Lab 8: Classification

Part 1. Making a speech detector

In this section we will design a simple classifier that will let us know if its input is speech or non-speech. Get the following set of sounds:

http://courses.engr.illinois.edu/cs498ps3/sounds/SpeechMusic.zip

In this archive you will find two directories, speech/ and music/

Randomly select 50 soundfiles from each directory to use as training data, and use the remaining sounds as testing data.

For all of the sounds we will compute a representation that makes the classification easier and we will use a simple Gaussian model to classify them. Do the following:

1. Perform an STFT for each sound, take it's magnitude and raise it to 0.3 to improve contrast

   We will consider each column of that to be a data point

2. Using the training data of each sound:

   Calculate the mean column and the diagonal covariance of the columns[1]

   You will thus get two sets of Gaussian parameters that model each sound class

3. For each testing data point:

   Calculate the likelihood of each column based on the above models

   To calculate the entire file likelihood add all the frame likelihoods

   Assign each soundfile to the class that gets the highest likelihood

For extra credit implement the parameter estimation and model likelihood yourself. If you are too lazy for that you can instead use `sklearn.mixture.GaussianMixture` to learn a diagonal single-Gaussian model per class.

How do the results look like? If you rerun this with a different training/testing set, is there an appreciable difference? On average over multiple training/testing sets what accuracy do you get? Can you think of a way to improve it?

Part 2. Making a music genre classifier

We will repeat the above, but this time we will perform music genre classification. To do so we will use a slightly more elaborate feature representation, and a stronger classification model. Get a subset of the CTZAN genre collection data from:

http://courses.engr.illinois.edu/cs498ps3/sounds/MusicGenres.zip

[1] The diagonal covariance of the data is a diagonal matrix that contains the variance of each dimension in its diagonal

Just as before, you will find a set of directories with examples of each sound class that we want to recognize. For each class, split the soundfiles into a training set (50% of data) and testing set (remaining 50% of data).

For a representation we will use MFCC features. For extra credit, code these yourself otherwise you can use the implementation from the `librosa` library. Once all the files are transformed we will have a series of MFCC frames for each recording (as opposed to spectral frames as is in the case of the STFT). We will use these as the data to classify.

For each class learn a Gaussian model (with a diagonal covariance again). This will be the same process as above.

In order to evaluate how good this works we will use the following procedure. For each sound in the training data, get the likelihood of each MFCC frame based on the learned Gaussian models and sum these over the entire file just as we did before. Use the resulting values to get a classification result for each . Report how accurate your results are. Now report the accuracy using your testing data instead.

Now will use a better classifier to hopefully get better accuracy. We will use a Gaussian Mixture Model (`sklearn.mixture.GaussianMixture`). Just as before you should learn one such model for each class using the corresponding training data.

How many Gaussians do you need in your GMM to get the best results? Do the MFCC parameters make a difference? Play around with the numbers to get the best possible results.