Assignment 1: Time and Frequency Domain Features

CS 4347: Sound and Music Computing

due Feb 4 (Monday), 11:59 pm

1. This assignment will make use of the "Music & Speech" dataset of Marsyas:
   - You can download the dataset from: `http://opihi.cs.uvic.ca/sound/music_speech.tar.gz`
   - This dataset has two copies of each song, delete the `music/` and `speech/` directories and use the files in `music-wav/` and `speech-wav/` directories. There are 64 music and 64 speech files. Each file has 30 seconds of audio stored as 16-bit signed integers at 22050 Hz.
   - Ground truth data for this dataset can be downloaded from IVLE. Format of the file is `filename` `\t` (tab) `label` `\n` (newline), one song per line:

     ```
     filename1\tlabel1\n
     filename2\tlabel2\n
     ...
     filename128\tlabel128\n
     ```

     The `label` field is either `music` or `speech`.

2. Follow the following steps to complete this assignment:
   - Read the ground-truth file (`music_speech.mf`).
   - Load each wav file and convert the data to floats by dividing the samples by 32768.0. Hint: use `scipy.io.wavfile.read()`
   - Split the data into buffers of length 1024 with 50% overlap (or a hopsize of 512). Only keep complete buffers, e.g. if the last buffer only has 1020 samples, omit it.
     Hint: the starting and ending indices for the first few buffers are:

Buffer number	start index	end index (not included in array)
0	0	1024
1	512	1536
2	1024	2048
...		

     We recommend that you use the "array slicing" feature provided by numpy:

     ```
     for i in range(num_buffers):
     start = ...
     end = ...
     buffer_data = whole_file_data[start:end]
     ```

   - For each file, calculate time domain features for each buffer according to the given formula. Given $X = \{x_0, x_1, x_2, \ldots x_{N-1}\}$ ($N = 1024$ for this assignment):

     (a) Root-mean-squared (RMS):

$$X_{\text{RMS}} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} x_i^2}$$

(b) Zero crossings (ZCR):

$$X_{\text{ZCR}} = \frac{1}{N-1} \sum_{i=1}^{N-1} \begin{cases} 1 & \text{if } (x_i \cdot x_{i-1}) < 0 \\ 0 & \text{else} \end{cases}$$

- After calculating the features for each buffer, calculate the mean and uncorrected sample standard deviation for each feature over all buffers for each file.
- Now you have finished calculating time domain features. To calculate frequency domain features, multiply each buffer with a Hamming window.
  Hint: use `scipy.signal.windows.hamming()`
- Perform a Discrete Fourier Transform for each windowed buffer.
  Hint: use `scipy.fft()`.
  Note: the DFT gives you both "positive" and "negative" frequencies, whose values are mirrored around the Nyquist frequency. Discard the negative frequencies (whose array indices are above $N/2$ for an FFT of length N).
- Calculate the following frequency domain features for each spectral buffer. Given a spectral buffer X:

  (a) Spectral Centroid (SC):

$$\text{SC} = \frac{\sum_{k=0}^{N-1} k \cdot |X[k]|}{\sum_{k=0}^{N-1} |X[k]|}$$

  (b) Spectral Roll-Off (SRO): which is the smallest bin index R such that L energy is below it. For this assignment, we will use $L = 0.85$.

$$\sum_{k=0}^{R-1} |X[k]| \geq L \cdot \sum_{k=0}^{N-1} |X[k]|$$

  (c) Spectral Flatness Measure (SFM):

$$\text{SFM} = \frac{\exp\left(\frac{1}{N} \sum_{k=0}^{N-1} \ln |X[k]|\right)}{\frac{1}{N} \sum_{k=0}^{N-1} |X[k]|}$$

  Note: using the log-scale is useful for avoiding multiplications which may exceed the bounds of double floating-point arithmetic.

- After calculating the features for each buffer, calculate the mean and uncorrected sample standard deviation for each feature over all buffers for each file.
- Output your results to a new ARFF file and name it as `results.arff`. The header of it should be like:

```
@RELATION music_speech
@ATTRIBUTE RMS_MEAN NUMERIC
@ATTRIBUTE ZCR_MEAN NUMERIC
@ATTRIBUTE SC_MEAN NUMERIC
@ATTRIBUTE SRO_MEAN NUMERIC
@ATTRIBUTE SFM_MEAN NUMERIC
@ATTRIBUTE RMS_STD NUMERIC
@ATTRIBUTE ZCR_STD NUMERIC
@ATTRIBUTE SC_STD NUMERIC
@ATTRIBUTE SRO_STD NUMERIC
@ATTRIBUTE SFM_STD NUMERIC
@ATTRIBUTE class {music,speech}
```

  The format of the data section should be:

```
@DATA
RMS_MEAN1,ZCR_MEAN1,SC_MEAN1,SRO_MEAN1,SFM_MEAN1,RMS_STD1,ZCR_STD1,SC_STD1,SRO_STD1,SFM_STD1,music
...
```

  Concretely, the `@DATA` section should be:

```
@DATA
0.057447,0.191595,128.656296,239.404651,0.329993,0.027113,0.036597,13.206525,27.957121,0.087828,music
...
0.062831,0.082504,78.481380,145.886047,0.198849,0.032323,0.070962,39.388633,66.942115,0.133545,speech
```

Note: Please keep at least 6 digits after the decimal point for output.

3. Submit a zip file to IVLE containing your source code (a single `.py` file) and the ARFF file. Name the zip file using your student number (e.g. `A0123456H.zip`).

4. **Note:** You may use any python standard libraries, numpy (including pylab / matplotlib) and scipy. No other libraries are permitted. Late submissions will receive no marks.

5. Grading scheme:

   - **4/6 marks**: correct ARFF file.
   - **2/6 marks**: readable source code (good variable names, clean functions, necessary comments).