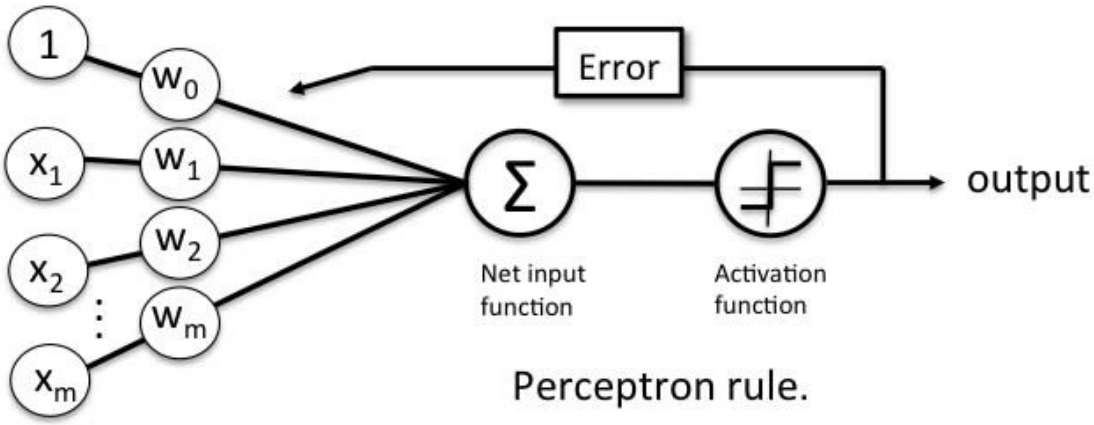
ELL409
Assignment 1
Achint Kumar Aggarwal
2018EE10433

Introduction:
The assignment required us to build a 10-class classifier for the MNIST data set of handwritten digits.
Neural Networks were used to achieve the same. The document first contains a brief description of the math and ideas behind neural networks, followed by specific focus on the given problem.

The Idea Behind Neural Networks:



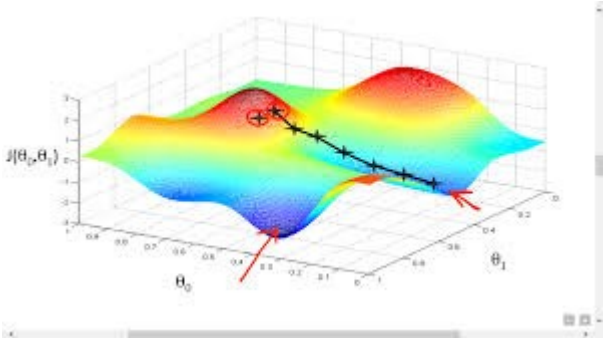
Forward Pass:

A linear hyperplane type function using the given parameters is applied. Following this linear input is a non-linear function, in order to prevent the network from collapsing into a simple matrix multiplication. The process is repeatedly applied in order to obtain output for each perceptron. Several perceptrons comprise a layer, several of these layers form the neural network.

Backward Pass:

Error is passed backwards in accordance with the weights between the considered layers. The new error hence obtained (given that we had the error of the following layer) is used to update the previous layer of weights, and so on.

The mathematical tool used to implement the same is gradient descent.

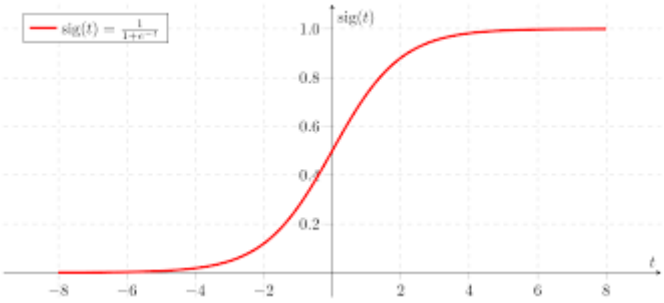


The specific type used here is stichastic gradient descent, wherein loss for only one input was considered at a given time.

$$\text{Repeat Until Convergence } \{$$
$$\text{for i} = 1...\text{m } \{$$
$$\omega \leftarrow \omega - \alpha * \nabla_w L_m(w)$$
$$\}$$
$$\}$$

Besides, the activation function used in all the layers (except the last one is sigmoid). The one used in the last layer is softmax function, which is the typical one ised in multi-class classification as is the case in the given problem.

Sigmoid function:



Softmax Function:

Softmax

$$f_i(\vec{a}) = \frac{e^{a_i}}{\sum_k e^{a_k}}$$

The derivative of both the functions happens to be: f'(x) = f(x).(1 – f(x)).

--

Initialization of weights: The weights have been initialized to be normal random variables (IID). We needed small values for the weights. Still, we couldn't let them all be the same or 0. This would have resulted the gradient w.r.t each to be in the same direction and would have thus failed to properly update the networks.

--

Updation: Using the aforementioned gradient descent rule.

--

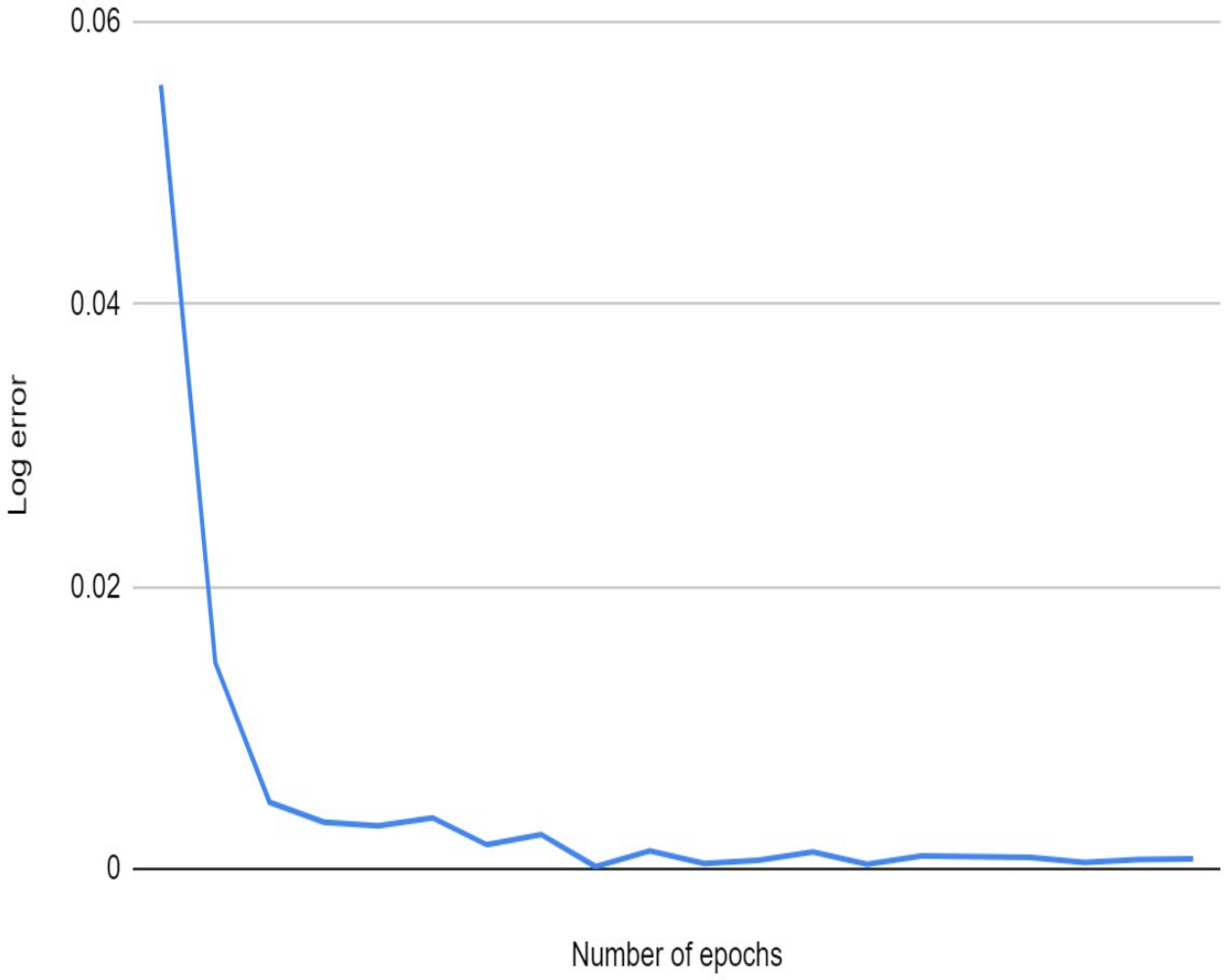
Overall, empirical risk has been minimised which ensures minimisation of the true risk as they former converges to the latter due to the weak law of large numbers.

Analyzing the results of the given problem:

Using the Plots of Log-error v/s No. Of epochs run-

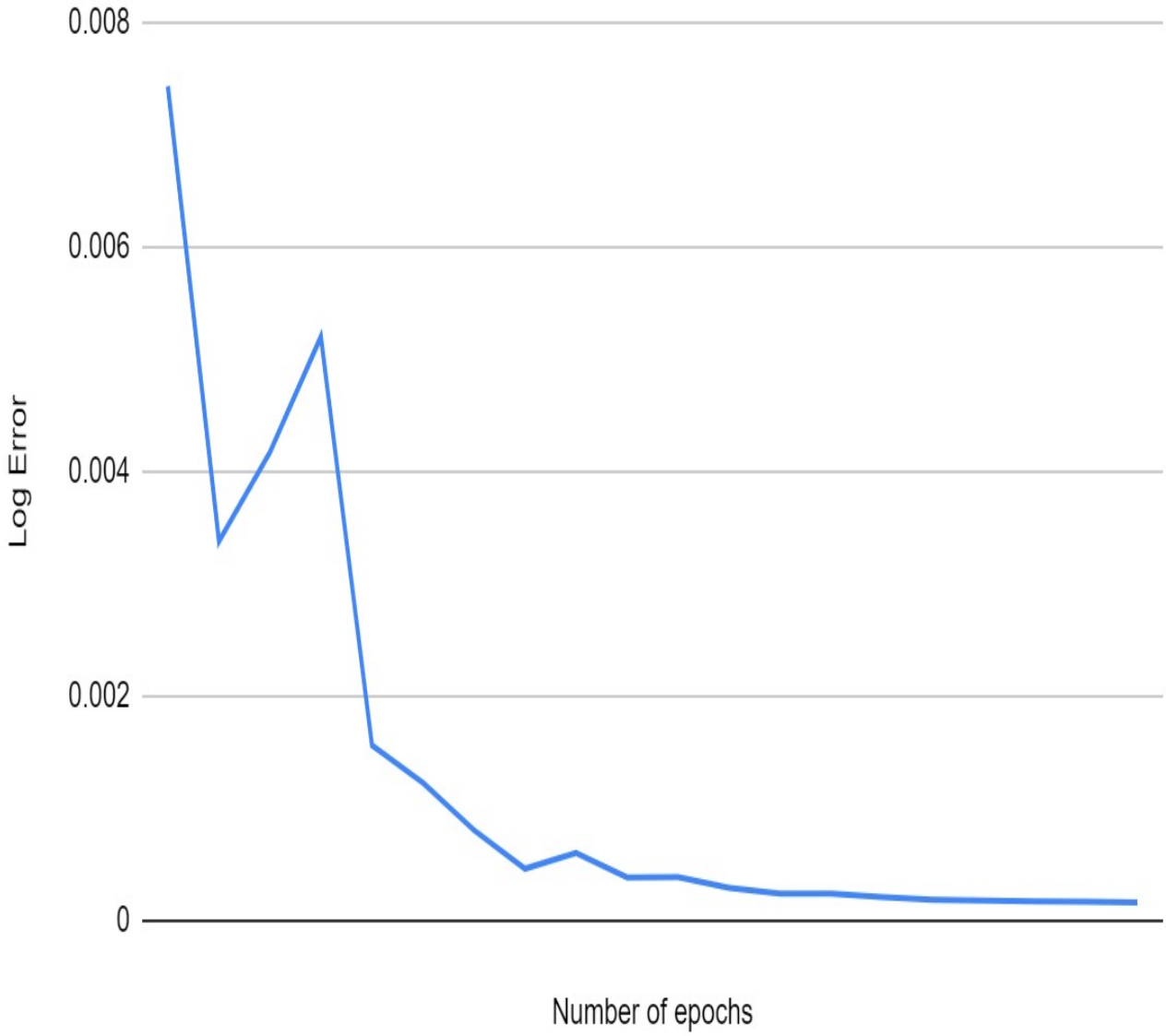
1. Architecture: 785-121-61-10:

Learning rate: 0.211
Regularization parameter: 0.001
Max Epochs = 20



2. Architecture: 785-121-10:

Learning Rate: 0.211
Regularization parameter: 0.001
Max Epochs: 20



3. **Architecture: 785-121-10:**

Learning Rate: 0.011
Regularization parameter: 0
Max Epochs: 20