Improving Neural-Network Classifiers Using Nearest Neighbor Partitioning

Lin Wang, Bo Yang, Yuehui Chen, Xiaoqian Zhang, and Jeff Orchard

***Abstract*— This paper presents a nearest neighbor partitioning method designed to improve the performance of a neural-network classifier. For neural-network classifiers, usually the number, positions, and labels of centroids are fixed in partition space before training. However, that approach limits the search for potential neural networks during optimization; the quality of a neural network classifier is based on how clear the decision boundaries are between classes. Although attempts have been made to generate floating centroids automatically, these methods still tend to generate sphere-like partitions and cannot produce flexible decision boundaries. We propose the use of nearest neighbor classification in conjunction with a neural-network classifier. Instead of being bound by sphere-like boundaries (such as the case with centroid-based methods), the flexibility of nearest neighbors increases the chance of finding potential neural networks that have arbitrarily shaped boundaries in partition space. Experimental results demonstrate that the proposed method exhibits superior performance on accuracy and average f-measure.**

***Index Terms*— Classification, neural networks, nearest neighbor partitioning.**

I. Introduction

SUPERVISED classification is one of the most important fields in data mining and pattern recognition. Once learned from a training data set, a classification model can predict the label of unknown samples. Although plenty of supervised classification approaches have been developed (e.g., decision trees [1], [2], and support vector machines [7]–[9]), the artificial neural network has some advantages. It is a general model for approximating a wide array of nonlinear functions that can map samples directly to labeled centroids [3]–[6], and thus has been used successfully for many classification tasks [10]–[15]. The neural network classifier maps each sample from data space to a new space, called the partition space, where it categorizes the sample. In partition space, a centroid marks the center of each class, and samples are classified according to their closest centroid. Conventionally, the number, positions, and labels of centroids are fixed before training. However, mapping samples toward these fixed centroids limits the range of potential neural networks during optimization. After all, a neural network that maps points into clearly discernible classes should be considered a success, even if the resulting clusters do not fit the sphere-like distribution of a centroid-based classifier. Although attempts have been made to generate floating centroids automatically, the sphere-like partitions cannot generate flexible decision boundaries.

The question that arises here is, can we generate flexible partitions and decision boundaries for different classes in partition space to improve the performance of a neural-network classifier?

To address this issue, a nearest-neighbor partitioning (NNP) method is proposed in this paper. In the optimization process, all of the training samples are mapped to partition space first by neural networks. Then, the quality of each neural network is evaluated based on how compact the separated the different class clusters are, as evaluated using a nearest-neighbor criterion. After optimization, we obtain the best neural network and its corresponding partition space (whose feature is reflected by the mapped training samples). Based on the distribution of mapped training samples, the category of any new sample can be predicted according to its neighborhood in partition space. There is no concept of a centroid in NNP. The categorization of a point is determined by the class of nearby points in partition space, not proximity to a handful of centroids. The NNP method generates flexible partitions and decision boundaries easily and increases the chance of finding potential neural networks during optimization.

The rest of this paper is arranged as follows. Section II reviews some important neural-network classification methods and their problems. Section III gives a detailed account of the NNP approach. Section IV discusses the user-defined parameters. Section V outlines and discusses our experiments, followed by the conclusions in Section VI.

II. Related Work

In traditional neural-network classifiers, the method of solving the classification problem is straightforward. For a binary classification problem, the neural network has one output that estimates the chance that a sample belongs to the "true" class. For a multi-class classification problem, the one-per-class method is adopted. It uses multiple output

Manuscript received November 30, 2015; revised February 5, 2016; accepted June 8, 2016. Date of publication November 30, 2015; date of current version September 15, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61573166, Grant 61572230, Grant 61373054, Grant 61472164, Grant 81301298, Grant 61302128, and Grant 61472163, in part by the Shandong Provincial Natural Science Foundation, China, under Grant ZR2015JL025, and in part by the Science and technology project of Shandong Province under Grant 2015GGX101025. *(Corresponding author: Bo Yang.)*

L. Wang, B. Yang, Y. Chen, and X. Zhang are with the Shandong Provincial Key Laboratory of Network Based Intelligent Computing, University of Jinan, Jinan 250022, China (e-mail: yangbo@ujn.edu.cn; wangplanet@gmail.com).

J. Orchard is with the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, N2L 3G1, Canada (e-mail: jorchard@cs.uwaterloo.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2016.2580570

units and categorizes a sample according to the class whose corresponding output gives the highest value among outputs.

Bridle [16] proposed a method that deals with a problem in one-per-class methods, in which the sum of the individual probabilities does not always equal to one. It normalizes the outputs of the neural network using a softmax activation function (SoftMax). In the corresponding geometric interpretation, the partition space in SoftMax is a hyperplane whose number of dimensions equals the number of classes. Centroids are defined as the intersection points between hyperplanes and axes.

Dietterich and Bakiri [17] proposed an error-correcting output code (ECOC) method to provide additional statistical robustness. Geometrically, the dimension of the partition space is expanded to a number that is larger than the number of classes. Redundant dimensions are used for correcting errors using a codewords table, each codeword corresponds to a centroid in partition space. The redundant dimensions and appropriate codewords table have the ability to correct the errors and restore the labels. Zhou *et al*. [18] proposed a method for learning ECOC with the help of a single-layered perceptron neural network. Bagheri *et al*. [19] proposed an evolutionary-algorithm-based approach to the design of an application-dependent code matrix for the ECOC framework.

Besides the centroids-based neural-network classifiers, radial basis function neural networks (RBFNNs) are a special subset of neural networks. These methods use the overlapping localized regions generated by radial basis functions to create complex decision boundaries [20]. Babu and Suresh [21] proposed a sequential projection-based metacognitive learning algorithm to improve the radial basis function network classifier. Zhang *et al*. [22] proposed a new discrete-continuous algorithm for the construction of an RBF model. Patrikar [23] showed that RBFNN with Gaussian basis functions and a Euclidean norm can be approximated accurately with a multilayer perceptron with quadratic inputs. Tian *et al*. [24] propose a novel algorithm for the construction of radial basis function neural network classifier based on subspace learning. Marko Robnik-Sikonja [25] proposes a radial basis function networks based generator of semiartificial data with similar properties to the original data, which enables the development and testing of different data mining algorithms and the optimization of their parameters.

III. Methodology of Nearest Neighbor Partitioning

In this section, the method of nearest-neighbor partitioning is described in detail. NNP uses evolutionary computation to optimize the neural-network parameters, i.e., the weights and biases of a multilayer perceptron. The core of the optimization rests in the fitness evaluation, guiding the direction of evolution. To evaluate a set of parameter values, we first map all of the training samples into partition space using the corresponding neural network. Then, the quality of the mapping (and, thus, the neural network) is evaluated by the distribution of nearest neighbors, assigning a higher fitness when samples from the same class get mapped close together, and samples from different classes get mapped farther apart. After optimization, we obtain the parameters for the best neural network, as well as its corresponding partition space, whose features are reflected by the mapped training samples. Based on the distribution of mapped training samples, the category of any new sample is predicted according to its nearest neighbors in partition space using the process of classical k-Nearest Neighbors (kNN).

Next, we describe the motivation and model of the NNP method. Then, the optimization processes for NNP is described in detail, including its mapping relationships, normalization, similarity, and optimization objective function. Finally, we explain the method to classify a new sample based on a trained NNP.

A. Motivation

Currently, the centroids used in neural-network classifiers are set manually before training and are fixed. Although they have been proven to be effective in many situations, the position, number, and label of these fixed centroids limits the scope of potential neural networks during optimization. A neural-network classifier could be considered effective if it yields clear decision boundaries between different classes.

In our previous research [26], we proposed a floating-centroids method (FCM) to remove the fixed-centroid constraint for neural network classifiers. It introduces a lot of floating centroids, spread throughout the partition space, each assigned a class label automatically according to the actual distribution of mapped samples. This approach divides partition space into many irregular partitions using floating centroids. It increases the chance of finding an optimal neural network, performs better than other classifiers, and shows favorable application prospects [26]–[29].

However, there are still significant problems in FCM that lower its performance. Its centroid-based nature still prefers sphere-like partitions and cannot generate flexible decision boundaries between classes; the decision boundaries in a classification task need not be spherical in practice. Furthermore, FCM does not have a strategy to handle anisotropic variation, where the dispersion of points along one axis is vastly different than the dispersion along another axis. It causes the model to prefer the dimension that has a broad range of values, impairing classification performance. Finally, the optimization target function used in FCM only applies to centroid-based methods and does not incorporate strategies to handle imbalanced data.

To address these problems, we propose NNP. There is no need for centroids in the NNP method. The categorization of the neural network is determined by the distribution of normalized mapped training samples in partition space, not a handful of centroids. The target of optimization for NNP is to find a neural network that maps points in the same class close together, while spreading points from different classes as far away from each other as possible. The target function includes a facility to account for class weights for imbalanced data. NNP generates flexible partitions and arbitrarily shaped decision boundaries easily and increases the chance of finding potential neural-network classifiers.

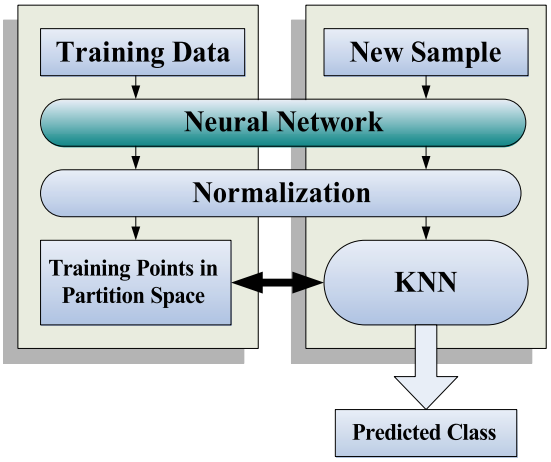


Fig. 1. Overview of nearest neighbor partitioning.

B. Model

An example NNP model is depicted in Fig. 1.During the training stage, the neural network is optimized iteratively. After training, the neural network's normalization parameters are set, thus fixing the location of the training points in partition space.

The class of a new sample is determined by mapping it into the partition space using the neural network, and then observing the class of its nearest neighbors. This method maps samples from data space to partition space, where points from the same class get mapped close together, while keeping points from different classes away from each other.

Fig. 2 illustrates an example of the NNP method for classification into three classes. The only requirement of the NNP is that there be clear decision boundaries between mapped points from different classes. The NNP frees us from the sphere-like boundaries that are typical in centroid-based methods, and generates arbitrarily shaped boundaries in partition space. This flexibility increases the opportunity to find an optimal neural network.

C. Optimization

The objective of the optimization process is to get the best neural network and its corresponding partition space, where features are reflected by the mapped training samples. The search space of the NNP is multimodal because there is no unique target solution. A neural network need only map points to regions with clear decision boundaries between clusters, regardless of the region's shape [30], eccentricity, or sparsity. Bayesian regularization [31] is an optimization method designed to produce solutions with better generalization ability. However, Bayesian regularization does not apply to the optimization in NNP because the Levenberg-Marquardt algorithm used for classic Bayesian regularization [31] only finds a local minimum. Instead, evolutionary computation has properties that enable a more global search [32]; we use evolutionary computation to optimize the NNP. Furthermore, since the time complexity of evolutionary computation is high, Bayesian regularization is not incorporated into the optimization framework because the calculation of the Hessian matrix would render the method infeasible.

Algorithm 1 Common Framework for Optimizing NNP

```
1  Initialize the population;
2  Calculate the weight for each class in the data set;
3  while termination condition has not been met do
4      for every individual in the population do
5          Decode the current individual to a neural network;
6          for every sample in the data set do
7              Map the current sample from data space to
               partition space using (1);
8          end
9          for every mapped point in partition space do
10             Perform normalization for current point using
               Z-score;
11         end
12         for every normalized point in partition space do
13             Calculate the summary of within-class
               similarities S_self in neighborhood of current
               point;
14             Calculate the summary of similarities between
               current point and points from the other classes
               S_nonself in neighborhood of current point;
15         end
16         Compute the value of optimization target function
           as fitness of current individual according to (6);
17     end
18     Individuals reproduce to make the next generation,
       according to their fitness;
19 end
20 Return the best neural network and its corresponding
   colored partition space;
```

All evolutionary computation algorithms are guided by the same general principle, the fitness of individuals/chromosomes. To illustrate this idea, the unified common framework of optimizing the NNP using evolutionary computation is described as follows. At the beginning of evolution, a population of individuals is initialized randomly. Each individual encodes a combination of neural-network parameters and can be decoded to a neural network. Every gene/dimension in the individual represents a parameter, e.g., the weight of a connection. Then, the individuals are evolved iteratively (the specifics of the process vary between different evolutionary methods). The core of this framework rests in the fitness evaluation for each individual, i.e., the optimization target function, guiding the direction of evolution. To evaluate an individual, its neural network is decoded first. Then, every sample in the data set is mapped to partition space using the neural network, and those mapped points are further normalized. For every normalized point, we calculate the summary of within-class similarities, and the summary of similarities between the current point and points from the other classes in the neighborhood. The fitness of the current individual is obtained by calculating the optimization target function. Algorithm 1 shows this common framework for optimizing an NNP.

Each stage of this algorithm is described in the following sections.

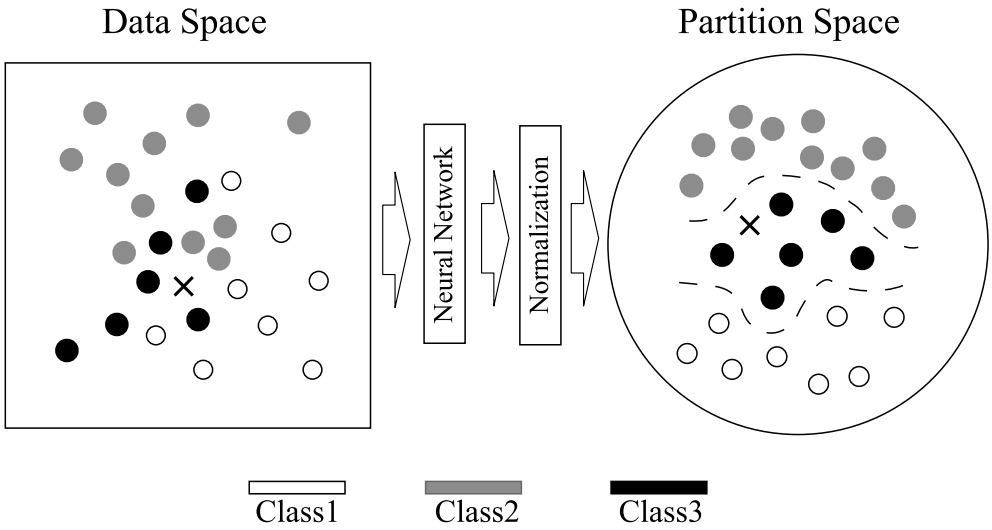


Fig. 2. Data space and partition space for classification into three classes using nearest neighbor partitioning.

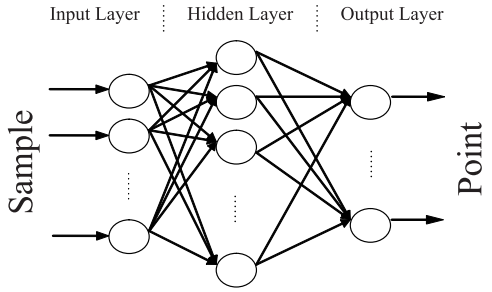


Fig. 3. Three-layer perceptron, mapping samples in data space to points in partition space.

1) Mapping: The neural network is able to approximate any nonlinear function. In this work, the neural network is used to map the original samples from data space to new points in partition space. The number of neurons in the input layer is equal to the dimension of the data space. The number of neurons in the output layer determines the dimension of the partition space, DPS. The mapping from data space to partition space is denoted

$$\vec{O} = \mathscr{G}(\vec{I}) \tag{1}$$

where $\mathscr{G}$ represents the mapping function (embodied in the neural network), $\vec{I}$ represents a sample in data space, and $\vec{O}$ represents the corresponding mapped point in partition space. Fig. 3 illustrates $\mathscr{G}$ in the form of a multilayer perceptron (MLP). An MLP is able to yield the required decision boundary after training. Furthermore, it has been proven to be able to approximate any nonlinear function [33].

2) Normalization: After mapping samples into partition space, the distribution of points will probably not be isotropic. Clusters might show little variance along one axis, but be more spread out along another axis, causing a bias in the resulting neural networks. To bring the different dimensions into a common scale, the Z-score [34] is adopted to standardize points because it is less sensitive to the presence of outliers in comparison with min-max normalization. While the Z-score does not normalize points into a compact range, e.g., [0, 1], we achieve a compact range using min-max normalization to calculate similarity (defined in the next section).

3) Similarity: The evaluation of similarity is an integral part of k-nearest neighbors. It is also essential in evaluating the distribution of points because the optimization target function is dependent upon this similarity. The optimization target function F, which will be discussed in the next section, decreases when the similarity of two points from the same class increases, and increases when the similarity of two points from different classes increases. Therefore, the similarity metric between points influences the design of the optimization target function.

The Euclidean distance is widely used in k-nearest neighbor implementations. However, the Euclidean distance itself is a *dissimilarity* metric and should be converted into a similarity metric. There is no definite upper bound for the distance between two normalized points. To facilitate the design of similarity (and therefore an optimization target function), we constrain the normalized points into a hypersphere of radius 1 using

$$h(\mathbf{x}) = \mathbf{x}\frac{\frac{1-e^{-|\mathbf{x}|/2}}{1+e^{-|\mathbf{x}|/2}}}{|\mathbf{x}|} = \mathbf{x}\frac{1-e^{-|\mathbf{x}|/2}}{|\mathbf{x}|+|\mathbf{x}|e^{-|\mathbf{x}|/2}} \tag{2}$$

where $\mathbf{x}$ represents a normalized point in partition space, and h is the function to constrain it to the hypersphere. The maximum distance between any points is the diameter of the hypersphere (which is 2). Then, based on the constrained points, the similarity s between two points is defined as

$$s(\mathbf{x}_1, \mathbf{x}_2) = 2 - |h(\mathbf{x}_1) - h(\mathbf{x}_2)| \tag{3}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ represent two normalized points.

4) Optimization Target Function: The optimization target function guides the direction of evolution of the neural network by evaluating the distribution of points that are mapped from data space into partition space. Geometrically, the optimization target is to find a neural network that maps points from the same class as close together as possible, and points from different classes as far apart as possible [26].

If all classes in the data set have the same number of samples (i.e., the data set is balanced), the optimization target function F can be defined

$$F = \sum_{i=1}^{n} \{S_{\text{nonself}}(\mathbf{x}_i) - \alpha S_{\text{self}}(\mathbf{x}_i)\} \tag{4}$$

where n is the number of points, $S_{\text{self}}(\mathbf{x}_i)$ is the sum of within-class similarities in the neighborhood of $\mathbf{x}_i$, and $S_{\text{nonself}}(\mathbf{x}_i)$ is the sum of similarities between $\mathbf{x}_i$ and points from the other classes in its neighborhood. The discrimination weight α is a coefficient that adjusts the weight between the two terms. S_{self} and S_{nonself} are defined as

$$S_{\text{self}}(\mathbf{x}) = \sum_{i=1}^{L} \begin{cases} s(\mathbf{x}, \mathbf{y}_i) & \text{if } c(\mathbf{x}) = c(\mathbf{y}_i) \\ 0 & \text{otherwise} \end{cases}$$

$$S_{\text{nonself}}(\mathbf{x}) = \sum_{i=1}^{L} \begin{cases} s(\mathbf{x}, \mathbf{y}_i) & \text{if } c(\mathbf{x}) \neq c(\mathbf{y}_i) \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{x}$ represents a point in partition space, $\mathbf{y}$ represents a point other than $\mathbf{x}$, c is the class of $\mathbf{x}$, and L is the number of nearest neighbors to $\mathbf{x}$. The points $\mathbf{y}_i$ are ordered such that

$$s(\mathbf{x}, \mathbf{y}_1) \geq \cdots \geq s(\mathbf{x}, \mathbf{y}_L) \geq \cdots \geq s(\mathbf{x}, \mathbf{y}_n).$$

This function is for dealing with balanced data sets because it views points in different classes equally. However, in many classification tasks, the collected data sets are imbalanced, which means the number of samples in each class differs; some classes have many samples, and some have far fewer. An imbalanced data set usually leads to a bias toward the majority class and therefore limits which classification methods can be used. To alleviate the influence of imbalanced data, each point is assigned a weight according to its class. The weight for a point $\mathbf{x}$ in partition space is

$$w(\mathbf{x}) = \frac{\text{Average number of points in all classes}}{\text{Number of points in class } c(\mathbf{x})}. \tag{5}$$

Thus, a point from a smaller, minority class has a greater weight than a point from a larger, majority class. In FCM, a point's weight dictates how much influence it has on the centroid of its class [26]. In NNP, all mapped points are equally treated in calculating the optimization target function. In this case, each point in partition space can be viewed as a "centroid." Therefore, a point's weight dictates how much influence it has on determining the class of points in its neighborhood. When weight is adopted, the optimization target function F becomes

$$F = \sum_{i=1}^{n} \{w(\mathbf{x}_i)(S_{\text{nonself}}(\mathbf{x}_i) - \alpha S_{\text{self}}(\mathbf{x}_i))\} \tag{6}$$

where S_{self} and S_{nonself} are defined in (7) and (8), as shown the bottom of this page, respectively. The $\mathbf{x}$ represents a point in partition space, and $\mathbf{y}$ represents a point other than $\mathbf{x}$. The unweighted version is equivalent to setting the weights of all the points to 1. In that case, L is not only the number of nearest neighbors, but also represents the total weights of the nearest neighbors. Therefore, after allowing for different weights for points, we generalize the notion of a neighborhood to include a fixed total weight. Furthermore, new criteria are added to deal with points that are located at the border of the total weights. The smaller the value of F, the better the neural network and its corresponding partition space.

D. Classifying New Samples

When a new sample arrives, it is mapped into partition space using the optimized neural network first. Then it is normalized using $(\mathbf{x} - \mu)/\sigma$, where μ and σ are determined by the mapped training data set. After this, the point is constrained into a hypersphere of radius 1 using (2). Finally, the class of this sample is determined via weighted k-nearest neighbors, using (5) as the weight for each sample/point. The sample/point is labeled as belonging to the class with the greatest neighborhood weight. K-nearest neighbors is selected to classify new samples because it uses the mapped points directly and does not need to train a model to learn the new distribution again.

IV. Analysis of User-Defined Parameters

Aside from parameters in the neural network and the optimizer, three user-defined parameters in the NNP need to be fine-tuned for different problems: the dimension of the partition space DPS, the number of nearest neighbors L, and the discrimination weight α. Figs. 4–6 show the effects of changing parameters on six data sets. The accuracy measure and data sets are defined in the next section.

A. Dimension of Partition Space

The dimension of partition space (DPS) is the number of neurons in the output layer. This parameter determines the

$$S_{\text{self}}(\mathbf{x}) = \sum_i \begin{cases} w(\mathbf{y}_i) s(\mathbf{x}, \mathbf{y}_i) & \text{if } c(\mathbf{x}) = c(\mathbf{y}_i) \wedge \sum_{j=1}^{i} w(\mathbf{y}_j) \leq L \\ \left(L - \sum_{j=1}^{i-1} w(\mathbf{y}_j) \right) s(\mathbf{x}, \mathbf{y}_i) & \text{if } c(\mathbf{x}) = c(\mathbf{y}_i) \wedge \sum_{j=1}^{i} w(\mathbf{y}_j) > L \wedge \sum_{j=1}^{i-1} w(\mathbf{y}_j) < L \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$S_{\text{nonself}}(\mathbf{x}) = \sum_i \begin{cases} w(\mathbf{y}_i) s(\mathbf{x}, \mathbf{y}_i) & \text{if } c(\mathbf{x}) \neq c(\mathbf{y}_i) \wedge \sum_{j=1}^{i} w(\mathbf{y}_j) \leq L \\ \left(L - \sum_{j=1}^{i-1} w(\mathbf{y}_j) \right) s(\mathbf{x}, \mathbf{y}_i) & \text{if } c(\mathbf{x}) \neq c(\mathbf{y}_i) \wedge \sum_{j=1}^{i} w(\mathbf{y}_j) > L \wedge \sum_{j=1}^{i-1} w(\mathbf{y}_j) < L \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

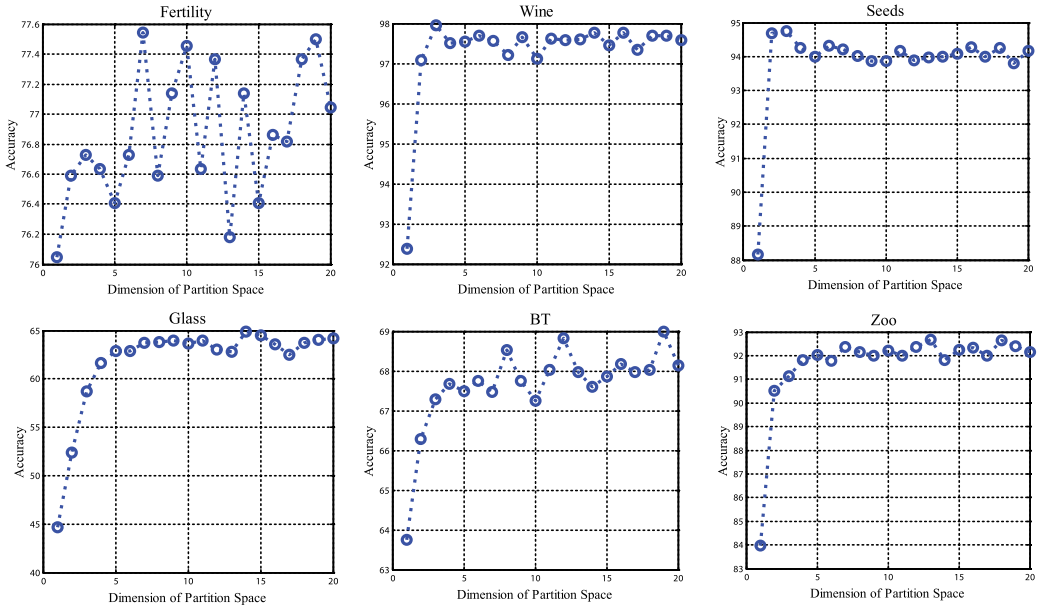


Fig. 4. Influence of dimension of partition space on accuracy for different data sets. A three-layer MLP neural network with 5 nodes in the hidden layer is adopted. The optimizer is particle swarm optimization whose maximum generation is set to 100, population size is set to 10, and both of φ_1 and φ_2 are set to 1.8. The number of nearest neighbors L is fixed at 10, and the discrimination weight is fixed at 1.

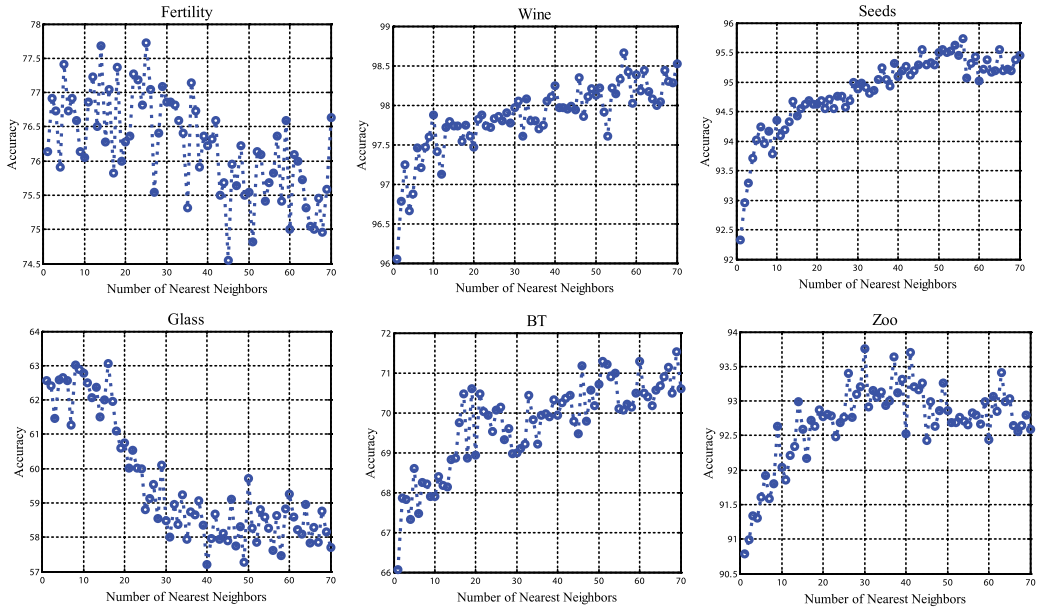


Fig. 5. Influence of number of nearest neighbors on accuracy for different data sets. A three-layer MLP neural network with 5 nodes in the hidden layer is adopted. The optimizer is particle swarm optimization whose maximum generation is set to 100, population size is set to 10, and both of φ_1 and φ_2 are set to 1.8. The dimension of partition space DPS is fixed at 5, and the discrimination weight is fixed at 1.

redundancy of dimension and influences the generalization of the classifier. A larger value of DPS adds redundant dimensions to improve the generalization ability and increases the chance of finding a satisfactory classier. Although the points from different classes may be indistinguishable along some axes, the redundant dimensions will restore the expected relationship between points if the number of "bad" dimensions is not too large. However, if DPS is set to a very large value, the performance of optimization decreases accordingly because of the increase of complexity of model.

Fig. 4 shows that accuracy increases as the dimension of partition space increases, while the growth trend slows down gradually for most data sets. Particularly, for Wine and Seeds, the accuracy is saturated when DPS is larger than 3. This suggests that if there are relatively clear decision boundaries between classes in the original data space, then a higher dimension only increases the complexity of the model and difficulty of optimization, but does not contribute to improved accuracy. In practice, we suggest selecting a DPS value that is a little larger than the number of classes. It should be noted that

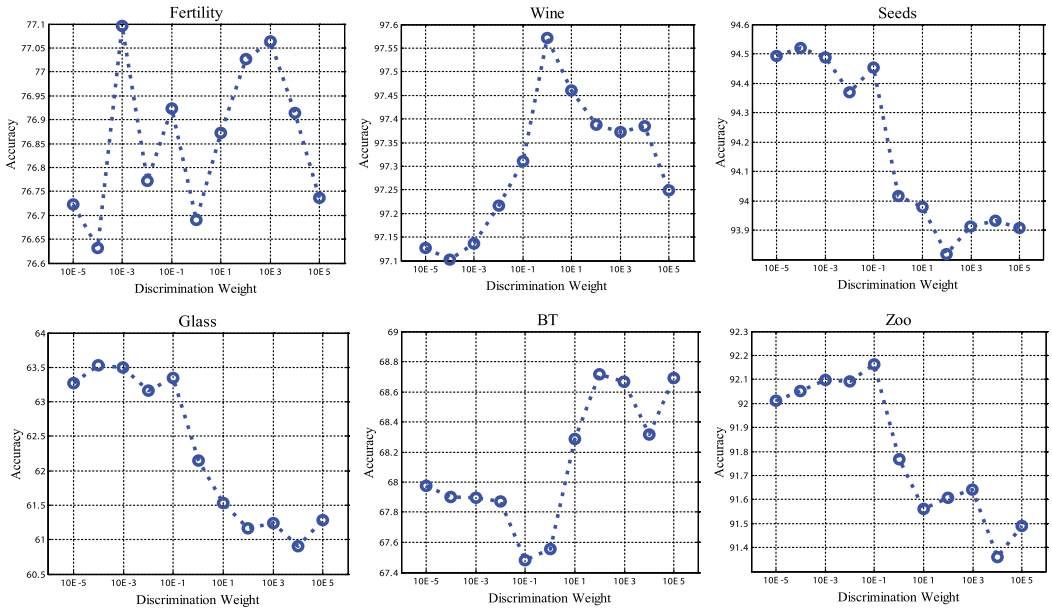


Fig. 6. Influence of discrimination weight on accuracy for different data sets. A three-layer MLP neural network with 5 nodes in the hidden layer is adopted. The optimizer is particle swarm optimization whose maximum generation is set to 100, population size is set to 10, and both of φ_1 and φ_2 are set to 1.8. The dimension of partition space DPS is fixed at 5, and the number of nearest neighbors L is fixed at 10.

the difference in accuracy between two dimensions and higher dimensions is less than 1% for Fertility, Wine, and Seeds, and is also very small for the other data sets. Therefore, apart from classification, the proposed method is also promising in data visualization by setting DPS to 2 or 3.

B. Number of Nearest Neighbors

The number of nearest neighbors L in the NNP is different from the k in the k-NN method. L represents the size of the neighborhood over which a point has influence, or can be influenced. If L is set to a very small value, each point barely influences the other points. In this case, it is hard to put a class's points together and keep different classes' points away. On the other hand, if L is set to a very large value, each point influences all the other points. In this case, the optimization efforts are, for the most part, wasted on dealing with points that are located very far from one another and are not at all helpful in forming clear decision boundaries. Furthermore, the increase of L also leads to an increase of computing time. As shown in Fig. 5, the accuracy is strongly influenced by the value of L. For the Fertility, Glass, and Zoo data sets, the accuracy increases at first with an increase in L, but then decreases again after the peak. For the other three data sets, the accuracy increases gradually with the increase in L. This phenomenon shows that if a class has multiple clusters or many outliers in partition space (e.g., Fertility, Glass, and Zoo), a very large L will make the neural network map points from different clusters to one cluster. This increases the complexity of the model, making the optimization more difficult, and therefore hinders the improvement of accuracy. On the contrary, a smaller L means the neural network only has to concentrate on nearby points and thus facilitates optimization. Experiments suggest that the parameter L should be tuned carefully. In practice, it can be set to a very small value and increased gradually until a decrease in accuracy is observed.

C. Discrimination Weight

The discrimination weight α is a coefficient that adjusts the relative push-pull of the within-class similarity term S_{self} and the between-class similarity term S_{nonself}. A larger α pushes points belonging to other classes away. Conversely, a smaller α pull points belonging to the same class closer.

Fig. 6 shows the impact of α on the final accuracy for different data sets. Some data sets are very sensitive to α (e.g., Glass), while others are not (e.g., Fertility). This sensitivity of clusters is presumably related to the distribution of points in the original data space between classes. If points from two classes are evenly distributed in the same region, e.g., Fertility, the accuracy is insensitive to α because attraction and repulsion each affect the generation of decision boundaries in partition space equally. If there are clear but not very large decision boundaries between classes, e.g., Wine, setting α to 1 is able to enlarge it because the points from the same class attract each other and points from different classes repulse each other equally at the same time. If only a few points from the two classes are distributed in the same region, e.g., Seeds, Glass, Zoo, selecting a smaller value for α is helpful because it will force the neural network to focus on creating a boundary between classes but not on contracting clusters. On the contrary, if most of the points from two classes are distributed in the same region, e.g., BT, selecting a larger α will force the neural network to focus on contracting the clusters. Experiments suggest selecting α from 10^{-3} to 10^{3} is enough to balance the within-class similarity and between-class similarity.

TABLE I
CHARACTERISTICS OF DATA SET

Data Set Name	Number of Classes	Number of Samples	Number of Features	Imbalance Ratio
Ionosphere	2	351	34	1.79
WBCD	2	569	30	1.69
Fertility	2	100	9	7.33
Haberman	2	306	3	2.78
Parkinsons	2	195	22	3.06
PR	2	182	12	2.50
IRIS	3	150	4	1.00
Wine	3	178	13	1.48
CMC	3	1,473	9	1.53
Seeds	3	210	7	1.00
VC	3	310	6	2.50
Vehicle	4	846	18	1.10
UKM	4	403	5	2.58
Glass	6	214	9	8.44
BT	6	106	9	1.57
Zoo	7	101	16	10.25
Segment	7	2310	19	1.00

V. EXPERIMENTS

To validate the effectiveness of the NNP, seventeen well-known classification data sets were selected from the UCI machine learning repository (http://archive.ics.uci.edu/ml/) for the experiments, including Ionosphere Data Set(Ionosphere), Wisconsin Breast Cancer Diagnostic Data Set (WBCD), Fertility Data Set (Fertility), Haberman's Survival Data Set (Haberman), Parkinsons Data Set (Parkinsons), Planning Relax Data Set (PR), Iris Data Set (IRIS), Wine Data Set (Wine), Contraceptive Method Choice Data Set (CMC), Seeds Data Set (Seeds), Vertebral Column Data Set (VC), Statlog Vehicle Silhouettes Data Set (Vehicle), User Knowledge Modeling Data Set (UKM), Glass Identification Data Set (Glass), Breast Tissue Data Set (BT), Zoo Data Set (Zoo), and the Statlog Image Segmentation Data Set (Segment). Table I summarizes the characteristics of these data sets, including: name of data set, number of classes, number of samples, number of features, and the imbalance ratio (number of samples in the majority class relative to the number of samples in the minority class)

We used accuracy and average f-measure (Avg. FM) to evaluate the results of our experiments. We evaluated accuracy from the test data set using (9), shown at the bottom of this page. This accuracy measure indicates a method's generalization capability. Average f-measure is a measure that considers both precision and recall in computing its score. The f-measure reaches its best score at 1 and worst score at 0. We calculated the average f-measure of the classes in a data set using

$$\text{Avg. FM} = \frac{\sum_{i=1}^{N} \left(2 \times \frac{\text{precision}_i \times \text{recall}_i}{\text{precision}_i + \text{recall}_i}\right)}{N} \times 100 \qquad (10)$$

where N is the number of classes.

For each data set, all of the features were normalized to the same scale using min-max normalization to eliminate the influence of different scales between features. The min-max normalization is adopted because it is able to constrain features to a compact range $[0, 1]$ and thus puts the neural inputs and outputs into the same scale when a sigmoid function is adopted. Features were re-scaled using

$$f' = \frac{f - f_{\min}}{f_{\max} - f_{\min}} \qquad (11)$$

where f is the original value of the feature, $f_{\max}$ represents the maximum value of the feature, and $f_{\min}$ represents the minimum value.

A tenfold cross-validation was adopted to perform a reliable comparison and analysis. First, the original data set was divided into ten subsets by way of allocating samples from each class to each of the subsets randomly and uniformly. After that, one of the subsets was used as the testing set, while the other nine subsets were used as training models. Following that, another subset performed the role of a testing set, while the other nine subsets were used for training. This procedure was repeated ten times until each subset had been used as the test data set.

A. Partition Space

We used a three-layered MLP neural network with 20 hidden neurons and two output neurons to illustrate the distribution in the partition space of nearest neighbor partitioning. Particle Swarm Optimization (PSO) [35] was used in the experiments as the optimizer. For comparison, we also show the distribution of points projected using principal component analysis (PCA), projecting the points onto the 2-D subspace spanned by the first two principal components.

Fig. 7 illustrates the data space (projected onto the two principal components), and the partition space in NNP after optimization for the data sets VC, Vehicle, and UKM. Since the selection of the first two components ignores the low score's dimensions and loses information, the data space reduced by PCA can only be viewed as a low-dimensional approximation to the original data space. It is difficult to differentiate points that come from different classes in that data space. By contrast, it is easy to differentiate points from different classes in the partition space obtained by NNP. The points in the same class are put together to form a cluster, while points from different classes are disambiguated from each other.

The NNP generates flexible partitions and arbitrarily shaped clusters and boundaries in partition space. Furthermore, the differentiable clusters shown in the two-dimensional case suggest that NNP can also be used for visualizing data and reducing dimensions.

$$\text{Accuracy} = \frac{\text{number of correctly classified samples in test data set}}{\text{number of total samples in test data set}} \times 100 \qquad (9)$$

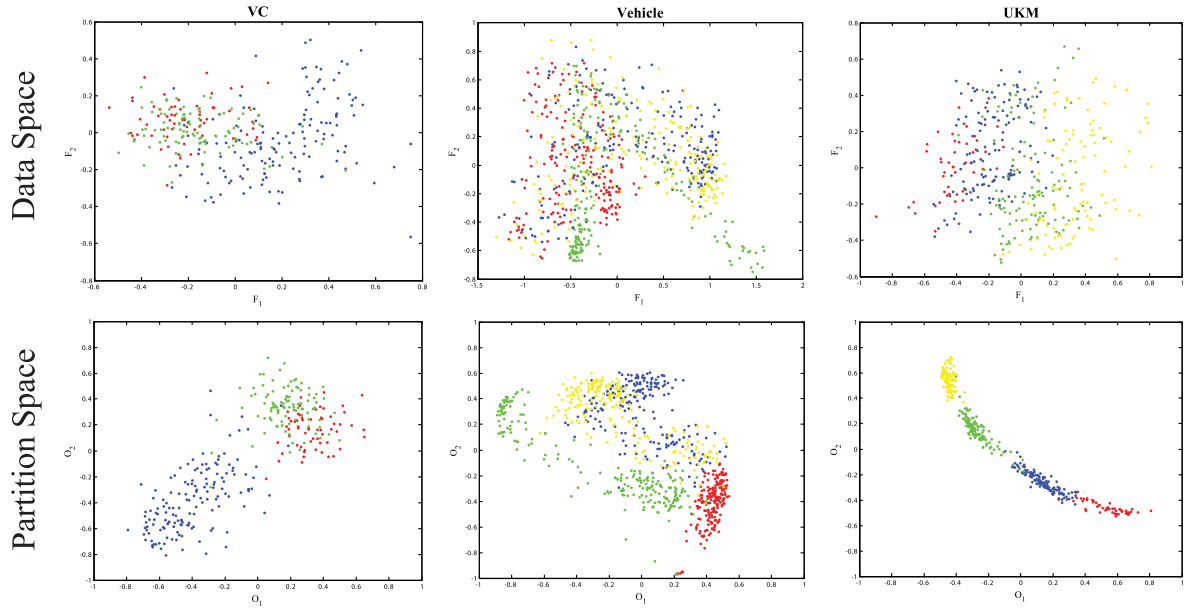


Fig. 7. Comparison of data space whose dimension is reduced by PCA, and the partition space in NNP after optimization.

B. Performance

We conducted experiments to compare 7 other classifiers with NNP, including four neural-network classifiers, k-Nearest Neighbor, Naive Bayes, and Support Vector Machines. Those algorithms are listed below:

1) Traditional neural network classifier (Traditional).
2) SoftMax neural network classifier (SoftMax) [16].
3) Error correcting output codes (ECOC) [17].
4) Floating-centroids method (FCM) [26].
5) Naive Bayes (NB).
6) k-Nearest Neighbor (kNN).
7) Support vector machines (SVM) [7].

For a fair comparison, the parameters of all compared methods were adjusted carefully by trial and error on each data set to achieve the optimal performance. For all five neural-network based classifiers (Traditional, SoftMax, ECOC, FCM, NNP), a three-layered MLP neural network with 20 hidden neurons and sigmoid activation function was selected.

1) The traditional neural-network classifier used a simple one-output method for binary classification and one-per-class method for multi-classification. Bayesian regularization was used for optimizing the weights and biases [31].
2) The SoftMax neural-network classifier used the SoftMax function at the output layer of the neural network classifier. Bayesian regularization was used for optimizing the weights and biases.
3) For the Error-correcting output code, the exhaustive code was used as the coding method. Bayesian regularization was used for optimizing the weights and biases.
4) For the Floating-centroids method, the parameter m was fixed at $2N$ while k was selected from $\{1.5N, 2N, \ldots, 4.5N\}$. Particle swarm optimization was used to optimize the weights and biases [35].
5) For KNN, the number of nearest neighbors, K, was selected from $\{1, 3, \ldots, 25\}$.
6) For SVM, the radial basis function was used as the kernel. Its cost parameter C was selected from $\{2^{-3}, 2^{-4}, \ldots, 2^{11}\}$ and the kernel parameter γ was selected from $\{2^{-11}, 2^{-10}, \ldots, 2^{3}\}$. It was trained by the sequential minimal optimization method [36].
7) For NNP, the dimension of partition space DPS was selected from 1 to 30, and the number of nearest neighbors L was selected from 1 to 50. The discrimination weight α was selected from 10^{-2} to 10^{1}. NNP was trained by particle swarm optimization.

The optimizer used for NNP was particle swarm optimization with parameters: Max Generation = 10 000, Population Size = 20, $\varphi 1 = 1.8$, $\varphi 2 = 1.8$, and VMAX = 0.4. For a fair comparison, the FCM also adopts these parameters for training.

The neural-network classifiers (including Traditional, SoftMax, and ECOC) were trained by Bayesian regularization and tested using MATLAB. The FCM and NNP were coded in C and tested on a machine with a Linux operating system. The other classifiers (i.e., k-Nearest Neighbors, Naive Bayes, and Support Vector Machines) were tested using WEKA [37].

Tables II and III provide a comparison of accuracy and average f-measure for all the data sets and methods. Since the differences between many of the results are not significant, the average rank of each method is further reported in Fig. 8. The rank of each method was normalized to [0, 1] on each data set, where 0 represents the worst method and 1 represents the best method. Then, the ranks on all data set were averaged for each method.

Several interesting findings can be obtained by analyzing the tables and figures. The NNP exhibits the best average rank on accuracy. Specifically, it achieves the best accuracy on eight data sets, including WBCD, Parkinsons, IRIS, Wine, Seeds, VC, Zoo, and Segment. Moreover, the improvement on average f-measure is more substantial than the improvement on accuracy, which shows that NNP has a more comprehensive

TABLE II

ACCURACY. FOR EACH DATA SET, THE BEST RESULT (AMONG THE COMPETING ALGORITHMS) IS HIGHLIGHTED

	Traditional	SoftMax	ECOC	FCM	NB	kNN	SVM	NNP
Ionoshpere	88.06(±5.41)	N/A	N/A	•**94.17(±4.80)**	80.16(±4.91)	85.00(±4.93)	93.33(±6.03)	93.89(±3.88)
WBCD	93.13(±5.14)	N/A	N/A	94.90(±3.15)	87.15(±6.52)	90.84(±6.07)	94.19(±3.02)	•**95.60(±1.91)**
Fertility	78.18(± 9.77)	N/A	N/A	•**83.64(±3.83)**	78.18(±6.36)	82.73(±5.16)	80.00(±8.35)	79.09(±13.59)
Haberman	72.81(± 4.43)	N/A	N/A	73.75(±3.67)	73.44(±4.72)	•**74.06(±2.57)**	73.75(±3.36)	71.56(±11.17)
Parkinsons	87.00(± 12.74)	N/A	N/A	87.00(±7.53)	67.00(±14.76)	88.00(±9.49)	84.50(±12.12)	•**90.00(±11.06)**
PR	•**68.42(± 0.00)**	N/A	N/A	65.79(±6.20)	61.58(±9.63)	•**68.42(±0.00)**	•**68.42(±0.00)**	62.11(±8.15)
IRIS	94.67(± 6.89)	96.67(± 5.67)	93.33(±6.29)	97.33(±4.66)	95.33(±4.50)	96.67(±4.71)	96.00(±5.62)	•**99.33(±2.11)**
Wine	97.78(±3.88)	98.33(±3.75)	97.78(±2.87)	•**98.89(±2.34)**	97.22(±3.93)	98.33(±2.68)	•**98.89(±2.34)**	•**98.89(±2.34)**
CMC	53.56(±3.69)	52.55(±2.45)	51.21(±2.69)	•**55.23(±2.90)**	49.06(±3.59)	49.46(±3.54)	54.09(±3.75)	54.16(±3.66)
Seeds	93.33(±6.02)	92.86(±7.19)	94.29(±7.38)	95.24(±5.02)	89.05(±11.68)	91.90(±9.80)	94.29(±7.03)	•**96.19(±4.92)**
VC	77.74(±3.86)	84.19(±5.15)	79.35(±7.78)	85.16(±6.31)	83.23(±8.16)	78.39(±8.61)	82.58(±7.32)	•**85.48(±6.32)**
Vehicle	81.74(±1.98)	83.26(±2.14)	79.88(±3.05)	79.19(±3.78)	46.05(±5.00)	71.63(±4.15)	•**85.35(±2.34)**	83.02(±2.46)
UKM	92.38(±5.12)	91.67(±6.66)	93.33(±2.46)	95.95(±1.96)	87.62(±4.87)	83.57(±4.82)	•**96.90(±1.61)**	95.24(±2.97)
Glass	65.27(±8.56)	61.94(±13.91)	59.04(±15.59)	•**66.58(±20.54)**	41.88(±6.56)	65.72(±11.16)	65.74(±10.42)	66.11(±9.91)
BT	67.86(±11.79)	72.14(±14.85)	65.00(±12.80)	•**73.57(±10.13)**	65.71(±8.78)	71.43(±11.66)	70.71(±9.79)	71.43(±14.29)
Zoo	94.00(±5.84)	94.00(±5.84)	94.00(±5.84)	93.33(±7.03)	94.00(±5.84)	94.00(±5.84)	94.00(±5.84)	•**94.67(±5.26)**
Segment	96.67(±1.78)	96.54(±1.43)	96.17(±1.38)	95.70(±1.71)	80.18(±1.78)	97.18(±1.70)	96.80(±1.31)	•**97.72(±1.35)**

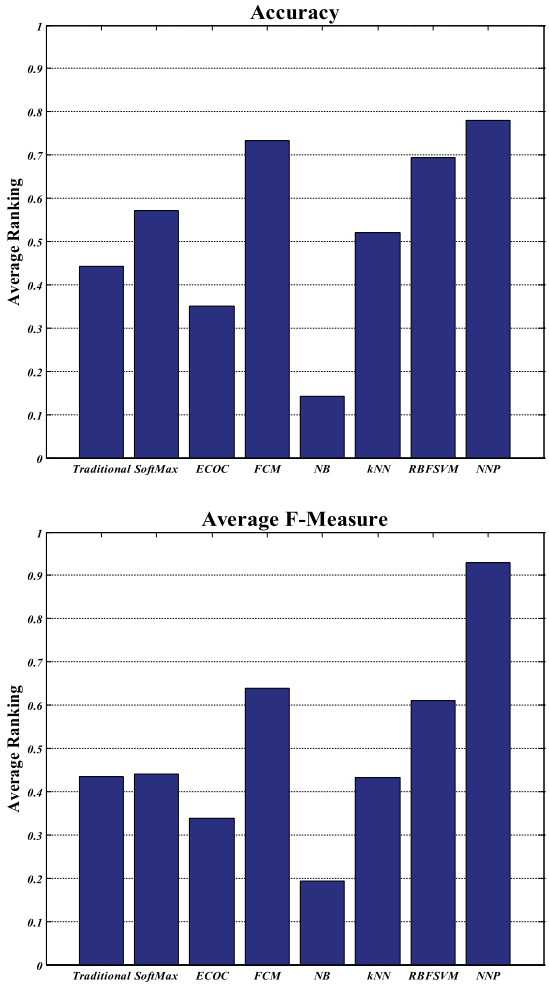


Fig. 8. Average rank of each method (based on Tables II and III). The rank was normalized to [0, 1] on each data set, where 0 represents the worst method and 1 represents the best method. For each method, the ranks on all data set are averaged and reported.

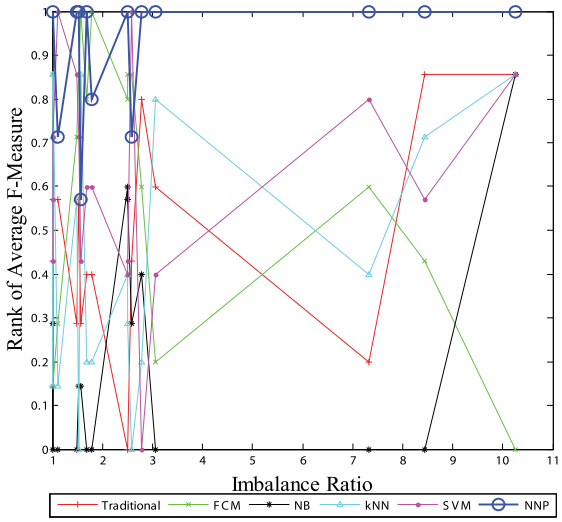


Fig. 9. Change of rank of average f-Measure with the increase of imbalance ratio for Traditional, FCM, NB, kNN, SVM, and NNP.

and balanced precision and recall for each class. The introduction of class weight also improves NNP's performance on imbalanced data sets. This phenomenon is clearly illustrated in Fig. 9, which shows the rank of average f-Measure versus imbalance ratio. It can be observed that NNP ranks higher on data sets with a higher imbalance ratio.

The NNP achieves better performance than FCM on more data sets since it removes the dependence on cluster centroids. All of the training points in the partition space can be viewed as centroids. NNP avoids the tendency for sphere-like boundaries like those found in FCM, and generates arbitrarily shaped boundaries instead.

Furthermore, it should be noted that NNP provides better results than kNN on most data sets. If the neural-network model in NNP is a transparent model, simply copying inputs straightforwardly to outputs, the NNP classification degenerates into k-nearest neighbor classification, performed on a normalized and constrained original data set. The results indicate that the neural-network nonlinear mapping from the original data space to partition space makes it easier to classify samples by kNN. NNP increases the similarity of samples that

TABLE III
AVERAGE F-MEASURE. FOR EACH DATA SET, THE BEST RESULT (AMONG THE COMPETING ALGORITHMS) IS HIGHLIGHTED

	Traditional	SoftMax	ECOC	FCM	NB	kNN	SVM	NNP
Ionoshpere	86.18(±6.42)	N/A	N/A	•**93.44(±5.43)**	79.34(±4.79)	82.04(±6.28)	92.45(±6.88)	93.21(±4.31)
WBCD	92.73(±5.13)	N/A	N/A	94.53(±3.24)	86.03(±6.70)	90.23(±6.16)	93.78(±3.08)	•**95.28(±2.00)**
Fertility	55.72(±14.67)	N/A	N/A	57.66(±17.11)	43.81(±2.10)	56.21(±15.47)	57.75(±16.42)	•**64.93(±21.57)**
Haberman	59.42(±8.02)	N/A	N/A	55.93(±9.61)	55.23(±10.97)	53.34(±7.88)	53.03(±11.42)	•**64.41(±13.54)**
Parkinsons	83.12(±16.57)	N/A	N/A	76.75(±17.39)	64.49(±16.72)	84.60(±12.52)	79.02(±16.79)	•**85.57(±16.07)**
PR	40.63(±0.00)	N/A	N/A	44.46(±7.76)	41.97(±7.60)	40.65(±0.00)	40.65(±0.00)	•**53.35(±10.20)**
IRIS	94.57(±7.01)	96.60(±5.78)	93.22(±6.40)	97.26(±4.82)	95.24(±4.64)	96.65(±4.73)	95.93(±5.73)	•**99.33(±2.13)**
Wine	97.85(±3.69)	98.33(±3.77)	97.78(±2.88)	98.88(±2.36)	97.32(±3.86)	98.41(±2.56)	98.93(±2.27)	•**98.97(±2.16)**
CMC	50.84(±4.88)	50.49(±2.89)	49.28(±3.16)	52.83(±2.55)	48.45(±3.28)	47.70(±4.34)	51.38(±4.22)	•**53.85(±3.69)**
Seeds	93.32(±6.01)	92.85(±7.15)	94.26(±7.43)	95.18(±5.08)	88.89(±11.63)	91.82(±9.90)	94.19(±7.14)	•**96.16(±4.96)**
VC	72.50(±5.72)	79.98(±6.31)	73.59(±9.61)	81.21(±7.77)	79.22(±9.35)	75.17(±9.84)	77.74(±8.35)	•**82.11(±7.60)**
Vehicle	81.88(±2.14)	83.28(±2.19)	80.10(±3.12)	78.54(±4.17)	42.50(±5.79)	70.57(±4.59)	•**85.47(±2.37)**	82.92(±2.56)
UKM	92.73(±5.03)	85.22(±14.79)	93.26(±2.98)	96.11(±2.03)	87.45(±6.32)	79.26(±8.11)	•**96.81(±1.84)**	94.88(±3.44)
Glass	60.74(±12.57)	51.73(±16.27)	52.83(±19.89)	55.52(±20.84)	39.84(±11.48)	60.08(±15.43)	56.49(±15.94)	•**67.30(±12.87)**
BT	62.88(±10.34)	66.14(±17.03)	60.13(±11.00)	•**69.78(±11.70)**	61.14(±9.67)	67.10(±10.85)	65.77(±8.29)	65.94(±16.12)
Zoo	86.88(±12.98)	86.64(±13.27)	86.88(±12.98)	86.45(±13.54)	86.88(±12.98)	86.88(±12.98)	86.88(±12.98)	•**87.61(±12.14)**
Segment	96.66(±1.79)	96.55(±1.40)	96.16(±1.37)	95.72(±1.69)	78.03(±2.34)	97.18(±1.70)	96.81(±1.31)	•**97.72(±1.35)**

come from the same class, and decreases the similarity of samples from different classes.

VI. CONCLUSION

In this paper, a nearest neighbor partitioning method is proposed for the purpose of improving the performance of a neural-network classifier. During the optimization process, each neural network maps the data points into a partition space. The quality of the mapping is evaluated using nearest neighbors, favoring dense clustering of points in the same class, with different class clusters separated by clear decision boundaries. Based on the resulting neural network, and the distribution of mapped training samples, the category of any new sample is dictated by its nearest neighbors in the partition space.

There is no need for the concept of a centroid in NNP. The categorization ability of the neural network is determined not by a handful of centroids, but rather by the distribution of training samples. A neural network need only map points in the same class close together, and put points from different classes as far away from each other as possible. In this manner, the NNP effortlessly generates flexible partitions and decision boundaries. That, in its turn, improves the chances of pinpointing optimal neural networks in the course of optimization.

We evaluated the performance of NNP on seventeen well-known classification data sets, measuring the method's accuracy and average f-measure. We compared those results with accuracy and average f-measure of seven other classification methods. Our experimental results indicate that NNP outperforms its competitors on the majority of the data sets by employing this novel approach. NNP has a more comprehensive and balanced precision and recall for each class, which yields improved performance of neural-network classifiers on imbalanced data. Furthermore, according to the results obtained by analyzing the parameters involved in the NNP, the difference in accuracy between two dimensions and higher dimensions is very small. This suggests that the proposed method is also a promising dimension-reduction technique for data visualization.

It should be noted that the proposed method focuses on the nature of the processing in partition space, after being mapped by the neural network. Therefore, in future work, the impacts of different neural network structures on the performance of NNP will be studied to further investigate its effectiveness. Furthermore, regularization strategies could be incorporated into NNP to improve its generalizability.

The NNP adopts k-nearest neighbors as a preliminary method to predict the label of unknown samples according to the distribution of mapped training samples in partition space. Finally, since NNP maps samples to a new distribution and makes the clusters from different classes more distinguishable, it is possible for other classifiers (e.g., SVM, instead of kNN) to classify samples based on the remapped data.

REFERENCES

[1] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[2] Y. Freund, "Boosting a weak learning algorithm by majority," *Inf. Comput.*, vol. 121, no. 2, pp. 256–285, Sep. 1995.

[3] H. Lu, R. Setiono, and H. Liu, "Effective data mining using neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 8, no. 6, pp. 957–961, Dec. 1996.

[4] B. B. Misra, S. Dehuri, P. K. Dash, and G. Panda, "A reduced and comprehensible polynomial neural network for classification," *Pattern Recognit. Lett.*, vol. 29, no. 12, pp. 1705–1712, 2008.

[5] D. Gao, Y. Ji, "Classification methodologies of multilayer perceptrons with sigmoid activation functions," *Pattern Recognit.*, vol. 38, no. 10, pp. 1469–1482, 2005.

[6] Y. F. Hassan, "Rough sets for adapting wavelet neural networks as a new classifier system," *Appl. Intell.*, vol. 35, no. 2, pp. 260–268, 2011.

[7] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag, 1995.

[8] J. López and J. A. K. Suykens, "First and second order SMO algorithms for LS-SVM classifiers," *Neural Process. Lett.*, vol. 33, no. 1, pp. 31–44, 2011.

[9] K. S. Chua, "Efficient computations for large least square support vector machine classifiers," *Pattern Recognit. Lett.*, vol. 24, nos. 1–3, pp. 75–80, 2003.

[10] A. Castaño, F. Fernández-Navarro, C. Hervás-Martínez, and P. A. Gutierrez, "Neuro-logistic models based on evolutionary generalized radial basis function for the microarray gene expression classification problem," *Neural Process. Lett.*, vol. 34, no. 2, pp. 117–131, 2011.

[11] S.-Y. An, J.-G. Kang, W.-S. Choi, and S.-Y. Oh, "A neural network based retrainable framework for robust object recognition with application to mobile robotics," *Appl. Intell.*, vol. 35, no. 2, pp. 190–210, 2011.

[12] E. Avci, "An expert target recognition system using a genetic wavelet neural network," *Appl. Intell.*, vol. 37, no. 4, pp. 475–487, 2012.

[13] S. N. Yaakob and L. Jain, "An insect classification analysis based on shape features using quality threshold ARTMAP and moment invariant," *Appl. Intell.*, vol. 37, no. 1, pp. 12–30, 2012.

[14] Y. V. Venkatesh and S. K. Raja, "On the classification of multispectral satellite images using the multilayer perceptron," *Pattern Recognit.*, vol. 36, no. 9, pp. 2161–2175, 2003.

[15] Q. Yu, R. Yan, H. Tang, K. C. Tan, and H. Li, "A spiking neural network system for robust sequence recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 621–635, Mar. 2016.

[16] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," *Neurocomputing: Algorithms, Architectures and Applications*. Berlin, Germany: Springer-Verlag, 1990, pp. 227–236.

[17] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *J. Artif. Intell. Res.*, vol. 2, no. 1, pp. 263–286, 1995.

[18] J.-D. Zhou, X.-D. Wang, H.-J. Zhou, Y.-H. Cui, and S. Jing, "Coding design for error correcting output codes based on perceptron," *Opt. Eng.*, vol. 51, no. 5, p. 057202, 2012.

[19] M. A. Bagheri, Q. Gao, and S. Escalera, "A genetic-based subspace analysis method for improving error-correcting output coding," *Pattern Recognit.*, vol. 46, no. 10, pp. 2830–2839, 2013.

[20] M. T. Musav, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels, "On the training of radial basis function classifiers," *Neural Netw.*, vol. 5, no. 4, pp. 595–603, 1992.

[21] G. S. Babu and S. Suresh, "Sequential projection-based metacognitive learning in a radial basis function network for classification problems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 2, pp. 194–206, Feb. 2013.

[22] L. Zhang, K. Li, H. He, and G. W. Irwin, "A new discrete-continuous algorithm for radial basis function networks construction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 11, pp. 1785–1798, Nov. 2013.

[23] A. M. Patrikar, "Approximating Gaussian mixture model or radial basis function network with multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 7, pp. 1161–1166, Jul. 2013.

[24] J. Tian, M. Li, F. Chen, and N. Feng, "Learning subspace-based RBFNN using coevolutionary algorithm for complex classification tasks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 1, pp. 47–61, Jan. 2016.

[25] M. Robnik-Šikonja, "Data generators for learning systems based on RBF networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 5, pp. 926–938, May 2016.

[26] L. Wang *et al.*, "Improvement of neural network classifier using floating centroids," *Knowl. Inf. Syst.*, vol. 31, no. 3, pp. 433–454, 2012.

[27] G. Czibula, I. G. Czibula, and R. D. Găceanu, "Intelligent data structures selection using neural networks," *Knowl. Inf. Syst.*, vol. 34, no. 1, pp. 171–192, 2011.

[28] L. Zhang, L. Wang, X. Wang, K. Liu, and A. Abraham, "Research of neural network classifier based on FCM and PSO for breast cancer classification," in *Hybrid Artificial Intelligent Systems* (Lecture Notes in Computer Science), vol. 7208. Berlin, Germany: Springer-Verlag, 2012, pp. 647–654.

[29] I. Czarnowski and P. Jędrzejowicz, "Agent-based approach to RBF network training with floating centroids," in *Proc. 4th Int. Conf. Comput. Collect. Intell.*, 2012, pp. 453–462.

[30] J. Zhou, L. Chen, C. L. P. Chen, Y. Zhang, and H.-X. Li, "Fuzzy clustering with the entropy of attribute weights," *Neurocomputing*, vol. 198, pp. 125–134, Jul. 2016.

[31] F. D. Foresee and M. T. Hagan, "Gauss–Newton approximation to Bayesian learning," in *Proc. Int. Conf. Neural Netw.*, vol. 3. Jun. 1997, pp. 1930–1935.

[32] L. Wang, B. Yang, and A. Abraham, "Distilling middle-age cement hydration kinetics from observed data using phased hybrid evolution," *Soft Comput.*, in press, doi: 10.1007/s00500-015-1723-4.

[33] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.

[34] E. Kreyszig, *Advanced Engineering Mathematics*, 4th ed. New York, NY, USA: Wiley, 1979.

[35] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, 1995, pp. 39–43.

[36] J. C. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-98-14, 1998.

[37] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.

Lin Wang was born in Shandong, China, in 1983. He received the B.Sc. and master's degrees in computer science and technology from the University of Jinan, Jinan, China, in 2005 and 2008, respectively, and the Ph.D. degree in computer science and technology from the School of Computer Science and Technology, Shandong University, Jinan, in 2011.

He is currently an Associate Professor with the Shandong Provincial Key Laboratory of Network Based Intelligent Computing, University of Jinan. His current research interests include classification, hybrid computational intelligence, and mathematical modeling.

Bo Yang was born in 1965.

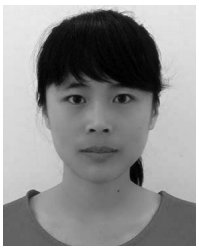
He is currently a Professor with the University of Jinan, Jinan, China, and the President of Linyi University, Linyi, China. He is the Director of the Shandong Provincial Key Laboratory of Network Based Intelligent Computing and also acts as the Associate Director of the Shandong Computer Federation. He has authored numerous papers. His current research interests include computer networks, artificial intelligence, machine learning, knowledge discovery, and data mining.

Prof. Yang is also a member of the IEEE Computational Intelligence Society. He received some of important scientific awards in his research area. He acts as the Chair of the ACM Jinan Chapter.

Yuehui Chen received the Ph.D. degree in electrical engineering from Kumamoto University, Kumamoto, Japan, in 2001.

He was the Senior Researcher with Memory-Tech Corporation, Tokyo, Japan, from 2001 to 2003. Since 2003, he has been a member of the Faculty of Electrical Engineering with the University of Jinan, Jinan, China, where he is currently the Head of the Laboratory of Computational Intelligence. He is currently the Professor and Vice President of the University of Jinan. He has authored or co-authored over 70 technique papers. His current 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.

Prof. Chen is a member of the IEEE Systems, Man and Cybernetics Society, the Computational Intelligence Society, the Young Researchers Committee of the World Federation on Soft Computing, and the CCF Young Computer Science and Engineering Forum of China.



Xiaoqian Zhang was born in 1989. She received the B.Sc. degree in computer science and technology from the University of Jinan, Jinan, China, in 2011, where she is currently pursuing the master's degree with the Shandong Provincial Key Laboratory of Network Based Intelligent Computing.

Her current research interests include neural network and classification.

Jeff Orchard received the B.Math. degree in applied mathematics from the University of Waterloo, Waterloo, ON, Canada, the M.Sc. degree in applied mathematics from the University of British Columbia, Vancouver, BC, Canada, and the Ph.D. degree in computing science from Simon Fraser University, Burnaby, BC, Canada, in 2003.

He has been a Faculty Member with the David R. Cheriton School of Computer Science, University of Waterloo. His research focuses on computational neuroscience and using mathematical models and computer simulations of neural networks in an effort to understand how the brain works. He has also published research in image processing and medical imaging. His current research interests include learning methods for deep perceptual networks, network dynamics, decision-making, spatial navigation, and neural population coding.