Hybrid Approach to Optimize the Centers of Radial Basis Function Neural Network Using Particle Swarm Optimization

Monir Foqaha, Mohammed Awad*
Department of Computer Science, Faculty Engineering and Information Technology, Arab American University Jenin, 240, Palestine.

* Corresponding author. Tel.: +970599559648; email: mohammed.awad@aauj.edu
Manuscript submitted February 8, 2016; accepted April 25, 2016.
doi: 10.17706/jcp.12.5.396-407

Abstract: Function approximation is an important type of supervised machine learning techniques, which aims to create a model for an unknown function to find a relationship between input and output data. The aim of the proposed approach is to develop and evaluate a function approximation models using Radial Basis Function Neural Networks (RBFN) and Particles Swarm Optimization (PSO) algorithm. We proposed Hybrid RBFN with PSO (HRBFN-PSO) approach, the proposed approach use PSO algorithm to optimize the RBFN parameters, depending to the evolutionary heuristic search process of PSO, here PSO use to optimize the best position of the RBFNN centers *c*, the weights *w* optimize using Singular Value Decomposition (SVD) algorithm and the Radii *r* optimize using K-Nearest Neighbors (Knn) algorithm, within the PSO iterative process, which means in each iterative process of PSO, the weights and Radii are updated depending the fitness (error) function. The experiments are conducted on three nonlinear benchmark mathematical functions. The results obtained on the training data clarify that HRBFN-PSO approach improved the approximation accuracy than other traditional approaches. Also, this result shows that HRBFN-PSO reduces the root mean square error and sum square error dramatically compared with other approaches.

Key words: Radial basis function neural networks, particles swarm optimization, function approximation.

1. Introduction

In the field of mathematical sciences, it is important determining unknown functions from a set of input/output data. It is common finding data with discrete values, this produce pattern of the relationship between input and output which called curve fitting [1]. The functions approximation generally is obtained by linear combination of elementary functions, which take the form of: $c_n f_n(x) + c_{n-1} f_{n-1}(x) + \ldots + c_0 f_0(x) : \ 0 \le i \le n$ where c is the constants values that we want to find and $f(x)$ is specific elementary functions [2]. Function approximation describes the behavior of complicated mathematics functions by collecting simpler functions and for reducing the computing time to calculate the value of function value [3]. Function approximation, used in the most real world applications [4], such as pattern recognition, prediction, classification and data mining [5]. To achieve good approximations of complex non-linear functions, the cluster of the input data must be grouped small patches [6]. In many preceding works, researchers focused basically on the most applicable way in function approximation

techniques, which is a supervised learning problem, which may be formulated as a regression task [7].

Neural Networks (NNs) are mathematical models inspired by biological neural networks; it is black-box models trained using input/output data. The basic component of the neural network is a simple processing unit (neuron), these neurons are interconnected by the weights attributed [8]. The principle of applying NNs for approximating unknown functions depends on the process of learning by applying examples of these functions. The weights in NNs are updated to produce the same output as in the examples (objective output). The goal behind adjusted the weights is when the NN is offered a new set of input variables; it will give a correct output and reduce the error between the objective output and the actual output to improve the performance of the NNs [9].

One of the important types of NNs is the Radial Basis Function Neural Networks (RBFN), which are characterized from other types of NNs, including better approximation, simpler network structures and faster-learning algorithms [10]. One important method to determine the centers and width of RBF is clustering techniques [11]. The centers can be determined randomly from the input data but in most cases it is better using traditional K-means Clustering Algorithm to determine a suitable position of the centers in the state space, use k-Nearest Neighbors (knn) to determine RBF radii [12]. Traditional algorithms are used to optimize the weights connections of RBFN, such Least Mean Square (LMS) [12], and Singular Value Decomposition (SVD) [13], orthogonal least squares (OLS) [14]. Evolutionary algorithms provide great flexibility in the optimization which currently allows the evaluations required to approximate complex mathematical problems. Evolutionary algorithms (EAs), which are optimization techniques that avoid various mathematical complications, manage populations of solutions, trying to identify individuals representing the best solutions for the problem [15], the most previous studies was based on Genetic Algorithms and Neural Networks [16]. A similar method of evolutionary computation techniques such as Genetic Algorithms (GAs) is Particle Swarm Optimization (PSO), which is a method for optimized several continuous nonlinear functions. PSO is inspired by the social behavior of bird flocks and school of fishes, it used in many applications, like neural network training and various areas of application including telecommunications, signal processing, data mining, control, power systems, and several other applications. PSO algorithm operates on a population (swarm), with the characteristic of no crossover and mutation calculation like genetic algorithm; which makes it is easy to implement. PSO only evolve their social behavior accordingly their movement towards the best solutions [17].

In this paper, we propose a hybrid approach that combines RBF neural network with PSO algorithm (HRBFN-PSO). We use Particle swarm optimization to optimize the centers of RBFN. Tradition algorithms like K-nn and SVD used to optimize the radii and the weights respectively. Each particle in PSO consists of centers of hidden neuron for RBFN. After running PSO algorithm number of iterations, the process selects the best values of the centers (Gbest of PSO). As a best center position for the approximation process radii and weights are optimized in all iterations of the PSO. To determine the optimal number of RBF (Neurons), we use the incremental method, which terminated when the process get the threshold value of based on the error value.

This paper is organized as follows; Section 2, presents the newer related works of the proposed field. Section 3, describes the RBFN in details. Section 4, explains the PSO algorithm in details. Section 5, describes the proposed approach HRBFN-PSO. Section 6, shows the experimental results of some benchmark mathematical functions. Lastly, conclusions are presented in Section 7.

2. Related Works

The advantages of evolutionary algorithms and intelligence swarm in function approximation are that these algorithms behave robustly and efficiently in set broad domains. PSO algorithm is widely used in

resolution problems of optimization, and it has become popular depends to its speed of convergence and the simplicity of its implementation. There are several studies that combined RBFN and PSO. The authors in [18] combined Incremental RBFN with Particles Swarm Optimization (IRBF-PSO) to improve the accuracy of classification in intrusion detection system, PSO was used to find optimal values for weight and bias. In [19], the authors have proposed a PSO-RBF to control the design of RBF Networks and evaluate parameter of RBF to solve pattern classification problems, in this model PSO used to finds the size of network, in addition to optimize the center and the width for each basis function. The authors in [20] presented an intelligent fault diagnostic approach for a steer-by-wire (SBW) system. They used clustering algorithm to find the number of the centers in RBF, and then optimize the centers and widths by using PSO algorithm. PSO-RBFNN proposed in [21] concentrated in finding the best center values of RBF using PSO in the induction motor parameter. The experimental results of PSO-RBFNN show this method can enhance the induction motor performance accuracy. In [22] present new approach called PSO-ELM algorithm that combines PSO and extreme learning machine (ELM) to train RBFN. Here PSO is used to find optimal parameters for center and width of neurons, while ELM uses to find the values connection weights. MuPSO-RBFNN presented in [23] is a learning algorithm that combines a RBFNN with PSO with mutation operation to train RBF parameters (center, width and weight). As for in [24] they presented a time variant multi-objective PSO of RBF called TVMOPSO for diagnosing the medical diseases. TVMOPSO is used to optimize centers and weights of RBF network. In [25] they proposed the transformer fault diagnosis approach based on RBF neural network improved by PSO algorithm. PSO algorithm is used to optimize (center, width and weight) values of RBF neural network. Hybrid PSO-RBF proposed in [26] was used to evaluate the levels of underground water quality in the ten monitoring points of the black dragon hole, the value of weight for output layer determined by PSO. In [27] the presented method growth the number of RBF, the basic idea in this model increases the model complexity by add one neuron at each iteration, it based on the PSO algorithm to optimize parameters of the newly RBF neuron only, while save the parameters of all older RBF neuron that optimized earlier and gradually decreased root mean square error RMSE at each iteration.

3. Radial Basis Function Neural Networks (RBFN)

Radial Basis Function Neural Networks are type of neural networks whose activation functions in the hidden layer are radially symmetrical, which means its output depends on the distance between a vector that stores the input data and a vector of weights, which is called the center. RBFN has been used in many applications, such as function approximation, system control, speech recognition, time-series prediction, and classification. [28]. The RBFN has three layers feed-forward connection: the input layer, the hidden layer and output layer. The input data flow from input layer to send the information to the hidden layer. The hidden layer neurons are activated depending on the distance between each input pattern with the centroid stores each hidden neuron, which determine the structure behavior of network; different types of activation functions may be used in hidden layer, but the most type commonly used in most application is the Gaussian Function. The output layer calculates the linear sum of values of the hidden neuron and outputs it [29]. Fig. 1 shows the architecture of RBFN that including three layers.

Two categories of parameters need to be determined in RBFN [30]: the first category is the center and radii; and the second one is the connection weights between the hidden layer and output layer. The Gaussian activation function in the hidden layer (Φ) is calculated as follows:

$$\Phi = \exp\left[\frac{-(x-cj)^2}{2\sigma j^2}\right] \tag{1}$$

where $x = (x_1, x_2, x_3 \ldots, x_n)$ is the input data, c_j is the center of j-th hidden neuron, and σ_j is the width of j-th hidden neuron. The output of RBFN is calculated as in the following expression:

$$Y = \sum_{k=1}^{m} Wjk * \Phi j(x) \tag{2}$$

where k=1, 2, 3 . . . , m is the number of nodes in hidden layer. W_{jk} are the connection weights values between the j-th hidden layer nodes and the k-th output nodes.

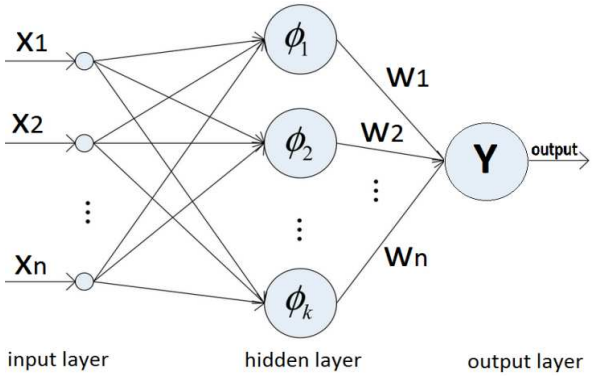

Fig. 1. Architecture of RBFNN.

In RBFN to reach the best accuracy for results is used fitness function measured, which takes the error between the output (target) of the RBF and the real output. There are many fitness functions can be used to calculate error such as mean square error (MSE), root mean square error (RMSE), sum square error (SSE), and Normalized/Root Mean Squared Error (N/RMSE). In this paper, we use two fitness functions namely root mean square error (RMSE) as shown in (3) and we use sum square error (SSE) as shown in (4), to compare our results with results of previous studies.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(T-Y)^2} \tag{3}$$

$$\mathrm{SSE} = \sum_{i=1}^{n}(T-Y)^2 \tag{4}$$

where n is the number of input data, T is the target output, and Y is the real output.

One important parameter of RBFN is the determination of the suitable number of neurons in hidden layer, which affects the network complexity and the generalization of the RBFN. If the number of neurons in RBFN is too small, the accuracy of the output will decrease. On the other hand, if the number of neurons is too large, this cause over-fitting for the input data [31]. The drawback of RBFN is that it treats all the input parameters with same level of significance [32]. In this paper, we use the incremental method to determine the suitable number of neurons in the hidden layer. The process starts with one neuron and increase the number of neuron by one in each iteration, the process stop increasing when the training process get the threshold value of the error.

4. Particle Swarm Optimization (PSO)

Particle Swarm Optimization mimics the behaviors of bird flocking. In bird flocking, a group of birds are randomly searching food in specific area. All the birds do not know where the food is found. But they know how far the food in each iteration. PSO algorithm is a fast optimization method due to need to adjust few

parameters, which has fast convergence speed, high robustness, and strong global search capability, does not require gradient information and is easy to implement [33].

The population in PSO is called swarm that is initialized with a group of random values called particles (solutions) and then searches for optimal position of particles by updating it for all iterations. In all iterations, each particle is updated by following two best fitness values are evaluated by using proper fitness function according to problem. The first value is the best position it has achieved so far for each particle; this value is called personal best (pbest). Another value is the best position for the entire swarm obtained so far by any particle in the swarm; this best value is called a global best (gbest). All particles have velocity which determine direction of the particles and move it to the new position. When calculating the velocities, it is possible for the magnitude velocities are become very large. So it is best limited a maximum velocity (Vmax) that can be set by the user that can affect the performance of PSO. Therefore the velocity values become limited between [-Vmax, Vmax].

The basic algorithm of PSO is as following steps:

- **Initialize** each particle *i* of the swarm, with random values for position *(Xi)* and velocity *(Vi)* in the search space according to the dimensionality of problem.
- **Evaluate fitness** value of particle by using fitness function.
- **Compare** the value obtained from the fitness function from particle *i*, with the value of Pbest.
- **If** the value of the fitness function is better than the Pbest value, then update the particle position to takes the place of Pbest.
- **If** the value of Pbest form any particle is better than gbest, then update gbest = Pbest.
- **Modify** the position *Xi* and velocity *Vi* of the particles using (5) and (6), respectively.
- **If** the maximum number of iterations or the ending criteria is not achieved so far, then return to step 2.

$$\mathrm{Vid}\left(t+1\right)=\omega*\mathrm{Vid}\left(t\right)+\mathrm{c1r1}\left(\mathrm{Pid}\left(t\right)-\mathrm{Xid}\left(t\right)\right)+\mathrm{c2r2}(\mathrm{Pgd}\left(t\right)-\mathrm{Xid}(t)) \tag{5}$$

$$\mathrm{Xid}\left(t+1\right)=\mathrm{Xid}\left(t\right)+\mathrm{Vid}\left(t+1\right) \tag{6}$$

where i = 1, 2, …, M; d = 1, 2, …, n, t+1 is the current iteration number, t is the previous iteration number, ω is the inertia weight, c_1 and c_2 are the acceleration constant which are usually between [0, 2], P_i = (P_{i1}, P_{i2},…, P_{in}) is the best previous position of particle i known as the personal best position (pbest), P_g = (P_{g1}, P_{g2}, …, P_{gn}) is the position of the best particle among all the particles in the swarm known as the global best position (gbest), and r_1 and r_2 are random numbers distributed uniformly in (0, 1).

5. Hybrid RBFN with PSO (HRBFN-PSO)

PSO has been used to improve RBF Network in several sides like network architecture, learning algorithm, and network connections. Every single solution of PSO called a particle. Using fitness function (MSE in this paper) to evaluate the particles for optimal solution. In this paper, we present approach to evolve the optimization accuracy, this approach is a combined RBF with PSO to reduce the number of neurons and error value between target and real output in Radial Base Function Neural Network, as well as select the best values of RBF centers by using PSO. The remaining parameters of RBF, we use tradition algorithms namely Knn and SVD to optimize radii and weight respectively. HRBFN-PSO approach is explained in the following pseudo code:

Start with one RBF.
Initialize RBFN parameters.

- *Initialize the centers c randomly*

- *Use **K-nn** to initialize Radii r*
- *Use **SVD** to initialize Weights w.*

Start optimizing centers of RBFs using PSO.
Initialize particles position and velocity randomly
While not reach the maximum numbers of iteration ***do***
For each particle do
Calculate fitness value (MSE between Real output of RBFN and Target)
If fitness value is better than best fitness value ***pbest*** in particle then
Set current position as ***pbest***
End
Select *gbest* of the particle which the best fitness value among all particles in current iteration
Calculate particle velocity based on (5).
Update particle position (centers) based on (6).
End
End
Take the ***gbest*** of particle as centers ***c*** of RBF
Complete training RBFN using K-nn and SVD.
Calculate the real output of RBFN
Repeat

It should be noted that inertia weight ω was first introduced by Shi and Eberhart [34] in order to control the search speed and make particles converge to local minima quickly. Inertia weight ω often have restricted between two numbers during a run. Here in this work, we calculate the inertia weight by using (7).

$$\omega = \omega\max - \frac{(\omega\max - \omega\min)*t}{T\max} \tag{7}$$

where ω is the inertia weight, ωmax is the maximum of ω (here ωmax=0.9), ωmin is the minimum of ω (here ωmin=0.4), t is the current iteration number, and Tmax is the maximum iteration number.

In this paper, the training RBFN is stopped if either reaches the maximum number of iterations or got the threshold value of error.

K-Nearest Neighbors is used to determine the width r of each RBF. Knn is a simple algorithm used for classification and regression [35]. Knn stores all available cases and classifies new cases based on a similarity measure (e.g. Euclidean distance functions) as shown in (8). Knn is types of lazy learning algorithm, where the function is only approximated locally and all computation is continue until classification [36] but Knn algorithm has been successful in many numbers of classification and regression problems; such as handwritten digits and satellite image scenes [37]. The number k is used to decide how many neighbors influence the classification for new value.

$$D(x,y)^n = sqrt\left[\sum_{i=1}^{k} (xi - yi)^2\right] \tag{8}$$

To optimize the weights of each RBF, we use Singular Value Decomposition, SVD is a powerful and useful matrix decomposition has been used in many fields such as data analysis, reducing dimension transformations of images, data compression, signal processing, and pattern analysis [13]. If $A \in R^{mxn}$, there exist orthogonal matrices $S \in R^{mxm}$ and $H \in R^{nxn}$ such that:

$$S^T A H = diag(\sigma_1,...,\sigma_p) \quad (9)$$

where p is the minimum of (m, n), σ are the singular values of A. The use of SVD technique to calculate the optimal weights w of the RBFN depends on using matrix notation described in the following reducing expression:

$$\vec{Y} = \vec{w}\ \Phi \quad (10)$$

where Y is the real output of RBFN, w are the weights vector, and Φ is the Gaussian activation function matrix.

Using the next following expression:

$$A = H\ diag(\frac{1}{k})\ S^T \quad (11)$$

where $k = diag(\sigma_1,...,\sigma_p)$, by replacing Φ in (10), using (11); the weights vector (12):

$$\vec{w} = \left[H\ diag(\frac{1}{k})\ S^T \right] \vec{y} \quad (12)$$

SVD can solve any equation system. In the proposed case SVD effect in reducing the of the output error, it can also be used to remove any RBF when its associated singular value had a small value or if the approximation error can't affect the result.

6. Experimental Result

To compare of performance in function approximation of our approach, we conduct different experiments of this work including run the proposed approach (HRBFN-PSO) on the training data and compared with other previous works. For evaluating HRBFN-PSO approach we used three nonlinear functions as it shown in Table 1. All functions described in Table 1 reflect varying degrees of complexity.

Table 1. The Benchmark Nonlinear Function Information

No	Nonlinear Functions	# Data Point	Range
1	$y = \frac{\sin(x)}{x}$	50	x ϵ [-10,10]
2	$y = -sin\left(\frac{\pi x}{10}\right) + cos\left(\frac{2\pi x}{5}\right)$	50	x ϵ [-10,10]
3	$y = 1.1\left(1 - x + 2x^2\right)\exp\left(-\frac{x^2}{2}\right)$	100	x ϵ [-4,4]

Table 2. Parameters for PSO Used in HRBFN-PSO

Parameter	Value
Number of Particle	20
Iteration	500
c1	0.5
c2	2
Vmax	1

For each of the benchmark functions, HRBFN-PSO optimizes the parameters of the RBFN; these parameters optimized using PSO and traditional algorithms.

The parameters of the PSO algorithm that are used in this paper were set as: inertia weight ω in calculated based on the (7), rest of parameters are shown in Table 2.

The particle size is an important factor in HRBFN-PSO, when the algorithm use small particle numbers it produce poor performance. But, large particle number produces a very residual improvement compared to the improvement that occurs when using Media particle size, but increase the computational cost of the algorithms. From the results obtained you can see that HRBFN-PSO approach has a good performance in the optimization of nonlinear functions and ability to reach the optimum value. The number of iterations in HRBFN-PSO helps to find the optimum value of the functions, by increasing the amount of iterations the computational cost is also increased, which should be taken into consideration when using HRBFN-PSO approach. As shown the Table 2, the proposed approach used suitable number of particles and iterations.

The proposed approach HRBFN-PSO is experimented using MATLAB 2012 under Windows 7 with i5-3210M CPU 2.5GHz, 4GB RAM memory. For each of the experiments, the proposed approach executes 3 times with each of the presented three experimental functions.

6.1. First Benchmark Example

In this section, we present the results of the training HRBFN-PSO approach. The experiments are number of conducted by using three nonlinear function examples as shown in Table 1. In this function, we conduct our experiments on 50 training data point as shown in Table 1. Table 3 shows the RMSE and number of hidden neuron obtained for our approach and [38] approach. Fig. 2 show the curve results between target data and real output data in HRBFN-PSO approach. This first experiment compares the results obtained by applying the [38] to the function, as well, by applying the HRBFN-PSO to them. The ability to achieve optimal value by using the proposed approach is increased, the approximation error decreases with less number of RBF.

Table 3. Results for Nonlinear Function # 1

Approach	# Hidden Neuron (RBF)	RMSE
Sun [38]	28	0.0035
Proposed Approach HRBFN-PSO	16	0.003830
	17	0.002831
	18	0.002172

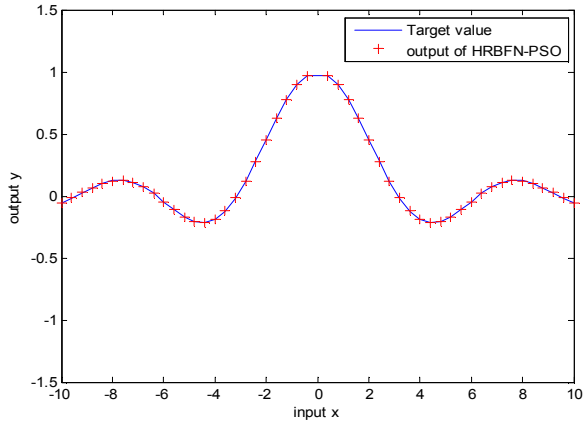

Fig. 2. Target and real output for 18 neurons.

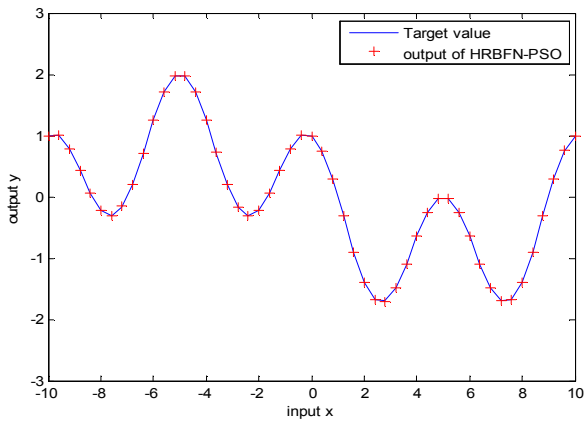

Fig. 3. Target and real output for 26 neurons.

From Table 3 and Fig. 2, we can see that the behavior of HRBFN-PSO in the approximation of the first nonlinear function. It is clear that the proposed approach converges to the optimum value with less number of RBF (hidden layer neurons) with better approximation error; take in account that the number of particles

is Median. This is because each particle has more opportunities to adjust their velocity and position of the RBFN centers. As we can see, the proposed approach performs very well comparing with the approach in [38]. It is improving the approximation performance.

Table 4. Results for Nonlinear Function # 2

Approach	# Hidden Neuron (RBF)	RMSE
Sun [38]	29	0.0079
Proposed Approach HRBFN-PSO	24	0.007703
	25	0.004136
	26	0.003295

6.2. Second Benchmark Example

In this function, we conduct our experiments on 50 training data point as shown in Table 1. Table 4 show the RMSE and number of hidden neuron obtained for our approach and [38] approach. Fig. 3 show the curve results between target data and real output data in HRBFN-PSO approach.

This experimental nonlinear function has been selected to demonstrate the ability of HRBFN-PSO in the approximation. This function has a very variable output.

6.3. Third Benchmark Example

In this function, we conduct our experiments on 100 training data point as shown in Table 1. Table 5 shows the SSE and number of hidden neuron obtained for our approach and [23] approach. Fig. 4 show the curve results between target data and real output data in HRBFN-PSO approach.

Table 5. Results for Nonlinear Function # 3

Approach	# Hidden Neuron (RBF)	SSE
Liu [23]	10	0.8756
Proposed Approach HRBFN-PSO	6	0.118994
	7	0.084387
	8	0.007835

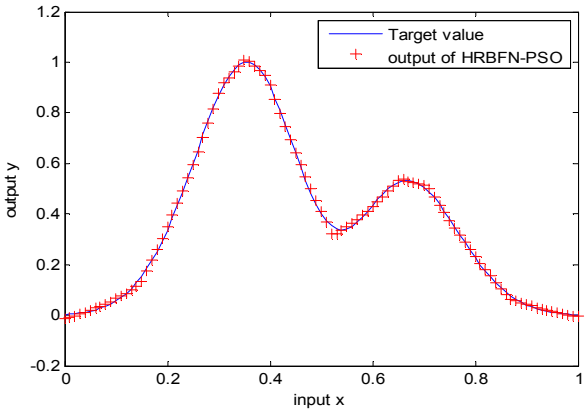

Fig. 4. Target and real output for 8 neurons.

From Tables 3, 4 and 5 explain the error functions (RMSE and SSE) of HRBFN-PSO approach are least among other approaches. In Fig. 2, 3, and 4, results are presented obtained using the proposed approach HRBFN-PSO to approximate the nonlinear benchmark functions with the aim to minimize squared error between the real output and the target of each one. The HRBFN-PSO which is optimized by PSO algorithm is

improved approximation accuracy than other approaches.

7. Conclusion

Radial Basis Function Neural Networks (RBFN) is one of the most important types of artificial neural networks (ANN); it is characterized by the following features: better approximation, simpler network structures, and faster learning algorithms. In this paper, we proposed a hybrid approach (HRBFN-PSO) that combining RBFN and PSO used for function approximation problem. PSO has been used to improve RBFN in several sides like network architecture, learning algorithm and network connections. In this paper, we use PSO to find optimal values for the centers of hidden neurons in RBFN; it is also used traditional Knn algorithm to optimize the width and SVD technique to optimize the weight. The proposed approach has been evaluated using three nonlinear mathematical functions and tests it on the specific training data. The results obtained are comparable with other approaches shows HRBFN-PSO approach improve approximation accuracy and reduce the root mean square error and sum square error compared with other approaches. The results of the simulations show that HRBFN-PSO is an effective method that is a reliable alternative for approximation nonlinear mathematical functions. The quality of the results improves the convergence.

References

[1] Yang, S., Ting, T. O., Man, K. L., & Guan, S. U. (2013). Investigation of neural networks for function approximation. *Procedia Computer Science, 17,* 586-594.

[2] Mordohai, P., & Medioni, G. (2010). Dimensionality estimation manifold, learning and function approximation using tensor voting. *Journal of Machine Learning Research, 11,* 411-450.

[3] Busoniu, L., Babuska, R., Schutter, B. D., & Ernst, D. (2010). *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press.

[4] Yang, S., Ting, T. O., Man, K. L., & Guan, S. U. (2013). Investigation of neural networks for function approximation. *Procedia Computer Science, 17*, 586-594.

[5] Vuković, N., & Miljković, Z. (2013). A growing and pruning sequential learning algorithm of hyper basis function neural network for function approximation. *Neural Networks, 46*, 210-226.

[6] Park, J., & Kim, K. Y. (2016). Instance variant nearest neighbor using particle swarm optimization for function approximation. *Applied Soft Computing, 40*, 331-341.

[7] Baudat, G., & Anouar, F. (2001). Kernel-based Methods and Function Approximation. *IJCNN, 2*, 1244-1249.

[8] Yao, X. (1999). Evolving Artificial Neural Networks . *Proceedings of the IEEE: Vol. 87* (pp. 1423-1447).

[9] Agatonovic-Kustrin, S., & Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis, 22,* 717–727.

[10] Lee, C. C., Chung, P. C., Tsai, J. R., & Chang, C. I. (1999). Robust radial basis function neural networks. *Systems, Man, and Cybernetics, Part B: Cybernetics, 29,* 674-685.

[11] Pomares, H., Rojas, I., Awad, M., & Valenzuela, O. (2012). An enhanced clustering function approximation technique for a radial basis function neural network. *Mathematical and Computer Modelling, 55(3-4),* 286 – 302.

[12] Widrow, B., & Lehr, M. A. (1990). 30 years of adaptive neural networks: perceptron madaline and back-propagation. *Proceedings of the IEEE: Vol. 78* (pp. 1415-1442).

[13] Golub, G. H., & Van Loan, C. F. (2013). *Matrix Computations*. Hopkins University Press.

[14] Chen, S., Cowan, C. F. N., & Grant, P. M. (1991). Orthogonal least squares learning algorithm for radial basis function networks. *Neural Networks, 2,* 302-309.

[15] Qasem, S. N., Shamsuddin, S. M., & Zain, A. M. (2012). Multi-objective hybrid evolutionary algorithms for radial basis function neural network design. *Knowledge-Based Systems, 27,* 475-497.

[16] Schaffer, J. D., Whitley, D., & Eshelman, L. (1992). Combinations of genetic algorithms and neural networks: a survey of the state of the art. *Combinations of Genetic Algorithms and Neural Networks,* 1-37.

[17] Elbeltagi, E., Hegazy, T., & Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics, 19,* 43-53.

[18] Prasad, M. V. S., & Gottipati, R. A novel incremental instruct dynamic intrusion detection system using PSO-RBF. *International Journal of Future Computer and Communication, 4,* 280-285.

[19] Qin, Z., Chen, J., Liu, Y., & Lu, J. (2005). Evolving RBF neural networks for pattern classification. *Computational Intelligence and Security, 3801,* 957-964.

[20] Wu, F., Kong F., & Yao, J. (2012). Intelligent fault diagnosis of steer-by-wire automobile. *Journal of Computers*, *7*, 1204-1211.

[21] Rashag, H. F., Koh, S. P., Tiong, S. K., Chong, K. H., & Abdalla, A. N. (2011). Investigation of induction motor parameter identification using particle swarm optimization-based RBF neural network (PSO-RBFNN). *International Journal of the Physical Sciences, 6(19)*, 4564-4570.

[22] Malali, P., & Kotinis, M. (2015). PSO-based training, pruning, and ensembling of extreme learning machine RBF networks. *International Journal of Computational Engineering Research, 5(6).*

[23] Liu, X. (2010). Radial basis function neural network based on pso with mutation operation to solve function approximation problem. *Advances in Swarm Intelligence, 6146,* 92-99.

[24] Qasem, S. N., & Shamsuddin, S. M. (2011). Radial basis function network based on time variant multi-objective particle swarm optimization for medical diseases diagnosis. *Applied Soft Computing, 11(1),* 1427-1438.

[25] Cui, L., Wang, C., & Yang, B. (2012). Application of RBF neural network improved by PSO algorithm in fault diagnosis. *Journal of Theoretical and Applied Information Technology, 46,* 268-273.

[26] Bi, Z., Jiayang, W., & Sulan, Z. (2012). A new PSO-RBF model for groundwater quality assessment. *Advanced Materials Research,* 463-464, 922-925.

[27] Vachkov, G., Stoyanov, V., & Christova, N. (2015). Growing RBF network models for solving nonlinear approximation and classification problems. *Proceedings of 29th European Conference on Modelling and Simulation: At Albena, &Bulgaria.*

[28] Niu, D., Lu, Y., Xu, X., & Li, B. (2014). Short-term power load point prediction based on the sharp degree and chaotic RBF neural network. *Mathematical Problems in Engineering*.

[29] Sangwan, O. P., Bhatia, P. K., & Singh, Y. (2011). Radial basis function neural network based approach to test oracle. *ACM SIGSOFT Software Engineering Notes, 36(5),* 1-5.

[30] Han, H. G., Chen, Q. L., & Qiao, J. F. (2011). An efficient self-organizing RBF neural network for water quality prediction. *Neural Networks, 24(7),* 717-725.

[31] Mashor, M. Y. (1999). Some Properties of RBF Network with Applications to System Identification. *International Journal of Computer and Engineering Management, 7*.

[32] Ou, Y. Y., Gromiha, M. M., Chen, S. A., & Suwa, M. (2008). TMBETADISC-RBF: Discrimination of β-barrel membrane proteins using RBF networks and PSSM profiles. *Computational Biology and Chemistry, 32(3),* 227–231.

[33] Modares, H., Alfi, A., & Sistani, M. N. (2010). Parameter estimation of bilinear systems based on an adaptive particle swarm optimization. *Engineering Applications of Artificial Intelligence, 23(7),* 1105-1111.

[34] Shi, Y., & Eberhart, R. (1998). A Modified Particle Swarm Optimizer. *Evolutionary Computation Proceedings of IEEE World Congress on Computational Intelligence* (pp. 1945–1950).

[35] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The*

American Statistician, 46(3), 175–185
[36] Song, Y., Huang J., Zhou, D., Zha, H., & Giles, C. L. (2007). IKNN: Informative K-nearest neighbor pattern classification. *Knowledge Discovery in Databases, 4702,* 248–264.
[37] Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research, 10,* 207-244.
[38] Sun, T. U., Liu, C. C., Lin, C. L., Hsieh, S. T., & Huang, C. S. (2009). *A Radial Basis Function Neural Network with Adaptive Structure via Particle Swarm Optimization*. Taiwan: National Dong Hwa University.

Monir Foqaha is a master student in computer science program in the Faculty of Engineering and Information Technology at Arab American University of Jenin, Palestine.

He received his bachelor's degree in computer science from An-Najah National University of Nablus, Palestine in 2007. His research interests include neural networks and optimization algorithms.

Mohammed Awad received the Ph.D. degree in computer engineering from the Granada University Spain. From 2005 to 2006, he was a contract researcher at Granada University. Since Feb. 2006, he has been an assistant professor in Computer Engineering Department, College of Engineering and Information Technology at Arab American University. At 2010 he has been an associate professor in computer engineering. He worked for more than 10 years at the Arab American University in academic Position, in parallel with various Academic administrative positions. (Departments chairman and dean assistant, and dean of scientific research/Editor-In-Chief, Journal of AAUJ). Through the research and educational experience, he has developed strong research record. His research interests include: artificial intelligence, neural networks, function approximation of complex systems, clustering algorithms, input variable selection, fuzzy expert systems ,genetic algorithms and particle swarm optimization.