Computer Science 315
Assignment 5

2014

Deadline: Friday, 16 May 2014

Overview

In this assignment, you will train Hidden Markov models (HMMs) using the Viterbi re-estimation procedure. You will then use the HMMs to perform classification on the speech and signature data sets used in earlier assignments.

Implementation

In this project you will implement the two basic algorithms of the HMM: The EM algorithm for training the HMM, and the Viterbi algorithm for finding the optimal state sequence given a model, i.e. given the transition probabilities, and the state probability density functions (pdf's).

Although the pdf's can potentially be anything, in this project you will only use a single multivariate Gaussian pdf for each state. This means that for the pdf of each state you need to estimate the mean and covariance.

Although you will implement the basic algorithms you are provided with a few useful utilities. Since you will train the HMM using Viterbi re-estimation, you need to implement the Viterbi algorithm first.

Note examples with sample output are provided in the doctrings of each function/method.

Note that due to concerns of numerical overflow and underflow in calculations, it is advisable to perform as many of your calculations in the log-domain as possible. This includes working with (negative) log-likelihoods rather than likelihoods, as well as with the log-determinant of covariance matrices rather than the determinant, where possible.

Investigation

Speech data

Add a Python module called `speech.py` to the project. In this module, place code to perform and document the topics below. Your module must be set up so that running the module outputs the results and any comments you have on the results.

1. Load the data in the file `data/speech.dat`. This is the same pickled dictionary used in the previous assignment.

2. Normalize the training and test data by subtracting the mean of all the 16-dimensional observations, of all the training signals, and scaling each component so that the standard deviation of the component over the training set is 1.

   Note: In general one has to be careful about normalizing the data. Inappropriate normalizations can easily destroy the distinguishing features between models. The normalization recommended here is purely for numerical reasons. For this reason the normalization is done across all observations, of all signals of all the different diphthongs in the training set.

3. We shall assume that each $n \times 16$ array is an independent signal generated by an underlying hidden Markov model, which is the same for all signals from a given diphthong. Thus, we can use all the signals for a specific diphthong in the training set to estimate the parameters of the HMM for that diphthong.

4. Once a model of each diphthong, is available, as well as information of the prior probability of each diphthong, one can calculate the most likely class for a given observed signal. This is done by appropriately adjusting the negative log-likelihood of each model for the given observation to take the prior proportion information into account. (Hint: you can use the `calcstates` — or `viterbi` — method to obtain the negative log-likelihood of the observed signal for each model.)

5. Apply the classification approach described here to both the training and the test set, using a non-uniform prior on the proportions. (Cheat a little by using a prior which matches the label distribution on the test set.)

6. Perform the classification approach described above for K from 1 to 6, using both full– and diagonal covariance matrices.

7. Plot typical training and test error rates calculated in the last step against K on a line plot, and try to explain what you see. (Note that performance on the test set can get worse for larger values of K, even though more hidden states are being used to model the data.) Which combination `(K,diagcov)` gives you the best error rate on the test set?[1]

8. Compare the best performance you achieved with the HMM approach to that you obtained with the GMM approach.

Signatures

Add a Python module called `sign.py` to the project. In this module, place code to perform and document the topics below. Your module must be set up so that running the module outputs the results and any comments you have on the results.

[1] In practice, we can not use our test set to select our model parameters as we do in this assignment. Instead, the training data is segmented into a training and a validation set, where the validation set gets used for model selection.

1. Apply the investigation approach described above to the signature data included in the resources file in order to perform signature classification. However, note the following differences:

   - You should pre-process the data as described in assignment 1, instead of normalizing it.
   - Use a uniform prior for the identity of the person signing.
   - Use the first three signatures of each person for training, and the last twelve for testing. You may also use a leave-one-out procedure.

Further investigation

Perform further investigation as you see fit, and write a short report on your findings.

Some ideas for further investigation in this assignment:

1. Investigate a good way to automatically detect a good value for the number of states to use in the HMM models. What impact does it have on your classifier performance if you allow the models for different diphthongs/signatories to have differing numbers of states?

2. Investigate the effect of pre-processing the training and test data for these problems in different ways.

3. Viterbi re-estimation is a specific implementation of a general EM algorithm. If you have already implemented a generic EM algorithm, can you use this framework to implement Viterbi re-estimation?

4. Visualize the states of an HMM fitted to a set of signatures, by plotting shaded ellipses to represent the various components over a plot of a sample normalized signature. Consider colouring the portions of the signature in each state the same as the corresponding ellipse.

Evaluation

1. Implementation: 30 marks (`hmm`: 15; `viterbi`: 15)

2. Investigation: 30 marks (speech: 15; signature: 15)

3. Report: 40 marks