A. K-means, GMM

1 Synthetic Data

IA. Steps Followed

First collected the data from given files.Passed each class features to first K-means algorithm,whose output means are choosed as initialization for GMM algorithm.We experimented with initialization of K-means algorithm itself by taking either first K(cluster count) data points as initial cluster centroids or took random K data points of given data points.In E-step,we estimated the gamma values using the aforementioned means,covariance initializations.Using these gamma values,we again estimated means and covariance of the gaussian mixtures.We repeated this for 4 to 5 times.

IB. Observations and Results

- **Normal Covariance Matrix :** If we consider normal covariance matrices,we can represent the most variant data distribution direction which is not parallel to given axis.Therefore,with less gaussian mixtures are required to get a good decision surface.For our given data,we got 100% accuracy starting from K = 5 gaussian mixtures per class.For best decision surface,this method took about K = 16 mixtures per classEach gaussian aligned themselves to the distribution both in terms of position and rotation inorder to describe the data in best possible way.Unlike bayesian classifier with unimodal gaussian distribution models,decision surface can be as non-linear as possible according to the data.

- **Diagonal Covariance Matrix :** For diagonal covariance matrix,the major and minor axis of gaussian contours are parallel to coordinate axis.So they can't orient themselves to describe the given data best.But interestingly,the same accuracy of 100% is obtained from just K = 6 mixtures per class although decision surface is not a good as in normal covariance matrix.For good decision surface it took nearly 20 gaussian mixtures in this case.After all,with very less estimations we could get very good accuracy using these diagonal covariance matrices.

IC.Plot Results

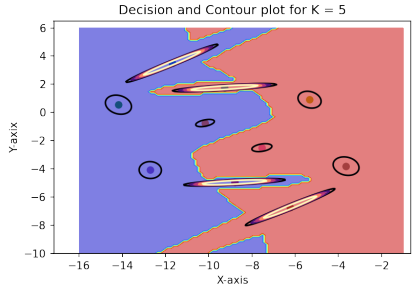

Figure 1.1: Normal Covariance Case

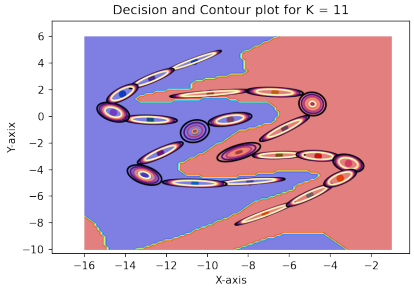

Figure 1.2: Normal Covariance Case

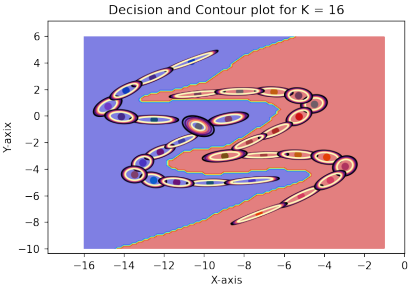

Figure 1.3: Normal Covariance Case

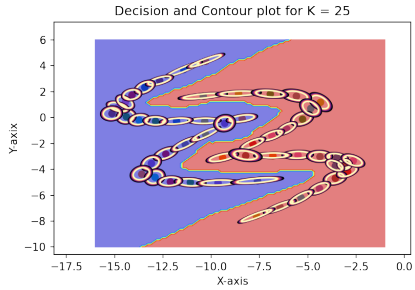

Figure 1.4: Normal Covariance Case

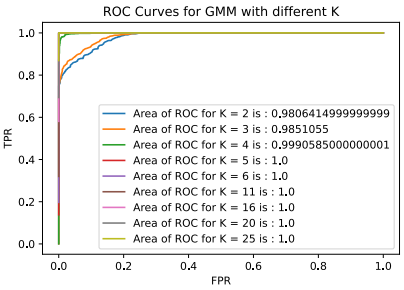

Figure 1.5: Normal Covariance Case

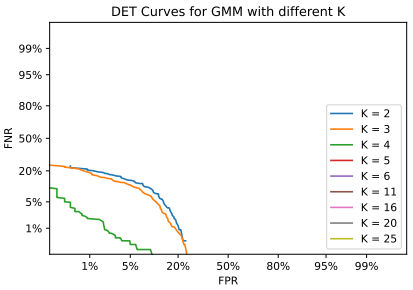

Figure 1.6: Normal Covariance Case

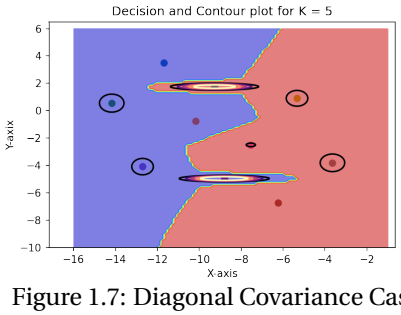

Figure 1.7: Diagonal Covariance Case

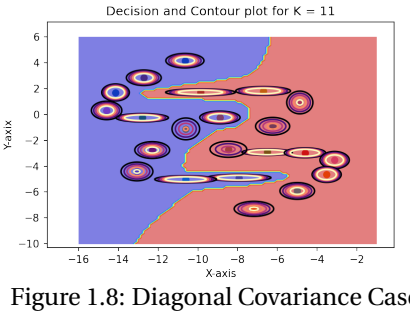

Figure 1.8: Diagonal Covariance Case

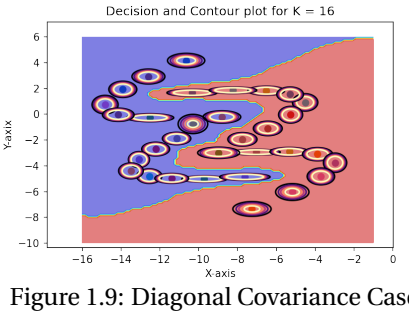

Figure 1.9: Diagonal Covariance Case

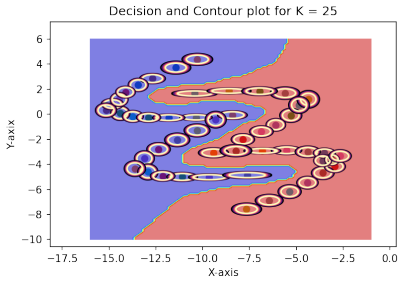

Figure 1.10: Diagonal Covariance Case

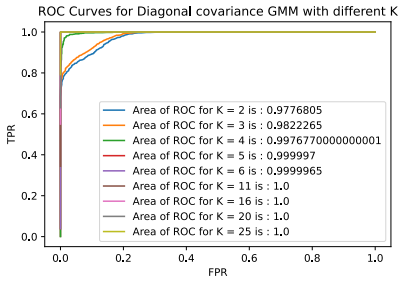

Figure 1.11: Diagonal Covariance Case

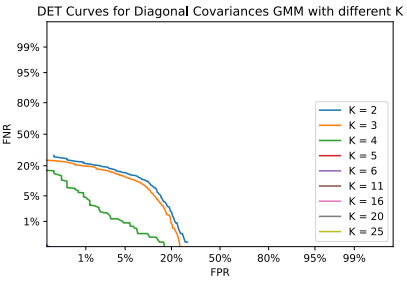

Figure 1.12: Diagonal Covariance Case

2 Real Data

2.1 Procedure

- **Normalisation:**
  - There are 23 features per each block, we have to normalise each and every feature independelty. We clubbed all the training data and found out the mean feature vector and shifted the features vectors so that the new mean becomes zero
  - After doing mean shift, we have to make sure that all the features are in the similar range. So we found out max - min and divided each feature by that. So we got all the features in the range (-0.5, 0.5). Finally we muliplied by 100 to make the range (-50, 50) so that it becomes each for the computer.
- We performed experiments using K(the number of mixtures in each class' model) = 5, 12, 16.
- We tried both diagonal non-diagonal covariance matrices
- We used multiple sets of iterations and reported results for these two combinations (GMM = 7, K-Means = 10) (GMM = 10, K-Means = 50)

2.2 Experimental Resuslts

Set1: GMM = 7 Iterations and K-Means = 10 Iterations

2.2.1 ROC and DET:

We used no. of clusters = 5, 12, 16 and both diagonal non-diagonal covariance matrices.

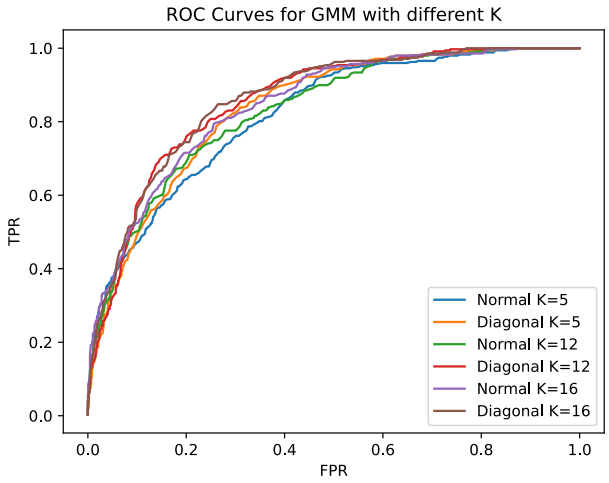

Figure 2.1

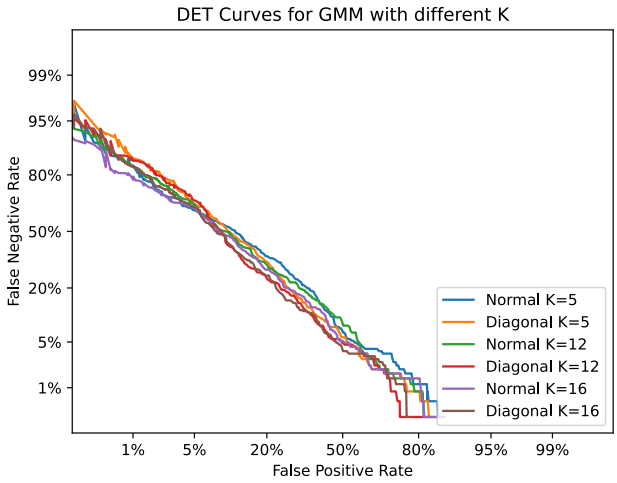

Figure 2.2

2.2.2 Confustion Matrix for Normal Covariance:

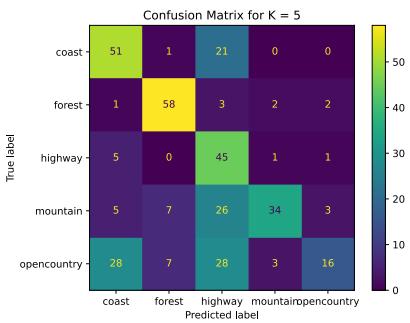

Figure 2.3

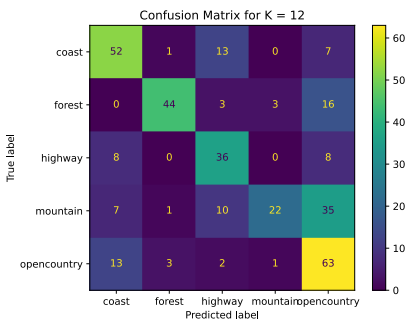

Figure 2.4

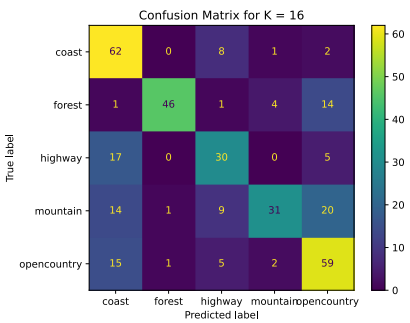

Figure 2.5

2.2.3 Confusion Matrices for Diagonal Covariance:

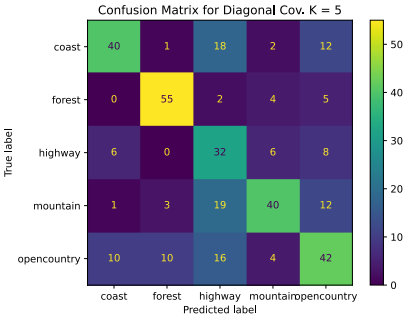

Figure 2.6

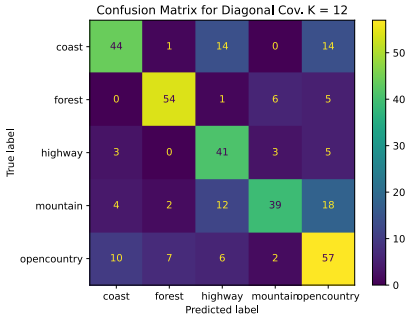

Figure 2.7

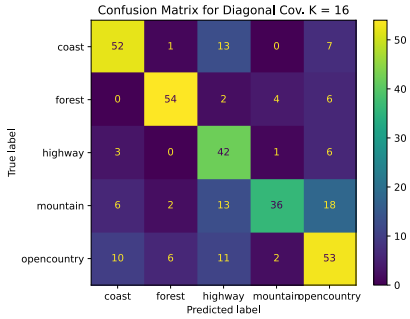

Figure 2.8

Set2: GMM = 10 Iterations and K-Means = 50 Iterations

2.2.4 ROC and DET:

We used no. of clusters = 5, 12, 16 and both diagonal non-diagonal covariance matrices.

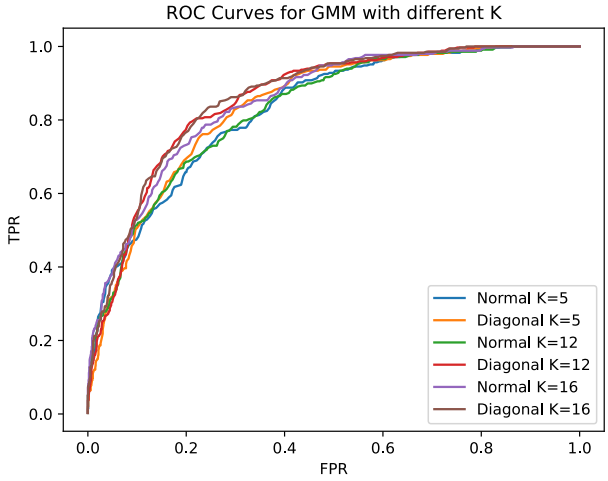

Figure 2.9

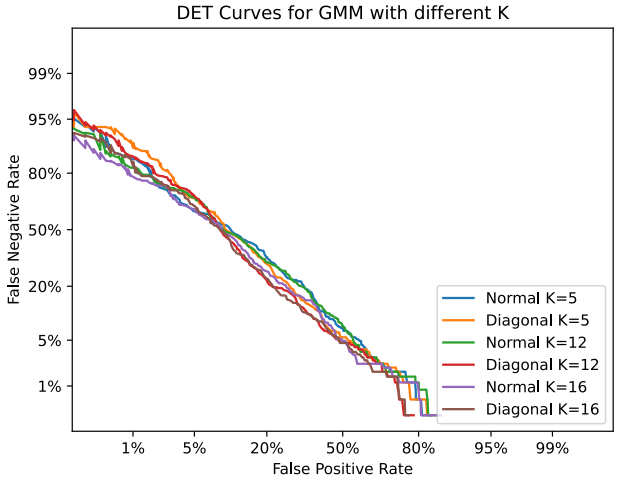

Figure 2.10

2.2.5 Confustion Matrix for Normal Covariance:

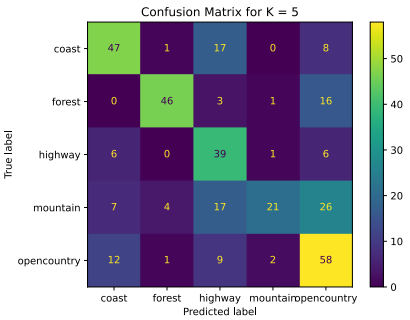

Figure 2.11

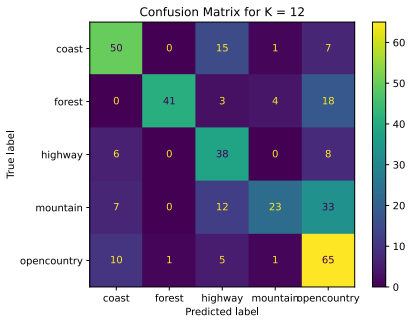

Figure 2.12

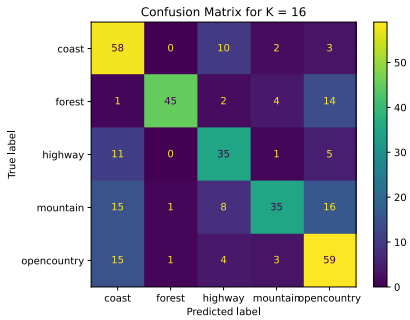

Figure 2.13

2.2.6 Confusion Matrices for Diagonal Covariance:

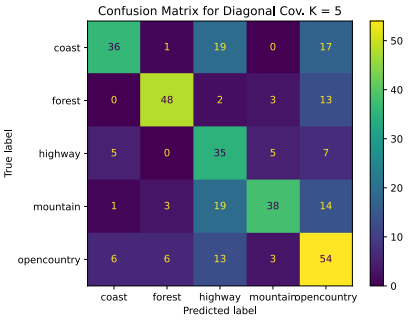

Figure 2.14

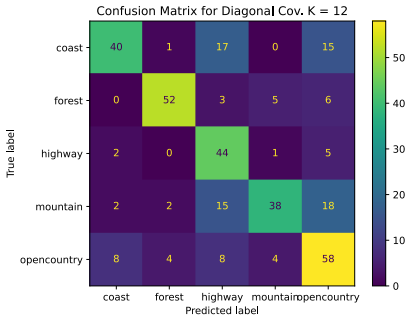

Figure 2.15

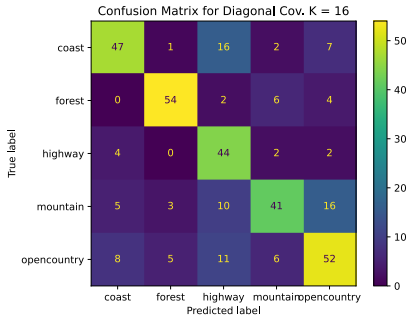

Figure 2.16

2.3 Observations and Inferences

- As we increased number of iterations of GMM & KMeans the accuracy of the model is increased.
- If we don't do feature normalisation python is throwing lot of precision errors, we spent a whole day debugging our code only to realise that the error is not because of the code but because of not normalising the data
- Diagonal covariance performed better in the case of real images.
- We got an accuracy of **68%** for the case: K = 16 GMM iters = 10 K Means iters = 50.
- One of the reasons for not so good accuracy is beacuse different images has attributes of different images, i.e., for example both forest and highway may have trees.
- For larger values of K above 16 accuracy is dropping, 16 is the place where we got highest accuracy.
- If we increase GMM iterations above 10 accuracy is not improving much, it is just taking lot more time. So we got 10 as the ideal GMM iterations for this case.

DTW

1 Online Handwritten-Character dataset

1.1 Procedure:

- **Normalisation:**
  - The letters are placed randomly in space, so we first shifted the characters such that their frames are centered at origin. We can do this by finding average of max and min X, Y coordinates and subracting those values from the respective coordinates of every point.
  - Once the characters are centered. We have to normalise the scale. Some example might be big some examples might be small but scale doesn't matter in our case. So we divided each coordinate such that they are in the range [-0.5, 0.5] finally.
- In DTW we used **window**. Window esentially means, for every point i in template we are allowed to map the point only to a point in the range i-w to i+w. But one catch here is as we must map every point to some point so window must have atleast abs(n-m) size. So window = max(w, abs(n-m))
- For finding the score of a dev data point w.r.t a class we are taking **least k** percentage scores and taking average.
- We did experiments varying k, w values and finding accuracies.
- DTW is taking a lot of time to run. So, we used just in time compilation and written our python code in such way that we can do JIT. This way we achieved an **insane speedup of 54x** (From 29minutes to 32seconds). It became nearly as fast as C/C++.

1.2 Experimental Resuslts

1.2.1 Best K-Scores vs. Accuracy for different window sizes:

For any test point out of all the scores for a given class we took average of least K percentage of scores. Here are the plots showing how the accuracy changes with varying k. We plotted for different window sizes:

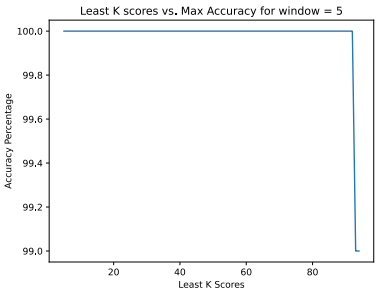

Figure 1.1

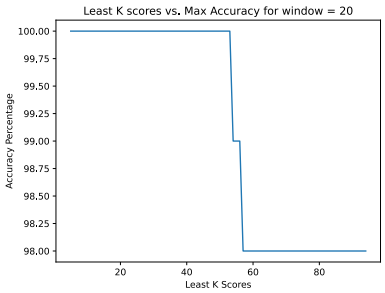

Figure 1.2

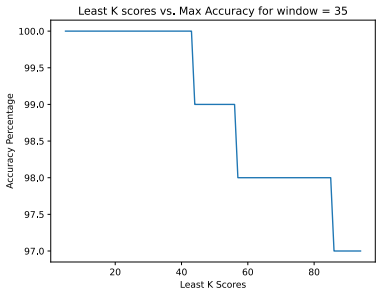

Figure 1.3

1.2.2 Window Size vs. Best Accuracy for that window:

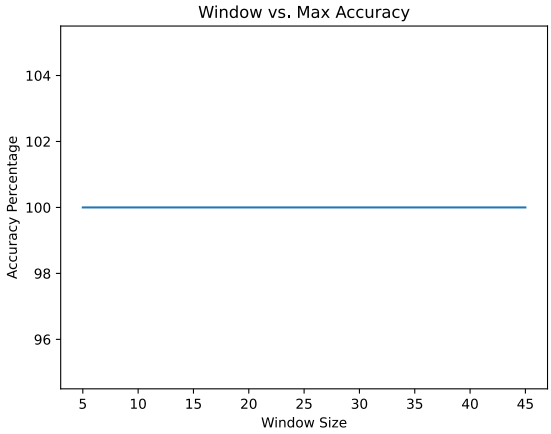

Figure 1.4

1.2.3 Confusion Matrices for different window sizes:

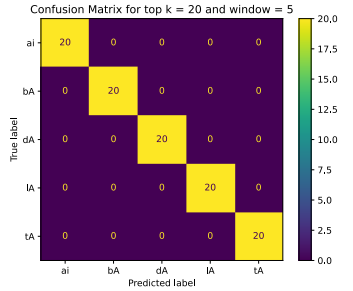

Figure 1.5

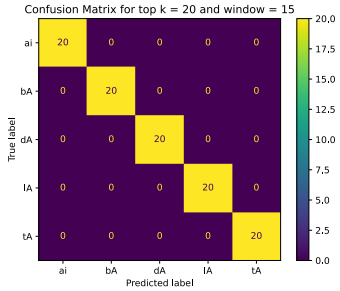

Figure 1.6

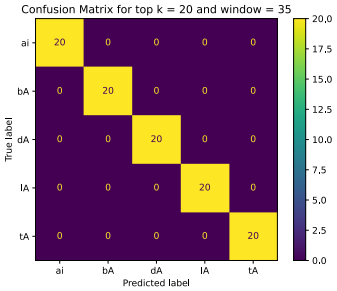

Figure 1.7

1.2.4 ROC and DET curves:

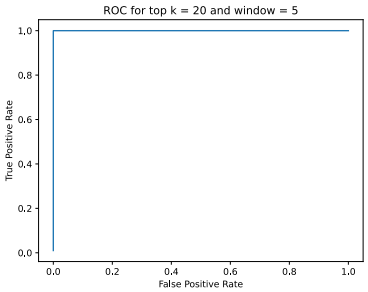

Figure 1.8

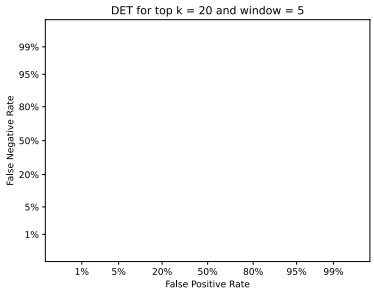

Figure 1.9

1.3 Observations and Inferences

- Best accuracy we got is 100%
- For Figure 1.1, 1.2, 1.3 we can see that as the K (Percentage of best scores we are considering) value increases, the accuracy decreases. Beacause, a dev point may be very similar to some of the training points but not all, so we are ignoring templates to which it is very different from.
- From Fig 1.4, As the window size increases best possible accuracy is exactly same. Only difference is it takes longer time to run.
- From Fig 1.5, 1.6, 1.7 confusion matrices we can see best possible accuracy for the given window sizes is exactly 100% confusion matrix is diagonal

2 Isolated Spoken-Digit dataset

2.1 Experimental Resuslts

2.1.1 Best K-Scores vs. Accuracy for different window sizes:

For any test point out of all the scores for a given class we took average of least K percentage of scores. Here are the plots showing how the accuracy changes with varying k. We plotted for different window sizes:

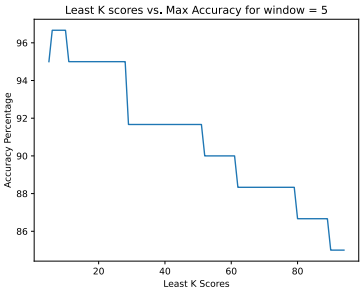

Figure 2.1

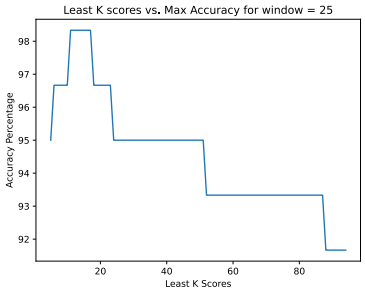

Figure 2.2

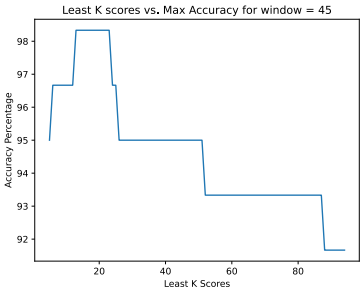

Figure 2.3

2.1.2 Window Size vs. Best Accuracy for that window:

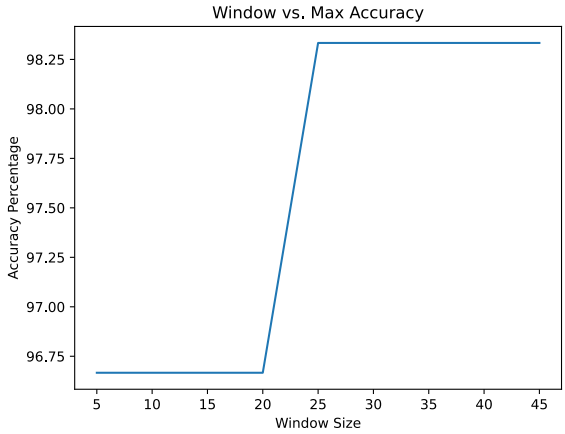

Figure 2.4

2.1.3 Confusion Matrices for different window sizes:

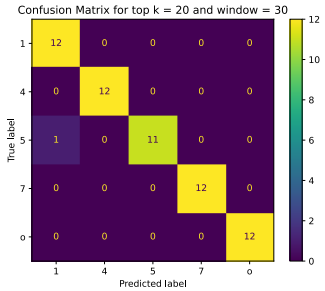

Figure 2.5

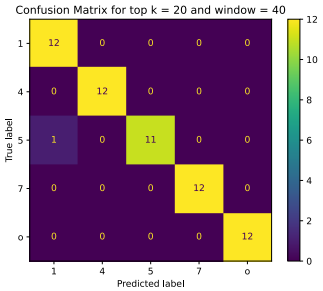

Figure 2.6

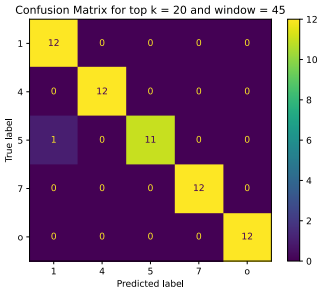

Figure 2.7

2.1.4 ROC and DET curves:

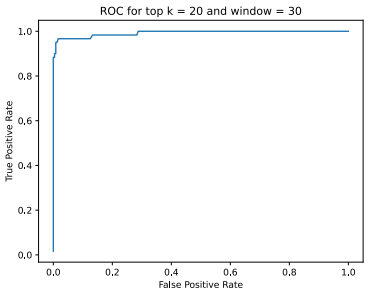

Figure 2.8

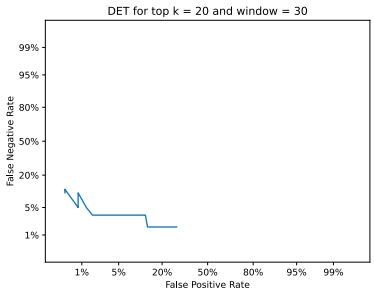

Figure 2.9

2.2 Observations and Inferences

- Best accuracy we got is 98.33333%
- For Figure 2.1, 2.2, 2.3 we can see that as the K (Percentage of best scores we are considering) value increases, the accuracy increases for a while but later decreases. Beacause, a dev point may be very similar to some of the training points but not all, so we are ignoring templates to which it is very different from, also we don't have to consider only very small percentage, it will again cause less accuracy.
- For every window size >= 45 we got 99.8333% accuracy for K = 20% best scores
- From Fig 2.4, As the window size increases best possible accuracy increases from 20 - 25. After 25 it remains constant so waste of time to train.
- From Fig 2.5, 2.6, 2.7 denote confusion matrices with 59 / 60 entries correct.

PART B : DTW and HMM Classification

II. HMM Method

Procedure for Speech and Character Recognition :

- **Reading Data and Normalization:** Collected training data from given files.For character data,since we want to detect character irrespective of their positions and dimensions in euclidean space,we normalized by making center of character as origin to standardise the position of character and scaled so that we get an aspect ratio of 1:1.Now character coordinates belong to $[-0.5, 0.5]$ both in X and Y directions.
- **Vector Quantization :**As Discrete HMM model seeks discrete symbols for training,we need to convert given d-dimensional feature vector or d-dimensional rather observation to a single symbol.For provided HMM-Code directory,we only need to pass symbols belonging to whole numbers.So for this converstion,we explored following methods...
  - K-Means Method : Collected all training observations from all classes and applied K-means algorithm on this cummulative data.Now,to determine a symbol for a new given observation,we simply return the index of the cluster mean to which it is closest(euclidean distance in multidimension).
  - Grid Method for Handwritten Characters : For handwritten characters,since observations are of 2-dimensional,to take advantage of that,we divided the $[-0.5, 0, 5]X[-0.5, 0.5]$ rectangle into NXN grid (we experimented with different values of N like $6, 8, 10$) and we assigned index to each cell of grid.So now we give symbol to a given (x, y) observation as its cell's index number.
- **Using given HMM-Code :** Converted given training observations into symbols of non-negative numbers as said in previous statement for each class.Using this new data,we trained each class HMM model with help of given hmm code and obtained state-transtition probabilities,symbol emission probabilites.As we are assuming left-right HMM model,starting probability of first state is 1 and 0 for remaining states.
- **Finding likelihood probability :** For classification,we need to determine likelihood probability of given test observation sequence for given class HMM model.In other words,we computed the probability of observing given test observation sequence wrt to a class model.Using dynamic programming algorithmic paradigm,we computed this as discussed in class.But the model description is slightly different like here symbol probailities as assigned and particular to given transtition,not state.We adopted the discussed DP algorithm to this case.
- **Classification** : Computed posterior probabilites using likelihood function described above and classified a test observation sequence to a class that has maximum posterior probability.

Observations and Results in Speech Recognition3 :

- Determining best number of states and number of symbols is challenging,because with higher states and symbols,the estimation of probabilites is poor unless we have good size of data.And on the other hand,taking very few symbols and states won't given correct probabilites and therefore,we experimented with multiple states (some same across all classes and some experiments different across classes) and symbols as shown in plots.For us the best recognition is happening for **states 10** (each class) and **25 symbols** with accuracy **98.33**%.

Observations and Results in Handwritten Character Recognition3 :

- Without normalizing the given data as said in procedure,we initally got poor accuracy but after normalization and grid vector quantization,we got an accuracy or true classification rate of **98**%.The remaining 2% misclassification happened because of similar structure of 'lA' and 'tA' characters.

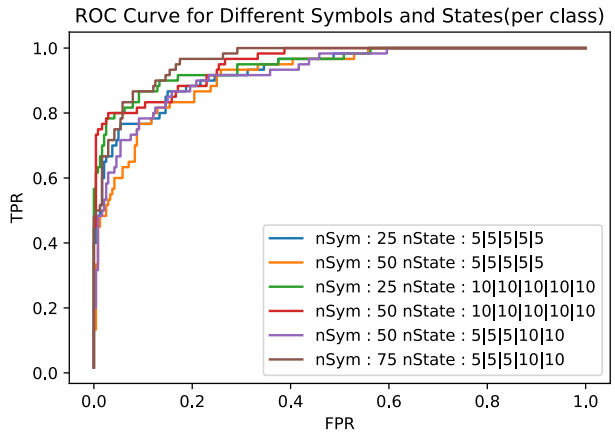

Figure 0.1: Speech Recognition

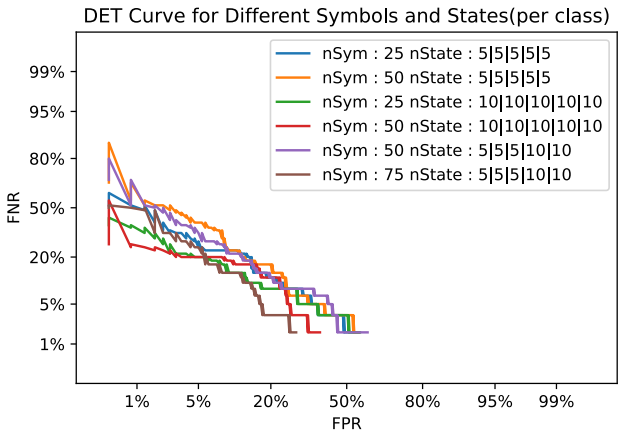

Figure 0.2: Speech Recognition

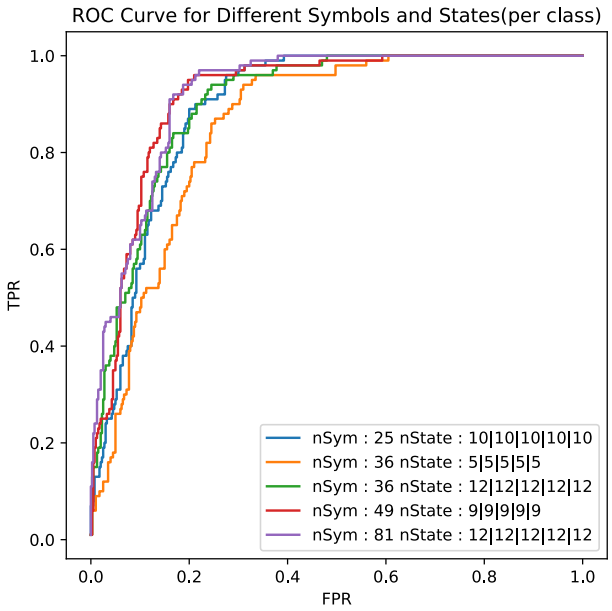

Figure 0.3: Character Recognition

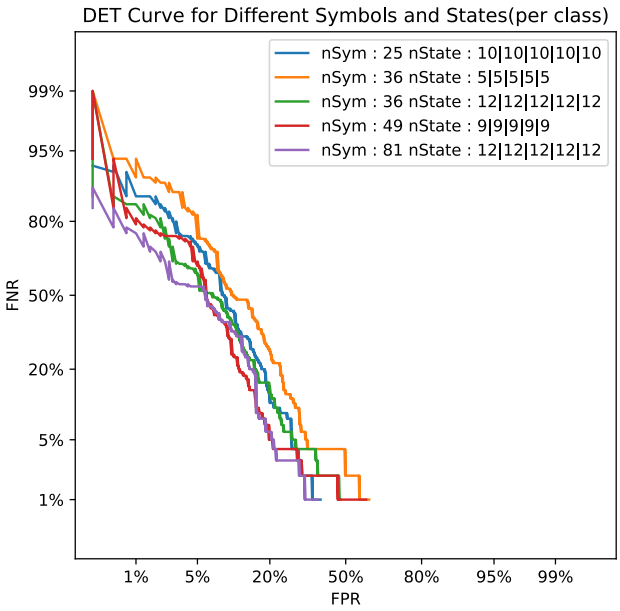

Figure 0.4: Character Recognition

Figure 0.5: Speech Recognition

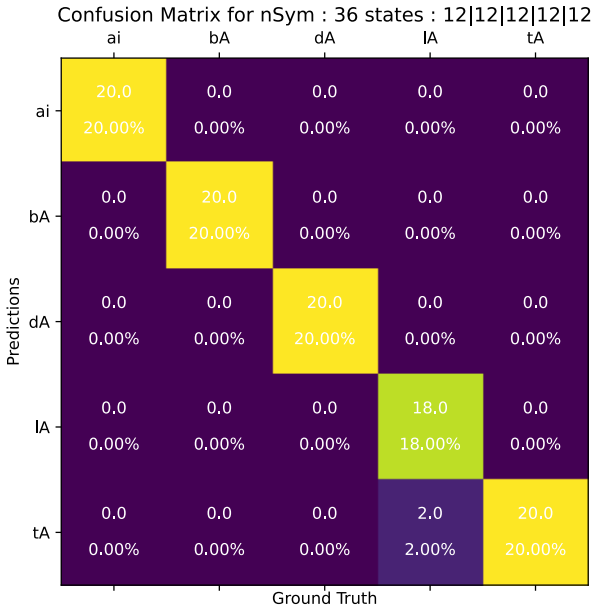

Figure 0.6: Character Recognition