# Improving the Performance of Back-Propagation Training Algorithm by Using ANN

Vishnu Pratap Singh Kirar

***Abstract*—**Artificial Neural Network (ANN) can be trained using back propagation (BP). It is the most widely used algorithm for supervised learning with multi-layered feed-forward networks. Efficient learning by the BP algorithm is required for many practical applications. The BP algorithm calculates the weight changes of artificial neural networks, and a common approach is to use a two-term algorithm consisting of a learning rate (LR) and a momentum factor (MF). The major drawbacks of the two-term BP learning algorithm are the problems of local minima and slow convergence speeds, which limit the scope for real-time applications. Recently the addition of an extra term, called a proportional factor (PF), to the two-term BP algorithm was proposed. The third increases the speed of the BP algorithm. However, the PF term also reduces the convergence of the BP algorithm, and criteria for evaluating convergence are required to facilitate the application of the three terms BP algorithm. Although these two seem to be closely related, as described later, we summarize various improvements to overcome the drawbacks. Here we compare the different methods of convergence of the new three-term BP algorithm.

***Keywords*—**Neural Network, Backpropagation, Local Minima, Fast Convergence Rate.

## I. Introduction

ARTIFICIAL Neural Network (ANN) is a model of reasoning based on the human brain with similar abilities such as associative and abstract thinking can only be achieved if it is based on architecture and working method which is similar to human brain; similar to human brain, neural network consists of a number of simple highly interconnected processors known as neurons, which are analogous to the biological neural cells of the brain. These neurons are connected by a large number of links, called weighted links. Learning is a fundamental and essential characteristic of ANN. It is capable of learning through the network experiences to improve their performance.

When ANN is exposed to a sufficient number of samples, it can generalize well to other data that they have not yet encountered. There are many algorithms to train ANN. BP is currently the most frequently and widely applied in ANN Architecture. BP has a colorful history. It was originally introduce by Bryson and Ho in 1969 [1] and then independently by Werbos in 1974 [2], by Parker in the 1980's [3]-[5] and by Rumelhart, Hinton and Williams in 1986 [6].

Generally, ANN can be trained using BP developed by Studies have shown that BP has been proven to be very successful in many diverse applications. ANN training usually carried out by iterative updating of weights based on the error signal. The negative gradient of a mean-squared error function is commonly used. In the output layer, the error signal is the difference between the desired and actual output values, multiplied by the slope of a sigmoidal activation function. Then the error signal is back propagated to the lower layers. BP is a descent algorithm, which attempts to minimize the error at each iteration. The algorithm adjusts the weights of the network such that the error is decreased along a descent direction. Traditionally, two parameters, called learning rate (LR) and momentum factor (MF), are used for controlling the weight adjustment along the descent direction and for dampening oscillations.

The BP algorithm is central to most recent work on learning in ANN. This algorithm is purely based on error correction learning technique. The very general nature of the BP training method means that a BP can be used to solve problem in many areas. In spite of these important properties, a major criticism is commonly moved against BP algorithm, like such as classification and function approximation, it often suffer from the local minima problem. And second one is its convergence rate is relatively slow, especially for networks with more than one hidden layer. To overcome this problem of local minima various methods have been proposed, such as Adaptive Learning Rate, Levenberg-Marquardt algorithm. The minimum for which the value of the error function is smallest is called global minimum, while other minima are called local minima. We have noted that many local minima difficulties are closely related to the neuron saturation in the hidden layer. Once such saturation occurs, neurons in the hidden layer will lose their sensitivity to input signals, and the propagation of information is blocked severely. In some cases, the network can no longer learn. The same phenomenon is also observed and discussed by Andreas Hadjiprocopis [7], Christian Goerick [8] and Simon Haykin [9].

Furthermore, methods have been proposed which embed chaotic dynamics into the neural network. Nozawa [10] showed the existence of chaos in Euler approximation of the Hopfield network [11] by adding a negative self-feedback connection. Chaotic simulated annealing (CSA) is proposed by Chen and Aihara [12] and uses a sufficiently large negative self-feedback to a Hopfield neural network and gradually reduces the self- feedback. Wang and Smith [13] suggested reducing the time step rather than the self-feedback in CSA. In [14], Wang et al. proposed stochastic chaotic simulated annealing, by adding a stochastic noise into CSA.

In the case of convergence speed, the reason for this is the

Vishnu Pratap Singh Kirar is with the Computer Science Department, University of Bedfordshire, Luton, United Kingdom. (Phone: +44 7405400182; +91 9826020913; e-mail: Vishnu.kirar@study.beds.ac.uk, vishnupskirar@live.com).

saturation behavior of the activation function used for the hidden and output layers. Since the output of a unit exists in the saturation area, the corresponding descent gradient takes a very small value, even if the output error is large, leading to very little progress in the weight adjustment. The selection of the LR and MF is arbitrary, because the error surface usually consists of many flat and steep regions and behaves differently from application to application. Large values of the LR and MF are helpful to accelerate learning. However, this increases the possibility of the weight search jumping over steep regions and moving out of the desired regions.

This paper presents a study of drawbacks of local minima and slow convergence rate. And we discus and studies the various approaches to overcome these drawbacks. We present the following main contributions: Section II summarizes the basic fundamental of BP. In Section III, we describe the local minima problem. Section IV describes the fast convergence of BP. In section V we give the performance evaluation. Finally conclude in last section.

## II. Theory of Back-Propagation

The back-propagation learning algorithm with multiplicative neural networks (MNN) has to be more efficient as both single-units and also in networks. The goal of learning is to update the network weights iteratively to minimize globally the difference between the actual output vector of the network and the desired output vector. The rapid computation of such a global minimum is a rather difficult task since, in general, the number of network variables is large and the corresponding non-convex multimodal objective function possesses multitudes of local minima and has broad flat regions adjoined with narrow steep ones. The basic building block of the MNN is a single neuron or node as depicted in Fig. 1 [15].

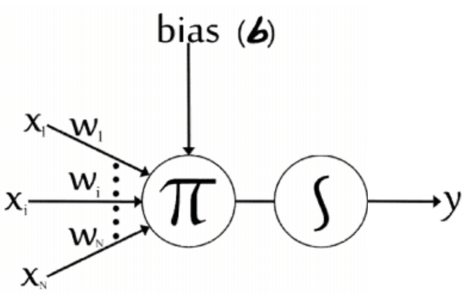

Fig. 1 Node Structure of MNN

A node receives a number of real inputs $x_1, x_2 ... x_n$, which are then multiplied by a set of weights $w_1, w_2 ... w_n$ and bias terms $b_1, b_2 ... b_n$ are added. The resultant values are multiplied to form a polynomial structure. This output of the node is further subjected to a nonlinear function $f$ defined as

$$f = \frac{1 - e^{-x}}{1 + e^{-x}}$$

The MNN are mainly employed in higher order neural Network [16]. In the MNN a number of nodes described above are arranged in layers. A multidimensional input is passed to each node of the first layer. The outputs of the first layer nodes then become inputs to the nodes in the second layer and so on. The output of the network is the output of the nodes of the final layer. Weighted connections exist from a node to every node in the succeeding node but no connections exist between nodes of the same layer. As the number of layers in the MNN is increased decision regions are formed which are considerably more complex and have highly nonlinear boundaries. Fig. 2 shows a general model of a feed forward MNN where the summation at each node is replaced by the product unit [17].

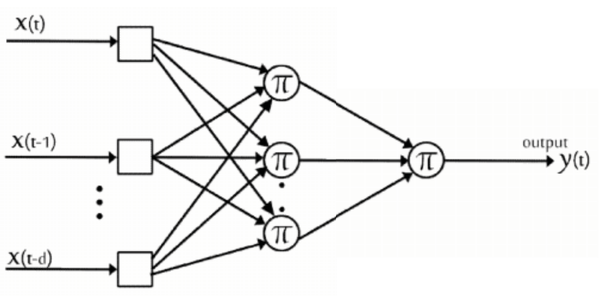

Fig. 2 Architecture of Feed Forward MNN

The MNN is trained using supervised learning where a set of input and the corresponding target vector is used to adjust the scalar parameters weight and bias. The network is trained incrementally so that the weights and biases are updated after each input is presented. The MNN is trained using supervised learning where a set of input and the corresponding target vector is used to adjust the scalar parameters weight and bias. The network is trained incrementally so that the weights and biases are updated after each input is presented. As the inputs are applied to the network, the network outputs are compared to the targets. The learning rule is then used to adjust the weights and biases of the network in order to move the network outputs closer to the targets.

The output of the node $u$ before applying activation function is given by [18].

$$u = \prod_{i=1}^{n} (w_i x_i + b_i) \tag{1}$$

The bipolar sigmoidal activation function $f$ is given by

$$y = f(u) = \frac{1 - e^{-u}}{1 + e^{-u}} \tag{2}$$

An error back propagation based learning rule is used for training. The MSE is given by

$$E = \frac{1}{2N} \left( y^p - y_d^p \right)^2 \tag{3}$$

where, $p$ is the number of input patterns.

The weight update equation for single layer algorithm is given by

$$\Delta w_i = -\eta \frac{dE}{dw_i} = -\frac{1}{2}\eta(y-d)(1+y)(1-y)\frac{u}{(w_i x_i + b_i)} x_i \tag{4}$$

where, $\eta$ is the learning rate and $d$ is the desired signal. If $\eta$ is large, learning occurs quickly, but if it is too large it may lead to instability and errors may even increase.

The bias is updated as

$$\Delta b_i = -\eta \frac{dE}{db_i} = -\frac{1}{2}\eta(y-d)(1+y)(1-y)\frac{u}{(w_i x_i + b_i)} \tag{5}$$

The standard BP algorithm calculates the new weights and biases as

$$w_i^{new} = w_i^{old} + \Delta w_i \tag{6}$$

$$b_i^{new} = b_i^{old} + \Delta b_i \tag{7}$$

Adding the momentum term and PF term further modifies the standard algorithm. The momentum term is a fraction of the previous weight change. The momentum term prevents extreme changes in the gradient due to anomalies and suppresses oscillations due to variations in the slope of the error surface [19] and prevents the network to fall into shallow local minima. The convergence still remains relatively slow because of the saturation behavior of the activation function. In the saturation area of the output activation function, the corresponding gradient descent takes very small value leading to small changes in weight adjustments. Adding a term proportional to the difference between the output and the target solves the problem of slow convergence. The improved BP weight update is calculated as

$$\Delta w_i^{improved} = \beta \Delta w_i^{old} + \Delta w_i + \gamma(y-d) \tag{8}$$

$$\Delta b_i^{improved} = \beta \Delta b_i^{old} + \Delta b_i + \gamma(y-d) \tag{9}$$

$\beta$ is the proportional term
$\Delta w_i^{old}$ is the previous weight change
$\gamma$ is the proportional term
$(y-d)$ is the difference between the output and the target at each iteration
$\Delta b_i^{old}$ is the previous bias change.

## III. Improvement in Local Minima

In this section we discuss the weight evolution algorithm for solving the local minimum problem of back-propagation by changing the weights of a multi-layer neural network in a deterministic way. During the learning phase of back-propagation, the network weights are adjusted intentionally in order to have an improvement in system performance. The idea is to work backward from the error components and the system outputs to deduce a deterministic perturbation on particular network weights for optimization purpose. From mathematical analysis, it can be found that the weight evolution between the hidden and output layers can accelerate the convergence speed, whereas the weight evolution between the input and hidden layers can assist in solving the local minima problem.

The existence of local minima is one of the difficulties in the adaptive learning of multi-layered feed forward networks. A local minimum is a suboptimal equilibrium point at which system error is non-zero and the hidden output matrix is singular [20]. Any local minima has a region of attraction such that if a descent path starts at any point inside the region, it will converge to the local minima inside the region and it will be impossible to leave the region to converge to the global minima. To solve the problem of local minima, we propose to change the hidden output matrix to a non-singular one by using evolutionary algorithms. In [21], we introduced a random perturbation to solve the problem of local minima. However, random perturbation sometimes may spend too many efforts in searching wrong descent paths before achieving the correct one and thus its computational complexity will be quite high. In [22], we proposed to use the weight evolution algorithm with deterministic perturbation to give global solution. The methodology is to select the worst input-output pairs (i.e. the worst output results with its corresponding input pattern) from the network. By considering its expected output, we then reverse the computation of the forward pass of the back-propagation learning algorithm with some approximations *to* find out their corresponding weights. We update the weights accordingly and continue the adaptation until the local minimum has been escaped. In this paper, the mathematical analysis of the weight evolution algorithm will be given. It is found that the local minima problem is related to the singularity of the hidden output matrix. A generalized scheme is developed to solve the singular hidden output matrix, which in turns will remedy the local minima problem.

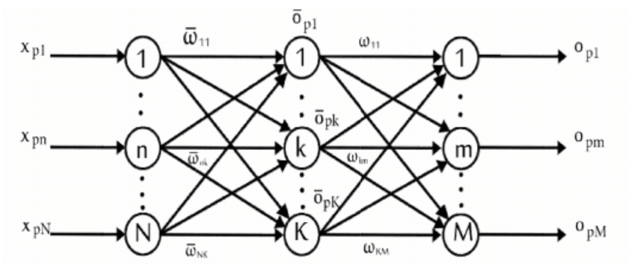

Fig. 3 Basic Structure of a two layered Feed Forward Network

Now we see how the deterministic weight evolution algorithm works. For this purpose we select the approach of S.C Ng et al. in which they proposed a network shown in Fig. 3. The network consists of *N* input nodes; *K* hidden nodes and *M* output nodes. Let $o_{pm}$ and $\bar{o}_{pk}$ *be* the output of the output node *m* and the hidden node *k* from the input pattern *p* respectively. Let $\omega_{km}$ be the network weights for the hidden node *k* and the output node *m,* and $\overline{\omega}_{nk}$ be the network weights for the input node *n* and the hidden node *k.* Assume $x_{pm}$ is the input value in the input node *n* for the input pattern *p* and $t_{pm}$ is the target output value in the output node *m* for the input pattern *p.*

Now the output of the k-th node in the hidden layer is given by:

$$\bar{o}_{pk} = f(\sum_{m=1}^{N} \bar{\omega}_{nk} x_{pm}) \quad (10)$$

where f is the sigmoidal function defined as

$$f(x) = \frac{1}{(1+e^{-x})} \quad (11)$$

Similarly, the output of the m-th node in the output layer and the sun1 of squared error of the system are:

$$\bar{o}_{pk} = f(\sum_{k=1}^{K} \omega_{km} \bar{o}_{pk}) \quad (12)$$

$$\mathrm{E} = \frac{1}{2}\sum_{p=1}^{P} \sum_{m=1}^{M} \left(t_{pm} - o_{pm}\right)^2 \quad (13)$$

To have a reduction in system error, we select an output node with the worst output to perturb deterministically. Consider a particular output neuron $m^*$, where $1 \leq m^* \leq$ Mat the output layer for which the output error is above the mean squared error value.

A number of experiments have been conducted on three different problems including the XOR problem, the 3-bit parity and the 5-bit counting problem to illustrate the performance of the weight evolution using deterministic perturbation as compared with other fast algorithms such as Quickprop [23] and RPROP [24]. For these problems, the input and the target patterns consist of 0's and l's, and the learning algorithms terminate when the system errors reach $10^{-3}$ within 30000 iterations. The initial weights are drawn at random from uniform distribution between -0.3 and 0.3.

Each experiment is performed 30 times for 30 different sets of initial weights on various learning algorithms, including standard back-propagation (BP) [25], Quickprop, RPROP and our new algorithm. Experimental results show that our new algorithm gives very promising results in terms of giving fast learning speed and global search capability as compared with other learning algorithms. From Table I, "Conv" means the average number of iterations for the algorithm to converge to system error less than the percentage of global convergence *("% of conv")* is to count the number of successful runs (over 30 different runs), which can converge to system error of less than $10^{-3}$.

### *A. The XOR Problem*

The network architecture for training the XOR problem is (2-2-1), it consists of 2 input nodes, 2 hidden nodes and 1 output node. The learning rate and momentum are assigned as 0.5 and 0.7 respectively. It can be shown from Table I that the weight evolution algorithm can always converge to global minimum with a reasonable fast manner, while other fast algorithms such as Quickprop and RPROP cannot give 100% global convergence.

### *B. The XOR Problem*

For the 3-bit parity problem, the network architecture is (3-3-l); it consists of three input nodes, three hidden nodes and one output node. The output denotes the odd parity of the three inputs, that is, the output will become '1' when there is an odd number of 1's in the input pattern. The learning rate and momentum are chosen as 0.3 and 0.7 respectively. From Table I, it is clearly reflected that the traditional learning algorithms cannot always converge while our proposed one is 100% guarantee.

### *C. The XOR Problem*

The counting problem described in [26]is particular useful to illustrate the learning of back-propagation always being trapped in local minima since there are many different local minima occurred at different error levels. The network contains *5* input units, 12 hidden units and *6* output units. Each output node corresponds to the number of 1's in the input vectors. Here the learning rate and momentum are set respectively at 0.1 and 0.7. In Table I, it is shown that our proposed one is 100% global convergence; it has a great improvement over the original back-propagation and RPROP which give only 0% global convergence. The latter two algorithms cannot converge to global minimum within 30000 iterations.

## IV. Fast Convergence Rate for BP Algorithm

Despite the learning the neural networks, several major deficiencies are still needed to be resolved. Firstly, the original back-propagation algorithm (BP) will get trapped in local minima especially for non-linearly separable problems [27] such as the XOR problem [28]. Having trapped into local minima, BP may lead to failure in finding a global optimal solution. Secondly, the convergence rate of BP is still too slow even if learning can be achieved. Furthermore, the convergence behavior of the back-propagation algorithm depends very much on the choices of initial values of connection weights and the parameters in the algorithm such as the learning rate and the momentum.

The main reason for the slow convergence of BP is due to the derivative of the activation function, which will lead to the occurrence of premature saturation [29] of the network output units. When the actual output $o_{pm}$ (where $o_{pm}$ is the actual output of the *m* output neuron for the p-th pattern) is approaching to either extreme values of the sigmoidal function, that is either 0 or 1, the derivative of the activation function having the factor $o_{pm}(1-o_{pm})$ will become extremely small, and the hack propagated error signal may vanish. Therefore, the output can be maximally wrong without producing a large error signal. The algorithm may then bc trapped into "flat spot". Consequently, the learning process and weight adjustment of the algorithm will be very slow or even suppressed. BP usually requires tens to thousands iterations to leave the flat spots, this causes the slow convergence of the algorithm.

Different approaches had been suggested to eliminate the flat spot problem so as to accelerate the convergence speed of BP [30]. Among all these methods, there are basically two approaches in solving the premature saturation (or flat spot) problem. They are the modification on either the definition of system error E, or the slope of the activation function in the weight update equation. Although many methods had been developed to solve the premature saturation of BP, these methods did not obtain very significant improvement in both the convergence speed and global convergence capability over the standard back- propagation algorithm.

In the proposed a modification on the derivative of the activation function so as to improve the convergence of the learning process by preventing the error signal drop to a very small value. The idea is to magnify the derivative term $o_{pm}$(l-$o_{pm}$),especially when the value of *Opm* approaches 0 or 1, by using a power factor. In that case, the derivative of the activation function will not be too small and the convergence of the algorithm can he improved. The new algorithm is shown to have the characteristics of faster convergence rate and greater chance to escape from flat spots as compared with BP.

To test the effectiveness of our new algorithm, its performance is compared with BP and other algorithms on some standard benchmark problems. A number of experiments have been conducted on different problems including the XOR, 3-bit parity, 5-bit counting problem, the regression problem and the character recognition problem to illustrate the performance of the MGFPROP algorithm. Let *(N, K, M)* be a network configuration with *N* input nodes, *K* hidden nodes and *M* output nodes respectively; then the network configuration of the above problems are (2-2-l), (3-2-l), (5,12,6), (l,6,l) and (64,20,26) respectively. Their learning rates are set as 0.5, 0.3, 0.1, 0.4 and 0.05 respectively. The momentum factor of all three problems is *0.7.* For the XOR problem, the output should be unity only when the two inputs are different. For the 3-bit parity problem, the outputs will he set to unity when the three inputs produce an odd parity. The 5-bit counting problem, which counts the number of 1's from *5* input units, contains many local minima and thus it is a standard experiment to illustrate the performance of a learning algorithm to avoid trapping in local minima [31].

## V. Performance Evaluation

The performance evaluations are as follow:

In Table I we evaluate performance for Local minima and in Table II we can see the performance evaluation for faster convergence.

TABLE I
PERFORMANCE EVALUATION FOR LOCAL MINIMA PROBLEM

| LEARNING ALGORITHM | XOR | | 3-BIT PARITY | | 5-BIT PARITY | |
|---|---|---|---|---|---|---|
| | conv | % of conv | conv | % of conv | conv | % of conv |
| Quick-prop | 62.9 | 56.7% | 98.2 | 93.3% | 487.1 | 66.6% |
| RPROP | 68.0 | 46.7% | 148.3 | 76.6% | >30000 | 0% |
| BP | 2376.0 | 100% | 616.6 | 100% | >30000 | 0% |
| Proposed Method | 83.4 | 100% | 198.6 | 100% | 1385.6 | 100% |

TABLE II
PERFORMANCE EVALUATION FOR FAST CONVERGENCE RATE

| Test Case | 5-Bit Counting | | Regression | | Character Recognition | |
|---|---|---|---|---|---|---|
| Algorithm | Convergence rate | % of global Convergence | Convergence rate | % of global Convergence | Convergence rate | % of global Convergence |
| BP | FAIL | 0 | 4111.6 | 46.7 | 121.1 | 100 |
| Quick-prop | 487.1 | 66.7 | 3621.7 | 76.6 | 57.0 | 3.3 |
| RPROP | FAIL | 0 | 2605.5 | 0 | 61.0 | 3.3 |
| SARPROP | 275.3 | 100 | 1293.6 | 96.7 | 54.9 | 100 |
| MGFPROP | 245.9 (S=5) | 100 | 510.2 (S=6) | 100 | 29.5 (S=3) | 100 |

## VI. Conclusion

We have studied an improved learning method for multilayer feed forward neural networks. In this method, each training pattern has its own activation functions of neurons in the hidden layer. The activation functions are adjusted by the adaptation of gain parameters during the learning process. These adjustments are made in order to prevent the network from trapping in to a local minimum caused by the neuron saturation in the hidden layer. When the network starts to approximate to the teacher signals, the gain parameters of all patterns will be adapted back to their original values. Finally, our proposed method has been indicated to be very effective in avoiding the local minima by testing it and comparing the results with those of the back propagation algorithm and the simulated annealing method on several benchmark problems.

## References

[1] Bryson, A.E., and Ho, Yu-Chi, Applied Optimal Control, Blaisdell, New York, 1969.
[2] Werbos, Paul J., Beyond Regression: "New Tools for Prediction and Analysis in the Behavioral Science," Ph.D. Thesis, Applied Mathematics, Harvard University, November, 1974.
[3] Parker, David B., "Optimal Algorithms for Adaptive Networks: Second Order Back Propagation, Second Order Direct Propagation, and second Order Hebbian Learning", Proc. 1987 IEEE International Conference on Neural Networks, II (593-600), IEEE Press, New York, 1987.
[4] Parker, David B., "A comparison of Algorithms for Neuron-Like Cells", in Denker, John (Ed.), proc. Second Annual Conference on Neural Network for Computing, Proceeding vol. 151, pp. 327-332, America Institute of Physics, New York, 1986.
[5] Parker, David B., "Learning-Logic," Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT, April 1985.
[6] D.E. Rumelhart, G.E. Hilton, R.J. Williams, "Learning Representations by Backpropagation Error," Nature 323, pp. 533-536, 1986.
[7] A. Hadjiprocopis, "Feed Forward Neural Network Entities," Ph.D. Thesis, Department of Computer Science, City University, London, UK, 2000.
[8] C. Goerick, W.V. Seelen, "On Unlearnable Problems or A Model for Premature Saturation in Back- Propagation Learning," Proceedings of the European Symposium on Artificial Neural Networks '96, Brugge, Belgium, 24–26 April 1996, pp.13–18.
[9] X.G. Wang, Z. Tang, H. Tamura, M. Ishii and W.D. Sun, "An improved Backpropagation Algorithm to Avoid Local Minima Problem," Neurocomputing, vol. 56, pp. 455-460, 2004.
[10] Faridesh Fazayeli, Lipo Wang and Wen Lui, "Back-propagation with Chaos," IEEE Int. conference Neural Network & Signal Processing, Zhenjiang, China, pp. 5-8, June 8-10, 2008.
[11] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," Proc. National Academic Science USA, vol. 81, pp. 3088-3092, May 1984.

International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:1, 2015

[12] L. N. Chen and K. Aihara, "Chaotic simulated annealing by a neural network model with transient chaos," Neural Network, vol. 8, no. 6, pp. 915-930, 1995.
[13] L. P. Wang and K. Smith, "On chaotic simulated annealing," IEEE Trans. Neural Network, vol. 9, no. 4, pp. 716-718, Jul. 1998.
[14] L. P. Wang, S. Li, F. Y. Tian, and X. J. Fu, "A noisy chaotic neural network for solving combinatorial optimization problems: Stochastic chaotic simulated annealing," IEEE Trans. System, Man, Cybern. B, Cybern, vol. 34, no. 5, pp. 2119-2125, May 2004.
[15] Kavita Burse, Manish Manoria, Vishnu P. S. Kirar, "Improved Back Propagation Algorithm to Avoid Local Minima in Multiplicative Neuron Model," World Academy of Science and Technology, vol. 72, pp. 429-432, 2010.
[16] Kavita Burse, R.N. Yadav, S.C. Srivastava, Vishnu P. S. Kirar, "A Compact Pi Network for Reducing Bit Error Rate in Dispersive FIR Channel Noise Model," International Journal of Electrical and Computer Engineering, vol. 4:11, pp.697-700, 2009.
[17] B. Mel, "Information processing in dendritic trees," Neural Computing, vol.6, pp.1031-1085, 1994.
[18] R. N. Yadav, V. Singh and P. K. Kalra, "Classification using single neuron," Proc. IEEE Int. Conf. on Industrial Informatics, pp. 124-129, 21-24 August, 2003, Banff, Alberta, Canada.
[19] C.C. Yu, and B.D. Liu, "A back propagation algorithm with adaptive learning rate and momentum coefficient," Proc. of the International Joint Conference on Neural Networks, IJCNN 2002, pp.1218-1223, May 2007.
[20] X. H. Yu, G. A, Chen, "On the local minima free condition of backpropagation learning," IEEE Transactions on Neural Networks, vol. 6, no. 5, Sept. 1995, pp. 1300-1303.
[21] S. C. Ng, S. H. Leung and A. Luk, "A Hybrid Algorithm of Weight Evolution and Generalized Back- propagationfor finding Global Minimum", Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN'99), 1999.
[22] X.H. Yu, "Can backpropagation error surface not have local minima," IEEE Transactions on Neural Networks, vol. 3, no. 6, Nov. 1992, pp. 1019-1021.
[23] M. Riedmiller, and H. Braun, "A direct adaptive method for faster back-propagation learning: The RPROP Algorithm," Proceedings of International Conference on Neural Networks, vol 1, pp. 586-591, 1993.
[24] S.E. Fahlman, "Fast learning variations on back-propagation: An empirical study," Proceedings of the 1988 Connectionist Models Summer School (Pittsburgh, 1988), D. Touretzky, G. Hinton, and T. Sejnowski, eds., pp. 38-51. Morgan Kaufmann, San Mateo, California, 1989.
[25] D. E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation, in Parallel Distributed Processing: Exploration in the Microstructure of Cognition," vol. 1. MIT Press, Cambridge, Mass, 1986.
[26] S.C. Ng, S.H. Leung, "On solving the local minima problem of adaptive learning by using deterministic weight evolution algorithm," IEEE Proceedings of the 2001 Congress on Evolutionary Computation, vol. 1, pp. 251 – 255, 2001.
[27] M. Gori, and A. Tesi, "On the problem of local minima in hack-propagation," IEEE Transactionson Pattern Analysis and Machine Intelligence,vol. 14,no. 1, pp. 7686, 1992.
[28] Yahya H. Zweiri, "Optimization of a Three-Term backpropagation Algorithm used for Neural Network Learning," International Journal of Engineering and Mathematical science, vol. 3:4, pp. 322-327, 2007.
[29] J.E. Vitela and J. Reifman, "Premature Saturation in Backpropagation Networks: Mechanism and Necessary Conditions," Neural Networks, vol. 10, no. 4, pp. 121-135, 1997.
[30] Z. Zainuddin, N. Mahat and Y. Abu Hassan, "Improving the Convergence of the backpropagation Algorithm using Local Adaptive Techniques," World Academy of Science, Engineering and Technology, vol. 1, pp. 79-82, 2005.
[31] S.C. Ng, C.C. Cheung, S.H. Leung, A. Luk, " Fast convergence for backpropagation network with magnified gradient function," IEEE Proceedings of The International Joint Conference on Neural Network 2003,vol.3, pp.1903-1908, 2003.