**How to get MFCC and Spectrum features from open-source python packages**

1. Step\_1: Install ffmpeg.
   * For mac, it’s much easier to install ffmpeg, which you can find the instructions from [ref\_1](https://www.idiotinside.com/2016/05/01/ffmpeg-mac-os-x/) or [ref\_2](http://macappstore.org/ffmpeg/).
   * For windows, I found a step-by-step installation guide in [here](https://www.wikihow.com/Install-FFmpeg-on-Windows).
   * Don’t forget to restart your system before ffmpeg can be callable.
2. Step\_2: After you’ve successfully installed and configured ffmpeg, now you need to install the python package(s) for audio signal processing.

 Install **librosa** library by typing in:

pip3 install librosa

in cmd or terminal windows.

1. Step\_3: Now you can write the code to compute the 1D **mfcc** features.
   * Several example code can be found at their [website](https://librosa.github.io/librosa/tutorial.html). You may search in google and learn.
   * Here, paste the main python syntaxes to get the **mfcc** features. You can download all the mentioned code and run in the last section.

**from** \_\_future\_\_ **import** print\_function

**import** librosa

*# ---Load the audio as a waveform `y`*

*# ---Store the sampling rate as `sr`*

filename **=** './UrbanSound/data/street\_music/6508.mp3'

sig, rate **=** librosa**.**load(filename, sr**=**44100)

*# ---Set the hop length; at 22050 Hz, 512 samples ~= 23ms*

hop\_length **=** 512

*# ---Compute MFCC features from the raw signal*

mfcc **=** librosa**.**feature**.**mfcc(y**=**sig, sr**=**rate, hop\_length**=**hop\_length, n\_mfcc**=**13)

The mfcc is what we need as the 1D feature.

You can visualize it by running:

**import** librosa.display

librosa**.**display**.**specshow(mfcc, x\_coords**=**None, y\_coords**=**None, x\_axis**=**None, y\_axis**=**None, sr**=**44100, hop\_length**=**512, fmin**=**None, fmax**=**None, bins\_per\_octave**=**12, ax**=**None)

And get:



For 2D spectrum:

1. Step\_4: Get the 2D features– **spectrum**.
   * Main python syntaxes to compute the **spectrum**:

**from** util\_audio\_processing **import** **\***

**import** matplotlib.pyplot **as** plt

**import** librosa

**if** \_\_name\_\_ **==** '\_\_main\_\_':

*### Parameters ###*

fft\_size **=** 2048 *# window size for the FFT*

step\_size **=** fft\_size **//** 16 *# distance to slide along the window (in time)*

spec\_thresh **=** 4 *# threshold for spectrograms (lower filters out more noise)*

lowcut **=** 500 *# Hz # Low cut for our butter bandpass filter*

highcut **=** 15000 *# Hz # High cut for our butter bandpass filter*

*# For mels*

n\_mel\_freq\_components **=** 64 *# number of mel frequency channels*

shorten\_factor **=** 10 *# how much should we compress the x-axis (time)*

start\_freq **=** 300 *# Hz # What frequency to start sampling our melS from*

end\_freq **=** 8000 *# Hz # What frequency to stop sampling our melS from*

*# ---Load the audio as a waveform `y`*

*# ---Store the sampling rate as `sr`, normally it's 44100Hz.*

*#----Load your audio file(s) ---#*

filename **=** './UrbanSound/data/street\_music/6508.mp3'

data, rate **=** librosa**.**load(filename, sr**=**44100)

**print**(data**.**shape)

wav\_spectrogram **=** pretty\_spectrogram(

data**.**astype("float64"),

fft\_size**=**fft\_size,

step\_size**=**step\_size,

log**=**True,

thresh**=**spec\_thresh,

)

fig, ax **=** plt**.**subplots(nrows**=**1, ncols**=**1, figsize**=**(20, 4))

cax **=** ax**.**matshow(

np**.**transpose(wav\_spectrogram),

interpolation**=**"nearest",

aspect**=**"auto",

cmap**=**plt**.**cm**.**afmhot,

origin**=**"lower",

)

fig**.**colorbar(cax)

plt**.**title("Original Spectrogram")

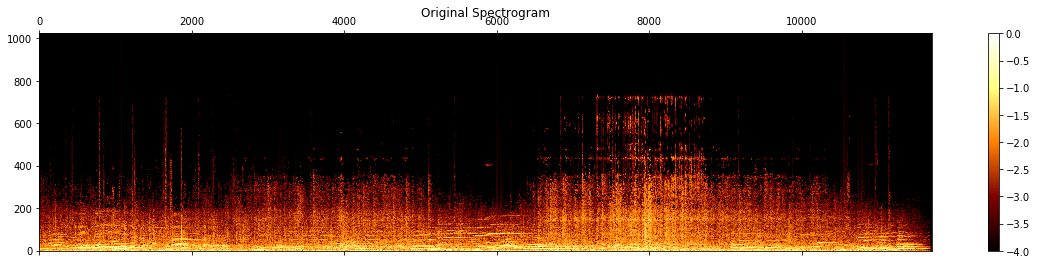
data **=** butter\_bandpass\_filter(data, lowcut, highcut, rate, order**=**1)

*# Only use a short clip for our demo*

**if** np**.**shape(data)[0] **/** float(rate) **>** 10:

data **=** data[0 : rate **\*** 10]

**print**("Audio length in time (s): ", np**.**shape(data)[0] **/** float(rate))

* + [Demo] spectrum feature:
  + You need to save the spectrum (e.g using *plt.savefig()*) as an image before sending it to your training model.

1. You can download the sample code from:
   *  [util\_audio\_processing.py](https://people.ryerson.ca/bowu1004/blogs/06/03/source_code/util_audio_processing.py)
   *  [mfcc\_demo.py](https://people.ryerson.ca/bowu1004/blogs/06/03/source_code/mfcc_demo.py)
   *  [spectrum\_demo.py](https://people.ryerson.ca/bowu1004/blogs/06/03/source_code/spectrum_demo.py)
   * Make sure you download all the three files, and change the *filename* to your data location in the two main files (*mfcc.demo.py* and *spectrum\_demo.py*).
2. What’s next?

\*Apply and search for more codes and run the program and use your own codes\*