Homework 3
Due on October 28 at 11:59 pm

Problem 1: GMMs for Speaker Recognition

I. Motivation and Background

In this problem, we will use Gaussian Mixture Models for speaker identification, and subsequently use these models to perform speaker classification. Model training and classification will be performed using mel fr[1]equency cepstrum vectors, which are computed from speech samples collected from ten speakers. Mel frequency cepstral coefficients (MFCCs) are a way of representing speech signals recorded from human beings, and you can read about them more from (https://en.wikipedia.org/wiki/Mel-frequency_cepstrum) and (http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstralcoefficients-mfccs/). For this problem, we will not be worried about converting speech samples into MFCC coefficients, we will work with MFCC coefficients directly. To motivate how we fit GMMs to MFCC coefficients, it's useful to understand a high-level picture of how MFCC coefficients are extracted from a speech utterance.

Mel Frequency Cepstral Coefficients

For performing speech recognition, it's very important to extract meaningful features from the audio signal. These features should identify the linguistic content and discard signal noise, emotional cadence etc. Since sounds generated by human beings are filtered by the vocal tract of a person, the shape of the vocal tract manifests in the envelope of the short time power spectrum. MFFCs, by virtue of their formulation, accurately represent this envelope. The broad strokes of the algorithm are as follows

1. Divide the audio signal into short frames (overlapping windows)
2. For each frame, calculate the periodogram estimate of the power spectrum
3. Apply mel filters to the power spectrum, and sum the energy obtained in each filter.
4. Take the logarithm of the filter energies
5. Take the DCT of the log filter energies, and keep 20 components

We don't need to worry about the exact algorithm, but it is important to remember that the raw audio signal is divided into multiple overlapping frames, and for each frame we calculate a 20-dimensional MFCC coefficient. The distribution of these MFCC coefficients across all the frames over an audio signal for a speaker are usually powerful discriminative tools. For this problem, you will fit GMMs to the MFCC coefficients for each speaker, which we'll discuss in more detail below.

II. Data

You can download the HW3_Problem1_template folder from **blackboard**. In the problem_1/data folder, you have ASCII-formatted training and testing data for each speaker. You have a train and test folder, along with a utt2spk file, explained below -

1. train: The train directory contains 2 files for each speaker, for a total of 20 files. Each file contains the 20-dimensional MFCC vectors from a speech recording from a subject. Therefore, each file contains

[1]

data of the form n × 20, where n is the number of frames that the raw audio recording was divided into, and each frame has a 20-dimensional MFCC vector associated with it.

2. test: The test directory contains 10 files, named testX, and have the same format as the training directory. Each test file contains a varying n number of 20-dimensional feature vectors from the same speaker.
3. utt2spk: This file contains 2 columns; column 1 represents the filename of the training sample, and column 2 represents the speaker identity. There are a total of 10 speakers whose identities are {110667-m, 106888-m, 3424-m, 102147-m, 103183-m, 101188-m, 4287-f, 2042-f, 7722-f, and 4177-m}. The m and f suffixes refer to the gender of the speaker, and the prefixes are the UIDs of the speakers.

III. Methodology

It is important to note that we use GMMs here in a somewhat different application to those discussed in the lectures. Here, we represent the distribution of parameters for each class (each speaker) using a different GMM model, for a total of 10 GMM models (one for each speaker). Then, given testing data, we calculate the posterior probability of the testing data belonging to each of these 10 classes and classify the testing data as belonging to the class whose GMM gave the highest posterior probability of explaining the data.

1. EM Algorithm for training GMMs: Estimate the Gaussian Mixture Model parameters for each speaker separately in the training data (we will have 10 different GMM models).

- The training data for speaker j is of the form $X^{(j)} = \{X_1^j, X_2^j, X_3^j, \ \dots, X_n^j\}$, where $X_i^{(j)} \in \mathbb{R}^{20}$
- For each speaker, train a 20-dimensional Gaussian Mixture Model with k = 64 Gaussians, each assuming diagonal covariance. Therefore, we're representing each MFCC vector as,

$$P\left(X_i^{(j)}\right) = \sum_{k=1}^{64} \pi_k^{(j)}\, G\left(X_i^{(j)} \middle| \mu_k^{(j)}, \Sigma_k^{(j)}\right) \dots\dots\dots (1)$$

where the multivariate Gaussian distribution for 20 dimensions is defined as

$$G\left(X_i^{(j)} \middle| \mu_k^{(j)}, \Sigma_k^{(j)}\right) = \frac{1}{\sqrt{(2\pi)^{20}\left|\Sigma_k^{(j)}\right|}} \exp\left(-0.5\left(X_i^{(j)} - \mu_k^{(j)}\right)^T \left(\Sigma_k^{(j)}\right)^{-1} \left(X_i^{(j)} - \mu_k^{(j)}\right)\right) \dots\dots\dots (2)$$

Here, $X_i^{(j)}, \mu_k^{(j)} \in \mathbb{R}^{20}, \Sigma_k^{(j)} \in \mathbb{R}^{20*20}$
- The joint probability distribution (likelihood) for the training data of one speaker, assuming each $X^{(j)}$ is independent, is simply the product of marginal probability distributions, given by

$$P(X^j|\pi^j, \mu^j, \Sigma^j) = \prod_{i=1}^{n} P\left(X_i^{(j)}\right) \dots\dots\dots (3)$$

And therefore,

$$\log P(X^j|\pi^j, \mu^j, \Sigma^j) = \sum_{i=1}^{n} \log P\left(X_i^{(j)}\right) \dots\dots\dots (4)$$

- For performing expectation-maximization, you will want to maximize the log-likelihood with respect to the parameters of the Gaussians $\pi^{(j)}$, $\mu^{(j)}$, $\Sigma^{(j)}$ for k = 1, . . ., 64. **Mathematically derive the closed-forms of the updates** for the Expectation Maximization algorithm for training the parameters of these GMMs. You can simplify the notation here to work with just one speaker for now for your mathematical derivation. Please submit your mathematical derivations and the steps of the EM algorithm as a PDF titled *EM_derivation.pdf.*
- Implement your derived EM algorithm above in MATLAB, and train 10 GMMs, one for each speaker's training data. Run your EM algorithm for 20 steps or until convergence to ε = 1e−4, whichever is first.
- You can use MATLAB's fitgmdist() function to compare with your results. Again, use exactly the same parameters; k = 64, and diagonal covariance matrices.

2. Classification: We can summarize the algorithm for classification given the trained GMM coeffi- cients as follows
   - Compute the log-likelihood of the test data sequence $X = \{X_1, X_2, \ldots, X_n\}$given the trained speaker GMM parameters $\Theta^{(j)} = \{\pi^{(j)}, \mu^{(j)}, \Sigma^{(j)}\}$ for j=1, ..., 10, which is given as
$$\log P(X|\theta^j) = \sum_{i=1}^{n} \log P(X_i|\theta^j) \ldots\ldots\ldots (5)$$
   - We can then classify the test data sequence X by assigning it to the speaker that gives the highest log-likelihood with their GMM model, as follows
$$\arg\max_j \log P(X|\theta^j) \ldots\ldots\ldots (6)$$
   - You can use the MATLAB function posterior() to calculate the posterior probabilities for the GMMs returned by fitgmdist() to compare with your results. posterior() estimates the posterior probability given data. Note that it returns the negative log-likelihood of the data, whereas we're calculating the log-likelihood. Make sure your account for this fact when you're comparing your implementation with MATLAB's functions.
   - Report classification results for both your implementation and using MATLAB's in-built func- tions (fitgmdist() and posterior()) for the 10 test samples. You should report your results in a text file test_predictions.txt, where the first column is the name of the test file, the second column is the prediction using your implementation, and the third column is the pre- diction using MATLAB's in-built functions. A valid submission would look like the following (assuming your classifications differ for the first test)

```
test1 110667-m 2042-f
test2 110667-m 110667-m
...
test10 106888-m 106888-m
```

Submission Instructions

Please follow these instructions closely. We will deduct points for not sticking to the template. Solutions should be uploaded to CANVAS.

The scripts must be written so that we can simply run your codes within the directory, without any additional commands or adding any additional file to the directories. If we cannot run the program, we cannot score you. Your results and derivation should go in the problem_1/results/ folder as a text file named test_predictions.txt, and as EM_derivation.pdf.

If you use MATLAB, use these templates. If you use Python, use the Jupyter notebook template for all problems in google collab in the following link-

https://colab.research.google.com/drive/1c5Jj4XqE5jmjemtCoSrBJPDHMbpgH_6o?usp=sharing

Note: Points will be deducted if your code doesn't match the required output format mentioned above. Your code should not output anything else.

Instructions for MATLAB:

You will submit a zip file containing all the required files specified below. Title your zip file as "HW3_yourJHID.zip", where 'yourJHID' is your JHED ID.

Instructions for Python:

You will have to use Jupyter Notebook for Python. We provided the template in Google Colab. Instructions are given in the link. You have to submit the specific deliverables mentioned below. Put them inside "results" folder inside the designated problem folder. You have to submit the "ipynb" file and "HW3_yourJHID.zip" zip file containing the required results or reports. In the text comments on Canvas, put the link to your Google colab notebook with proper permissions.