CS 534 Final Project

Automatic Detection of the Genre of Music

Team Members:

I-Shen Liao,

Kaijun Zhang,

Xuemeng Wang

Introduction

This project is the automatic detection of the genre of music. The classifier can learn how to differentiate between the different genres with accuracy.

Installation

In the beginning, we need to install the following packages which are NumPy, PyDub, SciPy, scikit-learn, statsmodels and python_speech_features. NumPy is the fundamental package for scientific computing with Python which contains a powerful N-dimensional array object, useful linear algebra, Fourier transform, and random number capabilities. Pydub has a facility for getting the audio data as an array of samples. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, and other tasks common in science and engineering [1]. Statsmodels is a Python library which uses for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. The python_speech_features provides common speech features for automatic speech recognition including MFCC (Mel Frequency Cepstral Coefficients) [2]. The Mel Frequency Cepstrum encodes the power spectrum of a sound. It is calculated as the Fourier transform of the logarithm of the signal's spectrum. The Talkbox SciKit (scikits.talkbox) contains an implementation of of MFC that we can directly use. The data that we feed into the classifier is stored as ceps, which contain 13 coeffecients to uniquely represent an audio file.

The Dataset

In this project, we use the dataset from GTZAN which is a well-known dataset in musical genre classification. The audio files are a variety of sources including personal CDs, radio, microphone recordings, so it can represent many different recording conditions [3]. Also, the dataset was used for the paper "Musical genre classification of audio signals" by G. Tzanetakis and P. Cook in IEEE Transactions on Audio and Speech Processing 2002. The dataset contains 10 genres including blue, jazz, hip-hop, rock, classical, and so on. Each genre contains 100 different songs, so the total songs are 1000 songs. As the Figure 1, it is the representation of the frequency in a song. It shows the intensity of the frequencies on the y axis, and the specified time intervals on the x axis. As you can see, the songs belonging to the same genre have similar spectrograms. We can design a classifier that can learn to differentiate between the different genres with accuracy.

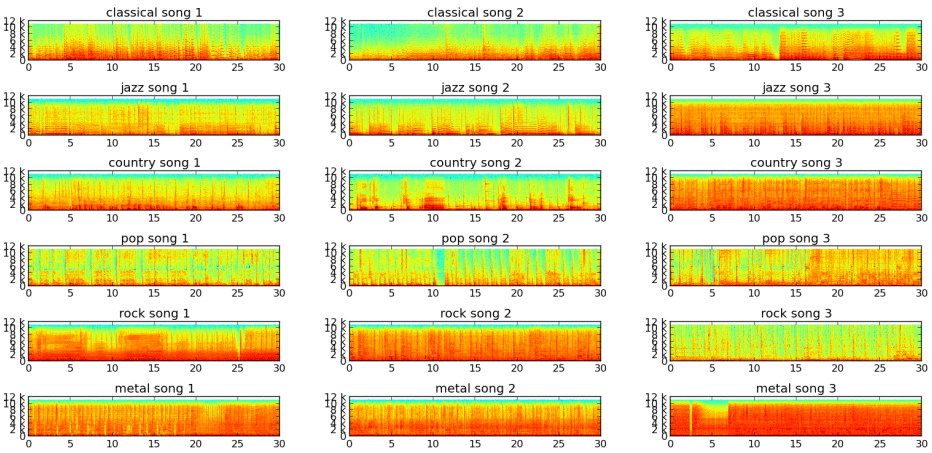

Figure 1: The representation of the frequencies in songs

Model Generation

There are many steps to generate the model. The first step is to converts each audio file in the GTZAN dataset to numpy arrays which can be used by the model. The way how we convert audio files to numpy arrays is that we use MFCC which is a library in the python speech features. The data we feed in the classifier contains 13 coefficients to represent an audio file.

After converting, the model is able to train the classifier. In this project, the machine learning algorithm are SVM and K-NN. Once the model has been generated, we use the model to predict the genres of the testing audio files.

Existing Approaches

The first algorithm we used is SVM which is the basic algorithm for machine learning, so we use SVM as our first algorithm. We treat one song as one training set. It does not use too much time for training and the results are bad. The accuracy rate is just 33%. It is too low to predict the genre of the music.

To judge the overall performance, a confusion matrix is produced. A confusion matrix is a specific table layout that allows visualization of the performance of an algorithm. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class.

The confusion matrix with all the genres selected is shown below.

According to figure 2, this is a chart about all the accuracy rate for genres. The depth of the color means the accuracy rate of the genre. The deeper color, the higher of the accuracy rate. The color on the diagonal shows the correct prediction, so we need to focus on those parts on diagonal. Then we can see from the chart that some genres predicts very good like classical and pop music. Their color on diagonal are deep and the rest of the part so light. It means that SVM can correctly regonsize this genre with others, but for

some gernes like jazz, all of the parts within jazz are so light. It means that SVM can not distinguish it with other genres. So it make problems. The accuracy for these genres are lower than normal genres.

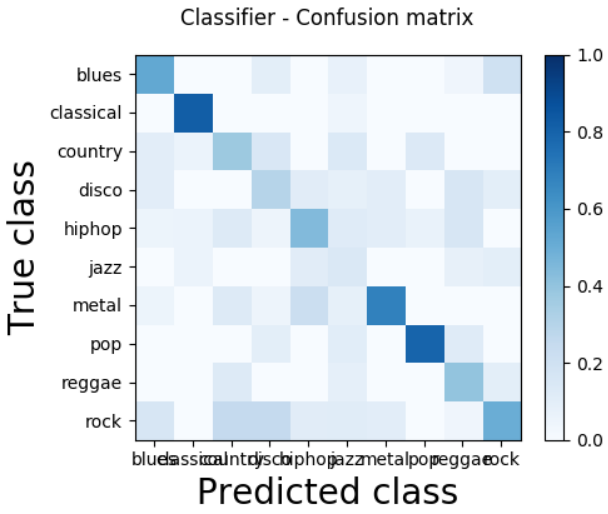

Figure 2: Confusion matrix for SVM & no data operation

Then we try to figure out what causes this problems. As figure 3 showed, classical song and jazz songs' frequencies are so similar with each other. The reason is that we treat the one song as one training data, so the average frequency of classical and jazz seems similar. The model treats them as same genre, and would not do update when this two different songs occurred on the training set. However, some prediction of genres are good becasue their frequencies are very different from others. So that the prediction are good enough when training data is one songs.

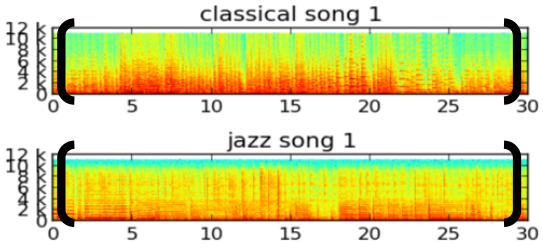

Figure 3: The frequency of songs

Above all, SVM with no data operation can not predict well. Therefore, we need to improve our algorithm as well as make some feature engineering on the training set.

Our Approaches

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive. The need for manual feature engineering can be obviated by automated feature learning. In our project, we chose to cut our training examples into ten pieces in order to improve our accuracy rate.

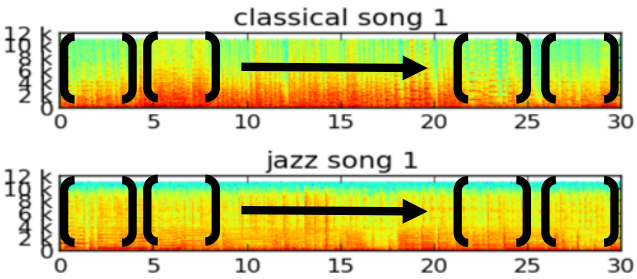

Figure 4: The frequency of songs with ten pieces

The advantage of doing this is we can add more training times. Initially, we have 1000 training examples. After cutting them into ten pieces, we have 9000 more training times. In this situation, we improve our accuracy rate from 33% to 49.8%.

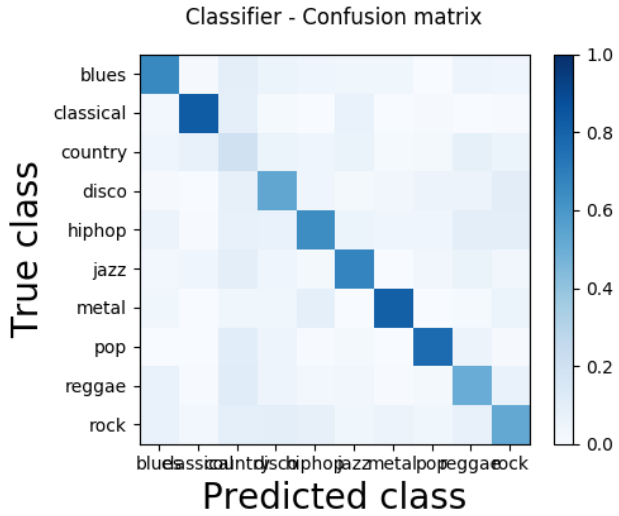

Figure 5: The confusion matrix for SVM & data operation

From the image above, you can see the diagonal line. Most colors are individual to the rest and have better color depth.

Results

KNN makes predictions using the training dataset directly. Predictions are made for a new instance (x) by searching through the entire training set for the K most similar instances (the neighbors) and summarizing the output variable for those K instances. For regression, this might be the mean output variable, in classification this might be the mode (or most common) class value.

To determine which of the K instances in the training dataset are most similar to a new input a distance measure is used. For real-valued input variables, the most popular distance measure is Euclidean distance. Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (xi) across all input attributes j.

$$\text{Euclidean Distance}(x, xi) = \text{sqrt}(\text{sum}((xj - xij)^2))$$

Our finally accuracy rate is based on KNN with K equals to five, and if setting K to values that greater than five, then the training would be overfitting.

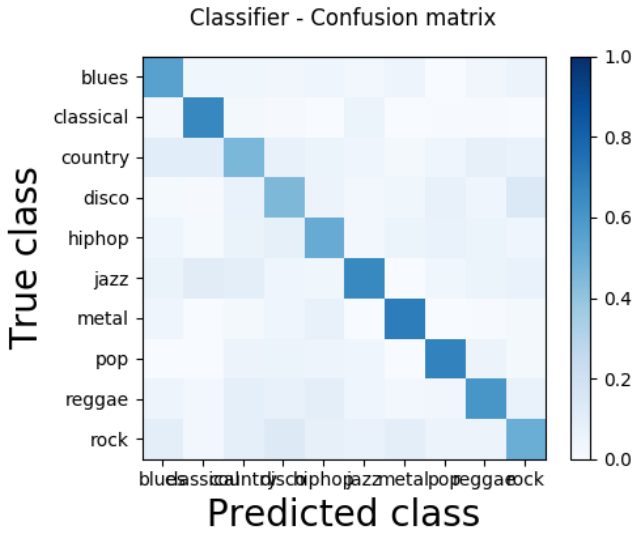

Figure 6: Confusion matrix for KNN (K = 5) & data operation

Our final result of KNN is 58% accuracy rate. Based on this accuracy rate, we did a prediction on jazz music genre with ROC (Receiver Operating Characteristic) curve. The best possible prediction method would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). ROC AUC (Area Under Curve) varies between 0 and 1, with an uninformative classifier yielding 0.5. According to the image below, the shape is closed to upper left corner, and AUC is closed to 1. It means the prediction is good.

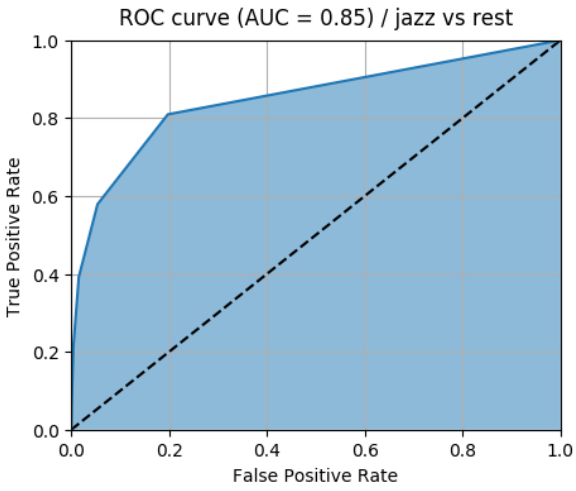

Figure 7: ROC curve of jazz music genre

Discussions

The reason leads to inaccurate result is that our sample number is not enough. We only have 1000 music samples so we find a method to cut them into ten pieces in order to increase the training sample number. Which algorithm is better? SVM is mostly used practically. SVM can be used in linear or non-linear ways with the use of a Kernel, when you have a limited set of points in many dimensions SVM tends to be very good because it should be able to find the linear separation that should exist. SVM is good with outliers as it will only use the most relevant points to find a linear separation (support vectors). KNN has some nice properties: it is automatically non-linear, it can detect linear or non-linear distributed data, it tends to perform very well with a lot of data points. On the minus side KNN needs to be carefully tuned, the choice of K and the metric (distance) to be used are critical.

Conclusions

Based on our final project, we learn how to convert music resources to numpy array. Besides, we also learn how to use SVM and KNN model to train music resources. Performance tuning is necessary and data operation plays an important role in this final project.

Reference

[1] SciPy. (2017). Documentation [Online]. Available: https://www.scipy.org/docs.html

[2] James Lyons. (2013). Welcome to python_speech_features's documentation [Online]. Available: http://python-speech-features.readthedocs.io/en/latest/

[3] MARSYAS. (2002). Data Sets. [Online]. Available: http://marsyasweb.appspot.com/download/data_sets/