COL774

Report of Assignment 3

DEVANSH DALAL (2012CS10224)

Note: All the questions are mainly done in Matlab 2013a or 2014b.

Q1. Newsgroup Classification

a. **Naive Bayes algorithm**

The python script *shuffle.py* processes the training data and randomly splits it as required by the question. The *Bayes* function is called for each split with the respective train and test files and final averaged accuracy is obtained as follows:

Kth-split	Accuracy
1	**0.955740**
2	**0.950899**
3	**0.953665**
4	**0.941909**
5	**0.957815**
Average	**0.9520**

b. **Comparison with random guessing**:
There are 8 newsgroups in the given data as shown below.

1	rec.autos
2	rec.motorcycles
3	rec.sport.baseball
4	rec.sport.hockey
5	talk.politics.guns
6	talk.politics.mideast
7	talk.politics.misc
8	talk.religion.misc

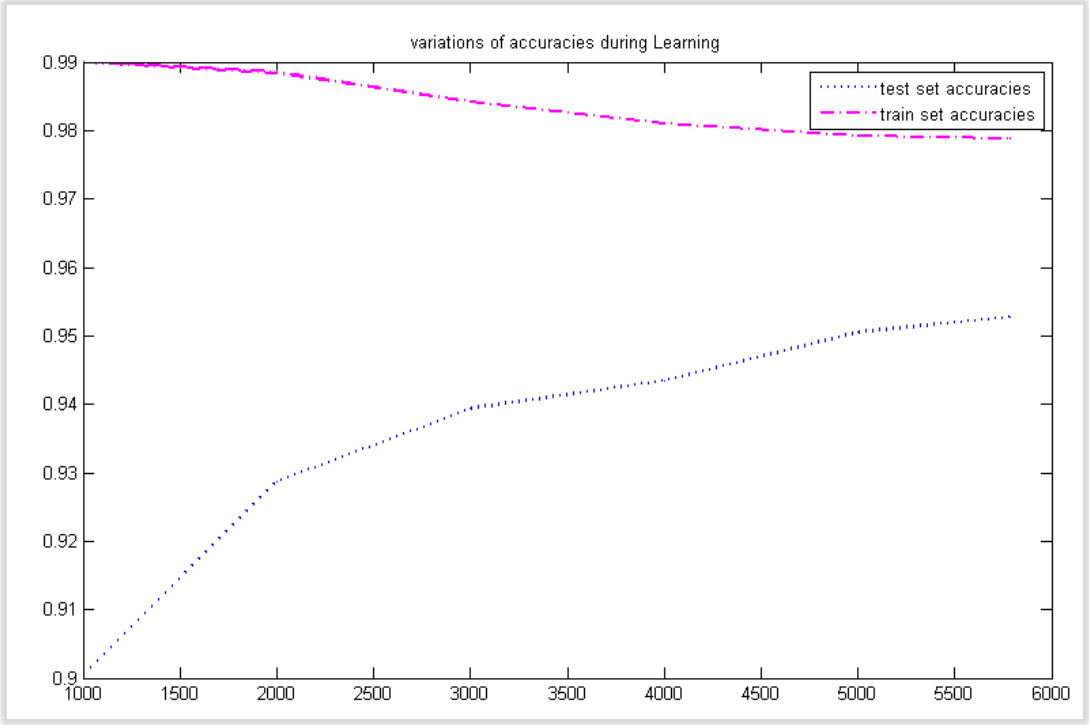
With random guessing probability of classifying an example correctly is **1/8 = 12.25 %**. While Naïve Bayes gives an average accuracy of about **95.2%** which is much better than random. So Naïve Bayes gives an improvement of about **82.95%.**

c. **Cross posted data:**
The high accuracy of 95% despite the cross-posting of data tells us that Naïve Bayes is not very sensitive to noise and wrong labels. The is due to the fact that Naïve Bayes doesn't make any

stronger assumptions on the data due to which the learnt model is not much sensitive to slight variations in the data. But if the noise is more than Naïve Bayes also start to learn incorrect values of probabilities leading do under-fitting and less accuracy.

d. **Learning curve:**

The outputs for this part is shown in *fig1.png.* The accuracies (train and test) are plotted as a function of size of training input file and plot is shown below.

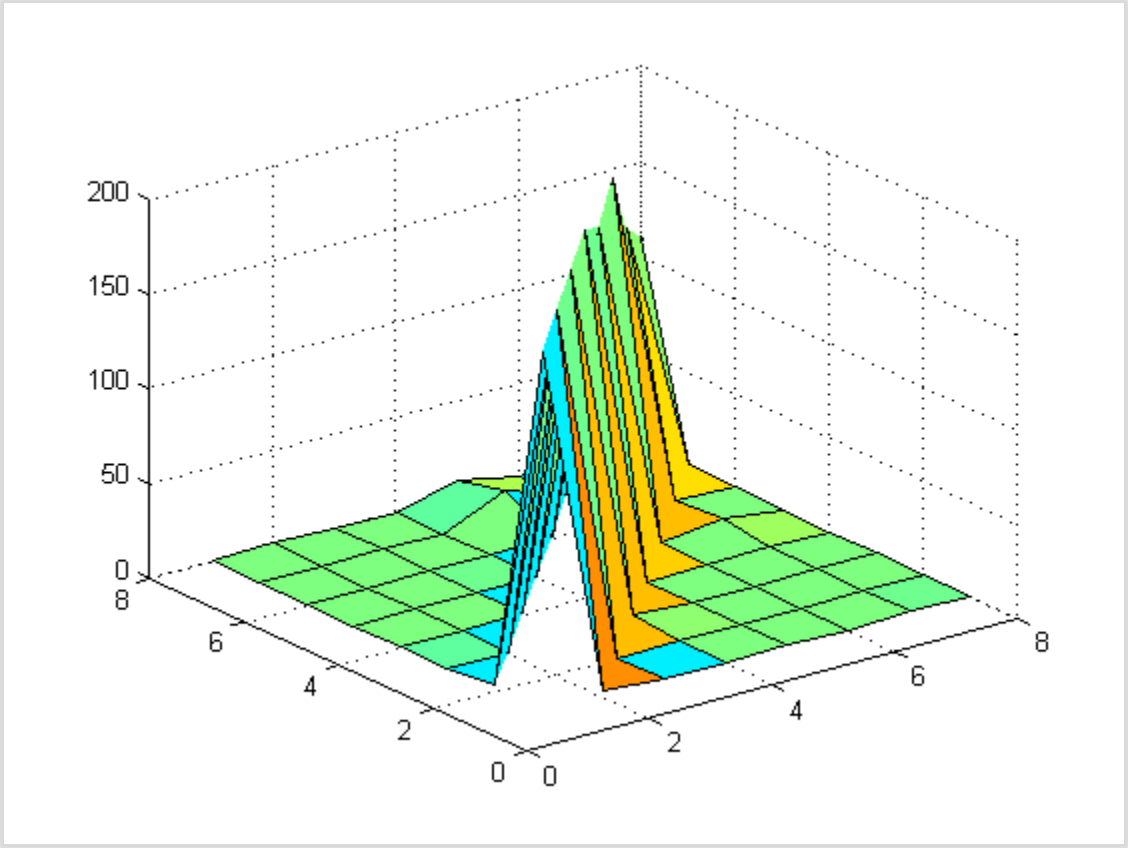


Observations:

- The test accuracy increases with increasing number of train examples because Naïve Bayes learn better with more examples.
- The train accuracy is initially high indicating the over-fitting but later it decreases when sufficient amount of data have been seen.
- The test accuracy becomes almost constant when sufficient number of train examples are used.

e. **Confusion matrix:**

The *confusion.m* script calculates the average confusion matrix for the estimated labels on test data. The plot is shown below.



CONFUSION MATRIX FOR EIGHT CLASSES

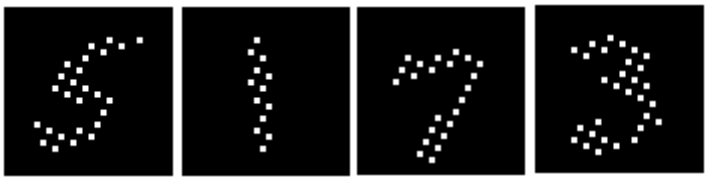
Observations:

- The entries of the confusion matrix varies significantly for different splits of the original data.
- Main observation was that the all diagonal entries were large numbers and non-diagonal entries were small.
- For a non-diagonal entry (I,j), the the entry (j,i) tends to be comparable also.
- For a particular run, ' *rec.sport.hockey* ' newsgroup had the highest diagonal entry in the confusion matrix indicating that these type of examples were least misclassified for the particular random split.
- The '*talk.politics.misc'* and '*talk.politics.guns'* were the maximum confused with each other. They are most confused with each other because they have similar kind of words in them.

Q2. Digit Recognition

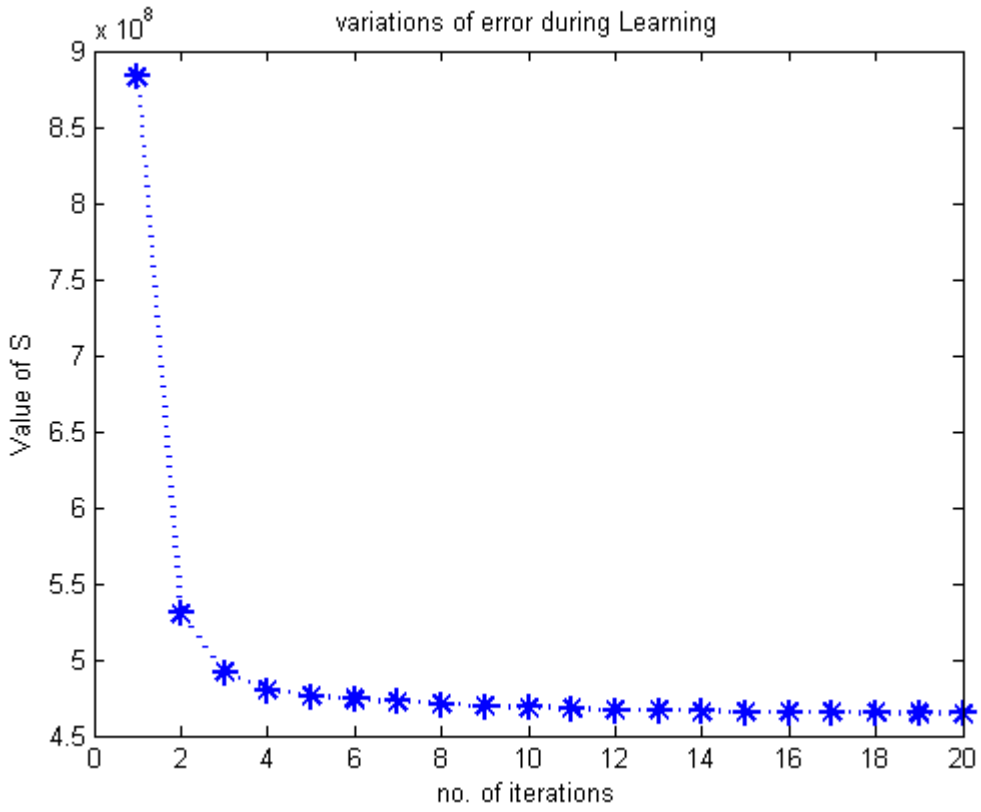
The script *parse.py* maps the text data to integers for efficient processing.

a. **Visualization**:
The *visualize()* function takes a row vector of size 784 as input and shows the corresponding image of size 28 x 28 pixels. A few examples are shown below.



b. **Classifier for digit 1,3,5 and 8**:
I have implemented a general k-means clustering algorithm with k=4 in file ***kmeans.m*** which initializes the clusters with random means. The accuracies on training data obtained are between **74% - 80.2%**.

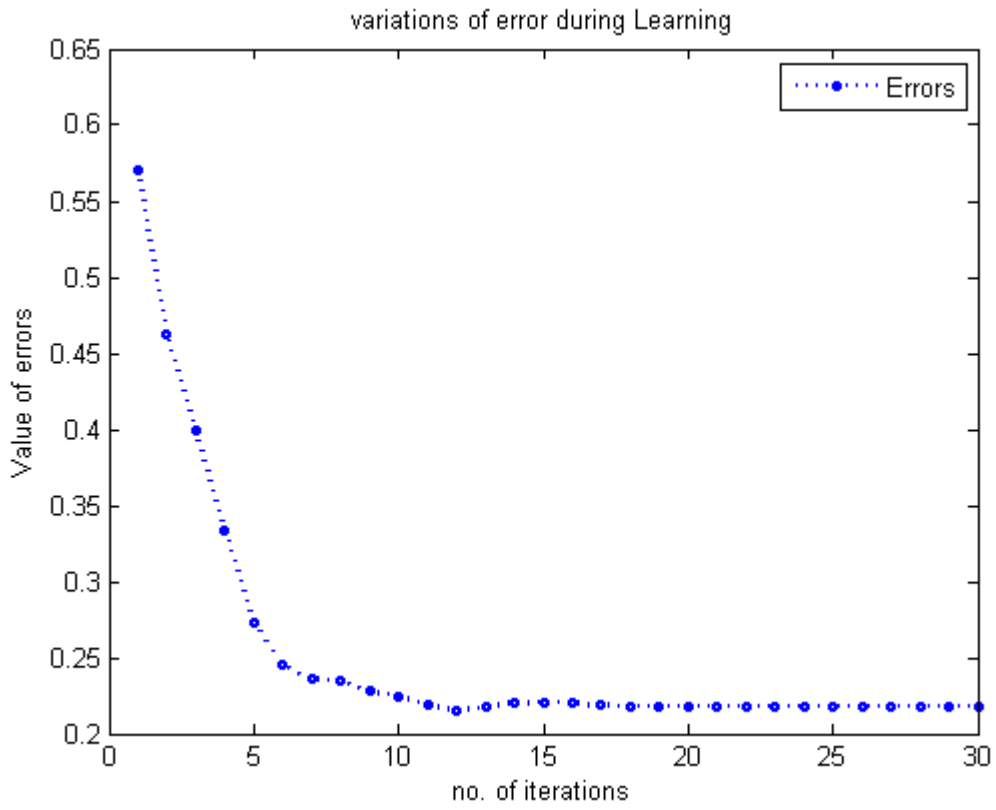
c. **Variation of sum of square distances**:



Obervations:

- The value of S decreases with increase in number of iterations
- S decreases mainly in the first few examples only (in first **10 iterations**).
- Final value of S are variable for different initializations of means which are generated randomly **(min: $.7 \times 10^8$)**.

d. **Variation of Mis-classifications/errors**:
The plot of error values also shows a decreasing behavior with the number of iterations as expected. Plot is shown below.



Obervations:

- Error values decreases with increase in number of iterations.
- Decrease is faster during first few iterations (first 10-12 iterations).
- Minimum error achieved was around **0.22**.
- The reason of such a large error is that kmeans doesn't converges to global maxima and susceptible to local maxima.

Q3. Expectation Maximization:

a. **CPTs and Likelihood with no '?'s:**
Estimated parameters using the fully observed data are calculated. Output the learned parameters in a tabular format for each of the variables in the network and log-likelihood of the test data using the ML estimate of the parameters obtained in the previous step are are present in file *CPT1.txt* shown below:

d^0	d^1
0.5986	0.4014

i^0	i^1
0.6952	0.3048

	g^1	g^2	g^3
i^0d^0	0.298493183449	0.403731164793	0.297775651758
i^0d^1	0.0588235294118	0.255864308914	0.685312161674
i^1d^0	0.90027700831	0.0792243767313	0.0204986149584
i^1d^1	0.50603378922	0.281576830249	0.212389380531

GRADE CONDITIONAL PROBABILITIES

	s^0	s^1
i^0	0.654344073648	0.345655926352
i^1	0.642388451444	0.357611548556

SAT CONDITIONAL PROBABILITIES

	l^0	l^1
g^1	0.104502046385	0.895497953615
g^2	0.390311418685	0.609688581315
g^3	0.990130624093	0.00986937590711

LETTER CONDITIONAL PROBABILITIES

b. **EM algorithm and convergence criteria**:

The file ***2.py*** has the implementation of em algorithm with **E** and **M** steps defined properly.

E step: In E step, we calculate the probability of the hidden variables given the current CPTs i.e. each example with '?' can be divided into many examples each with a weight/probability which is calculated in this step.

M step: In M step, we re-calculate the CPTs using all the weighted examples we have so far.

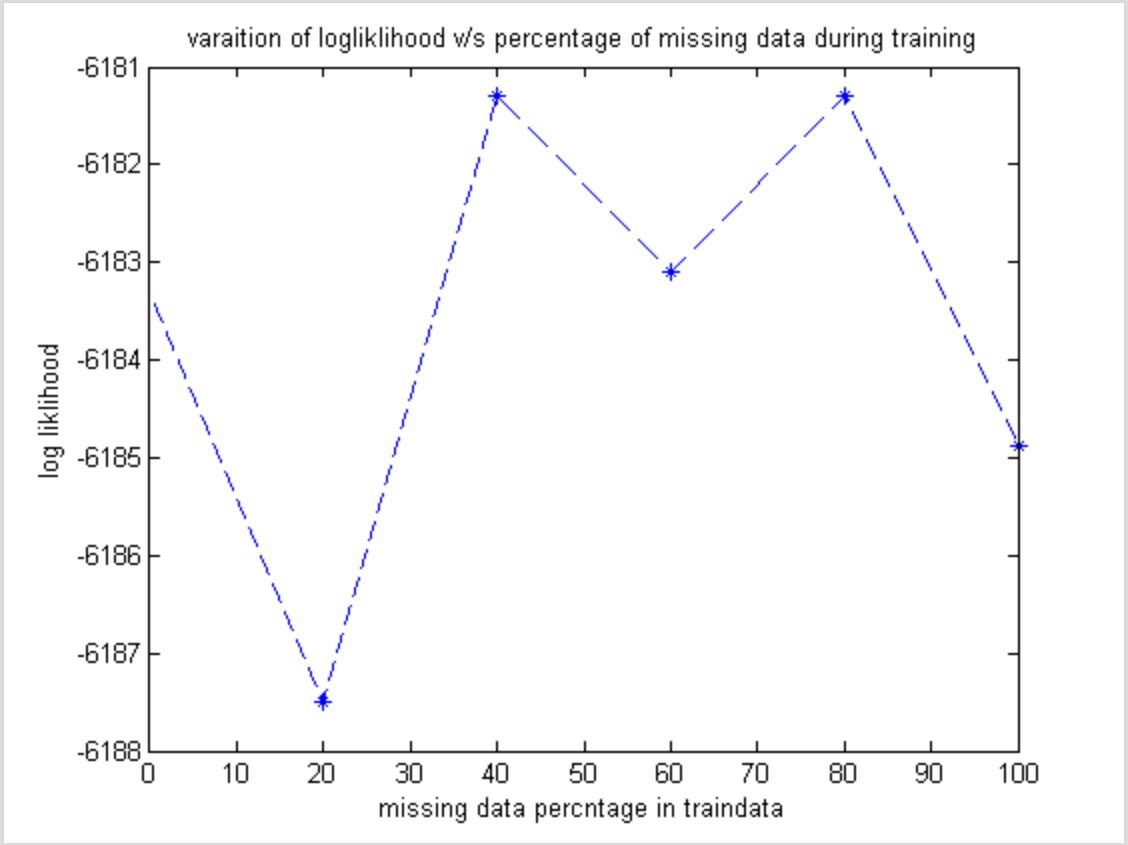
Stopping criteria: The algorithm terminates when the maximum change in weights of the **CPTs** becomes less than eps or a threshold number of iterations have occurred:

$$\epsilon < 0.0001$$

$$L(\theta) = -6183.31194866$$

c. **EM on missing data:**

The files **CPT2.txt** contains the final values of CPTs at convergence for the various kind of train files with missing data. The required graph is plotted for log likelihood v/s the mount of missing data while training.



Observations:

- The plot doesn't show any strong dependence of missing information on the log likelihood and all the 6 values are comparable. For this ambiguous behavior, It can be the case that the test and train data may have been generated using different models
- One reason can be that when little/no information is missing (0% or 20% missing), the model will tend to over-fit according for training data examples and will perform poorly for examples not in training data. But when sufficient missing labels(40%,60%,80%) are present, this over-fitting is avoided since weights of examples not in data are also included due to '?'s while learning.

- But when too much information is missing (i.e. 100% "?"s or all examples have missing information), the strength of learned model decreases and it tends to under-fit due to this the log likelihood also decreases again.

Q4. Face Recognition

a. **Average Face**:
The ***meanface()*** function shows the average image of the dataset of size 19 x 19 pixels as shown below.

MEAN FACE

b. **PCA (principal component analysis)**:
The ***pca1()*** function applies the PCA algorithm and stores the eign faces in the variable *V.mat* and also the folder named *faces/* . The images data's average is zeroed out before calling the svd function.

c. **Eigenfaces Visualization**:
The top 5 eigenfaces after proper scaling were obtained to be

Face no.	Eigen values
1	3.5056e+04
2	1.5449e+04
3	1.1711e+04
4	8.3370e+03
5	7.8369e+03

d. **Projection of images on eigenfaces**:

Projected Images on eigenfaces are almost similar to the parent images. The quality of various images increases with increase in number of face vectors with almost no contribution from lower eigen vectors. Few are shown below.

Original	Projected
	

Observations:

- The projected images are almost similar but more blurred as compared to actual ones.
- Average difference between pixel values of projected images and actual images was less than **10** for all images.

e. **Projection of images which are not eigenfaces**:
The PCA algorithm tries to estimate the closest facial representation for the input image. In some projected images the various parts of mouth are also visible. The projected image tends to be the sum of best matches from our image dataset.

Original	Projected