COL774 – Assignment 3

- **By Nishant Kumar**
2012CS10239

Problem – 1

a)

In this question, Naïve Bayes with **multinomial event mode** was implemented. Since this model takes into consideration the frequency of the words and in general performs better than the Bernoulli event model, this model was used.

The input data was preprocessed using python (code included in submission). The data was preprocessed to convert the articles into the form of feature vectors.

The data was shuffled randomly using python and 5 train and test files (as specified in question) was produced. Matlab was used to implement Naïve Bayes on these train and test files.

The average accuracy obtained = **95.4%.**

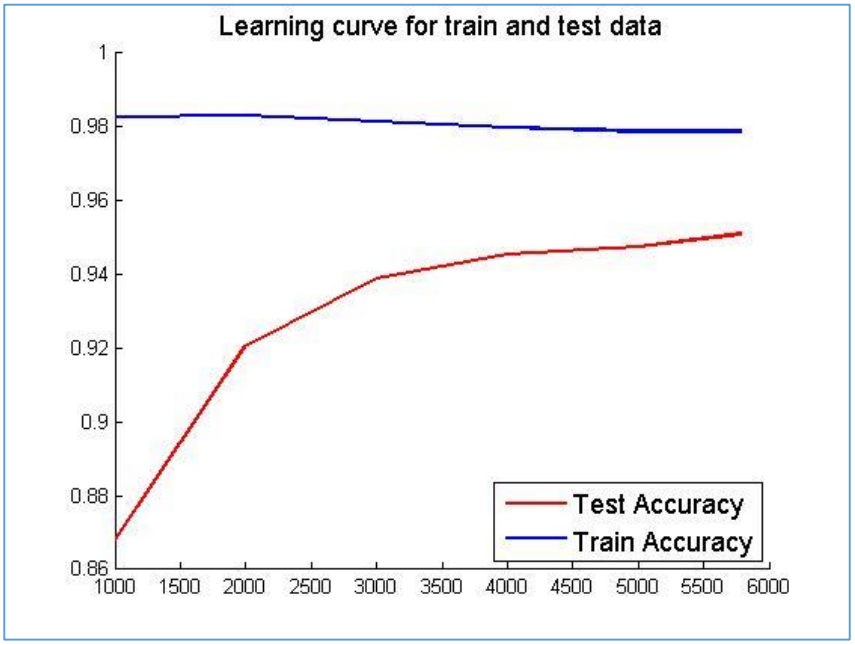
b)

Randomly guessing the class of the article gives an average accuracy of **12.2%.** Certainly Naïve Bayes performs much better than random prediction.

c)

Cross-posted data means that there are articles with more than one label assigned to it. This can potentially create problems for the Naïve Bayes learner for it needs to train on data where same feature vector has more than one y value associated with it, whereas Naïve Bayes learner can only predict one value. But it does not affect the actual working of the learner for firstly, such amount of cross-posted data is very low and secondly, the parameters of Naïve Bayes are not swayed because of this cross-posting (as they are calculated from all the data).

d) The learning curve obtained was as follows:



NOTE: Train accuracy was found on the same dataset on which the algorithm was trained and not the full dataset.

As can be seen from the learning curves, the train accuracy is decreasing (however slowly) as the number of training examples is increasing. On the other hand, the test accuracy is increasing with increase in the number of training examples. This is because when the amount of data available to train is low, it is more likely the data will overfit and hence test accuracy is very low. As the amount of train data increases, the likelihood of train data to overfit decreases and the test accuracy therefore increases.

e)

The actual labels obtained were as follows (taken from the preprocessing step of python):

Numerical Label	Actual Label
1	rec.motorcycles
2	talk.religion.misc
3	rec.autos
4	talk.politics.mideast
5	talk.politics.guns
6	talk.politics.misc
7	rec.sport.hockey

8	rec.sport.baseball

To find the confusion matrix for the part (a), the confusion matrices over all the train-test splits were found and averaged.

The **average confusion matrix** obtained was as follows:

194.8	1	0	2	0.4	0.6	0	1
0.6	193.2	0	0.2	0.6	3.6	0	1
0.4	0.8	180.8000	0.2	1.2	0.4	1.2	3
3.4	0.2	0	193.4	0.8	0.4	0	0.6
0	0.6	0.4	0	174.6	0	0.6	5.6
0	3.4	0.2	0.4	1	191.4	0.2	1.2
0	1.2	1.6	0.2	7.2	0.4	110.6	4.4
0	0.2	3	0.2	14.2	0.6	1.8	135

NOTE: The highlighted entries correspond to the diagonal entries and the green highlighted entry corresponds to the non-diagonal entry with the maximum value.

Using the table on the last page, it can be found that the **maximum diagonal entry is for rec.sport.hockey.**

And the **maximum non-diagonal entry is for talk.politics.misc and talk.politics.guns.** The reason for Naïve Bayes confusing the most with these 2 articles is for these 2 correspond to the same area – politics. Therefore articles marked with talk.politics.misc and talk.politics.guns contain similar words in the articles and hence have similar feature vectors and so, Naïve Bayes is confusing a lot in classifying these 2 types of articles.

Problem – 2

b)

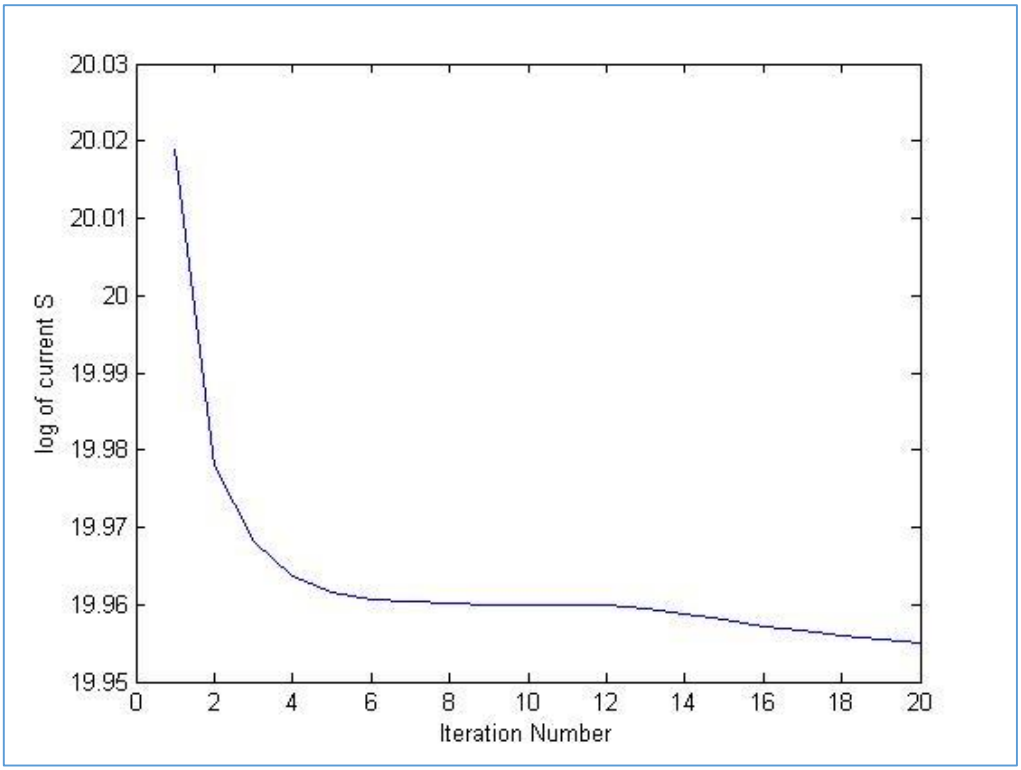
The convergence criteria used when running k-means was either the number of iterations exceeds 30 or the change in the value of $S = \sum_i \left|\left|x^i - \mu_{k_i}\right|\right|^2$ across 2 consecutive iterations is less than eps = 0.0001.

The following was observed on running the k-means algorithm starting with a random clustering.

Number of iterations = 26

Final Value of log(S) = 19.95483

c)

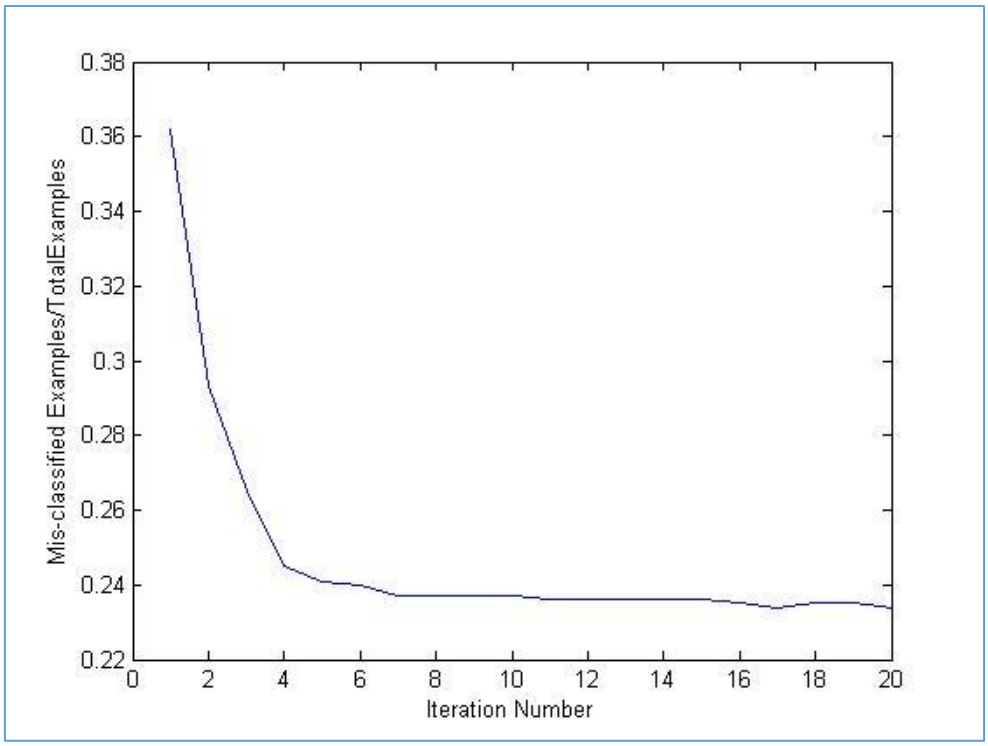


The above is the graph of log(S) vs the number of iterations (upper bounded to 20).

As can be seen the value of log(S) is decreasing with every iteration. It decreases sharply in the first few iterations and then the rate of decrease falls down sharply.

This is expected as S denotes the sum of the distances of the points from their cluster centroids. As the number of iterations increase, the intra cluster distances become smaller and the inter cluster distances become larger and so the value of S decreases.

c)



This graph shows the variation of the ratio of misclassified points across all clusters as number of iterations vary.

The error has decreased and the final/best value of error obtained after 20 iterations = 0.22.

As the number of iterations of k-means increase, points get assigned better clusters and so the error goes on decreasing. In the later iterations of k-means the cluster assignments donot change as much as they do in the initial iterations and hence, the error does not decrease as rapidly as it does in the initial iterations.

Problem – 3

a) The ML estimate of the parameters on training on train-0.data were as follows :

CPT of Difficulty

P(D=0)	P(D=1)
0.59860	0.40140

CPT of Intelligence

P(I=0)	P(I=1)
0.69520	0.3048

CPT of Grade

	P(G=1)	P(G=2)	P(G=3)
I=0,D=0	0.29849	0.40373	0.29778
I=0,D=1	0.05882	0.25586	0.68531
I=1,D=0	0.90028	0.07922	0.02050
I=1,D=1	0.50603	0.28158	0.21239

CPT of SAT

	P(S=0)	P(S=1)
I=0	0.65434	0.34566
I=1	0.64239	0.35761

CPT of Letter

	P(L=0)	P(L=1)
G=1	0.10450	0.89550
G=2	0.39031	0.60969
G=3	0.99013	0.00987

Log-likelihood of test data = -6183.31195

b)

Initialization:

To initialize the data all the examples were considered and if the probability calculation in the particular CPT entry did not require the missing value, then that missing data example was also considered in initialization.

E-Step:

To fill in the values in the E-step joint probability distribution of the data was used as follows:

$$P(D, I, G, S, L) = P(D) * P(I) * P(G|D, I) * P(S|I) * P(L|G)$$

And the values were filled in as follows (following is for an example where G is missing):

$$P(G|D, I, S, L) = \frac{P(D, I, G, S, L)}{P(D, I, S, L)} \; etc$$

M-step:

For this step the values found in the E-step was used. If an example had D missing and had probability obtained from E-step as $(d0, d1)$, then d0 fraction of the original 1 example was considered as having D=0 and d1 fraction of the original 1 example was considered as having D=1.

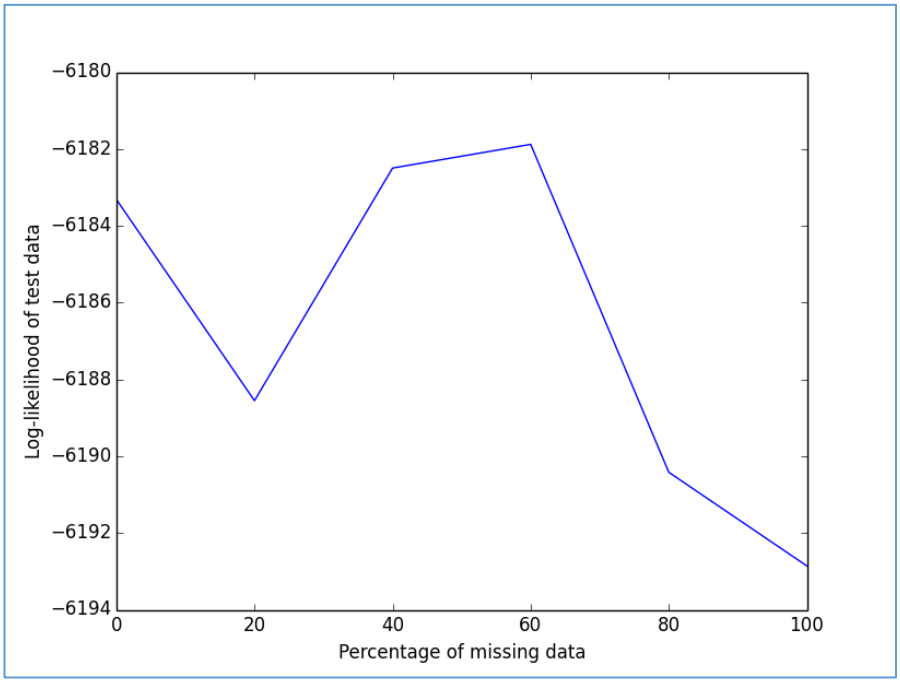
NOTE: Value of smoothing parameter used = 0.01

Convergence criteria

Either max difference between any 2 CPT entries across consecutive iterations < eps or number of iterations > 100.

Value of eps used = 0.0001

c)



The above graph shows the variation of the log-likelihood of the test data vs the amount of missing information in the train data.

As can be seen, the log-likelihood of the test data is maximum when missing data% = 60. Another point to note is that the variation in log-likelihood is very small – it varies from [-6194,-6182].

Also, there is no particular correlation between the amount of missing data and the log-likelihood of the test data. This is perhaps because the train and the test data come from different distributions. Another sort of possible explanation can be that when there is more missing data, the CPT calculation is more flexible and so better parameters are learnt. But if very high amount of data is missing, the model becomes weak enough to not the learn the parameters correctly.

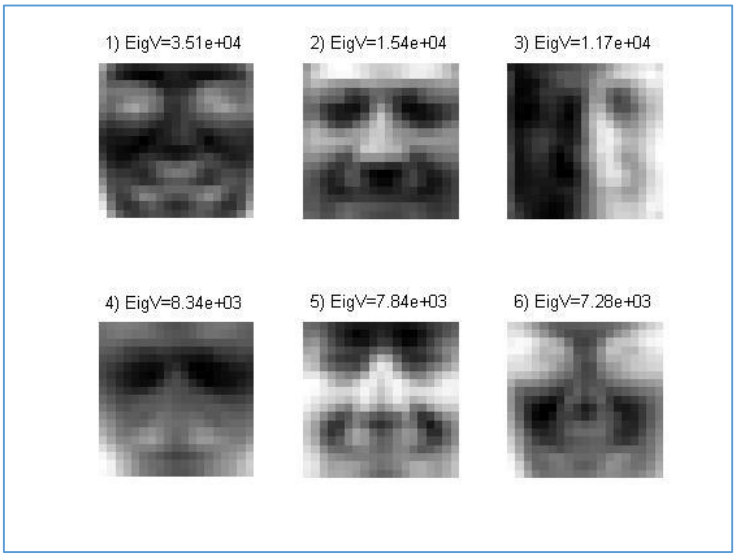
Problem – 4

a) To calculate the average face, all the images were read one by one and mean was found. Following was the average face found:

b) The images were first read into a design matrix, mean subtracted and SVD applied on this new matrix. Say, $[U, S, V] = SVD(NX), where\ NX = X - meanX$.

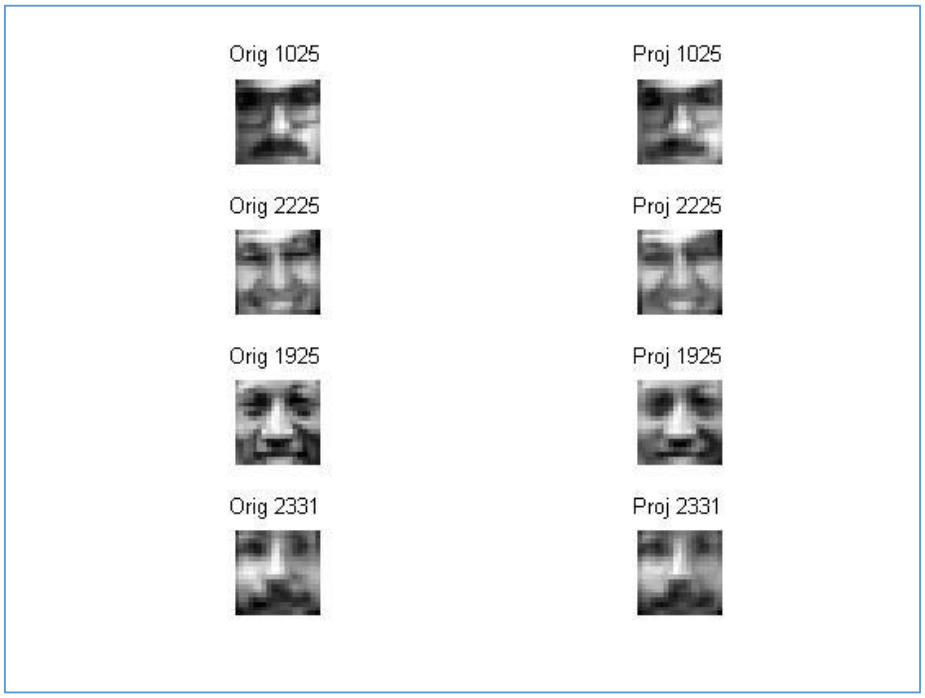
Then the columns of V are Eigen-vectors of the new data. This was saved in matlab file format to disk.

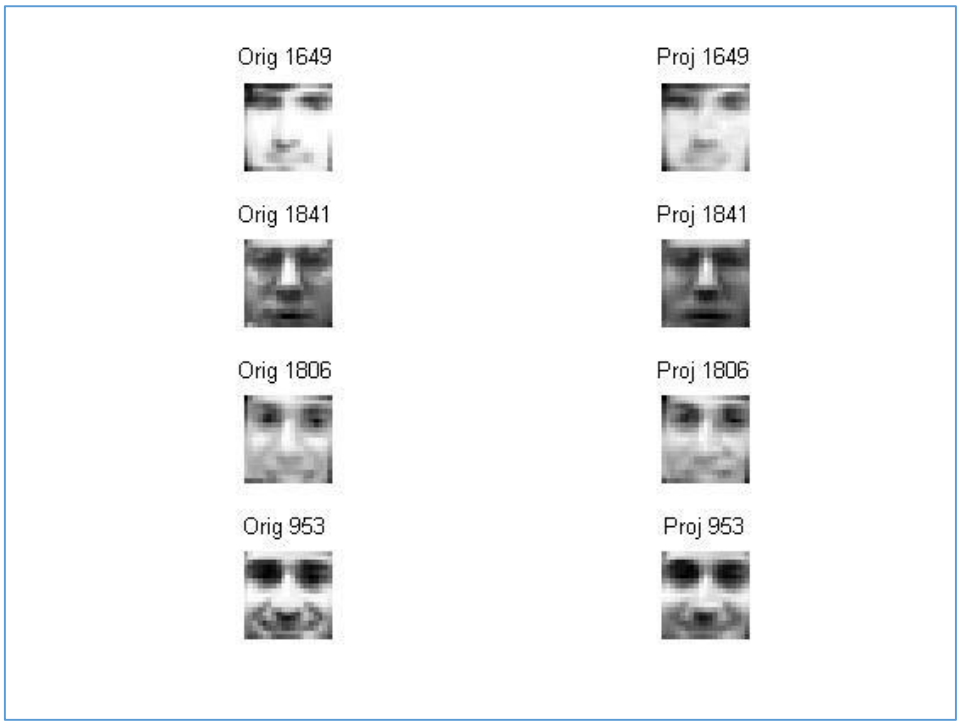
c) The following were the first 6 Eigen-faces. *Note that the Eigen-faces have a decreasing order of Eigen-value, which was expected. Also, the faces appear to be orthogonal to each other, which was also expected.*



Note: To display the Eigen faces the matlab inbuilt function 'imagesc' was used.

d) In this part, the script part4d.m chooses 4 random images from the face database and displays the original and the projected images. Following is a sample output from the script:

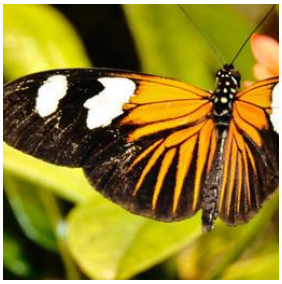




As can be seen from above, the projected images are not very different from the original images. Infact in all cases, they seem quite matching. This shows that 50 Eigen faces is actually enough for representing all the faces, as was expected.

e) Some non-face images were downloaded from the internet and are shown below. The script prob4d.m takes these images, resizes them to 19*19, converts them to grayscale and projects them on the Eigen-vectors as done previously.

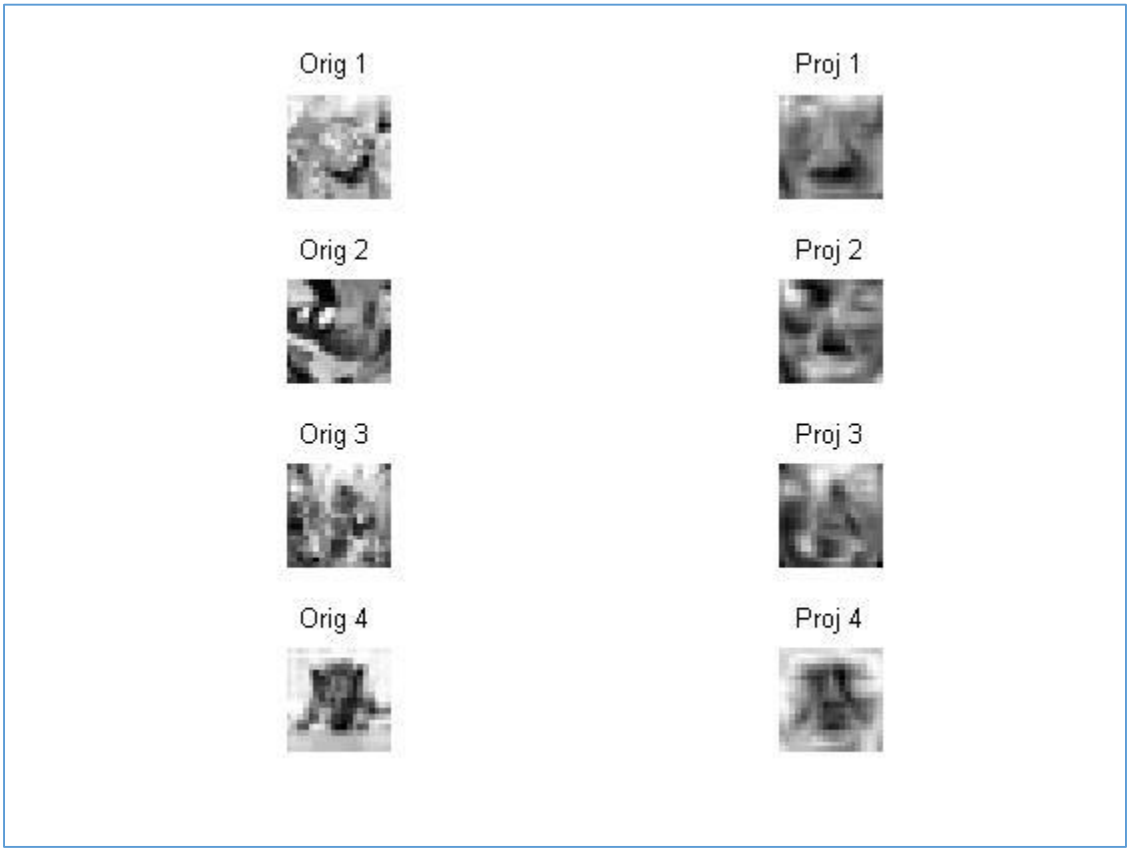
1) 

2) 

3) 

4) 

The following shows the grayscale images on the left side and the projected images on the right side column:



NOTE: The order of the images is the same as displayed in the color images. For eg, the first one is a butterfly and the last one is a dog.

Observation: As can be seen from the above images, on projecting the non-face images onto the Eigen-vectors, the algorithm tries to sort find the closest face to the image. For example, the last figure is of a dog and the algorithm tries to sort of match the dog's face to a human's face. This is happening because these non-face images are not from the same subspace as the other facial images and hence the problem.

Some other examples of running on non-face images are: