Chapter LVIII
Similarity Retrieval and Cluster Analysis Using R* Trees

Jiaxiong Pi
University of Nebraska at Omaha, USA

Yong Shi
University of Nebraska at Omaha, USA
Graduate University of the Chinese Academy of Sciences, China

Zhengxin Chen
University of Nebraska at Omaha, USA

INTRODUCTION

Data mining is aimed at the extraction of interesting (i.e., nontrivial, implicit, previously unknown, and potentially useful) patterns or knowledge from huge amounts of data. In order to make data mining manageable, data mining has to be database centered. Yet, data mining goes beyond the traditional realm of database techniques; in particular, reasoning methods developed from machine learning techniques and other fields in artificial intelligence (AI) have made important contributions in data mining. Data mining thus offers an excellent opportunity to explore the interesting fundamental issue of the relationship between data and knowledge retrieval and inference and reasoning. Decades ago, researchers made an important remark stating that since knowledge retrieval must respect the semantics of the representation language, knowledge retrieval is a limited form of inference operating on the stored facts (Frisch & Allen, 1982). The inverse side of this statement has also been explored, which views inference as an extension of retrieval. For example, Chen (1996) described a computer model that is able to generate suggestions through document structure mapping based on the notion of reasoning as extended knowledge retrieval; the model was implemented using a relational approach. However, although the issue of foundations of data mining has attracted much attention among data mining researchers (ICDM, 2004), little work has been done on the important relationship between retrieval and inference (or mining). A possible reason of lacking such kind of research is the difficulty of identifying an appropriate common ground that can be used to examine both data retrieval and data mining.

On the other hand, from the database perspective, an effective way to achieve efficient

data mining is by exploiting important features of database primitives. For example, as a multi-dimensional index structure for spatial data, R* tree (Beckmann, Kriegel, Schneider, & Seeger, 1990; Gaede & Günther**,** 1998) is a powerful database primitive and is widely used in database applications. A rich literature exists in regard to the application of R* trees for data mining as well (e.g., Keogh, Chakrabarti, Pazzani, & Mehrotra, 2000). Since the R* tree was originally developed for spatial data retrieval, recent developments in using R* tree for data mining reveals that R* tree structure can serve as a common ground to explore the relationship between retrieval and mining as discussed above.

As our first step to explore this interesting issue, in this article we examine time-series data indexed through R* trees, and study the issues of (a) retrieval of data similar to a given query (which is a plain data retrieval task), and (b) clustering of the data based on similarity (which is a data mining task). Along the way of examination of our central theme, we also report new algorithms and new results related to these two issues. We have developed a software package consisting of components to handle these two tasks. We describe both parts of our work, with an emphasis on dealing with the challenges of moving from retrieving individual queries similar to a given query to clustering the entire data set based on similarity. Various experimental results (omitted due to space limitation) have shown the effectiveness of our approaches.

BACKGROUND

Just like a B Tree, an R Tree (Guttman, 1984) relies on a balanced hierarchical structure, in which each tree node is mapped to a disk page. However, whereas B Trees are built on single-value keys and rely on a total order on these keys, R Trees organize rectangles according to a containment relationship. Each object to be indexed will be represented by a minimum bounding box (MBB) in the index structure except the point for which an MBB simply degrades to a point. All indexed objects will eventually be put in leaf nodes. A leaf node contains an array of leaf entries. A leaf entry is a pair (*mbb*, *oid*), where *mbb* is the MBB and *oid* is the object ID. Each internal node is associated with a rectangle, referred to as the directory rectangle (*dr*), which is the minimal bounding box of the rectangle of its child nodes. The structure of R Tree satisfies the following properties.

- For all nodes in the tree (except for the root), the number of entries is between m and M, where $0 \leq m\ M/2$.
- For each entry (*dr*, *node-id*) in a nonleaf node N, *dr* is the directory rectangle of a child node of N, whose page address is *node-id*.
- For each leaf entry (*mbb*, *oid*), *mbb* is the minimal bounding box of the spatial component of the object stored at address oid.
- The root has at least two entries (unless it is a leaf).
- All leaves are at the same level.

R* Tree is a variant of the R Tree that provides several improvements to the insertion algorithm. Among other things, R* tree reinserts entries upon overflow, rather than splitting. See Beckmann et al. (1990), and Gaede and Günther (1998) for more detail.

R* TREE FOR SIMILARITY ANALYSIS

As shown in Agrawal, Faloutsos, and Swami (1993), when R* tree is used for time-series data indexing, each time series of length n is mapped to a point in n-dimension space. Thus, a similarity query problem can be converted to finding those points close to a given point. The whole data set is indexed through an R* tree, and a similarity query is then carried out on the R* tree. Since R* tree indexes spatial objects according to spatial proximity and close points tend to be put in the same leaf node, a small amount of leaf nodes will be traversed before similar points are found. As a result, fast similarity analysis can be achieved. However, due