Grid Partition and Agglomeration for Bidirectional Hierarchical Clustering

Lei Wu[1], Hechang Chen[1,2], Xiangchun Yu[1], Sun Chao[1], Zhezhou Yu[1](✉), and RuiTing Dou[1]

[1] College of Computer Science and Technology, Jilin University, Changchun, China
yuzz@jlu.edu.cn

[2] Key Laboratory of Symbolic Computation and Knowledge Engineering, Ministry of Education, Changchun, China

Abstract. Clustering is an important data processing tool, which can be used to reveal the distribution structure of unfamiliar domain data, or as preprocess methods to magnify data object to accelerate subsequent processing or simplify models. However, the distribution of many real-world data in feature space is very complex or uneven. Besides, the similarity/distance is not easy to be properly defined in feature space with different dimensional quantity. Therefore, many existing clustering algorithms are not stable in real datasets, and better performance of different datasets relies on artificial special design, such as scale normalization. In this paper, we propose a bidirectional hierarchical clustering (BHC) algorithm with two phases. In the first phase (Top-down), based on the probability density function of data in different dimensions, the feature space is divided into over-segmented grids to adapt to the complex distribution of data. In the second phase (Bottom-up), based on statistical information, a robust distance instead of geometrical distance is defined to agglomerate the grids into a dendrogram. Compared with the individual data points, grids created in the first phase can carry more statistical information, and the magnified processing objects can accelerate the clustering process. The second phase enhances the algorithm's ability by the ability of recognize arbitrary shape data clusters. The effectiveness of BHC is compared with 20 popular or recent clustering algorithms on 8 artificial datasets and 6 real-world datasets. And the results show that our algorithm can achieve good results on most datasets. In particular, BHC surpasses all the comparison algorithms involved in the experiment on all real-world datasets. In addition, in order to test the efficiency of the algorithm, we design an experiment which can test the influence of dimension and data size on the operation time.

Keywords: Hierarchical clustering · Grid-based clustering · Top-down · Bottom-up

1 Introduction

As an unsupervised learning tool, without additional label information, clustering has a wide range of applications, such as biological gene classification [1–3], chemical molecular structure [4, 5], astrophysics [6–8] and business market analysis [9]. It can be used to demonstrate the inherent structure of spatial distribution of data, helping people explore unfamiliar areas from the perspective of data distribution. In addition, in the computer field, clustering technology is often used as a preprocessing method to reduce the complexity of subsequent processing or models [10]. Since a variety of data with different types and sizes are constantly generated, clustering algorithm has always been a hot research topic. At present, the research of clustering algorithm mainly focuses on three aspects: overcoming the effects of high-dimensional data and big data on experimental effect and efficiency [11, 12], processing complex data clusters [13–16] and reducing the dependence of algorithms on user-specified parameters [17, 18].

However, most of the above studies are lopsided without an integrated manner, which may lead to two problems: (1) Clustering algorithms using global partition strategy cannot process data with complex and uneven distribution. For example, the DBSCAN [19] algorithm can handle clusters of arbitrary shapes. But it cannot perform well when encountering data with uneven distribution. (2) Considering different dimensional quantity, it is very difficult to define a reasonable point-to-point distance, especially in high-dimensional data. For example, DPC [13] algorithm and its improved algorithms [14–16], which need to calculate the distance between points, are not robust on real-world datasets.

In this paper, we propose a bidirectional hierarchical clustering algorithm called BHC. As the point-to-point distance definition is unreliable and the data distribution is complex and uneven, we can divide the data into over-segmented subsets, where data points in a subset have the same spatial distribution and statistical information. That is the basic idea of BHC. Leveraging the statistical information of subsets to calculate robust distance between subsets, those over-segmented subsets can be re-agglomerated into a dendrogram. Accordingly, our bidirectional clustering algorithm consists of two opposite-directional phases: Top-down and Bottom-up. In the first phase, based on OptiGrid [20] algorithm, a coarse grid partition is performed recursively in dividing the feature space into non-interfering grids. Even if the data distribution is uneven or complex, the points in a grid have the same distribution characteristics. In the second phase, through the statistical information shared by the data points in the grid, a grid-to-grid distance is defined. And grids are agglomerated successively according to the order of the number of points in grids. Although the grids in the first stage are simple rectangles, the algorithm can handle clusters of arbitrary shapes through the second phase. The benefit of Top-down process is that a robust distance can be defined regardless of data size, dimension and distribution. In addition, statistical information in grids can be applied to identify outliers and noise. The major contributions of this paper are summarized as follows:

1. We proposed a spatial grid partition method based on data distribution, where points in same grids have the same distribution characteristics. In this method, performing coarse grid partition recursively results in over-segmented grids while the validity of segmentation is guaranteed. In addition, feature compositor is applied to select features that have more contributions to the formation of a particular cluster.

2. We defined a robust distance between grids that can be used to agglomerate the divided grids into dendrogram. This distance is calculated from the statistical information of the data points in the grid instead of the geometric information, thus avoiding the effects of high dimensional quantity.
3. We benchmark BHC with 20 popular or recent clustering algorithms on 8 artificial datasets and 6 real-world datasets, and our algorithm achieves the best result far superior to the comparison algorithms on all real-world datasets, which proves the stability and superiority of our algorithm in dealing with practical problems. What' more, our algorithm is less affected by the data size and data dimension in the efficiency experiment. An example of BHC flow is shown in Fig. 1.

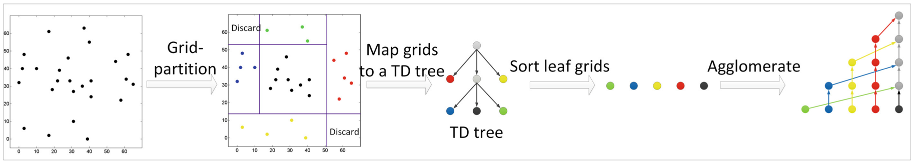

Fig. 1. An example of BHC flow

2 Methodology

In this section, we will introduce BHC with two component phases which are in opposite directions: Top-down and Bottom up. These two phases are respectively described in detail in Sects. 2.1 and 2.2.

2.1 Top-Down for Grid Partition

Partitioning the feature space into grids is a feasible strategy for big data, as magnified processing objects can accelerate the clustering process. Additionally, grids can carry more statistical information than individual data points. Therefore, this strategy is robust against outliers, since grids containing outliers are highlighted as their data points are relatively less. Besides, in BHC, the statistical information is used to define a robust distance between grids.

In this process, the feature space is partitioned recursively by the Top-down approach into multilevel coarse-grained grids to form a Top-Down (TD) tree structure, (see Fig. 2). However, axes-parallel partitions are not suitable for high-dimensional data due to data sparseness problem. Namely, the number of data points in each grid is so few that it may lose statistical significance. Hence, we adopt as wide a bandwidth as possible in one-dimensional Gaussian kernel estimation to promise a robust grid partition. And each partitioning is performed in some particular dimensions, which are selected from all dimensions according to their contribution of individual dimensions to the formation of a particular region or grid. Through recursive partitioning, an over-segmented grid set is created, while the validity of segmentation is guaranteed.

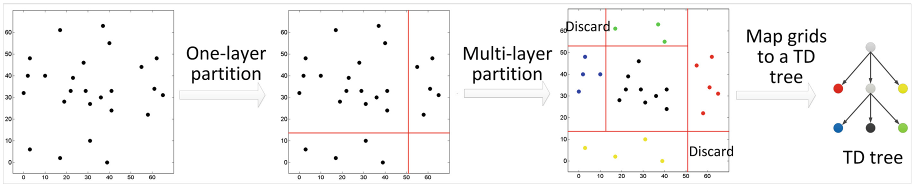

Fig. 2. An example of Top-down flow

Specifically, for a partition processing, based on OptiGrid [20] algorithm, the kernel density function is applied in the training set to calculate the probability density functions (PDFs) of all dimensions. For each kernel density estimation, the bandwidth is selected under the guidance of statistical information and a coarse-granularity approach. Then the partitioning is done by some cutting planes which are perpendicular to the point with minimal point densities in selected dimensions. These dimensions are obtained by selecting top part of the ordered dimensions, which are sorted by the difficulty of dividing the dimensions.

One-Layer Coarse-Grained Partitioning

OptiGrid provides a powerful framework for grid-partitioning by optimal cutting planes which are located at the minimal density points of PDFs in the corresponding contracting projection. In BHC, contracting projections are specifically referred to as axes projection.

For a dataset $D = \{X_1, X_2, \ldots, X_n\}$ with n d-dimensional points and $X_k = \{x_{k,1}, x_{k,2}, \ldots, x_{k,d}\}$, the PDF of D in the i-th dimension can be approximated by the one-dimensional Gaussian kernel $g(x_{k,i}, \lambda_i)$ respect to the bandwidth λ_i:

$$f_i(D, \lambda_i) = \frac{1}{n}\sum_{k=1}^{n} g\left(x_{k,i}, \lambda_i\right) = \frac{1}{\sqrt{2\pi}n\lambda_i}\sum_{k=1}^{n}\exp\left(\frac{\left(x - x_{k,i}\right)^2}{2\lambda_i^2}\right) \tag{1}$$

When data points spread sparsely in a dimension, a number of spikes (maxima) would appear in the PDF (see the top two graphs in Fig. 4). Obviously, such PDFs is not what we want as too many minimal density points for cutting planes. In DENCLUE [21] and OptiGrid, a threshold is defined to cut the impact of spikes, where the spikes with a density below the threshold would not be taken into account.

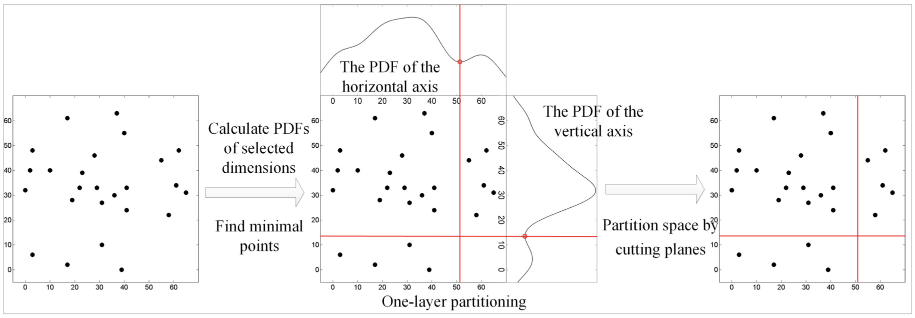
Calculate PDFs of selected dimensions
Find minimal points
The PDF of the horizontal axis
The PDF of the vertical axis
Partition space by cutting planes
One-layer partitioning

Fig. 3. One-layer partitioning processing

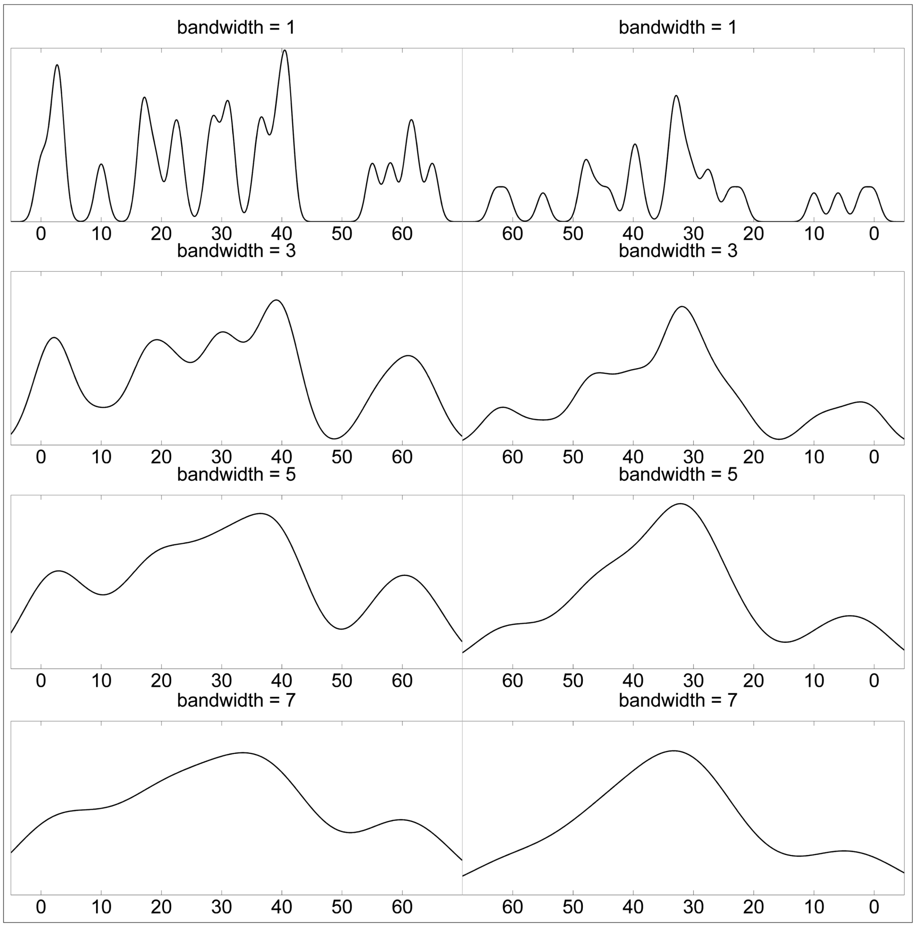

Fig. 4. PDFs with different bandwidth of two dimension of data points in Fig. 3

As shown in Fig. 4, with the increase of the bandwidth, the curve of the PDF becomes smooth and a number of useless spikes disappear. Hence, a larger value of bandwidth can filter the impact of sparse data. In this method, we attempt to use coarse-grained bandwidth to alleviate the existence of spikes. When estimating PDF, the value of the bandwidth decreased from one initial bandwidth, until more than one peak appears in the PDF. Determining cutting plane by fewer density peaks, the grids are more robust since fewer bad partitions would be done. As shown in Fig. 3, a one-layer partitioning is performed to divide a grid into four sub-grids by two cutting planes. Especially, over-segmentation is guaranteed by performing partitioning recursively.

A key problem in our method is the preset of the initial bandwidth. When setting a large value, a lot of computation is invalid; when setting a small value, the bandwidth cannot filter the impact of sparseness. To determine a proper initial bandwidth, we introduce a definition: statistical linkage.

Definition (**statistical linkage**): *dataset D is of statistical linkage in i-th dimension respect to λ_i as long as the PDF $f_i(D, \lambda_i)$ of D in i-th dimension has one maximal point at most.*

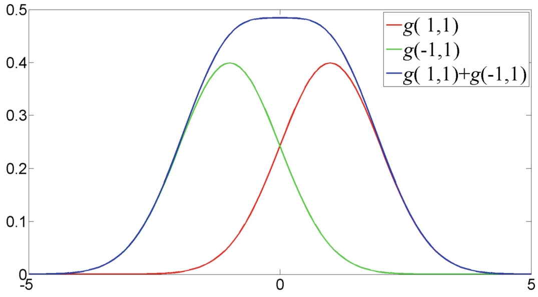

Fig. 5. Function graph of an ideal situation

According to the statistical linkage, the initial bandwidth of *i*-th dimension can be set as the minimal bandwidth making D is of statistical linkage in *i*-th dimension. However, such bandwidth is hard to be directly obtained. In an ideal situation where only two points X_1, X_2 exist in D (see in Fig. 5), it is easy to prove that the condition makes D of statistical-linkage in *i*-th is $\lambda_i \geq (x_{2,i} - x_{1,i})/2$, while $\lambda_i \geq (x_{2,i} - x_{1,i})/2$ is the standard deviation σ_i of D in the *i*-th dimension. Therefore, the initial bandwidth in BHC is initialized as the standard deviation σ_i of dataset D in the corresponding dimension.

When estimating PDFs of d dimensions, each dimension would get a separated bandwidth ranging from its standard deviation to 0, which has the maximal value making the PDF be not of statistical linkage. Obviously, data points in the dimension with a relatively large bandwidth are better divided. Hence, based on the bandwidth of a dimension, a variate can be defined to measure the difficulty of partitioning data points in this dimension.

Definition (**partitioning degree**): *A partitioning degree for the i-th dimension of the dataset D is defined as follows:*

$$degree_i^D = \arg\max H(d) \quad 0 < d \leq M \tag{2}$$

$$H(d) = \begin{cases} d, & \text{if } f_i\left(D, \frac{d}{M}\sigma_i\right) \text{has two or more maxima} \\ 0, & \text{else} \end{cases} \tag{3}$$

where an interval [0, σ_i] is divided into M segments and the bandwidth of the *i*-th dimension λ_i is assigned to σ_i *degree$_i$/M*. Figure 6 shows the calculation order of partitioning degree.

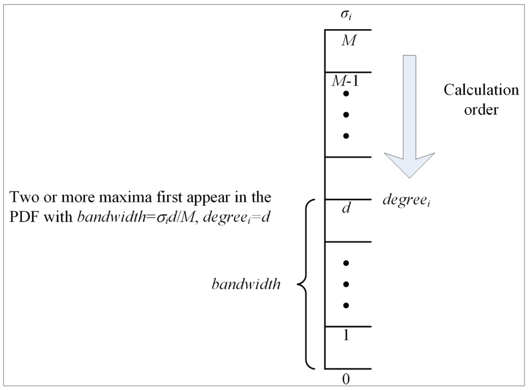
σ_i
M
M-1
Calculation order
Two or more maxima first appear in the PDF with *bandwidth*=$\sigma_i d/M$, *degree*$_i$=*d*
d
degree$_i$
bandwidth
1
0

Fig. 6. Calculation order of partitioning degree.

As for how to find the minima in PDFs, a stride detection approach is applied in our method, where the stride is set as half of the bandwidth (see Fig. 7). For two adjacent detection points x_k, x_{k+1}, if the first derivative of point x_{k+1} is less than or equal to 0 while the first derivative of x_k is bigger than 0, a minimum between x_k and x_{k+1} is approximated as follows:

$$x_{\min} = \frac{x_{k+1} + x_k}{2} + \frac{\lambda^2}{x_{k+1} + x_k} \cdot \ln\left(\frac{y_{k+1}}{y_k}\right) \tag{4}$$

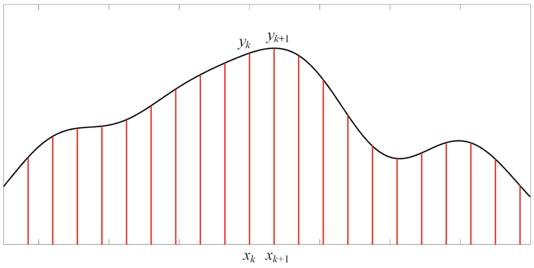

Fig. 7. Minima detection through stride detection approach

Dimension Sorting

When a coarse-grained partitioning is done in dataset *D*, each dimension gets a partitioning degree, which can be used to measure the contribution of individual dimensions to the formation of a particular sub-grid. According to the partitioning degree, part of dimensions is selected to reduce the impact of high dimensions. Since the partitioning degree is an integer within a certain range (0, *M*], there may be a number of equal degrees over dimensions, which results in a confusion of selecting dimensions. Hence, an extra variate is defined as supplementary.

Definition (**partitioning faith**): *a partitioning faith for a partitioning set {D_1, D_2, ..., D_s} over the i-th dimension of dataset D is defined as:*

$$faith_i^D = \sum_j^s \sigma_i^{D_j} \bigg/ \left(s \cdot \sigma_i^D\right) \tag{5}$$

Where σ_i^D denotes the standard deviation of the i-dimension of dataset D. After the partitioning degree of a dimension determined, the partitioning faith is confirmed. Obviously, the smaller the partitioning faith is, the better the partitioning performs. By the partitioning degree and faith, all dimensions of dataset D can be sorted according to the performance of partitioning, and only the former P dimensions are involved in grid-partitioning. In this way, not only can we reduce the computation of high dimensional data, but also avoid the data sparse problem.

Recursive Partitioning

One-layer partitioning processing is not enough to separate points belonging to different data clusters into different grids. Therefore, we need to partition data space recursively to guarantee data points in a grid have similar distribution characteristics, while different dimensions would be selected in each partition. Recursively partitioning stops for a sub-grid if the data points in this grid are below a threshold *max_data* or the depth of this node in TD tree is above a threshold L. A sub-grid is discarded if the data points in this grid are below a threshold *min_data* to alleviate the effect from outliers and noise. Hence, a total of five parameters need to be confirmed by the user:

- *max_data*: the maximal data points that a grid can contains
- *min_data*: the minimal data points that a grid need contains
- L: the threshold of the depth of TD tree
- P: the number of selected dimensions
- M: the number of partitioned segments of candidate interval

Although five parameters need to be defined by users, all those parameters are integers and are easily selected by cross-validation. In practice, the selection of *min_data* and *max_data* rely on the test dataset. Usually, *max_data* is less than the minimal number of data points belong to defined clusters and *min_data* is determined by the proportion of outliers in the dataset. As for parameter L and P, if we select a relatively large P and a small L can guarantee the precision of grid-partitioning, and algorithm runs faster; if we select a relatively small P and a large L can guarantee the precision, and algorithm is more robust. Therefore, we need to make a tradeoff between the running speed and the robustness of the algorithm when we preset L and P. The selection of M is usually unwarranted, while a range [5, 30] can be taken as a reference.

2.2 Bottom-Up for Grid Agglomeration

In this process, the leaf nodes in the TD tree created in the Top-down phase are agglomerated successively by the Bottom-up approach to form a hierarchical structure. In many multilevel grid-based clustering algorithms such as STING [22], grids that belong to different parent grids cannot be merged directly, so that may result in error accumulation. In addition, the traditional 4/8-collection strategy is not suitable for coarse-grained and feature-selected grids in our algorithm. Hence, in BHC, a robust statistical-based distance of any two grids is proposed, so that any leaf grids can be merged. Besides, adopting the corresponding strategy in the DPC, the agglomerative sequence of grids is sorted according to the number of data points in the grids.

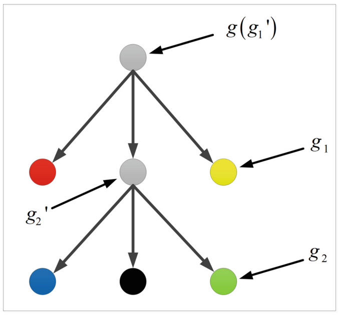

Fig. 8. An example of TD tree

For any two leaf grids $g_{1,}$ g_2 in the TD tree (see in Fig. 8), g is the lowest common ancestor of g_1 and g_2, and the distance of g_1 and g_2 is defined as follows:

$$dis(g_1, g_2) = \sum_{i=1}^{d} \frac{degree_i^g}{faith_i^g} \cdot F_i(g_1, g_2) \tag{6}$$

where $F_i(g_{1,}\ g_2)$ is a discriminant function that determines if g_1 and g_2 are linked in the i-th dimension. The criterion of $F_i(g_{1,}\ g_2)$ in the i-th dimension is whether the distance between the centers μ_i^{g1}, μ_i^{g2} of g_1 and g_2 is greater than the sum of the bandwidth $\lambda_i^{g1'}$, $\lambda_i^{g2'}$ of their parent nodes of their parent $g1'$ and $g2'$ instead of g:

$$F_i(g_1, g_2) = \begin{cases} 1, & \text{if} \left| \mu_i^{g1} - \mu_i^{g2} \right| \geq \lambda_i^{g1'} + \lambda_i^{g2'} \\ 0, & \text{else} \end{cases} \tag{7}$$

In DPC, an attractive idea is of a novel definition of the nearest point: the nearest neighbor of a node is defined as a node with the nearest distance and a higher density concurrently. Besides, data points in DPC are agglomerated in ascending order of their local density, which provides a novel agglomerative chain. This agglomerative chain is more robust in the result than that of the traditional definition of the nearest neighbor chains [26], since more typical data centers are detected.

Based on DPC, in our method, the nearest grid of g_i is defined as a grid g_t with the nearest distance and more data points:

$$t = \mathop{\arg\min}_{j:num\left(g_j\right) > num(g_i)} \left(dis\left(g_j, g_i\right)\right) \tag{8}$$

where *num*(g_j) denotes the number of points in grid g_j. In BHC, leaf grids are agglomerated in ascending order of the number of data points in grids. In other words, the smaller gird agglomerates with its nearest center first, since the effect of mis-agglomeration of smaller grids is more acceptable than that of larger grids. With the grid-partitioning and the agglomerative chain, we are now able to describe BHC generally.

```
Partitioning (dataset $D$, $layer$, $min\_data$, $max\_data$, $L$, $P$, $M$)
Output: grid set $G$;
1. Initialize the grid set $G \leftarrow \phi$;
2. Calculate partitioning degree and faith of all dimen-
   sion of dataset $D$;
3. Determine $P$ dimensions with the maximum degree and
   smaller faith;
4. Construct a $P$-dimensional grid set $G'$ by cutting
   planes perpendicular to the corresponding dimension at
   the point with minimal density;
5. Map all data points $x \in D$ into $G'$ and delete the grid
   with the number of data points less than $min\_data$;
6. For each grid $g \in G'$ do
     i. If $num(g) > max\_data$ and $layer < L$ Then
          a. partitioning ($g$, $layer$+1, $min\_data$,
             $max\_data$, $L$, $P$, $M$);
    ii. Else
          a. insert $g$ into $G$;
```

```
BHC (dataset $D$, $min\_data$, $max\_data$, $L$, $P$, $M$)
1. $G \leftarrow$Partitioning ($D$, 0, $min\_data$, $max\_data$, $L$, $P$, $M$);
2. Foreach pair of grids $g_1$ and $g_2 \in G$ do
     i. Calculate the distance of $g_1$, $g_2$;
3. Ascending sort grids in G according to the number of
   data in grids: $g_1$, $g_2$,… $g_k$;
4. For $i$=1 To $k$-1 do
     i. Find the nearest grid $g_t$ of $g_i$ according to (8);
    ii. Aggregate the two clusters that $g_i$ and $g_t$ belong;
```

3 Experiments

We designed three experiments to demonstrate the superiority of our proposed method. We first benchmark the algorithm on 8 artificial datasets. Then 6 real-world datasets from UCI datasets [23] are tested. At last, we test the efficiency of our method in datasets *Dim-sets* [25] with different sizes and dimensionalities. The concentrated introductions of above test datasets can be found in [24]. Each experiment is compared with several well-known clustering algorithms. Note that the results of unstable clustering algorithms whose clustering results are different with each clustering, including K-means [31], K-medoids [32], K-medians [33], are the averages of the results of repeated 50 times.

3.1 Experiment on Artificial Datasets

In this part, we benchmark the algorithm on 8 artificial datasets, and the results are presented in Fig. 9. The accuracies of our method and 20 well-known clustering algorithms for 8 artificial datasets are shown in Table 1. In Table 1, the corresponding database for each column are *aggregation*, *compound*, *pathbased*, *spiral*, *D31*, *R15*, *jain* and *flame*; the corresponding clustering algorithms for each row are Single [26], Complete [26], UPGMA [26], WPGMA [26], Ward [26], UPGMC [26], WPGMC [26], Rock [27], Chameleon [28], Cure [29], Birch [30], K-means [31], K-medians [33], K-medoids [32], DBSCAN [19], DPC [13], PERCH [11], DSets-DBSCAN [17], SNN-DPC [16], KNN-DPC [14], and our method BHC.

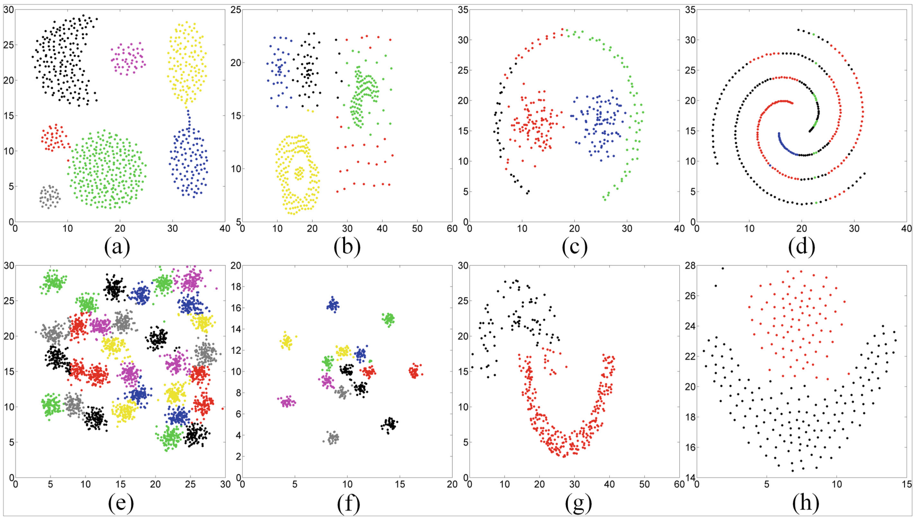

Fig. 9. Results of artificial datasets: (a) *aggregation*, (b) *compound*, (c) *pathbased*, (d) *spiral*, (e) *D31*, (f) *R15*, (g) *jain* and (h) *flame*

In the first column in Table 1 for dataset *aggregation* (see Fig. 9(*a*)) with 7 clusters of different shapes and varying size, UPGMA (100%), UPGMC (100%), KNN-DPC (99.23%) and our method (99.23%) can handle the dataset very well. Database *compound* (see Fig. 9(*b*)) contains 6 clusters of different shapes, sizes, and densities, where the lower-left ring-like cluster encircles another cluster, and the upper-right corner has two compounded clusters of different density. Due to its complex distribution structure, no algorithm can handle database *compound* well, and DSets-DBSCAN (93.98%), BHC (89.72%) perform best. Dataset *pathbased* (see Fig. 9(*c*)) and *spiral* (see Fig. 9

(*d*)) has non-convex clusters with large curvature, which are not what our method can deal with. Only SNN-DPC can cluster *pathbased* well and Single, Chameleon, Cure, DBSCAN, DPC and SNN-DPC perform well in dataset *spiral*. For dataset *D31* (see Fig. 9(*e*)) and *R15* (see Fig. 9(*f*)) with data points located with high overlap, our method successfully divides data points into correct clusters. Similar to *pathbased* and *spiral*, Dataset *jain* (see Fig. 9(g)) and *flame* (see Fig. 9(h)) also has non-convex clusters, while our method achieves considerable performance over those two datasets.

In Table 1, we can find that in artificial datasets with unified dimensions, DPC and its improved algorithms performed stable, which proves their ability to handle complex-shaped clusters. Due to the distortion caused by axes-parallel grid-partitioning, our algorithm cannot achieve best accurate results over test cases. This distortion is more likely to be an inherent defect of grid-partitioning, which can be released by increasing the depth of TD tree. However, a TD tree with a huge depth will reduce the number of data points in the leaf nodes to lose statistical significance, which may affect the precision of grid distance. Even though BHC is not the strongest algorithm when dealing with complex-shaped clusters, our algorithm achieves competitive results on all datasets except *spiral*.

Table 1. Results of artificial datasets

Method	*A*	*C*	*P*	*S*	*D*	*R*	*J*	*F*
Single	83.88	84.46	80.67	**100**	76.03	95.67	99.73	91.25
Complete	91.37	78.70	72.67	39.42	95.42	99.00	86.06	63.75
UPGMA	**100**	86.72	75.33	37.82	96.23	99.50	86.06	63.75
WPGMA	90.86	87.72	67.33	38.78	88.58	98.83	93.83	65.42
Ward	**100**	86.72	73.33	40.38	96.61	99.50	93.83	63.75
UPGMC	86.55	84.21	67.67	35.26	84.48	99.33	91.15	63.75
WPGMC	85.41	75.69	73.33	37.20	75.00	99.17	78.55	96.00
Rock	55.20	61.65	71.67	48.40	11.45	18.33	85.52	92.50
Chameleon	85.41	60.65	69.00	**100**	**97.68**	**99.67**	73.99	97.92
Cure	94.16	88.47	92.67	**100**	50.45	73.33	**100**	97.08
Birch	77.28	85.96	37.33	37.20	90.23	74.83	**100**	93.33
K-means	77.92	71.43	69.33	35.90	81.39	7332	78.55	75.42
K-medians	71.57	63.91	73.33	40.38	79.29	77.17	73.99	63.75
K-medoids	73.85	79.45	65.00	40.38	80.61	81.17	73.99	63.75
DBSCAN	95.18	81.45	68.67	**100**	77.97	97.33	92.49	96.67
DPC	89.72	88.22	78.00	**100**	78.10	99.33	95.17	98.33
Perch	75.29	66.04	69.03	44.70	84.61	95.65	67.86	67.04
DSets-DBSCAN	82.49	**93.98**	94.33	55.77	82.54	72.33	92.76	97.92
SNN-DPC	97.84	86.21	**97.67**	**100**	97.39	**99.67**	87.67	98.75
KNN-DPC	99.23	86.71	65.33	69.87	96.38	**99.67**	70.78	**100**
BHC	99.23	89.72	82.00	59.29	94.74	99.16	95.97	98.33

3.2 Experiment on Real-World Datasets

To illuminate the performance of our method on real-world problems, 6 real-world datasets from UCI datasets: *iris*, *wine*, *glass*, *breast cancer*, *yeast*, *thyroid*, are chosen to verify the effectiveness of BHC. The accuracy of our method and the compared algorithms on those datasets are shown in Table 2. It is obvious in Table 2 that our method is superior to all compared algorithms on all test datasets. Especially in dataset *wine* and *glass*, our algorithm achieved 97.19% and 51.75% clustering accuracy, while the second-best results of rest algorithms were 93.26% (Chameleon, Birch) and 58.88% (K-means) respectively. It proves the advantage of our method to deal with real-world problems. Besides, among these algorithms shown in Table 2, Single, Rock and DBSCAN have the worst performance, even Single and DBSCAN achieved good results on artificial datasets.

Considering Tables 1 and 2 together, we can find that many algorithms that performed well on artificial datasets performed poorly on real-world datasets. For DCP algorithm and its improved algorithm SNN-DPC, KNN-DPC, they achieved considerable results in all artificial datasets due to their ability of handling complex-shaped clusters. However, the dimensions of artificial datasets are simple and uniform in size. So, when applied in real-world datasets, those algorithms needing distance matrix, such as DPC, K-means, performed poorly, as point-to-point distance is difficult to be properly measured. In addition, for DBSCAN and other algorithms that use the global

Table 2. Results of real-world datasets

Method	*I*	*W*	*B*	*Y*	*G*	*T*
Single	82.67	55.05	63.09	31.94	43.46	77.67
Complete	96.00	89.89	79.61	36.05	42.52	85.12
UPGMA	96.67	89.33	91.39	41.37	48.60	81.86
WPGMA	96.00	86.52	92.44	41.51	50.47	69.77
Ward	96.67	84.83	91.39	44.74	43.46	87.44
UPGMC	96.00	88.20	89.81	41.85	47.20	87.44
WPGMC	96.00	65.73	85.59	38.48	45.79	82.79
Rock	66.67	59.55	78.91	34.37	43.46	70.70
Chameleon	96.67	93.26	93.32	43.94	57.48	83.26
Cure	96.67	91.01	91.74	43.80	55.14	92.09
Birch	**97.33**	93.26	91.92	43.16	56.54	88.84
K-means	94.00	92.13	92.27	38.41	58.88	86.98
K-medians	94.00	92.69	92.62	40.78	46.73	93.02
K-medoids	96.00	86.46	92.41	37.69	50.00	84.65
DBSCAN	84.00	60.67	62.92	31.94	45.79	77.67
DPC	96.00	91.01	91.74	37.40	51.40	88.37
Perch	80.13	60.28	74.48	34.47	50.93	57.81
DSets-DBSCAN	96.00	78.88	68.19	48.88	32.24	71.63
SNN-DPC	**97.33**	73.03	90.68	42.72	48.60	78.61
KNN-DPC	96.67	64.04	82.60	37.06	57.01	63.25
BHC	**97.33**	**97.19**	**94.20**	**51.75**	**62.62**	**94.42**

partitioning strategy, stable results can be obtained only when the data is evenly distributed. From Table 2, a conclusion can be drawn that BHC is robust and effective in processing real-world datasets.

3.3 Efficiency Analysis

To demonstrate the efficiency of our method, we performed analysis in datasets with different sizes and dimensionalities. We first test the clustering algorithms in dataset *Dim-sets* with six kind of dimensionalities: 32, 64, 128, 256, 512, 1024. We then draw several various-size datasets by copying the 1024-dimensional data points in *Dim-sets*. Figure 10 shows the running time of clustering algorithms versus dimensionality and data size. Note the run time is recorded when the algorithm obtains a 100% accuracy rate on the dataset.

As shown in Fig. 10(a), the running time has linear dependency with the increase of dimensionality, and Birch have a relatively small growing rate. The reason that the growth rate of running time of our method is higher than Birch, is our method takes a more time-consuming process in each dimension to perform grid-partitioning. As shown in Fig. 10(b), the running time of SINGLE, CHAMELEON, DBSCAN and DPC increases with the size of datasets by exponential growth; the running time of our method and Birch has been kept at a very low level. In addition, with the increase of data size, the running time of Birch exceeds our algorithm gradually.

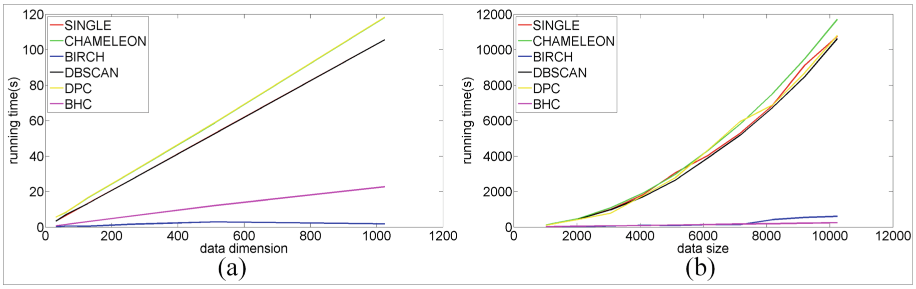

Fig. 10. Running time comparison: (a) with different data dimension, (b) with different data size

Generally, our method and Birch are both good at handling high-dimensional data and big data, while our method is better at big data and Birch is better at high-dimensional data.

4 Conclusions

In this paper we proposed a bidirectional hierarchical clustering algorithm called BHC where the Top-down and Bottom-up processes are performed to form a dendrogram. Through the strategy of over-segmentation and the definition of distance between grids, BHC can process complex and uneven data and avoid the influence of unstable geometric distance. Our method is good at large-scale and high-dimensional data, and is robust to outliers and noise. Experiment results verified the superiority of our method in efficiency and effectiveness over common clustering algorithms. However, one limitation of our algorithm is distortion caused by grid-partition. As a follow-up work, we will develop the kernel function as the projection to deal with nonlinear problems.

Acknowledgment. This work was supported by the Science and Technology Developing Project of Jilin Province No. 20150204007GX, the Key Science-Technology Project of the National Education Ministry of China under Grant No. 02090, the National Natural Science Foundation of China (Nos. 61472159), Project of Guizhou High-level Study Abroad Talents Innovation and Entrepreneurship (2018.0002), and Guangdong Natural Science Foundation (2018A030313339).

References

1. Lin, X., Stur, E., Ekrem, T.: Exploring genetic divergence in a species-rich insect genus using 2790 DNA barcodes. PLoS ONE **10**, e0138993 (2015)
2. Khaldi, N., Wolfe, K.H.: Evolutionary origins of the fumonisin secondary metabolite gene cluster in Fusarium verticillioides and Aspergillus niger. Int. J. Evol. Biol. **2011**, 423821 (2011)
3. Cimermancic, P., et al.: Insights into secondary metabolism from a global analysis of prokaryotic biosynthetic gene clusters. Cell **158**, 412–421 (2014)
4. Yan, C., Zou, X.: Predicting peptide binding sites on protein surfaces by clustering chemical interactions. J. Comput. Chem. **36**, 49–61 (2015)
5. Chu, C.W., Holliday, J.D., Willett, P.: Combining multiple classifications of chemical structures using consensus clustering. Bioorg. Med. Chem. **20**, 5366–5371 (2012)
6. Reddick, R.M., Tinker, J.L., Wechsler, R.H., Lu, Y.: Cosmological constraints from galaxy clustering and the mass-to-number ratio of galaxy clusters: marginalizing over the physics of galaxy formation. Astrophys. J. **783**, 118 (2014)
7. Krumholz, M.R.: The big problems in star formation: the star formation rate, stellar clustering, and the initial mass function. Phys. Rep. **539**, 49–134 (2014)
8. Anderson, L., et al.: The clustering of galaxies in the SDSS-III Baryon Oscillation Spectroscopic Survey: baryon acoustic oscillations in the Data Releases 10 and 11 Galaxy samples. Mon. Not. R. Astron. Soc. **441**, 24–62 (2013)
9. Li, Z., Wang, W., Yang, C., Ragland, D.R.: Bicycle commuting market analysis using attitudinal market segmentation approach. Transp. Res. Part A Policy Pract. **47**, 56–68 (2013)
10. Ramirez, I., Sprechmann, P., Sapiro, G.: Classification and clustering via dictionary learning with structured incoherence and shared features (2010)
11. Kobren, A., Monath, N., Krishnamurthy, A., Mccallum, A.: A hierarchical algorithm for extreme clustering. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 255–264 (2017)
12. Elankavi, R., Kalaiprasath, R., Udayakumar, D.R.: A fast clustering algorithm for high-dimensional data. Int. J. Civ. Eng. Technol. **8**, 1220–1227 (2017)
13. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. Science **344**, 1492–1496 (2014)
14. Du, M., Ding, S., Jia, H.: Study on density peaks clustering based on k-nearest neighbors and principal component analysis. Knowl.-Based Syst. **99**, 135–145 (2016)

15. Wu, B., Wilamowski, B.M.: A fast density and grid based clustering method for data with arbitrary shapes and noise. IEEE Trans. Ind. Inform. **PP**, 1 (2016)
16. Liu, R., Wang, H., Yu, X.: Shared-nearest-neighbor-based clustering by fast search and find of density peaks. Inf. Sci. (NY) **450**, 200–226 (2018)
17. Hou, J., Gao, H., Li, X.: DSets-DBSCAN: a parameter-free clustering algorithm. IEEE Trans. Image Process. **25**, 3182–3193 (2016)
18. Yang, X.-H., et al.: Parameter-free Laplacian centrality peaks clustering. Pattern Recogn. Lett. **100**, 167–173 (2017)
19. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: International Conference on Knowledge Discovery and Data Mining, pp. 226–231 (1996)
20. Hinneburg, A., Keim, D.A.: Optimal grid-clustering: towards breaking the curse of dimensionality in high-dimensional clustering. In: Proceedings of the International Conference on Very Large Data Bases, pp. 506–517 (1999)
21. Keim, D.A., Hinneburg, A.: An efficient approach to clustering in large multimedia databases with noise. In: International Conference on Knowledge Discovery and Data Mining, pp. 58–65 (1998)
22. Wang, W., Yang, J., Muntz, R.R.: STING+: an approach to active spatial data mining. In: Proceedings of the International Conference on Data Engineering, pp. 116–125 (1999)
23. Dheeru, D., Karra Taniskidou, E.: UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. Accessed 29 Nov 2018
24. Fränti, P., Sieranoja, S.: K-means properties on six clustering benchmark datasets. J. Appl. Intell. **48**, 4743–4759 (2018)
25. Fränti, P., Virmajoki, O., Hautamaki, V.: Fast agglomerative clustering using a k-nearest neighbor graph. IEEE Trans. Pattern Anal. Mach. Intell. **28**, 1875–1881 (2006)
26. Murtagh, F., Contreras, P.: Algorithms for hierarchical clustering: an overview. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **2**, 86–97 (2012)
27. Guha, S., Rastogi, R., Shim, K.: ROCK: a robust clustering algorithm for categorical attributes. Inf. Syst. **25**, 345–366 (2000)
28. Karypis, G., Han, E.-H., Kumar, V.: Chameleon: hierarchical clustering using dynamic modeling. Computer (Long. Beach. Calif) **32**, 68–75 (1999)
29. Guha, S., Rastogi, R., Shim, K.: CURE: an efficient clustering algorithm for large databases. ACM SIGMOD Rec. **27**, 73–84 (1998)
30. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: an efficient data clustering method for very large databases. ACM SIGMOD Rec. **25**, 103–114 (1996)
31. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297 (1967)
32. Kaufman, L., Rousseeuw, P.: Clustering by Means of Medoids. North-Holland, Amsterdam (1987)
33. Bradley, P.S., Fayyad, U.M., Reina, C., et al.: Scaling clustering algorithms to large databases. In: KDD 1998, pp. 9–15 (1998)