www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 6 Issue 1 Jan. 2017, Page No. 19860-19866
Index Copernicus Value (2015): 58.10, DOI: 10.18535/ijecs/v6i1.03

Training Feed forward Neural Network With Backpropogation Algorithm

RashmiAmardeep[1] and Dr.K ThippeSwamy[2]
[1]Sri Siddhartha Academy of Higher Education, Tumkur, India
[2]Visvesvaraya Technological University, PG Regional Center Mysore, India

ABSTRACT

Neural Networks (NN) are important data mining tool used for classification and clustering. NN learns by examples. NN when supplied with enough examples performs classification and even discover new trends or patterns in data. NN is composed of three layers, input, output and hidden layer. Each layer can have a number of nodes and nodes from input layer are connected to the nodes of hidden layer. Nodes from hidden layer are connected to the nodes of the output layer. Those connections represent weights between nodes.

This paper describes popular Back Propagation (BP) Algorithm is proposed for feed forward NN algorithms. The Back-propagation (BP) training algorithm is a renowned representative of all iterative gradient descent algorithms used for supervised learning in neural networks. The aim is to show the logic behind this algorithm. In BP algorithm, the output of NN is evaluated against desired output. If results are not satisfactory, weights between layers are modified and the process is repeated again and again until an error is minimized. BP example is demonstrated in this paper with NN 2-2-1 architecture considering momentum. It is shown that most commonly used back- propagation learning algorithms are special cases of the developed general algorithm. The Sigmoid activation function approach is used to analyze the convergence of weights, with the use of the algorithm, toward minima of the error function.

Keywords: Neural Networks, Artificial Neural Networks, Training, Back Propagation algorithm

INTRODUCTION

NN is one of the classification techniques used in data mining. Learning techniques in NN are of two types: they are supervised where output values are known beforehand (back propagation algorithm) and unsupervised where output values are not known (clustering).
Neural Networks are so widely studied by Computer Scientists, Electronic Engineers, Biologists and Psychologists, they are referred to as *Artificial Neural Networks (ANNs)*, *Connectionism* or *Connectionist Models*, *Multi-layer Perceptron's (MLPs)* and *Parallel Distributed Processing (PDP)*.There are a group of "classic" networks which are widely used and on which many others are based. These are Back Propagation, Hopfield Networks, Competitive Networks and networks using Spiky Neurons [3][4].
Feed forward neural networks (FNN) have been widely used for various tasks, such as pattern recognition, function approximation, dynamical modeling, data mining, and time series forecasting, to name just a few [1], [2].
The paper describes NN architecture, a number of nodes to choose, training the network and evaluating the results, advantages and disadvantages are proposed. Activation function gets mentioned together with learning rate and momentum. The most popular NN algorithm in Back propagation is demonstrated.

DOI: 10.18535/ijecs/v6i1.03

BACK PROPAGATION

Back Propagation network is considered to be quintessential Neural Network. Back Propagation is the training or learning algorithm rather than the network itself. To train the network we need to give the output called the *Target* for a particular input. The input and its corresponding target are called a *Training Pair*. Once the network is trained, it will provide the desired output for any of the input patterns.
The network is first initialized by setting up all its weights to be small random numbers – say between –1 and +1. Next, the input pattern is applied and the output is calculated this is called the *forward pass*. The calculation gives an output which is completely different to what is expected (the Target), since all the weights are random. We then calculate the *Error* of each neuron, which is essentially: *Target - Actual Output*. This error is then used mathematically to change the weights in such a way that the error will get smaller. In other words, the Output of each neuron will get closer to its Target (this part is called the *reverse pass*). The process is repeated again and again until the error is minimal.

Network size

The most commonly used networks consist of an input layer, a single hidden layer and an output layer as shown in fig 1.The input layer size is set by the type of pattern or input to the network for the process. The size of the output layer is set by the number of patterns to recognize. The number of neurons in hidden layer neurons should not be below min number of input neurons as the network hasn't got enough weights to store all the patterns, above output number of neurons the network becomes inefficient and doesn't perform as well. Thus, the number of hidden layer neurons needs to be experimented with for the best results.

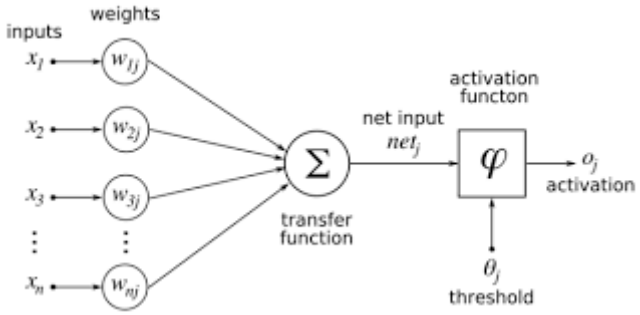

Fig 1 Neural Network architecture

The neurons have a sigmoid activation function. The network keeps training all the patterns repeatedly until the total error falls to some pre-determined low target value and then it stops.
When the network has fully trained, the Validation Set error reaches a minimum. When the network is overtraining (becoming too accurate) the validation set error starts rising. Fig 2 shows this idea.

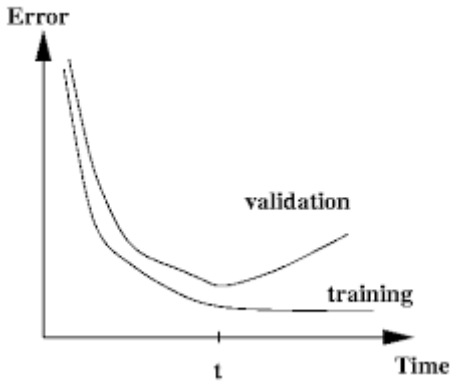

Fig 2 Plot of the error on training set and validation set.

DOI: 10.18535/ijecs/v6i1.03

Algorithm

Step1: Determine the architecture
Number of input and output neurons
Hidden neurons and layers

Step 2: Initialize all weights and biases to small random values, typically Є [-1, 1], choose a learning rate η .
Step 3: Repeat until termination criteria satisfied
Present a training example and propagate it through the network (Forward pass)
Calculate the actual output
- Inputs applied
- Multiplied by weights
- Summed
- squashed by sigmoid activation function
- Output passed to each neuron in next layer

Adapt weights starting from the output layer and working backwards (backward pass)

Pseucode
Input: Data Set D,Learning rate l network
Output: Trained neural Network

Initialize all weights and biases in network;
While terminating condition is not satisfied {
for each training tuple X in D {
// Propagate the input forward:
for each input layer unit j {
$O_j = I_j$; //output of an input unit is its actual input value
for each hidden or output layer unit j {
$I_j = \sum_i w_{ij} O_i + \Theta_j$;//compute the net input of unit j w.r.t toprevious layer, i
$O_j = 1/ (1 + e^{-t j})$; } //compute the output of each unit j
//Backpropagate the errors:
for each unit j in the output layer
$Err_j = O_j (1- O_j)(T_j – O_j)$;// compute the error
for each unit j in the hidden layesr, from the last to the first hidden layer
$Err_j = O_j (1- O_j) \sum_k Err_k w_{jk}$; //compute the error w.r.t next higher layer , k
for each weight w_{ij} in nework {
$\Delta w_{ij} = (l) Err_j O_i$; //weight increment
$w_{ij} = w_{ij} + \Delta w_{ij}$ } //weight update
for each bias Θ_j in network {
$\Delta \Theta_j = (l) Err_j$; //bias increment
$\Theta_j = \Theta_j + \Delta \Theta_j$;} //bias update
}}

DOI: 10.18535/ijecs/v6i1.03

Worked Example

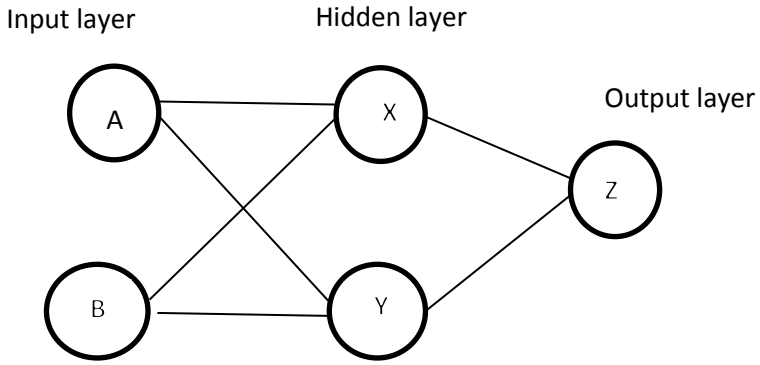

Fig 3: Neural Network (2-2-1)

Input		Output
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

Pattern data for OR

β=learning rate =0.35

α= Momentum =0.9

NN of fig. 3 has two nodes (A,B) in the input layer, two nodes in the hidden layer (X,Y)and one node in the output layer (Z). The values given to weights are taken randomly and will be changed during BP iterations. Initial weights of the top input nodes taken at random are 0.4,0.1 .Weights of bottom input node are 0.8 and 0.6.Weights of top hidden node is 0.3 and that of bottom hidden node is 0.9.

Learning rate of 0.35 proved to be popular choice when training Neural Network. Large value of momentum influences the adjustment in current weight to move in same direction as previous adjustment. Sigmoid function f(x) =1.0/ (1.0+exp (-x)) is used. Computation is solved for the last row where the input is 1 for both nodes and the expected output is 1.

Step 1: Feed forward computation:

Node x = f(x) =f (1*0.4 + 1* 0.8) =0.7685

Node y =f(y) =f (1*0.1 + 1*0.6) =0.6682

Node z =f (z) =f (0.3*0.7686 +0.9*0.6682) = 0.6982

Step 2: Reverse Pass (Target =1)

Error Calculation:

Z_{error} = output *(Target – output)*(1-output)

Z_{error} = 0.6982*(1-0.6982)*(1-0.6982) =0.0635

$\Delta W_{xz} = \beta * Z_{error} * f(x)$ = 0.35*0.0635*0.7686 = 0.0170

$\Delta W_{yz} = \beta * Z_{error} * f(y)$ = 0.35*0.0635*0.6682 = 0.0148

New weights from hidden node:

$W_{xz,new} = W_{xz} + \Delta W_{xz} + \alpha * \Delta(t-1)$ = 0.3 +0.0170+0.9 *0 = 0.3170

$W_{yz,new} = W_{yz} + \Delta W_{yz} + \alpha * \Delta(t-1) = 0.9 + 0.0148 + 0.9 * 0 = 0.9148$

The value of Δ (t-1) is previous delta change of the weight. Since there is no delta change so it is zero.

Error calculation for input node:

$X_{error} = Z_{error} * W_{xz,new} = 0.0635 * 0.3170 = 0.0201$

$Y_{error} = Z_{error} * W_{yz,new} = 0.0635 * 0.9148 = 0.0580$

$\Delta W_{Ax} = \beta * x_{error} * A = 0.35 * 0.0201 * 1 = 0.007035 = \Delta W_{Bx}$

$\Delta W_{Ay} = \beta * y_{error} * A = 0.35 * 0.0580 * 1 = 0.0203 = \Delta W_{By}$

New weights from Input node:

$W_{Ax,new} = W_{Ax} + \Delta W_{Ax} + \alpha * \Delta(t-1) = 0.4 + 0.007035 + 0 = 0.40735$

$W_{Bx,new} = W_{Ax} + \Delta W_{Bx} + \alpha * \Delta(t-1) = 0.8 + 0.007035 + 0 = 0.807035$

Similarly $W_{Ay,new} = 0.1203$ and $W_{By,new} = 0.6203$

Step 3: Weight Updates in second pass

Node x: f(x) =f (0.407 + 0.807) =0.771

Node y: f(y) =f (0.6203 + 0.1203) = 0.850

Node z: f (z) =f (0.3170 * 0.771 + 0.9148 * 0.8050) =0.735

Error Calculation:

$Z_{error} = 0.735 * (1-0.735) * (1-0.735) = 0.0516.$

Therefore after initial condition, calculated error was 0.0635 and in II pass the error has reduced to 0.0516.Hence the algorithm has improved. This gives a fair idea of working of BP Algorithm.

Problem with Back Propagation

BackPropagtion has some problems associated with it. 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, as shown in Fig 4. If this is the case, the algorithm “gets stuck” (because it can’t go uphill) and the error will not decrease further.

DOI: 10.18535/ijecs/v6i1.03

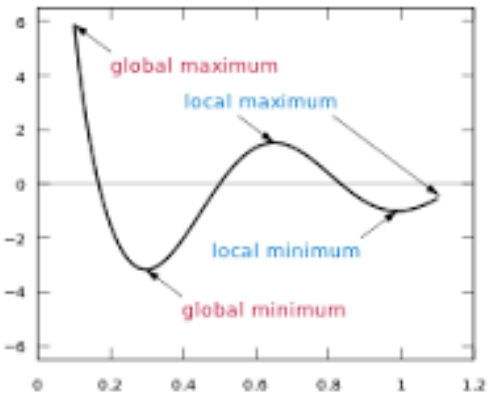

Fig 4 local minima

There are several solutions to this problem. One is very simple and that is to reset the weights to different random numbers and try training again. Another solution is to add "momentum" to the weight change. This means that the weight change and iteration depends not just on the current error, but also on previous changes.

Advantages

Back propagation (BP) is a well-known network that has been known for its accuracy because it allows itself to learn and improving itself thus it can achieve higher accuracy [6].
Multilayer Perceptron network can be trained by back propagation algorithm to perform any mapping between the input and the output.

Disadvantages

It requires labeled training data.- Almost all data is unlabeled

The learning time does not scale well. It is very slow in networks with multiple hidden layers.

It can get stuck in poor local optima. When faced against the larger datasets back propagation still stuck into local minima problem but researcher like Bumghi Choi et al.[5] has shown one of the novel ideas and proposed the algorithm to avoid the local minima problem in complex problems and shown that there is still scope to fasten the BP Algorithm for larger and complex problems.

CONCLUSION

A general FNN training back propagation algorithm which is a supervised learning method to train the ANN is proposed. BP Algorithm is known for its mathematical simplicity and accuracy
Its error minimization has been completely analyzed using the sine function theory. However, it should be emphasized that the strength of the general learning algorithm lies in its ability to handle time-varying inputs.
It is observed that introducing the momentum the error has been reduced.
BP has its own limitations of slow convergence rate and local minima problem which is still a big problem when dealing with large complex problems. There has been significant research done to overcome these problems and different variations of BPA has been proposed.

REFERENCES

[1] J. M. Zurada, *Introduction to Artificial Neural Systems*. St. Paul, MN: West, 1992.
[2] P. Mehra and B. W. Wah, *Artificial Neural Networks: Concepts and Theory*: IEEE Comput. Society Press, 1992.
[3] X. Liao and J.Yu, "Robust stability for interval Hopfield neural networks with time delay," *IEEE Trans. Neural Networks*, vol. 9, pp. 1042–1045.
[4] J. J. Hopfield, "Neurons with graded response have collective computational properties like these of two-state neurons," in *Proc. Nat. Academy Sci. USA*, 1984, pp. 3088–3092.
[5] Bumghi Choi,Ju-Hong Lee,Deok-Hwan Kim, "Solving local minima problem with large number of hidden nodes on two-layer feed forward artificial neural networks Neurocomputing"(2008).
[6] Yeremia, Hendy, et al. "Genetic Algorithm And Neural Network For Optical Character Recognition." Journal of Computer Science 9.11 (2013): 1435.
[7] Guoqiang Peter Zhang "Neural Networks for Classification: A Survey " IEEE transactions on systems, man, and cybernetics—part c: applications and reviews, vol. 30, no. 4, November 2000
[8] E. Barnard and E. C. Botha, "Back-propagation uses prior information efficiently," IEEE Trans. Neural Networks, vol. 4, pp. 794–802, 1993.
[9] P. Mehra and B. W. Wah, Artificial Neural Networks: Concepts and Theory: IEEE Comput. Society Press, 1992.
[10] Y. Zhao, "On-line neural network learning algorithm with exponential convergence rate," Electron. Lett., vol. 32, no. 15, pp. 1381–1382, July 1996.
[11] D. E. Rumelhart, B.Widrow, and M. A. Lehr, "The basic ideas in neural networks," Commun. ACM, vol. 37, pp. 87–92, 1994.
[12] R. P. Lippmann, "An introduction to computing with neural networks," IEEE Acoust. Speech Signal Process. Mag., vol. 4, no. 2, pp. 4–22, Apr. 1987.