Wen et al. EURASIP Journal on Advances in Signal Processing (2016) 2016:57
DOI 10.1186/s13634-016-0357-8

EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

An incremental learning algorithm for the hybrid RBF-BP network classifier

Hui Wen, Weixin Xie, Jihong Pei* and Lixin Guan

Abstract

This paper presents an incremental learning algorithm for the hybrid RBF-BP (ILRBF-BP) network classifier. A potential function is introduced to the training sample space in space mapping stage, and an incremental learning method for the construction of RBF hidden neurons is proposed. The proposed method can incrementally generate RBF hidden neurons and effectively estimate the center and number of RBF hidden neurons by determining the density of different regions in the training sample space. A hybrid RBF-BP network architecture is designed to train the output weights. The output of the original RBF hidden layer is processed and connected with a multilayer perceptron (MLP) network; then, a back propagation (BP) algorithm is used to update the MLP weights. The RBF hidden neurons are used for nonlinear kernel mapping and the BP network is then used for nonlinear classification, which improves classification performance further. The ILRBF-BP algorithm is compared with other algorithms in artificial data sets and UCI data sets, and the experiments demonstrate the superiority of the proposed algorithm.

Keywords: Radial basis function (RBF), Back propagation (BP), Incremental learning, Hybrid, Neural network

1 Introduction

In the field of pattern recognition and data mining, various methods and models are proposed to solve different problems. Existing methods can be divided into two levels including data level and algorithmic level. The data levels are mainly concerned with various sampling techniques [1]. The algorithmic level tried to apply or improve varieties of existing traditional learning algorithms such as fuzzy clustering [2], Markovian jumping system [3–5], k-nearest neighbors [6], and neural network, where single-layer feed-forward networks (SLFNs) have been intensively studied in the past several decades and applied to solve various problems in different fields, such as image recognition [7], signal processing [8], disease prediction [9], and industrial fault diagnosis [10]; in particular, radial basis function (RBF) neural networks offer an effective mechanism for nonlinear mapping and classification. In a typical RBF network, the number of hidden neurons is assigned a priori [11, 12], which leads to poor adaptability for different sample sets. Several sequential learning algorithms have been proposed to determine proper sizes of RBF network architectures. A resource allocation network (RAN) for constructing the RBF network is proposed in [13], which uses the novelty of incoming data as the learning strategy. A RAN algorithm based on an extended Kalman filter (RANEKF) is proposed in [14], which uses the extended Kalman filter algorithm instead of the least mean squares (LMS) algorithm. In [15], a minimal resource allocation network (MRAN) is proposed, which is allowed for the deletion of the previous center. The deletion strategy is based on the overall contribution of each hidden unit to the network output. A sequential learning algorithm for growing and pruning the RBF (GAP-RBF) is proposed in [16, 17]; this algorithm uses the significance of neurons as the learning strategy. In [18], a Gaussian mixture model (GMM) to approximate the generalized growing and pruning evaluation formula is proposed; the GMM can be used for problems with a high-dimensional probability density distribution. In [19], an error correction (ErrCor) algorithm is used for function approximation; this algorithm can achieve a desired error rate with fewer RBF units. Other methods have also been established to identify

* Correspondence: jhpei@szu.edu.cn
ATR Key Lab of National Defense, Shenzhen University, 518060 Shenzhen, China

a proper architecture while maintaining a desired accuracy [20–22].

Support vector machines (SVMs), which are maximal margin classifiers, can also be used to train SLFNs. RBFs and SVMs differ in that at the output layer, a SVM employs convex optimization to find an optimal linear classifier, whereas the output weights of RBF network are typically estimated by a linear least squares algorithm, such as the LMS or recursive least squares (RLS) algorithm. Regarding other training SLFNs, extreme learning machines (ELMs) are proposed in [23]; ELMs choose random hidden neuron parameters and calculate the output weights with the least squares algorithm. This method can achieve a fast training speed. Subsequently, an online sequence extreme learning machine (OS-ELM) algorithm that can learn one by one and data blocks of the input samples is proposed in [24]. In ELMs, the number of hidden nodes is assigned a priori, and many nonoptimal nodes may exist; thus, in [25–28], several types of growing and pruning techniques based on ELMs are proposed to effectively estimate the number of hidden neurons.

All of the algorithms for training SLFNs consist of two stages: (1) suitable feature mapping and (2) output weight adjustment. To train SLFNs efficiently, in this paper, a potential function is introduced in the feature mapping stage to train the sample space, and an incremental learning method of constructing RBF hidden neurons is proposed. Note that although the sequence learning RBF algorithms can also generate RBF hidden neurons automatically, because of the lack of global information in the sample space, the adaptability of complex sample space may be poor. In contrast to GAP-RBF, the proposed method does not require an assumption that the input samples obey a unified distribution. Furthermore, it does not need to fit the input sample distribution, such as the algorithm proposed in [18]. The proposed method utilizes global information about each class of training sample space and can generate RBF hidden neurons incrementally to adapt the sample space. By using a potential function to measure the density in each class of training sample space, the corresponding RBF hidden neurons that cover different sample areas can be established. The center of the Gaussian kernel function can be determined by learning the density of different regions in the training sample space. Once the width is given, a hidden neuron is generated and introduced into the RBF network, and a mechanism for eliminating the potentials of original samples is presented. This mechanism is ready for the next learning step, and thus, the RBF centers and number of hidden neurons can be effectively estimated. In this way, a suitable network size for RBF hidden layer that matches the complexity of the sample space can be built up. Thus, the proposed method solves the problem of dimension change from sample space mapping to feature space, and it reduces the restrictions on the sample sets, which is adaptable to more complex sample sets.

In this paper, a hybrid RBF-BP network architecture is designed for the output weight adjustment stage to further improve the generalization and classification performance. The output of the original RBF hidden layer is processed and connected with a new hidden layer, which means that the output of the original RBF hidden layer, the new hidden layer, and the output layer consists of a multilayer perceptrons (MLPs), and the output of the original RBF hidden layer is the input of the MLPs. Once the network architecture is established, a back propagation (BP) algorithm is used to update the weights of the MLPs. In the hybrid RBF-BP network, the RBF hidden neurons are used for nonlinear kernel mapping, the complexity of sample space is mapped onto the dimension of the BP network input layer, and the BP network is then used for nonlinear classification. The nonlinear kernel mapping can improve the separability of sample spaces, and a nonlinear BP classifier can then supply a better classification surface. In this manner, the improved network architecture combines the local response characteristics of the RBF network with the global response characteristics of the BP network, which simplifies the neuron number selection in the BP network hidden layer while further reducing the dependence on space mapping in the RBF hidden layer.

The incremental learning algorithm for the hybrid RBF-BP (ILRBF-BP), which is a batch learning algorithm, is proposed by combining the proposed incremental learning algorithm with the hybrid RBF-BP network architecture. In this paper, the performance of the ILRBF-BP algorithm is compared with other well-known learning algorithms, such as back propagation based on stochastic gradient descent (SGBP) [29], the RBF algorithm based on k-means clustering (KM-RBF) [12], GAP-RBF, SVM, and an ELM, on artificial data sets. To measure the unique features of the proposed method, the k-means clustering learning algorithm based on the hybrid RBF-BP network (KMRBF-BP) is also compared with ILRBF-BP on artificial data sets. Because SGBP and KM-RBF are not suitable for considering more complex problems, for multi-class data sets, in addition to batch learning algorithms, such as SVM and ELM, other well-known sequential algorithms, such as MRAN, GAP-RBFN, and OS-ELM, are also compared with the ILRBF-BP

algorithm. The results indicate that the ILRBF-BP algorithm can provide a higher classification accuracy with comparable complexity.

The remainder of this paper is organized as follows. Section 2 describes the principal ideas of the ILRBF-BP, followed by a summary of the algorithm. Section 3 presents the experimental results and performance comparisons with other existing batch and sequential algorithms. Section 4 provides the conclusions of this study.

2 Main concepts of the ILRBF-BP algorithm

In this section, the main concepts of the ILRBF-BP algorithm are described. First, we provide the problem definition of the basic RBF network and then present the incremental learning algorithm for constructing RBF hidden neurons. Then, a hybrid RBF-BP network architecture is designed, and the ILRBF-BP algorithm is summarized. Finally, a method of adjusting the output saturation for multi-class classification problem is proposed.

2.1 Problem definition

For a RBF network, the output can be given by

$$F(\mathbf{x}) = \sum_{k=1}^{K} \omega_k \phi_k(\mathbf{x}) \tag{1}$$

where

$$\phi_k(\mathbf{x}) = \exp\left(-\frac{1}{2\sigma_k^2}||\mathbf{x}-\mu_k||^2\right) \tag{2}$$

where K is the number of RBF hidden neurons; $\phi_k(\mathbf{x})$ is the response of the kth hidden node for an input vector $\mathbf{x}$, where $\mathbf{x} \in R^t$; ω_k is its connecting weight to the output node, which determines the classification surface; and μ_k and σ_k are the center and width of the kth hidden node, respectively, where $k = 1, 2, \ldots K$.

A RBF network can localize the input sample space, which maps input samples to the interior of the hypercube, and the localized area is near a vertex. The dimension of the hypercube is the number of RBF hidden neurons. Thus, when going through the RBF network, an input vector $\mathbf{x} \in R^t$ can be denoted as $f: R^t \rightarrow (0, 1]^K$. Figure 1 shows the results of mapping input samples going through the RBF hidden neurons, where the number of RBF hidden neurons is set as $K = 3$. In Fig. 1, we assume that every input sample vector is near the center of a RBF hidden neuron and that there is no overlap area covered by different RBF hidden neurons.

Figure 1 illustrates that in a RBF network, to achieve good training algorithms, an effective method of mapping the input sample space should be established, which means completing the estimation of the parameter set $\{K, \mu_k, \sigma_k\}_{k=1}^{K}$. Then, an effective classification surface is needed, which depends on output weight adjustment.

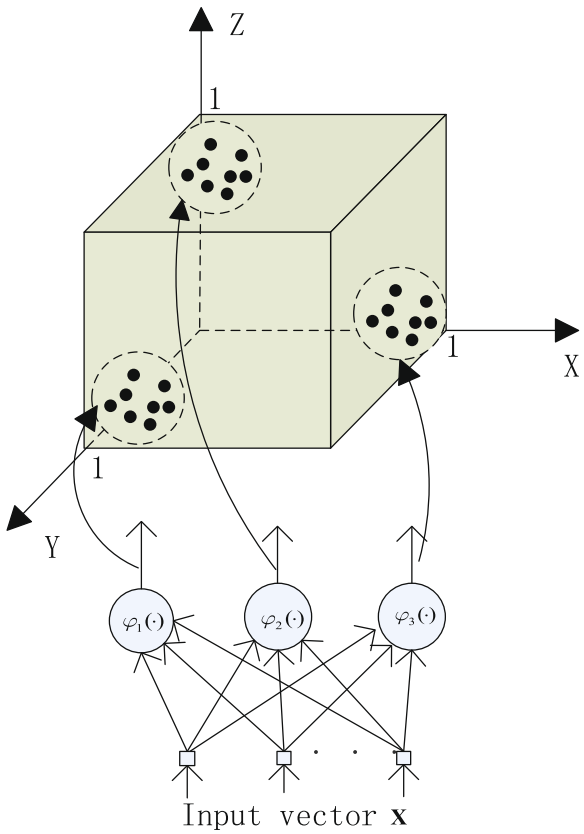

Fig. 1 Result of mapping input samples going through RBF hidden neurons

2.2 Incremental learning algorithm for constructing RBF hidden neurons

In the fields of data mining and pattern recognition, potential functions can be used for density clustering and image segmentation (IS) [30]. Several methods of constructing potential function are proposed in [31]; here, we choose the potential function

$$\gamma(\mathbf{x_1}, \mathbf{x_2}) = \frac{1}{1 + T \cdot d^2(\mathbf{x_1}, \mathbf{x_2})} \tag{3}$$

where $\gamma(\mathbf{x_1}, \mathbf{x_2})$ represents the interaction potential of two points $\mathbf{x_1}, \mathbf{x_2}$ in the input sample space, $d(\mathbf{x_1}, \mathbf{x_2})$ represents the distance measure, and T is a constant, which can be regarded as the distance weighting factor.

Given a training sample set S, where a specific label $\mathbf{y_i}, \mathbf{y_i} \in \{\mathbf{y_i}; i = 1, 2, \ldots h\}$ is attached to each sample vector $\mathbf{x}$

in S, h is the number of pattern class. Let S_i denote the set of feature vectors that are labeled $\mathbf{y_i}$, $S_i = \{\mathbf{x_1^i}, \mathbf{x_2^i}, ..., \mathbf{x_{N_i}^i}\}$, where N_i is the number of training samples in the ith pattern class. Thus, $S = \cup_{i=1}^{h} S_i$, $S_i \cap S_j = \emptyset, \forall\, i \neq j$. For a pair of samples $(\mathbf{x_u^i}, \mathbf{x_v^i})$ in S_i, its interaction potential can be denoted as

$$\gamma(\mathbf{x_u^i}, \mathbf{x_v^i}) = \frac{1}{1 + T \cdot d^2(\mathbf{x_u^i}, \mathbf{x_v^i})} \tag{4}$$

Let $\mathbf{x_v^i}$ be the baseline sample; then, the interaction potential of all other samples to $\mathbf{x_v^i}$ can be denoted as

$$\rho(\mathbf{x_v^i}) = \sum_{u=1, u\neq v}^{N_i} \gamma(\mathbf{x_u^i}, \mathbf{x_v^i}) \tag{5}$$

Therefore, the potentials of each sample in S_i is given by

$$\rho^i = \left\{\rho(\mathbf{x_1^i}), \rho(\mathbf{x_2^i}), ..., \rho\left(\mathbf{x_{N_i}^i}\right)\right\} \tag{6}$$

The potentials can be used to measure the density of different regions in the pattern class. Potentials are relatively large in the dense region, whereas they are relatively small in the sparse region. Once the potentials of each sample in S_i are given, the sample with the maximum potential can be selected, where it is assumed the sample is $\mathbf{x_p^i}$, that is,

$$\rho\left(\mathbf{x_p^i}\right) = \max\left\{\rho(\mathbf{x_1^i}), \rho(\mathbf{x_2^i}), ..., \rho\left(\mathbf{x_{N_i}^i}\right)\right\} \tag{7}$$

In a RBF network, the activation response of hidden neurons has local characteristics. The sample space is divided into different subspaces by establishing different Gaussian kernel functions. To generate valid Gaussian kernel functions, we find the most densely region in the sample space and then establish a Gaussian kernel to cover the region. For that purpose, the sample with the maximum potential is chosen as the center of Gauss kernel function, which is given below.

$$\mu_k = \mathbf{x_p^i} \tag{8}$$

where k refers to the number of RBF hidden neurons generated. To simplify the calculation, the width is fixed and selected by cross validation.

When a hidden neuron is established, it is necessary to eliminate the potentials of the region to find the next center in the remaining samples. This process can be updated by

$$\rho_{\text{new}}(\mathbf{x_v^i}) = \rho(\mathbf{x_v^i}) - \rho\left(\mathbf{x_p^i}\right) \cdot \exp\left(-\frac{1}{2\sigma_k^2} ||\mathbf{x_v^i} - \mathbf{x_p^i}||^2\right), \quad v = 1, 2, ...N_i \tag{9}$$

where $\mathbf{x_p^i}$ is the center of the current hidden neuron. For the potential value update process, Eq.(9) shows when a sample $\mathbf{x_v^i}$ is close to the center $\mathbf{x_p^i}$, the potential value of $\mathbf{x_v^i}$ is attenuated fast, whereas when a sample $\mathbf{x_v^i}$ is far away from the center, the potential value of $\mathbf{x_v^i}$ is attenuated slowly. When meeting the inequality

$$\max\left\{\rho_{\text{new}}(\mathbf{x_1^i}), \rho_{\text{new}}(\mathbf{x_2^i}), ..., \rho_{\text{new}}\left(\mathbf{x_{N_i}^i}\right)\right\} > \delta \tag{10}$$

a new hidden neuron is introduced into the RBF network and is ready to search for the next center; otherwise, the algorithm of constructing RBF hidden neurons in the current pattern class is over, where δ is a threshold.

The above process is called the incremental learning algorithm of constructing RBF hidden neurons. Figure 2 shows a schematic diagram of generating RBF hidden neurons incrementally, where the serial numbers in the training sample space represent the regions covered by different RBF hidden neurons. These covered regions transition from dense to sparse. The incremental learning algorithm of constructing RBF hidden neurons is summarized in Algorithm 1.

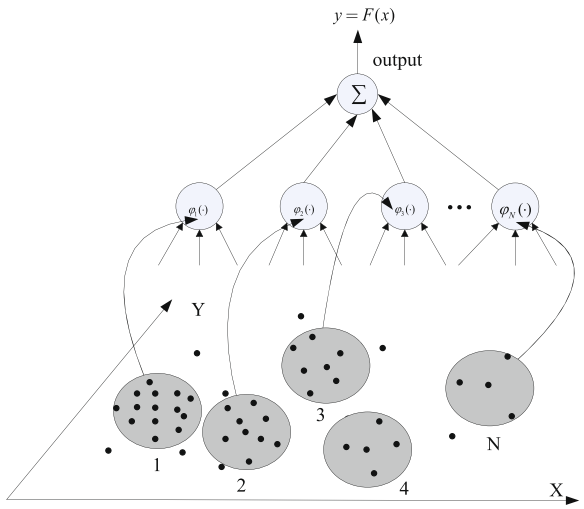

Fig. 2 Illustrative diagram of incrementally generating RBF hidden neurons

Algorithm 1. Incremental learning algorithm of constructing RBF hidden neurons

Initialize the number of RBF hidden neurons $k = 0$. Given the width σ and the distance weighting factor T. Given training samples $S = \bigcup_{i=1}^{h} S_i$, $S_i = \{\mathbf{x}_1^{\mathbf{i}}, \mathbf{x}_2^{\mathbf{i}}, ..., \mathbf{x}_{\mathbf{N_i}}^{\mathbf{i}}\}$. For each pattern class S_i, do

1. Compute the potential value of each sample according to Eq.(5);
2. Determine the maximum potential value of each sample according to Eqs.(6) and (7);
3. The number of RBF hidden neurons k counts plus 1. Use Eq.(8) to allocate a new hidden neuron center;
4. Eliminate the sample potential value of the region according to Eq.(9);
5. Set iteration termination condition

If $\max\{\rho_{new}(\mathbf{x}_1^{\mathbf{i}}), \rho_{new}(\mathbf{x}_2^{\mathbf{i}}), ..., \rho_{new}(\mathbf{x}_{\mathbf{N_i}}^{\mathbf{i}})\} > \delta$

Go to Step 2.

Else

The process of learning current pattern class is over. Go to learn other pattern classes.

EndIf

2.3 Hybrid RBF-BP network architecture

As noted above, in a typical RBF network, the output weights are typically estimated by a linear least squares algorithm, such as the LMS or RLS algorithm. In this section, we transform the linear least squares algorithm into a nonlinear algorithm. When classifying a problem, a nonlinear algorithm can supply a better classification surface to adapt the sample space. For that purpose, a hybrid RBF-BP network architecture is designed. The output of the RBF hidden neurons is processed and connected with a MLPs network, and then, the nonlinear BP algorithm is used to update the weights of the MLPs. The architecture of the hybrid RBF-BP network is shown in Fig. 3, which consists of four components:

1. The input layer, which consists of t source nodes, where t is the dimensionality of the input vector

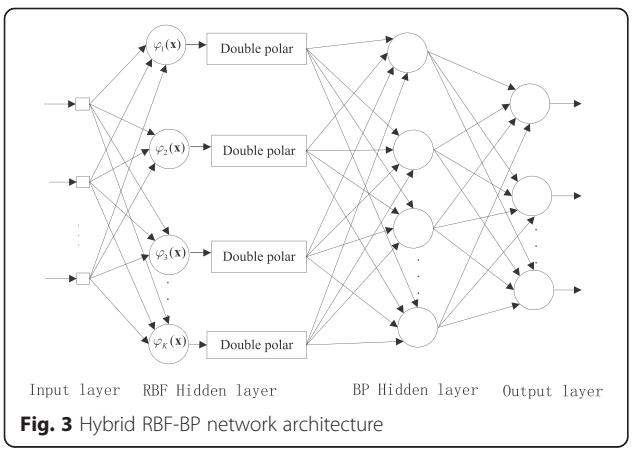

Fig. 3 Hybrid RBF-BP network architecture

2. The RBF hidden layer, which consists of a group of Gaussian kernel functions:

$$\phi_k(\mathbf{x}) = \exp\left(-\frac{1}{2\sigma_k^2}||\mathbf{x}-\mu_k||^2\right), \quad k = 1, 2, ...K \tag{11}$$

where μ_k and σ_k are the center and width of the hidden neuron, respectively, and K is the number of hidden neurons.

3. The BP hidden layer, which consists of the neurons between the RBF hidden layer and output layer. The induced local field $v_j^{(l)}$ for neuron j in layer l of the BP network is

$$v_j^{(l)} = \sum_i \omega_{ji}^{(l)} y_i^{(l-1)} \tag{12}$$

where $y_i^{(l-1)}$ is the output signal of the neuron i in the previous layer l-1 of the BP network and $\omega_{ji}^{(l)}$ is the synaptic weight of neuron j in layer l that is fed from neuron i in layer l-1. Assuming the use of a sigmoid function, the output signal of neuron j in layer l is

$$y_j^{(l)} = \phi_j(v_j) = a\tanh(bv_j) \tag{13}$$

where a and b are constants.

If neuron j is in the first BP network hidden layer, i.e., $l = 1$, set

$$y_j^{(0)} = g_j(\mathbf{x}) \tag{14}$$

where $g_j(\mathbf{x})$ is the double polar output of $\phi_j(\mathbf{x})$ and can be denoted as

$$g_j(\mathbf{x}) = 2 \cdot \phi_j(\mathbf{x}) - 1 \tag{15}$$

4. The output layer. Set L is the depth of the BP network, note the depth of the BP network is equal to the sum of the BP network input layer, the hidden layer, and the output layer, i.e., if $l = 1$, then $L = 3$, and the output can be given as

$$o_j = y_j^{(L)} \tag{16}$$

In Fig. 3, the double polar processing can ensure the validity of the BP network input. The hybrid RBF-BP network architecture is designed such that the RBF network has good stability, where the activation response in the RBF hidden neurons has local characteristics and maps the output value between 0 and 1. Thus, the original samples including outliers will be limited to a finite space. When the results of mapping the RBF hidden neurons are processed and used for the input of the BP network, the convergence rate of the BP algorithm can be increased and local minima can be avoided. For a BP network, the activation response in hidden neurons has global characteristics, especially those regions not fully displayed in the training set. Therefore, the hybrid RBF-BP network architecture is a reasonable model; it provides a new strategy that combines the local characteristics of the RBF network with the global characteristics of the BP network. In addition, the hybrid network simplifies the number of neurons in the BP hidden layer while further reducing the dependence on space mapping in the RBF hidden layer.

A single hidden layer MLP neural network with an input-output mapping can provide an approximate realization of any continuous mapping [32]. Combined with the above discussion, in the hybrid network, we set the number of BP network hidden layers as $l = 1$.

Combining the proposed incremental learning algorithm with the hybrid RBF-BP network architecture, the incremental learning RBF-BP (ILRBF-BP) algorithm is summarized in Algorithm 2.

Algorithm 2: The ILRBF-BP algorithm

1. Assign random initialized weights to each layer of the MLP network, initialize the BP iteration step m, set $a = 1.716$, $b = 2/3$.
2. Use the incremental learning algorithm of constructing RBF hidden neurons proposed in Algorithm 1.
3. Use Eqs.(11) and (15) to compute $g_j(\mathbf{x})$, let $g(\mathbf{x})$ be the input of the BP network, where

   $g(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x}), ..., g_K(\mathbf{x}))$.
4. Forward compute the BP network. Use Eqs.(12)-(14) and Eq.(16). Compute the error signal

   $$e_j = d_j - o_j$$

   where d_j is the jth element of the desired response vector $\mathbf{d}$.
5. Backward compute the BP network. Compute the local gradients of the network, which is denoted by

   $$\delta_j^{(l)} = \begin{cases} e_j^{(L)} \phi_j'\left(v_j^{(L)}\right) & \text{for neuron } j \text{ in output layer } L \\ \varphi_j'(v_j^{(l)}) \sum_k \delta_k^{(l+1)} \omega_{kj}^{(l+1)} & \text{for neuron } j \text{ in MLP hidden layer } l \end{cases}$$

   where $\phi_j'(\square)$ is the differentiation with respect to the argument. Adjust the synaptic weights of the network in layer l of MLP as shown below.

   $$\omega_{ji}^{(l)}(m+1) = \omega_{ji}^{(l)}(m) + \alpha\left[\omega_{ji}^{(l)}(m-1)\right] + \eta \delta_j^{(l)}(m) y_i^{(l-1)}(m)$$

   where α is the momentum constant and η is the learning rate.
6. Iteration. Iterate the forward and backward computations in Steps 4 and 5 by presenting new epochs of training examples to the network until the chosen stopping criterion is met.

2.4 Adjustment of the output label values

The ILRBF-BP algorithm can handle binary problems and multi-class problems. For multi-class classification problems, suppose that the observation data set is given as $\{\mathbf{x_n}, \mathbf{y_n}\}_{n=1}^{N}$, where $\mathbf{x_n} \in R^t$ is an t-dimensional observation features and $\mathbf{y_n} \in R^h$ is its coded class label. Here, h is the total number of classes, which is equal to the number of output hidden neurons. If the observation data $\mathbf{x_n}$ is assigned to the class label c, then the cth element of $\mathbf{y_n} = [y_1, ..., y_c, ... y_h]^T$ is 1 and other elements are −1, which can be denoted as follows:

$$y_j = \left\{ \begin{array}{ll} 1 & \text{if } j = c \\ -1 & \text{otherwise} \end{array} j = 1, 2, ..., h \right\} \tag{17}$$

The output tags of the ILRBF-BP classier are $\hat{\mathbf{y}}_\mathbf{n} = [\hat{y}_1, ..., \hat{y}_c, ... \hat{y}_h]^T$, where

$$\hat{y}_j = \operatorname{sgn}(o_j), \quad j = 1, 2, ...h \tag{18}$$

According to the coding rules, only one output tag value is 1 and the other value is −1. If this condition is not met, the output tag is saturated and must be adjusted. Therefore, we set an effective way to correct the saturation problem in the learning process, which can be denoted as the pseudo code in Algorithm 3.

3 Performance evaluation of the ILRBF-BP algorithm

In this section, we evaluate the performance of the ILRBF-BP algorithm using two artificial classification problems from [33] and three classification problems from the UCI machine learning repository [34]. The artificial binary data sets, including the Double-moon and Twist problems are used to measure the unique features of ILRBF-BP and the main advantages of the results over others. Table 1 provides a description of the classifying data sets, where Double-moon, Twist, and IS are well-balanced data sets and Heart and vehicle classification (VC) are imbalanced data sets. For balanced data sets, the numbers of training samples in each class are identical. For the heart problem, the numbers of training samples in classes 1 and 2 are 33 and 40, respectively. For the VC problem, the numbers of training samples in classes 1–4 are 119, 118, 98, and 89, respectively.

The performance of ILRBF-BP is compared with other well-known batch and sequential learning algorithms, such as SGBP, KM-RBF, KMRBF-BP, SVM and ELM, MRAN, GAP-RBF, and OS-ELM on different data sets. Note that the number of SGBP, KM-RBF, KMRBF-BP, ELM, and OS-ELM hidden neurons is selected manually. When changing the number of hidden neurons several times, the one with the lowest overall testing error is selected as the suitable number of hidden neurons. For multi-class problems, the method of adjusting output saturation problems is used. All simulations in each algorithm are performed ten times and are conducted in the MATLAB 2013 environment on an Intel(R) Core(TM) i5, 3.2 GHZ CPU with 4G of RAM. The simulations for the SVM are carried out using the popular LIBSVM package in C [35].

Algorithm 3 Method of adjusting the output saturation problem

Given an observation data set $\{\mathbf{x_n}, \mathbf{y_n}\}_{n=1}^{N}$, for every input vector $\mathbf{x_n}$,

While $j \pounds h$

If the number of $\hat{y}_j == -1$ is equal to h

Set $\max(o_j) = 1$ and hold other output values fixed.

EndIf

If the number of $\hat{y}_j == 1$ is more than 1

Set $\max(o_j) = 1$ and the other output values are -1.

EndIf

Endwhile

Table 1 Descriptions of the classifying data sets

Data sets	No. of features	No. of classes	No. of training	No. of testing	Attribute	Sources
Double-moon	2	2	200~2000	4000	Balance	Artificial
Twist	2	2	200~2000	4000	Balance	Artificial
Heart	13	2	73	230	Imbalance	UCI
IS	19	7	210	2100	Balance	UCI
VC	18	4	424	422	Imbalance	UCI

3.1 Performance measures

In this paper, the overall and average per-class classification accuracies are used to measure performance. The confusion matrix Q is used to obtain the class-level performance and global performance of the various classifiers. Class-level performance is measured by the percentage classification (η_i), which is defined as

$$\eta_i = \frac{q_{ii}}{N_i^T} \tag{19}$$

where q_{ii} is the number of correctly classified samples and N_i^T is the number of samples for the class $\mathbf{y_i}$ in the training/testing data set. The overall (η_o) and average per-class (η_a) classification accuracies are defined as

$$\eta_o = 100 \times \frac{1}{N^T} \sum_{i=1}^{h} q_{ii} \tag{20}$$

$$\eta_a = 100 \times \frac{1}{h} \sum_{i=1}^{h} \eta_i \tag{21}$$

where h is the number of classes and N^T is the number of training/testing samples.

3.2 Performance comparison

3.2.1 Artificial binary data sets: Double-moon problem

The prototype and data set of the Double-moon classification problem are shown in Fig. 4a, b, respectively, where $r = 10$, $\omega = 6$ and $d = -6$. The main parameters of distance weighting factor, width, incremental learning threshold, number of BP hidden neurons, and momentum constant in ILRBF-BP are set as $T = 1$, $\sigma = 3$, $\delta = 0.01$, $M = 5$, and $\alpha = 0$, respectively. Figure 4c shows the classification results for the testing samples under these parameters. The classification results illustrate that the proposed algorithm can provide a superior classification surface. Figure 5 shows using different width parameters to cover the training sample space, where each cover generates a RBF hidden neuron and the number of RBF hidden neurons is increased incrementally, the bold lines represent the first coverage region in each pattern class. In Fig. 5, with the increase of the width parameter, the corresponding region covered each RBF hidden neuron is increased accordingly, which will affect the location of the next center, thus generates different RBF hidden neurons. Though the number of RBF hidden neurons has changed, ILRBF-BP still can effectively cover each class of training samples. Thus, the incremental learning algorithm based on potential function clustering is feasible. ILRBF-BP can be well adapted to the sample space, which is an effective algorithm to incrementally generate RBF hidden neurons for the Double-moon problem.

Figure 6a, b demonstrates that when the number of training samples has changed, KMRBF-BP needs less

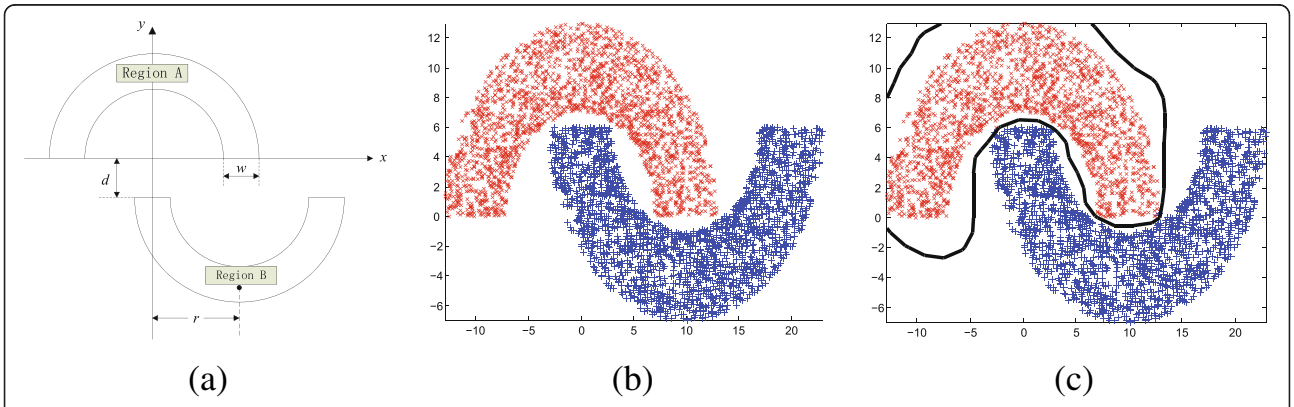

Fig. 4 **a** Double-moon classification problem. **b** Double-moon data set. **c** Classification result of the ILRBF-BP algorithm

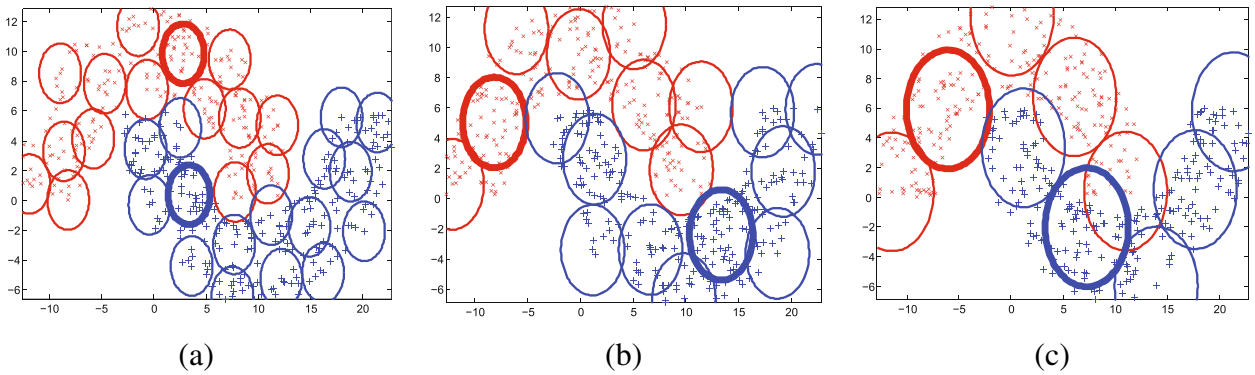

Fig. 5 Using different width parameters to cover the training sample space for Double-moon classification problem. **a** $\sigma = 2$. **b** $\sigma = 3$. **c** $\sigma = 4$

number of RBF hidden neurons than KM-RBF. When the number of training samples is more than 500, KMRBF-BP can get a higher classifying accuracy than KM-RBF. These results show that the hybrid RBF-BP network architecture is effective, which can improve the classifying accuracy and reduce the dependence on the original sample space mapping. In GAP-RBF and ILRBF-BP, the number of RBF hidden neurons is generated automatically. ILRBF-BP needs less number of RBF hidden neurons than GAP-RBF, and the overall testing accuracy outperforms GAP-RBF. The classifying accuracy of ILRBF-BP is comparable with SVM and KMRBF-BP and outperforms ELM and KM-RBF. Note that the number of KM-RBF and KMRBF-BP is selected manually. When changing the number of hidden neurons several times, the one with the highest overall testing accuracy is selected as the suitable number of hidden neurons. As ILRBF-BP utilizes global information about each class of training sample space, it can generate RBF hidden neurons incrementally to adapt the sample space, and the hybrid RBF-BP network architecture improves the network performance further.

3.2.2 Artificial binary data sets: Twist problem

The prototype and data set for the twist classification problem are shown in Fig. 7a, b, respectively, where $d_1 = 0.2$, $d_2 = 0.5$ and $d_3 = 0.8$. Compared to the Double-moon problem, the twist classification problem is more complex and can thus be used to evaluate the classification performance of the different algorithms. The main parameters of distance weighting factor, width, incremental learning threshold, number of BP hidden neurons, and momentum constant in ILRBF-BP are set as $T = 200$, $\sigma = 0.15$, $\delta = 0.01$, $M = 5$, and $\alpha = 0$, respectively. Figure 7c shows the classification results for the testing samples under these parameters. The classification results illustrate that the proposed algorithm still provides a superior classification surface for the Twist classification problem. Figure 8 shows using different width parameters to cover the training sample space, where each cover generates a RBF hidden neuron. In Fig. 8, the bold lines represent the first coverage region, which denote the most dense region in each pattern class. Although there are some overlap in different coverage regions, ILRBF-BP still can effectively cover each class of training

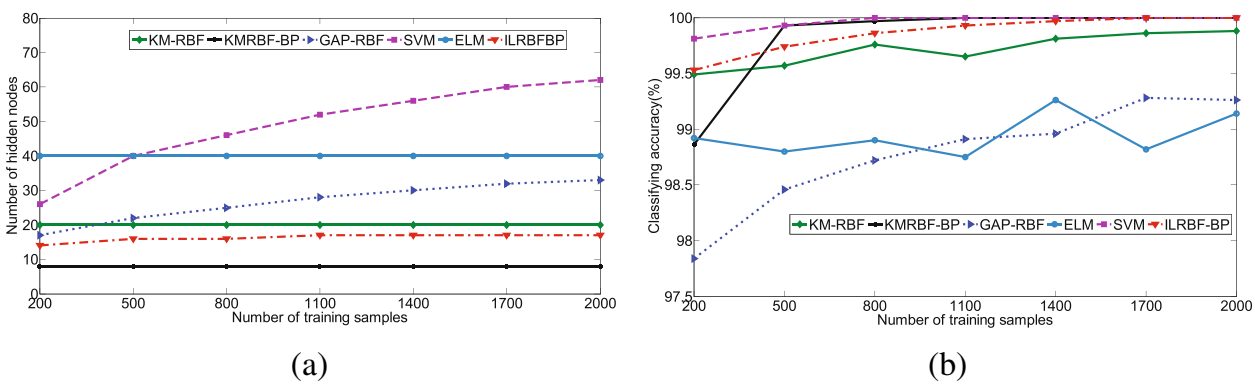

Fig. 6 Performance comparisons between ILRBF-BP and other algorithms on the Double-moon problem. **a** Number of training samples—number of hidden neurons. **b** Number of training samples—classifying accuracy

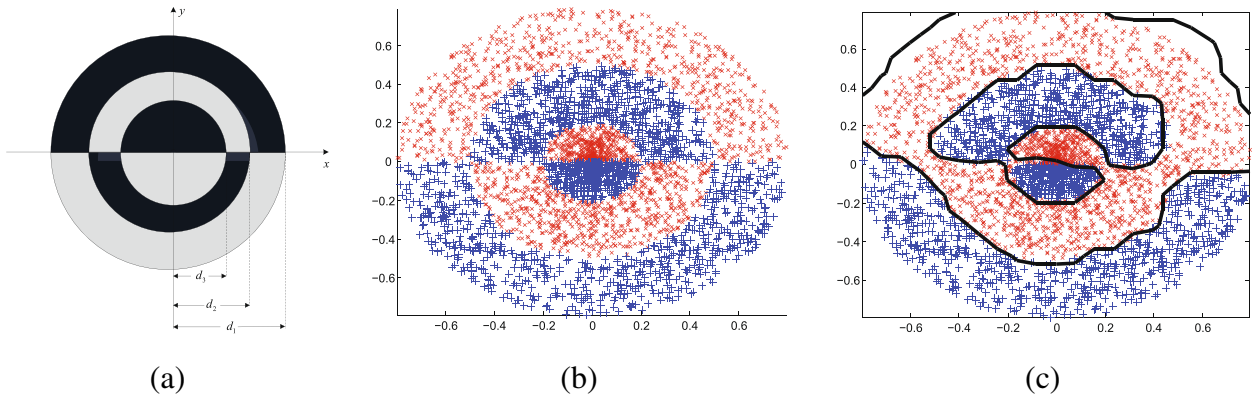

Fig. 7 **a** Twist classification problem. **b** Twist data set. **c** Classification result of the ILRBF-BP algorithm

samples and generate corresponding RBF hidden neurons incrementally.

Figure 9a, b demonstrates that when the number of training samples has changed, KMRBF-BP needs less number of RBF hidden neurons than KM-RBF and can get a higher classifying accuracy. Thus, the hybrid RBF-BP network architecture improves the classifying accuracy and reduces the dependence on the original sample space mapping. Note that in KM-RBF and KMRBF-BP, when the number of training samples is changed, the number of RBF hidden neurons has to be adjusted manually; otherwise, it will lead to a poor classification accuracy. Compared to KM-RBF and KMRBF-BP, ILRBF-BP can adapt the training sample space well; when the number of training samples is changed, the number of RBF hidden neurons in ILRBF-BP is changed accordingly and can get a higher classifying accuracy. Compared to GAP-RBF, ILRBF-BP can better adapt to the change of sample space. The classifying accuracy of ILRBF-BP outperforms GAP-RBF as well as SVM and ELM. Thus, the incremental learning algorithm based on potential function is effective, which utilizes global information about each class of training sample space to construct RBF hidden neurons incrementally, and the hybrid RBF-BP network architecture improves the network performance further.

3.2.3 UCI binary data set: Heart problem

In this section, the Heart problem in the UCI binary data set is used to evaluate the performance of the ILRBF-BP algorithm. In the Heart problem, the sample distribution values of each dimension are between 0 and 1, and the main parameters of distance weighting factor, width, incremental learning threshold, number of BP hidden neurons, and momentum constant in ILRBF-BP are set as $T = 1$, $\sigma = 1.2$, $\delta = 0.001$, $M = 5$, and $\alpha = 0.1$, respectively. As noted above, the Heart problem is an imbalanced classification problem. This, in addition to the overall testing η_o, the average testing η_a is also used to measure the performance of each algorithm.

The performance comparisons between ILRBF-BP and the other batch learning algorithms are shown in Table 2. For the Heart problem, the overall and average testing accuracy of ILRBF-BP are clearly higher than those of SGBP, and the proposed algorithm outperforms ELM

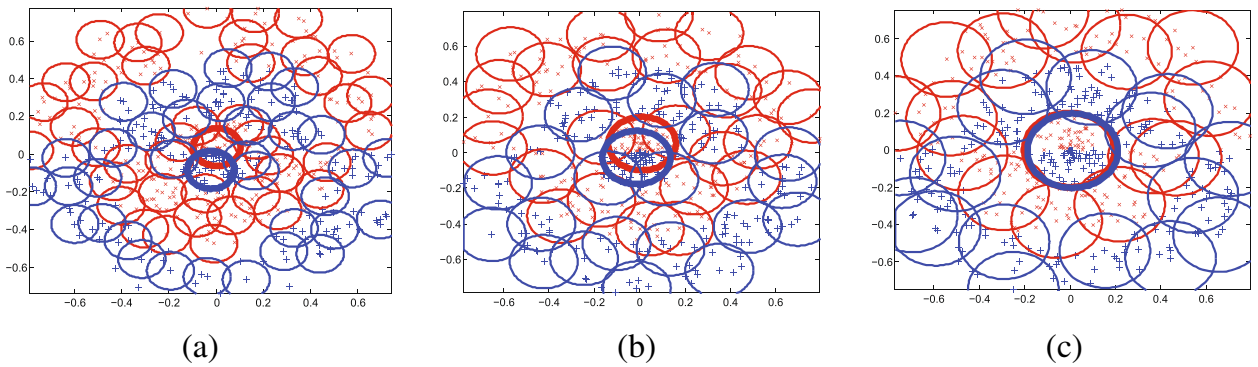

Fig. 8 Using different width parameters to cover the training sample space for the twist classification problem. **a** $\sigma = 0.1$. **b** $\sigma = 0.15$. **c** $\sigma = 0.2$

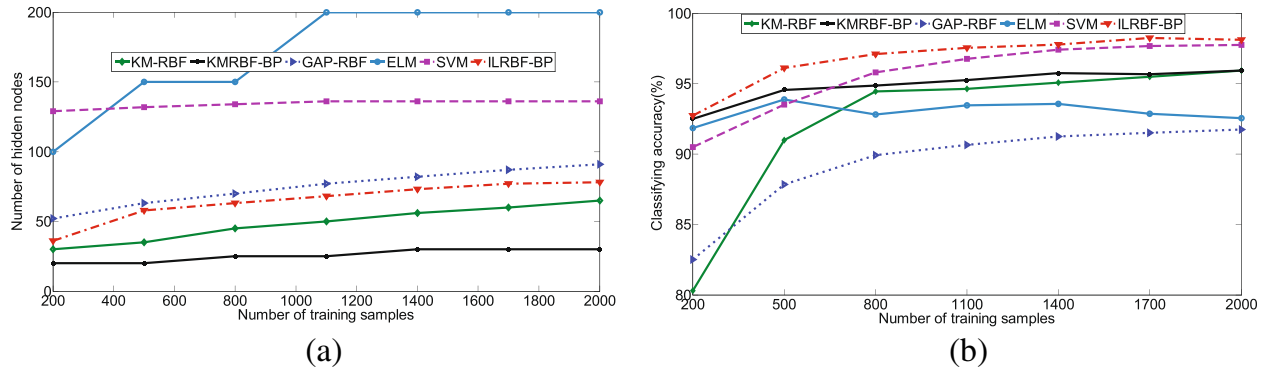

Fig. 9 Performance comparisons between ILRBF-BP and other algorithms on the Twist problem. **a** Number of training samples—number of hidden neurons. **b** Number of training samples—classifying accuracy

and KM-RBF by approximately 2.5–5 %. The average testing accuracy of ILRBF-BP is 1.74 % lower than that of the SVM; however, the overall testing accuracy is approximately 3 % higher than that of the SVM, and fewer hidden neurons are needed.

3.2.4 UCI multi-class data sets: IS and VC problems

In this section, the IS and VC problems are used to evaluate the performance of the ILRBF-BP algorithm. The output saturation is adjusted for the multi-class classifying problem in the ILRBF-BP algorithm. For the IS problem, the sample distribution range in each dimension is different, so the inputs of each algorithm are scaled appropriately between 0 and +1. The main parameters of distance weighting factor, width, incremental learning threshold, number of BP hidden neurons, and momentum constant in ILRBF-BP are set as $T = 1$, $\sigma = 0.3$, $\delta = 0.001$, $M = 8$, and $\alpha = 0.2$, respectively. The IS problem is a well-balanced data set; the number of training samples in each class is 30, and the overall testing η_o is used to measure the performance of each algorithm. For the VC problem, the sample distribution values of each dimension are between −1 and +1, and the main parameters of distance weighting factor, width, incremental learning threshold, number of BP hidden neurons, and momentum constant in ILRBF-BP are set as $T = 1$, $\sigma = 0.4$, $\delta = 0.001$, $M = 9$, and $\alpha = 0.1$, respectively. The number of training samples in each class is 119, 118, 98, and 89. The VC problem is a highly imbalanced data set, where the strong overlap between the classes influences the performance of each algorithm. The overall testing η_o and average testing η_a are used to measure the performance of each algorithm.

Table 3 shows the performance comparisons for the IS and VC problems. For the IS problem, the overall testing accuracy of ILRBF-BP is approximately 5–6 % higher than those of MRAN and GAP-RBF and approximately 0.9–1.3 % higher than those of OS-ELM, SVM, and ELM. For the VC problem, the overall and average testing accuracies of ILRBF-BP are approximately 9–11 % higher than those of MRAN and GAP-RBF and approximately 1.2–2.5 % higher than those of the SVM, ELM, and OS-ELM. The number of RBF hidden neurons and training time of ILRBF-BP are the greatest because the strong overlap of sample space increases the number of RBF hidden neurons and learning time, which yields a higher classification accuracy.

3.3 Analysis of the parameters in the ILRBF-BP algorithm

In this section, the parameter selection for the ILRBF-BP algorithm is discussed, which mainly refers to the

Table 2 Performance comparison for the Heart problem

Method	N_H neurons	Training time(s)	Training η_o	Testing η_o	Testing η_a
SGBP	7	0.95	95.01	46.09	48.42
KM-RBF	7	0.78	82.19	75.22	75.30
SVM	39[a]	0.08	100	77.39	81.81
ELM	10	0	87.67	77.83	77.66
ILRBF-BP	11 and 5[b]	0.66	91.78	80.43	80.07

[a]Support vectors
[b]RBF and BP hidden neurons

Table 3 Performance comparisons for the IS and VC problems

Data sets	Method	N_H neurons	Training time(s)	Testing η_o	Testing η_a
IS	SVM	96[a]	11.61	90.62	–
	MRAN	78	11.68	85.82	–
	GAP-RBF	87	5.77	86.34.	–
	ELM	49	0	90.23	–
	OS-ELM	100	0.01	90.67	–
	ILRBF-BP	77 and 8[b]	2.09	91.57	–
VC	SVM	234[a]	10.74	68.72	67.99
	MRAN	105	10.38	60.24	60.02
	GAP-RBF	81	9.87	58.94	58.17
	ELM	300	0.09	68.01	67.39
	OS-ELM	300	0.12	68.95	67.56
	ILRBF-BP	258 and 9[b]	11.53	70.17	69.43

[a]Support vectors
[b]RBF and BP hidden neurons

distance weighting factor T, width σ and number of BP hidden neurons.

3.3.1 Selection of distance weighting factor T

In this paper, parameter T is used for distance weighting, which can be used to control the interaction potential between two samples. By changing T, the nonlinear mapping of the potential γ can be achieved.

To determine a proper choice of T, in this paper, the standard deviation is considered to measure the impact on T. Here, the Twist classification problem is used in the experiment. Given the number of training samples is 500 and testing samples is 4000; other parameters are given as follows:

1) Twist 1: Set $d_1 = 0.2$, $d_2 = 0.5$, and $d_3 = 0.8$,the standard deviation in each dimension is 0.3281 and 0.3196, respectively. The width parameter is set as $\sigma = 0.1$.
2) Twist 2: Set $d_1 = 2$, $d_2 = 5$, and $d_3 = 8$,the standard deviation in each dimension is 3.2744 and 3.2689, respectively. The width parameter is set as $\sigma = 1$

Figure 10a shows that when the samples are not normalized, for the Twist 2 sample set, the standard deviation of each dimension is relatively large; with the increase of T, the classification performance is reduced. For the Twist 1 sample set, the standard deviation of each dimension is relatively small and the sensitivity of classifying accuracy on T is reduced; however, when the T is selected as 200, the maximum classification accuracy is achieved. Thus, the choice of T should be inversely proportional to the standard deviation of each dimension, that is, $T \propto {}^{1}/_{\max} i = 1, 2 \ldots t\{\alpha_i\}$, where α_i is the standard deviation of ith dimension and t is the sample dimension. Figure 10b further indicates that when the samples are normalized to [−1, 1], the dependence on T is reduced and a relatively stable classification accuracy can be achieved.

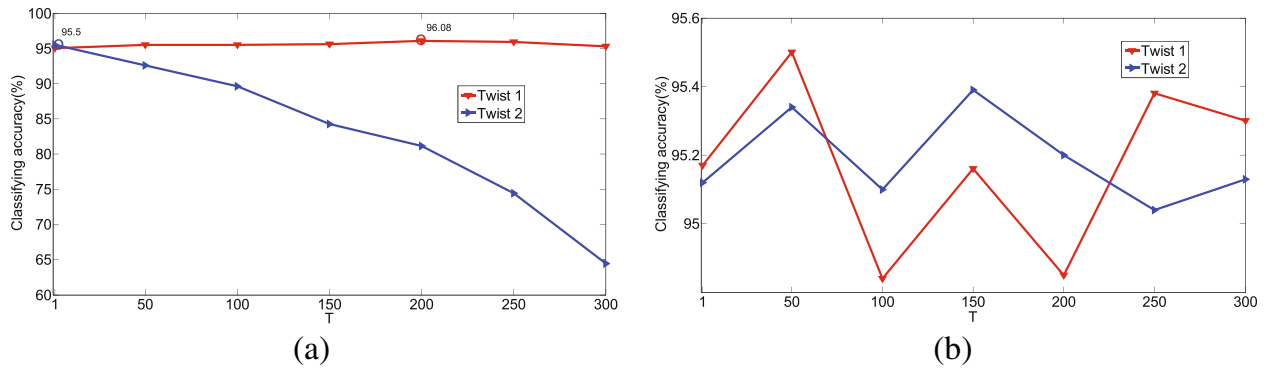

Fig. 10 Impact of T on ILRBF-BP. **a** Samples are normalized. **b** Samples are not normalized

In this paper, for the Double-moon data set, the maximum standard deviation of two dimensions is 8.6448, so a small T should be provided and T is set as $T = 1$. For the Twist data set, the maximum standard deviation of two dimensions is 0.3281, and T is set as $T = 200$.

In high-dimensional space, the sample distribution is often relatively sparse. The sample dimension is considered to be inversely proportional to T, thus $T \propto 1/t$. In this paper, for the IS classification problem, the input values in each dimension are scaled appropriately between 0 and +1. For the Heart and VC classification problems, the values in each dimension are between −1 and 1. Thus, the impact of standard deviation on T is eliminated. Taken into account the dimension information, for the IS, Heart, and VC classification problems, a small T should be provided and T is set as $T = 1$.

3.3.2 Impact of the width σ on ILRBF-BP

The width parameter σ can be used to control the classification accuracy and generalization performance in a RBF network. In the ILRBF-BP algorithm, the width is fixed and selected by cross validation. To reduce the range of the width parameter value selection, we conduct preprocessing for the sample space. If the sample distribution values of each dimension vary considerably, such as in the IS data set, the inputs to each algorithm are scaled appropriately between 0 and +1, whereas the inputs to each algorithm remain unchanged in the Heart and VC data sets.

In the proposed incremental learning algorithm, using a potential function approach to construct RBF hidden neurons incrementally has to complete the effective coverage of the training sample space. As the samples in high-dimensional space are relatively sparse, if the width is too small, it may lead to establish the corresponding Gaussian kernel at each sample, and the proposed incremental learning algorithm is invalid. The reason is that although the potential value of each sample in the training sample space is measured, in the process of eliminating the potential value of the sample, the generated RBF hidden neurons do not cover other samples, which will lead to a failure of Eq. (9), and excessive RBF hidden neurons will lead to the redundancy of the network architecture, which affects the classification performance of the BP network. Thus, in the proposed ILRBF-BP algorithm, an effective kernel width parameter should be provided, which can generate proper RBF hidden neurons to cover the sample space. Note that the number of generated RBF hidden neurons should not be close to the number of the training samples; otherwise, the proposed algorithm is invalid.

Figure 11a, b shows the impact of width on the overall classification accuracy and the number of RBF hidden neurons, respectively. Figure 11 illustrates that for the Heart and VC data sets, when the width parameter is small, such as $\sigma = 0.1$ and $\sigma = 0.2$, the overall classification accuracy is poor, and effective coverage of the input sample space is not achieved.

When the value of the width parameter is in a suitable range, the number of generated RBF hidden neurons will change, but a relatively stable classification accuracy can be achieved. For the proposed ILRBF-BP algorithm, once the width is given, it can learn the sample space automatically, and the changes in the width parameter will affect the coverage of RBF hidden neurons and generate different RBF hidden neurons. Thus, the incremental learning strategy can counteract the effect of the width to some extent.

4 Impact of the number of BP hidden neurons on ILRBF-BP

In the hybrid RBF-BP network architecture, the nonlinear BP algorithm is used to adjust the weights of the MLPs, which further improves the classification result. However, this method results in an increase in the number of parameters to be selected, especially the selection of the number of BP hidden neurons. For this problem, we conduct experiments on the UCI data sets and discuss the results.

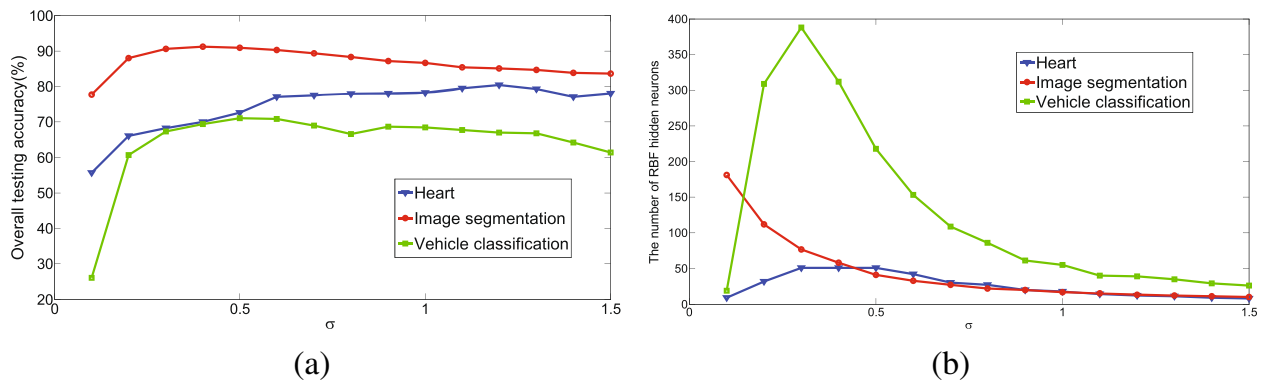

Fig. 11 Impact of width on ILRBF-BP. **a** σ—overall testing accuracy. **b** σ—number of RBF hidden neurons

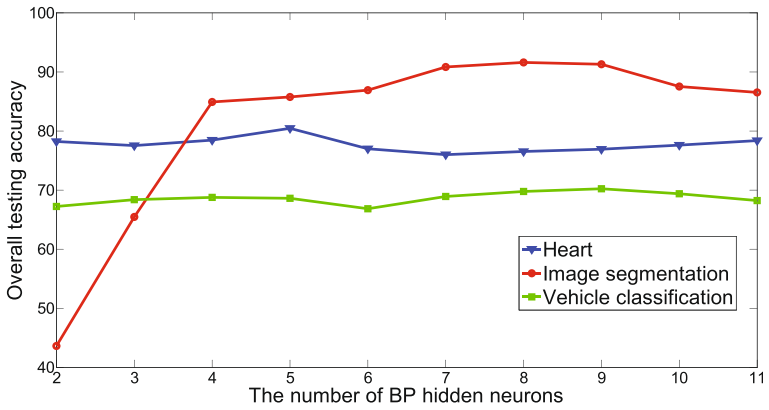

Fig. 12 Impact of the number of BP hidden neurons on ILRBF-BP

Figure 12 shows the impact of the number of BP hidden neurons on ILRBF-BP. For the Heart, IS, and VC problems, when the number of BP hidden neurons is greater or equal to 4, the overall classification accuracy does not change considerably. For the hybrid RBF-BP network, the mapping results of RBF hidden neurons are processed and used for the input of BP network, which improves the stability of the BP network and effectively avoids falling into local minima for the BP algorithm. Thus, the dependence on the number of BP hidden neurons is reduced. When the sample set is more complex, the momentum term can be used to improve the BP algorithm further.

5 Conclusions

In this paper, an incremental learning algorithm for the hybrid RBP-BP (ILRBF-BP) network classifier is proposed. The ILRBF-BP algorithm uses a potential function to measure the density of the training sample space and incrementally generates RBF hidden neurons, enabling the effective estimation of the center and number of RBF hidden neurons. In this way, a suitable network size for RBF hidden layer that matches the complexity of the sample space can be built up. A hybrid RBF-BP network architecture is designed to improve classification performance further, which shows good stability and generalization performance. The hybrid network simplifies the selection of the number of neurons in the BP hidden layer while further reducing the dependence on space mapping in the RBF hidden layer.

The performance of the ILRBF-BP algorithm has been compared with other batch learning algorithms, such as SGBP, KM-RBF, SVM, and ELM, and sequential learning algorithms, such as MRAN, GAP-RBF, and OS-ELM, in artificial data sets and UCI data sets. The method of adjusting output label values is used to prevent the output saturation problem for multi-class classification. Experiments demonstrate the superiority of the ILRBF-BP algorithm.

In the future, we will focus on the optimization of kernel width and imbalanced data classification problems. In the ILRBF-BP algorithm, the width is fixed and selected by cross validation and the adjustment of width parameter will affect the location of next center, as well as the network size. Therefore, it is necessary to design an adaptive width adjustment to adapt to the different regions of the sample space. In addition, for the imbalanced data classification problem, the samples in the boundary regions contain more classification information, thus how to measure and select these samples is particularly important. Further studies are needed to address these concerns.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

The authors thank the support provided by the National Science Foundation of China (No. 61331021, U1301251) and the Shenzhen Science and Technology Plan Project (JCYJ20130408173025036). The authors would like to thank the Editor-in-Chief, the Associate Editor, and the Anonymous Reviewers for their helpful comments and suggestions which have greatly improved the quality of presentation.

Received: 3 November 2015 Accepted: 27 April 2016
Published online: 10 May 2016

References

1. M Lin, K Tang, X Yao, Dynamic sampling approach to training neural networks for multiclass imbalance classification. IEEE Trans Neural Netw and Learning Systems **24**(4), 647–660 (2013)
2. L-Q Li, W-X Xie, Intuitionistic fuzzy joint probabilistic data association filter and its application to multitarget tracking. Signal Process **96**, 433–444 (2014)
3. Y-L Wei, J-B Qiu, HR Karimi, M Wang, H-infinity model reduction for continuous-time Markovian jump systems with incomplete statistics of mode information. Int J Syst Sci **45**(7), 1496–1507 (2014)
4. Y-L Wei, J-B Qiu, HR Karimi, M Wang, Filtering design for two-dimensional Markovian jump systems with state-delays and deficient mode information. Inform Sci **269**, 316–331 (2014)

5. Y-L Wei, J-B Qiu, HR Karimi, M Wang, A new design of H∞ filtering for continuous-time Markovian jump systems with time-varying delay and partially accessible mode information. Signal Process **93**(9), 2392–2407 (2013)
6. F-Y Meng, X Li, J-H Pei, A feature point matching based on spatial order constraints bilateral-neighbor vote. IEEE Trans Image Process **24**(11), 4160–4171 (2015)
7. L-X Guan, W-X Xie, J-H Pei, Segmented minimum noise fraction transformation for efficient feature extraction of hyperspectral images. Pattern Recogn **48**(10), 3216–3226 (2015)
8. HC Nejad, O Khayat, B Azadbakht, M Mohammadi, Using feed forward neural network for electrocardiogram signal analysis in chaotic domain. J Intelligent and Fuzzy Systems **27**(5), 2289–2296 (2014)
9. CH Weng, CK Huang, RP Han, Disease prediction with different types of neural network classifiers. Telematics Inform **33**(2), 277–292 (2016)
10. C Lu, N Ma, ZP Wang, Fault detection for hydraulic pump based on chaotic parallel RBF network. EURASIP J on Advances in Signal Processing **49**, (2011). doi: 10.1186/1687-6180-2011-49
11. J Moody, CJ Darken, Fast learning in networks of locally-tuned processing. Neurocomputing **1**(2), 281–294 (1989)
12. D Lowe, Characterising complexity by the degrees of freedom in a radial basis function network. Neurocomputing **19**(1-3), 199–209 (1998)
13. J Platt, A resource-allocating network for function interpolation. Neural Comput **3**(2), 213–225 (1991)
14. V Kadirkamanathan, M Niranjan, A function estimation approach to sequential learning with neural networks. Neural Comput **5**(6), 954–975 (1993)
15. L Yingwei, N Sundararajan, P Saratchandran, A sequential learning scheme for function approximation using minimal radial basis function. Neural Comput **9**(2), 461–478 (1997)
16. G-B Huang, P Saratchandran, N Sundararajan, An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. IEEE Trans Syst Man Cybern B Cybern **34**(6), 2284–2292 (2004)
17. G-B Huang, P Saratchandran, N Sundararajan, A generalized growing and pruning RBF (GAP-RBF) neural network for function approximation. IEEE Trans Neural Netw **16**(1), 57–67 (2005)
18. M Bortman, M Aladjem, A growing and pruning method for radial basis function networks. IEEE Trans Neural Netw **20**(6), 1030–1045 (2009)
19. H Yu, PD Reiner, T Xie, T Bartczak, BM Wilamowski, An incremental design of radial basis function networks. IEEE Trans Neural Netw and Learning Systems **2**(10), 1793–1803 (2014)
20. S Suresh, D Keming, HJ Kim, A sequential learning algorithm for self-adaptive resource allocation network classifier. Neurocomputing **73**(16-18), 3012–3019 (2010)
21. T Xie, H Yu, J Hewlett, P Rózycki, B Wilamowski, Fast and efficient second-order method for training radial basis function networks. IEEE Trans Neural Netw and Learning Systems **23**(4), 609–619 (2012)
22. C Constantinopoulos, A Likas, An incremental training method for the probabilistic RBF network. IEEE Trans Neural Netw **17**(4), 966–974 (2006)
23. G-B Huang, Q-Y Zhu, C-K Siew, A new learning scheme of feedforward neural, in *Proceedings of International Joint Conference on Neural Networks (IJCNN 2004)*, pp. 985–99
24. N-Y Liang, G-B Huang, P Saratchandran, N Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw **17**(6), 1411–1423 (2006)
25. G-B Huang, L CHEN, C-K Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw **17**(4), 879–892 (2006)
26. G-B Huang, L CHEN, Convex incremental extreme learning machine. Neurocomputing **70**(16-18), 3056–3062 (2007)
27. G-B Huang, L Chen, Enhanced random search based incremental extreme learning machine. Neurocomputing **71**(16-18), 3460–3468 (2008)
28. G Feng, G-B Huang, Q Lin, Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans Neural Netw **20**(8), 1352–1357 (2009)
29. Y LeCun, L Bottou, GB Orr, K-R Müller, Efficient backprop. Lecture Notes Comput Sci **1524**, 9–50 (1998)
30. J-H Pei, W-X Xie, Adaptive multi thresholds image segmentation based on potential function clustering. Chinese J Computers **22**(7), 758–762 (1999)
31. OA Bashkerov, EM Braverman, IB Muchnik, Potential function algorithms for pattern recognition learning machines. Autom Remote Control **25**(5), 692–695 (1964)
32. G Cybenko, Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signal, and Systems **2**, 303–314 (1989)
33. S Hayin, *Neural networks and learning machines. Third Edition* (China Machine Press, China, 2009), pp. 61–63
34. C Blake, C Merz, *UCI repository of machine learning databases* (Department of Information and Computer Sciences, University of California, Irvine, 1998). available at http://archive.ics.uci.edu/ml/
35. C-C Chang, C-J, *LIBSVM: a library for support vector machines* (Department of Computer Science and Information Engineering, National Taiwan University, Taiwan, 2003). available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html