Problem Set 2

As always, any time that you create a plot to turn in, be sure to label your axes and write a descriptive caption.

1. (20 points) – Explain *in your own words* why more data is needed to train models in higher dimensional spaces.

2. (40 points) – Complete class mydsp.DFTStream (see class on Blackboard). Like RMSStream, it takes an instance of a FrameStream object in its constructor.

   Write a function in module mydsp.plots called spectrogram. Given a list of files, and frame parameters, it creates a spectrogram across all of the files. Module mydsp.multifileaudiostream contains a class that will let you create a frame stream across multiple files.

   In your driver module, write function plot_narrowband_wideband that reads the audio file from a specified file and plots its spectrogram twice, once as a narrow-band spectrogram, and again as a wide-band spectrogram. Call the function from the top-level script environment with the shaken.wav file from the last assignment (this is the part in the "if __name__ == '__main__:'" block). Place these plots in a figure with both spectrograms and write a caption that describes the them. The wide-band spectrogram should be in the upper plot. The caption should describe the differences between the two plots (i.e. say more than Narrow-band and wide-band spectrograms of …) and the frame lengths of each one.

   Useful functions:
   numpy.asarray – numpy function to convert lists of lists to numpy matrices
   plt.subplot – matplotlib function for placing multiple plots in the same figure
   plt.pcolormesh – matplotlib function for showing a numpy matrix as an image (e.g. a spectrogram)
   plt.colorbar – matplotlib function for providing a sidebar scale for image data (e.g. isocontour data or pcolormesh)

   Caveat: If you are using an IDE that is producing small, inline plots (e.g. spyder), you may see aliasing of pixels that prevent you from seeing features of the spectrogram. Undock it and make larger for best results.

3. (40 points) –A subset of the female training data in the TIDIGITS corpus of spoken digits is available for you to use on Blackboard. Unzip this to your computer.

   Extend problem 2 to implement function pca_analysis. Write function pca_analysis which is called with a directory name. Invoke your function from

the script-level environment with the directory where you saved the TIDIGITS files.

The mydsp.utils module from the Blackboard attachment contains function get_corpus. Given a root directory, it will find all filenames contained in that directory or its children with a specified extension (default .wav).

Module mydsp.multifileaudioframes contains a class that will let you create a frame stream across a list of files. Generate a large spectrogram (do not plot it) using typical speech framing parameters (10 ms advance, 20 ms length).

Complete the missing sections of class PCA in modules mydsp.pca.PCA and conduct a PCA analysis of your data. Provide the following in the work you turn in:

a. Create a plot of the number of PCA components used versus the amount of variance captured.
b. Provide a tabular summary of your plot above. Each row should show how many components are needed for each decile of the variance (e.g. N_1 components to capture ≥10% of the variance, N_2 components to capture ≥20%, etc.). The first several rows may be the same as it is not uncommon for the first component to capture more than 20% of the variance.
c. Plot a spectrogram associated with file woman/ac/6a.wav (don't hardcode the file, use os.path.join to construct it relative to the root directory you specified. Transform the intensity data to PCA space, only using enough components to capture 60% of the data. Plot it as if it were a spectrogram, but the vertical axis is no longer frequency, but instead PCA component.

Note that the class constructor is set up to handle either variance-covariance analysis or correlation analysis. You are only responsible for providing the variance-covariance analysis, but the difference is only a couple lines of code, so feel free to implement it as well if you would like (satisfaction of a job well done, not extra credit).

Useful functions

scipy.signal.detrend – Removes the trend from a signal. Use type="constant" to remove the mean from your concatenated spectrogram. Be sure to do this so that it is the mean of each frequency bi n (use axis parameter to select)
numpy.linalg.eig – Computation of eigen values and vectors
numpy.cov/numpy.corrcoef – Computation of variance-covariance and correlation matrices
numpy.dot – Multiples two matrices

4. (40 points) Write function speech_silence in your driver module that takes a filename as an argument. The function will create an RMSStream from the file (20 ms frame length, 10 ms advance) and compute the RMS energy for each frame in the file. Use scikits learn's Gaussian mixture model (sklearn.mixture.GaussianMixture) to train a 2 mixture model from the RMS data. Note that the GaussianMixture class expects a column vector and if you use np.array to convert to a numpy array, it will create a row vector by default. You can use the array method's reshape method with the argument [-1, 1] to put it into a column vector that the GMM expects; be sure to capture the output, reshape returns a new matrix as opposed to changing the current one.

   Use the resultant model to predict the probability associated with each mixture. It will predict each frame as either belonging to mixture 0 or 1 based on the maximum posterior probability (assuming a uniform prior). You can associate the mixtures with speech or noise based on examination of the means_ attribute of the model (mean RMS of each of the Gaussians, give it a little thought as to how you should determine which one is noise and which one is speech). Create a plot that shows the RMS energy over time with distinctions (e.g. plot marker (x vs o), line style, or color if you have a color printer; read matplotlib.pyplot for options) for noise and speech. Add a legend with matplotlib.pyplot.legend.

   Call the function from the top-level environment on shaken.wav. Write a figure and caption.

   Listen to the speech, look at the spectrograms, and write a short paragraph describing how well qualitatively the classification performed. If you are unhappy with the performance, suggest how it might be made better.