Knowl Inf Syst
DOI 10.1007/s10115-011-0410-8

REGULAR PAPER

Improvement of neural network classifier using floating centroids

Lin Wang · Bo Yang · Yuehui Chen · Ajith Abraham · Hongwei Sun · Zhenxiang Chen · Haiyang Wang

Received: 22 April 2010 / Revised: 25 February 2011 / Accepted: 2 May 2011

Abstract This paper presents a novel technique—Floating Centroids Method (FCM) designed to improve the performance of a conventional neural network classifier. Partition space is a space that is used to categorize data sample after sample is mapped by neural network. In the partition space, the centroid is a point, which denotes the center of a class. In a conventional neural network classifier, position of centroids and the relationship between centroids and classes are set manually. In addition, number of centroids is fixed with reference to the number of classes. The proposed approach introduces many floating centroids, which are spread throughout the partition space and obtained by using K-Means algorithm. Moreover, different classes labels are attached to these centroids automatically. A sample is predicted as a certain class if the closest centroid of its corresponding mapped point is labeled by this class. Experimental results illustrate that the proposed method has favorable performance especially with respect to the training accuracy, generalization accuracy, and average F-measures.

L. Wang · B. Yang (✉) · Y. Chen · Z. Chen
Shandong Provincial Key Laboratory of Network based Intelligent Computing,
University of Jinan, 250022 Jinan, China
e-mail: yangbo@ujn.edu.cn

L. Wang · H. Wang
School of Computer Science and Technology, Shandong University,
250101 Jinan, China
e-mail: wangplanet@gmail.com

A. Abraham
Faculty of Electrical Engineering and Computer Science,
VSB-Technical University of Ostrava, Ostrava, Czech Republic

A. Abraham
Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation
and Research Excellence, Washington 98071, USA

H. Sun
School of Science, University of Jinan, 250022 Jinan, China

Published online: 28 May 2011

Keywords Classification · Neural networks · Floating Centroids Method

1 Introduction

Classification is an important research area in pattern recognition and data mining. In a supervised classification task, a classification model is usually constructed according to a given training set. Once the model has been built, it can map a test data to a certain class in the given class set. Many classification techniques including decision trees [1,2], neural networks [3–6], support vector machines [7–10], rule-based systems, etc., have been proposed. Among these techniques, neural network classifier, which is supervised, has been proven to be a practical approach with lots of success stories in several classification tasks [11–15].

In neural network classifier, partition space is a space that is used to categorize data sample after sample is mapped by neural network. Centroid is a point in partition space and denotes the center of a class. In a conventional neural network classifier, position of centroids and the relationship between centroids and classes are set manually. In addition, number of centroids is fixed with reference to the number of classes. This fixed-centroid constraint decreases the chance to find optimal neural network. Therefore, Floating Centroids Method (FCM) is proposed to remove the fixed-centroid constraint by introducing many floating centroids, which are spread throughout the partition space and obtained by using K-Means algorithm. Furthermore, general mathematical model and detailed proof to the coloring method are also given in this paper.

Rest of the paper is arranged as follows. Section 2 reviews some important neural network classification methods and their problems. Section 3 gives a detailed account of the FCM approach. Section 4 outlines and discusses our experiments followed by conclusions in Sect. 5.

2 Related works

It is relatively straightforward to understand how a neural network works for binary classification. A neural network for binary classification only has one output that estimates the probability that a sample belongs to the "true" class.

However, in some classification tasks with more than two mutually exclusive classes, neural network configurations and calculations are not so simple. A straightforward approach is to use multiple output units, one for each class (one-per-class). Each output unit is viewed as computing the probability that a sample belongs to its corresponding class. The class whose neural network output unit gives the highest probability estimate is chosen as the predicted class.

Since the classification categories are mutually exclusive, each sample can only be assigned to one of the classes, which means that the sum of these individual probabilities should always be equal to one. However, it is unlikely that the sum of outputs equals one if each output is the estimated probability for that class. Bridle [16] normalized the outputs of neural network using softmax activation function (SoftMax) to solve this problem. The sum of adjusted probabilities is always one.

If soft outputs are not required, Dietterich and Bakiri [17] proposed the error-correcting output code (ECOC) method to provide additional statistical robustness properties. In this method, each neural network output unit can be viewed as computing probability that its corresponding bit in the codeword is 1. Define these probability values $B = (b_1, b_2, \ldots, b_n)$ as a probability vector, where n is the length of codeword in the error-correcting code. To classify a sample, the L^1 distance between this probability vector B and each of the codewords W_i $(i = 1, \ldots, k)$ is computed. The class whose codeword has the smallest L^1 distance

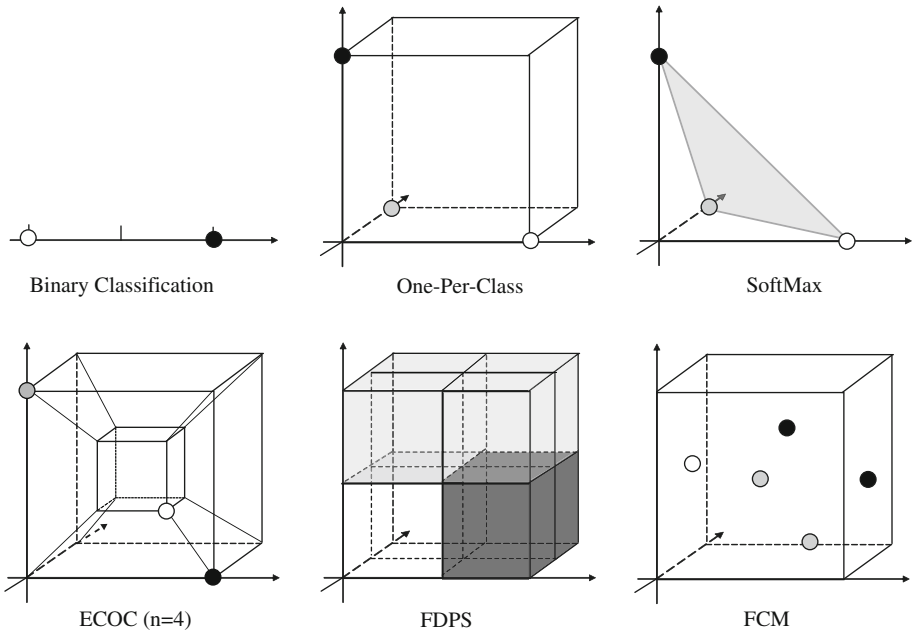


Fig. 1 Different types of partition space for binary classification (*top-left*) and 3-class classification (the others)

between B and W_i is chosen as the predicted class. Although some binary predictors may make errors in actual cases, if the number of errors is not too large, an appropriate codewords table can correct the errors and restore the correct multi-class label.

These methods also can be interpreted from their equivalent geometric meaning. First, we defined some important concepts.

Definition 2.1 In the case of a classification problem, neural network is a mapping relationship $\mathscr{F}$ maps sample from data space to a partition space.

Definition 2.2 The partition space $\mathscr{P}$ is used to categorize samples, according to the output value of $\mathscr{F}$.

Definition 2.3 The centroid $\mathbb{c}$ is a point, which denotes a center of class in partition space $\mathscr{P}$.

Definition 2.4 A partition $\mathbb{p}$ is a region, which is controlled by $\mathbb{c}$. Each point that is situated in $\mathbb{p}$ is linked to $\mathbb{c}$.

Definition 2.5 The classifier $\mathscr{D}$ is a pair ($\mathscr{F}$, $\mathscr{P}$).

In the geometric interpretation to binary classification (top left of Fig. 1), partition space is a line segment with two fixed centroids (0 and 1 using sigmoid activation function, -1 and 1 using hyperbolic-tangent activation function). Sample will be predicted as "true" if the mapped point in this line segment is more closer to 1 than to 0 or more closer to 1 than to -1.

For multi-classification one-per-class method, each neural network output unit is viewed as coordinate component of a point in partition space (top middle of Fig. 1). Each class label is attached to a centroid with corresponding coordinate component set to one while the other

components are set to zero. The class whose centroid is closet to the point is chosen as the predicted class. For example, as far as the 3-class classification problems are concerned, (1, 0, 0), (0, 1, 0), and (0, 0, 1) are centroids of class 1, 2, and 3, respectively. If the outputs of neural network denote a point (0.2, 0.3, 0.1), class 2 whose centroid (0, 1, 0) is closet to this point is chosen as the predicted class.

For SoftMax method, a sample will be mapped to the hyper plane $\{x_1 + x_2 \cdots + x_m = 1, 0 \leqslant x_i \leqslant 1, 1 \leqslant i \leqslant m\}$ (top right of Fig. 1), where m is the dimension of partition space. Intersection points between this hyper plane and axes are centroids. The class whose centroid is closest to the corresponding mapped point in the hyper plane is chosen as the predicted class.

For ECOC, dimension of partition space is expanded and not limited to the number of classes (bottom left of Fig. 1). This method adds many redundant dimensions to correct errors. Each codeword corresponds to a centroid in the high dimensional space. Selecting L^1 as the distance measure, the class whose centroid is closest to the corresponding mapped point in partition space is chosen as the predicted class.

In the above-mentioned geometric interpretations, all of these methods set the position of centroids and the relationship between centroids and classes manually. In addition, number of centroids is fixed to the number of classes. However, as far as geometric meaning is concerned, the classifier optimization object is to find a neural network, which allocates points in the same class together as close as possible and keep points in the different class away from the others as far as possible. Although these methods have been proven to be very effective, mapping samples to the position-fixed and number-fixed centroids limits the scope of neural network selection during optimization. A neural network can be viewed as "good" as long as there is clear decision boundary between centroids of different classes in the partition space.

In our previous research, we provided a preliminary attempt to break the conventional partition space and described a tentative method called further division of partition space (FDPS, bottom middle of Fig. 1) [18] in which the partition space is divided into many more shape-fixed partitions (total number of partitions > total number of classes) and is expanded to higher dimensional space. Class labels are attached to these partitions based on the distribution of mapped points during optimization process. The class whose partitions include the corresponding mapped point in partition space is chosen as the predicted class. There is no concept of centroid in this method. Partition exercise centroid's effect independently. In FDPS, optimization task is not to find a neural network, which makes points get close to some predefined centroid, but to find a neural network which concentrates points of same class into the same partition. Since the number and position of partitions, which is controlled by a class, are generated based on distribution of mapped points, the neural network is optimized relatively easily.

However, the shape-fixed partition division method used in FDPS will break the internal decision boundary between classes in partition space. Shape-fixed partition cannot distinguish the boundary clearly. When the boundary crosses through a partition, this partition can only categorize one class, and against the other classes. This leads to poor generalization ability. Hence, we propose a robust approach named Floating Centroids Method (FCM, bottom right of Fig. 1) in this paper.

3 Floating Centroids Method

In FCM, there is no manually divided partition. Some *floating centroids*, each of which has a class label, are spread throughout the partition space. A sample will be predicted as a certain

class if the closest centroid of its corresponding mapped point is labeled by this class. The irregular regions controlled by centroids are considered as partitions. To get these floating centroids during optimization, training samples are mapped to partition space by neural network first. Then, the corresponding mapped points in partition space will be partitioned into several number of disjoint subsets using the K-Means clustering algorithm [19]. The centroids are defined as the cluster centers of these subsets. Total number of centroids can be more than the number of classes. Finally, each of these centroids will be labeled by a class. Labeling principle is that if training points of one class are majority among all the points which belong to a centroid, this class will label the centroid. A class can label more than one centroid.

Since there is no constraint in the number of centroids and position and class label of centroids are computed by the actual distribution of mapped points, there is natural boundary between centroids. This method avoids boundary problem in FDPS. In fact, FCM fully performs the above-mentioned optimization objective of neural network classifier. Its main objective is to find a neural network, which allocates points in the same class together as close as possible and keep points in the different class away from the others as far as possible. Therefore, compared with one-per-class, SoftMax and ECOC methods, floating (involve number and position) centroids in FCM remove the fixed-centroid constraint and increase the chance to find optimal neural network. Specially, FCM is still effective in binary classification. Partition space is usually a line segment with two fixed centroids. In this case, floating centroids can also be computed according to points distribution in FCM.

At first, we interpret some concepts, including mapping relationship, partition space, centroids, partitions, and the model of classifier. We then illustrate the procedure for centroid generation. Coloring methods and corresponding proofs are described in this subsection. Finally, we introduce the learning process: optimization target function, learning process, and the user-defined parameters.

3.1 Concept interpretation

3.1.1 Mapping relationship

Mapping relationship $\mathscr{F}$ is to transform the training data set from original data space to partition space. A sample with n features is defined as an n-dimensional input vector $\vec{I}$ and the ith element of $\vec{I}$ is defined as I_i. For $\mathscr{F}$, in FCM, the dimension of input is defined as n and dimension of output is defined as m. Therefore, $\mathscr{F}$ is a mapping relationship from set with n elements to set with m elements. The mapping formula of $\mathscr{F}$ is

$$\vec{O} = \mathscr{F}\left(\vec{I}\right) \tag{1}$$

Since feedforward neural networks have the ability to approximate any nonlinear functions, it is very suitable for them to play the role of $\mathscr{F}$. The number of neurons in input layer is set to n, and the number in output layer is set to m. Therefore, one feedforward neural network is enough to approximate $\mathscr{F}$. Figure 2 illustrates the specific form of $\mathscr{F}$: a feedforward neural network with n input neurons and m output neurons.

3.1.2 Partition space, centroid, and partition

The partition space $\mathscr{P}$ is used to categorize samples, according to the output value of $\mathscr{F}$. Dimension of partition space equals to the dimension of output in $\mathscr{F}$, which is defined as m.

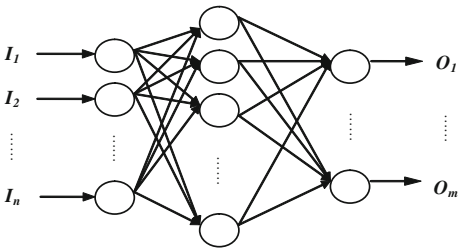


Fig. 2 A feedforward neural network with n input neurons and m output neurons. This neural network can play the role of $\mathscr{F}$

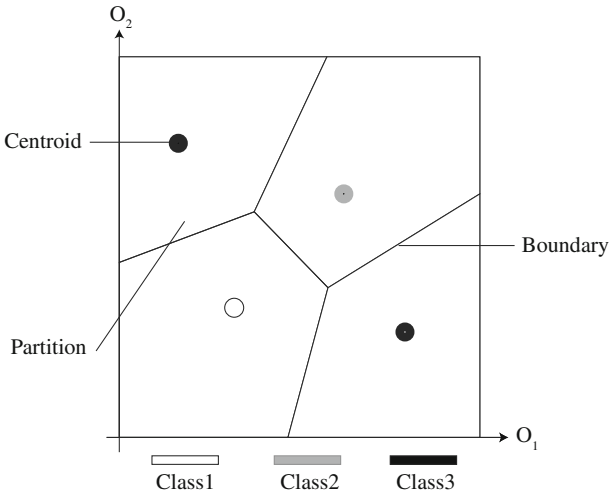


Fig. 3 A colored partition space of 3-class classification. The dimension of partition space is 2 and the number of centroids is 4

Definition 3.1 Each class is assigned a corresponding exclusive label, which is called $color$. Actually, $color_i$ is an abstract symbol to represent class i.

Centroids $\mathbb{c}$ are some key points in partition space with the purpose to classify mapped points. Each centroid has a color. They are spread throughout the partition space. The number of centroids is defined as k. The partition space that has colored centroids is called *colored partition space*.

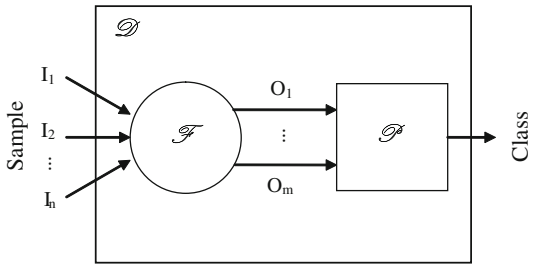
Definition 3.2 A point belongs to centroid $\mathbb{c}$ means $\mathbb{c}$ is the closest centroid to this point.

These centroids are tantamount to dividing the partition space into many irregular regions, which are called partitions $\mathbb{p}$. The point that is situated in a partition belongs to the centroid of this partition. The borders between partitions are classification boundaries. The number of partitions is the same as the number of centroids and equals to k. Figure 3 illustrates these concepts.

3.1.3 Model of FCM neural network classifier

After learning (in Sect. 3.3), an optimized neural network and its corresponding colored partition space are produced. The classifier $\mathscr{D}$ is a pair ($\mathscr{F}$, $\mathscr{P}$), where centroids $\mathbb{c}$ in $\mathscr{P}$ have been colored (the class label). The typical model of classifier $\mathscr{D}$ is depicted in Fig. 4.

Fig. 4 Model of classifier



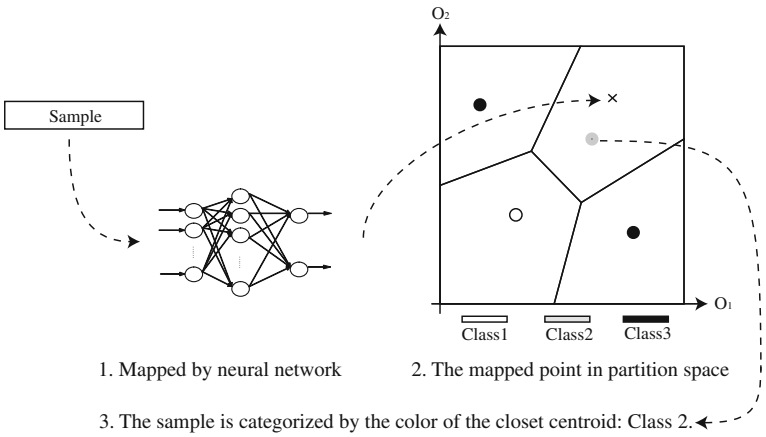


Fig. 5 Categorize a new sample. The dimension of partition space is set to two and the number of classes is three

Euclidian distance is selected as the distance measure. If a new sample need to be categorized, it should be mapped to $\mathscr{P}$ by $\mathscr{F}$ and then, the category will be predicted according to the class of centroid that the corresponding mapped point of this sample belongs to. This process is illustrated in the Fig. 5.

3.2 Centroids generation

In this Subsection, it is narrated how to get a colored partition space, which have many labeled centriods. In order to form a classifier $\mathscr{D}$, the centroids generation process must be performed to get a colored partition space $\mathscr{P}$, which corresponds to a certain $\mathscr{F}$. This process covers the calculation of centroids and assignment of class label for these centroids.

Training samples are mapped to partition space by neural network first. The corresponding mapped points in the partition space are called $color\ points$. Then, K-Means algorithm are used to partition color points in the partition space into a fixed number of k disjoint clusters and the center of each cluster is calculated as centroid.

After the calculation of k centroids, each of them should be labeled by a class. This process is called *coloring*. The coloring principle is that if color points of one class are majority among all the points which belong to a centroid, this class will color the centroid. One or more centroids can be colored by one class.

It is unfair for every class data to color the partition space without using weights or other solutions, because the number of samples between classes are different. A class will color the

centroid at a higher probability, if its samples number is larger than others. In order to describe this problem clearly, some definitions are shown first. Theorem 1 reveals the consequence, on condition that the number of samples in each class is different.

Definition 3.3 Given a training data set S, the number of classes in S is N.

Definition 3.4 S_i is the sample set of class i, $S_i \subseteq S$, $\bigcup_{i=1}^{N} S_i = S$, $\bigcap_{i=1}^{N} S_i = \varnothing$.

Definition 3.5 $S_{\mathbb{c}}$ is the set of samples which corresponding color points dropped into partition of $\mathbb{c}$.

Theorem 1 *P_i is the probability that class i colors centroid $\mathbb{c}$. In the hypothesis of uniform distribution of the outputs generated by the neural network, if $|S_i| \leqslant |S_j|$, then*

$$P_i \leqslant P_j. \tag{2}$$

Proof Let: $n_1 = |S_i|, n_2 = |S_j|$

$$P_i = \sum_{K_1=0}^{n_1} \left\{ C_{n_1}^{K_1} \left(\frac{1}{2}\right)^{n_1} \sum_{K_2=0}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_2} \right\} = \sum_{K_1=0}^{n_1} \sum_{K_2=0}^{K_1} C_{n_1}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2}$$

$$P_j = \sum_{K_2=0}^{n_1} \sum_{K_1=0}^{K_2} C_{n_1}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2} + \sum_{K_2=n_1+1}^{n_2} \sum_{K_1=0}^{n_1} C_{n_1}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2}$$

$$= \sum_{K_2=0}^{n_1} \sum_{K_1=0}^{K_2} C_{n_1}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2} + \sum_{K_2=n_1+1}^{n_2} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_2}$$

Since: $0 \leq K_2 \leq K_1 \leq n_1 \leq n_2$

$$\frac{C_{n_1}^{K_2} C_{n_2}^{K_1}}{C_{n_1}^{K_1} C_{n_2}^{K_2}} = \frac{(n_1 - K_1)!(n_2 - K_2)!}{(n_1 - K_2)!(n_2 - K_1)!} = \frac{(n_2 - K_1 + 1) \cdots (n_2 - K_2)}{(n_1 - K_1 + 1) \cdots (n_1 - K_2)} \geqslant 1$$

$$\therefore P_j \geqslant P_i + \sum_{K_2=n_1+1}^{n_2} C_{n_2}^{K_2} \left(\tfrac{1}{2}\right)^{n_2}$$

□

It can be drawn from Theorem 1 that a class will color a centroid at a higher probability, if the number of samples are larger than others. In order to alleviate the consequence about which different number of samples brings, weights are introduced to ensure the approximation of probabilities. Different weights are given to color points. Then, majority is the class which total weights of color points of this class are heaviest in the partition of $\mathbb{c}$. $\mathbb{c}$ will be colored by the color of majority. The weight defined in Definition 3.6 ensures the approximation of probability, which the color points that belongs to different class drop into the same partition.

Definition 3.6 The weight of a color point, which belongs to class i is defined as:

$$W_i = \frac{1}{|S_i|} \tag{3}$$

Theorem 2 *P_i^w represents the probability that the total weights of color points of class i are heaviest in one partition. In the hypothesis of uniform distribution of the outputs generated by the neural network, if $|S_i| \leq |S_j|$, then*

$$|P_j^w - P_i^w| \leq P_j - P_i. \tag{4}$$

Proof Let:$n_1 = |S_i|, n_2 = |S_j|$

$$P_i^w = \sum_{K_1=0}^{n_1} \left\{ C_{n_1}^{K_1} \sum_{K_2=0}^{\left[\frac{n_2}{n_1}K_1\right]} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2} \right\} = \sum_{K_1=0}^{n_1} \sum_{K_2=0}^{\left[\frac{n_2}{n_1}K_1\right]} C_{n_1}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2} \geqslant P_i$$

$$P_j^w = \sum_{K_2=0}^{n_2} \left\{ C_{n_2}^{K_2} \sum_{K_1=0}^{\left[\frac{n_1}{n_2}K_2\right]} C_{n_1}^{K_1} \left(\frac{1}{2}\right)^{n_1+n_2} \right\} = \sum_{K_2=0}^{n_2} \sum_{K_1=0}^{\left[\frac{n_1}{n_2}K_2\right]} C_{n_1}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2} \leqslant P_j$$

$$P_j^w = \sum_{K_2=0}^{n_2} \left\{ C_{n_2}^{n_2-K_2} \sum_{K_1=0}^{\left[n_1-\frac{n_1}{n_2}(n_2-K_2)\right]} C_{n_1}^{K_1} \left(\frac{1}{2}\right)^{n_1+n_2} \right\}$$

$$\overset{\text{replace}}{=} \sum_{K_2=0}^{n_2} \left\{ C_{n_2}^{K_2} \sum_{K_1=0}^{\left[n_1-\frac{n_1}{n_2}K_2\right]} C_{n_1}^{K_1} \left(\frac{1}{2}\right)^{n_1+n_2} \right\}$$

$$\geqslant \sum_{K_2=0}^{n_1} \left\{ C_{n_2}^{K_2} \sum_{K_1=0}^{\left[n_1-\frac{n_1}{n_2}K_2\right]} C_{n_1}^{K_1} \left(\frac{1}{2}\right)^{n_1+n_2} \right\}$$

$$= \sum_{K_2=0}^{n_1} \sum_{K_1=0}^{\left[n_1-\frac{n_1}{n_2}K_2\right]} C_{n_1}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2}$$

$$P_i = \sum_{K_1=0}^{n_1} \sum_{K_2=0}^{K_1} C_{n_1}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2}$$

$$\overset{\text{exchange}}{=} \sum_{K_2=0}^{n_1} \sum_{K_1=K_2}^{n_1} C_{n_1}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2}$$

$$= \sum_{K_2=0}^{n_1} \sum_{K_1=K_2}^{n_1} C_{n_1}^{n_1-K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2} \overset{\text{replace}}{=} \sum_{K_2=0}^{n_1} \sum_{K_1=K_2}^{n_1-K_2} C_{n_1}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2}$$

$$\because n_1 - \frac{n_1}{n_2}K_2 \geqslant n_1 - K_2 \qquad \therefore \left[n_1 - \frac{n_1}{n_2}K_2\right] \geqslant n_1 - K_2 \qquad \therefore P_j^w \geqslant P_i$$

$$P_j \overset{\text{exchange}}{=} \sum_{K_1=0}^{n_1} \sum_{K_2=K_1}^{n_2} C_{n_1}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2}$$

$$= \sum_{K_1=0}^{n_1} \sum_{K_2=K_1}^{n_2} C_{n_1}^{n_1} C_{n_2}^{n_2-K_2} \left(\frac{1}{2}\right)^{n_1+n_2} \overset{\text{replace}}{=} \sum_{K_1=0}^{n_1} \sum_{K_2=0}^{n_2-K_1} C_{n_1}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2}$$

$$P_i^w = \sum_{K_1=0}^{n_1} \left\{ C_{n_1}^{K_1} \sum_{K_2=0}^{\left[\frac{n_2}{n_1} K_1\right]} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2} \right\}$$

$$= \sum_{K_1=0}^{n_1} \left\{ C_{n_1}^{n_1-K_1} \sum_{K_2=0}^{\left[n_2-\frac{n_2}{n_1}(n_1-K_1)\right]} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2} \right\}$$

$$= \sum_{K_1=0}^{n_1} \left\{ C_{n_1}^{K_1} \sum_{K_2=0}^{\left[n_2-\frac{n_2}{n_1} K_1\right]} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2} \right\} = \sum_{K_1=0}^{n_1} \sum_{K_2=0}^{\left[n_2-\frac{n_2}{n_1} K_1\right]} C_{n_1}^{K_1} C_{n_2}^{K_2} \left(\frac{1}{2}\right)^{n_1+n_2}$$

$\because n_2 - K_1 \geqslant n_2 - \frac{n_2}{n_1} K_1 \geqslant \left[n_2 - \frac{n_2}{n_1} K_1\right] \quad \therefore P_j \geqslant P_i^w$

Thus $|P_j^w - P_i^w| \leqslant P_j - P_i$ □

Therefore, the weight W_i ensures the approximation of probabilities, which the color points that belongs to different class color and the same centroid. To sum up the points which we have just discussed, the centroid generation algorithm is depicted in Algorithms 3.1 and 3.2. Algorithm 3.1 is the main algorithm and Algorithm 3.2 is a sub-algorithm, which is affiliated to algorithm 3.1 to color centroids.

Algorithm 3.1 Centroid generation

Input: A neural network, training data set, and the number of centroids k.
Output: Colored partition space which corresponds to the given neural network.

Step 1: Map samples to partition space using the neural network, and form color points;

Step 2: Use K-Means algorithm to partition color points into a fixed number of k disjoint clusters;

Step 3: The center of each cluster is calculated as centroid;

Step 4: Get the first centroid;

Step 5: **while** *exist centroid which has not been colored* **do**

Step 6: Call algorithm centroid coloring to get the color of majority in the current centroid;

Step 7: Current centroid is colored by majority's color;

Step 8: Get the next centroid;

Step 9: **endwhile**

Step 10: **return** the colored partition space.

Algorithm 3.2 Centroid coloring

Input: Current centroid $\mathbb{c}$ and color points that belongs to $\mathbb{c}$.
Output: Majority's color.

Step 1: $majority := 1$;

Step 2: **for** $i := 1$ **to** $total\ number\ of\ classes$ **do**

Step 3: Let $|S_{\mathbb{c}_i}|$ be the number of color points of class i in $\mathbb{c}$, let $|S_{\mathbb{c}_{majority}}|$ be the number of color points of class $majority$ in $\mathbb{c}$;

Step 4: **if** $|S_{\mathbb{c}_i}| \cdot W_i > |S_{\mathbb{c}_{majority}}| \cdot W_{majority}$ **then** $majority := i$;

Step 5: **endfor**

Step 6: **return** the color of $majority$

3.3 Learning

The question how to obtain an optimized neural network and its corresponding colored partition space is answered in this subsection. In the first part, the optimization target function is introduced. Then, the learning process frame is described in the following part. Finally, every user-defined parameters used in FCM are summarized.

3.3.1 Optimization target function

We first defined a dependent variable Z to evaluate a mapped point in partition space.

$$Z = \begin{cases} \frac{d_{\min}^{\text{self}}}{d_{\min}^{\text{self}} + d_{\min}^{\text{nonself}}} & (\exists \mathbb{c}_{\text{self class}} \wedge \exists \mathbb{c}_{\text{nonself class}}) \\ M & (\text{Else}) \end{cases} \tag{5}$$

For a point in partition space, $d_{\min}^{\text{self}}$ represents the euclidian distance to the closest centroid which is colored by the same class as the mapped point (self class), $d_{\min}^{\text{nonself}}$ represents the euclidian distance to the closest centroid which is colored by a class that is different from that of the mapped point (nonself class) and M is the maximum euclidian distance in m dimensional partition space. In the limit case, if a sample is just mapped to the position of self class centroid, Z equals to 0. Conversely, if it is mapped to the position of nonself class centroid, Z equals to 1. Specially, Z equals to 0.5 if the point is equidistant from the closet self centroid and closet nonself centroid. That is to say, it is situated in the boundary. On this

basis of Z, optimization target function is defined as follows

$$E = \sum_{l=1}^{|S|} \frac{1}{1 + e^{a(1-2Z_l)}} \tag{6}$$

where Z_l is Z value of point of the lth sample, $|S|$ is the total number of samples in the training data set S and a is a real constant which determines tortuosity. The smaller E is, the better $\mathscr{F}$ and $\mathscr{P}$ is. Target of this function is to put points in the same class together as close as possible and keep points in the different class away from the others as far as possible. The introduce of sigmoid type function is to give priority to searching neural networks which have clear decision boundaries. Because the maximum slope is reached in $Z_l = 0.5$, if a point is situated near the boundary, point's or centroid's (boundary's) small movement will cause greater variation of E, which leads to neural network keeps points away from boundary as a priority. If tortuosity is too small or too large, the sigmoid type function will degenerate into linear function or jump function, and lose its differential ability. In practice, it is sufficient to set a to 6 to ensure suitable tortuosity.

3.3.2 Common frame of learning process

The optimizer used in the learning process is defined as $\mathscr{T}$. The goal of learning process is to get the best neural network and its corresponding colored partition space. Common frame of learning process is shown in Algorithm 3.3.

Algorithm 3.3 Common frame of learning process

Input: User-defined parameters and training data set.
Output: Best neural network and its corresponding colored partition space.

Step 1: Code neural network to form individual according to the requirements in $\mathscr{T}$;

Step 2: **while** *maximum generation has not been reached* **do**;

Step 3: **for** id:=1 **to** the number of individuals **do**

Step 4: Decode individual id to form a neural network;

Step 5: Perform centroid generation with this neural network to get colored partition space;

Step 6: Use this neural network and its corresponding colored partition space to compute the value of optimization target function as the fitness of individual id;

Step 7: **endfor**

Step 8: Perform one step iteration of $\mathscr{T}$, update individuals by their fitness;

Step 9: **endwhile**

Step 10: **return** the best neural network and its corresponding colored partition space.

3.3.3 Summary of user-defined parameters

In addition to user-defined parameters which used in optimizer $\mathscr{T}$, the following above-mentioned parameters or methods have to be set by the user before optimization:

m ($m \in \mathbb{Z}^+,\ m \neq 0$), the dimension of the feedforward neural network output. Furthermore, m is also the dimension of partition space. Larger m add many redundant dimensions to improve the generalization ability. Although distance relationship between point and centroid may make errors in some axes, if the number of errors is not too large, the total distance in m-dimensional space can correct the errors to restore correct distance relationship. However, if m is set too large, there will be too many output units, which increase neural network complexity and reduce optimization speed. Experiments suggest selecting m from the interval $[2N, 3N]$ is enough to ensure dimensions redundant (results shown in next section).

k ($k \in \mathbb{Z}^+,\ k \geqslant N$), the number of centroids in the partition space. Larger k increases the chance to find optimal neural network during learning process, because it means there are several classes, each of which colors more than one centroid, and point in partition space has more choices to get close to its own centroids. However, if k is set too large, the partition space will be over-partitioned which leads to the decline of accuracy. Moreover, increase in k will also lead to the time consumption of centroids generation increases dramatically. Experiments suggest selecting k from the interval $[1.5N, 4.5N]$ to ensure several classes have coupled centroids (results shown in next section).

4 Experiments and results

In order to evaluate the performance of this approach, three criterions are defined.

Training Accuracy (TA), a method to adapt training data to achieve better TA.

$$\mathrm{TA} = \frac{\text{number of correctly classified samples in training data set}}{\text{number of total samples in training data set}} \times 100 \tag{7}$$

Generalization Accuracy (GA), a method to obtain better generalization capability.

$$\mathrm{GA} = \frac{\text{number of correctly classified samples in test data set}}{\text{number of total samples in test data set}} \times 100 \tag{8}$$

Average F-Measure (Avg. FM), a popular measure of a testing accuracy. F-Measure considers both the precision and the recall of the test to compute the score. Precision is the number of correct results divided by the number of all returned results and recall is the number of correct results divided by the number of results that should have been returned. F-Measure reaches its best value at 1 and worst score at 0. Average F-Measure of classes on a data set is calculated in our experiments.

$$\text{Avg. FM} = \frac{\sum_{i=1}^{N} \left(2 \times \frac{\text{precision}_i \times \text{recall}_i}{\text{precission}_i + \text{recall}_i}\right)}{N} \times 100 \tag{9}$$

Ten well-known classification data sets are selected for the experiments, including Wisconsin Breast Cancer Diagnostic Data Set (WBCD), Wine Data Set (WINE), Statlog Vehicle Silhouettes Data Set (VEHICLE), Zoo Data Set (ZOO), Iris Data Set(IRIS), Ionosphere Data Set(IONOSPHERE), Statlog Image Segmentation Data Set(SEGMENT), Contraceptive Method Choice Data Set (CMC), Statlog Landsat Satellite Data Set (SATELLITE), and Glass Identification Data Set (GLASS). All these data sets are selected from the UCI

Table 1 Characteristics of data set

Data set name	No. of samples	No. of features	No. of classes
IONOSPHERE	351	34	2
WBCD	569	30	2
IRIS	150	4	3
WINE	178	13	3
CMC	1,473	9	3
VEHICLE	846	18	4
SATELLITE	6,435	36	6
GLASS	214	9	6
ZOO	101	16	7
SEGMENT	2310	19	7

machine learning repository, which are available from the web site: http://archive.ics.uci.edu/ml/. Main characteristics of these data sets are depicted in Table 1, including number of data samples, number of features (inputs), and number of data classes.

All features are normalized for each data set. For a feature F in a data set, if the maximum value is $f_{\max}$ and the minimum value is $f_{\min}$, original value of this feature of a sample is f. Then the feature value of this sample is normalized as follows:

$$f' = \frac{f - f_{\min}}{f_{\max} - f_{\min}} \tag{10}$$

Tenfold cross-validation are used to validate proposed method. Each data set was split into ten subsets randomly, and all samples of each class are uniformly assigned to these subsets. Then one subset was used as the testing set, the other nine subsets as a whole training set. One time validation was performed for this pair of training and testing sets. The procedure was repeated ten times and each subset was used as test set for one time and the mean and standard deviations were reported. A three-layered feedforward neural network with 15 hidden neurons is selected as $\mathscr{F}$. The optimizer $\mathscr{T}$ used in experiments is Particle Swarm Optimization(PSO) [20–22].

First, the two important parameters m and k are evaluated. GLASS ($N = 6$) and ZOO ($N = 7$) data sets are used in this evaluation. Variations of average generalization accuracy are reported in Fig. 6. For the analysis of m, generalization accuracy rises with the increase in m while the growth trend is slowed gradually. Selecting m from $[2N, 3N]$ is enough to ensure dimensions redundant. For the analysis of k, generalization accuracy first increase and then reduce gradually with the increase in k. Especially, this trend appears more obvious when m is small. Hence, we suggest selecting k from $[1.5N, 4.5N]$ to find optimal neural network.

We compare FCM with four other neural network classifiers, including Traditional method (simple one output method for binary classification and one-per-class method for multi-classification), SoftMax, ECOC (exhaustive code), and FDPS (partitions number in each axe is 2). In view that ECOC and SoftMax are multi-classifier, we do not evaluate them on IONOSPHERE and WBCD. In FCM, the parameter m is set to $2N$ and k is set to $[1.5N]$. Particle Swarm Optimization (PSO) algorithm is used as the optimization approach for all of the comparison methods. PSO parameters' settings have an important impact on searching the optimal solution. Therefore, for a fair comparison, PSO parameters in all of the

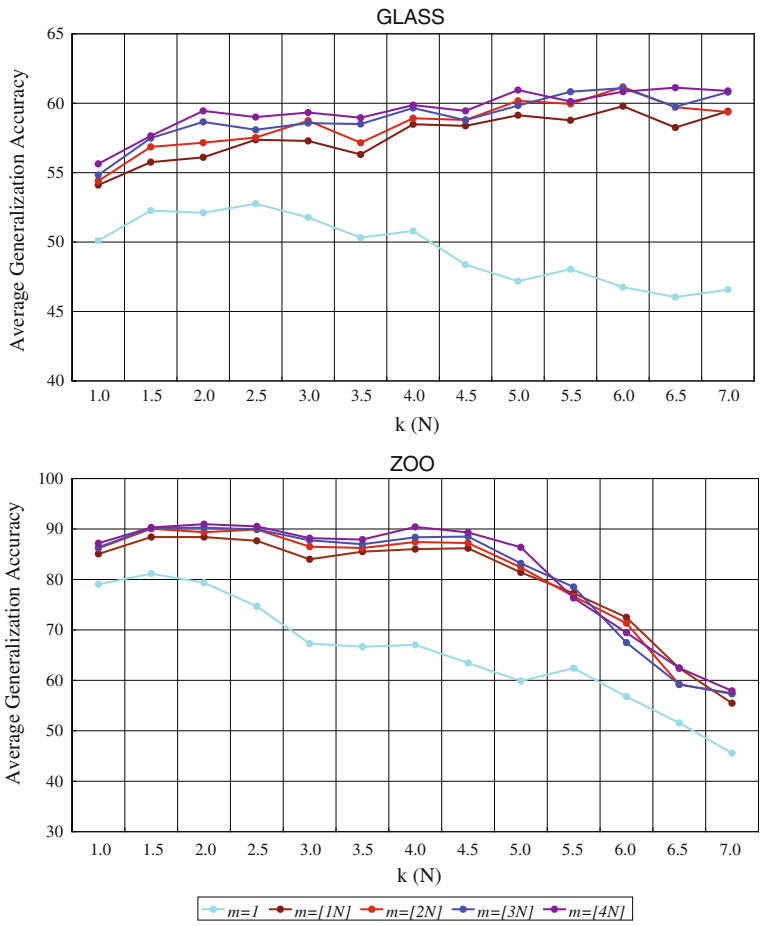


Fig. 6 Variation of generalization accuracy with the increase of k and m. k increases from $[1N]$ to $[7N]$ with step $0.5N$ while m is set to 1, $[1N]$, $[2N]$, $[3N]$ and $[4N]$. PSO parameters: Max Generation = 500, Population Size=20, $\varphi 0 = 1$, $\varphi 1 = 1.8$, $\varphi 2 = 1.8$ and VMAX = 3

comparison methods are the same. In PSO algorithm, with the expansion of population size and increase in the number of generations, probability to find the optimal solution increases gradually. Therefore, according to this characteristic, we first set up a small population and limit generation steps to find optimal parameters. Through trial and error on these data sets, the following settings are adopted in real experiments: Max Generation = 10,000, Population Size = 20, $\varphi 0 = 1$, $\varphi 1 = 1.8$, $\varphi 2 = 1.8$ and VMAX = 3.

Tables 2, 3 and 4 depict the accuracy results for comparison between FCM and the other methods. The average TA of FCM is obviously higher than other methods for most data sets. Moreover, the smallest standard deviation is also produced by FCM except VEHICLE, GLASS, and ZOO, which means it is more stable. These reflect that the introduction of floating centroids increases the chance to find optimal neural network during learning process. Furthermore, FCM achieves the highest average GA in the experiments on all of these data sets. The lowest standard deviation is also obtained by FCM on most data sets which ensures the stability of prediction. The results of Avg. FM in Table 4 further demonstrate FCM is

Table 2 Training accuracy results on these data sets

	Traditional	SoftMax	ECOC	FDPS	FCM
IONOSPHERE	**98.57** (±0.72)	N/A	N/A	96.51 (±1.03)	98.54 (±**0.58**)
WBCD	97.48 (±2.78)	N/A	N/A	96.88 (±0.90)	**98.54** (±**0.43**)
IRIS	88.59 (±14.55)	**99.33** (±**0.23**)	89.48 (±15.75)	99.11 (±0.47)	**99.33** (±**0.23**)
WINE	90.06 (±16.00)	99.94 (±0.20)	94.63 (±11.33)	99.75 (±0.32)	**100.00** (±**0.00**)
CMC	46.50 (±3.36)	60.35 (±4.37)	46.47 (±5.05)	55.48 (±1.91)	**61.17** (±**1.83**)
VEHICLE	49.13 (±13.42)	80.00 (±8.20)	76.95 (±4.70)	72.04 (±**1.81**)	**82.80** (±2.96)
SATELLITE	42.85 (±18.01)	75.35 (±7.73)	83.30 (±1.01)	83.96 (±0.95)	**87.57** (±**0.63**)
GLASS	39.13 (±19.28)	69.85 (±6.48)	74.28 (±2.70)	**78.91** (±**1.88**)	77.80 (±3.66)
ZOO	82.21 (±15.01)	**100.00** (±**0.00**)	99.42 (±1.13)	**100.00** (±**0.00**)	99.88 (±0.37)
SEGMENT	58.76 (±13.44)	85.45 (±7.42)	91.82 (±1.49)	91.19 (±1.71)	**96.50** (±**0.64**)

Table 3 Generalization accuracy results on these data sets

Traditional	SoftMax	ECOC	FDPS	FCM	
IONOSPHERE	91.94 (±3.33)	N/A	N/A	88.89 (±6.80)	**92.22** (±**2.55**)
WBCD	93.50 (±3.89)	N/A	N/A	92.96 (±5.07)	**94.71** (±**3.65**)
IRIS	82.67 (±14.47)	95.33 (±7.06)	86.67 (±14.74)	96.00 (±**5.62**)	**96.67** (±5.67)
WINE	97.78 (±3.88)	96.67 (±4.68)	91.67 (±15.33)	97.22 (±2.93)	**98.89** (±**2.34**)
CMC	46.71 (±4.03)	52.01 (±4.83)	45.50 (±**3.22**)	48.39 (±3.75)	**54.50** (±4.52)
VEHICLE	49.07 (±11.47)	71.74 (±8.00)	73.60 (±5.75)	67.21 (±3.54)	**78.37** (±**2.70**)
SATELLITE	42.12 (±18.17)	73.87 (±8.31)	82.48 (±3.80)	82.54 (±**2.67**)	**85.03** (±3.14)
GLASS	37.63 (±13.09)	54.00 (±**7.16**)	59.82 (±7.47)	58.17 (±11.90)	**62.79** (±16.19)
ZOO	73.79 (±15.04)	91.90 (±6.11)	90.00 (±7.20)	75.69 (±8.33)	**93.23** (±**5.45**)
SEGMENT	59.03 (±13.40)	83.85 (±6.88)	91.74 (±2.61)	88.79 (±2.48)	**95.49** (±**1.56**)

Table 4 Average F-Measure results on these data sets

Traditional	SoftMax	ECOC	FDPS	FCM	
IONOSPHERE	90.98 (±3.92)	N/A	N/A	87.40 (±7.93)	**91.21** (±**3.96**)
WBCD	92.93 (±4.23)	N/A	N/A	92.50 (±5.25)	**94.36** (±**3.67**)
IRIS	78.01 (±20.16)	95.24 (±7.19)	83.27 (±19.84)	95.93 (±**5.73**)	**96.60** (±5.78)
WINE	97.63 (±4.21)	96.73 (±4.62)	88.89 (±20.84)	97.14 (±3.02)	**98.93** (±**2.26**)
CMC	30.64 (±8.20)	43.99 (±8.31)	29.50 (±6.70)	44.48 (±**4.65**)	**52.37** (±4.66)
VEHICLE	37.44 (±14.10)	68.41 (±10.59)	69.63 (±8.53)	66.01 (3.78)	**77.57** (±**3.18**)
SATELLITE	28.23 (±16.23)	57.65 (±11.23)	72.26 (±3.79)	76.19 (±**3.04**)	**81.50** (±3.26)
GLASS	22.63 (±12.48)	36.48 (±14.46)	40.37 (±**6.74**)	46.69 (±11.68)	**54.15** (±15.10)
ZOO	54.23 (±16.77)	83.29 (±14.53)	76.82 (±16.24)	57.85 (±**8.63**)	**85.39** (±12.33)
SEGMENT	48.62 (±15.25)	79.41 (±9.19)	91.58 (±2.78)	88.55 (±2.61)	**95.53** (±**1.51**)

superior to the other four methods. These indicate that FCM is fully capable of improving the generalization ability of neural network classifier.

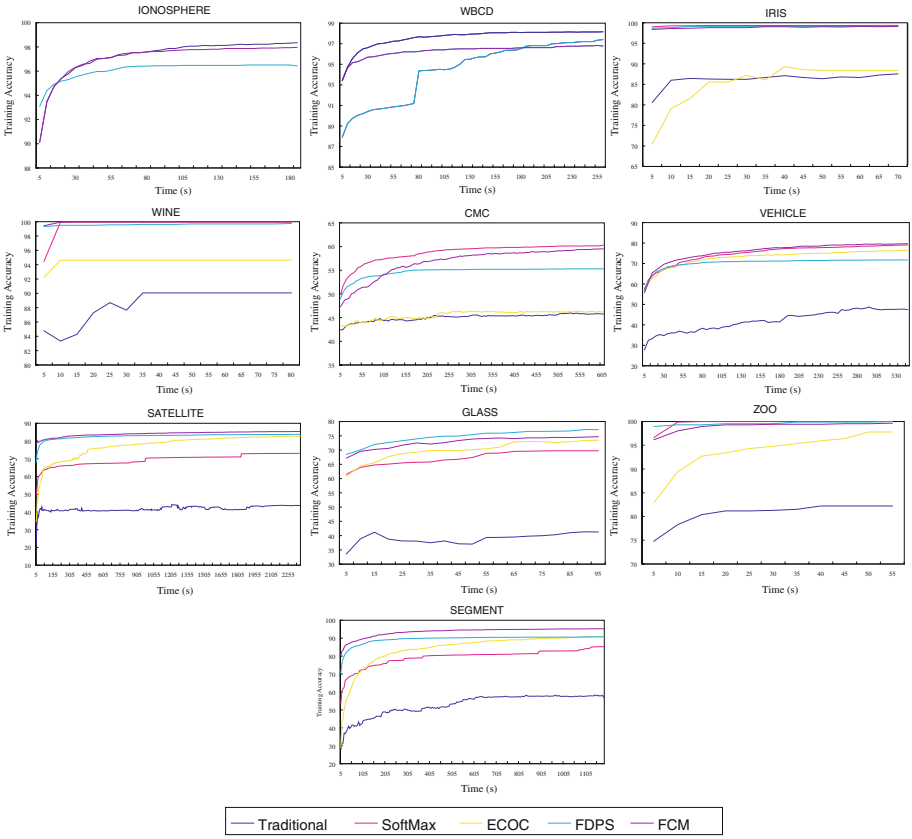


Fig. 7 Training accuracy versus time on these data sets

We admit that the use of K-Means algorithm will increase time complexity of FCM. However, the flexibility of floating centroids increases the chance to find optimal neural network and compensates for this disadvantage. Figure 7 illustrates the optimization speed over time, in which the average training accuracy at 5-second intervals is evaluated during learning process. FCM is the fastest method on WINE, VEHICLE, SATELLITE, and SEGMENT and is also the superior one on the other data sets. That is to say, FCM can be compared with the other methods in terms of speed even if it adopts K-Means algorithm.

We also compared FCM with other existing classification methods outside the area of neural networks, including Naive Bayes(NB), Support Vector Machines(radial basis function kernel, RBFSVMs) and K-Nearest Neighbor(KNN). In view of the marked differences between these methods, for fair comparison, we adjusted parameters of RBFSVMs, KNN, and FCM on each data set to get optimal performance. Many parameters were tried and only those which gave rise to the best results were retained. The RBFSVMs were trained by sequential minimal optimization method [23] using different values of the cost parameter $C = \{2^{-3}, 2^{-4}, \ldots, 2^{11}\}$ and kernel parameter $\gamma = \{2^{-11}, 2^{-10}, \ldots, 2^{3}\}$. For KNN, the parameter K was selected from $\{1, 3, \ldots, 25\}$. For FCM, the parameter m was fixed to $3N$ while k was selected from $\{[1.5N], [2N], \ldots, [4.5N]\}$. Furthermore, the number of hidden neurons was selected from $\{5, 10, \ldots, 50\}$. Figures 8 and 9 illustrate comparison of GA

Fig. 8 Comparison of generalization accuracy between FCM and the other classification methods

and Avg. FM between FCM and the other classification methods. The average accuracy of FCM is marginally lower than RBFSVMs but higher than NB and KNN. Furthermore, it should be noticed that FCM is the best classifier on half number of data sets (best one on IONOSPHERE, WINE, and CMC, tied for first place on IRIS and ZOO) and tied for the first place with RBFSVMs (best one on WBCD, VEHICLE, and SATELLITE, tied for first place on GLASS and ZOO). Although RBFSVMs also is able to identify clusters of classes in feature space and achieves the best average accuracy, FCM still shows better performance on several data sets. In addition, FCM is better than KNN on most data sets, which means that the mapping of neural network helps FCM to identify more samples correctly even if similarities are not enough. These reflect that neural network classifier using Floating Centroids Method is superior on many data sets.

5 Conclusions

In this paper, we proposed a classification approach called Floating Centroids Method, which is a novel improvement for neural network classifier. This approach divides partition space into many irregular partitions using floating centroids, which are spread throughout the partition space and obtained by using K-Means algorithm. A sample will be considered relating to a certain class if the closest centroid of its corresponding mapped point is labeled by this class. This approach removes the fixed-centroid constraint and increases the chance to find optimal neural network.

To evaluate the performance of the proposed approach, ten well-known classification data sets are selected to compare some accuracy measures and optimization speed. Experimental results manifest that these measures including training accuracy, generalization accuracy, and average F-Measure are improved by employing the novel approach.

Fig. 9 Comparison of average F-Measure between FCM and the other classification methods

Acknowledgments This work was supported by National Basic Pre-Research Program of China (973 Program) under Grant No. 2010CB635117. National Natural Science Foundation of China under Grant No. 60873089, No. 60573065, No. 61070130, No. 60903176, No. 60673130, and No. 90818001. Provincial Natural Science Foundation for Outstanding Young Schoolers of Shandong under Grant No. JQ200820. Program for New Century Excellent Talents in University under Grant No. NCET-10-0863.

References

1. Qinlan JR (1986) Introduction of decision trees. Mach Learn 1(1):81–106
2. Freund Y (1995) Boosting a weak learning algorithm by majority. Inf Comput 121(2):256–285
3. Hongjun Lu, Rudy S, Huan L (1996) Effect data mining using neural networks. IEEE Trans Knowl Data Eng 8(6):957–961
4. Misraa BB, Dehurib S, Dashc PK, Pandad G (2008) A reduced and comprehensible polynomial neural network for classification. Pattern Recognit Lett 29(12):1705–1712
5. Daqi G, Yan J (2005) Classification methodologies of multilayer perceptrons with sigmoid activation functions. Pattern Recognit 38(10):1469–1482
6. Chen Y, Abraham A, Yang Bo (2006) Feature selection and classification using flexible neural tree. Neurocomputing 70(1-3):305–313
7. Vapnik VN (1995) The nature of statistical learning theory. Springer, New York
8. Chua YS (2003) Efficient computations for large least square support vector machine classifiers. Pattern Recognit Lett 24(1-3):75–80
9. Koknar-Tezel S, Latecki LJ (2010) Improving SVM classification on imbalanced time series data sets with ghost points. Knowl Inf Syst (in print)
10. Benjamin XW, Japkowicz N (2010) Boosting support vector machines for imbalanced data sets. Knowl Inf Syst (in print)
11. Gomez-Ruiz JA, Jerez-Aragones JM, Munoz-Perez J, Alba-Conejo E (2004) A neural network based model for prognosis of early breast cancer. Appl Intell 20(3):231–238
12. Cho SB, Won H-H (2007) Cancer classification using ensemble of neural networks with multiple significant gene subsets. Appl Intell 26(3):243–250
13. Venkatesh YV, Kumar Raja S (2003) On the classification of multispectral satellite images using the multilayer perceptron. Pattern Recognit 36(9):2161–2175

14. Garfield S, Wermter S (2006) Call classification using recurrent neural networks, support vector machines and finite state automata. Knowl Inf Syst 9(2):131–156
15. Tsai C-Y, Chou S-Y, Lin S-W, Wang W-H (2009) Location determination of mobile devices for an indoor WLAN application using a neural network. Knowl Inf Syst 20(1):81–93
16. Bridle JS (1990) Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. Neuralcomputing: algorithms, architectures and applications, Springer, pp 227–236
17. Dietterich TG, Bakiri G (1995) Solving multiclass learning problems via error-correcting output codes. J Artif Intell Res 2(1):263–286
18. Wang L, Yang B, Chen Z, Abraham A, Peng L (2007) A novel improvement of neural network classification using further division of partition space. In: Proceedings of international work-conference on the interplay between natural and artificial computation, 2007. Lecture notes in computer science vol 4527. Springer, Berlin, pp 214–223
19. Hartigan JA, Wong MA (1979) A k-means clustering algorithm. Applied Stat 28(1):100–108
20. Kennedy J, Eberhart RC (1995) A new optimizer using paritcle swarm theory. In: Proceddings of the sixth international symposium on micromachine and human science, pp 39–43
21. Lin S-W, Chen S-C, Wu W-J, Chen C-H (2009) Parameter determination and feature selection for back-propagation network by particle swarm optimization. Knowl Inf Syst 21(2):249–266
22. Wang T, Yang J (2010) A heuristic method for learning Bayesian networks using discrete particle swarm optimization. Knowl Inf Syst 24(2):269–281
23. Platt JC (1998) Sequential minimal optimization: a fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research

Author Biographies

Lin Wang was born in 1983 in Shandong Province of China. In 2005 and 2008, he received his B.Sc. and Master degrees in Computer Science and Technology from the University of Jinan. In 2011, he received his PhD degree in Computer Science and Technology from the School of Computer science and technology, Shandong University, Jinan, China. Now, he is a researcher in Shandong Provincial Key Laboratory of Network-based Intelligent Computing, University of Jinan, Jinan, 250022, China. His research interests include Classification, Hybrid Computational Intelligence, and Mathematical Modeling.

Bo Yang is a professor and vice-president of University of Jinan, Jinan, China. He is the Director of the Provincial Key Laboratory for Network-based Intelligent Computing and also acts as the Associate Director of Shandong Computer Federation, and Member of the Technical Committee of Intelligent Control of Chinese Association of Automation. His main research interests include computer networks, artificial intelligence, machine learning, knowledge discovery, and data mining. He has published numerous papers and gotten some of important scientific awards in this area.

Yuehui Chen received his PhD degree in electrical engineering from the Kumamoto University of Japan in 2001. During 2001–2003, he had worked as the Senior Researcher of the Memory-Tech Corporation at Tokyo. Since 2003, he has been a member at the Faculty of Electrical Engineering in University of Jinan, where he is currently head of the Laboratory of Computational Intelligence. His research interests include Evolutionary Computation, Neural Networks, Fuzzy Logic Systems, Hybrid Computational Intelligence and their applications in time-series prediction, system identification, intelligent control, intrusion detection systems, web intelligence, and bioinformatics. He is the author and co-author of more than 70 technique papers. Professor Yuehui Chen is a member of IEEE, the IEEE Systems, Man and Cybernetics Society, and the Computational Intelligence Society, a member of Young Researchers Committee of the World Federation on Soft Computing, and a member of CCF Young Computer Science and Engineering Forum of CHINA.

Ajith Abraham received PhD degree in Computer Science, from Monash University, Australia and a Master of Science degree from Nanyang Technological University Singapore. He works in a multidisciplinary environment involving machine intelligence, terrorism informatics, network security, sensor networks, e-commerce, Web intelligence, Web services, computational grids, data mining, and applied to various real world problems. He has given more than 35 plenary lectures and conference tutorials in these areas. He has published over 600+ publications and some of the works have also won best paper awards at International conferences. Currently, he is also coordinating the activities of the Machine Intelligence Research Labs (MIR Labs), International Scientific Network of Excellence, which has members from over 60 countries. He is a Senior Member of IEEE, IEEE Systems Man and Cybernetics Society, IEEE Computer Society, IET (UK), IEAust (Australia) etc.

Hongwei Sun is an associate professor of University of Jinan, Jinan, China. He was born in 1966 and received his B.Sc. and Master degree in mathematics from the Zhejiang University of China in 1988 and 1991, and PhD degree in mathematics from the Beijing Normal University in 2008.

Zhenxiang Chen was born in Hunan Province, China, in 1979. He received his B.S and M.S. degrees from University of Jinan, Jinan, Shandong, China, in 2001 and 2004. In 2008, He received his PhD degree in the School of Computer science and technology, Shandong University, Jinan, China. His research interests include Classification, Computer Software Theory, Network Security, Hybrid Computational Intelligence.

Haiyang Wang was born in 1965. He is a professor and doctoral supervisor at the Shandong University and a CCF senior member. His research areas are software and data engineering, CSCW and business process management.