Practical Exercise - Task 7
GMMs for sound classification and Speaker Identification
Speech Technology - COM4511/6511

May 2020

1 Introduction

The aim of this task is to understand sound classification using Gaussian Mixture Models (GMMs). For a more detailed discussion of GMMs see the lecture notes. Included in this folder is also a brief GMM primer (Draft document). GMMs can be used for supervised classification of audio as well as for unsupervised clustering

For this task you have two options - either to write the Gaussian mixture model code yourself (which would have immense learning benefit), or to use "scikit-learn", a Machine Learning toolkit for Python. The latter has the advantage that you will be able to concentrate on speech issues.

For this task a helper script is provided, set up for classification of vowel sounds. A second data set is provided to use GMMs for speaker identification, with different types of features. The speaker ID data (20 speakers, 3 feature representations) consists of a training set and an independent test set.

2 Preparation

From the MOLE module website download vowel (and other) sounds and helper scripts. An MFCC computation script is provided to you as a package called "speechtech".

For this task we will use "scikit-learn", a Machine Learning toolkit for Python. First, you need to install "scikit-learn" if you haven't done so. It is recommended to also install a Python visualisation library called "seaborn".

`http://scikit-learn.org/stable/install.html`

```
sudo pip install -U scikit-learn
sudo pip install seaborn
```

In order to get started, take the following first steps:

1. Unzip "`vowels.zip`" and place the "`vowels`" folder that contains 4 vowel sounds in the same folder where you put the helper script "`speechtech_gmm.py`" . Note that other sound file collections are provided.
2. Unzip "speakers.zip" in the same directory, further use bunzip2 to uncompress the feature files.
3. Place the package "`speechtech`" in the same working folder.
4. Provided the package "`scikit-learn`" is installed, you should be able to run the "`speechtech_gmm.py`" helper script and see some results. Word through the code.

3 Objectives

The objective in this task is to

- Learn about the configuration of GMMs and how to use them for sound classification of complete utterances.

  Complete the code provided and explore changes in configuration. A function `train_gmm` based on `scikit-learn` is provided. With help from the scikit-learn website, you can fill the missing parts in the function "`train_gmm`" and "`eval_gmm`" with your own code. How can performance and training quality of these models be assessed aside from classification accuracy?

- Understand the usefulness of GMMs for sound classification.

  A separate GMM is trained for each vowel. Classification is done at first for for each frame independently. Measure frame and utterance level accuracy. Increase the number of mixture components and see if it improves the accuracy. Are there other ways to improve? Are there limits and why? Note in this task the training set is used for evaluation. Is this a good idea?

- Understand the usefulness of GMMs for speaker identification.

  Modify the helper code to perform speaker identification with the data sets provided. Train the models on the training sets provided, perform frame level and utterance level tests on the test data. Explore the use of different features and report your findings.

4 Desired outcome

The outcome of this task is expected to be

1. Source code

   You can either submit your own code or the extended helper program, with your own code for the purpose of vowel classification. The program should be performing in optimal configuration and include both frame and utterance level accuracy computations.

   A second copy of the helper program, adapted for speaker ID. The program should be performing in optimal configuration and include both frame and utterance level accuracy computations.

2. A brief report

   Report your findings on both vowel classification and speaker identification. Which configurations work best? What is the difference between training and test set? What is the difference between utterance and frame level classifications? Provide explanations for these differences. Be very concise in your reporting. An ideal experiment modifies only one parameter at a time, addresses a very clear question, and provides a clear answer to that question.

   If time allows you are free to extend your experiments in any form that may interest you. For example you might explore distortion of data with noise, perform normalisation or adaptation.