MLP Classification 2 classes

Banatwalla, Muhammad

March 21, 2021

# Introduction

For this homework, we will again be working with Multi-Layer Perceptrons. However, this time we will be using this MLP for classification of data into 3 classes. We will continue to work with a 3-layer MLP. To improve our MLP, we will use auto encoders as well as sparsity learning and evaluate the effects these have on performance of the MLP.

## Description of Data

For this assignment, we will get our data from the ”University of California Irvine Repository for Machine Learning Datasets”. The data we are working with consists of digitized images of typed characters. The fonts we chose are Century, Ebrima, and Gill. Before treatment of the data, the data had the following sizes: Century - 7994, Ebrima - 6892, Gill 5836. After selecting only the cases that are italic and with strength = .4, like we did in HW 2, they had the new following sizes: Century - 1999, Ebrima - 1723, Gill - 1459. The data consists of 400 features corresponding to a series of gray level image intensity values for different pixel positions. For this report, Century will correspond to Class 1, Ebrima will correspond to Class 2 and Gill will correspond to Class 3. Class 1, Class 2, and Class 3 were unioned into a larger data set we call DATA with size 5181. Further, we then standardized this data.

## PCA

We used PCA to get our values for the lower and upper bound of the hidden layer sizes. To get the lower bound, we ran PCA on the entire data set to compute the the number of principal components for 95% explained variance. Below is the plot for the number of components versus explained variance:

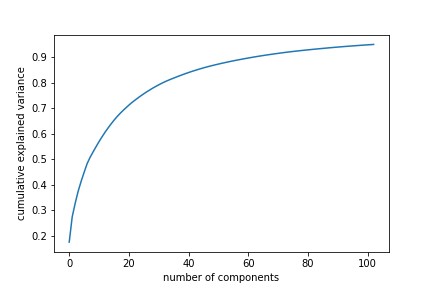


Figure 1: No. of Principal Components v. Cumulative Explained Variance

By principal component 102, we have reached 95% explained variance and therefore will choose h = 102 as the minimum for our h values.

To calculate the upper bound, we ran PCA on each class separately to get 3 values. This time, we computed the number of principal components for 99% explained variance. Below is the plot for the number of components versus explained variance:

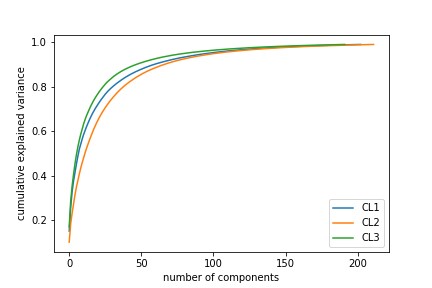


Figure 2: No. of Principal Components v. Cumulative Explained Variance

For class 1, by principal component 203, we reached 99% explained variance. For class 2, by principal component 212, we have reached 99% explained variance. For class 3, by principal component 192, we have reached 99% explained variance. To compute the upper bound for the hidden layer size we add these 3 values of principal components to get 607 as the max hidden layer size.

# Multi Layer Perceptrons

## Training MLP0

For MLP0, we will use a 3-layer MLP with a hidden layer size of 103 and an output size of 3 to represent the 3 classes. We will use cross entropy as our loss function, Adam as our optimizer and softmax as our optimization. We use a batch size of 25 and 150 epochs. Below, we visualize the loss value for train and test for each epoch and determine the ideal value for Tstop.

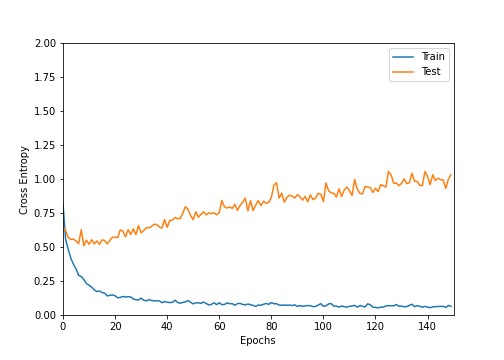


Figure 3: Epoch v. Loss

From this graph, we visually estimate values for stabT and overT. stabT is the time of stabilization, we choose an epoch value of 10 for stabT as this looks to be where the loss value begins to stabilize for the test set. overT should be the time that the test set begins to increase again, we chose an epoch value of 20 for overT. Then to get Tstop we get the minimum loss value in between stabT and overT. This minimum value was reached an epoch 14, this is our Tstop value.

Below, we will share the confusion matrices for the train and test set of MLP0:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL |
| Actual: CENTURY | 98% | 1% | 1% |
| Actual: EBRIMA | 2% | 97% | 1% |
| Actual: GILL | 1% | 0% | 99% |

Table 1: Confusion Matrix for MLP0 (*TrainingData*)

Overall, for the training data, the performance was nearly perfect for each font. The overall accuracy reached was 98%.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL |
| Actual: CENTURY | 87% | 10% | 3% |
| Actual: EBRIMA | 13% | 80% | 7% |
| Actual: GILL | 10% | 8% | 82% |

Table 2: Confusion Matrix for MLP0 (*TestingData*)

The testing set had an overall accuracy of 83%.In the testing data, for Ebrima, 13% true class Ebrima were wrongly predicted as century by our MLP. Also, 10% of true class Gill were wrongly predicted as Century and 10% of true class Century were wrongly predicted as Ebrima. These misclassifications had the largest effects on the accuracy for the test set.

## Training MLP\*

For MLP\*, we will use a 3-layer MLP with a hidden layer size of 607 and an output size of 3 to represent the 3 classes. We will use cross entropy as our loss function, Adam as our optimizer and softmax as our optimization. We use a batch size of 25 and 150 epochs. To find the best MLP we compared the performance of our MLP at sparsity level 10% and 20%. Sparsity level means that on average you have only that percent of neurons in the hidden layer that are active.

Below, we visualize the loss value for train and test for each epoch and determine the ideal value for Tstop for MLP\* at a sparsity level of 10%:

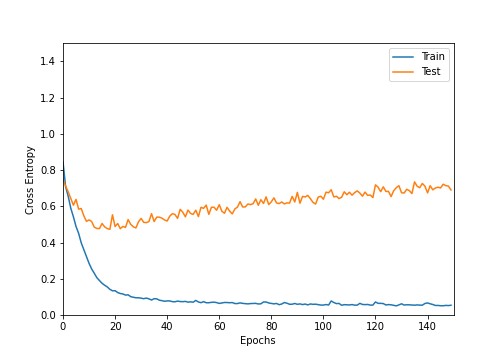


Figure 4: Epoch v. Loss

From this graph, we visually estimate values for stabT and overT. stabT is the time of stabilization, we choose an epoch value of 15 for stabT as this looks to be where the loss value begins to stabilize for the test set. overT should be the time that the test set begins to increase again, we chose an epoch value of 35 for overT. Then to get Tstop we get the minimum loss value in between stabT and overT. This minimum value was reached an epoch 18, this is our Tstop value.

Below, we will share the confusion matrices and confidence intervals for the train and test set of MLP\* at sparsity level 10%:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL |
| Actual: CENTURY | 100% | 0% | 0% |
| Actual: EBRIMA | 2% | 98% | 0% |
| Actual: GILL | 4% | 1% | 95% |

Table 3: Confusion Matrix for MLP\* Sparsity 10% (*TrainingData*)

Overall, for the training data, the performance was nearly perfect for each font. The overall accuracy reached was 97%.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL |
| Actual: CENTURY | 90% | 8% | 2% |
| Actual: EBRIMA | 15% | 78% | 7% |
| Actual: GILL | 10% | 7% | 83% |

Table 4: Confusion Matrix for MLP\* Sparsity 10% (*TestingData*)

The testing set had an overall accuracy of 83%.In the testing data, for Ebrima, 15% true class Ebrima were wrongly predicted as century by our MLP. Also, 10% of true class Gill were wrongly predicted as Century and 8% of true class Century were wrongly predicted as Ebrima. These misclassifications had the largest effects on the accuracy for the test set.

Below, we visualize the loss value for train and test for each epoch and determine the ideal value for Tstop for MLP\* at a sparsity level of 20%:

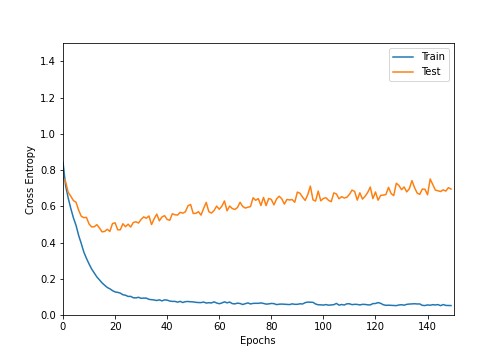


Figure 5: Epoch v. Loss

From this graph, we visually estimate values for stabT and overT. stabT is the time of stabilization, we choose an epoch value of 10 for stabT as this looks to be where the loss value begins to stabilize for the test set. overT should be the time that the test set begins to increase again, we chose an epoch value of 25 for overT. Then to get Tstop we get the minimum loss value in between stabT and overT. This minimum value was reached an epoch 15, this is our Tstop value.

Below, we will share the confusion matrices for the train and test set of MLP\* at sparsity level 20%:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL |
| Actual: CENTURY | 96% | 0% | 4% |
| Actual: EBRIMA | 0% | 98% | 2% |
| Actual: GILL | 0% | 0% | 100% |

Table 5: Confusion Matrix for MLP\* Sparsity 20% (*TrainingData*)

Overall, for the training data, the performance was nearly perfect for each font. The overall accuracy reached was 98%.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL |
| Actual: CENTURY | 87% | 7% | 6% |
| Actual: EBRIMA | 10% | 81% | 9% |
| Actual: GILL | 6% | 7% | 87% |

Table 6: Confusion Matrix for MLP\* Sparsity 20% (*TestingData*)

The testing set had an overall accuracy of 85%.In the testing data, for Ebrima, 10% true class Ebrima were wrongly predicted as century by our MLP. Also, 7% of true class Gill were wrongly predicted as Ebrima and 7% of true class Century were wrongly predicted as Ebrima. These misclassifications had the largest effects on the accuracy for the test set.

To compare performance between the sparsity levels we will look at the 95% confidence intervals for the test set:

|  |  |  |
| --- | --- | --- |
|  | Sparsity 10% Test | Sparsity 20% Test |
| Overall Accuracy | (82%*,*86%) | (82%*,*87%) |
| CENTURY | (86%*,*92%) | (73%*,*82%) |
| EBRIMA | (75%*,*83%) | (78%*,*86%) |
| GILL | (79%*,*87%) | (80%*,*89%) |

Table 7: Confidence Intervals for Accuracy in MLP\*

All of these intervals overlap except for the century confidence interval, for sparsity 10% performed better. However, overall the performance of MLP\* at sparsity level 10% was better and therefore this is what we will use as our best MLP\*.

The empirical sparsity can be best evaluated by looking at the percentage of neurons in the hidden layer that are less than the (average neuron activity)/2 (per 1/2) and percentage of neurons in the hidden layer that are less than the (average neuron activity)/2 (per 1/3). For our best MLP\*, per 1/2 = .8% and 1/3 = .2%.

**2.2.1 Neuron Activity**

We will look at the average activity ACTn on hidden layer for our best MLP\*. Below is the histogram for these activities:

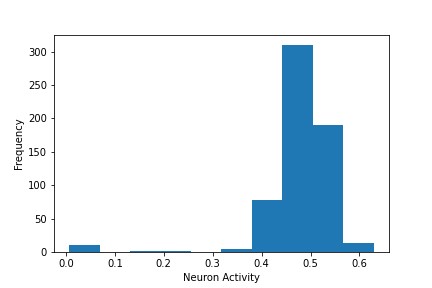


Figure 6: Neuron Activity Histogram

The mean of this histogram is .48, while the standard deviation is .07. Visually, we can see that this histogram is skewed to the left and most values lie around the mean, with a small group of neurons having close to 0 activity.

## Training MLP\*\* with AutoEncoder

Autoencoders are a specific type of feedforward neural networks that can be used to learn a compressed representation of raw data. The input is the same as the output. They compress the input into a lower-dimensional code and then reconstruct the output from this representation. The code is a compact “summary” or “compression” of the input. An autoencoder consists of 3 components: encoder, code and decoder. The encoder compresses the input and produces the code, the decoder then reconstructs the input only using this code. Autoencoders are trained the same way as ANNs via backpropagation. After training, the encoder model is saved and the decoder is discarded. Autoencoders are different from a standard data compression algorithm because they are data specific. Autoencoders can only compress data similar to what they have been trained on since they learn features specific for the given training data. In this project we used the autoencoder as a classifier.

To begin the building of our autoencoder, we performed PCA on the hidden layer Zn from our MLP\* model to find the number of principal components that have 95% explained variance.

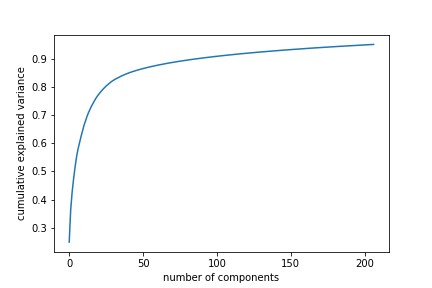


Figure 7: No. of Principal Components v. Cumulative Explained Variance

By principal component 207, we have reached 95% explained variance and therefore S = 207.

We first train the autoencoder using all of Zn as the inputs, with a hidden layer size of S and an output layer size of h\* = 607. For this model, we use the mean squared error as the loss function. For this autoencoder, we evaluate performance by calculating RMSE/ average (—— Zn——). For RMSE we got a value of 0.08, our rmse is close to zero, so it implies that our model is fitting the data really well, however since it is so small we are afraid that our model might be overfitting. The value we got for RMSE/ average (—— Zn——) was .17.

Now to begin building MLP\*\*, we stitch together the first two layers of MLP\* with the last two layers of the autoencoder. We now have a 3-layer MLP with an input layer Xn of size 400, a hidden layer Zn of size 607 and an output layer Kn of size 207. We then use Kn as the new inputs and train a short MLP with no hidden layer and an output of 3, using the cross entropy as the loss function. Now our final MLP\*\*\* has input layer Xn of size 400, hidden layer 1 of size 607, hidden layer 2 of size 207 and output of size 3. For this final MLP\*\*, we use the cross entropy loss function and softmax output. Below are the confusion matrices for MLP\*\*:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL |
| Actual: CENTURY | 100% | 0% | 0% |
| Actual: EBRIMA | 2% | 98% | 0% |
| Actual: GILL | 5% | 0% | 95% |

Table 8: Confusion Matrix for MLP\*\* (*TrainingData*)

Overall, for the training data, the performance was nearly perfect for each font. The overall accuracy reached was 98%.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predicted: CENTURY | Predicted: EBRIMA | Predicted: GILL |
| Actual: CENTURY | 87% | 7% | 6% |
| Actual: EBRIMA | 10% | 81% | 9% |
| Actual: GILL | 6% | 7% | 87% |

Table 9: Confusion Matrix for MLP\*\* (*TestingData*)

The testing set had an overall accuracy of 85%.In the testing data, for Ebrima, 10% true class Ebrima were wrongly predicted as century by our MLP. Also, 7% of true class Gill were wrongly predicted as Ebrima and 7% of true class Century were wrongly predicted as Ebrima. These misclassifications had the largest effects on the accuracy for the test set.

## Finding Best MLP

To determine which of our models performed the best, we will compare the 95% confidence intervals of the overall accuracy for the test set:

|  |  |  |  |
| --- | --- | --- | --- |
|  | MLP0 | MLP\* | MLP\*\* |
| Overall Accuracy Train | (97%*,*98%) | (98%*,*99%) | (98%*,*100%) |
| Overall Accuracy Test | (81%*,*85%) | (82%*,*87%) | (82%*,*86%) |

Table 10: Confidence Intervals for Accuracy

Because all of these intervals overlap, we will also compare the actual accuracy values. The overall accuracy of all of the models for the train set was 98%. For MLP0, the overall accuracy of test set was 83%, while MLP\* and MLP\*\* both had an overall accuracy for test set of 85%. Because MLP\* and MLP\*\* had equal performance, we cannot determine which is best, this would require further testing of models.

# Conclusion and Further Suggestions

The purpose here was to use a series of gray level image intensity values for different pixel positions to classify them into three different fonts: century, ebrima, and gill. We achieved reasonable and similar accuracy using sparsity learning for mlp\* and using the autoencoder for mlp\*\*. To further improve accuracy we could try more sparsity levels on our mlp. Also, we could improve the Autoencoder by using it with Dropout and checking if the results change.