DYNAMIC CLUSTERING AND INDEXING OF MULTI-DIMENSIONAL DATA & A CLUSTER-OUTLIER ITERATIVE DETECTION

K. Vinuthna,research scholar, Dept of CSE, Sunrise University, Alwar, Rajasthan
Dr. R.K Pandey, Supervisor, Dept of CSE, Sunrise University, Alwar, Rajasthan

Abstract

These days expansive volumes of data with high dimensionality are being created in numerous fields. Generally existing indexing methods corrupt quickly when dimensionality goes higher. ClusterTree is another indexing approach speaking to clusters created by any current clustering approach. It is a chain of command of clusters and subclusters which consolidates the cluster representation into the record structure to accomplish powerful and productive recovery.

The creators propose another ClusterTree✂ indexing structure which has new features from the time point of view. Each new data thing is added to the ClusterTree with the time data which can be utilized later as a part of the data upgrade process for the securing of the new cluster structure. This methodology ensures that the ClusterTree is dependably in the most upgraded status which can assist advance the effectiveness and viability of data insertion, inquiry and redesign. This methodology is exceedingly versatile to any sort of clusters.

Watchwords: *time indexing, ClusterTree, high-dimensional datasets, time-related question, time-related cancellation*

Introduction

Clustering in data mining [SADC93, CHY96] is a discovery process that groups a set of data such that the intracluster similarity is maximized and the intercluster similarity is minimized [JD88, KR90, PAS96, CHY96]. These discovered clusters can be used to explain the characteristics of the underlying data distribution, and thus serve as the foundation for other data mining and analysis techniques. The applications of clustering include characterization of different customer groups based upon purchasing patterns, categorization of documents on the World Wide Web [BGGC99a,BGGC99b], grouping of genes and proteins that have similar functionality [HHS92, NRSC95, SCCC95, HKKM98], grouping of spatial locations prone to earth

quakes from seismological data [BR98, XEKS98], etc.

Existing clustering algorithms, such as *K*-means [JD88], PAM [KR90], CLARANS [NH94], DBSCAN [EKSX96], CURE [GRS98], and ROCK [GRS99] are designed to find clusters that fit some static models. For example, *K*-means, PAM, and CLARANS assume that clusters are hyper-ellipsoidal (or globular) and are of similar sizes. DBSCAN assumes that all points within genuine clusters are density reachable 1 and points across different clusters are not.

Large volumes of data with high dimensionality are being generated in many fields. Most existing indexing techniques degrade rapidly when dimensionality goes higher. ClusterTree is a new indexing approach representing clusters generated by any existing clustering approach. It is a hierarchy of clusters and subclusters which incorporates the cluster representation into the index structure to achieve effective and efficient retrieval. In this chapter, we propose a new ClusterTree + indexing structure from the conceptual and theoretical point of view, which has new features from the time perspective. Each new data item is added to the ClusterTree with the time information which can be used later in the data update process for the acquisition of the new cluster structure. This approach guarantees that the ClusterTree is always in the most updated status which can further promote the efficiency and effectiveness of data insertion, query and update. This approach is highly adaptive to any kind of clusters.

Nowadays many data mining algorithms focus on clustering methods. There are also a lot of approaches designed for outlier detection. We observe that, in many situations, clusters and outliers are concepts whose meanings are inseparable to each other, especially for those data sets with noise. Thus, it is necessary to treat clusters and outliers as concepts of the same importance in data analysis. In this chapter, we present COID, a cluster-outlier iterative detection algorithm, tending to detect the clusters and outliers in another perspective for noisy data sets. In this algorithm, clusters are detected and adjusted according to the intra-relationship within clusters and the inter-relationship between clusters and outliers, and vice versa. The adjustment and modification of the clusters and outliers are performed iteratively until a certain termination condition is reached. This data processing algorithm can be applied in many fields such as pattern recognition, data clustering and signal processing.

Experimental results demonstrate the advantages of our approach.

Literature Review

Numerous approaches have been designed to analyze data sets with constant changes. For example, Abrantes etc. proposed a data clustering method that extends well-known static clustering algorithms, applying a motion model to track clusters that deform and translate. The method uses centroids, or points of reference within the data cluster that are drawn towards the center of clusters. The clusters are defined by their relevance to the centroids. In every instance of tracking the previous centroids are used to calculate translations and deformations of the cluster, and establish new centroids, based on previous calculations. They also demonstrated examples of this process in the context of object tracking using pixels and three dimensional linear calculations.

Garcia etc. proposed a method for clustering data with a dissimilarity measure and a dynamic procedure of splitting, giving examples of the method by using plot graphs of two-dimensional data sets. The dissimilarity measure uses the optimum path between each successive datum. The optimum path is chosen by finding the shortest distance between two successive vertices. This measure allows the clusters to take a unique shape, rather than clustered into quadrants. The optimal partitioning can be performed using the previous optimum path as a comparison, so clusters that are dense will be less likely to assume outliers or data belonging to another cluster. The authors also described a method to scale and smooth the derivatives.

ClusterTree+ and its Construction

A large proportion of a data set may be time related, and the existence of the obsolete data in the data set may seriously degrade the data set processing. However few approaches are proposed to implement the maintenance of the whole data set to get rid of the obsolete data and keep the data set always in the most updated status for the convenience and effectiveness of data set process such as query and insertion. Although ClusterTree can efficiently handle the processing of data sets with high dimensionality, it doesn’t consider the dynamic data update issue.

Here we present a modified version of ClusterTree: ClusterTree+, which is based on the design of the ClusterTree and enhance its capability of handling dynamic data insertions, queries and deletions.

There can be several different ways to associate the time information with the original ClusterTree structure. The typical three approaches are as follows:

- Directly add time information into the ClusterTree as another dimension.
- Use a simple queue to process the time issue.
- Use an individual B+ tree-similar structure to handle the time information.

Processing of the ClusterTree

There are three major processing of the ClusterTree+: insertion, query and deletion.

1. Insertion

For a new coming data point, we classify it into one of three categories:

Cluster points: are either the duplicates of or very close to some data points in a cluster within a given threshold.

Close-by points: are the data points which are neighbors to some points in the clusters within a given threshold.

Random points: are the data points which are either far away from all of the clusters and can not be bounded by any bounding sphere of the ClusterTree, or might be included in the bounding spheres of the clusters at each level, but they do not have any neighboring cluster points within a given threshold.

Thus depending on the type of the new coming data point, we can apply the insertion algorithm of ClusterTree to recursively insert data point to a certain leaf node of ClusterTree* accompanied by the insertion time information in the T(time) field of the new entry in that leaf node, while inserting a new entry which includes the insertion time info into the B*t* tree. We then point the L(link) field of the new entry in the certain leaf node in ClusterTree* to the new entry in the leaf node of B*t* tree, and point the L(link) field of the new entry in the certain leaf node in B*t* tree to the new entry in the leaf node of ClusterTree*.

2. Query

There are two kinds of queries, one is the time-irrelevant queries which include the range queries and *P* Nearest Neighbor queries, and the other is the time-related queries which include certain time period requirement. For example, some users may ask for the neighbors to a certain data point which are inserted into the structure in almost

the same time as the insertion time of that data point.

We can depend on the original ClusterTree query algorithm to solve the former kind of queries. As for the latter one, we can solve it as follows:

Algorithm: time-related queries

Input: a data point, a time stamp;

Output: the set of data points in ClusterTree∗ which satisfy the query requirement; Algorithm steps:

- find the set a of candidate data points in the ClusterTree∗;
- at the same time search the entry x in Bt tree whose time data is closest to the query time stamp;
- find the set of the entries in Bt tree which are in a certain threshold in time distance to the entry x;
- find the set b of corresponding data points in ClusterTree∗ using the L field in the entries in Bt tree;
- get the result set of data sets by the intersection of set a and b.

3. Deletion

In many systems the obsolete data should be deleted periodically. Managers may want to delete those data which are inserted into the system a certain time ago, or they want to delete those data inserted during a certain period. Also they can simply indicate to the data system to automatically adjust itself. The ClusterTree+ can support such user-specified deletions.

Algorithm: time-related deletion1

Input: a time stamp ts;Output: the new ClusterTree+ which has got rid of the obsolete data (the data inserted before the specified time stamp);

Algorithm steps:

- find the entry x in Bt tree whose time data is left-closest to the time stamp ts
- (“left- closest” means that it is the newest one which is older than the specified time stamp ts)
- get the set a of entries in the Bt tree which are older than the entry x; search for the set b of corresponding data points in ClusterTree∗ using the L field in the entries in Bt tree;

- cut those entries in the Bt tree recursively which are older than entry x;
- cut set a in the Bt tree;
- cut set b in the ClusterTree∗.

A Cluster-Outlier Iterative Detection

The concepts of cluster and outlier are related to each other. Real world data don't necessarily have natural clusters at all. And for those which do have clusters, there are seldom the cases in reality that the data objects (data points) in the data all belong to some natural cluster. In other words, there are normally outliers existing in the data. One of the aspects of the qualities of clusters and outliers is reflected by how much *diversity* they have inside and have to each other. Clusters and outliers are concepts whose meanings are inseparable to each other. Thus, it is necessary to treat clusters and outliers as the same important concepts in the data processing. Equal treatment to clusters and outliers can benefit applications in many fields.

Definition: *The diversity between two outliers O*1 *and O*2 *is defined as:*

$D_3(O_1, O_2) = d(O_1, O_2)$

The qualities of data groups

In this subsection we define the quality of a cluster and the quality of an outlier.

We propose a novel way to define the quality of a cluster C. The quality of C is reflected not only by the diversity between it and other clusters (how far away and different they are from each other), but also by the diversity between it and outliers. In other words, if C is near some outliers, its quality should certainly be impacted, because outliers are supposed to be far away from any cluster. So we take consideration of both the diversity between clusters and the diversity between a cluster and an outlier to define the quality of a cluster.

Experiments

Qualitative Comparison

CHAMELEON To cluster a data set using CHAMELEON, we need to specify the following parameters: the value of *k* for computing the *k*-nearest neighbor graph, the value of MINSIZE for the Phase I of the algorithm, and the choice of scheme for combining relative inter-connectivity and relative closeness and associated parameters. In the experiments presented in this section, we used the same set of parameter values for

all five data sets. In particular, we used k D 10, MINSIZE D 2:5% of the total items in the data set, and used the second scheme for combining RI and RC, and used α D 2:0 in Equation 4 for combining relative inter-connectivity and relative closeness of each pair of clusters. We also performed a parameter study to determine the sensitivity of CHAMELEON on the above set of parameters by using k D f5; 10; 15; 20g, MINSIZE D f2%; 3%; 4%g and α D f1:5; 2:0; 2:5; 3:0g. Our results (not shown here) show that CHAMELEON is not very sensitive on the above choice of parameters, and it was able to discover the correct clusters for all of these combinations of values for k, MINSIZE, and α. Figure 10 shows the clusters found by CHAMELEON for each one of the five data sets. The points in the different clusters are represented using a combination of different colors and different glyphs. As a result, points that belong to the same cluster have both the same color as well as their points are drawn using the same glyph. For example, in the clustering solution shown for DS4, there are two

clusters that have cyan color (one contains the points in the region between the two circles inside the ellipse, and the other contains the points that form a line between the two horizontal bars and the 'c' shaped cluster), and there are two clusters that have a dark blue color (one corresponds to the upside-down 'c' shaped cluster and the other corresponds to the circle inside the candy-cane); however, their points are represented using different glyphs (bells and squares for the first pair, and squares and bells for the second pair), so they denote different clusters.

Since CHAMELEON is hierarchical in nature, it creates a dendrogram of possible clustering solutions at different levels of granularity. The clustering solutions shown in Figure 10 correspond to the earliest point in the agglomerative process in which CHAMELEON was able to find the genuine clusters in the data set. That is, they correspond to the lowest level of the dendrogram at which the genuine clusters in the data set have been identified and each one has been placed together in one cluster. As a result, the number of clusters shown in Figure 10 for each one of the data sets can be larger than the number of genuine clusters, and these additional clusters contain points that are outliers.

Conclusion

In this thesis, we ignored the issue of scaling to large data sets that cannot fit in the main memory. These issues are orthogonal to the ones discussed here and are covered in [ZRL96, BFR98, GRS98, GRGC99].

We have proposed a modified version of ClusterTree – ClusterTree+ from the conceptual and theoretical point of view, which can efficiently support the time-related queries and user-specified deletions. The ClusterTree+ can keep the data set always in the most updated status to promote the efficiency and effectiveness of data insertion, query and update. Further experiments will be available to support the analysis of the ClusterTree+.

This approach can be helpful in the fields of data fusion where the data evolve dynamically and regular approaches often fail to solve the problem of keeping a certain system always containing the most updated data. This approach can dynamically supervise the data status of the system and efficiently get rid of obsolete data, and at the same time, reorganize the structure of the data set.

The fast-growing, tremendous amount of data has far exceeded our human ability for comprehension without powerful tools. It is really important to design the tools to extract the valuable knowledge embedded in the vast amounts of data. This dissertation focuses on effective and efficient mining of novel, interesting and significant patterns from real data sets.

To be specific, we study the following four problems:

(1) shrinking-based data preprocessing approach and shrinking-based clustering algorithm;

(2) shrinking-based dimension reduction approach;

(3) iteratively detecting clusters and outliers based on their inter-relationship, and on the intra-relationship within clusters, and within outliers, respectively;

(4) indexing time-related multi-dimensional data sets.

References

- A. Lister, “The Problem of Nested Monitor Calls,” ACM OperatingSystems Review, July 1977, pp. 5–7.
- Peter Lohr, “Concurrency Annotations for Reusable Software,”

Communications of the ACM, vol. 36, no. 9, Sept. 1993, pp.81–89.

- Ole Lehrmann Madsen, Birger Moller-Pedersen and Kristen Nygaard, Object-Oriented Programming in the Beta Programming Language, Addison-Wesley, Reading, Mass., 1993.
- Ciaran McHale, Bridget Walsh, Sean Baker and Alexis Donnelly, "Scheduling Predicates," Proceedings of the ECOOP'91 workshop on Object-Based Concurrent Computing, ed. M. Tokoro, O.
- Nierstrasz and P. Wegner, Lecture Notes in Computer Science, vol. 612, 1992, pp. 177–193.
- Ciaran McHale, Bridget Walsh, Sean Baker and Alexis Donnelly, "Evaluating Synchronisation.
- P attie Maes, "Concepts and Experiments in Computational Reflection," in proceedings OOPSLA'87,ACM 1987.
- SIGPLAN Notices, vol. 22, no. 12, Dec.

*****