Capstone Project

Machine Learning Engineer Nanodegree

Jialei Yang
September 29th, 2016

I. Definition

Project Overview

Music is one of the most ancient cultures of human beings. From religious purposes to entertainment, it plays an important role in our life. Music can be divided into genres and different people may like different genres of music. People who have music knowledge are able to distinguish different music genres. It is interesting to explore the capability of development of automatic music genre classification. The developed system can also be used as a tool for music recommendation system.

In the project, a music genre classification system was built to recognize five genres of music using the GTZAN Genre Collection[1]. They are: blues, classical, jazz, pop and rock. Each genre has 100 music pieces which are all 30 second long with 22050 HZ frequency.

Problem Statement

The goal of the project is to build a system to classify five music genres (blues, classical, jazz, pop and rock) based on audio signals. The original audios are 30 second long. But in real application, 30 second is a bit long and inconvenient for users to use. So in the project, audios are cut to only 10 second length. The whole development can be divided into following three tasks:

1. Data collection and preprocessing:

This project used public dataset: the GTZAN for analysis and development. Audio signals are then preprocessed for further using. The preprocessing includes: 1) convert format of audio signals from .au to .wav since there are more support for .wav files in python; 2) cut each 30 second audio to 10 second; 3) normalize the audio signals

2.Feature extraction

There are various proposed audio features can contribute to the classification. Typically they are divided into three parts: Timbre features, rhythmic features and pitch features. In this

system, three types of timbre features: Mel-frequency Cepstrum Coefficients (MFCCs) Zero-crossing Rate and Short-time Energy and the inter-quartile range of the signals are extracted

3.Classification

K-nearest Neighbor Classifier is the main classifier in this project.

Metrics

Because all five classes have same amount of samples, there is no need to consider unbalance of the data. So accuracy is used for the main metrics to estimate the performance. The formula is:

$$\text{Accuracy}=\text{Number of samples in correct classes}/\text{Number of all samples}$$

Besides accuracy, confusion matrix is built in order to have a direct look for the performance of all classes.

II. Analysis

Data Exploration

The GTZAN Genre Collection has 10 genres (Blues, Classical, Country, Disco, Hiphop, Jazz, Metal, Pop, Reggae and Rock), each represented by 100 audio tracks. Each track is 30 seconds long with 22050Hz and in .au format. In this project, five of the genres (Blues, Classical, Jazz, Pop and Rock) are selected. They are all well known genres and people who know music a little bit may be able to distinguish them from each other. Figure 1 shows a sample in Blues genre. It has 22050*30 values.

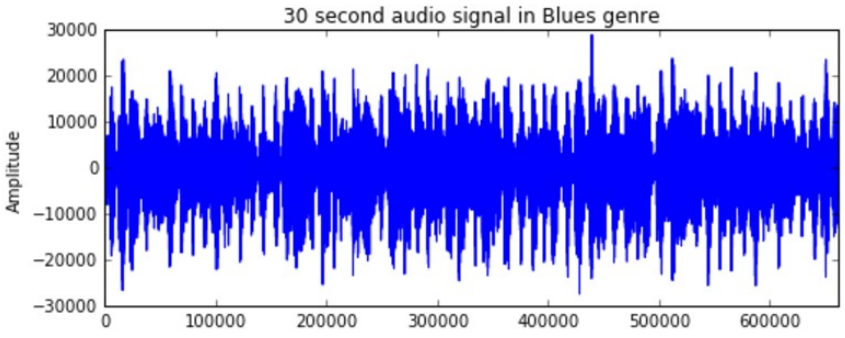

Figure 1. A piece of 30 second audio signals in Blues genre

Exploratory Visualization

Values of audio signals are called amplitude. They balance and control the loudness of sounds. This is one of the direct feature we can use when look at the signals. In order to study whether the distribution of amplitudes will be different with different genres, the inter-quartile range (IQR) of amplitude of each song track is calculated. IQR is a measure of variability, being equal to the difference between the upper and lower quartiles:

$$IQR=Q3-Q1$$

Then the box-plots of five genres are shown in Figure 2.

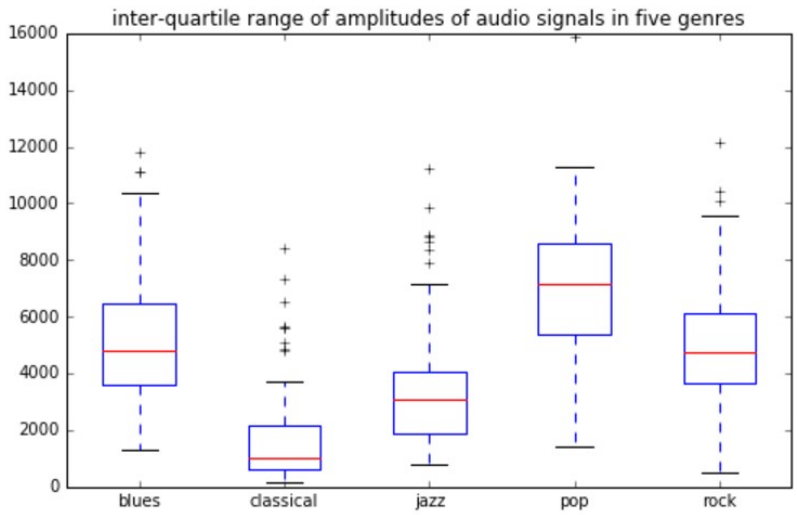

Figure 2. Inter-quartile range of amplitudes of audio signals in five genres. On each box, the central mark is the median, the edges of the box are the 25th (Q1) and 75th (Q3) quartiles. The black crosses indicate the outliers. Points are drawn as outliers if they are larger than Q3+1.5(Q3-Q1) or smaller than Q1-1.5(Q3-Q1).

From the plots we can see that five genres have different dispersion of amplitudes. Classical tend to have the lowest IQR, which means variability of amplitude of classical music is smaller than other four classes. Blues and Rocks are in the similar level, so this feature is not good enough to separate them. The plots show that the IQR can be used as one of the feature for music genre classification.

Algorithms and Techniques

In feature extraction part, four type of features will be extracted from the preprocessed audio signals. In classification part, K-Nearest neighbor (KNN) is used to fulfill the mission. These techniques are introduced below.

1) MFCCS (Mel-frequency Cepstral Coefficients)

MFCCS is a popular and famous method can describe the envelope of the power spectrum. It

is widely used in speech recognition. But recently more proposed methods use it to modeling music[2].

Mel-frequency is a feature space that simulates the human hearing system. Because human hearing system is not a linear system and more sensitive to low frequency sound, using Mel-frequency can extract more features which human ears will extract. The following formula is the relation between normal frequency and Mel-frequency:

$$M(f) = 2595 \times \log10(1 + \frac{f}{700})$$

in which, f is the normal frequency and M(f) is the corresponding Mel-frequency.

The basic process of calculating MFCCS shows below (Figure 3):

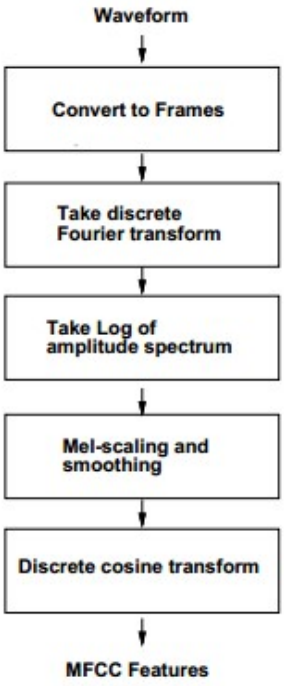

Figure 3: Process to create MFCCS features[2]

As a spectral feature, the first step is always to create the spectrum. This step is completed by first cut the whole audio signal to short time frames. Then take discrete Fourier Transform for each time frame. After get the spectrum, take log of the amplitude of the spectrum. The next step is to Mel scale and smooth. The Mel scale is to map the normal frequency to Mel frequency. Finally, in order to decorrelate the resulting feature vectors a discrete cosine transform is performed [1].

2) Zero-crossing rate

Zero-crossing rate is a temporal feature. It counts how many times the audio signal crosses the zero point. In order to obtain more detailed information, the signal is usually cut into short time period. The definition of zero-crossing rate is:

$$zcr = \frac{1}{T-1}\sum_{t=1}^{T-1} II(s_t s_{t-1} < 0)$$

Where s is the signal and T is the length, the function II(A) is 1 is A is true, or is 0.

3) Short-time energy

The short-time energy is also a feature in time domain. It is the energy in short-time period. Short-time energy is a simple and effective classifying parameter for voiced and unvoiced segments [3]. It also can extract the end point of the audio segments. The formula of short-time energy is:

$$E_n = \sum_{m=-\infty}^{\infty} [x(m)w(n-m)]^2$$

Where w () is the window for the time period, x is the signal.

4) K-Nearest Neighbor (KNN)

KNN is a non-parametric and instance-based learning method to classify different classes of data. The idea of KNN is to find which class has more points close to the test data. It chooses a specific k which means find k neighbors. Figure 4 shows an example [4]. In order to classify the testing data (green circle), k training samples near the green circle are found. When k=3, two red triangles and one blue square are the nearest three samples to the green circle (within solid line), in this case, the testing sample will be classified to the red class. when k=5, three blue squares and two red triangles are found (within dash line), so the green circle will be classified to blue class this time.

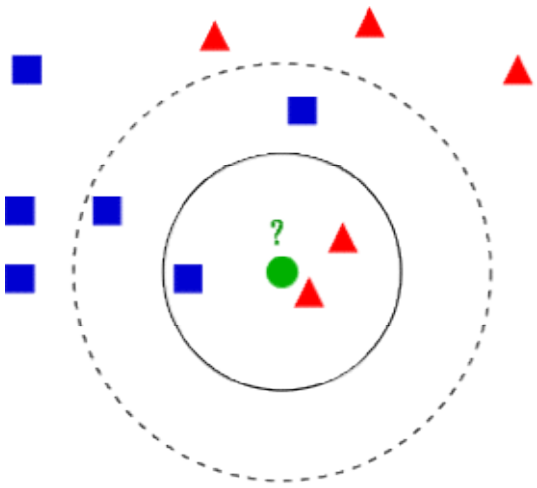

Figure 4: An example of KNN classifier. The green circle is a test sample. Blue squares and red triangles represent training samples in two different classes. The goal is to classify the green circle to either of the two classes

The 'nearest' in KNN is defined by distance measures. A commonly used distance metric is Euclidean distance, it is also the metric used in this project. As a non-parametric method, when the training sample goes to infinite, it will give a best result.

The parameter k is chosen based on data. When k=1, the classifier will predict the testing sample to be class of the closest training sample. In this project, a common grid search method is applied to select the best k for this data. The method basically loops through a range of candidate k and compute the performance of each of them. The one with the best performance will be selected as the k for the data set.

KNN is one of the simplest machine learning algorithms, however, because it is sensitive to the structure of the data. It requires good feature sets to support the classification.

Benchmark

In study of [5], performance of human music genre classification is reported. The study selected 190 music tracks in 19 different genres for experiments. Ordered classification accuracies of 24 participants are shown in Figure 5.

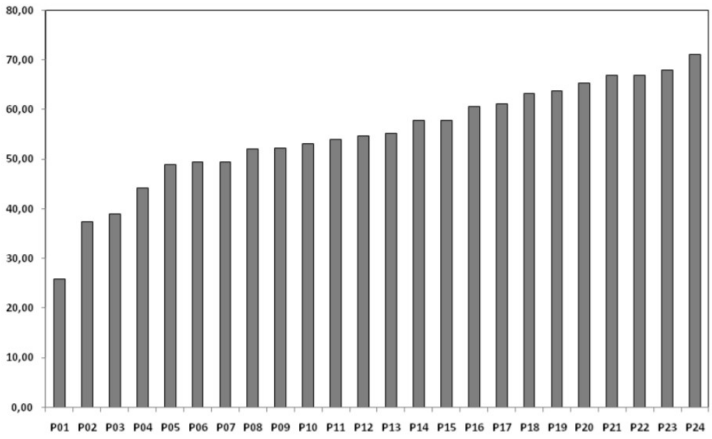

Figure 5: Ordered classification accuracies of 24 participants in [5]

Above figure shows that accuracies range from 26% to 71% and the average accuracy is 55%. However, the study used 19 different genres comparing with only 5 genres in this project. So my goal is to achieve at least 70% accuracy for 5 genre classification.

III. Methodology

Data Preprocessing

The GTZAN dataset is already a well defined and well organized dataset. As mentioned above, only three preprocessing steps were implemented: format conversion, music cut and audio signal normalization.

The original audios are in .au format. However, python doesn't have a good library to read in this format of file. So a piece of matlab code was written to convert the .au file to .wav file. The codes are shown below.

```
%read in music
data_path='.\genres\';
save_path='.\genres2\';
genres={'blues','classical','jazz','pop','rock'};%5 music genres
for i=1:5 %for 5 genres
    flist=dir(fullfile(data_path,genres{i},'*.au')); %read files in data_path
    save_list=[save_path genres{i} '\'];
    n=length(flist); %number of files
    for j=1:n
        [data,fs]=audioread(fullfile(data_path,genres{i},flist(j).name)); %read music
        audiowrite([save_list num2str(j) '.wav'],data,fs);
    end
end
```

The second step is to cut the music into 10 second length. As discussed above, a short piece of audio is more reasonable in real application. The project cut each audio signal from beginning to the 10 second point.

The third step is signal normalization. Since all audios are in different amplitude level, normalization can make their volume close to each other and lead to more robust feature. However, original signals are used to calculate the IQR feature because the variability of amplitude is important for this feature. The formula of normalization shown below:

$$signal = signal / \max(\text{abs}(signal))$$

in which *signal* is the 10 second audio.

Implementation

The preprocessing leaves us two types of signals: normalized and original ones. They both have 500 samples and each sample has 220500 (22050 Hz*10 sec) values. Then four type of features are extracted for each sample.

1) MFCCs (Mel-frequency Cepstral Coefficients)

Code of MFCCs feature extraction were written based on the processes of computing MFCCs shown above and an online tutorial[6]. The parameters were also selected according to the online tutorial.

1) The first step was to generate the power spectrum. This was completed by first cutting the normalized 10s signal to short time frames using a sliding window. The sliding window was 40ms long (882 points) with 10ms (220 points) overlap. Next, the DFT was applied with 1024 sampling points in each time frame to obtain the complex spectrum. Finally, the

power spectrum was obtained by taking the square of the absolute value of the complex spectrum.

2) The second step was to compute a mel filter bank. The filter bank consists of 40 filters and ranging from 300 - 10000HZ. According to the frequency range, 42 points were generated, linearly spaced between the minimum and maximum mel-frequency. Then these frequencies were converted to regular frequency and rounded to the nearest FFT bins obtained from DFT. Forty triangular filters were built through these points. Figure 6 show the filter bank.

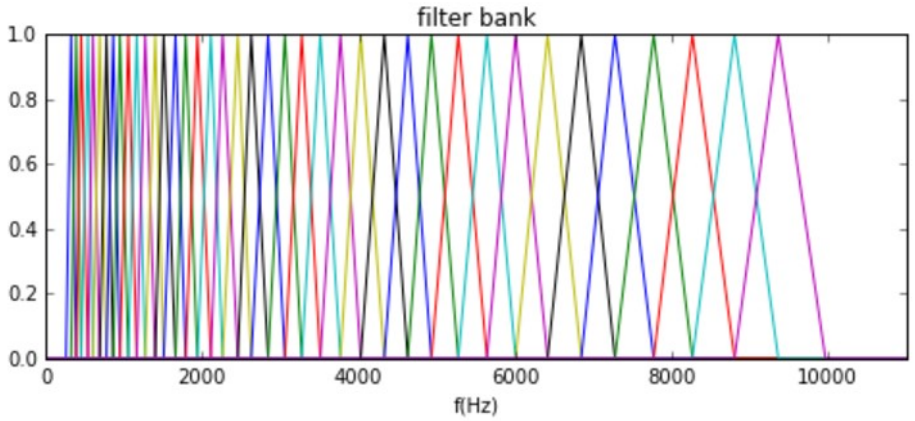

Figure 6: Filter bank contains 40 triangular filters ranging from 300 to 10000Hz

3) Then, each filter was multiplied with the power spectrum and summed to get 40 bank energies for each time frame.
4) The next step was to get the log bank energies. This was completed by taking the logarithm of each of the 40 bank energies.
5) The Discrete Cosine transform (DCT) of the 40 log energies was taken to give 40 cepstral coefficients for each time frame. Only the 2nd to the 13nd were kept for further use. The very first coefficient was discarded because it was the DC term. The higher coefficients were discarded because they represent fast changes in the filter bank energies and it turns out that these fast changes actually degrade ASR performance[6]. So for each time frame, there were 12 coefficients.
6) To combine information in all time frames, for each coefficient, compute its mean value and variance along the time axis. That gave 12 means of MFCCs and 12 variance of MFCCs. In the end, there were 24 features from MFCCs. They were 12 mean and 12 variance values of the coefficients.

2) Zero-crossing rate

To calculate the zero-crossing rate, the normalized signal was first cut into 2 blocks and each block represents 5 second. Then the adjacent amplitude of signals were compared to see if it cross the zero point. Number of times the amplitude crossing the zero point within the 5 second window was count. Finally the rate of zero crossing times in each window was calculated. So in the end, there are 2 features left.

3) Short-time energy

The short-time energy was also calculated in two 5 second window using normalized signals.

In each window, the square of the amplitude were added up and divided by the samples in each period. The process also results in 2 features.

4) **IQR**

IQR is the only feature that was calculated using the raw signals. For each 10s audio track, the 3rd quartile (Q3) and 1st quartile (Q1) were calculated. Then IQR was computed by subtracting Q1 from Q3.

After obtained all 29 features, feature normalization were applied. The min-max normalization was used to scale features between 0 and 1.

Finally, KNN were applied for classification purpose. Because the dataset only consists of 500 samples which is not a very large quantity. Considering the limited amount of samples, k-fold cross validation and leave-one-out strategy were applied for training and testing.

With k-fold cross validation, who data set was divided into k parts. Each time one part of samples were set aside for testing, the remaining ones (other k-1 part of samples) were used for training. The process was repeated for k times until all samples were tested.

Leave-one-out process is a special case of k-fold cross validation, in which the k is equal to the number of samples (500 in this case). Every time one sample was set aside for testing, the remaining ones (499 samples) were used for training. The process was repeated for 500 times until all samples were tested. Compared with k-fold cross validation, leave-one-out is more time consuming, but it tends to give a more unbiased validation since most of the data are involved in the training process.

The average accuracy of all folds were then calculated. The confusion matrix was also built according to the result of the k-fold cross validation and leave-one-night out process. The parameter k is set to 5, which is the default value of the KNN function in sklearn package.

Refinement

The initial training uses the default parameter k for KNN classifier. In order to improve the performance, the grid search parameter selection process was implemented. The candidate k is between 1 to $\sqrt{n}$ (n is the sample number, 500 in this case). Considering the time consumption, 10-fold cross-validation was used instead of leave-one-out during the grid search process. Figure 7 shows the training and validation accuracy with different k values.

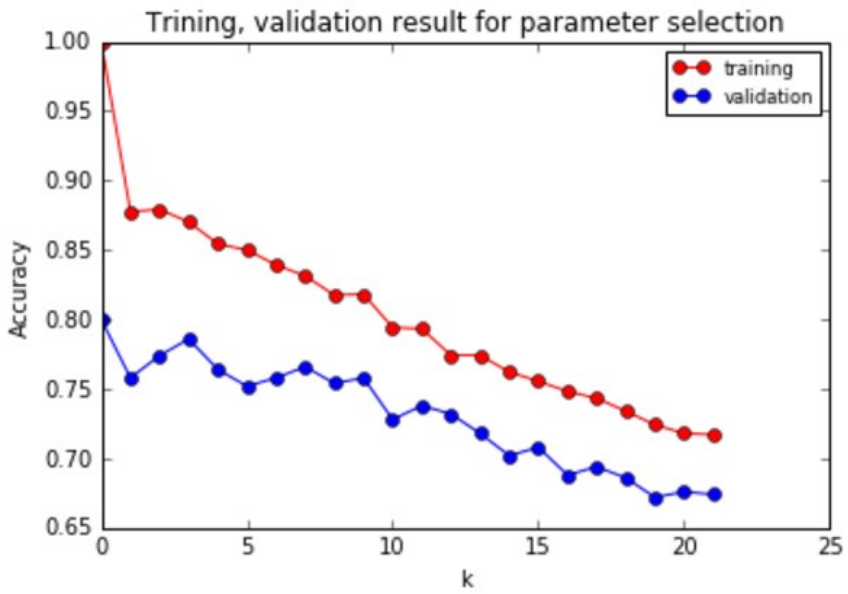

Figure 7: Filter bank contains 40 triangular filters ranging from 300 to 10000Hz

It is clearly that when *k* is 1, the model has the best performance.

IV. Results

Model Evaluation and Validation

After trying different features and parameter selection as discussed above, the final model was chosen based on the highest accuracy. It has 29 features and a KNN classifier with *k* equals to 1. Leave-one-out process was applied in order to get a reliable evaluation. The confusion matrix after leave-one-out process is shown below.

		Real Class				
		Blues	**Classical**	**Jazz**	**Pop**	**Rock**
Estimate Class	**Blues**	**82**	1	11	4	15
	Classical	1	**91**	8	1	3
	Jazz	2	3	**73**	3	6
	Pop	2	1	1	**85**	7
	Rock	13	4	7	7	**69**

The accuracy for the model is 80%.

In order to compare k-fold cross validation with leave-one-out process, two experiments with k=5 and k=10 were conducted. The results is shown below.

- 5-fold cross validation

		Real Class				
		Blues	**Classical**	**Jazz**	**Pop**	**Rock**
Estimate Class	**Blues**	**80**	0	10	3	14
	Classical	1	**89**	13	3	3
	Jazz	3	4	**67**	1	4
	Pop	2	2	2	**86**	7
	Rock	14	5	8	7	**72**

The accuracy for the model is 78.8%.

- 10-fold cross validation

		Blues	**Classical**	**Jazz**	**Pop**	**Rock**
Estimate Class	**Blues**	**81**	0	9	4	14
	Classical	1	**90**	8	1	3
	Jazz	2	5	**74**	2	6
	Pop	2	1	1	**85**	7
	Rock	14	4	8	8	**70**

The accuracy for the model is 80%.

The confusion matrixes and accuracies of two k-fold cross validation show that the number of k chosen for the k-fold had influence on the final results we will see. The characteristics of KNN make it sensitive to the change of training data. So if time permit, leave-one-out may be the most suitable evaluation model for KNN.

In order to test robustness, Gaussian noises with 0 mean and 1 standard deviation were added to the original audios. The amplitude for the noises were set to 1/100 of the maximum amplitude of each original music tracks. After that, the same process was applied and similar results were achieved.

		Real Class				
		Blues	**Classical**	**Jazz**	**Pop**	**Rock**
Estimate Class	**Blues**	**81**	2	6	6	15
	Classical	0	**83**	8	0	4
	Jazz	4	9	**76**	1	7
	Pop	2	2	2	**81**	7
	Rock	13	4	8	12	**67**

The accuracy for noisy data is 76.6%.

The confusion matrix and accuracy shows that the model is robust to Gaussian noises added to the original signals. The only drop of the performance is the Classical music.

Justification

First, the accuracy of 80% satisfied my initial goal of achieving at least 70% accuracy. With a close look at the confusion matrix, we can see that Classical music has the best performance. It can be understood since classical usually sounds different and most people can separate it with others. On the contrary, Rock seems to be easily mixed with others. It can also be explained by this introduction from Wiki:" *It has its roots in 1940s' and 1950s' rock and roll, itself heavily influenced by blues, rhythm and blues and country music. Rock music also drew strongly on a number of other genres such as electric blues and folk, and incorporated influences from jazz, classical and other musical sources.*" Its own characteristic could be the reason of the worst result.

Although the model meet my expectation, I think it may not have similar performance with other data set. Like Rock music, the boundary of different genres is not clear. One kind of music will easily take some features belong to other genre. So the performance of the system may not only depends on the algorithms, but also depends on the music themselves. In order to solve the music genre classification in real life, deeper exploration should be taken.

V. Conclusion

Free-Form Visualization

Basically, KNN classify samples based on distance measurement. In order to have a visualized look of the distances among all samples. A distance plot was generated in Figure 8. This was generated by:

1. Calculate the pair-wise distances of features for all samples

2. Normalize the distance values between 0 to 1

3. Plot the distance matrix as image.

Each point on the image is a distance between the row sample and column sample. The black diagonal means 0 distance between one sample itself. The darker the color is, the smaller the distance is. From the plot, we can roughly find 5 classes corresponding to 5 genres. The plot gives a idea of whether the features work for the problem and whether a distance measure is good enough for classification.

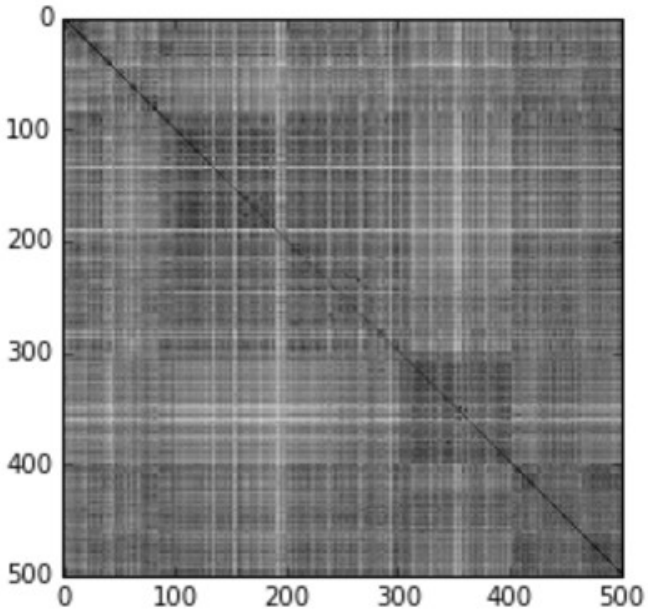

Figure 8: Distance plot

Reflection

The process of this project is summarized as following:

1. Search for public data set that is suitable for this problem

2. Analyze the structure of the data set

3. Preprocess the data based on need of the project

4. Read literature of similar problems and find useful features

5. Extract different type of features

6.Build the basic process of training and testing (Leave-one-out)

6. Explorer different classifiers and features based on the training process

8.Apply parameter selection for classifiers

9. Select the best combination

The most challenge part is to find the correct model for the problem. Because I don't have much music knowledge and experiences with audio signals. It takes a while to find the correct direction for the problem. There are many good features and classifiers, but how to choose the right ones is difficult. I believe there are combinations of features and classifiers that can generate better results. The interesting part is that I was able to learn more about music and also some signal processing techniques. I was also able to try different types of classifiers and got some feeling about what are the differences between them.

Improvement

In the project, only timbre features were extracted from the audio signals. In fact, there are many other types of features are shown good performance in audio or speech analysis. Adding pitch features or rhythm features should benefit the accuracy. However, with more features adding into the system, more data will be needed due to the "curse of dimensionality". In the case of limited data, feature selection or dimensionality reduction (such as PCA) can be applied.

---

[1] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. IEEE Transactions on Speech and Audio Processing, 10(5), July 2002.

[2] B. Logan, “Mel frequency cepstral coefficients for music modeling,” Proceedings of the International Coference in Music Information Retrieval (ISMIR), pp. 11-23, 2000.

[3] D.Enqing, L.Guizhong, Z.Yatong, c.Yu, "Voice Activity Detection Based on Short-Time Energy and Noise Spectrum Adaptation," 6th International Conference on Signal Processing, 2002.

[4] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[5] S. Lippens, J.-P. Martens, T. D. Mulder, et al., “A Comparison of Human and Automatic Musical Genre Classification,” in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Montreal, 2004

[6]http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/