OLAP Mining: An Integration of OLAP with Data Mining

Jiawei Han
Intelligent Database Systems Research Laboratory
School of Computing Science, Simon Fraser University, British Columbia, Canada
E-mail: han@cs.sfu.ca

Abstract

OLAP mining is a mechanism which integrates on-line analytical processing (OLAP) with data mining so that mining can be performed in different portions of databases or data warehouses and at different levels of abstraction at user's finger tips. With rapid developments of data warehouse and OLAP technologies in database industry, it is promising to develop OLAP mining mechanisms.

With our years of research into data mining, an OLAP-based data mining system, DBMiner, has been developed, where OLAP mining is not only for data characterization but also for other data mining functions, including association, classification, prediction, clustering, and sequencing. Such an integration increases the flexibility of mining and helps users find desired knowledge. In this paper, we introduce the concept of OLAP mining and discuss how OLAP mining should be implemented in a data mining system.

1 INTRODUCTION

With an enormous amount of data stored in databases and data warehouses, it is increasingly important to develop powerful data warehousing and data mining tools for analysis of this collected data and mining interesting knowledge from it (Fayyad, Piatetsky-Shapiro, Smyth & Uthurusamy 1996).

Among many different designs and architectures of data mining systems, OLAP mining, which integrates on-line analytical processing (OLAP) with data mining, is a promising direction based on the following reasoning.

1. Data mining tools need to work on integrated, consistent, and cleaned data, which often require data cleaning and data integration as preprocessing steps (Fayyad et al. 1996). A data warehouse constructed by such preprocessing serves as a valuable source of cleaned and integrated data for on-line analytical processing as well as for data mining.

2. OLAP mining facilitates interactive exploratory data analysis. Users often like to traverse flexibly through a database, select any portions of relevant data, analyze data at different levels of abstraction, and present knowledge/results in different forms. OLAP mining provides such a tool for drilling, pivoting, filtering, dicing and slicing on any sets of data in data cubes, for analyzing data at different levels of abstraction, and for interacting flexibly with the mining engine based on intermediate mining results.
3. It is often difficult for users to predict what kinds of knowledge to be mined beforehand. By integration of OLAP with multiple data mining modules, OLAP mining provides flexibility for users to select desired data mining functions and swap data mining tasks dynamically.

The above observations motivate us to study the desired ways to perform OLAP mining and their efficient implementation methods.

With our years of research and development, an OLAP data mining system, DBMiner, has been developed by integration of database, OLAP and data mining technologies (Han & Fu 1995, Han & Fu 1996, Han, Chiang, Chee, Chen, Chen, Cheng, Gong, Kamber, Liu, Koperski, Lu, Stefanovic, Winstone, Xia, Zaiane, Zhang & Zhu 1997). The system mines various kinds of knowledge at multiple levels of abstraction from large relational databases and data warehouses efficiently and effectively. In this paper, we examine the principles of OLAP mining and study its implementation techniques with the DBMiner system as a running example.

The remaining of the paper is organized as follows. Section 2 is a brief introduction to OLAP and data mining technologies. In Section 3, we examine OLAP mining and the desired OLAP mining functions. In Section 4, methods for efficient implementation of OLAP mining are studied. In Section 5, we summarize the study and propose some future research topics.

2 OLAP AND DATA MINING

To understand what is OLAP mining, we need to first understand what is OLAP and what is data mining.

OLAP (On-Line Analytical Processing) refers to a set of data analysis techniques developed for analyzing data in data warehouses since 1990s. A data warehouse stores a large collection of subject-oriented, integrated, time-variant, and nonvolatile data in support of management's decision-making process. It presents a multidimensional, logical view of the data, and is hence called a multidimensional database or data cube. A point in a data cube stores a consolidated measure of the corresponding dimension values in a multidimensional space. OLAP operations, such as drill-down, roll-up, pivot, slice,

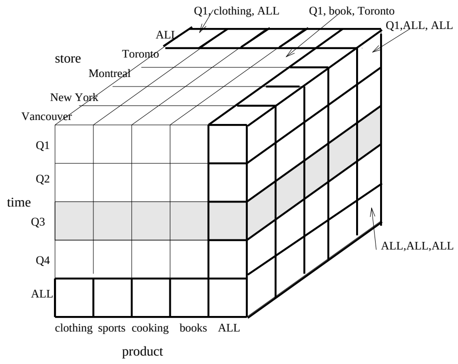

Figure 1 A 3-D data cube of a market data warehouse

dice, etc., are the ways to interact with the data cube for multidimensional data analysis.

Example 1 A market data warehouse shown in Figure 1 consists of three dimensions: *store, item*, and *time*, and two measures, *number-of-units-sold* and *profit*. A concept hierarchy is associated with each dimension as follows,

$$store(store\_id, street, city, province, country)$$
$$item(item\_id, item\_name, brand, category)$$
$$time(minute, hour, day, month, quarter, year).$$

Drill-down or roll-up operations can be performed along each dimension. For example, one may start with a low-level cube which consists of $store\_id$, $item\_name$, and $hour$, and roll-up to examine the number of items sold by category, by city, and by quarter and then drill-down to see the number of items sold by item name, by store, and by month.

In addition to performing drill-down and roll-up operations, there are several other popularly used OLAP operations: slicing, dicing and pivoting. Slicing is the extraction, from a data cube, of summarized data for a given dimension-value, or *slice*. For example, one may slice on a particular store to examine the total number of items solded by item name and by day. Dicing is the extraction of a "subcube", or intersection of several slices of the data cube. For example, to examine various kinds of TVs sold in Sears in August 1997, one need to dice using several constants or range values. *Filtering* is to perform selection on a data cube using some constants. Finally, *pivoting* rotates the axes of a data cube so that one may examine a cube from different angles.

The data cube can be browsed conveniently using the DBMiner cube browser as shown in Figure 2, where the size of a cuboid represents the entry count

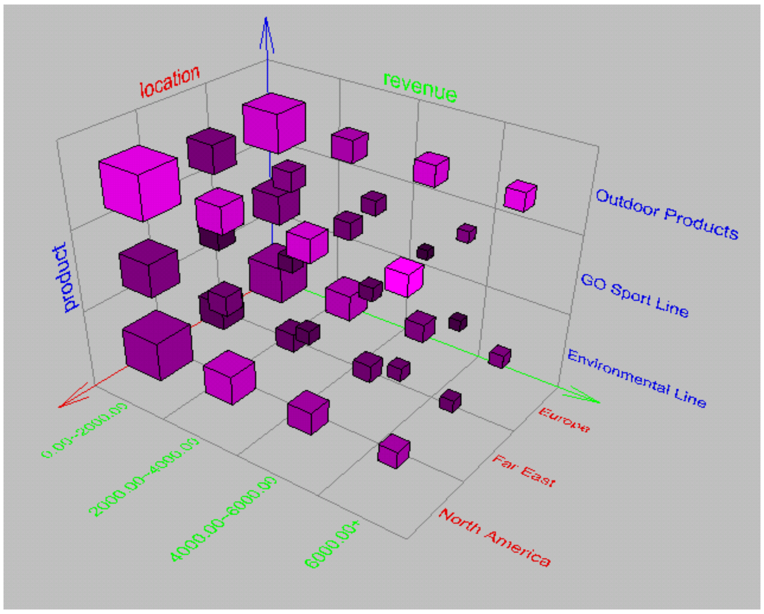

Figure 2 Browsing of a 3-dimensional data cube in DBMiner

in the corresponding cell, and the brightness of a cuboid represents the accumulated amount in the cell. Drilling and slicing/dicing operations can also be performed on the data cube with simple button clicking.

Moreover, OLAP operations in some systems include comprehensive statistical analysis packages, such as trend analysis, ratios and ranking, charting, browsing and surfing, iterative analysis, linear and non-linear modeling, regression analysis, time-series analysis, and multidimensional or complex correlation analysis. □

To facilitate our discussion of OLAP mining, the popularly used cube transformation operations, including drilling, rolling, slicing, dicing, filtering, and pivoting, are called cubing operations because they lead to the generation of new data cubes. Efficient computation of data cubes and efficient implementation of OLAP operations have been investigated with a spectrum of techniques proposed, such as multiway aggregation of multidimensional arrays (Agarwal, Agrawal, Deshpande, Gupta, Naughton, Ramakrishnan & Sarawagi 1996, Zhao, Deshpande & Naughton 1997), indexing data cubes (Sarawagi 1997), efficient OLAP computations, etc. These techniques are also essential for efficient implementation of OLAP mining.

Data mining is to discover some nontrivial and interesting knowledge or patterns from the data stored in large databases. It consists of several major functions as follows.

- Characterization which generalizes a set of task-relevant data into a *generalized data cube* which can then be used for extraction of different kinds of rules or be viewed at multiple levels of abstraction from different angles. In particular, it derives a set of characteristic rules which summarizes the general characteristics of a set of user-specified data (called the *target class*). For example, the symptoms of a specific disease can be summarized by a characteristic rule.
- Comparison which mines a set of discriminant rules which summarize the features that distinguish the class being examined (the *target class*) from other classes (called *contrasting classes*). For example, to distinguish one disease from others, a discriminant rule summarizes the symptoms that discriminate this disease from others.
- Classification which analyzes a set of training data (i.e., a set of objects whose class label is known) and constructs a model for each class based on the features in the data. A set of classification rules is generated by such a classification process, which can be used to classify future data and develop a better understanding of each class in the database. For example, one may classify diseases and provide the symptoms which describe each class or subclass.
- Association which discovers a set of association rules (in the form of "$A_1 \wedge \cdots \wedge A_i \rightarrow B_1 \wedge \cdots \wedge B_j$") at multiple levels of abstraction from the relevant set(s) of data in a database. For example, one may discover a set of symptoms often occurring together with certain kinds of diseases and further study the reasons behind them.
- Prediction which predicts the possible values of some missing data or the value distribution of certain attributes in a set of objects. This involves finding the set of attributes relevant to the attribute of interest (by some statistical analysis) and predicting the value distribution based on the set of data similar to the selected object(s). For example, an employee's potential salary can be predicted based on the salary distribution of similar employees in the company.
- Cluster analysis which groups a selected set of data in the database or data warehouse into a set of clusters to ensure the interclass similarity is low and intraclass similarity is high. For example, one may cluster the houses in Vancouver area according to house type, value, and geographical location.
- Time-series analysis which performs data analyses for time-related data in databases or data warehouses, including similarity analysis, periodicity analysis, sequential pattern analysis, and trend and deviation analysis. For example, one may find the general characteristics of the companies whose stock price has gone up over 20% last year or evaluate the trend or particular growth patterns of certain stocks.

Efficient data mining methods for large databases have been studied extensively, with many interesting methods developed. For example, attribute-oriented induction method for mining characteristic and comparison rules (Han & Fu 1996), Apriori algorithm for mining association rules (Agrawal & Srikant 1994), Cart algorithm for classification and decision tree construction (Quinlan 1993), and CLARANS, DBScan, Birch, and k-means algorithms for clustering analysis (Ng & Han 1994, Ester, Kriegel, Sander & Xu 1996, Zhang, Ramakrishnan & Livny 1996, Jain & Dubes 1988).

3 DESIRED OLAP MINING FUNCTIONS

By integration of OLAP and data mining, OLAP mining facilitates flexible mining of interesting knowledge in data cubes because data mining can be performed at multi-dimensional and multi-level abstraction space in a data cube. Cubing and mining functions can be interleaved and integrated to make data mining a highly interactive and interesting process.

Here we first examine what are the desired OLAP mining functions.

1. Cubing then mining: With the availability of data cubes and cubing operations, mining can be performed on any layers and any portions of a data cube. This means that one can first perform cubing operations to select the portion of the data and set the abstraction layer (granularity level) before a data mining process starts.
   For example, one may first tailor a cube to a particular subset, such as "*year = 1997*", and to a desired level, such as at the city level for the dimension *store*, and then execute a prediction mining module.
2. Mining then cubing: This means that data mining can be first performed on a data cube, and then particular mining results can be analyzed further by cubing operations.
   For example, one may first perform classification on a "market" data cube according to a particular dimension or measure, such as *profit_made*. Then for each obtained class, such as the *high_profit* class, cubing operations can be performed, e.g., drill-down to detailed levels and examine its characteristics.
3. Cubing while mining: A flexible way to integrate mining and cubing operation is to perform similar mining operations at multiple granularities by initiating cubing operations during mining. By doing so, the same data mining operations can be performed on different portions of a cube or at different abstraction levels.
   For example, for mining association rules in a "market" data cube, one can drill down along a dimension, such as *time*, to find new association rules at a lower level of abstraction, such as *from year to month*.
4. Backtracking: To facilitate interactive mining, one should allow a mining process to backtrack one or a few steps or backtrack to a preset marker and then explore alternative mining paths.

For example, one may classify market data according to *profit_made* and then drill down along some dimension(s), such as *store* to see its characteristics. Alternatively, one may like to classify the data according to another measure, *cost_of_product*, and then do the same (characterization). This requires the miner to jump back a few steps or backtrack to some previously marked point, and redo the classification. Such flexible traversal along the cube at mining is a highly desired feature for users.

5. Comparative mining: A flexible data miner should allow comparative data mining, that is, the comparison of alternative data mining processes.
   For example, a data miner may contain several cluster analysis algorithms. One may like to compare side by side the clustering quality of different algorithms, even examine them when performing cubing operations, such as when drilling down to detailed abstraction layers.

It is possible to have other combinations in OLAP mining. For example, one can perform "mining then mining", such as first perform classification on a set of data and then find association patterns for each class.

In a large warehouse containing a huge amount of data, it is crucial to provide flexibilities in data mining so that a user may traverse a data cube, select mining space and the desired levels of abstraction, and test different mining modules and alternative mining algorithms at his/her finger tips. By doing so, mining will be a highly interactive, enjoyable, and productive process.

4 EFFICIENT IMPLEMENTATION OF OLAP MINING

Assuming readers have general knowledge of data warehousing (Chaudhuri & Dayal 1997) and data mining (Fayyad et al. 1996), we examine the methods for efficient implementation of OLAP mining in this section.

With recent developments of data warehousing technology, data cubes can be computed and accessed efficiently. For example, one may use either a MOLAP (based on multi-dimensional array structures) or a ROLAP (based on relational structures) approach for efficient cube storage and computation (Agarwal et al. 1996, Gray, Chaudhuri, Bosworth, Layman, Reichart, Venkatrao, Pellow & Pirahesh 1997, Zhao et al. 1997), where a cube could be dense or sparse. Also, data cubes can be indexed or bit-mapped in several ways for efficient accessing (Chaudhuri & Dayal 1997, Sarawagi 1997). With these technologies available, our discussion will focus on how OLAP mining should be performed in cooperation with data mining functions.

4.1 OLAP-based characterization and comparison

Data characterization summarizes and characterizes a set of task-relevant data based on data generalization. For mining multiple-level knowledge, progressive deepening (*drill-down*) and progressive generalization (*roll-up*) techniques can be applied.

Progressive generalization starts with a conservative generalization process which first generalizes the data to a slightly higher level than the primitive data in the data cube. Further generalizations can be performed on it progressively by selecting appropriate attributes for step-by-step generalization.

Progressive deepening starts with a relatively high-level generalized cuboid, selectively and progressively specializes some of the generalized tuples or attributes to lower abstraction levels.

Conceptually, a top-down, progressive deepening process is preferable since it is natural to first find general data characteristics at a high abstraction level and then follow certain interesting paths to drill down to specialized cases. However, from the implementation point of view, it is easier to perform generalization than specialization because generalization replaces low level data by high ones through ascension of a concept hierarchy. Since a generalized cell does not register the detailed original information, it is difficult to get such information back when specialization is required later.

To facilitate specializations on a high-level cuboid, a typical technique is to save a set of "*lower-level cuboids*", especially the "*minimally generalized cuboid*" either at the preprocessing stage (i.e., cube computation stage) or in the early stage of generalization. For example, to compute the *minimally generalized cuboid*, each dimension in the relevant set of data can be generalized to minimally generalized concepts (which can be done in one scan of the database or warehouse), with the measures aggregated correspondingly. After that, both *progressive deepening* and *interactive up-and-down* can be performed with reasonable efficiency: If the data at the current abstraction level is to be generalized further, generalization can be performed on it directly; on the other hand, if it is to be specialized, the desired result can be derived by searching for the closest lower-level cuboid and generalizing such a cuboid to appropriate level(s) if necessary.

An output of the characterizer is shown in Figure 3.

Comparison is to find a set of discriminant rules which distinguish the general features of a target class from that of the contrasting class(es) specified by a user. It is implemented as follows.

First, the set of relevant data in the database has been collected by query processing and is partitioned respectively into a *target* class and one or a set of *contrasting* class(es). Second, attribute-oriented induction is performed on the target class to extract a *prime target cube*, where a *prime target cube* is a

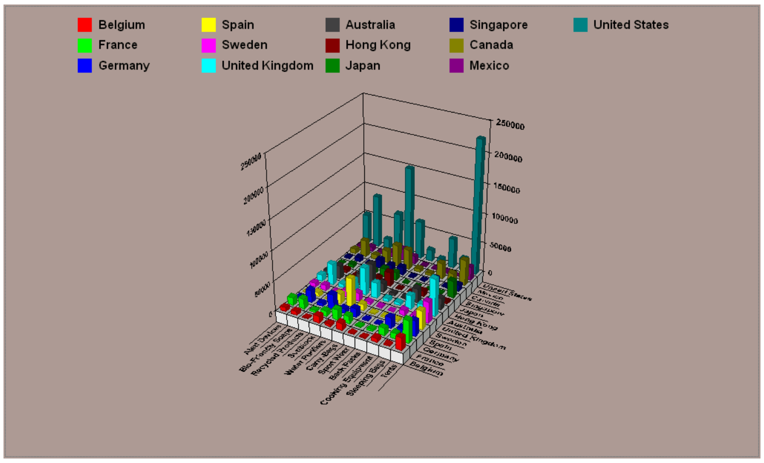

Figure 3 Graphical output of the Characterizer of DBMiner

generalized cuboid in which each attribute contains no more than but close to the threshold value of the corresponding attribute. Then the concepts in the *contrasting* class(es) are generalized to the same level as those in the prime target cube, forming the *prime contrasting cube*. Finally, the information in these two classes is used to generate qualitative or quantitative discriminant rules.

Moreover, interactive drill-down and roll-up can be performed synchronously in both target class and contrasting class(es) in a similar way as in characterization.

How can multi-level characterization and comparison be integrated into OLAP mining? Since at each step of drill-down or roll-up, characterization and comparison produce a new cuboid, with the same data structure, it is inherently suitable for integration with OLAP mining. That is, any mining module can treat the result of characterization and comparison as a data cube and mining can be performed directly on such a resulting cube. Furthermore, for any mining result on which cubing operations can be performed, characterization and comparison can be done as well.

4.2 OLAP-based association

Based on many studies on efficient mining of association rules (Agrawal & Srikant 1994, Srikant & Agrawal 1995, Han & Fu 1995), a multiple-level association rule miner (called "associator") has been implemented in DBMiner. An output of the associator is shown in Figure 4.

Different from mining association rules in transaction databases, a relational

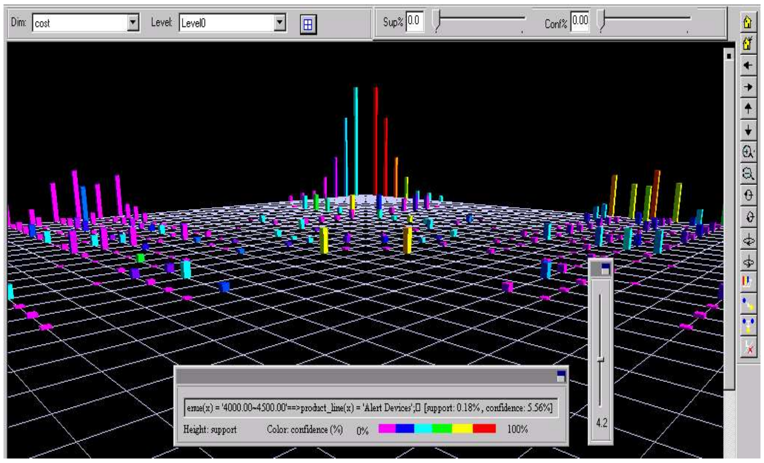

Figure 4 Graphical output of the Associator of DBMiner

associator may find two kinds of associations: *inter-attribute association* and *intra-attribute association*. The former is an association among different attributes; whereas the latter is an association within one or a set of attributes formed by grouping of another set of attributes. This is illustrated in the following example.

Example 2. Suppose the "course_taken" relation in a university database has the following schema:

$$course\_taken = (student\_id, course, semester, grade).$$

Intra-attribute association is the association among one or a set of attributes formed by grouping another set of attributes in a relation. For example, the associations between each student and his/her course performance is an intra-attribute association because a set of attributes, "$course, semester, grade$", are grouped according to $student\_id$, for mining associations among the courses taken by each student. From a relational database point of view, a relation so formed is a nested relation obtained by nesting "$(course, semester, grade)$" with the same $student\_id$. Therefore, an intra-attribute association is an association among the nested items in a nested relation.

Inter-attribute association is the association among a set of attributes in a *flat* relation. For example, the association between $course$ and $grade$, such as "*the courses in computing science tend to give good grades*", is an inter-attribute association.

Two associations require different mining algorithms.

For mining intra-attribute associations, a data relation can be transformed into a nested relation in which the tuples which share the same values in the nesting attributes are merged into one. For example, the *course_taken* relation can be folded into a nested relation with the schema,

$$\begin{aligned} course\_taken &= (student\_id, course\_history) \\ course\_history &= (course, semester, grade). \end{aligned}$$

With such transformation, it is easy to derive association rules like "*90% senior CS students tend to take at least three CS courses at 300-level or up in each semester*". Since the nested tuples (or values) can be viewed as data items in the same transaction, the methods for mining association rules in transaction databases, such as Apriori (Agrawal & Srikant 1994), can be applied to such transformed relations in relational databases. □

Moreover, it is preferable to have some user-specified constraints to guide an association rule mining process. Such constraints can be specified in a *meta-rule* (or *meta-pattern*) form (Kamber, Han & Chiang 1997), which confines the search to specific forms of rules. For example, a meta-rule "$P(x, y) \rightarrow Q(x, y, z)$", where P and Q are predicate variables matching different properties in a database, can be used as a rule-form constraint in the search.

The multi-dimensional data cube structure facilitates efficient mining of multi-level, inter-attribute association rules. A count cell of a cube stores the number of occurrences of the corresponding multi-dimensional data values; whereas a dimension count cell stores the sum of counts of the whole dimension. With this structure, it is straightforward to calculate the measures such as *support* and *confidence* of association rules based on the values in these summary cells. A set of such cuboids, ranging from the least generalized one to rather high level ones, facilitate mining of association rules at multiple levels of abstraction.

Association mining module generates a set of association rules. Since the rule form is quite different from a data cube structure, it is not easy to integrate the mining results with other mining/cubing processes. One choice is to take any rule which contains a few connected nodes as a cuboid, from which characteristics can be displayed and drill-down or roll-up can be performed. Another choice is to take a node as a one-dimensional cuboid and show data distribution and add additional attributes for cubing and mining. The third choice is to simply backtrack to a point where cubing and other mining operations can take place naturally.

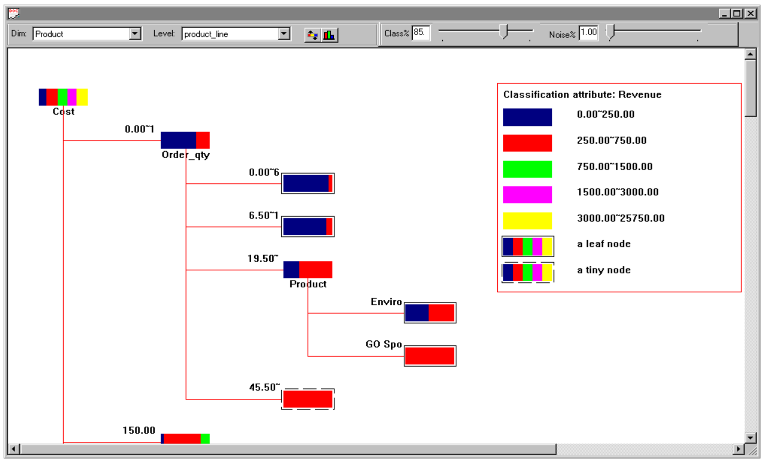

Figure 5 Graphical output of the Classifier of DBMiner

4.3 OLAP-based classification

Data classification is to develop a description or model for each class in a database, based on the features present in a set of class-labeled training data.

There have been many classification methods studied, including decision-tree methods, such as ID-3 and C4.5 (Quinlan 1993), statistical methods, neural networks, rough sets, etc. Recently, some database-oriented classification methods have also been investigated (Mehta, Agrawal & Rissanen 1996).

Our classification method consists of four steps: (1) collection of the relevant set of data and partitioning of the data into training and test data, (2) analysis of the relevance of the attributes, (3) construction of classification (decision) tree, and (4) test of the effectiveness of the classification using the test data set.

Attribute relevance analysis is performed based on the analysis of an uncertainty measurement, a measurement which determines how much an attribute is in relevance to the class attribute. Other measurements, such as entroy-based information gain (Quinlan 1993) and Gini index (Mehta et al. 1996), can be used for relevance analysis as well. Several top-most relevant attributes are retained for classification analysis; whereas the weakly or irrelevant attributes are not considered in the subsequent classification process.

In the classification process, our classifier adopts a generalization-based decision-tree induction method which integrates OLAP data cube technology with a decision-tree induction technique, by first performing minimal generalization on the set of training data, and then performing decision tree induction on the generalized data.

Since a generalized cell comes from the generalization of a number of original cells, the *count* information is associated with each generalized cell and plays an important role in classification. To handle noise and exceptional data and facilitate statistical analysis, two thresholds, *classification threshold* and *exception threshold*, are introduced. The former is used for justification whether it is needed to continue classification on a node if a significant set of the examples of the node belongs to a single class; whereas the latter is used to terminate further classification on a node if the node contains only a negligible number of examples.

There are several alternatives for doing generalization before classification: A data set can be generalized to either a minimally generalized abstraction level, an intermediate abstraction level, or a rather high abstraction level. Too low an abstraction level may result in scattered classes, bushy classification trees, and difficulty at concise semantic interpretation; whereas too high a level may result in the loss of classification accuracy.

For OLAP mining, classification can be associated with other cubing and mining functions as follows. For any cubing result, one attribute can be selected as class attribute and classification can be performed on the corresponding cuboid in the same way as our cube-based classification process. For any classification result, each class node can be treated as a portion of the cube selected by the class constraint. Subsequent cubing and mining operations can be performed on the selected class.

The multi-level classification process has been implemented in the DBMiner system. An output of the DBMiner classifier is shown in Figure 5.

4.4 OLAP-based prediction

A predictor predicts data values or value distributions on the attributes of interest based on similar groups of data in the database. For example, one may predict the amount of research grants that an applicant may receive based on the data about the similar groups of researchers.

The power of data prediction should be confined to the ranges of numerical data or the nominal data generalizable to only a small number of categories. It is unlikely to give reasonable prediction on one's name or social insurance number based on other persons' data.

For successful prediction, the factors (or attributes) which strongly influence the values of the attributes of interest should be identified first. This can be done by the analysis of data relevance or correlations by statistical methods, decision-tree classification techniques, or be simply based on expert judgement. To analyze attribute relevance, the uncertainty measurement similar to the method used in our classifier is applied. This process ranks the

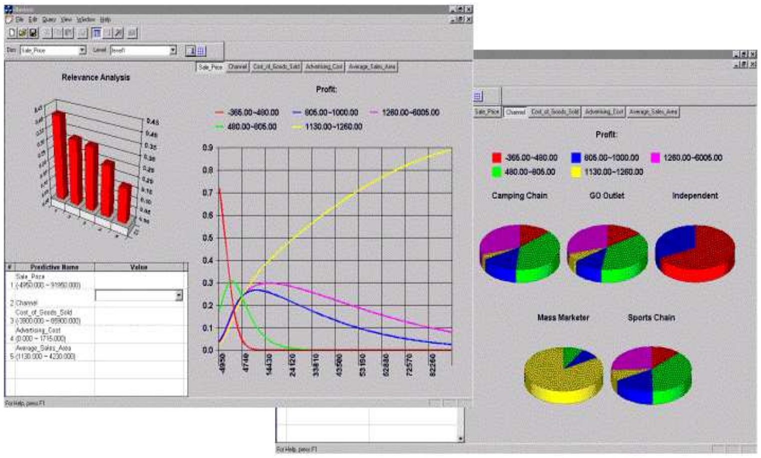

Figure 6 Graphical output of the Predictor of DBMiner: numeric predictive attribute (left) and categorical predictive attribute (right)

relevance of all the attributes selected and only the highly ranked attributes will be used in the prediction process.

After the selection of highly relevant attributes, a generalized linear model has been constructed which can be used to predict the value or value distribution of the predicted attribute. If the predictive attribute is a numerical data, a set of curves are generated, each indicating the trend of likely changes of the value distribution of the predicted attribute. If the predictive attribute is a categorical data, a set of pie charts are generated, each indicating the distributions of the value ranges of the predicted attribute.

When a query probe is submitted, the corresponding value distribution of the predicted attribute can be plotted based on the curves or pie charts generated above. Therefore, the values in the set of highly relevant predictive attributes can be used for trustable prediction.

The prediction output has two forms of presentation: curve graph and pie chart depending whether the predictive attribute is a numeric attribute or a categorical attribute. When the predictive attribute is a numeric one, the output is a set of curves as shown in the left half of Figure 6; whereas when the predictive attribute is a categorical one, the output is a set of pie charts as shown in the right half of Figure 6.

OLAP mining can be integrated with the prediction module as follows. For any predicted class, the class can be identified by a class selection criteria and its characteristics can be displayed. Then cubing operations can be performed on such a selected cuboid. Alternatively, one may backtrack to a point before prediction is performed and continue the exploration of other features of the previously selected data cube.

4.5 OLAP-based clustering analysis

Data clustering, also viewed as "unsupervised learning", is a process of partitioning a set of data into a set of classes, called *clusters*, with the members of each cluster sharing some interesting common properties. A good clustering method will produce high quality clusters, in which the intra-class (i.e., intra-cluster) similarity is high and inter-class similarity is low.

Clustering has many interesting applications. For example, it can be used to help marketers discover distinct groups in their customer bases and develop targeted marketing programs.

Data clustering has been studied in statistics, machine learning and data mining with different methods and emphases. Many clustering methods have been developed and applied to various domains, such as data classification and image processing.

Data mining applications deal with large high dimensional data, and frequently involve categorical domains with concept hierarchies. However, most of the existing data clustering methods can only handle numeric data, or cannot produce good quality results in the case where categorical domains are present.

Our cluster analyzer is based on the well-known *k-means* paradigm. Comparing to the other clustering methods, the *k-means* based methods are promising for their efficiency in processing large data sets. However, their use is often limited to numeric data. To adequately reflect categorical domains, we have developed a method of encoding concept hierarchies. This enables us to define a dissimilarity measure that not only takes into account both numeric and categorical attributes, but also at multiple levels. Due to these modifications, our cluster analyzer can cluster large data sets with mixed numeric and categorical attributes in a way similar to *k-means*. It can also perform multi-level clustering and select a desired level by a comparison of the clustering quality at different levels. On the other hand, the user or the analyst can direct the clustering process by either selecting a set of relevant attributes for the requested clustering query, or assigning a weight factor to each attribute, or both, so that increasing the weight of an attribute increases the likelihood that the algorithm will cluster according to that attribute.

OLAP mining can be integrated with cluster analyzer as follows. For any cluster so obtained, its characteristics can be displayed and cubing/mining operations can be performed on such a selected cluster. Alternatively, one may backtrack to a point before clustering is performed and continue the exploration of other features of the previously selected data cube.

4.6 Backtracking and comparative mining analysis

Backtracking is convenient for OLAP mining since a user may like to tentatively dig deep following some mining paths and later try alternatives if there have not been desired interesting patterns found.

One suggested technique for implementation of backtracking in OLAP mining is as follows. First, a status vector should be saved in a backtrack stack (if the backtrack pattern is simply tracing back step by step) or a backtrack list (if position marking or other traversal patterns is desired). The cuboid(s) associated with the vector should also be saved on the disk and linked with the vector. When backtracking, such a stack/list is used to trace back the appropriate status pointers. When a session is complete, all the saved backtrack pointers, and their associated vectors and cuboids should be deleted to release the disk space occupied.

Comparative mining analysis can be implemented similarly. For comparative analysis of two mining tasks, one needs to fork a new path of mining/cubing process stream and show both streams in separate windows. Comparative displays can also be synchronized when necessary by bundling together the two similar mining/cubing processes.

5 DISCUSSION AND CONCLUSIONS

OLAP mining integrates on-line analytical processing with data mining which substantially enhances the power and flexibility of data mining and makes mining an interesting exploratory process.

In this paper, we discussed the principles and some implementation techniques of OLAP mining by taking the OLAP mining functions of the DBMiner system as running examples.

With multiple data mining functions available, one question which naturally arises is how to determine which data mining function is the most appropriate one for a specific application. To select an appropriate data mining function, one needs to be familiar with the application problem, data characteristics, and the roles of different data mining functions. Sometimes one needs to perform interactive exploratory analysis to observe which function discloses the most interesting features in the database. Therefore, the building of exploratory analysis tools and the construction of an application-oriented semantic layer are two important solutions. OLAP mining provides an exploratory analysis tool, however, further study should be performed on the *automatic selection* of data mining functions for particular applications.

Another popularly posed question is how these data mining techniques are different from the set of existing statistical data analysis tools. Data mining is the confluence of multiple disciplines, including database systems, data warehouses, statistics, visualization, machine learning, and information science. Previous work on statistics has provided some foundational work for data mining. Most effective data mining systems are extending the power of database systems or data warehouse systems and re-examining the methods studied in statistics, visualization, and machine learning. Many methods developed in data mining systems are novel and scalable, and integrates well with existing database systems, which are quite different from existing statistical analysis tools and machine learning packages. Thus, data mining forms a new direction in the research into the methods for the analysis of data and knowledge in large databases and data warehouses.

We are currently working on the further enhancement of the power and efficiency of OLAP mining of DBMiner for exploratory data mining, including the improvement of system performance and rule discovery quality for the existing functional modules, and the development of techniques for mining new kinds of rules, especially on time-related data, and visual data mining.

Another important task for future study is the extension of OLAP mining techniques towards advanced and/or special purpose database systems, including extended-relational, object-oriented, text, spatial, temporal, multimedia, and heterogeneous databases and Internet information systems. We will report the progress on OLAP mining of complex types of structured, semi-structured and nonstructured data in the future.

REFERENCES

Agarwal, S., Agrawal, R., Deshpande, P. M., Gupta, A., Naughton, J. F., Ramakrishnan, R. & Sarawagi, S. (1996), On the computation of multidimensional aggregates, *in* 'Proc. 1996 Int. Conf. Very Large Data Bases', Bombay, India, pp. 506–521.

Agrawal, R. & Srikant, R. (1994), Fast algorithms for mining association rules, *in* 'Proc. 1994 Int. Conf. Very Large Data Bases', Santiago, Chile, pp. 487–499.

Chaudhuri, S. & Dayal, U. (1997), 'An overview of data warehousing and OLAP technology', *ACM SIGMOD Record* **26**, 65–74.

Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. (1996), A density-based algorithm for discovering clusters in large spatial databases, *in* 'Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96)', Portland, Oregon, pp. 226–231.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. & Uthurusamy, R. (1996), *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press.

Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F. & Pirahesh, H. (1997), 'Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals', *Data Mining and Knowledge Discovery* **1**, 29–54.

Han, J. & Fu, Y. (1995), Discovery of multiple-level association rules from large databases, *in* 'Proc. 1995 Int. Conf. Very Large Data Bases', Zurich, Switzerland, pp. 420–431.

Han, J. & Fu, Y. (1996), Exploration of the power of attribute-oriented induction in data mining, *in* U. Fayyad, G. Piatetsky-Shapiro, P. Smyth & R. Uthurusamy, eds, 'Advances in Knowledge Discovery and Data Mining', AAAI/MIT Press, pp. 399–421.

Han, J., Chiang, J., Chee, S., Chen, J., Chen, Q., Cheng, S., Gong, W., Kamber, M., Liu, G., Koperski, K., Lu, Y., Stefanovic, N., Winstone, L., Xia, B., Zaiane, O. R., Zhang, S. & Zhu, H. (1997), DBMiner: A system for data mining in relational databases and data warehouses, *in* 'Proc. CASCON'97: Meeting of Minds', Toronto, Canada, pp. 249–260.

Jain, A. K. & Dubes, R. C. (1988), *Algorithms for Clustering Data*, Printice Hall.

Kamber, M., Han, J. & Chiang, J. Y. (1997), Metarule-guided mining of multi-dimensional association rules using data cubes, *in* 'Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD'97)', Newport Beach, California, pp. 207–210.

Mehta, M., Agrawal, R. & Rissanen, J. (1996), SLIQ: A fast scalable classifier for data mining, *in* 'Proc. 1996 Int. Conference on Extending Database Technology (EDBT'96)', Avignon, France.

Ng, R. & Han, J. (1994), Efficient and effective clustering method for spatial data mining, *in* 'Proc. 1994 Int. Conf. Very Large Data Bases', Santiago, Chile, pp. 144–155.

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Sarawagi, S. (1997), 'Indexing OLAP data', *Bulletin of the Technical Committee on Data Engineering* **20**, 36–43.

Srikant, R. & Agrawal, R. (1995), Mining generalized association rules, *in* 'Proc. 1995 Int. Conf. Very Large Data Bases', Zurich, Switzerland, pp. 407–419.

Zhang, T., Ramakrishnan, R. & Livny, M. (1996), BIRCH: an efficient data clustering method for very large databases, *in* 'Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data', Montreal, Canada, pp. 103–114.

Zhao, Y., Deshpande, P. M. & Naughton, J. F. (1997), An array-based algorithm for simultaneous multidimensional aggregates, *in* 'Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data', Tucson, Arizona, pp. 159–170.