Backpropagation Networks and Multilayer Perceptron

Saroj Singh

Abstract: Backpropagation(BP) or generalized delta learning rule has been discussed. The algorithm uses least mean square error minimization strategy. Modifications in the weight are taken along the –ive gradient direction to reduce the error. The activation function used in the BP is sigmoidal function. The limitation of BP is that it is prone to local minima just like gradient descent formula. Backpropagation network is Multilayer Feed-forward network with different network transfer function in Artificial Neural Network and a more powerful learning rule. The learning rule is called Backpropagation which is a kind of gradient descent technique with backward error propagation. It is a supervised learning method and is an implementation of the Delta learning rule. It requires a teacher that knows or can calculate the desired output for any given input.

Index Term-Keywords: weights; perceptron; gradient; threshold; activation; enhancing; radial basis;

1. Introduction

Backpropagation is a systematic method of training multilayered artificial neural networks. It is built on high mathematical foundation and has very good application potential. Rumelhard, Hiklton and *Williams* (1986)[1] presented a clear and concise description of the backpropagation algorithm. Parker (1982) has also shown to have anticipated Rumelhart's work. An Artificial Neural Network (ANN) Architecture the multilayer Feedforward (MLFF) with Backpropagation (BP) learning. This type of network is sometimes called multilayer perception because of its similarity to perceptron networks with more than one layer. First we review the perceptron model to show how this is altered to form MLFF networks. The learning rule is called Backpropagation which is a kind of gradient[2] descent technique with backward error propagation

2. Perceptron

The perceptron is a program with true(1) or false(0) for inputs we present to it by repeatedly "studying".
Single layer prceptron:
A single layer perceptron consist of an input & an output layer. The perceptron function applied as hard-limiting function. An output unit will assume the value 1 if the sum of weight inputs is greater than its threshold i.e.
Σ Wji Xi >□
Where
Wji is weight from unit j to j
Xi is the input from i
◎j is the threshold on unit j
Let there are two classes A and B. if Σ Wji Xi >□then object will be classified as class A otherwise B. Suppose there are n inputs then the equation
Σ Wji Xi =□
Where i=1, 2, ……. n forms a hyper plane dividing the space in two halves.

Perceptron Algorithm:
1. Weight initialization
Set all the weights &mode thresholds to small random numbers.
2. Calculate of Activation
(a) The activation level of an input unit is determined by instance presented to the networks.
(b) The activation level Oj of an output unit i determined by
Oj=Fh(Wji Xi - ◎j)
Where Fh is hard-limiting function
Fh(a) = {1 1 if a>0, 0 false
3. Weigh training
(a) Adjust weights by
Wji (t+1)=Wji (t) + ΔWji
Where Wji(t) is the weigh from unit i to unit j at time t and ΔWji is the weigh adjustment.
(b) The weigh change may be computed by Delta learning rule
ΔWji = ηδjXi
Where
Δ = trial independent learning rate (0<η<1)
δj = the error at unit j and δj = Tj-Oj
Where Tj is the desired output activation and Oj is the actual output activation at output unit j.
(c) Repeat the iterations till convergence.
Example:
In a single layer perceptron unit 1 receives input from unit 2 and 3 given that:
W1,2=(-3), W1,3=(2), x2,x3=1, ◎1=1.
Calculate O1.
Suppose the desired output T1=1. How we adjust the

weights consider η=0.3.
According to activation level Oj
Oj=Fh(Wji Xi - ◎j) Wji=W1,2 and W1,3
O1= Fh (((-3)*1+2*1)-1)
= Fh (-2)
= 0
δj = 1-0=1
according to ΔWji = ηδjXi
ΔW1,2 = ηδjXi=((0.3)*1*1)=0.3
ΔW1,3 = ηδjXi=((0.3)*1*1)=0.3
W1,2 = W1,2(odd) + ΔW1,2
W1,3 = W1,3(odd) + ΔW1,3
= 0.3+2
The threshold is always negative of weight from bias unit let W1,b is the bias unit that is
W1,b=(- ◎1)= -1
ΔW1,b = 0.3*1*1=0.3
W1,b = -1+0.3 = 0.7
Thus the threshold is changed to 0.7

3. Multilayer perceptron

A multilayer perceptron is a network of simple neuron called perceptron. The basic concept of single perceptron was introduced by Rosenblatt in 1958. A multilayer perceptron is a feed-forward neural network with at least one hidden layer. It can deal with nonlinear classification problems.

Example

Suppose an output node receives 2 inputs its threshold is set to 1.5 and its input weights are both set 1. Threshold when both the input unit are active (i.e. 1) the output node will be active. In this case the output node performs a logical AND operation on the threshold is set to 0.5 then any active input can activate the node. In this case the output node performs a logical OR operation. This by choosing a different set of weights are threshold, a node can implement a different logical operation.

4. Architecture of a Backpropagation Network

Rosenblatt's perceptron was introduced and its limitation with regard to the solution of linearly inseparable or non-linearly separable problems was discussed[3].
The initial approach to solve such linearly inseparable problems was to have more than one perceptron each set up of the inputs then combining their outputs into another perceptron would produced a final indication of the class to which the input belongs. Let us take the example of XOR problems discuss the following diagram.

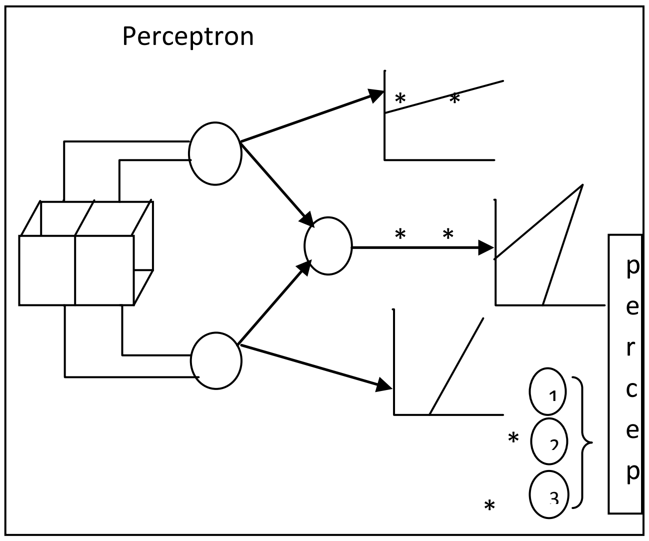
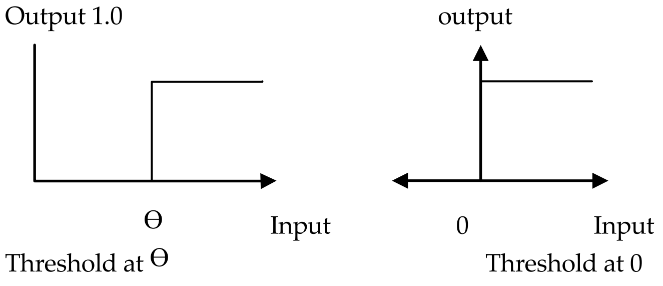


Figure 1 Combining perceptron to solve XOR problem

The combining of perceptrons to solve the XOR problem can solve problem.
In the structure takes the weighted sum of thresholds it and outputs one(1) or zero (0). For the perceptron the first layer, the input comes from the actual input of the problem, while for the perceptron in the second layer the inputs are outputs of the first layer. The perceptron of the second layer do not know which of the real inputs from first layer on or off.
It is impossible to strengthen the connections between active inputs and strengthen the correct part of the networks. The actual inputs are effectively marked off from the outputs units by intermediate layer. The two state of neuron being on or off shown in figure 2 do not give us any indication of the scale by which we have to adjust the weights. The hard hitting threshold functions remove the information that is needed if the network is to successfully learn. The network is unable to determine which of the input weight should be increased & which one and so, it is unable to work, to produce a better solution next time.



"Step" or Heaviside function
Figure 2 Hard hitting threshold function

The way to go around the difficulty using the step function as the threshold process [4] is to adjust it slightly & and to use a slightly different nonlinearity.

5. Multilayer Perceptron Characteristics

Multilayer perceptron have the following characteristics such as:

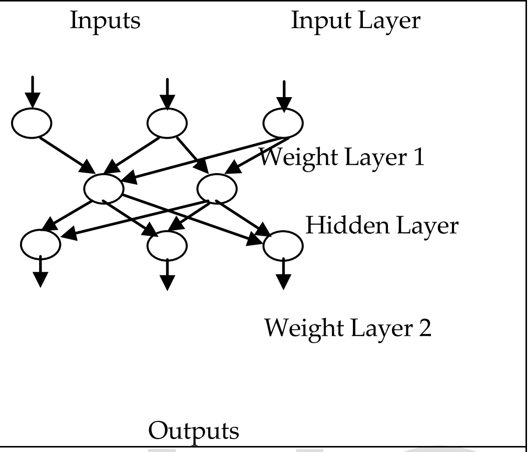
1.	Sample Structure	 Inputs Input Layer Weight Layer 1 Hidden Layer Weight Layer 2 Outputs
2.	Type	Feed-forward
3.	Neuron Layer	1 Input Layer 1 or More Hidden Layer 1 Output Layer
4.	Input Value Type	Binary
5.	Activation Function	Hard – Limiter (Hitting) Sigmoid
6.	Learning Method	Supervised
7.	Learning Algorithm	Delta Learning Back propagation (mostly used)

Table 1: Characteristics of multilayer perceptron

Learning methods:
The neural network has also been called the "connectionist". It consists a large number of neuron like processing elements and a large number of weighted connections between the elements. The weights on the connections encode the knowledge of network. It uses a highly parallel, distributed control and can learn to adjust itself automatically. The various algorithms are described below:
Back propagation (Generalized Delta learning rule)
Radial basis function network
Reinforcement learning
Art network

Back propagation (Generalized Delta learning rule)

It was first described by Paul Werbos in 1974, but it was not until 1986, through the work of David E. Rumelhart, Geoffrey E. Hinton and renold J. William, that it gained recognition.

Backpropagation network is multilayered feedforward network with different transfer function in ANN and a more powerful learning rule. The learning rule is called "Backpropagation", which is kind of gradient descent technique with backward error propagation.
It is a supervised learning method and is an implementation of the delta rule. It requires a teacher that knows, or can be calculate, the desired output for any given input.

Radial basis function network

A Radial basis function (RBF) network is a 2 layer network whose output unit forms a linear combination of the basis functions computed by hidden units. And each hidden unit implements s a radial activated function. The output unit implements a weighted sum of hidden output units. The input to RBF network is nonlinear while the output is linear. Due to their nonlinear properties, RBF networks are able to model complex mappings. The most common basis function can be activation function is the hidden layer. The most common basis function chose is Gaussian function, in which case the activation level Oj of the hidden unit j is calculated as
Oj = exp[-(X - Wj)* (X-Wj)/2sj2]
Where X is the input vector Wj is the weight vector associate with hidden unit j and sj2 is the normalization factor. The output of the hidden unit lie between 0 and 1; the closer the input is center of the Gaussian, the lager the response of the node. The activation level of an output unit is determined by
Oj = ӘWjiOi
The output units from a linear combination of nonlinear basis functions, and thus the overall network performs a nonlinear transformation of the input.

Reinforcement learning

This method of learning[5] is mixture of the previous two types. The network is presented with the input, but only told that that the answer is wrong or right. It is expected to organize in some manner to ensure a correct answer is more likely on the next presented of the input.

6. Art Network

Adaptive Resonance Theory(ART) networks are most useful for pattern clustering, classification and recognition. They can also perform pattern association with some modification. These networks work on binary or analog –value input. Their ability to generalize is limited because ART networks can lack the hidden layer of neurons which perform feature recognition or pattern recognition or pattern recognition in the backpropagation network.
Training: There are two basic methods of training ART[6] –based neural networks: slow and fast. In the slow learning method, the degree of training of the recognition neuron's weights towards the input vector is calculated to continuous values with differential equation s and thus dependent on the length of time the input vector is presented. With fast learning, algebric equations are used to calculate degree of weight adjustments to be made,and banary values are used. While fast learning is effective and efficient for a varity of tasks, the slow learnig method is more biologically possible and can be used with continous – time networks.

ART1
ART 1 is the simplest variety of ART networks, accepting only binary inputs.

ART 2
ART 2 extends network capabilities to support continous inputs.

ART 2-A
ART 2-A is a streamlined form of ART 2 with a drastically accelerated runtime and with qualitative results being only rarely inferior to the full ART 2 implementation.
ART 3
ART 3 builds on ART 2 by simulating rudimentary neurotransmitter regulation of synaptic activity by incorporating simulated sodium (Na+) and calcium (Ca2+) ion the system's equations, which results in a more physiologically realistic means of partially inhabiting categories that trigger mismatch resets.
Fuzzy ART
Fuzzy Art implements fuzzy logic into ART's pattern recognition, thus enhancing generalizability. On optional and very useful feature of fuzzy ART is complement coding, a means of incorporating the absence of features into pattern classifications, which goes a long way towards preventing efficient and unnecessary category proliferation.
ARTMAP
ARTMAP also known as Predictive ART, combines two slightly modified ART-1 or ART-2 units into a suprtvised learning structure where the first unit takes the input data and the second takes the correct output data, then used to make the minimum possible adjustment of the vigilance parameter in the first unit in order to make the correct classification.
Fuzzy ARTMAP
Fuzzy ARTMAP is merely ARTMAP using fuzzy ART units, resulting in a corresponding increase in efficiency.

Model for Multilayer perceptron
The adapted perceptrons are arranged in layers and so the model is termed as multilayer perceptron. This model has three layers:
An input layer
Hidden layer
Output layer

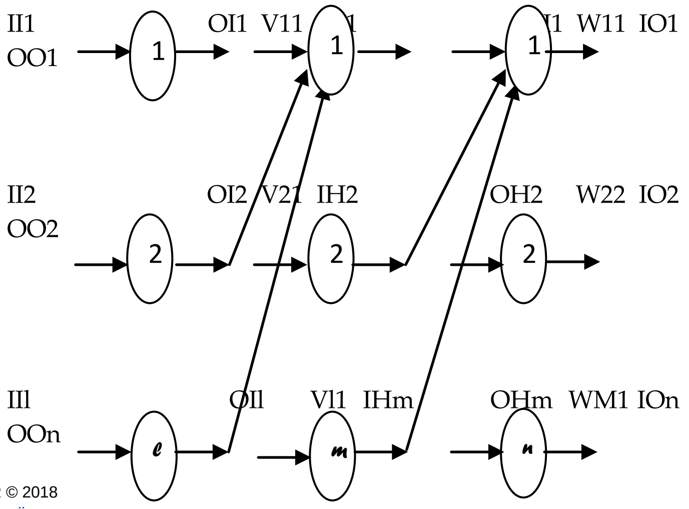
An Input Layer

The nodes in this layer are called input units which encodes the instance present to the network for processing. For example each input unit may be designated by an attribute value possessed by instance.

Hidden layer
The nodes in this layer are called hidden units, which are not directly observable and hence hidden. They provide the nonlinearities for the network.
Output layer
The nodes in this layer are called output nodes which encodes possible concepts or values to be assigned to the instance under consideration for example each output unit presents a class of objects.

7. Back Propagation Learning Methods

Consider the network as shown in figure 3 where the subscript I-input, H-hidden, O-output denoted neurons[7].



Input Layer Hidden Layer Output Layer
L nodes m nodes n nodes

Figure 3: Multilayer feed-forward back propagation network

Consider a problem in which an "nset" of "l" inputs and the corresponding "nset" of "n" output data is given as shown in table:

No. of set	Input	Output
	I1 I2 -------I l	O1 O2 -------On
1	0.3 0.4 ----- 08	0.1 0.56 ------- 0.82
2		
-		
-		
n set		

Table 2: Problem of nset

Input layer computation:
Consider linear activation function the output of the input layer is input of input layer (considering g=tanØ=1). Taking one set of data.
{O}I = {I}I
l x1 l x1
the hidden neurons are connected by synapses to input neurons and (denote) Vij is the weight of the arc between ith input neuron to jth hidden neuron as shown in equation (1).

IHP=VIPOI1+V2POI2 + ---------VIPOIl (1)
Where (P=1,2,3,--------m)

The input to the hidden neuronis the weighted sum of outputs of the input neurons to get IHP (i.e the input to the Pth hidden neuron). Denoting weight matrix or connectivity matrix between input neurons and hidden neurons as between [V]
lxm
We can get an input to the hidden neurons as
{I}H = [V]T {O}I
Mx1 mxl lx1

Hidden Layer Computation:
Considering sigmoidal function[8] or squashed – S function, the output of the Pth hidden neuron is given by

$$OHP = \frac{1}{((1+e^{-\lambda(IHP\_nP)})} \quad (2)$$

Where
OHP = output of the pth hidden neuron
IHP = input of the pth hidden neuron
⊚HP = threshold of the pth neuron
A non –zero threshold neuron is computationally equivalent to an input that is always held at –1 and the non-zero threshold becomes the connecting weight values shown in figure

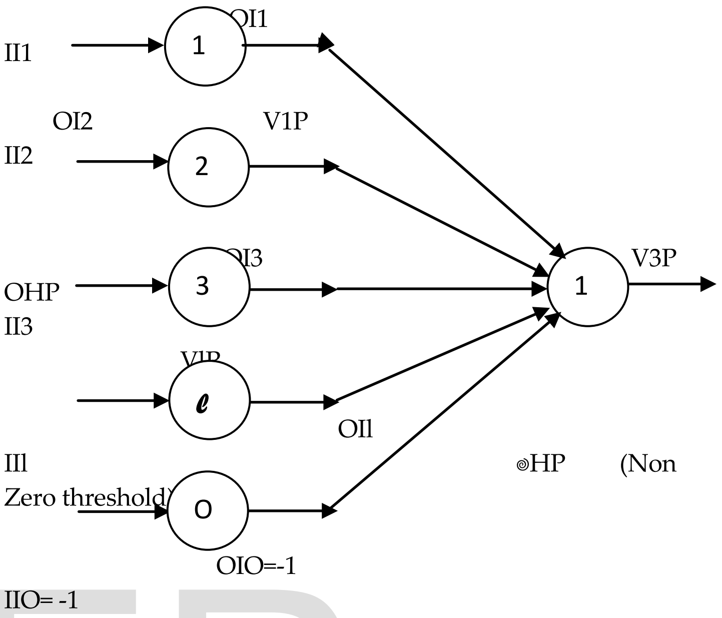


Figure 4: Hidden layer computation

Output layer computation:

Consider sigmoid function. The output of the qth output neuron is given by
Ooq = 1/(1 + e-λ(Ioq-θoq)
Where :
Ooq is the output of the qth output neuron,
Ioq is the input of the qth output neuron,
θoq is the threshold of the qth output neuron,

8. Backpropagation Algorithm

If given are P training pairs
{z1, d1, z2,d2, ………, zp, dp}
Where zi is (l x 1), di is (k x 1), and i=1,2,,…..,P.
Note
(i) lth component of each xi is of value -1 since input vectors have been segmented. Size J-1 of the hidden layer having outputs y is selected.
(ii) the Jth component of y is of value -1, since hidden layer outputs have also been augmented; y is (J x 1) and o is (K x 1).

Step1: $\eta > 0$, Emax chosen
Weight W and Y are initialized at small random values; W is (K x J), V is (J x I).
q <- 1, p <-1, E <- 0
Step2: Training step starts here
Input is presented and the layers output computed
z <- zp, d <- dp
yj <- f(vtj z), for j = 1,2, …. , J
where vj, a column vector , is the jth now of V and
ok <- f(wtk y), for k = 1,2, …. , K
where wk, a column vector , is the kth now of W.
Step3: Error value is computed:

$E \leftarrow \frac{1}{2}\ (dk- ok)2\ + E$ for k = 1,2, …. , K

Step4: Error signal vectors δ0 and δy of both layer are computed. Vector is δ0 is (K x 1), δy is (J x 1)
The error signal terms of the output layer in this step are

$\delta 0k = \frac{1}{2}\ (dk- ok)2\ (1 – o2)$ for k = 1,2, …. , K

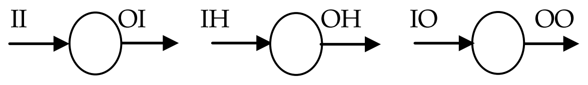
the error signal terms of the hidden layer in this step are

$\delta yj = \frac{1}{2}\ (1 – y2j)2 \sum_{k=1}^{k} \delta 0k\ wkj,$ for j = 1,2, …. , J

Step5: Output layer weights are adjusted :
Wkj <- wkj + η δ0k yj, for k = 1,2, …. , K and j = 1,2, …. , J
Step6: Hidden layer weights are adjusted :
vji <- vji + η δyj zi, for j = 1,2, …. , J and i = 1,2, …. , I
Step7: if p<P the p<-p+1, q<- q+1 and go to step 2 otherwise go to step 8.
Step8: The training cycle is completed.

For E< Emax terminate the training session. Output weights W, V, q, and E.
If E > Emax, then E<- 0, p<- 1, and initiate the new training cycle by going to step2.

How to solve the numerical base on BP
Consider three layers input, hidden and output layer.



(i) Input Layer (ii) Hidden Layer (iii) Output Layer

Figure 5: Layers of BP

The input to the input layer is represented by II and the output of this layer OI .
The input to the hidden layer is represented by IH and the output layer of hidden layer OH.
The input to the output layer is represented by OO and the output of output layer OO.
Write the input for input layer from the given question. The output of this layer is same as the input because there is no processing in the input layer.
Now calculate IH. IH = [V]t * OI. Where [V] is the weight between the input and hidden layer.
Next is to calculate the output of hidden layer OH.

$OH=\frac{1}{[1+\exp(-\lambda net)]}$ where λ is a constant and net is the value calculated above i.e. net = weight * input for the layer. i.e., OH = 1/[1 + exp(-λ. IH)]

Calculate Io . Io = [W]t * OH. where [W] is the weight between the hidden layer and the output layer.
Calculate OO . OO = 1/[1 + exp(-λnet)]. where λ is a constant and net is the value calculated above i.e. net = weight * input for the layer. i.e., Oo = 1/[1 + exp(-λ. Io)]
Calculate the error by E = [1/2]* (d – O)2.
Calculate the error signal term for the output layer

$\delta o = \frac{1}{2}\ [\ (dk- ok)2\ (1 – ok2\ )]^*$

Next Calculate error signal term for the hidden layer
δy = wtj δo fty. where fty = OH (1 - OH)
Adjust the weights of the output layer
Wnew = Wold + η δo . OHt
Adjust the weight of the hidden layer
Vnew = Vold + η δy II t

9. REFERRENCES

Rumelhart; Williams, Ronald J. (8 October 1986). Learning representations by back-propagating errors. doi: 10.1038/323533a0
Kelley, Henry J. (1960). Gradient theory of optimal flight paths. Ars Journal. 30 (10): 947–954. doi:10.2514/8.5282
R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In Back-propagation: Theory, Architectures and Applications. Hillsdale, NJ: Erlbaum, 1994.
Arthur E. Bryson (1961, April). A gradient method for optimizing multi-stage allocation processes. In Proceedings of the Harvard Univ. Symposium on digital computers and their applications.
Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8, 229-256.
Stuart Russell; Peter Norvig (1975). Artificial Intelligence A Modern Approach. p. 578.
Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. Neural Computation, 1, 270-280.
Werbos, Paul John (1975). Beyond Regression: New Tools for Prediction and Analysis in the Beha