CS545 2023 – PROBLEM SET 4

This problem set is due on the last day of classes. This is an optional problem set that you can use to perk up your grade, and is also necessary if you want to get an A+. This time you can use third-party toolkits for complex algorithms like Gaussian Mixtures or Hidden Markov Models, of MFCC feature extraction. I would still prefer that you implement simple things like K-Means yourself (and feel free to impress me with your implementation of GMMs or HMMs as well, they are not that difficult once you have K-Means).

Problem 1. Some clustering

For this problem download a snippet of a soundtrack from a TV sitcom episode (friends.wav). The goal is to automatically segment it to three sound classes, "Music", "Speech", and "Audience Laughter". Because we are too lazy to make classifiers we will use clustering instead.

Find a reasonable representation for this data and make sure that when you visualize it you can roughly see the difference between the three classes. You obviously have a ton of options, e.g. spectrogram, MFCCs, etc. Pick whatever you like.

You now will have to cluster that data. The idea is that the feature vectors that correspond to speech will be most similar to each other than the other classes, and so on. Try three different things; K-means clustering, GMM clustering and a 3-state HMM (one state for each sound class). Each of these approaches will label each feature vector as belonging to a cluster. Also provided is the file labels.txt which contains the start and stop time for each sound class. Compare your clustering results to that. Make some notes on how the three algorithms compare, speculate on why they differ the way that they do and provide a measure of their accuracy based on the ground truth.

Some hints:

a) You should ignore the input frames where the energy isn't significant, these will be frames containing silence and we don't really care to label them as being different from their neighbors.

b) If you try to learn a GMM on the high dimensional spectrogram data you will most likely encounter numerical errors (the covariance will not be invertible). You have two things that can help here, you can always use PCA to reduce the dimensionality, and you could also estimate diagonal covariance matrices, which are a little more forgiving.

c) You don't have to learn a full HMM, you can simply use the GMM Gaussians as state models and impose an appropriate transition matrix so that the label sequence doesn't flip states too much. This means that you simply need to run the Viterbi algorithm on the GMM posteriors using the transition matrix.

Problem 2. Recognition of temporal sequences

For this problem you have to create a simple spoken digit classifier. Unfortunately, you will also have to gather the data for it!

To solve this problem you will use Hidden Markov Models. I will be thoroughly impressed if you write your own code for HMMs, but if you want to spare yourself the agony I suggest you use an existing toolbox.

To generate some training data make a recording of yourself saying each of the ten digits (0 to 9) twenty times. For each digit keep fifteen of these utterances as training data, and five as test data. Make sure that your pronunciation/intonation is consistent and that there is not an excessive amount of background noise or reverberation.

Just as before you need to find a good representation, since the time samples aren't that useful. You can use features such as MFCCs, or a log magnitude spectrogram followed by PCA dimensionality reduction, or whatever else you prefer. You will train one left-to-right HMM for each digit. You can start with six states for each HMM and four Gaussians per state, but feel free to optimize. I suggest diagonal covariance matrices, but it could work otherwise as well (again, remember that for high dimensions full covariances won't work well, you should either reduce the dimensionality, or use diagonal or unit covariance matrices).

Show me the confusion matrix for the training data and the testing data. Try to optimize the process so that you get the best possible performance. Note that this time it might be that your data is the reason for poor performance. Be sure to not blame the algorithm immediately (remember: garbage in – garbage out).

Problem 3 . Activity recognition

For this problem you have to make a human activity recognizer from cell phone accelerometer data. Download the HAR data from: https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

We will need to distinguish a user's activity from these four classes: "Walking", "Walking Upstairs", "Walking Downstairs" and "Sitting" (we'll skip the "Laying" class). Ignore all the post-processed data in the archive and proceed to load the raw data from the files named `total_acc_*_test.txt`. Keep only the sequences from the above four classes. You should end up with 1878 sequences of 128 time samples of 3 dimensions (x,y,z triplets). Split these into a training and testing set (use a random 50%-50% split).

This time we will use a linear dynamical model for the classifier. Since the data is multi-dimensional (x,y,z) you will use a VAR model (Vector Auto Regressive). With this model we use the past N, 3-dimensional samples to predict the current 3-dimensional sample. Construct an appropriate input and output matrix to learn a linear mapping that does so.

The input matrix will have in its columns a concatenation of the past N samples, and the corresponding column of the desired output will have the current sample. For each column containing the past samples add one more dimension which will always have the value of 1 (the bias term). Solve this linear system and for each class find the matrix that best predicts the current sample from the past. Play around with various values of N to get the best results.

Now you can use the learned models to see how well they fit your test and train data. For each input sequence, predict every sample using its past N samples. See which model results in the least error in predicting each sequence, and classify that sequence as belonging to that model's class. Show me the confusion matrices for the training and testing data.

Enjoy!