Data Min Knowl Disc (2007) 15:29–54
DOI 10.1007/s10618-007-0063-0

Exploratory mining in cube space

Raghu Ramakrishnan · Bee-Chung Chen

Receive: 7 June 2006/Accepted: 4 January 2007 / Published online: 2 May 2007

Abstract Data Mining has evolved as a new discipline at the intersection of several existing areas, including Database Systems, Machine Learning, Optimization, and Statistics. An important question is whether the field has matured to the point where it has originated substantial new problems and techniques that distinguish it from its parent disciplines. In this paper, we discuss a class of new problems and techniques that show great promise for exploratory mining, while synthesizing and generalizing ideas from the parent disciplines. While the class of problems we discuss is broad, there is a common underlying objective—to look beyond a single data-mining step (e.g., data summarization or model construction) and address the combined process of data selection and transformation, parameter and algorithm selection, and model construction. The fundamental difficulty lies in the large space of alternative choices at each step, and good solutions must provide a natural framework for managing this complexity. We regard this as a grand challenge for Data Mining, and see the ideas discussed here as promising initial steps towards a rigorous exploratory framework that supports the entire process.

Responsible editor: Geoffrey Webb.

Bee-Chung Chen is supported by a Microsoft Research graduate fellowship.

R. Ramakrishnan (✉) · B.-C. Chen
Department of Computer Science, University of Wisconsion, 1210 W. Dayton street, Mdison, WI 53706, USA
e-mail: ramakris@yahoo-inc.com

B.-C. Chen
e-mail: beechung@cs.wisc.edu

Keywords Data mining · Exploratory analysis · OLAP · Cube · Feature space · Multidimensional data model

1 Introduction

The Data Mining community seeks to build on ideas from several disciplines, including Databases, Machine Learning, Optimization, and Statistics, to address the grand challenge of understanding and learning from massive datasets. Many interesting concepts, e.g., frequent itemsets (Agrawal 1993), have been introduced, and many scalable algorithms have been developed for previously studied problems such as clustering and decision tree construction. However, analyzing and mining massive databases requires more than just efficient and scalable algorithms; we need principled ways to explore a large space of alternatives in how to transform, subset, and analyze a dataset. For example, consider building a classification model (e.g., a decision tree) from a given database. It is well known that preparing the data, which involves identifying interesting and useful subsets of data, aggregating data at the right granularity, and creating the class labels, is a time-consuming and critical step. In this preparation step, there are a huge number of possibilities to consider, including different subsets, different aggregate granularities, and different ways to create the labels that we wish to learn to predict (including different label semantics). Further, each possibility leads to a different classification model in the model-construction step. How to allow analysts to explore these choices in a principled manner that automates routine steps and enables them to consider large parts of the search space with minimum effort is wide open, and addresses what is arguably the biggest bottle-neck in data-mining projects, namely analyst bandwidth. While this observation holds in general, it is especially the case for large datasets, where manual consideration of the choices is difficult, even impossible. Unfortunately, most current work has concentrated on algorithms for the data summarization or model-construction steps, and discussions of the process and the choices in mining have remained at a qualitative level.

In this paper, we focus on the problem of how to handle the huge space of possibilities. We describe a promising paradigm called *cube-space data mining*, which can be intuitively thought of as the integration of OLAP-style multidimensional data analysis (OLAP stands for On-Line Analytical Processing, which will be discussed in Section 2.2) with techniques from machine learning, statistical modeling, etc. The basic idea is to *use multiple structural dimensions as a conceptual way to organize data of interest into intuitive regions at various granularities, and then analyze and mine the data by applying model-building and summarization techniques systematically over these regions at varying granularities*. In this process, regions and granularities intuitively define the space of possible dataset generation scenarios. Then, data-mining models (e.g., models for classification, regression, clustering, association analysis, time-series analysis, etc) are built systematically for each possible scenario to find interesting patterns and trends. The patterns and trends that we are interested in are those

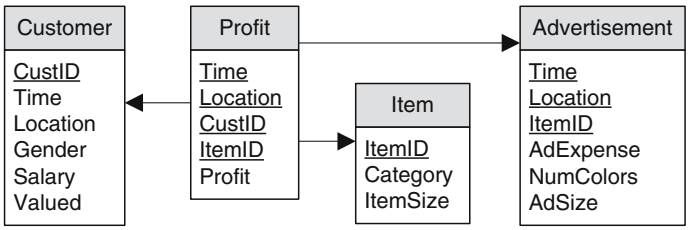


Fig. 1 The schema of the example database

that can only be identified by considering of the interplay between dataset generation and model construction. The goal of this paper is to illustrate this idea by describing many instances, and to make the case that it is a promising direction for future research with great potential value and many open problems.

In the following, we first describe a running example and provide background knowledge in Section 2, and then introduce cube-space data mining in Section 3. In Section 4, prediction cubes are used as a detailed example to illustrate the ideas of cube-space data mining. Then, in Section 5, other instances of cube-space data mining are briefly discussed. Finally, we conclude the paper in Section 6.

2 Preliminaries

In this section, we provide the background for the discussion in the rest of the paper. We first introduce a simple example database, which will be used to illustrate many ideas in later sections, and then review the key elements of the multi-dimensional data model used in OLAP systems, and finally the basics of predictive models.

2.1 Example database

Let us consider a store chain that has worldwide sales and maintains its customer information. In its data warehouse, there are four tables, which attributes (columns) are shown in Fig. 1 (in which the combination of the underlined attributes in each table forms a key of that table). The Customer table contains customer information including the customer ID, the time and location that he/she becomes a customer, his/her gender and salary, and whether he/she is a valued customer, which is determined by each store. Each tuple (row) in the Profit table records the profit earned from a particular customer's purchase of an item at a specific time and location. Item information is stored in the Item table, which contains the category and the size of each item. Advertisement information (i.e., the expense, the number of colors used and the size of each advertisement) is stored in the Advertisement table, where each advertisement is identified by the combination of a time, a location and an item ID.

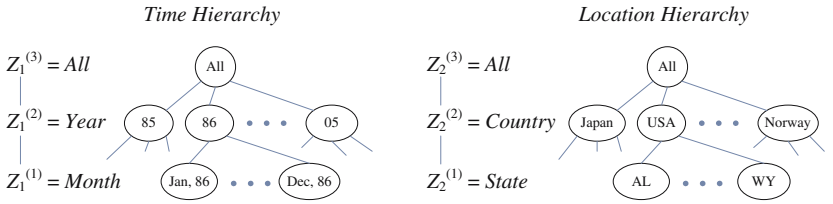


Fig. 2 The domain hierarchies of time and location

Some of the attributes have associated domain hierarchies. Fig. 2 shows the domain hierarchies of attributes Time and Location. For example, the time hierarchy contains three domains. From the lowest one to the highest one, they are Month, Year and All. The Category attribute in the Item table may also have an associated domain hierarchy, but we omit it here. For simplicity, we only discuss tree-structured hierarchies, but in general the domains of an attribute can be structured as a lattice (See Harinarayan et al. 1996 for details). Also, we assume that the values of any attribute (e.g., Time) recorded in any table are from the domain at the lowest level (i.e., finest granularity) of the hierarchy (e.g., Month).

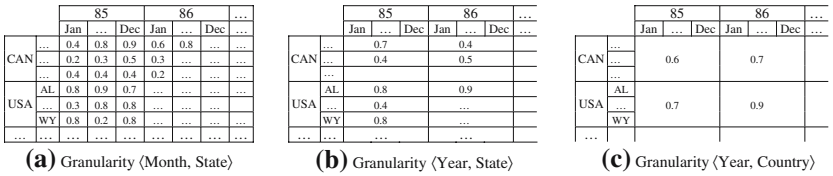
In the following subsections, we will first focus on the analysis using only the Profit table, and then discuss analyses using other tables.

2.2 Multi-dimensional data model and OLAP

In OLAP (On-Line Analytical Processing), the multi-dimensional data model is defined on a *fact table* **T** (e.g., the Profit table). On the fact table, attributes are divided into *dimension attributes* Z_1, …, Z_d (e.g., Time, Location, CustID, and ItemID) and *measure attributes* M_1, …, M_k (e.g., Profit). Each dimension attribute is associated with a **domain hierarchy**[1] $Hier(Z_i) = (Z_i^{(1)}, \ldots, Z_i^{(n)})$, where $Z_i^{(1)}$, …, $Z_i^{(n)}$ are the domains from the lowest level to the highest level (e.g., Month, Year, All for the Time attribute). We call $\mathrm{Dom}(Z_i) = \cup_{j=1,\ldots,n} Z_i^{(j)}$ the extended domain of attribute Z_i, which includes all the values in the domain hierarchy (e.g., Dom(Time) = {"1985/01", …, "2005/12"} ∪ {"1985", …, "2005"} ∪ {"All"}). We call $\mathbf{C} = \mathrm{Dom}(Z_1) \times \ldots \times \mathrm{Dom}(Z_d)$ the *cube space* spanned by Z_1, …, Z_d, and call each element $\boldsymbol{r} = [z_1, \ldots, z_d] \in \mathbf{C}$ a **region**. Each region is associated with a **granularity**: $gran(\boldsymbol{r}) = \langle g_1, \ldots, g_d \rangle$, where g_i is the domain of value z_i. Some examples of regions in the cube space spanned by Time and Location are as follows: region [2005, USA/AL] at granularity ⟨Month, State⟩, region [2005, USA] at granularity ⟨Year, Country⟩, and region [All, All] at granularity ⟨All, All⟩.

In regular OLAP, we are interested in aggregate numbers (e.g., total Profit) of the regions in cube space. An aggregate number of region $\boldsymbol{r}$ is obtained by

[1] Usually, domain hierarchies are defined by dimension tables, which are not shown in Fig. 1.



(a) Granularity ⟨Month, State⟩ **(b)** Granularity ⟨Year, State⟩ **(c)** Granularity ⟨Year, Country⟩

Fig. 3 An example data cube at different granularities

applying an aggregate function to a measure attribute (e.g., sum(Profit) over all the records in region $\boldsymbol{r}$). Given an aggregate function, a measure attribute and a set of dimension attributes, the set of all aggregate numbers generated by applying the aggregate function to the measure attribute for all the regions in the cube space spanned by the dimensions is called a *data cube*.

Figure 3 shows an example data cube at different granularities, where each cell value represents the average profit per customer in a region (the unit is thousand). The operation that changes a cube from a fine granularity to a coarse one (e.g., from ⟨Year, State⟩ to ⟨Year, Country⟩) is called *rollup*. The operation that changes a cube from a coarse granularity to a fine one (e.g., ⟨Year, State⟩ to ⟨Month, State⟩) is called *drilldown*. By using rollup and drilldown in a cross-tab-style interface, analysts can navigate through the space of interest and find interesting patterns. We call this kind of analysis the *OLAP analysis* (see Gray et al. 1997 for details).

2.3 Basics of predictive models

We now review some basics of predictive models (see Mitchell 1997 for details). Let **D** be a table of data (e.g., the Customer table) with attributes $X_1, \ldots, X_p$, Y, where $X_1, \ldots, X_p$ are called *features* (e.g., Gender and Salary), Y is called the *target* or label attribute (e.g., Valued), and each row in the table is called an *example*. We use $\boldsymbol{X} = \{X_1, \ldots, X_p\}$ to denote the set of features. A predictive model learns the relationship between $\boldsymbol{X}$ and Y from **D** and can predict the Y value of a new example based on its $\boldsymbol{X}$ value. **D** is called the **training set**. We use $h(\boldsymbol{X}; \mathbf{D})$ to denote a predictive model trained on **D** using $\boldsymbol{X}$ to predict Y, and $h(\boldsymbol{x}; \mathbf{D})$ returns the target value of example $\boldsymbol{x}$. If the training set can be inferred from the context, we use $h(\boldsymbol{X})$ and $h(\boldsymbol{x})$ for simplicity. If the target Y is a numeric value, h is called a *regression model*. If Y is a categorical value, h is called a *classification model*. Decision trees, support vector machines, neural networks and linear regression models are examples of predictive models.

The quality of a predictive model is usually measured by the error (or equivalently, accuracy) of the model, which is the expected discrepancy between the true target value and the predicted value for a new example. For classification models, the misclassification rate (i.e., the expectation of making an incorrect prediction) is a commonly used error measure, while for regression models, the mean squared error (MSE) and root mean squared error (RMSE) are commonly used. MSE is the expected value of the squared difference between the

true Y value and the predicted one, and RMSE is the square root of MSE. However, in reality, the true distribution of $(\boldsymbol{X}, Y)$ is generally unknown. Thus, the error of a model cannot be computed exactly, but needs to be estimated from the given data. Three commonly used error estimates are test-set error (which uses an additional set of test data), and cross-validation error and training-set error (which do not use any additional data).

- *Test-set error:* In addition to $\mathbf{D}$, given a test set Δ with attributes $(\boldsymbol{X}, Y)$, in which no example is used to train model h, the test-set error of model h is the percentage of examples in Δ that are incorrectly classified by h.
- *Cross-validation error:* To compute the cross-validation error, we first partition $\mathbf{D}$ into n non-overlapping subsets of examples: $\mathbf{D}_1, \ldots, \mathbf{D}_n$. For i from 1 to n, we train a model on $\cup_{j \neq i} \mathbf{D}_j$ and test the model on $\mathbf{D}_i$ to obtain an error value. Then, the cross-validation error is the mean of the n error values. A commonly used n is 10.
- *Training-set error:* Another way to estimate the error of a model is to train the model on $\mathbf{D}$, and then test it also on $\mathbf{D}$ to obtain the error value, which is called the training-set error. Usually the training-set error is overly optimistic. However, for simple models, e.g., linear regression models, the training-set error can approximate the true error.

3 Cube-space data mining

Intuitively, cube-space data mining seeks to fuse OLAP-style analysis and data mining. The basic idea is to view the data of interest as points in a multi-dimensional space and to systematically analyze it at multiple granularities using data-mining models, such as models for classification, regression, clustering, association analysis, time-series analysis, etc., as building blocks. In this section, we first discuss the benefits of fusing OLAP-style analysis and data mining, and then introduce the characteristics and challenges of cube-space data mining. In later sections, we illustrate the points made in this section with additional examples of this fusion.

3.1 Motivation

To understand the value in fusing OLAP-style analysis and data mining, it is useful to consider how these traditional approaches complement each other. The OLAP multi-dimensional data model offers:

- *A natural collection of data subsets:* Many data-mining tasks involve the identification of interesting subsets from a large dataset. This search for interesting subsets is sometimes a part of data preparation and sometimes a way to better understand the data. However, even for a modest dataset, the number of possible subsets is exponentially large, and not all of these subsets make intuitive sense. Thus, it is better to focus on a feasible-to-analyze collection of meaningful subsets. Cube space offers such a collection. Each

region in cube space defines a meaningful data subset that can be easily understood (e.g., [2005, USA]). Also, the nested structure of cube space (e.g., [2005/01, USA] is a sub-region of [2005, USA]) provides opportunities for efficient computation, e.g., computation sharing and pruning.

- *A framework for multi-granularity aggregation:* Data at different granularities usually has different behavior. For example, we do not expect the monthly profit of a company to have the same behavior as the annual profit of an individual product. Thus, the ability to explore data behavior at multiple granularities is important to certain kinds of mining tasks. Also, from another viewpoint, the volume of today's dataset is usually very large. Aggregation provides a form of data reduction. However, the right granularity for aggregation is usually not known in advance. The multi-dimensional data model provides a natural framework to systematically explore patterns in data aggregated at different granularities.
- *An interface for exploring data-mining results:* The data cube interface (which supports spreadsheet-style rollup, drilldown, pivoting, etc. through a simple GUI) has been widely used and proven to be easily understood. When we use the multi-dimensional data model to structure the data subsets to which we apply mining techniques, the same natural interface can be used to display the mining results, with appropriate extensions.
- *A mechanism for implementing privacy policies:* Privacy in data mining has received increasing attention. Rollup and aggregation provide ways to prevent individuals' private and sensitive information from being revealed. For example, we can rollup a table to a certain granularity so that no tuple in the table can be distinguished from a group of k. This protection is called k-anonymity (Samarati and Sweeney 1998). Related forms of privacy protection may also be achieved using rollup and aggregation (Dobra and Fienberg 2001; Machanavajjhala et al. 2002). While we want to protect privacy, we also want the data to be useful for mining. How to find a good balance between privacy and utility (Kifer and Gehrke 2006; LeFevre et al. 2006) is still an open problem. Cube-space data mining may provide ways to understand the tradeoff between privacy and utility, and ways to achieve privacy-preserving mining.

On the other hand, data-mining techniques also complement OLAP-style analysis:

- *The ability to analyze prediction or decision behavior:* In regular OLAP, analyses are based on simple aggregate numbers (e.g., count, sum, average) over regions in a cube space. With the power of data-mining tools, we can extend OLAP analysis to prediction or decision analysis (e.g., Chen, B.-C., et al., 2005a). For example, by appropriate use of data-mining techniques (details will be given in the next section), we can make the value in each cell of a data cube represent how important Gender is to the decision of whether a customer is a valued one. Then, we can answer questions such as: "At what times and locations did the decision making exhibit gender discrimination?"

- *Ways to handle uncertain and incomplete data:* In some applications, we may not have exact or complete data. Data-mining techniques can be used to handle uncertainty and imprecision, and thereby allow us to conduct reliable rollup and drilldown analysis (e.g., Burdick et al., 2005). An example is OLAP analysis on information extracted from text.
- *Methods to identify interesting regions:* In OLAP analysis, cube space may contain a huge number of regions. Manual rollup and drilldown may not be sufficient to identify all interesting regions. Data-mining techniques can help analysts find interesting regions (e.g., Sarawagi, 1999, 2000) in data cubes.

3.2 Characteristics of cube-space data mining

The idea of combining OLAP analysis and data mining is not new. For example, Han (1998) developed a system that uses data cubes and an OLAP engine to improve many data-mining tasks, such as concept description, association rules, classification, clustering and time-series analysis. Recent years have seen further advances on the idea of combining OLAP and data mining, which go well beyond using one to improve the other, and we believe this broad direction will continue to generate interesting and useful problems, analyses and techniques. We think of these advances as the results of deeper interplay between the two approaches. The following is an incomplete list of the different types of interaction between OLAP-style analysis and data-mining techniques that characterize cube-space data mining:

- *Using cube space to define the space of candidates for mining.* Each region in cube space represents a candidate. The candidates can be possible data generation scenarios, subsets of data over which we wish to find interesting patterns, or even time-series at various granularities. The use of cube space makes the space of candidates both meaningful and tractable (because cube space is defined by a set of informative dimension hierarchies, not just a set of arbitrary subsets of data).
- *Using OLAP queries to generate features and targets for mining.* Intuitively, each candidate defines a dataset over which we can build a data-mining model to describe the behavior of that candidate. The features and even targets (that we wish to learn to predict) of such a dataset sometimes can be naturally defined as OLAP aggregate queries over regions in cube space.
- *Using data-mining models as building blocks in a multi-step mining process.* In contrast to regular data-mining, where the goal is to build data-mining models, cube-space data mining usually consists of multiple steps, where data-mining models are building blocks used to describe the behavior of candidates, rather than the end results. Data-mining models may also be used in an unconventional way. For example, the predictiveness of a feature can be intuitively defined as the difference in accuracy between a model that uses all the features and a model that uses all the features except the chosen one. See Section 4.2 for details.

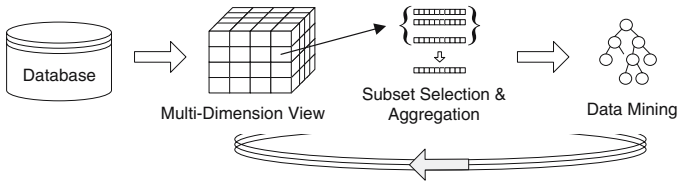


Fig. 4 The process of cube-space data mining

- *Using data-cube computation techniques to speed up repeated model construction.* In principle, cube-space data mining usually requires building a model for each candidate, which is usually too expensive to be feasible. However, by carefully sharing computation across model-construction for different candidates based on data-cube computation techniques, efficient mining is still achievable.

We call the research that results from such interplay between OLAP-style multi-dimensional analysis and data-mining techniques *cube-space data mining*. This term does not denote a specific mining task or algorithm; rather, it is a paradigm or a way of thinking that covers a broad class of problem formulations and algorithms.

The typical process of cube-space data mining is illustrated in Fig. 4. Starting from a relational database, we (conceptually) create one or more multi-dimensional views of the data of interest. Then, subsets of data are selected and aggregated at desired granularities. After that, data-mining methods (e.g., decision tree construction algorithms) are applied to the aggregated subsets and evaluated using various criteria. The key characteristic of cube-space data mining that distinguish it from regular data mining is that this process involves systematic evaluation of a large number of possibilities, such as candidate subsets (i.e., regions in cube space) and granularities, repeatedly using data-mining models. The mining results usually consist of interesting patterns found on different subsets of data at various granularities, and may be shown using a cube-like interface that support rollup and drilldown analysis.

3.3 Challenges

We now discuss the challenges posed by cube-space data mining.

- *Efficiency and scalability:* From the above discussion, it should be apparent that cube-space data mining poses great computational challenges. Constructing a single data-mining model on a large dataset can be expensive. Repeatedly evaluating a huge number of possibilities, each of which may involve model construction, makes the situation even worse. Thus, naïve computation approaches usually do not work, and the feasibility of any task of cube-space data mining is determined by whether efficient and scalable mining algorithms can be developed.

- *Compositionality:* In cube-space data mining, data-mining models are usually building blocks, rather than the end results. Thus, a cube-space mining task usually consists of multiple mining steps. How to represent such multi-step mining, how to optimize these steps for efficient execution, and how to manage multiple tasks in a unified framework are challenges central to the success of cube-space data mining.
- *Dimensional complexity:* When the number of dimensions increases, the size of the cube-space quickly becomes intractably large. However, real data-mining tasks usually involve many dimensions. Thus, the ability to identify useful dimensions and the ability to efficiently find interesting regions in a high dimensional space are very important in practice. The dimensional complexity comes not only from the number of explicitly listed data dimensions, but also from several additional sources:
  1. There is often a need to identify structure in the domains underlying dimension attributes when the hierarchies are not known before the analysis. For example, various kinds of time windows can be defined on a time dimension.
  2. We may want to define cube-space using non-data "attributes," such as parameters of mining algorithms (e.g., scaling factors for different data axes) and properties of mining models (e.g., the accuracy of models), in order to consider the full space of analysis "experiments".
- *Statistical significance:* Cube-space data mining usually involves finding interesting patterns over a large space of candidates. Although interesting patterns are usually rare events, when the number of candidates is large, rare events can occur just by chance. How to make sure that the patterns found do not overfit the data and how to justify that these patterns are statistically significant are challenges central to the usefulness of cube-space data mining.

4 Prediction cubes

We now use prediction cubes (Chen et al. 2005a) as a concrete example to illustrate the main ideas, challenges and techniques in cube-space data mining. Prediction cubes are a new tool for the identification of interesting subsets of data, where candidate subsets are defined by regions in cube space, and interestingness is defined using predictive models. A prediction cube is similar to a data cube. However, unlike a data cube in which each cell value is an aggregate number (e.g., count, sum and average) computed over the subset of data in that cell, each cell value in a prediction cube is computed by evaluating a predictive model built on the subset of data in that cell, thus representing the decision or prediction behavior of that subset. Predictive models are used in an unconventional way in prediction cubes. Instead of using them to make predictions, they are used to capture the behavior of regions in cube space, e.g., where there is gender discrimination in a decision process in a region. In the rest of this section, we first introduce a motivating example, define a type of prediction

cube to solve the problem in the motivating example, and then describe the computation techniques and point out some further research directions. For other types of prediction cubes, see Chen et al. 2005a.

4.1 Motivating example

Consider the company described in Section 2.1. The managers of the company want to analyze the decision process of whether a customer is highly-valued with respect to two dimensions: Time and Location. In particular, they are interested in the following question:

> Are there times and locations during which the value of a customer depended greatly on the customer's gender (or, in general, a set of attributes of interest)?

When predictive models built by training a machine-learning algorithm (on the Customer table in Fig. 1) are used to assist in such a decision process, the question essentially has to do with the predictiveness of Gender on different subsets of data. Social scientists have raised concerns that the use of data mining introduces the risk of discrimination that is hard to identify, because it is hidden in sophisticated models (see Danna and Gandy, 2002).

This question is also complicated by the fact that the candidate answers are *subsets* of data, partitioned by Time and Location values; clearly, there are a large number of candidates. Although the Time and Location hierarchies are known (e.g., Fig. 2), the right level (or granularity) for the analysis is unclear; e.g., is performing analysis using ⟨Month, Country⟩ better than using ⟨Year, State⟩? Thus, it is desirable to have a tool that allows the company's analysts to navigate through different granularities by rolling-up and/or drilling-down along the hierarchies, and we propose *prediction cubes* to support such analysis.

Figure 3 can be an example of a 2-dimensional prediction cube for answering the question. For example, each cell in Fig. 3 (c) is indexed by a ⟨Year, Country⟩ pair. Each cell value represents the *predictiveness* of the attribute of interest, calculated by evaluating two models trained on the subset of data in that cell. (In Section 4.2, we discuss how to measure predictiveness.) We call this type of prediction cube a *predictiveness cube*. Prediction cubes support navigation via roll-up (e.g., from ⟨Year, Country⟩ to ⟨Year, All⟩) and drill-down (e.g., from ⟨Year, Country⟩ to ⟨Year, State⟩). By navigating through this cube, we can quickly check whether the models trained on different subsets of data reflect, for example, gender discrimination.

4.2 Predictiveness cube

We now formally define a predictiveness cube. Consider a data table **D** (e.g., the Customer table). Let $\boldsymbol{X}$ be the set of features (e.g., {Gender, Salary}), Y be the class-label attribute (e.g., Valued), and $\boldsymbol{Z}$ be the set of dimension attributes (e.g., {Time, Location}). Each dimension attribute has an associated domain

hierarchy (e.g., Fig. 2). We first introduce a way to define the predictiveness of attribute(s) $\boldsymbol{V}$ (e.g., {Gender}). The *predictiveness*[2] of $\boldsymbol{V}$ on a subset of data can be quantified by the difference in accuracy between the model built on that subset using $\boldsymbol{X}$ to predict Y and the model built on that subset using $\boldsymbol{X}$–$\boldsymbol{V}$ (e.g., {Salary}) to predict Y. The intuition is that, if the difference is large, $\boldsymbol{V}$ must play an important role in the prediction of class label Y.

We note that using the difference in error (between a model using $\boldsymbol{X}$–$\boldsymbol{V}$ to predict Y and a model using $\boldsymbol{X}$ to predict Y) to quantify the importance of $\boldsymbol{V}$ is a common practice in linear regression analysis. Based on some distribution assumptions, statisticians have developed tests for significance. Here, we do not make any distribution assumption, but think of this difference in error (or equivalently, accuracy) as a heuristic value that quantifies the importance of $\boldsymbol{V}$. We also note that the predictiveness of $\boldsymbol{V}$ is defined with respect to $\boldsymbol{X}$–$\boldsymbol{V}$. Thus, to be more precise, we should say "the predictiveness of $\boldsymbol{V}$ in the presence of the set $\boldsymbol{X}$–$\boldsymbol{V}$ of attributes." In the current formulation of prediction cubes, the user needs to select $\boldsymbol{X}$-$\boldsymbol{V}$ based on domain knowledge. This selection is critical to the interpretation of the results.

Given the definition of predictiveness, a set of attributes $\boldsymbol{V}$, and a learning algorithm, the predictiveness cube at granularity $\langle l_1, \ldots, l_d \rangle$ (e.g., ⟨Year, State⟩) is a d-dimensional array (e.g., Fig. 2 (b)), in which the value in each cell (e.g., [2005, USA/WI]) is the predictiveness of $\boldsymbol{V}$ evaluated on the subset defined by the cell (e.g., the records in the Customer table with Time in 2005 and Location in WI). Similar to a data cube, we can rollup a predictiveness cube from a lower granularity to a higher one (e.g., from ⟨Year, State⟩ to ⟨Year, Country⟩ along the Location dimension) or drilldown a predictiveness cube from a higher granularity to a lower one (e.g., from ⟨Year, State⟩ to ⟨Month, State⟩ along the Time dimension).

4.3 Computation techniques

Supporting prediction cube navigation is very computationally costly. Thus, to achieve acceptable interactive responses, materializing cell values at different granularities is generally necessary. For simplicity, we only consider full materialization, i.e., precomputing all the cell values for all possible granularities. Partial materialization with budget constraints (as in, e.g., Harinarayan et al. 1996) is an interesting future research direction.

A naïve way to do full materialization is to exhaustively build models and evaluate them for each cell, for each granularity. We need to build $(\sum_l |Z_1^{(l)}|) \times \cdots \times (\sum_l |Z_d^{(l)}|)$ models (see Section 2.2 for the notation). We call this method the **exhaustive method**. Note that the sizes of the training data for these models are dramatically different. At one extreme, consider the cells in the cube at the finest granularity (e.g., ⟨Month, State⟩). The size of the data in each such

[2] This definition is slightly different from the definition in Chen, B.-C., et al. (2005a,b), but captures the same intuition.

cell is small, and building a model for such cells is relatively inexpensive. At another extreme, consider the cell in the cube at the most general granularity ⟨All, All⟩. The training data for this cell is the entire dataset; building a model requires extremely large resources. Further, it is very likely that building the single model of the most general cell is more expensive than building the models for all the cells at the finest granularity. These observations point out a great computational challenge in prediction cube materialization. In the following, we describe computation techniques that require model construction only for the finest-grained cells. To generate the values of coarser-grained cells, we use OLAP-style bottom-up aggregation.

We call these finest-grained subsets the *base subsets*, denoted by $b_1(\mathbf{D})$, …, $b_B(\mathbf{D})$. Note that the number B of base subsets is the product of the sizes of the least general domains, i.e., $|Z_1^{(1)}| \times \cdots \times |Z_d^{(1)}|$. Each base subset corresponds to a finest-grained cell. It can be easily seen that every cube subset (subset of data in a region of cube space) can be represented as the union of some base subsets of $\mathbf{D}$. Thus, for ease of expression, we use $\sigma_S(\mathbf{D}) = \cup_{i \in S} b_i(\mathbf{D})$, where $\boldsymbol{S} \subseteq \{1, \ldots, B\}$, to denote a cube subset.

4.3.1 Decomposable scoring functions.

Our main computation technique is scoring function decomposition, which reduces prediction cube computation to data cube computation. We first note that the prediction of a predictive model can be seen as finding a class label that maximizes a scoring function. Formally, consider a predictive model $h(\boldsymbol{X}; \sigma_S(\mathbf{D}))$ trained on subset $\sigma_S(\mathbf{D})$ using features $\boldsymbol{X}$ to predict the class label. The prediction of $h(\boldsymbol{X}; \sigma_S(\mathbf{D}))$ on input tuple $\boldsymbol{x}$ can be modeled as maximizing a scoring function $Score(y \mid \boldsymbol{x}; \sigma_S(\mathbf{D}))$; i.e.,

$$h(\boldsymbol{x}; \sigma_S(\mathbf{D})) = argmax_y \text{Score}(y \mid \boldsymbol{x}; \sigma_S(\mathbf{D})).$$

Usually, $\text{Score}(y \mid \boldsymbol{x}; \sigma_S(\mathbf{D}))$ has probabilistic meaning; i.e., we expect that, given $\boldsymbol{x}$, $\text{Score}(y \mid \boldsymbol{x}; \sigma_S(\mathbf{D}))$ has the same maximum as $p(Y = y \mid \boldsymbol{X} = \boldsymbol{x}, \sigma_S(\mathbf{D}))$.
Before we introduce the main properties of scoring functions, we first review the two types of aggregate functions that can be efficiently computed in a data cube (Gray et al. 1997). Let $\mathbf{D}_1, \ldots, \mathbf{D}_n$ be subsets of $\mathbf{D}$ that partition $\mathbf{D}$; i.e., $\cup_i \mathbf{D}_i = \mathbf{D}$, $\mathbf{D}_i \cap \mathbf{D}_j = \emptyset$, for $i \neq j$, and, for ease of exposition, we further require that no base subset be split into multiple $\mathbf{D}_i$'s. An aggregate function F is *distributive* if there is a function H such that $F(\mathbf{D}) = H(\{F(\mathbf{D}_i) : i = 1, \ldots, n\})$. SUM, MIN and MAX are distributive aggregate functions with $F = H$. COUNT is distributive with $H = \text{SUM}$. An aggregate function F is *algebraic* if there are a function G that returns a fixed-length tuple and a function H such that $F(\mathbf{D}) = H(\{G(\mathbf{D}_i) : i = 1, \ldots, n\})$. All distributive functions are algebraic. AVERAGE is another example of a distributive function, where $G(\mathbf{D}_i)$ returns $[\text{SUM}(\mathbf{D}_i), \text{COUNT}(\mathbf{D}_i)]$, and H sums up all the sums and counts separately and then divides the total sum by the total count. It can be clearly seen that if an aggregate function is distributive or algebraic, then the aggregate value of $\mathbf{D}$

can be computed directly from the aggregate results of $\mathbf{D}_1, \ldots, \mathbf{D}_n$. This means we do not need to compute the aggregate function from scratch each time. The same properties can be defined for scoring functions.

Definition Distributive decomposability. A scoring function Score$(y \mid \boldsymbol{x}; \sigma_S(\mathbf{D}))$ is distributively decomposable if Score$(y \mid \boldsymbol{x}; \sigma_S(\mathbf{D}))$ = $F(\{\text{Score}(y \mid \boldsymbol{x}; b_i(\mathbf{D})) : i \in \boldsymbol{S}\})$, where F is a distributive aggregate function. A predictive model based on a distributively decomposable scoring function is called a distributively decomposable model.

Definition Algebraic decomposability. A scoring function Score$(y \mid \boldsymbol{x}; \sigma_S(\mathbf{D}))$ is algebraically decomposable if Score$(y \mid \boldsymbol{x}; \sigma_S(\mathbf{D})) = F(\{G(y, \boldsymbol{x}, b_i(\mathbf{D})) : i \in \boldsymbol{S}\})$, where F is an algebraic aggregate function, and G is a function that returns a fixed-length tuple. A predictive model based on an algebraically decomposable scoring function is called an algebraically decomposable model.

It can be easily shown that, for predictive models with decomposable scoring functions, data cube computation techniques can be directly applied to the computation of prediction cubes. In Chen et al. 2005a, we show that the Naïve Bayes classifier has an algebraically decomposable scoring function, and the kernel-density-based classifier has a distributively decomposable scoring function. We also developed an ensemble method, called the Probability-Based Ensemble (PBE), to approximately make the scoring function of any predictive model distributively decomposable. We omit the details here.

4.4 Efficient predictiveness cube materialization.

In the following, we describe an algorithm to efficiently materialize a predictiveness cube. For simplicity, we make the following assumptions:

- Predictive model h is distributively decomposable. Models having algebraically decomposable scoring functions can also be handled similarly.
- Each base subset has a sufficient amount of data to build a reasonable predictive model.
- A labeled test set Δ is available.
- Given test example $\boldsymbol{x}$, model $h(\boldsymbol{X}; \sigma_S(\mathbf{D}))$ not only predicts the class label for $\boldsymbol{x}$, but also has the ability to output Score$(y \mid \boldsymbol{x}; \sigma_S(\mathbf{D}))$ for every class label y.
- We have a DBMS that implements extended SQL GROUP BY clauses that support a CUBE operator similar to Gray et al. 1997. However, in contrast to Gray et al. 1997, instead of using ROLL UP, we assume the CUBE operator will perform hierarchical roll-up for each dimension.

Note that the goal of the following algorithm is to show the main idea of transforming prediction cube computation to data cube computation, and the feasibility of implementing prediction cubes in database management systems. Thus, we use SQL statements to describe the algorithm. This may not be the most efficient computation method but is good enough for our illustration purposes.

Given predictive model h, data table $\mathbf{D}$ with dimension attributes $\boldsymbol{Z} = \{Z_1, \ldots, Z_d\}$, features $\boldsymbol{X}$ and the class attribute Y, and labeled test set Δ with attributes $(\boldsymbol{X}, Y)$, the full materialization table of a predictiveness cube of attributes $\boldsymbol{V} \subset \boldsymbol{X}$ is computed as follows:

1. *Generate the scores for each base subset:* We first create two score tables with attributes ($\boldsymbol{Z}$, TID, Y, Score), where TID is the test example ID. Then, for each base subset $b_i(\mathbf{D})$, we train a model $h(\boldsymbol{X}; b_i(\mathbf{D}))$, and, for each test example $\boldsymbol{x}_{tid} \in \Delta$ and for each class label y, we use model $h(\boldsymbol{X}; b_i(\mathbf{D}))$ to compute Score($y \mid \boldsymbol{x}_{\text{tid}}; b_i(\mathbf{D})$) and save the score in ScoreTable_1. Similarly, for each base subset $b_i(\mathbf{D})$, we train another model $h(\boldsymbol{X}\text{-}\boldsymbol{V}; b_i(\mathbf{D}))$, and, for each test example and each class label, save the score in ScoreTable_2.
2. *Materialize two score cubes:* Given ScoreTable_1 and ScoreTable_2, we materialize the score cubes using the following SQL query, where F is the distributive aggregate function for the scoring function, and k can be 1 or 2. The result is that, in ScoreCube_k, for each region $\boldsymbol{S}$ in the cube space spanned by $Z_1, \ldots, Z_k$, for each test example (identified by *tid*) and each class label y, we have a score $Score(y \mid \boldsymbol{x}_{\text{tid}}; \sigma_S(\mathbf{D}))$.

   ```
   INSERT INTO ScoreCube_k
   SELECT Z_1, ..., Z_d, TID, Y, F(Score)
   FROM ScoreTable_k
   GROUP BY TID, Y CUBE Z_1, ..., Z_d;
   ```

3. *Materialize two accuracy cubes:* We then materialize two accuracy cubes with schema [$\boldsymbol{Z}$, *Accuracy*] as follows, where *ts*_accuracy(•) is a function that computes the accuracy from the scores, and k can be 1 or 2. We omit the definition of *ts*_accuracy(•) since it is straightforward. The result is that, in AccuracyCube_k, for each region in the cube space spanned by $Z_1, \ldots, Z_d$, we have the accuracy of the model built on that region.

   ```
   INSERT INTO AccuracyCube_k
   SELECT Z_1, ..., Z_d, ts_accuracy (TID, Y,Score)
   FROM ScoreCube_k
   GROUP BY Z_1, ..., Z_d;
   ```

4. *Materialize the predictiveness cube:* Finally, we materialize the predictiveness cube by taking the difference between the two accuracy values for each pair of the corresponding cells in the two accuracy cubes.

   ```
   INSERT INTO PredictivenessCube
   SELECT C_1.Z_1, ..., C_1.Z_d, C_1 .Accuracy−C_2.Accuracy
   FROM ScoreCube_1 C_1, ScoreCube_2 C_2
   WHERE C_1.Z_1 = C_2.Z_1 AND ... AND C_1.Z_d = C_2.Z_d;
   ```

Note that the above predictiveness cube is computed based on test-set accuracy. We can also compute predictiveness cubes based on cross-validation or training-set accuracy.

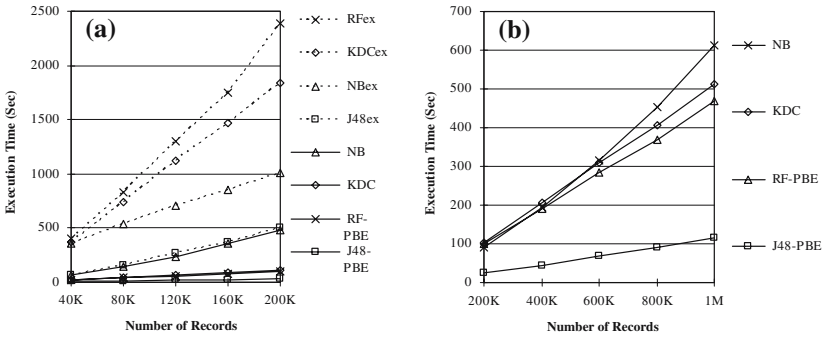


Fig. 5 Experiment results: efficiency and scalability

4.4.1 Experimental results

To understand how much the scoring function decomposition technique can improve the efficiency of prediction cube computation. We compare the running time of our technique with the running time of the exhaustive method on synthetic datasets. The details of the datasets can be found in Chen et al. 2005a. The predictive models evaluated are the Naïve Bayes classifier (NB), the kernel-density-based classifier (KDC), the probability-based ensemble of J48 decision trees (J48-PBE), and the probability-based ensemble of random forests (RF-PBE). The running times reported are based on an in-memory implementation of accuracy cube computation.

The results are shown in Fig. 5. The *x*-axis represents the size of the dataset and the *y*-axis represents the running time. In Fig. 5(a), the solid lines are for J48-PBE, RF-PBE, and the decomposable version of NB and KDC. The corresponding exhaustive methods are denoted by an "ex" suffix with dotted lines. It can be clearly seen that our computation technique significantly improves the computation efficiency. The improvement is roughly by an order of magnitude. Note that NB does not perform as well as what we expected. NB is a simple predictive model, and its running time is expected to be relatively short. However, since we use a kernel-density-based method to handle numeric attributes in NB and its algebraic scoring function is more complex than the scoring functions of other models, the actual running time of NB is quite long. The scalability of our computation technique is shown in Fig. 5(b). It can be seen that our technique scales linearly in the size of the dataset.

4.5 Discussion

Prediction cubes are a good example that demonstrates the main characteristics of cube-space data mining. Instead of seeing data-mining models as the end results, in prediction cubes, data-mining models are used as building blocks to define the interestingness of subsets of data. Instead of considering arbitrary subsets, candidate subsets are defined by regions in cube space, which are both meaningful and feasible. Because of the use of cube space, the data-cube

interface that supports rollup and drilldown analysis can be used to find interesting subsets at multiple granularities. Because of the observation that a predictive model can be seen as a scoring function and a distributively (or algebraically) decomposable scoring function can be computed as a distributive (or algebraic) aggregate function, efficient prediction-cube materialization can be achieved by adapting data-cube computation techniques.

In summary, some important research directions related to prediction cubes are:

- *Extended dimensions:* For example, parameters of model learning algorithms and properties of predictive models can be included as new kinds of dimensions. This opens the door to a significantly more general use of prediction cubes; beyond exploring data subsets, the paradigm can be used to explore alternatives in conditioning the data (e.g., choices for scaling different data axes) or tuning the learning algorithm (e.g., choices for various "magic thresholds"). Efficient cube computation for these generalizations is wide open.
- *Iceberg cubes:* Usually, we are interested in identifying regions with extreme cell values. That is, we want to select cells in a prediction cube with values higher (or lower) than a threshold value. Cubes that include only extreme cell values are called *iceberg cubes* and pruning techniques have been developed to compute them efficiently. It would be interesting to see whether pruning techniques can be used to speed up the computation of iceberg prediction cubes.
- *Automated identification of interesting subsets:* In prediction cubes, analysts use rollup and drilldown to find interesting subsets of data. Iceberg cubes provide one-step automation by assuming extreme cell values are interesting. However, some interesting subsets may be found in multiple steps. How to represent such multi-step mining, how to optimize these steps for efficient execution, and how to manage multiple tasks in a unified framework are open research problems.

5 Additional instances of cube-space data mining

In this section, we survey research related to cube-space data mining. We begin by describing several instances of cube-space mining, some proposed by us and some by others, to illustrate the potential in the idea of combining OLAP and data-mining concepts, and to demonstrate that it goes well beyond a specific instance such as prediction cubes. We conclude (in Section 5.6) with a discussion of some important early work that combines OLAP analysis and data-mining techniques, although the interplay of techniques is not as extensive as in some of the subsequently developed instances of cube-space mining.

5.1 Multi-structural databases

Fagin et al. (2005a,b) proposed multi-structural database mining, which is an instance of cube-space data mining that uses regions in cube space to mine and

summarize a set of objects in a database. For example, consider a document database where each document belongs to some elements of some dimensions (e.g., Time and Location). Given a subset of documents (e.g., a search result set), we may want to obtain a summary such as: "20% of them are from [2005, USA], 21% of them are from [All, Europe], and so on." The way that the operators in multi-structural databases are formulated is as follows. The input of any operation consists at least of a set of objects, a set of dimensions and an integer k. The output is a PDC (pair-wise disjoint collection) of k regions in the cube space spanned by the dimensions, such that the PDC optimizes an operator-dependent measure defined on the set of objects and the dimensions. Three such operators were proposed:

- *Divide:* The divide operator tries to divide the input set of objects as evenly as possible into k pair-wise disjoint regions. The returned regions can be at different granularities. This operator provides a way to succinctly summarize the input set of objects using the input dimensions.
- *Differentiate:* Given an additional "background set" of objects, the differentiate operator returns a PDC such that the input set of objects occurs significantly more frequent than our expectation estimated by the "background set." This operator provides a way to find "surprises" in the data.
- *Discover:* Given an additional set of "clustering dimensions," the discover operator returns a PDC such that the input objects in each region of the PDC are similar to one another and the input objects in different regions are quite different, where the similarity is defined according to the "clustering dimensions." This operator provides a way to discover hidden clusters in the data, where the dimensions that group objects are different than the dimensions used to define the distance.

Efficient evaluation of these operators is challenging; Fagin et al. (2005a,b) developed algorithms for some special types of PDCs.

To summarize, the research on multi-structural databases can be thought of as an effort to incorporate several kinds of cube-space mining into a unified system. Instead of using mining models to formulate queries, queries are formulated as optimization problems.

5.2 Multi-dimensional regression analysis of data streams

In this subsection, we describe an instance of cube-space data mining that incorporates linear regression models into cube space to support multi-dimensional analysis of data streams. Chen et al. (2002, 2005b) developed an architecture for the identification of unusual changes in data streams. To reduce the huge amount of streaming data, linear regression models were used to summarize the main trends in the stream at different granularities. The basic idea is, in principle, to build a regression model for each region in cube space between two user-specified granularities (called the m-layer and the o-layer), and then allow analysts to find interesting patterns by looking at the models between the

two granularities using rollup and drilldown. Exceptional regions can also be reported by setting a threshold on the slope of the regression lines. Regions having regression lines with slopes greater than the threshold value are defined to be exceptional.

Two kinds of rollup (or drilldown) were proposed based on the division of two kinds of dimension attributes. Let $X_1, \ldots, X_p, S_1, \ldots, S_q, Y$ denote the attributes, where $X_1, \ldots, X_p$ are the regression dimensions (e.g., Time), $S_1, \ldots, S_q$ are the standard dimensions (e.g., Location), and Y is the target attribute (e.g., Profit). Regression models are built using $X_1, \ldots, X_p$ to predict (possibly aggregated) Y on different regions of the cube space spanned by $X_1, \ldots, X_n, S_1, \ldots, S_q$. An example of such a model is a regression line that fits the time-series of total profits in a location. One important assumption is that, in every region defined by the standard dimensions, we always have the same set of regression dimension values (e.g., in every location, we always use the same set of time points to create the time-series of profits). Given a region $[x_1, \ldots, x_p, s_1, \ldots, s_q]$, the regression model for the region is conceptually built by: (1) selecting all the observed data records in the region, (2) grouping the selected data by $X_1, \ldots, X_p$, (3) generating the summed Y value for each group, and (4) fitting a linear regression model that uses $X_1, \ldots, X_p$ to predict the summed Y values on the selected and grouped data. For example, the regression model for region [2005, USA] is the regression line that fits the time-series of total profits in USA starting at the beginning of 2005 and ending at the end of 2005. The two kinds of rollup are:

- *Rollup along a standard dimension:* For example, we can rollup from ⟨Year, State⟩ to ⟨Year, Country⟩; i.e., rollup from time-series of state's total profits to time-series of country's total profits within a year. Note that the lengths of time-series are unchanged, but the Y values are aggregated at the corresponding time points.
- *Rollup along a regression dimension:* For example, we can rollup from ⟨Month, Country⟩ to ⟨Year, Country⟩; i.e., rollup from time-series of country's total profit within a month to time-series of country's total profit within a year. Note that month-length time-series are concatenated to generate year-length time-series, but no aggregation of Y is performed.

Efficient computation techniques were developed by extending data-cube computation techniques. Special treatment of the time dimension was also discussed. For details, see Chen et al. (2002, 2005b).

To summarize, we saw that an interesting and useful stream-mining approach was developed based on the idea of using data-mining models (i.e., regression models) as building blocks to characterize regions in cube space. Also, OLAP-style aggregation is used to generate the target values, so that regression analysis can be conducted at multiple granularities. Because of the choice of using cube space, efficient algorithms were developed and made this interesting and useful analysis feasible.

5.3 Bellwether analysis

In Chen et al. (2006a), we introduced bellwether analysis. The goal of *bellwether analysis* is to find a "cost-effective" region in cube space, such that we can accurately predict certain aggregated target values using only features collected in that region. Data-mining models are used as building blocks in a multi-step mining process, and cube space intuitively defines the space of interest. Further, aggregate queries are used to define features and even target values, minimizing the human effort required for labeling and thereby making the approach especially attractive for massive, unlabeled datasets.

Consider again the company introduced in Section 2.1. Suppose it wants to predict the worldwide profit of a new item over its first year of sales. After selling this item worldwide for one year, the company will know the exact profit. However, if the company can accurately predict this target value using features (e.g., regional profit) collected in a much shorter time (e.g., 1st week) and a much smaller area (e.g., only focus on USA), then it can quickly adapt its business strategy to minimize the loss or even maximize the profit. [1st month, USA] is an example of a *bellwether region*. The goal of bellwether analysis is to find such regions.

To find bellwether regions, the company must exploit a database containing historical sales data (shown in Fig. 1). We first consider a straightforward data-mining approach. We can aggregate the Profit table to obtain the target value (i.e., the 1^{st} year worldwide profit) for each historical item. Thus, a training set can be created by associating the **item table features** (i.e., Category and ItemSize) of each item with its target value. Then, we can train a predictive model (e.g., a linear regression model) on the training set, and use the model to predict the target value of a new item based on its features. If this model is very accurate, then no bellwether analysis is needed. However, since the item table features are usually not sufficiently predictive, the accuracy of the model is usually not acceptable.

To improve the accuracy of the predictive model, adding more informative features is necessary. Note that we have not yet used the information provided by the Profit table and Advertisement table as features to help predict the target value. However, collecting such features for a new item incurs a cost. At one extreme, if we sell the new item worldwide for one year, then we know the worldwide profit exactly. There is no need for prediction, but this incurs a very high cost. At another extreme, if we are not willing to pay anything, then we only have the item table information and no other features can be used. The goal of bellwether analysis is to find a cost-effective region, such that using new features collected from that region can best improve the accuracy of the model.

Candidate regions are defined as regions in the cube space spanned by dimension attributes (e.g., Time and Location). For each region (e.g., [1st month, USA]), new features of item i can be generated by aggregate queries over the database, such as:

- *Regional Profit:* This is the total profit of item *i* aggregated over the region (e.g., the profit of the 1st month in USA), which can be generated by a SQL query over the Profit table.
- *Regional AdExpense:* This is the total ad expense of item *i* aggregated over the region (e.g., the total ad expense in the 1st month in USA), which can be generated by a SQL query over the Advertisement table.
- *Other regional item information:* Examples of such features include maximum ad size for item *i* in the region, number of valued customers in the region who bought item *i*, etc. These can also be generated by SQL queries over the database.

Based on the company's experience, the cost of each region can be defined, which represents the cost of collecting regional features (i.e., Regional Profit, Regional AdExpense and other regional item information) from a region. For example, the company may define a cost for each finest-grained region, and then the cost of a coarser-grained region is the sum of the costs of all the finest-grained regions contained in the coarser-grained region.

Given query-generated features and costs of regions, we want to find a region with a small cost such that a highly accurate model can be built using features generated from that region. Data in the database is then used to find such a region and its corresponding model. Using queries, we create a training set for each region, in which each training example represents one item and contains: (1) the item table features, (2) the per-item query-generated regional features (i.e., Regional Profit, Regional AdExpense and other regional item information) from that region and (3) the query-generated target value of that item. Then, in principle, we build a predictive model for each region using the training set from that region, and evaluate the error. The region that has the minimum error (i.e., best accuracy) with a cost under a user-specified budget is the bellwether region. Thus, for a new item, we can collect data from the bellwether region at a cost within the budget, and expect the model that uses features generated from that region to have the minimum error over all other regions for which we could collect data at a cost under the budget.

Efficient algorithms based on two commonly used data-cube computation techniques, pruning and computation sharing, were described in Chen et al. (2006a). The basic ideas are to prune infeasible regions (e.g., regions with costs higher than the budget) using iceberg cube computation techniques, and to compute many aggregate queries all together so that some intermediate results can be reused.

Through experiments, we showed that bellwether regions do exist in real datasets, and efficient bellwether analysis is feasible. We also extended the bellwether search problem to a bellwether-based prediction problem, and developed bellwether cubes and bellwether trees so that we can make better predictions based on subsets of data. For details, see Chen et al. (2006a).

Table 1 An Example of the Extended Advertisement Table

Time	Location	ItemID	Ad Expense	…	Effective
2004–01	USA-WI	1	300 K	…	0.8
2004–03	USA-WA	1	500 K	…	0.9
2004	USA-WI	2	1,200 K	…	0.3
2004–03	USA	3	800 K	…	0.9
2005	USA	1	2,100 K	…	0.2
2005	Europe	2	1,400 K	…	0.4
…	…	…	…	…	…

5.4 Probabilistic OLAP

Another way to integrate OLAP analysis and data mining is to conduct standard OLAP analysis over data obtained by applying mining techniques. Results of data mining are usually uncertain, and the data to be mined is sometimes imprecise. To perform OLAP analysis over such results requires special care. In Burdick et al. (2005), a methodology was developed to conduct OLAP analysis over uncertain and imprecise data.

Let us consider the Advertisement table in Fig. 1. We extend this table in two ways. First, we add a new column, Effective, which tells whether an advertisement is effective or not. The values in the Effective column are probabilities of the ad's being effective, predicted using a data-mining model. Second, we allow the dimension attributes to contain imprecise values. Instead of recording the finest-grained values (at granularity ⟨Month, State⟩), the dimension values can be at any granularity. Table 1 shows an example of the extended Advertisement table. Our goal is to perform meaningful rollup and drilldown over such a table. For example, we may want to look at the total number of effective advertisements (i.e., Count(Effective)) in different times and locations at different granularities. Note that, from the example table, it is not straightforward to drilldown the total number of effective advertisements from [2004, USA] to [2004–01, USA], … , [2004–12, USA] because: (1) the measure attribute contains probability values, and (2) it is not clear how to divide the records in the table into twelve month periods.

Burdick et al. (2005) addressed this problem by defining two requirements for meaningful OLAP analysis over such uncertain and imprecise data, *consistency* and *faithfulness*, and Burdick et al. (2006) developed efficient algorithms to perform consistent and faithful rollup and drilldown.

5.5 Composite subset measures

Chen et al. (2006b) proposed a framework for multi-step computation of complex measures of regions in multi-dimensional datasets. A pictorial language allows users to easily specify complex queries compositionally, and is designed to facilitate effective optimization and scalable, efficient execution.

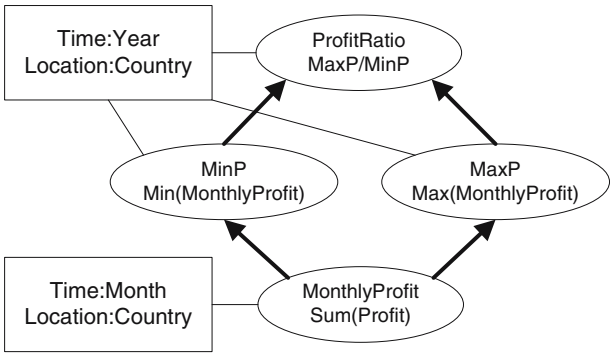


Fig. 6 An example aggregation workflow

The design goal of the pictorial language is to help users specify complex queries, called aggregation workflows, over cube space in an easy-to-use graphical interface. Fig. 6 shows an example aggregation workflow defined on the Profit table (in Figure 1), which computes four measures (ovals) for two region sets (rectangles). A region set is the set of all regions in cube space at a given granularity. For example, region set [Time: Month, Location: Country] represents the set of all regions at granularity ⟨Month, Country⟩. A measure is an attribute associated (using a thin line) with a region set. For example, MonthlyProfit is a measure (i.e., attribute) to be computed for each region at granularity ⟨Month, Country⟩, where the definition is also given in the oval. To compute the measure MonthlyProfit, we group the tuples in the Profit table by month and country, and then for each group (i.e., region) we sum up all the profit values in the group, which results in a new attribute (i.e., MonthlyProfit) for each ⟨Month, Country⟩ region. Measures can be compositionally defined, based on other measures. For example, the minimum monthly profit in each year for each country (i.e., the MinP measure of region set [Time: Year, Location: Country]) can be computed using the monthly profits of that country in that year (i.e., the MonthlyProfit measure of region set [Time: Month, Location: Country]).

Efficient algorithms have been developed, based on co-ordinated sorting and scanning of the dataset; for details, see Chen et al. (2006b). While this work rests directly on the notion of regions in cube-space, it currently addresses only simple aggregation-based measures. It would be interesting to see whether the proposed aggregation workflow framework can be extended to effectively represent and efficiently execute tasks of cube-space data mining that involve model construction and other sophisticated "measure" computations.

5.6 Early related work

Interesting early work combining ideas from OLAP and data mining includes the following:

- *Finding interesting regions in cube space:* In a large data cube, it can be difficult for analysts to manually find interesting regions. Mining techniques have been developed to help them to find interesting patterns. For example, Sarawagi et al. in several papers (Sarawagi et al., 1998, Sarawagi 1999, 2001; Sathe and Sarawagi 2001) developed: (1) a method to identify regions in which cell values are significantly different from the values anticipated by a predictive model, (2) a DIFF operator that helps analysts explore why drops and increases are observed between related regions, (3) a mechanism to identify the most informative unvisited regions in an interactive data cube exploration, and (4) a RELAX operator that summarizes generalizations and exceptions along various rollup paths from a region in a data cube. Another example of finding interesting patterns in data cubes is based on the idea of *gradients* in data cubes. Imielinski et al. 2002 generalized association rules to the notion of gradients in data cubes, where an interesting pattern is defined based on the change in measure when we rollup, drilldown, or substitute a cell with its sibling cell. Dong et al. (2001) also considered a similar problem, with the goal of finding pairs of neighboring cells (defined by rollup, drilldown and substitution) associated with big changes in measure in a data cube.
- *Approximating aggregate values using data-mining models:* Storing a large data cube requires a large amount of space. If cell values can be approximated by the predictions of predictive models, then we have a way to trade accuracy for space and efficiency. Based on this idea, there has been research on approximating cell values in a data cube using predictive models. For example, Barbará and Wu (2001) used statistical log-linear models to approximate dense regions in a data cube, while Margaritis et al. (2001) used Bayesian Networks built on a data cube to approximately answer count queries.
- *Subspace clustering:* Clustering analysis suffers from the curse of dimensionality. The goal of clustering analysis is to partition data records into groups such that the data records within each group is close to one another with respect to a distance measure. However, it is well known that when the number of dimensions increases, most data records are almost equidistant from one another. To find meaningful clusters in a high dimensional space, methods have been developed to find clusters in low dimensional subspaces; i.e., to find clusters by using distances defined on possibly different subsets of dimensions. These low dimensional subspaces are in fact regions in the cube space spanned by the dimensions. Thus, the idea of subspace clustering can be thought of as clustering analysis in multi-dimensional cube space. This idea ameliorates the curse of dimensionality and generates many interesting research results. For a survey of subspace clustering, see Parsons etal. (2004).

6 Conclusion

We discussed a class of new problems and techniques based on combining ideas from OLAP and data mining, which show promise for exploratory mining. The common underlying objective is to look beyond a single data-mining step (e.g.,

data summarization or model construction) and address the combined process of data selection and transformation, parameter and algorithm selection, and model construction. The fundamental difficulty lies in the large space of alternative choices at each step, and the ideas in this paper are initial steps towards a rigorous exploratory framework that supports the entire process. There are a number of challenging research topics to be addressed:

- How to efficiently conduct cube-space data mining in a scalable way.
- How to provide a unified framework to describe and execute different tasks of cube-space data mining.
- How to handle high dimensional cube spaces and complex dimensions such as parameters of mining algorithms and properties of mining models.
- How to judge whether the mining results are statistically significant.

We expect research on cube-space data mining, if technically successful, to produce significant practical gains.

References

Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. ACM SIGMOD Record 22(2):207–216

Barbará D, Wu X (2001) Loglinear-based quasi cubes. J Intell Inf Syst 16(3):255–276

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Burdick D, Deshpande PM, Jayram TS, Ramakrishnan R, Vaithyanathan S (2005) OLAP over uncertain and imprecise data. In: Proceedings of the 31st international conference on very large data bases (VLDB 05), pp 970–981

Burdick D, Deshpande PM, Jayram TS, Ramakrishnan R, Vaithyanathan S (2006) Efficient allocation algorithms for olap over imprecise data. In: Proceedings of the 32nd international conference on very large data bases (VLDB 06), pp 391–402

Chen Y, Dong G, Han J, Wah BW, Wang J (2002) Multi-dimensional regression analysis of time-series data streams. In: Proceedings of the 28th international conference on very large data bases (VLDB 02), pp 323–334

Chen B-C, Chen L, Lin Y, Ramakrishnan R (2005a) Prediction cubes. In: Proceedings of the 31st international conference on very large data bases (VLDB 05), pp 982–993

Chen Y, Dong G, Han J, Pei J, Wah BW, Wang J (2005b) Stream cube: an architecture for multi-dimensional analysis of data streams. Distribut Parallel Databases 18(2):173–197

Chen B-C, Ramakrishnan R, Shavlik JW, Tamma P (2006a) Bellwether analysis: predicting global aggregates from local regions. In: Proceedings of the 32nd international conference on very large data bases (VLDB 06), pp 655–666

Chen L, Ramakrishnan R, Barford P, Chen B-C, Yegneswaran V (2006b) Composite subset measures. In: Proceedings of the 32nd international conference on very large data bases (VLDB 06), pp 403–414

Danna A, Gandy O (2002) All the glitters is not gold: digging beneath the surface of data mining. J Bus Ethics 40:373–386

Dobra A, Fienberg SE (2001) Bounds for cell entries in contingency tables induced by fixed marginal totals with applications to disclosure limitation. Stat J U N 18:363–371

Dong G, Han J, Lam J, Pei J, Wang K (2001) Mining multi-dimensional constrained gradients in data cubes. In: Proceedings of the 27th international conference on very large data bases (VLDB 01), pp 321–330

Fagin R, Guha R, Kumar R, Novak J, Sivakumar D, Tomkins A (2005a) Multi-structural databases. In: Proceedings of the 24th ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems (PODS 05), pp 184–195

Fagin R, Kolaitis PG, Kumar R, Novak J, Sivakumar D, Tomkins A (2005b) Efficient implementation of large-scale multi-structural databases. In: Proceedings of the 31th international conference on very large data bases (VLDB 05), pp 958–969

Gray J, Chaudhuri S, Bosworth A, Layman A, Reichart D, Venkatrao M (1997) Data cube: a relational aggregate operator generalizing group-by, cross-tab, and sub-tables. J Data Min Knowl Discov 1:29–53

Han J (1998) Towards on-line analytical mining in large databases. ACM SIGMOD Record 27(1):97–107

Harinarayan V, Rajaraman A, Ullman JD (1996) Implementing data cubes efficiently. ACM SIGMOD Record 25(2):205–216

Imielinski T, Khachiyan L, Abdulghani A (2002) Cubegrades: generalizing association rules. J Data Min Knowl Disov 6:219–257

Kifer D, Gehrke J (2006) Injecting utility into anonymized datasets. In: Proceedings of ACM SIGMOD international conference on management of data (SIGMOD 06), pp 217–228

LeFevre K, DeWitt D, Ramakrishnan R (2006) Workload-aware anonymization. In: Proceeding of the international conference on knowledge discovery and data mining (KDD 06), pp. 277–286

Machanavajjhala A, Gehrke J, Kifer D, Venkitasubramaniam M (2006) l-Diversity: privacy beyond *k*-anonymity. In: Proceedings of the 22nd international conference on data engineering (ICDE 06), pp 24.

Margaritis D, Faloutsos C, Thrun S (2001) NetCube: a scalable tool for fast data mining and compression. In: Proceedings of the 27th international conference on very large data bases (VLDB 01), pp 311–320

Mitchell TM (1997) Machine learning. McGraw-Hill, New York

Parsons L, Haque E, Liu H (2004) Subspace clustering for high dimensional data: a review. ACM SIGKDD Explor Newsl 6(1):90–105

Samarati P, Sweeney L (1998) Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical Report SRI-CSL-98–04, SRI Computer Science Laboratory

Sarawagi S (1999) Explaining differences in multidimensional aggregates. In: Proceedings of the 25th international conference on very large data bases (VLDB 99), pp 42–53

Sarawagi S (2001) User-cognizant multidimensional analysis. VLDB J 10(2–3):224–239

Sarawagi S, Agrawal R, Megiddo N (1998) Discovery-driven exploration of OLAP data cubes. In: Proceedings of the 6th international conference on extending database technology (EDBT 98), 168–182

Sathe G, Sarawagi S (2001) Intelligent rollups in multidimensional OLAP data. In: Proceedings of the 27th international conference on very large data bases (VLDB 01), pp 531–540

Witten IH, Frank E (2000) Data mining: practical machine learning tools with java implementations. Morgan Kaufmann, San Francisco