Assignment 1: Time Domain Features Extraction

CS 4347: Sound and Music Computing

due Wednesday 8 February 2017, 11:59 pm

0. This assignment will make use of the "Music & Speech" dataset of Marsyas:
   - `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 is 30 seconds of audio stored as 16-bit signed integers at 22050 Hz.
   - Ground truth data for this dataset:
     `http://www.comp.nus.edu.sg/~duanzy/music_speech.mf`
     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` will be `music` or `speech`.

1. Write a python program that will:
   - Read the ground-truth `music_speech.mf` file
   - Load each wav file and convert the data to floats by dividing the samples by 32768.0. Hint: use `scipy.io.wavfile.read()`
   - Calculate 5 features for each file according to the given formulae. Use only one vector per file (don't use multiple buffers for each file).
     Given $X = \{x_0, x_1, x_2, \ldots x_{N-1}\}$,
     (a) Root-mean-squared (RMS)
     $$X_{\text{RMS}} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} x_i^2}$$
     (b) Peak-to-average-ratio (PAR)
     $$X_{\text{PAR}} = \frac{\max |x_i|}{X_{\text{RMS}}}$$
     (c) Zero crossings (ZCR)
     $$X_{\text{ZCR}} = \frac{1}{N-1}\sum_{i=1}^{N-1} \begin{cases} 1 & \text{if } (x_i \cdot x_{i-1}) < 0 \\ 0 & \text{else} \end{cases}$$
     (d) Median absolute deviation (MAD)
     $$X_{\text{MAD}} = \underset{i}{\text{median}}\left(\left| x_i - \underset{j}{\text{median}}(x_j) \right|\right)$$
     Hint: numpy has a built-in `numpy.median()` function!

(e) Mean Absolute Deviation (MEAN-AD):

$$X_{\text{MEAN\_AD}} = \frac{1}{N} \sum_{i=0}^{N-1} \left(\left| x_i - \frac{1}{N} \sum_{j=0}^{N-1} x_j \right| \right)$$

- Output the data to a comma separated value (CSV) text file in the format:

```
filename1,RMS1,PAR1,ZCR1,MAD1,MEAN_AD1\n
filename2,RMS2,PAR2,ZCR2,MAD2,MEAN_AD2\n
...
filename128,RMS128,PAR128,AC128,MAD128,MEAN_AD128\n
```

  Concretely, the beginning and ending of the file should be:

```
music_wav/bagpipe.wav,0.063492,8.149929,0.191660,0.031769,0.044588
music_wav/ballad.wav,0.029699,7.320233,0.039395,0.012695,0.018975
...
speech_wav/voice.wav,0.070688,4.163124,0.082435,0.031982,0.049338
```

  To pass our automated grading system, the format of your file must match this exactly. The order of filenames must match the order in the `music_speech.mf` file.

2. Upload your CSV file, and source code to **Assignment 1a** at:

   `http://cs4347.smcnus.org`

   This will automatically grade the values you calculated. If any mistake is found, please check your program and resubmit – you are welcome to submit as many versions as you wish before the submission deadline.

   Submit a zip file containing your program's source code files (as a `.py` file), the CSV file, and an optional README.txt file to the same website.

3. Make a copy of the previous program. Modify it such that it will:

   - Read the ground-truth `music_speech.mf` file
   - Load each wav file (divide by 32768.0) and split the data into buffers of length 1024 with 50% overlap (or a hopsize of 512). Only include complete buffers; if the final buffer has 1020 samples, omit that buffer.
     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
...		

     I recommend that you use the numpy "array slice" feature:

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

     This should give you 1290 buffers of length 1024. If you are comfortable with linear algebra and numpy, the program will run significantly faster if you store this as a single 1290x1024 matrix and avoid any loops during the feature calculations.
   - Calculate 5 features for each buffer: RMS, PAR, ZCR, MAD, MEAN_AD. This means that $N = 1024$ in the equations, instead of the previous $N = 661500$. You should end up with 1290 feature vectors of length 5, or a single 1290x5 matrix.

- After calculating the features for each buffer, calculate the mean and uncorrected sample standard deviation for each feature over all buffers for each file. You should end up with a single feature vector of length 10, or a 1x10 matrix for each file.
- For a later assignment we will be using a tool which reads ARFF files. Output the data to a new ARFF file with the header:

  ```
  @RELATION music_speech
  @ATTRIBUTE RMS_MEAN NUMERIC
  @ATTRIBUTE PAR_MEAN NUMERIC
  @ATTRIBUTE ZCR_MEAN NUMERIC
  @ATTRIBUTE MAD_MEAN NUMERIC
  @ATTRIBUTE MEAN_AD_MEAN NUMERIC
  @ATTRIBUTE RMS_STD NUMERIC
  @ATTRIBUTE PAR_STD NUMERIC
  @ATTRIBUTE ZCR_STD NUMERIC
  @ATTRIBUTE MAD_STD NUMERIC
  @ATTRIBUTE MEAN_AD_STD NUMERIC
  @ATTRIBUTE class {music,speech}
  ```

  The data lines should be:

  ```
  @DATA
  RMS_MEAN1,PAR_MEAN1,ZCR_MEAN1,MAD_MEAN1,MEAN_AD_MEAN1,RMS_STD1,PAR_STD1,ZCR_STD1,MAD_STD1,MEAN_AD_STD1,music
  ...
  ```

  Concretely, the `@DATA` section should be:

  ```
  @DATA
  0.057447,3.234547,0.191595,0.037308,0.044607,0.027113,0.397827,0.036597,0.018317,0.021898,music
  0.026583,2.590328,0.039416,0.016488,0.018977,0.013284,0.384978,0.015575,0.011143,0.012273,music

  ...
  0.062831,2.822102,0.082504,0.042271,0.049270,0.032323,0.799688,0.070962,0.027540,0.029103,speech
  ```

4. Upload your ARFF files, and source code to **Assignment 1b** at:

   `http://cs4347.smcnus.org`

   This will automatically grade the values you calculated. If any mistake is found, please check your program and resubmit – you are welcome to submit as many versions as you wish before the submission deadline.

   Submit a zip file containing your program's source code files (as a `.py` file), the ARFF file, and an optional README.txt file to the same website.

   - You may use anything in the python standard library, numpy (including pylab / matplotlib), and scipy libraries. No other libraries are permitted.

If you are familiar with python and understood the lecture, this should take about 2 hours.
Grading scheme:
- **2/6 marks**: correct CSV file (automatically graded by computer).
- **2/6 marks**: correct ARFF file (automatically graded by computer).
- **2/6 marks**: readable source code (good variable names, clean functions, comments when needed).