Journal of mathematics and computer science 12 (2014), 113-123

Contents list available at JMCS

Journal of Mathematics and Computer Science

Journal Homepage: www.tjmcs.com

Comparison of Particle Swarm Optimization and Backpropagation Algorithms for Training Feedforward Neural Network

Nasser Mohammadi[1*], Seyed Javad Mirabedini[2]

[1]*Department of computer engineering, Tehran Science and Research Branch, Islamic Azad University, Damavand, Iran*

[2]*Department of computer engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran*

***E-mail:** n.mohammadi_uni@yahoo.com*

Article history:
Received July 2014
Accepted August 2014
Available online August 2014

Abstract

An interesting tool for non-linear multivariable modeling is the Artificial Neural Network (ANN) which has been developed recently. The use of ANN has been proved to be a cost-effective technique. It is very important to choose a suitable algorithm for training a neural network. Generally Backpropagation (BP) algorithm is used to train the neural network. While these algorithms prove to be very effective and robust in training many types of network structures, they suffer from certain disadvantages such as easy entrapment in a local minimum and very slow convergence. In this paper, to improve the performance of ANN, the adjustment of network weights using Particle Swarm Optimization (PSO) was proposed as a mechanism and the results obtained were compared with various BP algorithms such as Levenberg-Marquardt and gradient descent algorithms. Each of these networks runs and trains for different learning ratios, activation functions and numbers of neurons within their hidden layer. Among different criteria Mean Square Error (MSE) and Accuracy are the main selected criteria used for evaluating both models. Also the MSE was used as a criterion to specify optimum number of neurons in hidden layer. The results showed that PSO approach outperforms the BP for training neural network models.

Keywords: Particle Swarm Optimization, Backpropagation, Artificial Neural Network.

1. Introduction

ANN is a parallel distributed processor which can express nonlinear and complicated relationship using input-output training patterns from the experimental data. Generally neural network learning is a nonlinear minimization issue with many local minimum [1] which depends on network weights, learning rules and architecture [2].One of the most common neural network architectures is Feedforward Neural Network (FNN).Feedforward means that data flows in one direction from input to

output layer (forward). Among the different FNNs, we choose Multilayer Perceptron (MLP) which is widely used for pattern classification, recognition, prediction and approximation. In fact, MLP can solve problems which are not linearly separable. These networks are normally arranged in three layers of neurons: the input and output layers which represent the input and output variables of the model and one or more hidden layers which lie between them and contain the network's ability to learn non-linear relationships [3].

ANNs have a special ability to approach the dynamics of nonlinear systems in many applications in a black box way. Generally, the development of a good ANN model depends on various operators. The first operator is identified with the input-output data driven, where model qualities are mostly affected by the quality of data being used. The second operator is related to the network architecture.Various network architectures result in different performances. The model size and its complexity are the third operator in which a small network may not be able to depict the real situation of the model assessment because of its limited ability, while a large network may have noise in the training data and therefore fail to provide good generalization ability. The last operator is related to the quality of the process model and strongly relies on the network training. Among all, the last issue may be the most important, because it is mainly an identification of model parameters that fits with the given data. Although the BP algorithm is commonly used in recent years to perform the training task, some drawbacks are often encountered using this gradient-based method, include: very slow training convergence speed and getting stuck in a local minimum easily [4, 5]. In order to solve these drawbacks different algorithms have been proposed [6, 7 and 8]. This disadvantage can be removed by an exploration ability of the evolutionary algorithms such as PSO. This paper represents the performance comparison of the FNNs using various BP algorithms with PSO. The various BP training algorithms used are gradient descent, gradient descent with adaptive learning, gradient descent with momentum, gradient descent with momentum and adaptive learning and also Levenberg-Marquardt.

Following this introductory section, the rest of the paper is organized as follows: In section 2 related works are presented then in section 3 the ANN training process is defined while the related subsections are allocated to explaining of three different algorithms for training ANNs; Gradient decent, LM and PSO. German dataset used in this paper is stated in section 4. Model performance assessment criteria, Accuracy and MSE, are specified in the next section, then in section 6 our proposed methodology with its flowchart are analyzed clearly. This paper is finished byillustrating the results in section 7 and the last section is allocated to the conclusions of this study.

2. Related works

There are many researches in the field of training neural networks. Lahmiri in 2011 measured the Accuracy of MLP networks trained with different heuristic and numerical algorithms. He found that BFGS conjugate algorithm and Levenberg-Marquardt are the best in terms of Accuracy and numerical algorithm outperform heuristic [9]. Hooshyaripor and Tahershamsi in 2012 provided a review of some methods for estimation of peak outflow from breached dams and presented an effective and efficient model for predicting peak outflow based on ANN. By comparing the results of ANN and empirical formulas they found the higher ANN performance and declared that such formulas are better to be replaced with a superior ANN model [10]. Mirjalili et al. in 2012 employed Gravitational Search Algorithm (GSA) and PSOGSA as new training methods for FNNs to examine the efficiencies of these algorithms in reducing the problems of trapping in local minimum and the slow convergence rate of current evolutionary learning algorithms. The results were compared with a standard PSO-based algorithm. The experimental results showed that PSOGSA outperforms both PSO and GSA for training FNNs [11]. Yaghini et al. in 2013 presented a method for training ANN. They combined the ability of meta-heuristics and greedy gradient based algorithms to obtain a hybrid improved opposition based

PSO and a BP algorithm with the momentum term. They proposed a new cross validation method to prevent over-fitting then the effectiveness and efficiency of the proposed method were compared with several other famous ANN training algorithms on various benchmark problems [12]. Das et al. in 2013 applied ANN trained with PSO for the problem of channel equalization. They employed PSO to optimize the number of layers, the type of transfer functions, input and hidden neurons etc. [13].

3. ANN training process

The training process of ANN is usually complicated and high dimensional. Until today, many researchers prefer to use BP algorithms to train ANNs. BP works by measuring the output error, calculating the gradient of this error and adjusting the ANN weights and biases in the descending gradient direction. These gradient methods estimate the error in the network's decision as compared to a supervisor and propagate the error to the weights throughout the network, so that one of the main obstacles due to the fact that searching of optimal weights is strongly dependent on initial weights, and if they are located near local minimum, the algorithm would be stuck at a sub-optimal solution. Hence, the conventional gradient search method is prone to be converged at local optima. Many solutions are proposed by neural network researchers to overcome the slow convergence rate and being trapped in a local minimum. Some powerful optimization algorithms, which are based on a simple gradient descent algorithm [14] such as conjugate gradient descent, scaled conjugate gradient descent, Quasi-Newton BFGS and Levenberg-Marquardt methods, have been devised to improve the convergence rate. Since evolutionary algorithms (EAs) do not use gradient information, it is advantageous for problems where such information is unavailable or very costly to obtain. These advantages make them more robust and attractive than many other search algorithms [15]. The PSO as an evolutionary algorithm is easy to implement and performs well on many optimization problems. Like other evolutionary techniques, PSO could also be used in neural network training. Among many algorithms for training ANNs the following algorithms were used in this study:

3.1. Gradient descent

This algorithm is one of the most common training algorithms in the field of neural networks. At first it measures the output error, then calculates the gradient of this error and finally adjusts the ANN weights and biases in the descending gradient direction. The learning rate is a tunable factor that controls the speed of the learning process. The ANN model will learn faster with a faster learning rate, but the training process will never converge if it is too high. In contrast, if the learning rate is too slow, the ANN model may be caught in a local minimum instead of the global minimum. In order to avoid local minimum, reduction of oscillation and to decrease the sensitivity of the network to fast changes of the error surface, the change in weight is made dependent on the past weight change by adding a momentum coefficient [16]. The proportion of the last weight change added into the new weight change is specified by momentum coefficient.

3.2. Levenberg–Marquardt

One of the most popular tools for non-linear minimum mean squares problems is LM which is another type of BP training algorithm. The LM algorithm approximates to the Newton Method and has been also used for ANN training. The Newton method estimates the error of the network with a second order expression which is in contrast to the prior category which follows a first order expression. LM is popular in the field of ANN as well as it is considered as the first approach for an unobserved MLP training task and is more powerful than conventional gradient descent techniques [17, 18].

3.3. Particle swarm optimization

Eberhart and Kennedy in 1995 [19] developed a global optimization technique, PSO, which is a group-based stochastic optimization technique for continuous nonlinear functions. In comparison with other Meta heuristics, PSO has obtained popularity and showed clearly to be an effective and competitive optimization algorithm. Each member in PSO algorithm known as particle flies around the multi-dimensional search space with a velocity, which is continuously brought up to date by the particle's own experience and the experience of the particle's neighbors or the experience of the entire swarm. It means that two discrepancies of the PSO algorithm are developed; PSO with a local neighborhood and PSO with a global neighborhood. According to the global surroundings, each particle moves towards its best previous position and towards the best particle in the entire swarm, called gbest model [20, 21 and 22]. On the other hand, according to the local discrepancy, called lbest, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood [19]. Whereas PSO has memory of the past, knowledge of good solutions is kept by all particles. Particles works together in a useful manner and can share information in the swarm.

3.3.1. PSO in ANNs: Unlike BP, PSO is a global search and population-based algorithm which has been used for training neural networks, finding neural network architectures, tuning network learning parameters, and optimizing network weights. PSO avoids trapping in a local minimum, because it is not based on gradient information [23]. The function of PSO in ANN is to get the best set of weights (particle position) where several particles are trying to move to get the best solution. The dimension of the search space is the total number of weights and biases.Through following the personal best solution of each particle and the global best amount of the entire swarm, the algorithm finishes the optimization. The success or failure of a population based algorithm depends on its ability to establish proper trade-off between exploration and exploitation. An unsuitable balance between exploration and exploitation may result in a weak optimization method which may suffer from premature convergence, trapping in a local optimum and stagnation.

4. Dataset

In this study, we use the UCI machine learning database, one of the 100 reliable databases, which has the most references in scientific papers available via http://archive.ics.uci.edu/ml/datasets.html. The German dataset is the credit dataset used in this study to evaluate the models. Table1 shows a summary of main features of this credit dataset.

Table 1. Some Details of dataset used in study

No.	Dataset	No. of Attributes	No. of Good Instances	No. of Bad Instances
1	German	21	700	300

In data normalization process the input data are normalized between 0 and 1. This is performed by finding the maximum number in each column (feature) for all instances and dividing the rest of the entries in each column to its maximum value.

5. Model performance assessment criteria

The performance assessment criteria used in this study include Accuracy and MSE. Accuracy indicates the proportion of the correctly classified cases on a particular dataset and is defined by Equation (1);

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \quad (1)$$

Where, TP (True Positive): is the proportion of positive cases that are correctly diagnosed as positive; FP (False Positive): is the proportion of negative cases that are wrongly diagnosed as positive; FN (False Negative): is the proportion of positive cases that are wrongly diagnosed as negative; TN (True Negative): is the proportion of negative cases that are correctly diagnosed as negative.

MSE is the difference between the actual and predicted value by the model. This measure is computed by Equation (2) and our target is to minimize this value;

$$e = \hat{y} - y \Rightarrow \text{MSE} = \frac{1}{N}\sum_{i=1}^{N} e_i^2 \quad (2)$$

Where, e is the difference between $\hat{y}$ as predicted value and y as the actual value.

6. Methodology

In this study, firstly, in order to find the best BP neural network different learning algorithmswere used which are shown in Table 2. ANNs are constructed based on learning algorithm, activation function and the number of neurons as the parameters. Also, log-sigmoid and hyperbolic tangent sigmoid are used as activation functions (other activation functions were not applicable in this case). The BP learning algorithms are examined by different numbers of training and test data ratios (or as we refer to them, learning ratios). The lower the ratio, the more challenging it is for a neural network, but the learning is more robust and meaningful. Learning rate of 0.01 and a momentum coefficient of 0.9 was considered in this work for BP Learning.

Table 2. Different learning algorithms

No.	Abbreviation	Learning algorithmdescription
1	LM	Levenberg-Marquardt
2	GD	Gradient descent
3	GDM	Gradient descent with momentum
4	GDA	Gradient descent with adaptive learning rate
5	GDMA	Gradient descent with momentum and adaptive learning rate

Another factor which should be considered in constructing an ANN is the number of neurons present in the hidden layer. Each of these five networks with considering the learning ratio and activation function runs and trains for different numbers of neurons from 20 up to 50 within their hidden layer. Training process performs 10 times for each number of these neurons as well. The optimum number of neurons in hidden layer is specified according to the minimum value of MSE on test data. We compare the performance of the neural network models under all schemes and then select the ideal neural model and learning scheme.

Secondly, to enhance the convergence rate and learning process, PSO is used to train MLP neural network. The learning process includes finding a set of weights to minimize the learning error. A set of weights for current iteration shows the position of each particle and the number of weights associated with the network represents the dimension of each particle. The goal in this algorithm is to minimize a learning error (cost function) which is calculated by using MSE. The particle will move inside the weight space trying to minimize learning error. The PSO procedure is presented in Figure 1. The iterative approach of PSO can be described by the following steps:

1) Initialize position and velocity of all the particles randomly in the d-dimension space.
2) Training the PSO-ANN by using the particles position and determine MSE (particle fitness) for each particle.
3) The current position and fitness achieved by particle p is set as its best history amount, also called the personal best (pbest).The pbest with best value in all particles are set as global best (gbest).
4) Change the velocity of the particle according to Equation (3).
5) Update particle position by adding the calculated velocity value to the current position value according to Equation (4).
6) Using the new sets of positions to generate new learning error.
7) Comparing the MSE of each particle with its pbest MSE then updating pbest, if the current MSE is lower than the pbest MSE.
8) Finding the minimum calculated MSE in the swarm then comparing it by the global best MSE then updating gbest, if the minimal MSE is lower than gbest MSE.
9) The optimization output is based on gbest position value. The iteration loop continues until reaching the MSE of the gbest lower than the desirable threshold or a maximum iteration number. The gbest weights are used as the training results when the iteration is finished.

$$v_{id}[t+1] = wv_{id}[t] + c_1 r_1 (p_{id}[t] - x_{id}[t]) + c_2 r_2 (g_d[t] - x_{id}[t]) \quad (3)$$

$$x_{id}[t+1] = x_{id}[t] + v_{id}[t+1] \quad (4)$$

Where, a d-dimensional vector in problem space $x_{id} = (x_{i1}, x_{i2}, \ldots, x_{id})$ represents the position of each particle, $v_{id} = (v_{i1}, v_{i2}, \ldots, v_{id})$ is the velocity of the ith particle,$p_{id} = (p_{i1}, \ldots, p_{id})$ is the best position encountered by ith particle (pbest), g_d shows the best position found by any member in the entire swarm population (gbest), t is iteration counter; c_1, c_2 are acceleration coefficients and r_1, r_2 are two similar random numbers in [0, 1]. Parameters used for running PSO are shown in Table 3:

Table 3. Parameters used for running PSO

Parameter	Value
No. of population	20
No. of generations	300
C_1,C_2	2
Inertia weight	1
Inertia Weight Damping Ratio	0.9
Search space range	(-1,1)
Maximum velocity	0.18
Minimum velocity	-0.18

The acceleration coefficients control how far a particle will move in a single iteration. While high values result in abrupt movement and low values allow particles to drift far from target regions. The effect of previous histories of velocities on present velocity is often used as a parameter to control the trade-off between exploration and exploitation by the inertia weight (w). The inertia weight was introduced by Shi and Eberhart in 1998 [24]to improve the convergence rate of PSO algorithm. Particle's velocity is limited by a user specified value, maximum velocity (v_{max}) and minimum velocity (v_{min}) to prevent the particles from moving too far from potential solution.

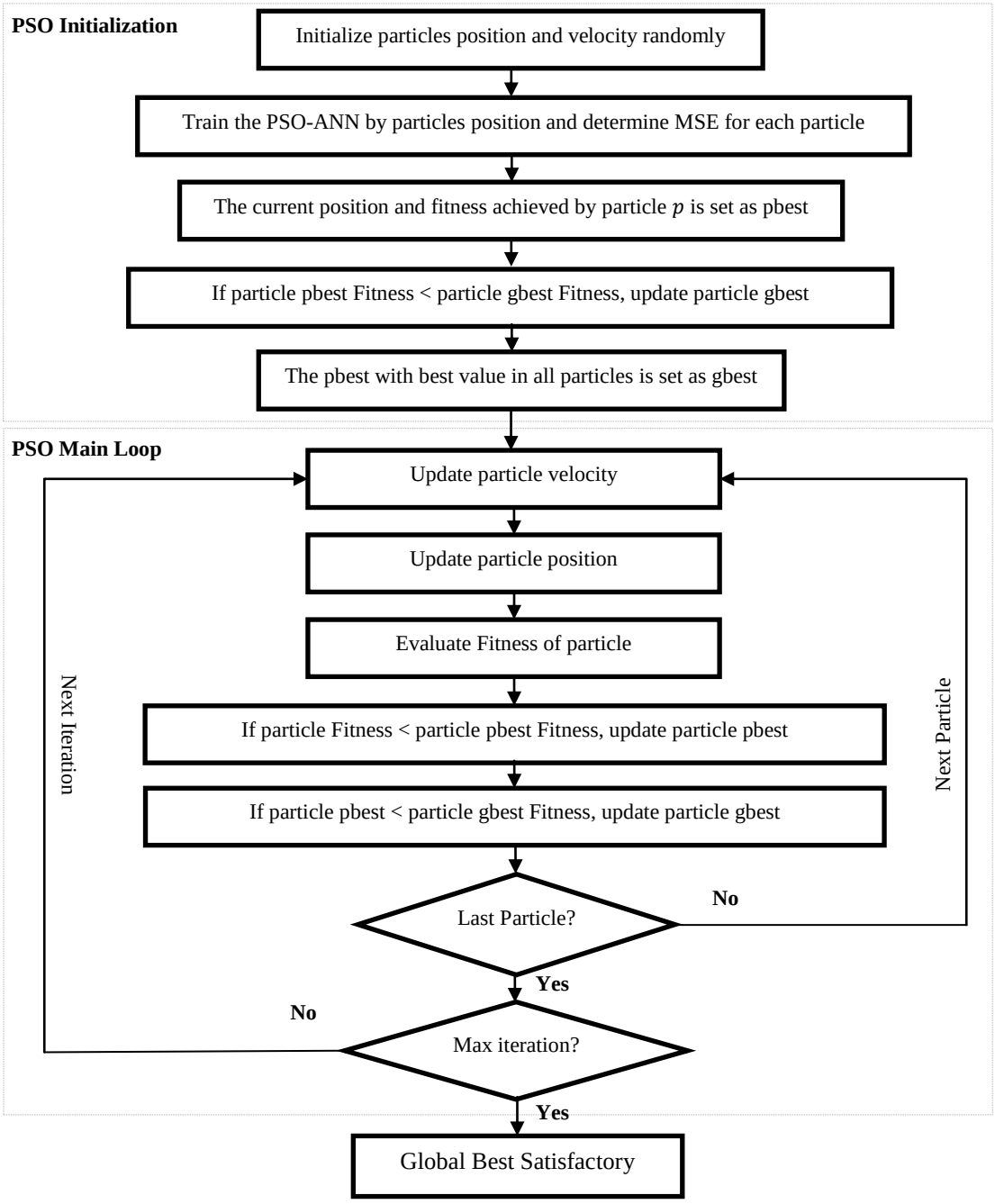


Figure 1. PSO flow chart

7. Results

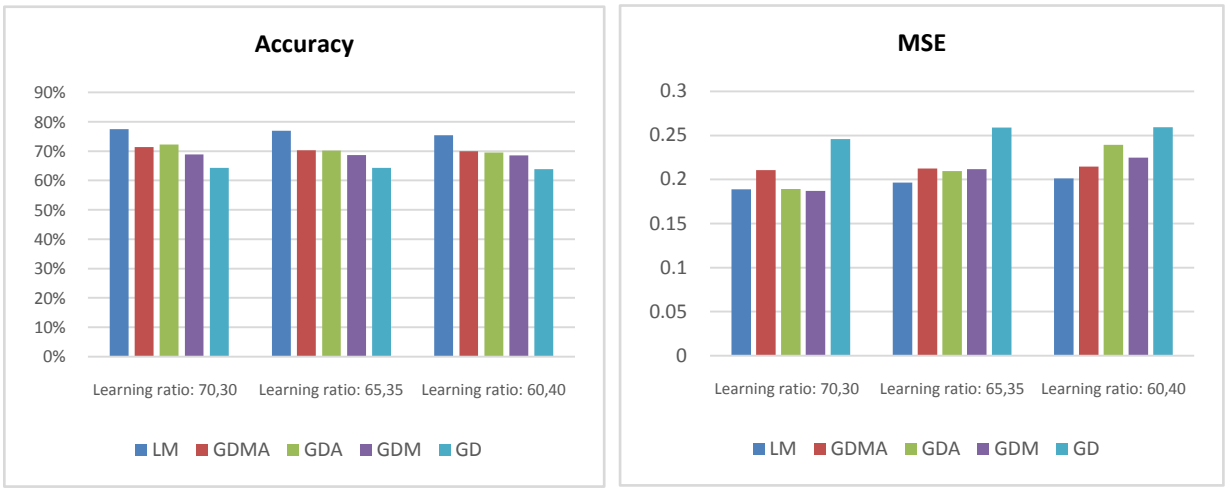
At first, the performance of the ANN Backpropagation (ANN-BP) models was evaluated based on the optimum number of neurons in hidden layer calculated according to MSE criterion, i.e. the models have the lowest MSE for this number of neuron. The obtained results of implementation of the MLP neural network models with five BP learning algorithms are listed in detail in Table 4, which shows that the preferred ANNs are the ones using LM as learning algorithm with Log-sigmoid and Tan-

sigmoid as activation functions in hidden and output layers respectively. The results show that accurate selection of the activation function has a great impact on network performance. For each learning algorithm, the model with the highest Accuracy is highlighted in boldface. In order to simulate the considered MLP neural network, MATLAB version R2012a software was used.

Table 4. Implementation results of different BP learning algorithms

No.	Learning Algorithm	Learning Ratio	Activation Functions (hidden, output layers)	Optimum No. of Neurons in Hidden Layer	MSE	Accuracy
1	LM	**70,30**	**Log-sigmoid, Tan-sigmoid**	**23**	**0.1890**	**77.52%**
			Tan-sigmoid, Tan-sigmoid	21	0.1989	77.45%
		65,35	Log-sigmoid, Tan-sigmoid	28	0.1964	76.90%
			Tan-sigmoid, Tan-sigmoid	25	0.1969	76.80%
		60,40	Log-sigmoid, Tan-sigmoid	24	0.2012	75.40%
			Tan-sigmoid, Tan-sigmoid	30	0.2058	75.20%
2	GD	**70,30**	**Log-sigmoid, Tan-sigmoid**	**34**	**0.2458**	**64.30%**
			Tan-sigmoid, Tan-sigmoid	21	0.2441	64.20%
		65,35	Log-sigmoid, Tan-sigmoid	24	0.2590	64.30%
			Tan-sigmoid, Tan-sigmoid	27	0.2417	64%
		60,40	Log-sigmoid, Tan-sigmoid	44	0.2593	63.90%
			Tan-sigmoid, Tan-sigmoid	38	0.2698	63.60%
3	GDM	**70,30**	**Log-sigmoid, Tan-sigmoid**	**35**	**0.1871**	**68.90%**
			Tan-sigmoid, Tan-sigmoid	42	0.2052	68.60%
		65,35	Log-sigmoid, Tan-sigmoid	39	0.2117	68.60%
			Tan-sigmoid, Tan-sigmoid	32	0.2556	67.80%
		60,40	Log-sigmoid, Tan-sigmoid	36	0.2250	68.50%
			Tan-sigmoid, Tan-sigmoid	37	0.2587	68.50%
4	GDA	**70,30**	**Log-sigmoid, Tan-sigmoid**	**42**	**0.1892**	**72.20%**
			Tan-sigmoid, Tan-sigmoid	27	0.2041	71.20%
		65,35	Log-sigmoid, Tan-sigmoid	33	0.2096	70.20%
			Tan-sigmoid, Tan-sigmoid	34	0.2119	70%
		60,40	Log-sigmoid, Tan-sigmoid	32	0.2394	69.50%
			Tan-sigmoid, Tan-sigmoid	42	0.2425	69.40%
5	GDMA	**70,30**	**Log-sigmoid, Tan-sigmoid**	**43**	**0.2106**	**71.40%**
			Tan-sigmoid, Tan-sigmoid	32	0.2007	70.30%
		65,35	Log-sigmoid, Tan-sigmoid	37	0.2125	70.30%
			Tan-sigmoid, Tan-sigmoid	30	0.2557	69.40%
		60,40	Log-sigmoid, Tan-sigmoid	37	0.2146	70%
			Tan-sigmoid, Tan-sigmoid	43	0.2676	68.80%

In this section BP learning algorithms with best activation functions (Log-sigmoid, Tan-sigmoid) under different learning ratios are plotted against Accuracy in Figure 2(a) and MSE in Figure 2(b).



(a) (b)

Figure 2. BP algorithms: (a) Accuracy (b) MSE

Figure 2 shows that by reducing the number of training and consequently increasing the number of testing observations, the Accuracy of the model decreases and the MSE increases. Therefore it confirmed that for a neural network if the ratio be lower, it would be more challenging. As can be observed, among these five models LM significantly is the best and GD is the worst model. Hence, the performance of LM Backpropagation (LMBP) learning algorithm is compared with PSO.

In the following the performance of the proposed PSO for training neural network (PSO-ANN) is evaluated. The obtained results of implementation of the PSO-ANN model with different learning ratios and activation functions for optimum number of neurons in hidden layer are listed in detail in Table 5.

Table 5. Implementation results of PSO algorithm

Learning Method	Learning Ratio	Activation Function (hidden, output layers)	Optimum No. of Neurons in Hidden Layer	MSE	Accuracy
PSO	70:30	**Tan-sigmoid, Log-sigmoid**	**22**	**0.1419**	**80.10%**
		Tan-sigmoid, Tan-sigmoid	22	0.1593	78.10%
	65:35	**Tan-sigmoid, Log-sigmoid**	**27**	**0.1434**	**79.80%**
		Tan-sigmoid, Tan-sigmoid	21	0.1601	77.40%
	60:40	**Tan-sigmoid, Log-sigmoid**	**21**	**0.1480**	**79.20%**
		Tan-sigmoid, Tan-sigmoid	23	0.1582	77.50%

The PSO-ANN models achieving the lowest MSE and best Accuracy are the ones used Tan-sigmoid and Log-sigmoid as activation functions in hidden and output layers respectively. These models are highlighted in bold face for each learning ratio. Figure 4 shows the learning curves (error convergence) of PSO-ANN model under different schemes. As mentioned before, the number of generations (max iteration) is considered as 300, but in order to overcomethe problem of over-fitting in neural network

model, a validation check criterion with a value of 5 is determined. Therefore, learning algorithms are stopped upon achieving either of these criteria (achieving validation check = 5 or max iteration = 300).

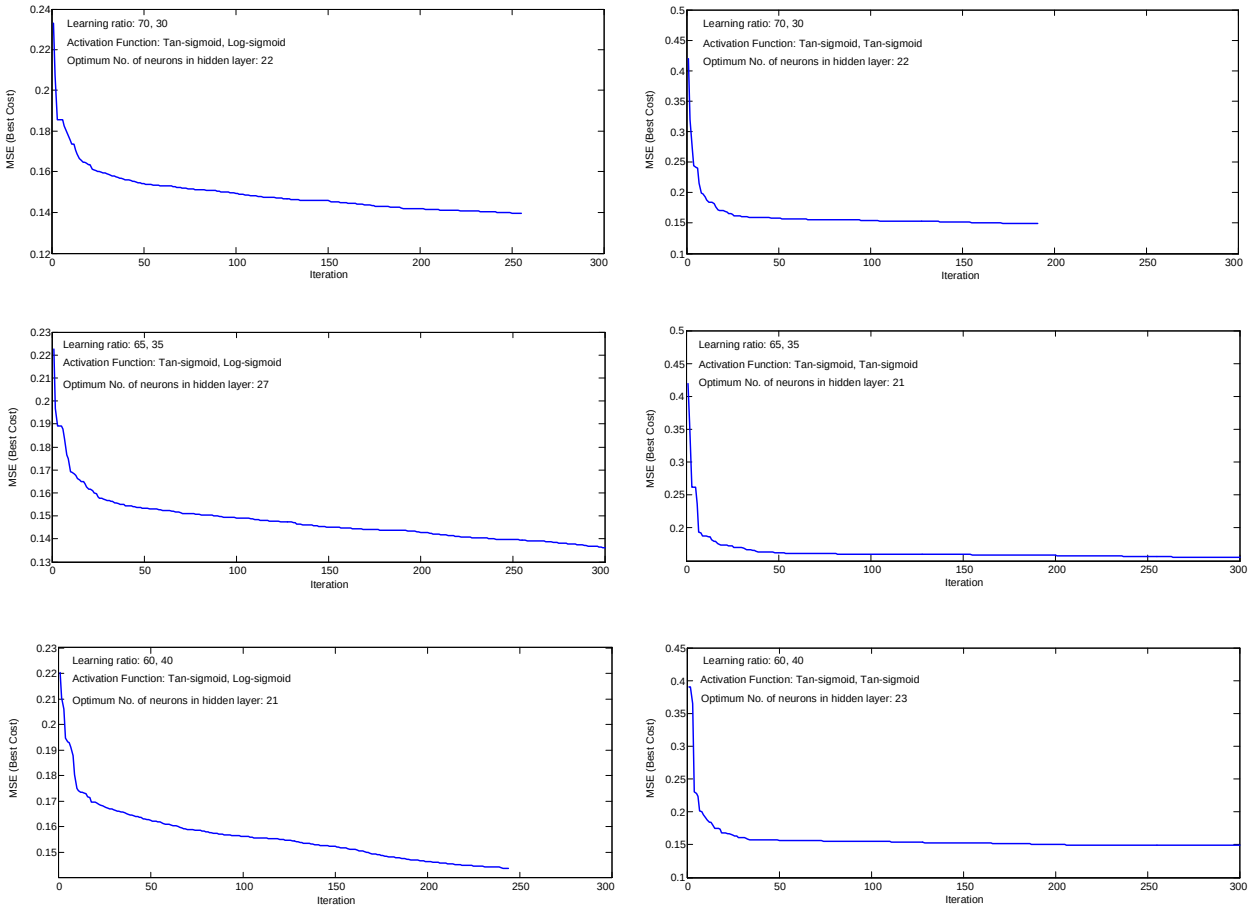


Figure 4. Learning curves (error convergence) of PSO-ANN models

From the results shown in Tables 4, 5, when the performance of PSO-ANN and LMBP-ANN models is compared, PSO-ANN can perform better (the lowest MSE and best Accuracy), meaning that PSO is very effective in training ANNs.

8. Conclusion

The use of ANNs has been shown to be an effective technique. Among the different neural networks, MLP was used for evaluating learning algorithms. It is very important to choose a proper algorithm for training a neural network, so this paper presents a comparison of various BP algorithms and PSO with different learning ratios and activation functions. BP algorithms update weights and biases in the direction of the negative gradient. ANNs training with BP algorithms is usually faced certain drawbacks such as very slow convergence and may trap in the local minimum. In contrast, weights and biases in PSO-ANN are represented by particles position. These particles velocity and position are updated, which search for personal best and global best values. This will avoid the weights and biases being trapped in local minimum. Experimental results showed that the LM algorithm achieved better performance than all other BP algorithms in training MLP neural networks. Comparing these results with PSO as a training algorithm showed the superiority of PSO.

References

[1] Y. Shang, B.W. Wah, Global Optimization for Neural Networks Training, IEEE Computer. 29 (1996) 45-54.
[2] A. Abraham, Meta learning evolutionary artificial neural networks, Neuro Computing. 56 (2004) 1-38.
[3] M.S. Mirtalaei, M. Saberi, O. K. Hussain, B. Ashjari, F. K. Hussain, A trust-based bio-inspired approach for credit lending decisions, Computing. 94 (2012) 541-577.
[4] Y.Y. Hsu, C.C. Yang, Fast voltage estimation using ANN, Electric Power System Research. 27 (1993) 1-9.
[5] T. Jain, L. Srivastava, S.N. Singh, Fast voltage contingency screening using Radial Basis Function neural network, IEEE Transactions on Power Systems. 18 (2003) 1359-1366.
[6] R. Govindaraju, A. Rao, Artificial Neural Networks in Hydrology, Kluwer Academic Publishers, Dordrecht, 2000.
[7] S.Y. Liong, W.H. Lim, G.N. Paudyal, River stage forecasting in Bangladesh: neural network approach, Journal of Computing in Civil Engineering. 14 (2000) 1-8.
[8] K.W. Chau, C.T. Cheng, Real-time prediction of water stage with artificial neural network approach, Lecture Notes in Artificial Intelligence. 2557 (2002) 715-715.
[9] S. Lahmiri, A comparative study of backpropagation algorithms in financial prediction, International Journal of Computer Science, Engineering and Applications. 1 (2011) 15-21.
[10] F. Hooshyaripor, A. Tahershamsi, Comparing the performance of Neural Networks for Predicting Peak Outflow from Breached Embankments when Back Propagation Algorithms Meet Evolutionary Algorithms, International Journal of Hydraulic Engineering. 1 (2012) 55-67.
[11] S.A. Mirjalili, S.Z.M. Hashim, H.M. Sardroudi, Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm, Applied Mathematics and Computation. 218 (2012) 11125-11137.
[12] M. Yaghini, M.M. Khoshraftar, M. Fallahi, A hybrid algorithm for artificial neural network training, Engineering Applications of Artificial Intelligence. 26 (2013) 293-301.
[13] G. Das, P.K. Patnaik, S.K. Padhy, Artificial Neural Network trained by Particle Swarm Optimization for non-linear channel equalization, Expert Systems with Applications. 41 (2013) 491-3496.
[14] C. M. Bishop, Neural networks for pattern recognition, Oxford University Press, New York, 1995.
[15] D.B. Fogel, An introduction to simulated evolutionary optimization, IEEE Transactions on Neural Networks. 5 (1994) 3-14.
[16] J.S.R. Jang, C.T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice-Hall, Upper Saddle River, 1997.
[17] M.T. Hagan, M. Menhaj, Training feedforward networks with the Marquardt algorithm, IEEE Trans Neural Networks. 5 (1994) 989-993.
[18] M.Y. El-Bakyr, Feed forward neural networks modeling for K–P interactions, Chaos, Solitons and fractals. 18 (2003) 995-1000.
[19] J. Kennedy, R. Eberhart, Particle swarm optimization, IEEE international conference on neural networks. 4 (1995) 1942-1948.
[20] R. Kiran, S.R. Jetti, G.K. Venayagamoorthy, Online training of generalized neuron with particle swarm optimization, IEEE International Joint Conference on Neural Networks. (2006) 5088-5095.
[21] N. Kwok, D. Liu, K. Tan, An empirical study on the setting of control coefficient in particle swarm optimization, Proceedings of IEEE Congress on Evolutionary Computation. (2006) 823-830.
[22] T.J. Richer, T.M. Blackwell, When is a swarm necessary?, IEEE Congress on Evolutionary Computation. (2006) 1469-1476.
[23] H.A. Abbass, R. Sarker, C. Newton, PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems, IEEE Congress on Evolutionary Computation. 2 (2001) 971-978.
[24] R.C. Eberhart, Y. Shi, Comparison between genetic algorithms and Particle Swarm Optimization, International Conference on Evolutionary Programming VII. 1447 (1998) 611-616.