Neural Networks

Lecture 14

Supervised Learning and Backpropagation Neural Networks (2)

Md. Mijanur Rahman, Prof. Dr.

Dept. of Computer Science and Engineering, Jatiya Kabi Kazi Nazrul Islam University, Bangladesh.

Web: www.mijanrahman.com | Email: mijanjkkniu@gmail.com; mijan@jkkniu.edu.bd

Supervised Learning and Backpropagation Neural Network

- **This chapter covers the following topics:**
  - Supervised Learning Network
  - Perceptron
  - Adaptive Linear Neuron (Adaline)
  - Multiple Adaptive Linear Neuron (Madaline)
  - Generalized Delta Learning Rule
  - Backpropagation Neural Networks
  - Backpropagation Algorithm
  - Types of Activation Functions

Generalized Delta Learning Rule…

- Delta rule works only for the output layer. On the other hand, generalized delta rule, also called as **back-propagation** rule, is a way of creating the desired values of the hidden layer.
- **Mathematical Formulation:**

For the activation function $y_k = f(y_{ink})$ the derivation of net input on Hidden layer as well as on output layer can be given by

$$y_{ink} = \sum_i z_i w_{jk}$$

And $y_{inj} = \sum_i x_i v_{ij}$

Now the error which has to be minimized is

$$E = \frac{1}{2} \sum_k [t_k - y_k]^2$$

Generalized Delta Learning Rule…

- **Mathematical Formulation:**

By using the chain rule, we have

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}}\left(\frac{1}{2}\sum_{k}[t_k - y_k]^2\right)$$

$$= \frac{\partial}{\partial w_{jk}}\left(\frac{1}{2}[t_k - t(y_{ink})]^2\right)$$

$$= -[t_k - y_k]\frac{\partial}{\partial w_{jk}} f(y_{ink})$$

$$= -[t_k - y_k] f(y_{ink})\frac{\partial}{\partial w_{jk}}(y_{ink})$$

$$= -[t_k - y_k] f'(y_{ink}) z_j$$

Now let us say $\delta_k = -[t_k - y_k] f'(y_{ink})$

Generalized Delta Learning Rule…

- **Mathematical Formulation:**

The weights on connections to the hidden unit $\mathbf{z_j}$ can be given by −

$$\frac{\partial E}{\partial v_{ij}} = -\sum_k \delta_k \frac{\partial}{\partial v_{ij}} (y_{ink})$$

Putting the value of y_{ink} we will get the following

$$\delta_j = -\sum_k \delta_k w_{jk} f'(z_{inj})$$

Generalized Delta Learning Rule

- **Mathematical Formulation:**

Weight updating can be done as follows −

For the output unit −

$$\Delta w_{jk} = -\alpha \frac{\partial E}{\partial w_{jk}}$$

$$= \alpha \, \delta_k \, z_j$$

For the hidden unit −

$$\Delta v_{ij} = -\alpha \frac{\partial E}{\partial v_{ij}}$$

$$= \alpha \, \delta_j \, x_i$$

What is Backpropagation?...

- Backpropagation training algorithm is a supervised learning algorithm for multilayer feed forward neural network.
- Since it is a supervised learning algorithm, both input and target output vectors are provided for training the network.
- The purpose of the supervised learning (or training) is to find out a set of weight that can minimize the error E over the complete set of training pair.
- Every cycle in which each one of the training patterns is presented once to the neural network is called an epoch.
- The error data at the output layer is calculated using network output and target output. Then the error is back propagated to intermediate layers, allowing incoming weights to these layers to be updated

What is Backpropagation?...

- The Backpropagation (BP) algorithm is based on the error-correction learning rule. The BP algorithm was introduced by Bryson and Ho in 1969.
- This BP algorithm looks for the minimum value of the error function in weight space using a technique called **the delta rule or gradient descent**.
- The weights that minimize the error function is then considered to be **a solution to the learning problem**.

What is Backpropagation?...

- The BP algorithm does gradient descent as it moves in direction opposite to the gradient of the error that is in direction of the steepest decrease of the error.
- Gradient based methods are one of the most widely used error minimization methods used to train back-propagation neural networks. Gradient descent is a first-order optimization algorithm.
- To find a local minimum of error function using gradient descent, the weight update is proportional to the negative gradient of the error function at the current point.

What is Backpropagation?...

- In the simplest form, the weight update is a scaled step in the opposite direction of the gradient, is multiplied by a constant value, the learning-rate η, is given by the following equation:

$$\Delta w(t) = -\eta \nabla E(t)$$

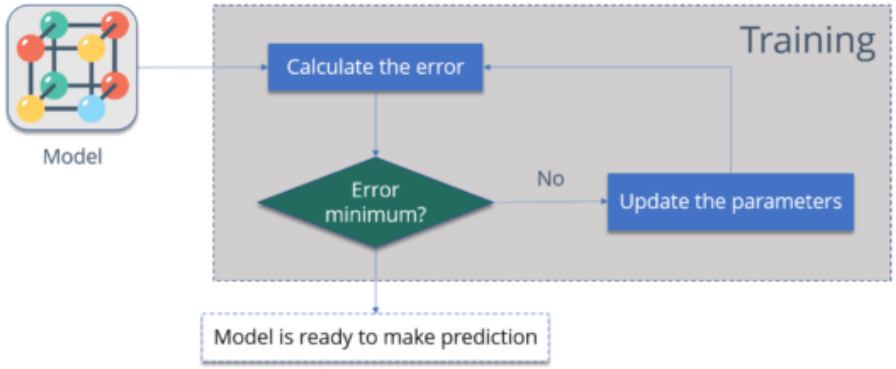
Or, for a single weight:

$$\Delta w_{ij}(t) = -\eta \frac{\partial E}{\partial w_{ij}}(t)$$

- **This minimization technique is commonly known as 'gradient descent'. It is also known as 'steepest descent'.**

What is Backpropagation?

- **The algorithmic steps are summarized below:**
  1. **Calculate the error** – How far is the desired output from the actual output.
  2. **Minimum Error** – Check whether the error is minimized or not.
  3. **Update the parameters** – If the error is huge then, update the parameters (weights and biases). After that again check the error. Repeat the process until the error becomes minimum.
  4. **Model is ready to make a prediction** – Once the error becomes minimum, you can feed some inputs to your model and it will produce the output.



History of Backpropagation

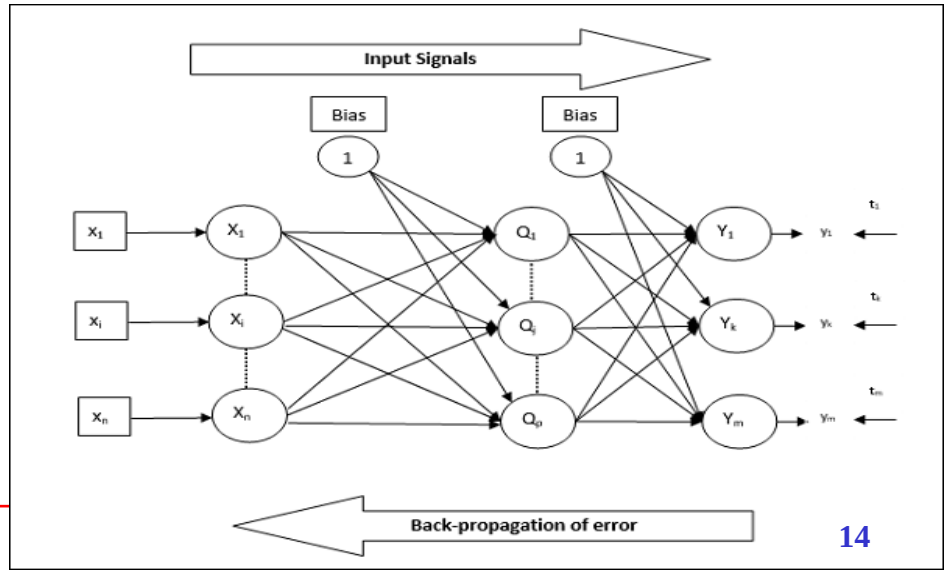
- In 1961, the basics concept of continuous backpropagation were derived in the context of control theory by J. Kelly, Henry Arthur, and E. Bryson.
- In 1969, Bryson and Ho gave a multi-stage dynamic system optimization method.
- In 1974, Werbos stated the possibility of applying this principle in an artificial neural network.
- In 1982, Hopfield brought his idea of a neural network.
- In 1986, by the effort of David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams, backpropagation gained recognition.
- In 1993, Wan was the first person to win an international pattern recognition contest with the help of the backpropagation method.

Backpropagation Neural Networks…

- **Back Propagation Neural (BPN)** is a multilayer neural network consisting of the input layer, at least one hidden layer and output layer.
- As its name suggests, back propagating will take place in this network.
- The error which is calculated at the output layer, by comparing the target output and the actual output, will be propagated back towards the input layer.

Backpropagation Neural Networks…

- **BPN Architecture:**
  - The architecture of BPN has three interconnected layers having weights on them.
  - The hidden layer as well as the output layer also has bias, whose weight is always 1, on them.
  - The working of BPN is in two phases:
    - One phase sends the signal from the input layer to the output layer, and
    - The other phase back propagates the error from the output layer to the input layer.



Backpropagation Neural Networks…

- **BPN Training Algorithm**
- For training, BPN will use binary sigmoid activation function. The training of BPN will have the following three phases.
  - Phase 1 − Feed Forward Phase
  - Phase 2 − Back Propagation of error
  - Phase 3 − Updating of weights

Backpropagation Neural Networks…

- **BPN Training Algorithm:**
- All these steps will be concluded in the algorithm as follows:

Step 1 − Initialize the following to start the training −

- Weights
- Learning rate α

For easy calculation and simplicity, take some small random values.

Step 2 − Continue step 3-11 when the stopping condition is not true.

Step 3 − Continue step 4-10 for every training pair.

Backpropagation Neural Networks…

- **Phase 1:**

Step 4 − Each input unit receives input signal x_i and sends it to the hidden unit for all **i = 1 to n**

Step 5 − Calculate the net input at the hidden unit using the following relation −

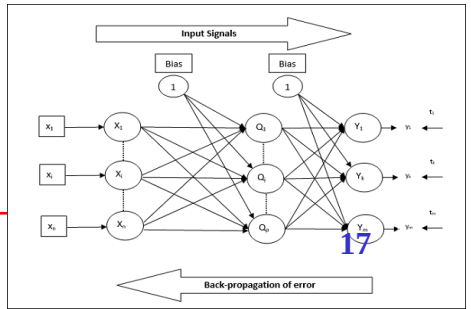
$$Q_{inj} = b_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad j = 1 \, to \, p$$

Here b_{0j} is the bias on hidden unit, v_{ij} is the weight on **j** unit of the hidden layer coming from **i** unit of the input layer.

Now calculate the net output by applying the following activation function

$$Q_j = f(Q_{inj})$$

Send these output signals of the hidden layer units to the output layer units.



Backpropagation Neural Networks…

- **Phase 1:**

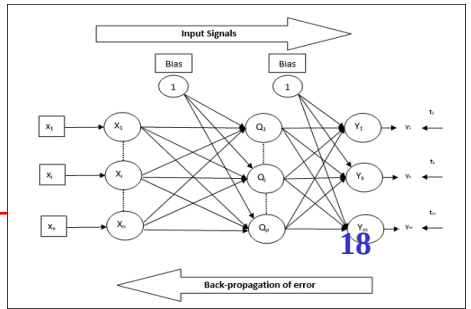
Step 6 − Calculate the net input at the output layer unit using the following relation −

$$y_{ink} = b_{0k} + \sum_{j=1}^{p} Q_j \, w_{jk} \quad k = 1 \, to \, m$$

Here $\mathbf{b_{0k}}$ is the bias on output unit, $\mathbf{w_{jk}}$ is the weight on **k** unit of the output layer coming from **j** unit of the hidden layer.

Calculate the net output by applying the following activation function

$$y_k = f(y_{ink})$$



Backpropagation Neural Networks…

- **Phase 2:**

Step 7 − Compute the error correcting term, in correspondence with the target pattern received at each output unit, as follows −

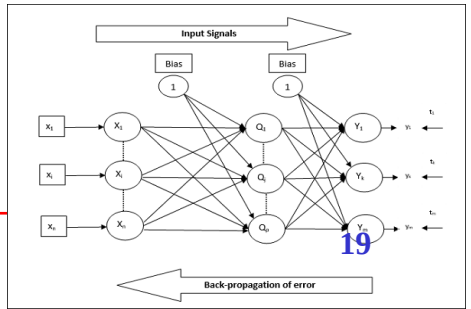
$$\delta_k = (t_k - y_k) f'(y_{ink})$$

On this basis, update the weight and bias as follows −

$$\Delta v_{jk} = \alpha \delta_k Q_{ij}$$

$$\Delta b_{0k} = \alpha \delta_k$$

Then, send δ_k back to the hidden layer.



Backpropagation Neural Networks…

- **Phase 2:**

Step 8 − Now each hidden unit will be the sum of its delta inputs from the output units.

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk}$$

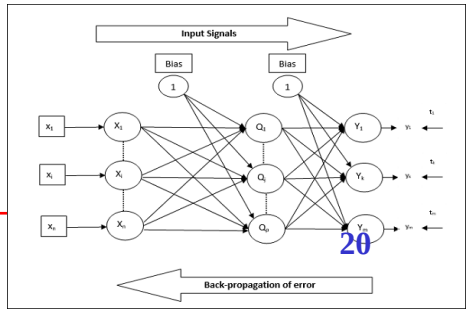
Error term can be calculated as follows −

$$\delta_j = \delta_{inj} f'(Q_{inj})$$

On this basis, update the weight and bias as follows −

$$\Delta w_{ij} = \alpha \delta_j x_i$$

$$\Delta b_{0j} = \alpha \delta_j$$



Backpropagation Neural Networks

- **Phase 3:**

Step 9 − Each output unit ***(y_kk = 1 to m)*** updates the weight and bias as follows −

$$v_{jk}(new) = v_{jk}(old) + \Delta v_{jk}$$

$$b_{0k}(new) = b_{0k}(old) + \Delta b_{0k}$$

Step 10 − Each output unit ***(z_j = 1 to p)*** updates the weight and bias as follows −

$$w_{ij}(new) = w_{ij}(old) + \Delta w_{ij}$$

$$b_{0j}(new) = b_{0j}(old) + \Delta b_{0j}$$

Step 11 − Check for the stopping condition, which may be either the number of epochs reached or the target output matches the actual output.

SUPERVISED LEARNING AND BACKPROPAGATION NETWORK PARADIGMS

To be continued…