Knowledge-Based Systems 99 (2016) 135–145

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Study on density peaks clustering based on k-nearest neighbors and principal component analysis

Mingjing Du[a,b], Shifei Ding[a,b,*], Hongjie Jia[a,b]

[a] School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China
[b] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100090, China

ARTICLE INFO

Article history:
Received 11 July 2015
Revised 31 January 2016
Accepted 1 February 2016
Available online 9 February 2016

Keywords:
Data clustering
Density peaks
k Nearest neighbors (KNN)
Principal component analysis (PCA)

ABSTRACT

Density peaks clustering (DPC) algorithm published in the US journal Science in 2014 is a novel clustering algorithm based on density. It needs neither iterative process nor more parameters. However, original algorithm only has taken into account the global structure of data, which leads to missing many clusters. In addition, DPC does not perform well when data sets have relatively high dimension. Especially, DPC generates wrong number of clusters of real-world data sets. In order to overcome the first problem, we propose a density peaks clustering based on k nearest neighbors (DPC-KNN) which introduces the idea of k nearest neighbors (KNN) into DPC and has another option for the local density computation. In order to overcome the second problem, we introduce principal component analysis (PCA) into the model of DPC-KNN and further bring forward a method based on PCA (DPC-KNN-PCA), which preprocesses high-dimensional data. By experiments on synthetic data sets, we demonstrate the feasibility of our algorithms. By experiments on real-world data sets, we compared this algorithm with k-means algorithm and spectral clustering (SC) algorithm in accuracy. Experimental results show that our algorithms are feasible and effective.

1. Introduction

Clustering, used mostly as an unsupervised learning method, is a major technique for data mining. The main aim of cluster analysis is to divide a given population into groups or clusters with common characteristics, since similar objects are grouped together, while dissimilar objects belong to different clusters. Clustering is useful in exploratory pattern-analysis, grouping, decision-making, and machine-learning situations, including data mining, document retrieval, image segmentation, and pattern classification [1]. Clustering methods are generally divided into five groups: hierarchical clustering, partitioning clustering, density-based clustering, grid-based clustering and model-based clustering [2]. Each method has its own strengths and weaknesses.

Density-based clustering [3–7] is represented by DBSCAN [3]. In density-based clustering, clusters are defined as areas of higher density than the remainder of the data set. Density-based clusters can have an arbitrary shape in the feature space. In addition, DBSCAN does not require one to specify the number of clusters in the data a priori. However, it is very sensible to the user-defined parameter values, often producing very different clustering results in the data set even for slightly different parameter settings [2].

Like DBSCAN and the mean-shift method [8], density peaks clustering (DPC) algorithm [9] proposed by Rodriguez and Laio is able to detect non-spherical clusters and does not require one to specify the number of clusters. This method is robust with respect to the choice of d_c as the only parameter. DPC is based on the idea that cluster centers are characterized by a higher density than their neighbors and by a relatively large distance from points with higher densities. Several researches [10–15] have been going on around this method.

But DPC still has some defects. The local structure of data has not been taken into account in DPC when it calculates the local density. For example, DPC does not perform well when clusters have different densities. Having clusters of different densities is very common in data sets. The local density of DPC will lead to missing many clusters. Fig. 1 presents that clusters cannot be all detected with the local density of DPC. If p is small, two clusters in the lower-left corner are detected as a single cluster. However, if p is high, two clusters near the bottom are detected as a single cluster. In this case, DPC is not able to find clusters.

In order to overcome this problem, we propose a novel DPC based on k nearest neighbors (DPC-KNN). The proposed method makes use of the ideas of the k nearest neighbors for the local density computation.

* Corresponding author at: School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China. Tel.: +86 15852494396.
E-mail address: dingsf@cumt.edu.cn (S. Ding).

http://dx.doi.org/10.1016/j.knosys.2016.02.001
0950-7051/

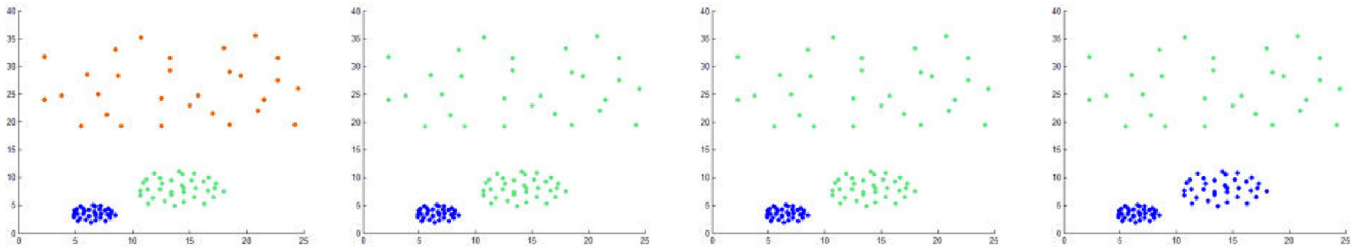


Fig. 1. DPC on these clusters of different densities.

In addition, it does a poor job of finding the clusters of high-dimensional data. It may generate wrong number of clusters of real-world data sets. This is because many of the dimensions in high dimensional data are often irrelevant. These irrelevant dimensions can confuse DPC by hiding clusters in noisy data. Another reason is its overwhelming dependence on the distances between points. Two main quantities of DPC are both relevant to the distances. And for this reason, the problem, "the curse of dimensionality", is exacerbated. As the number of dimensions in a dataset increases, distance measures become increasingly meaningless [16]. Additional dimensions spread out the points until, in very high dimensions, they are almost equidistant from each other. More specific details are shown in Section 4. On the basis of the former, we further bring forward a method based on principal component analysis (DPC-KNN-PCA).

We test our algorithms on synthetic data sets to demonstrate their feasibility. In order to assess the performance of proposed algorithms, we compare proposed algorithms with other algorithms on some UCI data sets. Our algorithms have achieved satisfactory results in most data sets. The rest of this paper is organized as follows. In Section 2, we describe the principle of the DPC method, and introduce the k nearest neighbors and principal component analysis. In Section 3, we make a detailed description of DPC-KNN and DPC-KNN-PCA. In Section 4, we present experimental results in synthetic data sets and UCI data sets, then we analyze the performance of proposed algorithms. Finally, some conclusions and the intending work are given in the last section.

2. Related works

The proposed DPC-KNN is based on DPC and KNN. The proposed DPC-KNN-PCA is based on former theories and PCA. This section provides brief reviews of DPC, KNN, and PCA.

2.1. Density peaks clustering

Rodriguez and Laio proposed an algorithm published in the US journal Science. Its idea is that cluster centers are characterized by a higher density than their neighbors and by a relatively large distance from points with higher densities [9]. This method utilizes two important quantities: One is the local density ρ_i of each point i, and the other is its distance δ_i from points of higher density. The two quantities correspond to two assumptions with respect to the cluster centers. One is that the cluster centers are surrounded by neighbors with a lower local density. The other is that they have relatively larger distance to the points of higher density. In the following, we will describe the computation of ρ_i and δ_i in much more detail.

Assume that the data set is $\boldsymbol{X}_{N\times M} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]^T$, where $\boldsymbol{x}_i = [\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i}, \boldsymbol{x}_{Mi}]$ is the vector with M attributes and N is the number of points. The distance matrix of the data set needs to be computed first. Let $\mathrm{d}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ denote the Euclidean distance between the point $\boldsymbol{x}_i$ and the point $\boldsymbol{x}_j$, as follows:

$$\mathrm{d}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \|\boldsymbol{x}_i - \boldsymbol{x}_{j_2}\| \tag{1}$$

The local density of a point $\boldsymbol{x}_i$, denoted by ρ_i, is defined as:

$$\rho_i = \sum_j \chi\left(\mathrm{d}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - d_c\right)$$

$$\chi(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases} \tag{2}$$

where d_c is a cutoff distance. ρ_i is defined as the number of points that are adjacent to point $\boldsymbol{x}_i$. There is another local density computation in the code presented by Rodriguez and Laio. If the former is called a hard threshold, the latter will be called a soft threshold. Specifically, ρ_i is defined as a Gaussian kernel function, as follows:

$$\rho_i = \sum_j exp\left(-\frac{\mathrm{d}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)^2}{{d_c}^2}\right) \tag{3}$$

where d_c is an adjustable parameter, controlling the weight degradation rate.

d_c is the only variable in Formulas (2) and (3). The process for selecting d_c is actually that for selecting the average number of neighbors of all points in data set. In the code, d_c is define as:

$$d_c = d_{N_d \times \frac{p}{100}} \tag{4}$$

where $N_d = \binom{N}{2}$ and $d_{N_d \times \frac{p}{100}} \in D = [d_1, d_1, \ldots, d_{N_d}]$. D is a set of all the distances between every two points in data set, which are sorted in ascending order. N denotes the number of points in data set. $N_d \times \frac{p}{100}$ is the subscript of $d_{N_d \times \frac{p}{100}}$, where $[\ \cdot\ ]$ is the ceiling function and p is a percentage.

The computation of δ_i is quite simple. The minimum distance between the point of $\boldsymbol{x}_i$ and any other points with higher density, denoted by δ_i, is defined as:

$$\delta_i = \begin{cases} \min\limits_{j:\rho_i > \rho_j} \left(\mathrm{d}(\boldsymbol{x}_i, \boldsymbol{x}_j)\right), & if\ \exists j\ s.t. \rho_i > \rho_j \\ \max\limits_{j} \left(\mathrm{d}(\boldsymbol{x}_i, \boldsymbol{x}_j)\right), & otherwise \end{cases} \tag{5}$$

Only those points with relative high ρ_i and high δ_i are considered as cluster centers. The points with high ρ_i and δ_i value are also called as peaks that have higher densities than other points. A point is assigned to the same cluster as its nearest neighbor peak.

After cluster centers have been found, DPC assigns each remaining points to the same cluster as its nearest neighbors with higher density. A representation named as decision graph is introduced to help one to make a decision. This representation is the plot of δ_i as a function of ρ_i for each point.

The following algorithm is a summary of DPC.

Algorithm 1. DPC algorithm.

Inputs:
The samples $\boldsymbol{X} \in \mathbb{R}_{N\times M}$
The parameter d_c
Outputs:
The label vector of cluster index: $\boldsymbol{y} \in \mathbb{R}_{N\times 1}$
Method:
Step 1: Calculate distance matrix according to Formula (1)
Step 2: Calculate ρ_i for point i according to Formula (2) or (3)
Step 3: Calculate δ_i for point i according to Formula (5)
Step 4: Plot decision graph and select cluster centers
Step 5: Assign each remaining point to the nearest cluster center
Step 6: **Return y**

2.2. k Nearest neighbors

k Nearest neighbors (KNN) has already been exploited for classification [17–21]. This approach has been shown to be a powerful technique for density estimation [22], clustering [23–25] and other fields. As the name implies, the goal of this approach is to find the k-nearest neighbors of a sample among N samples. In general, the distances between points are achieved by calculating the Euclidean distance.

To compute the local density preferably, we use an idea based on k nearest neighbors (KNN).

We always assume that we are given N data points $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]^T$ which have been drawn in M dimensional space. As distance function between points we use the Euclidean distance, which is denoted by $\mathrm{d}(\cdot, \cdot)$. It is required to compute the k nearest neighbors of a sample $\boldsymbol{x}_i$ among $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N\}$. Sorting these distances in ascending order is to find the first K distances. The kth distance corresponds to the kth nearest neighbor. By $\mathrm{kNN}(\boldsymbol{x}_i)$ we denote the set of the k nearest neighbors of $\boldsymbol{x}_i$ among $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N\}$. More specifics of the idea are discussed in Section 3.

The nearest neighbor step is simple to implement, though it scales in the worst case as $\mathrm{O}(N^2)$, if performed without any optimizations. Some efficient methods such as K-D tree [26] or ball trees [27] can be used to compute the neighbors in $\mathrm{O}(N\log N)$ time.

2.3. Principal components analysis

Principal components analysis (PCA) is a dimensionality reduction algorithm that can be used to significantly speed up unsupervised feature learning algorithm. The basic idea of PCA is to project the original data onto a lower-dimensional subspace, which highlights the principal directions of variation of the data.

The following steps describe this algorithm procedure.

(1) Make each of the features have the same mean (zero) and variance.
(2) Calculate the covariance matrix Σ.
(3) Calculate the eigenvectors $\boldsymbol{u}_i$ and the eigenvalues λ_i of Σ.
(4) Sort these eigenvalues in decreasing order and stack the eigenvectors $\boldsymbol{u}_i$ corresponding to the eigenvalue λ_i in columns to form the matrix $\mathbf{U}$.

3. Density peaks clustering based on KNN and density peaks clustering based on KNN and PCA

There are still some defects in DPC. To solve these problems, we propose the following solutions.

3.1. Density peaks clustering based on k nearest neighbors

Firstly, the local structure of data is not assessed by the local density in DPC. For this, we propose a novel DPC based on k nearest neighbors (DPC-KNN).

A shortcoming of the local density is that it is not sensitive to the local geometric of the data. Especially, when there is a great difference between the clusters in the density, there is a great difference between cluster centers on the local density. If d_c is so low that the distances between cluster centers are distinguished on decision graph, it is hard to select cluster centers. In our work, we introduce the idea of KNN into the calculation of the local density.

Let $\boldsymbol{x}_i \in \boldsymbol{X}$, $\mathrm{d}(\cdot, \cdot)$ the Euclidean distance function, and the $\mathrm{NN}_k(\boldsymbol{x}_i)$ be the kth nearest point to $\boldsymbol{x}_i$ according to d, the k-nearest neighbors ($\mathrm{kNN}(\cdot)$) of $\boldsymbol{x}_i$ is defined as:

$$\mathrm{kNN}(\boldsymbol{x}_i) = \left\{ j \in X | \mathrm{d}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \le \mathrm{d}(\boldsymbol{x}_i, \mathrm{NN}_k(\boldsymbol{x}_i)) \right\} \tag{6}$$

We can use $\mathrm{kNN}(\boldsymbol{x}_i)$ to calculate the local density. This new local density is calculating the mean distance to k nearest neighbors, as follows:

$$\rho_i = exp\left(-\left(\frac{1}{k} \sum_{x_j \in \mathrm{kNN}(\boldsymbol{x}_i)} \mathrm{d}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)^2 \right) \right) \tag{7}$$

where k is computed as a percentage (p) of the number of points N, so $k = p \times N$. Because a higher value of the local density means a higher density, so Formula (7) is a Gaussian kernel function as an inverse measure of the distance.

Then we can adopt the idea of KNN to the local density in DPC. The following algorithm is a summary of the proposed DPC-KNN.

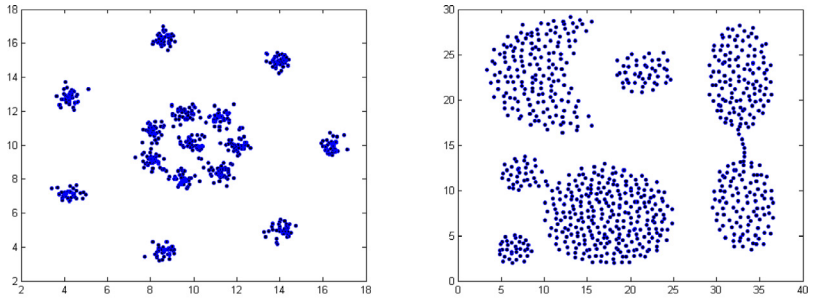
Algorithm 2. DPC-KNN algorithm.

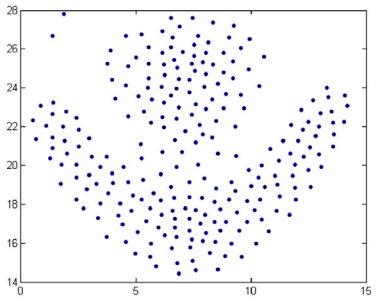
Inputs:
The samples $\boldsymbol{X} \in \mathbb{R}_{N\times M}$
The parameter p
Outputs:
The label vector of cluster index: $\boldsymbol{y} \in \mathbb{R}_{N\times 1}$
Method:
Step 1: Calculate distance matrix according to Formula (1)
Step 2: Calculate ρ_i for point i according to Formula (7)
Step 3: Calculate δ_i for point i according to Formula (5)
Step 4: Plot decision graph and select cluster centers
Step 5: Assign each remaining point to the nearest cluster center
Step 6: **Return y**

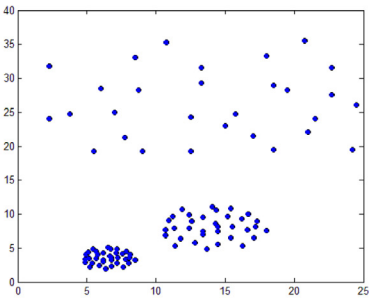
Complexity Analysis: Suppose N is the total number of points in data set. The complexity in calculating the similarity matrix is $\mathrm{O}(N^2)$. DPC-KNN also needs $\mathrm{O}(N^2)$ to compute the local density. In addition, we cost $\mathrm{O}(N\log N)$ in the sorting process with quick sort. For the progress to determine the cluster centers, we take no account of the time. As the complexity in assignment procedure is $\mathrm{O}(N)$, the total time complexity of our DPC-KNN method is $\mathrm{O}(N^2) + \mathrm{O}(N^2) + \mathrm{O}(N\log N) + \mathrm{O}(N) + \mathrm{O}(N) \sim \mathrm{O}(N^2)$.

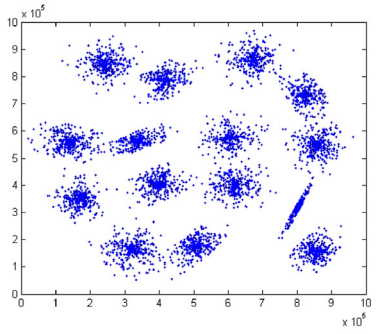
3.2. Density peaks clustering based on k nearest neighbors and principal component analysis

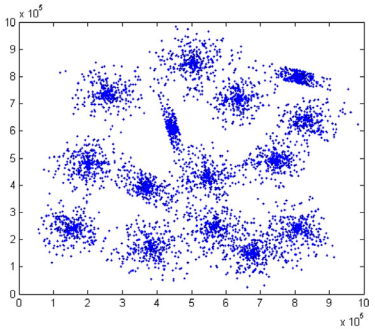
In addition, DPC does not perform well when data has relatively high dimension. Especially, in real-world data sets, DPC generates wrong number of clusters of real-world data sets. On the basis of DPC-KNN, we further bring forward a method based on principal component analysis (DPC-KNN-PCA).

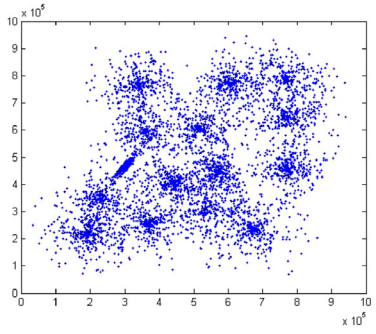












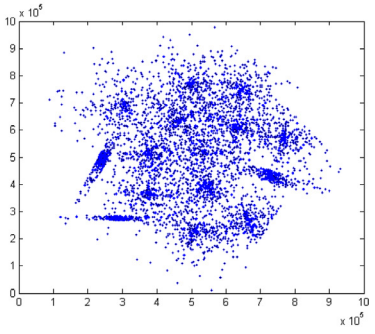


Fig. 2. Visualization of two-dimensional data sets.

Firstly, we make each of the features have the same mean (zero). Then, compute the matrix Σ as follows:

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T \tag{8}$$

If $\mathbf{x}$ has zero mean, then Σ is exactly the covariance matrix of $\mathbf{x}$. We can compute the eigenvectors of Σ, and stack the eigenvectors in columns to form the matrix $\mathbf{U}$:

$$\mathbf{U} = \begin{bmatrix} | & | & | & | \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_M \\ | & | & | & | \end{bmatrix} \tag{9}$$

where $\boldsymbol{u}_1$ is the principal eigenvector (corresponding to the largest eigenvalue), $\boldsymbol{u}_2$ is the second eigenvector, and so on. Also, let $\lambda_1, \lambda_1, \ldots \cdots, \lambda_M$ be the corresponding eigenvalues.

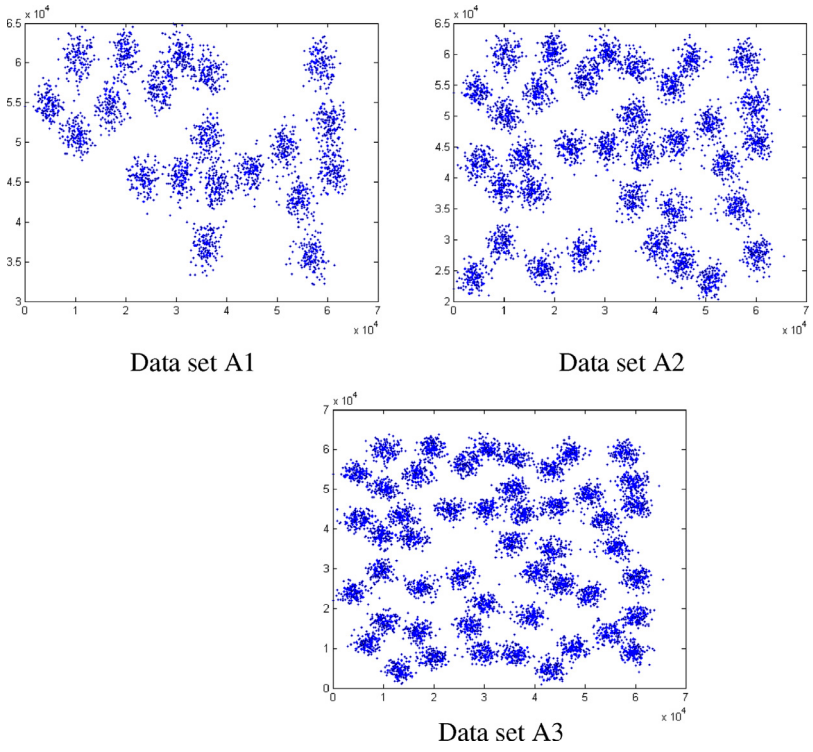


Fig. 2. (Continued).

We can represent $\mathbf{x}$ in the $(\boldsymbol{u}_1, \boldsymbol{u}_1, \cdots\cdots, \boldsymbol{u}_M)$-basis by computing

$$\mathbf{x}_{rot} = \boldsymbol{U}^T\boldsymbol{x} = \begin{bmatrix} \boldsymbol{u}_1^T\boldsymbol{x} \\ \boldsymbol{u}_2^T\boldsymbol{x} \\ \vdots \\ \boldsymbol{u}_M^T\boldsymbol{x} \end{bmatrix} \tag{10}$$

The subscript "rot" comes from the observation that this corresponds to a reflection of the original data. We can compute $\boldsymbol{x}_{rot}^{(i)} = \boldsymbol{U}^T\boldsymbol{x}^{(i)}$ for every point i. If we want to reduce this data to one dimension (the principal direction of variation of the data), we can set

$$\tilde{\boldsymbol{x}}^{(i)} = \boldsymbol{x}_{rot,1}^{(i)} = \boldsymbol{u}_1^T\boldsymbol{x}^{(i)} \in \mathbb{R} \tag{11}$$

If $\mathbf{x}$ in $\mathbb{R}^M$ and we want to reduce it to a k dimensional representation $\tilde{\boldsymbol{x}} \in \mathbb{R}^k$ (where $k < M$), we would take the first k components of $\mathbf{x}_{rot}$, which correspond to the top k directions of variation. In other words, our definition of $\tilde{\boldsymbol{x}}$ can also be arrived at by using an approximation to $\mathbf{x}_{rot}$ where all but the first k components are zero, as follows:

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} \mathbf{x}_{rot,1} \\ \vdots \\ \mathbf{x}_{rot,k} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \approx \begin{bmatrix} \mathbf{x}_{rot,1} \\ \vdots \\ \mathbf{x}_{rot,k} \\ \mathbf{x}_{rot,k+1} \\ \vdots \\ \mathbf{x}_{rot,M} \end{bmatrix} = \mathbf{x}_{rot} \tag{12}$$

The reason that it drops the later components of $\mathbf{x}_{rot}$ is that the first few components are considerably larger than the later components.

To decide how to set k, we will usually look at the percentage of variance retained for different values of k. Generally, let $\lambda_1, \lambda_1, \cdots, \lambda_M$ be the eigenvalues of Σ (sorted in decreasing order), so that λ_i is the eigenvalue corresponding to the eigenvector $\boldsymbol{u}_i$. Then if we retain k principal components, the percentage of variance retained is given by:

$$\frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{M}\lambda_i} \tag{13}$$

In this paper, we pick the smallest value of k that satisfies

$$\frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{M}\lambda_i} \geq 0.99 \tag{14}$$

The following algorithm is a summary of the proposed DPC-KNN-PCA.

Algorithm 3. DPC-KNN-PCA algorithm.

Inputs:
The samples $\boldsymbol{X} \in \mathbb{R}_{N\times M}$
The parameter p
Outputs:
The label vector of cluster index: $\boldsymbol{y} \in \mathbb{R}_{N\times 1}$
Method:
Step 1: Make all of the features have the same mean (zero) and variance
Step 2: Compute the covariance matrix Σ according to Formula (8)
Step 3: Compute the eigenvectors $\boldsymbol{u}_i$ and the eigenvalues λ_i of Σ
Step 4: Reduce the data and keep 99 % principal component to Formula (14)
Step 5: Calculate distance matrix according to Formula (1)
Step 6: Calculate ρ_i for point i according to Formula (7)
Step 7: Calculate δ_i for point i according to Formula (5)
Step 8: Plot decision graph and select cluster centers
Step 9: Assign each remaining point to the nearest cluster center
Step 10: **Return y**

Complexity Analysis: Suppose N is the total number of points in data set and M is the number of features of each point. Covariance matrix computation is $O(M^2N)$. Its eigenvalues decomposition is. So, $O(M^3)$ the complexity of PCA is $O(M^3 + M^2N)$. The total time complexity of our DPC-KNN-PCA method is $O(M^3 + M^2N + N^2)$.

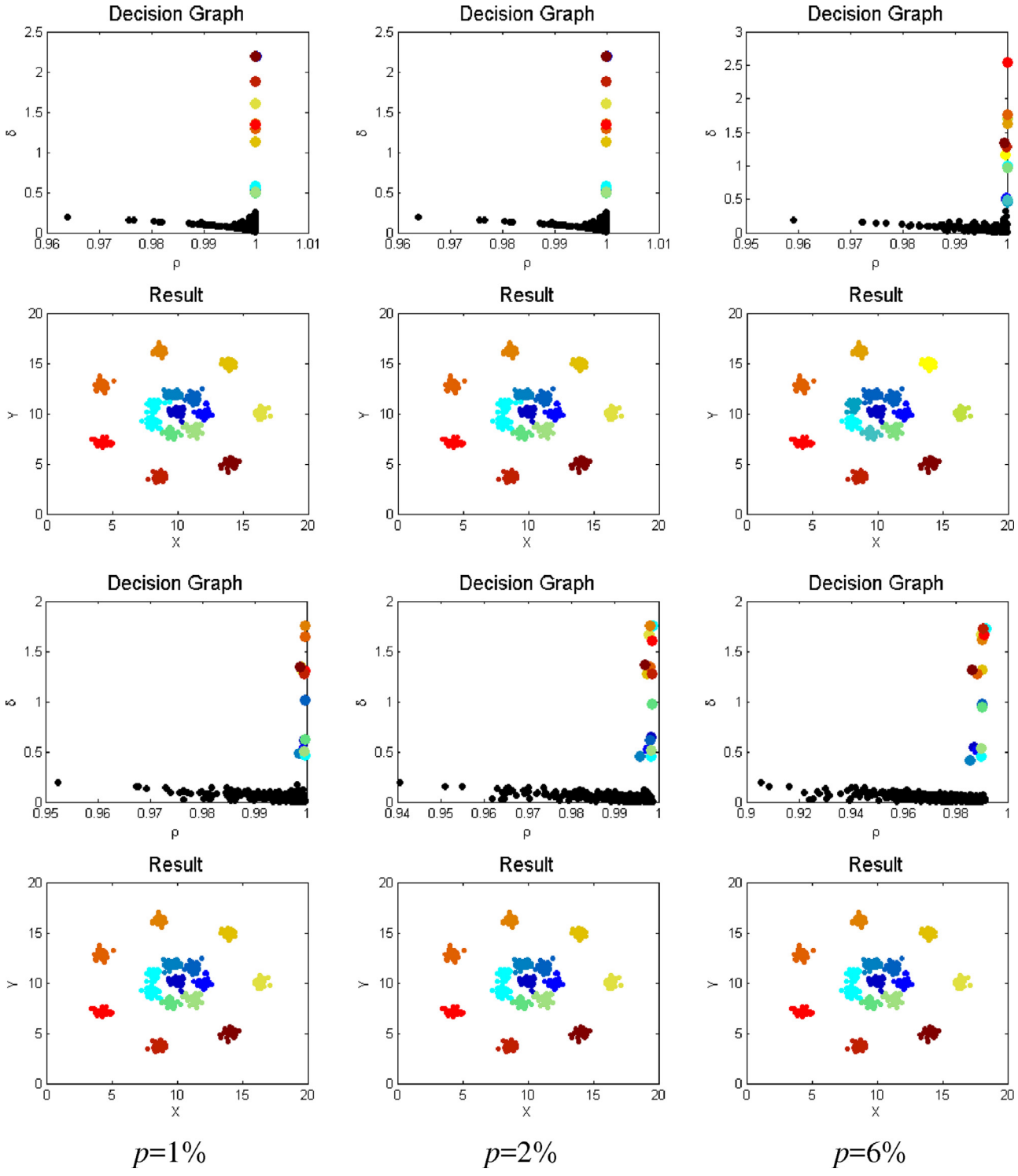


Fig. 3. DPC-KNN-PCA on R15 set with different values of *p*.

4. Experiments and results

In this section, we will test the performance of DPC-KNN-PCA through two types of the experiments. By experiments on synthetic data sets, we demonstrate the feasibility of the algorithm. By experiments on real-world data sets, we compared this algorithm with k-means algorithm, spectral clustering (SC) algorithm in accuracy.

We do experiments in a work station with a core i7 DMI2-Intel 3.6 GHz processor and 18 GB RAM running MATLAB 2012B. We run k-means algorithm, SC algorithm, 10 times in real-world data sets. This paper measures the similarity between data points with the famous Euclidean distance, which is used widely to measure the similarity of spatial data, as shown Formula (1). In DPC-KNN and DPC-KNN-PCA, we select the parameter p from [0.1% 0.2% 0.5% 1% 2% 6%]. In DPC, the parameter d_c is also selected from [0.1% 0.2%

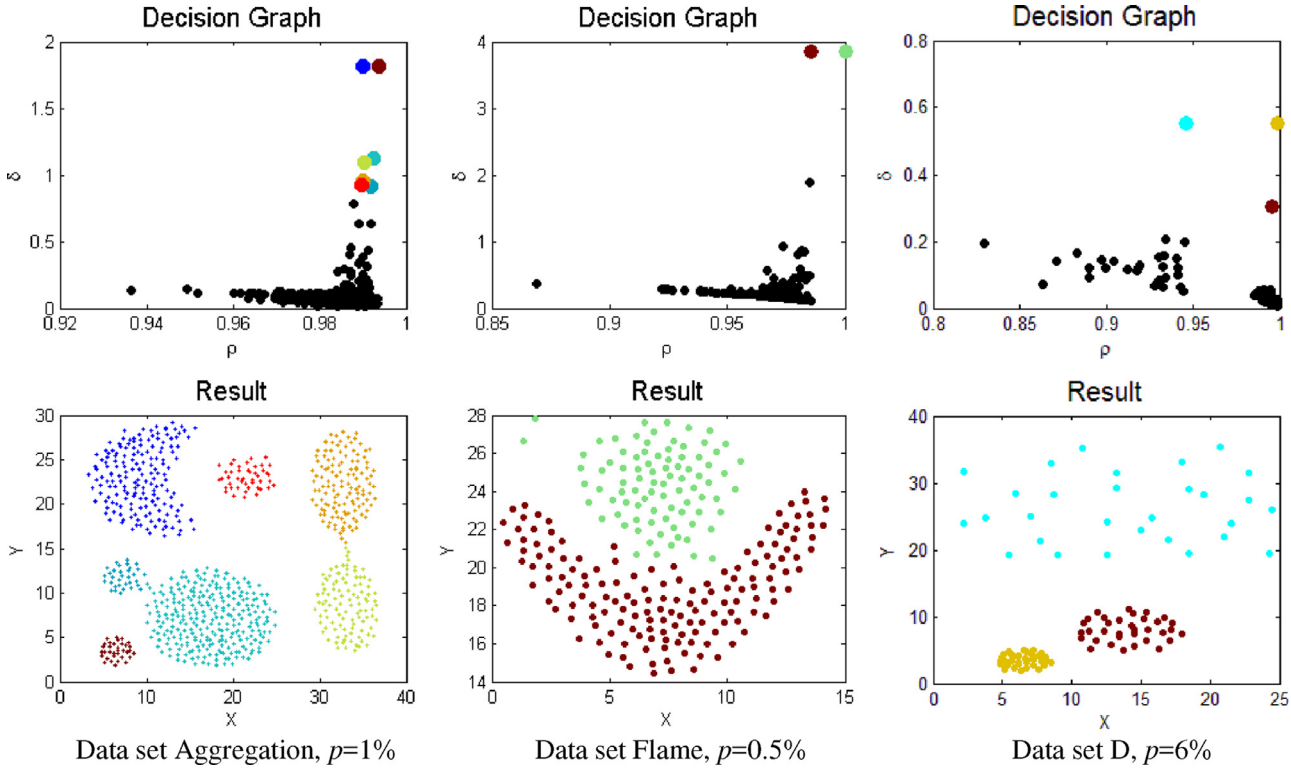


Fig. 4. DPC-KNN-PCA on Aggregation and Flame sets.

0.5% 1% 2% 6%]. The kernel of spectral clustering (SC) algorithm is the Gaussian kernel. In SC, we select the parameter δ from [0.5 1 2 3 4].

4.1. Experiments on synthetic data sets

We test the performance of our algorithms on synthetic data sets. The synthetic data sets are two dimensional, which makes things easy from the visualization point of view. Because the performances of DPC-KNN and DPC-KNN-PCA are similar in 2-dimensional data sets, we only test the performance of DPC-KNN-PCA.

4.1.1. Synthetic data sets

Our algorithm is tested by 11 data sets whose geometric shapes are shown in Fig. 2. The first data set, R15 [28] is generated as 15 similar 2-D Gaussian distributions that are positioned in rings. The second data set, aggregation [29] consists of the seven perceptually distinct groups of points where there are non-Gaussian clusters. The third data set, Flame [30], is of different size and shape. The fourth data set, *D*, is of clusters of different densities. S1, S2, S3 and S4 [31] are four 2-D data sets with varying complexity in terms of spatial data distributions: i.e., in S1 the overlap is the smallest, whereas in S4 the overlap is the greatest. A1, A2 and A3 [32] are two-dimensional sets with varying number of circular clusters ($M = 20,\ 35,\ 50$). We demonstrate the power of these algorithms on these test cases.

4.1.2. The evaluation of clustering results in synthetic data sets

There are 15 clusters, 600 points in R15. In this case, our approaches are robust respect to parameter *p*. Fig. 3 presents a lot of clusters found by our algorithms with different *p* and selections of cluster centers. As shown, DPC-KNN-PCA gets perfect results, except when $p = 0.5\%$.

There are 2 clusters, 240 points in Aggregation set and are 7 clusters, 788 points in Flame set. The two data sets consist of some clusters that are of different size and shape. D set has 2 clusters and 97 points. Its clusters are of different densities. Clustering results proposed by DPC-KNN-PCA have been given in Fig. 4.

The data sets S1 to S4 are two-dimensional sets with varying complexity in terms of spatial data distributions. The data sets have 5000 points around 15 clusters with a varying degree of overlap. As shown in Fig. 5, the performance of DPC-KNN-PCA is perfect for data sets with varying complexity.

A1, A2 and A3 sets are large data sets with varying number of clusters. A1 has 3000 points around 20 clusters. A2 has 5250 points around 35 clusters. There are 7500 points and 50 clusters on A3. We demonstrate the robustness of the DPC-KNN-PCA for the quantity, as shown in Fig. 6.

As these experiments illustrate our algorithms are very effective in finding clusters of arbitrary shape, density, distribution and number.

4.2. Experiments on real-world data sets

The performances of our algorithms are compared with classical methods (k-means algorithm and spectral clustering algorithm).

4.2.1. Real-world data sets

The data sets used in the experiments are all from the UCI Machine Learning Repository, which include Iris, LED digits, Seeds, Heart, Pen-based digits, Waveform, Sonar. The details of those data sets are given in Table 1.

4.2.2. Quality of the clustering results

This paper uses clustering accuracy (ACC) [33] to measure the quality of the clustering results. For *N* distinct samples $\mathbf{x}_i \in \mathbb{R}^j$, y_i and c_i are the inherent category label and the predicted cluster label of $\mathbf{x}_i$, the calculation formula of ACC is

$$\mathrm{ACC} = \sum_{i=1}^{N} \delta(y_i, map(c_i))/N \tag{15}$$

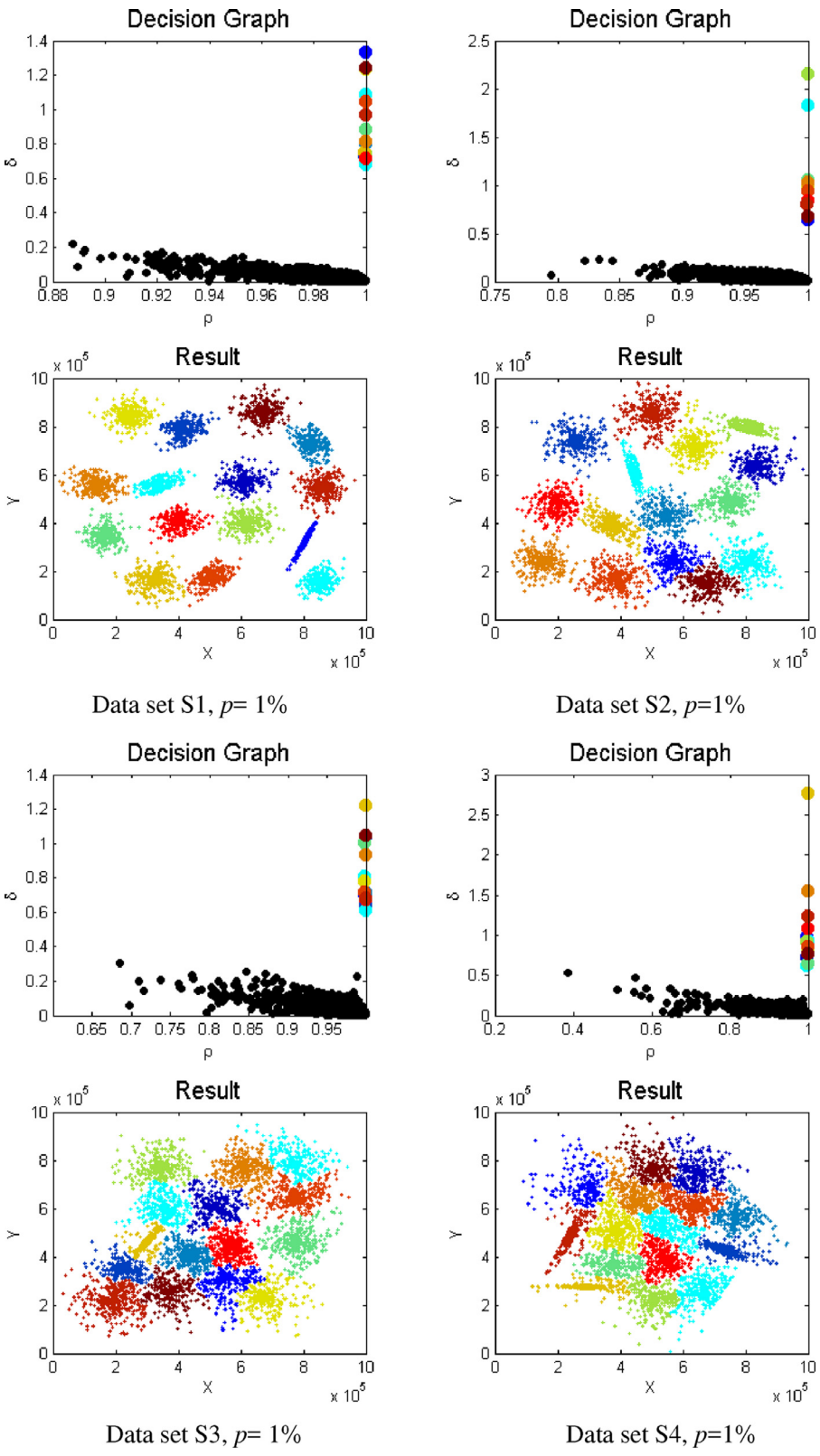


Fig. 5. DPC-KNN-PCA on the S1, S2, S3, and S4 sets.

where *map*(·) maps each cluster label to a category label by the Hungarian algorithm [34] and this mapping is optimal, let $\delta(y_i, c_i)$ equal to 1 if $y_i = c_i$ or equals to 0 otherwise. The higher the values of the ACC are, the better the clustering performance will be.

The comparison of these algorithms is shown in Table 2. In Table 2, the symbol – means that the algorithm cannot work in the data set.

There are less than ten features in the first three data sets. And the last two data sets have more than 10 features. As shown

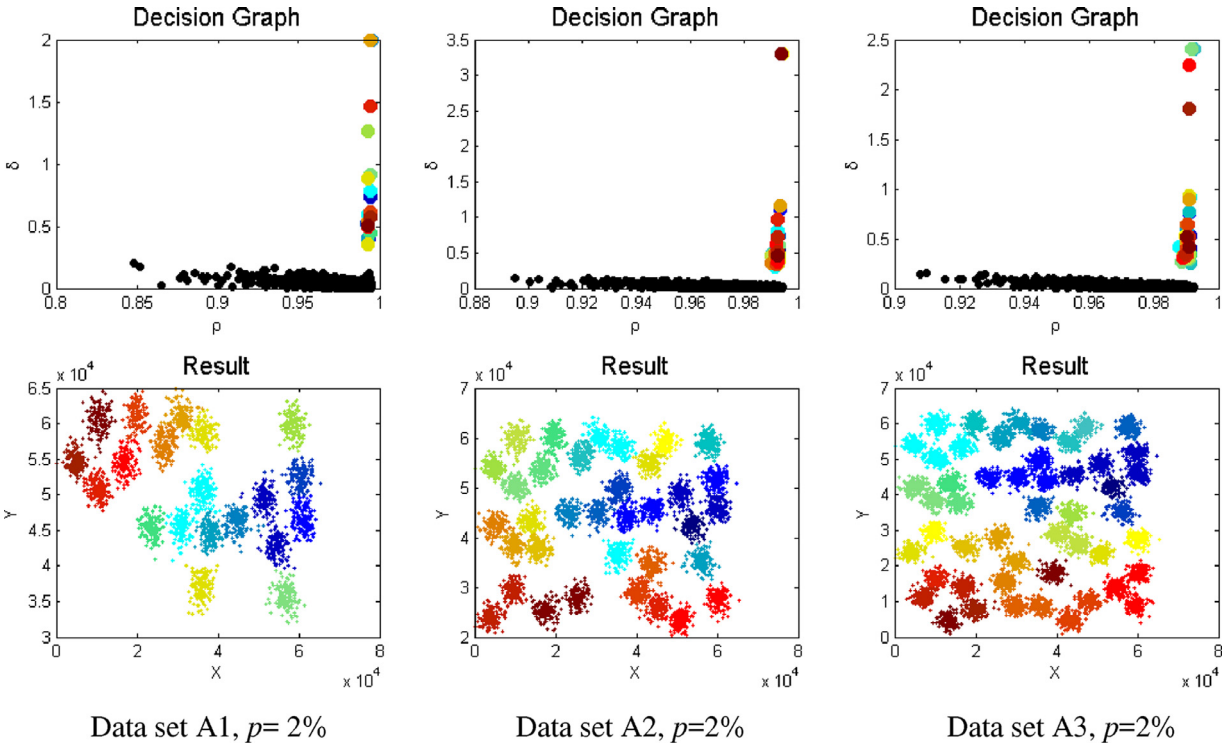


Fig. 6. DPC-KNN-PCA on the A1, A2, and A3 sets.

Table 1
The details of UCI data sets.

Data sets	Cluster	Dimension	*N*
Iris	3	4	150
LED digits	10	7	500
Seeds	3	7	210
Heart	2	13	270
Pen-based digits	10	16	109,962
Waveform	3	21	5000
Sonar	2	60	208

in Table 2, DPC-KNN outperforms others in low-dimensional data sets such as Iris, LED digits and Seeds. However, DPC-KNN-PCA has a better performance compared to DPC-KNN in relatively high-dimensional data sets. To be specific, the higher the features of the data set are, the greater advantage DPC-KNN-PCA has over DPC-KNN. It is obvious that DPC-KNN-PCA algorithm has achieved gratifying results in most data sets.

Then, the reason that produced above results is discussed. DPC-KNN, without missing any information of features, performed better than DPC-KNN-PCA in low-dimensional data set. Due to the phenomenon of the so called curse of dimensionality, the similarity between samples becomes meaningless in high dimensional data spaces. Although similarity between high dimensional samples is not meaningful, similarity according to subsets of attributes is still meaningful. PCA not only reduces the dimensionality of the data, but also maintains as much information as possible. When some data sets have relatively high dimensions, DPC does a poor job of finding the clusters, which we need to pay extra attention to. For example, Sonar data set has relatively high dimensions compared to the number of samples. Fig. 7 shows only one cluster center found by DPC with different d_c on decision graph. In this case, it is unacceptable that we were incapable of making the right choices. DPC-KNN-PCA has a favorable performance compared to the

Table 2
The performance comparison of proposed algorithms.

Data sets	Accuracy	DPC	DPC-KNN	DPC-KNN-PCA	SC	k-means
Iris	Mean	0.94	0.96	0.88	0.8867 ± 0	0.8560 ± 0.097
	Parameter	$d_c = 0.1\%$	$p = 1\%$	$p = 4\%$	$\delta = 0.5$	
LED digits	Mean	–	0.7460	0.6700	0.6360 ± 0.0552	0.5208 ± 0.0713
	Parameter		$p = 6\%$	$p = 6\%$	$\delta = 0.5$	
Seeds	Mean	0.8952	0.9143	0.9143	0.9048 ± 0	0.8905 ± 0
	Parameter	$d_c = 1\%$	$p = 2\%$	$p = 2\%$	$\delta = 1$	
Heart	Mean	–	0.8111	0.8259	0.7963 ± 0	0.7166 ± 0.0640
	Parameter		$p = 1\%$	$p = 6\%$	$\delta = 4$	
Pen-based digits	Mean	–	0.7618	0.7623	0.7177 ± 0.0276	0.7004 ± 0.0466
	Parameter		$p = 1\%$	$p = 0.2\%$	$\delta = 3$	
Waveform	Mean	0.5676	0.5840	0.6452	0.5054 ± 0	0.5012 ± 0
	Parameter	$d_c = 0.5\%$	$p = 0.2\%$	$p = 0.1\%$	$\delta = 0.5$	
Sonar	Mean	–	–	0.6442	0.5433 ± 0	0.5433 ± 0.0124
	Parameter			$p = 1\%$	$\delta = 2$	

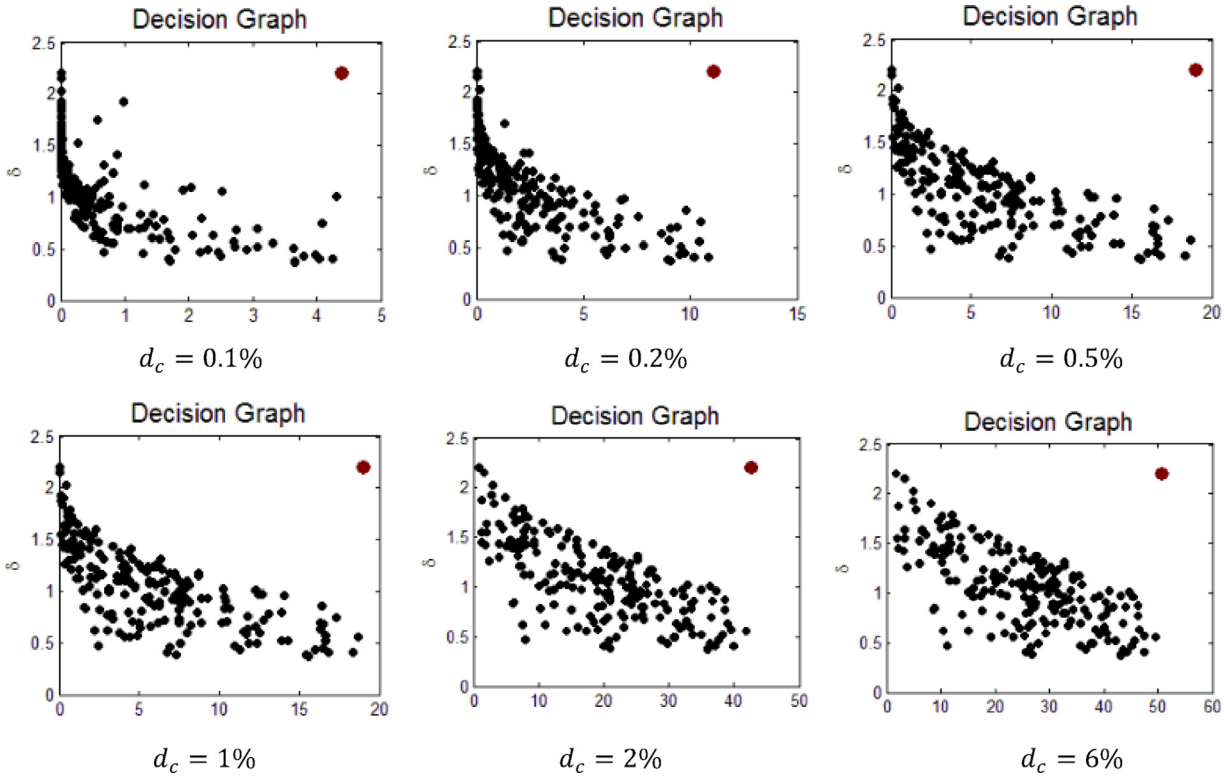


Fig. 7. DPC on Sonar set with different values of d_c.

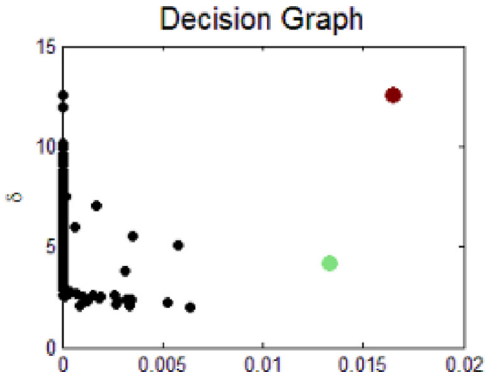


Fig. 8. DPC-KNN-PCA on Sonar set.

original algorithm, as shown Fig. 8. In consequence, DPC-KNN-PCA outperforms DPC-KNN in high-dimensional data sets.

5. Conclusions

DPC-KNN has another option based on k nearest neighbors (KNN) for the local density computation. On the basis of DPC-KNN, a method based on principal component analysis (DPC-KNN-PCA) is presented to improve the performance of the former on real-world data sets. In this paper, presented algorithms show the power in some synthetic data sets. Besides the good feasibility, proposed algorithms get better clustering performances compared to classical methods (k-means algorithm and spectral clustering algorithm) and DPC in UCI data sets. In low-dimensional data sets, DPC-KNN outperforms others. And DPC-KNN-PCA has achieved gratifying results in relatively high-dimensional data sets.

However, the proposed algorithm does not perform well when there is a collection of points forming vertical streaks in data set. Future research is to improve the performance of DPC-KNN-PCA algorithm on manifold data set.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61379101), and the National Key Basic Research Program of China (No. 2013CB329502).

References

[1] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (3) (1999) 264–323.
[2] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufman, San Francisco, California, USA, 2000.
[3] M. Ester, H.P. Kriegel, J. Sander, X.W. Xu., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of Second International Conference on Knowledge Discovery and Data Mining, 96, 1996, pp. 226–231.
[4] J. Sander, M. Ester, H.P. Kriegel, X. Xu, Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications, Data Min. Knowl. Discov. 2 (2) (1998) 169–194.
[5] M. Ankerst, M.M. Breunig, H.-P. Kriegel, J. Sander, OPTICS: ordering points to identify the clustering structure, in: Proceedings of ACM SIGMOD Conference, 28, 1999, pp. 49–60.
[6] X. Xu, M. Ester, H.P. Kriegel, J. Sander, A distribution-based clustering algorithm for mining in large spatial databases, in: Proceedings of the 14th International Conference on Data Engineering, 1998, pp. 324–331.
[7] R.J.G.B. Campello, D. Moulavi, J. Sander, Density-based clustering based on hierarchical density estimates, in: Proceedings of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2013, pp. 160–172.
[8] Y. Cheng, Mean shift, mode seeking, and clustering, IEEE Trans. Pattern Anal. Mach. Intell. 17 (8) (1995) 790–799.
[9] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[10] K. Sun, X. Geng, L. Ji, Exemplar component analysis: a fast band selection method for hyperspectral imagery, IEEE Geosci. Remote Sens. Lett. 12 (5) (2015) 998–1002.
[11] Y. Zhang, Y. Xia, Y. Liu, W.M. Wang, Clustering sentences with density peaks for multi-document summarization, in: Proceedings of Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL, 2015, pp. 1262–1267.

[12] K. Xie, J. Wu, W. Yang, C.Y. Sun, K-means clustering based on density for scene image classification, in: Proceedings of the 2015 Chinese Intelligent Automation Conference, 2015, pp. 379–438.
[13] Y.W. Chen, D.H. Lai, H. Qi, J.L. Wang, J.X. Du, A new method to estimate ages of facial image for large database, Multimed. Tools Appl. (2015) 1–19, doi:10.1007/s11042-015-2485-9.
[14] W. Zhang, J. Li, Extended fast search clustering algorithm: widely density clusters, no density peaks. arXiv:1505.05610, 2015, doi:10.5121/csit.2015.50701.
[15] G. Chen, X. Zhang, Z.J. Wang, F.L. Li, Robust support vector data description for outlier detection with noise or uncertain data, Knowledge-Based Syst. 90 (2015) 129–137.
[16] L. Parsons, E. Haque, H. Liu, Subspace clustering for high dimensional data: a review, ACM SIGKDD Explor. Newslett. 6 (1) (2004) 90–105.
[17] H.L. Chen, B. Yang, G. Wang, J. Liu, X. Xu, S.J. Wang, D.Y. Liu, A novel bankruptcy prediction model based on an adaptive fuzzy k-nearest neighbor method, Knowledge-Based Syst. 24 (8) (2011) 1348–1359.
[18] T. Basu, C.A. Murthy, Towards enriching the quality of k-nearest neighbor rule for document classification, Int. J. Mach. Learn. Cybern. 5 (6) (2014) 897–905.
[19] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.
[20] M. Hao, Z. Qiao, Identification of the pesticide fluorescence spectroscopy based on the PCA and KNN, in: Proceedings of 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), 3, 2010, pp. 184–186.
[21] P.S. Hiremath, M. Hiremath, 3D face recognition based on radon transform, PCA, LDA using KNN and SVM, Int. J. Image Graph. Signal Process. 6 (7) (2014) 36–43.
[22] D.O. Loftsgaarden, C.P. Quesenberry, A nonparametric estimate of a multivariate density function, Ann. Math. Stat. 36 (1) (1965) 1049–1051.
[23] R. Jarvis, E. Patrick, Clustering using a similarity measure based on shared near neighbors, IEEE Trans. Comput. 100 (11) (1973) 1025–1034.
[24] J. Schneider, M. Vlachos, Fast parameterless density-based clustering via random projections, in: Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management, 2013, pp. 861–866.
[25] E. Aksehirli, B. Goethals, E. Muller, J. Vreeken, Cartification: A neighborhood preserving transformation for mining high dimensional data, in: Proceedings of the 13th IEEE International Conference on Data Mining (ICDM), 2013, pp. 937–942.
[26] J.L. Bentley, Multidimensional binary search trees used for associated searching, Commun. ACM 18 (9) (1975) 509–517.
[27] S.M. Omohundro, Five Balltree Construction Algorithms. Technical Report TR-89-063, International Computer Science Institute, 1989.
[28] C.J. Veenman, M.J.T. Reinders, E. Backer, A maximum variance cluster algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 24 (9) (2002) 1273–1280.
[29] A. Gionis, H. Mannila, P. Tsaparas, Clustering aggregation, ACM Trans. Knowl. Discov. Data 1 (4) (2007) 341–352.
[30] L. Fu, E. Medico, FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data, BMC Bioinf. 8 (1) (2007) 399–408.
[31] P. Fränti, O. Virmajoki, Iterative shrinking method for clustering problems, Pattern Recognit. 39 (5) (2006) 761–775.
[32] I. Kärkkäinen, P. Fränti, Dynamic Local Search Algorithm for the Clustering Problem. Technical Report A-2002-6, Department of Computer Science, University of Joensuu, 2002.
[33] S.F. Ding, H.J. Jia, Z.Z. Shi, Spectral clustering algorithm based on adaptive Nyström sampling for big data analysis, J. Softw. 25 (9) (2014) 2037–2049.
[34] C.H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Courier Dover Publications, Mineola, New York, USA, 1998.