Volume : 2 | Issue : 1 | Jan 2013 • ISSN No 2277 - 8160

Research Paper | Engineering

A review on back propagation algorithms for Feedforward Networks

Shital Solanki — Department of Information Technology, L. D. Collegeof Engineering, Gujarat Technological University, Ahmedabad

H.B.Jethva — Associate Professor, L. D. College of Engineering, Gujarat Technological University, Ahmedabad

ABSTRACT

The Back-propagation Neural Network (BPNN) Algorithm is widely used in solving many real time problems in world. It is highly suitable for the problems which involve large amount of data and there is no relationships found between the outputs and inputs. However BPNN possesses a problem of slow convergence and convergence to the local optimum. Over the years, many improvements and modifications of the BP learning algorithm have been reported to overcome these shortcomings. In this paper, the back propagation algorithm and several variations to improve the performance of the algorithm has been thoroughly reviewed.

KEYWORDS: Back propagation, convergence, feed forward neural networks, training, local Minima

1. Introduction

Artificial Neural Networks (ANNs) are logical methods modelled on the learning processes of human brain. Artificial Neural Networks (ANNs) works by processing information like biological neurons in the brain and consists of small processing units known as Artificial Neurons, which can be trained to perform complex calculations. As human being, we learn how to write, read, understand speech, recognize and distinguish pattern – all by learning from examples. In the same way, ANNs are trained rather than programmed.

ANN have been successfully solved many complex real world problem such as predicting future trends based on the huge historical data of an organization. ANN have been successfully implemented in all engineering fields such as biological modelling, decision and control, health and medicine, engineering and manufacturing, marketing, ocean exploration and so on [1-5].

A multilayer feed-forward neural network consists of an input layer, hidden layer and an output layer of neurons. Every node in a layer is connected to every other node in the neighbouring layer. A FFNN has no memory and the output is solely determined by the current input and weights values. A feed forward neural network consists of one or more layers of usually non-linear processing units (can use linear activation functions as well). The output of each layer serves as input to the next layer. The objective of training a NN is to produce desired output when a set of input is applied to the network The training of FNN is mainly undertaken using the back-propagation (BP) based learning.

Back-Propagation Neural Network (BPNN) algorithm is the most popular and the oldest supervised learning multilayer feed-forward neural network algorithm proposed by Rumelhart, Hinton and Williams [6].

II. BACK PROPOGATION ALGORITHM

A. Overview of the Algorithm

The traditional Back-propagation Neural Network (BPNN) Algorithm is widely used in solving many practical problems . The BPNN learns by calculating the errors of the output layer to find the errors in the hidden layers. Due to this ability of Back-Propagating, it is highly suitable for problems in which no relationship is found between the output and inputs. Due to its flexibility and learning capabilities, it has been successfully implemented in wide range of applications [7].

A Back-Propagation network consists of at least three layers of units: an input layer, at least one intermediate hidden layer, and an output layer. Typically, units are connected in a feed-forward fashion with input units fully connected to units in the hidden layer and hidden units fully connected to units in the output layer.

The input pattern is presented to the input layer of the network. These inputs are propagated through the network until they reach the output units. This forward pass produces the actual or predicted output pattern. Because back propagation is a supervised learning algorithm, the desired outputs are given as part of the training vector. The actual network outputs are subtracted from the desired outputs and an error signal is produced. This error signal is then the basis for the back propagation step, whereby the errors are passed back through the neural network by computing the contribution of each hidden processing unit and deriving the corresponding adjustment needed to produce the correct output. The connection weights are then adjusted and the neural network has just "learned" from an experience. Once the network is trained, it will provide the desired output for any of the input patterns.

B. Training in Back-Propagation Algorithm

The feed forward back-propagation network undergoes supervised training, with a finite number of pattern pairs consisting of an input pattern and a desired or target output pattern. An input pattern is presented at the input layer. The neurons here pass the pattern activations to the next layer neurons, which are in a hidden layer. The outputs of the hidden layer neurons are obtained by using perhaps a bias, and also a threshold function with the activations determined by the weights and the inputs. These hidden layer outputs become inputs to the output neurons, which process the inputs using an optional bias and a threshold function. The final output of the network is determined by the activations from the output layer. The computed pattern and the input pattern are compared, a function of this error for each component of the pattern is determined, and adjustment to weights of connections between the hidden layer and the output layer is computed. A similar computation, still based on the error in the output, is made for the connection weights between the input and hidden layers. The procedure is repeated with each pattern pair assigned for training the network. Each pass through all the training patterns is called a cycle or an epoch. The process is then repeated as many cycles as needed until the error is within a prescribed tolerance. The adjustment for the threshold value of a neuron in the output layer is obtained by multiplying the calculated error in the output at the output neuron and the learning rate parameter used in the adjustment calculation for weights at this layer. After a Back-Propagation network has learned the correct classification for a set of inputs from a training set, it can be tested on a second set of inputs to see how well it classifies untrained patterns. Thus, an important consideration in applying Back-Propagation learning is how well the network generalizes.

C. Mathematical Analysis of Algorithm

Assume a network with N inputs and M outputs. Let xi be the input to ith neuron in input layer, Bj be the output of the jth neuron before activation, yj be the output after activation, bj be the bias between input and hidden layer, bk be the bias between hidden and output layer, wij be the weight between the input and the hidden layers, and wjk be the weight between the hidden and output layers. Let η be the Learning rate, δ the error. Also, let i, j and k be the indexes of the input, hidden

and output layers respectively.

The response of each unit is computed as:

$$Bj = \sum_{i=1}^{n} Xi * Wij \quad \text{(I)}$$

$$Yj = (1/(1 + \exp(-Bj))) \quad \text{(II)}$$

Weights and bias between input and hidden layer updated as follows:

For input to hidden layer, for I = 1 to n,

$$Wij(t+1) = Wij(t) + \eta\, \delta j yi \quad \text{(III)}$$

$$bj(t+1) = bj(t) + \eta\, \delta j \quad \text{(IV)}$$

δj is the error between input and hidden ayers and calculated as follows:

$$\delta j = yj * (1 - yj) * \sum_{k} \delta k\, Wjk \quad \text{(V)}$$

Weights and bias between hidden and output layer updated as follows:

For input to hidden layer, for j = 1 to n,

$$Wjk(t+1) = Wjk(t) + \eta\, \delta k yj \quad \text{(VI)}$$

$$bk(t+1) = bk(t) + \eta\, \delta k \quad \text{(VII)}$$

δk is the error between , hidden and output layers and calculated as follows:

$$\delta k = yk * (1 - yk) * (\delta k - yk) \quad \text{(VIII)}$$

D. Difficulties With Backpropagation Learning

Backpropagtion has some problems associated with it which include network paralysis, local minima and slow convergence.

- Perhaps the best known is called "Local Minima". This occurs because the algorithm always changes the weights in such a way as to cause the error to fall. But the error might briefly have to rise as part of a more general fall, If this is the case, the algorithm will "gets stuck" (because it can't go uphill) and the error will not decrease further.
- Network paralysis occurs when the weights are adjusted to very large values during training, large weights can force most of the units to operate at extreme values, in a region where the derivative of the activation function is very small.
- A multilayer neural network require many repeated presentations of the input patterns, for which the weights need to be adjusted before the network is able to settle down into an optimal solution

E. Solutions to these learning difficulties

Approaches to improve the performance range from finding the optimal learning rate to finding the optimal network architecture. Some of the most promising approaches are:

- Adaptive learning rate and momentum factor: Rather than fixed learning rate in training, the learning rate and momentum can be adjusted dynamically during training [Weir 1990, Fausett 1994]. Decreased training time and improved convergence have been achieved using adaptive learning rate and momentum. A careful selection of the learning rate is often necessary to ensure smooth convergence. A large learning rate can cause network paralvsis and a small learning rate causes slow convergence. A momentum factor is used to smooth error oscillation. The learning rate and momentum should therefore be varied according to the region where the weight adjustment is.
- Random weight initialization: The choice of initial weight influences whether the network converges quickly or not [Fausett 1994]. The weight update between two units depends on both the derivative of the objective (error) function with respect to weights, as well as the activation value of units. weights are usually initialized randomly to small values [Rumelhart et al1986]
- Optimal network architecture selection: The achievement of good performance in a trained network is through careful selection of the network size. An oversized network can lead to over fitting of the data but on the other hand small sized (simple) network can lead to under fitting. optimal architecture selection is split into three areas: growing the network during training by adding more parameters to the network [Hirose et al 1991, Jutten et al 1995], pruning the network by removing redundant parameters during training[Sietsma et al 1988, Engelbrecht et al 1996Le cnn 1990] or regularization through penalty terms added to the objective function[Weigend et al 1991, Kamimura et al 1994, Karayiannis ef, al 1993]
- Training with jitter: Jitter is artificial noise deliberately added to inputs during training. Training with jitter is a form of regularization, such as weight decay. An advantage of jitter is that the NN can be brought out of a local minimum [Beale et al l1990] Injecting artificial noise into inputs during training is very effective in improving generalization performance when small training sets are used.
- Adaptive learning function: Activation functions can be adapted and trained just like the weights of NN. This adaptation improves learning in terms of faster convergence and more accurate results [Zurada 1992a, Engelbrecht et al1995, Fletcher et all994]. Zurada [Zurada 1992a] and Fletcher et al]
- Active learning involves making optimal use of the training data. Much research has been done in developing active learning models [Engelbrecht et al 1998, Engelbrecht et al1999a, Engelbrf'cht et oJ 1999 a] Active learning refers to the selection of a subset of the available training data dynamically during training, where the subset contains the most informative data. Active learning has been found to save computational cost and reduce training time

II. RELATED WORK DONE

Back propagation algorithm uses Gradient descent learning rule which requires careful selection of parameters such as initial weights and biases, learning rate value, activation function should be selected carefully. An improper choice of these parameters can lead to slow network convergence, network error or failure. Seeing these problems, many variations in gradient descent BPNN algorithm have been proposed by previous researchers to improve the training efficiency.

Some of the variations are the use of learning rate and momentum to speed-up the network convergence and avoid getting stuck at local minima. These two parameters are frequently used in the control of weight adjustments [8]

Back-propagation with Fixed Momentum (BPFM) shows acceleration results when the current downhill of the error function and the last change in weights are in similar directions, when the current gradient is in an opposing direction to the previous update, BPFM will cause the weight direction to be in the upward direction instead of down the slope as desired, so in that case it is necessary that the momentum-coefficient should be adjusted adaptively instead of keeping it fixed [9], [10].

Over the past few years several adaptive-momentum modifications are proposed by researchers. One such modification is Simple Adaptive Momentum (SAM) [11], proposed to further improve the convergence capability of BPNN. Sam works by scaling the momentum-coefficient according to the similarities between the changes in the weights at the current and previous iterations. SAM is found to lower computational overheads then the Conventional BPNN.

In 2009, R. J. Mitchell adjusted momentum-coefficient in a different way than SAM [11], the momentum-coefficient was adjusted by considering all the weights in the Multi-layer Perceptrons (MLP). This technique was found much better than the previously proposed SAM [12].

In 2011, M. Z. Rehman , N. M. Nawi adaptively changing the momentum for all nodes in the neural network. The GDAM has performed well for all classification problems than the previous methods.[13]

In 2005.Anil K Ahlawat,Ankit Gupta,Aniruddha gupta,Gaurav Malik,Rahul Ramchandani proposed a variant of back propagation algorithm, momentum and speed factor and gradient following has been added to the algorithm, the learning rate and momentum dynamically adusted to increase the overall efficiency of the algorithm and to increase the speed of convergence of the system.[14]

In JULY 2007, V.V.Joseph ,Rajapandian, N.Gunaseeli

Has used optimum initialization method for weight initialization, ensures that the outputs neurons are in the active region and the range of activation function is fully utilized. it has been implemented on 2 bit parity problem, 4 bit parity checker and encoder problem and produced good results [15]

In 2012,. S.P.Kosbatwar, S.K.Pathan used Back-Propagation algorithm for Pattern Association and produced good results in character recognition (IJCSES) Vol.3, No.1, February 2012 [16]

IV. Conclusions

The Back-propagation Neural Network (BPNN) is a supervised learning neural network model highly applied in different engineering fields around the globe. Although it is widely implemented in the most practical ANN applications and performed relatively well, it is suffering from a problem of slow convergence and convergence to local minima. This makes Artificial Neural Network's application very challenging when dealing with large problems. In the present paper, a deep and thorough review of back propagation algorithms for feedforward structure is carried out . From this paper we can conclude that even though several variations to this algorithm have been suggested to improve the performance of BPNN, no one guarantees global optimum solution which is still need to be answered.

REFERENCES

[1] Kosko, B.: Neural Network and Fuzzy Systems. 1sted., Prentice Hall of India (1994) | [2] Krasnopolsky, V. M. and Chevallier, F.Some Neural Network application in environmental sciences. Part II: Advancing Computational Efficiency of environmental numerical models. In: Neural Networks. (eds.), vol. 16(3-4), pp. 335--348. (2003) | [3] Coppin, B.: Artificial Intelligence Illuminated," Jones and Bartlet Illuminated Series, USA, Chapter 11, pp. 291-- 324. (2004) | [4] Basheer, I. A., Hajmeer, M.: Artificial neural networks: fundamentals, computing, design, and application." J. of Microbiological Methods, 43(1), 03--31 (2000) | [5] Zheng, He., Meng, Wu., Gong, B.: Neural Network and its Application on Machine fault Diagnosis," In: ICSYSE 1992, 17-19 September, pp. 576--579, (1992) | [6] Rumelhart, D. E., Hinton, G.E., Williams, R. J.: Learning Internal Representations by error Propagation. J. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. (1986) | [7] Lee, K., Booth, D., and Alam, P. A.: Comparison of Supervised and Unsupervised Neural Networks in Predicting Bankruptcy of Korean Firms. J. Expert Systems with Applications. 29, 1- -16 (2005) | [8] Zweiri, Y.H., Seneviratne, L D., Althoefer, K.: Stability Analysis of a Three-term Back-propagation Algorithm. J. Neural Networks. 18, 1341- -1347 (2005) | [9] Shao, H., and Zheng,H.: A New BP Algorithm with Adaptive Momentum for FNNs Training," In: GCIS 2009, Xiamen, China, pp. 16--20 (2009) | [10] Rehman, M. Z., Nawi, N.M., Ghazali, M. I.: Noise-Induced Hearing Loss (NIHL) Prediction in Humans Using a Modified Back Propagation Neural Network, In: 2nd International Conference on Science Engineering and Technology, pp. 185--189 (2011) | [11] Swanston, D. J., Bishop, J.M., and Mitchell, R. J.: Simple adaptive momentum: New algorithm for training multilayer Perceptrons. J. Electronic Letters. 30, 1498--1500 (1994) | [12] Mitchell, R. J., On Simple Adaptive Momentum. In: CIS 2008, London, United Kingdom, pp.01--06 (2008) | [13] M. Z. Rehman , N. M. Nawi " Improving the Accuracy of Gradient Descent Back Propagation Algorithm (GDAM) on Classification Problems" (IJNCAA) 1(4): 838-847ns, (2011) (ISSN: 2220-9085) | [14] Anil K Ahlawat,Ankit Gupta,Aniruddha Gupta,Gaurav Malik,Rahul Ramchandani "A Variant Of Back Propagation Algorithm For Feed Forward Network",2005 | [15] V.V.Joseph ,Rajapandian, N.Gunaseeli Modified Standard Backpropagation Algorithm With Optimum Initialization For Feedforward Neural Networks" JISE,GA,USA,ISSN:1934-9955,vol.1,no.3, july 2007 | [16] S.P.Kosbatwar, S.K.Pathan "Pattern Association for character recognition by Back-Propagation algorithm using Neural Network approach" (IJCSES) Vol.3, No.1, February 2012 [16] |