©2016 Published in 4th International Symposium on Innovative Technologies in Engineering and Science 3-5 November 2016 (ISITES2016 Alanya/Antalya - Turkey)

A Novel Modeling Network Structure and Its Heuristic Learning Performance

[1]Gizem Ataç Kale and*[1]Cihan Karakuzu
[1]Department of Computer Engineering, Bilecik Şeyh Edebali University, Bilecik,TR

Abstract

In this research, we introduce a novel network, Hybrid Radial Basis Function Neural Network (*HyrRbfNN*), an artificial neural network (ANN) in which a hidden layer of radial basis function (RBF) is integrated. The training of weight parameters is accomplished using improved particle swarm optimization (iPSO). In total, the network has four layers. (3 hidden, 1 output) The learning performance of our network has been compared to learning performance of adaptive-network based fuzzy inference systems (ANFIS). Training measurement graphs, output and error surfaces of trained methods are displayed for the both methods. The results has shown that our method provides much better training performance.

Key words: System modeling, RBF, ANN, iPSO, Heuristic learning

1. Introduction

Artificial neural networks (ANN), influenced by human brain, are models that learn out of available data and then provide appropriate output on different data [1]. ANN are used in the modeling of nonlinear system relationships. In this manner, ANN has a wide range of applications such as data association, data interpretation, data filtering, signal processing, image and speech recognition, etc.

Radial basis function network (RBFN) is a specific type of multi layered and feed forward neural networks [2]. RBFN is a structure which is formed using a hidden layer of radial basis functions(RBF) and they are used in classification and prediction applications. Lu et al. developed a sequential learning scheme for function approximation using minimal RBF neural networks [3]. Yang et al. designed a novel self-constructing RBF neural-fuzzy system [4].

Particle swarm optimization (PSO) is a population based optimization technique which is influenced by the manners of bird swarms and designed for the solution of non-linear problems [5]. System is commenced with a population that includes random solutions and investigates optimum solution by updating the generations. PSO does not necessarily need derivation information that are different than the classical optimization techniques. PSO is simple because of its low number of parameters. Successful applications of PSO include function optimization, fuzzy system control, training of ANN. Sermpinis et al. investigated forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and particle swarm optimization [6]

Adaptive-network based fuzzy inference systems, namely ANFIS, was developed by Jang and is

*Corresponding author: Address: Faculty of Engineering, Department of Computer Engineering Bilecik Şeyh Edebali University, Bilecik TURKEY. E-mail address: cihan.karakuzu@bilecik.edu.tr.

a kind of ANN that is based on Takagi-Sugeno fuzzy inference system [7], [8]. ANFIS integrates neural networks and fuzzy logic principles in a single framework. Using a set of fuzzy IF-THEN rules, ANFIS inference system has the learning capability to approximate non-linear functions which makes it a universal estimator.

In this research, by integrating RBF, we design an ANN in which the training of weight parameters is accomplished using improved PSO (iPSO). Since the rival of our method is ANFIS, we have compared our results with the performance of ANFIS. In the following sections, it will be demonstrated that our network learning performance is much better, especially on uneven surfaces.

2. Hybrid Radial Basis Function Neural Network (*HybRbfNN*)

This section introduces the novel network structure which is announced the first-time in [9] by Cihan Karakuzu.

2.1. Structure of HybRbfNN

In this research, we define a novel network. This is *HybRbfNN* that designed in 4 layers with {2, 10, 2, 3, 1} units, by integrating RBF to ANN and the *Peaks* function [10] is what we aim to model. Regarding Figure 1, *HybRbfNN* has a network structure that has x_1 and $x_2$as inputs. In the *RBF layer*, the Gaussian function which is an RBF is used. Following the *RBF layer*, in the *summation layer* (*S layer*), summation operation is performed and $\boldsymbol{\theta}$ is a matrix that includes this layer weighting parameters. In the *NN layer*,ω is a matrix that includes that this layer's weighting parameters and the ANN activation function *tansig* is used. Finally, at the output layer, $\boldsymbol{\gamma}$ is a vector that includes this layer weighting parameters.

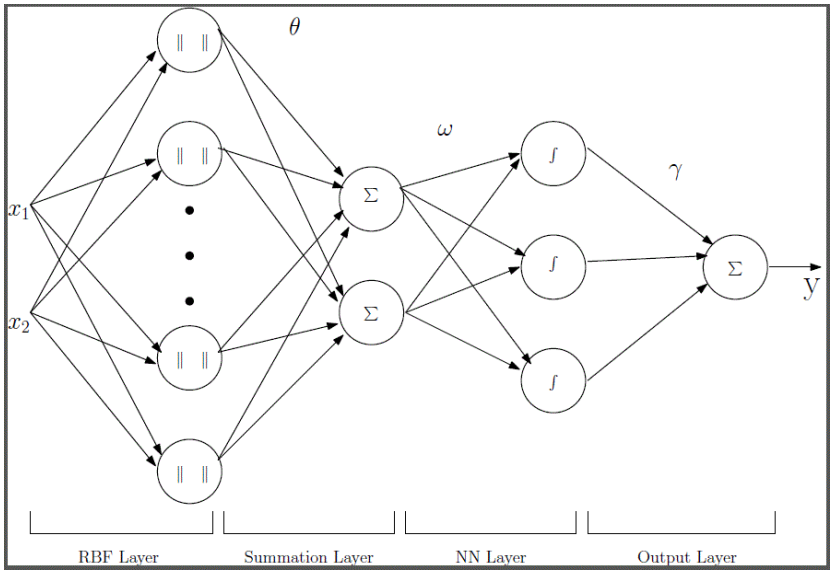

Figure 1: ***HybRbfNN***network structure.

The first hidden layer which is the *RBF layer* associates an input data to a certain cluster. Each neuron' s function of the first hidden layer is defined in Equation 1. The advantage of RBF lies in its flexible modeling ability for the sharp transitions and multi minima/maxima on input-output mapping surface. Connections of this layer are unit and they forward inputs to the neurons in this layer. For this layer, let us define *nRBF* as the number of neurons within this particular hidden layer.

$$o_{1_i} = \mu_{c_i}\left(\left|\left|\mathbf{x} - \mathbf{c_i}\right|\right|\right) = e^{-\frac{||x-\mathbf{c_i}||^2}{{\sigma_i}^2}} \quad (1)$$

Here, o_{1i} denotes the output. *RBF layer*'s parameters are $\mathbf{c}_i = [c_{i,1} \;\; c_{i,2}]$ center vectors and σ_i standard deviations for each neuron of this layer for two inputs.

The second hidden layer which is defined as *summation layer(S layer)*, has weighting connections θ, between output of the first hidden layer and this layer's neurons. The neurons of this layer weight RBF layer output values with θ as shown in Equation 2 and then sum up them. The output is o_{2i} . In this layer, *nS* stands for the number of neurons in this hidden layer.

$$o_{2_i} = o_{1_i}\theta_{ij} = \mu_{c_i}\left(\left|\left|\mathbf{x} - \mathbf{c_i}\right|\right|\right)\theta_{ij} \quad (2)$$

Neural network layer(NN layer) is the third and final hidden layer. The functionality of each neuron is displayed in Equation 3. The extra gradient parameter *A* differs this neuron from conventional neuron activation functions. There are two types of parameters in this hidden layer which are connection weights (ω) and gradients (A). Output is o_3 . Integer variable *nNN* describes the number of neurons in this hidden layer.

$$o_3 = \varphi(o_2 \times \omega) = \frac{2}{\left(1+e^{-2A\times(o_2\times\omega)}\right)} - 1 \quad (3)$$

The last layer is the *output layer* which forms the output by combining the weighted (γ) the previous layer outputs and the functionality of each neuron is described in Equation 4. o_4 denotes the output.

$$o_4 = o_3 \times \gamma \quad (4)$$

2.2 Heuristic Learning of HybRbfNN

The center vectors $\mathbf{c}_i$ point out the central point of multi dimensional Gaussian functions in the input domain. Let us denote the set of these vectors with **C** matrix. The *S layer* weighting parameters are collated in the matrix **θ**. *NN layer* weighting parameters are listed in the matrix $\boldsymbol{\omega}$ as mentioned in the previous subsection and gradient parameter vector is **A**. Finally, **γ** is the vector that includes the output layer parameters.

Using iPSO, training of the four parameters are displayed below in Equation 5. Here, a particle structure $\mathbf{p_i}$ is defined which is the *i*th row of a particle swarm matrix.

$$\mathbf{p}_i = \left[c_{1,1}c_{1,2} \dots c_{m,1}\, c_{m,2}\sigma_1 \dots \sigma_m\, \theta_1 \dots \theta_j\, \omega_1 \dots \omega_k A_1 \dots A_l \gamma_1 \dots \gamma_l\right] \quad (5)$$

In the Equation 5, *m* represents the number of RBF layer neurons (*nRBF)*, whereas *j* is an integer of the product of *nRBF* and *nS*. The integer product *nS* x *nNN* is *k* and finally *l* is the number of neurons in the NN layer.

These parameters mentioned in the previous paragraph are determined by a heuristic learning algorithm in this research using an improved PSO (iPSO) [11] algorithm as shown in Equation 6 and 7. Detailed information about the algorithm and its parameters can be seen in [11].

$$\mathrm{v_i(n+1)} = \xi\left[\mathrm{v_i(n)} + \alpha_1 \mathrm{r_1}\left(\mathrm{p_{b,i}} - \mathrm{p_i(n)}\right) + \alpha_2 \mathrm{r_2}\left(\mathrm{g_{b,i}} - \mathrm{p_i(n)}\right)\right] + [\alpha_3 \lambda(\mathrm{n})] \quad (6)$$

$$p_i(n+1) = p_i(n) + v_i(n+1) \quad (7)$$

Here, $\mathrm{v_i}$ is the velocity of the particle, i.e. the Cartesian Coordinate update value. ξ is the construction factor and in most applications equals to 0.76. α_1 and α_2 are the learning rate constants which are picked as 2.1. $\mathrm{r_1}$ and $\mathrm{r_2}$ uniformly distributed random values which has a range of [0, 1). $\mathrm{p_i}$ is the particle structure as mentioned in the previous paragraph. $\mathrm{p_b}$ is the local best which is the best position memorized by each particle and $\mathrm{g_b}$ is the global best which is the best position of every local best obtained. α_3 is the additive learning constant and has to be picked according to the Equation 8. λ is a normally distributed random number vector.

$$\propto_3 \ll \frac{1}{max\{eig(XX^T)\}} \quad (8)$$

3. System Modeling Example and Its Comparison with ANFIS

In this research, modeling efficiency of *HybRbfNN* has been shown using *Peaks* function which has multiple extremums since it is useful in three-dimensional plots. *Peaks* is obtained by translating and scaling two variable Gaussian distributions and mathematically expressed in Equation 9 [10].

$$z = peaks(x,y) = 3(1-x)^2 e^{\left(-x^2-(y+1)^2\right)} - 10\left(\frac{x}{5} - x^3 - y^5\right) e^{\left(-x^2-y^2\right)} - \frac{1}{3} e^{\left(-(x+1)^2-y^2\right)} \quad (9)$$

As a demonstration, *HybRbfNN* is used to model the function shown in Equation 9. For this purpose, parameters of *HybRbfNN* is tuned or learned by iPSO with a single swarm using the concept described in the previous paragraphs. During the learning process, each particle is evaluated by the fitness value given in Equation 10 which considers the error between desired output (z_d) and actual output of *HybRbfNN* (z_a).

$$fit = \frac{1}{N}\sum_{i=1}^{N}(z_d - z_a)^2 \quad (10)$$

Learning of *HybRbfNN* has been implemented using the following iPSO algorithm parameters: swarm size (the number of the particles) is 186 for the swarm; learning constants ξ=0.76, $\alpha_1=\alpha_2$=2.1, α_3=0.0012; the number of generation/iteration G_{max}= 300. Learning of ANFIS has been implemented using the same iPSO algorithm parameters of *HyrRbfNN* except the swarm size of ANFIS is 60.

Figure 2 shows the learning performance of the modeling. For the given learning, standard deviations of the first layer's nodes were kept fixed as unity.

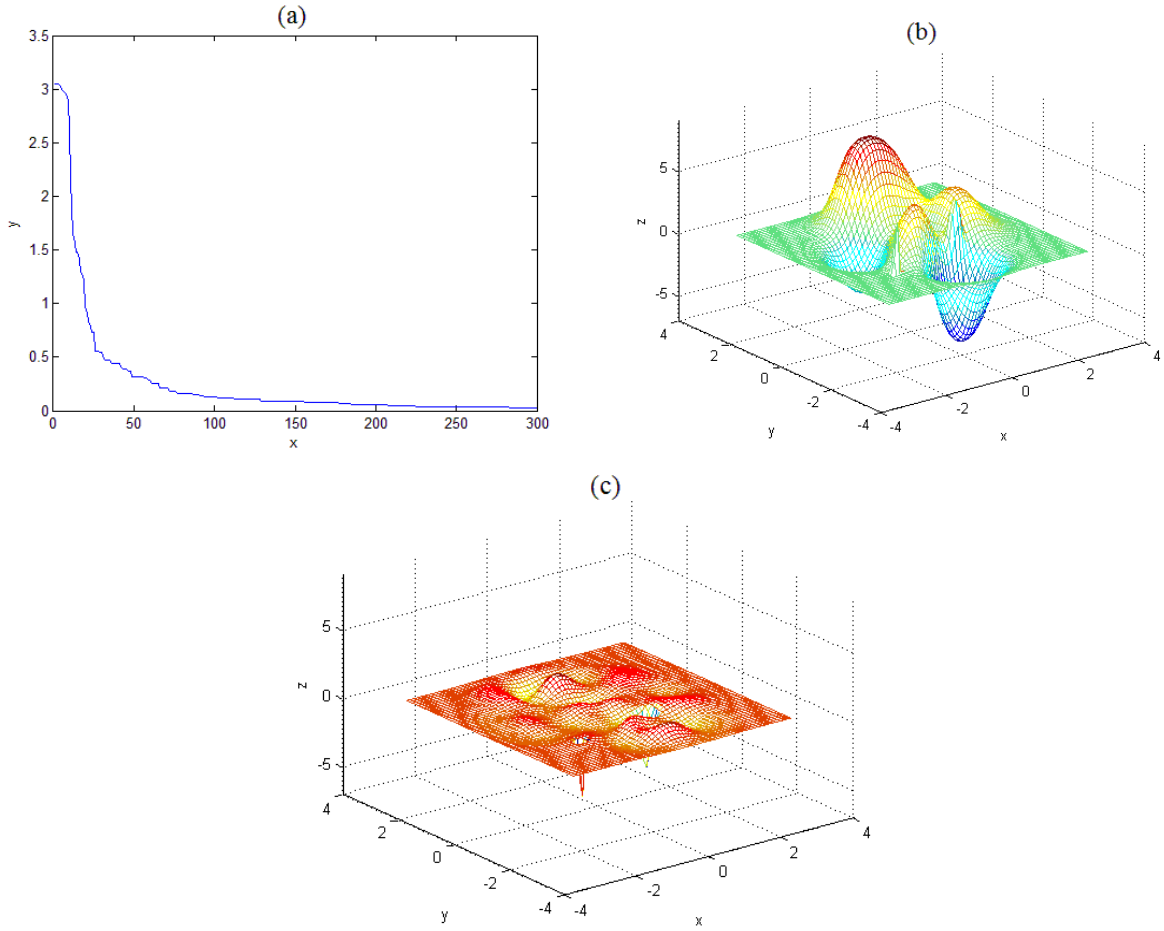

Figure 2:Learning performance of *HyrRbfNN*: Training course (a), Output surface (b) and error surface (c) of trained network

The 2-input, 1-output and 5-layer (total) ANFIS structure we use has 4 rule Sugeno type fuzzy inference method. The structure has two fuzzy sets each having two neurons and as membership function, Gaussian activation function is used. The central points and standard deviations

(antecedent parameters), rule or consequent parameters are trained via iPSO. Figure 3 displays learning performance of ANFIS.

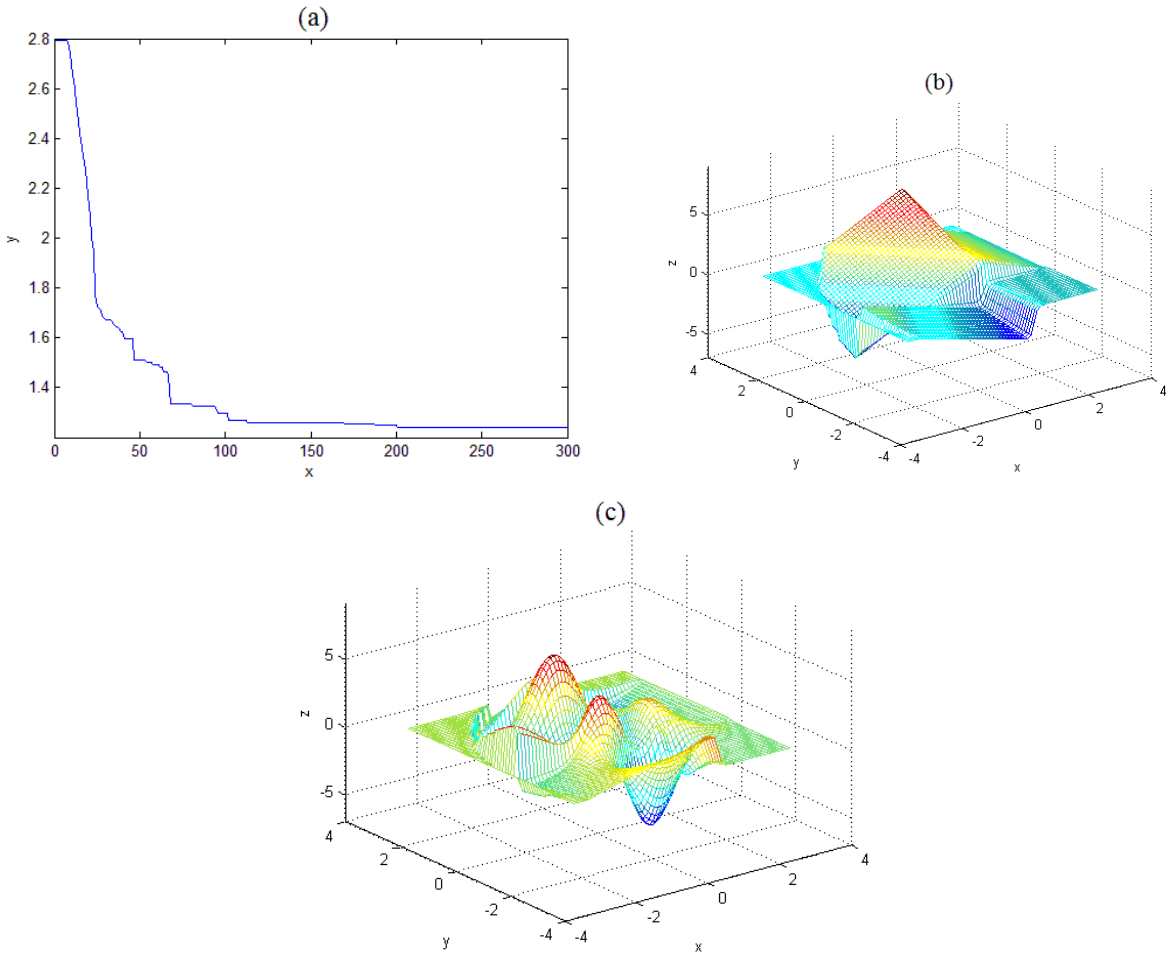

Figure 3: Learning performance of *ANFIS*: Training course (a), Output surface (b) and error surface (c) of trained network

4. Discussion

Although ANFIS is a popular network model, our network model *HyrRbfNN* clearly outperforms it as shown in Figure 2 and 3. Heuristic learning of the two networks using iPSO has been performed ten times to compare statistically with each other. Obtained statistical results are given in Table 1. In the table, initial and final fitness values of the global best particle at the first and the last generation are given in the columns named as *First* and *Last* respectively. Mean of the global best particles' fitness values throughout training course is given in the column named as *Mean*. As can be seen clearly on both the figures and the table, *HyrRbfNN* outperforms than ANFIS. For example, regarding Table 1, in the 7th run, the last training course value of ANFIS is

1.59982566. On the other hand, the last training course value of *HyrRbfNN* is 0.04305355 which is far too close to 0 in the same run. *HyrRbfNN* has more parameters than ANFIS; however, it is much more successful, especially on slopes, sharp transitions and uneven surfaces.

Table 1. Comparison of learning performance of *HyrRbfNN* and ANFIS

	HyrRbfNN			ANFIS		
Run	First	Last	Mean	First	Last	Mean
1	3.05247084	0.09704854	0.55308577	2.81004351	1.66240576	1.84384968
2	3.05246870	0.09973391	0.44641125	2.80639244	1.94190508	2.07711224
3	3.05246191	0.95657352	1.15240545	2.82877895	1.95365058	2.07041582
4	3.05245971	0.15725067	0.43714378	2.84856796	2.20795477	2.25250595
5	3.05247025	0.07178905	0.35753075	2.82092960	1.95819235	2.02286796
6	3.05246125	0.04843998	0.34222334	2.80083656	2.24005271	2.27247340
7	3.05245696	0.04305355	0.34138577	2.81939214	1.59982566	1.87428082
8	3.05245740	0.05183595	0.31814243	2.79383574	2.41294341	2.43651952
9	3.05246208	0.06584072	0.29879856	2.82108382	2.39282785	2.43137310
10	3.05246964	0.09857568	0.69321935	2.79224192	1.95536564	2.03181721

Conclusions

HyrRbfNN is a novel network structure where we have integrated RBF into ANN in which the training of weight parameters were accomplished using iPSO. The learning performance of *HyrRbfNN* has been compared to the learning of performance of ANFIS. The given results and Discussions section clearly show that *HyrRbfNN* provides better learning performance. *HyrRbfNN* can be a useful in problems including uneven surfaces and sharp transitions.

References

[1] Simon Haykin, *Neural Networks : A comprehensive Foundation*.: Prentice Hall, 1998.

[2] Martin D. Buhmann, *Radial Basis Functions: Theory and Implementations*.: Cambridge University Press, 2003.

[3] N. Sundararajan, P. Saratchandran Lu Yingwei, "A Sequential Learning Scheme for Function Approximation Using Minimal Radial Basis Function Neural Networks," *Neural Computation*, vol. 9, no. 2, pp. 461-478, Feb 1997.

[4] Tsung-Ying Sun, Chih-Li Huo, Yu-Hsiang Yu, Chan-Cheng Liu, Cheng-Han Tsai Ying-Kuei Yang, "A novel

self-constructing Radial Basis Function Neural-Fuzzy System," *Applied Soft Computing*, vol. 13, no. 5, pp. 2390-2404, May 2013.

[5] J. Eberhart, R. C. Kennedy, "Particle Swarm Optimization," in *IEEE International Conference on Neural Networks*, vol. IV, Piscataway, NJ, 1995, pp. 1942-1948.

[6] Konstantinos Theofilatos, Andreas Karathanasopoulos, Efstratios F. Georgopoulos, Christian Dunis Georgios Sermpinis, "Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and Particle Swarm Optimization ," *European Journal of Operational Research* , vol. 225 , no. 3 , pp. 528-540 , March 2013.

[7] Jyh-Shing R. Jang, "Fuzzy Modeling Using Generalized Neural Networks and Kalman Filter Algorithm," , Anaheim, CA, USA., 1991, pp. 762-767.

[8] Jyh-Shing R. Jang, "ANFIS: adaptive-network-based fuzzy," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665-685, 1993.

[9] Cihan Karakuzu, "New Fuzzy-Neural Network Model and Its Learning Using Heuristic," in *International Congress on Natural and Engineering Sciences Abstract Book*, Sarajevo, Bosnia and Herzegovina, 9-13 September, 2015, p. 10.

[10] Anonymous. Peaks Function (Matlab Style). [Online]. http://finzi.psych.upenn.edu/library/pracma/html/peaks.html

[11] Cihan Karakuzu, Fuat Karakaya Mehmet Ali Çavuşlu, "Neural identification of dynamic systems on FPGA with improved PSO learning," *Applied Soft Computing*, vol. 12, no. 9, pp. 2707-2718, Sep. 2012.