Information Sciences 354 (2016) 19–40

Contents lists available at ScienceDirect

Information Sciences

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

Robust clustering by detecting density peaks and assigning points based on fuzzy weighted *K*-nearest neighbors

Juanying Xie[a,*], Hongchao Gao[a], Weixin Xie[b], Xiaohui Liu[c], Philip W. Grant[d]

[a] School of computer science, Shaanxi Normal University, Xi'an 710062, PR China
[b] School of Information Engineering, National Key Laboratory of ATR, Shenzhen University, Shenzhen 518006, PR China
[c] Department of Computer Science, Brunel University, London UB8 3PH, UK
[d] Department of Computer Science, College of Science, Swansea University, Singleton Park, Swansea SA2 8PP, UK

ARTICLE INFO

Article history:
Received 14 May 2015
Revised 4 February 2016
Accepted 6 March 2016
Available online 12 March 2016

Keywords:
Local density
Density peaks
Clustering
K-nearest neighbors
Fuzzy weighted *K*-nearest neighbors

ABSTRACT

Clustering by fast search and find of Density Peaks (referred to as DPC) was introduced by Alex Rodríguez and Alessandro Laio. The DPC algorithm is based on the idea that cluster centers are characterized by having a higher density than their neighbors and by being at a relatively large distance from points with higher densities. The power of DPC was demonstrated on several test cases. It can intuitively find the number of clusters and can detect and exclude the outliers automatically, while recognizing the clusters regardless of their shape and the dimensions of the space containing them. However, DPC does have some drawbacks to be addressed before it may be widely applied. First, the local density ρ_i of point i is affected by the cutoff distance d_c, and is computed in different ways depending on the size of datasets, which can influence the clustering, especially for small real-world cases. Second, the assignment strategy for the remaining points, after the density peaks (that is the cluster centers) have been found, can create a "Domino Effect", whereby once one point is assigned erroneously, then there may be many more points subsequently misassigned. This is especially the case in real-word datasets where there could exist several clusters of arbitrary shape overlapping each other. To overcome these deficiencies, a robust clustering algorithm is proposed in this paper. To find the density peaks, this algorithm computes the local density ρ_i of point i relative to its *K*-nearest neighbors for any size dataset independent of the cutoff distance d_c, and assigns the remaining points to the most probable clusters using two new point assignment strategies. The first strategy assigns non-outliers by undertaking a breadth first search of the *K*-nearest neighbors of a point starting from cluster centers. The second strategy assigns outliers and the points unassigned by the first assignment procedure using the technique of fuzzy weighted *K*-nearest neighbors. The proposed clustering algorithm is benchmarked on publicly available synthetic and real-world datasets which are commonly used for testing the performance of clustering algorithms. The clustering results of the proposed algorithm are compared not only with that of DPC but also with that of several well known clustering algorithms including Affinity Propagation (AP), Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and *K*-means. The benchmarks used are: clustering accuracy (Acc), Adjusted Mutual Information (AMI) and Adjusted Rand Index (ARI). The experimental results demonstrate that our proposed clustering algorithm can find cluster centers, recognize clusters regardless of their shape and dimension of the space in which they are embedded, be unaffected by outliers, and can often outperform DPC, AP, DBSCAN and *K*-means.

* Corresponding author. Tel.: +86 13088965815.
E-mail address: xiejuany@snnu.edu.cn, juanyingxie@gmail.com (J. Xie).

http://dx.doi.org/10.1016/j.ins.2016.03.011
0020-0255/

1. Introduction

Clustering is the process of grouping objects together according to their similarities, so that objects in the same cluster are similar to one another and dissimilar to objects in any other cluster [12,15,18,27,30,33]. Clustering is important for uncovering the inherent, potential and unknown knowledge, principles or rules in the real-world, and has been widely used in scientific and engineering applications [12,15,18,27,30,33]. With the emergence of *big data*, there is an ever increasing interest in clustering algorithms that can automatically understand, process and summarize the data [9,29].

Many different methods of clustering exist including partitioning, hierarchical, density-based, grid-based, or combinations of these [15,33]. One very popular partitioning method is the *K-means* clustering algorithm [18,24]. It is well known that *K*-means is heavily dependent on the initial cluster centers or initial partitions, and is unable to find clusters with arbitrary shape, with clustering easily affected by noise or outliers. Furthermore, the value of K must be pre-specified [15,18,33]. The *Global K-means* algorithm and its variations were proposed to overcome the disadvantages of *K*-means [21,32], and other algorithms developed to remedy the sensitivity of *K*-means to the outliers [4,16,20]. However, because a data point is always assigned to the nearest center, these approaches are not able to detect nonspherical clusters. Clusters with arbitrary shape are easily detected by approaches based on the local density of data points. Density-Based Spatial Clustering of Applications with Noise (*DBSCAN*) [8] is the typical density-based clustering algorithm which can detect clusters with any arbitrary shape only if the density thresholds are specified, such as ϵ the neighborhood radius and *MinPts* the minimum number of points included in the neighborhood with radius ϵ [15,17,33]. However, choosing the appropriate thresholds may be nontrivial. A relative density based *K*-nearest neighbors clustering algorithm was introduced in [22] to remedy these limitations of DBSCAN.

Affinity Propagation (*AP*) [12] is another well-known clustering algorithm which takes as input measures of similarity between pairs of data points. Real-valued messages are exchanged between data points until a high-quality set of exemplars and corresponding clusters gradually emerge. AP takes as input a real number $s(k, k)$ (where $s(i, k)$ is the similarity indicating how well the data point with index k is suited to be the exemplar for data point i) for each data point k so that data points with larger values of $s(k, k)$ are more likely to be chosen as exemplars. These values are referred to as *preferences*. The number of identified exemplars (number of clusters) is influenced by the values of the input preferences. However, by simultaneously considering all data points as candidate exemplars and gradually identifying clusters by a message-passing procedure, AP is able to identify the exemplars and detect the clusters of a dataset. AP was tested as a simple and efficient clustering algorithm, especially for cases where the number of the identified exemplars is relatively large [31], but it does not work well for clusters with arbitrary shape. A new hierarchical clustering technique based on synchronization was proposed by Huang et al. [17]. It was reported that the clustering algorithm can detect the clusters of any size and shape. However, the drawbacks of hierarchical clustering cannot be avoided.

Recently, a novel clustering algorithm was proposed by Alex Rodríguez and Alessandro Laio [27], based on the assumption that cluster centers are surrounded by the neighbors with lower local density and that they are at a relatively large distance from any points with a higher local density. We refer to this algorithm as *DPC* (Density Peak Clustering) in this paper. It was demonstrated on several test cases that DPC can efficiently find the cluster centers (i.e., the density peaks) and assign the remaining points to their appropriate clusters as well as detect outliers. However, DPC does have some shortcomings. First, the density metric is different for large and small datasets, and the arbitrarily selected cutoff distance d_c can greatly influence the clustering of a small dataset, and furthermore no criterion is given for determining whether a dataset is small or large. Secondly, after the density peaks have been found, the strategy of assigning the remaining points to the same cluster as its nearest neighbor of higher density can cause propagation of errors. Once a point i is assigned to a wrong cluster, then there may be several points with lower density than point i being assigned to incorrect clusters, producing poor clustering. This is specially true for real-world datasets which usually contain highly overlapped clusters of arbitrary shape. Such is the case for the commonly used test dataset Iris, contained in the UCI machine learning repository [1], on which DPC performs poorly.

In order to remedy these problems with DPC, we introduce several modifications to the basic algorithm. Our new algorithm defines the local density ρ_i of point i based on its *K*-nearest neighbors, so that the deficiencies of DPC in defining the local density of a point can be avoided. Two new assignment strategies are proposed based respectively on the idea of *K*-nearest neighbors and fuzzy weighted *K*-nearest neighbors. We refer to our proposed algorithm as *FKNN-DPC* (Fuzzy weighted *K*-Nearest Neighbors Density Peak Clustering).

The new features of our FKNN-DPC are (i) a uniform local density metric is proposed based on the *K*-nearest neighbors, so that the local density ρ_i of point i can be computed by one density metric independent of the size of the dataset, and the density peaks (cluster centers) will be found efficiently and correctly; and (ii) two new strategies for assigning the remaining points to their most likely clusters are introduced. These new strategies are applied consecutively. The points assigned by strategy one comprise the core of the clusters, and the other points assigned by strategy two are the *halo* of the clusters. The core together with halo make up the whole cluster. The power of our FKNN-DPC to find the density peaks and detect the clusters of a dataset will be demonstrated in Section 4 of this paper.

This paper is organized as follows: Section 2 briefly describes the idea of DPC and analyzes its deficiencies by displaying its clustering on a synthetic dataset which is often used to test the performance of a clustering algorithm. Section 3 introduces our robust clustering algorithm and gives a detailed analysis. Section 4 tests our proposed algorithm on several synthetic and real-world datasets, and compares its performance with DPC, AP, DBSCAN and *K*-means in terms of several

very popular criteria for testing a clustering algorithm, namely clustering accuracy (Acc), adjusted mutual information (AMI), and adjusted rand index (ARI). Section 5 draws some conclusions.

2. DPC algorithm and its analysis

Clustering by fast search and find of density peaks (DPC) was published recently in Science [27] and has become of interest to many machine learning researchers. DPC can detect cluster centers and exclude outliers and recognize the clusters regardless of their shape and the dimension of the space in which they are embedded. The algorithm builds on an assumption that the ideal cluster centers are characterized by the following two properties: (i) they are surrounded by neighbors with lower local density; and (ii) they are at a relatively large distance from any points with a higher local density. In order to find the ideal cluster centers, DPC introduces the *local density metric* of a point i defined by (1) and the *distance* δ_i of point i defined by (2) [27].

$$\rho_i = \sum_{j \neq i} \chi(d_{ij} - d_c) \tag{1}$$

$$\delta_i = min_{j:\rho_j > \rho_i}(d_{ij}) \tag{2}$$

where d_{ij} is the Euclidean distance between data points i and j, and d_c is a *cutoff* distance. $\chi(x) = 1$ if $x < 0$, otherwise $\chi(x) = 0$. So, the local density ρ_i of point i is equal to the number of points j that are closer to i than d_c. DPC is sensitive only to the relative magnitude of ρ_i at different points, implying that, for large data sets, the results are robust with respect to the choice of d_c [27]. From (2), we can see that δ_i is the minimum distance between point i and any other point j of higher density. It should be noted that for the point i with the highest density we conventionally take δ_i to be given by (3) [27]. It is can be seen from (2) and (3) that δ_i is much larger than the typical nearest neighbor distance only for points that are local or global maxima with respect to the density. DPC searches for the density peaks as cluster centers thus are recognized as points for which the value of δ_i is anomalously large.

$$\delta_i = max_j\ d_{ij} \tag{3}$$

How does DPC search and find the density peaks, i.e., the cluster centers? The critical technique used in DPC is to plot the decision graph which is the plot of δ_i as a function of ρ_i for each point (i.e., the collection of points (ρ_i, δ_i)). Cluster centers can then be detected by analysis of the decision graph, as the only points of high δ and relatively high ρ. After the cluster centers have been found, each of the remaining points are assigned to the same cluster as its nearest neighbor of higher density. The cluster assignment is performed in a single step, which results in efficient execution of DPC.

However, there are cases where the 'ideal' number of clusters is comparable with the number of elements in the dataset, making a reliable estimate of the densities (density peaks) difficult. As a consequence of the sparsity of the data points, the exact number of density peaks is unclear, so a hint for choosing the number of centers is provided by the plot of $\gamma_i = \rho_i \delta_i$ sorted in decreasing order [27]. In addition, the density is estimated by a Gaussian kernel with a pre-specified cutoff distance d_c given by (4) [27], and a slightly more restrictive criterion is used to assign an instance to a cluster, such as in [27] for the Olivetti face database from [28]. An image is assigned to the same cluster of its nearest image with higher density only if their distance is smaller than d_c, and the images further away than d_c from any other image of higher density remain unassigned.

$$\rho_i = \sum_{j \neq i} exp\left(-\left(\frac{d_{ij}}{d_c}\right)^2\right) \tag{4}$$

DPC does not introduce a noise-signal cutoff. Instead, a border region for each cluster is defined, which is the set of points assigned to that cluster which are within a distance d_c from points belonging to other clusters. For each cluster the point of highest density within its border region is then found and its density is denoted by ρ_b. The points of the cluster whose densities are higher than ρ_b are considered to be the cluster core, and the other points are the cluster halo which can be understood to be noise [27].

From the results of testing DPC on several different datasets it is seen that it can perform well in many instances, though the following drawbacks are apparent:

(i) DPC uses different methods to determine the local density of points in datasets of different sizes. As a rule of thumb, (1) is used to calculate the local density ρ_i of point i when the dataset is composed of a large number of points. Otherwise the exponential kernel, described in [5], is used instead, with a pre-specified cutoff distance d_c, such as that in (4). However, there is no objective metric for deciding whether the dataset is small or large and this choice can produce very different clusterings. In addition, although for large datasets, the results of DPC are robust with respect to the choice of d_c, this is not the case for small data sets, where the clustering results of DPC can be greatly affected by the cutoff distance d_c.

(ii) the assignment strategy for remaining points after the cluster centers have been found may cause a kind of "Domino Effect" so that once a point is assigned erroneously, then this may propagate so that more points will be assigned incorrectly.

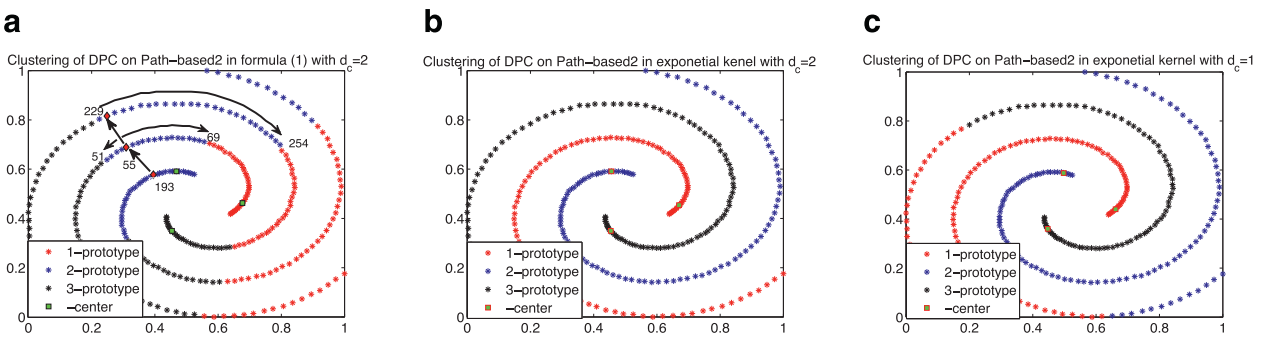

Fig. 1. The clustering of Path-based2 by DPC. (a) Local density for points is calculated in (1) with $d_c = 2$, (b) exponential kernel in (4) is used to calculate the local density for points with $d_c = 2$, (c) exponential kernel in (4) is used to calculate the local density for points with $d_c = 1$.

Fig. 1(a) and (b) show the clusterings produced by DPC on Path-based2, taken from [3], using the different metrics in (1) and (4) to determine the density of a point with the same cutoff distance d_c. Fig. 1(b) and (c) display the clusterings of DPC on Path-based2 using the metric in (4) with different cutoff distance d_c. It is evident that the clustering in Fig. 1(b) is much better than that in Fig. 1(a) and (c). Although the cluster centers are quite similar, the clusterings are completely different, which shows that the clustering of DPC is influenced both by the metrics used to calculate the density of a point and by the cutoff distance d_c.

To see what is going wrong in case (a), consider the assignment strategy used by DPC after the cluster centers have been found assigning each remaining point to the same cluster as its nearest neighbor of higher density. This will lead to the fact that if the density ρ_i of point i is relatively large compared to that of its neighbors', then the point i must be surrounded by its neighbors, so that the distance between point i and its true cluster point, the nearest neighbor of point i of higher density, may become larger than that from point i to another cluster point of higher density, which will result in the erroneous assignment for point i. For example, in Fig. 1(a), point 55 has the local highest density in the set of points $S = \{51, \ldots, 69\}$. The distance from point 55 to its true cluster point, its nearest neighbor of higher density, is larger than that from point 55 to another cluster point of higher density, say, point 193, so the point 55 will be wrongly assigned to the same cluster as point 193. This will result in all points in S being assigned erroneously. For the same reason the points in $\{229, \ldots, 254\}$ will be assigned to the wrong clusters because of the error in assignment of point 230.

The results in Fig. 1(b) and (c) demonstrate that the choice of cutoff distance d_c will heavily influence the clustering of DPC. All the results in Fig. 1 demonstrate that the clusterings produced by DPC not only depend on the metric to calculate the densities of points, but also on the cutoff distance d_c. The poor results of DPC in Fig. 1 are as a consequence of the potential issues hidden in its very simple and efficient strategy for assigning the remaining points after the cluster centers have been found.

To remedy these limitations in DPC, we propose a robust clustering algorithm, Fuzzy weighted K-Nearest Neighbors Density Peak Clustering (*FKNN-DPC*), in this paper. This algorithm is based on using the K-nearest neighbors of a point to fast search and find the density peaks (i.e., the cluster centers) semi-automatically by computing the local density for points in a general metric without any influence from cutoff distance, and assigning remaining points using two proposed strategies, in particular using the fuzzy weighted K-nearest neighbors of a point in the second strategy to assign a point to its most probable cluster. The uniform local density calculation can be used for any size of dataset large or small. FKNN-DPC was tested on several synthetic and real-world datasets which are often used to test the performance of a clustering algorithm. The results demonstrate that FKNN-DPC can recognize the true distribution of a dataset and has improved performance over that of DPC.

3. FKNN-DPC algorithm

We will improve DPC in two respects by using a uniform definition for the local density of a point and changing the assignment strategy for the remaining points after the cluster centers have been found. The main idea of our algorithm is based on using the K-nearest neighbors of a point to compute its local density and using the K-nearest neighbors and the fuzzy weighted K-nearest neighbors of a point in turn to assign the remaining points to their correct clusters after the cluster centers have been found. This section will present the essential details of the proposed clustering algorithm and analyze its complexity and its clustering performance theoretically.

3.1. The main idea of FKNN-DPC

FKNN-DPC uses the exponential kernel with constant width $\delta(= 1)$ to define the local density for points. The local density ρ_i of a point i combines the information from its neighbors so that it to some extent can embody the distribution of the neighbors of point i. The local density ρ_i of point i is defined by (5), where KNN_i is the set of the K nearest neighbors of

point i, and d_{ij} is the Euclidean distance between points i and j.

$$\rho_i = \sum_{j \in KNN_i} exp(-d_{ij}) \tag{5}$$

This definition can guarantee that the local density ρ_i of point i will be calculated using the distribution information of its K nearest neighbors, while the original definition of the local density ρ_i of point i in [27] depends on the cutoff distance d_c, and it cannot guarantee the number of points which are closer than cutoff distance d_c to point i when calculating the local density ρ_i of point i. Furthermore to determine the parameter K is easier than determining the cutoff distance d_c in [27] to ensure the average number of neighbors of a point is between 1 and 2% of the total number of points in the dataset.

Comparing (5) with (1), it can be seen that the evaluation of the local density for point i in FKNN-DPC only depends on the K points in KNN_i, whereas the density of point i in DPC depends on the *whole* dataset. As a consequence the complexity of computing the density of a point in FKNN-DPC in general is much less than that in DPC without considering the cost of searching for the K nearest neighbors of point i, that is the KNN_i is assumed to be known in advance.

The cluster centers are first found by constructing the decision graph as for DPC but using ρ_i given by (5). Then two strategies respectively based on KNN and fuzzy weighted KNN are proposed and in turn used to assign the remaining points to their proper clusters. To avoid two clusters being merged into one cluster by the influence of outliers, FKNN-DPC will first delete the outliers before assigning the remaining points, and will assign the outliers using a special strategy. Outliers are defined as follows. First, if X is the whole dataset of size N, then the KNN distance δ_i^K of a point i is defined by (6) [19], and if the threshold τ is given by (7) (the mean of the KNN distances), then the set of outliers is defined by (8).

$$\delta_i^K = max_{j \in KNN_i} d_{ij} \tag{6}$$

$$\tau = \frac{1}{N} \Sigma_{i=1}^{N} \delta_i^K \tag{7}$$

$$Outliers = \{o | \delta_o^K > \tau\} \tag{8}$$

Refering to the two proposed strategies as strategy 1 and strategy 2, then the assignment procedure is as follows. FKNN-DPC first finds the outliers, then deletes the outliers from X and uses strategy 1 to attempt to assign the points in the set $X - Outliers$ to their proper clusters. The points assigned by strategy 1 comprise the cores of the clusters. Strategy 2 is used to assign the outliers and the unassigned remaining points in $X - Outliers$ by strategy 1. After applying the two assignment strategies, there can still be points which have not been assigned which are the true noise, and they will be assigned to the same cluster as the nearest neighbor which has already been assigned.

The basis of assignment strategy 1 is the breadth first search for the K nearest neighbors of a point starting from cluster centers on the connected weighted graph where the vertices are points of non-outliers, and the weights on the edges are the Euclidean distances between points. The K nearest neighbors of a point are assigned to their clusters under the constraints described in strategy 1. In order to get the optimal clusters, a fuzzy weighted KNN approach is developed in assignment strategy 2 and semi-supervised learning [34] is adopted, so that the cluster information obtained from strategy 1 can be used, as far as possible, to assign the outliers and the points unassigned by strategy 1 to their most probable clusters. The key aspect of strategy 2 is to learn the probability p_i^c that a point i belongs to cluster c, then assign the point i to its most similar cluster c with the highest value of p_i^c.

In order to define p_i^c, we first define the similarity w_{ij} between points i and j by (9), which means the smaller the distance between points i and j, the more similar they are. In (9) as d_{ij} can be close to zero, 1 is added to d_{ij} in the denominator of (9) to constrain w_{ij} to lie between 0 and 1.

$$w_{ij} = \frac{1}{1 + d_{ij}} \tag{9}$$

The calculation of p_i^c uses the cluster information of the K nearest neighbors of point i. The more points j belong to cluster c among the K nearest neighbors of i, and the more similar are points i and j, that is the larger the value of w_{ij}, then the greater is the probability that point i belongs to the cluster c, so the higher is the value of p_i^c. The value of p_i^c is defined by (10)

$$p_i^c = \sum_{j \in KNN_i, y_j = c} \gamma_{ij} * w_{ij} \tag{10}$$

where $\gamma_{ij} = w_{ij} / \Sigma_{l \in KNN_j} w_{lj}$ and y_j is the cluster label of point j (so γ_{ij} is w_{ij} normalized by the sum of the similarities of point j to its K nearest neighbors). The term $\gamma_{ij} * w_{ij}$ is the weighted contribution of point j to the p_i^c, and its value not only depends on the similarity between points i and j, but also on the distribution of the K nearest neighbors of point j.

Assignment strategy 2 will add a point to the cluster with the highest membership in the fuzzy weighted KNN. It is reported in [7] that the fuzzy weighted KNN rules are more robust to the choice of K than the traditional KNN rules in [6], and have a comparative low error.

Here are the primary steps of our FKNN-DPC and the details of the proposed assignment strategies 1 and 2.

The main steps of FKNN-DPC:

Input: dataset X, parameter K, cluster c_i for point i, $c_i = -1$ means point i hasn't been assigned.

Output: the clustering C.

Step 1 *data preprocessing: to make up the missing values in a dataset and normalize the data;*

Step 2 *compute the Euclidean distance between points and calculate the local density* ρ_i *and the distance* δ_i *for each point* $i \in X$ *using (5) and (2) or (3);*

Step 3 *find the cluster centers by analysis of the decision graph of* δ_i *as a function of*ρ_i*, and let the cluster centers be the set* CI;

Step 4 *find outliers using (8), and delete the outliers from dataset* X;

Step 5 *assign the remaining points using strategy 1;*

Step 6 *use strategy 2 to assign the outliers and the remaining points still not assigned by step 5;*

Step 7 *group the few points unassigned by the steps 5 and 6 to the cluster of its nearest neighbor which has been assigned.*

Strategy 1: Assigning the remaining non-outlier points

1. *select one unvisited cluster center* $c \in CI$*, and mark* c *as visited;*
2. *assign the points in* KNN_c *to the same cluster as* c*, and initialize a queue* Q *to be* KNN_c;
3. *remove the head* p *of* Q*, and choose each point* $q \in KNN_p$ *satisfying the following conditions:*
   (a) q *has not been assigned, and*
   (b) q *is not an outlier, and*
   (c) *the distance* $d_{pq} \le (\Sigma_{j \in KNN_q} d_{qj})/K$,
   then assign the point q *to the same cluster as* p *and add* q *to the end of the queue* Q
4. *if the queue* Q *is not empty, then go to 3;*
5. *if there is an unvisited point in CI , then got to 1, else halt strategy 1.*

Strategy 2: Assigning the outliers and non-outliers unassigned by strategy 1

We suppose the number of points still to be assigned is m.

1. *for each unassigned point* i *calculate its membership* p_i^c *for* $c = 1, \ldots, |CI|$ *using (10) storing as a vector with* $|CI|$ *elements. A recognition matrix* A *with* m *rows and* $|CI|$ *columns can then be constructed for all unassigned points;*
2. *form vectors VA and VP each of length* m *to store for each point* i $(i = 1, \ldots, m)$ *the highest membership* p_i^c *from the recognition matrix* A*, and the most likely cluster* c *respectively, i.e.,*

$$VA[i] = max\{p_i^c | c = 1, \ldots, |CI|\};$$
$$VP[i] = argmax_c\{p_i^c | c = 1, \ldots, |CI|\}$$

3. *choose the point* p *with the highest membership by searching vector VAif there is an element* > 0 *and assign the point* p *to its most likely cluster* c *denoted by* $VP[p]$*, then go to 4, i.e., the next step, otherwise stop strategy 2;*
4. *update the recognition matrix* A *and the vectors VA and VP, i.e., let* $VA[p] = 0$*, and* $VP[p] = -1$*. For each unassigned point* $q \in KNN_p$*, update the entry* $A[q][c]$ in the recognition matrix A, that is, the membership degree of *point* q *to cluster* c*,* p_q^c *by letting* $p_q^c = p_q^c + \gamma_{qp} * w_{qp}$*, and update*$VA[q] = max\{p_q^c | c = 1, \ldots, |CI|\}$ and $VP[q] = argmax_c\{p_q^c | c = 1, \ldots, |CI|\}$;
5. *if there are no points to be assigned, then terminate the strategy 2, otherwise go to 3.*

3.2. The analysis of FKNN-DPC

This subsection will analyze the complexity of our proposed FKNN-DPC, then present a theoretical analysis to explain its good clustering properties.

3.2.1. Complexity analysis of FKNN-DPC

The space complexity of DPC is $O(n^2)$, where n is the size of dataset, which is mainly due to storing the distance matrix. FKNN-DPC also needs space to store the distance from each point to its K-nearest neighbors, and to store the recognition matrix in strategy 2, but the space complexity of FKNN-DPC does not increase more than $O(n^2)$ because the extra space required by FKNN-DPC does not exceed $O(|CI|n)$, and the number of clusters $|CI|$ is often small (and anyway bounded by n), so the space complexity of FKNN-DPC is of the same order as DPC in [27].

The time complexity of DPC depends on the following three aspects: (a) the time for computing the distance between points; (b) the time to calculate the local density ρ_i for point i and (c) the time used to compute the distance δ_i for each point i. The time complexity of each is $O(n^2)$, so the total approximate time complexity of DPC is $O(n^2)$.

The time complexity of FKNN-DPC depends on the following four parts: (a) computing the distance between points $(O(n^2))$; (b) evaluating the local density for each point. To compute this we need to search the K-nearest neighbors of point i, whose time complexity is $O(n)$, so the time complexity for searching K-nearest neighbors for n points is $O(n^2)$. As a result the time complexity to calculate the local density for each point is of order $O(n^2)$; (c) evaluating the distance δ_i of point i, which has time complexity $O(n^2)$; d) assigning points to their most appropriate clusters with time complexity also

$O(n^2)$. The detailed analysis for the assignment process is as follows: There are two strategies for assigning points to their most likely clusters. Strategy 1 is similar to the breadth-first search of a graph, and its time complexity is $O(n_1(|CI|+1))$, where n_1 is the number of non-outlier points. From (6)–(8), the KNN distance δ_o^K of outlier o is far greater than that of non-outliers, and the threshold estimated in (7) is the average of the KNN distance of all points in a dataset, so n_1 is relatively large, and the number of core points in a cluster is also large for the core is a subset of non-outliers. Strategy 2 assigns the points unassigned by strategy 1, and its time complexity mainly depends on steps 3 and 4 whose functions are to assign a point to the most probable cluster and its time complexity is $O(n_2)$, $n_2 = n - n_1$, so the total time complexity of strategy 2 is $O(n_2^2)$ to assign all n_2 points to their most probable clusters.

The above analysis for the two strategies of FKNN-DPC demonstrate that the time complexity of the assignment procedures is $O(n^2)$. Therefore, the overall time complexity of FKNN-DPC is $O(n^2)$ which is the same as that for DPC.

3.2.2. Theoretical analysis of FKNN-DPC

The key feature of DPC is its heuristic to search and find the density peaks (cluster centers). FKNN-DPC keeps this feature, but FKNN-DPC uses a new approach to evaluate the local density ρ_i of point i based on the K-nearest neighbors of i. The new density metric is much more general than that of DPC because it can be used for any size dataset to accurately evaluate the local density without any reference to the cutoff distance d_c. This overcomes the shortcomings of DPC which had to use different approaches for computing the local density for points for different sizes of dataset and the local density of a point may be influenced by cutoff distance d_c.

Furthermore the new definition of local density ρ_i of point i guarantees that ρ_i depends only on the K nearest neighbors, whilst for the original definition in [27] the number of the points which are closer than cutoff distance d_c to point i is variable depending on d_c. Determining the parameter K is easier than determining the cutoff distance d_c to ensure that the average number of neighbors of a point is between 1% and 2% of the total number of points in the dataset. Therefore, the new general density metric is an improvement on the original for search and find of density peaks.

In addition, FKNN-DPC adopts strategy 1 followed by strategy 2 to assign points after the cluster centers have been found. The points assigned by strategy 1 comprise the core of a cluster, and the points in cores are close to their cluster centers, which reduces the misclassification rate of clustering. Strategy 2 assigns points left by strategy 1 to their most probable clusters denoted by their highest membership degree calculated in (10). As a consequence strategy 2, as far as possible, uses the distribution of points which have been assigned, which guarantees the highest accuracy in clustering, so FKNN-DPC is more robust than DPC in [27].

However, our FKNN-DPC is based on the K-nearest neighbors of a point. The value of K may affect the local density of a point, but it has little influence on the cluster centers because variations in the value of K does not change the fact that the cluster centers are the density peaks. Furthermore, to some extent, the fuzzy weighted KNN in strategy 2 reduces the influence of the value of K on the clustering [7]. In addition, strategy 2 only assigns the point with the highest membership degree to its most likely cluster, which guarantees the correctness of clustering.

From the above analysis we can see that the new definition of the local density of a point together with the proposed assignment strategies will lead to better clustering results than that of DPC, which will be demonstrated in Section 4.

4. Experimental results and their analysis

Experiments were conducted on two collections of synthetic and real-world datasets to test the power of FKNN-DPC. The datasets are those ones commonly used to test the performance of a clustering algorithm. The number of features and data points varied from small to large with various numbers of clusters and the distribution of points acted as a challenge to detect the clusters. The performance of FKNN-DPC was compared with well-known clustering algorithms including AP in [12], DBSCAN in [8], and K-means in [24] besides DPC in [27]. The code of AP and DBSCAN were provided by their authors. The DPC code is optimized from the original DPC code provided by Alex Rodríguez and Alessandro Laio in [27] by using matrix operations instead of iterative loops to reduce its run time.

We evaluated the performance of the above clustering algorithms in clustering accuracy (Acc), adjusted mutual information (AMI) and adjusted rand index (ARI). These three metrics are very popular for testing the performance of clustering algorithms [25]. The larger the benchmark values the better is the clustering. The upper bounds of the three benchmarks are all 1.

We display the clustering of each algorithm for the synthetic datasets embedded in two dimensional space as colored plots. Because the primary idea of DPC and FKNN-DPC is to search and find cluster centers, we also compare the number of cluster centers found by these two algorithms with the actual number of clusters.

The five clustering algorithms being compared require various parameters to be set. FKNN-DPC needs the value of K, the number of neighbors of a point, to be pre-specified. In DPC the cutoff distance d_c is required to compute the local density. As a rule of thumb, one can choose d_c so that the average number of neighbors is between 1% and 2% of the total number of points in the data set [27], clustering is robust to cutoff distance d_c when the size of the dataset is large enough. In our experiments we adopt (1) to estimate the local density of a point in DPC when the size of a dataset is greater than 2000, otherwise the exponential kernel in (4) is used to calculate the local density of a point, and *Par* represents the percentage in setting the cutoff distance. *Par* may vary for different datasets, but DPC is robust to the change in *Par* [27]. AP takes as input a real number $s(k, k)$ for each data point k so that data points with larger values of $s(k, k)$ are more likely to be chosen as

Table 1
Synthetic datasets.

Dataset	No records	No attributes	No clusters	Source
Path-based1	300	2	3	[3]
Path-based2	312	2	3	[3]
Flame	240	2	3	[13]
Aggregation	788	2	7	[14]
SD(noise)	2000	2	5	[27]
S2(noise)	5000	2	15	[10]
DIM512	1024	512	16	[11]
DIM1024	1024	1024	16	[11]

Table 2
Real-world datasets.

Dataset	No records	No atributes	No clusters	Source
Iris	150	4	3	[1]
Wine	178	13	3	[1]
Parkinsons	195	23	2	[1]
Seeds	210	7	3	[1]
Ionosphere	351	34	2	[1]
Dermatology	366	34	6	[1]
Libras movement	360	91	15	[1]
WDBC	569	30	2	[1]
Pima-indians-diabetes	768	8	2	[1]
Segmentation	2310	19	7	[1]
Waveform	5000	21	3	[1]
Waveform(noise)	5000	40	3	[1]
Olivetti face	400	92*112	40	[28]

exemplars. The number of identified exemplars (number of clusters) is influenced by the values of the input preferences. If the preferences are set to the median of the input similarities, it will result in a moderate number of clusters, else if it is set to the minimum, then a small number of clusters will result [12,31]. DBSCAN has two parameters to specify, the maximum radius ϵ and the minimum points *MinPts* included in the neighborhood [8]. For the *K*-means clustering algorithm the value of *K* and the initial cluster centers must be given. We used a grid search strategy to find the optimal parameters for DBSCAN in our experiments, and the best parameters of other compared algorithms are found by repeating experiments a number of times.

4.1. Data preprocessing

The sources of the real-world datasets used in our experiments were the UCI machine learning repository [1] and the Olivetti face dataset [28]. The synthetic datasets used come from research published in [3,10,11,13,14,27] and are commonly used benchmark datasets to test the performance of clustering algorithms. The size, number of features and the number of clusters varies in each of the datasets. The datasets have been chosen to test the power of FKNN-DPC to recognize the clusters for datasets of arbitrary shape without any influence from noise, the size or the dimensionality. Table 1 describes the synthetic datasets used in the experiments and the synthetic dataset from [27] is named SD. Table 2 displays the real-world datasets from [1,28]. It should be noted that 2 real-world datasets named Waveform are included. The second one displayed in Table 2 is tagged with 'noise' to indicate that it has 19 noise features, so the total number of attributes in the dataset is 40.

In data preprocessing, we replace the missing attribute values by the mean of the attribute set, and normalize the data using a min-max normalization given by (11),

$$x_{ij} \leftarrow \frac{x_{ij} - min(x_j)}{max(x_j) - min(x_j)} \tag{11}$$

where x_{ij} represents the value of attribute j of point i and $min(x_j)$ and $max(x_j)$ are the minimum and maximum values of attribute j.

Min-max normalization preserves the relationships between the original data values [15], and reduces the influence on experimental results from different metrics for different attributes and also reduces the run time of algorithms.

4.2. Analysis of the experimental results on synthetic datasets

FKNN-DPC, DPC [27], AP [12] and DBSCAN [8] can all search and find clusters automatically after their parameters are given. This section will first show the performance of the four clustering algorithms on 6 synthetic datasets which are

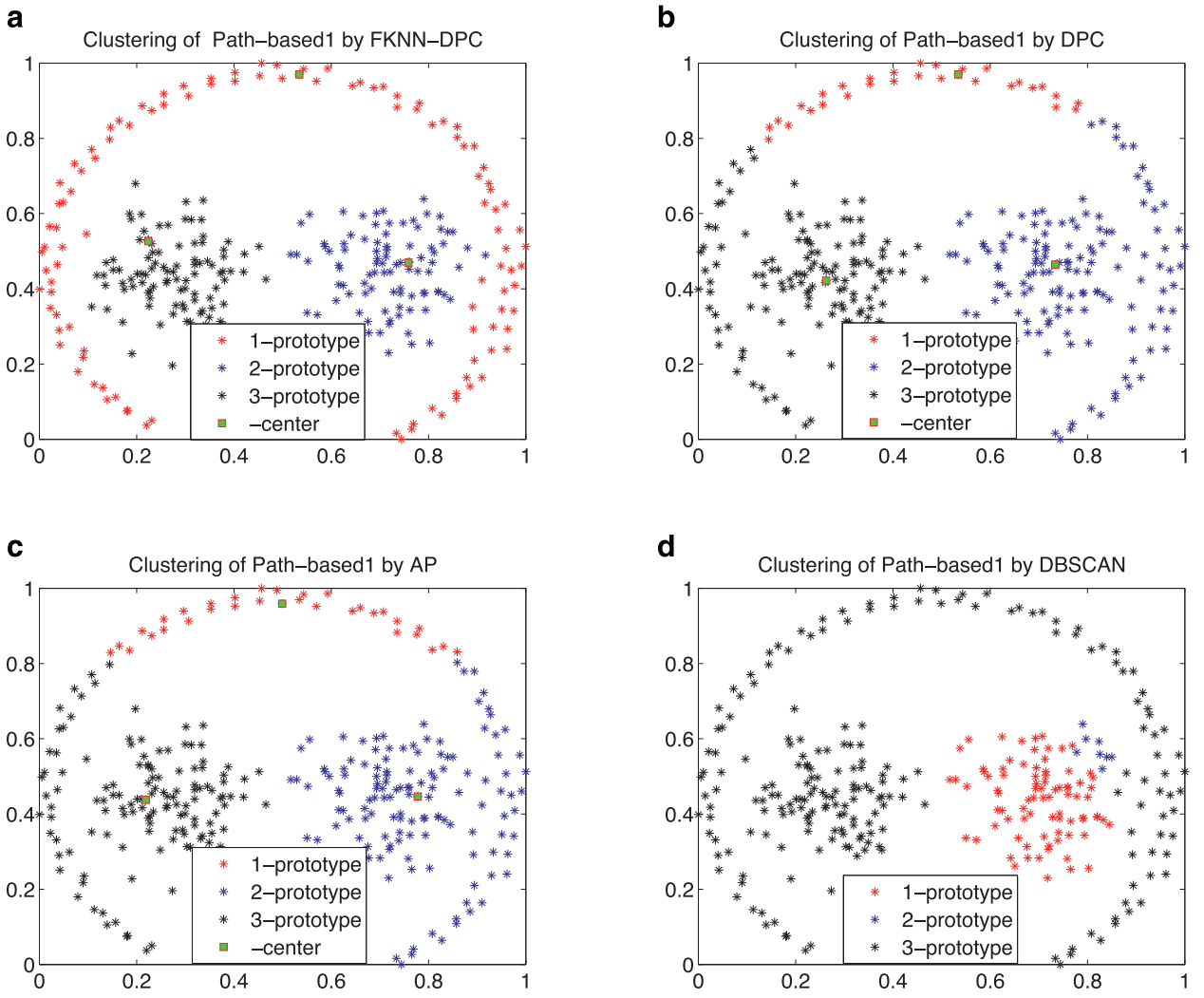

Fig. 2. The clustering of Path-based1 by 4 clustering algorithms. (a) FKNN-DPC, (b) DPC, (c) AP, and (d) DBSCAN.

embedded in two dimensional space. Figs. 2–7 show the clusterings of these algorithms on the 6 synthetic datasets, where the cluster centers have been marked except for DBSCAN. In DBSCAN's case cluster centers are randomly chosen among those points satisfying the condition that there are at least *MinPts* points in the neighborhood with radius ϵ.

Table 3 displays clustering results in terms of Acc, AMI and ARI of the five clustering algorithms on all eight synthetic datasets shown in Table 1. For *K*-means the average values over 20 runs are given as results may vary with its initial centers. We did not provide the statistical results, for the deviations of the algorithms are all zero except for *K*-means, which is caused by the nature of the algorithms or by the code provided by authros. The number of clusters found by FKNN-DPC, DPC and AP in the datasets is compared in Table 3 with the number of clusters in which the cluster centers sit indicated by F/P. F refers to the number of cluster centers found by algorithms and P the number of clusters in which the cluster centers lie. The parameters pre-specified for each algorithm are also shown in Table 3. It must be noted that there are no corresponding results in Table 3 of synthetic dataset SD from [27] for there is no true distribution information about this dataset for comparison. The clustering results of the five clustering algorithms are very similar on the two high dimensional synthetic datasets DIM512 and DIM1024, so only the results for DIM512 are displayed in Table 3. The symbol "-" in Table 3 means there are no corresponding values. The bold font in Table 3 indicates the best of the results.

Fig. 2 shows that only FKNN-DPC can find both the cluster centers and correct clusters for the Path-based1 dataset. DPC, AP and DBSCAN cannot recognize the clusters while DPC and AP can only identify the cluster centers. Although the cluster centers found by FKNN-DPC and DPC and AP are similar, the clustering results are very different because they adopt different strategies to assign points to their clusters. From the Fig. 2(b) and (c), the clustering results of AP and DPC are similar, however the reasons causing the error in these figures are completely different. The assignment strategy in DPC can lead to the fact that once a point with a comparable higher density is assigned to an incorrect cluster, then there will be

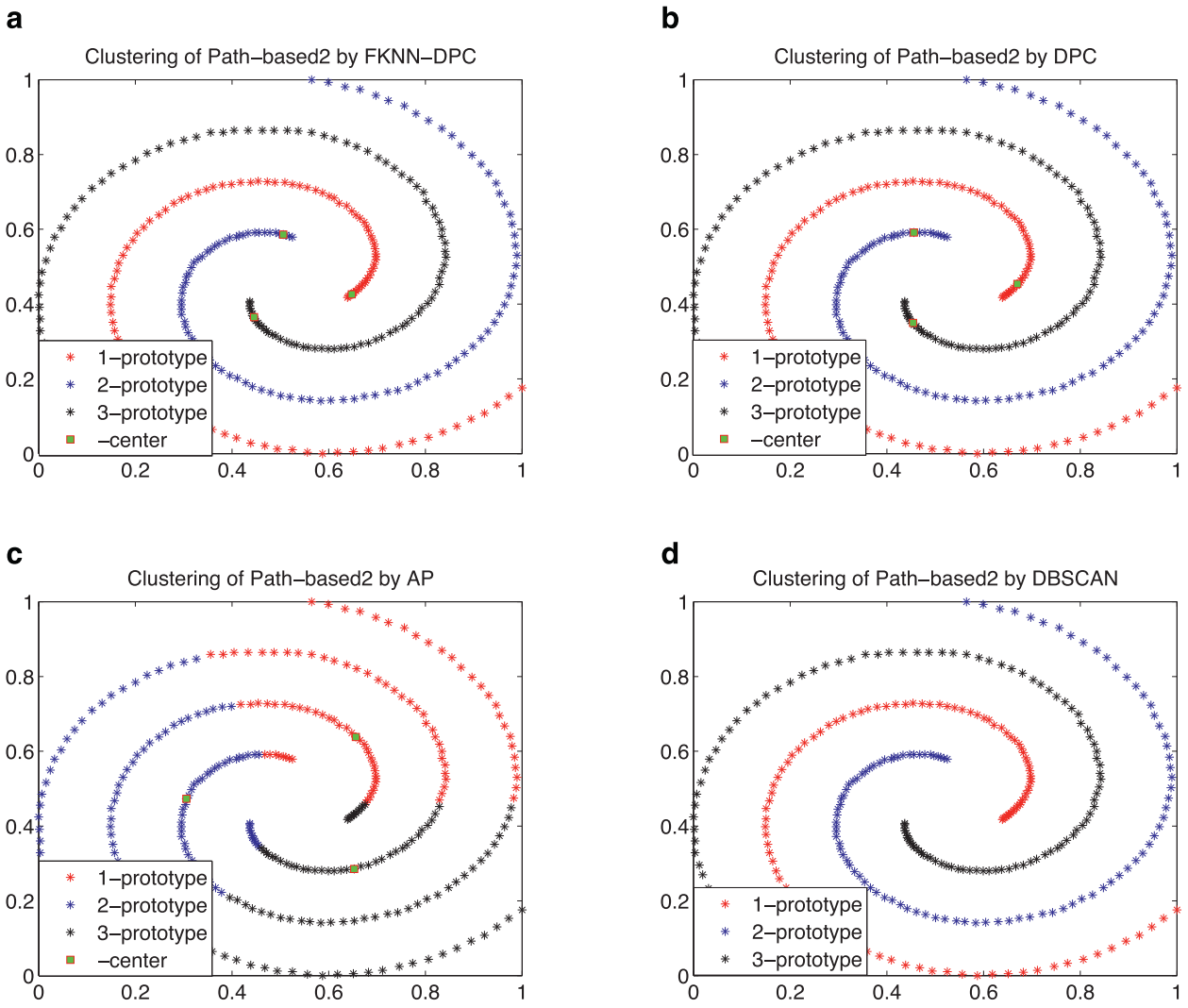

Fig. 3. The clustering of Path-based2 by 4 clustering algorithms. (a) FKNN-DPC, (b) DPC, (c) AP, and (d) DBSCAN.

several more points assigned erroneously. That is why in Fig. 2(b) there are some points which should be in cluster 1 which have been placed in clusters 2 or 3. AP will add point i to cluster C only if the attraction from C (exemplar) to the point i is large. The attraction is computed by two kinds of messages exchanged between points, *responsibility* and *availability*, which respectively reflect how well the points are exemplars for all other points, and how appropriate the exemplars are for every other point. That is why the clustering in Fig. 2(c) has arisen. DBSCAN is based on the connection between points by similarity to assign points to their clusters. It is the classical density based clustering algorithm. The similarities between points and the parameters pre-specified by hand may heavily affect its clustering. For this reason, in Fig. 2(d), DBSCAN merges clusters 1 and 3 into one cluster, and partitions cluster 2 into two clusters .

The clusterings shown in Fig. 3 demonstrate that the three algorithms including FKNN-DPC, DPC in [27] and DBSCAN in [8] can detect the clusters of dataset Path-based2, but AP from [12] cannot. The reason for this is that the exemplars in the concave dataset may attract points belonging to other clusters. Furthermore, from the clusterings shown in Fig. 4, we can see that FKNN-DPC, DPC and AP can all find the correct cluster centers of Path-based2. It is obvious that the cluster centers found by FKNN-DPC, DPC and AP are distant from the initial points of the clusters in turn.

The clusterings displayed in Fig. 4 demonstrate that FKNN-DPC, DPC, and DBSCAN can detect the clusters of Flame but AP can only find the centers. The cluster centers found by FKNN-DPC, DPC and AP are not completely same.

The clusterings in Fig. 5 reveal that FKNN-DPC, DPC and DBSCAN have recognized the clusters in the Aggregation dataset but AP has not. FKNN-DPC and DPC have also detected the cluster centers. Although the cluster centers are not marked in Fig. 5(d) for DBSCAN, the clustering and the number of clusters found by it are correct. AP cannot detect the correct cluster

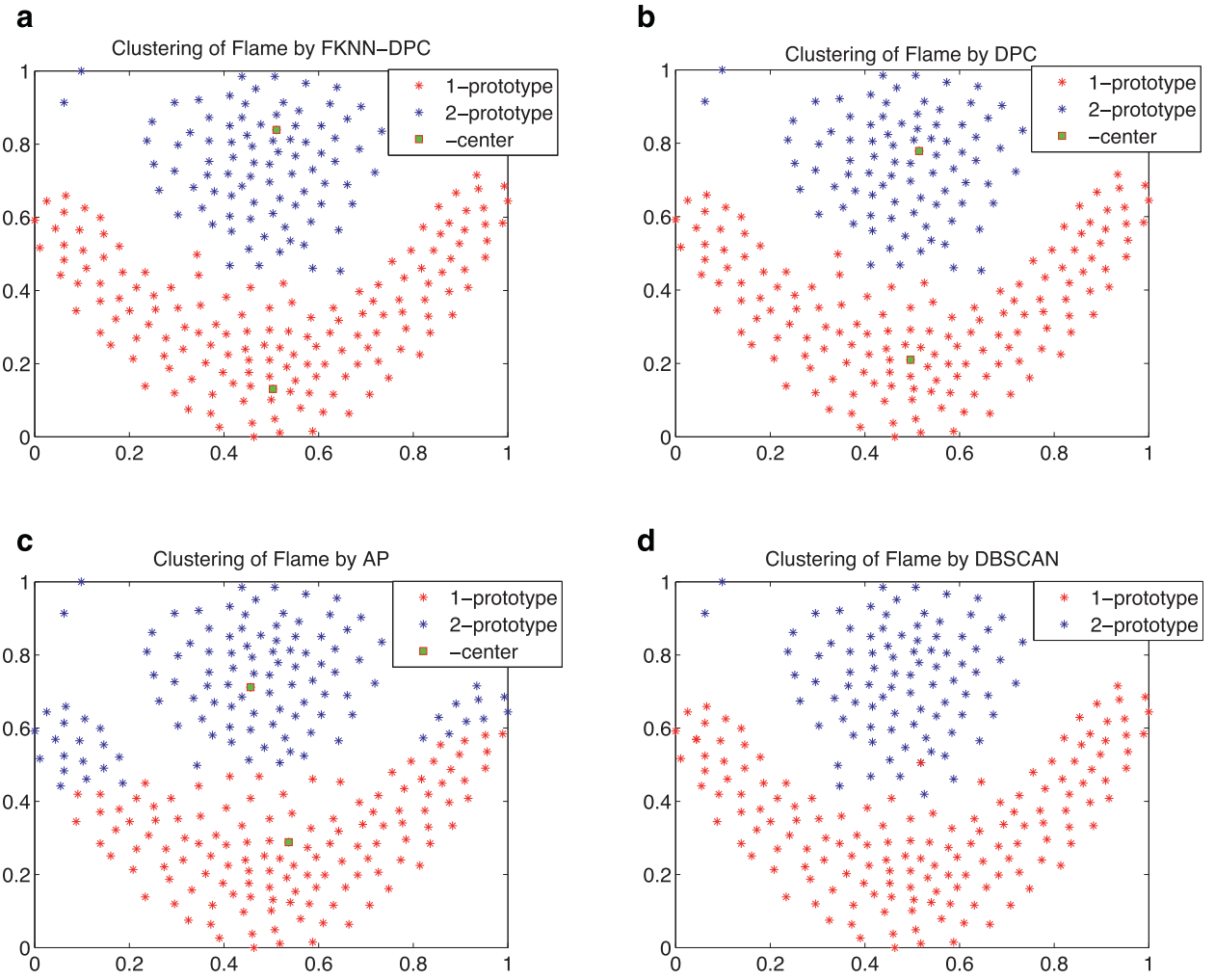

Fig. 4. The clustering of Flame by 4 clustering algorithms. (a) FKNN-DPC, (b) DPC, (c) AP, and (d) DBSCAN.

Table 3
The comparison of Acc, AMI and ARI benchmarks for 5 clustering algorithms on synthetic datasets.

Algorithm	Path-based1					Path-based2				
	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
FKNN-DPC	**0.987**	**0.941**	**0.960**	3/3	5	**1.000**	**1.000**	**1.000**	3/3	6
DPC	0.733	0.500	0.453	3/3	2	**1.000**	**1.000**	**1.000**	3/3	2
AP	0.740	0.506	0.458	3/3	10	0.343	−0.006	−0.006	3/3	19
DBSCAN	0.647	0.543	0.500	–	0.05/6	**1.000**	**1.000**	**1.000**	–	0.1/5
K-means	0.743	0.510	0.461	–	3	0.343	0.006	0.006	–	3
			Flame					Aggregation		
Algorithm	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
FKNN-DPC	**1.000**	**1.000**	**1.000**	2/2	5	**0.999**	**0.995**	**0.997**	7/7	8
DPC	**1.000**	**1.000**	**1.000**	2/2	5	0.997	0.992	0.996	7/7	4
AP	0.863	0.452	0.524	2/2	35	0.770	0.805	0.713	7/5	7.7
DBSCAN	0.983	0.872	0.934	–	0.09/8	0.996	0.987	0.992	–	0.05/8
K-means	0.848	0.421	0.483	–	2	0.747	0.789	0.695	–	7
			S2(noise)					DIM512		
Algorithm	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	
FKNN-DPC	0.963	0.936	0.924	15/15	13	**1.000**	**1.000**	**1.000**	16/16	8
DPC	**0.970**	**0.945**	**0.937**	15/15	2	**1.000**	**1.000**	**1.000**	16/16	2
AP	0.908	0.913	0.878	14/14	20	**1.000**	**1.000**	**1.000**	16/16	20
DBSCAN	0.724	0.800	0.654	–	0.23/7	**1.000**	**1.000**	**1.000**	–	0.3/7
K-means	0.872	0.897	0.840	–	15	0.765	0.867	0.751	–	1

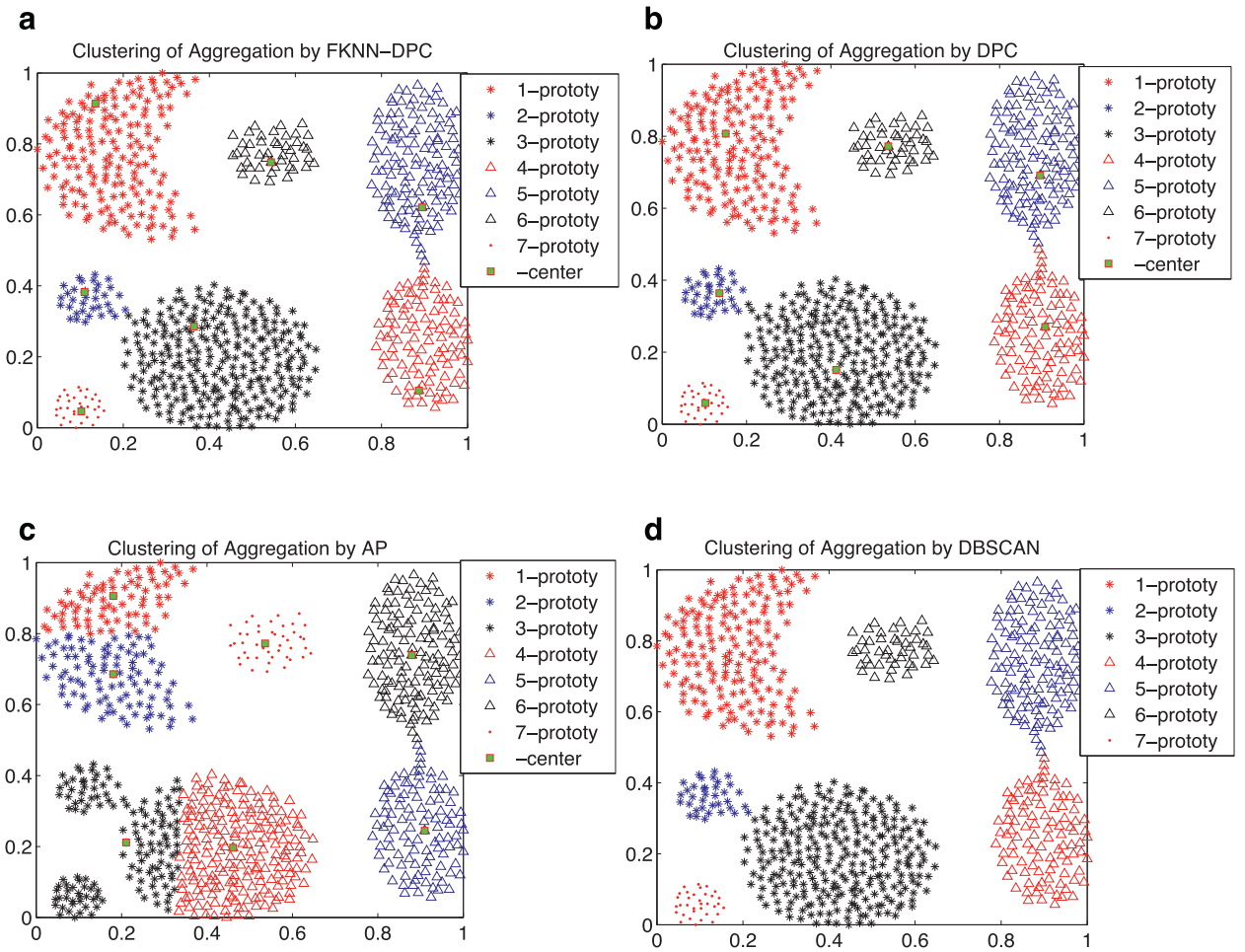

Fig. 5. The clustering of aggregation by 4 clustering algorithms. (a) FKNN-DPC, (b) DPC, (c) AP, and (d) DBSCAN.

centers, even worse it selects two cluster centers from one group and one cluster center from several groups of data points. However, the correct cluster centers found by AP lie in the center of their clusters.

The clusterings shown in Fig. 6 for dataset SD tell us that both of FKNN-DPC and DPC can detect the cluster centers and assign the points to their correct clusters. AP finds two cluster centers from the cluster 3 with relatively high density, and splits the cluster into two clusters. DBSCAN merged clusters 1 and 2 as these two clusters heavily overlapped each other, and grouped the outliers into one cluster showing it is sensitive to the intersection of clusters and to outliers or noise.

S2 is a dataset with noise and heavily overlapping clusters so is a very challenging dataset on which to test a clustering algorithm. From the clustering displayed in Fig. 7(a)–(d), it is obvious that both FKNN-DPC and DPC can find the cluster centers. FKNN-DPC can recognize all the clusters in S2 except that several outliers around cluster 14 have been assigned to cluster 8 or cluster 6. DPC can also recognize the clusters in S2 and produces a good clustering of S2. Fig. 7(c) shows that the number of cluster centers found by AP is 14 instead of 15, with 13 cluster centers lying in the center of the related clusters but the 14th cluster center lies in the intersection of cluster 2 and cluster 15. The worst case is produced by DBSCAN. The number of clusters detected is 9, with clusters 2, 5 and 15 merged into one cluster and clusters 10, 4 and 7 also grouped together. Furthermore, some outliers are recognized as two new clusters.

From the benchmark data shown in Table 3 it is clear that overall FKNN-DPC achieved the best results among the five clustering algorithms. FKNN-DPC outperformed the other clustering algorithms on nearly all these synthetic datasets, except for DPC on dataset S2, but FKNN-DPC can find the correct cluster centers of S2. It also can be seen from Table 3 that DPC compares well to FKNN-DPC except that on the Path-based1 it performs even worse than AP and *K*-means.

It is obvious that FKNN-DPC, DPC, AP and DBSCAN can all obtain the optimal results on the high dimensional datasets of DIM512 and DIM1024, but *K*-means cannot. From the results of the clustering algorithms on the dataset Pathe-based2, in terms of Acc, AMI and ARI, it is very clear that FKNN-DPC, DPC and DBSCAN can all achieve the optimal results but AP and *K*-means cannot. From the benchmark data in terms of Acc and AMI and ARI shown in Table 3, it can be seen that the very simple and popular clustering algorithm *K*-means can achieve similar results to AP on the datasets embedded in a relatively

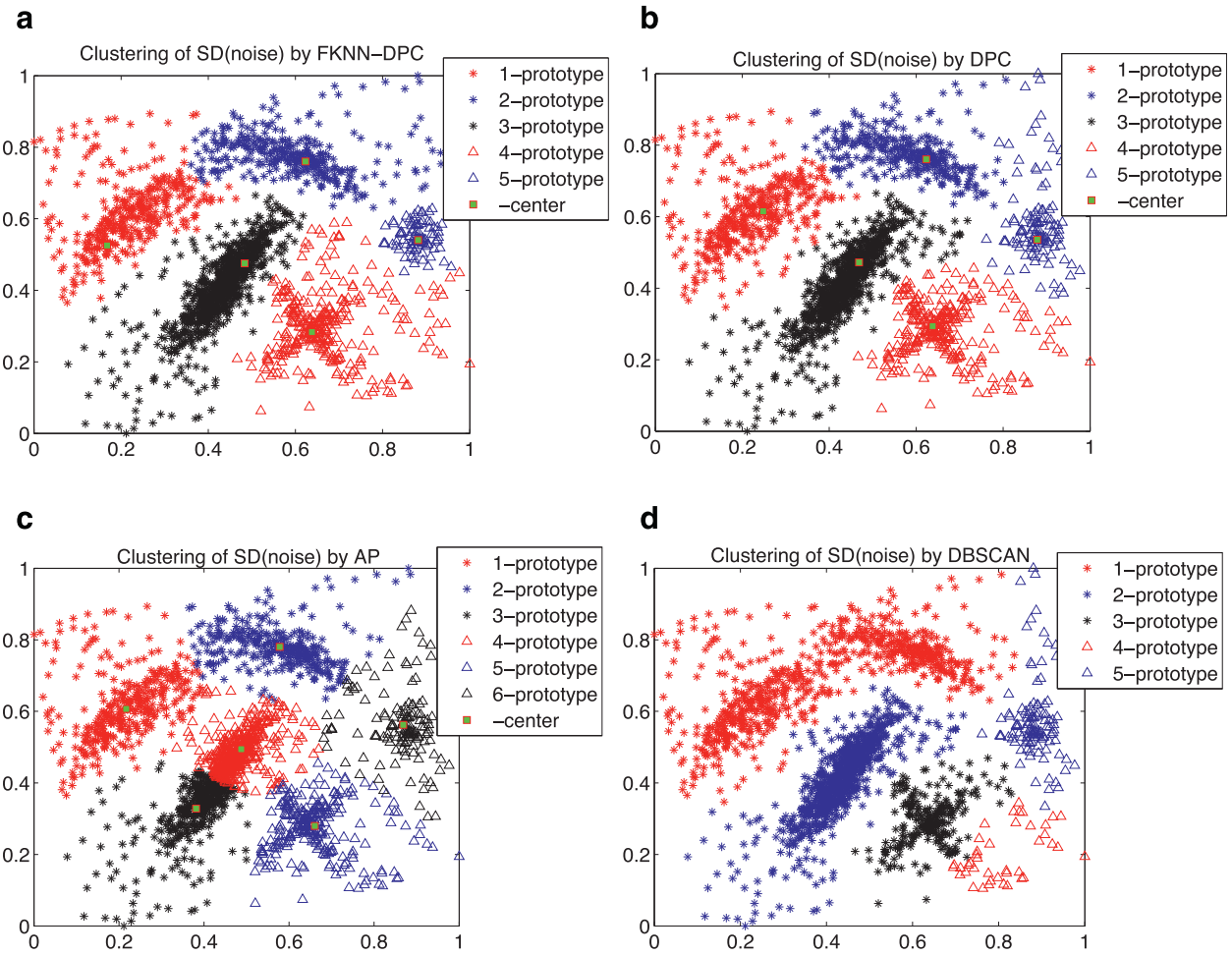

Fig. 6. The clustering of SD by 4 clustering algorithms. (a) FKNN-DPC, (b) DPC, (c) AP, and (d) DBSCAN.

low dimensional space, but its performance on the high dimensional datasets, such as DIM512 and DIM1024, is worse than that of AP. DBSCAN can obtain a comparable good clustering result, but its performance is bad when it is used to cluster datasets with heavy intersections between clusters or with noise, such as datasets S2 and Path-based1.

We see that FKNN-DPC and DPC can detect the correct cluster centers in each synthetic dataset but AP cannot perform as well on several synthetic datasets. For example, on Aggregation AP only found 5 cluster centers instead of 7 and on S2 only 14 cluster centers instead 15. The clusterings of AP on the two synthetic datasets are respectively shown in Figs. 5(c) and 7(c), indicating that AP may erroneously partition one cluster into two clusters or merge two clusters into a single one.

All the clustering algorithms compared in this paper require the setting of the parameters by hand. It is reported in [27] that for large datasets DPC is robust to changes in the cutoff distance d_c, say, to the parameters shown in Table 3 as Par. FKNN-DPC needs to pre-specify the value of K. It has been analyzed in Section 3.2 that FKNN-DPC is robust to the parameter K. However, the performance of AP is influenced by the value of input preferences, and the optimal value of the preferences varied greatly for different datasets. There are two parameters which must be given by hand for DBSCAN, and the clustering of DBSCAN is sensitive to these. It would be a challenging task to select the optimal parameters for DBSCAN on each dataset.

4.3. Analysis of experimental results on real-world datasets

This subsection will test the performance of our FKNN-DPC in 12 real-world datasets from UCI machine learning repository [1] and in the Olivetti face dataset [28]. We chose these datasets as the test cases because they are commonly used for testing the performance of a clustering algorithm. The size of the datasets (number of points contained in the datasets) and the number of clusters of these datasets and the dimension in which that the datasets are embedded are varied, allowing us to test the performance of our algorithm under many different conditions.

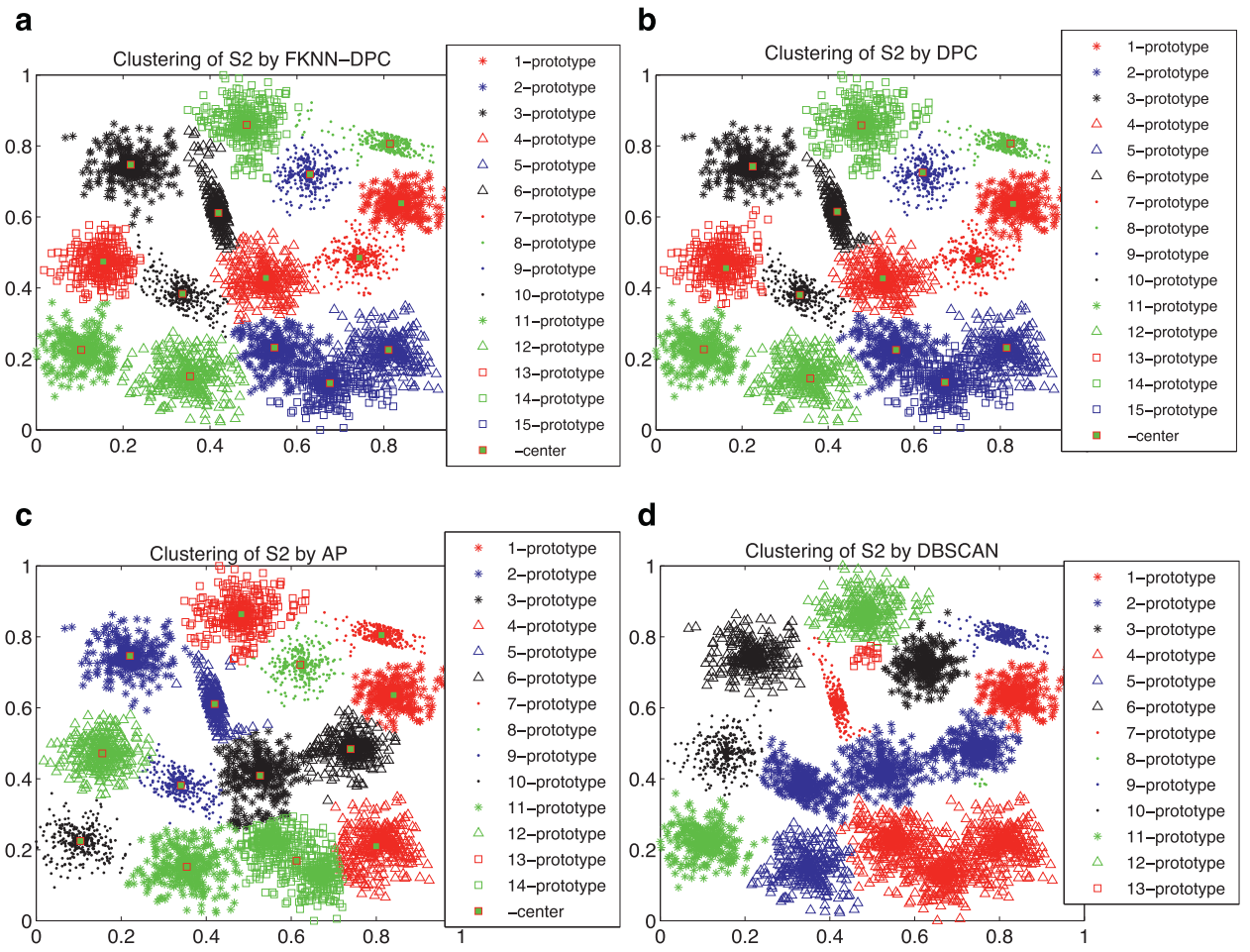

Fig. 7. The clustering of S2 by 4 clustering algorithms. (a) FKNN-DPC, (b) DPC, (c) AP, and (d) DBSCAN.

4.3.1. The analysis of the experimental results on real-world datasets from UCI

Again the performance of FKNN-DPC, DPC, AP, DBSCAN and *K*-means are benchmarked in terms of Acc, AMI and ARI. The number of cluster centers found by FKNN-DPC, DPC and AP are compared to the number of true clusters containing the cluster centers. Table 4 displays the clustering results and parameters of FKNN-DPC and other clustering algorithms on the 12 real-world datasets. The symbol "-" in Table 4 means there is no value for that entry and the best results are shown in boldface. Par in Table 4 is the parameters specified, and F/P respectively indicates the number of cluster centers found and the number of clusters containing those cluster centers.

Iris is composed of 3 clusters, where clusters 2 and 3 are heavily overlapped. From the results shown in Table 4 we can see that FKNN-DPC, DPC and AP can all find the cluster centers, but the benchmark data of these three algorithms are very different. The clustering accuracy (Acc) of FKNN-DPC reaches 0.973 with only four points in cluster 3 classified as being in cluster 2. Until now, the best clustering results for Iris were obtained by the simulated annealing algorithm using a Bayesian Ying–Yang (BYY) model [23], where the clustering accuracy (Acc) is 0.98 with only three points incorrectly classified. This algorithm, however, has a heavy computational load in searching for the optimal parameter of a model. In addition, the benchmark results for FKNN-DPC are much better than that for DPC, AP, DBSCAN and *K*-means in terms of AMI and ARI.

From the results in Table 4, we can see that FKNN-DPC, DPC and AP can all find the cluster centers of Wine, but the clustering results of FKNN-DPC evaluated by Acc, AMI and ARI are much better than that of DPC and AP. Furthermore from the results of each algorithm on Wine we can see that FKNN-DPC is best among the five clustering algorithms. The results of DPC are a little better than that of AP and DBSCAN. The average clustering results of the very simple and popular clustering algorithm *K*-means on Wine ranked in second place only after FKNN-DPC.

It is obvious from Table 4 that both FKNN-DPC and AP can find the cluster centers of the breast cancer database WDBC, but DPC cannot. Considering this in more detail, we see only one peak (the 236*th* point) can be found easily, for it is obviously far away from the other points in the decision graph of DPC when the cutoff distance $d_c = 9/100$, while the other peak (the 367th point) is very close to the remaining points. These two peaks (cluster centers) are all in the cluster of

Table 4
Comparison of Acc, AMI and ARI benchmarks for 5 clustering algorithms on UCI datasets.

Algorithm	Iris					Wine				
	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
FKNN-DPC	**0.973**	**0.912**	**0.922**	3/3	7	**0.949**	**0.831**	**0.852**	3/3	7
DPC	0.887	0.767	0.720	3/3	2	0.882	0.706	0.672	3/3	2
AP	0.907	0.756	0.757	3/3	6	0.854	0.686	0.616	3/3	6
DBSCAN	0.893	0.775	0.732	–	0.14/9	0.876	0.678	0.660	–	0.42/10
K-means	0.825	0.692	0.660	–	3	0.932	0.815	0.830	–	3
			Ionosphere					Seeds		
Algorithm	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
FKNN-DPC	**0.752**	**0.284**	**0.355**	3/2	8	**0.924**	**0.759**	**0.790**	3/3	8
DPC	0.681	0.238	0.276	3/2	0.65	0.900	0.717	0.734	3/3	2
AP	0.709	0.127	0.173	2/1	15	0.895	0.685	0.715	3/3	10
DBSCAN	0.607	0.086	0.036	–	0.2/7	0.881	0.644	0.686	–	0.17/8
K-means	0.712	0.129	0.178	-	2	0.890	0.671	0.705	–	3
			Segmentation					Libras movement		
Algorithm	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
FKNN-DPC	**0.716**	**0.655**	**0.555**	6/5	7	0.436	0.508	**0.308**	13/9	9
DPC	0.684	0.651	0.550	6/5	3	0.361	0.390	0.214	9/7	0.5
AP	0.670	0.605	0.502	7/5	25	**0.450**	0.497	0.277	16/10	2.5
DBSCAN	0.441	0.435	0.227	–	0.25/10	0.350	0.408	0.154	–	0.96/5
K-means	0.602	0.578	0.483	–	6	0.443	**0.519**	0.304	–	15
			Dermatology					Parkinsons		
Algorithm	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
FKNN-DPC	0.768	**0.847**	**0.718**	6/5	7	**0.851**	**0.273**	**0.391**	2/1	5
DPC	0.697	0.588	0.490	4/4	2	0.610	0.201	0.027	2/1	5
AP	**0.814**	0.771	0.717	7/6	5	0.672	0.147	0.127	3/1	15
DBSCAN	0.787	0.709	0.727	–	0.7/3	0.672	0.205	0.225	–	0.4/6
K-means	0.691	0.786	0.654	–	6	0.615	0.205	0.035	–	2
			WDBC					Pima-indians-diabetes		
Algorithm	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
FKNN-DPC	**0.944**	**0.679**	**0.786**	2/2	7	0.648	0.001	0.013	2/2	6
DPC	0.613	0.009	−0.011	2/1	9	0.650	0.034	0.078	2/1	4
AP	0.924	0.602	0.718	2/2	40	0.624	0.045	0.089	3/1	35
DBSCAN	0.870	0.419	0.542	–	0.27/7	0.540	0.017	0.035	–	0.15/6
K-means	0.928	0.611	0.730	–	2	**0.668**	**0.050**	**0.102**	–	2
			Waveform					Waveform (noise)		
Algorithm	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
FKNN-DPC	**0.703**	0.324	**0.350**	3/3	5	**0.648**	0.247	**0.253**	3/3	5
DPC	0.586	0.318	0.268	3/3	0.5	0.535	0.184	0.164	3/3	0.3
AP	–	–	–	–	–	–	–	–	–	–
DBSCAN	–	–	–	–	–	–	–	–	–	–
K-means	0.501	**0.364**	0.254	–	3	0.512	**0.364**	0.252	–	3

benign cancers. The corresponding clustering results of DPC by Acc, AMI and ARI are the worst among the five clustering algorithms, while our FKNN-DPC performs best. The benchmark results for Acc and AMI and ARI of *K*-means are ranked in the second place followed by that of AP and DBSCAN. We tried to tune the cutoff distance d_c for DPC in order to find the correct cluster centers of WDBC, but when the cutoff distance $d_c = 0.1/100 \sim 20/100$, the number of the cluster centers found obviously is always only one in the cluster of benign cancers. Not until we set $d_c = 0.00001$, that is the parameter (Par) in Table 4 for DPC is 0.001, is number of cluster centers found 3 from 2 real clusters. The 3 peaks are respectively the 111th, 188th and 234th point, where the 111th and the 188th points are in the cluster of malignant, whilst the 234th point is in the cluster of benign cancer. The corresponding clustering results in terms of Acc, AMI and ARI are respectively 0.629174, 0.001758 and 0.004816. This result is still the worst among the compared clustering algorithms. The obvious peaks found by DPC when tuning the cutoff distance are respectively the 234th point when $d_c = 0.01/100 \sim 3/100$, and the 236th point when $d_c = 4/100 \sim 20/100$. These two points are all in the cluster of benign cancers. The above detailed analysis further demonstrates that DPC works poorly in WDBC database.

The experimental results in Table 4 reveal that FKNN-DPC, DPC and AP can all detect the cluster centers of Seeds dataset. The benchmark results in terms of Acc and AMI and ARI in Table 4 demonstrate that FKNN-DPC performs better than DPC and AP, and these three clustering algorithms are better than DBSCAN and *K*-means. So the performance of FKNN-DPC is the optimal among the five clustering algorithms, while DBSCAN has the worst performance on Seeds dataset.

The results on Segmentation dataset shown in Table 4 show that FKNN-DPC and DPC can only find six cluster centers which are dispersed over five real clusters. Although AP can find seven cluster centers of the dataset, the cluster centers are only spread over five clusters. The performance of FKNN-DPC is the best one in terms of Acc, AMI and ARI. It is obvious from the experimental results displayed in Table 4 that FKNN-DPC and DPC can achieve better performance than AP, DBSACN and *K*-means on the dataset. Furthermore, it can also be seen that DBSCAN has the worst performance on this dataset.

The Libras movement dataset brings some challenges to FKNN-DPC and DPC as the number of instances in each class is small. FKNN-DPC found 13 cluster centers scattered over nine real clusters, and AP found 16 exemplars from 10 classes, and DPC found nine cluster centers from seven clusters. The performance of the five clustering algorithms shown in Table 4 demonstrate that FKNN-DPC is the best one in terms of ARI. The best Acc and AMI results are obtained by AP and *K*-means respectively. DPC has trouble with this dataset.

The experimental results of Ionosphere dataset shown in Table 4 demonstrate that FKNN-DPC and DPC both found three cluster centers, although there are only two classes in it. AP found two exemplars for this dataset, but the two exemplars are in the same class. The benchmark results shown in Table 4 reveal that FKNN-DPC achieves the best performance among the five clustering algorithms in terms of Acc, AMI and ARI. The performance of *K*-means is ranked in the second place according to the clustering accuracy, followed by AP, DPC and DBSCAN. But in terms of AMI and ARI, the next best-performing algorithm is DPC, followed by *K*-means and AP, while DBSCAN performs worst. We analyzed the details of the cluster centers found by DPC and AP on the Ionosphere dataset, and found that the exemplars found by AP are {345, 285} both belonging to one cluster. We selected 10 nearest neighbors of the exemplars, and found that all 10 neighbors of instance 345 belong to the cluster one, and 8 out of 10 nearest neighbors of exemplar 285 lie in the cluster two. This fact may illustrate why the clustering accuracy (Acc) of AP is higher than that of DPC.

The experimental results of Dermatology dataset shown in Table 4 disclose that our FKNN-DPC can detect 6 density peaks from 5 real groups of the dataset, and has got the clustering results with the best AMI and ARI, but its clustering accuracy is only the third one among five clustering algorithms. AP has obtained the highest clustering accuracy followed by DBSCAN. The DPC performs worst on the Dermatology dataset among the 5 clustering algorithms in terms of AMI and ARI, and its clustering accuracy on Dermatology dataset is only a little better than *K*-means. Furthermore, DPC cannot detect the number of clusters of the dataset.

The experimental results of the clustering algorithms on Parkinsons shown in Table 4 tell us that our FKNN-DPC defeats all the other four clustering algorithms in terms of Acc, AMI and ARI, though it detects 2 density peaks from one cluster. DPC performs worst among the five clustering algorithms in terms of Acc, AMI and ARI on Parkinsons dataset, and the two density peaks found by DPC are from the same cluster. DBSCAN has a little better performance than AP on Parkinsons dataset, and its performance on the dataset is not bad following our FKNN-DPC. *K*-means performs a little better than DPC.

However, the clustering results of algorithms on Pima-indians-diabetes shown in Table 4 show that *K*-means has got the best performance among the five clustering algorithms, followed by DPC, our FKNN-DPC, AP and DBSCAN. Although the performance of our FKNN-DPC is not better than that of DPC in terms of Acc, AMI and ARI on this dataset, our FKNN-DPC is the only one among the five clustering algorithms that can detect the right density peaks from real clusters of Pima-indians-diabetes dataset.

The Waveform and Waveform (noise) are different versions of the waveform database with 3 classes of waves including noise, and the latter one with 19 more noise attributes with mean 0 and variance 1. There are no benchmark results in Table 4 of DBSCAN and AP on these waveform databases in terms of Acc, AMI and ARI for the heavy computational costs make it infeasible to search for the optimal parameters for the algorithms on these datasets. From the results in Table 4 it can be concluded that both FKNN-DPC and DPC can find the cluster centers of the two Waveform databases. The clustering results of FKNN-DPC are much better than those of DPC in terms of Acc, AMI and ARI. This indicates that FKNN-DPC is more robust to noises than DPC. In addition, the figures in Table 4 demonstrate that FKNN-DPC has the best clustering performance on these two waveform databases among the FKNN-DPC, DPC and *K*-means in terms of Acc and ARI, and the *K*-means has the best benchmark values for AMI among these three clustering algorithms. Furthermore, from the performance of FKNN-DPC, DPC and *K*-means on both Waveform databases in terms of Acc, AMI and ARI shown in Table 4, we can see that FKNN-DPC and DPC can obtain better clustering results on the former database than they do on the latter one containing noise attributes. This demonstrates that although FKNN-DPC and DPC are robust to noises, their power are, to some extent, influenced by noises. However the performance of *K*-means on these two Waveform datasets is very similar, with even a little higher clustering accuracy obtained on the latter noisy dataset. This may be due to the fact that *K*-means randomly selects its initial cluster centers.

From the above detailed analysis of the performance of FKNN-DPC with other clustering algorithms including DPC, AP, DBSCAN and *K*-means on the real-world datasets from the UCI machine learning repository, we can conclude that FKNN-DPC has given an overall good performance in clustering. It can find the cluster centers for a dataset and can recognize its clusters and overall its performance is better than that of DPC and other commonly used clustering algorithms.

4.3.2. Experimental analysis of Olivetti face dataset

To further test the power of our FKNN-DPC, we conducted experiments on the Olivetti face database to search and find the density peaks and detect the clusters (i.e., the subjects in the database), and compared the performance of FKNN-DPC with that of DPC, AP, DBSCAN and *K*-means.

The Olivetti face database is a widespread benchmark for machine learning algorithms, with the aim of identifying, without any previous training, the number of subjects in the database [27]. This dataset consists of 40 subjects with 10 different images of each. It poses a serious challenge to DPC because the ideal number of clusters (number of distinct subjects) is comparable with the number of elements in the dataset (number of different images, 10 for each subject). This makes a reliable estimate of the densities difficult [27]. However, it makes little difference to the local density of a point in FKNN-DPC because it depends only on the *K*-nearest neighbors of the point.

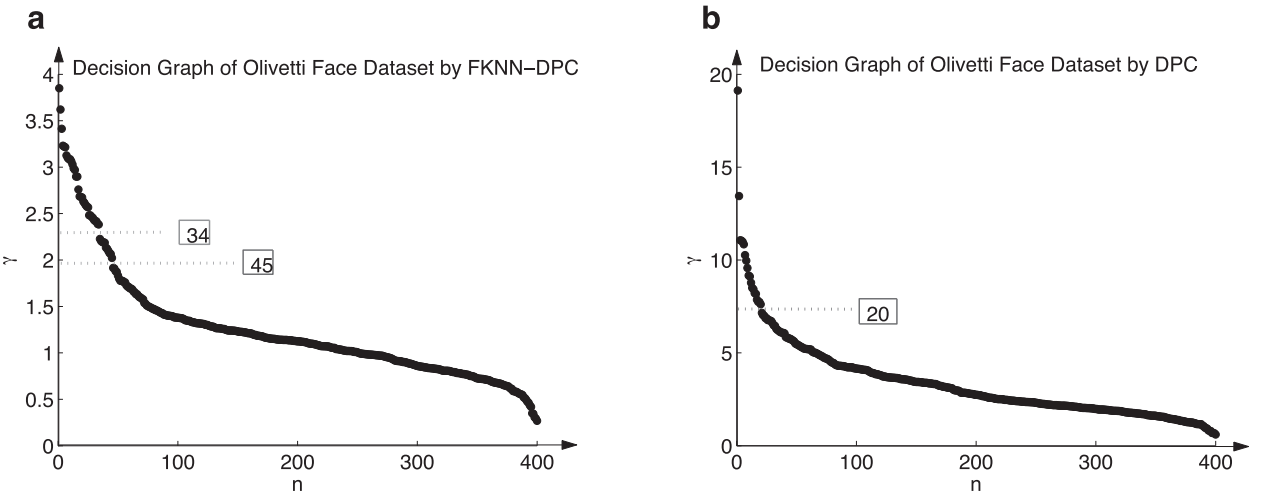

Fig. 8. Decision graphs for the images in Olivetti face database. (a) FKNN-DPC ($K = 4$) and (b) DPC ($d_c = 0.5$).

To reduce the storage cost and the computational load in our experiments, we compressed each image in the Olivetti face database from a 92 × 112 matrix to a 15 × 15 matrix, and performed PCA (Principal Component Analysis) [26] on the condensed image matrix by preserving its 90% accumulation variance so reducing the dimension of an image instance. The distance between the compressed images is computed using the Euclidean metric. The decision graphs of FKNN-DPC and DPC for the images in the Olivetti face database are respectively displayed in Fig. 9(a) and (b), where the *Y*-axis is the value of γ ($\gamma_i = \rho_i \delta_i$) in decreasing order for the images in the database, and the *X*-axis refers to the corresponding images.

From the experimental results shown in Fig. 8 we can see that FKNN-DPC can detect 34 or 45 density peaks, while only 20 density peaks are found by DPC, so we can say that FKNN-DPC outperforms DPC in searching and finding the cluster centers on the Olivetti face database.

After the density peaks of the Olivetti face database have been found, a slightly more restrictive criterion in assignment strategy is adopted by DPC to assign an image to the same cluster of its nearest image with higher density only if the distance between them is smaller than the cutoff distance d_c [27]. As a consequence, the images further than d_c from any other image of higher density remain unassigned. The assigned images comprise the core of a cluster in DPC, which correspond to those images assigned in the assignment strategy 1 by our FKNN-DPC.

The cores of clusters achieved by FKNN-DPC and DPC for the Olivetti face database are respectively represented in Fig. 9(a) and (b). We show, with different colors, the clusters in Fig. 9(a) corresponding to the 34 cluster centers found by FKNN-DPC and in Fig. 9(b) the 20 cluster centers found by DPC. Each cluster of one color corresponds to one subject. A gray image means that the image wasn't assigned to any cluster. The white dot at the upper right corner of an image indicates that image is a density peak.

We analyzed the performance of FKNN-DPC represented in Fig. 9(a), and found those 34 cluster centers covered 30 subjects. There is only one image that has been assigned by FKNN-DPC for subjects 1, 28 and 40, and they are respectively assigned to subjects 17, 37 and 5 erroneously. There may be two reasons for this erroneous assignment: the approach used by FKNN-DPC to preprocess the data for the Olivetti face database loses information embedded in the data; and the assignment strategy 1 of FKNN-DPC causes the error, such as S_1^{10} (the 10th image of subject 1) is one of the *K*-nearest neighbors of S_{17}^5 (a density peak), so S_1^{10} was assigned to the same cluster as S_{17}^5 according to the assignment strategy 1 of FKNN-DPC. From the performance of DPC represented in Fig. 9(b), we can see that the 20 cluster centers found by DPC are dispersed among 19 subjects. Similar to FKNN-DPC, DPC erroneously identified images of different subjects as one subject, e.g., S_5^{10} and S_{40}^5 are recognized as images of subject 40, where the image S_{40}^5 is the density peak. The pictorial representations shown in Fig. 9(a) and (b) indicate that both FKNN-DPC and DPC have the risk that a few subjects are split between two clusters. This phenomenon is more prevalent in FKNN-DPC than that in DPC because FKNN-DPC can find many more density peaks than DPC, which makes the risk of recognizing one subject as two different ones more likely.

In order to thoroughly compare the performance of the clustering algorithms including FKNN-DPC, DPC, AP, DBSCAN and *K*-means on detecting the subjects of the Olivetti face database, we evaluate these algorithms in terms of Acc, AMI and ARI by clustering images in the database into 40 clusters, for there are 40 subjects in the database. We first ranked the instances by their γ ($\gamma_i = \rho_i \delta_i$) value in descending order and selected the top 40 as the cluster centers for FKNN-DPC and DPC, then assigned the images to their proper clusters according to the related assignment strategies. For AP and DBSCAN we complete the clustering process by adjusting their parameters, so that the clustering is composed of 40 clusters. For *K*-means we group images into 40 clusters. We compared the clustering results of these algorithms in Table 5 in terms of Acc, AMI and ARI. The number of the cluster centers found by FKNN-DPC and DPC or the exemplars found by AP are also compared to the number of subjects in which the cluster centers dispersed in the Olivetti face database in Table 5 by F/P

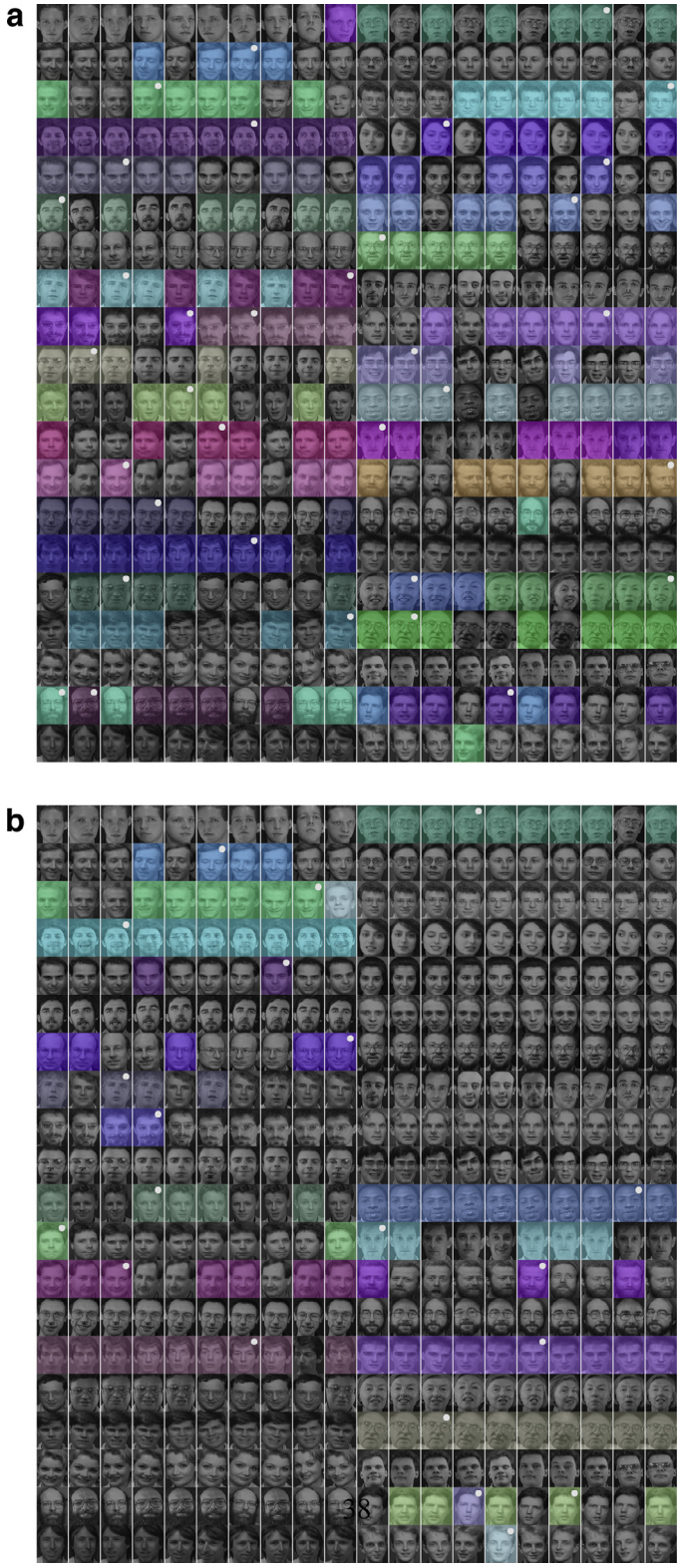

Fig. 9. Pictorial representation of cluster cores for the Olivetti face database. Faces with the same color belong to the same cluster, whereas gray images are not assigned to any cluster. Cluster centers are labeled with white dots. (a) The cores of 34 centers by FKNN-DPC and (b) the cores of 20 centers by DPC.

Table 5
Performance comparison of algorithms by clustering criteria for Olivetti face database.

Algorithm	FKNN-DPC					DPC				
	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
	0.818 34 clusters	0.832	**0.714**	40/36	4	0.665 20 clusters	0.728	0.560	40/32	0.5
	0.690 45 clusters	0.744	0.581	34/30	4	0.460	0.587	0.401	20/19	0.5
	0.785	**0.837**	**0.714**	45/37	4					
Algorithm	DPC_2					AP				
	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
	0.738	0.791	0.655	40/32	0.5	0.763	0.737	0.619	40/36	2.1
Algorithm	DBSCAN					*K*-means				
	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
	0.648	0.691	0.526	–	0.6/4	0.681	0.742	0.585	–	40

with F the number of cluster centers found and P the subjects containing the cluster centers. The pre-specified parameters of each algorithm, Par, are also displayed in Table 5. The bold font in Table 5 indicates the best results.

It should be noted that the clustering of FKNN-DPC is obtained by using its assignment strategies 1 and 2 in turn. The clustering of DPC is achieved by using a two-step assignment after the 40 cluster centers have been found. The first step is to get the cores of clusters by using the aforementioned strict assignment strategy then in the second step those remaining images are assigned to the same cluster of the nearest image which has been assigned in the first step. In addition, we used the assignment strategy 2 of FKNN-DPC instead of the second assignment step of DPC, in order to obtain a clustering with a much better chance to be correct than that in DPC. The modified DPC is referred to as DPC_2, and its clustering result is also shown in Table 5. Because FKNN-DPC can find **34** or **45** density peaks in the Olivetti face database, and DPC can find **20** density peaks, we also display in Table 5 the results for FKNN-DPC in clustering all images into **34** and **45** clusters, and those of DPC to cluster all 400 images into **20** clusters.

The figures in Table 5 reveal that the 40 cluster centers with the highest values of γ selected by FKNN-DPC covered 36 subjects while for DPC or DPC_2 only 32 subjects are detected. The related benchmark results for FKNN-DPC are much better than that for DPC or DPC_2 in terms of Acc, AMI and ARI. The clustering results of DPC_2 are better than that of DPC even though the only difference between them is the assignment strategy for the remaining images except for cores. The detailed analysis of DPC and DPC_2 will show the reasons why the latter can obtain better clustering results. The core produced by DPC is composed of images that are assigned to the same cluster as its nearest image with higher density only if their distance is smaller than the cutoff distance d_c. The strict assignment strategy in DPC caused the cores to have a small number of images, e.g., the cores produced by DPC only include 182 images. The remaining 218 images are assigned by DPC to the same cluster as its nearest neighbor which has been assigned. The assignment strategy of DPC causes large errors in its clustering compared to the true distribution. However, DPC_2 assigns each remaining image, except for those in cores, to its most probably correct cluster according to its highest membership degree. As a consequence the clustering result of DPC_2 is better than that of DPC. This difference appearing in Table 5 demonstrates the great power of the assignment strategy 2 proposed in this paper. The 40 exemplars found by AP covered 36 subjects in the Olivetti face database. The clustering results of AP in terms of Acc, AMI and ARI are worse than those of FKNN-DPC but better than that of DPC, DBSCAN and *K*-means. DBSCAN has the worst clustering results among the clustering algorithms, while *K*-means achieves a competitive clustering result, even better than that of DPC.

From the above detailed analysis for the benchmark data of each clustering algorithm on the Olivetti face database, we can see that FKNN-DPC has performed best followed by AP, DPC_2, *K*-means, and DPC in turn. The DBSCAN gave the worst performance.

In addition, from the figures in Table 5, we can see that the 34 cluster centers found by FKNN-DPC sit in 30 clusters (subjects) and the 45 cluster centers by FKNN-DPC are in 37 clusters (subjects), while the 20 cluster centers produced by DPC are scattered across 19 clusters (subjects). That is to say FKNN-DPC can recognize about 37 subjects in the Olivetti face database out of 40, whereas DPC can recognize no more than 19 subjects out of 40. Furthermore, the clustering results shown in Table 5 corresponding to the 34 or 45 cluster centers by FKNN-DPC and the 20 cluster centers by DPC demonstrate that the performance of FKNN-DPC is much better than that of DPC in terms of Acc, AMI and ARI, and the benchmarks of FKNN-DPC for 45 cluster centers defeat that of 34 cluster centers.

4.3.3. Statistical test of algorithms

This subsection will undertake statistical tests on clustering algorithms, including our FKNN-DPC, the original DPC, AP, DBSCAN and *K*-means, to judge whether or not our FKNN-DPC results are statistically significant. We adopt the Friedman's test [2] to discover the significant difference between the compared clustering algorithms. If a significant difference has been detected by Friedman's test, then the Multiple comparison test is used as a *post hoc* test to detect the significant

Table 6
Paired rank comparison of 5 clustering algorithms in Acc. Upper triangle shows the difference between algorithms. Lower triangle shows pairs with statistical significance.

Algorithm	FKNN-DPC	DPC	AP	DBSCAN	*K*-means
FKNN-DPC		2.0000	1.1364	2.5909	1.5455
DPC	*		−0.8636	0.5959	−0.4545
AP				1.4545	0.4091
DBSCAN	*				−1.0455
K-means					

Table 7
Paired rank comparison of 5 clustering algorithms in AMI. Upper triangle shows the difference between algorithms. Lower triangle shows pairs with statistical significance.

Algorithm	FKNN-DPC	DPC	AP	DBSCAN	*K*-means
FKNN-DPC		2.0000	1.8182	2.6818	1.2273
DPC	*		−0.1818	0.6818	−0.7727
AP				0.8636	−0.5909
DBSCAN	*				−1.4545
K-means					

Table 8
Paired rank comparison of 5 clustering algorithms in ARI. Upper triangle shows the difference between algorithms. Lower triangle shows pairs with statistical significance.

Algorithm	FKNN-DPC	DPC	AP	DBSCAN	*K*-means
FKNN-DPC		2.0909	1.5455	2.4545	1.6364
DPC	*		−0.5455	0.3636	−0.4545
AP				0.9091	0 .0909
DBSCAN	*				−0.8182
K-means					

difference between pairs of clustering algorithms. Friedman's test is considered preferable for comparing algorithms over several datasets without any normal distribution assumption [2]. We conduct Friedman's test at α=0.05 using the clustering results of algorithms in terms of Acc, AMI and ARI on real-word datasets including those from the UCI machine learning repository and those from the Olivetti face dataset (Sections 4.3.1 and 4.3.2). In all three cases, Friedman's test showed significant difference between the compared clustering algorithms. For Acc, $\chi^2 = 16.8584$, $df = 4$, $p = 0.0021$; for AMI, $\chi^2 = 17.9594$, $df = 4$, p=0.0013; and for ARI, $\chi^2 = 15.4909$, $df = 4$, $p = 0.0038$. So we can say that there are strong significant differences between these clustering algorithms.

We show the multiple comparison test between each pair of algorithms at the confidence level of 0.95 in Tables 6–8. The mean rank difference between algorithms is shown in the upper triangle of the tables, and the statistical significance of pairs of algorithms in the lower triangle. The multiple comparison tests, respectively, by clustering accuracy, clustering measurement AMI and ARI shown in Tables 6–8 all reveal that our FKNN-DPC is much better than the original DPC and DBSCAN. From these statistical tests we can conclude that our FKNN-DPC is a very powerful clustering algorithm.

4.4. Run time comparison of algorithms

This subsection will compare the time performance of our proposed FKNN-DPC with that of DPC, AP, DBSCAN and *K*-means on the real-world datasets from the UCI machine learning repository [1]. We have analyzed the time complexity of FKNN-DPC and DPC in Section 3.2.1, and know that both DPC and our FKNN-DPC have the approximate time complexity of $O(n^2)$, where n is the size of the dataset. However, the time consumed by DPC arises mainly from computing distances between points, the local densities and the δ values of points, while the time consumed by FKNN-DPC is mainly from the search of K nearest neighbors of point i to compute its local density ρ_i, and from the assignment for a few instances in strategy 2. So although the approximate time complexity of DPC and FKNN-DPC are both of the same magnitude $O(n^2)$, their execution times on a real-world dataset may be different. Table 9 shows the run time of each algorithm in seconds on the real-world datasets from the UCI machine learning repository.

The figures in Table 9 show that the run time of FKNN-DPC is about three or more times that of DPC on a relatively small datasets, where the difference mainly comes from the assignment strategy 2 of FKNN-DPC for instances not in the cores. Although the computational load of the local densities for points grows very quickly with the size of a dataset, the time consumed by assignment strategy 2 in FKNN-DPC grows slowly with the increasing size of a dataset. This led to the gap in run time between DPC and FKNN-DPC being reduced, which can be seen from the execution time of DPC and FKNN-DPC on the two relatively large datasets of Waveform and Waveform (noise).

Table 9
Run time of 5 clustering algorithms in seconds on UCI datasets.

Dataset	FKNN-DPC	DPC	AP	DBSCAN	*K*-means
Iris	0.148	0.049	0.565	0.059	0.014
Wine	0.168	0.048	0.832	0.098	0.013
WDBC	0.464	0.092	6.115	0.884	0.018
Seeds	0.164	0.05	0.973	0.122	0.014
Libras movement	2.602	0.068	3.016	0.309	0.075
Ionosphere	0.309	0.064	2.018	0.349	0.021
Segmentation	0.313	0.806	66.79	8.727	0.062
Dermatology	0.409	0.063	2.185	0.513	0.007
Pima-indians-diabetes	0.892	0.126	9.709	2.018	0.009
Parkinsons	0.263	0.048	0.866	0.114	0.003
Waveform	7.775	3.51	–	–	0.067
Waveform(noise)	7.525	3.784	–	–	0.109

The time complexity of AP mainly comes from the iterations before it converges. So even though its time complexity for computing the similarity matrix is $O(n^2)$, its total execution time is very large, which can be seen from the figures in Table 9. In addition, it is very challenging to search for the optimal parameters for AP when the dataset is relatively large. That is why the run time of AP on the two Waveform datasets was not evaluated.

The approximate time complexity of DBSCAN is also $O(n^2)$. The figures in Table 9 show that the execution time of DBSCAN is similar to that of DPC or FKNN-DPC when the size of a datasets is relatively small, but it grows quickly when the size of a dataset increases. It is well known that the performance of DBSCAN depends on two parameters, but it is a challenge to search for their two optimal values when the dataset is big. So the run time of DNSCAN on two Waveform datasets was also not evaluated.

K-means is a widely used clustering algorithm because of its efficiency. Its approximate time complexity is $O(n)$. The figures in Table 9 of its run time on the real-world datasets from the UCI machine learning repository demonstrate that it is indeed a fast clustering algorithm. Although *K*-means is a very popular clustering algorithm, there are several deficiencies which can be found in the literature.

5. Conclusions and future work

A robust clustering algorithm was proposed in this paper. This proposed algorithm introduced a uniform metric to calculate the local density of a point for any size of datasets, and developed two assignment strategies to detect the true distribution of a dataset. The proposed clustering algorithm performs fast search and find of density peaks, say, cluster centers of a dataset of any size, and recognizes clusters with any arbitrary shape or dimensionality. The proposed algorithm is named FKNN-DPC which means that it finds cluster centers via calculating the local density of a point by its *K*-nearest neighbors, and detects clusters via assigning remaining points after the density peaks have been found to their most appropriate clusters in turn using two strategies respectively based on the *K*-nearest neighbors of a point starting from cluster centers or the membership degree of a point to a cluster calculated via the fuzzy weighted *K*-nearest neighbors of the point. The FKNN-DPC successfully addressed several issues arising from the clustering algorithm of Alex Rodríguez and Alessandro Laio [27] including that in its density metric and the potential issue hidden in its single step assignment strategy.

The power of FKNN-DPC was tested on several synthetic datasets and on the real-word datasets from the UCI machine learning repository and the well known Olivetti face database. The experimental results on these data demonstrate that our FKNN-DPC is powerful in finding cluster centers and in recognizing clusters regardless of their shape and of the dimensionality of the space in which they are embedded and of the size of the datasets, and is robust to outliers. It performs much better than the original algorithm DPC.

However, the constant *K* of our FKNN-DPC is pre-specified by hand, and the density peaks, say the cluster centers of each dataset, are found by analysis of the decision graph. How to chose the *K* and discover the density peaks of a dataset automatically need further research.

Acknowledgment

We are much obliged to professor Witold Pedrycz for his very valuable comments and advice on our manuscript. We also thank the anonymous referees for their helpful comments which have led to many improvements in the paper. We acknowledge professors Alex Rodríguez and Alessandro Laio for sharing with us their original code of their algorithm (referred to as DPC in this paper) published in Science at June 2014. We also acknowledge many other researchers who provided the original code of their algorithms.

This work is supported in part by the National Natural Science Foundation of China under Grant no. 31372250, is also supported by the Key Science and Technology Program of Shaanxi Province of China under Grant no. 2013K12-03-24, and is

at the same time supported by the Fundamental Research Funds for the Central Universities under Grant no. GK201503067, and by the International Science& Technology Cooperation Program of China under Grant no. 2011DFA12910.

References

[1] K. Bache, M. Lichman, UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml
[2] A. Borg, N. Lavesson, V. Boeva, Comparison of clustering approaches for gene expression data., in: Proceedings of the SCAI, 2013, pp. 55–64.
[3] H. Chang, D.-Y. Yeung, Robust path-based spectral clustering, Pattern Recognit. 41 (1) (2008) 191–203.
[4] J.-L. Chen, J.-H. Wang, A new robust clustering algorithm-density-weighted fuzzy c-means, in: Proceedings of the IEEE SMC '99 Conference on Systems, Man, and Cybernetics, 1999. 1999 IEEE International Conference on, vol. 3, 1999, pp. 90–94vol.3, doi:10.1109/ICSMC.1999.823160.
[5] Y. Cheng, Mean shift, mode seeking, and clustering, Pattern Anal. Mach. Intell. IEEE Trans. 17 (8) (1995) 790–799.
[6] T. Cover, P. Hart, Nearest neighbor pattern classification, Inf. Theory, IEEE Trans. 13 (1) (1967) 21–27.
[7] S.A. Dudani, The distance-weighted k-nearest-neighbor rule, Syst. Man Cybern. IEEE Trans. SMC-6 (4) (1976) 325–327.
[8] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA, 1996, pp. 226–231.
[9] D. Feldman, M. Schmidt, C. Sohler, Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering, in: Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, in: SODA '13, SIAM, 2013, pp. 1434–1453. URL http://dl.acm.org/citation.cfm?id=2627817.2627920
[10] P. Fränti, O. Virmajoki, Iterative shrinking method for clustering problems, Pattern Recognit. 39 (5) (2006) 761–775.
[11] P. Fränti, O. Virmajoki, V. Hautamaki, Fast agglomerative clustering using a k-nearest neighbor graph, Pattern Anal. Mach. Intell. IEEE Trans. 28 (11) (2006) 1875–1881.
[12] B.J. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (5814) (2007) 972–976.
[13] L. Fu, E. Medico, Flame, a novel fuzzy clustering method for the analysis of DNA microarray data, BMC Bioinf. 8 (2007).
[14] A. Gionis, H. Mannila, P. Tsaparas, Clustering aggregation, ACM Trans. Knowl. Discov. Data 1 (1) (2007) Article 4, doi:10.1145/1217299.1217303.
[15] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, The Morgan Kaufmann Series in Data Management Systems, 3rd ed., Morgan Kaufmann, 2011.
[16] R. Hathaway, Y. Hu, Density-weighted fuzzy c-means clustering, Fuzzy Syst. IEEE Trans. 17 (1) (2009) 243–252, doi:10.1109/TFUZZ.2008.2009458.
[17] J. Huang, J. Kang, J. Qi, H. Sun, A hierarchical clustering method based on a dynamic synchronization model, Sci. China: Inf. Sci. 43 (5) (2013) 599–610.
[18] A.K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognit. Lett. 31 (8) (2010) 651–666.
[19] W. Jin, A.K.H. Tung, J. Han, W. Wang, Ranking outliers using symmetric neighborhood relationship, in: W.-K. Ng, M. Kitsuregawa, J. Li, K. Chang (Eds.), Advances in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science, vol. 3918, Springer Berlin Heidelberg, 2006, pp. 577–593.
[20] C. Li, Y. Tan, A weighted fuzzy clustering algorithm based on density, in: Advances in Future Computer and Control Systems, Springer, 2012, pp. 205–211.
[21] A. Likas, N. Vlassis, J. J. Verbeek, The global k-means clustering algorithm, Pattern Recognit. 36 (2) (2003) 451–461.
[22] Q.-B. Liu, S. Deng, C.-H. Lu, B. Wang, Y.-F. Zhou, Relative density based k-nearest neighbors clustering algorithm, in: Proceedings of the International Conference on Machine Learning and Cybernetics, 2003, vol. 1, 2003, doi:10.1109/ICMLC.2003.1264457. 133–17
[23] J. Ma, J. Liu, The BYY annealing learning algorithm for gaussian mixture with automated model selection, Pattern Recognit. 40 (7) (2007) 2029–2037.
[24] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, in: *vol. 1: Statistics*, University of California Press, 1967, pp. 281–297.
[25] X.V. Nguyen, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: is a correction for chance necessary? in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009, 2009, pp. 1073–1080.
[26] K. Pearson, On lines and planes of closest fit to systems of points in space, Philos. Mag. 2 (6) (1901) 559–572.
[27] A. Rodríguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[28] F. Samaria, A. Harter, Parameterisation of a stochastic model for human face identification, in: Proceedings of the Second IEEE Workshop on Applications of Computer Vision, 1994, 1994, pp. 138–142.
[29] H. Tong, U. Kang, Big data clustering, in: A.C. C., R.C. K. (Eds.), Data Clustering: Algorithms and Applications, CRC Press, 2013, pp. 259–276.
[30] U. von Luxburg, R.C. Williamson, I. Guyon, Clustering: science or art? J. Mach. Learn. Res. - Proc. Track 27 (2012) 65–80.
[31] Y. Xiao, J. Yu, Semi-supervised clustering based on affinity propagation algorithm, J. Softw. 19 (11) (2008) 2803–2813.
[32] J. Xie, S. Jiang, W. Xie, X. Gao, An efficient global K-means clustering algorithm, J. Comput. 6 (2) (2011) 271–279.
[33] R. Xu, I. Wunsch D., Survey of clustering algorithms, Neur. Netw. IEEE Trans. 16 (3) (2005) 645–678.
[34] X. Zhu, A.B. Goldberg, Introduction to Semi-Supervised Learning, *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Morgan & Claypool Publishers, 2009.