DBMiner: A System for Data Mining in Relational Databases and Data Warehouses*

Jiawei Han Jenny Y. Chiang Sonny Chee Jianping Chen Qing Chen
Shan Cheng Wan Gong Micheline Kamber Krzysztof Koperski
Gang Liu Yijun Lu Nebojsa Stefanovic Lara Winstone
Betty B. Xia Osmar R. Zaiane Shuhua Zhang Hua Zhu

Data Mining Research Group, Intelligent Database Systems Research Laboratory
School of Computing Science, Simon Fraser University, British Columbia, Canada V5A 1S6
URL: http://db.cs.sfu.ca/ (for research group) http://db.cs.sfu.ca/DBMiner (for system)

Abstract

A data mining system, `DBMiner`, has been developed for interactive mining of multiple-level knowledge in large relational databases and data warehouses. The system implements a wide spectrum of data mining functions, including characterization, comparison, association, classification, prediction, and clustering. By incorporating several interesting data mining techniques, including OLAP and attribute-oriented induction, statistical analysis, progressive deepening for mining multiple-level knowledge, and meta-rule guided mining, the system provides a user-friendly, interactive data mining environment with good performance.

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 such data and mining interesting knowledge from it [14, 4].

With our years of research and development on data mining and knowledge discovery in databases, a data mining system, `DBMiner`, has been developed by integration of database, OLAP and data mining technologies [6, 8, 9, 10]. The system mines various kinds of knowledge at multiple levels of abstraction from large relational databases and data warehouses efficiently and effectively, with the following distinct features:

1. It incorporates several interesting data mining techniques, including data cube and OLAP technology [3], attribute-oriented induction [6, 9], statistical analysis, progressive deepening for mining multiple-level rules [8, 9, 12], and meta-rule guided knowledge mining [5, 11]. It also implements a wide spectrum of data mining functions including characterization, comparison, association, classification, prediction, and clustering.
2. It performs interactive data mining at multiple levels of abstraction on any user-specified set of data in a database or a data warehouse using an SQL-like Data Mining Query Language, `DMQL`, or a graphical user interface. Users may interactively set and adjust various thresholds, control a data mining process, perform *roll-up* or *drill-down* at multiple levels of abstraction, and generate different forms of outputs, includ-

*Research was supported in part by a research grant and a CRD grant from the Natural Sciences and Engineering Research Council of Canada, a grant NCE:IRIS/Precarn from the Networks of Centres of Excellence of Canada, and grants from B.C. Advanced Systems Institute, MPR Teltech Ltd., National Research Council of Canada, and Hughes Research Laboratories.

ing crosstabs, bar/pie charts, curves, classification trees, multiple forms of generalized rules, visual presentation of rules, etc.

3. Efficient implementation techniques have been explored using different data structures, including multiple-dimensional data cubes and generalized relations. The implementations have been integrated smoothly with relational database systems and data warehouses.
4. The data mining process may utilize user- or expert-defined set-grouping or schema-level concept hierarchies which can be specified flexibly, adjusted dynamically based on data distribution, and generated automatically for numerical attributes. Concept hierarchies are being taken as an integrated component of the system and are stored as a relation in the database.
5. The system adopts a client/server architecture and is running on Windows/NT system. It communicates with various commercial database systems for data mining using the ODBC technology.

The system has been tested on several medium to large relational databases, including NSERC (Natural Science and Engineering Research Council of Canada) research grant information system, and U.S. City-County Data Book, with satisfactory performance. Additional data mining modules are being designed and will be added incrementally to the system along with the progress of our research.

2 Architecture and Functionalities

The general architecture of `DBMiner`, shown in Figure 1, tightly integrates a relational database system, such as a Microsoft SQL/Server, with a concept hierarchy module, and a set of knowledge discovery modules. The discovery modules of `DBMiner`, shown in Figure 2, include characterizer, comparator, classifier, associator, meta-pattern guided miner, predictor, cluster analyzer, time series analyzer, and some planned future modules.

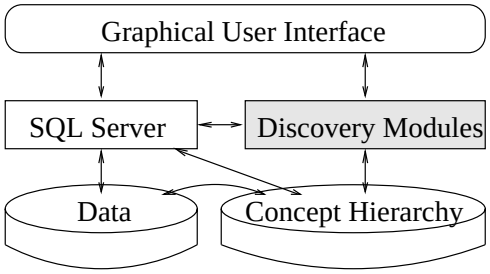

Figure 1: General architecture of `DBMiner`

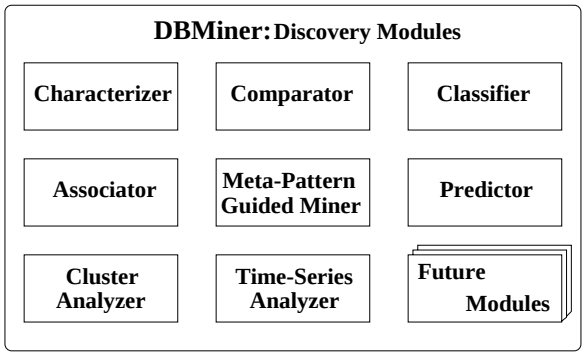

Figure 2: Knowledge discovery modules of `DBMiner`

The functionalities of the knowledge discovery modules are briefly described as follows:

- The *characterizer* 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.
- A *comparator* 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.
- A *classifier* 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.

- An *associator* 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.

- A *meta-pattern guided miner* is a data mining mechanism which takes a user-specified meta-rule form, such as "$P(x, y) \wedge Q(y, z) \rightarrow R(x, z)$" as a pattern to confine the search for desired rules. For example, one may specify the discovered rules to be in the form of "$major(s{:}student, x) \wedge P(s, y) \rightarrow gpa(s, z)$" in order to find the relationships between a student's major and his/her gpa in a university database.

- A *predictor* 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.

- A *cluster analyzer* 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 their house type, value, and geographical location.

- A *time-series analyzer* performs several kinds of data analyses for time-related data in the database or data warehouse, 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.

Another important function module of `DBMiner` is concept hierarchy which provides essential background knowledge for data generalization and multiple-level data mining. Concept hierarchies can be specified based on the relationships among database attributes (called *schema-level hierarchy*) or by set groupings (called *set-grouping hierarchy*) and be stored in the form of relations in the same database. Moreover, they can be adjusted dynamically based on the distribution of the set of data relevant to the data mining task. Also, hierarchies for numerical attributes can be constructed automatically based on data distribution analysis [7].

3 DMQL and Interactive Data Mining

`DBMiner` offers both an SQL-like data mining query language, `DMQL`, and a graphical user interface for interactive mining of multiple-level knowledge.

Example 1. To characterize CS grants in the *NSERC96* database related to discipline code and amount category in terms of count% and amount%, the query is expressed in `DMQL` as follows,

use NSERC96
mine characteristic rules as "CS_Discipline_Grants"
with respect to disc_code, amount,
count%, sum(amount)%
from award A, grant_type G
where A.grant_code = G.grant_code
and A.disc_code = "Computer Science"

The query is processed as follows: The system collects the relevant set of data by processing a transformed relational query, constructs

a multi-dimensional data cube, generalizes the data, and then presents the outputs in different forms, including generalized crosstabs, multiple (including visual) forms of generalized rules, pie/bar charts, curves, etc.

A user may interactively set and adjust various kinds of thresholds to control the data mining process. For example, one may adjust the generalization threshold for an attribute to allow more or less distinct values in this attribute. A user may also *roll-up* or *drill-down* the generalized data at multiple levels of abstraction. □

A data mining query language such as `DMQL` facilitates the standardization of data mining functions, systematic development of data mining systems, and integration with commercial relational database systems. Various kinds of graphical user interfaces can be developed based on such a data mining query language. A graphical user interface facilitates interactive specification and modification of data mining queries, concept hierarchies, and various kinds of thresholds, selection and change of output forms, roll-up or drill-down, and dynamic control of a data mining process. Such interfaces have been implemented in `DBMiner`.

4 Implementation of DBMiner

4.1 Data structures: Generalized relation vs. multi-dimensional data cube

Data generalization is a core function of `DBMiner`. Two data structures, *generalized relation*, and *multi-dimensional data cube*, are considered in the implementation of data generalization.

A *generalized relation* is a relation which consists of a set of (generalized) attributes (storing generalized values of the corresponding attributes in the original relation) and a set of "aggregate" (*measure*) attributes (storing the values resulted from executing aggregate functions, such as *count*, *sum*, etc.), and in which each tuple is the result of generalization of a set of tuples in the original data relation. For example, a generalized relation *award* may store a set of tuples, such as "$award(AI, 20\_40k, 37, 835900)$", which represents the generalized data for discipline code is "AI", the amount category is "20_40k", and such kind of data takes 37 in count and $835,900 in (total) amount.

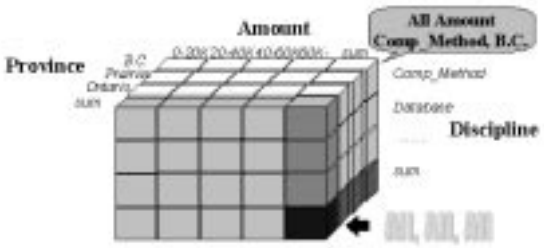

Figure 3: A multi-dimensional data cube

A *data cube* can be viewed as a multi-dimensional array structure, as shown in Figure 3, in which each dimension represents a generalized attribute and each cell stores the value of some aggregate attribute, such as *count*, *sum*, etc. For example, a cube *award* may have two dimensions: "discipline code" and "amount category". The value "AI" in the "discipline code" dimension and "20-40k" in the "amount category" dimension locate the corresponding values in the two aggregate attributes, *count* and *sum*, in the cube. Then the values, *count%* and *sum(amount)%*, can be derived easily.

In comparison with the generalized relation structure, a multi-dimensional data cube structure has the following advantages: First, it may often save storage space since only the measurement attribute values need to be stored in the cube and the generalized (dimensional) attribute values will serve only as dimensional indices to the cube; second, it leads to fast access to particular cells (or slices) of the cube using indexing structures; third, it usually costs less to produce a cube than a generalized relation in the process of generalization since the right cell in the cube can be located easily. When a cube structure is quite sparse, the sparse cube technology [1, 18] should be applied to compute and store sparse cube efficiently.

Both data structures have been implemented in the evolution of the `DBMiner` system: the

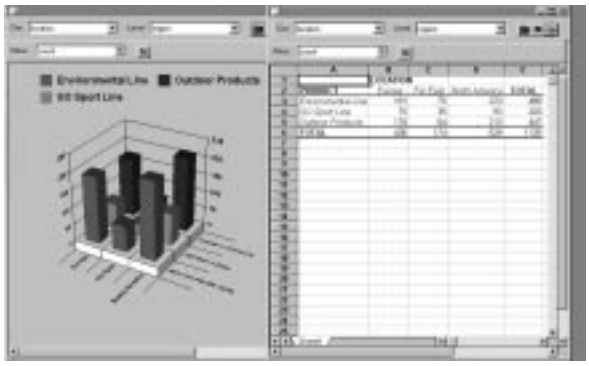

Figure 4: A snapshot of the output of `DBMiner` Characterizer

generalized relation structure is adopted in version 1.0, and a multi-dimensional data cube structure without using sparse cube technology in version 2.0. The version 3.0 adopts sparse cube technology in the cube construction and switches between array-based cube and relation-based cube according to its size to ensure good performance in databases and data warehouses with different sizes.

Besides designing good data structures, efficient implementation of each discovery module has been explored, as discussed below.

4.2 Multiple-level characterization

As shown in Figure 4, data characterization summarizes and characterizes a set of task-relevant data, usually based on 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 slightly higher abstraction levels than the primitive data in the relation or 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 cube/relation, 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 step down to specialized cases. However, from the implementation point of view, it is easier to perform generalization than specialization because generalization replaces low level tuples by high ones through ascension of a concept hierarchy. Since generalized tuples do not register the detailed original information, it is difficult to get such information back when specialization is required later.

Our technique which facilitates specializations on generalized relations is to save a "*minimally generalized relation/cube*" in the early stage of generalization. That is, each attribute in the relevant set of data is generalized to minimally generalized concepts (which can be done in one scan of the data relation) and then identical tuples in such a generalized relation/cube are merged together, which derives the minimally generalized relation. 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 generalizing the minimally generalized relation/cube to appropriate level(s).

4.3 Discovery of discriminant rules

The *comparator* of `DBMiner` finds a set of discriminant rules which distinguishes the general features of a target class from that of 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 relation/cube*, where a *prime target relation* is a generalized relation in which each attribute contains no more than but close to the threshold value of the corresponding attribute. Then the

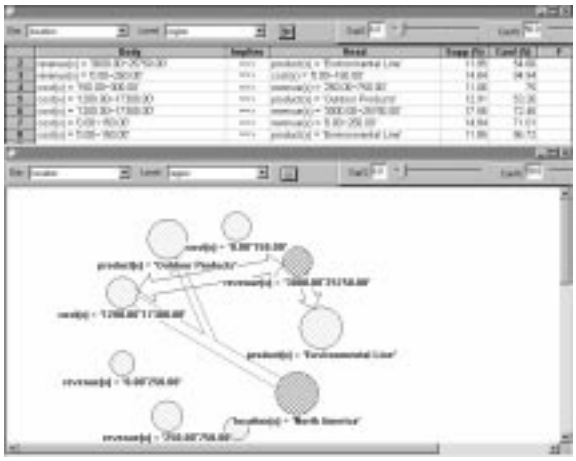

Figure 5: Graphic output of the Associator of DBMiner

concepts in the *contrasting* class(es) are generalized to the same level as those in the prime target relation/cube, forming the *prime contrasting relation/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. These functions have been implemented in the discriminator.

4.4 Multiple-level association

Based on many studies on efficient mining of association rules [2, 17, 8], a multiple-level association rule miner (called "associator") has been implemented in DBMiner. An output of the associator is shown in Figure 5.

Different from mining association rules in transaction databases, a relational 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 one or 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 the attributes "$(course, semester, grade)$" 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 following is an inter-attribute association: the association between *course* and *grade*, such as "*the courses in computing science tend to have good grades*", etc.

Two associations require different data mining techniques.

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,

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

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 [2, 8], can be applied to such transformed relations in relational databases.

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 measurements such as *support* and *confidence* of association rules based on the values in these summary cells. A set of such cubes, ranging from the least generalized cube to rather high level cubes, facilitate mining of association rules at multiple levels of abstraction. □

4.5 Meta-rule guided mining

Since there are many ways to derive association rules in relational databases, it is preferable to have users to specify some interesting constraints to guide a data mining process. Such constraints can be specified in a *meta-rule* (or *meta-pattern*) form [16], 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.

In principle, a meta-rule can be used to guide the mining of many kinds of rules. Since the association rules are in the form similar to logic rules, we have first studied meta-rule guided mining of association rules in relational databases [5]. Different from the study by [16] where a meta-predicate may match any relation predicates, deductive predicates, attributes, etc., we confine the match to only those predicates corresponding to the *attributes* in a relation. One such example is illustrated as follows.

Example 3. A meta-rule guided data mining query can be specified in `DMQL` as follows for mining a specific form of rules related to a set of attributes: "*major, gpa, status, birth_place, address*" in relation *student* for those born in Canada in a *university* database.

mine associations
with respect to major, gpa, status, birth_place, address
from student
where birth_place = "Canada"
set rule template
$$major(s : student, x) \wedge Q(s, y) \rightarrow R(s, z)$$

Multi-level association rules can be discovered in such a database, as illustrated below:

$$major(s, \text{“}Science\text{”}) \wedge gpa(s, \text{“}Excellent\text{”}) \rightarrow status(s, \text{“}Graduate\text{”}) \quad (60\%)$$
$$major(s, \text{“}Physics\text{”}) \wedge status(s, \text{“}M.Sc\text{”}) \rightarrow gpa(s, \text{“}3.8\_4.0\text{”}) \quad (76\%)$$

The mining of such multi-level rules can be implemented in a similar way as mining multiple-level association rules in a multi-dimensional data cube [11]. □

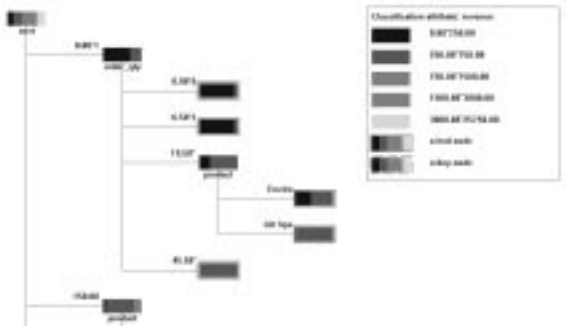

Figure 6: Classifier of `DBMiner`

4.6 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 data classification methods studied, including decision-tree methods, such as ID-3 and C4.5 [15], statistical methods, neural networks, rough sets, etc. Recently, some database-oriented classification methods have also been investigated [13].

Our classification method consists of four steps: (1) collection of the relevant set of data and partitioning of the data into training and testing 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. Several top-most relevant attributes

retain 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 attribute-oriented induction and OLAP data cube technology with a decision-tree induction technique, by first performing minimal generalization on the set of training data to generalize attribute values in the training set, and then performing decision tree induction on the generalized data.

Since a generalized tuple comes from the generalization of a number of original tuples, the *count* information is associated with each generalized tuple 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 helps justification of the classification at a node when a significant set of the examples belong to the same class; whereas the latter helps ignore a node in classification if it 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.

Currently, we are testing several alternatives at integration of generalization and classification in databases, such as (1) generalize data to some medium abstraction levels; (2) generalize data to intermediate abstraction level(s), and then perform node merge and split for better class representation and classification accuracy; and (3) perform multi-level classification and select a desired level by a comparison of the classification quality at different levels. Since all three classification processes are performed in relatively small, compressed, generalized relations, it is expected to result in efficient classification algorithms in large databases.

The generalization-based multi-level classification process has been implemented in the DBMiner system. An output of the DBMiner classifier is shown in Figure 6.

4.7 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 simply be 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 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 predicting 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 predicting 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 predicting attributes

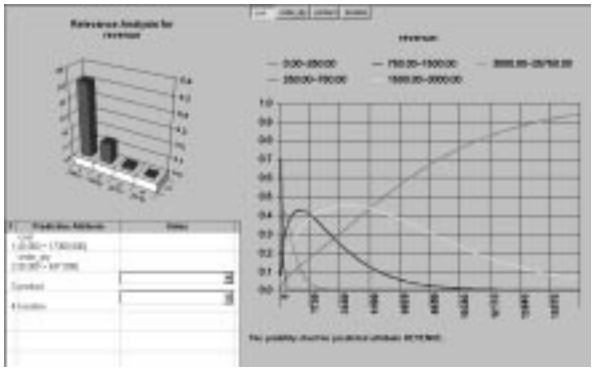

Figure 7: Prediction output when the predicting attribute is a numeric one

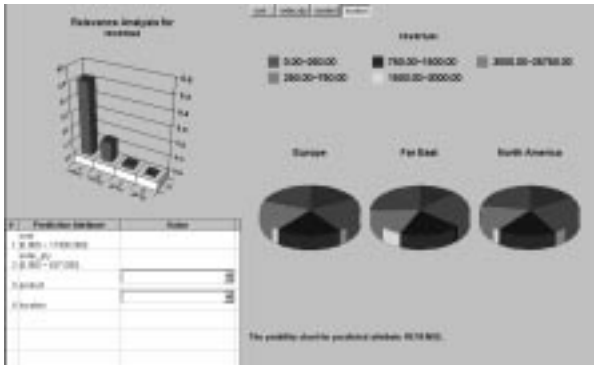

Figure 8: Prediction output when the predicting attribute is a categorical one

can be used for trustable prediction.

Figure 7 shows the prediction output when the predicting attribute is a numeric one; whereas Figure 8 shows the prediction output when the predicting attribute is a categorical one.

4.8 Clustering

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 can not 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.

5 Further Development of DBMiner

The `DBMiner` system is currently being extended in several directions, as illustrated below.

- Further enhancement of the power and efficiency of data mining in relational database

systems and data warehouses, 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.

- Further enhancement of the performance of data mining in large databases and data warehouses by exploration of parallel processing using NT clusters. Some algorithm design and experimental work have been performed in this direction.

- Integration, maintenance and application of discovered knowledge, including incremental update of discovered rules, removal of redundant or less interesting rules, merging of discovered rules into a knowledge-base, intelligent query answering using discovered knowledge, and the construction of multiple layered databases.

- Extension of data mining technique towards advanced and/or special purpose database systems, including extended-relational, object-oriented, textual, spatial, temporal, multi-media, and heterogeneous databases and Internet information systems. Currently, three such data mining systems, *GeoMiner*, *LibMiner*, and *WebMiner*, for mining knowledge in spatial databases, library databases, and the Internet information-base respectively, are being under design and construction.

About the Authors

- **Jiawei Han** (Ph.D. Univ. of Wisconsin at Madison, 1985), Professor of the School of Computing Science and Director of Intelligent Database Systems Research Laboratory, Simon Fraser University, Canada. He has conducted research in the areas of data mining and data warehousing, deductive and object-oriented databases, spatial databases, multimedia databases, and logic programming, with over 100 journal and conference publications. He is a project leader of the Canada NCE/IRIS:HMI-5 project ("data mining and knowledge discovery in large databases"), and has served or is currently serving in the program committees of over 30 international conferences and workshops, including ICDE'95 (PC vice-chair), DOOD'95, ACM-SIGMOD'96, VLDB'96, KDD'96 (PC co-chair), CIKM'97, SSD'97, KDD'97, and ICDE'98. He has also been serving as an editor for IEEE Transactions on Knowledge and Data Engineering, Journal of Intelligent Information Systems, and Data Mining and Knowledge Discovery.

- **Jenny Y. Chiang** received her B.Sc. degree in Computing Science at Simon Fraser University. Currently, she is working on the design and implementation of the DBMiner system. Her research interests are data cube construction, OLAP implementation, and association rule mining.

- **Sonny Chee** received his MASc. degree in Aerospace at University of Toronto. Currently, he is a Ph.D. candidate of School of Computing Science, Simon Fraser University. He is working on the design and implementation of the DBMiner System. His research interests are active database mining, OLAP implementation, and time-series data mining.

- **Jianping Chen** received his B. Sc. in Mathematics in the University of Science and Technology of China in 1987, and M.Sc. degree in Mathematics at Simon Fraser University in 1996. He worked in the Institute of Applied Mathematics, Chinese Academy of Sciences, Beijing, China from 1987 to 1994. Currently, he is a M.Sc. student in the School of Computing Science, Simon Fraser University. He is working on parallel database mining, and data classification techniques.

- **Qing Chen** received her B.Sc. degree in Computer Science from the University of Science and Technology of China, in 1993. She is currently working on the M.Sc. degree at School of Computing Science, Simon Fraser University. Her research interests include spatial data mining.

- **Shan Cheng** received a B.Sc. degree in statistics from Fudan University, China, and an M.Sc. degree in statistics from Simon Fraser University. Currently, she is involved in the research and implementation of DBMiner system. Her research interests are data mining, data warehousing, and data analysis.

- **Wan Gong** received her B.Sc. degree in Math/Computer Science at Mount Allison University, N.B. Currently, she is working on a M.Sc. Degree in Computing Science, Simon Fraser University, and is involved in the research and development of the DBMiner System. Her research interest is time-related data mining.

- **Micheline Kamber** is a doctoral candidate in Computing Science at Simon Fraser University. Her research interests in data mining include data cube-based mining of association rules, metarule-guided mining, pattern interestingness, and the development of a data mining query language.

- **Krzysztof Koperski** received the M.Sc. degree in Electrical Engineering from Warsaw University of Technology, Poland. He is currently Ph.D. student in Computer Science at Simon Fraser University. His research interests are spatial data mining, spatial reasoning, and data warehousing.

- **Gang Liu** is a Ph.D. student at School of Computing Science at Simon Fraser University. His research interests include visual data mining. Gang received an M.Sc. in Computer Science from National University of Singapore.

- **Yijun Lu** received his B.Sc. degree in applied mathematics from Huazhong Univ. of Sci. and Tech., China, in 1985 and an M.Sc. degree in applied and computational mathematics from Simon Fraser University, in 1995. He is currently a M.Sc. student in the School of Computing Science. His research interests include knowledge discovery, data mining and scientific computing.

- **Nebojsa Stefanovic** is an M.Sc. student at School of Computing Science at Simon Fraser University. His major interests are in spatial data warehousing and spatial data mining. He has implemented several visualization tools for the DBMiner, and the OLAP component for the GeoMiner system. Stefanovic received a B.Sc. in computer engineering from University of Belgrade, Yugoslavia.

- **Lara Winstone** is currently a M.Sc. student in Computing Science at Simon Fraser University. She received her B.Sc. (Honours Co-op) from the University of Manitoba. Her research interests include classification, domain analysis, and case based reasoning.

- **Betty Bin Xia** received her M.Sc. degree in Computer & Application from Jilin University, China, in 1993. She is currently working on the M.Sc. degree at School of Computing Science, Simon Fraser University. Her research interests include data mining, user interface design and implementation, and the similarity mining in time-related databases.

- **Osmar R. Zaiane** received the M.Sc. degree in Computer Science from Laval University, Canada. He is currently a Ph.D. candidate in Computer Science at Simon Fraser University. His research interests are WWW technology, data mining, multimedia databases, and Web-based data mining systems.

- **Shuhua Zhang** received his Ph.D. degree in mathematics from Simon Fraser University in 1992. He is currently pursuing his M.Sc. degree in computing science at Simon Fraser University. His research interests include data mining and universal algebra.

- **Hua Zhu** is currently an M.Sc. student of School of Computing Science at Simon Fraser University. Her research interests include database mining, user interface, especially the web mining. Hua received a B.S. in Computing Science from University of Sci. & Tech. of China.

References

[1] S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. In *Proc. 1996 Int. Conf. Very Large Data Bases*, pages 506–521, Bombay, India, Sept. 1996.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases*, pages 487–499, Santiago, Chile, September 1994.

[3] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26:65–74, 1997.

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

[5] Y. Fu and J. Han. Meta-rule-guided mining of association rules in relational databases. In *Proc. 1st Int'l Workshop on Integration of Knowledge Discovery with Deductive and Object-Oriented Databases (KDOOD'95)*, pages 39–46, Singapore, Dec. 1995.

[6] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Trans. Knowledge and Data Engineering*, 5:29–40, 1993.

[7] J. Han and Y. Fu. Dynamic generation and refinement of concept hierarchies for knowledge discovery in databases. In *Proc. AAAI'94 Workshop on Knowledge Discovery in Databases (KDD'94)*, pages 157–168, Seattle, WA, July 1994.

[8] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proc. 1995 Int. Conf. Very Large Data Bases*, pages 420–431, Zurich, Switzerland, Sept. 1995.

[9] J. Han and Y. Fu. Exploration of the power of attribute-oriented induction in data mining. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 399–421. AAAI/MIT Press, 1996.

[10] J. Han, Y. Fu, W. Wang, J. Chiang, W. Gong, K. Koperski, D. Li, Y. Lu, A. Rajan, N. Stefanovic, B. Xia, and O. R. Zaiane. DBMiner: A system for mining knowledge in large relational databases. In *Proc. 1996 Int'l Conf. on Data Mining and Knowledge Discovery (KDD'96)*, pages 250–255, Portland, Oregon, August 1996.

[11] M. Kamber, J. Han, and J. Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. In *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96)*, pages 207–210, Newport Beach, California, August 1997.

[12] M. Kamber, L. Winstone, W. Gong, S. Cheng, and J. Han. Generalization and decision tree induction: Efficient classification in data mining. In *Proc. of 1997 Int. Workshop on Research Issues on Data Engineering (RIDE'97)*, pages 111–120, Birmingham, England, April 1997.

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

[14] G. Piatetsky-Shapiro and W. J. Frawley. *Knowledge Discovery in Databases*. AAAI/MIT Press, 1991.

[15] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[16] W. Shen, K. Ong, B. Mitbander, and C. Zaniolo. Metaqueries for data mining. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 375–398. AAAI/MIT Press, 1996.

[17] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. 1995 Int. Conf. Very Large Data Bases*, pages 407–419, Zurich, Switzerland, Sept. 1995.

[18] Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. In *Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data*, pages 159–170, Tucson, Arizona, May 1997.