 67–73.

THOMAS, S., BODAGALA, S., ALSABTI, K., AND RANKA, S. 1997. An efficient algorithm for the incremental updation of association rules in large databases. In *Knowledge Discovery and Data Mining.* 263–266.

WANG, W., YANG, J., AND YU, P. 1997. Efficient mining of weighted association rules (war).