Music Key Prediction – An Approach Using Machine Learning Algorithms to Predict Major Keys in Popular Music

By: Isaac Smith

Motivation

Music, at its most abstract level, is defined by its notes, tempo, and by extension the key signature. In my own life as a singer, knowing the key signature is perhaps the most important information to have when it comes to singing a piece. It narrows the range of usable notes, and defines harmonies and a foundation for improvisation, for instance knowing a song is in the key of "A Major" tells us that our seven primary notes, from which chords and harmonies are built from the melody, are A, B, C#, D, E, F#, and G#. My goal with this project is to produce an algorithm that can take as an input a piece of music, or a section of a song, and output its determination of the key signature (or key signatures if it changes during the song). In particular, I want to make an algorithm that can predict the key at least as accurately as a person with perfect pitch.

As it is, there are only a handful of programs that will listen to a recording and return key signature information. The most prevalent software that exist for this purpose is targeted at DJ's. DJ's, as a consequence of often mixing songs together, tend to be looking for songs with the same keys so that those transitions are as seamless as possible. But while the programs that exist for this purpose can be very effective, they are often expensive or are platform-limited applications. I believe that making a highly accurate and versatile key-detection software is itself fairly ambitious, but it can be scaled up to determine more information, such as tempo or genre.

Approaches

My approach to solving this problem took several forms. The first proof-of-concept was a simple linear regression model to show that it was possible to differentiate between two keys, C and C#. With that as a starting point, I then implemented a Support Vector Machine to allow for multi-class classification. I trained my model using a set of 12-dimensional arrays that corresponded to normalized note occurrences for each of the 12 notes (A, A#, B, C, C#, D, D#, E, F, F#, G, and G#) in 10-second song chunks. Thus far I have trained my algorithm on 23 popular songs in six of the keys (A, A#, B, C, C#, and D), for a total of nearly 3,500 labeled features. In order to get these 12-dimensional features for each song chunk, I first split the songs up, then applied a one-dimensional Fourier transform to the raw waveform sections associated with a song chunk, and binned the note frequencies that were in the range of A1 (27 Hz) to A8 (3520 Hz). Frequencies outside of this range were discarded, since the majority of human/instrument produced frequencies would not lie outside the range of 27-3500 Hz.

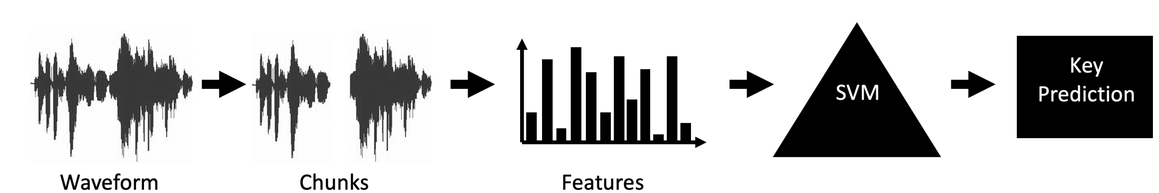


Figure 1: A simplified diagram showing the general workflow of the project. Starting with a raw waveform, it's chunked, and then the notes represented are binned into normalized abundances pet note. These are the features that train the SVM. When predicting music key, the same preparation work is done, and the SVM predicts on unlabeled features.

Challenges

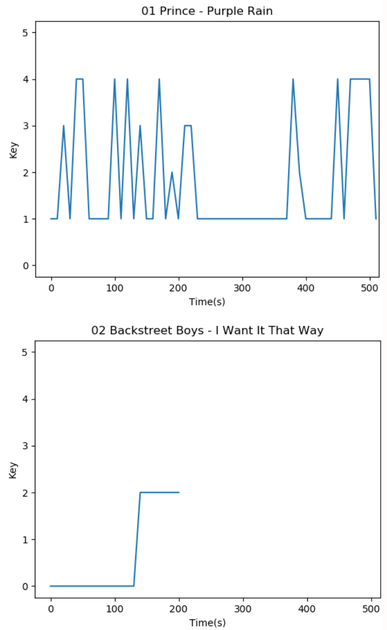
There are several challenges associated with solving this problem. One of the first ones I encountered was just collecting the data. The best method I could manage was to manually create playlists on YouTube with every desired song, and then use the command line program, youtube-dl, to download each playlist as audio files and convert each of them to the .wav file format. This was mostly a challenge just for the manual labor involved and is the reason why my model is only trained on 6 of the 12 keys at the moment.

Another challenge is the evaluation of the algorithm versus someone with perfect pitch. For this though, I am fortunate to know several people with it, and paid one of them to annotate 96 songs with me, 8 in each of the 12 major musical keys.

Results and Error Analysis

My algorithm currently assigns a key to a song based on which key the song "spends most of its time in", or the key that is most common among all of the chunks for a song. Using this assignment method, out of the 48 test songs for the **6 tested keys**, my algorithm correctly assigned keys to 42 of them, for **88% accuracy**, compared to my friend with perfect pitch who correctly assigned 43 of them, for an accuracy of 90%.

In two example plots generated by my algorithm, to the right, which have key prediction plotted over the course of a song, we can visually see how my algorithm makes its assignments. For Prince's "Purple Rain", my algorithm assigned the key associated with index 1, or A#, to the majority of the song, and so correctly assigned the key of A# Major to the song. In "I Want It That Way", by the Backstreet Boys, the algorithm incorrectly assigned the key associated with index 0, or A, to a song that my source dataset lists as being in B. An interesting note about this song in particular, is that the person who I had annotate each song said that it "starts in A and moves to B", which my algorithm agrees with and shows graphically. One additionally interesting bit is that of the 6 songs my algorithm incorrectly assigned, 3 were also incorrectly assigned by the person with perfect pitch, and 2 were in the same way.



Comments and Next Steps

The current implementation has exceptionally promising and exciting results. Music is an incredibly complex and variable form of art, and it's a testament to my feature extraction algorithm that an SVM is able to perform so well. But there is still room for improvement. In most of the songs, there the 10 second chunks give a good enough resolution to clearly show key changes, but it's clear in a song like "Purple Rain" that my algorithm is somewhat confused and frequently spikes away from the root key of A# major to the keys of B, C, and C#. 10 seconds was chosen somewhat arbitrarily, so I will need to do some experimenting with other chunk sizes to see how that affects key prediction.

With time as a limiting factor, it wasn't possible to incorporate all 12 major keys into the model so far, but that is perhaps the most-clear next step, and will allow me to more completely evaluate the overall effectiveness of my algorithm.