CSE 291 PATTERN RECOGNITION : ASSIGNMENT 1

Ayush Jasuja - A5310338
Nivetha Thiruverahan - A53205226

MNIST DATA

Description of methods :

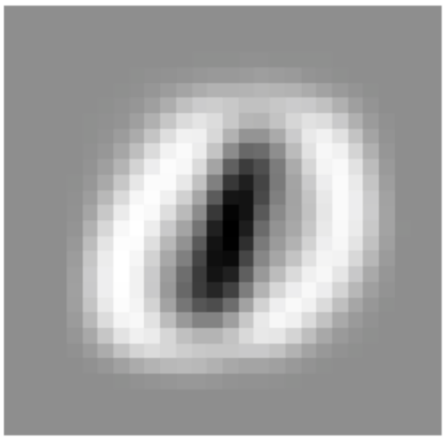
The MNIST dataset contains handwritten digit images that are 28*28 pixels and thus has a lot of features per image. Thus compression of the feature set is a crucial first step in order to run learning/classification algorithms on it in a cost effective manner without compromising accuracy too much. Also there are a lot of pixel values that have a zero variance and don't provide any information for classification. Hence compression algorithms such as PCA and LDA were applied to the training data and the reconstruction performance was observed by taking the most important eigen values that capture the maximum information.

Post this, the actual problem of classification of the handwritten image comes in. Two classifiers were studied and applied to the classification task, LDA and Naive Bayes with compressed (using PCA which seemed to work better in tandem with classification) as well as raw data. We considered the performance of both the algorithms on both versions of the dataset. A 'good' number of components to use when applying on compressed dataset was a decision made by either tuning it on a validation set or by using the most important eigen values capturing maximum information (80-90%) according to the results in the previous part.

Most important Eigen Vector and 20th Eigen vector :

PCA :

0-1 Classification :

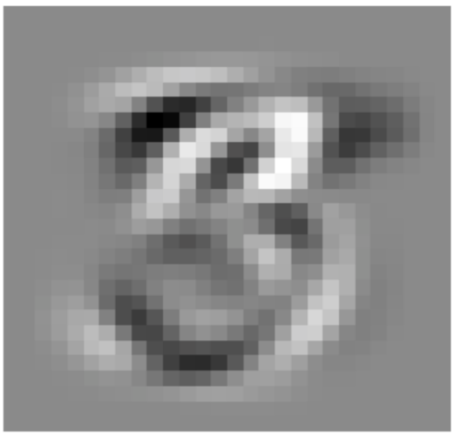


Most important

20th Eigen vector

3-5 Classification :

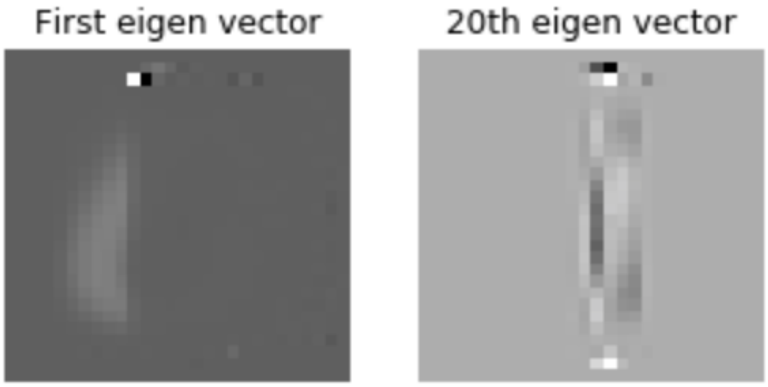
Most important



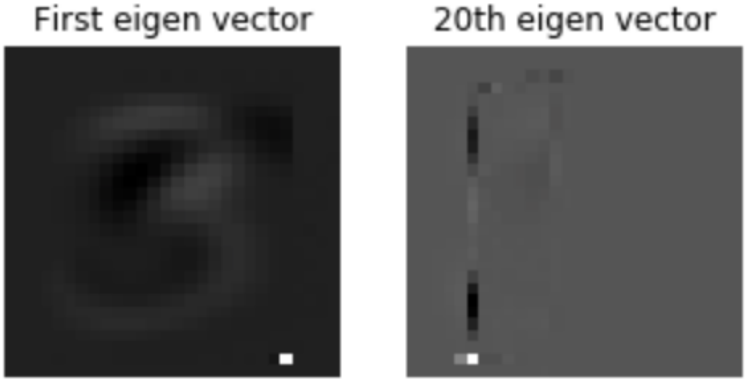
20th Eigen vector

LDA :

0-1 Classification



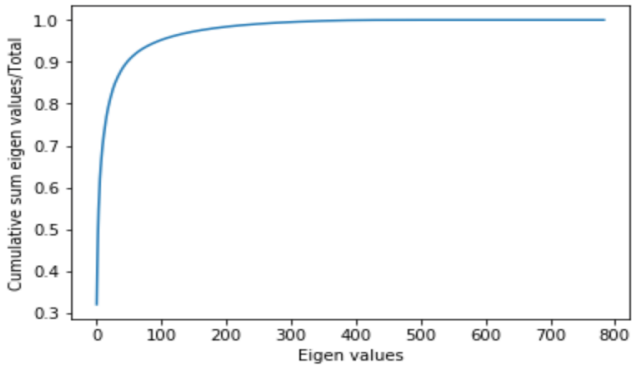
3-5 Classification

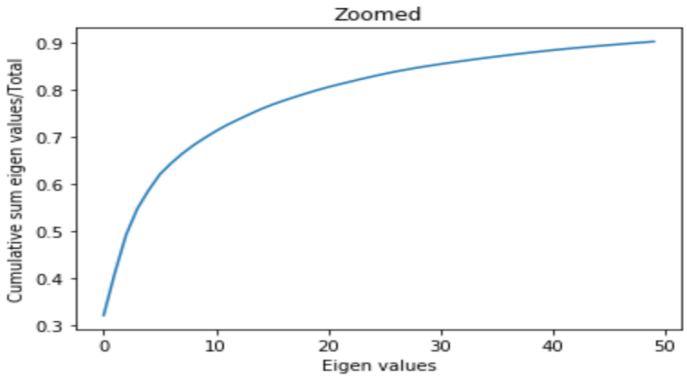


Plot of Cumulative Sum / Total sum across Eigen Values :

PCA :

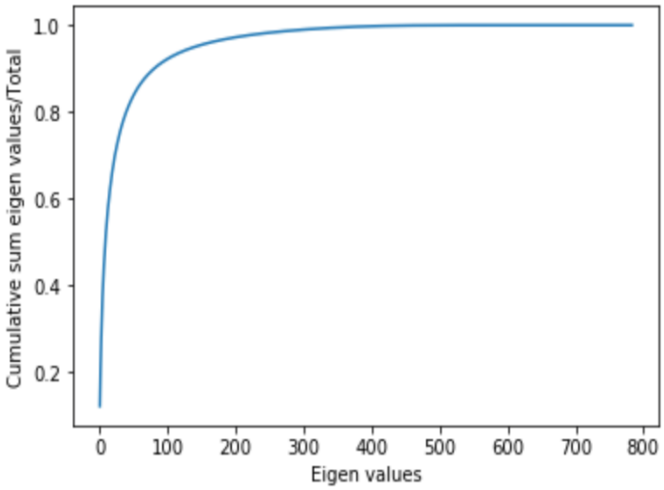
0-1 Classification :

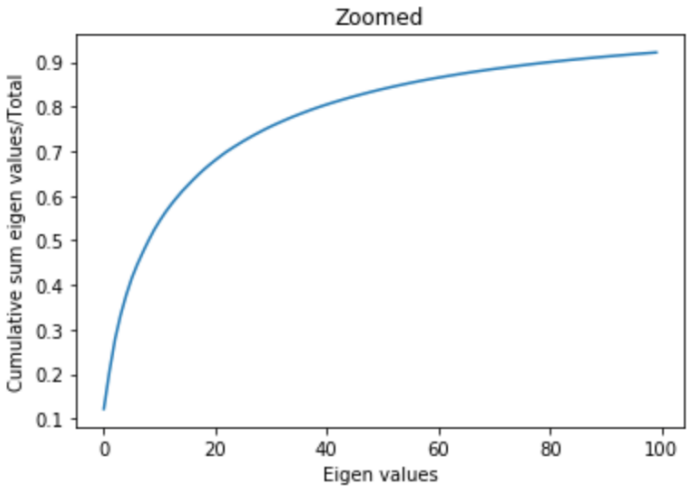




We can see that right from around 30 eigen values, the slope isn't changing too much and it tapers towards a plateau. Since 30 components hold 90% of the information, it should be enough to get an accurate reconstruction at the cost of including more components.

3-5 Classification :

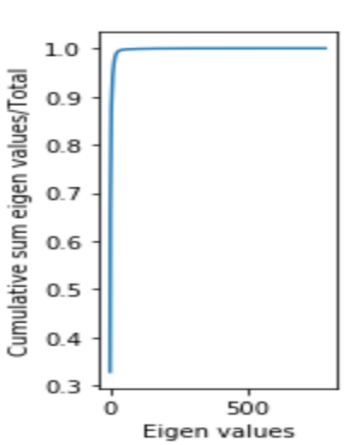


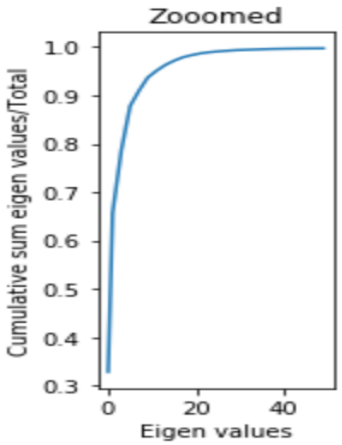


For the 3-5 case, only from around 100 eigen values, the slope isn't changing too much and it tapers towards a plateau. 100 components hold about 90% of the information and thus should be enough to get an accurate reconstuction at the cost of including more components.

LDA :

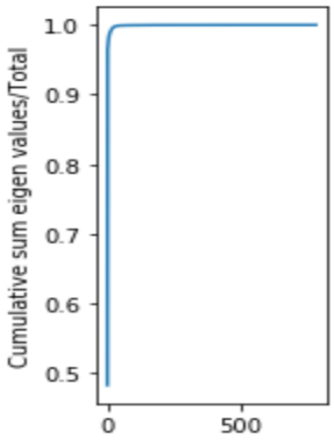
0-1 Classification :

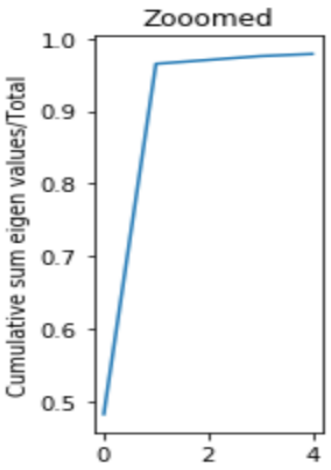




Here too, we can see that right from around 30 eigen values, the slope isn't changing too much and it tapers towards a plateau. 30 components hold more than 90% information and should be enough to get an accurate reconstuction at the cost of including more components.

3-5 Classification :





Here, we can see that just 1 component holds more than 90% of the information and seems sufficient to get an accurate reconstruction at the cost of adding more components.

Reconstruction of a test example :

PCA :

0-1 Classification ;

Based on the graph, we pick 30 components to do a reconstruction of a test example randomly chosen which is a zero. The left image shows the original and the right two images show the reconstruction with 30 components and the reconstruction error in terms of difference.

3-5 Classification :

Based on the graph, we pick 100 components to do a reconstruction of a test example randomly chosen which is a three. The left image shows the original and the right two images show the reconstruction with 100 components and the reconstruction error in terms of difference.

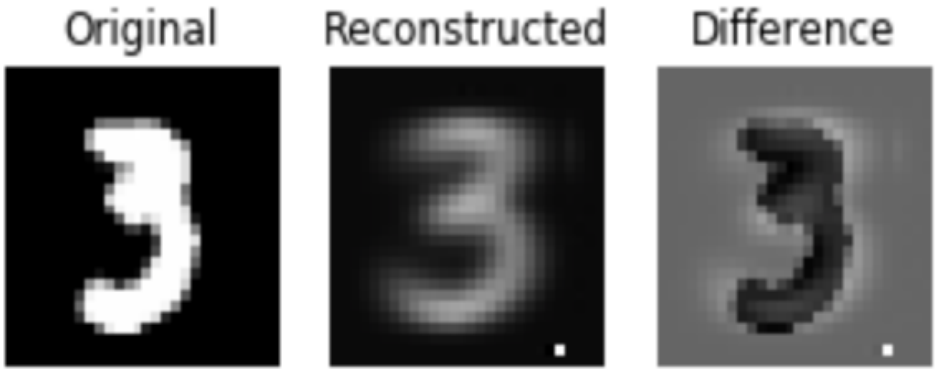
LDA :

0-1 Classification :

Based on the graph, we pick 100 components to do a reconstruction of a test example randomly chosen which is a three. The left image shows the original and the right two images show the reconstruction with 100 components and the reconstruction error in terms of difference.

3-5 Classification :

Based on the graph, we pick just 1 component to do a reconstruction of a test example randomly chosen which is a three. The left image shows the original and the right two images show the reconstruction with 1 component and the reconstruction error in terms of difference.



Classifier :

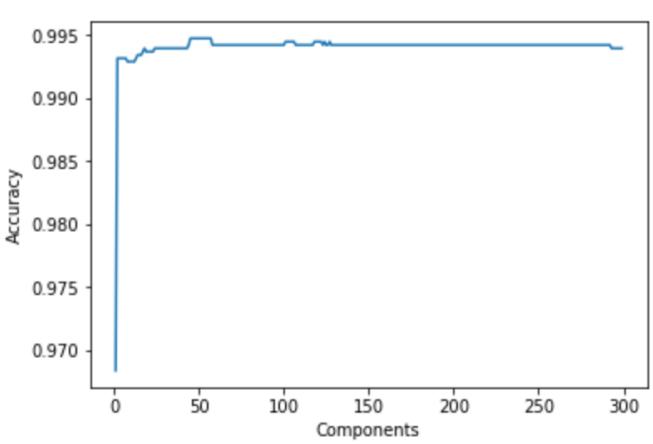
0-1 Classification :

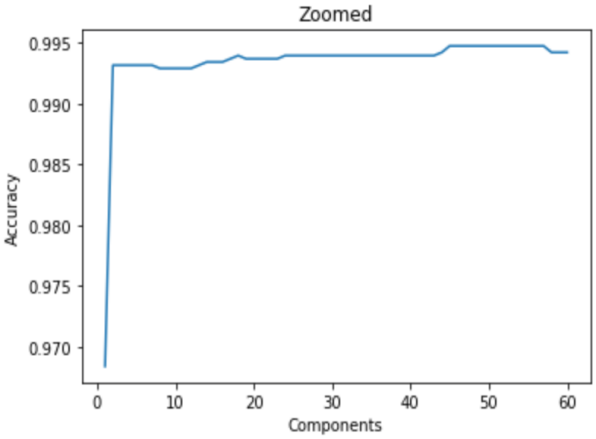
LDA :

We train the LDA classifier on 70% of the training data, using the remaining to tune the number of compinents that the LDA compressor should compress to and look at the accuracy for 1-300 components to figure out a good accuracy that can be afforded given the tradeoff of having more components.

First a sanity check is performed with all 784 components by comparing our implementation accuracy results to the scikit learn library's. The accuracies match fairly well with a difference of about 0.03%.

Following are the results of accuracy on the validation set plotted against the number of components that the data was compressed to by the LDA compressor and used by the classifier.



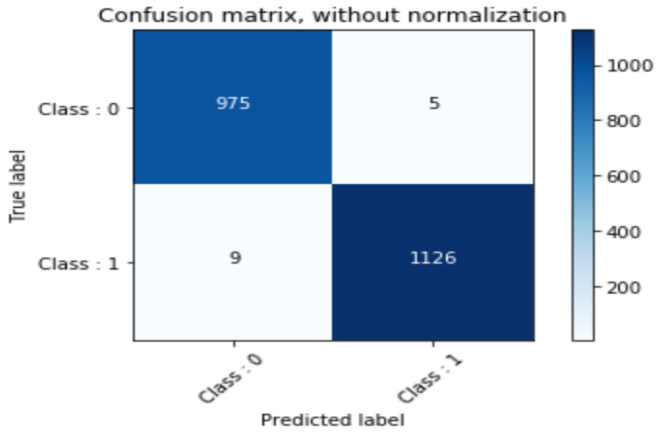


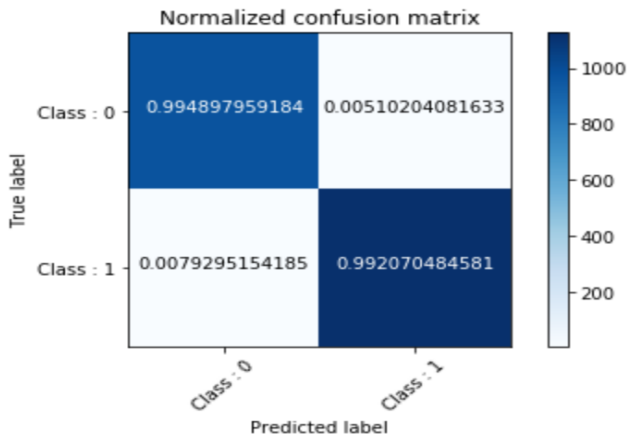
We can see that we have achieved a plateau at around 30 components and a good performance of around 99.395%. We can improve performance only slightly relative to the cost of adding more components in learning. This matches with the result of 30 components being enough to provide a good reconstruction as inferred from the cumulative sum proportion graph.

Thus the LDA classifier which uses 30 components derived from the LDA compressor is used to make predictions on the test set. An accuracy of 99.338% was achieved.

Confusion Matrix Results : LDA Classifier with 30 components.

```
Confusion matrix, without normalization
[[ 975    5]
 [   9 1126]]
Normalized confusion matrix
[[ 0.99489796  0.00510204]
 [ 0.00792952  0.99207048]]
```





Naive Bayes :

Let's take a look at the Naive Bayes with gaussian distributions classifier applied to the problem. We'll pass the raw data and use all components and also look at passing PCA compressed data to the classifier and compare performance in terms of accuracy achieved on the validation set.

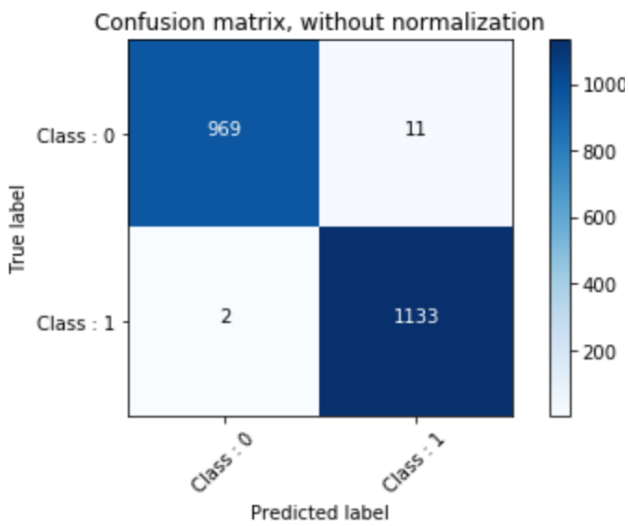
First a sanity check is performed with all 784 components by comparing our implementation accuracy results to the scikit learn library's. The accuracies match fairly well with a difference of < 1%.

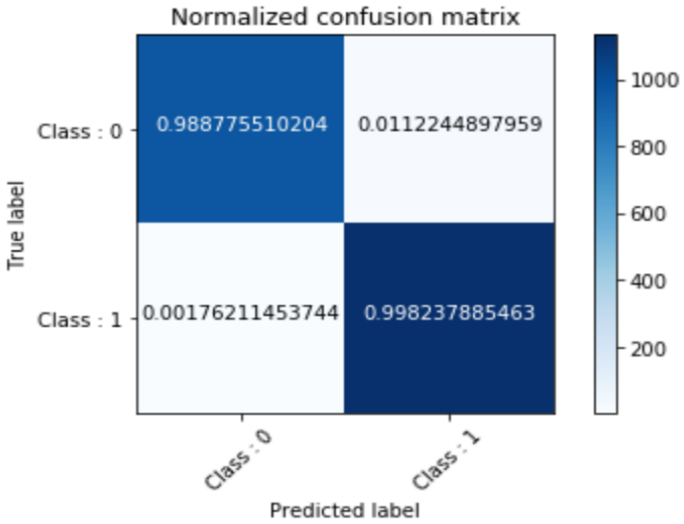
We believe the difference in accuracy is due to our simplifying assumption that variance across both classes is the same and usage of a class independent sigma in the algorithm. The library performs classification taking into account class specific sigma vectors.

Naive Bayes with all components gives an accuracy of 99.385% for the 0-1 classification problem.

Confusion Matrix Results :

```
Confusion matrix, without normalization
[[ 969   11]
 [   2 1133]]
Normalized confusion matrix
[[ 0.98877551  0.01122449]
 [ 0.00176211  0.99823789]]
```



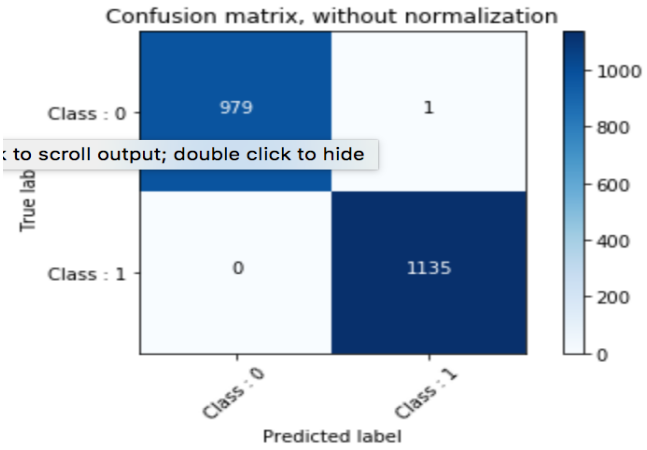


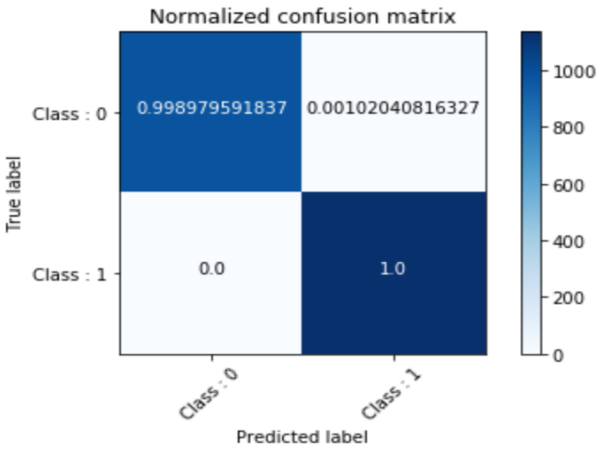
From the cumulative sum proportion graph we inferred 30 component compressed data from PCA seemed good enough for reconstruction. We passed this compressed data to the Naive Bayes Classifier and achieved a much better performance of 99.953%.

This leads to a conclusion that considering all components turned out to be 'unnecessary noise' that affected the performance of the classifier.

Confusion Matrix Results :

```
Confusion matrix, without normalization
[[ 979    1]
 [   0 1135]]
Normalized confusion matrix
[[ 0.99897959  0.00102041]
 [ 0.          1.        ]]
```



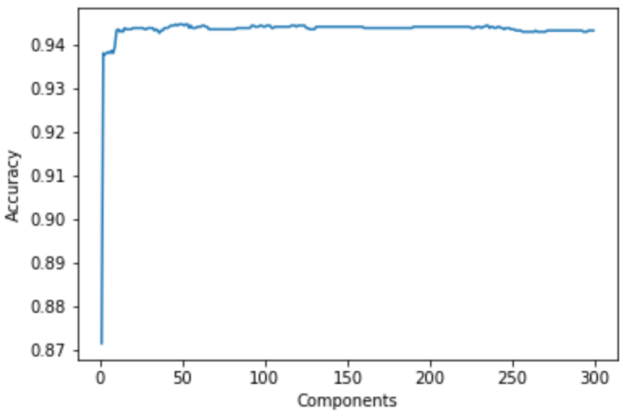


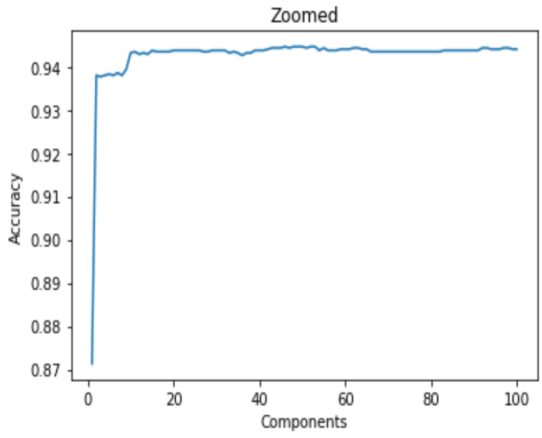
3-5 Classification :

LDA :

We train the LDA classifier on 70% of the training data, using the remaining to tune the number of compinents that the LDA compressor should compress to and look at the accuracy for 1-300 components to figure out a good accuracy that can be afforded given the tradeoff of having more components.

First a sanity check is performed with all 784 components by comparing our implementation accuracy results to the scikit learn library's. The accuracies match fairly well with a difference of about 0.03%.





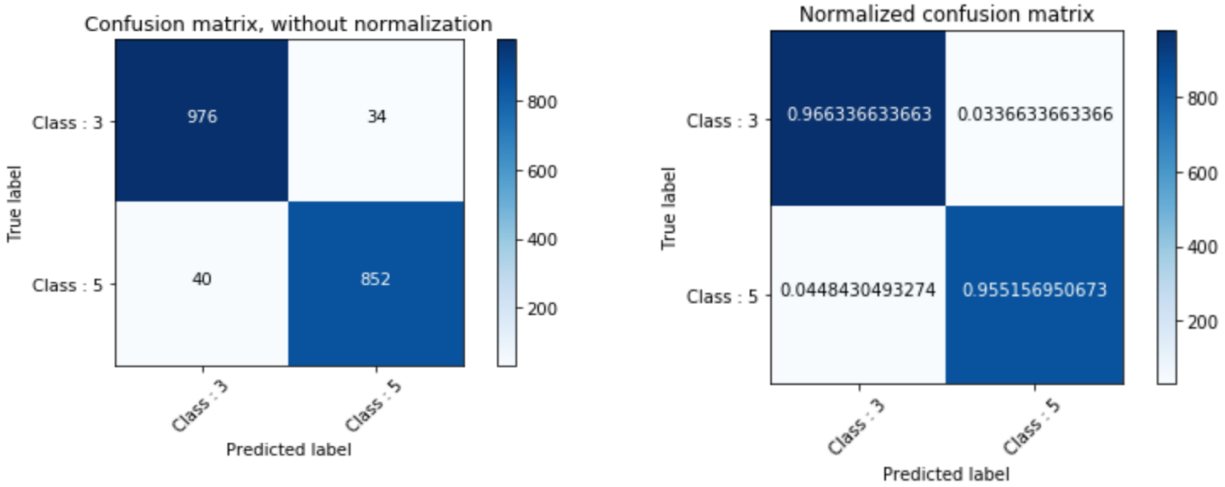
Above are the results of accuracy on the validation set plotted against the number of components that the data was compressed to by the LDA compressor and used by the classifier.

We can see that the accuracy has achieved a plateau at around 20 components and a performance of around 94.37%. We can improve performance only slightly relative to the cost of adding more components in learning. For example, choosing 100 components improves performance only by about 0.03%, giving 94.40%. Also taking just one component although provided good reconstruction, doesn't give a good accuracy measure when fed into the classifier (around 87.136%).

Thus the LDA classifier which uses 20 components derived from the LDA compressor is used to make predictions on the test set. An accuracy of 96.109% was achieved.

Confusion Matrix Results :

```
Confusion matrix, without normalization
[[976  34]
 [ 40 852]]
Normalized confusion matrix
[[ 0.96633663  0.03366337]
 [ 0.04484305  0.95515695]]
```



Naive Bayes :

Let's take a look at the Naive Bayes with gaussian distributions classifier applied to the problem. We'll pass the raw data and use all components and also look at passing PCA compressed data to the classifier and compare performance in terms of accuracy achieved on the validation set.

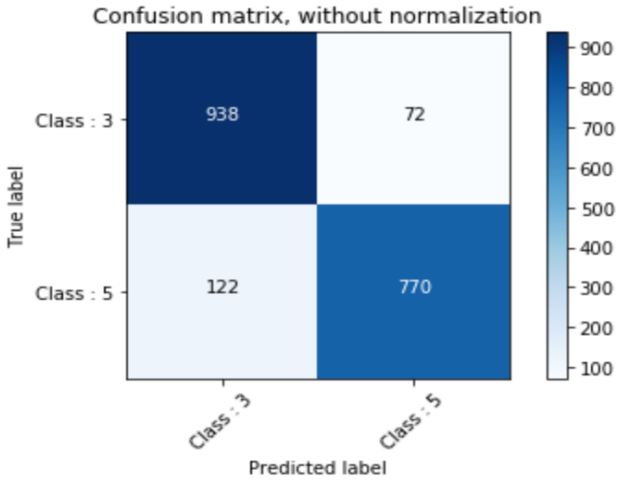
First a sanity check is performed with all 784 components by comparing our implementation accuracy results to the scikit learn library's. The accuracies match fairly well with a difference of 3%.

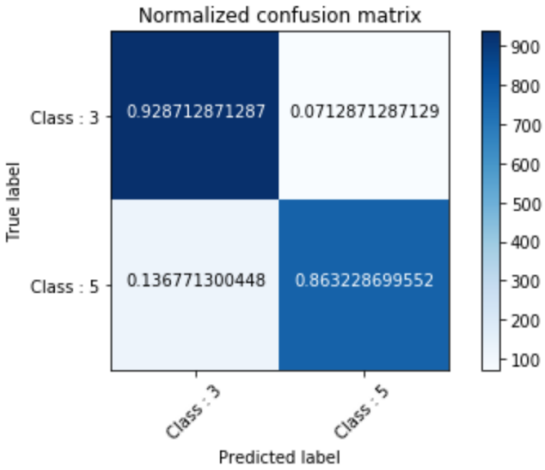
We believe the difference in accuracy is due to our simplifying assumption that variance across both classes is the same and usage of a class independent sigma in the algorithm. The library performs classification taking into account class specific sigma vectors.

Naive Bayes with all components gives an accuracy of 89.80% for the 3-5 classification problem.

Confusion Matrix Results :

```
Confusion matrix, without normalization
[[938  72]
 [122 770]]
Normalized confusion matrix
[[ 0.92871287  0.07128713]
 [ 0.1367713   0.8632287 ]]
```



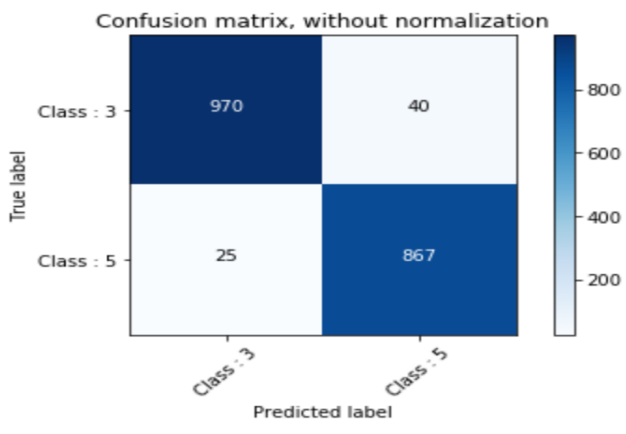


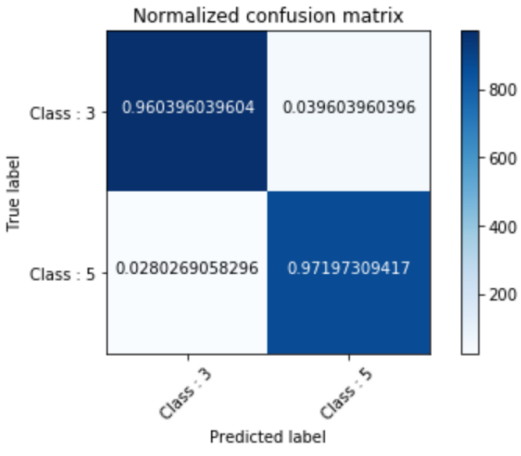
We will try passing compressed data to the Naive Bayes classifier. The LDA compressed data with 1 component which seemed good for reconstruction gave not such a great performance of about 89%.
Looking at PCA, from the cumulative sum proportion graph we inferred 30 component compressed data from PCA seemed good enough for reconstruction. We passed this compressed data to the Naive Bayes Classifier and achieved a much better performance of 96.582%

This leads to a conclusion that considering all components turned out to be 'unnecessary noise' that affected the performance of the classifier and also PCA compressed data works better in tandem with Naive Bayes for this classification problem compared to LDA compressed data although LDA compressed data required only 1 component for good reconstruction.

Confusion Matrix Results :

```
Confusion matrix, without normalization
[[970  40]
 [ 25 867]]
Normalized confusion matrix
[[ 0.96039604  0.03960396]
 [ 0.02802691  0.97197309]]
```





Accuracy Table :

0-1 Classification :

LDA with 30 components (tuned on validation set)	99.338%
Naive Bayes on raw data	99.385%
Naive Bayes on PCA compressed data(30 components)	99.953%

3-5 Classification :

LDA with 20 components (tuned on validation set)	96.109%
Naive Bayes on raw data	89.80%
Naive Bayes on PCA compressed data(30 components)	96.582%

Results :

In case of 0-1 dataset compression, PCA and LDA both required about 30 components to achieve good reconstruction. While in the case of 3-5 dataset, PCA required about 100 components while LDA required only 1 component to achieve a good reconstruction.

In terms of classification, in both cases, Naive Bayes trained on PCA compressed data outperforms Naive Bayes applied to whole data as well as LDA with a good number of components for accuracy determined by regularisation on validation set. We also noticed that in the case of 3-5 classification, although LDA compression was able to reconstruct with just one component, in tandem with Naive Bayes classifier it performed poorly.

Hypothesis on Results :

LDA assumes same covariance for different classes and its classification takes into account that this fact. One for the reasons PCA with NB is better for MNIST dataset is the same covariance assumption across multiple classes is not valid, infact it is quite different for both classes. Also, since PCA tries to extract the features across data which have maximum variance, it does better than LDA.

Distinction between 0-1 classification and 3-5 classification :

In general, it is noticed that 0-1 classification is an easier task than 3-5 classification. This we believe is due to the fact that the shapes of 0 and 1 vary more, with less overlaps,with respect to each other compared to 3 and 5 which are almost similar to each other resulting in some overlaps which are not well discernible by the classification algorithms.

Multiclass Classification :

In case of multiple class we can take a one vs all approach and the one which gives the best probability could be chosen. We can also consider applying Multi-class LDA which in most of the practical scenarios performs better than 2-class LDA (or one-vs-all scenario).