Research Paper

Volume : 2 | Issue : 8 | August 2013 • ISSN No 2277 - 8179

Analysis of Back Propagation Algorithm

Mathematics

KEYWORDS : multilayer network, back propagation

Vikas Chahar | Associate Professor, Vaish Institute of Management & Technology - Rohtak

ABSTRACT

In this paper we will discuss the analysis of back propagation algorithm by implementation of this algorithm in metlab. And after implementation try to determine why or why not this algorithm is used in feed forward and feed backward multilayer neural network.

History of algorithm

Minsky and Papert (1969) showed that there are many simple problems such as th exclusive-or problem, which linear neural networks [1] cannot solve. Note that term "solve" means learn the desired associative links. Arguments is that if such networks cannot solve such simple problems how could they solve complex problems in vision, language, and motor control. Solutions of these problems were:

- Select appropriate recoding scheme which transforms inputs
- Perceptron learning rule – requires that you correctly guess an acceptable input to hidden unit mapping
- Back propagation learning rule – learn both sets of weights simultaneously.

Back Propagation Algorithm

The back propagation algorithm [2, 3] is mathematical techniques which require iterative processes and thus is easily implement able on the computer.

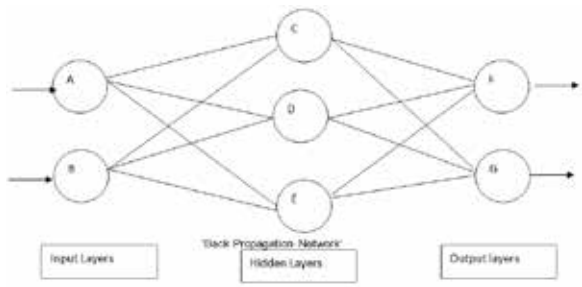
- Weight changes for hidden to output weights just like widrow-hoff learning rule
- Weight changes for input to hidden weights just like widrow-hoff learning rule but error signal is obtained by backpropagating error from the output units.

The process starts by applying the first input patterns Xi and the corresponding target output Tk. The input causes a response to the neurons of the first layer, which in turn cause a response to the neurons of next layer, and so on, until a response and the difference (error signal) is calculated. From the error difference at the output neurons, algorithm computes the rate at which the error changes as the activity level of neuron changes. So far, the calculations were computed forward(from the input layer to output layer).

Then algorithm reschedule by steps back one layer before that output layer and recalculate the weights of the output layer, so that the output error is minimized. Then calculate the error output at the last hidden layer and computes new values for its weights. Processes is continue for calculating the error and computing new weight values, moving layer by layer backward, toward the input. When the input is reached and the weights do not change, then the algorithm selects the next pair of input-target patterns and repeats the process. Although response moves in a forward direction, weights are calculated by moving backward, hence name back propagation.

Implementation of algorithm

While algorithm is implemented in any language following steps must be considered to determine error rate by distinguishing them with actual and desired output.



The algorithm has following steps to determine efficiency of output based on desired outputs:

1. First step is to determine error of output neuron, by specifying each input that is manipulated by a weight that would be either inhabit input or excite output.

   $Xj = \sum_{i=0}^{n} Yi\ Wij$

   Yi – the activity level of ith unit in previous layers

   Wij- is the weight of connection between ith and jth unit

   Then weight of Xj is passed to sigmoid function that would scale the output in between 0 & 1

2. Next step to change output layer weights by calculating the activity Yi, using some function of total weight input

   $$Yj = \frac{1}{1 + e^{-xj}}$$

   Once the output is determine it is compared with the required output and total error E is compared.
3. Once the activity of all output unit have been determined then is to determine hidden layer errors, which is defined by

   $$E = \frac{1}{2}\sum_{i}^{n}(Yi - Dj)^2$$

   Yi – is the activity level of ith unit in the top layer
   Dj- is the desired output of the ith unit
4. Then change hidden layer weight

a. To compute how fast error changes as the activity of an output unit is changed

$$EAi = \frac{\delta E}{\delta Yi} = Yi - Dj$$

EA – is the error derivative that is the differences between the actual and desired activity

b. Then compute how fast the error changes as a weight on connection into an output unit is changed

$$EIj = \frac{\delta E}{\delta Xj} = \frac{\delta E}{\delta Yi} * \frac{Dyi}{Dxj}$$

$$= EAiYj(1 - Yj)$$

c. Then compute how fast the error changes as a weight on connection into an output unit is changed

$$EWij = \frac{\delta E}{\delta Dij} = \frac{\delta E}{\delta Xj} * \frac{\delta Xj}{DWij} = EIj\ Yi$$

d. Finally, determine how fast the error changes as the activity of a unit in previous layer changed

$$EAj = \frac{\delta E}{\delta Yi} = \frac{\delta E}{\delta Xj} * \frac{\delta Xj}{\delta Yj} = \sum_{j}^{n} EIj\, Wij$$

Discussion

1. Back propagation algorithm is widely used however, it is not escaped criticism. Here backwards-calculating weights does not seem to be biological plausible; neurons do not seem to work backward to adjust the efficacy of their synaptic weights. Thus, the back propagation learning algorithm is not viewed by many as a learning process that emulates the biological world but as a method to design a network with the learning.
2. This algorithm uses a digital computer to calculate weights. When final network is implemented in hardware, it lost its plasticity. And if changes are necessary computer calculates a new weight values and updates them. This means that neural network implementation still depends on a digital computer.
3. This algorithm suffers from extensive calculations, hence slows training speed. The time required to calculate the error derivatives and to update the weights on a given training exemplar is proportional to the size of the network.
4. The computation speed inefficiency of this algorithm has triggered an effort to explore techniques that accelerated the learning time by at least a factor of 2. Due to this this algorithm is not suitable for many real time applications.
5. Due to its wide applicability, this algorithm cannot be applied to all neural network systems which can be imagined. This algorithm requires that the activation functions of each of the neurons in the network, both continuous and differentiable.

REFERENCE

1. Laurenne Fausett. "Fundamentals of Neural Network". Prentice Hall. 1994. | 2. Uthayakumar Gevaran, "Back Propagation", Brandeis University, Department of Computer Science. | 3. Pete Mc Collum, "An Introduction to Back Propagation Neural Networks", The Newsletter of the Seattle Robotics Society.