American Journal of Intelligent Systems 2016, 6(3): 59-65
DOI: 10.5923/j.ajis.20160603.01

Classification with Some Artificial Neural Network Classifiers Trained a Modified Particle Swarm Optimization

Erdinç Kolay[1], Taner Tunç[2], Erol Eğrioğlu[3,*]

[1]Department of Statistics, Sinop University, Sinop, Turkey
[2]Department of Statistics, Ondokuz Mayıs University, Samsun, Turkey
[3]Department of Statistics, Giresun University, Giresun, Turkey

Abstract In this paper, we propose a new modified particle swarm algorithm for training some neural network classifiers for the most used classification problems in literature. Researches in artificial neural network field are based on different network architectures including multilayer perceptron, single multiplicative neuron and pi-sigma neuron model. To obtain a satisfactory performance for these classifiers, one of the most important issues is network training. Evolutionary algorithms are commonly used for training neural network classifiers. Particle swarm algorithm is population based, stochastic and meta-heuristic algorithm to solve optimization problems. As all evolutionary algorithms, the particle swarm algorithm may fall into local optimum and convergence rate may incredibly decline in iterative process. To overcome these shortcomings we refer to modify the particle swarm optimization with changing position matrix for each generation at iterative process. Experimental results show that training network with the proposed modified particle swarm optimization improve the classification performance for artificial neural network classifiers.

Keywords Neural Network Classifiers, Multilayer Perceptron, Pi-sigma Neural Network, Single Multiplicative Neuron, Particle Swarm Optimization

1. Introduction

Classification problem is defined as assigning an object to one of the predefined groups. The oldest known classification method is Discriminant Analysis and this method has well enough classification performance when only the assumptions are supplied. Artificial neural networks (ANNs) are mostly used for classification problems since a development of intelligent training algorithm [1] is called back propagation algorithm (BPA). The most of studies for classification problems with ANNs until the 2000s are summarized in [2].

Multi-layer perceptron (MLP) is most well-known ANN architecture. Even though MLP trained BPA has good classification performance, BPA suffers from a number of shortcomings, like its slow rate of convergence. Therefore, researchers have attempted to increase the performance of MLP by either modified BPA [3, 4] or different training algorithms [5-7].

Pi-sigma neural network, firstly introduced by [8], is higher-order feedforward network. Here probabilistic learning rule is used for training pi-sigma network which is applied to approximation and classification problems. Moreover pi-sigma network is used for image coding in [9], time series prediction in [10] and classification problems in [10, 11].

Single multiplicative neuron (SMN) model is firstly introduced by Yadav et. al. in [12] to obtain time series prediction. They used BPA for training network and the conclusions are compared with standard MLP. In [13], SMN network is trained by improved particle swarm optimization (PSO) and results are compared with BPA, standard PSO and genetic algorithm. To avoid falling into local optimum for BPA, [14] proposes improved BPA and implement to XOR and parity problems with SMN. Finally, [15] proposes using improved glow-worm swarm algorithm for training network for SMN.

PSO is firstly introduced by [16]. PSO, commonly used for neural network training [13, 17-19], is a population based, stochastic and meta-heuristic algorithm. In PSO, each particle searches global optimum point in multi-dimensional search-space. In iterative process each particle changes its position according to the individual best position and all particles' best position at history. As all evolutionary algorithms, the PSO may fall into local optimum and its

* Corresponding author:
erole1977@yahoo.com (Erol Eğrioğlu)
Published online at http://journal.sapub.org/ajis

convergence rate may incredibly decline in iterative process. To overcome these shortcomings, we refer to modify the PSO which is called "RPSO".

The rest of this paper is organized as follows. Section 2 reviews some artificial neural network including MLP, Pi-sigma and SMN. Section 3 reviews the standard PSO and our proposed RPSO is introduced in Sec. 4. Experiments and results are reported in Sec. 5. The paper is concluded in Sec. 6.

2. Artificial Neural Networks

2.1. Multilayer Perceptron

MLP, the most used neuron model in ANN literature, has a usage in very large areas in science. MLP's architecture can be shown as in Fig 1. In this study, we limit the number of hidden layer to one and only use sigmoid activation function for easy implementation and to avoid complexity. In Fig. 1 network's weights and thresholds can be optimized with any learning algorithm. Classification by this network is defined as follows,

$$\Sigma_i = (x_1 w_{i1} + x_2 w_{i2} + \ldots + x_q w_{iq}) + b_i,\ \text{i=1,...,k} \quad (1)$$

$$u_i = 1/(1+e^{-\Sigma i}) \quad (2)$$

$$\Sigma = \sum_{i=1}^{k} u_i v_i + b' \quad (3)$$

$$output_{(j)} = \frac{1}{1+e^{-\Sigma(j)}} \quad (4)$$

Here q is the number of variables, k is the number of hidden layer units, u is a sigmoid activation function, j is the number of observation, w_{ij} and b_i are network's weights and thresholds, respectively. Throughout the neural network training process, the variation between output value and desired value of class label will be done as possible as small by optimized parameters which are shown in Fig. 1 and also parameters vector (PV) which shown in (5). The vector of parameters including $k\times(q+2)+1$ element is to be optimized for train network:

$$PV_{MLP} = [w_{11}\ w_{12}\ldots w_{1q}\ b_1 | \ldots | w_{k1}\ w_{k2}\ldots w_{kq}\ b_k | v_1 \ldots v_k\ b'] \quad (5)$$

2.2. Pi-Sigma Neural Network

Pi-sigma neural network is higher order feedforward introduced by [8]. Pi-sigma network is similar to the MLP for their architectures which can be shown in Fig. 1 and Fig. 2. The most of important difference between two networks is to use the products of sum of k summing units for pi-sigma neural network. Outputs of the network are not affected from hidden layer thanks to fixed weights. Additionally, the lack of decision for hidden layer number may be advantage for pi-sigma neural network. To obtain output for pi-sigma network, (6) and (7) can be used. Calculations of the sums are similar to MLP in (1) and (2). Moreover, a vector of parameters including $k\times(q+1)$ element is to be optimized given as (8):

$$\Pi = \prod_{i=1}^{k} \Sigma_i \quad (6)$$

$$output_{(j)} = 1/(1+e^{-\Pi(j)}) \quad (7)$$

$$PV_{Pi-sigma} = [w_{11}\ w_{12}\ \ldots\ w_{1q}\ b_1 | \ldots | w_{k1}\ w_{k2}\ \ldots\ w_{kq}\ b_k] \quad (8)$$

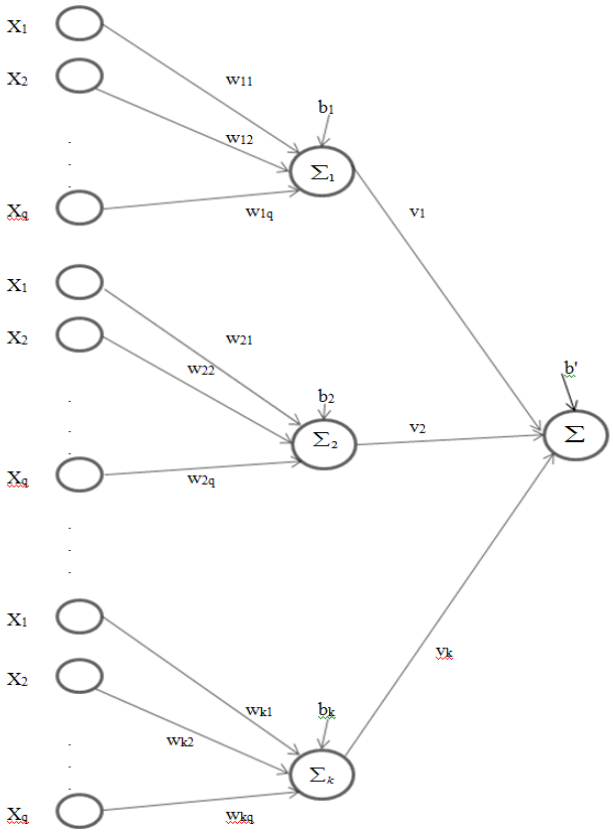

Figure 1. Multilayer Perceptron with k units in one hidden layer

2.3. Single Multiplicative Neuron Model

SMN model is firstly used for times series prediction by [12] and better performance compared with MLP is trained BPA. SMN's architecture is very simple and created only from one unit. This network has fewer parameters compared to others. This can be advantage in process of optimization. Fig. 3 shows SMN's architecture. To get output, (9) and (10) are used. Lack of any restriction for implementation of this model leads the application of the network easily to any problems as classification problems. Parameters vector (11) which consists of $2\times q$ parameter is to be optimized:

$$\Omega = \prod_{i=1}^{q} (w_i x_i + b_i) \quad (9)$$

$$output_{(j)} = \frac{1}{1+e^{-\Omega(j)}} \quad (10)$$

$$PV_{SMN} = \begin{bmatrix} w_1 & w_2 & \dots & w_q & b_1 & b_2 & \dots & b_q \end{bmatrix} \quad (11)$$

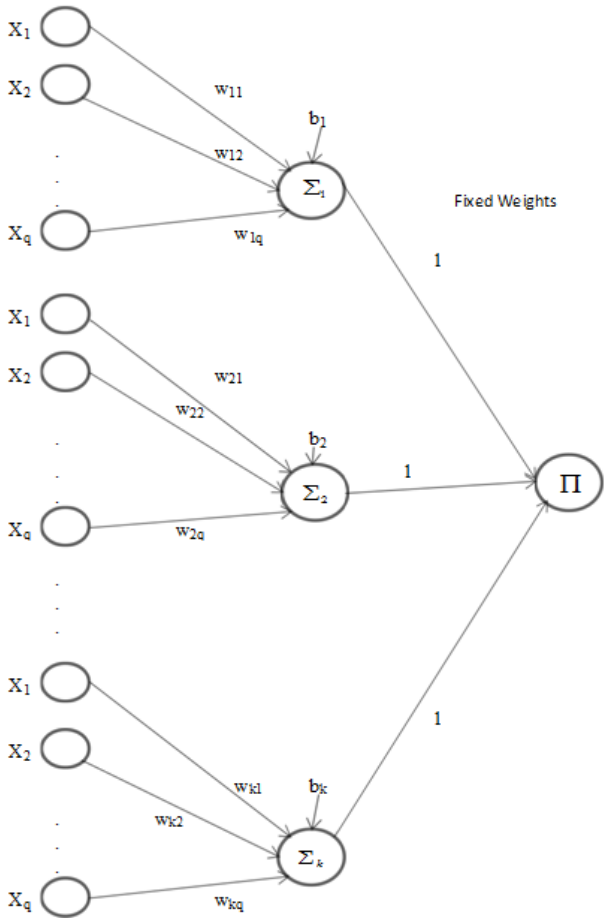

Figure 2. Pi-Sigma Neural Network with k summing units

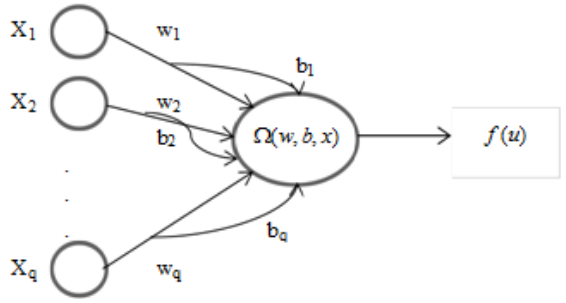

Figure 3. Single Multiplicative Neuron Model

3. Particle Swarm Optimization

Population based algorithms are very popular to solve optimization problems including neural network training. The PSO is a population-based algorithm, firstly introduced by [16]. In PSO each member of population called "particle" and the entire collection of particles are called as "swarm". This optimization technique simulates the social behaviour of swarms, such as bird flocking and fish schooling. Each particle has a random position, when the swarm initially departures for a destination. In the successive steps, each particle goes a new position by using its own previous experience (pbest) and the experience of the best positioned member of the swarm (gbest). In a *d*-dimensional search space, position and velocity vectors of the i^{th} particle can be represented as $X_i = (x_{i1}, x_{i2}, \dots, x_{id})$ and $V_i = (v_{i1}, v_{i2}, \dots, v_{id})$ respectively. These vectors update according to equations as follows

$$V_{id}(t+1) = w * V_{id}(t) + C_1 * rand * (P_{id}(t) - X_{id}(t)) + C_2 * rand * (P_{gd}(t) - X_{id}(t)) \quad (12)$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1) \quad (13)$$

where t is the current generation, w is the inertia weight, C_1 and C_2 are constants known as acceleration coefficients, P_{id} is the best position of i^{th} particle in the *d*-dimensional search-space called "personal best" and P_{gd} is the overall best solution obtained by swarm called "global best", "rand" is random number in the range [0 1]. The inertia weight is not included in the original PSO. Shi and Eberhart [20] referred to initial weight and improve the PSO performance. Most studies on PSO in the direction of bringing innovation are based on their model.

The most of the recent studies which are to improve the performance of PSO are based on modification of parameters in (12) which shows the change in location. Inertia weight gets an important effect on balancing the global search and local search. Some studies have modified inertia weight [21-24] to get better performance. Clerc [25] indicates that use of a constriction factor may be necessary to insure convergence of the particle swarm optimization. Previous studies of constriction coefficients show that there arises a new parameter detailed by [26]. Other studies that have been made to develop the performance of PSO, add a factor, etc.

4. A Modified PSO

In particle swarm optimization all particles are updated by (12) and (13). In this study, the usage of median which is a measure of central tendency is proposed, at each dimension, instead of the position of the particle which is giving the worst value of the objective function for all iterations. Also, inertia weight and constriction factors are calculated periodically as given in (14)-(16) which is referred in [27].

$$c_1(t) = (c_{1\max} - c_{1\min})(t / t_{\max}) + c_{1\min} \quad (14)$$

$$c_2(t) = (c_{2\max} - c_{2\min})(t / t_{\max}) + c_{2\min} \quad (15)$$

$$w(t) = (w_{\max} - w_{\min})(\frac{t_{\max} - t}{t_{\max}}) + w_{\min} \quad (16)$$

Here (C_{1min}, C_{1max}), (C_{2min}, C_{2max}) and (w_{min}, w_{max}) are intervals for the C_1, C_2 and w, respectively. Additionally, to avoid becoming trapped in a local optimum we propose to restart the position matrix during every i^{th} iteration, where i is the arbitrary integer, also P_{id} and P_{gd} remain in memory throughout the iteration process.

The purpose of using median is being a robust measure

among extreme values [28]. By using median, the particle giving the worst value of the function namely the farthest particle from the swarm is getting closer to the centre of the swarm. Therefore, the members of the swarm move closer to each other and reach faster to the optimum point. Since the acceleration is caused by an external force and the position of the particle is updated by this new acceleration, this can be interpreted as the entire swarm enforces the farthest particle to centralize through the swarm itself. Fig. 4 shows the particle's movement via using median. In this figure, the worst particle which is also the farthest one to the entire swarm is approximated to the coordinates labelled with a lighter colour. The lightly labelled coordinate approaches the centre of the swarm at each dimension, through the median. The working of optimization process can be shown with Pseudo Code of RPSO.

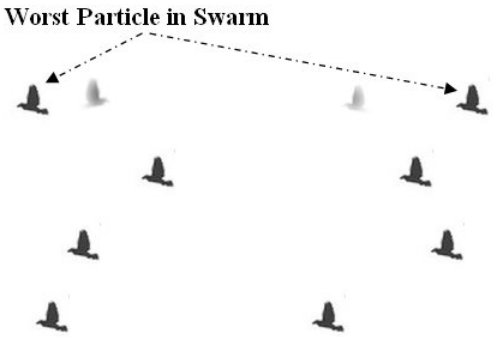

Figure 4. Movement of Worst Particle Position in Swarm

5. Experimental Design and Results

5.1. Experimental Design

Firstly, we first compare RPSO and the standard PSO algorithm with benchmark functions to search RPSO's convergence rate. Table 1 consists of the most commonly used benchmark functions. In our experiment, we use these functions for comparing two algorithms. The values of the learning factor are set C_1 = (1.2, 1.4995), C_2 = (1.2, 1.4995). Interval of inertia weight is taken w = (0.4, 0.9) referred by [29]. Population size is 30 and the dimensions (D) for all function in Table 1 are 30 and also the maximum iteration number is 1000 for two algorithms. During the iterative process, we restart the position matrix all fortieth iteration for iterative process and run this experiment 100 times independently.

Secondly, we use the most commonly used classification problems in Table 2 from UCI repository [30]. To train networks for all these classifiers, we try to optimize the parameters vector of each individual network given by (5), (8) and (11), respectively, using PSO and RPSO. MLP's hidden layer unit k and k summing unit for pi sigma network are taken 8 and 3, respectively. Moreover, we choose randomly 50% of total sample for training data and rest of data for testing for all datasets. We run this experiment 50 times independently and investigate misclassification rate for training and testing data.

Table 1. Benchmark Functions Used in the PSO and RPSO

Function's Name	Mathematical Representation	Range of Search
Sphere	$f_1(x)=\sum_{i=1}^{D} x_i^{\,2}$	$[-100,100]^D$
Griewank	$f_2(x)=\frac{1}{4000}\sum_{i=1}^{D} x_i^{\,2}-\prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]^D$
Rastrigin	$f_3(x)=\sum_{i=1}^{D}(x_i^{\,2}-10\cos(2\pi x_i)+10)$	$[-5.12,5.12]^D$
Rosenbrock	$f_4(x)=\sum_{i=1}^{D-1}(100(x_{i+1}-x_i^{\,2})^2+(x_i-1)^2)$	$[-30,30]^D$
Ackley's	$f_5(x)=-20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}}-e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)}+20+e$	$[-32,32]^D$
Step	$f_6(x)=\sum_{i=1}^{D}(\lvert x_i+0.5\rvert)^2$	$[-100,100]^D$

Table 2. Dataset from UCI repository

Dataset	*Number of attribute*	*Number of observation*
Statlog (Australian Credit Approval)	14	690
Breast Cancer (Diagnostic)	30	569
Pima Indians Diabetes	8	768
Ionosphere	34	351

Table 3. Comparison of the PSO and RPSO in term of Better Performance and Less Bias

Test Function	*PSO Mean(f)*	*PSO Std(f)*	*RPSO Mean(f)*	*RPSO Std(f)*
$f_1(x)$	9.5893e-05	7.5255e-05	8.0314e-05*	5.2325e-05
$f_2(x)$	3.5724e-05	2.5546e-05	2.4183e-05*	1.3223e-05
$f_3(x)$	13.7248	4.9175	11.3279*	7.2878
$f_4(x)$	28.3936	0.4489	28.4960**	0.3850
$f_5(x)$	0.0073	0.0031	0.0063*	0.0024
f6(x)	0.0011	8.0612e-04	3.1732e-05*	2.5586e-05

(*= 'Statistically Better', **=Statistically has no difference).

5.2. Experimental Results

Fig. 5 shows that the proposed RPSO converges faster than the standard PSO for five benchmark functions in Table 1 except for Rosenbrock function. We perform 100 independent runs to investigate the performance and bias of the standard PSO and RPSO. In Table 3, it is represented that the RPSO converges to the minimum values of the objective functions in Table 1 with better performance and less bias than the standard PSO. When comparing of the value of the mean for two algorithms, RPSO is statistically better with respect to Student-t distribution (Sig.<0.05).

Table 4 shows the decrease in the misclassification rate for both training sample and test sample when MLP, pi-sigma and SMN classifiers are trained with RPSO. Moreover misclassification rate's bias is less than standard PSO's bias. Training network with RPSO increase considerably the correct classification rates for all data.

This results show that RPSO is robust learning algorithm for solving optimization problems like approximation and classification problems.

6. Conclusions

The performance of correct classification is characterized by the neural network's architectures and intelligent learning algorithms. For obtaining good classification performance either various the neural network's architecture or a modified learning algorithm can be used. To avoid falling into error researchers pay attention choosing neural network's architectures. According to "no free lunch" theorem, which is referred by [31], the best classifier is not same for all dataset and we see this theorem's results computationally when we analyse Table 4. From this point of view, we use three kind of neural network classifiers trained the modified particle swarm optimization to enhance the classification performance or minimise the misclassification error rate. As all evolutionary algorithms, PSO can easily fall into local optimum and speed of convergence may be reduced. To overcome this shortcoming, there are several ways to modify the PSO. To improve the PSO performance, some studies modify the parameters like inertia weights, C_1 and C_2 coefficients and random value etc. For this issue, we propose upgrading swarm position matrix by using median instead of the position of the particle giving the worst value of the objective function for all iterations. The results of using our modified PSO show that the proposed RPSO for performance of training neural networks is robust and better performance to estimate misclassification rates both training sample and test sample. RPSO is significantly different and has less deviation than PSO statistically (Sig<0.05). Experimental results verify this conclusion.

Table 4. Comparison of the PSO and RPSO for Neural Network Classifiers

Datasets	Neural Network's Structure	Train Set		Test Set	
		PSO	RPSO	PSO	RPSO
Statlog	MLP	0.1943(0.0391)	0.1227(0.0169)	0.1984(0.0451)	0.1530(0.0161)
	Pi-Sigma	0.1853(0.0342)	0.1352(0.0218)	0.1968(0.0371)	0.1589(0.0171)
	SMN	0.1659(0.0437)	0.1539(0.0298)	0.1612(0.0483)	0.1512(0.0316)
Breast Cancer	MLP	0.1185(0.0353)	0.0304(0.0149)	0.1161(0.0348)	0.0551(0.0220)
	Pi-Sigma	0.0777(0.0223)	0.0321(0.0149)	0.0981(0.0287)	0.0522(0.0191)
	SMN	0.1283(0.0374)	0.1073(0.0255)	0.1386(0.0360)	0.1249(0.0388)
Pima	MLP	0.3187(0.0258)	0.2267(0.0222)	0.3285(0.0261)	0.2512(0.0192)
	Pi-Sigma	0.2837(0.0231)	0.2373(0.0209)	0.3020(0.0314)	0.2657(0.0296)
	SMN	0.3097(0.0448)	0.2665(0.0337)	0.3238(0.0440)	0.2819(0.0411)
Ionosphere	MLP	0.2527(0.0404)	0.0968(0.0318)	0.2885(0.0576)	0.1732(0.0404)
	Pi-Sigma	0.1857(0.0304)	0.1444(0.0278)	0.2373(0.0322)	0.2170(0.0371)
	SMN	0.2722(0.0373)	0.2135(0.0354)	0.3113(0.0524)	0.2548(0.0465)

Pseudo Code of RPSO
Initialize of velocity and position matrix of particles
Calculate the objective functions for all particles and find pbest and gbest in (12).
For i=1:max$_{iteration}$
Update velocity by (12) and Calculate position matrix by (13).
Find the particle giving the worst function's value and use medians of coordinates of each particle at each dimension instead of worst particle's position.
For every j. iteration (e.g., 40 or 50) position matrix is restart in the other words position matrix will be initialize again.
If new pbest is better than last pbest set the current value new pbest.
Choose the particle with the best objective function value of all of the particles as gbest.
End

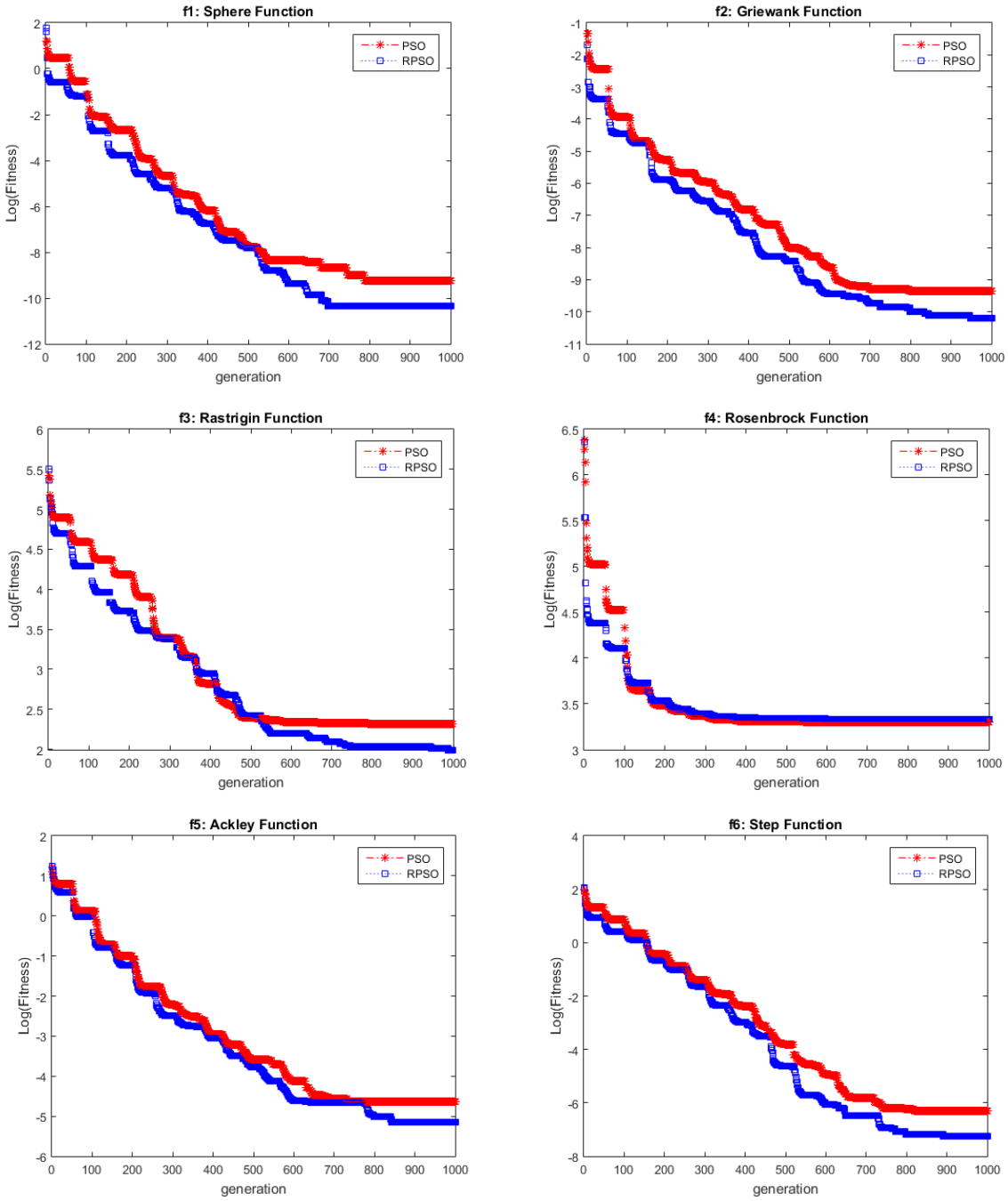

Figure 5. Convergence Curves for the Standard PSO and RPSO

REFERENCES

[1] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning Internal Representations by Error Propagation, in: Readings Cogn. Sci., Elsevier, 1988: pp. 399–421.

[2] G.P. Zhang, Neural networks for classification: a survey, IEEE Trans. Syst. Man Cybern. Part C (Applications Rev. 30 (2000) 451–462.

[3] B.B. Chaudhuri, U. Bhattacharya, Efficient training and improved performance of multilayer perceptron in pattern classification, Neurocomputing. 34 (2000) 11–27.

[4] S. Abid, R. Fnaicch, M. Najim, A fast feedforward training algorithm using a modified form of the standard backpropagation algorithm, IEEE Trans. Neural Networks. 12 (2001) 424–430. doi:10.1109/72.914537.

[5] P.A. Castillo, J.J. Merelo, A. Prieto, V. Rivas, G. Romero, G-Prop: Global optimization of multilayer perceptrons using GAs, Neurocomputing. 35 (2000) 149–163.

[6] J. Ilonen, J.-K. Kamarainen, J. Lampinen, Differential Evolution Training Algorithm for Feed-Forward Neural Networks, Neural Process. Lett. 17 (2003) 93–105.

[7] D. Wang, W.Z. Lu, Forecasting of ozone level in time series using MLP model with a novel hybrid training algorithm, Atmos. Environ. 40 (2006) 913–924.

[8] Y. Shin, J. Ghosh, The pi-sigma network: an efficient higher-order neural network for\npattern classification and function approximation, IJCNN-91-Seattle Int. Jt. Conf. Neural Networks. i (1991) 13–18.

[9] A. J. Hussain, P. Liatsis, Recurrent pi-sigma networks for DPCM image coding, Neurocomputing. 55 (2003) 363–382.

[10] C.K. Li, A sigma-pi-sigma neural network (SPSNN), Neural Process. Lett. 17 (2003) 1–19.

[11] S. Panigrahi, A.K. Bhoi, Y. Karali, A Modified Differential Evolution Algorithm trained Pi-Sigma Neural Network for Pattern Classification, (2013) 133–136.

[12] R.N. Yadav, P.K. Kalra, J. John, Time series prediction with single multiplicative neuron model, Appl. Soft Comput. 7 (2007) 1157–1163. doi:10.1016/j.asoc.2006.01.003.

[13] L. Zhao, Y. Yang, PSO-based single multiplicative neuron model for time series prediction, Expert Syst. Appl. 36 (2009) 2805–2812. doi:10.1016/j.eswa.2008.01.061.

[14] K. Burse, M. Manoria, V.P.S. Kirar, Improved back propagation algorithm to avoid local minima in multiplicative neuron model, Commun. Comput. Inf. Sci. 147 CCIS (2011) 67–73.

[15] H. Cui, J. Feng, J. Guo, T. Wang, A novel single multiplicative neuron model trained by an improved glowworm swarm optimization algorithm for time series prediction, Knowledge-Based Syst. 88 (2015) 195–209.

[16] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, MHS'95. Proc. Sixth Int. Symp. Micro Mach. Hum. Sci. (1995) 39–43.

[17] Y. Da, G. Xiurun, An improved PSO-based ANN with simulated annealing technique, Neurocomputing. 63 (2005) 527–533.

[18] G. Das, P.K. Pattnaik, S.K. Padhy, Artificial Neural Network trained by Particle Swarm Optimization for non-linear channel equalization, Expert Syst. Appl. 41 (2014) 3491–3496.

[19] A. P. Engelbrecht, Cooperative Learning in Neural Networks using Particle Swarm Optimizers, Annu. Res. Conf. South African Inst. Comput. Sci. Inf. Technol. (2001) 84–90.

[20] Y. Shi, R. Eberhart, A modified particle swarm optimizer, 1998 IEEE Int. Conf. Evol. Comput. Proceedings. IEEE World Congr. Comput. Intell. (Cat. No.98TH8360). (1998) 69–73.

[21] J.C. Bansal, P.K. Singh, M. Saraswat, A. Verma, S.S. Jadon, A. Abraham, Inertia weight strategies in particle swarm optimization, Proc. 2011 3rd World Congr. Nat. Biol. Inspired Comput. NaBIC 2011. (2011) 633–640.

[22] H. Geng, Y. Huang, J. Gao, H. Zhu, A self-guided Particle Swarm Optimization with Independent Dynamic Inertia Weights Setting on Each Particle, Appl. Math. Inf. Sci. 7 (2013) 545–552.

[23] Y. Feng, G.F. Teng, A.X. Wang, Y.M. Yao, Chaotic inertia weight in particle swarm optimization, Second Int. Conf. Innov. Comput. Inf. Control. ICICIC 2007. (2008) 7–10.

[24] Yuhui Shi, R.C. Eberhart, Fuzzy adaptive particle swarm optimization, in: Proc. 2001 Congr. Evol. Comput. (IEEE Cat. No.01TH8546), IEEE, 2001: pp. 101–106.

[25] M. Clerc, The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization, Proc. 1999 Congr. Evol. Comput. CEC 1999. 3 (1999) 1951–1957.

[26] D. Bratton, J. Kennedy, Defining a Standard for Particle Swarm Optimization, 2007 IEEE Swarm Intell. Symp. (2007) 120–127.

[27] U. Yolcu, E. Egrioglu, C.H. Aladag, A new linear & nonlinear artificial neural network model for time series forecasting, Decis. Support Syst. 54 (2013) 1340–1347.

[28] R.A. Maronna, R.D. Martin, V.J. Yohai, Robust Statistics, John Wiley & Sons, Ltd, Chichester, UK, 2006.

[29] B. Gao, X. Ren, M. Xu, Procedia Engineering An improved particle swarm algorithm and its application, 15 (2011) 2444–2448.

[30] C.L. Blake, C.J. Merz, UCI Repository of machine learning databases, Univ. Calif. (1998) http://archive.ics.uci.edu/ml/.

[31] D.H. Wolpert, The Lack of A Priori Distinctions Between Learning Algorithms, Neural Comput. 8 (1996) 1341–1390.