G-TREACLE: A New Grid-Based and Tree-Alike Pattern Clustering Technique for Large Databases

Cheng-Fa Tsai and Chia-Chen Yen

Department of Management Information Systems,
National Pingtung University of Science and Technology,
91201 Pingtung, Taiwan
{cftsai,m9556001}@mail.npust.edu.tw

Abstract. As data mining having attracted a significant amount of research attention, many clustering methods have been proposed in past decades. However, most of those techniques have annoying obstacles in precise pattern recognition. This paper presents a new clustering algorithm termed G-TREACLE, which can fulfill numerous clustering requirements in data mining applications. As a hybrid approach that adopts grid-based concept, the proposed algorithm recognizes the solid framework of clusters and, then, identifies the arbitrary edge of clusters by utilization of a new density-based expansion process, which named "tree-alike pattern". Experimental results illustrate that the new algorithm precisely recognizes the whole cluster, and efficiently reduces the problem of high computational time. It also indicates that the proposed new clustering algorithm performs better than several existing well-known approaches such as the K-means, DBSCAN, CLIQUE and GDH algorithms, while produces much smaller errors than the K-means, DBSCAN, CLIQUE and GDH approaches in most the cases examined herein.

Keywords: data clustering, data mining, hybrid clustering algorithm.

1 Introduction

Cluster analysis in data mining is a critical business application, which has recently become a highly active topic in data mining research [1]-[7]. Most of existing clustering techniques have high computational time, or may have pattern recognition problems when using large databases. To solve limitations of the previous existing clustering methods, this work presents a new algorithm named "**G**rid-based and **TRE**e-**A**like **C**lustering technique for **L**arge databas**E**s" (G-TREACLE) by integrating with grid-based, density-based and hierarchical clustering approaches. Performance studies show that the proposed G-TREACLE approach is a highly robust clustering technique.

T. Washio et al. (Eds.): PAKDD 2008, LNAI 5012, pp. 739–748, 2008.

2 Preliminaries

Several clustering algorithms regarding this work are described as follows.

K-means is the one of popular partitional algorithm [4]. It takes the input parameter, k, and partitions a set of n objects into k clusters. K-means always converges to a local optimum and it can not filter noise.

The grid-based clustering algorithm defines clusters as a multisolution grid data structure. It quantizes the object space into a finite number of cells that form a grid structure on which all of the operations for clustering are performed. The major advantage of the approach is its fast processing time. CLIQUE is one of the most famous grid-based techniques [7]. However, its cluster boundaries are either horizontal or vertical, due to the nature of the rectangular grid.

To identify clusters with arbitrary shape, density-based clustering approaches have been proposed. Those typically regard clusters as dense regions of objects in the data space that are separated by regions of low density (representing noise). DBSCAN is the one of well-know density-based approaches. Although it can accurately recognize any arbitrary pattern and different size clusters, and filters noise [5]. However, the time complexity of DBSCAN is high when the database size is large.

GDH integrates the idea of grid-based, density-based and hierarchical clustering methods, developed by Wang [2]. GDH refers the conception of density function and gradient decrease and concept of sliding window [2]. Although GDH can significantly eliminate the problem of indentation boundaries resulted from traditional grid-based algorithms, it may fail in grouping objects to the right position if two clusters are the same time in the populated hypercube.

3 The Proposed G-TREACLE Clustering Algorithm

This section describes the concepts of the proposed new G-TREACLE clustering algorithm. Ideally, the G-TREACLE algorithm creates a feature space through "hypercubes map constructing" in which all of objects are located on appropriate

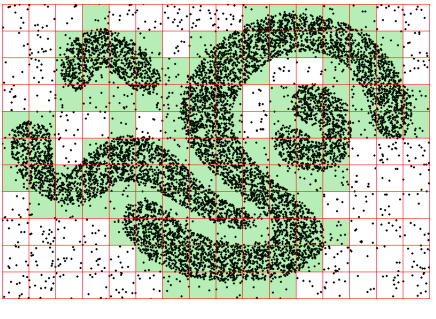

Fig. 1. In the 2-D hypercubes map, the hypercubes with dark colors are termed populated hypercube [6]

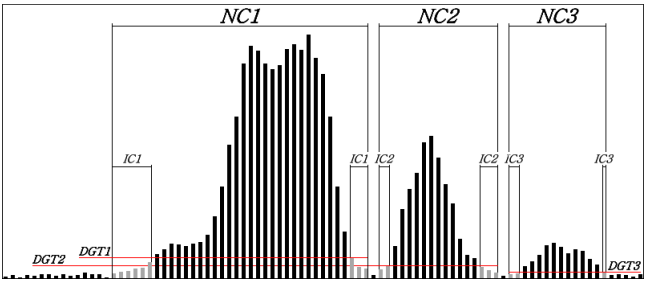

Fig. 2. Sample of solid framework recognizing in 1-D feature space

position. Then, "recognizing solid framework" is employed to fleetly identify the framework of clusters, and subsequently adopt "tree-alike pattern" within "edge shaping" to discover "blurred region", which may contain noises and cluster objects. Finally, the parts resulted from the above concepts will be integrated to acquire the complete clusters. The implemented details of concepts are illustrated with four parts as follows:

(1) **Hypercubes map constructing:** Reducing the number of searching spaces is the main idea of this step. Initially, G-TREACLE constructs a hypercubes map by splitting the feature space in accordance with a hypercube's length. Then, each object is assigned to an appropriate hypercube. If the total number of objects in the hypercube is greater than the threshold Hd, this hypercube is named "populated hypercube" [6]. Fig. 1 illustrates the concept. The searching expansion through the initial point will be performed. Notably, a populated hypercube is called "initial point" of search space if it has the highest number of objects among all populated hypercubes.

(2) **Recognizing solid framework:** This investigation adopts the "dynamic-gradient-threshold" as a measure of hypercube-volume, namely the number of objects in the populated hypercube, detecting preprocesses to discover the solid framework of clusters excluding the blurred region. The dynamic-gradient-threshold is obtained as follows:

$$DGT = |HC| \times PSV \tag{1}$$

where $|HC|$ indicates the number of objects in the most populated hypercube HC in the cluster, and PSV is the percentage of the submontane value, which is an input parameter. Fig. 2 depicts an example of the usage of dynamic-gradient-threshold. Every bar in Fig. 2 indicates the number of objects in each populated hypercube. Since every bar within a cluster may be different, dynamic-gradient-threshold can dynamically determine whether a populated hypercube can be treated as the solid framework of clusters in which every object can be assigned to a cluster without calculation. In Fig. 2, $NC1$, $NC2$ and $NC3$ represent the complete cluster. After computing the dynamic-gradient-threshold, such as $DGT1$, $DGT2$ and $DGT3$ in Fig. 2, for each cluster, the solid framework of clusters will be identified and assigned directly to a cluster but excluding

the "blurred region" representing the areas whose number of objects is under dynamic-gradient-threshold, given as $IC1$, $IC2$, $IC3$ and the areas between the clusters. Subsequently, the edge shaping step has to be utilized to detect those "blurred region", as displayed on populated hypercubes A, B, D, F and G of Fig. 3.

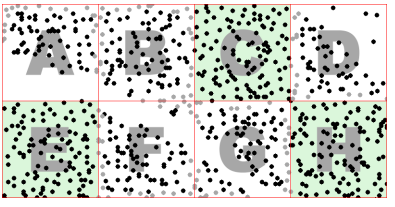

Fig. 3. Illustration of border objects for edge shaping in 2-D hypercubes map

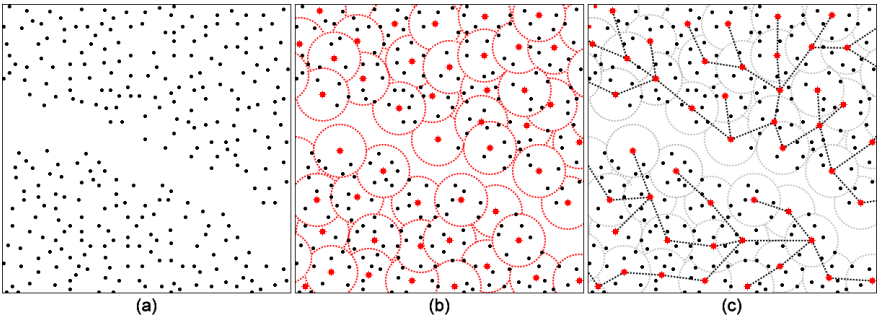

Fig. 4. Concept of searching expansion through the tree-alike pattern. (a) The original datasets (b) The neighbor-area set (c) The tree-alike pattern.

(3) **Edge Shaping:** The aim of this step is to define accurately the blurred region of a cluster. In this work, the new density-based clustering method is proposed. In contrast to conventional density-based clustering algorithms, e.g., DBSCAN, the proposed density-based method processes searching expansion through a "tree-alike pattern" comprising many centroids for each cluster, thus decreasing time complexity. Fig. 4 displays the procedure of how does the proposed density-based method work. In the 2-D hypercubes map, displayed in the diagram (a) of Fig.4, there is a given original data set $D = \{x_1, x_2, ..., x_m\}$, and a centroid set $C = \{c_1, c_2, ..., c_n\}$. For an object x_j picked from D, the centroid c_i choosing process is defined as:

$$c_i = \{x_j, \text{if } C = \phi\} \tag{2}$$

or

$$c_i = \{x_j, \text{if } d(x_j, c_p) > w, c_p \in C, p = 1, ..., i-1\} \tag{3}$$

where w is the radius of the search circle and the distance function $d(x_j, c_p)$ is the Euclidean distance function:

$$d(x_j, c_p) = \sqrt{\sum_{r=1}^{k} (x_{jr} - c_{pr})^2} \tag{4}$$

where k represents the dimension. If the centroid set C is empty or the distance between the object x_j and each centroid c_p in C is greater than w, the object x_j is chosen as new centroid. Otherwise, the object x_j is assigned to its closest centroid c_p in C. As displayed in the diagram (b) of Fig. 4, each zone surrounded by dotted circle is termed "neighbor-area" in which the largest point is illustrated as centroid. And the neighbor-area NA_p must satisfy:

$$NA_p \supset \{x_j \in D, c_p \in C : d(x_j, c_p) \leq w\} \tag{5}$$

where c_p is the centroid of NA_p. Subsequently, we need to identify which neighbor-area consisting of noise. In order to achieve this purpose, the density of every neighbor-area NA_p is determined by deriving density function [6] rather than directly counting the number of objects contained in the neighbor-area. The assumption is that the density value of the neighbor-area (namely region) comprising noise is generally lower than that of the populated neighbor-area containing normal clusters objects since its distribution is always sparser than that of the populated neighbor-area [6]. In other words, this means that although the neighbor-areas consisting of noise have the same number equivalent to the ones consisting of normal clusters objects, but the derived density value of former generally lower than that of latter. Consider some neighbor-areas within the clusters displayed in the diagram (b) of Fig. 4 that are not surrounded completely by dotted circle, those areas consist of fewer normal objects but cannot be labeled as noise-area since the density of those areas is greater than the density of noise-areas that not belong to any cluster.

In [6], influence function is defined as a mathematical description that the influence of an object has within its neighborhood, while the density function is defined as the sum of influence function of all objects in the region, and can be any arbitrary function. For simplicity, this work applies the Euclidean density function and Gaussian representation. The Gaussian density function is given by [6]:

$$f_{Gauss}^{D}(x) = \sum_{i=1}^{N} e^{-\frac{d(x_i, x_j)^2}{2\sigma^2}}, \tag{6}$$

where N represents the number of objects within the region, $d(x_i, x_j)$ denotes the distance between x_i and x_j, and σ is the standard deviation. If derived density value of the neighbor-area is greater than the threshold $MinDensityVal$, it will be preserved as a "node". Otherwise, the neighbor-area will be pruned and labeled as noise-area.

After the pruning process, each node searches its neighbor nodes and links them through the virtual edges, which are illustrated in the diagram (c) of Fig. 4.

The connection between the nodes means that their distance is less than twice the w stated above. After neighbor nodes searching recursively, a "tree-alike pattern" can be constructed as a cluster mapping. On the other hand, a broken connection between the patterns makes them into different clusters or noises. The complete algorithm is described as follows.

```
TAClustering(PartialDataSets,Width,MinDensityVal)
   NeighborAreaSet = null;
   FOR i FROM 1 TO PartialDataSets.Size DO
      Object = PartialDataSets.get(i);
      IF NeighborAreaSet.Size <> Empty
         FOR j FROM 1 TO NeighborAreaSet.Size DO
            NeighborArea = NeighborAreaSet.get(j);
            IF Object.isCloseToCentroid(NeighborArea,Width) == TRUE
               Object.assignTo(NeighborArea);
               Object.isAssigned = TRUE;
               break;
            END IF
         END FOR
         IF Object.isAssigned == FALSE
            NeighborAreaSet.setCentroid(Object);
         END IF
      ELSE
         NeighborAreaSet.setCentroid(Object);
      END IF-ELSE
   END FOR

   FOR i FROM 1 TO NeighborAreaSet.Size DO
      IF NeighborAreaSet.get(i).DensityValue < MinDensityVal
         NeighborAreaSet.prune(i);
      END IF
   END FOR

   FOR i FROM 1 TO NeighborAreaSet.Size DO
      Centroid = NeighborAreaSet.getCentroid(i);
      searchNeighborNode(Centroid,2*Width,NeighborAreaSet);
   END FOR
END TAClustering
```

`PartialDataSets` represents a partial dataset. `Width` is a search radius, and `MinDensityVal` denotes the minimal density threshold value in the region.

The neighbor node searching process `searchNeighborNode()` is as follows:

```
searchNeighborNode(CCentroid,DWidth,NeighborAreaSet)
   FOR i FROM 1 TO NeighborAreaSet.Size DO
      NCentroid = NeighborAreaSet.getCentroid(i);
      IF NCentroid.PROCESSED == FALSE && NCentroid.isCloseTo(CCentroid,DWidth) == TURE
         NCentroid.linkTo(CCentroid);
         NCentroid.PROCESSED = TURE;
         searchNeighborNode(NCentroid,DWidth,NeighborAreaSet);
      END IF
   END FOR
END searchNeighborNode
```

After running the new density-based clustering method `TAClustering()`, a set of sub-clusters can be gained from the populated hypercube that not belongs to the solid framework of the cluster. These populated hypercubes may contain objects belonging to two different clusters, as mentioned above and depicted on populated hypercubes F and G in Fig. 3. Border objects of sub-cluster and noise can be recognized at the same time [5]. In order to produce the precise combination, the proposed algorithm connects sub-cluster resulted from

`TAClustering()` run with the solid framework of cluster through the border objects of sub-cluster. Border objects are redefined as objects resulting from a `TAClustering()` run that are close to the populated hypercube's border. This redefinition shortens the computational time in `TAClustering()`. The light color objects (on the border) on populated hypercubes A, B, D, F and G of Fig. 3 indicate border objects.

(4) **Consolidation stage:** After the edge shaping stage, the algorithm merges the parts resulted from method `TAClustering()` with the solid framework of the cluster, depending on which border objects are close to the solid framework of cluster. The proposed algorithm repeats the process to recognize all clusters.

The complete clustering algorithm described as follows:

```
G_TREACLE(DataSets,Cl,PSV,Hd,Width,MinDensityVal)
   Initialization();
   ClusterId = 1;
   constructHCubeMap(Cl);
   PopulHCubeSet = getPopulHCubeSet(DataSets,PSV,Hd);
   WHILE(TRUE) DO
      IPHCube = getInitialPoint(PopulHCubeSet);
      IF IPHCube == NULL
         END ALGORITHM
      END IF
      DGT = IPHCube.ObjcetNumber * PSV;
      changeClusterId(IPHCube,ClusterId);
      searchNeighborHCubes(IPHCube,ClusterId,DGT);
      ClusterId++;
   END WHILE
END G_TREACLE
```

`DataSets` is an entire database. `Cl` represents the length of a hypercube, `PSV` denotes the percentage of the submontane value, and `Hd` is the threshold of the populated hypercube's volume. `Width` represents a search radius, and `MinDensityVal` denotes the minimal density threshold value in the region.

The neighbor searching process `searchNeighborHCubes()` is as follows:

```
searchNeighborHCubes(HCube,ClusterId,DGT)
   NeighborHCubes = getNeighborHCubes(HCube);
   WHILE NeighborHCubes.Size <> Empty DO
      CurrHCube = getHighestVolumeNeighborHCubes(NeighborHCubes);
      IF CurrHCube.ObjectNumber > DGT
         changeClusterId(CurrHCube,ClusterId);
         searchNeighborHCubes(CurrHCube,ClusterId,DGT);
      ELSE
         NCs = TAClustering(CurrHCube,Width,MinDensityVal);
         FOR i FROM 1 TO NCs.Size DO
            IF NCs.getSubCluster(i).Borders.areNear(HCube) == TRUE
               changeClusterId(NCs.getSubCluster(i),ClusterId);
            END IF
         END FOR
         searchNeighborHCubes(CurrHCube,ClusterId,DGT);
      END IF-ELSE
      NeighborHCubes.deleteNeighborHCube(CurrHCube);
   END WHILE
END searchNeighborHCubes
```

The process is repeated to construct the entire cluster.

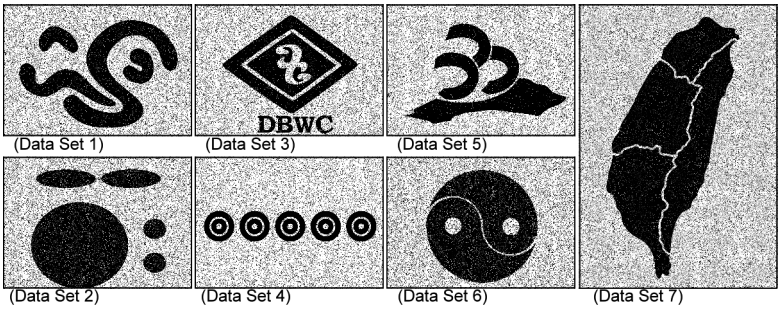

Fig. 5. The original datasets for experiment

4 Performance Studies

In this study, G-TREACLE was implemented in a Java-based program, and run on a desktop computer with 256MB RAM, an Intel 1.5GHz CPU on Microsoft MS Windows XP professional Operational System. For simple visualization, seven synthetic 2-D datasets were utilized to evaluate the performance of the proposed algorithm [3]. Among these datasets, the patterns of dataset 1, 2 and 4 were sampled from [2] and [5], Fig. 5 shows the original datasets. The results of the proposed algorithm were compared with DBSCAN, K-means, CLIQUE and GDH. Four kinds of data sizes in seven synthetic 2-D datasets, with 11,500, 115,000, 230,000 and 575,000 objects in seven synthetic 2-D datasets, and all with 15% noise, were employed in this experiment. For clustering performance comparisons, the clustering correctness rate (CCR) and noise filtering rate (NFR) are introduced. Notably, CCR represents the percentage of cluster objects correctly recognized by algorithm, while NFR denotes the percentage of noise objects correctly filtered by algorithm. Due to the computational time of DBSCAN increases significantly as the number of databases increases, hence Table 1 does not list the simulation results for DBSCAN (N/A means that the simulations were not performed). Table 1 shows the clustering experimental results with G-TREACLE, K-means, DBSCAN, CLIQUE and GDH by utilizing 575,000 object datasets. Owing to the limitation of length, not all experimental results are shown. It is observed that G-TREACLE can handle arbitrary patterns for clustering, while K-means cannot recognize arbitrary shapes. Although CLIQUE and GDH could handle the complex patterns in Dataset 4 to 7, CLIQUE could not smoothly identify clusters' edge due to the nature of the rectangular grid, and then it caused in inaccurate results. Additionally, the gradient decrease function in GDH placed some clusters the wrong position if the populated hypercubes were neighbors but the gradient decrease between the populated hypercubes was too high. In complex datasets such as DataSets 4, 5, 6 and 7, GDH and CLIQUE need to set small capacity of populated hypercube for distinction between cluster's borders that are close to each other. Therefore, the time cost of GDH and CLIQUE raises with increasing numbers of populated

Table 1. Comparisons with G-TREACLE, K-means, DBSCAN, CLIQUE and GDH using 575,000 objects data sets with 15% noise; item 1 represents time cost (in seconds); item 2 denotes the CCR (%), while item 3 is NFR (%).

Algorithm	Item	DataSet-1	DataSet-2	DataSet-3	DataSet-4	DataSet-5	DataSet-6	DataSet-7
K-means	1	18.531	16.391	36.625	59.437	43.203	7.828	19.906
	2	49.925%	51.149%	25.887%	60.837%	57.612%	50.007%	54.49%
	3	0%	0%	0%	0%	0%	0%	0%
DBSCAN	1	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	2	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	3	N/A	N/A	N/A	N/A	N/A	N/A	N/A
CLIQUE	1	5.016	8.031	8.906	12.281	30.094	31.219	46
	2	98.763%	99.104%	98.615%	95.926%	97.274%	95.647%	93.547%
	3	95.92%	98.149%	97.568%	99.305%	99.608%	99.79%	99.805%
GDH	1	8.188	9.516	10.063	13.359	31.75	26.297	51.469
	2	99.213%	99.642%	98.884%	98.299%	98.153%	96.456%	96.4%
	3	96.618%	97.477%	97.387%	98.932%	99.408%	99.736%	99.71%
G-TREACLE	1	6.156	5.594	6.547	7.766	8.469	10.64	15.75
	2	99.392%	99.511%	99.138%	98.376%	99.767%	99.754%	99.127%
	3	98.694%	99.051%	98.998%	98.894%	98.377%	98.74%	98.949%

hypercubes to be searched and processed. As shown in Table 1, G-TREACLE usually yields more accurate results and performs fast than K-means, DBSCAN, CLIQUE and GDH.

5 Conclusion

This work develops a new clustering algorithm named G-TREACLE for data mining. It can accurately identifies large patterns that are close to each other by using tree-alike pattern and is capable of successfully eliminate edge indention, so that it may improve the clustering performance of large databases as well as eliminate outliers. In addition, simulation results demonstrate that the proposed new clustering approach performs better than some existing well-known methods such as the K-means, DBSCAN, CLIQUE and GDH algorithms.

Acknowledgments. The authors would like to thank the National Science Council of the Republic of China, Taiwan for financially supporting this research under Contract No. NSC 96-2221-E-020-027.

References

1. Tsai, C.F., Tsai, C.W., Wu, H.C., Yang, T.: ACODF: A Novel Data Clustering Approach for Data Mining in Large Databases. Journal of Systems and Software 73, 133–145 (2004)
2. Wang, T.P., Tsai, C.F.: GDH: An Effective and Efficient Approach to Detect Arbitrary Patterns in Clusters with Noises in Very Large Databases. In: Degree of master at National Pingtung University of Science and Technology, Taiwan (2006)

3. Tsai, C.F., Yen, C.C.: ANGEL: A New Effective and Efficient Hybrid Clustering Technique for Large Databases. In: Zhou, Z.-H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4426, pp. 817–824. Springer, Heidelberg (2007)
4. McQueen, J.B.: Some Methods of Classification and Analysis of Multivariate Observations. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
5. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226–231 (1996)
6. Hinneburg, A., Keim, D.A.: An Efficient Approach to Clustering in Large Multimedia Databases with Noise. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, pp. 58–65 (1998)
7. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 94–105. ACM Press, Seattle, Washington (1998)