Deep Learning Homework 1 Report

Assem Kussainova
Data Science
Nazarbayev University
Nur-Sultan, Kazakhstan
assem.kussainova@nu.edu.kz

I. Task

It was asked to implement backpropagation for a 2 hidden-layer network, which has the following architecture, where x1 and x2 are inputs, o1 and o2 are outputs, and h1-h4 are the units of two hidden layers.

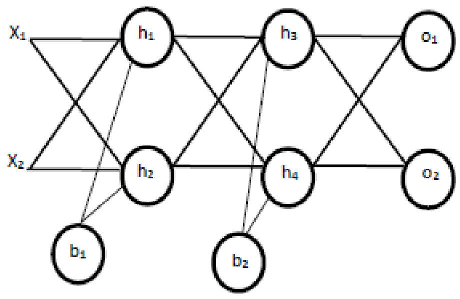

Fig. 1. Neural network architecture

Given inputs x1=0.2 and x2 = 0.6, and the outputs are o1=0.1 and o2=0.9. In this network, it is needed to use the sigmoid activation in h1-h4 and in o1, o2, to set the initial weights randomly to fall in range between 0.001 and 0.5, and to use squared error as loss function. Taken biases associated with first and second hidden layers are 0.25 and 0.45, respectively. Initial learning rate is 0.01.

II. Implementation

In given network, each neuron has a set of weights: one weight for each input connection and an additional weight for the bias, except outputs. A dictionary was used to represent each neuron and store related values. A network is organized into layers. Layers are organized as arrays of dictionaries and treat the whole network as an array of layers.

Neuron activation is calculated as the weighted sum of the inputs. Once a neuron is activated, it is needed to transfer the activation to see what the neuron output actually is. It is done by using sigmoid function. Next, forward propagation is done through the network. The outputs for each neuron are calculated through each layer of the network. All of the outputs from one layer become inputs to the neurons on the next layer. This function returns calculated values for two outputs using initial weights.

Error is calculated between the expected outputs and the outputs forward propagated from the network. These errors are then propagated backward through the network from the output layer to the hidden layer, assigning blame for the error and updating weights.

Once errors are calculated for each neuron in the network via the back propagation method, they are used to update weights. When the weights have been updated, the process starts again from forward propagation until the correct values of outputs have been reached.

III. Results

For the network were used inputs with 10 different values of weights as the code was running 10 times. Each iteration took 10000 epochs as it was needed to obtain outputs as close as possible to real targets. Following is the dependency of error rate to the number of epochs.

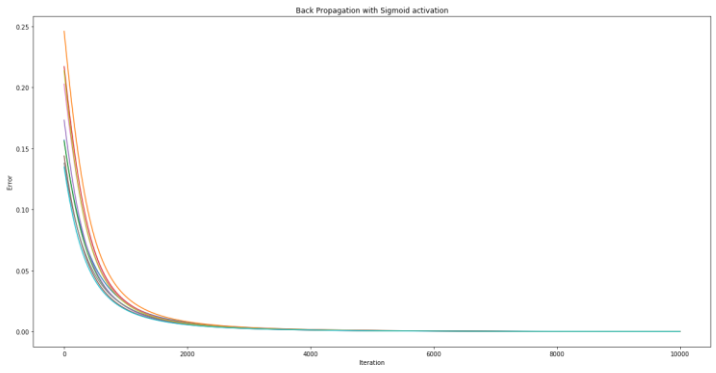

Fig. 2. Back propagation with sigmoid function

It can be seen that graph reflects relationship of error rate and iterations as follows: the higher number of epochs, the lower is error. As for the different weights at each input of 10 cases, all of them have different initial error rate, but have the same tendency, lowering to 0.

IV. Conclusion

In this assignment, the aim of becoming familiar with backpropagation in networks was achieved. Given network architecture was implemented using Python programming language. Algorithm included implementation of a forward pass, the error measurement, and performing the backpropagation to adjust the weights including the biases. Then, process was repeated 10000 times and using that, error rate graph was plotted, showing how error changes after each forward pass. After generating the graph, process was made multiple times with random weights and new graph was obtained in order to observe tendency in different cases. It was identified that error rate changes in the same way for all cases independently from weights.