Assignment SCOA-6

Title: Backpropagation Algorithm and Neural Network

Problem Statement:

The figure shows a single hidden layer neural network. The weights are initialized to is as shown in the diagram and all biases are initialized to as. Assume all the neurons have linear activation functions. The neural network is to be trained with stochastic (online) gradient. descent. The first training example output is is [x_1=1, X_2=0] and the desired output is1. Design the back propagation algorithm to find the upidated value for Wi after back propagation

Objectives:

- ❖ To learn and implement back propagation using gradient descent.
- ❖ To understand need of back propagation algorithm.

Outcomes:

Students will be able to:

- Implement back propagation using Suitable algorithm.

Software and Hardware Requirements:

4GB RAM, Processor i3 and above, 500GB HDD, OS Fedora / Ubuntu, Text Editor, Jupiter Notebook, Python 2.7 or 3.6

Theory

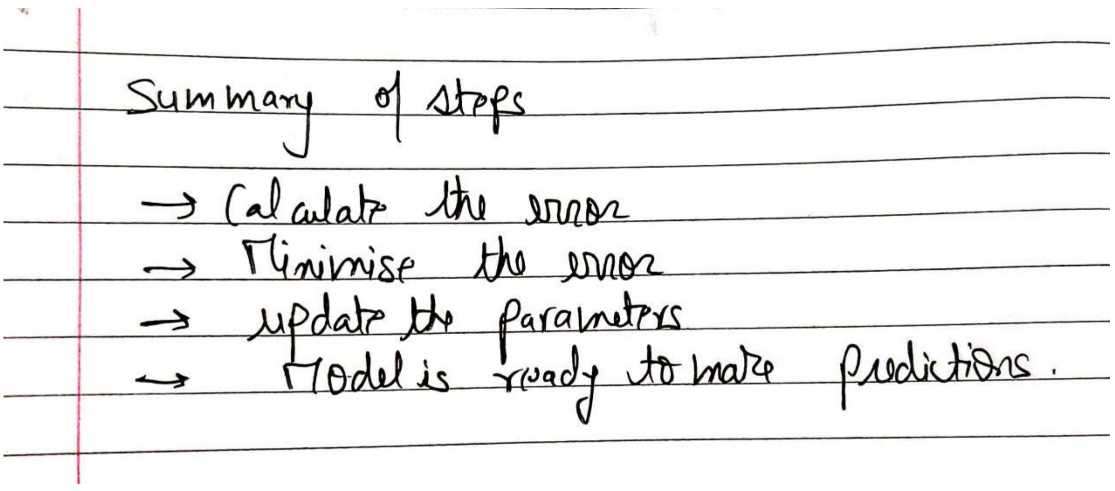
Backpropagation, short for "backward propagation of errors," is an algorithm for supervised learning of artificial neural networks using gradient descent. Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural network's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

Backpropagation'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.

It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (le iteration). Proper tuning of the weights allows us to reduce error rates and to make the model reliable by increasing its generalization. This method helps to calculate the gradient of a loss function with respect to all the weights in the network.

Backpropagation error is computed at the output and is attributed backward throughout the network layer. Insiest commonly used to train a neural network



Summary of steps

→ Calculate the error

→ Minimise the error

→ update the parameters

→ Model is ready to make predictions.

Steps :

Step 1 : Forward propagation.

$$\begin{tikzcd}
I_1 \arrow[r] & x_1 \arrow[r, "w_{11}"] \arrow[rd, "w_{12}"] & h_1 \arrow[rd, "w_{31}"] & & \\
& & & O \arrow[r] & {} \\
I_2 \arrow[r] & x_2 \arrow[r, "w_{22}"'] \arrow[ru, "w_{21}"] & h_2 \arrow[ru, "w_{32}"'] & &
\end{tikzcd}$$

Step 2 : Back propagation

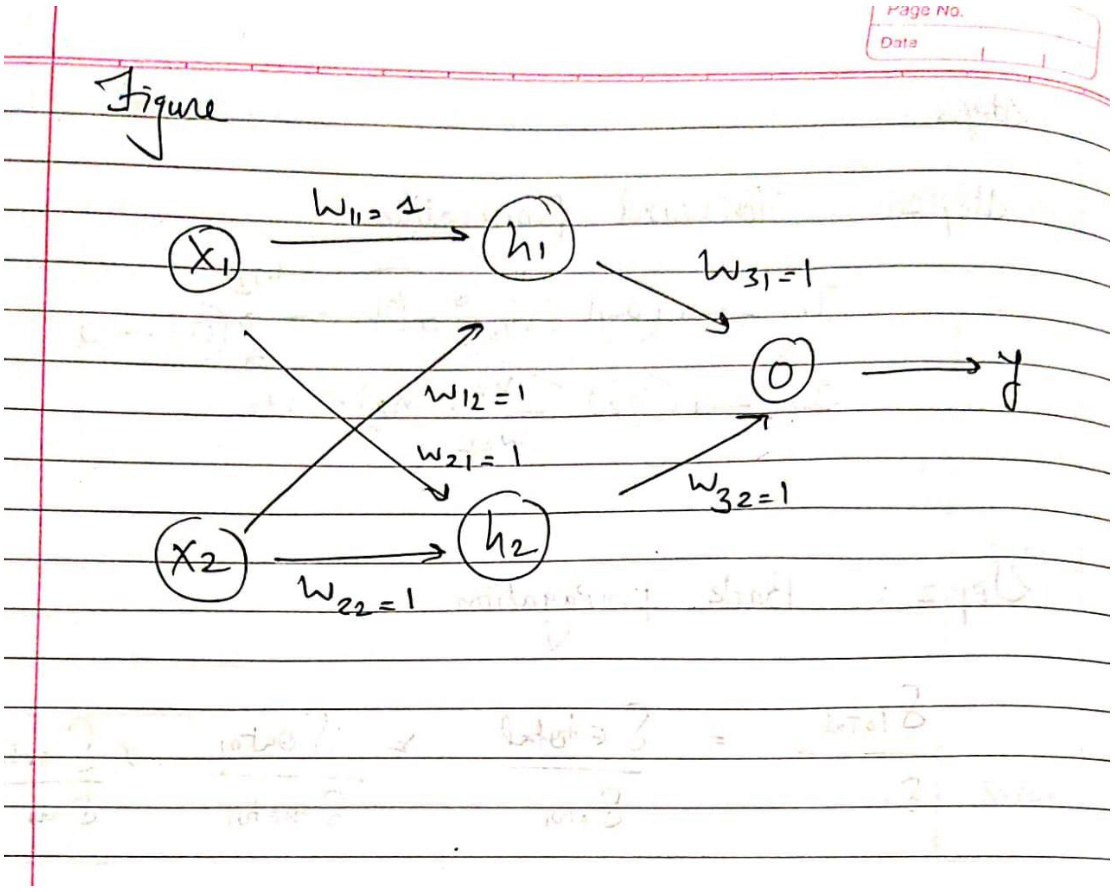
$$\frac{\delta_{Total}}{\delta_o} = \frac{\delta E_{total}}{\delta out_{01}} * \frac{\delta out_{01}}{\delta net_{01}} * \frac{\delta net_{01}}{\delta w_i}$$

Similarly error for every hidden layer node is calculated.

weights are updated as follows.

$$W_i = W_s - n \int \frac{\delta total}{\delta w}$$

After calculating errors and setting weights, forward propagation is calculated again.



Conclusion:

We have successfully implemented stochastic gradient for back propagation algorithm.