Indian Journal of Science and Technology, *Vol 8(12), 70636, June 2015*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Optimization of the Artificial Neural Networks Structure for Filtering Applications in Wind Energy Conversion System

Karim Beddek[1*], Mohamed Kesraoui[1] and Adel Merabet[2]

[1]Laboratory of Applied Automation (LAA), University of Boumerdes Avenue de l'independance - 35000, Boumerdes, Algeria; k_beddek@yahoo.fr

[2] Laboratory of Control Systems and Mechatronics (LCSM), Division of Engineering, Saint Mary's University, Halifax, NS, Canada

Abstract

The approximation of a complex function by an Artificial Neural Networks (ANN) of Radial basis Function (NRF) conducts to large size architectures (a considerable number of neurons in the hidden layer) which minimize the quality of the network generation. This paper develops and implements an algorithm based on the Newton's method for learning and searching for an optimal structure of a radial basis network. The algorithm reduces automatically the number of neurons in the hidden layer of the ANN and also reaches the minimal architecture without deteriorating the learning error. Tovalidate the algorithm, simulation of filtering noise from a noisy sinusoidal signal has been performed using MATLAB/SIMULINK. Satisfying results have been obtained and application as a filter for fault detection in a Wind Energy Conversion System (WECS) is intended.

Keywords: Complex Systems Modeling, Filtering for Fault Detection, Learning Newton's Method, Neural Networks Optimization, WECS

1. Introduction

Neural networks with radial basis function are Feedforward networks with one hidden layer (Figure 1). The first use of this type of neural networks dates from 1970s. They were used to solve problems in multivariable interpolation. The theoretical bases of these networks have been depened and further works have been done; like NRF applications development and enlargement[1]. These networks are distinguished by their ability to provide a local representation of the space using radial basis functions $\rho\left(\|\cdot\|\right)$, where influence is restricted to some regions of the space.

In order to minimize the number of neurons an algorithm that eliminates the weak neurons is proposed.

$\|\cdot\|$ represents the Euclidean norm.

*Author for correspondence

2. Neural Networks of Radial basis Function (NRF)

2.1 Basic Concepts

A radial basis function is as follows:

$$\rho_i(x) = \rho\left(\left\|x - \mu_i\right\|, \sigma_i\right) \tag{2.1}$$

two parameter can be distinguished:

$$\begin{cases} \mu_i : \text{reference vector} \\ (\text{center or prototype}) \\ \sigma_i : \text{dimension of the influence field} \\ (\text{influence radius}) \end{cases}$$

The mostly used basis function is the Gaussian[2, 3]. It is expressed in its most general form as follows:

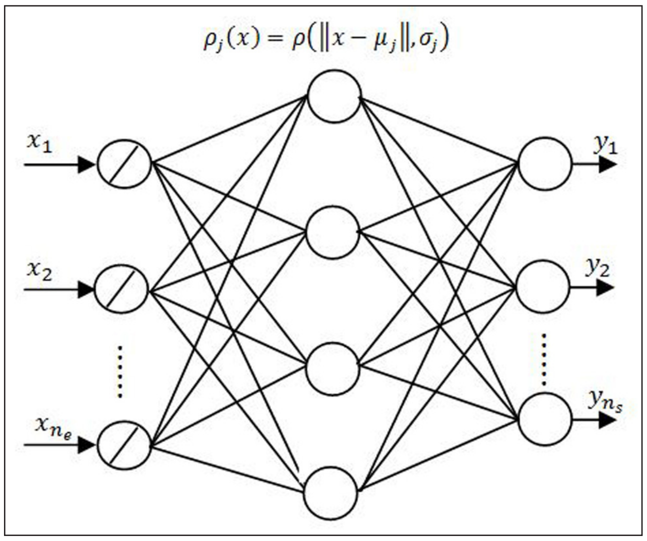

Figure 1. Network of radial based functions

$$\rho_i(x) = \exp\left(-\frac{1}{2\sigma_i^2}(x-\mu_i)(x-\mu_i)^t\right) \tag{2.2}$$

Where σ_i^2 designates the variance associated to the cell. The Gaussian decreases in the same way in all directions of the space. The iso-activation curves of the hidden cells are then hyper spheres[6]. For a given input data, a restricted number of basic functions contribute to the output calculation.

The *NRF* can be classified into two categories, depending on the output neuron[4,5].

- Normalized : $y(x) = \dfrac{\sum_j w_j \rho_j(x)}{\sum_j \rho_j(x)}$ (2.3)

- Non-Normalized : $y(x) = \sum_j w_j \rho_j(x)$ (2.4)

In this paper, the values of the centers and the radii are initially fixed then they are varied by the algorithm. The arrangement of centers and radii, defines a kind of paving of the input space. In this case, this procedure represents a preliminary phase for a network of neurons called neutral.

The centers are uniformly arranged in the domains of approximation. The number *n* of centers and the number n_e of system inputs define a unique neutral network whose architecture is of neurons n^{n_e} .Then our algorithm will reduce this number by removing weak neurons.

Figure 2 illustrates the recovery rate τ of a radial basis neural network into a unidirectional regular lattice given by[6]:

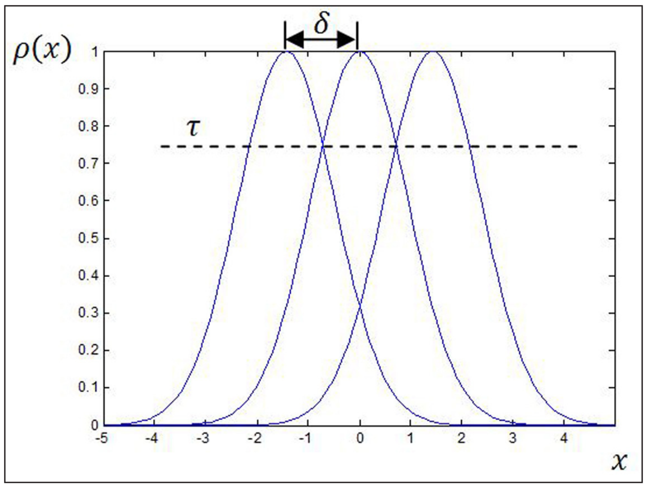

Figure 2. Gaussians with one-dimensional and regular lattice.

$$\tau = \exp\left[-\frac{1}{2}\frac{\left(\frac{\delta}{2}\right)^2}{\sigma^2}\right] \tag{2.5}$$

with δ being the distance between two adjacent centers.

From equation (2.5), we conclude the equation of the radius, for a fixed value of the recovery rate and the distance between the centers of the Gaussian functions.

$$\delta = (c_1 - c_0)/(n-1) \tag{2.6}$$

The distance between the centers δ is a function of the first center c_0, the last center c_1 and the number of centers *n*:

$$\sigma = \frac{\delta}{(2\times\sqrt{-2\times\log(\tau)})} \tag{2.7}$$

It is therefore, the approach of the dynamics of a complex function that we need, to do so, samples are *t* drawn uniformly from an interval of time, with a given sampling period. We define the cost function *J* by the following equation.

We define the cost function *J* by the following equation [7], [8]:

$$J = \frac{1}{2t}\sum_{i=1}^{t}\|\hat{y}_i - y_i\|^2 \tag{2.8}$$

Where y_i represents the output vector of the real process at each incremental instant*i*, $\hat{y}_i$ is the output vector of the neural network as it is given by the following equation:

$$\hat{y}_i = \frac{1}{R}\sum_{j=1}^{n^{ne}} w_{ij}\rho_j \tag{2.9}$$

with R being the normalization term given by the following equation:

$$R = \sum_{j=1}^{n^{ne}} \rho_j \tag{2.10}$$

Where:

$$\rho_j(u) = exp\left(-\frac{1}{2}\sum_{k=1}^{n_e}\frac{(x_k - c_{jk})^2}{\sigma^2 jk}\right)$$
$$= exp\left(-\frac{1}{2}(x - c_j^T)^T Q_j (x - c_j^T)\right) \tag{2.11}$$

The vector of center c_j^T has the same dimension as the input vector of the system (x) and Q_j is a square matrix which represents the radii of the Gaussian, given by:

$$Q_j = \left(diag\left[\frac{1}{\sigma_{j1}}\frac{1}{\sigma_{j2}}\frac{1}{\sigma_{j3}}\cdots\frac{1}{\sigma_{jn_e}}\right]\right)^2 \tag{2.12}$$

3. Proposed Algorithm Strategy

3.1 Weak Neurons

The response of the base function depends on the length of the input vector x (distance from x to the prototype μ_i), and the size of the influence field σ_i .The function $\rho_i(x)$ is generally maximal when ($x = \mu_i$) , and decreases monotonically to 0 when $\|x - \mu_i\| \to \infty$.

In case of a network with several inputs (Figure 1), x is a matrix of dimension ($n_e \times n_i$). With n_e being the number of inputs and n_i is the number of learning samples:

$$X = \begin{bmatrix} x_1(1) & x_1(2) & \cdots & x_1(n_i) \\ x_2(1) & x_2(2) & \cdots & x_2(n_i) \\ \vdots & \vdots & \vdots & \vdots \\ x_{n_e}(1) & x_{n_e}(2) & \cdots & x_{n_e}(n_i) \end{bmatrix} \tag{3.1}$$

The matrix ϕ represents the responses of radial basis functions in each sampling instant.

$$\phi = \begin{bmatrix} \rho_1(1) & \rho_1(2) & \cdots & \rho_1(n_i) \\ \rho_2(1) & \rho_2(2) & \cdots & \rho_2(n_i) \\ \vdots & \vdots & \vdots & \vdots \\ \rho_{n_c}(1) & \rho_{n_c}(2) & \cdots & \rho_{n_c}(n_i) \end{bmatrix} \tag{3.2}$$

n_c: The number of neurons in the hidden layer.

The matrix ϕ can be written as follows:

$$\phi = \begin{bmatrix} \rho_1(i) \\ \rho_2(i) \\ \vdots \\ \rho_{n_c}(i) \end{bmatrix}; \text{with} : i = \overline{1 : n_l} \tag{3.3}$$

Given F_j the function defined as:

$$F_j = \sum_{i=1}^{n_i} \rho_j(i) \quad ; \text{with} : j = \overline{1 : n_c} \tag{3.4}$$

The weight of connection between the output neuron i and j of the hidden layer is given by w_{ij}. The weight matrix W id given by this equation:

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n_c} \\ w_{21} & w_{22} & \cdots & w_{2n_c} \\ \vdots & \vdots & \vdots & \vdots \\ w_{n_s1} & w_{n_s2} & \cdots & w_{n_sn_c} \end{bmatrix} \tag{3.5}$$

With n_s being the number of output neurons.

The variation of these weights is done by the Newton's method.

By referring to the equations describing the output $\hat{y}_i$ of the network in function of the inputs, the values of the Newton's method are given by the following equations:

First derivative of the criterion:

$$\frac{\partial J}{\partial w} = \frac{\partial}{\partial w}\left(\frac{1}{2t}\sum_{i=1}^{t}\|\hat{y}_i - y_i\|^2\right) = \frac{1}{t}(\hat{y}_i - y_i)\frac{\rho^T}{R} \tag{3.6}$$

Second derivative of the criterion:

$$\frac{\partial}{\partial w}\left(\frac{\partial J}{\partial w}\right) = \frac{\partial}{\partial w}\left(\frac{1}{t}(\hat{y}_i - y_i)\frac{\rho^T}{R}\right) = \frac{1}{t}\left(\frac{\rho}{R}\frac{\rho^T}{R}\right) \tag{3.7}$$

if we put:

$$\begin{cases} D1 = t\dfrac{\partial J}{\partial w} \\ D2 = t\dfrac{\partial}{\partial w}\left(\dfrac{\partial J}{\partial w}\right) \end{cases} \tag{3.8}$$

The correction of the weights is given as:

$$w^{k+1} = w^k - [D2^{-1}D1^T]^T \tag{3.9}$$

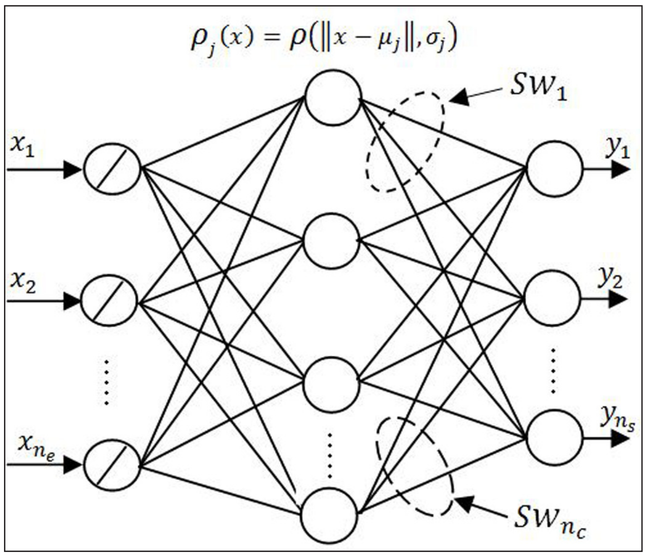

Figure 3. Intermediate layer neurons weights summation.

with k being an index of iteration.

given Sw_j (see figure 3) a function defined as follows :

$$Sw_j = \sum_{i=1}^{n_s} w_{ij} \quad ; \text{with} : j = \overline{1 : n_c} \qquad (3.10)$$

We define A_j as follows:

$$A_j = Sw_j \times F_j \qquad (3.11)$$

A_j represents the amount of influence of the neuron j of the hidden layer on the network outputs. If A_j is weak, the influence of the neuron number j of the hidden layer over the outputs of the network will be weak and can be neglected. It is on this reasoning that the principle of elimination of weak neurons in our algorithm is based.

3.2 Learning

The learning methods of neural networks with radial basis functions, such as the gradient method, the method of conjugate gradients, the gradient method with optimal step, the method of Newton, have achieved great success in many applications. However, a problem is frequently encountered when doing learning of complex networks of neurons (that is, when the function to be minimized is multimodal). This problem is convergence to local optima. At the end of the iteration process, we obtain not the global optimum of the error function, but one of its local optima.

Our algorithm is based on the following principle: it gives a significant number of centers for the network (a large number of neurons in the hidden layer) arranged uniformly in the areas of approximation (see Section 2.1), then whenever weak neurons are eliminated (see Section 3.1), we vary the centers and radii and we refine the learning of the weights, in such a way to get a decrease or stability of the criterion to be minimized.

4. Weak Neurons Elimination Method

```
1. Begin (parameters initialization and network construction) :
   Choice of the recovery rate ($\tau$).
   Choice of the number of centers ($n$).
   Choice of the step: $\Delta\sigma$ and $\Delta n$ .
   Choice of the error constant: $err$.
   Calculation of the distance between centars ($\delta$).
   Calculation of radius $\sigma$ .
   Calculation of the number of neurons in the hidden layer ( $n^{n_e}$ ).
   Random initialization of the networks weights $w$.
   Neutral network construction and network outputs calculation ($\hat{y}$).
   Calculation of the criterion $J$.
2. Doing the learning of weights by the Newton's method
3. **While** the stop criterion is not verified, **do:**
   **While** $J(t+1) =< J(t) + err$ **do**
       a) Calculation of $A_j$ .
       b) Classify the neurons of the hidden layer.
       c) Eliminate the weak neurons.
       d) Relearning of the weights.
   **End while**
   **While** $J(t+1) =< J(t)$ **do**
       1) Variation of radius : $\sigma = \sigma + \Delta\sigma$ .
       2) Relearning the weights.
   **End while**
   **For** i = 1 to $n_c$ **do**
       $n_i = n_i + \Delta n$.
       **While** $J(t+1) =< J(t)$ **do**
           $n_i = n_i + \Delta n$.
       **End while**
       $n_i = n_i - 2\Delta n$.
       **While** $J(t+1) =< J(t)$ **do**
```

```
        $n_i = n_i - \Delta n$.
      End while
    1) $n_i = n_i + \Delta n$.
    2) Relearning of weights.
    3) $i = i + 1$
    End for
4. go to step 3.
```

5. Application of the Method

Our aim consists to the use of the neural networks as a filter with two inputs $x_i(k)$ and two outputs $\hat{y}_i(k)$, for this we chose two noisy sine functions with a zero-mean Gaussian white noise sequence with covariance Q_{v_i}, and we initialize the network by four centers (n = 4), that is, sixteen neurons in the hidden layer ($n^{n_e} = 16$).

The neural networks inputs are given by the following equation:

$$\begin{cases} x_1(k) = 380\sqrt{2}\sin(\omega k) + V_1(k) \\ x_2(k) = 380\sqrt{2}\sin\left(\omega k + \frac{3}{2}\pi\right) + V_2(k) \end{cases} \quad (5.1)$$

When $V_i(k)$ is a zero-mean Gaussian white noise sequence with covariance Q_{v_i}, k being an index of iteration and ω is the frequencies of the signal.

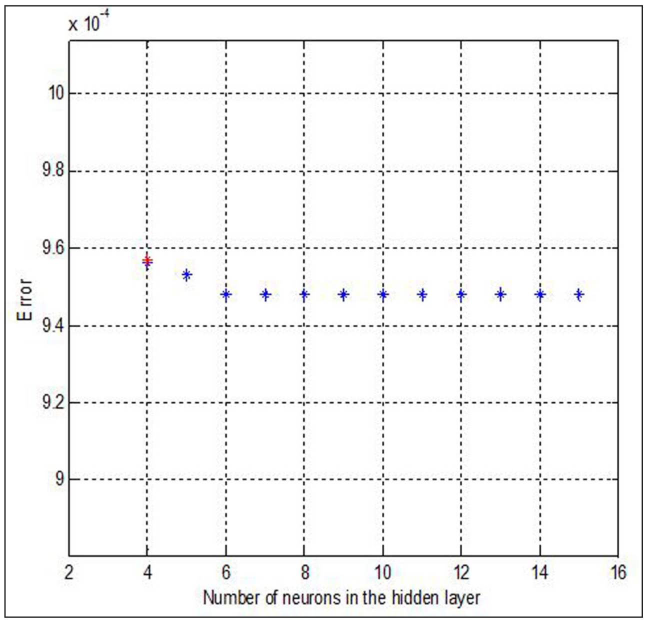

Figure 4. Number of neurons in the hidden layer and learning error, in red the number of neurons retained.

To use the neural networks as a filter to be used as desired outputs of network the inputs without noise whether:

$$\begin{cases} \hat{y}_1(k) = 380\sqrt{2}\sin(\omega k) \\ \hat{y}_2(k) = 380\sqrt{2}\sin\left(\omega k + \frac{3}{2}\pi\right) \end{cases} \quad (5.2)$$

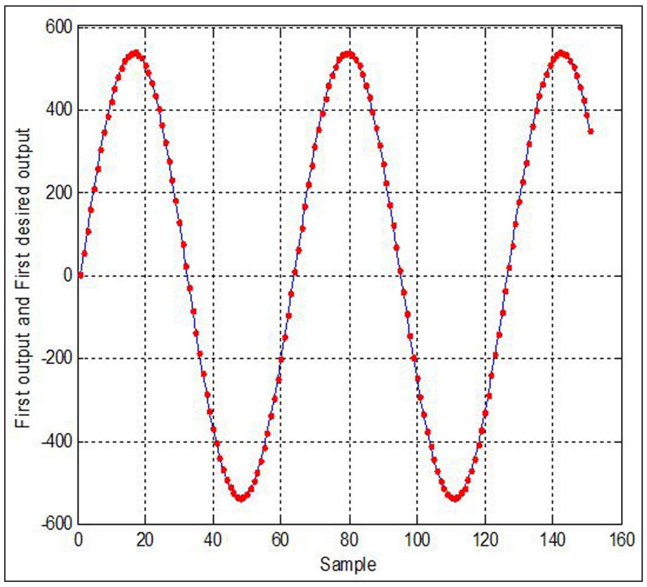

Figure 5. Learning phase, first network output in red and first desired output in blue

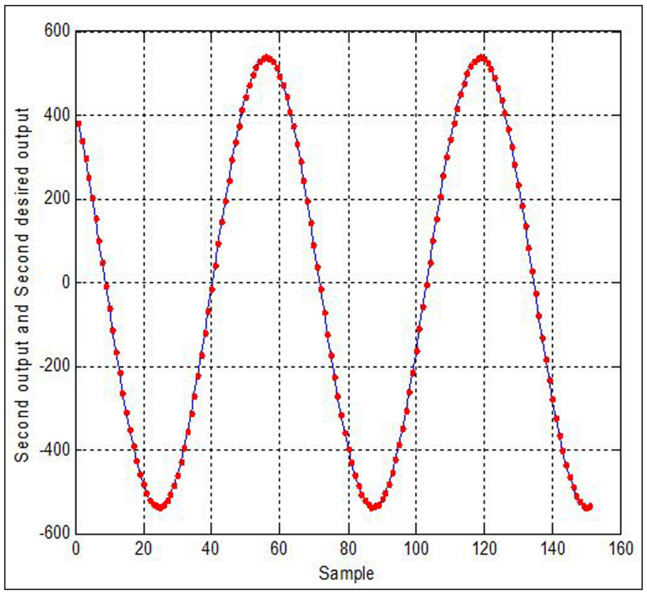

Figure 6. Learning phase, second network output in red and second desired output in blue.

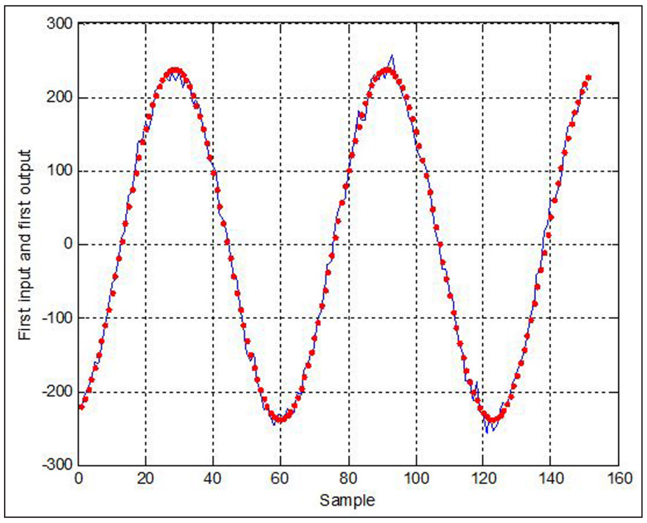

Figure 7. Generalization phase, first network output in red and first input in blue.

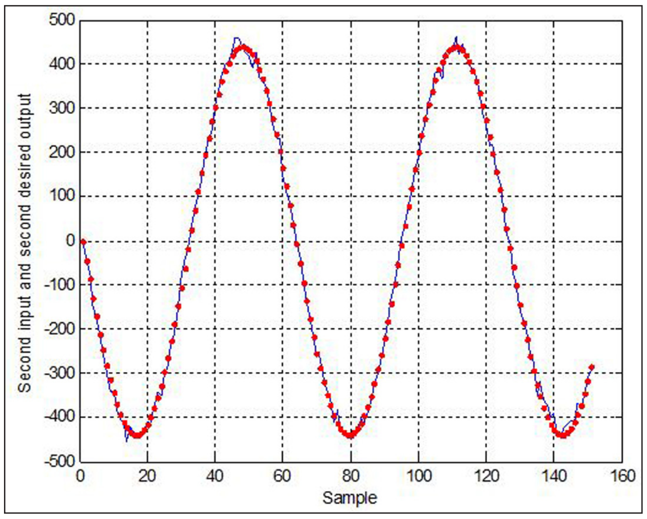

Figure 8. Generalization phase, second network output in red and second input in blue.

By applying the algorithm to eliminate weak neurons, we reach 4 neurons in the hidden layer for a learning error *j* almost unchanged.

The figure 4 to 6 represents respectively the elimination of a weak hidden layer neurons and output of the neural networks.

For the generalization phase, the last values of weight, centers and radius are used. And we use sinusoidal inputs with a larger noise that used in the learning phase.

The figure 7 and 8 represents the input and output of the neural networks for the generalization phase.

6. Conclusion

In this paper, an algorithm for optimizing the structure of neural networks of radial basis function has been developed and tested. This algorithm eliminates the weak hidden layer neurons without deteriorates the learning error. The same performance level as an ANN with a considerably large size used can be achieved with only a network of much less size and also a better quality of generalization. Validating simulation of the algorithm by an example that allowed the filtering of a noisy sinusoidal signal has also been performed. Filtering noise from a sensed fault signal is an interesting application.

As a future work authors intend to use the developed algorithm as a filter for the faults detection and isolation in a wind energy conversion system.

7. References

1. Karayiannis NB. Learning algorithms for reformulated radial basis neuralnetworks. IEEE Trans World Congress on Computational Intelligence. 1998; 2230–5.
2. Xu XZ, Ding SF, Shi ZZ. Optimizing radial basis fonction neural network based on rough sets and affinity propagation clustering algorithm. Journal of Zhejiang University. 2012; 31–138.
3. Celikoglu HB, Cigizoglu HK. Modelling public transport trips by radial basis function neural networks. Math Comput Model. 2007; 45(3-4):480–9.
4. Ebrahimzadeh A, Ghazalian R. Modulation classification using genetic algorithm and radial basis neural network based on the HOS. IEEE Trans. Digital Content, Multimedia Technology and its Applications Conference. 2010 ; 375–8.
5. Chagas SH,. MartinsJB,, de Oliveira LL. An approach to localization scheme of wireless sensor networks based on artificial neural networks and Genetic Algorithms. IEEE Trans. New Circuits and Systems Conference. 2012; p. 137–40.
6. Ram D, Srivastava L, Pandit M, Sharma J. Corrective action planning using RBF neural network. Applied Soft Computing. 2007; 7:1055–63.
7. Mulero-Martinez JI. Boundedness of the nominal coefficients in Gaussian RBF neural networks. Neurocomputing, 2007; 71 :197–220.
8. Langoni D, Weatherspoon MH, Foo SY, Martinez HA. A speed and accuracy test of backpropagation and RBF neural networks for small signal models of active devices. Eng Appl Artif Intell. 2006; 883–90.