Competitive algorithms for the clustering of noisy data

Tai-Ning Yang^a, Sheng-De Wang^{b,*}

^a*Department of Computer Science, Chinese Culture University, Taipei 111, Taiwan*

^b*Department of Electrical Engineering, National Taiwan University, EE Building, Taipei 106, Taiwan*

Received 20 July 2001; received in revised form 7 October 2002; accepted 12 November 2002

Abstract

In this paper, we consider the issue of clustering when outliers exist. The outlier set is defined as the complement of the data set. Following this concept, a specially designed fuzzy membership weighted objective function is proposed and the corresponding optimal membership is derived. Unlike the membership of fuzzy *c*-means, the derived fuzzy membership does not reduce with the increase of the cluster number. With the suitable redefinition of the distance metric, we demonstrate that the objective function could be used to extract *c* spherical shells. A hard clustering algorithm alleviating the prototype under-utilization problem is also derived. Artificially generated data are used for comparisons.

© 2002 Elsevier B.V. All rights reserved.

Keywords: Clustering; Outlier set; Algorithms

---

1. Introduction

Clustering algorithms try to partition a set of unlabeled input vectors into a number of subsets (clusters) such that data in the same subset are more similar to each other than to data in other subsets. There are two kinds of unsupervised clustering algorithms: hierarchical versus partitional. Hierarchical clustering generates a sequence of nested partitions from the proximity matrix. In social, biological, and behavioral sciences, hierarchical clustering techniques are popular because of the necessity to produce taxonomies. Partitional clustering is used frequently in the engineering applications such as data compression and image segmentation. A single partition of the data is generated in partitional clustering. Clustering algorithms can also be divided into two types: hard versus fuzzy. Hard (crisp or conventional) clustering assigns each input vector to exactly one cluster. In fuzzy clustering, a given pattern does not necessarily belong to only one cluster but can have varying

---

* Corresponding author. Tel.: +886-22-3635251; fax: +886-22-3671909.

E-mail address: sdwang@cc.ee.ntu.edu.tw (S.-D. Wang).

degrees of memberships to several clusters. Many clustering algorithms could be found in the related books [5,19,18,12,2].

Recently the on-line competitive learning in neural networks is often associated with the sequential partitional clustering [3]. The searching and updating process for an appropriate prototype is considered as the competition between the hidden neurons. It was pointed out that the basic form of the competitive learning is the same as the sequential version of hard c -means. Kohonen proposed a competitive algorithm called learning vector quantization (LVQ) [14,15]. LVQ has attracted a lot of attentions because of its simplicity and efficiency. There are many variants of LVQ. For example, Kohonen proposed two refinements, call LVQ2 and LVQ3. Unfortunately LVQ suffers the so-called prototype under-utilization problem that can be serious in some applications.

Some algorithms based on the fuzzy clustering theory have been proposed recently. The approach to cope with the prototype under-utilization problem is to update not only the winner but also the other prototypes for each input. Many fuzzy clustering algorithms including GLVQ-F adopted the membership from fuzzy c -means (FCM) algorithms. Since the objective function in FCM is constrained by a probability premise, the total sum of the membership shared by all classes for an input data must be one. If the number of classes is large, the fuzzy membership may be minute. It has been shown that if the number of classes is large, the traditional clustering algorithms may be better than the fuzzy ones. Moreover, the outliers that may appear are not considered in FCM. We propose a new family of clustering algorithms called fuzzy robust clustering (FRC) with the consideration of outliers. The size of updating prototype is independent of the number of prototypes and the influence of outliers is reduced.

The remaining parts of this paper are organized as follows. In Section 2, we review fuzzy c -means (FCM) algorithms. We propose the fuzzy robust clustering (FRC) algorithm in Section 3. Fuzzy robust c spherical shells (FCSS) is proposed in Section 4. The prototype under-utilization problem of LVQ is discussed and is solved by the proposed hard robust clustering (HRC) algorithms in Section 5. Section 6 demonstrates the simulation results. Section 7 contains the conclusions.

2. Fuzzy c -means and the outliers

Let $X = \{x_1, x_2, \dots, x_n\} \subset R^m$ denote the input sample set and c denote the number of clusters. The prototypes $V = \{v_1, v_2, \dots, v_c\}$ are cluster centers, where $v_i \in R^m$ for $1 \leq i \leq c$. Let U denote the membership matrix. The objective function in *Fuzzy C-Means* (FCM) [1] is

$$E(V, X) = \sum_{i=1}^c \sum_{j=1}^n u_{ij}^m \|x_j - v_i\|^2. \quad (1)$$

In the above equation, $m \in [1, \infty)$ is a weighting factor called the fuzzifier. The elements u_{ij} in the matrix U satisfy the following constraints:

constraint 1. $u_{ij} \in [0, 1]$ for all i, j .

constraint 2. $0 < \sum_{j=1}^n u_{ij} < n$ for all i, j .

constraint 3. $\sum_{i=1}^c u_{ij} = 1$ for all j .

Constraint 3 is a probabilistic premise. If we discard constraint 3, minimizing the objective function (1) will lead to a trivial solution in which all memberships are zero.

Fig. 1. Two clusters and one outlier.

Bezdek [1] applied the technique of Lagrange multiplier and derived the optimal membership function,

$$u_{ij} = \left(\sum_{r=1}^c \frac{\|x_j - v_i\|^{2/(m-1)}}{\|x_j - v_r\|^{2/(m-1)}} \right)^{-1}. \quad (2)$$

A simple example in Fig. 1 illustrates the problems caused by the probabilistic constraint. For FCM-like algorithms with probabilistic constraint, points A and B may have the same membership value. But apparently point A is an outlier. Although points C and D are symmetric to the left cluster, their memberships are different. Probabilistic constraint forces the total sum of the membership for each input data must be one, so the outlier could not be distinguished from the data. Since the total membership is shared by all classes, this makes point D to transfer some left class membership value to right class. If the number, c , of classes is large, the fuzzy membership may be minute. Moreover, this objective function does not consider the situation where outliers exist. To tackle this problem, we have to clarify the concept of outliers first. Assume that A is the universal set including all the possible elements and B is the data set. We define the outlier set as the complement, $(A-B)$.

3. Fuzzy robust clustering algorithms

Now, we propose the objective function that takes into consideration of outliers. Assume that a noise cluster exists outside each data cluster. The fuzzy complement of u_{ij} , $f(u_{ij})$ may be interpreted as the degree to which x_j does not belong to the i th data cluster. Thus the fuzzy complement can be viewed as the membership of x_j in the noise cluster with the distance η_i . The general form of the proposed objective function is

$$FE(V, X) = \sum_{i=1}^c \sum_{j=1}^n u_{ij}^m d^2(x_j, v_i) + \sum_{i=1}^c \sum_{j=1}^n (f(u_{ij}))^m \eta_i. \quad (3)$$

$d(x_j, v_i)$ is the distance measure from the data point x_j to prototype v_i . A clustering algorithm with a special version of (3) as the objective function, called *possibilistic c-means* (PCM) algorithm, was proposed by Krishnapuram and Keller [16], where the standard fuzzy complement, $f(u_{ij}) = 1 - u_{ij}$ was used.

To minimize (3), various methods such as the genetic approach, the simulated annealing, and the alternating optimization can be used [23]. We choose the alternating optimization method for its popularity. The alternating optimization method is based on the necessary conditions of minimization. We select a complement of the form $f(u_{ij}) = 1 + u_{ij} * \log(u_{ij}) - u_{ij}$ because its derivatives can be simply represented. It is easy to prove that this setting of $f(u_{ij})$ satisfies the axioms of fuzzy complements, boundary conditions and monotonicity [13]. To simplify the following discussion, we set $m = 1$. Thus, the proposed fuzzy objective function becomes

$$FE(V, X) = \sum_{i=1}^c \sum_{j=1}^n u_{ij} d^2(x_j, v_i) + \sum_{i=1}^c \sum_{j=1}^n (1 + u_{ij} * \log(u_{ij}) - u_{ij}) \eta_i. \quad (4)$$

In (4), the distance is usually set as the Euclidean distance, that is $d(x_j, v_i) = \|x_j - v_i\|$. First, we compute the gradient of $FE(\cdot, \cdot)$ with respect to u_{ij} . By setting $\partial FE(V, X) / \partial u_{ij} = 0$, we get

$$u_{ij} = \exp\left(-\frac{\|x_j - v_i\|^2}{\eta_i}\right). \quad (5)$$

Substituting this membership back and after simplification, we get

$$FE(V, X) = \sum_{i=1}^c \eta_i - \sum_{i=1}^c \sum_{j=1}^n \exp\left(-\frac{\|x_j - v_i\|^2}{\eta_i}\right) \eta_i. \quad (6)$$

The above reformulation is based on the method proposed by Hathaway and Bezdek [10]. Following the multidimensional chain rule, when x_j is the current input data the gradient of $FE(\cdot, \cdot)$ with respect to v_i is

$$\frac{\partial FE(V, X)}{\partial v_i} = \left(\frac{\partial FE}{\partial \|x_j - v_i\|^2}\right) \left(\frac{\partial \|x_j - v_i\|^2}{\partial v_i}\right) \quad (7)$$

$$= u_{ij} \left(\frac{\partial \|x_j - v_i\|^2}{\partial v_i}\right) \quad (8)$$

$$= -2u_{ij}(x_j - v_i). \quad (9)$$

In (5), η_i plays the role of a normalization parameter for the distance $\|x_j - v_i\|^2$ for the membership u_{ij} . In Fig. 2, we show the membership relative to $\eta_i = 1, 2, \dots, 10$, respectively. Note that $\exp(-\|x_j - v_i\|^2/1) < \exp(-\|x_j - v_i\|^2/2) < \dots < \exp(-\|x_j - v_i\|^2/10)$.

Fuzzy Robust Clustering (FRC) algorithm:

Input: the training feature vector set $X = \{x_1, x_2, \dots, x_n\}$ and the number, c , of clusters.

Output: the final prototypes of clusters $V = \{v_1, v_2, \dots, v_c\}$.

Procedure:

Step 1: Initially set the iteration count $t = 1$, iteration bound T , learning coefficient $\alpha_0 \in (0, 1]$.

Step 2: Set the initial prototype set $V = \{v_1, v_2, \dots, v_c\}$ with a strategy.

Fig. 2. Plot of the membership generated with different η .

Step 3: Compute $\alpha_t = \alpha_0(1 - t/T)$ and adjust η_i with a strategy.

Step 4: Sequentially take every sample x_j from X and update each prototype with $v_i^{\text{new}} = v_i^{\text{old}} + \alpha_t(x_j - v_i^{\text{old}})(\exp(-\|x_j - v_i\|^2/\eta_i))$.

Step 5: Add 1 to t and repeat steps 3–5, until t is equal to T .

η_i plays an important role in (5) and determines the mobility of the corresponding prototype. There are many strategies for the adjusting of η_i . We propose two strategies. One is initializing η_i with the result of another clustering algorithm. The other is initializing the prototypes at different places with larger distances between each other and setting a smaller $\alpha_0 \in (0, 1]$ and a larger T . In step 5, η_i is adjusted by the rule: $\eta_i = \min_j \{\|v_j - v_i\|^2\}$, $j \neq i$. The concept is to set η_i with the consideration of minimizing influences on other prototypes. This is often referred as the alternation optimization approach [23]. It is hard to find the membership function that is the critical point of the following objective function:

$$FE(V, X) = \sum_{i=1}^c \sum_{j=1}^n u_{ij}^m d^2(x_j, v_i) + \sum_{i=1}^c \sum_{j=1}^n (1 + u_{ij} * \log(u_{ij}) - u_{ij}) \eta_i. \quad (10)$$

Alternatively, we could seek other optimization approaches. The proposed membership (5) can also be derived from the M estimator approach in the field of robust statistics [7,9,11]. The M estimator approach tries to minimize the function

$$ME(v) = \sum_{i=1}^n \theta(r_i), \quad (11)$$

where $r_i = x_i - v$ is the residual and $\theta(\cdot)$ is a symmetric positive-definite function. Various functions for $\theta(\cdot)$ can be selected to reduce the influence of large residuals. We propose $\theta(r_i) = \exp(-r_i^2)$. The necessary condition for the minimum of (11) is $\partial ME(v)/\partial v = 0$. Thus,

$$\sum_{i=1}^n (x_i - v) \exp(-r_i^2) = 0. \quad (12)$$

We get

$$v = \frac{\sum_{i=1}^n \exp(-r_i^2) x_i}{\sum_{i=1}^n \exp(-r_i^2)}. \quad (13)$$

Eq. (13) shows that v is a weighted mean of x . The only difference between the weighting function and the proposed fuzzy membership function is just the normalization. Thus FRC algorithm can be viewed as c M estimators work independently and simultaneously.

Here, we propose the corresponding fuzzy objective function,

$$FF(V, X) = \sum_{i=1}^c \sum_{j=1}^n u_{ij} d^p(x_j, v_i) + \sum_{i=1}^c \sum_{j=1}^n (1 + u_{ij} * \log(u_{ij}) - u_{ij}) \eta_i^p. \quad (14)$$

Following the alternation optimization approach, we compute the gradient of $FF(\cdot, \cdot)$ with respect to u_{ij} . By setting $\partial FF(V, X) / \partial u_{ij} = 0$, we get

$$u_{ij} = \exp \left(- \left(\frac{\|x_j - v_i\|^2}{\eta_i} \right)^p \right). \quad (15)$$

Note that the membership function (5) is a special case of (15) that is used by Runkler and Bezdek without indicating that it is derived by an objective function [11]. The gradient of $FF(\cdot, \cdot)$ with respect to v_i is

$$\frac{\partial FF(V, X)}{\partial v_i} = -2u_{ij} p d^{(p-1)}(x_j, v_i)(x_j - v_i). \quad (16)$$

To simplify the representation of gradient (16), we can remove the product term $p d^{(p-1)}(x_j, v_i)$. Since the term is always positive, the removal does not change the search process of v_i . We call the corresponding algorithm FRC2, which modifies the FRC algorithm in the computation of the membership function in step 4.

Step 4. Sequentially take every sample x_j from X and update each prototype with $v_i^{\text{new}} = v_i^{\text{old}} + \alpha_t(x_j - v_i^{\text{old}})(\exp(-(\|x_j - v_i\|^2/\eta_i)^p))$.

It may be interesting to compare the proposed function (15) and the membership function proposed by Krishnapuram et al. [17]

$$u_{ij} = \frac{1}{1 + (\|x_j - v_i\|^2/\eta_i)^p}. \quad (17)$$

The membership values of these two functions relative to the normalized distance $\|x_j - v_i\|^2/\eta_i$ are shown in Fig. 3. p is set from 1 to 10, respectively. Properly used, the exponential parameter p can produce various membership functions. The two functions have the similar form because their objective functions share the similar design logic.

Runkler and Bezdek [23] proposed a fuzzy clustering scheme called alternating cluster estimation with the above two membership functions as the membership function tool bars.

Fig. 3. Plots of the membership generated with different p . The left plot is the proposed membership function and the right plot is proposed by Krishnapuram et al.

4. Fuzzy robust c spherical shells algorithm

The fuzzy c spherical shells algorithm is designed for the searching of the clusters belonging to the spherical shells. Krishnapuram et al. [17] derived the *fuzzy c spherical shells* (FCSS) algorithm with the membership of fuzzy c -means (2). We define the prototype $v_i = (o_i, r_i)$ and set $d^2(x_j, v_i) = (\|x_j - o_i\|^2 - r_i^2)^2$, where o_i is the center of the i th hypersphere and r_i is the radius. Following (5), the membership weighted loss function is

$$FE(V, X) = \sum_{i=1}^c \sum_{j=1}^n \exp\left(-\frac{(\|x_j - o_i\|^2 - r_i^2)^2}{\eta_i}\right) (\|x_j - o_i\|^2 - r_i^2)^2. \quad (18)$$

We rewrite $d^2(x_j, v_i) = p_i^t M_j p_i + y_j^t p_i + b_j$ where

$$b_j = (x_j^t x_j)^2, \quad y_j = 2(x_j^t x_j) \begin{pmatrix} x_j \\ 1 \end{pmatrix},$$

$$p_i = \begin{pmatrix} -2o_i \\ o_i^t o_i - r_i^2 \end{pmatrix} \quad \text{and} \quad M_j = \begin{pmatrix} x_j \\ 1 \end{pmatrix} \begin{pmatrix} x_j \\ 1 \end{pmatrix}^t.$$

It has been shown in [17] that the updating of p_i in the membership weighted form is

$$p_i = -(\frac{1}{2})(H_i)^{-1} w_i, \quad (19)$$

where

$$H_i = \sum_{j=1}^n u_{ij} M_j \quad (20)$$

and

$$w_i = \sum_{j=1}^n u_{ij} y_j. \quad (21)$$

Fuzzy Robust C Spherical Shells (FRCSS) algorithm:

Procedure:

Step 1: Initially set the iteration count $t = 1$, iteration bound T .

Step 2: Set the initial prototype set $V = \{v_1, v_2, \dots, v_c\}$ with a strategy.

Step 3: Compute H_i and w_i for each cluster with (20) and (21). Compute p_i for each cluster with (19).

Step 4: Compute u_{ij} with (5).

Step 5: Add 1 to t and repeat steps 3–5, until t is equal to T .

5. Prototype under-utilization problem of LVQ and hard robust clustering algorithms

Learning vector quantization (LVQ) is a neural network based method to find a good set of prototypes to store as the reference set for representing the cluster structure of the input data. Let $X = \{x_1, x_2, \dots, x_n\} \subset R^m$ denote the input sample set and c denote the number of nodes in the output layer. The objective of LVQ is to minimize the following function:

$$E(V, X) = \sum_{i=1}^c \sum_{j=1}^n w_{ij} \|x_j - v_i\|^2, \quad (22)$$

where $w_{ij} = 1$ if v_i is the winner else $w_{ij} = 0$. Kohonen proposed the following algorithm [15].

LVQ algorithm:

Input: the training feature vector set $X = \{x_1, x_2, \dots, x_n\}$ and the number, c , of clusters.

Output: the final prototypes of clusters $V = \{v_1, v_2, \dots, v_c\}$.

Procedure:

Step 1: Initially set the iteration count $t = 1$, iteration bound T , learning coefficient $\alpha_0 \in (0, 1]$.

Step 2: Set the initial prototype set $V = \{v_1, v_2, \dots, v_c\}$ with some strategy.

Step 3: Compute $\alpha_t = \alpha_0(1 - t/T)$.

Step 4: Randomly take a sample x_k from X .

Step 5: Find the winning neuron v_i , such that $\|x_k - v_i\| = \min_{1 \leq j \leq c} \{\|x_k - v_j\|\}$.

Step 6: Update the winner: $v_i^{\text{new}} = v_i^{\text{old}} + \alpha_t * (x_k - v_i^{\text{old}})$.

Step 7: Add 1 to t and repeat steps 3–6, until t is equal to T .

It was found by Grossberg [8], Rumelhart and Zipser [22] that traditional hard clustering algorithms like the sequential hard c -means (SHCM) and learning vector quantization (LVQ) suffer from a severe initialization problem. If the initial values of the prototypes are not in the convex hull formed by the input data, the clustering algorithms may not produce meaningful results. This is called the prototype under-utilization or the dead units problem since some prototypes, called dead units,

Fig. 4. An initialization problem of LVQ and SHCM.

may never win the competition. The cause of this problem is that these algorithms update only the winning prototype for every input.

We use the following example to show that LVQ suffers from the initialization problem. Suppose that the input data $X = \{x_1, x_2, \dots, x_8\}$. There are two classes $X_1 = \{x_1, x_2, x_3, x_4\}$ and $X_2 = \{x_5, x_6, x_7, x_8\}$ in the data set. $v_{i,j}$ is the prototype v_i in the iteration j . The initial positions of two prototypes $v_{1,0}$ and $v_{2,0}$ are at x_1 and x_2 as shown in Fig. 4. Since $v_{2,0}$ is closer to other six input data than $v_{1,0}$, only $v_{2,0}$ is updated. $v_{1,0}$ never gets a chance to win. This result is not desirable because it is only a local optimum.

A popular approach proposed to solve the initialization problem is frequency sensitive competitive learning (FSCL) [4]. In FSCL, each prototype incorporates a count of the number of times it has been the winner. The distance measure is modified to give prototypes with a lower count value a chance to win the competition.

FSCL algorithm:

Step 1: The same as LVQ except winning count $u_i = 0$.

Steps 2 and 3: The same as LVQ.

Step 4: For $k = 1, 2, \dots, n$, find the winning neuron v_i , such that $u_i * \|x_k - v_i\| = \min_{1 \leq j \leq c} \{u_j * \|x_k - v_j\|\}$. Update the winner and winning count with $v_i^{\text{new}} = v_i^{\text{old}} + \alpha_t(x_k - v_i^{\text{old}})$ and $u_i = u_i + 1$.

Step 5. Add 1 to t and repeat steps 3–5, until t is equal to T .

According to the analysis by Pal et al. [21], the problem has two causes: (1) an improper initialization of prototypes and (2) only one prototype is updated in each input. To solve (1), some researchers suggest to initialize prototypes with random input vectors, reducing the probability of falling into the under-utilization situation but does not eliminate it.

Pal et al. [20] suggest an algorithm called GLVQ that gives the full membership to the winner and the partial membership to the loser. We found the inconsistent situation produced by GLVQ for a certain scaling of input data and proposed a modified algorithm called generalized competitive clustering (GCC) [24]. This situation is also analyzed by Gonzalez et al. [6]. Another modified algorithm called GLVQ-F is designed by Nikhil et al. [21], where a modified function is proposed. Since GLVQ-F uses the membership derived from the FCM, it has the same disadvantages of FCM when the cluster number is large.

Following the concept of GLVQ-type algorithm, we modify the FRC algorithm to give the winner full membership. That is, $\eta_i = \infty$ if the i th prototype is the winner. While the losers use the membership (5) with a finite value of η . We call the proposed algorithm *hard robust clustering* (HRC). Like the situation in FRC, the initial value and the adjustment of η in HRC are very important. In the current version, we let the losers use the same η and reduce it as the training process proceeds. The initial value of η is set as a multiple of the squared variance of the training data. Depending on the distribution of the data and a priori information, η can also be set with other strategies.

Hard Robust Clustering (HRC) algorithm:

Procedure:

Step 1: Initially set the iteration count $t = 1$, iteration bound T , learning coefficient $\alpha_0 \in (0, 1]$.

Step 2: Set the initial prototype set $V = \{v_1, v_2, \dots, v_c\}$ with a strategy.

Step 3: Compute $\alpha_t = \alpha_0(1 - t/T)$ and $\eta = K * \text{var}^2 * \eta(1 - t/T)$. K is a constant and var is the variance of the data set X .

Step 4: For $j = 1, 2, \dots, n$, find the winning neuron v_i such that $\|x_j - v_i\| = \min_{1 \leq k \leq c} \{\|x_j - v_k\|\}$. Update the winner with $v_i^{\text{new}} = v_i^{\text{old}} + \alpha_t(x_j - v_i^{\text{old}})$. Update the others with $v_k^{\text{new}} = v_k^{\text{old}} + \alpha_t(x_j - v_i^{\text{old}})(\exp(-\|x_j - v_i\|^2/\eta_t))$.

Step 5: Add 1 to t and repeat steps 3–5, until t is equal to T .

6. Simulations

6.1. Comparison of LVQ, FSCL and HRC with same number of data in four clusters

Input data: There are four clusters of samples, and each cluster has 100 samples from four Gaussian distributions centered at $(0.1, 0)$, $(0, 0.1)$, $(-0.1, 0)$, $(0, -0.1)$, marked by asterisks.

Initialization: Initial positions of four prototypes are all at $(0.25, 0.25)$. We set $T = 40$, $c = 4$, $K = 3$ and $\alpha_0 = 0.8$.

Results:

LVQ: The final positions of four prototypes trained by LVQ are shown in Fig. 5. Two prototypes stay at initial positions.

HRC and FSCL: All prototypes are used by HRC and FSCL and the final positions are near the actual centroids as shown in Figs. 6 and 7.

Discussions: The results correspond to the designing purpose of the HRC and FSCL. Note that the performance of HRC is not sensitive to the setting of the constant K . A choice of K in [5, 16] produces the similar result.

6.2. Comparison of FSCL and HRC with different number of data in four clusters

This simulation is designed to test the robustness of FSCL and HRC when the number of data in each cluster is different. As shown in Fig. 7, FSCL performs well. But if we change the number of data in clusters to 50, 50, 200, 200, the final centroids trained by FSCL shift along the direction from the sparse cluster to the dense cluster as shown in Fig. 8. This is caused by the winning frequency

Fig. 5. Final positions of LVQ prototypes. Actual centroids: '*'. LVQ prototypes: 'o'.

Fig. 6. Final positions of HRC prototypes. Actual centroids: '*'. HRC prototypes: 'o'.

used in FSCL. HRC does not use the frequency as one of the training coefficients, and therefore the produced prototypes will not be affected by the number of data in clusters as shown in Fig. 9. Note that in the above simulations, the results of FRC depend on the initialization. It performs like HRC if the prototypes are initialized properly or it fails like LVQ.

Fig. 7. FSCl prototypes when the number of data in each cluster is equal.

Fig. 8. FSCl prototypes when the number of data in each cluster is not equal.

6.3. Comparison of FRC and HRC

Input data: There are four clusters of samples, and each cluster has 100 samples from four Gaussian distributions marked by dots. There are 100 outliers marked by “+”.

Fig. 9. HRC prototypes when the number of data in each cluster is not equal.

Fig. 10. Final positions of HRC prototypes when outliers exist. Actual centroids: '*'. HRC prototypes: 'o'.

Initialization: Initial positions of four prototypes are at $(0.2, 0.2)$, $(0.2, -0.2)$, $(-0.2, 0.2)$ and $(-0.2, -0.2)$. We set $T = 40$, $c = 4$, $K = 3$ and $\alpha_0 = 0.2$. The initial value of v_i and η_i of FRC is set with the result of HRC.

Results: As shown in Fig. 10, the final positions of prototypes in HRC are greatly affected by the outliers. The final prototypes of FRC are near the actual centroids as shown in Fig. 11.

Discussions: HRC is designed for solving the prototype under-utilization problem of LVQ, where the winner always owns full membership. Following the progress of the training process, η of the losers reduces and the behavior of HRC is more and more like the LVQ. Thus the final prototypes are affected by the outliers. Since all prototypes in FRC are updated proportionally to the fuzzy

Fig. 11. Final positions of FRC prototypes when outliers exist.

membership, the influence of outliers would be quite limited. FRC reduces the effect from the outliers but its prototype mobility is also restricted. It may be a good idea to initialize FRC with HRC as this simulation suggests. Note that from our simulations, the final prototypes of HRC are near the final prototypes of a properly initialized LVQ. This meets the design goal of HRC.

6.4. Extraction of 2-D circles

Input data: There are three 2-D circles and each circle has 100 boundary points marked by dots. There are 100 outliers marked by “+”.

Initialization: Initial positions of prototypes in FCSS are randomly set and the final prototypes are initial prototypes of FRCSS. We set $T=40$, $c=4$ and $\alpha_0=0.2$.

Results: As shown in Fig. 12, the extracted circles with FCSS are greatly affected by the outliers. The extracted circles with FRCSS are near the actual circles as shown in Fig. 13.

6.5. Extraction of 3-D spherical shells

Input data: There are three 3-D spheres centered at $(0,0,0)$, $(1.1,0,0)$ and $(0.5,0.5,0.5)$. Each sphere has 100 boundary points marked by dots. There are 100 outliers marked by “+”.

Initialization: Initial positions of prototypes in FCSS are randomly set and the final prototypes are used as initial prototypes of FRCSS. We set $T=40$, $c=4$ and $\alpha_0=0.2$.

Results: As shown in Fig. 14, the extracted spherical shells with FCSS are greatly affected by the outliers. The extracted spherical shells with FRCSS are near the actual circles as shown in Fig. 15.

6.6. Comparison of PCM and FRC

Although the PCM and FRC2 use a similar membership function, they still perform differently in some simulations.

Fig. 12. Final positions of FCSS prototypes when outliers exist and the 2-D testing data. “+” represents the outlier.

Fig. 13. Final positions of FRCSS prototypes when outliers exist and the 2-D testing data. “+” represents the outlier.

Input data: There are two clusters of samples. The left cluster has 100 and the right cluster has 300 samples from two Gaussian distributions marked by dots.

Initialization: Initial positions of two prototypes are at $(0,0)$. We set $T=40$, $p=2$, $c=2$ and $\alpha_0=0.2$.

Fig. 14. Final positions of FCSS prototypes when outliers exist and the projection of the 3-D testing data on the x - y and x - z plane. “+” represents the outlier.

Results: As shown in Fig. 16, the resultant prototypes in FRC are quite near the actual centroids, while the left prototype derived by the PCM is affected by the right cluster as shown in Fig. 17.

Discussions: From the memberships shown in Fig. 3, when the normalized distance is greater than 1, PCM membership is usually greater than FRC membership. Thus the left prototype of PCM is affected by the right cluster and the greater bias appears.

Fig. 15. Final positions of FRCSS prototypes when outliers exist and the projection of the 3-D testing data on the x - y and x - z plane.

7. Conclusions

Through the definition of outliers as the fuzzy complement, we have proposed a family of robust clustering objective functions without the assumption of the probabilistic constraint. We have carried

Fig. 16. Final positions of FRC prototypes when two overlapped clusters exist.

Fig. 17. Final positions of PCM prototypes when two overlapped clusters exist.

out simulations for the proposed methods and the existing algorithms, showing that the proposed method performs much better than the existing approaches. The corresponding fuzzy c spherical shells algorithms are derived by a redefinition of the distance metric.

As compared with other clustering algorithms, FRC and FRCSS reduce the influence from the outliers. We also modify FRC to be a hard robust clustering algorithm for solving the prototype under-utilization problem. As supported by the experiments, the proposed algorithms not only perform well under the assumed model but also produce a satisfactory result under the noisy environment.

References

- [1] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum, New York, 1981.
- [2] J.C. Bezdek, S.K. Pal (Eds.), *Fuzzy Models for Pattern Recognition*, IEEE Press, New York, 1992.

- [3] J. Buhmann, H. Kuhnel, Complexity optimized data clustering by competitive neural networks, *Neural Comput.* 5 (1993) 75–88.
- [4] D. Desieno, Adding a conscience to competitive learning, in: *Proc. IEEE Internat. Conf. on Neural Networks*, vol. I, 1988, pp. 117–124.
- [5] P. Devijver, J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [6] A.I. Gonzalez, M. Grana, A. D’Anjou, An analysis of the GLVQ algorithm, *IEEE Trans. Neural Networks* 6 (1995) 1012–1016.
- [7] C. Goodall, *M*-estimator of location: an outline of the theory, in: D.C. Hoaglin, F. Mosteller, J.W. Tukey (Eds.), *Understanding Robust and Exploratory Data Analysis*, Wiley, New York, 1983, pp. 339–403.
- [8] S. Grossberg, Competitive learning: from interactive activations to adaptive resonance, *Cognitive Sci.* 11 (1987) 23–63.
- [9] F.R. Hampel, E.M. Ponchotti, P.J. Rousseeuw, W.A. Stahel, *Robust Statistics: The Approach Based on Influence Functions*, Wiley, New York, 1986.
- [10] R.J. Hathaway, J.C. Bezdek, Optimization of clustering criteria by reformulation, *IEEE Trans. Fuzzy System* 3 (2) (1995) 241–245.
- [11] P.J. Huber, *Robust Statistics*, Wiley, New York, 1981.
- [12] A.K. Jain, R.C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- [13] G.J. Klir, B. Yuan, *Fuzzy Sets and Fuzzy Logic*, Prentice-Hall Inc., Englewood Cliffs, NJ, 1995, pp. 51–61.
- [14] T. Kohonen, The self-organizing map, *Proc. IEEE* 78 (1990) 1464–1480.
- [15] T. Kohonen, *Self-organization and Associative Memory*, 3rd Edition, Springer, Berlin, Germany, 1989.
- [16] R. Krishnapuram, J.M. Keller, A possibilistic approach to clustering, *IEEE Trans. Fuzzy Systems* 1 (1993) 98–110.
- [17] R. Krishnapuram, O. Nasraoui, H. Frigui, The fuzzy *c* spherical shells algorithm: a new approach, *IEEE Trans. Neural Networks* 3 (5) (1992) 663–671.
- [18] G.J. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*, Wiley, New York, 1992.
- [19] K. Nadler, E.P. Smith, *Pattern Recognition Engineering*, Wiley, New York, 1993.
- [20] N.R. Pal, J.C. Bezdek, E.C. Tsao, Generalized clustering networks and Kononen’s self-organizing scheme, *IEEE Trans. Neural Networks* 4 (4) (1993) 549–557.
- [21] N.R. Pal, J.C. Bezdek, E.C. Tsao, Repair to GLVQ: a new family of competitive learning schemes, *IEEE Trans. Neural Networks* 7 (1996) 1062–1070.
- [22] D. Rumelhart, D. Zipser, Feature discovery by competitive learning, *Cognitive Sci.* 9 (1985) 75–112.
- [23] T.A. Runkler, J.C. Bezdek, Alternating cluster estimation: a new tool for clustering and function approximation, *IEEE Trans. Fuzzy Systems* 7 (4) (1999) 377–393.
- [24] T.N. Yang, S.D. Wang, A generalized competitive clustering algorithm, in: *Internat. Symp. on Artificial Neural Networks*, 1994, pp. 365–373.