Music Genre Classification–Base line report

Hung-Ting Wen[*] and Yun-shao Sung[†]

1. Baseline approaches. For baseline approach, we use the same data set as the reference paper used, GTZAN data set. We use SOX to sample at 22050Hz and partition data sets into intervals, each for 3 seconds. Therefore, each clip will have 66150 values representing music signal intensity within the 3 seconds window.

After data preprocessing, we read outputs and transfer into python readable files, so that we can use scipy tools on it.

Second approach is to use pyfilterbank packages for mel-frequency bank generation. As trying to reproduce the method of our reference paper, we prepared 12 of the frequency banks covering signal from 0 to 6000 Hz range. As scattering method need to be convoluted with a lowpass filter at the end, we designed the lowpass filter by scipy.signal.butter for 6500 Hz cutoff frequency, 22050 Hz sampling rate, and in the order of 6. Then scipy.signal.lfilter was performed to filter the signal on frequency domain, which in other word, performing convolution with signal clips on time domain. For the convolution part during scattering, numpy.convolve was performed during eachtime of scattering process.

During scattering process, each clip in a song will produced 1 product at 0-ordered scattering, 12 products with each mel-frequency banks at 1-ordered scattering, and then the summation of 11 to 1 during 2-ordered scattering wich yield 66 products. Therefore, given total 10 clip per song and each clip will produced 79 products during scattering process, each song now will have 790 vectors of scattering-modified clips. As each clip contains 66150 values and this scattered song will genreate huge size of data which is not pratical for any machine with limited memory, therefore we performed energy conversion for each of the clip converting 790 signal vectors into 790 signal values.

2. Software.

2.1. Data Preprocessing.

1. SOX is used to sampel source file at given frequency and help partition into clips with given interval.
2. scipy.wavfile is used to help read wav file into numpy array.

2.2. Scattering.

1. pyfilterbank is used to generate frequency bank that we need to use to filter out unwanted signals.
2. numpy is used to do necessary convolution.

2.3. Learning Algorithm.

1. scikit-learn.SVM is used as our baseline learning algorithm. We use SVC algorithm with degree of 2 to classify samples.

[*] htw230@nyu.edu

[†] yss265@nyu.edu

Our main difficulties is the performance. The performance for scaterring each of the song is very slow. Each song has to perform 1320 times of convolutions, 790 times of lowpass filtering, and then 790 times of energy conversion. Therefore approxmately each song will need 10 to 15 minutes for scattering process.

3. Dataset. For baseline approach, we download the GTZAN data set. It contains 10 different music generes, and 100 of 3 minutes musics in .au format per gerere, which is 1000 songs in toal and about 3GB for its data size. Then SOX was performed for data preprocessing that we sampling each songs with 22050Hz and divided it into 10 clips in .wav, each for 3 seconds. Therefore, each clip will have 66150 values representing music signal intensity within the 3 seconds window.

REFERENCES

[1] Low pass filter by FFT convolution, *http://www.dsprelated.com/freebooks/sasp/Example_1_Low_Pass_Filtering.html*

[2] Multiscale Scattering for Audio Classifications, *http://www.cmap.polytechnique.fr/scattering/ismir-final.pdf*

[3] Digital image processing: p067- Dictionary Learning, *https://www.youtube.com/watch?v=XLXSVLKZE7U*

[4] Interior-point methods, *https://web.stanford.edu/class/ee364a/lectures/barrier.pdf*

[5] Iterative Shrinkage/Thresholding Algorithms, *http://people.ee.duke.edu/ lcarin/figueiredo.pdf*

[6] Matching pursuits with time-frequency dictionaries, *http://www.cmap.polytechnique.fr/ mallat/papiers/MallatPursuit93.pdf*

[7] Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit, *http://www.cs.technion.ac.il/ ronrubin/Publications/KSVD-OMP-v2.pdf*