Report Lab 2
Pattern Recognition

NTUA 9th Semester

Efthymios Tzinis 031212007
Konstantinos Kallas 03112057

December 2, 2016

Introduction

In our work we experimented on a typical digit recognition system from speech data provided by the laboratory exercise files. We implemented our system in Matlab according to the steps provided by the description of the exercise. The final result is quite remarkable as the evaluation of our model demonstrated estimable accuracy by using the HMM tool in Matlab. The training and the test data sets are comprised by 15 speakers and 4 speakers respectively, speaking the digits from 1 to 9. Throughout the exercise we found deviant approaches for further increasing the accuracy of our model which are discussed below. Policies and assumptions are also described and analyzed. The implemented model is based on the model described by Rabiner in [2].

Step 11

In this step we used the extracted 13 MFCC's as features for our model and of course we implemented a Hidden Markov Model for the cause of recognition. The HMM we used is a Markovian Chain in which every state is connected only to itself and to the next state which generates a transition matrix A of dimensions $N_s \times N_s$ where $a_{ij} = 0,\ i < j\ or\ i > j + 1$. The initial probabilities for the transition matrix were selected in order to be normalized (every row will sum up to 1, using **normalize()**). Except of the first and the final state where $a_{11} = a_{N_s N_s} = 1$ all the other initial probabilities are considered equal to $\frac{1}{2}$. For the prior probabilities we consider the vector $\pi = [1, 0, ..., 0]$ before calling the Expectation Maximization Algorithm. For a first implementation of our model we used only one Gaussian in order to represent the probability of every state. Because the features we are using are the 13 MFCC's we have to create a 13 dimensional Gaussian which is actually 13 different gaussians implemented in every dimension.

Data Pre-processing

We notice that every speaker has a different way of saying the same words as another speaker. Consequently, the duration of every utterance of the data set is different and because the window length was constant the total number of MFCC's generated were actually deviant for the speakers. We had to find a way in order to make the number of MFCC's for every digit of the train set equal for all the speakers. After trial and error we experimented upon a few different ways of making the same length for the MFCC's features. This was a necessity because otherwise the function **mhmm_em()** could not run properly.

We endorsed a way of finding the maximum duration for every digit according to the train and the test data set. Now for the utterances which have smaller total duration than the max we padded them in order to be equal. Below some of these methods are referred and we discuss briefly the pros and cons of each one.

- **Padding with zeros:** When we try to pad our speech signals with zeros in the beginning of the utterance or at the end of it or equally at both of them we saw an inappropriate behavior in the EM function. Although,

zeropadding generally helps the speech processing applications because it does not change the information in time domain and does not make a more noisy signal in frequency domain either, in our case it ruins the apt behavior of the EM function. This is because in the quefrency domain a stationary sound in the beginning or the end of a utterance will ultimately change the MFCC features because it will destroy the energy of some windows. By minimizing the energy will eventually conclude to extreme values for the respective MFCC's in the start or the end of the utterance. This approach was declared problematic and was abandoned.

- **Padding with Repetition:** Same as above by repetition of the same values in the end or the start of the signal was also devastating for the quefrency domain information by creating negative infinite values for the maximum likelihood in EM algorithm.

- **Padding by mirroring some samples of the utterance:** This approach was substantially better than the two aforementioned methods. In particular, we padded the final part of every utterance with mirror repetition of the same spoken digit. In this way we built a system with minimum data distortion. From now on, the models we will represent are based upon this pre-process policy.

- **Dynamic Time Wrapping (DTW):** This method was not implemented in our models because padding with repetition had proven to be a very good method of pre-processing and it was quite easier to implement. Although, in [1] the algorithm of finding the best path between 2 different sequences with different length is described.

After the mirror padding, we tried to call the function of EM algorithm (**mhmm_em()**) in Matlab in order to train our model. For every training model corresponding to a digit we can see that after some iterations Log likelihood increases and converges to a specific value. There is no use to train our model further because after 5 iterations of EM Log likelihood does not increases further for each model but it causes overfitting to our training data and eventually diminishing our classification ratio on the test data set.

Finally, we trained our model with 11 speakers for all the digits and tested on 4 speakers. The speakers who had invalid utterances for some digits (2 instances in total) were used for training and not for testing.

Important Notice: For the initialization of the Gaussians we followed a more deterministic policy because the random selection of mean value and the variance matrix led our model to fluctuations in the final classification accuracy result. For this purpose we divided our train data set to N_m parts and we initialized every corresponding mean value for every gaussian as the mean value of all these MFCC's. As we saw in the previous laboratory exercise the results when we used the same unitary matrix as variance for all the corresponding Gaussians we achieved higher classifications results. By this intuition we initialized deterministically, all the mean values and the variances for all the Gaussians.

Step 12

For every instance of our training set we used our trained HM models in order to compute the corresponding likelihood of each model. In the matrix below, we can see the likelihoods of all our models and of course the maximum of them which is the final decision of our classifier. Our models in the training phase have been endowed with $N_s = 9, N_m = 1, N_{iterations} = 5$. This configuration demonstrated the best results from the recommended values by the presentation of the exercise. Moreover, in relevant paperwork we saw that the first MFCC is useless and actually decreases our classification accuracy so we used only 12 MFCC's (2-13) for our testing purpose. In the matrix below the misclassified instances which have maximum likelihood in wrong corresponding model it is pinpointed by red color. With this configuration we achieved 94.44%.

Finally, we used another configuration by endowing our model with $N_s = 15, N_m = 2, N_{iterations} = 10$ we achieved an even more remarkable result of classification 97.22%. This result divulges some shortcomings of our previous model because of the lower number of gaussians selected because of the restricted number of states are offered. See step 14 for a better explanation.

Maximum Likelihood Matrix for every Speaker for every digit of the test data set								
Dig1	Dig2	Dig3	Dig4	Dig5	Dig6	Dig7	Dig8	Dig9
-1342.7	-1643.2	-1555.1	-1601.4	-1534.9	-1581.8	-1526.6	-1688.7	-1444.5
-1541.2	**-1346.1**	-1431.6	-1644.6	-1571.3	-1458.7	-1375.9	-1470.7	-1518.9
-1698.5	-1487.0	**-1452.9**	-1806.0	-1734.8	-1580.9	-1681.1	-1636.7	-1687.3
-1408.1	-1530.8	-1521.6	**-1360.0**	-1475.2	-1544.5	-1455.5	-1539.2	-1504.0
-1442.4	-1501.7	-1424.8	-1390.3	**-1261.2**	-1342.3	-1282.3	-1552.3	-1442.7
-1910.7	-1563.9	-1631.7	-1770.2	-1725.8	**-1491.4**	-1623.8	-1821.4	-1788.6
-1758.9	-1513.5	-1581.8	-1680.7	-1578.8	-1466.3	**-1423.0**	-1755.5	-1639.1
-1533.7	-1437.5	-1454.5	-1749.6	-1587.5	-1476.5	-1516.6	**-1331.0**	-1535.7
-1345.0	-1561.3	-1445.6	-1498.1	-1352.4	-1495.3	-1372.6	-1412.4	**-1292.1**
-1373.7	-1606.1	-1540.9	-1656.9	-1572.4	-1601.5	-1516.0	-1660.5	-1446.0
-1567.5	**-1350.6**	-1513.2	-1611.1	-1590.3	-1381.6	-1435.5	-1616.1	-1538.7
-1651.9	-1549.7	**-1380.7**	-1869.2	-1778.9	-1615.8	-1647.7	-1634.0	-1685.0
-1543.3	-1639.3	-1564.1	**-1449.0**	-1676.1	-1653.8	-1676.1	-1673.1	-1832.5
-1550.2	-1726.7	-1700.6	-1458.4	**-1400.3**	-1711.8	-1633.9	-1577.2	-1481.3
-1837.9	-1610.2	-1600.4	-1780.2	-1733.3	**-1387.8**	-1421.6	-1774.3	-1770.7
-1610.7	-1483.8	-1542.2	-1616.0	-1543.1	-1414.6	**-1347.4**	-1634.0	-1579.4
-1399.2	-1324.5	-1322.5	-1610.8	-1464.1	-1339.9	-1403.3	**-1248.9**	-1411.3
-1355.2	-1530.8	-1425.7	-1495.3	-1429.9	-1648.8	-1443.8	-1425.3	**-1339.2**
-1347.0	-1602.9	-1534.5	-1557.6	-1559.9	-1611.3	-1594.3	-1451.5	-1553.4
-1448.0	**-1367.9**	-1372.5	-1421.0	-1487.9	-1469.6	-1425.1	-1509.9	-1477.7
-1638.2	-1530.0	**-1398.1**	-1844.0	-1696.8	-1534.8	-1676.3	-1485.8	-1538.0
-1512.4	-1658.7	-1570.2	**-1392.8**	-1516.6	-1614.7	-1623.9	-1616.0	-1697.6
-1462.5	-1645.2	-1504.5	-1439.0	**-1295.9**	-1537.9	-1474.1	-1534.3	-1431.2
-2043.3	-1851.0	-1887.4	-2090.7	-2066.4	**-1583.5**	-1676.8	-1924.9	-2166.0
-1511.1	-1477.6	-1447.2	-1482.0	-1409.7	-1461.2	**-1405.2**	-1570.0	-1448.2
-1401.2	-1413.7	-1355.8	-1644.8	-1542.9	-1421.8	-1480.0	**-1318.8**	-1415.3
-1353.1	-1425.0	-1393.8	-1539.0	-1462.0	-1395.8	**-1346.9**	-1441.2	-1366.2
-1372.9	-1709.8	-1601.5	-1609.9	-1581.2	-1712.2	-1557.2	-1692.8	-1520.2
-1519.3	**-1280.2**	-1418.9	-1517.2	-1584.7	-1375.7	-1424.2	-1508.5	-1473.5
-1684.0	-1533.8	**-1446.2**	-1785.5	-1696.2	-1546.2	-1568.7	-1473.1	-1579.1
-1516.0	-1727.9	-1667.2	**-1458.3**	-1664.8	-1644.7	-1653.8	-1768.8	-1713.7
-1348.9	-1460.9	-1390.2	-1356.1	**-1283.3**	-1415.1	-1382.5	-1340.3	-1380.5
-1747.2	-1508.4	-1541.4	-1626.0	-1549.5	**-1442.2**	-1478.3	-1579.3	-1626.6
-1504.1	-1569.9	-1497.7	-1474.2	-1438.7	-1455.2	**-1385.3**	-1540.6	-1497.3
-1583.0	-1587.7	-1589.1	-1719.6	-1643.7	-1588.9	-1603.7	**-1492.5**	-1579.3
-1353.9	-1486.5	-1449.3	-1601.1	-1519.1	-1427.2	**-1387.9**	-1443.5	-1397.4

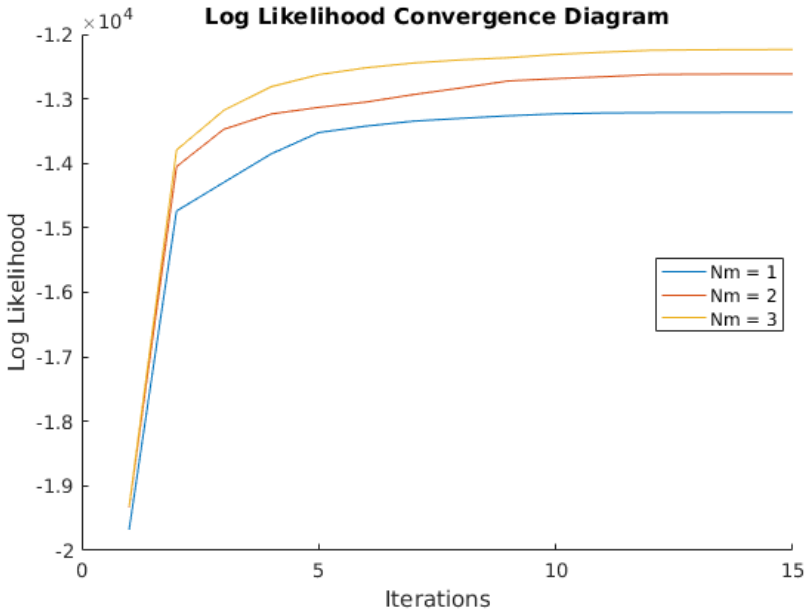
For every model to the corresponding digit we compute the total number of the classified instances. In particular, the models which contain the maximum likelihood for every test instance are selected for the decision of our classifier. In the matrix below the respective matrix is showing the total number of classified instances for the test data set.

	Dig1	Dig2	Dig3	Dig4	Dig5	Dig6	Dig7	Dig8	Dig9
Num of Classifications	5	4	4	4	4	4	5	4	2

Step 13

For this step we increased the maximum number of iterations to 15 (Niter = 15) in order to better visualize the Log Likelihood convergence as the training proceeds. We also plotted the same diagram for 3 different configurations $N \in \{1, 2, 3\}$ in order to get a better view. The diagram is presented below.
Note: Having a higher log-likelihood during training doesn't necessarily mean that the model has better classification accuracy. A higher log-likelihood only means better data-fitting. An HMM decides the classification of every instance only by comparing the values of the likelihoods of every model for the corresponding digit and does not care for the values themselves.

Figure 1: Log Likelihood Convergence during Training



Step 14

In this step we will present the classification success with the best model configurations using a confusion matrix. In general confusion matrices are a good way of representing classification success because they summarize a lot of results in one matrix. However in this specific case where the test occurrences are just 36(!) and the classification success high, the confusion matrix is very sparse and doesn't offer a much better representation than the whole classification result table. Both are presented below for comparison.

Confusion Matrix									
	Dg 1	Dg 2	Dg 3	Dg 4	Dg 5	Dg 6	Dg 7	Dg 8	Dg 9
Dg 1	4	0	0	0	0	0	0	0	0
Dg 2	0	4	0	0	0	0	0	0	0
Dg 3	0	0	4	0	0	0	0	0	0
Dg 4	0	0	0	4	0	0	0	0	0
Dg 5	0	0	0	0	4	0	0	0	0
Dg 6	0	0	0	0	0	4	0	0	0
Dg 7	0	0	0	0	0	0	4	0	0
Dg 8	0	0	0	0	0	0	0	4	0
Dg 9	1	0	0	0	0	0	1	0	2

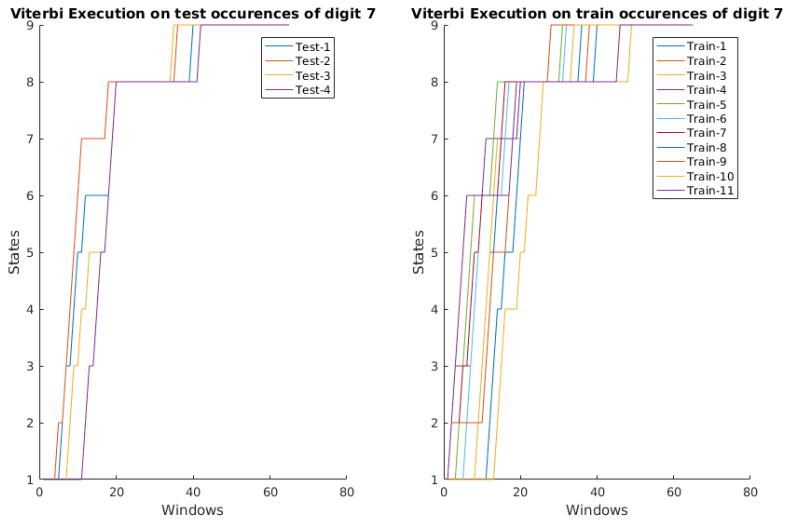
Classification Results				
	Classified as			
Dg 1	1	1	1	1
Dg 2	2	2	2	2
Dg 3	3	3	3	3
Dg 4	4	4	4	4
Dg 5	5	5	5	5
Dg 6	6	6	6	6
Dg 7	7	7	7	7
Dg 8	8	8	8	8
Dg 9	9	9	7	1

The total classification recall for our initial configuration is **94.44%** which is a remarkable result. After that, we tried other configurations and we saw that an increased number of states in the HMM could actually increase our classification ratio because of the nature of our data set. By using more states in our Hidden Markov Model and a linear mixture of multiple Gaussians as predictors for every state, more iterations are demanded for achieving higher classifications ratios. We had to initialize the mixture of 2 gaussians by different mean values from the train data set because if we initialised them with the same then the two Gaussians would demonstrate a totally correlated demeanor concluding to only one gaussian. For this reason we split the data in 2 and we used the 2 mean values from our MFCC's features for every digit to initialize the mean values of the corresponding digits models. For the initialization of the Variance of both gaussians we used the diagonal unitary matrix in order to properly let the EM algorithm to operate on it and change it adequately. By all these we obtained a supreme classification result of **97.22%**, with only one misclassified instance over all of our test data set.

Step 15

In this section we executed the Viterbi algorithm in order to find the most probable state sequence for the train and test data. The most probable state sequences for all the occurrences of digit 7, are plotted below in two diagrams one for the train and one for the test data.

Figure 2: Most Probable State Sequence for each occurrence of 7



The differences of the above diagram for each occurrence can be attributed to the time length of each occurrence. Slower occurrences stay for longer periods of time on the same state while the faster ones are changing constantly.

Folder Code

- **lab2.m** Preparation file (Demanded for running) for extracting all the MFCC's features.
- **lab2_main.m** File for the Laboratory Exercise, Training HMM's and testing our models in the train and test data sets.
- **find_mfccs.m** Preparation file, Self Explanatory.
- **triangly.m** Preparation file, triangular filter implementation.

References

[1] Sakoe,H. and Chiba, S. Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. on Acoust., Speech, and Signal Process., ASSP 26, 43-49 (1978).

[2] B.H.Juang, R.Rabiner, Hidden Markov Models, Technometrics, Vol. 33, No. 3. (Aug., 1991), pp. 251-272.