CS 655: Analyzing Sequences
Homework 4

Part 1: HMM with separately trained GMMs

The observations are divided into 5 classes based on the 'gold-standard' labels. A GMM is trained for every class with 16 mixtures each. The starting probabilities and transition probabilities are assumed to be equal for all states. The GMM parameters of all 5 classes are plugged into the HMM model. The overall accuracy obtained using this model is 55.48%.

Part 2: HMM with separately trained GMMs, with transition matrix calculated from 'gold-standard' labels

Here the transition matrix (given as an input for HMM) is calculated from the 'gold-standard' labels. By adding this information the accuracy is increased to 71.89%. This increase in accuracy is due to the addition of extra information as to which state has higher probability to follow a given state. This gives more weight to a state that is more probable, leading to higher accuracy.

Part 3: Plotting posterior probabilities

The posterior probability of each state for each observation in the range 6000:7000 is shown in fig. 1.

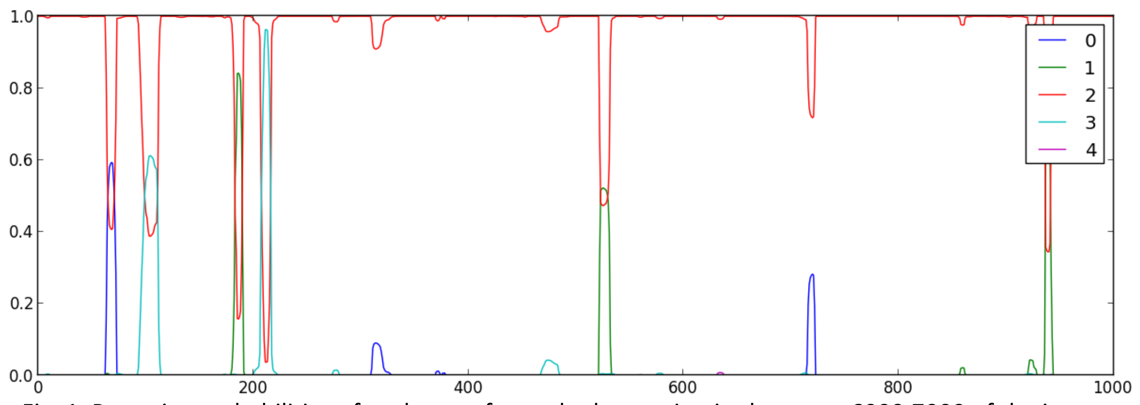


Fig. 1. Posterior probabilities of each state for each observation in the range 6000:7000 of the input

Posterior probability is a measure of the responsibility a component of the mixture takes in explaining an observation. The observations in the range 6000:7000 belong to component 2 (from 'gold-standard' labels). That is the reason the red line in the graph (representing component 2), has a posterior probability of one in most of the observations. In some observations, there are downward and upward peaks. In these areas the posterior probability of component 2 is reduced and proportionate increase is observed in the posterior probability of another component. These are observation points near the intersection of two GMMs. There is a contribution from both GMMs. The component of the HMM which has the maximum posterior probability for a given observation, is taken to be the component in which this observation is seen. The places in the graph where other components have higher posterior probability than component 2 are the places where the model predicted wrong output.

Part 4: HMM from GMMs trained with corrupted labels

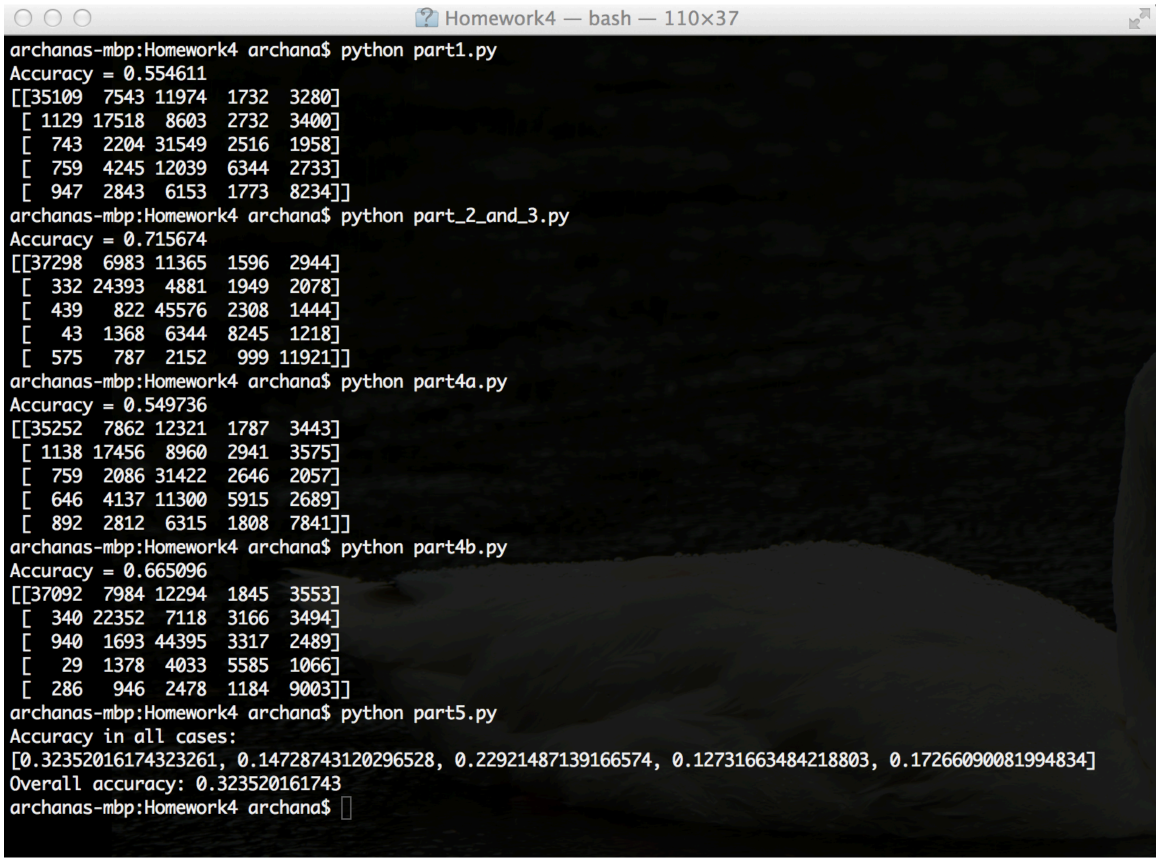
For this part, the experiments in part 1 and 2 are repeated by training GMMs using the corrupted labels and calculating the transition probabilities also from corrupted labels. A similar trend of increase in accuracy is observed by adding the transition matrix information. Accuracy with equal transition probability to all states is 54.88, while with calculated transition matrix it is 66.37. There is a decrease in accuracy compared to earlier experiments as the GMMs are trained using corrupted data.

Part 5: Training HMM using .fit()

In this part the HMM was trained using EM (.fit()). The accuracy obtained using this model is 44.52%, which is lower than that obtained by training individual GMMs. In the previous cases, individual GMMs were trained with data that belonged only to that class alone. This led to a better approximation of the GMM parameters. But here the HMM parameters are estimated using expectation maximization algorithm on all the observations. Therefore, the appropriateness of estimated values is lesser compared to the previous case. The accuracies obtained in all experiments are summarized in table 1 and sample terminal output displaying the accuracy and confusion matrix in different experiments is shown in fig. 2.

Table.1 Accuracy obtained in different experiments

HMM	Transition matrix	Accuracy (%)
Individual GMMs gold standard	Equal	55.48
	From gold-standard labels	71.89
Individual GMMs Corrupted labels	Equal	54.88
	From corrupted labels	66.37
HMM using .fit()	-	44.52



```
Homework4 — bash — 110×37
archanas-mbp:Homework4 archana$ python part1.py
Accuracy = 0.554611
[[35109  7543 11974  1732  3280]
 [ 1129 17518  8603  2732  3400]
 [  743  2204 31549  2516  1958]
 [  759  4245 12039  6344  2733]
 [  947  2843  6153  1773  8234]]
archanas-mbp:Homework4 archana$ python part_2_and_3.py
Accuracy = 0.715674
[[37298  6983 11365  1596  2944]
 [  332 24393  4881  1949  2078]
 [  439   822 45576  2308  1444]
 [   43  1368  6344  8245  1218]
 [  575   787  2152   999 11921]]
archanas-mbp:Homework4 archana$ python part4a.py
Accuracy = 0.549736
[[35252  7862 12321  1787  3443]
 [ 1138 17456  8960  2941  3575]
 [  759  2086 31422  2646  2057]
 [  646  4137 11300  5915  2689]
 [  892  2812  6315  1808  7841]]
archanas-mbp:Homework4 archana$ python part4b.py
Accuracy = 0.665096
[[37092  7984 12294  1845  3553]
 [  340 22352  7118  3166  3494]
 [  940  1693 44395  3317  2489]
 [   29  1378  4033  5585  1066]
 [  286   946  2478  1184  9003]]
archanas-mbp:Homework4 archana$ python part5.py
Accuracy in all cases:
[0.32352016174323261, 0.14728743120296528, 0.22921487139166574, 0.12731663484218803, 0.17266090081994834]
Overall accuracy: 0.323520161743
archanas-mbp:Homework4 archana$
```

Fig. 2. Screenshot showing accuracy and confusion matrix in different experiments