Music Genre Classification - Project Status Report

Ryan O'Gorman ROGORMAN@SEAS.UPENN.EDU
Luke Mainwaring LUKEMAIN@SAS.UPENN.EDU
Alex Matthys AMATTHYS@SEAS.UPENN.EDU

Abstract

As music streaming services such as Spotify and Apple Music become increasingly popular, customers are often searching these streaming sites for music based on a specific genre. This means firms like Spotify must be able to determine the genre for all of their songs, and with hundreds of songs being uploaded daily, these firms are presented with a genre classification problem. In this paper, we analyze the effectiveness of different machine learning algorithms on predicting a song's genre.

1. Project Introduction

There does not exist a lot of research into the field of music genre classification. One primary reason for this is that genre classification can at times be subjective, as songs often times lie in a grey area between two specific genres. As such, we began our studies by analyzing classification algorithms between objectively different genres. Specifically, we have looked into genre classification for songs that are either Metal or Classical. As next steps, we plan to develop models that will pinpoint where on the spectrum between Genre 1 and Genre 2 a specific song lies, effectively dealing for this grey-area issue in genre classification. Further, we will add more genres of songs into our data set, effectively turning this into a multiclassification ML problem.

There are many real world applications for music genre classification. The primary use case, as discussed briefly in the Abstract, is for digital music streaming companies such as Spotify, Apple Music, and Pandora to be able to immediately classify the genre of songs as they are being uploaded to their platform. Further, if an artist uploads his/her song to a music streaming service with an attached genre label which is in fact inconsistent with the mainstream notion of what that genre represents, these streaming services must be able to detect that the uploaded song's genre does match with how that genre is defined for the music streaming site overall.

University of Pennsylvania, CIS 419/519 Course Project.

2. Methodology

2.1. Data Preprocessing

The first step in our solution was to obtain data in the form of music files and prepare it for analysis. At this point in our project, we were working with two genres: metal and classical. In order to obtain a variety of songs typical to each genre, we selected them from online Top 10 lists and downloaded the songs individually. However, the vast majority of music found online is encoded in the MP3 file format, which is compressed and can be difficult to parse.

In order to prepare our data for feature extraction, we performed two operations. First, we converted these MP3 files to the WAV format, which is uncompressed and is essentially a long list of amplitudes that occur at a given rate within a song. Second, we took these stereo WAV files and flattened them to mono. This greatly simplified our data by reducing the number of audio streams from two to one, without losing any of the music's quality. Both of these steps were completed using Audacity, a free piece of audio processing software.

2.2. Feature Extraction

Upon completion of our data preprocessing, we had a large array of amplitude values, but in order to create accurate classifiers that ran in a reasonable time we needed useful features. After some research, we discovered Mel-Frequency Cepstral Coefficients (MFCCs), which represent the power spectrum of a sound. These are commonly used in speech recognition systems. (Ganchev et. al, 2005) They have also been shown to measure the timbre (sound quality) of a sound. For this reason, we decided to try and apply them to a different application - music genre classification. MFCCs are performed by the following operations: 1.) Take the Fourier transform of a signal. 2.) Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows. 3.) Take the logs of the powers at each of the mel frequencies. 4.) Take the discrete cosine transform of the list of mel log powers, as if it were a signal. 5.) The MFCCs are the amplitudes of the resulting spectrum. (Min Xu, 2004) Fortunately, there exists a python library, `python_speech_features`, that creates MFCCs. We were able to convert our song from a large array of amplitude values into MFCCs to use as a feature vector for our classifiers.

We also came up with four more features to extract from the amplitude data of a specific song. To calculate these features, we first created a new numpy array of the amplitude data, where the ith amplitude value in the new array equals the absolute value of the ith amplitude value in the original array. Given this array of absolute values of the amplitude data, we then immediately generated three more features for each song: the mean, standard deviation, and variance of this new absolute value array. The mean will be a good representation of how loud the song is, while the standard deviation and variance will represent how consistent the sound of the song is (i.e. whether the song is very smooth and mellow, or if the song has very loud choruses and very quiet verses). For our fourth feature, we split the absolute value amplitude data into 100 segments of equal size, calculated the mean values for each of these 100 segments, and then calculated the standard deviation of these 100 calculated means. This metric gives us an even better, more precise idea of whether a song has very loud choruses and very quiet verses, or if the song remains at relatively the same volume throughout the song.

2.3. Algorithms

We are currently examining two different classification algorithms and approaches to go along with our two sets of features. Our first classifier uses only the MFCC features for its classification. We compute the MFCC matrix for each song, flatten it into a single list, and creates the feature matrix using these flattened lists as the rows. We then feed this feature matrix, along with the array of labels corresponding to each row, into the `AdaBoostClassifier` provided by `sklearn`.

For our second classifier, we use a Support Vector Machine (SVC) implementation provided by `sklearn`, based solely off of the four features calculated using the absolute value amplitude data. This was to give us a good idea of how strong these four features will be when used together in classifying a song's genre.

3. Results

To analyze our AdaBoost classifier, we ran 10 iterations of 5-fold cross-validation on our dataset of 10 metal and 10 classical songs using `sklearn`'s `cross_val_score` function. We obtained a mean score of 0.8933 and a standard deviation of 0.148174.

For our second classifier, based solely off the four features extracted from the absolute value amplitude data, we found that after tuning the regularization parameter C and running cross-fold validation, the SVC implementation returned an accuracy of 52 percent on average, meaning that clearly these four features by themselves are not a good indicator of genre. One possibility for this error is that the feature representing the overall mean of all the amplitude data may contain significant noise, as the volume of the different songs is not standardized (meaning a classical song may be stored in WAV format as louder than a metal song).

4. To Do's

As there is a gray area in classifying many songs on whether they are genre 1 or genre 2, we can develop algorithms to return probabilities that a specific song belongs to each genre. Thus, instead of classifying exactly which genre a song belongs to, this algorithm could potentially pinpoint where on a spectrum between genre 1 and genre 2 does a certain song lie.

Further next steps include analyzing the effectiveness of other ML models on genre classification, as well as coming up with potentially new features to extract from the amplitude data of a song. We also need to determine the best way to combine our two sets of features, so that we can use a single classifier to analyze both the MFCCs and the amplitudes. Finally, we must find a way to standardize the volume of our songs stored in wav format, as this will prevent the feature representing the overall mean of the amplitude data from containing significant noise.

References

Ganchev, T., Fakotakis, N., and Kokkinakis, G.. Comparative evaluation of various MFCC implementations on the speaker verification task. *10th International Conference on Speech and Computer (SPECOM 2005)*,1:191-194, 2005.

Moffat, D., Ronan, D., Reiss, J. An Evaluation of Audio Feature Extraction Toolboxes. *Proc. of the 18th Int. Conference on Digital Audio Effects*, 2015.

Xu, Min, et. al. HMM-based audio keyword generation. *Advances in Multimedia Information Processing PCM 2004: 5th Pacific Rim Conference on Multimedia*, 2004.