Download and Analyze Sounds from Freesound - Examples

Audio Signal Processing for Music Applications

Before starting the assignment A9, make sure you have downloaded the script `freesound.py` from `https://github.com/MTG/freesound-python` into the folder A9 containing the scripts `soundDownload.py` and `soundAnalysis.py`. Also, apply and obtain a Freesound API key from `www.freesound.org/apiv2/apply/` . We provide you some examples to use these scripts, which will be useful to solve A9.

1 Downloading sounds (mp3 previews) and descriptors

These are the steps to download and store sounds and descriptors from Freesound using the scripts provided with this assignment.

1. Go to the working directory for this assignment `sms-tools/workspace/A9`

   ```
   $ cd <path to A9 directory in workspace>
   ```

2. Create/select a directory to store descriptors and sound previews downloaded from freesound.

   ```
   $ mkdir testDownload
   ```

3. To download descriptors and sound previews use `downloadSoundsFreesound()` function in the file `soundDownload.py`.

   ```
   user:/sms-tools/workspace/A9$ ipython
   In [1] : import soundDownload as SD
   In [2] : SD.downloadSoundsFreesound(queryText='double_bass_mezzoforte',
            API_Key=<your key>, outputDir='testDownload/', topNResults=5,
            duration=(0,3), tag = 'pizzicato')
   ```

In order to run examples for the other questions in this assignment, download another class of sounds by running the following code in ipython shell. After this step you will have two classes of the sounds in texttttestDownload directory.

```
In [3] : SD.downloadSoundsFreesound(queryText='violin', API_Key=<your key>,
outputDir='testDownload/', topNResults=5, duration=(0,3))
```

NOTE: Before using this script for downloading descriptors make sure that you have chosen the query text, tag and duration of the sound that returns you good representative sounds for analysis in this assignment. You can do this selection using the advance search options in `http://freesound.org/search/?q`.

2 Descriptor scatter plot (plotting descriptor pairs for every sound)

You can do a scatter plot of descriptor pairs (any two descriptors can be selected for this plot) for all the downloaded sounds. Using this plot you can determine how relevant are the chosen set of descriptor pairs for differentiating the sound classes (instruments in our case). For doing the scatter plot you can use `descriptorPairScatterPlot()` function in the script `soundAnalysis.py`.

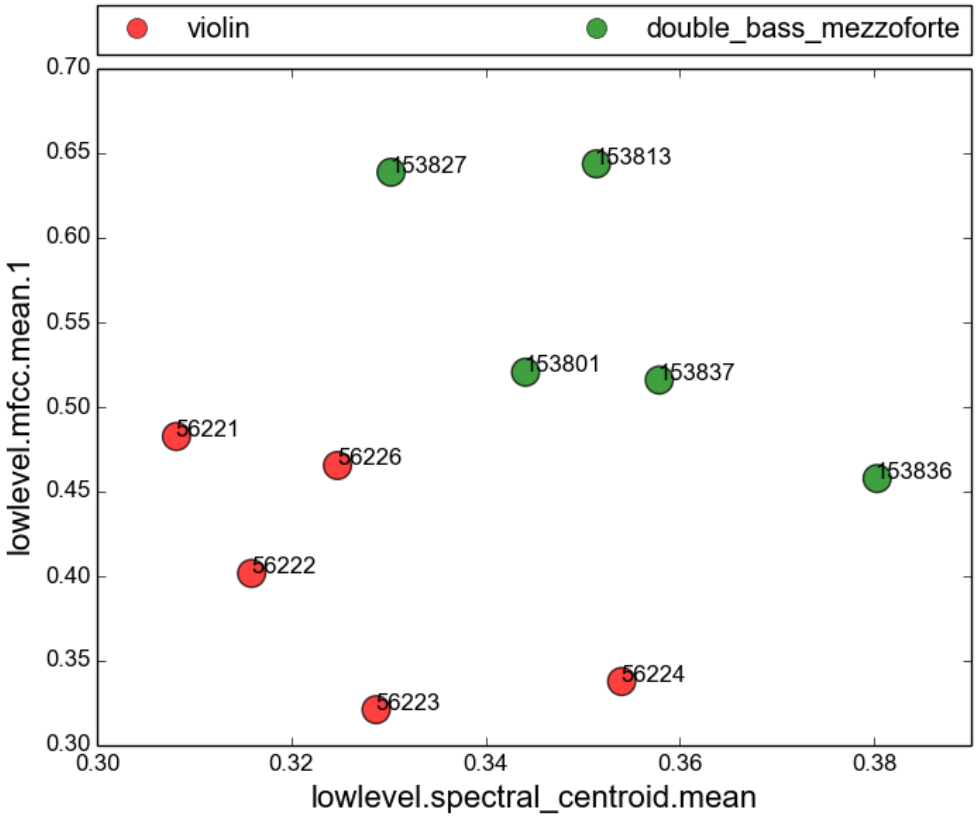

Figure 1: Scatter plot for the mean of the spectral centroid and 1^{st} MFCC coefficient for all the sounds downloaded from Freesound in Section 1.

```
In [4] : import soundAnalysis as SA
In [5] : SA.descriptorPairScatterPlot('testDownload/', descInput=(0,12))
```

The parameter `descInput` is a python tuple with two elements corresponding to the indices of descriptors to be used for doing scatter plot. To know the mapping between an index and the descriptor name use this function:

```
In [5] : SA.showDescriptorMapping()
```

Additionally, if your scatter plot is not already cluttered, or if you want to spot an outlier sound, you can set `anotOn = 1` to annotate the points in scatter plot with the sound-id. The output plots are shown Figure 1.

3 Clustering sounds downloaded from Freesound

You can perform a k-means clustering of the sounds that you downloaded from the freesound. You can use the function `clusterSounds()` in the file `soundAnalysis.py`.

```
In [6] : SA.clusterSounds('testDownload/', nCluster=2, descInput=[0,12])
```

The output of this code is as follows:

```
(Cluster: 1) Using majority voting as a criterion this cluster belongs to
class: double_bass_mezzoforte
Number of sounds in this cluster are: 6
```

```
sound-id, sound-class, classification decision
[['56224' 'violin' '0']
 ['153827' 'double_bass_mezzoforte' '1']
 ['153801' 'double_bass_mezzoforte' '1']
 ['153813' 'double_bass_mezzoforte' '1']
 ['153836' 'double_bass_mezzoforte' '1']
 ['153837' 'double_bass_mezzoforte' '1']]
(Cluster: 2) Using majority voting as a criterion this cluster belongs to class : violin
Number of sounds in this cluster are: 4
sound-id, sound-class, classification decision
[['56226' 'violin' '1']
 ['56222' 'violin' '1']
 ['56223' 'violin' '1']
 ['56221' 'violin' '1']]
Out of 10 sounds 1 sounds are incorrectly classified considering that one cluster should
ideally contain sounds from only a single class
You obtain a classification (based on obtained clusters and majority voting) accuracy
of 90.00 percentage
```

NOTE: k-means clustering results can vary quite significantly across runs (specifically since there are only a small number of data points). You might get different results for this part after running the function above, and different results every time you run.

4 Classifying sounds using k-nearest neighbors classifier

You can classify any sound into sound categories corresponding to the downloaded sounds ('double_bass_mezzoforte' and 'violin' in above examples). You can use `classifySoundkNN()` function in the file `soundAnalysis.py`.

```
In [7] : SA.classifySoundkNN('testDownload/violin/56224/56224_692375-lq.json',
'testDownload/', 3, descInput=[0,12])
```

Ideally you should not use a query that is also a part of the target search space. This is the reason why the command above produces a warning. This is just an example case to show you how to run the function. For the assignment you will choose a sound that is not included in the target directory.