# 4

# Evolution and Scaling of Data Mining Algorithms

***Objectives:***

- Data mining is an application-driven field where research questions tend to be motivated by real-world data sets.
- While many data mining applications focus on commercial applications, such as credit scoring, fraud detection, and Web personalization, data mining as a tool for scientific discovery also motivates research interest.
- The data mining innovations being implemented worldwide often involve collaborations among domain experts, computer scientists, and statisticians.
- These application-driven developments will continue to proliferate as data owners seek new and better ways to gain insight into their data.
- The research effort on scaling data mining algorithms to large databases has now given analysts the ability to model and discover valid, interesting patterns over these large data sets.
- General scaling principles include use of summary statistics, data compression, pruning the search space, and incremental computation.
- We described these principles in the practical context of mining algorithms, expecting they would be useful in other areas of computer science as well.

**Abstract.** Traditional modeling methods from statistics and machine learning, including linear regression, logistic regression, discriminate analysis, and naïve Bayes models, are often the first tools used to model multivariate data. Newer predictive models, including additive regression, decision trees, neural networks, support vector machines, and Bayesian networks, have attracted attention in data mining research and applications, as modern computing power has allowed data miners to explore more complex models. These predictive models often sacrifice interpretability for increased flexibility in the functional forms they accommodate. The trade-off between flexibility and the interpretability often drives the choice of method applied to particular multivariate data set.

Recent research has shown that combining different models can be effective in reducing the instability that results from predictions using a single model fit to a

S. Sumathi and S.N. Sivanandam: *Evolution and Scaling of Data Mining Algorithms*, Studies in Computational Intelligence (SCI) **29**, 151–164 (2006)
www.springerlink.com

single set of data. A variety of model-combining techniques (with exotic names like bagging, boosting, and stacking) combine massive computational search methods with variance-reduction ideas from statistics; the result is relatively powerful automated schemes for building multivariate predictive models. As the data miner's multivariate toolbox expands, a significant part of the art of data mining is the practical intuition of the tools themselves. Also in this section scaling the mining algorithms to large databases is discussed.

## 4.1 Data-Driven Evolution of Data Mining Algorithms

Data mining is an application-driven field where research questions tend to be motivated by real-world data sets. In this context, a broad spectrum of formalisms and techniques has been proposed by researchers in a large number of applications. Organizing them is inherently rather difficult; that is why we highlight the central role played by the various types of data motivating the current research.

We begin with what is perhaps the best-known data type in traditional data analysis, namely, $d$-dimensional vectors x of measurements on N objects or individual, or N objects where for each of which we have $d$ measurements or attributes. Such data is often referred to as multivariate data and can be thought of as an N $\times$ d data matrix. Classical problems in data analysis involving multivariate data include classification (learning a functional mapping from a vector x to $y$ where $y$ is a categorical, or scalar, target variable of interest), regression (same as classification, except $y$, which takes real values), clustering (learning a function that maps x into a set of categories, where the categories are unknown a priori), and density estimation (estimating the probability density function, or PDF, for $x, p\ (x)$).

The dimensionality $d$ of the vectors x plays a significant role in multivariate modeling. In problems like text classification and clustering of gene expression data, $d$ can be as large as $10^3$ or $10^4$ dimensions. Density estimation theory shows that the amount of data needed to reliably to estimate a density function scales exponentially in $d$ (the so-called "curse of dimensionality"). Fortunately, many predictive problems including classification and regression, do not need a full $d$ dimensional estimate of the PDF $p(x)$, relying instead on the simpler problem of determining of a conditional probability density function $p(y|x)$, where $y$ is the variable whose value the data miner wants to predict.

Recent research has shown that combining different models can be effective in reducing the instability that results from predictions using a single model fit to a single set of data. A variety of model-combining techniques (with exotic names like bagging, boosting, and stacking) combine massive computational search methods with variance-reduction ideas from statistics; the result is relatively powerful automated schemes for building multivariate predictive models. As the data miner's multivariate toolbox expands, a significant part of the art of data mining is the practical intuition of the tools themselves.

### 4.1.1 Transaction Data

A common form of data in data mining in many business contexts is record of individuals conducting "transactions," examples include consumers purchasing groceries in a store (each record describes a market basket) and individuals surfing a Web site (each record describes the pages requested during a particular session). Employing the multivariate viewpoint, we can conceptually view this data as a very sparse $N \times d$ matrix of counts, where each of the N rows corresponds to an individual basket or session, each of the $d$ columns corresponds to a particular item, and entry (i, j) is 1 if item j was purchased or requested as part of session i and is 0 otherwise.

Both N and $d$ can be very large in practice. For example a large retail chain or e-commerce Web site might record on the order of $\mathrm{N} = 10^6$ baskets per week and have $\mathrm{d} = 10^5$ different items in its stores available for purchase or downloading. These numbers pose significant challenges from both the point of view of being computationally tractable and being amenable to traditional statistical modeling. For example, in a store with $10^5$ different items and $10^6$ baskets per week simply computing a pairwise correlation matrix requires $\mathrm{O}(\mathrm{N}d^2)$ time and O $(d^2)$ memory, resulting in numbers of $10^{16}$ for time and $10^{10}$ for memory.

However, data miners routinely take advantage of the fact that transaction data is typically sparse; for example, since the average grocery basket might contain only 10 items, having only a few items in a basket means that only 10/50,000 or 0.02%, of the entries in the $\mathrm{N} \times d$ transaction matrix are nonzero. A substantial body of work in data mining research focuses on the idea of using subsets of items represented in each market basket, the so-called item sets $I$ as "information nuggets" in large high-dimensional transaction data sets; an example of an item set is the combination of products *bread, wine, and cheese* in baskets in a grocery store. Several variants of efficient algorithms are available to find all frequent item sets from a sparse set of transaction data. Frequent items are item sets $I$ such that $f_1 > T$, where the frequency $f_1$ is the number of rows in which all the items in $I$ were purchased and T is preselected count threshold, such that $\mathrm{T} = 0.001 \times \mathrm{N}$.

Another strand of research takes a more statistical view of market basket data as a density estimation problem rather than a search problem. A methodology for finding statistically significant item sets, i.e., item sets $I$ whose empirical frequency varies significantly from the frequency expected by a base line model (see illustrative visualizations at www.ics.uci.edu/∼smyth/cacm02/). Determining statistical significance in this contest is a subtle problem. A Bayesian approach can uncover complex multi-item associations ignored by more traditional hypothesis testing techniques. It has been used by the U.S. Food and Drug Administration to search large postmarket surveillance databases for significant but relatively rare adverse reactions, – a good example of the marriage of computationally oriented data mining ideas with more traditional inferential theories from statistics. Increasingly, much of the research

work in data mining occurs at this interface of computational and inferential approaches.

Frequent item sets can also be viewed as constraining on the set of all possible high-order probability models for the data. The technique of maximum entropy estimation provides theoretical framework for estimating joint and conditional probability distribution from the frequent item sets that can then be used for forecasting and answering queries. Unfortunately the maximum entropy approach scales exponentially the number of variables as to model (in both time and memory), limiting the technique in practice to relatively short queries or low-dimensional models.

Viewing transaction data as a sparse $N \times d$ matrix is a gross oversimplification of the true structure of the data in most applications. Typically, real transaction data has significant additional structure at various levels of detail; for example retail items are usually arranged in product hierarchies, and Web pages can be related to each other (through hyper links) or can be instances of a more general database scheme. Thus the columns of a data matrix such as products and Web pages can themselves have attributes (such as price and content) as well as implicit inter item relationships. Similarly the rows in a transaction data set can also have significant structure manifested by hourly, weakly, and seasonal temporal patterns. While some of these techniques explicitly exploit this structure, many open research challenges remain. Clear, however, is that techniques exploiting special structure in the data are likely to produce much more valuable insights and predictions than techniques that choose to ignore this structure.

### 4.1.2 Data Streams

The term *data stream* pertains to data arriving over time in a nearly continuous fashion. In such applications, the data is often available for mining only once, as it flows by. Some transaction data can be viewed this way, such as Web logs that continue to grow as browsing activities occur over time. In many of these applications, the data miner's interest often centers on the evaluation of user activity; instead of focusing on the relationships of items (columns), the data miner focuses on modeling individuals or objectives (rows).

Data streams have prompted several challenging research problems, including how to compute aggregate counts and summary statistics from such data. A related problem is that of incremental learning, whereby a global model assumed for the data stream, and the model is estimated incrementally as data arrives. A good example of the approach for online adaptation of classification tree models uses analytical probabilistic bounds to guide the degree to which the model needs to be updated over time.

Another aspect of data stream research involves scaling traditional ideas in statistical data analysis to massive, heterogeneous, nonstationary environments. Using large streams of call-record data in, for example, the telecommunications industry, statistical models (called signatures) can be built for

individual telephone customers. Note that the collection of customers' signatures resulting from this methodology can be viewed in a database context as a statistical view of the underlying transaction data. Thus the derived data can help provide approximate (statistical sense) answers to queries. Numerous applications of these techniques tackle problems in forecasting, fraud detection, personalization, and change detection.

### 4.1.3 Graph and Text-Based data

The possibility of discovering patterns in large graphs also motivates data mining interest. We can think of representing relationships among objects. Such "data graphs" appear in multiple settings; for example, the Web can be viewed as a graph where nodes are pages and hypertext links are edges. Similarly, user browsing can be viewed as a bipartite graph where nodes are either users or Web pages, and the edges or pages users have visited. An inevitable question arising from a graphical view of the Web is: What kind of structure can be automatically discovered from its topology? Research suggests, for example, the graph structure underlying the Web is distinctly nonrandom and possesses many interesting proprieties.

Graphs can be represented by an adjacency matrix conveying the nodes as row/column labels and edges as cell entries. Such matrices are indeed large, and fortunately, sparse. That is all nodes in a real graph are not created equal; some have an extremely high degree, outgoing or incoming edges, while the vast majority barely have degree 1. If the nodes are sorted according their degree, the result is often "laws" of the form.

*Degree* $\alpha$ $1/\text{rank}^a$, where $a$ is often termed as the "degree" exponent.

The matrix representation of a graph suggests that many classical methods in linear algebra are likely to be extremely useful for analyzing the properties of graphs. Indeed, the singular value decomposition is the engine behind many powerful tools, including latent semantic indexing, the hubs and authorities algorithm, and Google's PageRank algorithm. Reflecting what can be discovered from connectivity information alone, page rank uses a recursive system of equations defining the importance of each in terms of the importance of the pages pointing to it. The importance (or page rank) of each page can then be determined by solving this set of linear equations. Once again, sparseness is important from a computational point of view. Since the number of out links per page is on average extremely low relative to the total number of pages on the Web, this system of linear equations is sparse, and an iterative algorithm typically converges on a solution rather quickly.

Hyperlink connectivity represents only one type of Web data. The navigation patterns of Web surfers, obtained from Web logs, also represent opportunities for prediction, clustering, personalization, and related techniques, often referred as to as "Web mining."

Web content, including text documents, is another vast and ready available data source for data mining. Considerable progress in text classification

and clustering has been made by representing text as *term vectors* (a vector where component $j$ is 1 if the document contains term j and 0 otherwise). Nevertheless, modeling documents at a richer semantic level is clearly worthwhile for, say, trying to identify the relations among sets of objects such as documents.

### 4.1.4 Scientific Data

While many data mining applications focus on commercial applications, such as credit scoring, fraud detection, and Web personalization, data mining as a tool for scientific discovery also motivates research interest. For example, data in the form of DNA and protein sequences, microarray-based gene expression measurements, and biological images have revolutionized the fields of biology and medicine. Biologists often spend more time looking at data than through a microscope. Since much biological research is data-rich and relatively theory-poor, data mining research promises significant opportunities for assisting biologists pursuing new scientific discoveries. Rather than viewing the field of computational biology as just applications, data miners find themselves confronted with interesting and fundamental research challenges from a number of perspectives, including modeling, interference, and algorithmic. For example, the discovery of "motifs" in DNA sequences is an example of a biologically motivated data mining problem. Motif discovery can involve prior knowledge as to the number of motifs (such as one per sequence) and their exact or expected lengths. However, little knowledge is typically available as to where the motifs occur in each sequence or what symbols they contain. Related research is driven by development of both score functions for patterns (to be interesting, pattern must differ from the back ground in a systematic way) and efficient search techniques to locate the likely candidates from the combinationally large space of possible patterns in a set of sequences. Ideas from a systematic search, heuristic search, and stochastic search have all proved useful in this context. Several publicly available algorithms are used in computational biology for motif discovery, each combining basic statistical models with massive search capabilities.

Scientists in other disciplines also have an increased awareness of the importance of data mining; for example, in astronomy, the Sloan Digital Sky Survey generates 5TB of data annually, leading to significant data engineering challenges (see www.sdss.org). An important research topic concerning such data is how to develop efficient algorithms to perform common data analysis tasks, including clustering and density estimations, on massive data sets. Multiresolution *kd*-trees, or a flexible data structure for indexing data in multiple dimensions, can provide orders of magnitude speed-ups in the density estimations of astronomical data using mixture models.

One research area conspicuous by its absence in data mining research though tremendously important in practically any scientific context, is human operator interaction for discovery; for example how can the algorithm

designer and the scientist represent prior knowledge so the data mining algorithm does not just rediscover what is already known? And how can scientists "get inside" and "steer" the direction of a data mining algorithm? While some research on this topic has been pursued in a number of areas, including artificial intelligence and statistics, it has had relatively little effect on data mining in general.

## 4.2 Scaling Mining Algorithms to Large DataBases

Data mining is increasingly recognized as a key to analyzing, digesting, and understanding the flood of digital data collected by business, government, and scientific applications. Achieving this goal requires the scaling of mining algorithms to very large databases. Many classic mining algorithms require multiple database scans and/or random access to database records. Research today focuses on overcoming limitations imposed when it is costly or impossible to scan large databases multiple times or access records at random, while developing innovative algorithms and data structures to speed computation.

### 4.2.1 Prediction Methods

Predictive modeling is often a high-level goal of data mining in practice. After outlining the predictive modeling problem, we focus on two classes of algorithm: decision tree methods and support vector machines. Input into predictive modeling algorithms is a data set of training records. The goal is building a model that predicts a designated attribute value from the values of the other attributes (see Fig. 4.1). Many predictive models have been proposed in the literature, including neural networks and Bayesian methods.

*Decision Tree Construction.* Decision trees are especially attractive in data mining environments since human analysts readily comprehend the resulting models. Their construction does not require an analyst to provide input parameters: prior knowledge about the data is also not needed. A record can be associated with a unique leaf node by starting at the root and repeatedly choosing a child node based on the splitting criterion, which evaluates a condition on the input records at the current node.

Decision tree construction algorithms consist of two stages: tree building and pruning. In the former, most decision tree construction algorithms grow the tree top down in the following greedy way. Starting with the root node, the database is examined by "split selection method" for selecting the split condition at each node. The database is then partitioned and the procedure applied recursively. In the pruning stage, the tree constructed in the tree-building phase is pruned to control its size, and sophisticated pruning methods select the tree in a way that minimizes prediction errors.

The training database is accessed extensively while the tree is constructed; if the training database does not fit in memory, an efficient data-access method

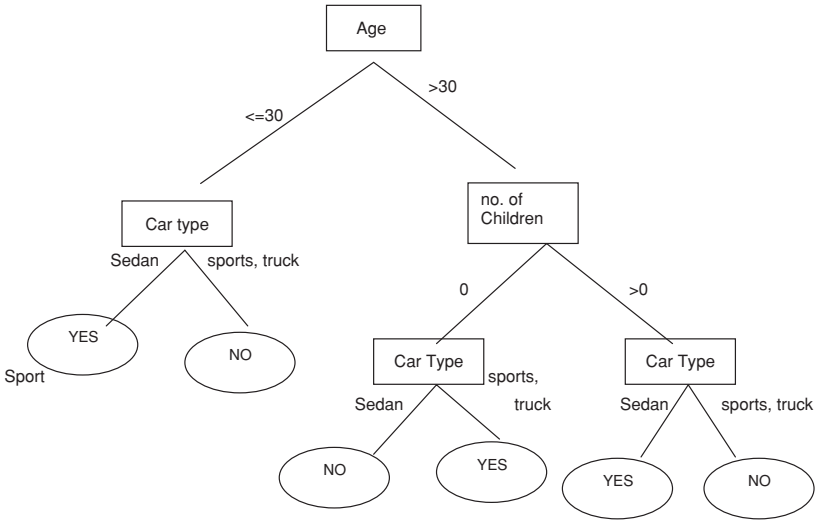


**Fig. 4.1.** Magazine subscription example classification tree

is needed to achieve scalability. Many scalable algorithms incorporate the observation that only a small set of sufficient statistics (such as aggregate measures, like counts) is necessary for applying popular split selection methods. The aggregated data is much smaller than the actual data. The statistics can be constructed in memory at each node in a single scan over the corresponding database partition that is, it satisfies the splitting criteria, leading to the node.

Although the sufficient statistics are often quite small, there are situations where the sufficient statistics are about as large as the complete data set. One way to deal with the size of the sufficient statistics is to observe that a large class of split selection methods searches over all possible split points and all attributes. The sufficient statistics at each step of the search are small – small enough to fit in memory. One way to utilize this observation is to create index structures over the training data set, thus permitting fast incremental computation of the sufficient statistics between adjacent steps of the search. For example, one class of algorithm vertically partitions the data set, sorts each partition by attribute value, then separately searches splitting criteria for each attribute by scanning the corresponding vertical partition.

Another way to deal with the size of the sufficient statistics is to split the problem into two phases. In the first, the algorithm scans the data set, constructing sufficient statistics in memory at a coarse granularity. Using the in-memory information, the algorithm prunes large parts of the search space of possible splitting criteria due to smoothness properties (such as bounds on the derivative) of the splitting criteria. In the second, the data set is scanned a second time, constructing exact sufficient statistics for only the parts of the search space that could not be eliminated in the first phase. Variations of

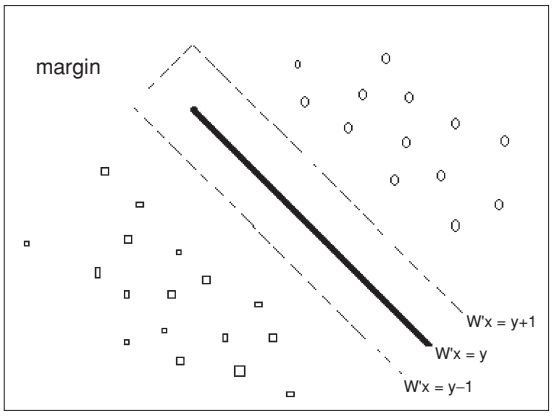


**Fig. 4.2.** SVM classification data points with dependent attribute = 0 are labeled with ∘'s, data points with dependent attribute = 1 are labeled with s □

this idea eliminate only part of search space with high probability in the first phase; the algorithm then checks decisions in the second phase. Algorithms based on this two-phase approach appear to be the fastest known methods for classification tree construction.

*Support Vector Machines.* Support vector machines (SVMs) are powerful and popular approaches to predictive modeling with success in a number of applications, including hand written digit recognition, charmed quark detection, face detection, and text categorization.

SVMs fit in the context of classification where the attribute whose value is to be predicted (the dependent attribute) has two possible values: 0 or 1. SVM classification is performed by a surface in the space of predictor attributes separating points with dependent attribute = 0 from those with dependent attribute = 1. An optimal separating surface is computed by minimizing the margin of separation (see Fig. 4.2). The margin of separation is the distance between the boundary of the points with dependent attribute = 0 and the boundary of those with dependent attribute = 1. The margin is a measure of "safety" in separating the two sets of points - the larger the better. In the standard SVM formulation, computing the optimal separating surface requires solving a quadratic optimization problem.

The burden of solving the SVM optimization problem grows dramatically with the number of training records. To reduce this burden, a method called "chunking" iteratively updates the separator parameters over chunks of training cases; the size of a chunk is chosen, so it fits into main memory. To obtain optimal classification chunks often need to be revisited, implying, multiple passes over the data.

A data compression approach called *squashing* is also applicable to SVM training, where the training records are summarized by a smaller data set emulating the distribution of the original training records. The training records

are clustered utilizing the likelihood profile of data. The SVM is then trained over the clusters, where the number of data points it contains weights each cluster.

Another approach to scaling SVMs involves reformulating the underlying optimization problem, resulting in efficient iterative algorithms. Methods require the solution of a linear system of equations with size $m+1$at each iteration or deal directly with the underlying optimality conditions (Karush-Kuhn-Tucker conditions) to incrementally improve the classifier at each iteration; $m$ is the number of predictor attributes.

The SVM predictive function can be decomposed as the linear combination of functions can be decomposed as the linear combination of functions of training data points (called kernels). Projection methods aim to approximate the combination of all training data with a subset of points. Some projection methods use $m$ randomly selected points on which to base the separating surface. Sparse, greedy matrix approximations try to determine the best $m$ points to use.

### 4.2.2 Clustering

Clustering aims to partition a set records into several groups such that "similar" records are in the same group according to some similarity function, identifying similar subpopulations in the data. For example a cluster could be a group of customers with similar purchase histories, interactions, and other factors.

One scalability technique for clustering algorithms is to incrementally summarize dense regions of the data while scanning a data set. Since a cluster corresponds to a dense region, the records within this region can be summarized collectively through a summarized representation called a *cluster feature* (CF), such as the triple consisting of the number of points in the cluster, the cluster centroid, and the cluster radius. More sophisticated cluster features are also possible.

CFs are efficient for two reasons: they occupy less space than maintaining all objects in a cluster; and if designed properly, they are sufficient for calculating all intercluster and intracluster measurements for making clustering decisions. Moreover, these calculations can be performed faster than all objects in clusters. Distances between clusters, radii of clusters and CFs – and hence other properties of merged clusters can all be computed quickly from the CFs of individual clusters.

CFs have also been used to scale iterative clustering algorithms, such as the k-means algorithm and expectation-maximization algorithm. When scaling iterative clustering algorithms, the algorithm identifies sets of discardable points, sets of compressible points, and a set of main-memory points. A point is discardable if its membership in a cluster can already be ascertained with high confidence; only the CF of all discardable points in a cluster is retained while the actual points are discarded. A point is compressible if it is not discardable but belongs to a tight subcluster consisting of a set of points that always

move between clusters simultaneously. The remaining records are designated as main-memory records, as they are neither discardable nor compressible. The iterative clustering algorithm then moves only the main-memory points and the CFs of compressible points between clusters until a criterion function is optimized.

Other research on scalable clustering focuses on training databases with large attribute sets. The search methods involve discovering the appropriate subspace of attributes in which the clusters are most likely to exist. These methods help analysis trying to understand the results, as they focus only on the attributes associated with a given cluster.

### 4.2.3 Association Rules

Association rules capture the set of significant correlation's present in a given data set. Given a set of transactions, where each transaction is a set of items, an association rule is an implication of the form X $\Rightarrow$ Y, where X and Y are sets of items. This rule has support $s$ if $s\%$ of transactions include all the items in both X and Y, and confidence $c$ if $c\%$ of transactions containing X also contain Y. for example, the rule "[carbonated beverages] and [crackers]$\Rightarrow$ [milk]" might hold in a supermarket database with 5% support and 70% confidence. The goal is to discover all association rules with support and confidence greater than the user-specified minimum support and minimum confidence, respectively. This formulation has been extended in many directions, including the incorporation of taxonomies, quantitative associations, and sequential patterns.

Algorithms for mining association rules usually have two distinct phases. First, they find all sets if items with minimum support (in other words, the frequent item sets). Since the data may consist of millions of transactions, and the algorithm may have to count millions of potentially frequent (candidate) item sets to identify the frequent ones, this phase can be computationally expensive. Next, rules can be generated directly from the frequent item sets, without having to go back to the data. The first step usually consumes most of the time: hence, research on scalability focuses here. Scalability techniques can be partitioned into two groups: those that reduce the number of candidates that need to be counted: and those that make the counting of candidates more efficient.

In the first group, identification of the antimonotonicity property that all subsets of a frequent item set must also be frequent proved to be a powerful pruning technique that dramatically reduces the number of item sets that need to be counted. Subsequent research has focused on variations of original problem. For instance, for data sets and support levels where the frequent item sets are very long, finding all frequent item sets is intractable, since a frequent item set with 'n' items has $2^n$ subsets. However, the set of maximal frequent item sets can still be found efficiently by looking again: once an item set is identified as frequent none of its subsets need to be counted. The key

is to maximize the probability that item sets counted by looking ahead are actually frequent. A good heuristic is to bias candidate generation, so the most frequent items appear in the most candidate groups. The intuition behind this heuristic is that items with high frequency are more likely to be part of long frequent item sets.

In the second group of techniques, nested hash tables can be used to efficiently check those candidate item sets that are contained in a transaction. This is very effective when counting shorter candidate item sets less so for longer candidates. Techniques for longer item sets include database projection, where the set of candidate item sets is partitioned into groups such that the candidates in each groups share a set of common items. Then before counting each candidate group the algorithm first discards the transactions that do not include all the common items for the remaining transactions, discards the common items (since it knows they are present), as well as items not present in any of the candidates. This reduction in the number and size of the remaining transactions can yield substantial improvements in the speed of counting.

### 4.2.4 From Incremental Model Maintenance to Streaming Data

Real-life data is not static, evolving constantly through additions and deletions of records; in some applications, such as network monitoring, data arrives in such high-speed data streams it is practically impossible to store it for offline analysis. A framework we call "block evolution" illustrates these models of evolving data. The input data set to the data mining process is not static, as it is updated with a new block of tuples at regular time intervals, say, everyday at midnight (a "block" is a set of tuples added simultaneously to the database). For large blocks, this model captures the common practice in many data warehouse installations, whereby updates from operational databases are batched together and performed in the block update. For small blocks (in the extreme, a single record), the model captures streaming data.

For evolving data, two classes of problem are of particular interest: data mining model maintenance and change detection. Model maintenance aims to maintain a data mining model undergoing insertion and deletion of blocks of data. Change detection aims to qualify the difference in terms of data characteristics between two blocks of data.

Recent research has focused on mining evolving data. Incremental model maintenance has also received attention, since it is desirable to go from incremental updates of the data warehouse to only incremental updates of existing data mining models, especially in the light of very large size of data warehouses. Incremental model maintenance algorithms concentrate on computing exactly the same model as if the original model construction algorithm was run on the combined collection of old and new data. One scalability technique widespread in model maintenance algorithms is the localization of changes for inserting new records. For example, for density-based clustering algorithms,

inserting a new record affects only clusters in the neighborhood of the record; thus efficient algorithms "localize" the change to the model without having to recompute the complete model. Another example involves decision tree construction, whereby the split criteria at the tree might change only within certain confidence intervals under insertion of records, assuming the underlying distribution of training records is static.

When working with high-speed data streams, algorithms have to be able to construct data mining models while looking at the relevant data items only once and in a fixed order (determined by the stream arrival pattern) with a limited amount of main memory. Data stream computation has given rise to several recent (theoretical and practical) studies of online and one-pass algorithms with limited memory requirements for data mining and related problems; examples include computating quantiles and order statistics; estimating frequency moments and join sizes; clustering and decision tree construction; estimating correlated aggregates and computing one dimensional, or single-attribute, histograms; and Haar wavelet decompositions. Scalability techniques include: sampling, summary statistics, sketches (small random projections with probable performance guarantees), and online compression of sufficient statistics.

Scalability in data mining is an active area of research, though many challenging questions remain, including the following:

- Can we mine patterns from huge data sets while preserving the privacy of individual records and the anonymity of the individuals who provided the data?
- What are suitable data mining models for high-speed data streams, and how can we construct them? and
- In light of the plethora of huge and growing sets of linked data available today, including in the Internet, newsgroups, and news stories, what type of knowledge can we mine from these resources, and can we design scalable algorithms for them?

## 4.3 Summary

The data mining innovations being implemented world wide often involve collaborations among domain experts, computer scientists, and statisticians. We expect these application-driven developments will continue to proliferate as data owners seek new end better ways to gain insight into their data. We can hope that more synergistic view of data mining, combining ideas from computer science and statistics will gradually emerge to provide a unifying theoretical framework for many of these efforts.

Large and growing databases are commonplace in business organizations, governments, and scientific applications. Prior to the invention of scaling techniques, sampling was the primary method for running conventional machine

learning, statistical, and other analysis algorithms on these databases. However, this approach also involved having to determine sufficient sample size, as well as the validity of discovered patterns or models (the patterns may be an artifact of the sample). We view sampling as orthogonal and complementary to scaling techniques, since scaling techniques allow the use of much larger data sets.

The research effort on scaling data mining algorithms to large databases has now given analysts the ability to model and discover valid, interesting patterns over these large data sets. General scaling principles include use of summary statistics, data compression, pruning the search space, and incremental computation. We have described these principles in the practical context of mining algorithms, expecting they would be useful in other areas of computer science as well.

## 4.4 Review Questions

1. Write short note on transaction data used in the mining algorithms.
2. Briefly describe on the data streams applied in mining process.
3. Explain about graphical, text-level, and scientific data used in mining.
4. Define predictive modeling and explain various techniques involved in this modeling.
5. Discuss clustering and about the association rules used in scaling.