Pattern Recognition 42 (2009) 689–698

Contents lists available at ScienceDirect

Pattern Recognition

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

A multi-prototype clustering algorithm

Manhua Liu[a,*], Xudong Jiang[b], Alex C. Kot[b]

[a]Department of Instrument Science and Engineering, Shanghai Jiao Tong University, No. 800, Dong Chuan Road, Shanghai, 200240, PR China
[b]EEE, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore

ARTICLE INFO

Article history:
Received 21 November 2006
Received in revised form 28 May 2008
Accepted 9 September 2008

Keywords:
Data clustering
Cluster prototype
Separation measure
Squared-error clustering

ABSTRACT

Clustering is an important unsupervised learning technique widely used to discover the inherent structure of a given data set. Some existing clustering algorithms uses single prototype to represent each cluster, which may not adequately model the clusters of arbitrary shape and size and hence limit the clustering performance on complex data structure. This paper proposes a clustering algorithm to represent one cluster by multiple prototypes. The squared-error clustering is used to produce a number of prototypes to locate the regions of high density because of its low computational cost and yet good performance. A separation measure is proposed to evaluate how well two prototypes are separated. Multiple prototypes with small separations are grouped into a given number of clusters in the agglomerative method. New prototypes are iteratively added to improve the poor cluster separations. As a result, the proposed algorithm can discover the clusters of complex structure with robustness to initial settings. Experimental results on both synthetic and real data sets demonstrate the effectiveness of the proposed clustering algorithm.

1. Introduction

Clustering is the unsupervised classification of patterns into groups [1]. It is widely used in data analysis such as data mining, pattern recognition and information retrieval. The Voronoi diagram also provides a means of naturally partitioning space into subregions to facilitate spatial data analysis and has been applied for data clustering [2–5]. But this technique often implies emphasis on the shape and arrangement of patterns, i.e., the geometric aspect of groups. Clustering techniques have been widely studied in Refs. [6–19]. They suggest more on grouping behavior and can be broadly classified into hierarchical or partitional clustering [1].

Hierarchical clustering is a procedure of transforming the proximity matrix of the data set into a sequence of nested groups in an agglomerative or divisive manner. The agglomerative hierarchical clustering has been widely studied as it allows for more feasible segments to be investigated [7–11]. The *Single-link* [7], *Complete-link* [8] and *average-link* [7] algorithms produce a sequence of clusterings based on the rank order of proximities. The *Single-link* and *Complete-link* algorithms use the distance between two closest and farthest points of two clusters as the cluster distance, respectively. Dependence on a few data points to measure the cluster distance makes these algorithms sensitive to noise. The *average-link* algorithm is more robust to noise by using the average distance of all pairs of patterns from different clusters as the cluster distance. A CURE algorithm [9] represents each cluster with a certain fixed number of well scattered points and shrinks these points toward the cluster center by a specified fraction. This algorithm achieves an improvement of noise robustness over the *Single-link* algorithm. A Chameleon algorithm [10] partitions a constructed k-nearest neighbor graph into a number of subclusters followed by dynamically merging the subclusters. In general, the hierarchical clustering algorithms can provide an easy understanding of the inherent structure of the data set. But they often require high computational cost and large memory space which make them inefficient for large data sets.

Partitional clustering produces a single partition of the data set which aims to optimize a certain cluster criterion function. Many partitional clustering algorithms have been proposed based on different cluster criterions [20–26]. In fact, each cluster criterion imposes a certain structure on the data set. The model-based clustering algorithms assume that the data distribution of a cluster fit a given probability density model such as Gaussian model [20,21]. They can discover the hyper-ellipsoidal clusters. But assumption of a static model makes them ineffective to adequately capture the characteristics of individual clusters, especially when the data set contains the clusters of diverse shapes and densities. Some nonparametric clustering algorithms based on density and grid are proposed to identify clusters by searching the regions of high data density separated by sparse valleys [22–24]. Although these algorithms can find clusters of arbitrary shape, their performances usually degrade for the high dimensional data set. The squared-error clustering algorithm is

* Corresponding author. Department of Instrument Science and Engineering, School of Electronic, Information and Electrical Engineering, Shanghai Jiao Tong University, No. 800, Dong Chuan Road, Shanghai 200240, China. Tel.: +86 34205206; fax: +86 34201372.
E-mail address: pg05080538@ntu.edu.sg (M. Liu).

0031-3203/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.09.015

based on squared error criterion [1,25,30]. It tends to work well with compact clusters of hyper-spherical shape and similar size and is widely studied and used [25–31]. Some new distance measures are proposed to detect clusters with specific characteristics [28,29]. Besides the squared error, other criterions such as the Davies–Bouldin index [32] and cluster variance are imposed as a global criterion to determine the optimum number of clusters [26,31]. Most partitional clustering algorithms require less memory space and computation cost than the hierarchical clustering algorithms. But their clustering results are usually not as good as those of hierarchical clustering. Recently, support vector clustering is proposed to generate the cluster boundaries of arbitrary shape by transforming the original space to a high dimensional space with a kernel function [19]. Although this algorithm can solve some difficult clustering problems, it is not easy to choose a suitable kernel parameter and the clustering result cannot provide information about the representation of cluster.

The hybrid clustering algorithms are proposed to combine the merits of partitional and hierarchical clustering algorithms for better data grouping [12–15,33,34]. They usually partition the data set into a relatively large number of small subclusters and construct a hierarchical structure for them based on a certain cluster distance (similarity) measure. A given number of clusters can be found on the hierarchical structure. A BIRCH algorithm [12] arranges the data set into a number of subclusters represented by *cluster feature* (CF) vectors in a tree structure. It is efficient for large data sets.

In some applications, we may need to efficiently represent data and reduce the data complexity through clustering. A single prototype for each cluster, e.g., the centroid or medoid of cluster in K-means type clustering, may not adequately model the clusters of arbitrary shape and size and hence limit the clustering performance on the complex data structure. This paper proposes a clustering algorithm to represent a cluster by multiple prototypes. The remaining of this paper is organized as follows. Section 2 reviews the related work along with the discussion of their differences. Section 3 presents the proposed multi-prototype clustering algorithm. In Section 4, the proposed algorithm is tested on both synthetic and real data sets and the results are compared to some existing clustering algorithms. Section 5 gives the conclusions.

2. Related work

The squared-error clustering algorithm produces a partition of the data set which aims to minimize the squared error [1,25,30]. Let $X=\{X_1,X_2,\ldots,X_N\}$ where $X_i=[x_{i,1},x_{i,2},\ldots,x_{i,M}]\in\mathbb{R}^M$ be a set of N patterns represented as points in M-dimensional space and K be the number of clusters. The cluster prototypes are denoted by a set of vectors $Z=\{Z_1,Z_2,\ldots,Z_K\}$. The squared error function is computed as

$$E(U,Z)=\sum_{l=1}^{K}\sum_{i=1}^{N}u_{i,l}d^2(X_i,Z_l), \tag{1}$$

subject to

$$\sum_{l=1}^{K}u_{i,l}=1,\quad 1\leqslant i\leqslant N, \tag{2}$$

where $u_{i,l}\in\{0,1\}$ and $u_{i,l}=1$ indicates that pattern i belongs to cluster l. $d(X_i,Z_l)$ is the distance between pattern i and the prototype of cluster l and Euclidean distance measure is often used. K-means clustering algorithm is a simple squared-error clustering algorithm with the number of clusters prespecified [25,27]. The processing steps of the K-means clustering algorithm can be summarized as follows [1]:

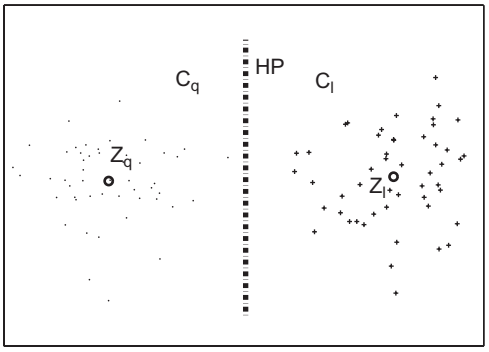

Fig. 1. The separating hyperplane of cluster C_q and C_l represented by prototype Z_q and Z_l, respectively, in squared-error clustering.

(1) Randomly choose K cluster prototypes.
(2) For each pattern, compute its distances to all prototypes and assign it to the closest cluster as

$$u_{i,l}=\begin{cases}1 & \text{if } d^2(X_i,Z_l)=\min\limits_{t=1}^{K}d^2(X_i,Z_t),\\ 0 & \text{otherwise.}\end{cases} \tag{3}$$

(3) Update the prototype of each cluster as

$$Z_l=\frac{\sum_{i=1}^{N}u_{i,l}X_i}{\sum_{i=1}^{N}u_{i,l}}\quad \text{for } 1\leqslant l\leqslant K. \tag{4}$$

(4) Go to step (2) until convergence is achieved.

The K-means clustering algorithm has been widely used in data partitioning because of its low computation and memory space requirements and yet good performance in finding the regions of high density. But the result is sensitive to the initial partition. ISODATA (iterative self-organizing data analysis techniques) algorithm alleviates this problem by iteratively deleting small clusters, merging similar clusters and splitting large clusters until a desired partition is obtained [30]. However, ISODATA algorithm uses single prototype (i.e., centroid) to represent each cluster, which may not adequately model the clusters of arbitrary shape and size no matter how well the data set is partitioned. In the squared-error clustering, the separating boundary for each pair of clusters is the hyperplane through the midpoint of the cluster prototypes and perpendicular to the line connecting these prototypes (see Fig. 1). It may reside in the region of high density for complex clusters (e.g., the separating hyperplanes for two clusters of different size in Fig. 2a and arbitrary shape in Fig. 2c). The hybrid algorithm [13] begins with K-means clustering to partition the data set into a set of subclusters and apply the *Single-link* algorithm to merge them into a given number of clusters according to their cohesions. More complex boundary can be constructed to separate the clusters. But the boundary may still reside in the high-density region resulting in poor cluster separation when the initial partition of K-means clustering is not properly set.

We propose a multi-prototype clustering algorithm to discover the clusters of complex structure. The proposed algorithm begins with the squared-error clustering to partition the data set into a relatively large number of high-density subclusters represented by prototypes. A new cluster separation criterion is proposed to evaluate how well two prototypes are separated by the sparse region. Multiple prototypes with poor separations are grouped to model a given number of clusters in the agglomerative method. New prototypes, i.e., additional prototypes, are iteratively added to improve the poor cluster boundaries until all of them move to sparse region. Therefore, the significant difference of the proposed method from

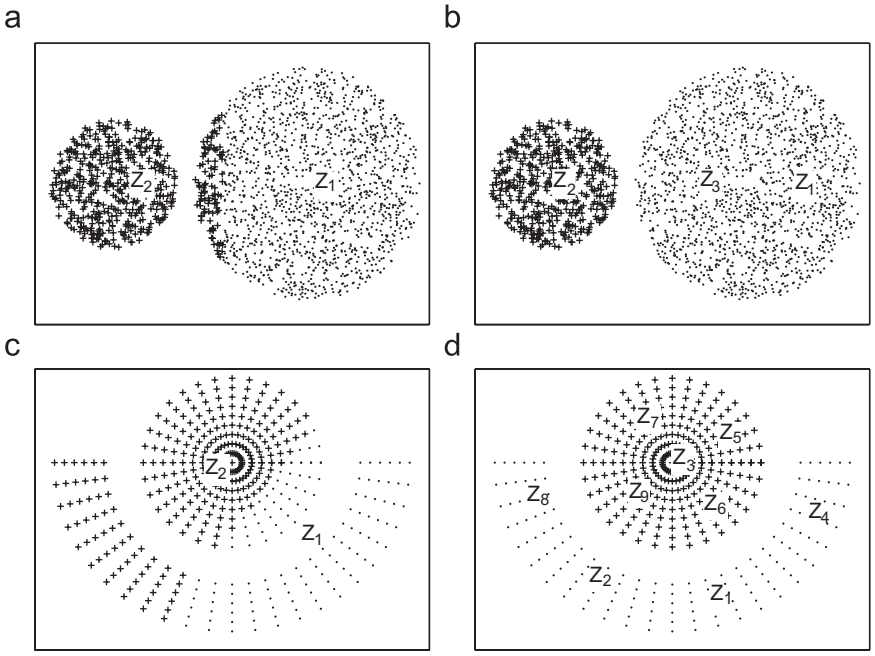

Fig. 2. Clustered data represented by different marks and prototypes denoted by text 'Z_l': (a) poor result with two clusters of different size modelled by 'Z_1' and 'Z_2', (b) good result with the small cluster modelled by 'Z_2' and the large cluster modelled by 'Z_1' and 'Z_3', (c) poor result with two clusters of arbitrary shape modelled by 'Z_1' and 'Z_2' and (d) good result with one cluster modelled by 'Z_1,Z_2,Z_4,Z_8' and another cluster modelled by 'Z_3,Z_5,Z_6,Z_7,Z_9'.

others is that we propose a new cluster separation criterion and our algorithm is targeted at improving the cluster separation.

3. The proposed clustering algorithm

In this section, we first propose a separation measure to evaluate how well two cluster prototypes are separated. Next, we present the proposed multi-prototype clustering algorithm based on the separation measure. Finally, the complexity analysis of the proposed algorithm is provided.

3.1. Separation measure

The separation of two clusters measures how well the clusters are separated. Conceptually, large separation of two clusters indicates less inclination of integrating these clusters into a larger one. It is also called cluster distance or similarity in the literature [7–10]. The distances between the closest or farthest data points of two clusters are used to measure the cluster separation in the agglomerative clustering algorithms [7,8]. They are not only computation expensive but also sensitive to noise due to the dependence on a few points. In the prototype-based clustering algorithms, the separation of two clusters (or prototypes) is often measured using the distance between their prototypes. Although this measure is computationally efficient and robust to noise, it cannot distinguish the clusters of different sizes and shapes. For example, four pairs of clusters in Fig. 3 have equal prototype distances, but their separations are obviously different. A measure of within-to-between cluster spread $R_{q,l} = (e(Z_q) + e(Z_l))/d(Z_q, Z_l)$, where $e(Z_l)$ is the within-cluster variance and $d(Z_q, Z_l)$ is the distance between the prototype Z_q and Z_l, is introduced to evaluate the cluster separation [32]. Including the information of within-cluster variance solves the problem of cluster size but leave the cluster shape problem unsolved. For example, the R of two clusters in Fig. 3d is 0.2931 which is larger than that in Fig. 3c, 0.2455. But two clusters in Fig. 3d are obviously better separated than those in Fig. 3c. A similarity measure of two clusters is proposed by assuming the data distribution of each cluster follows a static model [13]. Although this measure is effective in some cases, it may not adapt to the internal characteristics of the clusters especially when the data set contains the clusters of diverse shapes and distributions.

The cluster prototypes produced by the squared-error clustering are often located in the high-density region. Two prototypes connected by a region of high density are more likely to belong to one cluster than those connected by a sparse region. We propose a separation measure based on the data distribution between two cluster prototypes to evaluate how well the prototypes are separated by a sparse region. Firstly, two cluster prototypes are connected by a line segment.The data points of two clusters are projected onto the line connecting the prototypes since the separating hyperplane is perpendicular to it. Let Z_q and Z_l denote the prototypes of cluster C^q and C^l, respectively. The projections of the data points are computed by

$$x' = \frac{(X - m_0)^T (Z_q - Z_l)}{\|Z_q - Z_l\|^2}, \quad X \in C^q \cup C^l, \tag{5}$$

where $m_0 = (Z_q + Z_l)/2$ is corresponding to the origin of the projections x'. Two cluster prototypes are projected at the positions $-\frac{1}{2}$ and $\frac{1}{2}$ of the line.

Subsequently, we compute the distribution of the projections between the prototypes. For simplicity, we use the histogram to compute the 1-D projected data distribution. The bin center of the histogram is confined in the range of $[-1\ 1]$. To obtain the data densities at the prototypes and origin, the positions -1, $-\frac{1}{2}$, 0, $\frac{1}{2}$ and 1 are specified as the bin centers. $4B+1 (B \geqslant 1)$ bins of equal size (i.e., $1/2B$) are formed and the number of projections x' falling in each bin is counted to produce the histogram. Fig. 3 shows some examples of the 1-D projected distribution where B is set to 6. Let $f(c)$ be the data density at the position c. The data distribution between the prototypes is denoted by the $2B+1$ densities $\left\{f(-\frac{1}{2}), f(-\frac{1}{2}+1/2B), \ldots, f(\frac{1}{2})\right\}$. The smoothness of the data distribution depends on the bin size. To obtain a smooth distribution, Gaussian filter is repetitively applied

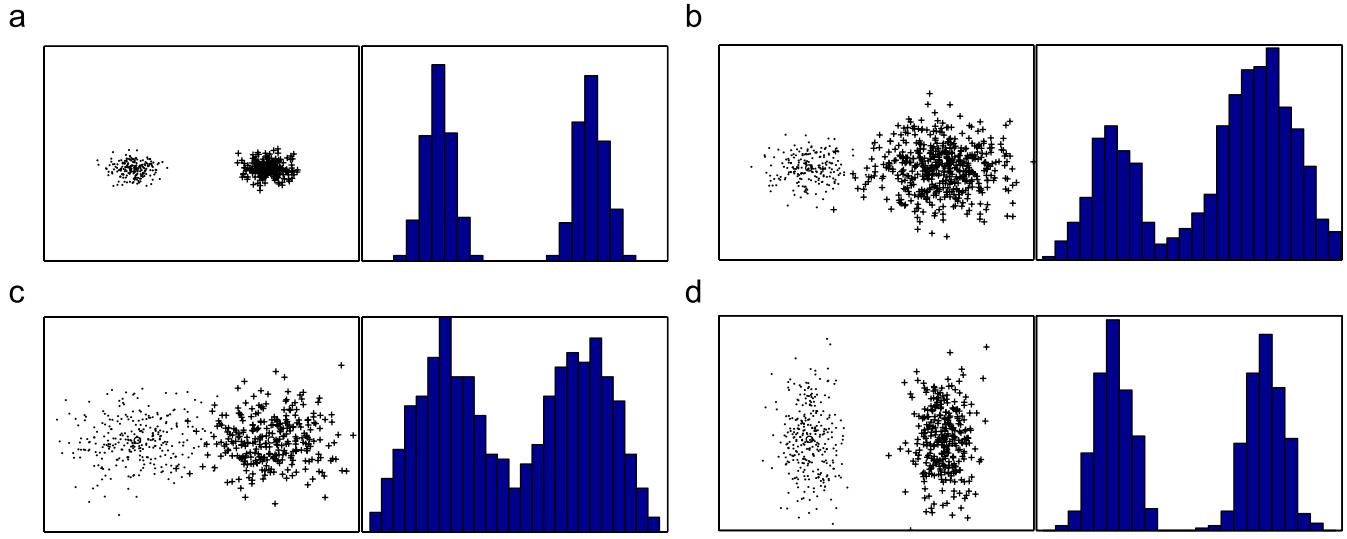

Fig. 3. Four pairs of clusters and their 1-D projected distributions. Their separations are: (a) $sp = 1$ (b) $sp = 0.8219$ (c) $sp = 0.6372$ and (d) $sp = 0.9930$.

on these data densities until only one local minimum exists on them.

Finally, the separation is computed based on the projected data distribution between the prototypes. If the minimum of the $2B+1$ densities between two prototypes is large, the prototypes are connected by a relatively high-density region and hence are inclined to belong to one cluster. Based on the minimum density normalized by the average of those at two prototypes, the separation is computed by

$$sp_{q,l} = 1 - \frac{2\min_{k=1}^{2B+1} f\left(-\frac{1}{2} + \frac{k-1}{2B}\right)}{f\left(-\frac{1}{2}\right) + f\left(\frac{1}{2}\right)}. \tag{6}$$

Large $sp_{q,l}$ indicates that cluster C^l and C^q or their prototypes are well separated by a sparse region. Some examples of the separations between two clusters are shown in Fig. 3. Instead of assuming a static distribution model, the data distribution is automatically estimated in this separation measure which can adapt to the internal characteristics of individual clusters.

The separation in Eq. (6) is based on the minimum data density so that the separating boundary is assumed at the hyperplane through the most sparse region between two prototypes. However, two clusters are separated by the hyperplane through the midpoint of the prototypes in practice. By replacing the minimum density with the density at the midpoint of two prototypes in Eq. (6), we compute the separation of two prototypes based on the current separating hyperplane as

$$sp^0_{q,l} = 1 - \frac{2f(0)}{f(-\frac{1}{2}) + f(\frac{1}{2})}. \tag{7}$$

If $sp^0_{q,l}$ is small, the separating hyperplane resides in the high-density region between two prototypes. Obviously, $sp^0_{q,l} \leqslant sp_{q,l}$.

3.2. The proposed multi-prototype clustering algorithm

For a given data set, the natural clusters often exist in the continuous regions of relatively high density separated by the sparse areas in the pattern space. Using single prototype to represent each cluster often result in the cluster boundaries residing in the region of high density (see Fig. 2a and c). By adding one or more prototypes to model the clusters, the cluster boundary can move to the sparse region of the pattern space (see Fig. 2b) or a more complex boundary can be constructed to separate the complex clusters (see Fig. 2d). We propose a clustering algorithm in which multiple prototypes coexisting within a continuous region of relatively high density are grouped to model the cluster and new prototypes are iteratively added to improve the poor cluster boundaries.

Firstly, we partition the data set into a relatively large number of small subclusters with each one represented by a prototype. Let P ($K \leqslant P < N$) be the number of prototypes for the K clusters. The squared-error clustering has good performance in finding the regions of high density with high computational efficiency and low memory space. It is thus employed in this stage to produce P prototypes. However, the P prototypes may not be appropriately distributed in the high-density regions. Some prototypes may represent the large subclusters consisting of the patterns belonging to different clusters and form a connection between two natural clusters. In addition, some prototypes may reside in the outliers which do not belong to any cluster. Thus, we try to add prototypes in the large subclusters and remove the noise prototypes. For each subcluster l, we compute the within-cluster squared error E_l and the number of patterns N_l. The large subcluster usually has both large N_l and E_l while the noise subcluster often has small N_l. We remove the noise prototypes with $N_l < N_{min}$ ($N_{min} = 0.3N/P$ in our experiments) and add a new prototype in the large subcluster with $N_l > 2.5N_{min}$ and $E_l > E_{max}$ ($E_{max} = \sum_{k=1}^{P} E_k/P + \eta std(E_k), \eta > 0$). The squared-error clustering is repeated until no prototypes are added or removed.

Next, multiple prototypes coexisting within a continuous region of relatively high density are grouped to model the cluster based on the separation measure in Eq. (6). We compute the separation $sp_{q,l}$ between each pair of prototypes and form a separation matrix of $P \times P$. If $sp_{q,l}$ is small, prototype q and l coexist in a high-density region and can be grouped into one cluster. The separation of two multi-prototype clusters C^m and C^n is defined as the smallest separation of two prototypes from different clusters

$$csp_{m,n} = \min_{Z_l \in C^m, Z_q \in C^n} sp_{l,q}. \tag{8}$$

In cluster organization, each prototype forms a cluster initially. Two clusters with the smallest separation are iteratively grouped until K clusters are obtained. This process is similar to the agglomerative *Single-link* clustering [7]. The P prototypes are finally

organized to represent K clusters in the agglomerative method. Thus, the separating boundary of two clusters is composed of multiple hyperplanes determined by the pairs of prototypes from different clusters. For example, the separating boundary of two clusters in Fig. 2d is composed of five hyperplanes determined by the pairs of prototypes: $\{Z_7,Z_8\},\{Z_9,Z_8\},\{Z_9,Z_2\},\{Z_6,Z_1\},\{Z_6,Z_4\}$ and $\{Z_5,Z_4\}$. By grouping multiple prototypes to represent a cluster, complex boundaries can be obtained to separate the non-linearly separable clusters.

The last step is to improve the poor cluster boundaries. The cluster boundary may not reside in the sparse region between two clusters due to the poor initial settings. Adding new prototype can push the poor cluster boundary move to the sparse region or construct a more complex boundary to separate the clusters. The $sp^0_{q,l}$ in Eq. (7) is used to check the separation of cluster hyperplane. If $sp^0_{q,l} < T$, the cluster separating hyperplane is poor. We compute $sp^0_{q,l}$ for each pair of prototypes from different clusters and sort the poor ones ($sp^0_{q,l} < T$) in increasing order. For each pair of prototypes sorted by $sp^0_{q,l}$, if $N_q + N_l > 3N_{min}$ and the separating boundary of the clusters the prototypes belong to has no new prototype already been added to, a new prototype is added to improve the poor cluster boundary. The new prototype is computed as the mean vector of the patterns whose projections x' locate in $[-\frac{1}{4},\frac{1}{4}]$.

After adding new prototypes, the multi-prototype clustering algorithm repeats the above steps until the prototypes do not change. Each pattern in the data set belongs to the cluster consisting of the closest prototype. The processing steps of the proposed algorithm can be summarized as:

(1) Initially set $P \geqslant K$ and randomly choose P cluster prototypes from the data set.
(2) Apply the squared-error clustering on the data set to obtain P subclusters with each one represented by a prototype.
(3) Remove the prototypes of subclusters with $N_l < N_{min}$ and decrease P accordingly. Add a new prototype in the large subclusters with $N_l > 2.5N_{min}$ and $E_l > E_{max}$ and increase P accordingly. If there are prototypes removed or added, go to step (2).
(4) Calculate the separations $sp_{q,l}$ between each pair of prototypes and produce a separation matrix.
(5) Organize the P prototypes into K clusters based on the separation matrix. Two clusters with the smallest separation are iteratively grouped into one cluster until K clusters are obtained.
(6) Compute the separation $sp^0_{q,l}$ between each pair of prototypes from different clusters and sort the poor ones ($sp^0_{q,l} < T$) in increasing order.
(7) For each pair of prototypes sorted by $sp^0_{q,l}$, if $N_q + N_l > 3N_{min}$ and the separating boundary of the clusters the prototypes belong to has no new prototype already been added to, add a new prototype between these two prototypes and increase P accordingly. The new prototype is computed as the mean vector of the patterns whose projections x' locate in$[-\frac{1}{4},\frac{1}{4}]$.
(8) Go to step (2) until the prototypes do not change.
(9) Output the clustering result.

3.3. Complexity analysis

Let m be the number of iterations in the squared-error clustering. The time complexity is $O(NPm)$ for partitioning the data set to produce P prototypes. It is $O(P^2 \log P)$ for the prototype organization which is similar to the *Single-link* algorithm. Thus, the time complexity of the proposed clustering algorithm is $O(NPm + P^2 \log P)$. The space complexity is $O(N)$ for the partitioning of the data set in the squared-error clustering. In the prototype organization, the space complexity is $O(P^2)$ because a separation matrix of size $P \times P$ has to be stored. Thus, the space complexity of the proposed clustering algorithm is $O(N + P^2)$. The number of prototypes P is much smaller than N. Thus, the proposed clustering algorithm requires less memory space and computation cost than the commonly used hierarchical clustering algorithms such as *Single-link* and *Complete-link* while preserves much of the speed and efficiency of the squared-error clustering algorithm.

4. Experimental results and comparisons

The proposed multi-prototype clustering algorithm can be applied for any numerical data set. We conduct a series of experiments on both synthetic and real data sets to demonstrate the clustering performance of the proposed algorithm. The results are compared to some existing clustering algorithms.

4.1. Synthetic data

Clusters on two-dimensional (2D) data sets are easy to visualize and compare. This section tests the proposed clustering algorithm on five synthetic 2D data sets. For good visualization of the clustering results, we represent the clustered data by different marks and denote the prototypes with texts (For example, 'Z_l' denotes prototype l).

To illustrate the process of the proposed algorithm, we first consider the data set shown in Fig. 2c which consists of two clusters of arbitrary shape and size. The clustering result with $P = K$ ($K = 2$) is shown in Fig. 2c. K prototypes cannot adequately model the clusters. Our clustering algorithm initializes P as 5 and set T to 0.8. Since only two clusters exist in the data set, a new prototype is added in each iteration. Fig. 4a shows an initial clustering state of five prototypes produced by the squared-error clustering. The separation $sp_{q,l}$ between each pair of the five prototypes is computed and Table 1 shows the separation matrix. In cluster organization, prototype Z_1 and Z_2 with the smallest separation are first grouped into one cluster $\{Z_1,Z_2\}$. The second smallest separation is $sp_{3,4}$ so that Z_3 and Z_4 are organized into another cluster $\{Z_3,Z_4\}$. The left prototype Z_1 are organized into the same cluster as Z_5 i.e. $\{Z_1,Z_2,Z_5\}$ since $sp_{1,5}$ is the third smallest separation. The cluster boundary is composed of the hyperplanes separating four pairs of prototypes, $\{Z_3,Z_2\}$, $\{Z_3,Z_1\}$,$\{Z_4,Z_1\}$ and $\{Z_4,Z_5\}$. We compute the $sp^0_{q,l}$ between these pairs of prototypes. A new prototype is added between Z_1 and Z_4 since the $sp^0_{1,4} = 0.4253$ is the smallest among them. If the above steps are repeated, an intermediate clustering state of eight prototypes is obtained as shown in Fig. 4b. Since the smallest separation $sp^0_{q,l}$ between two clusters is $sp^0_{4,5} = 0.7460 < T$, a new prototype is added between Z_4 and Z_5. The clustering algorithm terminates at $P = 10$ when the smallest separation between two clusters $sp^0_{4,10} = 0.8400 > T$. Fig. 4c shows the final clustering result of 10 prototypes. We can see that two clusters are correctly discovered.

In addition, we apply the proposed algorithm on four 2D data sets which are often used to test the clustering algorithms [9,10,12,13,22]. Fig. 5 shows the four 2D data sets denoted as DB1, DB2, DB3 and DB4, respectively. DB1 is obtained from Ref. [9]. It contains one big and two small circles and two ellipsoids connected by a chain of

Table 1
The separation matrix of five prototypes in Fig. 4a

sp	Z_1	Z_2	Z_3	Z_4	Z_5
Z_1	–	0.0330	0.8388	0.5747	0.3474
Z_2	0.0330	–	0.8426	0.9008	1.0000
Z_3	0.8388	0.8426	–	0.2663	1.0000
Z_4	0.5747	0.9008	0.2663	–	0.8579
Z_5	0.3474	1.0000	1.0000	0.8579	–

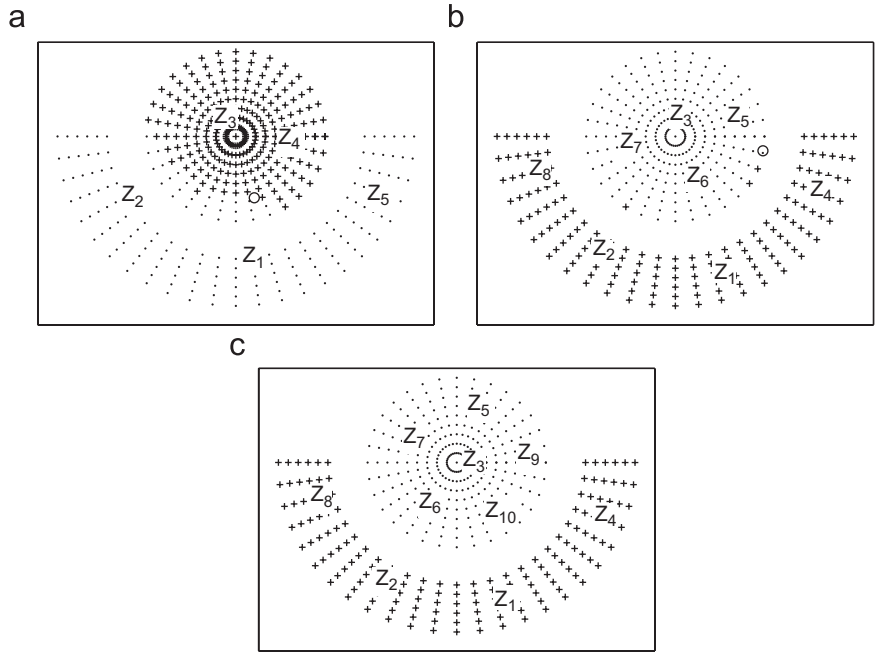

Fig. 4. An illustrative example of our clustering algorithm where '○' denotes the new added prototype: (a) initial clustering state of five prototypes, (b) an intermediate clustering state of eight prototypes and (c) final clustering result of 10 prototypes.

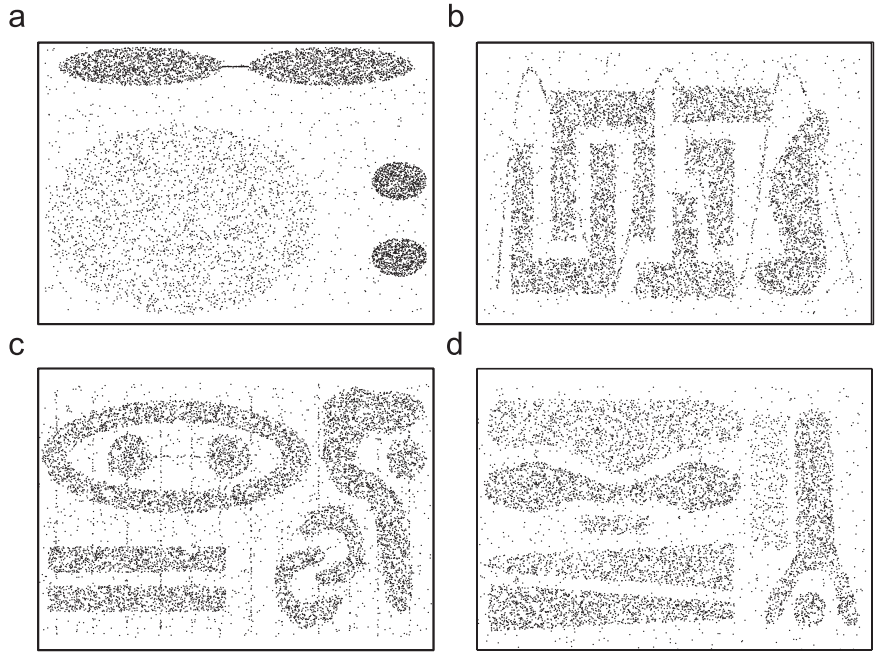

Fig. 5. Four 2D data sets used in our experiments: (a) DB1 with 8000 data points, (b) DB2 with 8000 data points, (c) DB3 with 10,000 data points and (d) DB4 with 8000 data points.

outliers. The other three data sets DB2, DB3 and DB4 are obtained from [10]. These data sets with 8000 to 10,000 data points consist of the clusters of arbitrary shape and size and the outliers are scattered on the data sets, which represent some difficult clustering instances.

Fig. 6 shows the clustering results of the proposed algorithm on these 2D data sets. The total number of prototypes finally obtained to model the clusters are 11, 29, 38 and 59 for the DB1, DB2, DB3 and DB4, respectively. From these figures, we can see that the proposed algorithm successfully discovers the clusters on these data sets.

To show the robustness to initial settings, we perform the proposed clustering algorithm on a poor initialization. DB1 and DB3 are used in this experiment. The initial number of prototypes is set to 20 for both data sets. The separation threshold T is set to 0.45. Fig. 7a and c show the initial clustering results of 19 prototypes on the DB1 and 20 prototypes on the DB3, respectively. DB3 has more complex data structure than DB1 so that the initial 20 prototypes may not adequately model the clusters on DB3. Although 20 prototypes are enough to model the clusters on DB1, the inappropriate

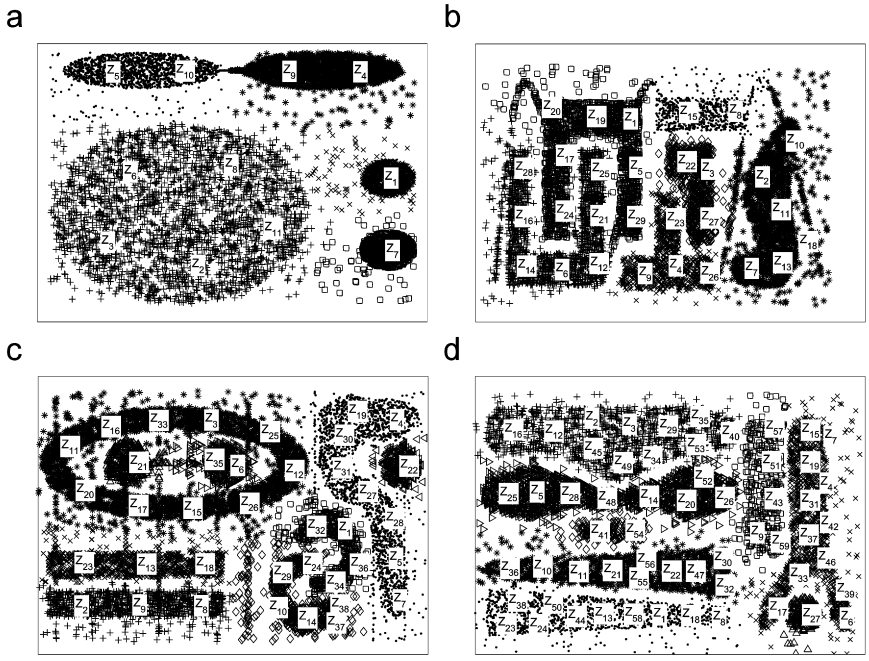

Fig. 6. The clustering results of our proposed algorithm on the four 2D data sets: (a) DB1 with 11 prototypes, (b) DB2 with 29 prototypes, (c) DB3 with 38 prototypes and (d) DB4 with 59 prototypes.

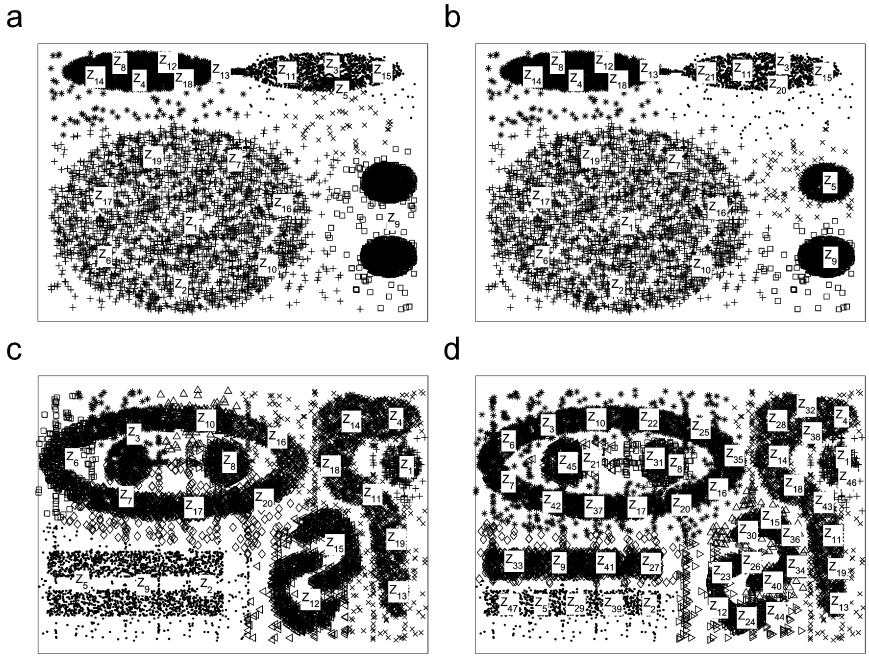

Fig. 7. The clustering results of the proposed algorithm on poor initialization (a) the initial partition of DB1; (b) final result of 21 prototypes on DB1; (c) the initial partition of DB3 and (d) final result of 47 prototypes on DB3.

initial prototypes also result in poor clustering result where two small circles on the right are modelled into one cluster by a prototype. Our proposed algorithm iteratively adds new prototypes to improve the poor cluster boundaries resulted by the inappropriate initial settings. Fig. 7b and d show the final clustering results of 21 prototypes on DB1 and 47 prototypes on DB3, respectively. We can see the clusters on these data sets are successfully discovered.

We compare the proposed algorithm with some existing clustering algorithms on the four 2D data sets. As these data sets contain the clusters of arbitrary shape and size, most variants of the squared-error clustering algorithm such as *K*-means [25] and ISODATA [30], which use single prototype to represent each cluster, cannot correctly discover the clusters on the data sets. DB1 is used to test the CURE clustering algorithm [9]. As stated in Ref. [9], the BIRCH algorithm [12] cannot distinguish between the big and small clusters and the *Single-link* algorithm cannot handle the chain of outliers connecting two ellipsoids. The hybrid clustering algorithm [13] is also tested on the four data sets and performs better than some

existing clustering algorithms such as the *Single-link* [7], *Complete-link* [8], CURE [9], *K*-means [25], BIRCH [12] and DBScan [22] algorithms. As stated in Ref. [13], the *Complete-link*, *K*-means and BIRCH algorithms cannot discover the clusters on the four data sets, while the *Single-link* equipped with outlier elimination, DBScan and CURE algorithms can correctly discover the clusters on one or two of the first three data sets but all fail on the complex DB4. The hybrid algorithm [13] is able to discover the clusters of the four data sets. However, its clustering result is sensitive to the initial settings. As reported in Ref. [13], the probabilities of successful partitions are about 95%, 90%, 65% and 40% on the DB1, DB2, DB3 and DB4, respectively, after performing this hybrid algorithm 20 times with random initialization on each data set. Similarly, the proposed algorithm is performed on each of the four 2D data sets over 20 random runs with the initial number of prototypes set to $3K$. The probabilities of the successful partitions are 100%, 100%, 95% and 90% on the DB1, DB2, DB3 and DB4, respectively. The average numbers of prototypes to finally model the clusters are 17, 26, 39 and 62 for DB1, DB2, DB3 and DB4, respectively. Thus, more prototypes are usually required to represent the clusters which are more complex in shape and size.

4.2. Real data

To show the practical applicability of our proposed multi-prototype clustering algorithm, we apply it on three real data sets: Iris data, Wine Recognition Data and Wisconsin Breast Cancer (WBC) Data, which are available at UCI Machine Learning Repository [35]. The class label is given for each pattern in these data sets. It is ignored during the clustering but used for evaluation of clustering performance. In this work, the clustering error rate is used to evaluate the performance of clustering algorithm. It is computed by [27]:

$$Error = \frac{\text{the number of misclassified patterns}}{\text{the number of patterns in data set}} \times 100\%. \quad (9)$$

To compute the clustering error rate, the major problem is the correspondence between the given class labels and the found clusters. We perform the matching between them. A cluster l corresponds to class label q if the number of patterns labelled as q in l is larger than those of other class labels. The best matching of the clusters is selected as the correspondence to the class labels.

We compare our proposed algorithm with some existing clustering algorithms such as *K*-means, agglomerative hierarchical clustering and hybrid algorithms on these real data sets. The proposed algorithm is performed on each data set over 20 random runs and the best clustering result is presented. We implement the *K*-means algorithm with a good initialization of the cluster centers in Ref. [27] on the WBC Data while the clustering results on the other two real data sets are reported in Ref. [27]. We perform the three common agglomerative clustering algorithms: *Single-link* [7], *Complete-link* [8] and *Average-link* [7] on each data set and present the best clustering result of them. In addition, the hybrid algorithm [13] is also implemented on these real data sets with our best effort. We give the best clustering result after 20 random runs of it on each data set. For other clustering algorithms, we present the results reported in the literature.

The Iris data set consists of 150 patterns and each one is represented by four numerical features: sepal length, sepal width, petal length and petal width. Three types of Iris flowers: setosa, versicolor and virginica are labelled as class I, II and III, respectively. Each class consists of 50 patterns. This data set is often used to test the clustering algorithms and the clustering results of three clusters are reported in Refs. [15,27]. Ref. [15] also gives the clustering results by the *Single-link* and *Complete-link* algorithms. Our algorithm produces five prototypes to model the three clusters on this data set. One prototype is used to represent cluster C1 which is easier to be separated from others. Two prototypes are used to represent each of cluster C2 and C3. Table 2 summarizes the clustering results for this real data set. We can see our algorithm performs better than other clustering algorithms.

Table 2
The clustering results for Iris data

Found cluster	Given class			Clustering error rate (%)				
	I (50)	II (50)	III (50)	Our algorithm	[27]	[13]	*Complete-link*	[15]
C1	50	0	0	2.67	11.33	4.0	4.0	7.4
C2	0	47	1					
C3	0	3	49					

Table 3
The clustering results for Wine Recognition data

Found cluster	Given class			Clustering error rate (%)				
	I (59)	II (71)	III (48)	Our algorithm	[27]	[13]	*Average-link*	[23]
C1	59	2	0	2.25	5.05	3.93	5.62	17.42
C2	0	67	0					
C3	0	2	48					

Table 4
The clustering results for Wisconsin Breast Cancer data

Found cluster	Given class		Clustering error rate (%)				
	I (444)	II (239)	Our algorithm	[27]	[13]	*Average-link*	[11]
C1	427	2	2.78	3.95	3.66	8.20	3.37
C2	17	237					

The Wine Recognition data set contains the results of a chemical analysis of the wines grown in the same region in Italy but derived from three different cultivars. The wines from three cultivars represent three types of wine data labelled as class I, II and III, respectively. This data set consists of 178 patterns and each one is represented by 13 features such as alcohol, magnesium, color intensity, etc. The feature values are normalized to $[0,1]$ to balance the effects of the features measured on different scales. This data set is also used to test the clustering algorithm [23,27]. Five prototypes are produced in our algorithm to model the three clusters on this data set. Two prototypes are used to represent each of cluster C1 and C2. Cluster C3 is represented by one prototype. Table 3 shows the clustering results for this real data set. We can see that the clustering result of our algorithm is better than those of others.

The WBC data set consists of 683 patterns which belong to two types of patterns: 444 benigns and 239 malignants labelled as class I and II, respectively. Each pattern is represented by nine features. This data set is also used to test the clustering algorithm [11]. As stated in Ref. [11], its clustering result is better than those in Refs. [17,18]. Our algorithm produces four prototypes to model the two clusters on this data set. One prototype is used to represent cluster C1 and the other three prototypes represent cluster C2. The clustering results for this real data set are shown in Table 4. Our algorithm performs better than other clustering algorithms on this data set.

The major computation load of the proposed clustering algorithm is in the squared-error clustering of data set and the prototype grouping process. It is well known that the squared-error clustering of data set has low computation cost. Since the number of prototypes is much smaller than the size of data set, the prototype grouping process is also time efficient. The total computation time of the proposed algorithm depends on the number of iterations (squared-error clustering and prototype grouping) required for adding new prototypes to improve the cluster separation. It is heavy as compared

Table 5
The comparison of computational complexity on the three data sets

Data sets	Time consumption in second			
	K-means [27]	Hybrid [13]	*Average/Complete-link* [7]/ [8]	Our algorithm
Iris	0.0015	0.0056	0.0620	0.0150
Wine	0.0017	0.0062	0.1100	0.0160
WBC	0.0019	0.0113	7.8130	0.0310

with the *K*-means and hybrid [13] algorithms due to the iteration of adding new prototypes. To alleviate this problem, we can initially set more prototypes in the squared-error clustering so that fewer prototypes will be added afterwards. In addition, we can consider re-clustering the data of the clusters affected by new prototypes instead of the whole data set to reduce the computation cost. Table 5 shows the computation time of our algorithm compared with the *K*-means, hierarchical and hybrid clustering algorithms on the three real data sets. All the algorithms are implemented in MATLAB and executed on a Dell Precision PWS390 1.86 GHz PC with 1 GB memory running Windows XP professional. Although our algorithm requires a little more computation than the *K*-means and hybrid [13] algorithms, it is much faster than the hierarchical clustering algorithms.

5. Conclusions

In this paper, we have proposed a multi-prototype clustering algorithm which can discover the clusters of arbitrary shape and size. The squared-error clustering is used to produce a number of prototypes and locate the regions of high density because of its low computation and memory space requirements and yet good performance. A separation measure is proposed to evaluate how well two prototypes are separated by a sparse region. Multiple prototypes with small separations are organized to model a given number of clusters in the agglomerative method. New prototypes are iteratively added to improve the poor cluster boundaries resulted by the poor initial settings. The proposed algorithm requires less memory space and computation cost than the commonly used hierarchical clustering algorithms such as *Single-link* and *Complete-link* while preserves much of the speed and efficiency of the squared-error clustering algorithm. Experimental results on both synthetic and real data sets show the effectiveness of the proposed clustering algorithm.

References

[1] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[2] A. Okabe, B. Boots, K. Sugihara, S.N. Chui, Spatial Tessellations, Concepts and Applications of Voronoi Diagrams, Wiley, New York, 2000.
[3] H. Koivistoinen, M. Ruuska, T. Elomaa, A Voronoi diagram approach to autonomous clustering, discovery science, Proceedings of the 9th International Conference, Springer, Berlin, 2006, pp. 149–160.
[4] C. Reyes, M. Adjouadi, A clustering technique for random data classification, IEEE International Conference on Systems, Man and Cybernetics, vol.1, October 1995, pp. 316–321.
[5] J. Li, P. Hao, Hierarchical structuring of data on manifolds, IEEE Conference on Computer Vision and Pattern Recognition, June 2007, pp. 1–8.
[6] A.K. Jain, M. Murthy, P. Flynn, Data clustering: a review, ACM Comput. Surveys 31 (3) (1999) 264–323.
[7] P.H.A. Sneath, R.R. Sokal, Numerical Taxonomy, Freeman, San Francisco, London, 1973.
[8] B. King, Step-wise clustering procedures, J. Am. Statist. Assoc. 69 (1967) 86–101.
[9] S. Guha, R. Rastogi, K. Shim, Cure: an efficient clustering algorithm for large databases, Proceedings of the Conference on Management of Data (ACM SIGMOD), 1998, pp. 73–84.
[10] G. Karypis, E.-H.S. Han, V. Kumar, Chameleon: a hierarchical clustering algorithm using dynamic modeling, IEEE Comput. 32 (8) (1999) 68–75.
[11] A.L. Fred, J.M. Leitao, A new cluster isolation criterion based on dissimilarity increments, IEEE Trans. Pattern Anal. Mach. Intell. 25 (8) (2003) 944–958.
[12] T. Zhang, R. Ramakrishnan, M. Livny, Birch: an efficient data clustering method for very large databases, Proceedings of the Conference on Management of Data (ACM SIGMOD), 1996, pp. 103–114.
[13] C.-R. Lin, M.-S. Chen, Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging, IEEE Trans. Knowl. Data Eng. 17 (2) (2005) 145–159.
[14] M.N. Murty, G. Krishan, A hybrid clustering procedure for concentric and chain-like clusters, Int. J. Comput. Inform. Sci. 10 (6) (1981) 397–412.
[15] E.Y. Cheu, Z.Z. Chee Keong Kwoh, On the two-level hybrid clustering algorithm, The International Conference on Artificial Intelligence in Science and Technology (AISAT), 2004, pp. 138–142.
[16] A. Topchy, A.K. Jain, W. Punch, Clustering ensembles: models of consensus and weak partitions, IEEE Trans. Pattern Anal. Mach. Intell. 27 (12) (2005) 1866–1881.
[17] R. Kothari, D. Pitts, On finding the number of clusters, Pattern Recognition Lett. 20 (1999) 405–416.
[18] S.V. Chakravarthy, J. Ghosh, Scale-based clustering using the radial basis function network, IEEE Trans. Neural Networks 7 (1996) 1250–1261.
[19] A. Ben-Hur, D. Horn, H.T. Siegelmann, V. Vapnik, Support vector clustering, J. Mach. Learn. Res. (2) (2001) 125–137.
[20] G. McLachlan K. Basford, Mixture Models: Inference and Application to Clustering, Marcel Dekker, New York, 1988.
[21] M. Figueiredo, A. Jain, Unsupervised learning of finite mixture models, IEEE Trans. Pattern Anal. Mach. Intell. 24 (3) (2002) 381–396.
[22] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD), 1996, pp. 226–231.
[23] Y. Shi, Y. Song, A. Zhang, A shrinking-based clustering approach for multidimensional data, IEEE Trans. Knowl. Data Eng. 17 (10) (2005) 1389–1403.
[24] M. Ankerst M.M. Breunig H.-P. Kriegel, J. Sander, Optics: ordering points to identify the clustering structure, Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD), 1999, pp. 49–60.
[25] J. McQueen, Some methods for classification and analysis of multivariate observations, Fifth Berkeley Symposium on Mathematical Statistics and Probability 1 (1967) 281–297.
[26] M. Sarkar, B. Yegnanarayana, D. Khemani, A clustering algorithm using an evolutionary programming-based approach, Pattern Recognition Lett. 18 (1997) 975–986.
[27] S.S. Khan, A. Ahmad, Cluster center initialization algorithm for k-means clustering, Pattern Recognition Lett. 25 (2004) 1293–1302.
[28] M.-C. Su, C.-H. Chou, A modified version of the k-means algorithm with a distance based on cluster symmetry, IEEE Trans. Pattern Anal. Mach. Intell. 23 (6) (2001) 674–680.
[29] D. Charalampidis, A modified k-means algorithm for circular invariant clustering, IEEE Trans. Pattern Anal. Mach. Intell. 27 (12) (2005) 1856–1865.
[30] G.H. Ball, D.J. Hall, Isodata, a novel method of data analysis and classification, Technique Report, Stanford University, 1965.
[31] C.J. Veenman, M.J. Reinders, E. Backer, A maximum variance clustering algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 24 (9) (2002) 1273–1280.
[32] D. Davies, D. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. 1 (4) (1979) 224–227.
[33] E.W. Tyree, J.A. Long, The use of linked line segments for cluster representation and data reduction, Pattern Recognition Lett. 20 (1) (1999) 21–29.
[34] M.-C. Su, Y.-C. Liu, A new approach to clustering data with arbitrary shapes, Pattern Recognition 38 (11) (2005) 1887–1901.
[35] C.J. Merz, P.M. Murphy, Uci repository of machine learning databases, Department of Information and Computer Science, University of California ⟨http://www.ics.uci.edu/mlearn/MLRepository.html⟩.

About the Author—MANHUA LIU received B.Eng. degree in 1997 and M.Eng. degree in 2002 in automatic control from North China Institute of Technology and Shanghai JiaoTong University, China, respectively. She got Ph.D. degree in 2007 from Nanyang Technological University (NTU), Singapore. She was a research fellow in NTU. Currently, she is a lecturer in Shanghai Jiao Tong University, PR China. Her research interests include biometrics, pattern recognition, image processing and machine learning and so forth.

About the Author—XUDONG JIANG received B.Eng. and M.Eng. from University of Electronic Science and Technology of China in 1983 and 1986, respectively, and Ph.D. from University of German Federal Armed Forces Hamburg, Germany in 1997, all in electrical and electronic engineering. From 1986 to 1993, he was a Lecturer at the University of Electronic Science and Technology of China where he received two Science and Technology Awards from Ministry for Electronic Industry of China. He was a recipient of German Konrad-Adenauer Foundation young scientist scholarship. From 1993 to 1997, he was with the University of German Federal Armed Forces Hamburg, Germany as a scientific assistant. From 1998 to 2002, He was with the Centre for Signal Processing, Nanyang Technological University, Singapore, as Research/Senior Fellow, where he developed a fingerprint verification algorithm achieving top in speed and second top in accuracy in the International Fingerprint Verification Competition (FVC2000). From 2002 to 2004, he was a Lead Scientist and Head of Biometrics Laboratory at the Institute for Infocomm Research, Singapore. Currently he is an Assistant Professor at the School of EEE, Nanyang Technological University, Singapore. His research interest includes pattern recognition, image processing, computer vision and biometrics.

About the Author—ALEX CHICHUNG KOT was educated at the University of Rochester, New York, and at the University of Rhode Island, Rhode Island, USA, where he received the Ph.D. degree in electrical engineering in 1989. He was with the AT&T Bell Company, New York, USA. Since 1991, he has been with the Nanyang Technological University (NTU), Singapore, where he is Vice Dean of the School of EEE. His research and teaching interests are in the areas of signal processing for communications, signal processing, watermarking, and information security. Dr. Kot served as the General Co-Chair for the Second International Conference on Information, Communications and Signal Processing (ICICS) in December 1999, the Advisor for ICICS'01 and ICONIP'02. He received the NTU Best Teacher of the Year Award in 1996 and has served as the Chairman of the IEEE Signal Processing Chapter in Singapore. He is the General Co-Chair for the IEEE ICIP 2004 and served as Associate Editor for the IEEE Transactions on Signal Processing and the IEEE Transactions on Circuits and Systems for Video Technology.