SIT789 – Robotics, Computer Vision and Speech Processing

Pass Task 8.1: Speech emotion recognition using MFCC features

---

Objectives

The objectives of this lab include:

- Extracting MFCC features from audio signals using librosa.feature.mfcc
- Applying MFCC features to speech emotion recognition

Tasks

1. MFCC

In this section, we will learn how to extract MFCC features using librosa.feature.mfcc. You can practise MFCC features with the audio file arctic_a0005.wav used in Task 7.1. Suppose that we want to extract 12 MFCC features from the cepstrum of the signal in arctic_a0005.wav. The code below shows how to extract MFCC features.

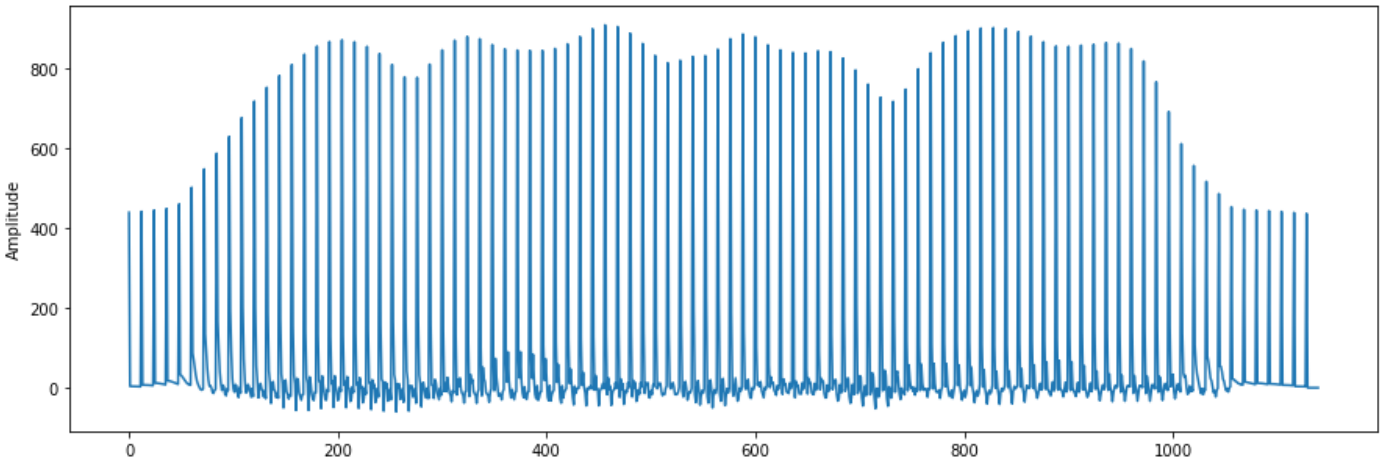
```
import numpy as np
import matplotlib.pyplot as plt
import librosa
import librosa.display
import IPython.display as ipd

from pydub import AudioSegment
from pydub.utils import mediainfo

speech = AudioSegment.from_wav('arctic_a0005.wav')  #Read audio data from file
x = speech.get_array_of_samples()  #samples x(t)
x_sr = speech.frame_rate  #sampling rate f - see slide 24 in week 7 lecture slides

mfcc = librosa.feature.mfcc(
    y = np.float32(x),
    sr = x_sr, #sampling rate of the signal, which is determined from the signal
    hop_length = int(x_sr * 0.015), #15 ms
    n_mfcc = 12) #number of mfcc features
```

librosa.feature.mfcc returns an array including 12 rows (as n_mfcc=12), and 95 columns (as hop_length is set to be equivalent to 15ms - see Section 2 in Task 7.1 for more details). We can check the dimension of mfcc using the following command.

```
print(mfcc.shape)
```

Recall that the MFCC features are the cepstral features extracted on the cepstrum (i.e., the inverse Fourier transform of the log magnitude spectrum). The following code visualises the MFCC features. In this code, we first get the transpose of mfcc, and then flatten it using reshape.

```
mfcc_flattened = np.reshape(mfcc.T, (mfcc.shape[0] * mfcc.shape[1]))
plt.figure(figsize = (15, 5))
plt.plot(mfcc_flattened)
plt.ylabel('Amplitude')
```



2. Speech emotion recognition

In this section, we will apply the MFCC features to describe speech signals for speech emotion recognition. We will use the EmotionSpeech dataset from the task's resources supplied in OnTrack. This dataset is a part of the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS).

The EmotionSpeech folder includes two sub-folders: Train and Test. Each Train/Test contains 128 clips (of about 4s) used for training and testing speech emotion recognition models respectively. In each Train/Test folder, there are 4 sub-folders containing audio data for 4 different emotions: calm, happy, sad, and angry. The audio clips for each emotion type are captured by 8 different actors (4 males and 4 females). However, audio clips from the same actor are used for either training or testing (but not both), e.g., clips from the first 4 actors are used for training and clips from the other 4 actors are used for testing. Each actor speaks two statements including *"Kids are talking by the door"* and *"Dogs are sitting by the door"*. Each statement is spoken two times. More details of the RAVDESS dataset can be found here.
Similarly to Task 4.1, we first define emotions and then load the data from audio files to computer memory for further processing.

```
import os

emotions = ['Calm', 'Happy', 'Sad', 'Angry']
path = 'EmotionSpeech/'
training_file_names = []
training_emotion_labels = []
for i in range(0, len(emotions)):
    sub_path = path + 'Train/' + emotions[i] + '/'
    sub_file_names = [os.path.join(sub_path, f) for f in os.listdir(sub_path)]
    sub_emotion_labels = [i] * len(sub_file_names)
    training_file_names += sub_file_names
    training_emotion_labels += sub_emotion_labels
```

Using the code sample in Section 1, we implement a method, named mfcc_extraction to extract mfcc features from an input audio file. Since audio files may have varying lengths, they would have different numbers of frames (given the same sampling rate is applied). To address this issue, we fix the number of frames in a given input audio clip to a predefined value. Specifically, let num_frames be the variable containing the number of frames. Suppose that an input audio clip has n frames. If $n \geq$ num_frames, only the first num_frames from the clip will be used. Otherwise, if $n <$ num_frames frames, all the n frames in the clip will be used and the missing frames, i.e., (n+1)-th to num_frames-th frames will be padded by zeros.

```
import numpy as np
import librosa
from pydub import AudioSegment
from pydub.utils import mediainfo

def mfcc_extraction(audio_filename, #.wav filename
                    hop_duration, #hop_length in seconds, e.g., 0.015s (i.e., 15ms)
                    num_mfcc, #number of mfcc features
                    num_frames #number of frames
                   ):
    speech = AudioSegment.from_wav(audio_filename)  #Read audio data from file
    samples = speech.get_array_of_samples()  #samples x(t)
    sampling_rate = speech.frame_rate  #sampling rate f

    mfcc = librosa.feature.mfcc(
        y = np.float32(samples),
        sr = sampling_rate,
        hop_length = int(sampling_rate * hop_duration),
        n_mfcc = num_mfcc)

    mfcc_truncated = np.zeros((num_mfcc, num_frames), np.float32)
    for i in range(min(num_frames, mfcc.shape[1])):
        mfcc_truncated[:, i] = mfcc[:, i]

    #output is a 1D vector
    return np.reshape(mfcc_truncated.T,
                      mfcc_truncated.shape[0] * mfcc_truncated.shape[1])
```

Your tasks here include:

1. Given the mfcc_extraction method, extract MFCC features from the training audio clips. Suppose that, in this task, each clip is sampled at 200 frames (i.e., num_frames=200), the hop length between two consecutive frames is 15ms, and 12 MFCC features are extracted on each frame. **Hint:** You can use a list to store MFCC feature vectors, each element in the list is an array of MFCC features returned by the mfcc_extraction method.

2. Speech emotion recognition using SVM (revisit Task 4.1 for examples of SVMs)
   a. Define an SVM model with RBF kernel and C set to 10.0.
   b. Train your SVM model with the speech data supplied in the Train folder.
   c. Use the trained SVM model to recognise emotions from the speech data supplied in the Test folder. Report the overall recognition accuracy, the recognition accuracy on each emotion, and confusion matrix of your SVM model.
   d. Vary num_mfcc in the range [10, 12, 14, 16, 18, 20] and measure the corresponding overall recognition accuracy, recognition accuracy on each emotion class, and confusion matrices. What is the best num_mfcc?

Submission instructions

1. Perform tasks required in Section 1 and 2.
2. Complete the supplied answer sheet with required results
3. Submit your code (.py) and answer sheet (.pdf) to OnTrack.