2011 International Conference on Telecommunication Technology and Applications
Proc .of CSIT vol.5 (2011) © (2011) IACSIT Press, Singapore

OMBP: Optic Modified BackPropagation training algorithm for fast convergence of Feedforward Neural Network

Omar Charif[1][2+], Hichem Omrani[1], and Philippe Trigano[2]

[1] CEPS/INSTEAD, Differdange, Luxembourg

[2] University of Technology of Compiegne, UTC, France

Abstract: In this paper, we propose an algorithm for a fast training and accurate prediction for Feedforward Neural Network (FNN). In this algorithm OMBP, we combine Optic Backpropagation (OBP) with the Modified Backpropagation algorithm (MBP). The weights are initialized using Yam and Chow algorithm to insure the stability of OMBP and to reduce its sensitivity against initial settings. The proposed algorithm had shown an upper hand over three different algorithms in terms of number of iterations, time needed to reach the convergence and the accuracy of the prediction. We have tested the proposed algorithm on several benchmark data and compared its results with those obtained by applying the standard BackPropagation algorithm (SBP), Least Mean Fourth algorithm (LMF), and the Optic Backpropagation (OBP). The criterions used in the comparisons are: number of iterations, and time needed for convergence, the error of prediction, and percentage of trials failed to converge.

Keywords: Artificial Neural Network(ANN), Pattern Recognition, Algorithms and Techniques.

1. Introduction:

For decades, researchers have devoted a lot of effort to understand and imitate the functionality of the human brain. This work led to the creation of the well known artificial intelligence technique Artificial Neural Network (ANN). Researchers from various scientific domains (e.g. networking, biology, medicine, social sciences, and statistic) have successfully implemented ANN models to compute all kind of functions. Neural networks are considered a nonlinear adaptive system which is able, after learning, to generalize. A well built neural network model has been considered as universal approximators capable of performing any linear and nonlinear computation. Multilayer perceptron has been applied to various problems. Researchers have used several approaches to train multilayer networks, some of them make use of artificial intelligence techniques (e.g. genetic algorithm [Montana&Davis, 1989], simulated annealing [Sexton et al., 1999], and particle swarm optimization [zhang et al., 2007]). The Backpropagation is the most used algorithm for training a multilayer network. Despite the popularity of this method, it has a lot of downsides: it is slow to converge, and it doesn't ensure the convergence to global minima. Recently, researchers have invested in improving the Backpropagation algorithm, focusing on the following areas of improvement:

The development of new advanced algorithms: Various modifications to SBP algorithm have been proposed such as Modified Backpropagation [Abid et al., 2001], LMF [Abid and Fnaiech, 2008], OBP [Otaire and Salameh, 2004]. Researchers have also developed new methods for multilayer network training. These methods and many other have achieved significant improvements in terms of the number of epochs (i.e. neural network iteration) and time needed to reach the convergence.

[+] Corresponding author. Tel.: +352 58 58 55 315; fax: +352 58.55.60
E-mail address: omar.charif@ceps.lu.

The use of adaptive learning rate and momentum term: The adaptive learning rate and momentum have been introduced to insure the fast convergence of the network as well as to avoid the local minima (e.g. [Moreira and Fiester, 1995], [Behra et al., 2006]).

The use of methods to find optimal initial weights: This approach requires some pre-treatment of the input data or at least some knowledge about the architecture of neural network model. Methods of this approach try to find optimal initial weights. The Nguyen-Widrow [1993], one of the most used methods for initializing weights, requires the number of input neurons and neurons in the hidden layer to execute the initialization process. Other methods such as Yam and Chow [2000, 2001] use the input data set into the weights initialization algorithm.

The use of advanced methods to find the optimal structure of the neural network model: Defining the topology of the neural network is the most important criteria in building a successful model able to converge and to generalize. Deciding the parameter of the neural network is not a straight forward task, several decisions should be taken e.g. the activation function to use, type of the multilayer network (e.g. fully connected or not), number of hidden layers(see table 1) and the number of neurons in each hidden layer. The number of hidden layers and the number of neurons in each one of these layers is a very critical decision to make. Narayan [1999] proposed a method "The Generalized Sigmoid Function" to avoid the use of hidden layers or to reduce the number of hidden neurons. On the other hand, the number of nodes in the hidden layers has a big effect on the outcome. Network with too few hidden nodes may suffer Underffitting and will not be able even to learn the training dataset. While a network with a big number of hidden nodes would suffer from Overffitting. Overfitting occurs when the limited amount of information contained in the training set is not enough to train all of the neurons in the hidden layers. It is also worth to mention that a large number of hidden neurons would dramatically increase the time needed for training. Several methods have been used to determine the best topology for the neural network. For instance, White and Ligomenides [1993], and Gwang-Hee [2004] have proposed Genetic Algorithm (GA) based methods to determine the best topology of the neural network models.

Table 1: Hidden layer effect

Number of hidden layer	Computation capabilities
0	Capable of approximating only linearly separable functions and in specific cases simple nonlinear functions [Narayan, 1999].
1	Can approximate any function which contains a continuous mapping from one finite space to another.
2	Represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.

In this paper, we propose a new training method, OMBP, which combines MBP algorithm, OBP, and the Yam and Chow (2001) weights initialization algorithm to train the neural network. The idea behind this combination is first to initialize the weights using Yam and chow algorithm in a way to ensure the output of neurons is in the active region (i.e. where the derivative of the active function has a large value). The neural networks with initialized weights will then be trained using OMBP algorithm.

2. Review of Standard Backpropagation

The standard Backpropagation is the most used algorithm to train Multilayer FNN. The linear and nonlinear outputs are respectively given by:

$$u_j^l = w_{j0}^l + \sum_{i=1}^{n_{l-1}} w_{ji}^l y^{l-1} \qquad (1)$$

$$f(u_j^l) = \frac{1}{1+e^{-u_j^l}} = y_j^l \qquad (2)$$

Where $l = 1 \ldots L$ denotes the number of corresponding layerS, w_{j0}^l is the threshold (or bias) of the j^{th} node in the layer l, y_i^{l-1} is the input coming from the previous layer. In equation (2), the sigmoid has been chosen as

activation function. The Backpropagation algorithm uses the gradient descent algorithm to minimize the mean squared error which is given below:

$$E_p = \sum_{p=1}^{n_p} \sum_{j=1}^{n_L} \frac{1}{2} (d_j^L - y_j^L)^2 \qquad (3)$$

Where d_j^L and y_j^L are respectively the desired and the actual outputs for the j^{th} neuron. And n_p is the number of patterns in the training data set.

The gradient descent method consists on minimizing the error by updating the weights. The training ends after reaching an acceptable error or when processing the maximum number of iterations.

The weights update is made using the equation (6), first the error signals of the output and the hidden layers should be calculated and, SUBSEQUENTLY, they are given by:

For the output neurons:

$$e_j^L = f'(u_j^L).(d_j^L \quad y_j^L) \qquad (4)$$

For the hidden layers $l = 1 \ldots L-1$:

$$e_j^l = f'(u_j^l) \sum_{r=1}^{n_{l+1}} (e_r^{l+1} w_{rj}^{l+1}) \qquad (5)$$

The weights in the hidden and output are updated using the following equation $= 1 \ldots L$:

$$w_{jk}^l(k+1) = w_{jk}^l(k) + \alpha e_j^l y_k^{l-1} \qquad (6)$$

Where α denotes the learning, generally $\alpha \in [0,1]$. The bigger is the learning rate the faster is the training. A big learning rate may produce an unstable training process

Standard Backpropagation Algorithm**:** Below is presented the standard Backpropagation training algorithm for a neural network with one hidden layer.

1) Define the network structure, assign initial weights randomly.
2) Chose a pattern to process.
3) For each node in the hidden layer
   evaluate the linear (1) and nonlinear outputs (2).
4) For each node in the output layer
   i. using the results of step3, evaluate the linear (1) and the non linear output (2)
   ii. calculate the error signals (4)
5) For each node of the hidden layer
   evaluate the error signals (5)
6) Update the weights of the hidden and output layers (6)
7) Repeat the 3 - 6 for all patterns
8) Evaluate the error of the network and evaluate the stopping criteria. If stopping criteria is not reached repeat steps 2-5.

3. The proposed algorithm

In this section, we present the algorithm we followed to achieve fast convergence while keeping very good prediction accuracy. First, we used the Yam and Chow (2001) algorithm to initialize the weights. This method determines the optimal initial weights using a multidimensional geometry. It decreases significantly the number of iterations to reach the convergence. This method showed good performance even when big training data bases are being treated. This ability of processing of big databases is very important for our future work, as we are planning to apply the developed algorithm on real data set with thousands of rows. Indeed, we will use this model to predict the transport mode choice. We will have a set of 17 explanatory variables as input and 4 different choices as output. The algorithm of the Yam and Chow (2001) method is presented below:

1) Calculate the maximum $\max(x_i)$ and minimum $\min(x_i)$ for all input variables x_i where $i = [1, \ldots, N]$.
2) Calculate the maximum distance between two points of the input space D^{in}, D^{in} is given by:

$$D^{in} = \sqrt{\sum_{i=1}^{N} [\max(x_i) - \min(x_i)]^2}$$, where N is the number of input neurons

3) Calculate the w_{max} to define the interval $[-w_{max}, w_{max}]$ from which the random weights for the hidden layer will be drawn. The w_{max} is given by: $w_{max} = \frac{0.7 s}{D^{in} \sqrt[N]{s}}$

4) Evaluate the centre of the input space C^{in} using the following equation:

$$C^{in} = \left(\frac{\max(x_1) + \min(x_1)}{2}, \frac{\max(x_2) + \min(x_2)}{2}, \ldots, \frac{\max(x_n) + \min(x_n)}{2}\right)$$

5) Calculate the threshold w_{j0} given by: $w_{j0} = -\sum_{i=1}^{n} c_i^{in} w_{ji}$

6) Calculate the v_{max} to define the interval $[-v_{max}, v_{max}]$ from which the random weights for the output layer, v_{max} is given by: $v_{max} = \frac{2.12}{H}$, where H is the number of node in the hidden layer.

7) Evaluate the threshold v_{j0} of the output layer given by: $v_{j0} = -0.5 \sum_{i=1}^{H} v_{ji}$.

The network is then trained using our proposed algorithm OMBP. In this method, two variation of the standard Backpropagation is combined, OBP[Otaire and Salameh, 2004] and the MBP algorithm [Abid et al., 2001]. The OBP presents a variant of the Backpropagation [Cichocki and Unbehauen, 1993] detailed in section 2. In this algorithm, the output layer Backpropagate the error after applying a nonlinear function on it. Thus, the algorithm of OBP is the same as SBP; just equation (4) is substituted with (7) the conditional one given by:

$$\left\{ \begin{array}{ll} e_j^L = f'(u_j^L).\left(1 + e^{(d_j^L - y_j^L)^2}\right) & if\ (d_j^L - y_j^L) \geq 0 \\ e_j^L = f'(u_j^L).\left(-\left(1 + e^{(d_j^L - y_j^L)^2}\right)\right) & if\ (d_j^L - y_j^L) < 0 \end{array} \right\} \quad (7)$$

This method is very simple to implement once you have already developed the SBP algorithm and it decreases significantly the number of iterations needed to converge.

The MBP consists of a very powerful algorithm which can increase the convergence speed of the SBP if appropriates initials settings are chosen (in specific learning rate and the range of the random weights). Beside the acceleration of the training process, the use of the Yam and Chow initialization has banished the effect of the initial weights and also increased the resilience of MBP method against the learning choice effect.

The MBP algorithm adds the concept of the linear error to the SBP algorithm. This algorithm minimizes the mean squared error given by:

$$E = \sum_{i=1}^{n_L} \frac{1}{2}(e_{1_j}^L)^2 + \sum_{i=1}^{n_L} \frac{1}{2}(e_{2_j}^L)^2 \quad (8),$$

where the $e_{1_j}^L$ and $e_{2_j}^L$ are respectively the non linear and the linear errors for the output layer. They are respectively calculated using equations (9) and (10):

$$e_{1_j}^L = d_j^L - y_j^L \quad (9)$$

$$e_{2_j}^L = Id_j^L - u_j^L \quad (10)$$

Id_j^L is the desired summation, and it is given by: $Id_j^L = f^{-1}(d_j^L)$ (11)

We have modified the equation (9) to introduce the optic Backpropagation concept. The equation (7) is used in the proposed algorithm instead of (9).

The linear and non linear errors signals are given by:

$$e_{1_j}^{L-1} = \sum_{i=1}^{n_L} f'(u_i^L) e_{1_i}^L w_{ij}^L \quad (12)$$

$$e_{2_j}^{L-1} = f'(u_j^L) \sum_{i=1}^{n_L} e_{2_i}^L w_{ij}^L \quad (13)$$

Thus, the weights update equation is given by:

$$w_{ij}^L(k+1) = w_{ij}^L(k) + f'(u_j^L)\,\mu\, e_{1_j}^L y_i^{L-1} + \mu\lambda\, e_{2_j}^L y_i^{L-1} \quad (14)$$

The proposed algorithm:

- Chose network, apply Yam and Chow to initialize weights.
- Select a pattern to process
- For each node in the hidden layer calculate the linear (1) and non linear (2) output.
- For each node in the output calculate the
  - Linear (1) and non linear (2) outputs
  - Non linear (7) and linear error (10)
- For each node in the hidden layer, calculate the linear (12) and non linear error (13)

- Update the weights using equation (14)
- Repeat the 3 - 6 for all patterns
- Evaluate the error of the network and evaluate the stopping criteria. If stopping criteria is not reached repeat steps 2-5.

4. Experiments

To demonstrate the performance of the proposed method, we tested the proposed algorithm on 4 parity problem. The results produced by the proposed method are compared to those produced using BSP, LMF, OBP, and MBP algorithms. The neural network architecture used is (4, 8, 1) with weights initialized randomly within the range [-2, 2]. The used convergence threshold is 10-4. The MBP requires in case of using Sigmoid as an activation function that the desired values must be different than 0 and 1. Thus, the output pattern 0.1 and 0.9 to present not pair and pair respectively. The table below shows the result produced by applying the different algorithms. We run each of this algorithm 100 times, then we calculated the average of the comparison criterions which are: number of epoch needed for convergence (NI), number of failure (NF), average error while prediction (AE), the time per iteration in milliseconds (TPI), NBP – the percentage of badly predicted patterns, a pattern p is badly predicted pattern iff desired_outputp – predicted_outputp > 0.2. The results are shown in table 1[1].

Table 2: Result of 4-bit parity problem

Method	NI	NF	AE	TPI	NBP
SBP	1412	1	0.1045384385336883	55	11%
SBP*	377	0	0.03036902532952025	17	1.62%
OBP	567	5	0.21857484650762002	23	25%
OBP*	294	2	0.16285951077058886	12	25%
LMF	467	8	0.13862155929278688	52	51%
LMF*	297	0	0.15367005728730704	12	50%
OMBP	**81**	**0**	**0.05388159994720001**	**5**	**4.62%**

Another test has been made on XOR problem. The architecture of neural network used this case for all methods (2, 3, 1). The results are shown in table2.

Table 3: Result of XOR experiment

Method	NI	NF	AE	TPI	NBP
SBP	337	6	0.21970360613593135	55	5.875%
SBP*	138	0	0.11015917944641399	6	3.25%
OBP	164	5	0.24053145385247257	3	48%
OBP*	35	0	0.255754310268325	2	33%
LMF	360	1	0.24378479039943973	6	96%
LMF*	135	2	0.23116313503691602	4	83%
OMBP	**64**	**0**	**0.027047435550979784**	**2**	**3.0%**

The result of these experiments proved that the proposed algorithm presents a very good compromise in terms of fast convergence, stability and prediction accuracy.

5. Conclusion

In this paper, we have proposed a new algorithm for training FNN to achieve a fast convergence and a better prediction. We have used the Yam and Chow initialization to make the proposed algorithm (OMBP) more resistant for the weight input change. The proposed algorithm gives us very good results in terms of convergence speed and prediction accuracy. As we believe that this method can be improved if an adaptive learning rate is introduced, we are planning to use Behera's adaptive learning rate to make this algorithm more resistant to the initial of the learning rate. We will then apply this algorithm on the transport system data.

[1] * signify that Yam and Chow has been used to initialize the weights.

6. Acknowledgements

This work is supported by CEPS/INSTEAD research institute, and national Research fund (FNR), Luxembourg. The authors would like to thank two anonymous reviewers for their useful and constructive comments on the earlier version of this paper. Any errors in this paper are the responsibility of the authors.

7. References

[1] Abid, A., Fnaiech., F., 2008 . Fast Training of Multilayer perceptrons with Least Mean Fourth (LMF) Algorithm. *International Journal of Soft Computing*, Vol**. 3**(5), pp. 359-367

[2] Abid, S., Fnaiech,.F., Najim., M., 2001. A Fast Feedforward Training Algorithm Using a Modified Form of The Standard Backpropagation Algorithm. *IEEE Transactions on Neural Networks*, Vol. **12**(2), pp. 424 - 430

[3] Behera, L., Kamur.,S. & Patnaik, A., 2006. On Adaptive Learning Rate That Guarantees Convergence in Feedforward networks. *IEEE Transactions on Neural Networks*, Vol. **17**(5), pp. 1116-1125

[4] Cichocki, A. and Unbehauen, R., 1993. Neural network for optimization and signal processing. *Chichester, U.K.*: Wiley, 1993

[5] Gwang-Hee, K., Jie.-Eon., Y., Sung-Hoon., A., Hun-Hee, C. & Kyung-In, K., 2004. Neural network model incorporating a genetic algorithm in estimating construction costs. *Building and Environment*, Vol. **39**(11), pp. 1333 - 1340

[6] Montana, D. & Davis, L., 1989. Training Feedforward Neural Networks Using Genetic Algorithms Kaufmann, M. (ed.). *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, pp. 762-767.

[7] Moreira, M. & Fiesler, E., 1995. Neural Networks with Adaptive Learning Rate and Momentum Terms

[8] Nguyen, D. & Widrow, B., 1990. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. *Proceedings of the International Joint Conference on Neural Networks*, Vol. **3**, pp. 21-26

[9] Otair, M. & Salameh, W., 2004. An improved back-propagation neural networks using a modified non-linear function. *Proceedings of the IASTED International Conference*, pp. 442-447

[10] Sexton, R.S., Dorsey.,R.E., Johnson, J.D., 1999. Beyond Backpropagation: Using Simulated Annealing for Training Neural Networks. *Journal of End User Computing*, Vol. **11**, pp. 3-10

[11] White, D. & Ligomenides, P., 1993. GANNet: A genetic algorithm for optimizing topology and weights in neural network design . *New Trends in Neural Computation*. Springer Berlin / Heidelberg, Vol. **686**, pp. 322-327.

[12] Yam, J.Y. & Chow, T.W., 2001. Feedforward networks training speed enhancement by optimal initialization of the synaptic coefficients. *IEEE Transactions on Neural Networks*, Vol. **12**(2), pp. 430-434

[13] Yam, J.Y. & Chow, T.W., 2000. A Weight initialization method for improving training speed in Feedforward neural network. *Neurocomputing*, Vol. **30**, pp. 219-232

[14] Zhang, Jing-Ru., Zhang., J., Lok.,T.-M.Lyu., M.R., 2007. A hybrid particle swarm optimization -back-propagation algorithm for Feedforward neural network. *Applied Mathematics and Computation*, Vol. **185**, pp. 1026-1037