Learning Activation In Neural Network

Initially, the goal of this project is to construct a simple machine learning model fro ground level, in this case Multi-Layer Perceptron (MLP), for classifying iris datasets.

Elements of a Neural Network :-
Input Layer :- This layer accepts input features. It provides information from the outside world to the network, no computation is performed at this layer, nodes here just pass on the information(features) to the hidden layer.
Hidden Layer :- Nodes of this layer are not exposed to the outer world, they are the part of the abstraction provided by any neural network. Hidden layer performs all sort of computation on the features entered through the input layer and transfer the result to the output layer.
Output Layer :- This layer bring up the information learned by the network to the outer world.

MLP Algorithm
Step 1: Set the weight and bias to a small-random value.

Step 2: Forward Propagation, which involves propagating all values from the input layer to the output layer.

Step 3: Using backward propagation from output to input, update the weight and bias in the inner layer.

Step 4: Iterate until the stop condition is met.

the thing left is to training the network. For training a neural network we need to have a loss function and every layer should have a feed-forward loop and backpropagation loop. Feedforward loop takes an input and generates output for making a prediction and backpropagation loop helps in training the model by adjusting weights in the layer to lower the output loss. In backpropagation, the weight update is done by using backpropagated gradients using the chain rule and optimized using an optimization algorithm. Backpropagation follows a series of steps to find the desired value for weights and bias

Now, Training the network by finding the actual and correct values

Final MLP Classifier summary
I finally trained my model for a total of 700 epoch and learning rate was set to 0.005

1. Input Layer: 4
2. Hidden Layer: 5
3. Output Layer: 3
4. Epoch: 700
5. Learning Rate: 0.005
6. Activation Function: sigmoid

Weight value of hidden layer during training

```
[[-1.2144479901761935, -1.0237235867746801, 1.1593402180504824, -0.9510205697571245, 0
.21282400398954074], [-1.828308534920255, -0.6174988029663058, 0.8822982250033891, 0.1
0538148610868076, 0.5623782407598591], [2.376995940983432, 2.673320503208607, -1.89468
08650670917, -0.22637674368743033, -0.9738675597590902], [0.8274442502250032, 0.544902
904567468, -0.4176102462987161, 0.6713496504613647, -0.263023627900034 86]]
```

Weight value of output layer during training

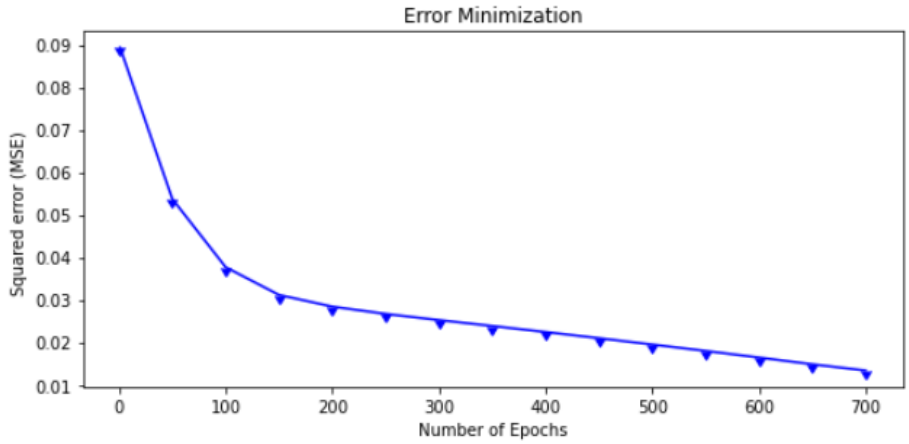
```
[[-1.7322627770013823, -2.9665545091908316, 2.091391511704845], [-3.4584624952157936,
1.6162436967706781, 0.9452808698855428], [1.6823684352932098, -0.12123161097583737, -3
.538001870802378], [-0.03755381210005162, 0.41622361402886393, 0.357845723366052 97], [
1.048241273174239, -1.8167493762545408, -1.6188648081621941]]
```

Predict Test Data using MLP Model

We can see that after every 50 epochs the error function is gradually decreasing to obtain the minimum value.

```
Epoch  1 - Total Error:  0.0893635715002401
Epoch  50 - Total Error:  0.053725282357467645
Epoch  100 - Total Error:  0.037678300466887256
Epoch  150 - Total Error:  0.031150650050135013
Epoch  200 - Total Error:  0.028400763121875304
Epoch  250 - Total Error:  0.026684640459743395
Epoch  300 - Total Error:  0.02523095496911937
Epoch  350 - Total Error:  0.02383533441803151
Epoch  400 - Total Error:  0.022422161169091452
Epoch  450 - Total Error:  0.02096794197543499
Epoch  500 - Total Error:  0.019480678845146756
Epoch  550 - Total Error:  0.017973271213314133
Epoch  600 - Total Error:  0.016422907059039912
Epoch  650 - Total Error:  0.014827759472965859
Epoch  700 - Total Error:  0.013362750473605812
```

Plotting the loss function vs. epochs graph



The better way to understand the accuracy is by implementing a confusion matrix. Here, for setosa we can see the precision, recall, f1 score is 1 for every label here and similarly for other two classification. Its is good habbit to look into f1 score as it gives the average value of precision and recall.

```
[[ 8.  0.  0.]
 [ 0. 15.  1.]
 [ 0.  0.  6.]]
   label      precision  recall  f1_score
   setosa         1.000  1.000   1.000
   versicolor     1.000  0.938   0.968
   virginica      0.857  1.000   0.923

   Average accuracy:   0.9666666666666667
   Average precision:  0.9523809523809524
   Average recall:     0.9791666666666666
   Average F1 score:   0.965588084232152
```

As we can see we have successfully trained a MLP which was written from scratch with high accuracy!