Aristotle University of Thessaloniki
Department of Informatics
Course: Pattern Recognition
Dimitriadis Spyridon A.E.M. 15318
spring semester 2017
key words: *custom dimension reduction, PCA, k-means, Gaussian Naïve Bayes Classifier*

1. In this assignment all the problems are implemented in MATLAB, a subset from the MNIST dataset has been used. It consists of all four classes of 0's, 1's, 2's and 3's. In the clustering problems, there will be used 5000 feature vectors from each class (20000 vectors total) and 20000 labels which has been taken from the train_set and train_label respectively. Moreover, for the Bayes Classification 6000 feature vectors and labels will be used for test, from the test_set and test_label respectively.

2. Reduction of dimensions-features from 784 to 2. For every 28x28 (pixel) matrix after the means of rows and columns that have been taken, I tried a lot of statistic measures and operations between them (i.e. mean, max - mean, median - mean, mean - variance, max etc.) to evaluate two numbers from the mean of rows and from the mean of columns. In Fig.1a,b we can see that these measures did not work that well to separate the classes, the only class that can be separated easily is the 1`s, Fig.1a.

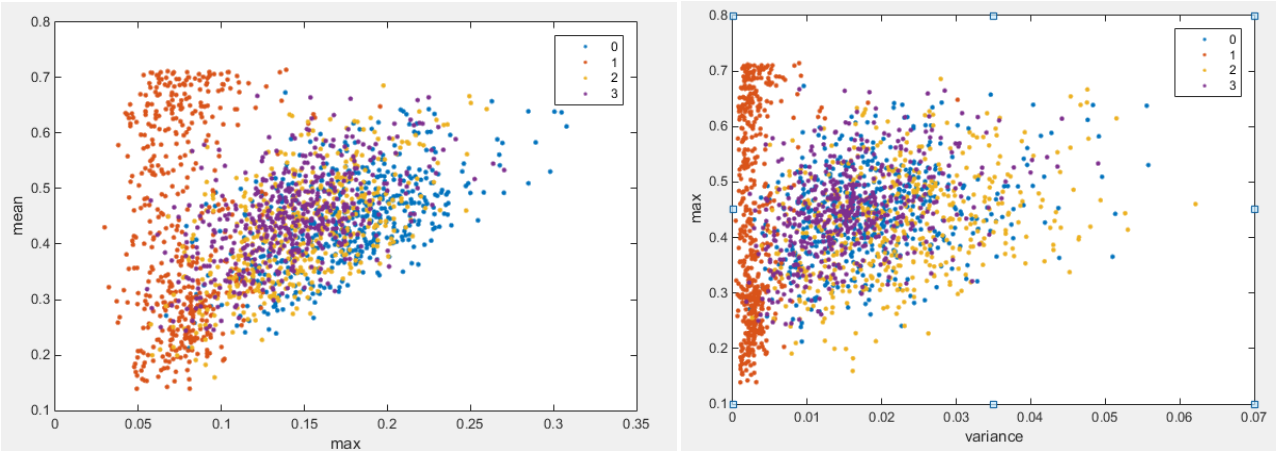

Figure 1a. visualization of the data with different statistic measures,

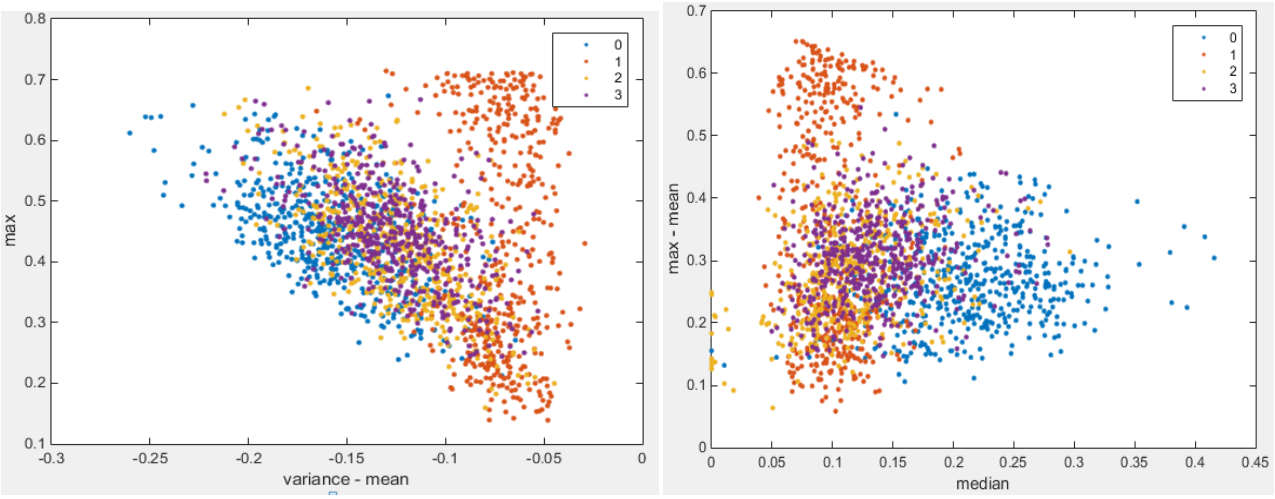

Figure 1b. visualization of the data with different statistic measures

In conclusion, kurtosis and skewness Fig.2 worked better than the previous ones. Although even with these statistic measures concerning, 2`s and 3`s did not achieve a complete separation between these classes.

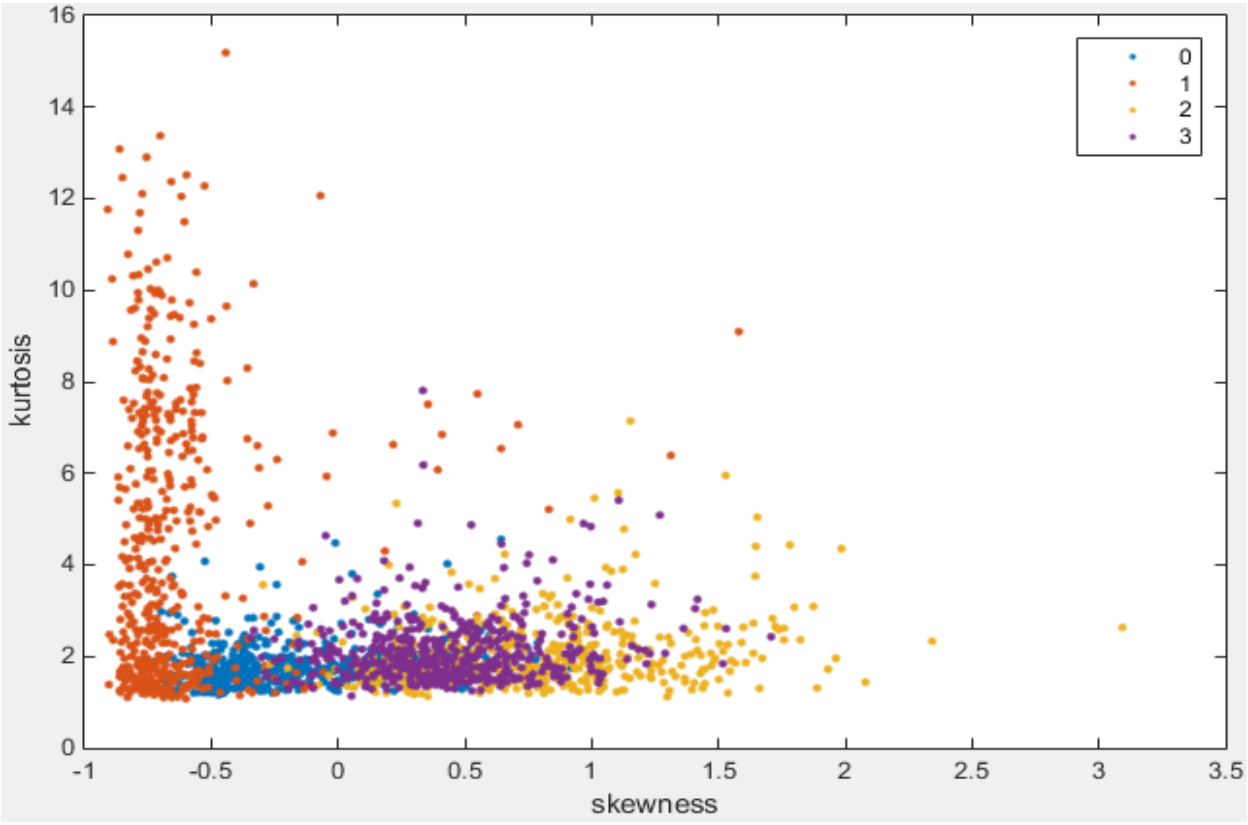

Figure 2. skweness - kurtosis

Note : For visualization reasons I chose to scatter 500 feature vectors from each class, i.e. 2000 feature vectors.

3. For the K-means algorithm, I did a simple implementation by creating a function named my_kmeans.m (reference to the code file for further details). To initialize the clusters my simple minimax algorithm has been used and random vectors from the dataset. There were implemented many testing and there were not much difference noticed, so for simplicity reasons random selection has been chosen. As a metric the squared Euclidean distance has been used. The algorithm was applied to the set of data prepared as described previously and is demonstrated in Fig.3. Having a quick look in the figure we can easily conclude that the separation is not the expected. The class of 0's is completely misplaced as we notice in Fig.2 and Fig.3. As an evaluation criterion for clustering quality, the purity has been taken. Purity is up to 48.56%.

   Note: Every time before k-means is implemented the data are shuffled as well as the labels, following the same order

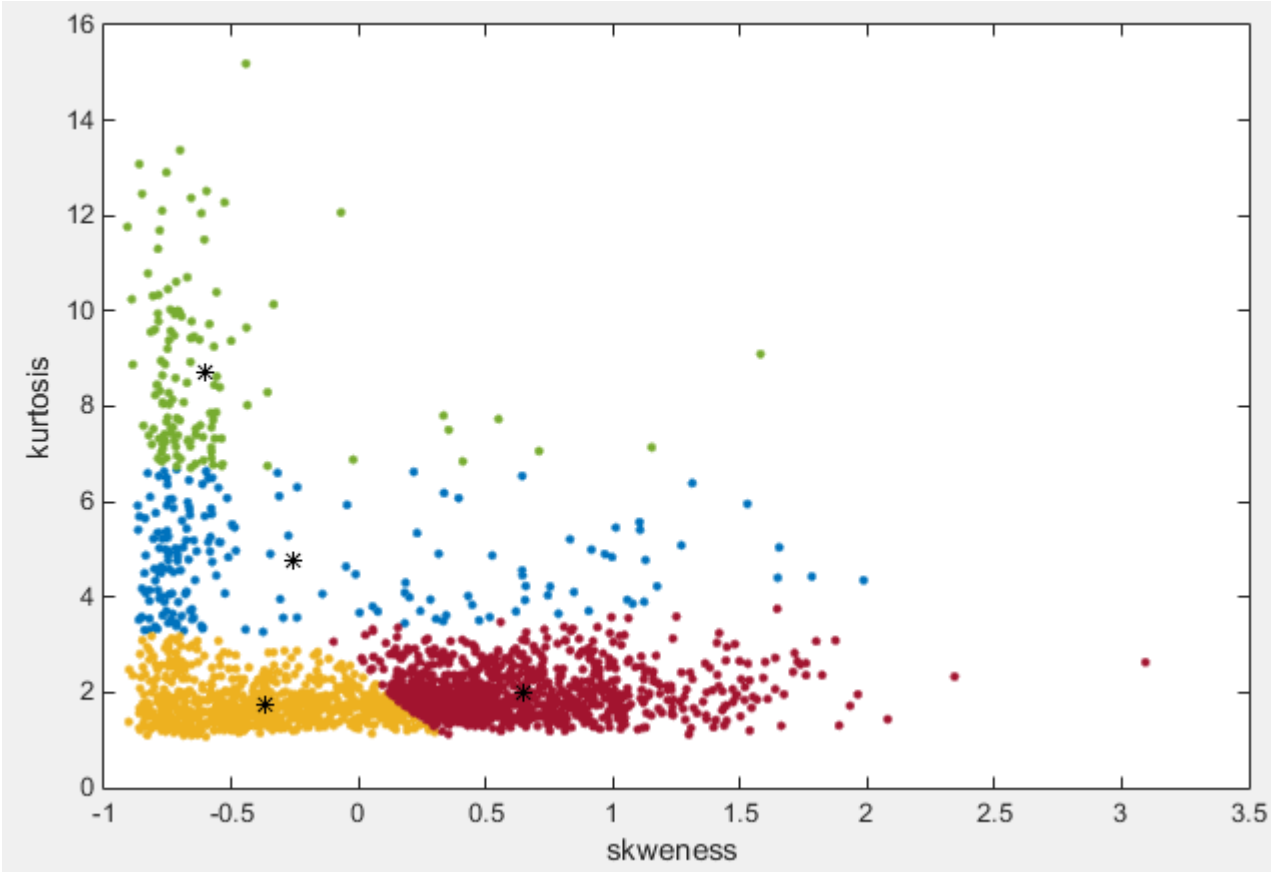

Figure 3. the centroids are noted with black stars and the classes with different colors (we do not know which color is each number because we are talking about clustering)

4. For Principal Component Analysis I used my implementation of PCA, named as my_PCA.m . First, PCA is used for V=2 (eigenvectors), to reduce the dimensions from 784 to 2. Then the dataset is visualized in Fig.4. Easily we can conclude that the classes are better separated than the Fig.2 where another dimension reduction was based on other criteria, that is statistical measures of rows and columns. Specifically, the 2's and 3's are way better separated.
   In this dataset, dimensionally reduced by PCA for V=2, my_kmeans algorithm is applied and visualized in Fig.5. We can notice that the PCA is working better than the previous custom dimension reduction. The purity in this case is up to 72.75%.
   Then PCA and k-means have been applied for various values of V (number of principle components). The evaluation is shown in Table 1.

V (number of eigenvectors)	V-th eigenvalue	Purity (%)
2	4.253	72.75
25	0.398	82.90
50	0.150	86.65
100	0.045	87.25

Table 1. Purity for different number of principle components (eigenvectors)

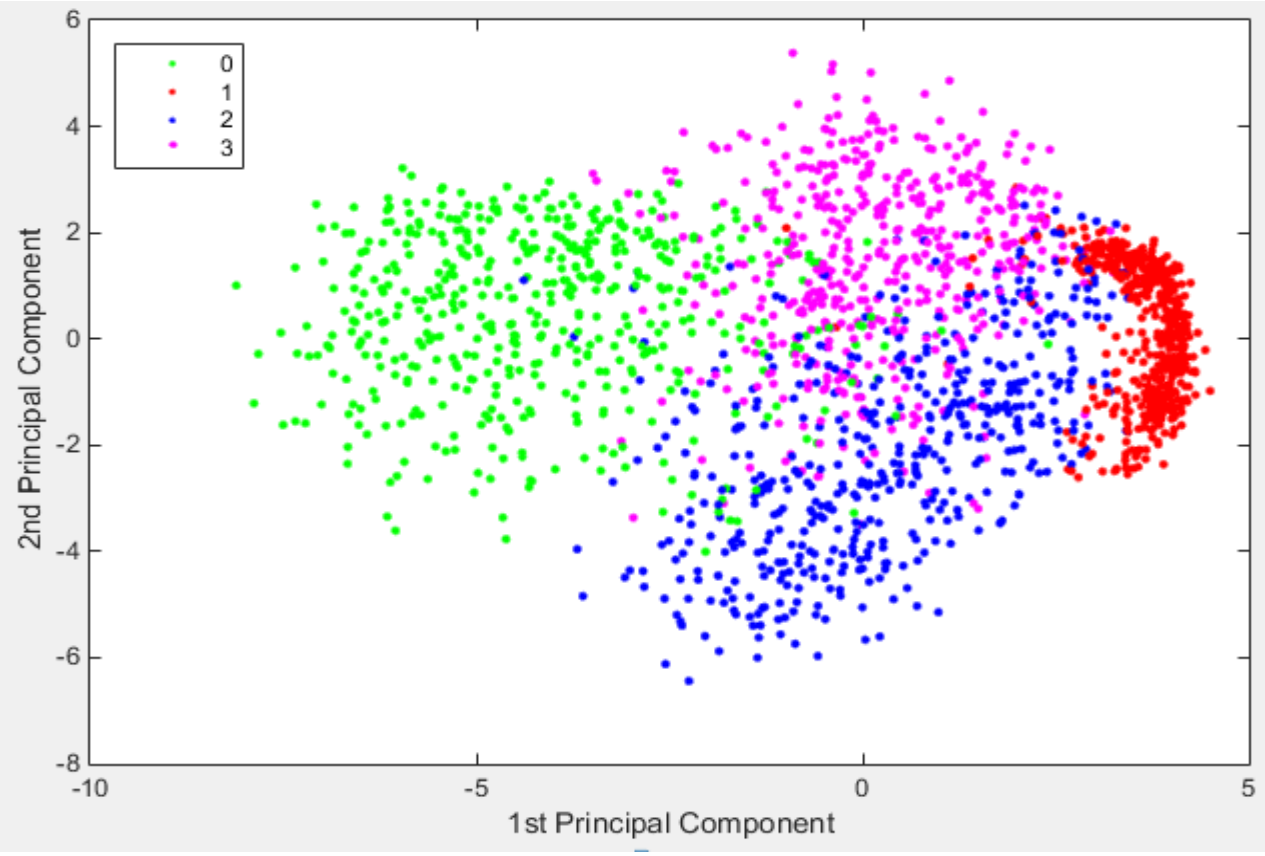

Figure 4. The dataset visualized with labels after PCA

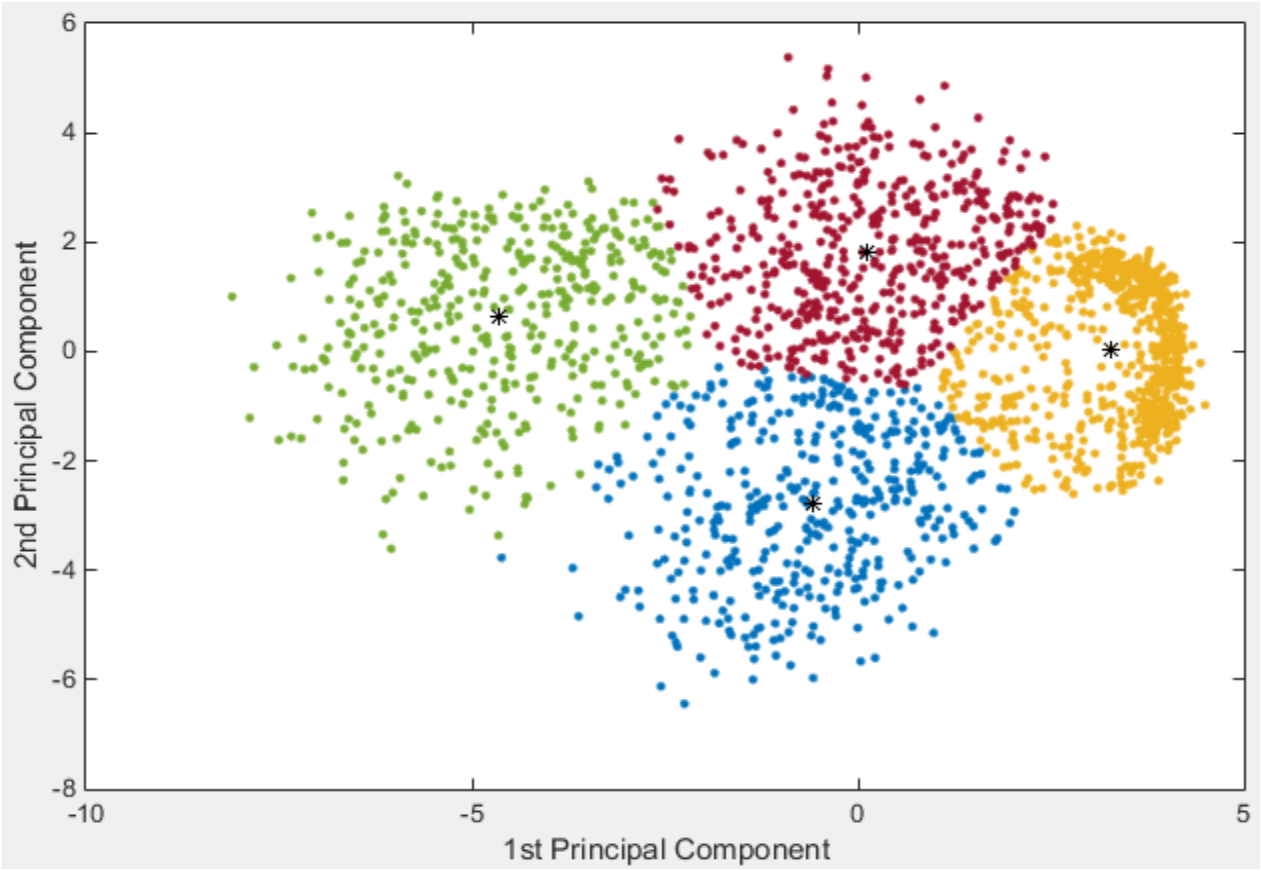

Figure 5. visualized k-means after PCA

In conclusion, the theory is verified if you look at the Table 1, as the value of eigenvalue decrease as the importance of the eigenvector decrease, shown in Fig.6 (theory: as the eigenvalue goes to zero as less information its eigenvector 'carries'). So the eigenvectors with small eigenvalues makes less affect in the evaluation. Furthermore, we notice in Table 1 that between 50 and 100 eigenvectors the purity is not changed noticeable and after some implementations we get as a conclusion that Vmax = 60 that has λ=0.1086 (eigenvalue) and purity 87.2%.

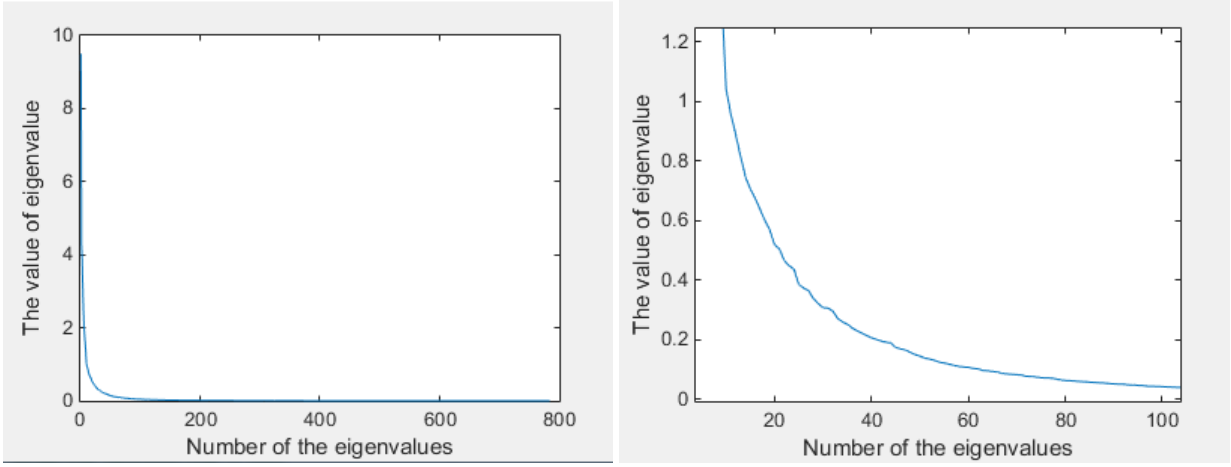

Figure 6. left: value of eigenvalues and the number of them. right: zoomed in

5. In the final question, I implemented a Gaussian Naïve Bayes Classifier to classify the 0's, 1's, 2's and 3's from MNIST dataset. The data reduced in 1 dimension to visualize them and have a glimpse and in Vmax=60 dimensions with the use of PCA. The same method of dimensionality reduction is used for the test set (the same eigenvector to project the data).
   For the evaluation of the means, the standard deviations and the evaluation of the classifier the labels of train set and the labels of test set have been used. The four classes are equiprobable, 5000 vectors from each class i.e. the priori probability is $P(\omega_i)=0.25$ for i=1,2,3,4. So the test becomes the maximum likelihood between the four classes. For evaluation accuracy has been used, i.e. how many correct labeled vectors we had, over the total number of the vectors.
   First, in Fig.7 it is shown the Gaussian Naïve Bayes Classifier in 1 dimension (dimension reduction with PCA), we have a quick look and evaluate the accuracy for the train set. The accuracy is 71.15%. It is easily noticeable from the Fig.7 that the Gaussian distribution is not the probability distribution function that the data follow but it is a fair estimation. Moreover, the separation of the class of 2's and 3's is not satisfied at all, so we will have a major error. On the other hand, 0's and 1's are well separated.

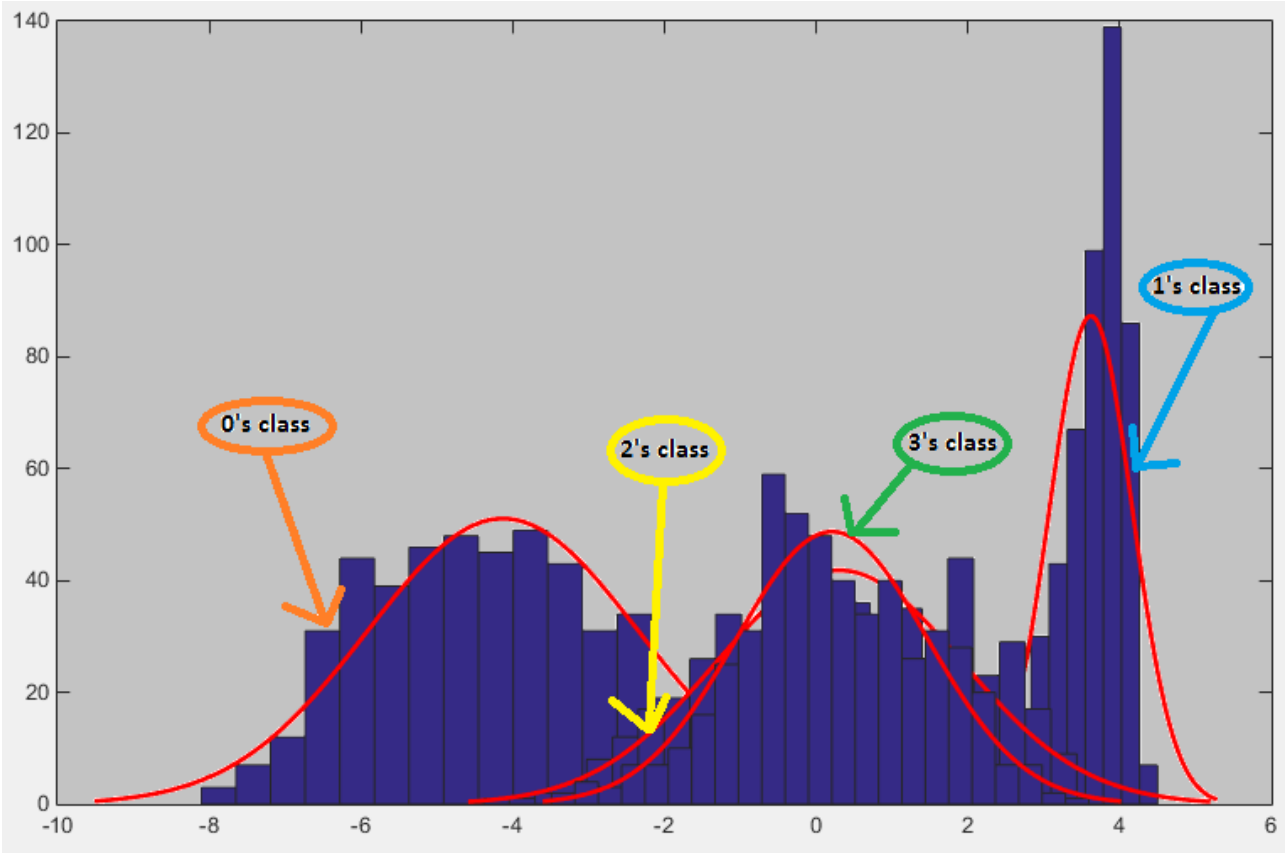

Figure 7. the different classes of train_set fit with Gaussian in 1 dimension (reduced with PCA)

In a nutshell, in 60 dimensions we have accuracy in train set 94.72% which is a great presentence considering that we have four classes.

Note: I tried to evaluate the test set too but I had really low accuracy 25%. Probably I was mistaken when I tried to reduce the dimensions of the test set as it was reduced in the train set, i.e. to use the same normalization and the same eigenvectors to project the feature vectors. Because of the dead line there were no more time to reconsider and fix the problem in the script.