Assignment 2: Machine Learning with Time-Domain Audio

CS 4347: Sound and Music Computing

due Wednesday 11 February 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 1 and begin modifying it. Your program should begin by doing the same steps as assignment 1, namely:
   - 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.
   - Calculate 4 features for each file: RMS, PAR, ZCR, MAD. Use only one vector per file (don't use multiple buffers for each file).
   - (optional) You may wish to improve your program's speed. Well-written python+numpy can calculate all these features on the entire dataset in less than a minute on a modern desktop computer (mine completes assignment 1 in 9.97 seconds).
     The key is to avoid any loops for calculating features. Everything can be done with built-in numpy commands. Some features can be calculated with a single line, for example:

     ```
     def rms(time_domain_data):
         return numpy.sqrt(numpy.mean(numpy.square(time_domain_data)))
     ```

   Your program should output different data, however:

   - Output the data to an `ARFF` file with this format:

     ```
     @RELATION music_speech
     @ATTRIBUTE RMS NUMERIC
     @ATTRIBUTE PAR NUMERIC
     @ATTRIBUTE ZCR NUMERIC
     @ATTRIBUTE MAD NUMERIC
     @ATTRIBUTE class {music,speech}

     @DATA
     RMS1,PAR1,ZCR1,MAD1,music
     RMS2,PAR2,ZCR2,MAD2,music
     ...
     RMS128,PAR128,ZCR128,MAD128,speech
     ```

     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. Note that the the ARFF format does not include the filenames, merely the label.
   - Output the follow pairs of features as a graph (use `pylab.savefig` to save as PNG files):
     - ZCR (x-axis) and PAR (y-axis)
     - MAD and RMS

     Each graph must have axis labels, clearly distinguishable markers & colors for music and speech, and a legend.

Hint: don't try to do the plotting in the main loop of your program! Intead, fill two matrices of features for music and speech:

```
features_music  = numpy.zeros((num_audio_files, num_features))
features_speech = numpy.zeros((num_audio_files, num_features))
...
for i in range(num_audio_files):
    ...
    if label == "music":
        features_music[i]  = features
    else:
        features_speech[i] = features
```

These can then be plotted very easily by extracting the relevant columns (e.g., column 2 is ZCR, column 1 is PAR):

```
pylab.plot(features_music[:,2], features_music[:,1])
pylab.plot(features_speech[:,2], features_speech[:,1])
```

The resulting plots do not clearly distinguish between these sets of features, so make sure you improve the plots before submitting the assignment. This hint gives you the beginning of the plotting task, not the whole answer!

2. Load the ARFF file in Weka: `http://www.cs.waikato.ac.nz/ml/weka/`

   Classify the data with `trees.J48` with 10-fold cross-validation and save the results. In detail, load weka, go to the Explorer, and open your ARFF file. Click on the "Classify" tab, then click "Choose" under "Classifier". Select the `trees.J48` option, then press "Start". Right-click on the value in the "Result list" and save the result buffer (the text containing the Classifier model, Summary, and Confusion Matrix).

   In addition to classifying with `trees.J48`, 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 two PNG files, 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, this should take about 1 hour.
Grading scheme:

- **1/6 marks**: correct ARFF file (automatically graded by computer).
- **2/6 marks**: readable source code (good variable names, clean functions, comments when needed).
- **2/6 marks**: visual quality of plots (axis labels, markers, colors, legend)
- **1/6 marks**: Discussion of weka output