Research Journal of Applied Sciences 8 (7): 383-387, 2013
ISSN: 1815-932X

Prey-Predator Algorithm as a New Optimization Technique Using in Radial Basis Function Neural Networks

Nawaf Hamadneh Surafel Luleseged Tilahun, Saratha Sathasivam and Ong Hong Choon
School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia

Abstract: Prey-Predator algorithm is a new metaheuristic algorithm developed for optimization problems. It is inspired by the interaction between a predator and preys of animals in the ecosystem. In this study, researchers have used the Prey-Predator algorithm to train the radial basis function neural networks. The most important in the training is finding the parameters including the centers, the widths and the output weights. Researchers have compared the performance of the new algorithm with the genetic algorithm on a logic programming data, iris flowers data set and new thyroid data set. The sum square error function was used to evaluate the performance of the algorithms. From the computational results, researchers found that Prey-Predator algorithm is better in improving the performance of radial basis function neural.

Key words: Radial basis function neural network, Prey-Predator algorithm, Genetic algorithm, Metaheuristic, algorithm, optimization

INTRODUCTION

Radial Basis Function Neural Network (RBFNN) is a three layer feed forward neural network which is addressed by viewing a network design as an approximation problem in a high dimensional space (Dhubkarya *et al*., 2010; Rojas, 1996; Taghi *et al*., 2004; Venkatesan and Anitha, 2006; Xiaobin, 2009). The importance of the network comes from that it has simple network structure, faster learning algorithm, high approximation capabilities and has been widely applied in many science and engineering fields (Hamadneh *et al*., 2012a, b; Rojas, 1996). For training the networks, deferent metaheuristic optimization algorithms have been used such as genetic algorithm and particle swarm optimization algorithms (Eberhart and Kennedy, 1995; Kennedy and Eberhart, 1995; Ustun, 2007). Genetic algorithm is suitable to the problem of training RBFNNs because it is good at exploring the entire space in an intelligent way to find values close to the global optimum (Ustun, 2007). In general, the problem of training networks is actually an unconstrained nonlinear minimization problem (Zhang *et al*., 1998).

For a long time scientists have been trying to develop methods for the problem of training neural networks (Parker and Lee, 2003; Zhou and Liu, 2006). Several metaheuristic algorithms had been developed which offer better results such as an error as small as possible. PPA is developed for optimization problems. The algorithm is a more general algorithm where some of well known algorithms including simulated annealing, particle swarm optimization, firefly algorithm and evolutionary strategy become its special case. It is inspired by how the predators run after their preys and how the preys try to survive in the ecosystem. PPA is a trajectory based metaheuristic optimization algorithm which used the concept of distance to update the location of the solutions using a term called velocity. Whereas GA is not a trajectory based rather it switches chromosomes which are the building block of the solutions to create a new solution. The aim of this study is developing the prey predator algorithm to use in training the neural networks.

RADIAL BASIS FUNCTION NEURAL NETWORKS

The advantages and the importance of the RBFNNs comes from that they have a simple topological structure and their ability to reveal how learning proceeds in an explicit manner. RBFNN has typically three layers (Moody and Darken, 1989): namely, an input layer, a hidden layer (Sathasivam *et al*., 2011) and a linear output layer as shown in Fig. 1. The hidden layer neurons receive the input information followed by certain decomposition, extraction and transformation steps to generate the output information. RBFNNs with good specification should have less hidden units and high prediction accuracy. So, the effective number of the hidden neurons is a relatively

Corresponding Author: Nawaf Hamadneh, School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia

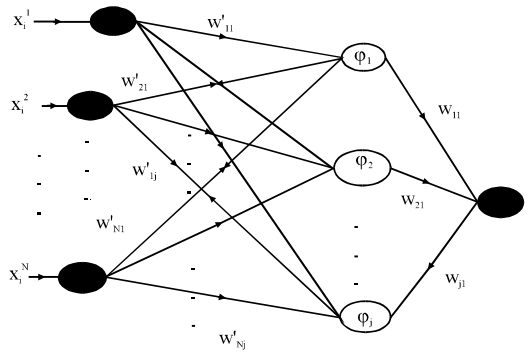

Fig. 1: Structure of a radial basis function neural network with one output neuron

small number. The interpolating function has to pass through all the training data points. Accordingly, the RBFNN technique has the following form:

$$F_L(x_i) = \sum_{m=1}^{j} w_{mL}\varphi_m(x_i) \quad (1)$$

Where:

$F_L(x_I)$ = The actual output value of the output neuron L which corresponds to the input value $x_I \in \Re^N$

w_{mL} = The output weight between the hidden neuron m and the output neuron L

φ_m = The activation function in hidden neuron m

j = The number of hidden neurons

The following equation gives the activation function (Gaussian function) which is used in RBFNN (Sathasivam *et al*., 2011):

$$\varphi_m(x_i) = e^{\frac{\left(\sum_{h=1}^{N} w'_{hm} \times x_i^h - c_m\right)^2}{2\sigma_m^2}} \quad (2)$$

Where:

φ_m = The radial basis function in hidden neuron m

c_m and σ^2_m = The center and the width of the hidden neuron m, respectively

$x_i = (x^1_i, x^2_i,..., x^N_i)$ = A value entered in the input layer

$x^h_i \epsilon x_i$ = An element in x_i entered in the input neuron h

w'_{hm} = The constant input weight between the input neuron and the hidden neuron m

N = The number of the input neurons (i.e., Dimension of x_i)

The training set (Lowe, 1989; Moody and Darken, 1989; Sathasivam *et al*., 2011) in RBFNN is represents associations to a given finite set of input-output data.

The actual output value is represented by the map F_L that is encoded by the binary values {0, 1}. Researchers have replaced F and T by 0 and 1, respectively to emphasize false and true. The difference between the actual values and the output target values can be obtained by using a differentiable function such as the sum of squared error SSE function (Ling *et al*., 2011; Liu, 2010; Schwenker *et al*., 2001):

$$SSE = \sum_{i=1}^{R} (O_i - Y_i)^2 \quad (4)$$

Where:

Y_i = The actual output value that represents the output target value O_i

R = The number of input data

PREY-PREDATOR ALGORITHM

Prey-Predator Algorithm (PPA) is a new metaheuristic algorithm introduced for optimization problems (Tilahun, 2013). It is inspired by the interaction between a predator and preys of animals in the ecosystem. In the algorithm, randomly generated solutions will be assigned as a predator and preys depending on their performance on the objective function. A solution with least performance will be assigned as a predator and the others preys. A prey with better performance in the objective function will be called best prey. After the assignment of predator and preys, the preys will run away from the predator and follow preys with better performance. The predator does the exploration by running randomly and chasing the prey with least performance. The updating is done using two basic concepts; direction and step length. These are two step lengths, one for the purpose of exploitation λ_{min} and the other for exploration λ_{max}. The direction of the predator is computed in such a way that it will run after a pray with least performance in the objection function and also run randomly. Suppose x_p is the predator then its new location will be:

$$x_p new = x_p old + \lambda_{max}\ rand + \lambda_{min}(x' - x_p old) \quad (4)$$

where, x is a prey with least survived value. The best prey will totally focus on exploitation. m random direction will be generated where m is an algorithm parameters which determine the number of the directions for local search. And also, if there is any direction which improves the performance of the best prey if it go in that direction then the best prey will done in that direction. Otherwise it will remain in its current position. The other prey follows

better prey where better preys are preys with better performance of the objective function provided that probability of following is met. If probability of follow up is not they will randomly run away from the predator. Hence, the predator works as an agent for exploration when as the best prey for exploitation.

The best prey in the other hand does only a local search for exploitation purpose. The main steps of the algorithm are as follows:

Step 1: Generate random solutions
Step 2: Calculate the performance of the solutions in the objective function and assign the solution with least performance as a predator, the solution with the best performance as best prey and the rest as preys
Step 3: Move the predator randomly and also towards the prey with least performance
Step 4: Generate random directions around the best prey. If there is any direction which increases the performance of the best prey then the best prey moves in that direction. Otherwise, keep the best prey in its current position
Step 5: If the probability of follow-up is met move the preys towards better preys and also with a local search. If the probability of follow-up is not met move then the preys randomly away from the predator
Step 6: Update list of preys best prey and predator
Step 7: If a termination criterion is met stop else go to step 3

RESULTS

Logic programming in radial basis function neural network: A proportional logic programming is a set of clauses which in turn consist of literals, i.e., atoms and negated atoms only. A clause has the form:

$$\bigvee_{i=1}^{n} A_i \leftarrow \bigvee_{j=1}^{m} B_j$$

Where:
A_i = An atom
B_j = A literal

A logic programming is in Conjunctive Normal Form (CNF) if and only if has the form:

$$P = \bigwedge_{i=1}^{n} F_i,\ n \in \mathbb{N}$$

where, F_i is a clause in Disjunctive Normal Form (DNF). A clause is in DNF if it is in form:

$$Z = \bigvee_{i=1}^{n} A_i,\ n \in \mathbb{N}$$

where, A_i are literals. Consider the clause (5) as an example of representing a logic programming in RBFNNs (Hamadneh *et al*., 2012a, b). By using Eq. 6, researchers have generated the training data as shown in Table 1. After 1000 iterations, researchers indicate that the SSE through PPA is considerably smaller than the SSE through GA as shown in Table 2. In addition, the performance of PPA on convergence speed is slightly better as shown in Fig. 2:

$$A_1 \vee A_2 \vee A_3 \vee A_4 \vee \neg A_5 \quad (4)$$

where, A_i is an atom for all i:

$$x = \sum \text{Boolean values of the atomes} - \sum \text{Boolean values of the negated atomes} \quad (5)$$

where, x is the input data in training data in a clause. The Boolean values are {0, 1}. Researchers have replaced F and T in the neural networks by 0 and 1, respectively to emphasize false and true.

Iris data set: In this study, researchers used the iris flowers data set (Merz and Murphy, 1996) as an example of data analysis in radial basis function neural network to evaluate the performance of GA-RBFNNs and PPA-RBFNNs. The structure of the networks is as follows: 4 input neurons, 12 hidden neurons and three output neurons. In iris flowers data set there are three iris

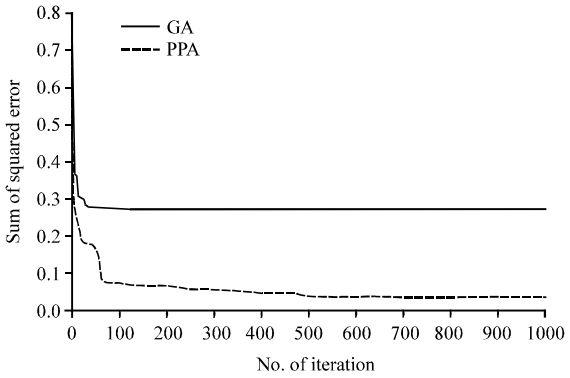

Fig. 2: SSE of PPA-RBFNN and GA-RBFNN on clause (6)

Table 1: Training data of clause (6)

Input data	Output target data
-1	0
0	1
1	1
2	1
3	1
4	1

Table 2: Result of PSO-RBFNN and GA-RBFNN on clause (6)

The algorithms	No. of input neurons	No. of hidden neurons	No. of output neurons	SSE
GA	5	3	1	0.2736
PPA	5	3	1	0.0359

Table 3: Results of PSO-RBFNN and GA-RBFNN on iris data

Algorithm	No. of input neurons	No. of hidden neurons	No. of output neurons	$\sqrt{\text{Total SSE}}$ of training data	$\sqrt{\text{Total SSE}}$ of testing data
GA	4	12	3	1.5143	2.1788
PPA	4	12	3	1.3217	1.9567

Table 4: Result of PSO-RBFNN and GA-RBFNN on the new thyroid data set

Algorithm	No. of input neurons	No. of hidden neurons	No. of output neurons	$\sqrt{\text{Total SSE}}$ of training data	$\sqrt{\text{Total SSE}}$ of testing data
GA	5	15	3	3.3818	3.5373
PPA	5	15	3	2.3622	2.6976

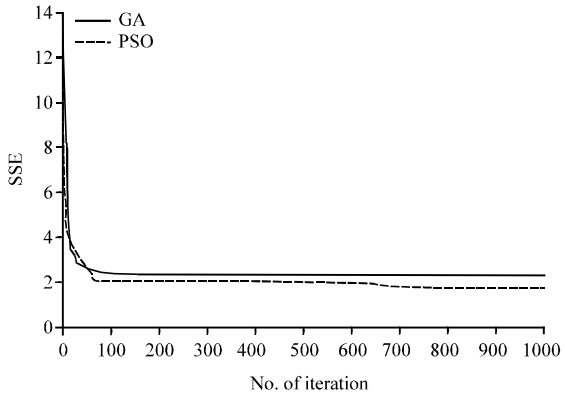

Fig. 3: SSE of PPA-RBFNN and GA-RBFNN on iris data

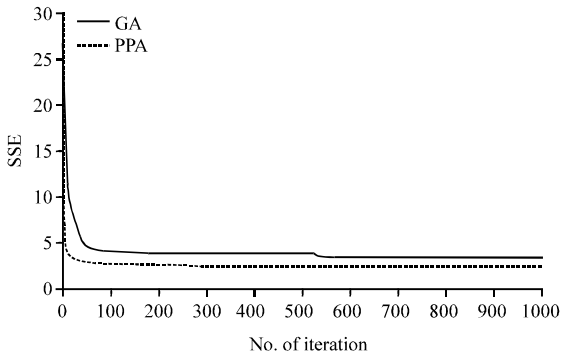

Fig. 4: SSE of PPA-RBFNN and GA-RBFNN on new thyroid data set

species, iris setosa, iris versicolor and iris virginica. The data consists of 150 data points from three classes divided to 50 data points for each class.

There are four dimensions (a multidimensional data set) to discriminate the data which are petal lengths, petal widths, sepal length and sepal widths. So, researchers used four input neurons in RBFNNs. To examine the performance of GA-RBFNN and PPA-RBFNN on iris flowers data set, the first ten points (represents 20%) for each kind for measurements will suffice these data are called training data. On the other hand, the second ten points (represents 20%) for each kind for measurements was used as a testing data. As a result of training GA-RBFNN and PSO-RBFNN, the sums of the squared errors for 1000 iterations are calculated as shown in Fig. 3. A comparison between the two algorithms indicates that even though the minimum SSE in PPA-RBFNNs is considerably smaller than the minimum SSE in GA-RBFNNs as shown in Table 3. The performance of PPA-RBFNNs on convergence speed is slightly better as shown in Fig. 3.

New thyroid data set: The new thyroid set was used for trying to predict the state of the thyroid gland (Merz and Murphy, 1996). It consists of three classes which are normal, hyper and hypo. Their classes consist of 150, 35 and 30 points, respectively. Attributes are T3-resin uptake test (A percentage). Total serum thyroxin as measured by the Isotopic Displacement Method, total serum triiodothyronine as measured by radioimmuno assay, basal Thyroid-Stimulating Hormone (TSH) as measured by radioimmuno assay and maximal absolute difference of TSH value after injection of 200 μg of thyrotropin-releasing hormone as compared to the basal value. Researchers used the first ten points in each class as a training data. On the other hand, the second ten points for each kind for measurements was used as a testing data. In this study, the structure of the network is as follows: 5 input neurons, 15 hidden neurons and 3 output neurons. As a result of training GA-RBFNN and PSO-RBFNN, the sums of the squared errors for 1000 iterations are calculated as shown in Fig. 4. A comparison between the two algorithms indicates that even though the minimum SSE in PPA is considerably smaller than the minimum SSE in GA as shown in Table 4. The performance of PPA on convergence speed is slightly better as shown in Fig. 4.

CONCLUSION

The artificial neural networks have been employed widely in a number of applications. This is because of their fault tolerance, fascinating characteristics of robustness, adaptive learning ability and massive parallel processing capabilities. In this study, researchers have developed successfully, the prey predator algorithm to be a new neural learning algorithm.

The computational results for training a logic programming data, iris data and new thyroid data set show the performance of GA and PPA in enhancing RBFNNs learning. The results are shown that the prey predator algorithm performs better than genetic algorithm in training radial basis function neural networks in minimizing the error. In addition, the convergent speed of the prey predator algorithm is better. In other words, the prey predator algorithm is faster than genetic algorithm in the training.

ACKNOWLEDGEMENT

Researchers would like to acknowledge the financial support of USM (short term grant 304/PMATHS/6312053) and MOSTI (E science grant 305/PMATHS/613142).

REFERENCES

Dhubkarya, D.C., D. Nagariya and R. Kapoor, 2010. Implementation of a radial basis function using VHDL. Global J. Comput. Sci. Technol., 10: 16-19.

Eberhart, R.C. and J. Kennedy, 1995. A new optimizer using particle swarm theory. Proceedings of the 6th International Symposium on Micro Machine and Human Science, October 4-6, 1995, Nagoya, Japan, pp: 39-43.

Hamadneh, N., S. Sathasivam and O.H.Choon, 2012a. Higher order logic programming in radial basis function neural network. Applied Math. Sci., 6: 115-127.

Hamadneh, N., S. Sathasivam, U.L. Tilahun and O.H. Choon, 2012b. Learning logic programming in radial basis function network via genetic algorithm. J. App. Sci., 12: 840-847.

Kennedy, J. and R.C. Eberhart, 1995. Particle swarm optimization. Proc. IEEE Int. Conf. Neural Networks, 4: 1942-1948.

Ling, T.L., M. Ahmad and L.Y. Heng, 2011. Quantitative determination of ammonium ion in aqueous environment using riegler's solution and artificial neural network. Sains Malaysiana, 40: 1105-1113.

Liu, X., 2010. Radial basis function neural network based on PSO with mutation operation to solve function approximation problem. Adv. Swarm Intell., 6146: 92-99.

Lowe, D., 1989. Adaptive radial basis function nonlinearities and the problem of generalisation. Proceedings of the 1st IEE International Conference on Artificial Neural Networks, October 16-18, 1989, London, UK., pp: 171-175.

Merz, C.J. and P.M. Murphy, 1996. UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Moody, J. and C.J. Darken, 1989. Fast learning in networks of locally tuned processing units. Neural Comp., 2: 281-294.

Parker, G.B. and Z. Lee, 2003. Evolving neural networks for hexapod leg controllers. Proceedings of the 2003 IEEE/RSJ International Conference on October 27-31, 2003 as Vegas, NV, USA, 1376-1381.

Rojas, R., 1996. Neural Networks: A Systematic Introduction. Springer, Berlin, ISBN-13: 978354060 5058, Pages: 502.

Sathasivam, S., N. Hamadneh and O.H. Choon, 2011. Comparing neural networks: Hopfield network and RBF network. Applied Math. Sci., 5: 3439-3452.

Schwenker, F., H.A. Kestler and G. Palm, 2001. Three learning phases for radial-basis-function networks. Neural Networks, 14: 439-458.

Taghi, M., V. Baghmisheh and N. Pavesic, 2004. Training RBF networks with selective backpropagation. Neurocomputing, 62: 39-64.

Tilahun, S.L., 2013. Prey-predator algorithmppa): A new metaheuristic optimization algorithm. PhD, Thesis, Universiti Sains Malaysia, Penang, Malaysia.

Ustun, S.V., 2007. GA-based optimization of PI speed controller coefficients for ANN-modelled vector controlled induction motor. J. Applied Sci., 7: 4001-4006.

Venkatesan, P. and S. Anitha, 2006. Application of a radial basis function neural network for diagnosis of diabetes mellitus. Curr. Sci., 91: 1195-1199.

Xiaobin, L., 2009. RBF neural network optimized by particle swarm optimization for forecasting urban traffic flow. Proceedings of the 3rd International Symposium on Intelligent Information Technology Application, November 21-22, 2009, University of Technology, Nanchang, China, pp: 124-127.

Zhang, G.P., B.E. Patuwo and M.Y. Hu, 1998. Forecasting with artificial neural networks: The state of the art. Int. J. Forecast., 14: 35-62.

Zhou, Z.H. and X.Y. Liu, 2006. Training cost-sensitive neural networks with methods addressing the class imbalance problem. Knowl. Data Engin. IEEE Trans., 18: 63-77.