Neurocomputing 72 (2009) 1797–1802

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Three-phase strategy for the OSD learning method in RBF neural networks

Gh. A. Montazer [a,*], Reza Sabzevari [b], Fatemeh Ghorbani [c]

[a] School of Engineering, Tarbiat Modares University, P.O. Box 14115-179, Tehran, Iran
[b] School of Engineering, Islamic Azad University of Qazvin, Member of YRC, P.O. Box 14115-179, Tehran, Iran
[c] School of Basic Sciences, Tarbiat Modares University, P.O. Box 14115-179, Tehran, Iran

ARTICLE INFO

Article history:
Received 27 February 2008
Received in revised form
25 April 2008
Accepted 4 May 2008
Communicated by R. Capobianco Guido
Available online 29 July 2008

Keywords:
Optimum Steepest Descent method
Radial basis functions
Artificial neural networks
Three-phase OSD

ABSTRACT

This paper presents a novel approach in learning algorithms commonly used for training radial basis function (RBF) neural networks. This approach could be used in applications that need real-time capabilities for retraining RBF neural networks. The proposed method is a Three-Phase Learning Algorithm that optimizes the functionality of the Optimum Steepest Decent (OSD) learning method. This methodology focuses to attain greater precision in initializing the center and width of RBF units. An RBF neural network with well-adjusted RBF units in the train process will result in better performance in network response. This method is proposed to reach better performance for RBF neural networks in fewer train iterations, which is the critical issue in real-time applications. Comparing results employing different learning strategies shows interesting outcomes as have come out in this paper.

1. Introduction

Radial basis function (RBF) networks were introduced into the neural network literature by Broomhead and Lowe in 1988 [3]. These networks have been extensively used for interpolation regression and classification due to their universal approximation properties and simple parameters estimation [11].

RBF neural networks are some standard three-layer feed forward networks with the first layer consisting of n input units, a hidden layer consisting N_h RBF units and an output layer of m linear units [19]. Thus, the network computes the following function:

$$Y=(y_1,\ldots,y_s,\ldots,y_m):R^n\to R^m:y_s(X)=\sum_{j=1}^{N_h}w_{js}\varphi\left(\frac{\|X-C_j\|}{\sigma_j}\right) \quad (1)$$

where $w_{js}\in R$, $\varphi:R\to R$ is a proper activation function, typically a Gaussian one like

$$\varphi\left(\frac{\|X-C\|}{\sigma}\right)=e^{-(\|X-C\|/\sigma)^2} \quad (2)$$

where C is called *center* and the parameter σ usually called *width*. Also, the term $X=(x_1,\ldots,x_n)$ refers to the input vector.

The learning process for RBF neural networks is composed of initiating centers and widths for RBF units and computing weights for connectors of these units. Poggio et al. [19] considered the learning process of RBF networks as an ill-posed problem. They believe that training data could not be covering enough to construct a unique surface in some regions with unavailable data. From this point of view, learning will be closely related to classical approximation techniques, such as generalized splines and the regularization theory [19].

Besides the curve fitting capabilities of RBF networks, they are also suitable for pattern classification applications [22,8]. In classification problems, the RBF network has to perform a process to map continues input space R^d into a finite set of classes $Y=\{1,\ldots,K\}$, where K is the number of classes. Classification is performed by assigning the input vector X to the class of the output unit with maximum activation.

According to different applications of RBF neural networks, there is a wide variety of learning strategies that have been proposed in the literature for changing the parameters of a RBF network through the train process. These strategies are of two main categories. The first category contains strategies, in which centers and variances of the network are subjects of change, including the following:

1. fixed centers selected at random [25];
2. self-organized selection of centers, containing [18]:
   (a) K-mean clustering procedure,
   (b) the self-organizing feature map clustering procedure;
3. supervised selection of centers [25];
4. supervised selection of centers and variances [20].

* Corresponding author.
E-mail address: montazer@modares.ac.ir (G.A. Montazer).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.05.011

The second category includes strategies, in which weights of the network are changed, including the following:

1. the pseudo-inverse (minimum-norm) method [4],
2. the Least-Mean-Square (LMS) method [15],
3. the Steepest Decent (SD) method [5],
4. the Quick Propagation (QP) method [16],
5. optimized version of previous methods [16] including General Optimum Steepest Decent (GOSD), Optimum Steepest Decent (OSD) and Optimum Quick Propagation (OQP).

In previous works [16] the set of modified learning methods for RBF neural networks has been presented. These methods were achieved by improving the classical ones. In this paper, we introduce a three-phase learning strategy for RBF networks. This method is a hybridization of the OSD learning method and the classical two-phase learning. The two-Phase strategy is employed to improve the train performance while using the OSD learning method. Experimental results show that applying this method on real-time applications, such as the helicopter sound identification system [16], will result in better outcomes in comparison with the OSD methods.

The organization of this paper is as follows. The two-phase learning strategy is presented in Section 2. Employing the OSD method in two-phase learning is described in Section 3 as the Three-Phase Learning Method with OSD. Section 4 presents the implementation of the proposed method on several benchmark data sets. Also the performance of this method in comparison with previous ones are discussed in this section. And finally in the last section, Section 5, conclusion of this work is presented.

2. Two-phase learning for RBF networks

Using conventional learning methods, while employing RBF neural networks for real-time applications, will not satisfy the desired speed and performance in the train process. This fact made researchers develop new learning methods to be fast and accurate enough for their desired applications [16].

In this section, we discuss a learning method taking advantage of the well-defined meaning of RBF network parameters. In this approach, the learning process is divided into two consequent steps. The first step consists of determining centers of hidden units and center widths. Positions of centers should reflect the density of data points; thus various clustering or vector quantization techniques can be used for this reason. To set the center widths, we can use simple heuristic methods. The often-used one, called the p-nearest neighbor rule, simply evaluates the widths with respect to the average distance of the p-nearest neighboring units [16].

The second step is a supervised learning. The only parameters to be set are the weights between hidden and output layers representing the coefficients of linear combinations of RBF unit outputs. The objective is to minimize the overall error function with respect to these weights. Assume $\Phi_{ij}=\varphi_j(x_i)$ as the outcome of the jth radial basis function with the ith element of X, as the input vector, and Y_{ij} as the jth component of the ith target vector Y_i. Given these two matrices $\Phi=(\Phi_{ij})$ and $Y_d=(Y_{ij})$, the matrix representing weights of the output layer would be the result of minimizing the error function:

$$J(W)=\|Y_d-W\Phi^{\mathrm{T}}\|^2 \tag{3}$$

The vector

$$(W^*)^{\mathrm{T}}=\Phi^{+}Y_d^{\mathrm{T}} \tag{4}$$

represents the solution for such minimization problems, having the smallest Euclidean norm. Where Φ^{+} denotes the pseudo-inverse matrix of Φ, which in the classical two-phase learning method is computed by singular value decomposition (SVD) [23]. In this case, term Φ^{+} is defined as

$$\Phi^{+}=\lim_{\alpha\to 0}(\Phi^{+}\Phi+\alpha I)^{-1}\Phi^{\mathrm{T}} \tag{5}$$

It is obvious that a unitary matrix P and an upper triangular matrix R can be found such that $P\Phi=[R\,0]^{\mathrm{T}}$, so we have:

$$\min_W\|Y_d-W\Phi^{\mathrm{T}}\|=\min_W\|Y_d^{\mathrm{T}}-\Phi W^{\mathrm{T}}\|=\min_W\left\|PY_d^{\mathrm{T}}-\begin{bmatrix}R\\0\end{bmatrix}W\right\| \tag{6}$$

Introducing the residual as $r=Y_d-W^*\Phi^{\mathrm{T}}$ it follows by

$$\|W^*\|\leqslant\frac{\mathrm{cond}(R)\|r\|}{\|\Phi\|_\infty} \tag{7}$$

and

$$\|Y_{\mathrm{d}}\|\geqslant\|\Phi\|_\infty\cdot\|W\| \tag{8}$$

The linear least-square problem becomes more strongly ill-conditioned as the cond(R) increases, or as either of the inequalities Eqs. (7) and (8) is more nearly satisfied. For more details see [24].

As stability of this problem, we point out that if the matrix Φ^{T} does not have full rank, the solution W^* is not necessarily a continuous function of the data, so that small changes on these latter might produce large variations in W^*. This problem raised in real-applications when the data are noisy or when they are not easily discernible, e.g. the Two-Spiral classification problem. In practice, solutions to this problem are ill-conditioned due to small variances in some projections of data. In these cases, the term Φ^{+} in Eq. (4), which is used to compute the weights of the output layer, will cause augmented overall error in network performance [6]. This fact is a critical issue in the two-phase learning method. The term Φ^{+} is appeared by minimizing the $J(W)$ in Eq. (3). This fact is a critical issue in the two-phase learning method.

In our approach the steepest decent method with optimum learning rate (OLR) is used to minimize the error function. In this method, output weights approach optimum values from randomly generated initial ones through an iterative procedure [16]. The Two-Phase strategy is also applied to the OSD learning method to attain more speed in training RBF neural networks [17]. In the Two-Phase OSD method [17], the OSD algorithm is used instead of SVD to calculate weights between hidden and output layers. In that approach, the weights are initially set with random values.

3. Three-Phase Learning Method with Optimum Steepest Decent

In Multilayer Perceptron (MLP) neural networks, all parameters are usually adapted simultaneously by an optimization procedure [23]. This training procedure is supervised, since it minimizes an error function measuring the difference between the network output and the correct output. In contrast to training MLP neural networks, learning in RBF neural networks can be done in some distinct stages [16]. In this section, we propose a learning method in which the learning process is presented in three different steps. In our approach, the first phase is dedicated to adjusting the center of the RBF units; the second phase calculates centers' widths, and the calculating weights between hidden and output layers are assigned to the third phase. In the third phase the OSD learning algorithm is employed and the initial values for weights in this method are tried to be obtained from the SVD method. This technique decreases the chance of facing an ill-posed condition while calculating Φ^{+}. Also initial weights obtained from the SVD method, in comparison with random ones, increase the convergence speed of network performance. In our algorithm we first try to use SVD for initial weights but when

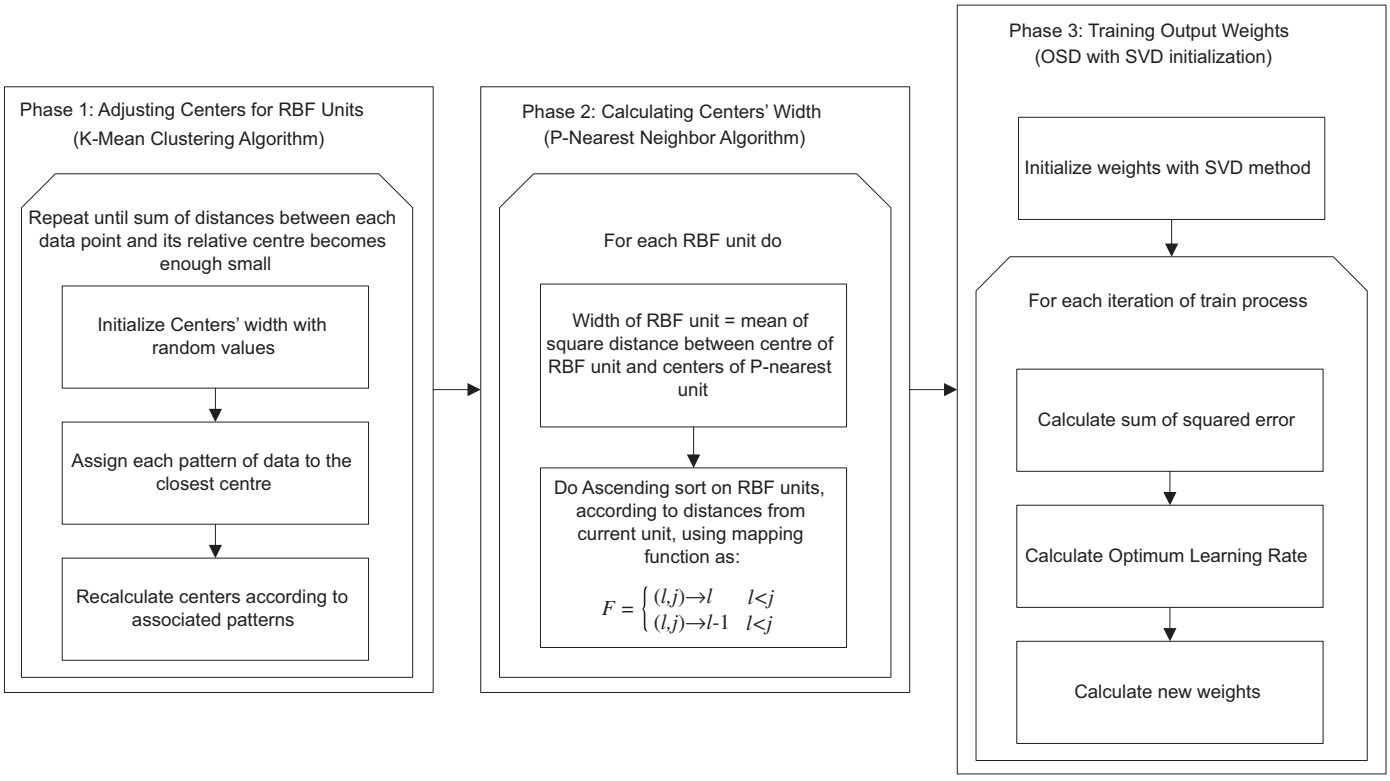

Fig. 1. Structure of the Three-Phase OSD method.

an ill-posed condition happens, random values are employed for this reason. Fig. 1 shows the proposed algorithm.

In the Three-Phase Learning Method, the error function is defined as

$$J = \tfrac{1}{2}\mathrm{E}\mathrm{E}^{\mathrm{T}} \tag{9}$$

where J is the sum squared error as in Eq. (3) and so E would be

$$E = Y_d - W\Phi^{\mathrm{T}} \tag{10}$$

The steepest decent method with OLR is used to minimize the term J. In this method, output weights approach optimum values from randomly generated initial ones through an iterative procedure [17]. Thus the term Φ^{+} would be omitted in computing the weights vector W. This fact is the reason of using the Three-Phase Learning Method.

3.1. Adjusting the centers of the RBF layer

To determine the centers of the RBF networks, typically an unsupervised training procedure from clustering techniques is employed [9]. In the remaining of this section we briefly describe K-means clustering to initialize the RBF centers.

3.1.1. K-means clustering algorithm

The K-means clustering algorithm starts with K random values as initial centers. Each pattern in the data set is assigned to the closest center, and finally the centers are recalculated according to the associated patterns. This process is repeated until a stopping criterion is satisfied [10]. This algorithm has the following steps:

1. Randomly initialize the K centers,
2. For each pattern, Z_p, in the data set compute its membership, $u(m_k|Z_p)$, to each center m_k and its weight $w(Z_p)$
3. Recalculate the K centers, using

$$m_k = \frac{\sum_{\forall Z_p} u(m_k|Z_p)w(Z_p)Z_p}{\sum_{\forall Z_p} u(m_k|Z_p)w(Z_p)} \tag{11}$$

The process will be repeated until some stopping criterion is satisfied. The most frequently used stopping criterion is a specified maximum number of iterations. Another termination criterion is the case in which the objective function of K-means (sum of distances between each data point and its centers) becomes smaller than some specified threshold.

The membership and weight functions for K-means are defined as

$$u(Z_p, m_k) = \begin{cases} 1 & \text{if } \mathrm{d}^2(Z_p, m_k) = \min\{\mathrm{d}^2(Z_p, m_j)\} \quad j = 1, \ldots, K \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

and $w(Z_p) = 1$.

When RBF neural networks are used for classification problems, parameter K is set to the number of classes. On the other hand, when the network is used in approximation problems, K is set to the number of neurons in the hidden layer, as in this literature is N_h. In approximation problems, the more the neurons in the hidden layer, the more the approximation precision.

3.2. Calculating the centers' width

Adjusting the centers' width is a critical issue in designing the architecture of an RBF neural network. Large widths will cause over-smoothing in the estimated probability density. Also the nature of the underlying true density may be lost. On the other hand, when widths are determined with too small values, some over-adaptation may happen in the particular data set. In

addition, very small or large widths tend to cause numerical problems with gradient descent methods as their gradients vanish [1]. Different schemes are investigated for initial settings of center widths in RBF networks [23,13].

In this paper, the p-nearest neighbor algorithm is used for investigating the initial settings of center widths in RBF units. In which, the width for each center is set to the mean of distances from the p-nearest prototypes of C_j [23], where the mean of squared distances between the center of cluster j and its p-nearest neighbors is calculated and the resulting value is assigned to parameter σ_j. All distances, $d_{ij}=\|C_l-C_j\|$, $l=1,\ldots,K$, $l\neq j$, are calculated and renumbered using a mapping function:

$$F=\begin{cases}(l,j)\rightarrow l & l<j\\ (l,j)\rightarrow l-1 & l>j\end{cases} \tag{13}$$

where j is the index of the current RBF unit and l is the index of clusters. Also, there is a permutation τ such that $d_{\tau(1)}\leqslant d_{\tau(2)}\leqslant\cdots\leqslant d_{\tau(K-1)}$ and σ_j is set to:

$$\sigma_j=\alpha\frac{1}{p}\sum_{i=1}^{p}d_{\tau(i)} \tag{14}$$

where $\alpha>0$ has to be set heuristically [23,12].

3.3. Training the output weights of the RBF network

To evaluate the output weights, we used the OSD learning method, which uses an OLR for each iteration of the train process [17].

3.3.1. Optimum Steepest Decent (OSD) method

In an RBF neural network we have $Y=[y_i]=W\Phi^{\mathrm{T}}$, $i=1,\ldots,M$ where Y is the estimated output vector. It is obvious that the error vector is

$$E=Y_d-Y=Y_d-W\Phi^{\mathrm{T}} \tag{15}$$

and the sum squared error, which should be minimized through the learning process, will be

$$J=\tfrac{1}{2}\mathrm{E}\mathrm{E}^{\mathrm{T}} \tag{16}$$

In the conventional SD method, the new weights are computed using the gradient of J in the W space:

$$\mathrm{OJ}=\frac{\partial J}{\partial W}=\frac{\partial((1/2)EE^{\mathrm{T}})}{\partial W}=(Y_d-Y)\frac{\partial Y}{\partial W}=E\frac{\partial(W\Phi^{\mathrm{T}})}{\partial W}=E\Phi \tag{17}$$

$$\Delta W=\mathrm{OJ}=E\Phi \tag{18}$$

$$W_{\mathrm{new}}=W_{\mathrm{old}}+\lambda\Delta W \tag{19}$$

where the coefficient λ is called learning rate, and remains constant throughout the learning process. It is clear that although Eq. (18) shows the optimum direction of delta weight vector, in the sense of first-order estimation, it still does not specify the optimum length of this vector and therefore the OLR. To achieve the OLR, the sum squared error of the new weights should be computed employing Eqs. (15), (16) and (19) as follows:

$$\begin{aligned}J(W)+\lambda\Delta W)&=\frac{1}{2}(Y_d-(W+\lambda\Delta W)\Phi^{\mathrm{T}})(Y_d-(W+\lambda\Delta W)\Phi^{\mathrm{T}})^{\mathrm{T}}\\&=\frac{1}{2}(E-\lambda\Delta W\Phi^{\mathrm{T}})(E-\lambda\Delta W\Phi^{\mathrm{T}})^{\mathrm{T}}\\&=\frac{1}{2}\mathrm{E}\mathrm{E}^{\mathrm{T}}-\lambda E\Phi\Delta W^{\mathrm{T}}+\frac{1}{2}\lambda^2\Delta W\Phi^{\mathrm{T}}\Phi\Delta W^{\mathrm{T}}\\&=A+B\lambda+C\lambda^2\end{aligned} \tag{20}$$

where $A=(1/2)\mathrm{E}\mathrm{E}^{\mathrm{T}}$, $B=-E\Phi\Delta W^{\mathrm{T}}$ and $C=(1/2)\Delta W\Phi^{\mathrm{T}}\Phi\Delta W^{\mathrm{T}}$ are scalar constants. Thus, $J(W+\lambda\Delta W)$ is a quadratic function of λ with constant coefficients A, B and C. $J(\lambda)$ will define a quadratic function of Φ with positive coefficients of the second-order term. Thus, it would have a minimum that can be found computing the derivative of $J(\lambda)$:

$$\frac{\partial J}{\partial\lambda}=\frac{\partial(A+B\lambda+C\lambda^2)}{\partial\lambda}=B+2\lambda C=0 \tag{21}$$

hence,

$$\lambda_{\min}=-\frac{B}{2C}=\frac{(E\Phi)(E\Phi)^{\mathrm{T}}}{(E\Phi\Phi^{\mathrm{T}})(E\Phi\Phi^{\mathrm{T}})^{\mathrm{T}}} \tag{22}$$

This learning rate minimizes the $J(\lambda)$, and so we can call it the OLR:

$$\lambda_{\mathrm{opt}}=\frac{(E\Phi)(E\Phi)^{\mathrm{T}}}{(E\Phi\Phi^{\mathrm{T}})(E\Phi\Phi^{\mathrm{T}})^{\mathrm{T}}}\geqslant 0 \tag{23}$$

Now the Optimum Delta Weight Vector (ODWV) can be determined as

$$\Delta W_{\mathrm{opt}}=\lambda_{\mathrm{opt}}\Delta W=\frac{(E\Phi)(E\Phi)^{\mathrm{T}}E\Phi}{(E\Phi\Phi^{\mathrm{T}})(E\Phi\Phi^{\mathrm{T}})^{\mathrm{T}}} \tag{24}$$

hence

$$W_{\mathrm{new}}=W_{\mathrm{old}}+\frac{(E\Phi)(E\Phi)^{\mathrm{T}}E\Phi}{(E\Phi\Phi^{\mathrm{T}})(E\Phi\Phi^{\mathrm{T}})^{\mathrm{T}}} \tag{25}$$

for which the initial value for W is set at random.

In the GOSD method the following new weight vector is used [16]:

$$W_{\mathrm{new}}=W_{\mathrm{old}}+\sum_{i=1}^{K}\lambda_i\Delta W_i \tag{26}$$

where K is the number of previous stages used to calculate the learning rate for the current stage.

As is apparent from Eqs. (24)–(26), matrices E, Φ and Φ^{T} are utilized to compute W. So, there would be no need of using the pseudo-inverse of matrix Φ. In other words we take a shortcut to solve the problem of the Two-Phase Learning method.

4. Experimental results

To benchmark the performance of the proposed learning method, we picked an audio processing application. In this application, we used samples of aerodynamically generated sounds of helicopters' main rotors to train an RBF neural network as the core of a helicopters' type identification system [2]. Different designs in helicopters cause variations in the aerodynamic behavior of different types of helicopters, which results in variations in wave shapes obtained form sound patterns [7,14]. Using this fact, our system recognizes the type of helicopters when it receives a sample sound generated by its rotors, of course if the system was trained for that type of helicopter. This sample application was previously used with some other optimized learning methods [16].

We have provided a vector of signal amplitudes for each sound pattern, which is sampled in 10 signal cycles as network inputs. The size of these vectors is assumed to be varying to make the system work on different sampling frequencies. Due to limitations in providing sample rotor sounds for creating train data, our system is initially trained with 14 cleared sample sounds from different types of military affairs helicopters categorized by their rotor configurations: Conventional, Tandem, Coaxial and Cynchropter helicopters. In this research we have assigned K to the number of helicopter types, so $K=14$. Also according to [21] parameters α and P are set to $\alpha=1$ and $P=2$.

In this paper we compare the results of our proposed methods with the classical optimized method, OSD and GOSD, on the same

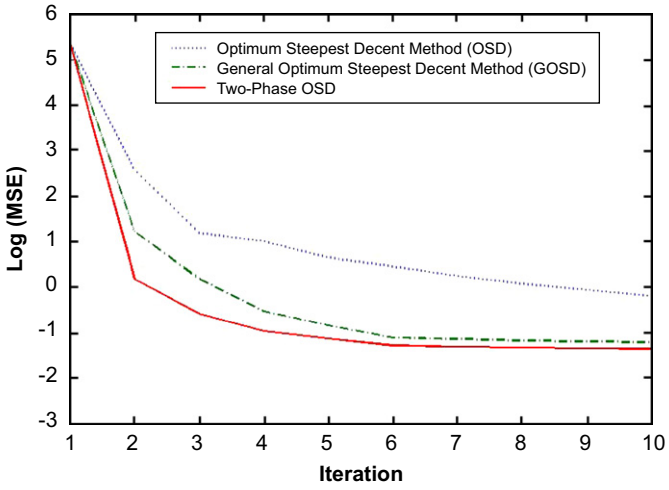

Fig. 2. Comparison of OSD, GOSD and Two-Phase OSD methods.

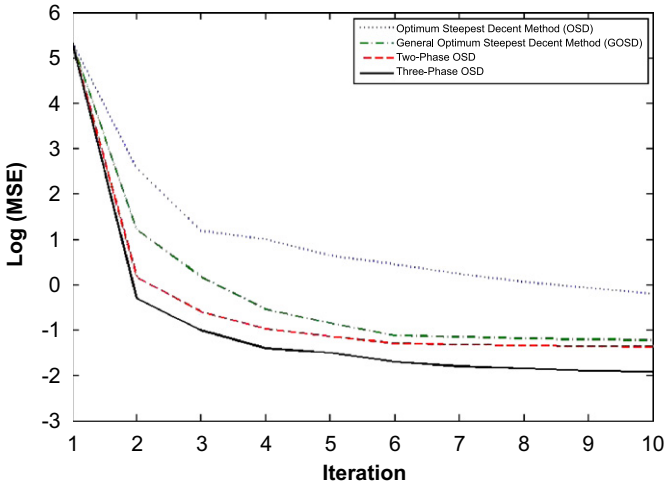

Fig. 3. Comparison of OSD, GOSD, Two-Phase OSD and Three-Phase OSD methods.

problem. We trained the RBF network with classical OSD, classical GOSD (when $K=3$), Two-Phase OSD and Three-Phase OSD methods, and the result of comparing these methods is shown in the Figures below. In Fig. 2, it is obvious that GOSD is faster than OSD and also Two-Phase OSD is faster than GOSD. The Two-Phase OSD method has fewer errors in fewer train iterations. This fact is because of well-adjusted centers and center widths at initial time. Also graphs in Fig. 3 show that the Three-Phase OSD reaches the desired performance in fewer epochs of the train.

Using different learning methods (Steepest Descent, Optimum Steepest Descent, Quick Propagation, Optimum Quick Propagation, General Optimum Steepest Descent, Two-Phase OSD and Three-Phase OSD), we trained our system for 90 epochs. To examine our system, we carried out sampling in a real noisy environment and passed the filtered signals to the neural network, as network inputs. Finally, the acquired results from different learning methods are compared together and show the better performance for our proposed method, as shown in Table 1.

In the phase of examining the system, as it is apparent in Table 1, we have tested the performance of training network using different train methods. For this reason, some different sample signals from different helicopter types are employed. According to diagrams showing network errors in consecutive train epochs,

Table 1
Comparing results obtained from conventional and modified training methods

Input signal	Identified helicopter type				Real signal types
	OSD	GOSD	Two-Phase OSD	Three-Phase OSD	
Signal 1	Type1	Type1	Type1	Type1	Type1
Signal 2	Type11	Type6	Type6	Type6	Type6
Signal 3	–	–	Type11	Type3	Type3
Signal 4	Type8	Type8	Type8	Type8	Type8
Signal 5	Type7	Type7	Type7	Type7	Type7
Signal 6	Type10	Type10	Type10	Type10	Type10
Signal 7	Type11	Type6	Type6	Type6	Type6
Signal 8	Type12	Type12	Type12	Type12	Type12
Signal 9	Type2	Type2	Type2	Type2	Type2
Signal 10	Type11	Type11	–	–	Type14

Figs. 2 and 3, the classical OSD method reaches a higher error value in 90 epochs of train in comparison with the newly introduced ones. This fact results in classification mistakes for our input data, which commonly have many similarities and also are noise dependent. On the other hand, as shown in Table 1, our proposed methods give more satisfactory results in this case.

Comparing the runtime of these methods, two major points will be apparent on this subject. As the Three-Phase OSD is involved with some calculations for the position of RBF centers and their width, it takes more time, while the network is being constructed, compared with previous versions of OSD. But as the network parameters, such as position and width of centers, are initially set to proper values, the train process will be accomplished in fewer epochs of train. Of course, when the same computer is used for this purpose, the fewer the number of epochs, the fewer the time to train. Results of this section are obtained from the same situation of equipments as in computer configuration and tuned parameters in software. As the affecting parameters of the network are center and width of the RBF units and weight of connections between these units, some comments on tuning these three parameters could be helpful. In the Three-Phase OSD, center and width of the RBF units are tuned using *K*-mean clustering and *P*-nearest neighborhood methods, respectively; however, in conventional OSD methods they are initially set randomly. The other parameter, weight of connections, is normally set through the learning process in all methods of training.

Generally, we can conclude that the training sample system employing convenient OSD resulted in 60% of truly identified sample signals, while using GOSD learning method increased the performance of system to 80% and using the Two-Phase OSD resulted in 86% of truly identified signals. And finally, employing the Three-Phase strategy on the OSD method increases the performance of the system up to 91%.

5. Conclusion

Developments of RBF neural networks and their learning methods are an important issue owing to the increasing interest of researchers for using this kind of neural network in different applications of engineering. The wide use of RBF neural networks is because of their strong abilities in classification and pattern recognition. This fact leads us to introduce a new strategy for training RBF neural networks. In this method we have hybridized the classical Three-Phase learning scheme with the OSD method. In the first two phases, radial parameters σ_j in the form of centers and widths for centers of RBF units are adjusted to proper values.

For this reason we have used the *K*-mean clustering algorithm for adjusting centers of the RBF layer as the first phase and the *P*-nearest neighbors' algorithm for calculating center widths as the second phase. In the third phase, the output weights W_k are determined by the OSD method. This method is performed on the basis that an audio processing application needs rapid retrain, which is the starting point for the idea of the Three-Phase OSD method as the previous methods could not satisfy the performance needed in the helicopter sound identification system proposed in [16]. Also the performance of the system is examined and compared with conventional methods, OSD and GOSD methods, and the Two-Phase OSD. Experimental results show that the proposed method, in comparison with other ones, converges to better performance in fewer epochs of train. This fact results in the increased accuracy and speed of the train process.

References

[1] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, USA, 1995.
[2] K.S. Brentner, F. Farassat, Modeling aerodynamically generated sound of helicopter rotors, Prog. Aerosp. Sci. 39 (2003) 83–120.
[3] D.S. Broomhead, D. Lowe, Radial basis functions, "Multi-Variable Functional Interpolation and Adaptive Networks," Complex Syst. 2 (1988) 321–355.
[4] R. Cancelliere, M. Gai, A comparative analysis of neural network performances in astronomical imaging, Appl. Numer. Math. 45 (2003) 87–98.
[5] X. Chen, Deformation measurement of the large flexible surface by improved RBFNN algorithm and BPNN algorithm, Lect. Notes Comput. Sci. 4493 (2007) 41.
[6] S. Cohen, N. Intrator, Global optimization of RBF networks, IEEE Trans. Neural Networks (2000).
[7] A.T. Conlisk, Modern helicopter rotor aerodynamics, Prog. Aerosp. Sci. 37 (2001) 419–476.
[8] T.M. Cover, Geomeasureal and statistical properties of systems of linear inequalities in pattern recognition, Electron. Comput. 14 (1965) 326–334.
[9] R. Gray, Vector quantization, IEEE Signal Process. Mag. 1 (1984) 4–29.
[10] H. Greg, C. Elkan, Alternative to the *K*-means algorithm that find better clustering, ACM Conference on Information and knowledge Management (CIKM 2002) (2002) 600–607.
[11] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice-Hall PTR Upper Saddle River, NJ, USA, 1994.
[12] N.B. Karayiannis, G.W. Mi, Growing radial basis neural networks: merging supervised and unsupervised learning with network growth techniques, IEEE Trans. Neural Networks 8 (1997) 1492–1506.
[13] M. Kubat, Decision trees can initialize radial-basis function networks, IEEE Trans. Neural Networks 9 (1998) 813–821.
[14] J.G. Leishman, Principles of Helicopter Aerodynamics, Cambridge University Press, Cambridge, 2006.
[15] I. Lin, C. Liou, Least-mean-square training of cluster-weighted modeling, Lect. Notes Comput. Sci. 4669 (2007) 301.
[16] G.A. Montazer, R. Sabzevari, H.G. Khatir, Improvement of learning algorithms for RBF neural networks in a helicopter sound identification system, Neurocomputing 71 (2007) 167–173.
[17] G.A. Montazer, R. Sabzevari, F. Ghornabi, Improvement Of Learning Rate For RBF Neural Networks In A Helicopter Sound Identification System Introducing Two-Phase OSD Learning Method, 5th International Symposium on Mechatronics and Its Applications, Amman, Jordan, 2008.
[18] T. Mu, A.K. Nandi, Detection of breast cancer using v-SVM and RBF networks with self organized selection of centers, The Third IEE International Seminar on Medical Applications of Signal Processing (2005) 47–52.
[19] R. Neruda, P. Kudová, Learning methods for radial basis function networks, Future Gener. Comput. Syst. 21 (2005) 1131–1142.
[20] K. Okamato, S. Ozawa, S. Abe, A fast incremental learning algorithm of RBF networks with long-term memory, in: Proceedings of the International Joint Conference on Neural Networks, vol. 1, 2003.
[21] P. Picton, Neural Networks, Palgrave Houndmills, Basingstoke, Hampshire, 2000.
[22] T. Poggio, F. Girosi, C. MIT, Networks for approximation and learning, Proc. IEEE 78 (1990) 1481–1497.
[23] F. Schwenker, H.A. Kestler, G. Palm, Three learning phases for radial-basis-function networks, Neural Networks 14 (2001) 439–458.
[24] J. Stoer, R. Bulirsch, Introduction to Numerical Analysis, Springer, Berlin, 2002.
[25] K.K. Tan, K.Z. Tang, Adaptive online correction and interpolation of quadrature encoder signals using radial basis functions, IEEE Trans. Control Syst. Technol. 13 (2005).

Gh. A. Montazer received his B.Sc. degree in Electrical Engineering from Kh. N. Toosi University of Technology, Tehran, Iran, in 1991, his M.Sc. degree in Electrical Engineering from Tarbiat Modares University, Tehran, Iran, in 1994, and his Ph.D. degree in Electrical Engineering from the same university, in 1998. He is an Assistant Professor of the Department of Information Engineering in Tarbiat Modares University, Tehran, Iran. His areas of research include Information Engineering, Knowledge Discovery, Intelligent Methods, System Modeling, E-Learning and Image Mining.

Reza Sabzevari received his B.Sc. degree in Computer-Hardware Engineering from Islamic Azad University of Qazvin, Iran, in 2003 and his M.Sc. degree in Mechatronics Engineering from the same university in 2007. His research interests center around Machine Vision, Machine Learning, Artificial Intelligence, Information Engineering and Data Mining. He is a member of Young Researchers' Club.

Fatemeh Ghorbani received her B.Sc. degree in Applied mathematics from University of Birjand, Iran, in 2004 and her M.Sc. degree in Applied Mathematics from Tarbiat Modares University, Tehran, Iran, in 2007. Her research interests center around Artificial Intelligence, Meta-Heuristic Algorithms, Clustering and Pattern Recognition.