

Chapter 5

Mining Data Bases and Data Streams

Carlo Zaniolo and Hetal Thakkar

Computer Science Department
University of California, Los Angeles
zaniolo@cs.ucla.edu, hthakkar@cs.ucla.edu

Abstract. Data mining represents an emerging technology area of great importance to homeland security. Data mining enables knowledge discovery on databases by identifying patterns that are novel, useful, and actionable. It has proven successful in many domains, such as banking, e-commerce, genomic, investment, telecom, web analysis, link analysis, and security applications. In this chapter, we will survey the main methods and applications of data mining and the information systems recently developed for supporting the mining process. We then overview the key areas of data mining research, in particular, on-line mining of massive data streams, such as those that flow continuously on the Internet and other communication channels. We show that the traditional store-now & mine-later techniques are no longer effective either because of the size of the data stream or because of the real-time response requirement. Thus new fast & light algorithms and suitable systems must be developed for mining data streams.

5.1 Introduction and Historical Perspective

The term “Data Mining” describes the processes and the techniques that discover and extract novel, nontrivial, and useful knowledge from large datasets [43].

The theory and practice of Data Mining (DM) have been attracting growing interest as a result of (i) their success in many diverse application domains, (ii) the progress in the enabling technology, and (iii) the growing number, size, and accessibility of data sets available for fruitful mining. As early as in the nineties, the large databases created by commercial enterprises represented the first area of opportunities. It was soon realized that although created to support specific operational needs, these databases contain troves of information about customers’ purchasing patterns and preferences: once these are discovered via data mining, the knowledge so acquired can be invaluable for improving the profitability of the company. For instance, retailers can use this information for targeted marketing, airlines can use it to design fare structures that maximize their profits, and so on. These early DM applications were deployed by companies as part of their “decision-support” and “business-intelligence” activities that were based on *Data Warehouses* specifically built for these activities. Thus, data collected from a company’s operational databases are integrated into a data warehouse to support business intelligence activities [26].

In the last decade, the DM field has been significantly broadened by the arrival of (i) the Web, whereby the whole world became accessible as a global data warehouse, and (ii) several new applications such as link analysis, security applications, and text mining. Furthermore, it became clear that

much of the information that is continuously exchanged on the Internet should be mined as it flows on the wire rather than from a stored database. In fact, a more traditional store-now & mine-later approach is often not possible because of real-time response requirements or simply because the data stream is too massive. In these situations, the information must be processed and mined as the data arrives. While this new research area, namely data stream mining, shares many similarities with database mining, it also poses unique challenges and requirements discussed in Section 5.4.

The progress made by DM technology is due to the confluent contributions of several research fields. Indeed, while failures tend to be orphans, DM has seen very many paternity claims emerge as a result of its success and interdisciplinary nature. We now address this topic briefly, and refer to [60] for a more comprehensive discussion of the subject. *Machine learning* is the Artificial Intelligence area where work on data mining started where research continues on very advanced methods for knowledge extraction. However, while learning on sparse data remains very difficult, great success has been achieved in applications working with large data sets. Then, scalability and computing efficiency become a paramount concern, prompting much research work to support and integrate data mining methods in Data Base Management Systems (DBMS). The research vision of inductive DBMS [29, 58], where DM queries are supported as regular DBMS queries, has yet to be realized, but much progress has been made and is also discussed in Section 5.4.

Many other areas and disciplines, such as visualization and statistics have contributed to the progress of DM. In particular, the field of statistics has been contributing to data analysis for more than a century, and is now challenged with the task of providing rigorous analytical foundations for data mining methods of proven empirical effectiveness [45]. Statistical data mining is discussed in [45], where it is also observed that DM plays a complementary role with respect to traditional data analysis. While DM's exploratory data analysis aims at discovering a pattern or a model from a given dataset data analysis often focused on 'confirmatory data analysis', where an assumed model needs to be confirmed and its parameters derived.

DM systems and software packages recently produced, by various groups and disciplines, also deserve much credit for the popularity of DM. Along with stand-alone DM systems such as WEKA [18], there are those that are part of statistical-software packages such as STATISTICA, CLEMENTINE and R [50, 16, 44], and those that are provided as extensions of commercial DBMS [54, 5, 40], besides the packages designed to support DM for specialized application domains.

Examples of DM Methods and Applications. Different DM methods are best suited for different applications. For instance, in border control, risk factor of any crossing vehicles can be computed on the basis of various attributes, such as vehicle type, number of times the vehicle crossed the border in last few months, driver's license type, driver's visa status, etc. Once the data has been integrated from various sources, *predictive* mining models, such as classification, are often used to determine the risk involved. These techniques have also been widely used in financial applications, where banks often perform loan payment prediction and risk assessment on the basis of attributes such as income and credit history. describing the customer. Another well-known application is targeted marketing, where potential customers are segmented into groups of similar traits and characteristics. *Clustering* techniques represent the mining tool of choice in this second type of applications. Samples that do not fall in the clusters are called *outliers*.

In crime analysis, outliers are often associated with individuals or situations that deserve to be further investigated. For instance, when monitoring computer networks for intrusion detection, outliers often denote anomalous activities by users, which in turn indicate a possible attack by hackers. Along with classification and clustering, *association* represents the third core mining method. The objective of this method is to discover patterns and predictive rules. For instance, by mining records of sale transactions we might be able to discover that a customer who buys certain books is also likely to buy others.

In the next section we discuss these three core methods—i.e., classification, association, and

clustering—along with simple applications of such methods. More complex applications, including security applications, which often require a combination of several DM techniques, are discussed in Section 5.3.

For instance, to detect money laundering, we might employ text mining tools (to identify the entities involved in company documents, emails, text messages, etc.), link analysis tools (to identify links between different individuals, companies, documents, and accounts), outlier-detection tools (to detect unusual money transfers), and various visualization tools. Similarly, intrusion detection, which aims to prevent attacks to the integrity, confidentiality, and/or availability of computers, involves classification, and detection of sequential patterns, besides outlier analysis. Furthermore, all data mining applications, and in particular homeland security applications, must be preceded by extensive preparation steps to collect and clean the relevant data (e.g., record consolidation for name matching), discretization, and other preprocessing steps discussed in the next section.

The organization of this chapter reflects the historical evolution of the DM field and its data-driven applications: in the next section we discuss knowledge discovery from databases, then we discuss new applications particularly, those driven by the web (which can be viewed as a global database), and Section 5.4 covers data stream mining.

5.2 Data Mining Methods

We now discuss the more established methods and techniques of *Knowledge Discovery in Databases (KDD)*. KDD is the algorithmic analysis of large data repositories to identify novel and useful patterns in the data. The algorithms that are at the heart of the KDD process can be grouped into methods that perform similar DM tasks, such as classification, association, and clustering. These three core methods are described in more detail in this section.

5.2.1 Taxonomy of Data Mining Methods

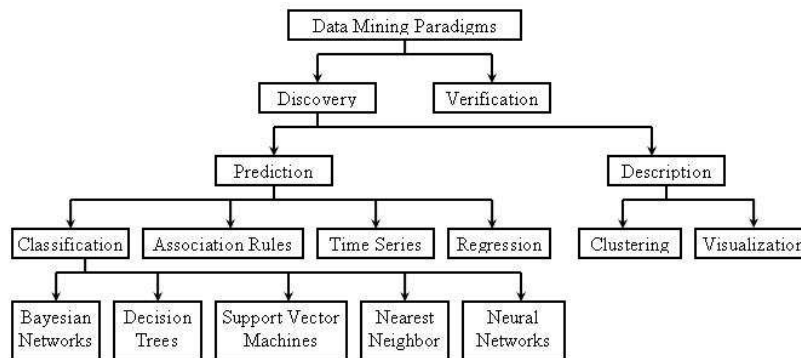


Figure 5.1: Taxonomy of Data Mining Methods

Mining Tasks and Algorithms. Figure 5.1 provides a skeleton taxonomy of the main mining tasks discussed in the next section.

In addition to the hierarchy presented in Figure 5.1, distinction between supervised and unsupervised learning is also common in DM literature. Here again clustering and classification are at the opposite extremes: clustering requires little input from the users (e.g., the number of clusters to be constructed), while classification requires numerous examples that the classifier uses to learn and emulate the behavior of the system that has produced those examples.

5.2.2 Classification and Prediction

Classification is the task of predicting the value of a particular attribute in a tuple on the basis of the values of other attributes, using a model learned from training data. In other words, classifiers are trained on a large set of examples from which they must learn how to best assign a category to new tuples. Because of their ability of making predictions based on training examples, classifiers represent a fundamental DM method used in many and diverse applications. Examples include detecting spam email messages, given the message header, content, subject, etc., or deciding whether a cargo entering US is high-risk (e.g. contains drugs) given the vehicle type, driver’s license type, driver’s age, time of day, etc. Classification is a supervised learning method, insofar as it is driven by an initial set of examples describing the behavior that the classifier must learn to emulate. These examples are normally given in the form of a database relation such as that of Table 5.1, below.

VehicleType	DriverLicType	DriverAge	TimeOfDay	HighRisk
passenger	regular	>= 35	night	N
passenger	regular	>= 35	day	N
workvan	regular	>= 35	night	Y
truck	disabled	>= 35	night	Y
truck	commercial	< 35	night	Y
truck	commercial	< 35	day	N
workvan	commercial	< 35	day	Y
passenger	disabled	>= 35	night	N
passenger	commercial	< 35	night	Y
truck	disabled	< 35	night	Y
passenger	disabled	< 35	day	Y
workvan	disabled	>= 35	day	Y
workvan	regular	< 35	night	Y
truck	disabled	>= 35	day	N

Table 5.1: Examples of past border crossings.

In this table, we have a tuple with four attributes describing a vehicle crossing the border and a binary HighRisk attribute, whose value is either *Y* or *N*. Here *Y* (versus *N*) denote if the vehicle may contain drugs or illegal weapons (versus not). This example shows data of potential interest to the Automated Commercial Environment (ACE) program of the U.S. Department of Homeland Security [1]. ACE attempts to integrate data from different government agencies to enable accurate data analysis for border control [1]. Independent of the interpretation of the values in the table, the objective of classification is (i) to learn the function \mathcal{F} that maps the known vehicle attributes into the Y/N values (possible values) of HighRisk (unknown attribute), and (ii) to evaluate the accuracy of \mathcal{F} . For this purpose the original set is divided into (i) a training set of rows for building the classifier, and (ii) a testing set of rows for determining the accuracy of the classifier¹. Thus, the known vehicle attributes of each testing row is fed to the learned classifier and the prediction is compared with the actual label. The number of correct decisions over the number of tuples tested, defines the accuracy of the classifier. While high accuracy can be expected in many cases, this is normally well below 100% because of the following reasons: (i) the limitations of the classification algorithm used in learning complex \mathcal{F} functions, and (ii) the function \mathcal{F} is non-deterministic—i.e., a vehicle might be not involved in criminal activities although another with identical attribute values is. Several classifier algorithms have been proposed in the literature and are discussed next. The applicability and accuracy of the classification algorithms depends on the application domain, i.e. not all algorithms apply to any application domain.

¹For instance 70% of the rows could be used for training and 30% for testing.

Naive Bayesian Classifiers

The training set for a mini-example, containing 14 tuples, is shown in Table 5.1 (Note: real-life applications contain thousands or even millions of tuples and many more attributes). The *HighRisk* value is Y in 9 of these examples and N in the remaining 5. Thus, if we had to attempt a prediction with only this information, we would most likely go with Y , since the probability of Y , $pr(Y) = 9/14$, is larger than $pr(N) = 5/14$. However, the tuples also contain information about the vehicle attributes associated with the Risk values, thus we can compute conditional probabilities to improve the quality of our predictions. The conditional probabilities of the VehicleType being either passenger, or workvan, or truck, for *HighRisk* = Y is estimated (by counting the entries in the table) as:

$$pr(passenger|Y) = 2/9, \quad pr(workvan|Y) = 4/9, \quad pr(truck|Y) = 3/9.$$

Similarly, for *HighRisk* = N we find: $pr(passenger|N) = 3/5$, $pr(workvan|N) = 0$, and $pr(truck|N) = 2/5$. Similar count-based estimations of conditional probabilities can be easily obtained for the remaining vehicle attributes. Table 5.2 shows these probabilities with the unconditional probability in the last row.

Value	Ys	Ns
passenger	2/9	3/5
workvan	4/9	0
truck	3/9	2/5
regular	2/9	2/5
disabled	4/9	2/5
commercial	3/9	1/5
>= 35	3/9	4/5
< 35	6/9	2/5
day	3/9	3/5
night	6/9	2/5
all	9/14	5/14

Table 5.2: Probabilities for the Training set of Table 5.1

Having collected this information, the classifier must predict, which one of the Risk values, Y or N , is more likely given a vehicle tuple with four attributes:

$$\langle \text{VehicleType}, \text{DriverLicType}, \text{DriverAge}, \text{TimeOfDay} \rangle.$$

If X is the current value of this 4-tuple, then the right prediction is the value of C that maximizes the value of $pr(C|X)$. But by Bayes' theorem we obtain that:

$$pr(C|X) = \frac{pr(X|C) \times pr(C)}{pr(X)}$$

Since X is given, $pr(X)$ is a constant that can be ignored in finding the maximum C . Also, $pr(C)$ was previously estimated as $pr(Y) = 9/14$ and $pr(N) = 5/14$. However, estimating the value of $pr(X|C)$ to any degree of accuracy can be very difficult. If we make a naive simplifying assumption that the weather attributes are independent of each other then we have:

$$pr(\langle X_1, \dots, X_k \rangle | C) = pr(\langle X_1 \rangle | C) \times \dots \times pr(\langle X_K \rangle | C)$$

Then, we simply plug the correct values from Table 5.2 into this formula and compare the results to find the one with the highest probability. For instance, say that our input sample is $X = \langle \text{truck}, \text{regular}, \geq 35, \text{night} \rangle$: then we have: $pr(X|Y) \times pr(Y) =$

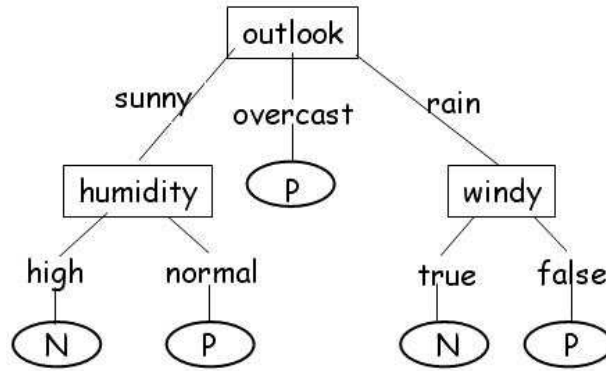


Figure 5.2: Decision Tree Classifier

$$pr(truck|Y) \times pr(regular|Y) \times pr(>= 35|Y) \times pr(night|Y) \times pr(all|Y) = \\ 3/9 \times 2/9 \times 3/9 \times 6/9 \times 9/14 = 0.0105$$

and $pr(X|N) \times pr(N) =$

$$pr(truck|n) \times pr(regular|N) \times pr(>= 35|N) \times pr(night|N) \times pr(all|N) = \\ 2/5 \times 2/5 \times 4/5 \times 2/5 \times 5/14 = 0.0182$$

Since 0.0182 is larger than 0.0105, our prediction is N .

Observe that, these computations can be derived by restricting the Table 5.2 to the rows containing the values *truck*, *regular*, ≥ 35 , *night*, and *all* (unconditional) and then performing a vertical multiplication on the Y and N columns. Thus a Naive Bayesian Classifier(NBC) can be easily implemented on a spreadsheet system or on a relational DBMS using the basic aggregate functions in their standard language SQL². Although so simple, NB classifiers normally produce accurate results, even in the situation where the naive assumption of independence between the different attributes is not satisfied. Simple generalizations of NBC are also at hand [53] for continuous attributes, whereby the actual driver age replaces the categorical ≥ 35 and < 35 in the training set: then in Table 5.1, we can use entries that describe the statistical distribution of these values rather than the values themselves. Finally, it is possible to overcome the naive assumption of independence between attributes, and use Bayesian networks to describe their correlations. The more complex techniques required by this approach are outside the scope of this paper.

Decision-Tree Classifiers

Although NB classifiers can provide efficient and accurate predictions, their ability to offer insights into the nature of the problem and the attributes involved, is quite limited. On the other hand, decision-tree classifiers are effective at revealing the correlation between decisions and the values of the pertinent attributes. Furthermore, these correlations are expressed through simple rules.

For instance, Figure 5.2 shows a simple decision-tree classifier for the training set of Table 5.1. Each node of the tree represents an attribute and the branches leaving the nodes represent the values of the attribute. Much understanding of the nature of decision process is gained through this tree. Thus, we conclude that when the VehicleType is *workvan*, then the decision is Y , but when the VehicleType is passenger then the decision also depends on DriverAge, while TimeOfDay becomes important when it is a truck. A more formal statement of this understanding can come in the form of rules, one for each leaf node in the tree. For the example at hand, we have five rules as follows:

²SQL is an acronym for Structured Query Language.

VehicleType = passenger, DriverAge >= 35	→	N
VehicleType = passenger, DriverAge < 35	→	Y
VehicleType = workvan	→	Y
VehicleType = truck, TimeOfDay = day	→	N
VehicleType = truck, TimeOfDay = night	→	Y

Clearly, more compact trees yield terser and more incisive rules, and provide a better descriptive model for the data. Therefore, a number of techniques have been proposed for pruning the decision tree for limiting its growth, while assuring that the predictive accuracy of the resulting tree is not impaired.

The following two steps are repeated recursively to generate a decision tree (i) select the ‘best’ attribute, say B , for classification, and then (ii) partition the examples in the training set according to the value of attribute B . The selection of the best attribute is normally performed greedily to minimize the heterogeneity (a.k.a. impurity) of the generated partitions. Several criteria have been proposed to select the best partition, for instance the Gini index³ is probably the simplest among them. The Gini index of a partition S containing both N and P samples can be measured as:

$$gini(S) = 1 - F_P^2 - F_N^2$$

where $F_P(F_N)$ denotes the number of $P(N)$ samples divided by the total number of samples in S . Thus, when S contains the same number of P and N samples $gini(S) = 1/2$. However, $gini(S) = 0$, when S is pure (i.e., homogeneous) and contains only P samples or only N samples. With the objective of partitioning our training set into classes that are as homogeneous as possible. Thus, we select the attribute with the minimal Gini index, where Gini index of an attribute is the weighted sum of the Gini index of each individual attribute value. Here weight is defined as the fraction of tuples that attain a given attribute value. For instance, we can group the the rows in Table 5.1 by the *VehicleType* attribute and partition the table into three groups. The first group is the *workvan* group with four Y s and 0 N s: thus its Gini index is zero with weight $4/14$. The second group is the *truck* group with three Y s and two N s: thus its Gini index is $(1 - 3^2/5^2 - 2^2/5^2)$ and its weight is $5/14$. The third and final group is the *passenger* group with two Y s and three N s with Gini value $(1 - 2^2/5^2 - 3^2/5^2)$ and weight $5/14$ thus the sum of the Gini values multiplied by their weight produces a Gini value of $9/35$ for the *VehicleType* partition. The same computation repeated for the other three columns produce higher Gini indexes. Thus we put *VehicleType* at the root of our decision tree and proceed to partition the samples according to their *VehicleType* values and obtain three subtables, one for each value of *VehicleType*. Now, we recursively repeat this attribute selection+partition procedure on each of these subtables. Thus, for the subtable where *VehicleType* = *passenger* we find that the best (lowest Gini index) partition is on *DriverAge*, while for *VehicleType* = *truck* the best partition is on *TimeOfDay*. However, the subtable generated from *VehicleType* = *workvan* is homogeneous since *HighRisk* = Y in all of these tuples: thus we simply label this branch as Y .

The situation where all tuples are homogeneous (or the level of heterogeneity is below a certain threshold) represents the first and most desirable stopping condition for the algorithm. Two less desirable outcomes are (i) all attributes have already been used, but the qualifying tuples are not homogeneous and (ii) the last partition does not contain any qualifying tuples. Thus, any testing tuples that are processed at case (i) node will have to be assigned a majority label, whereas a random label will have to be assigned for case (ii).

Decision-tree classifiers are unique in their ability to achieve both high descriptive quality and high predictive accuracy. Much research work has been devoted to improve decision tree classifiers and produce robust/scalable algorithms. Some of these algorithms use information gain or χ^2 -based measures, as opposed to the Gini index, for attribute selection. Algorithms and strategies for decision tree pruning have also been proposed. Some algorithms use multi-way splitting rather than binary

³Named after the Italian statistician Corrado Gini who introduced it his 1912 paper “Variabilità e mutabilità” (“Variability and Mutability”).

splitting as described in the examples. Binary split algorithms can also be extended to deal directly with continuous data (i.e., without requiring them to be discretized) by simply finding the optimum split value V for a given attribute A . Thus, samples with A -value less than V are assigned to one branch and the remaining ones to the other branch. Detailed discussion on these advanced decision tree methods can be found in [2, 53].

5.2.3 Association Rules

Whenever a market basket is pushed through the checkout counter of a grocery store, or a “checkout-and-pay” button is clicked during a web transaction, a record describing the items sold is entered in the database. A business enterprise can then use such records to identify the frequent patterns in such transactions. These patterns can then be described by the following rule stating that if a market basket contains both Diapers and Milk it also contains Beer.

$$r_1 : \text{Diapers, Milk} \rightarrow \text{Beer}$$

A marketing analyst would only find such rules useful if the following three conditions hold.

1. the rule is valid, at least to a certain level of statistical *confidence*,
2. the rule occurs frequently, i.e., has certain level of *support* and,
3. the rule is new and interesting, i.e., the rule was not previously known and can potentially be put to use to improve the profitability of company (e.g., by bundling these three items in a sale campaign or by placing them in nearby locations on the shelf).

Obviously condition 3 can only be decided by the analyst who is intimately familiar with the domain. Thus a DM system generates and presents to the analyst all rules that satisfy conditions 1 and 2: such rules can be found by counting the frequency of the itemsets in the database. Consider for instance the following database, containing five transactions.

TID	Itemset
1	{ Bread, Milk }
2	{ Bread, Beer, Diapers, Eggs }
3	{ Beer, Cola, Diapers, Milk }
4	{ Beer, Bread, Diapers, Milk }
5	{ Bread, Cola, Diapers, Milk }

Table 5.3: Examples of Market Basket Transactions

We are interested in rules that have a high level of *support* and *confidence*. The *support* of a rule is defined by the fraction of transactions that contain all the items in the rule. The example database contains five transactions and two of those contain all the items of rule r_1 (i.e., the itemset {Beer, Diapers, Milk}). So the support for r_1 is $S_1 = 2/5 = 0.4$.

The confidence for a rule is instead the ratio between the support of the rule and the support of its left side: the left side of r_1 is {Diapers, Milk} which has support $S_2 = 3/5 = 0.6$. Therefore, the confidence for the rule is $S_1/S_2 = 0.66$. In many situations, only rules that have a high level of confidence and support are of interest to the analysts. Thus, for instance if the analyst specifies a minimum confidence ≥ 0.66 and a minimum support ≥ 0.6 , then r_1 fails because of insufficient support. However, the following two rules meet this criteria:

$$r_2 : \text{Milk} \rightarrow \text{Bread}$$

$$r_3 : \text{Bread} \rightarrow \text{Milk}$$

In fact, both r_2 and r_3 have support equal to $3/5 = 0.6$, and confidence equal to $3/4 = 0.75$.

The symmetric role played by Bread and Milk in these rules, illustrates the fact that the rules we construct only establish a strong correlation between items, without establishing a causality implication. While a more in-depth analysis of the dataset can reveal more about the nature of the relationship between the items, the analyst is normally the best and final judge on the significance of the rule. For instance, a rule frequently discovered when mining the data of a large department store, is that people who buy maintenance contracts are likely to buy large appliances. While this represents a statistically valid association, it is useless in predicting the behavior of customers. In fact, we know that the purchase of a maintenance contract represents the consequence, not the cause of purchasing a large appliance. Only the analyst can discard such trivial rules. Thus we must make sure that the algorithms do not overwhelm the analysts by returning too many candidate rules. Also, we are dealing with hundreds of thousands of transactions and finding all those rules can become computationally intractable as the database size increases. The problems of limiting the number of rules generated and the running time for generating them can both be effectively addressed by, setting a high minimum support level. Under this assumption, there exist efficient algorithms to generate all frequent itemsets from very large databases of transactions as we discuss next.

The Apriori algorithm operates by generating frequent itemsets of increasing size: the algorithm starts with singleton sets and then from these it generates frequent pairs, then frequent triplets from frequent pairs, and so on. The algorithm exploits the downward closer property (*If an itemset is frequent then all its subsets must also be frequent*). Let us use the market-basket example with minimum support of 0.60 to illustrate how this principle is applied.

Step 1: The algorithm first find singleton items with frequency $\geq 3/5 = 0.60$. Thus we get the following items (with frequency). {Beer}:3, {Bread}:4, {Diapers}:4, {Milk}:4.

Step 2: Next the frequent pairs are obtained by combining the singleton frequent items: thus we obtain the following candidate pairs: {Beer, Bread}, {Beer, Diapers}, {Beer, Milk}, {Bread, Diapers}, {Bread, Milk}, {Diapers, Milk}.

Step 3: Then we filter these pairs to find the ones with frequency greater than or equal to the minimum support threshold (frequency ≥ 3 in this case):

{Beer,Diapers}:3, {Bread,Diapers}:3, {Bread,Milk}:3, {Diapers,Milk}:3.

Step 4: Next the algorithm joins these pairs based on the shared items to find candidate triplets—Note that these triplets may not qualify as candidates, if any of its subset is not frequent (the Apriori principle). Thus as the first two pairs are joined to form {Beer, Bread, Diapers}, we find that its subset {Beer, Bread} is not frequent: thus this triplet is not a candidate frequent itemset. Likewise, {Beer, Diapers, Milk} is missing the frequent subset {Beer, Milk}. Thus the only triplet that can possibly be frequent is, {Bread, Diapers, Milk}.

Step 5: Then the algorithm counts the occurrences of {Bread, Diapers, Milk} for which we find three occurrences—thus it is frequent. With no pairs of triplets to combine into a quadruplet, the algorithm terminates.

Thus, the algorithm constructs all candidate sets with support better than 0.6. The minimum confidence requirement is also easy to implement since it is now trivial to compute their confidence. For instance, let us assume that the required minimum confidence is 0.9. Then, we see that a rule Bread, Diapers \rightarrow Milk qualifies, since {Bread, Diapers, Milk} and {Bread, Diapers} have the same support and thus the rule has confidence 1. However, the rule Bread \rightarrow Milk does not qualify, since the support of {Bread, Milk} is $3/5$ while that of {Bread,} is $4/5$; thus the confidence is only 0.75.

While Apriori is rather simple and efficient, several algorithms, such as FP-growth [27], have been proposed to reduce the cost of finding frequent itemsets and to eliminate the need of performing multiple passes through the database.

Sequential Patterns Often there is a temporal correlation between items purchased, whereby, say customers who buy a suite and a shirt, often return to purchase a belt and a tie. This behavior can be modeled as a sequence of three itemsets: $S_1 = \langle \{suit, shirt\}, \{belt\}, \{tie\} \rangle$. Then, S_2 is said to be a subsequence of S_1 if it can be obtained from S_1 by eliminating some itemsets. Thus $S_2 = \langle \{suit\}, \{belt\}, \{tie\} \rangle$ and $S_3 = \langle \{suit, shirt\}, \{belt\} \rangle$, are both subsequence of S_1 . Thus we can now determine the support for a temporal rule such as: $\langle \{suit\}, \{belt\} \rangle \Rightarrow \langle \{tie\} \rangle$, by simply counting the number of sequences in our database that have $\langle \{suit\}, \{belt\}, \{tie\} \rangle$ as their subsequence. For example say that our database only contains sequences S_1, S_2 and S_3 ; then its support of this rule is $2/3$. The confidence in the rule is also $2/3$. Observe that the Apriori principle also holds here: if a sequence occurs k times in the database, then its subsequences must occur at least k times. Thus algorithms inspired by the Apriori algorithm can be used to find sequences whose frequency and confidence exceed given thresholds.

5.2.4 Clustering Methods

Given a large set of objects described by their attributes, clustering methods attempt to partition them into groups such that the similarity is maximized between the objects in the same group. The similarity is computed based on the attributes of the objects. Clustering often represent the first step in a knowledge discovery process. For instance, in marketing applications clustering for customer segmentation makes it possible to learn the response of the population to a new service or a product by interviewing sample customers in each cluster. In intrusion detection, clustering is the first step to learn the behavior of regular users making possible for the analyst to concentrate on outliers and anomalous behavior.

Clustering methods have been used in a wider variety of applications and longer than any other DM method. In particular, biologists have long been seeking taxonomies for the living universe and early work in cluster analysis sought to create a mathematical discipline to generate such classification structures. Modern biologists instead use clustering to find groups of genes that have similar expressions. Librarians and archivists have long used clustering to better organize their collections. Moreover, as discussed in the next section, clustering represents a very valuable tool for organizing and searching the billions of pages available on the web and monitor the innumerable messages exchanged as email. Let's assume that we want to analyze this data to detect terrorist activities or other suspicious activities. This data is highly dynamic, which precludes the use of classification algorithms. Therefore, after appropriate cleaning, each document can be converted to a feature vector containing the terms and their occurrence in the document. Then the documents can be clustered based on these feature vectors, such that the documents with similar feature vectors are assigned to the same cluster. Such clustering can result in interesting clusters that can be further investigated by human experts, e.g. a cluster where the top distinctive words are *biological virus* and *jihad*.

Many clustering algorithms have been proposed over the years and their detailed discussion is well beyond the scope of this paper. Therefore, we instead group them into a crude taxonomy and briefly discuss sample algorithms from the main classes of the taxonomy.

- **Hierarchical Clustering Methods.** These methods assume an underlining conceptual structure that is strictly hierarchical. Thus agglomerative methods start with individual points and, at each step, merge the closest pair of clusters—provided that their distance is below a given threshold (otherwise we end up with one all-inclusive cluster containing all points). Divisive methods instead start from one all-inclusive cluster and at each step split clusters that contain dissimilar points.
- **Density Based Clustering.** This method assigns clusters to points based on the neighborhood of the point, i.e. if many neighbors of a point belong to cluster C then the point is also assigned to cluster C, even if the center of cluster C is not close to the point. Therefore density-based methods can generate clusters of any shape and as such can describe trajectories and epidemic phenomena. A popular density-based algorithm called DBSCAN is described in Section 5.4.

- **Prototype-Based Methods.** These seek to cluster the objects near a set of prototype points that summarize the clusters—typically the mean or the median of the points in the cluster. A very popular method in this class is the **K-means** algorithm which is discussed next.

The K-Means Algorithm:

1. Select K points as initial centroids
 2. **repeat** the following two steps:
 - (i) Form K clusters by assigning each point to its closest centroid.
 - (ii) Recompute the centroid of each cluster.
 3. **until** the centroids do not change.
-

While clustering is considered an unsupervised learning method, most clustering algorithms require a fair degree of guidance from the analyst. Furthermore, the correct value of K must be specified by the user. A value of K that is too low will cause some clusters to be too sparse: on the other hand a value of K that is too large will generate clusters that have centroids very close to each other and should be merged. In practice therefore, the application of the algorithm could be an iterative process that, e.g., starts with a low value of K and then increases it by splitting sparse clusters. Even after the correct value of K is determined, the selection of the initial K points remains a delicate operation. Indeed, the choice of different initial points can result in different final centroids. One solution to this problem consists in randomly selecting a small sample of the dataset and clustering it using hierarchical clustering methods.

A centroid is computed as the average (i.e., the center of mass) of its points. This point is overly sensitive to noise and outliers, and rarely corresponds to an actual object. An alternative, and often preferable solution, consists of using medoids, i.e., the mean points of the clusters. Another decision that the analyst must make is which matrix to use for similarity between the objects. Common matrices include Euclidian distance, Manhattan distance, etc. Other notions of similarity are often preferable for particular applications. For instance, the cosine distance is often used for documents and the Jaccard and Tanimoto coefficients are used as similarity measures for binary data.

5.2.5 Other Mining Techniques

The core techniques described so far are supplemented by assortment of other techniques and methods, which we only summarize due to space limitations. Among the many techniques proposed for predictions, we find Support Vector Machine (SVM), a computationally expensive classification method that achieves excellent accuracy and avoids the “curse of dimensionality” problem. The goal of SVM is to separate sets of examples belonging to different classes by hyperplanes, as to maximize the distance between the hyperplane and the examples in each class. Nearest-Neighborhood classifiers instead attempt to achieve compact representations for the training set, so that any new sample can be classified on the basis of the most similar ($k \geq 1$) examples in the training set [53]. Thus, rather than providing a higher-level descriptive model, nearest-neighborhood classifiers justify their decision on the basis of similar instances from the training samples. Moreover, both descriptive model and intuitive connections with training samples can be lost when using *Neural Networks* [53]. This is a popular method that requires a long training phase but can deliver good predictive performance in applications such as handwritten character recognition and speech generation.

Regression is a popular statistical method based on well-developed mathematical theories, which is not restricted to linear functions, but can also be used for non-linear, multivariate functions, and event probabilities (logistic regression). Even so it cannot match the performance, and generality of classifiers in highly non-linear problems. Take for instance the analysis of *time series* and *event sequences*, such as stock quotations on a ticker tape. To perform trend analysis in this environment, the analyst

can use regression-based methods (after suitable smoothing of the data). However in most advanced applications the analyst is interested in detecting patterns, such as the ‘double bottom’ pattern, that provide a reliable predictor of future market gains. Special techniques are used to recognize these patterns and similar ones defined by relaxing the matching requirements both in the amplitude and time (time warping). Special query language constructs, indexing, and query optimization techniques (Section 5.4) are used for this purpose [49].

In many respects, *outlier analysis* complements these methods by identify data points that do not fit into the descriptive mining model. Therefore, outlier mining is often a ‘search & destroy’ mission, since outliers are either viewed as noisy data (e.g., incorrect values due to faulty lines) , or true but inconvenient data that can hamper derivation of mining models. However, applications such as fraud analysis or intrusion detection focus on the recognition and in-depth study of outliers, since these are frequently an indication of criminal activities. An assortment of other techniques ranging from OLAP data cubes to logistic regression [24] can be used for outlier detection.

5.2.6 The KDD Process

Figure 5.3 summarizes the overall KDD mining process consisting of the several steps, including

- *Data Collection, Integration and Cleaning*
- *Data Exploration and Preparation*
- *Selection and Application of DM tasks and algorithms*
- *Evaluation and application of DM results*

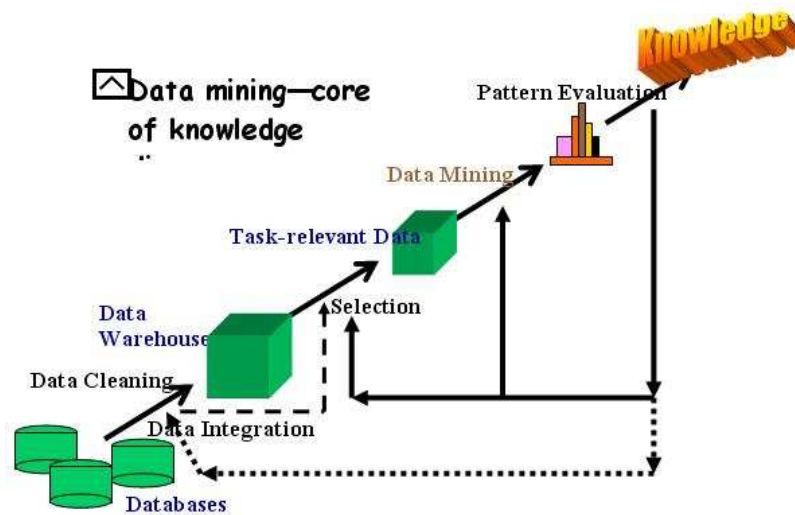


Figure 5.3: The KDD Process: main steps

Because of its exploratory nature the DM process is often iterative: if at any point in the process, the analyst becomes dissatisfied with the results obtained so far, he or she returns to previous steps to revise them or restart from scratch. We can now elaborate on these steps.

Data Collection, Integration, and Cleaning. Before any mining can begin, data must be collected and integrated from multiple sites. Furthermore, it must be “cleaned” to remove noise and eliminate discrepancies (e.g., by consolidating different names used for the same entity). These operations constitute time-consuming and labor-intensive tasks that lead to construction of a *Data Warehouse*. Thus,

a data warehouse is an integrated database into which a company consolidates the information collected from its operational databases, for the purpose of performing business-intelligence and decision-support functions. These functions often include a library of data mining methods—along with simpler tools for On-Line Analytical Processing (OLAP), visualization, and statistical analysis. We next discuss these functions and various data-preparation tasks that often precede and pave the way to the actual deployment of DM methods.

- *OLAP Analysis.* OLAP tools have met great success in business applications, inasmuch as (i) they provide users with easy-to-understand multidimensional views of their data, and (ii) they are now supported by most commercial DBMSs. Thus, by a simple SQL query, the manager analyzing the sales can retrieve the summary of the recent sales for each region, city, and each store within the city—and thus quickly identify outliers requiring further analysis. This analysis can e.g., be performed in the dimensions of (i) location, hierarchically structured by region, city, and store; (ii) time, hierarchically structured by year, season, month, week, and day; and (iii) product types. Thus a simple OLAP query can return a hypercube-based representation of all the sales of interest hierarchically partitioned along the dimensions of interest. Similar SQL queries can also produce roll-up, drill-down, slice, and dice views of this data [26]. An application of OLAP analysis to associate criminal events is discussed in the next section.
- *Visualization.* People’s ability to absorb large amounts of information and recognize patterns therein is limited by the available views of the data. Histograms, pie charts, scatter plots, density plots, vector plots, and spatio-temporal charts are very effective in visualizing low-dimensional data and the results produced by statistical and OLAP functions. On the other hand, visualization methods suffer from the “dimensionality curse” since they cannot handle high-dimensional data well—although techniques based on parallel coordinates, star coordinates, and Chernoff faces have proved effective in special cases.
- *Data Statistics.* Standard notions from statistics can be used to describe continuous data by their central tendency and dispersion. Measures of central tendency include mean, median, and midrange, while measures of data dispersion include quantiles, interquartile range, and variance. These statistics and related analytics are efficiently supported on large databases by OLAP queries of commercial DBMS, and can be visualized via histograms, quantile plots and box plots.

DM mining methods and in particular clustering, can also be used to gain a better understanding of the structure of the data in preparation for the actual mining tasks. The natural outcome of the data exploration step is that the analyst achieves a better understanding of which DM algorithms should be applied to the data. However, before such algorithm can be applied, one or several of the following preprocessing steps are frequently needed:

- *Eliminating missing values.* Special symbols called nulls are normally used to represent information in the collected data set. In certain cases, the DM algorithm can treat nulls as any other value and produce satisfactory results. In many cases however, this approach results in errors or decreased accuracy. Thus, null values must be eliminated, by either removing the table rows containing these null values all together, or by replacing each null with a probable value derived as the mean of the column (non-null) values or by other probability-based estimation.
- *Discretization of Numerical Values.* Table attributes can either be categorical (i.e., allowing only a small set of values) or numerical (i.e., representing continuous variables, such as access time or driver’s age). Many mining algorithms support continuous attributes or perform the discretization as part of the algorithm itself. But in some situations, such as driver’s age in border control, or employee salary in intrusion detection, discretization is best performed on the basis of the analyst’s understanding of semantic significance of the various intervals in the domain.

- *Dimensionality Reduction.* In some large applications this step can be used to either reduce the number of rows or columns in the dataset. Sampling and aggregation over a dimension (see OLAP cubes) can be used to reduce the number of rows. While discretization does not reduce the number of rows involved, it can make their representation more compact; compression techniques based on wavelet transforms and principal component analysis methods have also been used for the same purpose. Feature selection methods can instead be used to reduce the number of columns, by identifying and eliminating the columns that are not critical for the mining task.

Evaluation of DM results. The quality and the usefulness of the results produced must be evaluated for the application at hand. For predictive tasks, the quality of the model is evaluated in terms of the accuracy of the predictions. The usefulness of the results is normally evaluated in terms of (i) the novelty of the patterns discovered and (ii) their generality. For (i) it is clear that while confirmation of preexisting knowledge produces greater confidence in the accuracy of the DM approach, it does not deliver immediate business value to the company. The business value of new knowledge is often measured by its ability to inspire actions that increase the profitability of the business (e.g., by detecting a previously unknown pattern of frauds or vulnerability in the system). Finally, the benefits grow with the realm of applicability of the new rule or pattern that is learned: a widely applicable business rule, is better than a very specific one.

5.3 Web Mining and New Applications

The last decade has seen a significant growth in DM technology and applications; in particular, the web has tremendously expanded the information available for DM and created new application opportunities. Two new research areas have thus emerged: one is web mining—i.e., the global mining of information retrievable from web sites; the other is data-stream mining—i.e., the online mining of information exchanged over the communication lines of the internet. In the meantime, much progress has been achieved in DM methods, applications, and systems, and success stories from fields as diverse as security, biology, and marketing have inflated perceptions on the power and reach of DM technology. As a result, major information management software vendors are moving aggressively to provide systems that integrate a rich assortment of DM methods and can support the whole mining process discussed in the previous section. A parallel development has been the creation of large datasets for mining that, eventually, can be shared between cooperating companies and agencies, or even published for research purposes. This trend has been met with serious concerns about the privacy of individuals being compromised as a result, and the new research area of *Privacy-Preserving DM* has emerged to address this concern. In this section, we briefly discuss web mining, security, and privacy.

5.3.1 Web Mining

The web can be regarded as the largest database available and thus offers endless opportunities for knowledge discovery. However, the discovery process is more difficult here than in traditional databases, because of the lack of uniformity between web sites, and the prevalence of unstructured and semistructured information.

Web Usage Mining

Web usage mining is the study of the data generated by the web surfers' sessions. For instance, users' click streams recorded in the web logs of the server can be mined for frequent association patterns, which can then be used (i) to improve the design of the web site and (ii) to improve the recommendations and ads presented to the user.

Web Content Mining

In many respects, the growth of the web is intimately connected with that of search engines that use (i) crawlers to systematically download pages and documents from the web and (ii) Information Retrieval (IR) methods to index and rank the web pages on the basis of their contents, as to promptly return the URL of the relevant pages in response to keyword-based searches. While content mining of multimedia documents represents a topic of growing interest, the mining of text documents remains the cornerstone of search engines, which often combine classical IR techniques, with web-oriented enhancements (e.g., those based on link analysis).

In the keyword-based approach of IR, the document categorization is performed using the terms/words that occur in the documents. Then stop-words are removed: these are words such as article that are very common in the languages, and thus have little discriminative value. Additionally, word consolidation is often applied whereby synonyms and words represented by same stems are merged.

Therefore, a document is now represented by a vector of terms, where a frequency is associated with each term. These frequencies are first normalized relative to the total number of words in the document; then, it is further scaled down by using a log scale. Finally, to increase the discriminating power of the term-frequency vector, the term-frequency is also scaled down in proportion to the number of documents in the collection containing this particular term.

Now, when a query containing a vector of search terms is submitted by a user, the system computes the normalized inner product of the query vector with the term-frequency vector (cosine measure) for each document in the collection, and ranks the documents based on the cosine measure.

While the techniques described above apply to traditional text documents and web pages alike⁴, the latter also contain *tags* and *links*, which are critical pieces of information not found in the former.

Tags define the structure of a web page and can also be used to pin-point the semantics of its elements: it can thus provide precise descriptors to web applications and search engines. While opportunities created by tag mining have yet to be fully exploited, this situation can be reasonably expected to change once the Semantic Web progresses toward actual realization [30]. On the other hand, link mining has already made a dramatic impact through link analysis and Google.

Link Analysis

Link analysis is the activity of mining the web to determine its structure [12]. The model is a graph where nodes represent pages and arcs represent links. More often than not, links to a page are expressions of interest and act as approval for the page; therefore the number of links pointing to a page is normally a measure of its quality and usefulness. Thus, for a particular topic, an *authoritative page* is one which is pointed to by many other pages while a *hub* is a page that provides many links to such authorities. In scientific literature, for instance, survey papers play the role of hubs, by referring to authorities who introduced or best describe particular topics and techniques.

A major development in link analysis came with the introduction of the PageRank algorithm, and the extraordinary success of Google that used it to prioritize the relevant pages. Indeed, given the enormity of the web, keyword searches tend to return large number of URLs, whereas users tend to explore only a limited number of such URLs—typically those displayed in the first page (or first few pages) of their web browser. Thus, it is essential that the top URLs returned by the search engine are those that are approved by the largest number of most reliable users. Given that approval of a page is denoted by linking to it, in the PageRank algorithm the rank of a page is determined by (i) the number of pages that point to it and (ii) their ranks. This recursive definition leads to equations that are akin to those of Markov models where links to a webpage can be seen as transition probabilities. A number of efficient numeric and symbolic techniques are at hand to solve these equations and determine the PageRank of each page discovered by the web crawler.

⁴In fact similar distances and techniques are often used to cluster documents in libraries.

User Modeling

Link analysis is also used to discover accounts, companies, and individuals in a network of terrorists or other criminal organizations. Link analysis can find hidden links between the accounts and find a cluster of such accounts that are interrelated. These links include deposits to/withdrawals from the account, wire transfers, shared account owners, etc. Once such groups are identified, additional information about the group plans and goals can be obtained. Similarly, link analysis can also discover non-criminal *social networks*, by identifying highly connected subgraphs in the web. We note that individuals/companies participating in social or terror networks are normally connected in many different ways in addition to links in their web-pages. For instance, these entities may work together, share emails or phone calls or text messages, participate in same recreational activities, etc. Much of this information might not be available in electronic form, or could be protected by privacy. Even with these restrictions the discovery of such networks can be invaluable for homeland security, investigation of criminal activities, and viral marketing.

Viral Marketing is based on the idea that recommendation (about movies, products, etc.) spread through word-of-mouth, email and other informal contacts (i.e., in ways similar to those in which viral infections spread through the population). Thus, a very cost-effective strategy for marketing is to focus on a small subset of well-connected people in the network and turn them into enthusiastic early adopters, who will then propagate the message to the social network. For terrorist cell detection, user modeling takes the form of assigning roles to the entities involved in the cell. This identifies financial sources of the cell and influential entities in the cell. This labelling is based on many different attributes such as node degree, closeness, connectedness, and the degree of separation.

In traditional targeted marketing applications, potential customers are first partitioned into segments through clustering algorithms that group them according to their age, sex, income, education, ethnicity, and interests. Then the responses from samples collected from these segments are used to decide on a marketing strategy. However in many cases, this approach cannot be used on the web, since most users prefer to order goods and services without revealing personal information. A technique called *collaborative filtering* has been developed to address this problem. In this approach, the system makes recommendations during live customer transaction. For instance, if a customer is buying a book, the system (i) identifies customers, called neighbors, who have interests similar to the customer involved in the transaction (e.g., similar purchase patterns or product ratings) and then (ii) recommends other related items frequently bought (or highly rated) by neighbors. Many shopping sites such as amazon.com and movie rental sites such as netflix employ this technique with great success. Collaborative filtering can also be used to analyze criminal records and predict the behavior of felons based on the behavior of 'neighbor' felons. Thus collaborative filtering has attracted much research and commercial interest [12].

5.3.2 Security Applications

Data Mining is now regarded as a key technology in security applications that use DM techniques of increasing complexity and sophistication. For instance, simple OLAP techniques have been used to associate outlier crimes committed by the same perpetrator [35]. The significance of the outliers can be explained by the following example [35]:

1. Of 100 robberies, in 5 the weapon-used attribute had value 'Japanese sword' while a gun was used in the other 95. Obviously, the japanese swords are of more discriminative significance than gun.
2. For the method-of-escape attribute we have 20 different values: 'by car', 'by foot', etc. Although both 'Japanese sword' and 'by car' both occurred in 5 crimes., they should not be treated equally.
3. The combination of 'Japanese sword' and 'by car' is more significant than 'Japanese sword' and/or 'by car' alone.

Thus, the basic assumption is that criminals keep their modus operandi and reoccurrences of uncommon traits in successive crimes suggest that they are probably due to the same perpetrator. An OLAP-based approach can be used to count the occurrences of crimes for different values of attributes such as weapon-used and escape-method, and for all possible combinations of these attributes. OLAP aggregates also support natural generalizations over attribute values, whereby e.g., city blocks can be generalized into larger quarters, and person heights might be represented by coarser granularities. As a preprocessing step, clustering was used to select the most significant attributes for analysis, and partition their values into discrete intervals. Finally, an outlier score function was used [34] to measure the extremeness level of outliers, and an association method based on these scores was employed to associate criminal incidents. The method was applied to the robbery incident dataset in Richmond, Virginia of 1998, with promising results [34].

Effective border control represents a top priority for homeland security [1]. Simple classifiers such as those discussed in the previous section represent an important DM technique that can be used in such application. More sophisticated DM techniques or combination of techniques can be used in more advanced applications. For instance, telecommunication companies have been early adopters of DM technology, and are now sophisticated users of this technology in diverse applications, including identification of network faults, overload detection, customer profiling, marketing, discovery of *subscription* fraud, and *superimposition* fraud. *Subscription* fraud is very common in the business world and is perpetrated when new subscribers use deception in applying for calling cards, telephone accounts, and other services for which they are not qualified, and/or have no intention to ever pay for. Mining techniques such as classification and clustering have been used with good success in these and other risk-management situations (e.g., approving loan applications).

Cellular cloning fraud is instead a classical example of superposition fraud. This fraud is perpetrated by programming into one or more clones the identity of a legitimate cellular phone (whose account is then charged with the calls made from the second one). More traditional techniques based on rules such as “no two calls can be made within a short time from two distant places”, have now been successfully enhanced with DM methods based on (i) building a typical behavior profile for each customer and (ii) detecting deviation from this typical behavior using techniques such as neural nets and classifiers. An *anomaly detection* approach has also proven effective in *intrusion detection* applications as discussed next.

The fast growth of the internet has been accompanied with a surge of activities to attack the security of the network and intrude into its nodes, via increasingly sophisticated tricks and approaches. As a result, intrusion detection tools have become indispensable for network administrators. Early systems based on signatures that describe known intrusion scenarios, have now evolved into data mining algorithms that first build a model of “normal” behavior and then detect anomalies using techniques such as classifiers, clustering, and outlier detection [26]. Due to the transient and dynamic nature of intrusion these tasks must be performed in a data stream mining environment, and will thus be discussed in more details in the next section.

A third and more elaborate kind of fraud is *collusive agent fraud* that is perpetrated when several agents collaborate in fraudulent or illegal activities. An example of this is the laundering of dirty money resulting from drug trafficking. Of more recent interest is the analysis of funding mechanisms for terrorism—e.g., the situation where charity money is used for weapons and terroristic training. The approach used aims at identifying suspicious and otherwise unusual electronic transactions while minimizing the number of ‘false positive’ returned by the system. Traditional methods based on simple rules written by experts often lack in precision and cannot cope with new money laundering schemes continuously devised by the criminals. Relational data mining methods for the discovery of rare patterns have been used to overcome these problems [37]. Link analysis can also play a major role in detecting large networks of colluding agents. For instance the following quote is taken from Wikipedia [4].

Data mining has been cited as the method by which the U.S. Army unit Able Danger supposedly had identified the September 11, 2001 attacks leader, Mohamed Atta, and three

other 9/11 hijackers as possible members of an al Qaeda cell operating in the U.S. more than a year before the attack ⁵

Able Danger (a classified military intelligence program, created in 1999, against transnational terrorism—specifically al-Qaeda) relied on link analysis techniques to associate publicly available information with classified information in an attempt to make associations between individual members of terrorist groups. Whether factual or overstated, Able Danger’s reported success in investigating social (or criminal) networks by combining secret information with public-domain information has added to the fear about the government breaching the privacy of individuals by mining their massive databases ⁶. An even more pressing concern is that malicious users and organizations could employ similar mining techniques and public-domain databases for nefarious purposes. Propelled by these concerns, privacy-preserving DM has emerged as a very active topic of current research.

5.3.3 Privacy-Preserving Data Mining

In this Information Age, a great deal of scientific, medical, and economic information is collected on every aspect of human activity. The benefits of this information can be maximized by sharing it among agencies and researchers. Unfortunately, this can also lead to misuses and the privacy-breach of the citizens. A vivid example of this problem was provided by Latanya Sweeney, who in 2002 [51], started from a large public domain database of medical diagnoses and procedures performed on people. The database had been sanitized to remove information, such as social security number, address, etc., which could be used to identify the actual individuals. However, the person’s birthdate, zip code and sex were still reported in the database. Sweeney bought the voter registration list for Cambridge, Massachusetts (56 K records), listing the birthdate, zip, and sex for each voter, and was thus able to derive the medical record of William Weld, who was state governor at the time. The problem here is clearly identifiable in the fact that birthdate, zip, and sex is a quasi-key: it uniquely identifies 87% of population in the USA. The concept of *K-Anonymity* was thus introduced to avoid the quasi-key problem. K-Anonymity requires that, at least, $K \gg 1$ records share the same values for any combination of attribute values. In order to make a database publishable, while avoiding the quasi-key problem, researchers are now seeking to provide techniques to introduce modifications in the database to achieve K-anonymity, while still preserving the statistics of the data sets to enable accurate mining. However, K-anonymity is only a necessary condition that must be satisfied when seeking privacy-preserving data mining. The problem of finding sufficient conditions is much harder inasmuch as it has been shown that the mere publication of mining results, such as association rules or decision trees, can compromise privacy in certain situations. This problem provides a fertile ground for much current research.

5.4 Mining Data Streams

The tremendous growth of the internet has created a situation where huge amount of information is being exchanged continuously in the form of data streams. A store-now and process-later approach is often impractical in this environment, either because the data rate is too massive, or because of applications’ real time (or quasi real-time) requirements. A new generation of information management systems, called Data Stream Management Systems(DSMSs) are thus being designed to support

⁵According to statements by Lt. Col. Anthony Shaffer and those of four others, Able Danger had identified the September 11, 2001 attack leader Mohamed Atta, and three of the 9/11 plot’s other 19 hijackers, as possible members of an al Qaeda cell linked to the ’93 World Trade Center bombing. This theory was heavily promoted by Representative Curt Weldon, vice chairman of the House Armed Services and House Homeland Security committees. In December 2006, an investigation by the US Senate Intelligence Committee concluded that those assertions were unfounded. . . . However, witness testimony from these hearings is not publicly available . . .

⁶Indeed in US, Able Danger is only one (and hardly the largest) of the many classified government programs that use data mining for homeland security applications [48].

advanced applications on continuous data streams. For instance, in a fraud detection system for web purchases, clicks have to be marked as spam or non-spam as soon as they arrive, so that the rest of the system does not get affected by the fraudulent clicks. We next discuss network monitoring and intrusion detection represent key application area for this technology

Intrusion Detection

Intrusion Detection The threat to homeland security posed by intrusions became obvious in the Spring of 2007 when cyberwar against Estonia broke out after the ‘Bronze Soldier’ statue was removed from the Russian World War II memorial in the center of Tallinn. This cyber-assault disabled most of the web services of government ministries, political parties, newspapers and banks in this Baltic nation, whereby Nato scrambled to the rescue by sending its top terrorism experts [52]. The crisis was triggered by hackers, who using malware⁷ penetrated masses of computer and launched synchronized bursts of requests to overwhelm computer servers and bring them to a halt. This Distributed Denial of Service attack illustrates that these ‘virtual’ cyber-attacks can jeopardize the infrastructure of whole countries. Therefore, Intrusion Detection Systems (IDS) are now widely used to protect computer networks from cyber attacks by monitoring audit data streams collected from network and systems for clues of malicious behavior. The misuse detection techniques of more traditional IDS detect intrusions by directly matching them against known patterns of attacks (called signatures or rules). These methods have low rate of false alarms, but they cannot protect the systems from new patterns of attack. Moreover, the process of learning and encoding signatures accurately is slow and laborious and prevents a fast response to attacks. This problem can be addressed effectively by using classifiers and other predictive methods that learn from instances of IP logs labeled as ‘normal’ or ‘intrusive’ [32].

More recently approaches based on anomaly detection have found interesting uses in IDS. These approaches focus on the modeling the regular patterns of normal user activities: then user activities that deviate from his/her normal behavior pattern, are regarded as a possible intrusion activities (by malware disguising as the user). While statistical techniques can detect anomalous user behavior with only limited accuracy, density-based clustering algorithm DBSCAN was proposed in [42] to cluster the salient parameter of past behavior for the user, as to enable sudden changes in such behavior to be detected as outliers in the data. However, gradual shifts in user behavior should be allowed without raising false alarms. To incorporate the new data without requiring re-computation from scratch, incremental clustering algorithms are used in [42], making the approach more suitable for real time computations on data streams.

Data Stream Management Systems. Since data intensive applications often involve both streaming data and stored data (e.g., to validate a requested credit card transaction against historical account data), the approach taken by most general purpose DSMS is that of generalizing database languages and their technology to support continuous queries on data streams. Research in this vibrant area has led to many new techniques for response time/memory optimization, synopses, and quality-of-service [55, 8, 31]. DSMSs have now moved beyond the research phase [6, 17, 21, 10] and are used in many important application areas, including publish/subscribe, traffic monitoring, sensor networks, and algorithmic trading. Unfortunately, DSMSs cannot yet support well mining functions and queries and research advances in two main areas are needed to correct this situation, by:

- enriching database query language with the power and extensibility to express generic data stream mining functions, and
- taming the computational complexity of mining queries by developing faster data stream mining algorithms and better synoptic primitives (e.g., windows and sampling) for complex mining tasks such as frequent itemset mining for association rules.

⁷A generic name for evil software—<http://en.wikipedia.org/wiki/Malware>

The goal of supporting mining queries in database management systems proved very difficult to achieve because of technical challenges—discussed next since they help clarifying the similar problems faced by DSMS on the first point above. Then we address the second item, and discuss a ‘fast & light’ mining algorithms for mining data streams.

5.4.1 Data Mining Systems

In the mid-90s, DBMS vendors were able to integrate support for OLAP and data warehousing functions into their systems via limited extensions to SQL. This extraordinary technical and commercial success was promptly followed by abject failure when the same vendors tried to integrate support for data mining methods into their SQL systems. Indeed performing data mining tasks using DBMS constructs and functions proved to be an exceedingly difficult problem even for DBMS supercharged with OR-extensions [47]. Efforts to solve this difficult problem developed along two main paths, that were described as the high-road approach and the low-road approach in [29].

In their visionary paper, Imielinski and Mannila [29], called for a quantum leap in the functionality and usability of DBMSs, to assure that mining queries can be formulated with the same ease of use as current queries in relational DBMSs. The notion of Inductive DBMS was thus born, which inspired much research from the fields of knowledge discovery and databases [58], including a number of proposals for new data mining query languages such as MSQL [28], DMQL [25], and Mine Rule [39]. These proposals made interesting research contributions, but suffered limitations in terms of generality and performance.

DBMS vendors entering the DM market instead opted a rather low-road approach based on cache-mining. In this approach, the data of data of interest is first moved from the database into a file or main memory, where the cached data is then mined using the special DM functions from the vendor-provided library. Various enhancements, including graphical user interfaces and integration tools with the DBMS, have also been. For instance, *IBM DB2 intelligent miner* [40] provides a predefined set of mining functions, such as decision tree classification, neural network based classification, clustering, sequential pattern mining algorithm, linear regression, and other mining functions, along with selected preprocessing functions. These functions are supported by combination Java Stored Procedures and Virtual Mining Views that enhance the usability of the system, however other limitations remain. In particular, the system incurs data transfer overhead and does not allow modification or introduction of new mining algorithms (at least not without significant effort). *Oracle Data Miner* supports similar functionality and closer integration with the DBMS, via PL/SQL [5]. Furthermore, it supports more complex data mining functions, such as Support Vector Machines. We next discuss in more detail the Microsoft SQL Server that, among database vendors, claims to achieve the closest integration and best inter-operability of the mining functions with the DBMS [54].

OLE DB for DM. Microsoft SQL Server supports OLE DB for DM⁸, which views mining models as first class objects, that can be created, viewed, modified etc., just like other database tables. OLE DB for DM maintains schema rowsets, which are special system tables that store meta-data for all defined mining objects. These schema rowsets allow easy management and querying of mining objects. For instance, a “Create Model” construct, which is similar to the “Create Table” construct of SQL, can be used to create, say classifiers. Then, “insert into model” constructs can be used to insert new examples into the model, thus training the classifier just created. Finally, predicting class of testing tuples requires a special join, called prediction join, which takes a trained data mining model and a database table with testing data to classify the tuples in the table. Therefore, SQL is further extended to allow “prediction join” operator between a mining model and a database table. With these special constructs, OLE DB for DM achieves a closer integration of mining with SQL and the DBMS.

⁸Stands for Object Linking and Embedding Data Base for Data Mining.

To compensate for the fact that their proprietary DM extensions lack in flexibility and user-extensibility, DSMS vendors also allow inspecting and importing/exporting descriptive models through the XML-based representation called PMML (Predictive Model Markup Language). PMML is a markup language proposed to represent statistical and data mining information [19]. Even so, DBMS-based mining systems have only gained limited popularity with respect to the many existing stand-alone DM packages, and analytics software packages such STATISTICA [50], MATHEMATICA [38], SAS [3], and R [44] (open source) that are now supporting a rich set of DM methods. Among the several stand-alone systems at hand, we next describe the WEKA system [18], from the university of Waikato, which is very popular among data mining researchers.

WEKA. The Waikato Environment for Knowledge Analysis is a general purpose tool for machine learning and data mining, implemented in Java. It supports many algorithms for different data mining needs such as, data cleaning, visualization, filtering, classification, clustering, association rule mining, etc. Furthermore, these algorithms are supported *generically*, i.e., independently from the schema used by the tables containing the data. This is achieved via a configuration file, which describes the data to DM functions. The configuration file lists the attributes of the table, their types, and other meta information. Therefore, the main advantages of WEKA are as follows:

- All algorithms follow standard Java interface for extensibility,
- A large set of mining algorithms for classification, clustering, and frequent itemsets mining are also available, and
- A comprehensive set of data pre-processing and visualization tools are provided.

5.4.2 DSMSs and Online Mining

Much research interest has recently emerged on the problem of mining data streams to support a variety of applications including click stream analysis, surveillance, and fraud detection. These applications require the ability to process huge volumes of continuously arriving data streams and to deliver the results in a timely fashion even in the presence of bursty data. Thus we are faced with twin challenges of providing (i) mining algorithms that are fast and light enough to process bursty streams in a timely fashion, and (ii) a data stream mining system with functionalities parallel to those provided by, say, WEKA or Microsoft OLE DB on stored data. To achieve this goal, the power of DSMS and their query language must be extended with the ability of expressing and supporting continuous mining queries efficiently. While requiring the extension to current DSMS discussed next, this approach of the primitives and technology currently provided by DSMS: these include techniques for buffering, sampling, load shedding, scheduling, windows, punctuation, approximate query answering, and many others⁹

5.4.3 Stream-Mill: an Inductive DSMS

Research on DSMSs has led to the development of many DSMS prototypes[23, 7, 22] and DSMSs that are now being used in the commercial world [11]. Most of these systems support some dialect of SQL as their continuous query language and thus inherit the limitations

of SQL in terms of their ability of supporting data mining applications. In fact, these limitations are also restricting DSMSs, such as Aurora/Borealis [21, 22], which provide attractive ‘boxes and arrows’ graphical interface improving the usability, but not the expressive power of the system.

Moreover, the expressive power of SQL and other query languages on data stream applications is further diminished by the fact that only non-blocking queries [31] can be supported on data streams

⁹On the other hand, data mining systems on data bases make less use DSMS primitives such as, say, transaction support and query optimization [21].

¹⁰. Furthermore, as shown in [31], non-blocking queries are exactly monotonic queries; but, if we disallow the use of non-monotonic operators, (such as EXCEPT), in SQL, we also lose the ability of expressing some of its monotonic queries. Thus, SQL is not complete w.r.t. non-blocking queries [31]. Fortunately, the following positive result was also proved in [31]: SQL becomes Turing-complete, and also complete w.r.t. non-blocking queries, once it is extended with user defined aggregates (UDAs). The Stream Mill System developed at UCLA [10] builds on this important result by supporting the Expressive Stream Language (ESL) [10], which extends SQL with the ability of defining UDAs. We now briefly discuss the constructs used in ESL to efficiently support window-based synopses on UDAs.

User Defined Aggregates (UDAs) Example 1 defines a continuous UDA equivalent to continuous version of the standard AVG aggregate in SQL. The second line in Example 1 declares a local table, state, where the sum and the count of the values processed so far, are kept. Furthermore, while in this particular example state contains only one tuple, it is in fact a table that can be queried and updated using SQL statements and can contain any number of tuples. The INITIALIZE clause inserts the value taken from the input stream and sets the count to 1. The ITERATE statement updates the tuple in state by adding the new input value to the sum and 1 to the count and returns the ratio between the sum and the count as a result.

Example 1 *Defining the Standard Aggregate Average*

```
AGGREGATE avg(Next Real) : Real
{
  TABLE state(tsum Real, cnt Int);
  INITIALIZE :
  {
    INSERT INTO state VALUES (Next, 1);
  }
  ITERATE :
  {
    UPDATE state SET tsum=tsum+Next, cnt=cnt+1;
    INSERT INTO RETURN SELECT tsum/cnt FROM state;
  }
}
```

In addition to the INITIALIZE and ITERATE state ESL also allows the use of a TERMINATE state to specify computation to be performed once the end of the input is detected. Thus, to implement the standard avg aggregate of SQL the line INSERT INTO RETURN SELECT tsum/cnt FROM is moved from the ITERATE state to the TERMINATE. This change produces a blocking aggregate that can be used on database tables, but not on data streams [31]. Instead UDAs without TERMINATE, such as that of Example 1, can be freely applied over data streams, since they are non-blocking. As shown in [57] data mining algorithms can be expressed through UDAs (typically containing 40 lines of code rather than the 6 lines above): in fact, as shown in Section 5.2 all DM algorithms can be naturally viewed as complex aggregates that compute complex statistics over a given data set. In ESL, user is not restricted to write aggregates using SQL: external functions coded in a procedural language can also be used, e.g., to speed up execution or take advantage of external libraries [10].

Windows Because of memory limitation, only synopses of past data can be kept, such as samples, histograms and a window of most recent arrivals. ESL provides powerful constructs for specifying synopses, such as windows, slides, tumbles on arbitrary UDAs, not just built-in ones [10]. These synoptic constructs are critical in data stream applications: For instance, in a traffic monitoring application, users might be interested in retrieving every minute the average traffic speed at each sensor over the

¹⁰Blocking queries are those where the result can only be returned after all input tuples have been processed. Whereas, non-blocking queries are those that can return results incrementally, as the data arrives.

last 10 minutes. Such queries require computing average over a moving window of 10 minutes with a slide of one minute [10]. To optimize the execution over these windows, ESL allows windowed UDAs to be customized with a special EXPIRE state that is invoked when a tuple expires out of a window. For instance, a windowed version of AVG aggregate is defined as in Example 2.

Example 2 *Windowed Average*

```
WINDOW AGGREGATE avg(Next Real) : Real
{
  TABLE state(tsum Real, cnt Int);
  INITIALIZE :
  {
    INSERT INTO state VALUES (Next, 1);
  }
  ITERATE :
  {
    UPDATE state SET tsum=tsum+Next, cnt=cnt+1;
    INSERT INTO RETURN SELECT tsum/cnt FROM state;
  }
  EXPIRE :
  {
    UPDATE state SET tsum=tsum-oldest().Next, cnt=cnt-1;
  }
}
```

Thus, in the EXPIRE state we perform customized delta maintenance for AVG, by decrementing the sum with the expiring value (oldest value in the window) and the count with 1, respectively. This delta-maintenance technique results in efficient implementation of many UDAs as discussed in [10]. Moreover, by combining the basic blocking version and the window version of a given UDA, the ESL compiler derives efficient implementations for logical (time based) windows, physical (count based) windows, and other complex synopses based on slides and tumbles [10]. Arbitrary windowed UDAs can then be invoked in ESL using the SQL:2003 OLAP aggregate syntax.

Genericity Unlike the simple aggregates such as AVG, complex mining aggregates are often applied to tables and streams comprised of an arbitrary number of columns. Therefore, like WEKA, mining algorithms implemented in the system should be generically applicable over different data streams. Given a data tuple, WEKA converts it to a homogeneous record that can have arbitrary number of columns — this technique is general is called verticalization, since a table tuple is verticalized in to a record. Thus, all the mining algorithms can operate over this generic record. This record is essentially a array of real values, where categorical values are stored as the sequence index of the attained value. For example, if an attribute can get one of three values, such as passenger, workvan, truck, then internally WEKA stores 1, 2, or 3, for passenger, workvan, or truck, respectively. Date and time columns is converted to real and the storage of integer and real values is trivial. Much in the same way as WEKA, data mining algorithms implemented as ESL aggregates, should also be generically applicable over various application domains.

5.4.4 Data Stream Mining Algorithms

Although the mining methods used on data streams are conceptually similar to those on databases, the actual algorithms must be significantly modified or replaced with new ones to minimize their computational costs and the number of passes required over the data. The state-of-the-art here has progressed differently for different mining tasks. For instance, while the prediction task is effectively supported using classifier ensembles, the problem of computing frequent itemsets for association rules on data streams is significantly more challenging and has been the focus of many research

efforts [14, 33, 13, 41]. Windows containing only the most recent data are critical for mining data streams since they provide better performance and adaptability in the presence of *concept shift*. Concept shift (concept drift) denotes an abrupt (gradual) change of the statistical and logical properties of the data, whereby, e.g., the old classifier stops being an accurate predictor and a new classifier is required. Ensembles of classifiers can provide effective solutions to the concept shift/drift problem and are discussed next.

Classifier Ensembles Continuous adaptation is possible for Bayesian classifiers that use count-based statistics, which can be updated upon the arrival or expiration of each tuple via ESL aggregates with the expire clause as discussed previously (delta-maintenance). However, for most mining algorithms, including decision-tree classifiers, delta maintenance is not possible. For those cases, we instead prefer to build a new model every N tuples. Thus we partition the incoming stream into windows (these are often called tumbling windows to distinguish from continuously sliding windows used, say, for Bayesian classifiers): we build a new classifier for each window and use an ensemble containing K such classifiers to achieve more accurate predictions via advanced techniques, such as *bagging* and *boosting* [56, 15]. Ensemble methods have long been used to improve the predictive accuracy of classifiers built from different samples of the same data set. However, for data streams, classifiers are continuously built from successive windows non-overlapping windows (which are often called tumbling windows). The records in the new window are used as training data to build a new classifier. Furthermore the same training data are also used to assess the accuracy of the old classifiers. Since older classifiers are continuously replaced with the ones built on a new window, a simple adaptation to concept drift is automatically achieved. Whenever a rapid concept shift occurs, this causes a sudden fall in the accuracy of several classifiers, whereby several (or all) the old classifiers are discharged and we restart by rebuilding a largely (or totally) new ensemble [56].

Using bagging, the decision on how to classify each new arriving tuple is taken by combining the votes of each classifier, either by straight majority or by weights based on their accuracy. Therefore with bagging, the errors of classifiers can impact the weight assigned to their votes—thus impacting the decision phase. However, boosting affects the training phase, since each new arriving training tuple is assigned a weight that increases with the classification errors incurred by the previous classifiers. In other words, the new classifier is trained to compensate for the errors of its predecessors. Boosting ensembles can learn fast and produce accurate answers using weak learners, such as shallow decision tree-classifiers that are constructed in a fast & light computations [15]. However, boosting methods are known to be very sensitive to noise.

Stream Mill allows ensemble methods, based on either bagging and boosting, to be applied over arbitrary classifiers implemented as UDAs.

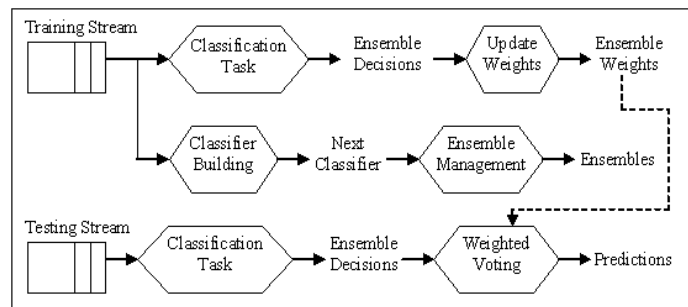


Figure 5.4: Generalized Weighted Bagging

For instance, Figure 5.4, shows the generic support for weighted bagging in Stream Mill. The training stream is first fed to a UDA, named ‘Classifier Building’, which learns the next classifier model to be stored with other models. The training stream is also sent to a ‘Classification’ UDA that predicts

the class for each tuple using each of the existing classifiers in the ensemble. These predictions are then used to assign weights to the classifiers (based on accuracy) for the next window of testing tuples. The newly arriving testing tuples are first classified using each of the classifiers from the ensemble. Then, a weighted voting scheme is employed to determine the final classification of the tuples. The general flow of data tuples, both training and testing, does not depend on the particular classification algorithm. In fact, only the boxes labelled ‘classifier building’ and ‘classification task’ are dependent on the particular classifier. Thus, any classification algorithm that provides implementation for these two ‘boxes’ can be used as a base classifier for the weighted bagging approach.

Other Mining Tasks

Other mining tasks in addition to classification are also supported by the Stream Mill system. For instance, online frequent itemsets mining algorithms are also supported in Stream Mill system. Research over online frequent itemsets mining has led to many algorithms, for instance Moment [14], CanTree [33], SWIM [41]. However, these algorithms have performance problems when dealing with large windows. Therefore, Stream Mill uses the IMLW algorithm to check and discover efficiently frequent itemsets over large sliding windows [41]. Mining of large sliding windows leads to more reliable results as compared with small sliding windows or large tumbling windows, since both of the latter cases suffer from lack of statistical support. Furthermore, IMLW can handle large itemsets and can also maintain frequency of any interesting patterns, not just the ones that are frequent ones. Stream Mill fully supports this algorithm, by allowing user specify confidence and support thresholds. Furthermore, user can also specify a set of itemsets that should be always maintained.

For clustering, Stream Mill also supports stream oriented version of K-means and DBScan [36, 20] both tumbling and continuous window setting. Unlike K-means that tend to produce spherical clusters around centroids, DBScan is a density-based method that can discover clusters of arbitrary shape: as such it is quite useful in detecting the spreading of viral diseases and other trajectories. Both algorithms can be expressed quite naturally using ESL UDAs.

Sequence queries represent a very useful tool for identifying sequential patterns in time-series and data streams. As demand for sequence pattern queries has grown in both database and data stream applications, SQL standard extensions are being proposed by collaborating DBMS and DSMS companies [59]. These extensions are based on allowing Kleene-closure expressions in the SQL query. These constructs, and their supporting query optimization techniques, were first introduced in SQL-TS [46], which is currently supported in Stream Mill [9].

5.5 Conclusion

The explosive growth of data that is available in the modern information age has created a fertile application ground for data mining (DM) technology—i.e., methods and techniques for knowledge discovery from databases. The DM technology is powerful and it has been successfully used on a broad spectrum of advanced applications from business, science, and security. However, DM remains a complex interdisciplinary field that combines techniques from different areas and requires significant application domain expertise for a successful deployment in real-world applications. In this chapter, we have provided an up-to-date introduction to DM, designed for researchers and technical experts coming from other fields. Thus, we have first discussed the more established DM methods and core techniques, including classification, association, and clustering. Then, turning our attention to the latest trends and applications, we discussed web mining and data stream mining.

Bibliography

- [1] Richard Skinner. Automated Commercial Environment http://www.dhs.gov/xoig/assets/mgmttrpts/oig_06-56_aug06.pdf December 2007.
- [2] Ross Quinlan. C4.5 decision tree classifier. <http://www.rulequest.com/personal/c4.5r8.tar.gz>. June 2005.
- [3] SAS: Statistical Analysis Software. <http://www.sas.com/technologies/analytics/statistics/stat/>. August 2005.
- [4] Wikipedia on Data Mining. http://en.wikipedia.org/wiki/data_mining. August 2006.
- [5] ORACLE. Oracle Data Miner Release 10gr2. <http://www.oracle.com/technology/products/bi/odm>, March 2007.
- [6] A. Arasu, S. Babu, and J. Widom. Cql: A language for continuous queries over streams and relations. In *DBPL*, pages 1–19, 2003.
- [7] Arvind Arasu and Jennifer Widom. Resource sharing in continuous sliding-window aggregates. In *VLDB*, pages 336–347, 2004.
- [8] B. Babcock, S. Babu, M. Datar, R. Motawani, and J. Widom. Models and issues in data stream systems. In *PODS*, 2002.
- [9] Yijian Bai, Chang Luo, Hetal Thakkar, and Carlo Zaniolo. *Stream Data Management*. Kluwer, 2004.
- [10] Yijian Bai, Hetal Thakkar, Chang Luo, Haixun Wang, and Carlo Zaniolo. A data stream language and system designed for power and extensibility. In *CIKM*, pages 337–346, 2006.
- [11] Stream Base. <http://www.streambase.com/>, September 2007.
- [12] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kaufman, 2002.
- [13] W. Cheung and O. R. Zaiane. Incremental mining of frequent patterns without candidate generation or support. In *DEAS*, 2003.
- [14] Y. Chi, H. Wang, P. S. Yu, and R. R. Muntz. Moment: Maintaining closed frequent itemsets over a stream sliding window. In *Proceedings of the 2004 IEEE International Conference on Data Mining (ICDM'04)*, November 2004.
- [15] F. Chu and C. Zaniolo. Fast and light boosting for adaptive mining of data streams. In *PAKDD*, volume 3056, 2004.
- [16] Clementine. <http://www.spss.com/clementine>. August 2005.
- [17] C. Cranor, T. Johnson, O. Spatscheck, V. Shkapenyuk, and O. Spatscheck. Gigascope: A stream database for network applications. In *SIGMOD*, pages 647–651. ACM Press, 2003.
- [18] Weka 3: data mining with open source machine learning software in java. <http://www.cs.waikato.ac.nz>. August 2005.
- [19] Data Mining Group (DMG). Predictive model markup language (pmml). <http://sourceforge.net/projects/pmml>. August 2005.

- [20] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.
- [21] D. Abadi et al. Aurora: A new model and architecture for data stream management. *VLDB Journal*, 12(2):120–139, 2003.
- [22] D. Abadi et al. The design of the borealis stream processing engine. *CIDR*, 12(2):120–139, 2005.
- [23] Sirish Chandrasekaran et al. Telegraphcq: Continuous dataflow processing for an uncertain world. In *CIDR*, 2003.
- [24] Jing Gao and Pang-Ning Tan. Converting output scores from outlier detection algorithms into probability estimates. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 212–221. IEEE Computer Society, 2006.
- [25] J. Han, Y. Fu, W. Wang, K. Koperski, and O. R. Zaiane. DMQL: A data mining query language for relational databases. In *Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD)*, pages 27–33, Montreal, Canada, June 1996.
- [26] J. Han and M. Kamber. *Data Mining, Concepts and Techniques*. Morgan Kaufman, 2001.
- [27] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, 2000.
- [28] T. Imielinski and A. Virmani. MSQL: a query language for database mining. *Data Mining and Knowledge Discovery*, 3:373–408, 1999.
- [29] Tomasz Imielinski and Heikki Mannila. A database perspective on knowledge discovery. *Commun. ACM*, 39(11):58–64, 1996.
- [30] Lalana Kagal, Tim Berners-Lee, Dan Connolly, and Daniel J. Weitzner. Using semantic web technologies for policy management on the web. In *AAAI*, 2006.
- [31] Yan-Nei Law, Haixun Wang, and Carlo Zaniolo. Data models and query language for data streams. In *VLDB*, pages 492–503, 2004.
- [32] Wenke Lee and Salvatore J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Trans. Inf. Syst. Secur.*, 3(4):227–261, 2000.
- [33] C.K.-S. Leung, Q.I. Khan, and T. Hoque. Cantree: A tree structure for efficient incremental mining of frequent patterns. In *ICDM*, 2005.
- [34] Song Lin and Donald E. Brown. Criminal incident data association using the olap technology. In Hsinchun Chen et al. (eds.), editor, *Intelligence and Security Informatics*, pages 13–26. Springer, 2003.
- [35] Song Lin and Donald E. Brown. An outlier-based data association method for linking criminal incidents. In *Proceedings of Third SIAM Int. Conference on Data Mining, San Francisco, CA, USA, May 1-3, 2003*, 2003.
- [36] C. Luo, H. Thakkar, H. Wang, and C. Zaniolo. A native extension of sql for mining data streams. In *SIGMOD*, pages 873–875, 2005.
- [37] Oded Maimon and Lior Rokach, editors. *Chapter 57 of The Data Mining and Knowledge Discovery Handbook*. Springer, 2005.
- [38] Mathematica. <http://www.wolfram.com/products/mathematica/index.html>. December 2007.
- [39] R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. In *VLDB*, pages 122–133, Bombay, India, 1996.
- [40] IBM. DB2 Intelligent Miner. <http://www-306.ibm.com/software/data/iminer>. August 2005.
- [41] Barzan Mozafari, Hetal Thakkar, and Carlo Zaniolo. Verifying and mining frequent patterns from large windows over data streams. In *International Conference on Data Engineering (ICDE)*, 2008.

- [42] Sang-Hyun Oh and Won-Suk Lee. Anomaly intrusion detection based on dynamic cluster updating. In Zhi-Hua Zhou, Hang Li, and Qiang Yang (eds.), editors, *Advances in Knowledge Discovery and Data Mining, 11th Pacific-Asia Conference: PAKDD 2007*, pages 737–744. Springer, 2007.
- [43] Gregory Piatetsky-Shapiro and William J. Frawley, editors. *Knowledge Discovery in Databases*. AAAI/MIT Press, 1991.
- [44] The R Project. <http://www.r-project.org/>, December 2007.
- [45] C. R. Rao, E. J. Wegman, and J. L. Solka. *Handbook of Statistics, Volume 24: Data Mining and Data Visualization (Handbook of Statistics)*. North-Holland Publishing Co., 2005.
- [46] Reza Sadri, Carlo Zaniolo, Amir Zarkesh, and Jafar Adibi. Optimization of sequence queries in database systems. In *PODS*, Santa Barbara, CA, May 2001.
- [47] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. In *SIGMOD*, 1998.
- [48] Jeffrey Seifert. Data mining and homeland security: An overview. Technical report, Congressional Research Service: Resources, Science, and Industry Division, 2007.
- [49] Dennis Shasha and Yunyue Zhu. *High Performance Discovery In Time Series: Techniques And Case Studies (Monographs in Computer Science)*. SpringerVerlag, 2004.
- [50] Statistica. <http://www.statsoft.com/products/products.htm>. December 2007.
- [51] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. In *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 2002.
- [52] Mark Sweney. Is Eastern Europe’s cyberwar the shape of things to come? <http://blogs.guardian.co.uk/news/2007/05/eastern.europes.html>, May 17, 2007.
- [53] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., 2005.
- [54] Z. Tang, J. Maclennan, and P. Kim. Building data mining solutions with OLE DB for DM and XML analysis. *SIGMOD Record*, 34(2):80–85, 2005.
- [55] N. Tatbul, U. Setintemel, S. Zdonik, M. Cherniack, and M. Stonebraker. Load shedding in a data stream manager. In *VLDB*, 2003.
- [56] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *SIGKDD*, 2003.
- [57] Haixun Wang and Carlo Zaniolo. ATLaS: A native extension of sql for data mining. In *SIAM International Conference on Data Mining (SDM)*, San Francisco, CA, 5 2003.
- [58] Carlo Zaniolo. Mining databases and data streams with query languages and rules. keynote paper. In *KDID 2005: Knowledge Discovery in Inductive Databases, 4th International Workshop*, volume 3933 of *Lecture Notes in Computer Science*, pages 24–37. Springer, 2006.
- [59] Fred Zemke, Andrew Witkowski, Mitch Cherniak, and Latha Colby. Pattern matching in sequences of rows. Technical report, Oracle and IBM, 2007.
- [60] Jan M. Żytkow and Willi Klösen. Interdisciplinary contributions to knowledge discovery. In Willi Klösen and Jan M. Żytkow, editors, *Handbook of Data Mining and Knowledge Discovery*, pages 22–32. Oxford University Press, 2002.