Clustering by Ordering Density-Based Subspaces

Kan LIU*, Dongru ZHOU, Xiaozheng ZHOU

School of Computer, Wuhan University
430072 Wuhan, China
*lk2000@public.wh.hb.cn

Abstract. Finding clusters on the basis of density distribution is a traditional approach to discover clusters with arbitrary shape. Some density-based clustering algorithms such as DBSCAN, OPTICS, DENCLUE, and CLIQUE etc have been explored in recent researches. This paper presents a new approach which is based on the ordered subspace to find clusters. The key idea is to sort the subspaces according to their density, and set a new cluster for the maximal subspace of the subspace list. Since the number of the subspaces is much less than that of the data, very large databases with high-dimensional data sets can be processed with high efficiency. We also present a new method to project high-dimensional data, and then some results of clustering with visualization are demonstrated in this paper.

Keywords: Clustering, Density-based, Data visualization

1 Introduction

The process of grouping the physical data or abstract data based on their similarity is called clustering. Clustering is an important analysis method in data mining, which could help people to better understand and observe the natural classification or structure of the data. Clustering algorithms are used to automatically classify data items into the relative, meaningful clusters. After clustering, the items within any cluster are highly relevant and the items across different clusters are lowly relevant. The factors listed below are always being considered when evaluating a clustering algorithm.

- Scalability: A good clustering algorithm can deal with large datasets including up to millions data items.
- Discovery of clusters with arbitrary shape: A cluster may have an arbitrary shape. A clustering algorithm should not only apply to the regular clusters.
- Minimum parameters input: It is a heavy burden for the users to input those important parameters. In the meantime, this brings more trouble in getting good quality clustering.

- Insensitive to order of input records: Inputting data in different order should not lead to different results.
- High-dimensionality: A dataset may include many attributes, clustering data in high-dimensional space is highly demanded in many applications.

Current clustering algorithms can be broadly classified into two categories: hierarchical and partitional. Hierarchical algorithms, such as BIRCH [7], CURE [8] and CHAMELEON [9] etc, decompose a dataset into several levels of nested partitions. They start by placing each object in its own cluster and then merge these atomic clusters into larger and larger clusters until all objects are in a single cluster. Or they reverse the process by starting with all objects in a cluster and subdividing into smaller pieces. Partitional algorithms, such as CLARA [5] and CLARANS [6], partition the objects based on a clustering criterion. The popular methods, K-means and K-medoid, use the cluster average, or the closest object to the cluster center, to represent a cluster.

New clustering algorithms have been proposed in recent researches [4]. For example, DBSCAN, OPTICS and CLIQUE are based on density; STING, WAVE CLUSTER are based on grid; and COBWEB, SOM are based on model.

2 Related Work

The main idea of density-based approaches is to find regions of high-density and low-density, with high-density regions being separated from low-density regions. These approaches can make it easy to discover arbitrary clusters. A common way is to divide the high-dimensional space into density-based grid units. Units containing relatively high densities are the cluster centers and the boundaries between clusters fall in the regions of low-density units. For example, the CLIQUE [1] algorithm processes dense units level-by-level. It first determines 1-dimensional dense units by making a pass over the data. Having determined (k-1)-dimensional dense units, the candidates for k-dimensional units are determined. A cluster is a maximal set of connected dense units in k-dimensions. This algorithm automatically finds subspaces of the highest dimensionality such that high-density clusters exist in those subspaces, but the accuracy of the clustering result may be degraded at the expense of simplicity of the method.

The alternative way is to calculate parameter ‘directly density-reachable’ or ‘reachability-distance’ of the object. For example, the DBSCAN [3] aims at discovering clusters of arbitrary shape based on the formal notion of density-reachability for k-dimensional points. OPTICS [2] solves the problem of DBSCAN, which only computes a single level clustering. OPTICS produces a special order of the database with respect to its density-based clustering structure, and according to the order of the reachability-distance of each object, it can quickly reach the high-density region. OPTICS is good for both automatic and interactive cluster analysis, including finding intrinsic clustering structure, and is not limited to one global parameter setting. But it is infeasible to apply it in its current form to a database containing several million high-dimensional objects. In this paper, we propose an integrative

clustering method which is to partition the data space based on the density and grid-based techniques and realize the visualization of the clustered result.

3 Clustering by Ordering Dense Units

3.1 Basic statement

The density-based technique is adopted in our approach, in which the data space is partitioned into non-overlapping rectangular units and the density distribution will be deduced by calculating the data volume of each rectangular unit.

Suppose D is a n-dimensional dataset with m items: $D=\{X_1, X_2, \ldots, X_i, \ldots, X_m\}$, where $X_i=(X_{i1}, X_{i2}, \ldots, X_{ij}, \ldots, X_{in})$, ($i \leq m$), and $X_{ij}$is the value of the j th attribute of X_i. If each dimension of the dataset is equally divided into t parts, then all the items in the dataset fall into k units: $U=\{U_1, U_2, \ldots, U_i, \ldots, U_k\}$ ($k \leq t^n$), where $U_i=(U_{i1}, U_{i2}, \ldots, U_{ij}, \ldots, U_{in})$ is the vector of each equally divided attribute. Two units U_1 and U_2 are defined to be adjacent only when any attribute of one unit is adjacent to that of the other unit: $| U_{1j} - U_{2j} | = 1$, $U_{1s} = U_{2s}$, ($j, s \leq k, j \neq s$). We define density peaks as those units whose densities are larger than those of the adjacent units; similarly, we define density valleys as the units whose densities are lower than those of the adjacent ones.

3.2 Algorithm

When high-dimensional space is divided into k equal subspaces (units), the density peaks are regarded as the clusters centers. So the key process is how to find the density peaks. In our approach, CODBU (Clustering by Ordering Density-Based Units), the units with densities greater than threshold are ranked by the density value. The change from density peaks to density valleys is expressed by hierarchical level. The density peak is positioned in the first level of the cluster, the adjacent units are in the second level, and finally, the density valley is positioned in the last level of the cluster. First, we calculate the density values of each unit, and then rank the units whose densities are greater than threshold value. The largest-density unit will be analysed first. Each unit is compared with its adjacent units (neighbours) in density, if its density is greater than that of any other adjacent unit, it is considered as a density peak and then be set as the first level, a new cluster is emerging. If its density is less than one of adjacent units, then this unit will be grouped into the cluster of the adjacent unit; if its value is less than many of adjacent units, then this unit will be grouped into the cluster of the lowest-level unit. Fig.1 describes the process of cluster, in which only 2 basic parameters are required: the number of subdivisions for each dimension and the density threshold value. These two values are entered manually according to the size of the dataset and the required accuracy of the result.

```
CODBU (MinDen, t)
BEGIN
    int cluster_no = 0;
    int k = 0;
    Divide each attribute into t equal parts, initialize U;
    Read data x from dataset
    If (x ∈ Uj)  Uj.density++;
    For all Uj
            if (Uj.density >= MinDen){
            U.addElement(Uj);
            Uj.cluster_no=0;
            Uj.layer_no=0;
            k++; }
      Quicksort ( U );                          // sort in the descending order
      for (j=0; j<k; j++){
        for all neighbors of Uj
          if (neighbor.density > Uj.density)
            if (Uj.layer_no = 0) {          // group into the high-density unit cluster
              Uj.cluster_no = neighbor.cluster_no;
              Uj.layer_no = neighbor.layer_no+1;}
            else                                // group into the low-level unit cluster
              if (Uj.layer_no > neighbor.layer_no+1){
                Uj.cluster_no=neighbor.cluster_no;
                Uj.layer_no=neighbor.layer_no+1;}
        if (Uj.layer_no=0){                 // form a new cluster
          cluster_no++;
          Uj.cluster_no=cluster_no;
          Uj.layer_no=1;}
    }
END.
```

Fig.1 CODBU: Clustering by Ordering Density-Based Units

Figure 2 shows a simple 2-dimension data set. The sequence number of each unit is shown in the unit, and ' * ' stands for the spread points among them. Now sort all the squares whose density values are larger than 3, the result is (sorted by density): 4 (11), 5 (9), 8 (8), 10 (8), 1 (7), 9 (7), 11 (7), 3 (6), 6 (5), 12 (5), 7 (4), 2 (3). Based on the above result, we can find 2 clusters (sorted by level):

C1={4 (1), 5 (2), 1 (2), 10 (2), 3 (2), 6 (3), 11 (3)};

C2={8 (1), 9 (2), 12 (2), 7 (2), 2 (2)}.

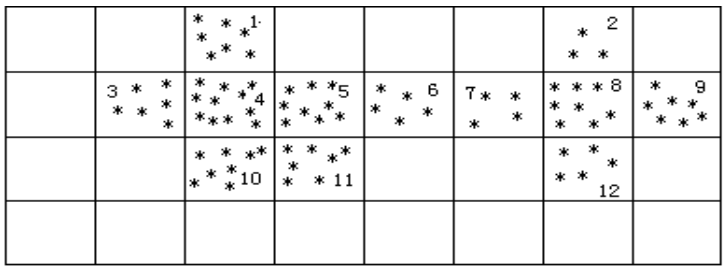


Fig.2 Two clusters based on the density values of units

We can see that, although the density of unit 5 is larger than that of unit 8, as it is adjacent to unit 4 which has higher density, unit 5 is still grouped into the first cluster. Since the density value of unit 8 is greater than that of any adjacent unit, it forms a new cluster. The density of unit 7 is lower than those of unit 6 and 8, but the unit 8 has a lower level than unit 6, so unit 7 is grouped into the second cluster.

3.3 Algorithm analysis

The dataset is only scanned once in our approach. Suppose k is the number of subspaces whose densities are greater than threshold value, the time complexity of applying quick sort is O (k*log k). By building up a search tree, the time complexity of comparing the density value of each unit with those of its adjacent units is O(nk), and n is the number of dimensions. Since the total number of units is much less than that of the data items in the dataset, the time complexity is decreased dramatically. Furthermore, the analysis of the data space based on the density order can better reflect various clusters than the traditional approach in which the data items are simply grouped together if the densities are greater than a set threshold. Therefore our approach can cluster high-dimensional data space more quickly and accurately. In addition, the quality of the clustered result will not be influenced by the shape of the high-dimensional clusters or the order of the data input, and the parameters are easily set up and modified, so all the criteria mentioned in the Section 1 have been met.

4 Visualization

Clustering high-dimensional datasets is used to help users to better understand the data structure and relationships. Visualization techniques play a very important role in displaying data and clustered results, making it more clear and reliable. The visualization techniques for high-dimensional dataset [10] [11] can be divided into two types. One type, such as “parallel coordinates” [12], is to divide the 2-d plane into several parts, each part representing an attribute. The other type is to reduce the dimensions, which is implemented through giving weights to the attributes of n-dimensional data according to the relative importance and then combine them linearly. We present a new method to project high-dimensional data, which uses stimulation spectrum to project high-dimensional data on a 3-d space.

The natural color is the summation of energy distribution in each wavelength in the range of visible spectrum. This energy distribution is called stimulation spectrum $\Phi(\lambda)$. Every stimulation spectrum can be transferred to a point in RGB color space. The quantitative relationship between stimulation spectrum $\Phi(\lambda)$ and RGB color coordinate are listed in the following formula:

$$R = k * \sum_{\lambda} \Phi(\lambda) * r(\lambda) * \Delta\lambda$$
$$G = k * \sum_{\lambda} \Phi(\lambda) * g(\lambda) * \Delta\lambda \quad (4.1)$$
$$B = k * \sum_{\lambda} \Phi(\lambda) * b(\lambda) * \Delta\lambda$$

In this formula, k is the ratio. λ refers to wavelength of visible spectrum, ranging from 400_{nm} to 700_{nm}. r(λ), g(λ) and b(λ) stand for the spectrum tristimulus functions of red, green and blue, and the value of r(λ), g(λ) and b(λ) at every 5nm is measured by CIE, which is already known. Fig.3 is the spectrum tristimulus functions graph of CIE 1931 standard colorimetric system.

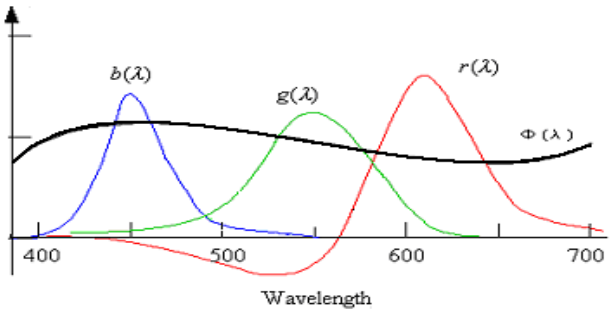


Fig. 3 Spectrum tristimulus functions graph

Each data item $X_i = (X_{i1}, X_{i2}, \ldots, X_{ij}, \ldots, X_{in})$ in high-dimensional space can be viewed as a stimulation spectrum, and spreads equally in the range of visible spectrum with the wavelength from 400_{nm} to 700_{nm}. Here X_i can be regarded as the function of λ, and the change of the attribute values corresponds to that of the spectrum tristimulus. For example, $X_i(400) = X_{i1}, \ldots, X_i(700) = X_{in}$. We can work out the 3-d coordinate of data X_i in projection space according to formula (4.2), and through adjusting the value of k can make projection space not only the RGB color space, but also any 3-d space**.**

$$x = k * \sum_{\lambda} \mathrm{Xi}(\lambda) * r(\lambda) * \Delta\lambda$$
$$y = k * \sum_{\lambda} \mathrm{Xi}(\lambda) * g(\lambda) * \Delta\lambda \quad (4.2)$$
$$z = k * \sum_{\lambda} \mathrm{Xi}(\lambda) * b(\lambda) * \Delta\lambda$$

We can see from the above that the process of projecting the data items as stimulation spectrums can also be viewed as a kind of weight linear combination of n-dimensional data through spectrum tristimulus functions r(λ), g(λ) and b(λ). Taking advantage of spectrum tristimulus functions to convert high-dimensional data can completely project the data in projection space. This is because the fundamental

function of r(λ), g(λ) and b(λ) is to project stimulation spectrums in color space, so it can well reflect the feature of the original data. From fig.3, we can see that the n attributes can be divided into 3 parts, and b(λ) corresponds to the attributes of the former part of the data while g(λ) and r(λ) mainly correspond to the middle and the last part of the data attributes. In this way, the projection result of a data item will be described by all of the 3 coordinate values. On the other hand, because the spectrum tristimulus functions cover the equal area, the ranges of the coordinate axes in projection space are equal, and the data will not be over-concentrated around some coordinate axes.

5 Experiments

A 6-dimensional dataset containing 400 points was used in our CODBU testing experiment (it is a car dataset from http://stat.cmu.edu/datasets/). The attributes in the data set are: fuel economy in miles per U.S. gallon, number of cylinders in the engine, engine displacement in cubic inches, output of the engine in horsepower, 0 to 60 mph acceleration, and vehicle weight in U.S. pounds. Each attribute was divided into 5 parts, so there were 5^6=15,625 units and the 400 points scattered in 53 units. The threshold was set up as 1 which means all the points were processed. 7 clusters were obtained through linking the associated units.

Two visualization techniques were explored in displaying the result: parallel coordinates and the spectrum tristimulus functions projection. Fig.4 shows the result using parallel coordinates, in which different clusters are represented by different colors but the characteristics of the clusters are not obvious.

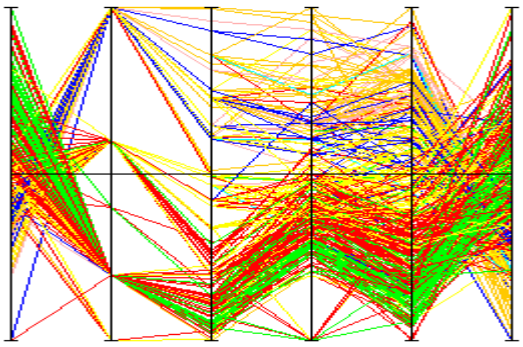
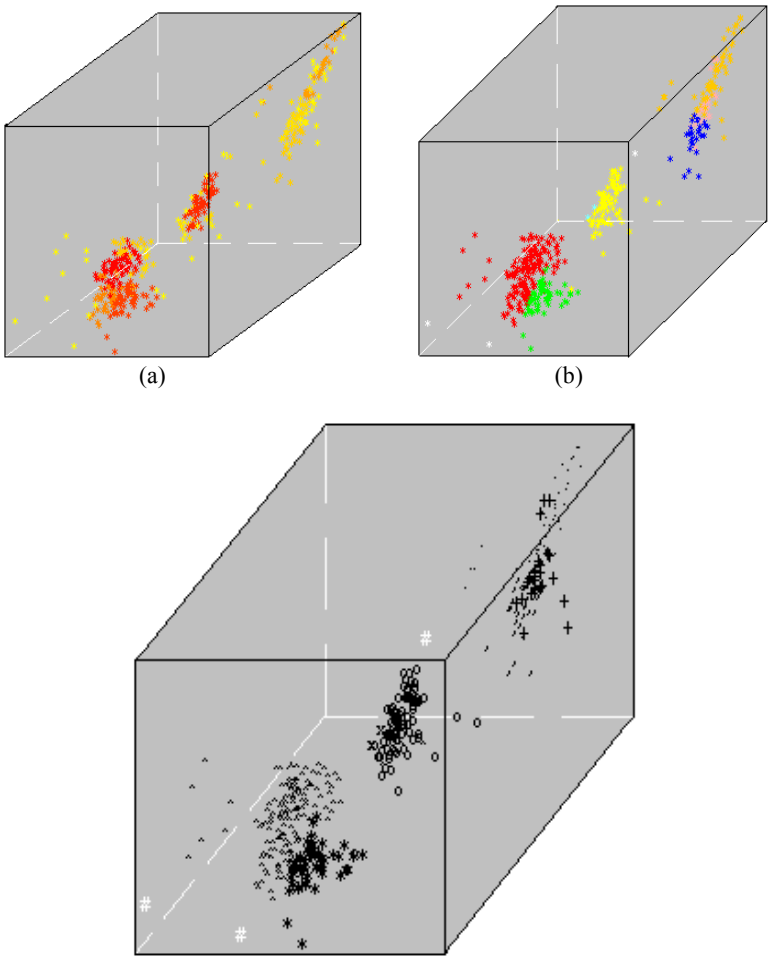


Fig. 4 Visualization of the clusters using parallel coordinates

Fig.5 shows the result using the spectrum tristimulus functions projection. The side length of the projection space is 200, Δλ=5nm, and r(λ), g(λ), b(λ) are given by CIE. In fig.5 (a), the values of the densities are reflected by color: the darker the color, the higher the density. 3 clusters are found. Fig.5 (b) displays the result of clustering using our algorithm; clusters are represented by different colors. 7 clusters are obtained based on the previous 3 big clusters. This is because that there are 7 density peaks being discovered. The clusters in white are removed due to only one data point

included. Fig.5 (c) uses symbols (such as *, +, o, ^, etc.) instead of colors to display the clusters of the data.



(a) (b)

(c)

Fig. 5 Visualization of the clusters using the spectrum tristimulus functions projection. Shades in (a) reflect the different density, and different colors in (b) represent the different clusters, while in (c) the different clusters are represented by the symbols instead of the colors.

6 Conclusions

The paper presents a new clustering approach by sorting density-based units. The basic idea is to rank the units in high-dimensional data space according to the values of the density, and start from the highest density unit to search for the density peaks in order to discover clusters. The experimental results are very promising. Clusters extend from the density peaks to density valleys and this will not be affected by the

shape of data items. Arbitrary clusters can be obtained through our approach. We also propose the method of projecting high-dimensional data on the basis of the spectrum tristimulus functions, by which the data and its distribution can be displayed in the 3-dimensional space. The combination of the data mining and the visualization techniques makes it easier for users to observe and understand data, and make better use of the data to do prediction and decision.

References

1. R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan: *Automatic Subspace Clustering of High dimensional Data for Data Mining Applications.* Proc. ACM SIGMOD'98 Int. Conf. on Management of Data, Seattle, WA (1998) .94~105
2. M. Ankerst, M.M. Breunig, H. Kriegel, J. Sander: *OPTICS: Ordering Points To Identify the Clustreing Structure.* Proc. ACM SIGMOD'99 Int. Conf. on Management of Data, Philadelphia, PA (1999)
3. M. Ester, H.P. Kriegel, J. Sander, X. Xu: *A density-based algorithm for discovering clusters in large spatial databases.* Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining (KDD'96), Portland, OR, Aug (1996) 226-231
4. J.W. Han, M. Damber: *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers (2001)
5. L. Kaufman, P.J. Rousseeuw: *Finding Groups in Data: An Introduction to Cluster Analysis*. New York, John Wiley & Sons (1990)
6. R. Ng, J. Han: *Efficient and effective clustering method for spatial data mining*. Proc. Int. Conf. Very Large Data Bases (VLDB'94) (1994) 144-155
7. T. Zhang, R. Ramakrishnan, M. Livny: *BIRCH: An efficient data clustering method for very large databases*. Proc. ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96), Montreal, Canada, June (1996) 103-114
8. S. Guha, R. Rastogi, K. Shim: *Cure: An efficient clustering algorithm for large databases*. Proc. ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98), Seattle, WA, June (1998) 73-84
9. G. Karypis, E. H. Han, V. Kumar: CHAMELEON: *A hierarchical clustering algorithm using dynamic modeling*. COMPUTER, 32 (1999) 68-75
10. I. Herman, G. MelancËon, M.S. Marshall: *Graph Visualization and Navigation in Information Visualization: A Survey*. IEEE Transactions on Visualization and Computer Graphics, Vol.6, No.1, January-March (2000) .24~43
11. A. Buja, D. Cook, D. Swayne: *Interactive High-Dimensional Data Visualization*. Journal of Computational and Graphical Statistics, No.5 (1996) 78-99.
12. H. Siirtola: *Direct Manipulation of Parallel Coordinates.* Proc. of the IEEE International Conference on Information Visualization (IV2000), London (2000) 373-378