A Hierarchical Approach for Clusters in Different Densities

Baoying Wang
ewang@wayneburg.edu
Waynesburg College
Waynesburg, PA 15370
Tel: (724) 852-3285
Fax: (724) 627-6408

William Perrizo
william.perrizo@ndsu.edu
North Dakota State University
Fargo, ND 58105
Tel: (701) 231-7248
Fax: (701) 231-8255

Abstract

Clustering has the following challenges: 1) clusters with arbitrary shapes; 2) minimal domain knowledge to determine the input parameters; 3) scalability for large data sets. Density-based clustering has been recognized as a powerful approach for discovering clusters with arbitrary shapes. However, the other two challenges still remain in most existing clustering algorithms. In this paper, we explore a hierarchical and iterative density-based clustering method for large data sets with clusters in different densities. We meet the second challenge by reducing input parameters and solve the third challenge by means of hashing techniques and a vertical data structure, P-tree.

Our experiments with three different data sets show that our approach is more efficient and robust than DBSCAN, TURN*, and K-means with better clustering qualities.

Keywords: clustering, density-based, large data sets, data mining.

1. INTRODUCTION

With the rapid growth of large quantities of data collected in various application areas, such as remote sensing, geographical information systems (GIS), medical applications, satellite image analysis, and microarray gene expression data, efficient clustering methods are in great demand. Clustering is a discovery process that partitions the data set into groups such that data points within a group have high similarity in comparison to one another, but are very dissimilar to points in other groups [5].

Clustering on large data sets has the following challenges [2]: (1) clusters with arbitrary shapes; (2) minimal domain knowledge to determine the input parameters; (3) scalability for large data sizes. Density-based clustering has been recognized as a powerful approach for discovering arbitrary-shape clusters. However, the other two challenges still remain in most existing clustering algorithms. This paper is to explore a hierarchical and iterative density-based clustering method for large data sets with clusters in different densities. We meet the second challenge by reducing input parameters. We solve the third challenge by means of hashing techniques and a vertical data structure, P-tree.

The contribution of our method is: 1) the algorithm divides the data set into groups based on different densities; 2) it automatically determines the optimal neighborhood radius for each density group; and 3) hashing techniques and P-trees are used to make the algorithm more efficient and scalable. Experiments show that our approach is more efficient and more robust than DBSCAN [3], TURN* [4], and K-means [6] with better clustering results.

2. RELATED WORK

Clustering techniques can be categorized in many ways [2][5]. According to cluster structures, clustering can be divided into partitioning clustering and hierarchical clustering. Hierarchical clustering is a nested sequence of partitions, whereas a partitioning clustering is a single partition.

2.1. Partitioning Methods

Partitioning clustering methods generate a partition of a data set in an attempt to recover natural groups present in the data. Partitioning clustering can be further divided into distance-based clustering and density-based clustering according to the definition of similarity measure.

A distance-based partitioning method breaks a data set into k subsets, or clusters, such that data points in the same cluster are closer to each other than the data points in other clusters. The most classical distance-based partitioning methods are k-means [6] and k-medoid, where each cluster has a gravity center. There are several problems with distance-based partitioning methods: (1) k is the input parameter and needs to be predetermined; (2) the methods are only suitable for clusters with spherical shapes; and (3) the methods do not work well for clusters which are very different in sizes [9].

In density-based clustering, clusters are dense areas of points in the data space that are separated by areas of low density (noise) [5]. Density-based clustering can usually

1 Patents are pending on the P-tree technology.
discover clusters with arbitrary shapes without predetermining the number of clusters. However, density-based clustering is very sensitive to density-related parameters. Figure 1 shows that the clustering results are sensitive to density thresholds [7]. The higher the density threshold, the fewer points fall into clusters and the more points become noise.

![Density-based clustering](image)

The most typical density-based clustering is DBSCAN [3]. The basic idea of DBSCAN is that each cluster is a maximal set of density-connected points. Points are connected when they are reachable. DBSCAN has two input parameters, the core radius (r) and a minimum number of neighbors (k) within a core. Different values of r and k may lead to different clustering results.

### 2.2. Hierarchical Clustering

Hierarchical clustering algorithms create a hierarchical decomposition of a data set [8]. Hierarchical algorithms are more flexible than partitioning algorithms. It does not need input parameters from users. However, the computational complexities of hierarchical algorithms are typically higher than those of the partitioning algorithms.

### 2.3. Parameter Reduction

Parameter reduction is a big challenge in clustering area. There have been many efforts to make clustering process parameter-free [9]. TURN* [4] is one of the most recent researches on input parameter reduction. TURN* focuses on reducing the input parameter, the optimal core radius, of a density-based partitioning method. It first decreases the core radius to such a small value that every data point becomes a noise, which forms the first sequence of clusters. Then the core radius is doubled to get the second sequence of clusters. The process continues doubling the core radius and generates more sequences of clusters until the number of clusters and other validation criteria become stabilize, which is called a turn point.

### 3. Motivations and Observations

There are two input parameters in DBSCAN: the optimal neighborhood radius, r, and the minimum number of core neighbors, k. In fact, k is the size of the smallest cluster. It is set to 4 in DBSCAN [3]. TURN* also sets it to a fixed value [4]. By experiments, we have found that the clustering results are not very sensitive to k except when there is chaining noise between clusters.

On the other hand, a small change of r can lead to a very different clustering result. Figure 2 shows three clustering results with r = 7, 8, and 9 respectively. When r = 7, there are too many clusters. When r = 8, nine clusters are clearly separated. When r = 9, several nearby clusters are merged together. In this section, we present observations based on experiments.

![Different clustering results due to different neighborhood radii](image)

In this section, we present observations based on experiments of two data sets: DS-3 which contains clusters in three different densities and DS-4 which contains clusters in four different densities.

#### 3.1. Observations on a data set in three densities

**Observation 1:** We define R as the minimum neighborhood radius of x with k neighbors (k = 7). The points are then sorted based on the value of R in ascending order. Figure 3 shows the distribution of the data set DS-3 and its R-x graph. DS-3 is reproduced from a data set used by CHAMELEON [10]. In order to test our algorithm, we insert more data in the 3 clusters on the left top corner. The size of DS-3 is 17.5K.

![The data set and the sorted R-x graph](image)
The diagram R-x also shows the cleanness of the data. As we can see from Figure 3, if there is noise in the data set, there will be a turning point in the R-x graph where R starts to change dramatically. Our experiments show most points on the right side of the turning point are noise.

**Observation 2**: Set the neighborhood radius r for all points to be the radius at the turning point of R-x graph. We calculate the number of neighbors for each point within r neighborhood, which is denoted as K, sort the points in descending order based on the value of K, and get the sorted K-x graph. Figure 4 shows the sorted K-x graphs for DS-3. Notice that the K-x graph shows “knees” very clearly.

**Observation 3**: We partition the sorted data set based on the K-x graph above. In case of DS-3, we partition it into three subsets at two “knees” in Figure 4. The two “knees” are at positions of $X_{10052}$ and $X_{16558}$. Therefore the three partitions are (a) Partition $X_0 - X_{10052}$, (b) Partition $X_{10053} - X_{16558}$, and (c) Partition $X_{16559} - X_{17524}$, which are shown in Figure 5. We can see that partition (a) consists of the denser clusters; partition (b) consists of the less dense clusters; and partition (c) contains mainly noise.

In summary, if a data set consists of $\eta$ different density subsets, there will be $\eta-1$ “knees” in the sorted K-x graph, which divide the data set into $\eta$ density groups.

**4. THE PROPOSED METHOD**

**4.1. Determination of the neighborhood radii**

Based on observations in the previous section, we develop an algorithm to determine the optimal neighborhood radii for different density subsets. To find optimal neighborhood radii for each density groups, we need to locate the knee points in the K-x graph, denoted...
as \(k_1, k_2 \ldots k_m\), where \(m\) is the number of knees. With the knee points, we can easily calculate their minimum neighborhood radii. The calculated neighborhood radii are optimal neighborhood radii for each density groups. A smaller radius is for a denser group, and a larger one is for a sparser group.

4.2. Hashing based on \(K\)

If we hash the \(K\)-x graph in Figure 8 (a) based on \(K\), the size of hash buckets would be like Figure 8 (b).

![Hash K-x graph](image)

(a) Hash K-x graph

![Sizes of buckets](image)

(b) Sizes of buckets

Figure 8. Hash K-x graphs into hash buckets.

Among hash buckets, we only need to study the peak buckets, e.g. \(b_1\) and \(b_3\) in Figure 8 (b). We call them the effective hash buckets. An effective hash bucket is used to determine the optimal neighborhood radius for a density group. Generally, we need to locate big drops of bucket sizes between adjacent buckets to find the effective hash buckets. If there is no big drop found in consecutive reducing buckets, the algorithm will rehash the buckets using a finer value. The process continues until big drops are found.

4.3. The Hierarchical Method

Given the effective hash buckets, we simply pick a point with the minimum \(K\) value from each of these buckets, denoted as \(p_i\) from the \(i^{th}\) effective hash bucket. Finally, we can easily calculate the minimum neighborhood radius \(r_i\) of \(p_i\).

The proposed hierarchical and iterative density-based clustering method (IDCUP) is based on the optimal neighborhood radii calculated above. Our approach is hierarchical but not in the sense of the traditional hierarchical clustering. Rather we divide the data set into groups based on density and cluster the groups further using density-based clustering with the optimal neighborhood radii iteratively. After the iteration is done, we merge the clusters which share boundary points to handles the case when a cluster is composed of different density parts.

We implemented our approach using a vertical data structure, P-tree [11], to make IDCUP more efficient to conquer the third challenge of data clustering. P-trees have been used in many data mining techniques and approved successful in terms of scalability for large data sets [12].

5. EXPERIMENTAL STUDY

In this section, we show our experimental study in terms of clustering results. We compare our proposed method, IDCUP, with DBSCAN [3], TURN* [4], and K-means [5] using three different data sets. K-means and DBSCAN both need input parameters. K-means needs the number of clusters, \(k\), and DBSCAN asks for the neighborhood radius \(r\). For K-means, we set \(k\) equal to the real number of clusters in the data set for the best clustering result of the algorithm. For DBSCAN, we choose a reasonable neighborhood radius based on our knowledge of the data distribution. Notice that such choices of input parameter setting are favorable to both K-means and DBSCAN.

5.1. Data Sets

We use three typical data sets: DS1, DS2, and DS3. The distributions of the data sets are shown in Figure 9. Data set DS1 and DS2 are reproduced and enlarged based on the data set used by OPTICS [1]. Data set DS3 is reproduced and enlarged based on the data set used by CHAMELEON [10].

![Visualization of three data sets](image)

(a) DS1  (b) DS2  (c) DS3

Figure 9. Visualization of three data sets

5.2. Clustering Results

In this section, we show the clustering results in Figure 10 through Figure 12. In each figure, we show four clustering results of the four algorithms on the same data sets.
set, one from K-means algorithm, one from DBSCAN, one from TURN*, and one from IDCUP. Points in the same color are in the same cluster. Noise is shown in grayish color in all the figures.

Figure 10 shows the clustering results on DS1. DS1 is a data set with six clusters in similar densities. We can see from the figure that every method produces fairly good results except K-means. K-means merges the three upper clusters together and breaks the two lower clusters in halves respectively. DBSCAN merges together the two clusters at upper-right corner. Both TURN* and IDCUP produce almost perfect results.

Figure 11 are the clustering results on DS2. DS2 is a data set that contains six clusters in different sizes. K-means merges together the two clusters at the upper-left corner and divides the largest cluster at the upper-right corner. When $r = 4$, DBSCAN merges two close clusters at the upper-left corner. TURN* and IDCUP both produce good results.

Figure 12 are the clustering results of DS3. DS3 is a noisy data set that contains clusters with arbitrary shapes and in two different densities. K-means generates nine clusters, but the clustering results do not uncover any real cluster. When $r = 12$, DBSCAN merges five clusters including the three dense clusters into one cluster and also merges the two bar-shape clusters into one. TURN*
generates the three dense clusters well, but breaks some other clusters and marks many boundary points as noise. IDCUP produces very good clusters except that a few noise points are attached to the nearby clusters.

In summary, K-means generates fairly good results for DS1 when clusters are spherical and in similar sizes. DBSCAN is good for clusters with arbitrary shapes and in similar densities. But the clustering results are very sensitive to the input parameter. TURN* works well for arbitrary-shape clusters without input parameter. However, when clusters are in different densities, TURN* can generate the denser clusters well but tends to split the sparse clusters. IDCUP outperforms TURN* remarkably when there is a big difference among cluster densities.

6. CONCLUSION

This work is motivated by the current challenges of clustering methods: (1) clusters with arbitrary shapes; (2) minimal domain knowledge to determine the input parameters; (3) scalability for large data size. IDCUP is a hierarchical and iterative density-based clustering. It first automatically determines the optimal neighborhood radius for each density group based on the data distribution, and then iteratively carries out density-based clustering with the neighborhood radii. Generally, IDCUP is more accurate than other methods. When the density of clusters are similar, TURN* and IDCUP can generate good results. But when the clusters are in different densities, IDCUP is much better than all other methods. Hashing techniques and P-trees make IDCUP more efficient and more scalable than the other methods.

7. REFERENCES