ENCM 509
Fundamentals of Biometric Systems Design
Department of Electrical and Software Engineering
Schulich School of Engineering

Laboratory Project #7

Speaker Recognition using GMM model

1 Introduction

The purpose of this Laboratory Project is to investigate speaker recognition approach using Gaussian Mixture Model (GMM). The performance of the algorithm is to be evaluated using confusion matrix. The steps shown here are in MATLAB language but you can use Python if you want to.

There are two variations of this project that you might consider, pick one:

a) Speaker recognition using GMM in MATLAB.

b) Speaker recognition using GMM in Python.

---

2 Project Report

The total Project mark is 20. The lab report shall include:

- Description of the implemented project (introduction, procedure and analysis, conclusion) (6 marks).
- The project flow block-diagram, illustrations, graphs (such as DET, ROC) and their analysis (6 marks).
- Code or any modifications to the code (8 marks).

3 Recommendations on the project on Speaker recognition using GMM in MATLAB

A demo of a speaker recognition used in this lab project, together with sample `.wav` files, is located on D2L in the Projects folder.

The input samples are in `.wav` format, but Matlab can read other audio formats too.

File `speaker_recognition_demo2018.m` includes demo that performs speech feature extraction from the training and testing on three training samples, and three testing samples of the same three individuals. It creates a statistical models of features of the training files. It tests each of the test files with the created models. It returns the results in the form of similarity scores.

In this demo, Matlab's `wavread` fucntion is used to read files. For example:

```
training_data1 = wavread('01_train.wav');
```

The features to be extracted are spectrum (one of the transforms applied on audio signals). For example,

```
training_features1 = melcepst(training_data1, Fs);
```

For statistical modeling, the Gaussian Mixture Model with M Gaussian components is used on data X. The function `gmm_estimate` returns the means mu, variances $sigma$ and weights c of the models created:

```
[mu,sigma,c] = gmm_estimate(X, M);
```

For testing, the function that computes multigaussian log-likelihood, using the test data X, the means mu, the diagonal covariance matrix of the model $sigma$ and the matrix representing the weights of each of the models, c:

```
[lYM,lY] = lmultigauss(X, mu, sigma, c);
```

The results are score obtained from the matching of the models of the speakers to the test samples. The speaker who has obtained the maximum score corresponds to the model speaker. The results of such matching are represented as a square matrix, which rows and columns correspond to the training and testing samples, respectively, such that the diagonal shows the scores for the same individuals.

3.1 Sample acquisition

In order to create your speech fragments, record audio files (`.wav`, or try other formats, too) of several subjects for training (30-60 sec per speaker), and audio files of the same subjects for testing (10 sec per speaker). For validation (probing a speaker who is not in database), more samples of another subject can be collected.

3.2 Speaker Recognition

Use the demo code fragments, and expand to perform the steps below.

- The samples you collected for each subject, must be divided into the training set and testing set. Keep one or two subjects for validation; they must not be in the training set.
- Perform feature extraction for each recording.
- Perform GMM training using the training samples.
- Perform classification using the testing samples. Record all he scores (you can use the code from demo to generate the confusion matrix).
- Perform validation by submitting the probe sample (of a speaker who is not on the database) and matching it against the trained models. Record the scores.
- Evaluate the error rates using the confusion matrix.

4 Recommendations on the project on Speaker recognition using Python

In Python you can easily read `.wav` files using the SciPy library[1]:

```python
from scipy.io import wavfile
samplerate, wave_data = wavfile.read('myfile.wav')
```

For the GMM, as you saw in Lab 3 (Signature with Gaussian Mixture Models), the Scikit-Learn library can be used[2]:

```python
# Training the GaussianMixture model from Scikit-Learn library
# the .fit(...) will perform the training step using the ''train_set''
gmm = GaussianMixture(n_components=50).fit(wave_data)
```

You can evaluate the classification using the same approach shown in the Lab: record the scores from the training samples and compare again a score of a probe sample. Remember that `n_components`, which is the number of Gaussian components used, is a hyperparameter of the GMM which needs to be tunned according to your data.

5 Acknowledgments

The original version of the Matlab demo code of a speaker recognition is credited to A. Alexander from EFPL.

March 7, 2022

[1]https://docs.scipy.org/doc/scipy/reference/generated/scipy.io.wavfile.read.html
[2]https://scikit-learn.org/stable/modules/mixture.html