Neural Comput & Applic (2007) 16: 395–405
DOI 10.1007/s00521-006-0063-9

ORIGINAL ARTICLE

D. Huang · T. W. S. Chow

Improving the effectiveness of RBF classifier based on a hybrid cost function

Received: 6 February 2006 / Accepted: 12 June 2006 / Published online: 19 July 2006

Abstract This paper presents a new minimum classification error (MCE)–mean square error (MSE) hybrid cost function to enhance the classification ability and speed up the learning process of radial basis function (RBF)-based classifier. Contributed by the MCE function, the proposed cost function enables the RBF-based classifier to achieve an excellent classification performance compared with the conventional MSE function. In addition, certain learning difficulties experienced by the MCE algorithm can be solved in an efficient and simple way. The presented results show that the proposed method exhibits a substantially higher convergence rate compared with the MCE function.

Keywords Classification · Gradient decent · Discriminant function · Minimum classification error · Mean square error · RBF neural network

1 Introduction

Neural-based classifiers have emerged in recent years and have exhibited many advantages over conventional statistical methods [1]. These include their learning ability and distribution-free characteristics that make neural-based classifiers less restrictive in performing classification. Radial basis function (RBF) network is a class of neural network providing the above advantages. On top of those, RBF networks provide the following salient features: (1) they exhibit an outstanding functional approximation property [2]; (2) the two components of RBF (non-linear and linear) can be separately treated. With these properties, RBF network is known to be a better classifier than multiplayer perceptrons (MLPs) [3–5]. Accordingly, RBF network has been applied to various real world classification tasks [6–8].

In order to enhance the classification ability and convergence rate, many research activities were conducted to modify the structure and learning algorithms of RBF network. There are, however, relatively few studies on modifying the cost function for the above purposes. Hitherto, the mean square error (MSE) function is the most commonly used cost function, although there are reports showing that other cost functions such as cross-entropy (CE) function are more suitable than MSE function for training a neural-based classifier [5, 9]. It is noted that when the above cost functions are to be used in classification tasks, there are differences between the objectives of training a neural classification model and the requirements of the classification task. By minimising the above cost functions, the training algorithms are aimed at making the output of an RBF network approximate the discrete values (such as 0 and 1) as much as possible. But a classification task poses a special feature in that the classification decision is generally made on the basis of hard decision rules. In other words, we classify a given observation to the class having the maximum output in order to minimise the classification error probability. In this sense, the requirements on the output of a classifier can be relaxed. It is sufficient that the output of the classifier can correctly distinguish the class of a data pattern, while the value of the output may not be required to approximate the designated value (1 or 0) as much as possible. In a training process based on the above-mentioned cost function, the patterns that have been correctly recognised still have an unnecessary contribution to the subsequent training process. This is likely to reduce the convergence rate and cause overfitting [10]. In [10], the above issue was addressed and an improved threshold type MSE function was proposed to train RBF networks. Together with an appropriate algorithmic scheme, certain improvement on the classification performance was obtained. The convergence rate was significantly speeded up than the algorithm used to

D. Huang (✉) · T. W. S. Chow
Electronic Department, City University of Hong Kong, Hong Kong, China
E-mail: dihuang@ee.cityu.edu.hk
E-mail: eetchow@cityu.edu.hk

optimise the MSE function. But it is also worth noting that the convergence rate of this type of training algorithm cannot be further accelerated because the linear parameters of the RBF network cannot be optimised by a much speedy linear least square (LLS) algorithm [5].

With the aim of enhancing the classification ability of a neural classifier, another type of cost function, the minimum classification error function (MCE), was introduced and used in [11, 12]. Although the MCE function enables the neural classifier to be constructed directly to minimise the classification error, there is a main difficulty in implementing MCE-based methods. The selection of the smoothness parameter ξ of MCE has a marked effect on training classifiers. A large ξ results in a rather rugged error surface in which a lot of local minima are very likely to cause sub-optimal results of the MCE function. On the other hand, when ξ is set to a small value, the error surface of the MCE becomes smoother. The gradient of MCE, however, becomes small at any point in that case, which results in a low convergence rate. Although the overall performance of the MCE can be adjusted by selecting an appropriate ξ, it has never been easy because the effect of ξ varies with the given problem. As a result, dynamical change of ξ was introduced in an attempt to improve both the classification performance and the convergence rate [13]. Basically, it is an algorithmic approach to adjust the smoothness parameter ξ when the training proceeds. Despite its promising results, this type of approach is heuristic and problem dependent. Implementation of an appropriate adaptive strategy has never been straightforward.

In this paper, we introduce a new type of hybrid cost function, called the MCE–MSE function, which is a combination of the MCE and MSE functions. The proposed MCE–MSE function is capable of providing a neutral property of maintaining the high classification performance and fast convergence rate. Thus, the MCE–MSE function provides an efficient and effective way to overcome the problems experienced by the MSE and MCE functions. When the classification error is relatively large, the effect of MSE function helps speed up the training process. In this way, the problem of the MCE-based method having low convergence rate is addressed in a simple way. The presented results show that the proposed MCE–MSE based approach outperforms both the MCE-based and the MSE-based approaches. In this study, the parameters of RBF are adjusted in a popular way. The LLS scheme is used for updating the linear component of RBF networks. Moreover, the parameters of the non-linear component of RBF networks are adjusted based on the MCE–MSE function. The results of our study show that this hybrid training scheme based on the newly introduced MCE–MSE function can produce promising results in terms of classification performance and convergence rate. The classification performance in this paper is measured by the classification error. The convergence rate is measured in terms of the number of training epochs and the running time.

This paper is organised as follows. Section 2 gives the background of this study. The hybrid MCE–MSE function is introduced and evaluated in Sect. 3. The MCE–MSE based method for training the RBF classifier in detailed in Sect. 4. Our study results are presented and discussed in Sect. 5. Finally, conclusions are drawn.

2 Background

2.1 RBF neural networks

Geometrically, RBF network is considered to partition the whole input space by hyper-ellipsoids. In the interpolation theory, the RBFs are usually combined with a polynomial term for functional approximation [14]. In this paper, in order to enhance the classification accuracy on data sets having a complex distribution, RBF network is combined with a simple polynomial. With this type of RBF network, the input domain can be divided in a more flexible and effective way when dealing with complex classification problems.

The RBF neural network used in this paper has M input units, H hidden units and J output units, as shown in Fig. 1. For an input data pattern $x_i = (x_{i1}, x_{i2}, ..., x_{iM})$, the output of the jth output units is presented as

$$y_j = f_j(x_i) = \sum_{k=1}^{H} w_{kj}^{(2)} g_k(x_i) + \sum_{u=1}^{M} w_{uj}^{(1)} x_{iu} + b_j, \tag{1}$$

where, as shown in Fig. 1, $w_{kj}^{(2)}$ is the weight between the kth hidden unit and the jth output unit and $w_{uj}^{(1)}$ is the weight between the uth input unit and the jth output unit. Each basis function $g(x)$ is determined by a centre vector c and a width vector σ. The basis function of the hidden neuron is typically chosen as a Gaussian-like function:

$$g_j(x_i) = \exp\left(-\sum_{k=1}^{M} \frac{(x_{ik} - c_{jk})^2}{2\sigma_{jk}^2}\right). \tag{2}$$

The above RBF network is defined by non-linear parameters (H, c, σ) and linear weights ($W^{(2)}$, $W^{(1)}$, B). There are many supervised or unsupervised methods for performing initialisation of H, c, σ, such as SOM [15], regression tree [16], k-means, SVM [17], etc. After initialising a network, the centres and the widths (c, σ) can be adjusted by gradient decent type learning methods [4, 8]. Linear weights $W^{(2)}$, $W^{(1)}$ and B, can be adjusted using LLS, which is the most commonly used because it is generally faster than the gradient paradigm and is able to avoid the sub-optimal points [5].

2.2 Mean square error and minimum classification error

Assume that patterns in data set $X = (x_1, x_2, ..., x_N)'$ have fallen into J categories. In this paper, the classifiers have J output units, i.e., one output unit corresponds to

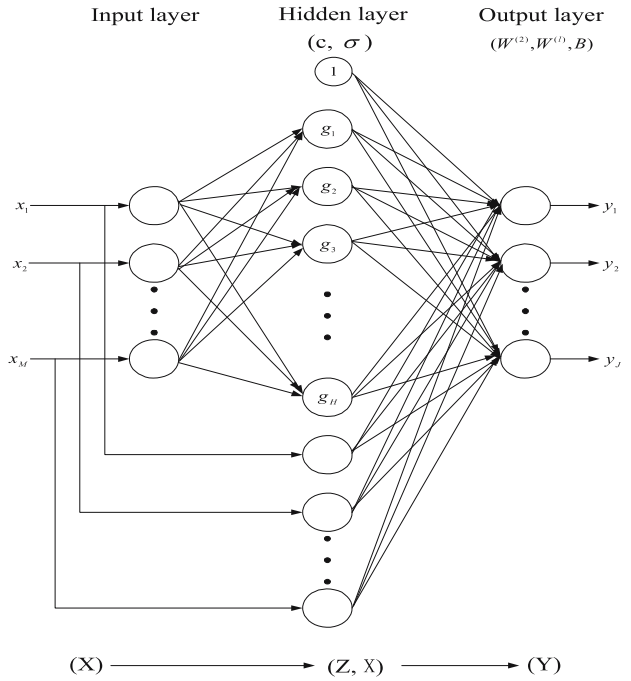

Fig. 1 Architecture of the RBF network used in this paper

one class. For convenience, for a data pattern x_i, let $t_i = (t_{i1}, t_{i2}, ..., t_{iJ})$ and $y_i = (y_{i1}, y_{i2}, y_{i3}, ..., y_{iJ})$ be the output target and the actual output of a classifier, respectively. For a pattern (say, x_i) belonging to the class cl_j ($j = 1, 2, ..., J$), it is defined that $t_{ik} = \begin{cases} 1 & k = j \\ 0 & k \neq j \end{cases}$.

Mean square error function E_{mse}, commonly used for training classifiers, can be expressed as

$$E_{\mathrm{mse}} = \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{J}\left(t_{ij} - y_{ij}\right)^2. \tag{3}$$

As mentioned before, in (3), the classification decision rules are not directly reflected. By minimising (3), the output of the classifier approximates the target (1 or 0) as close as possible.

Compared with the MSE function, a hard decision rule is directly adopted in the MCE function [11, 12]. In the MCE function, $y_i = (y_{i1}, y_{i2}, y_{i3}, ..., y_{iJ})$ is considered as the outputs of a set of discriminant functions. Generally, the class is determined as

$$x \in \mathrm{cls}_k \quad \text{if } y_k(x) = \arg\max_{1\le j\le J} y_j(x).$$

Based on this type of decision rule, for x belonging to the class cl_m, recognising x correctly requires that y_m is larger than the other ys. Also, the larger the difference between y_m and other ys, the smaller the misclassification risk. Thus, the difference between y_m and the largest y can be used to measure the classification ability of the discriminant function set just built. This is the rationale behind the misclassification measure (4) defined in [11]:

$$d^{(\mathrm{m})}(x,\Lambda) = -y_m(x,\Lambda) + \left[\frac{1}{M-1}\sum_{j,j\neq m} y_j(x,\Lambda)^p\right]^{1/p}, \tag{4}$$

where p is a positive constant and Λ the parameter set of the discriminant functions. The small value of $d^{(\mathrm{m})}$ indicates the small misclassification probability. With different p, one can take all the potential class into consideration to a different degree. When p is large enough, we have $\arg\max_{j,j\neq m}(y_j(x,\Lambda)) \approx \left[\frac{1}{M-1}\sum_{j,j\neq m} y_j(x,\Lambda)^p\right]^{1/p}$. In other words, for a large p, (4) is able to directly reflect the hard decision rule mentioned above. We set $p = 10$. For convenience, we rewrite (4) in a form of output vector y_i and target vector t_i:

$$d^{(\mathrm{m})}(x_i,\Lambda) = -t_i \times y_i' + \left[\frac{1}{M-1}(1-t_i)\times (y_i^p)'\right]^{1/p}, \tag{5}$$

where $(\bullet)'$ means the transpose of vector $\bullet$. In (4) or (5), $d^{(\mathrm{m})} < 0$ indicates a correct class determination, whereas $d^{(\mathrm{m})} \geq 0$ results in a misclassification determination. With $d^{(\mathrm{m})}$, the misclassification error function (MCE) [11] is introduced as

$$\ell(x_i,\Lambda) = \frac{1}{1+\mathrm{e}^{-\xi(d^{(\mathrm{m})}(x_i,\Lambda))}}, \tag{6}$$

where $\xi > 0$, and $x_i \in \mathrm{cls}_m$. Obviously, (6) increases with the decrease in $d^{(\mathrm{m})}$. Optimising (6) is able to directly maximise the classification ability of the constructed model. Equation (6) is a smooth zero–one function and can be optimised by gradient decent type algorithms. The MCE on the whole data set $X = \{x_1, x_2, ..., x_N\}$ is the mean of MCE results on all patterns, i.e.

$$E_{\mathrm{mce}} = \frac{1}{N}\sum_{i=1}^{N}\ell(x_i,\Lambda). \tag{7}$$

As mentioned above, MCE function (7) evaluates the classification ability of a classifier in a direct way, which is a clear advantage of using MCE (7) over MSE (3) from the perspective of enhancing the classification performance.

For the discriminant-based classification approach, the choice of the discriminant function is crucial for classification performance. As discussed in [11], the discriminant functions are required to enable the corresponding loss function $\ell(x,\Lambda)$ to approach the error counts. In the case of RBF-based classifier, as is well known, the jth output unit corresponds to the discriminant function for the class cls_j, and Λ is the parameter set of the RBF network. Owing to the outstanding capability of approximating an arbitrary function to any degree of precision [2], RBF network can be used as a set of non-linear discriminant functions when each class is designed to each output unit. Hence, the MCE function can be used to train the RBF network for classification.

2.3 Problems of MCE and MSE

The problems of using MSE function in classification have been briefed in the first section. This section focuses on analysing the MCE function. In a batch version of gradient descent learning, the direction of updating is the average of negative gradients of all patterns. It is expected that the patterns far from being correctly classified should impose more effect on adjusting the parameters, while the patterns close to the target should have less effect. In this sense, the MSE function satisfies the requirement well. But, the MCE function (7) has the problem of satisfying the above requirement because $|E'_{\text{mce}}|$ decreases with the increase of d^{m} for misclassified patterns (i.e., $d^{\text{m}} > 0$). This shows that the data patterns that are far from being correctly classified may have small or even negligible effect on adjusting the parameters.

For example, for a data pattern x, the real output is $y = (y_1, y_2)$ and the target output is $t = (1, 0)$. We have the 2-dimensional MCE function

$$E_{\text{mce}} = \frac{1}{1 + \mathrm{e}^{-\xi(-y_1+y_2)}}.$$

Figure 2 illustrates the relationship between $y = (y_1, y_2)$ and the derivative $\partial E_{\text{mce}}/\partial y_1$. Table 1 lists several values of $\partial E_{\text{mce}}/\partial y_1$ and $\partial E_{\text{mse}}/\partial y_1$ and their corresponding outputs. Take points 1, 2 and 3 as examples. The situation of point 1 certainly satisfied the above requirement because its output is close to the target and $\partial E_{\text{mce}}/\partial y_1$ is very small. But for points 2 and 3, it is desirable to have a larger value of $\partial E_{\text{mce}}/\partial y_1$ to minimise the classification error. The $\partial E_{\text{mce}}/\partial y_1$ values of these points, however, are small. This illustrates that using gradient decent type training methods to optimise E_{mce} may not assure a satisfactory convergence rate and the subsequent classification results. As mentioned before, this shortcoming was addressed in [13] by dynamically controlling the smoothness parameter ξ. In [13], they started with a rather large value of ξ for a coarse but likely global searching and then gradually decreased the value of ξ for fine searching. This type of approach is, however, problem dependent and is not easy to implement.

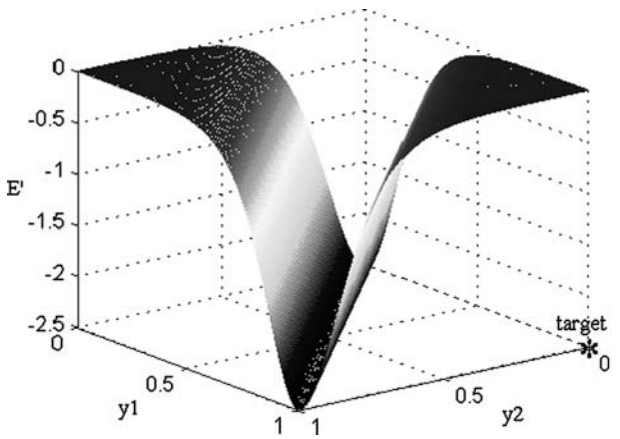

Fig. 2 The relationship between $\partial E_{\text{mce}}/\partial y_1$ and the classifier output $y = (y_1, y_2)$. Suppose that, for this 2-class, the target for an input data pattern is $t = (1, 0)$. Its MCE function is $E_{\text{mce}} = \frac{1}{1+\mathrm{e}^{-\xi(-y_1+y_2)}}$

3 Hybrid MCE–MSE cost function

The MCE function can construct an RBF-based classifier in a more direct way than the MSE function. But the MSE function exhibits certain advantages over the MCE cost function when gradient descent based training algorithms are used. In this paper, we propose a hybrid cost function:

$$E = \frac{1}{2}(E_{\text{mse}} + E_{\text{mce}}). \tag{8}$$

The hybrid cost function is able to encompass the properties of both MCE and MSE. The MSE part of this cost function enables patterns with large classification errors to be taken into account irrespective of the value of ξ in the MCE. In this sense, the shortcoming of MCE is compensated in a more simple way compared with heuristically decreasing ξ. On the other hand, the proposed hybrid cost function maintains the desirable property of MCE—the effect on adjusting the classifier parameters diminishes when the patterns are close to the targets. As a result, the proposed hybrid cost function provides a balanced performance on convergence rate and classification results.

In the following, the cost functions (3), (7) and (8) are compared by applying them to train RBF classifiers. The training algorithms based on the different cost functions were implemented in the same way, i.e., they employed the same learning scheme (the fast BP available in Matlab toolbox) and started from the same initial points. The structure of RBF network is detailed in Sect. 2.1. All these RBF classifiers are trained to solve the two-spiral classification problem (detailed in Sect. 5.1). The iterative training processes are stopped when the classification error has not decreased for 100 training epochs continuously. The averaged results of 10 trials are listed in Table 2 where ξ of the MCE function was set to 1. It shows that both the number of epochs and the running time required by the MCE cost function are significantly larger compared with the proposed MCE–MSE based cost function. Typical convergence curves are shown in Fig. 3. Due to the complex distinguish boundary of this problem, the training process generally begins with 50% classification error. In other words, a great part of patterns are far away from being correctly recognised. As mentioned above, to rapidly reduce the classification error requires that the misclassified patterns have more contributions than the correctly classified ones. MSE function (3) is able to meet this requirement, whereas MCE function (7) has difficulties to do so. Thus, the MSE and the MCE–MSE based gradient algorithms are much faster than the MCE-based one, as shown in Fig. 3. Also, in this figure,

Table 1 Comparisons of gradient value of MCE function and MSE function

Target $[y_1, y_2] = [1, 0]$			
Point no.	Output $[y_1, y_2]$	Gradient of MCE $\left[\frac{\partial E_{mce}}{\partial y_1}, \frac{\partial E_{mce}}{\partial y_2}\right]$	Gradient of MSE $\left[\frac{\partial E_{mse}}{\partial y_1}, \frac{\partial E_{mse}}{\partial y_2}\right]$
1	[0.9, 0.1]	[0.0031, − 0.0031]	[0.1000, − 0.1000]
2	[0.5, 0.9]	[0.3133, − 0.3133]	[0.5000, − 0.9000]
3	[0.1, 0.9]	[0.0061, − 0.0061]	[0.9000, − 0.9000]

Table 2 Comparisons of different cost functions through implementing the fast BP algorithms based on these functions

Cost function	Number of epochs	Running time (s)	Classification error (%)
MCE–MSE	5,380	267	2.0
MSE	6,040	383	12.6
MCE	15,340	1.31×10^3	1.9

we observe that attributed to the MCE function, the MCE–MSE based algorithm is able to achieve better classification results finally.

In this study, we also evaluate the effects of ξ briefly. Figure 4 shows comparative results between MCE–MSE and MCE for different values of ξ. Obviously, the results presented in Table 2 and Figs. 3 and 4 are consistent with the above analysis. The MCE and MCE–MSE based algorithm can enable RBF classifiers to achieve better classification results. In all our studied cases, MCE–MSE based algorithm consistently delivered a higher convergence rate than MCE. Also, we note that a relative small ξ, such as $\xi = 1$, is able to guarantee a respectable classification performance. Accordingly, a small ξ is used in our study.

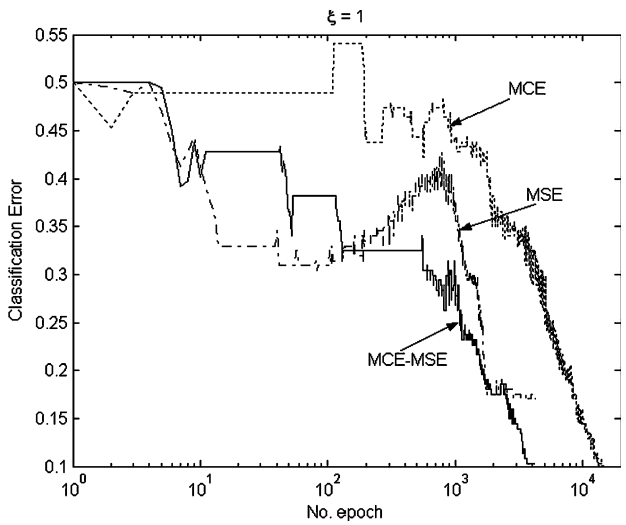

Fig. 3 A typical convergence curve of different cost functions (MCE–MSE, MCE, MSE). These comparisons are made on the two-spiral classification problem

Similar to the MCE-based approach, the setting of ξ must affect the performance of the proposed MCE–MSE based learning algorithm. However, ξ is fixed to 1 throughout this study in order to illustrate the natural properties of MCE–MSE function. It is worth mentioning that the overall performance of MCE–MSE can further be enhanced by dynamically adjusting the smoothness parameter ξ.

4 The MCE–MSE based training algorithm

As the initialisation of RBF network is not the focus of this paper, a commonly used method is employed. In this method, the centres of the RBF networks were initialised, using the SOM algorithm [15]. As all data are normalised to the interval [0, 1], the widths of hidden neurons in RBF network are determined as $\sigma = r_0 \times \mathrm{Id}$, where Id is the identity matrix, and we set $r_0 = 0.7$. The linear weights ($W^{(2)}$, $W^{(1)}$, B) of RBF network are randomly initialised with the values between 0 and 1.

After initialisation, the hybrid learning algorithm, as shown in Fig. 5, is adopted. The adjustment of parameters of hidden units is a non-linear process based on the MCE–MSE function (8), whereas the learning of weights between the hidden and output layers can be conducted in a linear way based on the MSE function (3). Based on Fig. 1, the function defined by RBF network can be expressed as

$$Y = f(X) = Z \times W^{(2)} + X \times W^{(1)} + B,$$

where Z is the output matrix of the hidden layer. The linear parameters in $f(X)$, i.e., $W^{(2)}$ and $W^{(1)}$, can be calculated by minimising the MSE function,

$$\begin{bmatrix} W^{(2)} \\ B \end{bmatrix} = \begin{bmatrix} Z \\ 1 \end{bmatrix}^{+} T, \qquad W^{(1)} = X^{+} T, \tag{9}$$

where $[\cdot]^{+}$ is defined as the pseudoinverse of $[\cdot]$.

In order to enhance the classification ability of RBF network, the MCE–MSE function (8) is employed to adjust the non-linear parameters, i.e., the parameters of the hidden layer. Both the MSE and the MCE functions are smooth and differentiable and can be conveniently minimised by the gradient descent method. The updating rule of the hidden layer parameter based on the MCE–MSE function (8) is

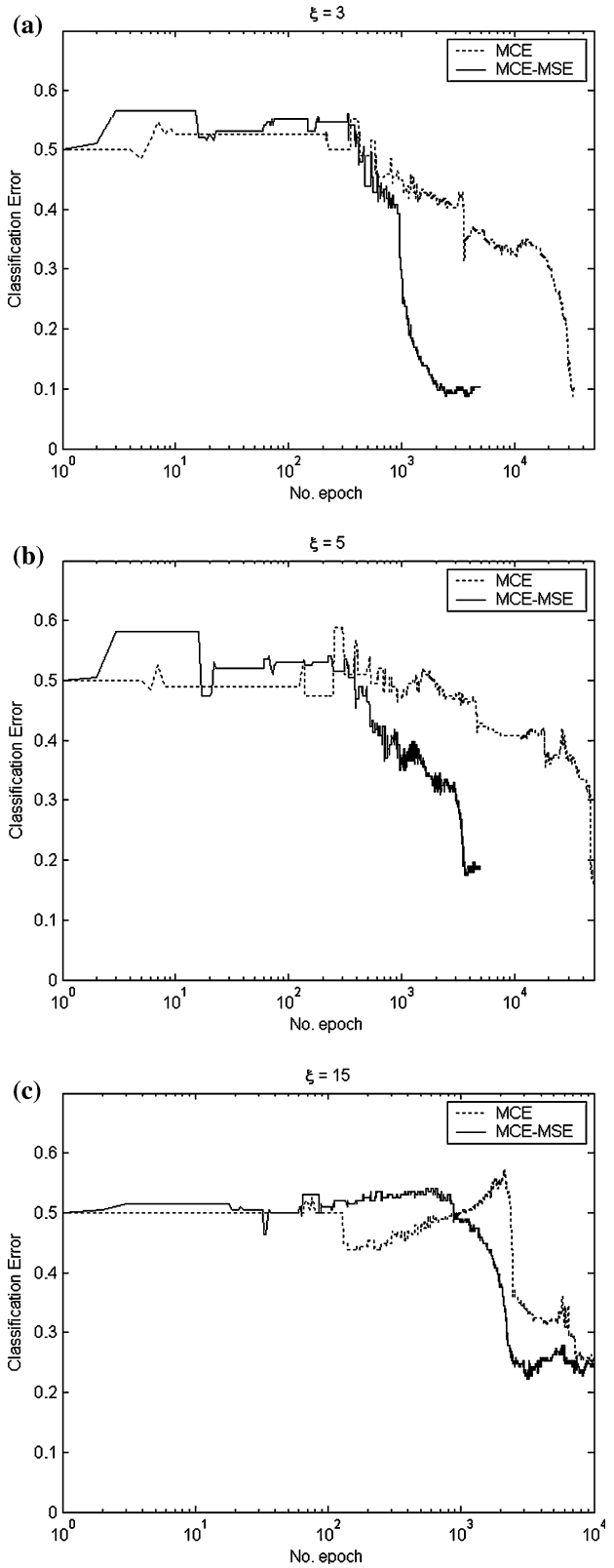

Fig. 4 The comparison of MCE–MSE and MCE function with different ξ. These comparisons are made on the two-spiral classification problem

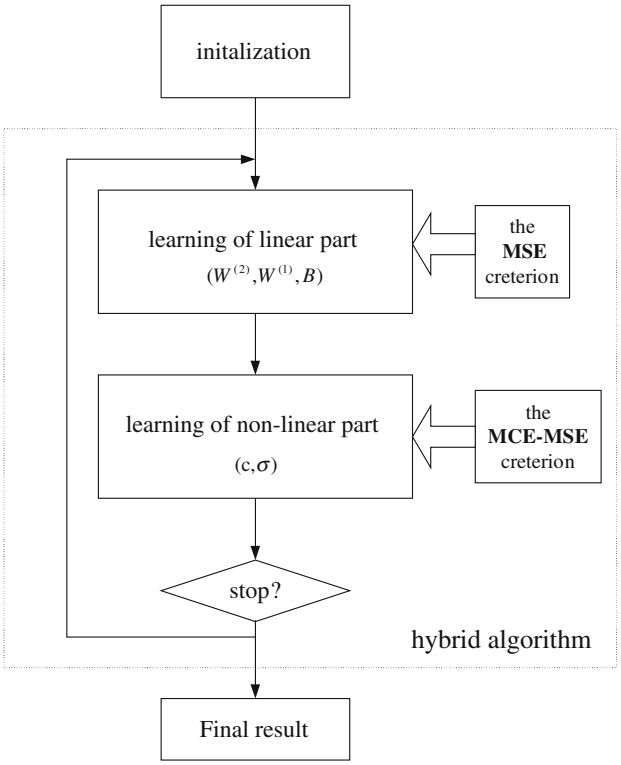

Fig. 5 Block diagram of the proposed hybrid algorithm based on the MCE–MSE function

$$\Lambda^{(\tau+1)} = \Lambda^{(\tau)} - \frac{1}{2}(\Delta\Lambda^{\text{mse}} + \Delta\Lambda^{\text{mce}})\big|_{\Lambda=\Lambda^{(\tau)}}, \tag{10}$$

where Λ^{τ} denotes the parameter c or σ at the τth step. The updating rules of Λ based on MSE, i.e. $\Delta\Lambda^{\text{mse}}$, have been explained in [3, 8]

$$\left(\Delta c_{jk}^{\tau+1}\right)_{\text{mse}} = -\eta(\tau)\sum_{i=1}^{N} g_i(x_i)\frac{x_{ik} - c_{jk}^{(\tau)}}{\left(\sigma_{jk}^{(\tau)}\right)^2} \times \sum_{r=1}^{J} w_{r=1}^{(\tau)}\left(t_{jr} - y_{ir}(\tau)\right), \tag{11}$$

$$\left(\Delta\sigma_{jk}^{(\tau+1)}\right)_{\text{mse}} = -\eta(\tau)\sum_{i=1}^{N} g_j(x_i)\frac{(x_{ik} - c_{jk}^{(\tau)})^2}{\left(\sigma_{jk}^{(\tau)}\right)^3} \times \sum_{r=1}^{J} w_{jr}^{(\tau)}(t_{ir} - y_{ir}(\tau)). \tag{12}$$

And, based on the definition of the MCE function (7), $\Delta\Lambda^{\text{mce}}$ is solved as follows:

$$\left(\Delta c_{jk}^{(\tau+1)}\right)_{\text{mce}} = \eta(\tau)\sum_{i=1}^{N} g_j(x_i)\frac{x_{ik} - c_{jk}^{(\tau)}}{\left(\sigma_{jk}^{(\tau)}\right)^2}\sum_{r=1}^{J} w_{jr}^{(\tau)}\frac{\partial E_{\text{mce}}}{\partial y_{ir}}. \tag{13}$$

Setting

$$\ell_i \equiv \ell(x_i, \Lambda), d_i^{(\mathrm{m})} \equiv d^{(\mathrm{m})}(x_i, \Lambda),$$

according to (5) and (7), we have

$$\frac{\partial E_{\mathrm{mce}}}{\partial y_{ir}} = \frac{\partial \ell_i}{\partial y_{ir}}, \tag{14}$$

$$\frac{\partial \ell_i}{\partial y_{ir}} = \xi \ell_i (1 - \ell_i) \frac{\partial d_i^{(\mathrm{m})}}{\partial y_{ir}}, \tag{15}$$

$$\frac{\partial d_i}{\partial y_i} = -t_i + \left[\frac{1}{M-1}\right]^{1/p} \left[(1 - t_i) \times (y_i^p)'\right]^{(1/p)-1} \times \left[(1 - t_i) \cdot \times y_i^{p-1}\right]. \tag{16}$$

In (16), $\cdot \times$ is a type of matrix operator, and $A \cdot \times B$ is the entry-by-entry product of matrix A and B. The modification rule of the centre c based on the proposed MCE–MSE (8) is

$$c_{jk}^{(\tau+1)} = c_{jk}^{(\tau)} - \frac{1}{2}\left(\left(\Delta c_{jk}^{(\tau+1)}\right)_{\mathrm{mse}} + \left(\Delta c_{jk}^{(\tau+1)}\right)_{\mathrm{mce}}\right),$$

where $\left(\Delta c_{jk}^{(\tau+1)}\right)_{\mathrm{mse}}$ and $\left(\Delta c_{jk}^{(\tau+1)}\right)_{\mathrm{mce}}$ are given in (11) and (13)–(16), respectively. In a similar way, the updating rule for the width σ based on the MCE–MSE function is

$$\sigma_{jk}^{(\tau+1)} = \sigma_{jk}^{(\tau)} - \frac{1}{2}\left(\left(\Delta \sigma_{jk}^{(\tau+1)}\right)_{\mathrm{mse}} + \left(\Delta \sigma_{jk}^{(\tau+1)}\right)_{\mathrm{mce}}\right),$$

where $\left(\Delta \sigma_{jk}^{(\tau+1)}\right)_{\mathrm{mse}}$ has been given in (12). And, $\left(\Delta \sigma_{jk}^{(\tau+1)}\right)_{\mathrm{mce}}$ can be delivered as

$$\left(\Delta \sigma_{jk}^{(\tau+1)}\right)^{\mathrm{mce}} = \eta(\tau) \sum_{i=1}^{N} g_j(x_i) \frac{(x_{ik} - c_{jr}^{(\tau)})^2}{\left(\sigma_{jr}^{(\tau)}\right)^3} \sum_{r=1}^{J} w_{jr}^{(\tau)} \frac{\partial E_{\mathrm{mce}}}{\partial y_{ir}}, \tag{17}$$

where $\partial E_{\mathrm{mce}} / \partial y_{ir}$ is calculated using (14)–(16).

5 Experimental results and discussion

In this section, the proposed MCE–MSE based method was compared to the MCE-based one and the MSE-based one by applying them to train RBF classifiers. The MSE-based method and the MCE–MSE based method are same as to using the LLS paradigm to adjust the linear weights, $W^{(2)}$ and $W^{(1)}$. The difference between the MSE and the MCE–MSE based methods lies in that the MCE–MSE method employs the MCE–MSE function to guide a gradient descent algorithm to update c and σ, while the MSE-based approach depends on the MSE function to fulfil this task. As to the MCE-based method, all parameters are adjusted by using the gradient descent algorithm based on the MCE function. In order to fairly compare these methods, the same gradient descent scheme used by different methods are implemented under the same scheme, the fast back-propagation (FBP) provided in Matlab 6.0. The parameters of FBP were set to default values. Using FBP, different methods employ different objective functions. In detail, the MSE-based FBP uses MSE (3) to update c and σ. And the MCE–MSE based FBP employs MCE–MSE (8) to undertake this task. The MCE-based FBP depends on MCE (7) to adjust all the parameters of the RBF network, including $W^{(2)}$, $W^{(1)}$, c and σ.

Several classification benchmarks and one industrial application were used for comparing these three training methods. In all cases, simulation results are the averages of 20 trials. And during each trial, with the same initial points, all three methods were tried. As mentioned above, the training methods are to be evaluated in two perspectives. To evaluate the classification ability, the classification error is used. To evaluate the convergence rate, the running time and the number of training epochs are employed. All simulations in the paper were done on an Intel Pentium IV 1.3 GHz PC with 256 RAM.

5.1 Two-spiral classification problem

The task of two-spiral classification benchmark [18] is to learn to discriminate between two sets of training points that lie on two distinct spirals in the 2-dimensional plane. The two-spiral benchmark consists of 194 data points, 97 points for each class. It is a highly non-linear and demanding classification problem. Without the test data set, various neural-based classifiers [19–22] are compared mainly in terms of convergence rate, i.e. the number of epochs required to achieve 100% training accuracy. The learning process stops when 100% training accuracy is obtained or when the training accuracy ceases to improve in the consecutive 100 training epochs. In this paper, besides the original two input variables, x_1 and x_2, two additional input variables were generated. They are $x_3 = (x_1^2 + x_2^2)$ and $x_4 = 1/(x_1^2 + x_2^2)$. Thus, the RBF classifier constructed in this section consists of 4 input neurons, 12 hidden neurons and 2 output neurons, 1 neuron for each class. The four input neurons are for x_1, x_2, x_3 and x_4, respectively.

In Table 3, the comparative results are listed, and the typical performances of the three types of training methods are shown in Fig. 6. Due to the complex distinguish boundary, at the beginning of the training process, the MCE–MSE based method exhibits the

Table 3 Performance comparison of the three training methods on the two-spiral problem

Method	Number of epochs	Running time (s)	Classification error (%)
MCE–MSE based	317	33	1.7 ± 3.1
MSE based	847	76	6.7 ± 3.5
MCE based	15,340	1.31×10^3	1.9 ± 1.5

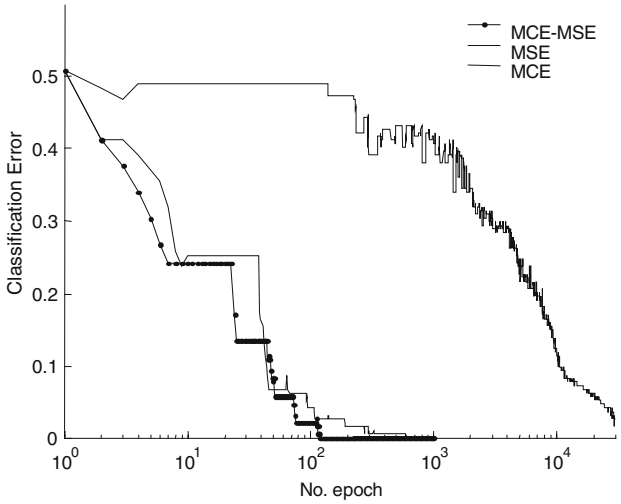

Fig. 6 A typical convergence curve of the three methods on the two-spiral data set

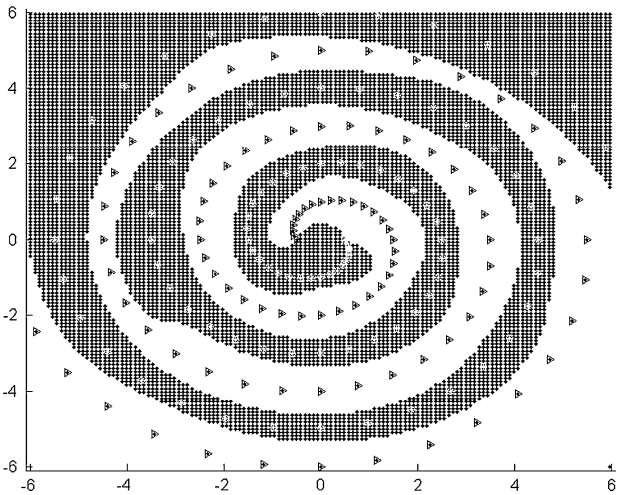

Fig. 7 The decision regions for two-spiral data by the RBF classifier trained by the MCE–MSE based method

largest convergence rate. The computational time required by the MCE–MSE based algorithm is 53 times less than that required by the MCE-based one. Also, compared with the MSE function, the MCE–MSE function enables the RBF-based classifiers to achieve better classification results. In Fig. 7, the decisions made by the RBF classifier trained by the MCE–MSE based method are illustrated.

In this section, RBF classifier trained by using the MCE–MSE based method is further compared with other neural classifiers. With similar classification performance—about 1% classification error—all these neural classifiers are compared in terms of the number of trainable parameters and convergence rate, as shown in Table 4. Obviously, the proposed MCE–MSE is the best one for the two-spiral classification problem in that it is the fastest one and has the smallest size of trainable parameter set.

Table 4 Comparisons on the two-spiral classification problem

Learning method/network	Number of trainable parameters	Number of epochs
BP (CE) [19]	138	11,000
CC [20]	181.75	1,700
MLP-IRO [21]	147.40	–
BP-NWP [22]	71	676
MCE–MSE (RBF)	168	317
MCE (RBF)	168	27,340

In [21], the network MLP-IRO was compared with CC in terms of the training time, instead of the number of epochs. The comparative results are that CC and MLP-IRO require 1.00×10^4 and 2.93×10^4 s for convergence, respectively

5.2 UCI classification problems

Several real classification data sets were employed in our investigation, which are from UCI archives [23]. Different from the two-spiral data set, the generalisation performance of the classifiers should be considered. In order to obtain better generalisation performance, there are many strategies. In our simulations, an easy one, "early stopping", was adopted. All iterative learning processes were periodically tested on the validation data set. The learning processes were stopped when the classification error on the validation set was larger than the best value achieved ten times consecutively. The number of hidden units of RBF-based classifier is related to several factors, such as the distribution of data patterns and the complexity of the decision surface of classification. It is, however, not the focus of this paper. In order to compare these training methods thoroughly, different numbers of hidden units were tried in the simulations. The classification problems used in our study include:

1. *Glass classification problem*. The glass data set is based on the chemical analysis of glass splinters. There are 214 data patterns, which split into three sets, 80 for training and 54 for validation and 80 for test. Each pattern consists of nine features and exactly belongs to one of six classes that correspond to the types of glass. The RBF-based classifiers had nine input units and six output units, and we considered RBF networks with hidden units of 4, 9, 12, 16.
2. *Waveform classification problem*. This task is to classify an instance into three categories of waves. There are 5,000 data patterns, and each pattern has 21 features. The 5,000 data patterns were split into three disjoint sets, i.e., 2,000 in training data set, 1,000 in validation data set and 2,000 in test data set. The RBF classifiers had 21 input units and 3 output units, and we tried RBF networks with hidden units of 9, 16, 25, 30.
3. *Other classification problems*. In Table 5, the other five classification data sets used in our study are listed.

Table 5 The UCI data sets used in this paper

Data set	Number of training patterns	Number of validation patterns	Number of test patterns	Number of attributes	Number of classes
Iris	60	30	60	4	3
Pima	270	228	270	9	2
Sonar	80	48	80	60	2
Wine	60	58	60	13	3
Yeast	600	284	600	8	10

Table 6 Comparison of the three training methods on glass classification problem. The results are average of 20 trials. H is the number of hidden units of RBF network. The running time is in second. (a) The performance of the compared methods. (b) The t test results to evaluate the classification ability improvement of the proposed method, the MCE–MSE based method. A small value indicates large improvement

(a)

H	Classification error on test data (%)			Convergence rate (number of epoch; running time)		
	MCE–MSE	MSE	MCE	MCE–MSE	MSE	MCE
4	35.4 ± 2.2	38.9 ± 1.1	35.3 ± 5.2	21; 0.8	24; 0.8	981; 36
9	34.1 ± 2.1	36.3 ± 3.9	34.2 ± 8.6	17; 0.9	20; 0.9	812; 46
16	33.9 ± 2.7	36.1 ± 4.3	33.8 ± 7.1	17; 2.0	19; 2.0	581; 50

(b)

H	t test in terms of classification accuracy	
	MCE–MSE versus MSE	MCE–MSE versus MCE
4	0.008	0.8
9	0.003	0.8
16	0.005	0.9

Table 7 Comparison of the three training methods on the waveform classification problem. The results are average of 20 trials. H is the number of hidden units of RBF network. (a) The performance of the compared methods. (b) The t test results to evaluate the classification ability improvement of the proposed method, the MCE–MSE based method. A small value indicates large improvement

(a)

H	Classification error on test data (%)			Convergence rate (number of epoch; running time)		
	MCE–MSE	MSE	MCE	MCE–MSE	MSE	MCE
9	20.3 ± 0.75	24.0 ± 1.05	19.5 ± 2.08	21; 13	23; 14	725; 419
16	19.1 ± 0.67	23.6 ± 0.74	19.9 ± 1.35	22; 35	32; 34	1077; 922
25	19.7 ± 0.82	22.2 ± 1.47	19.2 ± 1.93	30; 47	37; 55	623; 612

(b)

H	t test in terms of classification accuracy	
	MCE–MSE versus MSE	MCE–MSE versus MCE
9	3×10^{-5}	0.2
16	1.4×10^{-5}	0.45
25	8×10^{-4}	0.9

The comparative results are listed in Tables 6, 7 and 8. Besides calculating the average and the standard deviation of the obtained results, the results of the t test for unequal variances are presented in Tables 6b, 7b and 8b to further evaluate the significance of the improvement caused by the proposed MCE–MSE method. The small the presented t test result, the more the significance of the performance difference between the compared methods. Based on these results, two main conclusions can be drawn. First, from the perspective of

Table 8 Comparison of the three training methods on several UCI data sets. (a) The performance of the compared methods. (b) The *t* test results of certain comparisons. A small value indicates large difference of the compared results

(a)

Data set	*H*	Classification error on test data (%)			Convergence rate (number of epoch; running time)		
		MCE–MSE	MSE	MCE	MCE–MSE	MSE	MCE
Iris	9	5.7 ± 1.8	9.7 ± 1.8	6.3 ± 1.9	24; 0.17	25; 0.17	23; 0.29
Pima	16	25.0 ± 0.7	25.7 ± 0.9	24.9 ± 0.6	13 0.65	16; 1.0	24.8; 2.2
Sonar	9	33.9 ± 0.3	24.6 ± 0.8	30.1 ± 0.5	13; 0.43	13; 0.5	35; 1.2
Wine	6	5.5 ± 0	5.7 ± 0	5.4 ± 0.2	13; 0.2	13; 0.2	29; 0.4
Yeast	9	42.5 ± 0.9	43.3 ± 0.9	41.2 ± 4.2	26; 1.8	25; 1.8	25; 1.6

(b)

Data	*t* test in terms of classification accuracy		*t* test in terms of running time	
	MCE–MSE versus MSE	MCE–MSE versus MCE	MCE–MSE versus MSE	MCE–MSE versus MCE
Iris	7×10^{-5}	0.43	0.7	0.007
Pima	0.003	0.45	0.07	1.6×10^{-4}
Sonar	3×10^{-4}	0.02	0.88	1×10^{-5}
Wine	0.61	0.75	0.2	8×10^{-6}
Yeast	0.03	0.7	0.83	0.45

the classification performance, the MCE-based method and the proposed MCE–MSE one are better than the MSE-based method. Also, this type of improvement is significant in all examples but the one of wine. Second, the MCE–MSE based method exhibits faster convergence rate than that of the MCE cost function. Even, in certain examples, the efficiency of the MCE–MSE based method is substantially larger than that of the MCE-based one. In the example of waveform, the computational time required by the MCE–MSE based method is 13–32 times less than those required by the MCE-based one. In the glass classification problem, the computational time required by the MCE–MSE based method is 25–51 times less than those required by the MCE-based method.

5.3 Machine fault detection

In this section, RBF networks are applied for intelligently determining the degree of unbalance fault of a three-phase machine system. This practical problem has been detailed in [24–27]. The data employed in this section consists of 300 patterns, 100 for training, 50 for validation and 150 for testing. Each pattern has four attributes obtained from the vibration analysis of machine. All these patterns fall into three categories: light, medium and heavy degrees of electronic fault. In this section, the RBF classifiers have four input nodes and three output nodes corresponding to three classes. And learning schemes have been mentioned in the last section.

The simulation results of this application are presented in Table 9. According to these results, a similar conclusion can be drawn. In the perspective of the classification performance, the MCE-based method and the proposed one are much better than the MSE-based one. On the other hand, the proposed method is much efficient than the MCE-based method. In this example, the running time required by the MCE methods is eight times that of the proposed method.

6 Conclusion

A new type of cost function is proposed for enhancing the classification performance and convergence rate of RBF-based classifier. The proposed hybrid function can deliver improved classification results compared with the MSE function. Also, the MCE–MSE function itself exhibits a natural property of faster convergence rate than the MCE function. It is worth noting that the en-

Table 9 Comparison of the three methods on the machine default problem

H	Classification error on test data (%)			Convergence rate (number of epoch; running time)		
	MCE–MSE	MSE	MCE	MCE–MSE	MSE	MCE
4	5.6 ± 1.49	10.9 ± 6.90	5.3 ± 3.30	17; 0.3	15; 0.3	301; 3.4
8	5.6 ± 1.07	8.7 ± 5.27	4.5 ± 1.37	11; 0.3	15; 0.3	268; 4.5

H is the number of hidden units of RBF network

hanced convergence rate does not require the use of complicated algorithm for adaptively adjusting ξ. Hence, it is computationally efficient and easy for implementation. In all the presented cases, the proposed hybrid MCE–MSE based method had best performances, with much better classification result than the MSE-based one and much faster convergence rate compared with the MCE-based one.

References

1. Zhang GP (2000) Neural networks for classification: a survey. IEEE Trans Syst Man Cybern Part C 4:451–462
2. Girosi F, Poggio T (1990) Networks and the best approximation property. Biol Cybern 63:169–176
3. Poggio T, Girosi F (1990) Regularization algorithms for learning that are equivalent to multiplayer networks. Science 247:978–982
4. Haykin S (1999) Neural networks: a comprehensive foundation. Prentice Hall, New Jersey
5. Bishop C (1995) Neural networks for pattern recognition. Clarendon Press, Oxford
6. Howell AJ, Buxton H (1998) Learning identity with radial basis function networks. Neurocomputing 20:15–34
7. Ceccarelli M, Hounsou JT (1996) Sequence recognition with radial basis function networks: experiments with spoken digits. Neurocomputing 11:75–88
8. Schwenker F, Kestler HA, Palm G (2001) Three learning phases for radial-basis-function networks. Neural Netw 14:439–458
9. Rumelhart DE, Durbin R, Golden R, Chauvin Y (1995) Bachpropagation: the basic theory. In: Chauvin Y, Rumelhart DE (eds) Backpropagation: theory, architectures, and application. LEA, Hillsdale, pp 1–34
10. Lampariello F, Sciandrone M (2001) Efficient training of RBF neural networks for pattern recognition. IEEE Trans Neural Netw 12(5):1235–1242
11. Juang B-H, Katagiri S (1992) Discriminative learning for minimum error classification. IEEE Trans Signal Process 40(12):3043–3054
12. Watanabe H, Yamaguchi T, Katagiri S (1997) Discriminative metric design for robust pattern recognition. IEEE Trans Pattern Anal Mach Intell 45(11):2655–2662
13. Holmes CC, Mallick BK (1998) Bayesian radial basis function of variable dimension. Neural Comput 10:1217–1233
14. Kohonen T (1997) Self-organizing maps. Springer, Berlin Heidelberg New York
15. Kubat M (1998) Decision trees can initialize radial-basis function networks. IEEE Trans Neural Netw 9:813–821
16. Scholkopf B, Sung KK (1997) Comparing support vector machine with Gaussian kernels to radial basis function classifiers. IEEE Trans Signal Process 45:2758–2765
17. Available at http://www.openresource.com/openres/archives/P/AIR.shtml
18. Lang KJ, Witbrock MJ (1998) Learning to tell two spirals apart. In: Proceedings of the 1988 connectionist models summer school. Morgan Kaufmann, San Mateo
19. Fahlman SE, Lebiere C (1990) The cascade correlation learning architecture. In: Toursky DS (ed) Advances in neural information processing system 2. Morgan Kaufmann, San Mateo
20. Lengelle R, Denoeux T (1996) Training MLPS layer by layer using an objective function for internal representations. Neural Netw 2(1):83–97
21. Garavaglia SB (1999) The two spirals benchmark: lessons from the hidden layers. Int Jt Conf Neural Netw 2:1158–1163
22. Available at http://www.ics.uci.edu/~mlearn/MLRepository.html
23. Katagiri S, Juang BH, Lee CH (1998) Pattern recognition using a family of design algorithms based upon the generalized probabilistic descent method. Proc IEEE 86(11):2345–2373
24. Sito W, Chow TWS (2004) Induction machine fault detection: using SOM-based RBF neural networks. IEEE Trans Ind Electron 51(1):183–194
25. Chow TWS, Hai S (2004) Induction machine fault diagnostic analysis with wavelet analysis. IEEE Trans Ind Electron 51(3):558–565
26. Chow TWS, Tan HZ (2000) HOS-based nonparametric and parametric methodologies for machine fault detection. IEEE Trans Ind Electron 47(5):1051–1059
27. Chow TWS, Fei G (1995) Three phase induction machines asymmetrical fault identification using bispectrum. IEEE Trans Energy Convers 10(4):688–693