Assignment 4: Machine Learning and Spectral Features of Audio

CS 4347: Sound and Music Computing

due Wednesday 4 March 2015, 11:59 pm

0. This assignment will use the same "music / speech" dataset that we used in asn 1.

1. Make a copy of your python program from assignment 3 and modify it so that it:
   - Read the collection file
   - Load each wav file and split the data into buffers of length 1024 with 50% overlap. Only include complete buffers; if the final buffer has 1020 samples, omit that buffer.
   - Multiplies each buffer with a Hamming window. Hint: `scipy.signal.hamming()`
   - Performs a Discrete Fourier Transform. Hint: `scipy.fftpack.fft()`
   - Remember that the DFT gives you "positive" and "negative" frequencies, whose values are mirrored around the Nyquist frequency. Discard the negative frequencies (array indices above $N/2$ for an FFT of length N).
   - Calculate the following features for each spectral buffer. Since all these features use the absolute value of the spectrum, you may find it useful to convert the entire spectrum into absolute values (hint: `numpy.abs()`).
     Given a spectral buffer X,
     (a) Spectral Centroid (SC): the "brightness" of a sound

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

     (b) Spectral Roll-Off (SRO): measures how the energy is concentrated throughout the spectrum. Specifically, SRO is the smallest bin index R such that L energy is less than the sum it. For audio analysis, 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): how "spiky" the spectrum is, calculated by dividing the geometric mean by the arithmetic mean

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

     Using the log-scale is useful for avoiding multiplications which may exceed the bounds of double floating-point arithmetic.
     (d) Peak-to-average ratio (PARFFT): as defined in assignment 1.

(e) Spectral Flux (SF): difference between two successive buffers of DFT data. Let $|X[k]|_n$ be "the n^{th} analysis buffer".

$$\text{Flux}_n = \sum_{k=0}^{N-1} H(|X[k]|_n - |X[k]|_{n-1})$$

$$H(y) = \begin{cases} y & \text{if } y > 0 \\ 0 & \text{else} \end{cases}$$

We define $|X[k]|_{-1}$ as an array of zeros.

- After calculating the features for each buffer, calculate the mean and uncorrected sample standard deviation for each feature over all buffers for each file.
- Outputs the data to an `ARFF` file with the same format as before, but the headers should be `SC`, `SRO`, `SFM`, `PARFFT`, `SF`; each of them with a corresponding `MEAN_` and `STD_`.
  Concretely, the `@DATA` section should be:

```
@DATA
128.656296,239.404651,0.329993,9.164712,65.645735,13.206525,27.957121,0.087828,1.748588,45.773579,music
35.920625,63.362016,0.070470,14.030788,9.253576,5.640378,12.961779,0.017258,2.971727,11.300829,music
...
78.481380,145.886047,0.198849,13.112994,73.083275,39.388633,66.942115,0.133545,3.576990,64.046635,speech
```

  The header names must be:

```
SC_MEAN, SRO_MEAN, SFM_MEAN, PARFFT_MEAN, FLUX_MEAN,
SC_STD, SRO_STD, SFM_STD, PARFFT_STD, FLUX_STD
```

2. Classify the data with `trees.J48` with 10-fold cross-validation and save the results. In addition, select 2 other classification algorithms and record their output. Write a file called `classifications.txt` which compares the results of these three algorithms. The file should begin with 1-3 sentences stating which was the best algorithm and summarizing its performance in comparison to the other two algorithms. This should be followed by the Weka results ordered from best to worst.

3. Upload your ARFF file to:

   `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 (as a `.py` or `ipynb` file), the ARFF file, the `classifications.txt` file comparing different machine learning algorithms, 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 1–2 hours.
Grading scheme:

- **3/6 marks**: correct ARFF file (automatically graded by computer).
- **2/6 marks**: readable source code (good variable names, clean functions, comments when needed).
- **1/6 marks**: Discussion of weka output