BT3041 – Analysis and Interpretation of Biological Data

Assignment

Submitted by Sahana G (BE17B038)

Question 1

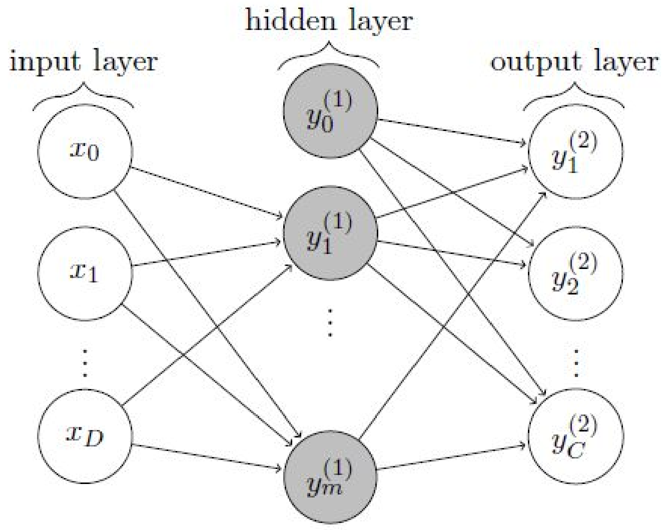
Implement Multilayer Perceptron for classifying the given MNIST dataset. The given dataset contains 800 training data and 200 testing data. Train the network and plot the loss and accuracy of the training and testing datasets.

Write a report on the following things:

- Network architecture
- Activation functions - Sigmoid, Softmax functions
- Loss function - Cross entropy loss
- Learning Algorithm – Stochastic gradient descent
- Learning Rate - (hyperparameter)
- Loss and accuracy for testing dataset.

Solution

Network architecture -



- Each image which is an input to the model is of size 28*28 = 784 pixels. The input layer of the network, therefore, comprises of 784 neurons, one for each pixel.
- There is only one hidden layer in the network. The hidden layer of the MLP network has 100 neurons.
- The hidden layer is then fed to the output layer, which classifies each input image as a number. Therefore, there are 10 neurons corresponding to numbers {0,1,2,...9} in the output layer.

The data fed to the model are as labelled - X_train = Images, Y_Train = Labels

Activation functions -

In the hidden layer of the perceptron, the "**Sigmoid**" activation function is used. This function is expressed as -

```
1/(1+ exp(-x))
h = sigmoid(x.W1 + b1)
```

In the output layer, "**Softmax**" activation function is used. This function helps in classifying an input into several categories. In our case, the input can be classified into any number {0,1,2,...9} with a certain associated probability. These probabilities will add up to 1. The entry with the maximum probability will correspond to the prediction by the network. This function is expressed as -

```
exp(x) / sum(exp(x))
Y = softmax(h.W2 + b2)
```

Loss function -

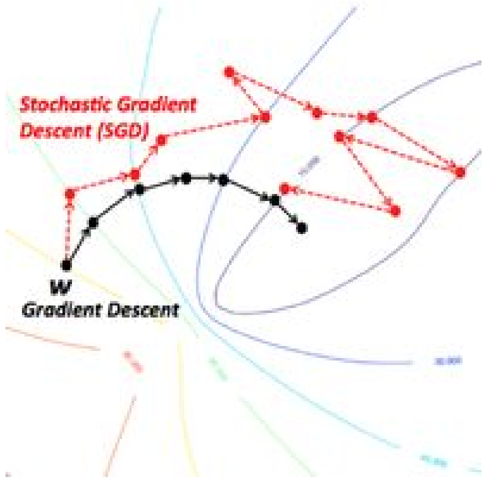
Cross-Entropy loss (CEL) is employed in the network. This loss function along with Stochastic gradient descent as the training function will simplify the mathematics later. We will need to compute the derivative of CEL with softmax function. That will yield a simple subtraction of prediction - expected outcomes. CEL function is expressed as -

$$CrossEntropyLoss = -\sum_i y_i log(p_i)$$

$$\frac{\partial Loss}{\partial o_i} = y^* - y = prediction - expected$$

Learning algorithm -

Stochastic Gradient descent is an iterative algorithm, that starts from a random point on a function and travels down its slope in steps until it reaches the lowest point of that function. SGD randomly picks one data point from the whole data set at each iteration to reduce the computations enormously instead of solving for each data point in every iteration, like in Gradient descent.



Learning rate -

Learning rate is a parameter that specifies how fast the network learns and updates its parameters. If the rate is too high, we may never converge on a good training error and if the rate is too low, the network learns very slowly which might extend computational time. In this model, the learning rate is set to a value of 0.001.

Loss and accuracy for testing dataset -

Note - In each epoch, after calculating the corresponding weights and biases with respect to training_datapoints, predictions are made for both training and testing data to calculate the loss and accuracy of both training and testing data.

Loss is calculated as the summation of all the cross-entropy losses upon the length of test_data samples.

```
test_loss = loss/len(X_test)
```

Test_Loss = 0.19994511050731906

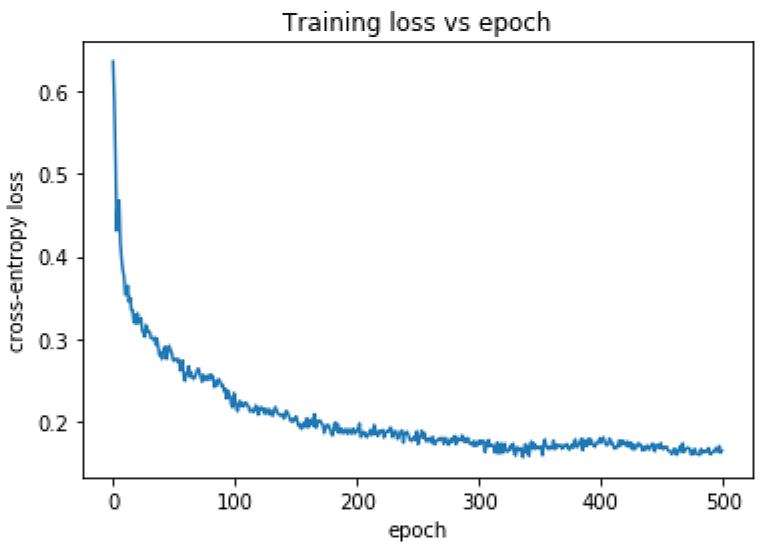
Once each prediction has been made with the calculated weights and biases as parameters, the predicted label (a number between 0 and 9) is compared with the Y_label of the test data. The number of correct predictions upon the total length of predictions made, i.e., the length of test-data samples, is returned as the accuracy.

```
test_accuracy = float(correct_predictions)/float(len(Y_test))
```

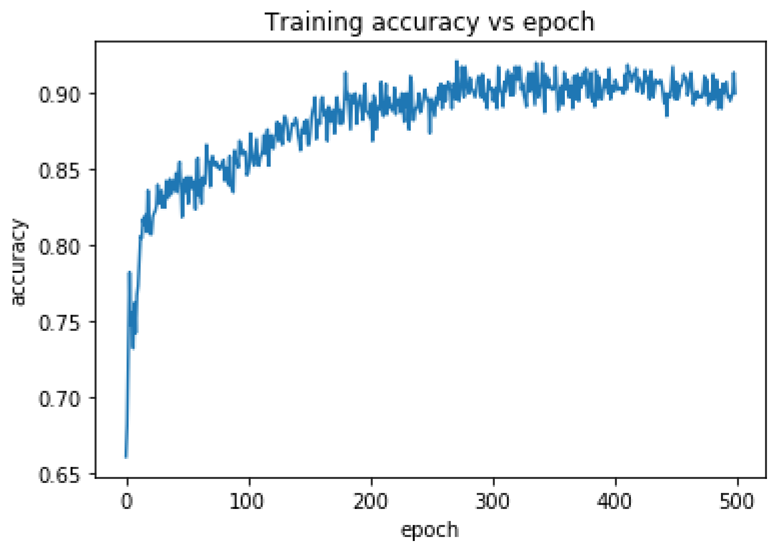
Test_Accuracy = 0.75

Plots -

Training loss vs epochs



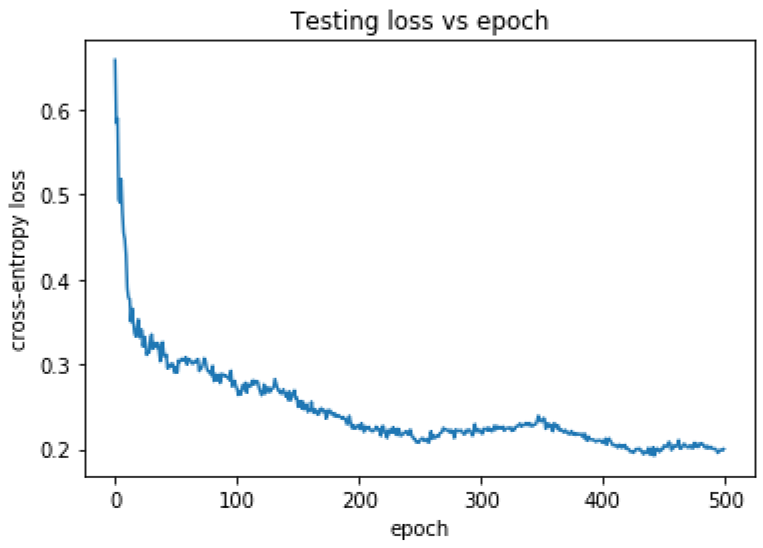
Training accuracy vs epochs



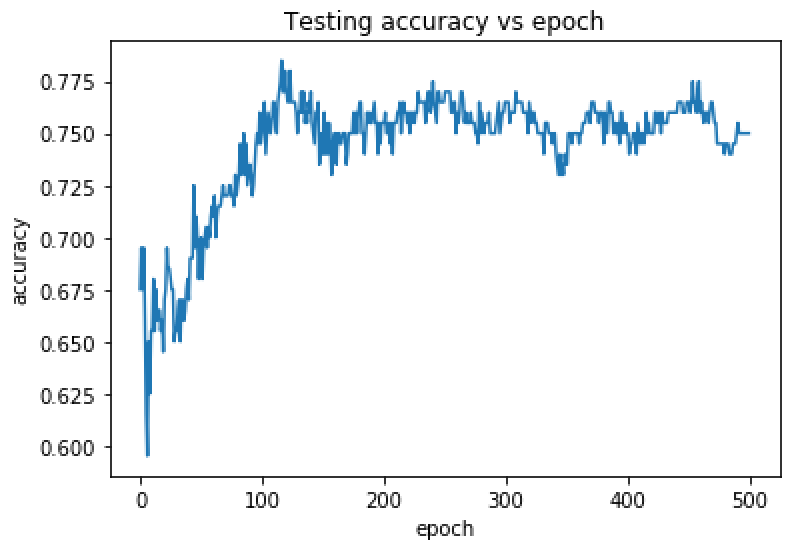
Training_accuracy value = 0.9 and Training_loss value = 0.16544

Inference - The training accuracy has been steadily increasing with respect to epochs, reaching a value of 0.9 at the end of training and the loss has also been decreasing with respect to the epochs, reaching a value of 0.16544.

Testing loss vs epochs



Testing accuracy vs epochs



Test_accuracy value = 0.75 and Test_loss value = 0.19995

Inference - Ignoring the high-frequency fluctuations, the accuracy plot is overall showing an increasing trend meaning that the model has not overfitted the training dataset yet. This is also shown by the test loss plot which is still decreasing after 500 epochs.

Question 2

Implement the DBSCAN algorithm to cluster the given 2000 data points. Report the right parameters to get four clusters and plot the clusters.

Solution

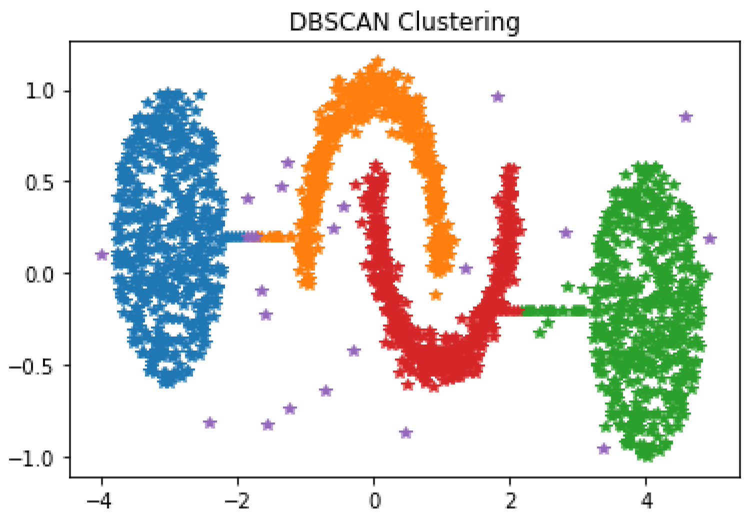
DBSCAN is a density-based clustering algorithm that categorically clusters data points based on a minimum threshold distance (**ε**) and a minimum number of points (Minpts) within eps. The abstract algorithm is as follows -

- Find the points at eps neighbourhood of every point and identify the core points that have more than Minpts in their neighbourhood.
- Find all the core points that are connected by being within the same eps neighbourhood from one-another, ignoring the non-core points.
- If the non-core point is in eps distance from a cluster, assign it to the cluster, else assign it as a noise point.

Various combinations of eps and Minpts can result in a specific number of clusters. In this case, as mentioned in the question, we'll need to identify 4 clusters. Through trial and error, it was identified that the following values result in 4 clusters.

- Eps - 0.15, Minpts - 5
- Eps - 0.19, Minpts - 10

The final clustering by the algorithm looks like this -



Cluster 1 - Blue, Cluster 2 - Orange, Cluster 3 - Red, Cluster 4 - Green, Noise - Purple