# MNIST DATASET FISHER LINEAR DISCRIMINANT AND LOGISTIC REGRESSION APPROACH

## INTRODUCTION

The purpose of this project is to understand how machine and deep learning algorithms work. For that, the MNIST handwritten digit dataset will be used and two algorithms will be implemented: A Fisher linear discriminant and a Logistic regression with a neural network mindset.

This database consists of 60000 training images and 10000 testing images of handwritten digits from 0 to 9. The size of this images is 28x28 = 784 pixels.

In the next section, we will implement the fisher discriminant for the classification of two chosen digits: 0’s and 1’s and 5’s and 6’s.

## PART A - Fisher Discriminant

### 1. Feature collection

First, we will load the MNIST dataset to our repertory and separate the training and testing data. For the moment, we will leave the testing data unused and will focus on training data. That’s always a good habit because we do not want to know how testing data looks like and improve our model to have a better accuracy.

We have 6742 images of 1’s and 5923 images of 0’s. So we can say that our training set is quite balanced and we won’t have any biased class.

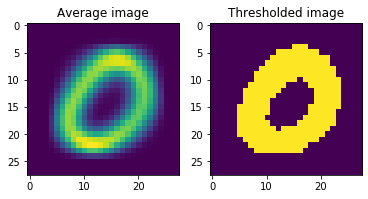
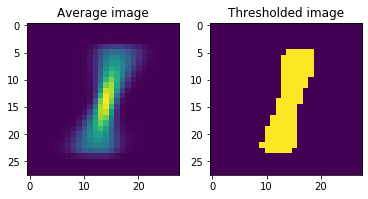
If we plot the average and the thresholded images of these two values:

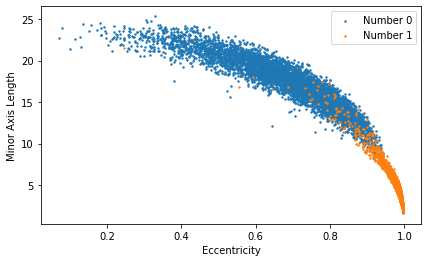
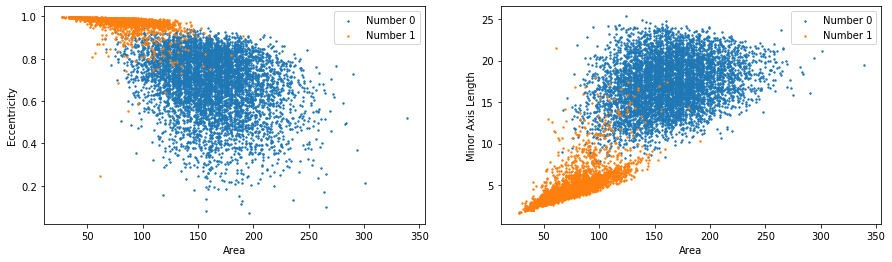
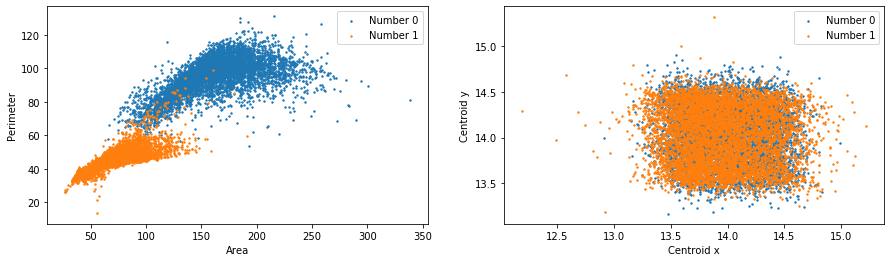
Figure 1. On the left, the average of all images of 0’s and the thresholded image. On the right same for the number one.

Both classes can be differentiated easily, let’s see if the algorithm can do it that well.

Now, we are going to decide which are better features to classify zeros and ones. Using measure.regionprops() function many features of images can be calculated. For the first approach, we are going to define the next five features: Area, perimeter, centroid, eccentricity and minor axis length.

On the next graph combinations of those features are plotted for both classes:

Figure 2 Scatter plots of different features for handwritten digits 0’s and 1’s



Note that except centroid, other combinations separate relatively well the two classes.

### 2. Classification of 0’s and 1’s on training data

First, we will use area and perimeter to classify 0’s and 1’s. Once we have the features of images, we will calculate the threshold that best discriminates our two classes. This is the histogram of the multiplication W\*X\_featuresT :

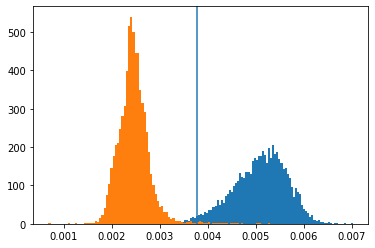


Figure 3 Histogram of the thresholded values

This are the results:

|  |  |  |
| --- | --- | --- |
|  | **Accuracy** | **Total errors** |
| **Predictions of 0’s** | 98.06% | 115 |
| **Predictions of 1’s** | 99.24% | 51 |
| **Total predictions** | 98.69% | 166 |

This model commits more errors predicting 0’s. However, accuracy is quite good and very few errors are committed totally.

### 3. Test set classification

On the test set, this are the accuracy results:

|  |  |  |
| --- | --- | --- |
|  | **Accuracy** | **Total errors** |
| **Predictions of 0’s** | 98.57% | 14 |
| **Predictions of 1’s** | 99.56% | 5 |
| **Total predictions** | 99.10% | 19 |

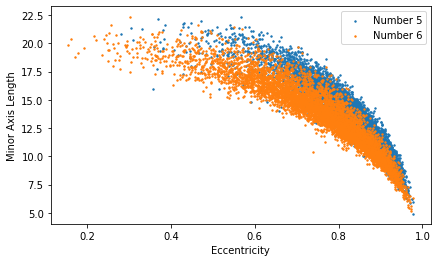
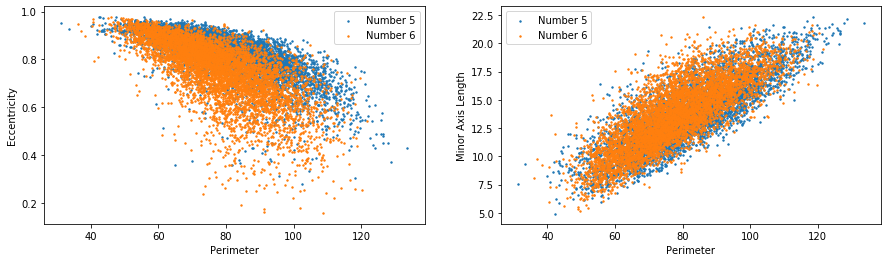
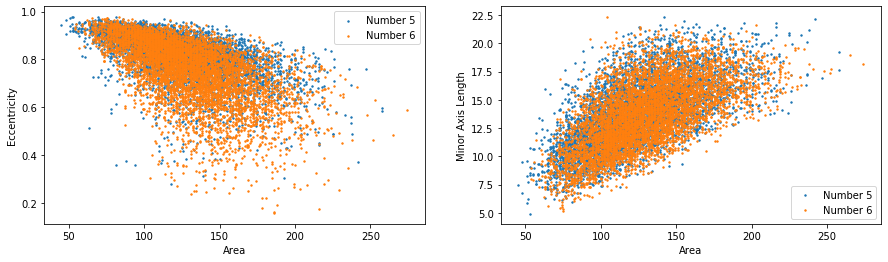
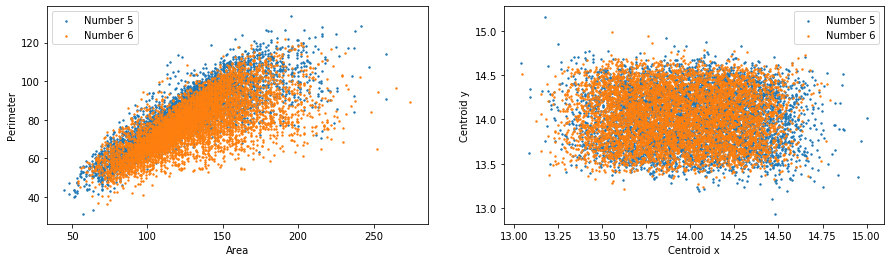
Very few errors are committed on the test set and this leads to a high accuracy.

### 4. Classification of 5’s and 6’s on training data

Next, we will deal with the problem of classification of 5’s and 6’s. This problem is quite harder since both numbers have a similar shape and doing it with a linear discriminant can be hard.

We will analyze how combinations of features used before can help in the classification. On the next figure, the mentioned seven combinations are shown:

Figure 4 Scatter plots of different features for handwritten digits 5’s and 6’s



This time, both classes have very similar features and it will be harder to classify them.

In order to compare this problem with the previous one, we will use the same features to train the model: area and perimeter.

With this selection, on the next figure the threshold is not able to separate two classes because they are overlapping each other.

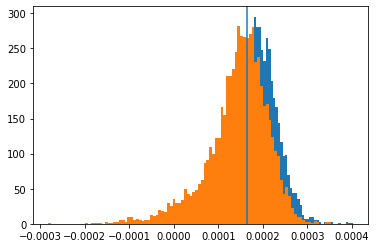


Figure 5 Histogram of the thresholded values

The accuracy of the obtained results is obviously lower than the previous one:

|  |  |  |
| --- | --- | --- |
|  | **Accuracy** | **Total errors** |
| **Predictions of 5’s** | 70.25% | 1613 |
| **Predictions of 6’s** | 58.74% | 2442 |
| **Total predictions** | 64.24% | 4055 |

### 5. Algorithm testing

Since training accuracy is quite low, we do not expect testing accuracy being higher.

|  |  |  |
| --- | --- | --- |
|  | **Accuracy** | **Total errors** |
| **Predictions of 5’s** | 67.04% | 294 |
| **Predictions of 6’s** | 58.66% | 396 |
| **Total predictions** | 62.70% | 690 |

### 6. Model improvement

In order to improve these results, other features will be selected to train the model. If we analyze the scatter plots in figure 4, we can see that the combination of minor axis length and eccentricity is the one that best separates both classes.

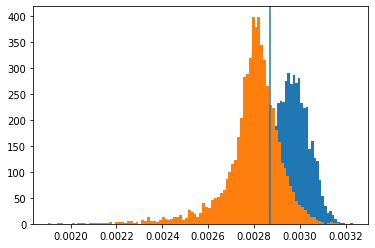
Using those two features we get this histogram:

Figure 6 Histogram of the improved model

Separation between classes is higher than before but still is worse than the case of 0’s and 1’s. The accuracy results, obviously, are also better in this case. For the training set estimation accuracy:

|  |  |  |
| --- | --- | --- |
|  | **Accuracy** | **Total errors** |
| **Predictions of 5’s** | 80.02% | 1083 |
| **Predictions of 6’s** | 76.25% | 1405 |
| **Total predictions** | 78.05% | 2488 |

Regarding to the testing accuracy we have that:

|  |  |  |
| --- | --- | --- |
|  | **Accuracy** | **Total errors** |
| **Predictions of 5’s** | 83.18% | 150 |
| **Predictions of 6’s** | 77.77% | 213 |
| **Total predictions** | 80.38% | 363 |

### 7. Results and discussion

Results in classification of 0’s and 1’s were more accurate than in the classification of 5’s and 6’s as expected. On the first classification problem, combinations of two features can easily classify the two classes but on the second problem, we can see that both classes have similar characteristics and that makes it difficult to distinguish 5’s and 6’s.

Regarding to the training and testing accuracy, there is no significant differences between both datasets. On the first case, training prediction accuracy is 0.987 and the prediction accuracy in the testing dataset is even higher: 0.991. This means that our threshold value can predict well the testing data and note that we did not use that data to train the model.

On the second case, training prediction accuracy is 0.64 and the testing accuracy of 0.627. This accuracy is way lower than the other case due to the reasons discussed before.

We saw that with minor axis length and eccentricity as features, results are better than the ones of the first approach, accuracy improves both in training and in testing sets.

Other combinations were also tried, for example, eccentricity and perimeter give an accuracy around 0.71 on training set and 0.72 on testing. However, the highest accuracy obtained was with the model of the previous section.

## PART B - Logistic Regression with a Neural Network mindset

### 1. NN programing

In this section, we will implement a simple neural network of 784 inputs and one output with a sigmoid activation function. When a 28x28 image is analyzed and set as an input, each x(i) takes the value of one pixel of the image.

### 2. Data preprocessing

The real image is reshaped to a 1D array of 784 entries, so that it can be used as the input.

Then, flat images are normalized from 0-255 to 0-1. This is done because normalizing the data generally speeds up learning and leads to faster convergence. Next formula is used for normalization where Xmin = 0 and Xmax = 255.

Xchanged=X−Xmin/Xmax−Xmin

Other advantage that this technique have is that normalizing data makes the mean close to zero and it speeds up the training.

Instead of the sigmoid activation function using a tanh function would speed up even more. That’s because tanh function gives values between -1 to 1 while sigmoid function give values from 0 to 1.

### 3. Classification of 0’s and 1’s on training data

The first step after preparing the data is to initialize the weights and the scalar value that corresponds to the bias. The array w will be initialized with zeros as well as the bias scalar.

Then we will decide which will be the learning rate and the number of training iterations. We will begin with LR=0.005 and 2000 iterations.

After defining that, we will call the gradient\_descent function which is the main function on data training. Inside this function, 2000 iterations or epochs will be run. In each iteration it will follow these steps:

1. Forward pass. Beginning from the inputs, the cost is calculated with a sigmoid activation function.
2. Backward propagation. Here we calculate gradients dw and db differentiating the loss function with respect to w and b respectively.
3. Weights are updated with the negative gradient step direction and a step size defined by the learning rate. The cost is calculated to show it later on a graph. Then we repeat the first step until the defined iterations are completed.



### 4. Cost training vs iteration

The cost function is defined by this formula:

The next graph shows the cost plotted every 5 iterations. We can see that after starting with a value of 10 in a few iterations the cost is reduced to 2 and after 2000 iterations, the cost is around 0.088.

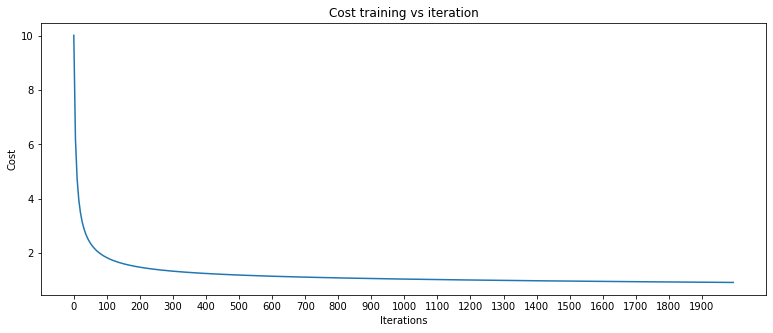
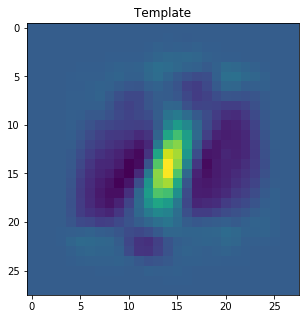


Figure 8 Heatmap of weights

Figure 7 Cost vs Iterations

Next template shows how the weights are distributed on the 28x28 image. We can see that in order to distinguish between two classes, central pixels have a yellow color while pixels on the surrounding circle have darker color. This seems reasonable in order to predict you have a 1 you should put weight to the central pixels and when predicting 0’s, surrounding pixels have more weight.

### 5. Training and testing accuracy

In this section we will analyze how well can the neural network predict the numbers in training and testing datasets.

Accuracy is measured by the following formula: mean(abs(y\_predicted – y\_real))

On the training dataset, accuracy was 99.874% and the loss was 0.088.

On the test set, accuracy was of 99.953%. This value is higher than the previous so we can ensure this model is not overfitted.

### 6. Classification of 5’s and 6’s on training data

For this problem we will follow the same steps as before but instead loading 0’s and 1’s, we will work with 5’s and 6’s.

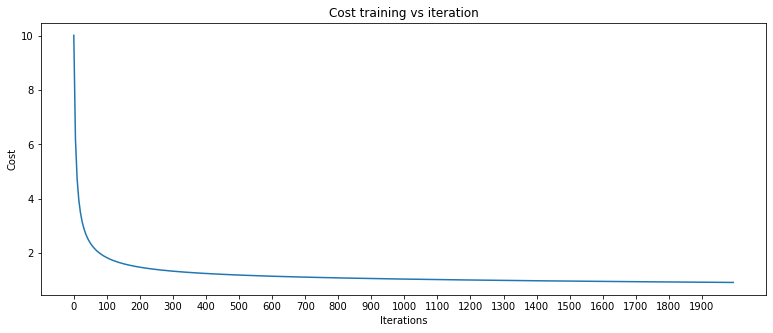
The following cost vs iteration graphic is similar to the previous one. Starting values are rapidly reduced and on the last iterations very reduced values are shown.

Figure 9 Cost vs Iteration

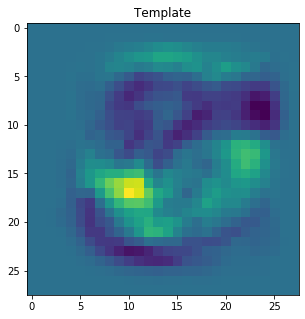
Note that if we analyze the template of the weights, this case is not as clear as the previous one. Yellow area shows crucial pixels to predict a 6 while dark blue area shows common pixels on 5’s

Figure 10 Heatmap of weights

### 7. Algorithm testing

Training accuracy is 97.88% and in the las epoch loss has a value of 0.907. Clearly, this are good results, however, they are worse than previous ones and this seems obvious since predicting 0’s and 1’s is easier.

On the test set, accuracy is 98.05%. Again, accuracy in testing set is higher than in the training set so the model is not overfitting the training data.

### 8. Comparison with Fisher discriminant

We have seen that classification of 0’s and 1’s is an easy task for both models and accuracy results obtained are good in both cases. However, when trying to classify 5’s and 6’s, those values have similarities on their features. That’s why linear discriminant model has difficulties to predict correctly.

On the other hand, the neural network model can maintain similar results for both classification problems. One of the reasons is that while Fisher’s discriminant only uses few features, we used only two but more can be used, the neural network takes into account every single pixel of the image.

### 9. Conclusion and future work

Due to the reduced maximum length of this project, some improvements were not implemented but some possible future work will be discussed on the next lines.

Regarding the Fisher’s Discriminant, in order to improve the accuracy, more than two features shall be used. That way, with a higher dimensionality, it may be easier to separate the two classes, however plotting the feature space is only possible with 3 or less dimensions.

In the Neural Network, we could try adding a hidden layer. This addition would increase the computational cost, but the accuracy could be also improved.

Other future work would be creating the model to classify any number from 0 to 9. On the neural network we should have 10 outputs to classify each class. On the other hand, the linear discriminant we have implemented can only separate two classes so the solution would be to create a multiclass discriminant. Alternatively, a one vs the rest technique could be implemented where one class is classified against every other class.