Christiaan Dageforde
General Assembly, DSI

Music Genre Classification with Machine Learning

Overview

I am interested in the prospects of machine learning capabilities for learning based on analog signals, such as audio. My goal, initially, was to build a model that can detect samples in music. [Sampling is a technique in music composition wherein the composer takes a selection of pre-recorded sound or music and repurposes it as an element in their own work.] Such a task, however, would require greater familiarity with digital signal processing/audio feature extraction methods and would have shifted resources away from the modeling process. Instead, I shifted my focus to a simpler problem - can a statistical model identify musical genres based on the spectral features of a given song.

Goals

1. Use the Python library, Librosa, to extract spectral features from audio files in a way that they can be used in a machine learning model
2. Build a model that can accurately predict the musical genre of an audio signal, given a set of spectral features as a Numpy array

Data

For this project, I sourced the GTZAN audio dataset. The dataset consists of 1,000 audio files, each of which is 30 seconds long. The audio files represent ten distinct genres, with 100 files per genre. All tracks are 22040 Mono 16-bit audio files in .wav format. I converted these audio files into images of their spectrograms and employed a Python package called Librosa to perform audio feature extraction. The features I extracted include Mel-Cepstral coefficients (a snapshot of the spectral envelope; here I've taken 20 such "snapshots"), chromagram (a measure of absolute frequency or pitch), spectral centroid (the "center of mass" for a sound), spectral bandwidth, spectral rolloff (a measure of spectral energy beneath a given threshold), and zero crossing rate (the rate of change in amplitude or “loudness”). Each of these features was translated into a Numpy array and stored in a .csv for use in the modeling process.

Modeling

I began by building a neural network to handle the classification task. The final model was designed with a dense input layer with 256 nodes, one dense hidden layer with 128 nodes, a second hidden layer with 64 nodes, and an output layer with 10 nodes (one for each of our classification labels). The model then fit to run 250 epochs with a batch size of 256. I found that it was overfit even after adjusting the topology of the network and regularizing. The “black box” qualities of a neural network made it difficult to further examine the ways in which my model was generating predictions. As a way of validating the results that were being returned by my neural network, I built a random forest classifier and also an Extra Trees classifier to diminish the effect of correlation between observations. I gridsearched over both of these models in order to find an optimal configuration of tuning parameters. The final random forest model was tuned to ‘None’ maximum depth, ‘3’ minimum sample split, and 70 estimators. The optimal parameter grid for the extra trees model consisted of a ‘None’ value for maximum depth, ‘4’ minimum sample split, and 50 estimators.

Findings

I evaluated these three models on the basis of precision (the number of predicted positive values), recall (the number of *correctly* predicted positive values), and F1 score (the harmonic average of precision and recall).

BEST GROUPINGS

Model	**Label**	**Precision**	**Recall**	**F1**
Neural Network	Classical [1]	.93	1	.96
Random Forest Classifier	Classical [1]	.93	.1	.96
Extra Trees Classifier	Classical [1]	.76	.89	.82

WORST GROUPINGS

Model	Label	Precision	Recall	F1
Neural Network	Hip-Hop [9]	.34	.48	.40
Random Forest Classifier	Rock [9]	.44	.33	.38
Extra Trees Classifier	Rock [9]	.39	.5	.44

As can be observed in the tables above, all three models performed best when classifying classical music. My neural network performed worst when trying to classify hip-hop, while the random forest and extra trees classifiers performed worst when trying to classify rock music. To give an estimated guess (informed by domain knowledge) as to why these labels might be easier or more difficult to predict, I would venture that the mixing and mastering techniques for genres in these groups plays a factor. Rock and hip-hop music tends to be highly compressed in the mastering process and therefore their spectrogram features would be similarly nondescript. [Think “walls of sound,” with broad, dense frequency ranges and little dynamic change in amplitude.] Classical music, on the other hand, tends to be mixed in such a way that each instrument in the ensemble is well-represented in the overall texture and color of the recording. This results in a more unique spectrogram image.

Additionally, I was able to calculate feature importances for the Random Forest and Extra Trees models. Out of our 25 features, the top five most important ones in Random Forest were chromagram, mfcc4, mfcc1, spectral bandwidth, and mfcc9. In extra trees, those features were chromagram, mfcc1, mfcc4, spectral centroid, and spectral bandwidth.

Future Steps

One easily identifiable way to improve this project would be to collect more data. All things considered, a dataset of 1000 tracks is relatively small. Furthermore, given that each track is only 30 seconds long, we are missing out on a great deal of structural information that could be helpful in classifying genre. I could combat this problem by having full-length tracks in my dataset. Spending additional time tuning the tree models and experimenting with different neural network topologies could also improve results.

In general, this project can provide the building blocks for more complex projects. Genre classification factors heavily into recommender systems or building tools for musical instrument or artist identification.