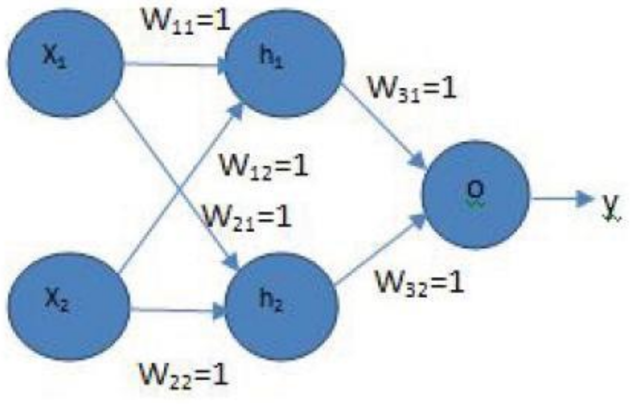
Assignment No: 03

Aim: The figure shows a single hidden layer neural network. The weights are initialized to 1‟s as shown in the diagram and all biases are initialized to 0‟s. Assume all the neurons have linear activation functions. The neural network is to be trained with stochastic (online) gradient descent. The first training example is [x1=1, x2=0] and the desired output is 1. Design the back-propagation algorithm to find the updated value for W11 after back propagation. Choose the value that is the closest to the options given below: [learning rate =0.1]



Objectives:

1. To design a back-propagation algorithm.

Software Requirements:

Ubuntu 18.04

Hardware Requirements:

Pentium IV system with latest configuration

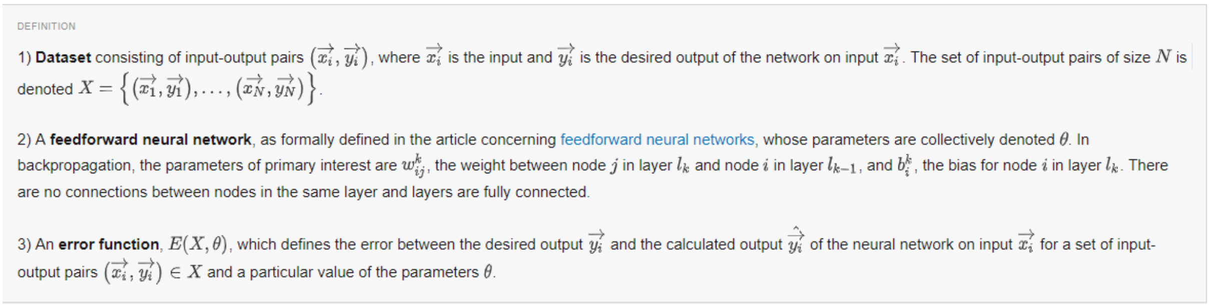
Theory:

Back propagation, short for "backward propagation of errors," is an algorithm for supervised learning of artificial neural networks using gradient descent. Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural network's weights. It is a generalization of the delta rule for perceptrons to multilayer feed forward neural networks.

The "backwards" part of the name stems from the fact that calculation of the gradient proceeds backwards through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last. Partial computations of the gradient from one layer are reused in the computation of the gradient for the previous layer. This backwards flow of the error information allows for efficient computation of the gradient at each layer versus the naive approach of calculating the gradient of each layer separately.

Back propagation's popularity has experienced a recent resurgence given the widespread adoption of deep neural networks for image recognition and speech recognition. It is considered an efficient algorithm, and modern implementations take advantage of specialized GPUs to further improve performance.

Formal Defination



DEFINITION

1) **Dataset** consisting of input-output pairs $\left(\vec{x_i}, \vec{y_i}\right)$, where $\vec{x_i}$ is the input and $\vec{y_i}$ is the desired output of the network on input $\vec{x_i}$. The set of input-output pairs of size N is denoted $X = \left\{\left(\vec{x_1}, \vec{y_1}\right), \dots, \left(\vec{x_N}, \vec{y_N}\right)\right\}$.

2) A **feedforward neural network**, as formally defined in the article concerning feedforward neural networks, whose parameters are collectively denoted θ. In backpropagation, the parameters of primary interest are w_{ij}^k, the weight between node j in layer l_k and node i in layer l_{k-1}, and b_i^k, the bias for node i in layer l_k. There are no connections between nodes in the same layer and layers are fully connected.

3) An **error function**, $E(X, \theta)$, which defines the error between the desired output $\vec{y_i}$ and the calculated output $\hat{\vec{y_i}}$ of the neural network on input $\vec{x_i}$ for a set of input-output pairs $\left(\vec{x_i}, \vec{y_i}\right) \in X$ and a particular value of the parameters θ.

The Backpropagation Algorithm

Using the terms defined in the section titled Formal Definition and the equations derived in the section titled Deriving the Gradients, the backpropagation algorithm is dependent on the following five equations:

For the partial derivatives,

$$\frac{\partial E_d}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1}.$$

For the final layer's error term,

$$\delta_1^m = g_o'(a_1^m)(\hat{y_d} - y_d).$$

For the hidden layers' error terms,

$$\delta_j^k = g'\left(a_j^k\right) \sum_{l=1}^{r^{k+1}} w_{jl}^{k+1} \delta_l^{k+1}.$$

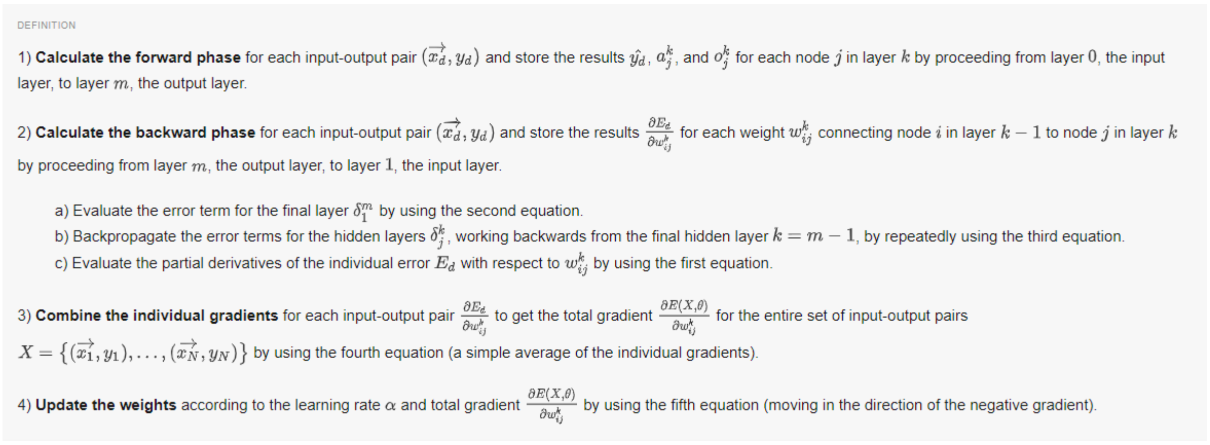
For combining the partial derivatives for each input-output pair,

$$\frac{\partial E(X,\theta)}{\partial w_{ij}^k} = \frac{1}{N} \sum_{d=1}^{N} \frac{\partial}{\partial w_{ij}^k} \left(\frac{1}{2}\left(\hat{y_d} - y_d\right)^2\right) = \frac{1}{N} \sum_{d=1}^{N} \frac{\partial E_d}{\partial w_{ij}^k}.$$

For updating the weights,

$$\Delta w_{ij}^k = -\alpha \frac{\partial E(X,\theta)}{\partial w_{ij}^k}.$$

The General Algorithm



DEFINITION

1) **Calculate the forward phase** for each input-output pair $(\overrightarrow{x_d}, y_d)$ and store the results $\hat{y_d}$, a_j^k, and o_j^k for each node j in layer k by proceeding from layer 0, the input layer, to layer m, the output layer.

2) **Calculate the backward phase** for each input-output pair $(\overrightarrow{x_d}, y_d)$ and store the results $\frac{\partial E_d}{\partial w_{ij}^k}$ for each weight w_{ij}^k connecting node i in layer $k-1$ to node j in layer k by proceeding from layer m, the output layer, to layer 1, the input layer.

a) Evaluate the error term for the final layer δ_1^m by using the second equation.
b) Backpropagate the error terms for the hidden layers δ_j^k, working backwards from the final hidden layer $k = m - 1$, by repeatedly using the third equation.
c) Evaluate the partial derivatives of the individual error E_d with respect to w_{ij}^k by using the first equation.

3) **Combine the individual gradients** for each input-output pair $\frac{\partial E_d}{\partial w_{ij}^k}$ to get the total gradient $\frac{\partial E(X,\theta)}{\partial w_{ij}^k}$ for the entire set of input-output pairs $X = \{(\overrightarrow{x_1}, y_1), \dots, (\overrightarrow{x_N}, y_N)\}$ by using the fourth equation (a simple average of the individual gradients).

4) **Update the weights** according to the learning rate α and total gradient $\frac{\partial E(X,\theta)}{\partial w_{ij}^k}$ by using the fifth equation (moving in the direction of the negative gradient).

Conclusion:

Thus we have designed back propagation algorithm.