REPORT

Q1 : **K MEANS**

Part (a) :

- I generate random indexes to pick random centroids initially from the given data points. Convergence is checked on the centroid assignments to all the data points. My convergence criteria is : 1 percent of m examples allowed to have errors in their centroids assignments.
- Stats for a particular example are
    - iterations = 15
    - J = 1.7385e+05
    - Elapsed time is 43.6498 seconds.
- For the rest of the random initialisations I get these results close to above figures only. Values for these examples can be viewed from the next part.

Part (b) :

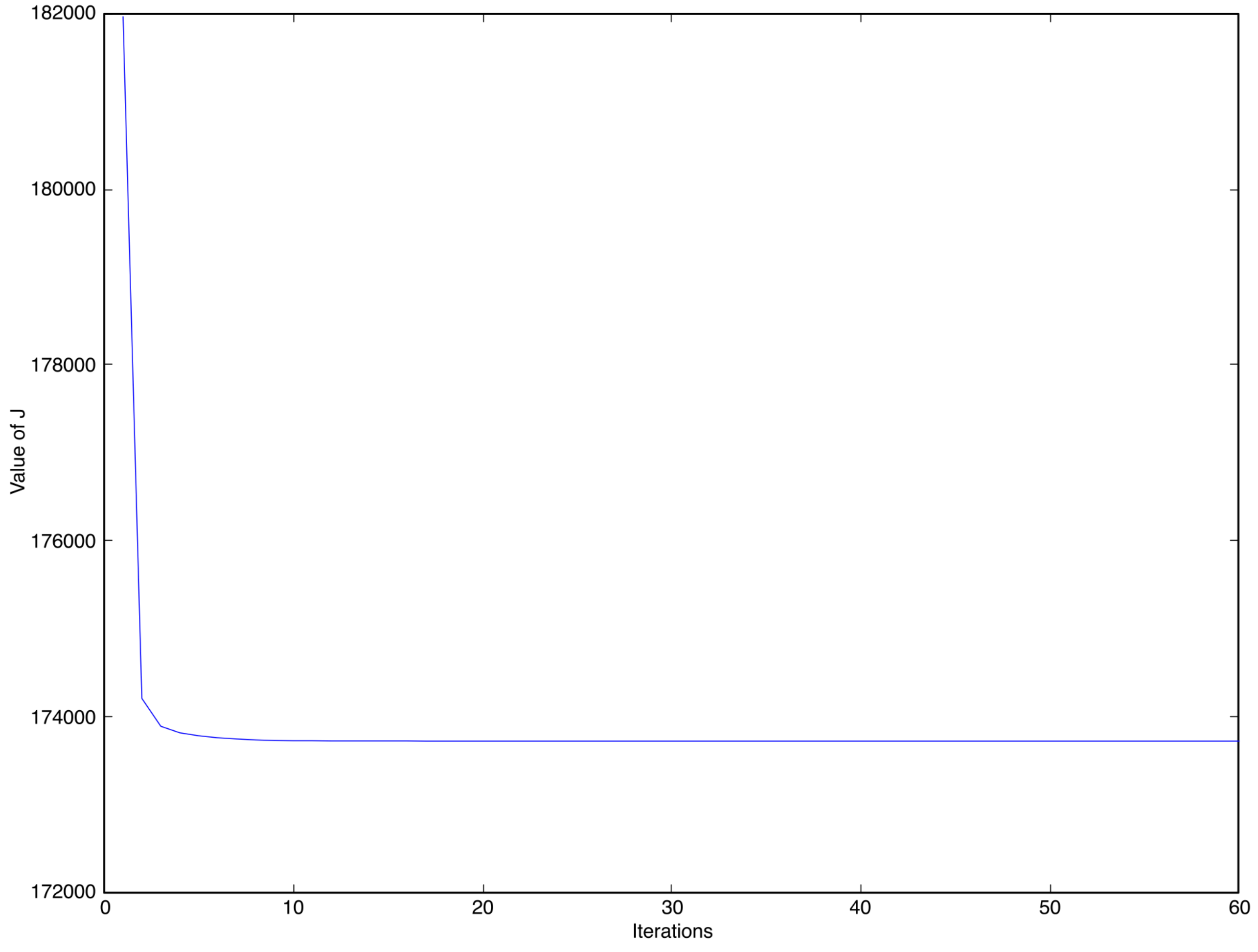
- I observed from the 10 random initialisations, that I don't get the same optimal value of J (distortion) on convergence. This is because this J is not purely convex in the variables. Its dependent on C and μ. Where C is discrete variable and μ is continuous. So in our algorithm we converge to different local minima for each random initialisation. Hence we get such results where no. of iterations taken to converge each time is different with the J value too different each time. Sometimes in less no. of iterations it converges to a better value than it coverages to in more number of iterations.

```
iter =  19
ans =    1.7373e+05

iter =  11
ans =    1.9183e+05

iter =  19
ans =    1.7373e+05

iter =  11
ans =    1.7438e+05

iter =  25
ans =    1.7374e+05

iter =  13
ans =    1.7372e+05

iter =  15
ans =    1.7466e+05

iter =  22
ans =    1.7375e+05

iter =  14
ans =    1.7438e+05

iter =  9
ans =    1.7393e+05
```

- Now lets see the case with the best value of J

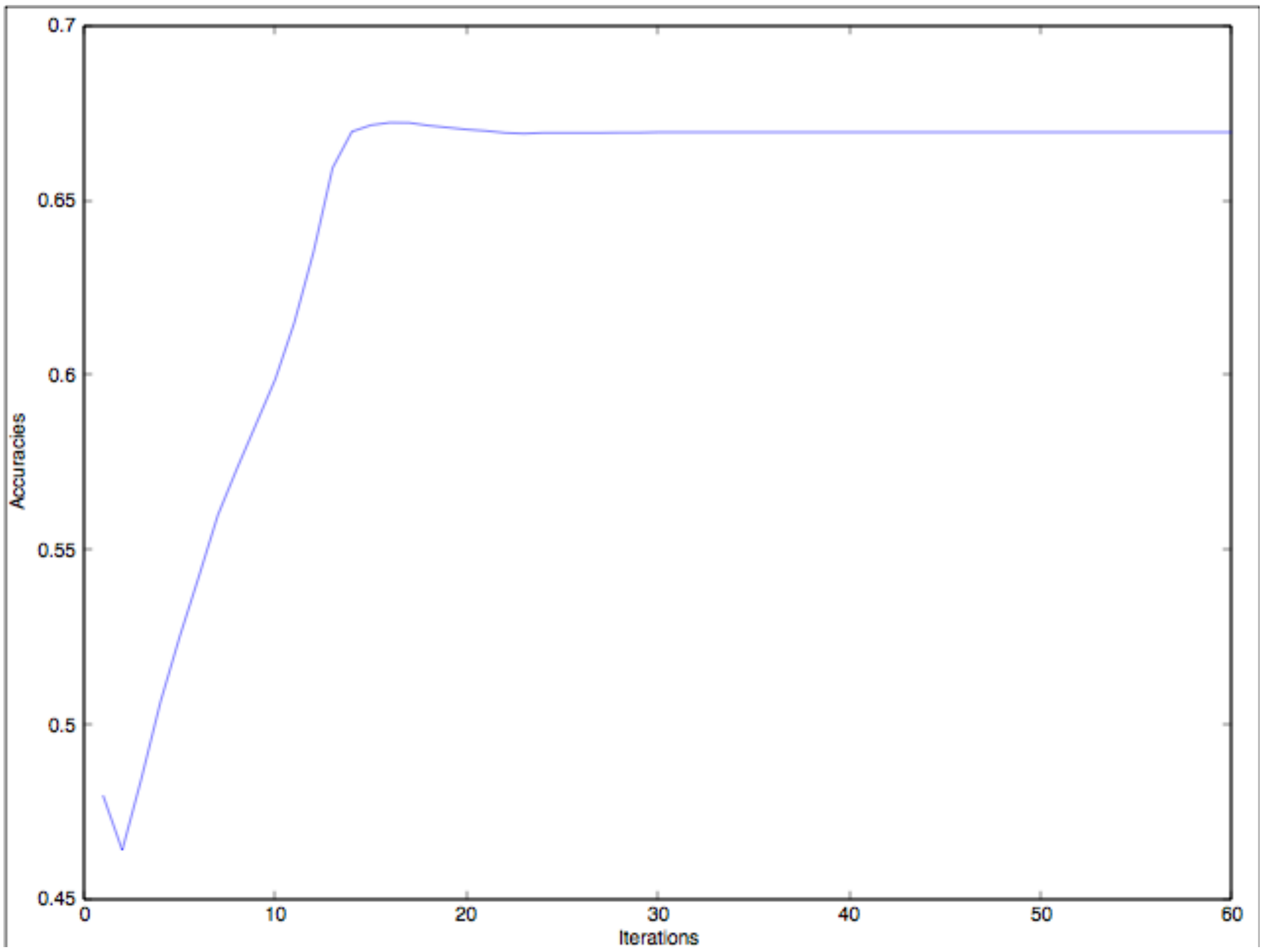
```
r = Random initialisation for 1.7372e+05 case !!

   9486
   4865
   5503
   8060
    424
   7863
```

- These are the indexes of the array of data points in the training set.
- From the graph I observe that J starts from a very high value and drops very fast to close to optimal value after around 10 iterations, and then remains constant for all the further iterations.
- Hence the algorithm had converged very much around 10 to 15 iterations and remains stuck on the local minima till iteration no. 60.



Part (c) :

- In the graph I observe that initially the accuracy was a bit low, below 50%, since it was a random initialisation. It increases linearly as did the J value drop in the previous graph. Then as soon as the algorithm has converged, the accuracy remains constant at its maximum value, as did the J value rained constant after the convergence. It reaches the max value of ~ 67 %, which I observed for multiple random initialisations.
- So this graph is pretty similar to the previous graph, just that instead of decreasing and then being constant this graph increases and the remains constant. Already described in above point.



Part (d) :

- The 10 cross validation accuracies I get are :

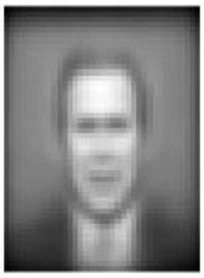
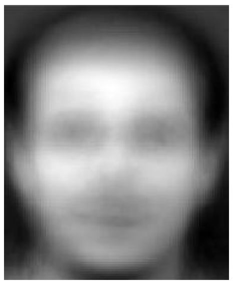
```
[ 0.96168582  0.91371045  0.86372361  0.93378119  0.94044188  0.95677233
  0.93948127  0.94038462  0.93551492  0.96628131]
```

- I observe that these are pretty high as compared to the accuracies I obtained in the previous part. Hence knowing the labels before hand improved the accuracies a lot. One more factor which plays a role is that multi class SVM was used for classification. This can be the feature of the data that it was easily separable using SVM.

Q2 : **PCA**

Part (a) :

- Loaded the data sets by going for each of those folders inside the main directory. Each image was loaded as a 2D matrix, I converted each of these into a vector, and that vector behaves as a feature vector for the images. Now when I need to show an image then I reshape it to desired dimensions.
- The average face of the datasets are :

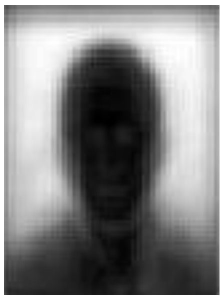
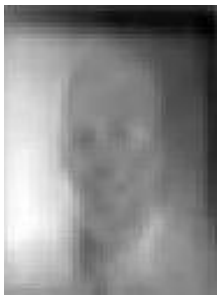
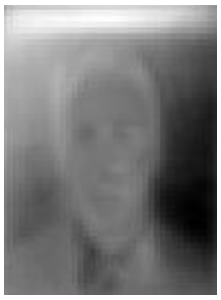
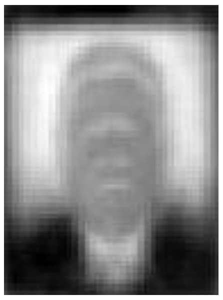
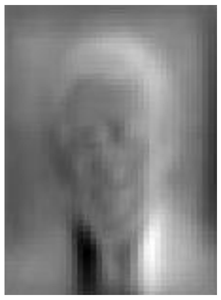



Part (b) :

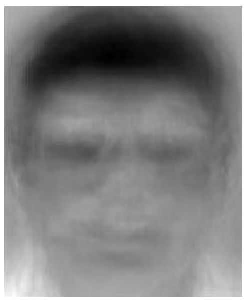
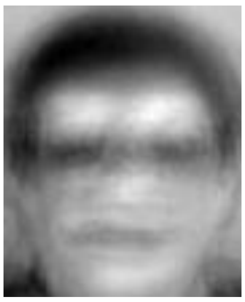
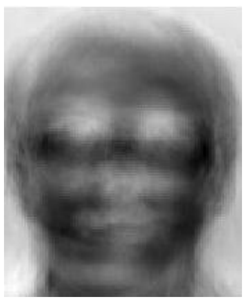
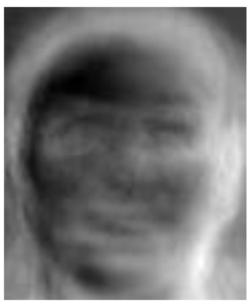
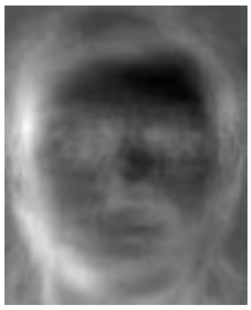
- The principal components are stored in files PC1.txt and PC2.txt
- Due to no normalisation, the change in the calculation of the X (the input to SVD) will be because of the fact the mean is non zero, there will no effect due to the fact that the variance is not unity, because we never used that while deriving the SVD formulation. Now since the mean is not zero the mean faces we calculated above will be subtracted from each of the figures in their respective datasets. That will give our matrix X, which will be the input to SVD function.

Part (c) :

- The top 5 Eigen faces for the dataset 1 are (in decreasing order of the Eigen values, left to right):

- For the dataset 2 :

Part (d) :

- The projections are stored in projections1.txt and projections2.txt

Part (e) :

- We can see from the below analysis that time taken for SVM to learn the model reduces by a factor of 10 when trained on projected set of attributes while giving almost same accuracy for dataset 2 as compared to that of original set of attributes.
- In case of dataset 1, accuracy does decrease when projected set of attributes were used but the time taken is much less as compared to the original one.
- Decreased accuracy can be explained because we did not give all the information available in case of projected attributes to the SVM learner as we restricted ourselves to top 50 principal components.
- Accuracies for dataset1 : [0.78, 0.81, 0.75, 0.76, 0.80, 0.77, 0.73, 0.84, 0.82, 0.74] with time taken = 31.9 seconds
- Accuracies for dataset1 using the projected set of attributes = [0.61, 0.7, 0.60, 0.64, 0.63, 0.66, 0.57, 0.64, 0.63, 0.67] with time = 3.41 sec
- Acc for d2 = [1, 0.95, 1, 1, 1, 1, 1, 0.95, 0.975, 0.975] with time = 30.5 sec
- Acc for d2 using projections = [1, 0.975, 0.975, 1, 0.975, 1, 1, 0.95, 0.95, 0.975, 0.95] with time = 0.27 sec

Part (f) :

Projected images do give a feel of original images but these are more blurred and not very close to the original ones since we are using only 50 dimensional data out of 1850 or 10304 dimensional datas to represent them as compared to original ones where the dimensionality of the data was much more.