Report on
Artificial Neural Networks

Team

1. Sai Satvik Vuppala (2017B4A71449H)
2. Shreya Kotagiri (2017B3AA0296H)
3. Dhanush Karupakula(2017B3A71011H)

Introduction

We implement a simple artificial neural network with one and two hidden layers that can perform multi-class (1-out-of-10) classification using non-linear activation function(sigmoid) for the hidden and output layers. We use Stochastic Gradient Descent to train the model. A random 70:30 split on the given data is done.

Libraries Used

1. Numpy
2. Pandas
3. Matplotlib
4. Math
5. Random

Implementation

This algorithm is implemented to solve multiclass(1-out-of-10) classification problem. The dataset has 6 input features, one of ten possible output classes. Using this algorithm, we try to emulate a biological neural network, where multiple layers of processing elements(neurons) are connected, with output of one layer given as weighted the input to the next layer, with some processing done at each layer.

The NN makes some random predictions which are then matched with the correct output class. The difference between these predicted and actual values(errors) are calculated, which make up the cost function. The aim is to minimize this cost function, which helps to predict the estimates of the weights of input to each layer as close to the correct class as possible. So higher number of iterations(epochs) will help to get better predictions.

One-Layered NN:

The neural network is going to have the following architecture:

6 input features mapped to a hidden layer with 5 nodes. The output layer will have 10 nodes, depicting the multilayer classification problem with possible output classes as 10.

The input features will be the input to the hidden layer and the hidden layer in turn will be the input to the output layer.

The activation function used, which defines how the linear combination of input is translated into output at a node is sigmoid function in this case. With logistic activation function used, we define the cost function to be a stochastic gradient descent function for cost optimization.

The neural network works in two steps- forward propagation and backward propagation.

In forward propagation, we first make predictions based on input features and the initialized weights values, taken as a dot product of weights and input features.

Using the errors found between the predicted values found above and the actual values, back propagation is done to correct the weights. We find the cost function first as part of this- in this case, stochastic gradient descent has been used. We then minimize this cost function, thereby reducing the error between the predicted and actual values to reach the correct classification of our input tuple.The back propagation cost function derivative will be represented using a chain rule to arrive at the weights w.

Two-layered ANN:

Two-layered ANN implemented will have 6 input features layer, with two hidden layers with 5 nodes each and an output layer with 10 nodes, for multiclass classification.

The input features layer will be an input to the first hidden layer, and the first hidden layer will in turn be the input to the second hidden layer, which will be an input to the output layer.

We use a sigmoid activation function, to minimize the cost function to arrive at the weights.

Forward propagation is fairly straightforward, where we make predictions of weight values.

We take the advantage of chain and power rules to find weights at each layer.

Brief description of your chosen hyper-parameters

- We implement the NN using one and two hidden layers, with five nodes present in each of the hidden layers in both one-layered and two-layered ANN. Input layer consists of 6 nodes(6 features) and the output layer has 10 nodes(representing the 10 classes)
- Sigmoid function has been used as the activation function to determine the weight values for input of every layer
- Three learning rates are used to minimize the cost function using stochastic gradient descent- 0.01, 0.1, 0.9 for two layered neural network and 0.01, 0.3, 0.9 for one layered neural network.

The amount through which the weights are updated during training is known as the step size or the learning rate. It is a configurable hyper-parameter during the training of neural networks. The learning rate governs the time a model takes to adapt to a given problem. A lower value of learning rates implies more training epochs are required given the smaller changes that will be made to the weights each update, similarly a larger learning rate implies rapid changes and require fewer training epochs.

Train and test metrics

Two Layered Neural Network

1. **With η= 0.9**
   _____TRAIN DATA_____
   Average Cost: 550.8346816406967
   Average Accuracy: 75.9557142857143

   _____TEST DATA_____

   Average Accuracy: 75.793%

2. **With η= 0.1**

   _____TRAIN DATA_____

   Average Cost: 580.1864723703909

   Average Accuracy: 74.71357142857146

   _____TEST DATA_____

   Average Accuracy: 74.624%

3. **With η= 0.01**
   _____TRAIN DATA_____

   Average Cost: 764.6568751939219

   Average Accuracy: 66.71357142857147

   _____TEST DATA_____

   Average Accuracy: 75.626%

One Layered Neural Network

4. **With η= 0.9**

   _____TRAIN DATA_____
   Average Cost: 0.0027779495309844847
   Average Accuracy: 74.7207142857143
   _____TEST DATA_____

   Average Accuracy: 76.294%

5. **With η= 0.3**

   _____TRAIN DATA_____

   Average Cost: 0.013120242697323252

   Average Accuracy: 74.97642857142857

   _____TEST DATA_____

   Average Accuracy: 75.960%

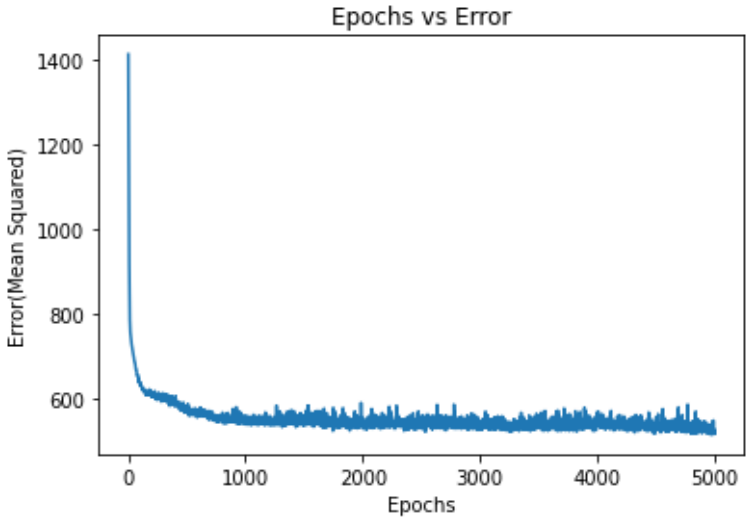
6. **With η= 0.01**

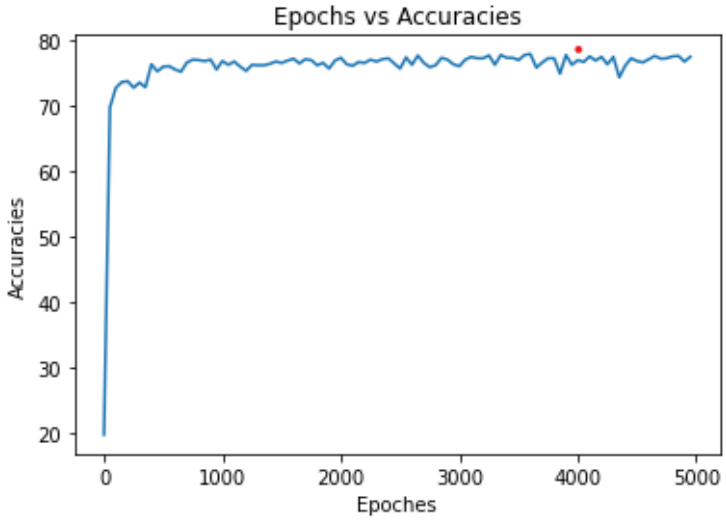
   _____TRAIN DATA_____
   Average Cost: 0.07359157273724484
   Average Accuracy: 71.87928571428569
   _____TEST DATA_____
   Average Accuracy: 76.962%

Plots

Two Layered Neural Network

1. With η= 0.9



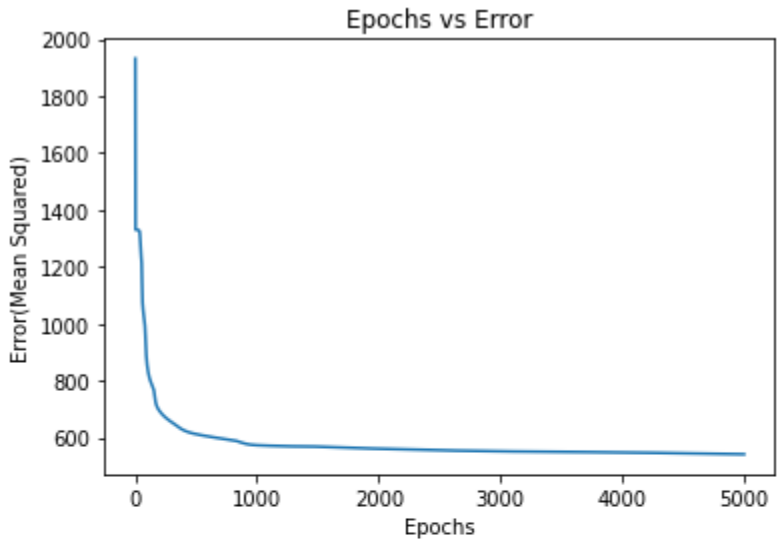


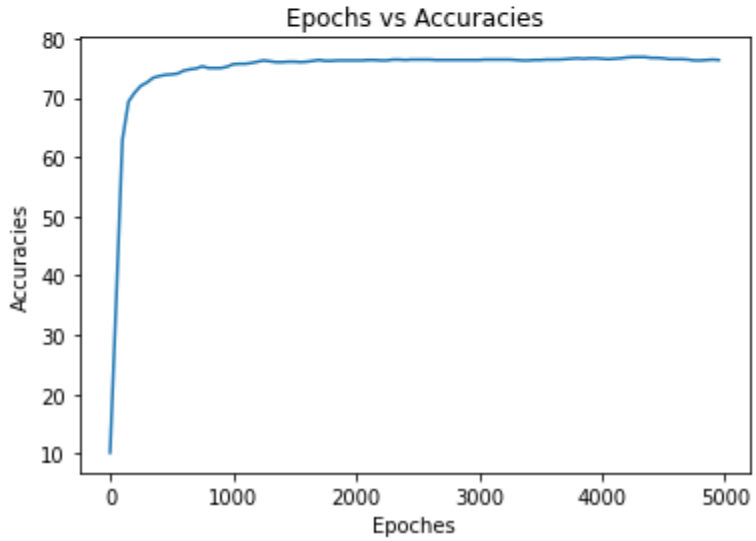
_____TRAIN DATA_____
Average Cost: 550.8346816406967
Average Accuracy: 75.9557142857143

_____TEST DATA_____

Average Accuracy: 75.793%

2. With η= 0.1





_____TRAIN DATA_____

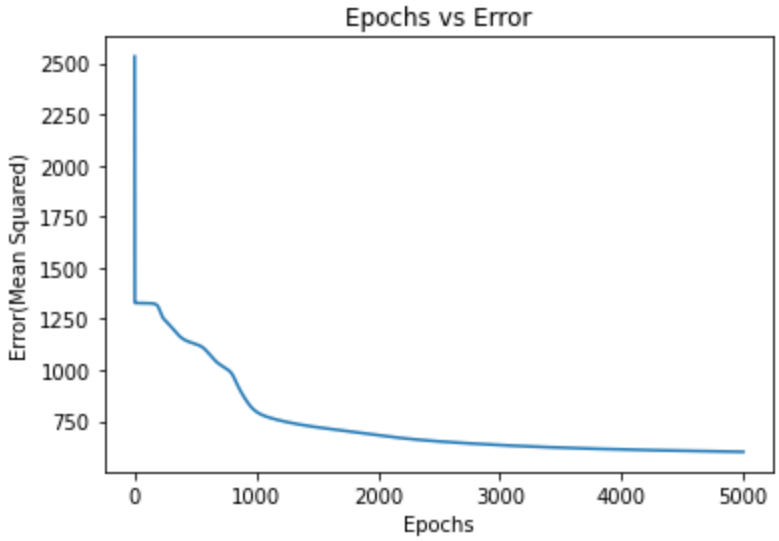
Average Cost: 580.1864723703909

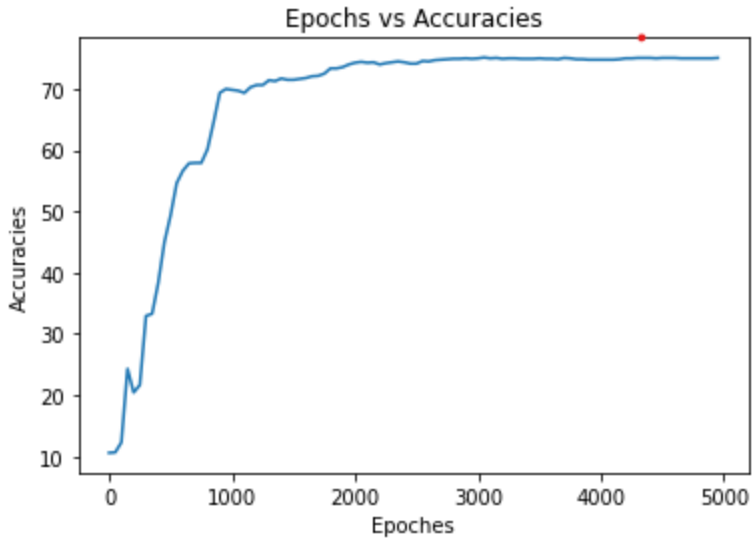
Average Accuracy: 74.71357142857146

_____TEST DATA_____

Average Accuracy: 74.624%

3. With η= 0.01





_____TRAIN DATA_____

Average Cost: 764.6568751939219

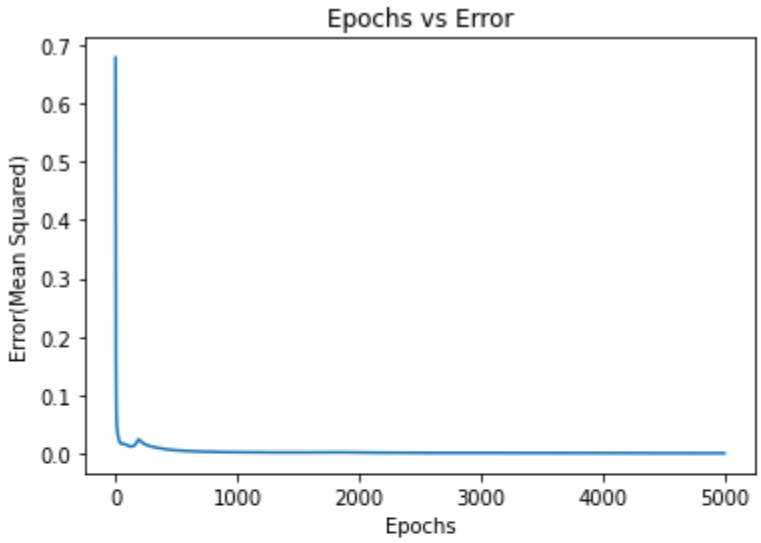
Average Accuracy: 66.71357142857147

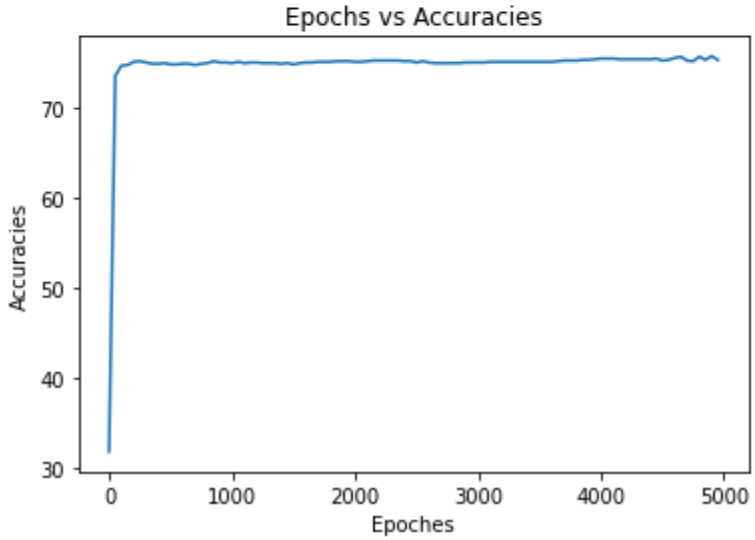
_____TEST DATA_____

Average Accuracy: 75.626%

One Layered Neural Network

4. With η= 0.9

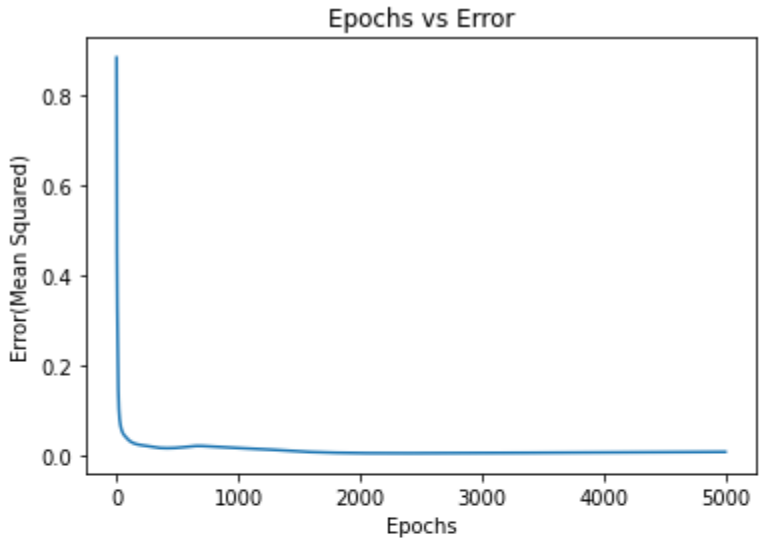


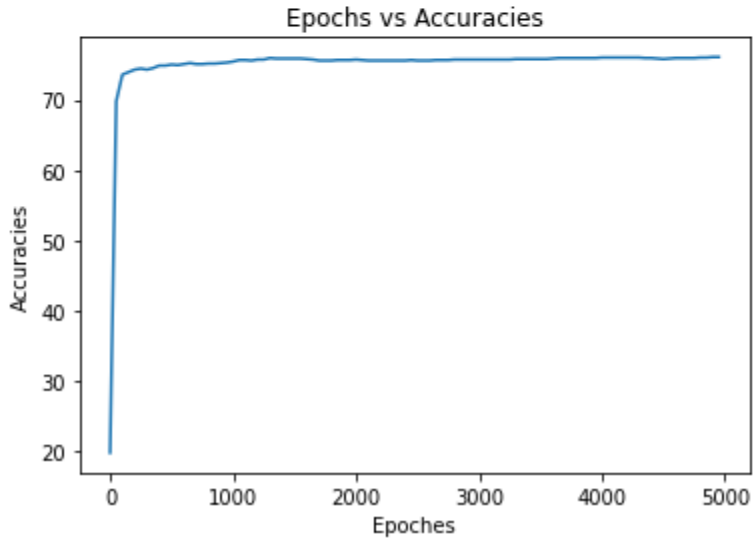


_____TRAIN DATA_____
Average Cost: 0.0027779495309844847
Average Accuracy: 74.7207142857143
_____TEST DATA_____

Average Accuracy: 76.294%

5. With η= 0.3





_____TRAIN DATA_____

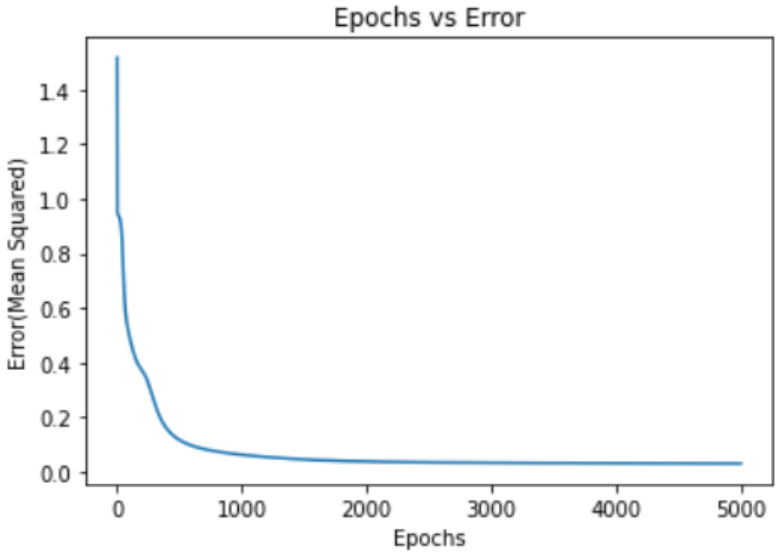
Average Cost: 0.013120242697323252

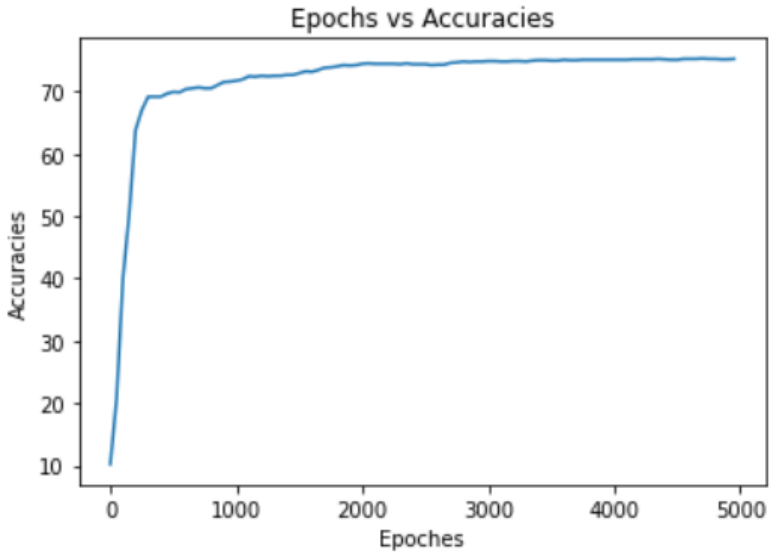
Average Accuracy: 74.97642857142857

_____TEST DATA_____

Average Accuracy: 75.960%

6. With η= 0.01





_____TRAIN DATA_____
Average Cost: 0.07359157273724484
Average Accuracy: 71.87928571428569
_____TEST DATA_____
Average Accuracy: 76.962%