A9: Sound and music description

Audio Signal Processing for Music Applications

Overview

[Download code.]

This is a peer assessed assignment in which you will learn how to describe sounds within a sound collection using a set of temporal and spectral descriptors. You will learn to use the Freesound API to download sounds and their descriptors. Then with the set of obtained descriptors and by using a similarity measure, you will cluster the sounds and also classify them into predefined classes. You will work with instrumental sounds, thus learning what audio features are useful for their characterization in particular tasks.

Learning outcomes

By the end of the assignment, you will have learnt about:

- Several common audio descriptors, and how they can capture different aspects of a sound.
- Selecting relevant descriptors for characterizing a particular category of sound for a particular application (e.g., clustering or classification).
- Using a simple audio/sound similarity based on different types of descriptors (Euclidean distance).
- Using a simple classifier (k-NN) and a simple clustering algorithm (k-means).
- Using web APIs to access data (Freesound API)

There are four parts in this assignment, each one using the results and output of the previous one, and hence to be done in that order:

- **Task 1:** Download several sounds and their descriptors from Freesound, of different acoustic instruments playing single notes.
- **Task 2:** Choose two descriptors that result into a good clustering of the downloaded sounds.
- **Task 3:** Cluster the downloaded sounds using k-means to obtain clusters corresponding to different instruments.
- **Task 4:** Classify a sound, using nearest neighbour classifier, into one of several instrumental classes.

We provide the code for each task, thus no need to get involved in any programming. Read the example document to see examples of using the code.

Relevant Concepts

Code

The assignment package includes two python scripts and an example pdf. In addition to these scripts you need to add another script, `freesound.py`, which you can download from `https://github.com/MTG/freesound-python`. Download the `freesound.py` script and put it along

with the other two scripts in your working directory. You don't have to write any additional code or modify the scripts, you will just have to make calls to the provided functions. If you are interested in knowing more about the Freesound API, you can see examples of using it from python in: `https://github.com/MTG/freesound-python/blob/master/examples.py` and you can read the API documentation here: `http://www.freesound.org/docs/api/`

These are the files that you should have in your working directory for doing this assignment:

- `examples.pdf`: Usage examples of the functions used in this assignment.
- `soundDownload.py`: Scripts with functions to download sounds and descriptors from Freesound.
- `soundAnalysis.py`: Script with functions that do the analysis using the downloaded sound descriptors.
- `freesound.py`: The Freesound API client needed for API queries.

Freesound API to query and download sounds

Freesound provides an API to access its data repository. The API is comprehensive, but in the scope of this assignment, we will use it only to download sounds and their descriptors by defining several search criteria. With the API you can do text searches similar to what you can do from the advance searches in the website `http://freesound.org/search/?q` plus you can also other types of more advance queries. The main advantage of a Web API is that we can do the queries from software.

To perform advanced searches from the website you can enable 'show advanced search options' and, for example, add a duration filter. Once you get the sounds from a search using a query text, you can filter them based on the tags that are shown on the right side of the screen. We can do the same search using the Freesound API. In this assignment, the task will be to download single notes/strokes of several instruments using a python function that is available in the accompanying file soundDownload.py file, the function call is: `downloadSoundsFreesound(queryTex="", tag=None,duration=None,API_Key="",outputDir="",topNResults=5, featureExt='.json'`

The input parameters used are:

- queryText: A single word or a string of words without spaces (use hyphens), typically the name of the instrument. e.g. (eg. "violin", "trumpet", "cello", "bassoon", etc.)
- tag: tag to be used for filtering the searched sounds (e.g., "multisample", "single-note", "velocity", "tenuto", etc.).
- duration: min and max duration (seconds) of the sound to filter, e.g., (0.2,15).
- API_Key: your api key, which you can obtain from: `www.freesound.org/apiv2/apply/`
- outputDir: path to the directory where you want to store the sounds and their descriptors.
- topNResults: number of results (sounds) that you want to download.
- featureExt: file extension for storing sound descriptor (.json, typically).

You will have to choose the appropriate queryText, tag, and duration, to return single notes/strokes of instrumental sounds. The first twenty results of the query should be "good". Note that the tag can be empty. Example of a query to obtain single notes of violin could be: `downloadSoundsFreesound(queryText='violin', API_Key=<your key>, outputDir='testDownload/', topNResults=20, duration=(0,8.5), tag='single-note')`. This returns 20 single notes of violin sounds and the script stores them in the testDownload directory (the directory has to be created beforehand).

Before using the API to download the sounds, we recommend to do the same query using Freesound website and checking that the top 20 results are good.

Sound Descriptors

In this assignment, you will automatically download the following temporal and spectral descriptors for every sound that the query finds. You will then select a subset of those for clustering and classification. Most of these are spectral descriptors and widely used in various Music Information Retrieval tasks. For information about them you can read the Essentia documentation: `http://essentia.upf.edu/documentation/algorithms_reference.html`

Here is the list of descriptors that are downloaded:

Index	Descriptor
0	lowlevel.spectral_centroid.mean
1	lowlevel.dissonance.mean
2	lowlevel.hfc.mean
3	sfx.logattacktime.mean
4	sfx.inharmonicity.mean
5	lowlevel.spectral_contrast.mean.0
6	lowlevel.spectral_contrast.mean.1
7	lowlevel.spectral_contrast.mean.2
8	lowlevel.spectral_contrast.mean.3
9	lowlevel.spectral_contrast.mean.4
10	lowlevel.spectral_contrast.mean.5
11	lowlevel.mfcc.mean.0
12	lowlevel.mfcc.mean.1
13	lowlevel.mfcc.mean.2
14	lowlevel.mfcc.mean.3
15	lowlevel.mfcc.mean.4
16	lowlevel.mfcc.mean.5

Task 1: Download sounds and descriptors from Freesound

- Apply for and obtain a Freesound API key (if you have not already): `http://www.freesound.org/apiv2/apply/`
- Select three instruments to be used out of this set: violin, guitar, bassoon, trumpet, clarinet, cello, naobo (cymbals used in Beijing Opera). Specify a good query text, tag, and duration to query for the the chosen instruments. Use your API key and download twenty sound examples of each instrument using the downloadSoundsFreesound() function in `soundDownload.py` script. The examples need to be representative of the instrument, single notes (melodic instruments) or single strokes (percussion instruments), and shorter than 10 seconds. Refine your search parameters until you get twenty good samples for each instrument. Listen to the sounds downloaded and look at the descriptor .json files.

Write a short paragraph mentioning the query text, tag and duration used for each of the three instruments you chose. Explain why you chose those instruments, and why you selected the specific search query text, tag and duration.

Attach the `<queryText>_soundList.txt` file for each instrument that the script created in each instrument folder. You should compress them into one zip file which you upload.

Task 2: Select two descriptors for a good clustering of sounds in 2D

Select two of the sound descriptors obtained from Task 1 in order to obtain a good clustering of the sounds of three instruments in a two dimensional space. By visualizing the descriptor values of the sounds in a 2D plot you can choose the features that can help to better cluster these instruments. The function `descriptorPairScatterPlot()` in `soundAnalysis.py` script takes as inputs the downloaded sounds folder (targetDir) and the descriptor pair indices (descInput) (see mapping above) to create a 2-D scatter plot of the descriptor pair. The data points, sounds, from

different instruments are shown with different colors. In addition, you can also plot the Freesound ID of the sounds with the points. Only plot the sounds of the 3 instruments chosen. Make sure that in targetDir you only have the 3 instruments chosen.

Choose a good pair of descriptors for the sounds of the 3 instruments you downloaded in Task 1. A good pair of descriptors leads to a point distribution where all the sounds of an instrument cluster together, with a good separation from the other instrument clusters. Try out different combinations of descriptor pairs. Write a short paragraph on the descriptor pairs you tried out, justifying your choices for selecting those particular descriptors. Based on the spectral and temporal features of the instruments and sounds, give an explanation of why (or why not) a good clustering is (or is not) achieved with the chosen pairs of descriptors.

Attach the 2-D scatter plots for the best descriptor pairs. You can upload more than one scatter plot, up to a maximum of three plots. You can save the plot as a .png files from the plotting window. You should compress them into one zip file which you upload.

Task 3: Cluster sounds of different instruments using kmeans in n-dimensions

After visualizing the sound descriptors, you will now cluster the sounds using more than two descriptors. You can use as many descriptors as you need for the best clustering. Use the same set of sounds obtain in Task 1, starting from the descriptors that you found were good in Task 2, and then adding other descriptors that you feel can improve the kmeans clustering of sounds. The function `clusterSounds()` in `soundAnalysis.py` script takes the sounds folder (targetDir), number of clusters (nCluster) and the descriptor indices (descInput) as input. It then performs a kmeans clustering using the selected descriptors. Make sure that in targetDir you only have the 3 instruments chosen.

For this part, you can use as many descriptors as you need to achieve good clustering and classification performance. However it is best to use as few descriptors as possible in order to make it easier to explain the contribution of each descriptor. Choose the number of clusters to be the same as the number of instruments (i.e., 3). Ideally in such a case, all the sounds of an instrument should go into a single cluster. In reality however, there might be sounds that are outliers and can go into a different cluster. The algorithm takes a majority vote on the sounds in each of the three clusters and assigns each cluster to an instrument. We compute the performance of the clustering by checking the number of points (sounds), that have been wrongly assigned to a cluster. The function clusterSounds() prints the clusters and the sounds assigned to each one. The function also prints the resulting classification obtained with the choice of descriptors you made.

Write a short paragraph explaining the descriptors you used, the resulting classification accuracy you obtained, and your observations on why you obtained (or not) those errors in clustering. Comment if you see any systematic errors (such as a consistent mix up of sounds from two instruments) and possible reasons for that. You should also try to cluster with different subsets of descriptors and mention the classification accuracy you obtain in each case.

Note: Since the cluster centers are randomly initialized every time in k-means, you might see different results every time you run the function. You can report the best result you obtained.