Vertical Data Mining

William Perrizo
North Dakota State University, USA

Qiang Ding
Concordia College,USA

Qin Ding
Pennsylvania State University, USA

Taufik Abidin
North Dakota State University, USA

INTRODUCTION

The volume of data keeps increasing. There are many data sets that have become extremely large. It is of importance and a challenge to develop scalable methodologies that can be used to perform efficient and effective data mining on large data sets. Vertical data mining strategy aims at addressing the scalability issues by organizing data in vertical layouts and conducting logical operations on vertical partitioned data instead of scanning the entire database horizontally.

BACKGROUND

The traditional horizontal database structure (files of horizontally structured records) and traditional scan-based data processing approaches (scanning files of horizontal records) are known to be inadequate for knowledge discovery in very large data repositories due to the problem of scalability. For this reason, much effort has been put on sub-sampling and indexing as ways to address and solve the problem of scalability. However, sub-sampling requires that the sub-sampler know enough about the large dataset in the first place in order to sub-sample "representatively." That is, sub-sampling requires considerable knowledge about the data, which, for many large datasets, may be inadequate or non-existent. Index files are vertical structures. That is, they are vertical access paths to sets of horizontal records. Indexing files of horizontal data records does address the scalability problem in many cases, but it does so at the cost of creating and maintaining the index files separate from the data files themselves.

A new way to organize data is to organize them vertically, instead of horizontally. Data miners are typically interested in collective properties or predictions that can be expressed very briefly (e.g., a yes/no answer). Therefore, the result of a data mining query can be represented by a bitmap vector. This important property makes it possible to do data mining directly on vertical data structures.

MAIN THRUST

Vertical data structures, vertical mining approaches and multi-relational vertical mining will be explored in detail to show how vertical data mining works.

Vertical Data Structures

The concept of vertical partitioning has been studied within the context of both centralized and distributed database systems for a long time, yet much remains to be done (Winslett, 2002). There are great advantages of using vertical partitioning; for example, it makes hardware caching work really well, it makes compression easy to do, and it may greatly increase the effectiveness of the I/O device since only participating fields are retrieved each time. The vertical decomposition of a relation also permits a number of transactions to be executed concurrently. Copeland & Khoshafian (1985) presented an attribute-level Decomposition Storage Model called DSM, similar to the Attribute Transposed File model (ATF) (Batory, 1979), which stores each column of a relational table into a separate table. DSM was shown to perform well. It utilizes surrogate keys to map individual attributes together, hence requiring a surrogate key to be associated with each attribute of each record in the database. Attribute-level vertical decomposition is also used in Remotely Sensed Imagery (e.g., Landsat Thematic Mapper

Imagery), where it is called Band Sequential (BSQ) format. Beyond attribute-level decomposition, Wong et al. (1985) presented the Bit Transposed File model (BTF), which took advantage of encoded attribute values using a small number of bits to reduce the storage space.

In addition to ATF, BTF, and DSM models, there has been other work on vertical data structuring, such as Bit-Sliced Indexes (BSI) (Chan & Ioannidis, 1998; O'Neil & Quass, 1997; Rinfret et al., 2001), Encoded Bitmap Indexes (EBI) (Wu & Buchmann, 1998; Wu, 1998), and Domain Vector Accelerator (DVA) (Perrizo et al., 1991).

A Bit-Sliced Index (BSI) is an ordered list of bitmaps used to represent values of a column or attribute. These bitmaps are called bit-slices, which provide binary representations of attribute's values for all the rows.

In the EBI approach, an encoding function on the attribute domain is applied and a binary-based bit-sliced index on the encoded domain is built. EBIs minimize the space requirement and show more potential optimization than binary bit-slices.

Both BSIs and EBIs are auxiliary index structures that need to be stored twice for particular data columns. As we know, even the simplest index structure used today incurs substantial increase in total storage requirements. The increased database size, in turn, translates into higher media and maintenance costs, and results in lower performance.

Domain Vector Accelerator (DVA) is a method to perform relational operations based on vertical bit-vectors. The DVA method performs particularly well for joins involving a primary key attribute and an associated foreign key attribute.

Vertical mining requires data to be organized vertically and be processed horizontally through fast, multi-operand logical operations, such as AND, OR, XOR, and complement. Predicate tree (P-tree[1]) is one form of lossless vertical structure that meets this requirement. P-tree is suitable to represent numerical and categorical data and has been successfully used in various data mining applications, including classification (Khan et al., 2002), clustering (Denton et al., 2002), and association rule mining (Ding et al., 2002).

P-trees can be 1-dimensional, 2-dimensional, and multi-dimensional. If the data has a natural dimension (e.g., spatial data), the P-tree dimension is matched to the data dimension. Otherwise, the dimension can be chosen to optimize the compression ratio.

To convert a relational table of horizontal records to a set of vertical P-trees, the table has to be projected into columns, one for each attribute, retaining the original record order in each. Then each attribute column is further decomposed into separate bit vectors, one for each bit position of the values in that attribute. Each bit vector is then compressed into a tree structure by recording the truth of the predicate "purely 1-bits" recursively on halves until purity is reached.

Vertical Mining Approaches

A number of vertical data mining algorithms have been proposed, especially in the area of association rule mining. Mining algorithms using the vertical format have been shown to outperform horizontal approaches in many cases. One example is the Frequent Pattern Growth algorithm using Frequent Pattern Trees introduced by Han et al. (2001). The advantages come from the fact that frequent patterns can be counted via transaction_id_set intersections, instead of using complex internal data structures. The horizontal approach, on the other hand, requires complex hash/search trees. Zaki & Hsiao (2002) introduced a vertical presentation called diffset, which keeps track of differences in the tidset of a candidate pattern from its generated frequent patterns. Diffset drastically reduces the memory required to store intermediate results; therefore, even in dense domains, the entire working set of patterns of several vertical mining algorithms can be fit entirely in main-memory, facilitating the mining for very large database. Shenoy et al. (2000) proposes a vertical approach, called VIPER, for association rule mining of large databases. VIPER stores data in compressed bit-vectors and integrates a number of optimizations for efficient generation, intersection, counting, and storage of bit-vectors, which provides significant performance gains for large databases with a close to linear scale-up with database size.

P-trees have been applied to a wide variety of data mining areas. The efficient P-tree storage structure and the P-tree algebra provide a fast way to calculate various measurements for data mining task, such as support and confidence in association rule mining, information gain in decision tree classification, Bayesian probability values in Bayesian classification, and etcetera.

P-trees have also been successfully used in many kinds of distance-based classification and clustering techniques. A computationally efficient distance metric called Higher Order Basic Bit Distance (HOBBit) (Khan et al., 2002) has been proposed based on P-trees. For one dimension, the HOBBit distance is defined as the number of digits by which the binary representation of an integer has to be right-shifted to make two numbers equal. For more than one dimension, the HOBBit distance is defined as the maximum of the HOBBit distances in the individual dimensions.

Since computers use binary systems to represent numbers in memory, bit-wise logical operations are much faster than ordinary arithmetic operations such as addi