# Creating an Encoding Independent Music Genre Classifier

John-Ashton Allen, Sam Oluwalana

December 16, 2011

## Abstract

The field of machine learning is characterized by complex problems that are solved by algorithms run over data models. While time can be spent making gains in the algorithmic area, many speculate that with enough data many problems can be solved. This study utilizes that theory to develop an encoding independent music classifier. In particular, given sets $S_1, \ldots, S_n$, where each $S_i = \{S_{i_1}, \ldots, S_{i_m}\}$ is a set of labeled data corresponding to music encoding $i$ of files in $m$ genres. The problem is represented as $n$ classification problems. In this study it was found that independent of file encoding, an accuracy in the high 70's could be achieved (an average of 76.93) in our case. However the average is much higher (in the 80%s) for genres with different sounding music (i.e. Rock and Hip-Hop as opposed to Hip-Hop and Rap)

## Set Up

To set up our problem of classification we first collected a corpus of diverse songs with genre tags. We hypothesized that a normal human being can classify the genre of a song based on a clip of the song, so we first converted the collection of songs which totaled 120 GB of data to a standard encoding using the mp3 format with a bit rate of 32KB/s and a frequency of 44100 HZ. Using this as our base we then converted the entire corpus of data to clips of size 20 seconds and 10 seconds in the formats mp3, m4a, and wav. In each of these encodings we made sure to strip all metadata that could be used to help the classifiers learn decision boundaries based on data besides the encoded song clips. With this data set we moved into developing the classifiers.

We then wrote code that took these encodings and created feature vectors out of the raw bytes of the files. This resulted in different sized feature vectors for each of the encodings. We chose to use block sizes of data of 1 byte, 2 bytes, 4 bytes, and 8 bytes. We chose these sizes in order to get an encoding independent method of analyzing data. The different size block sizes of the data gave different feature vector sizes, with the 1 byte block size giving the largest feature vector size, especially on the lossless encoding, wav. The 8 byte block size gave much smaller feature vector sizesand faster run time on the algorithms but with a higher loss of accuracy. With this we focused our analysis mainly using the 32 bit block size of raw data for our feature vectors to balance the trade-off of space complexity and computational complexity.

## Results & Reflection

Our study started by training a logistic classifier on two of our genres to test the feasibility of our project. Choosing many different pairs of genres we found that accuracy using logistic regression was fairly consistent in the range of 85%-95% range, much higher than random guessing. However when we added another genre this accuracy dropped to the 77%-83% range. Adding an additional genre brought us to the 73%-80% range using 4 genres and approximately 5GB of data. These tests were run on the mp3, m4a, and wav datasets of 20 second clips. We found that using the smaller block sizes yielded computations that took several hours and offered very little improvement over the native 32 bit chunks supported by our machine. Accuracy dropped significantly, however, between the 32 bit chunk sizes and the 64 bit chunk sizes in our test. This led us to investigate the cause of our decreased accuracy when adding additional genres.

Investing further we found that the for set of $i$ genres, our best performance was bound by the weakeast classifier for $i - 1$ genres. To see this more clearly, we present an example. Jazz (possibly because it influences many genres) did bad pairwise with each genre. Pairwise, Latin and Jazz had a classifier that was correct on average 85% of the time, while a classifier on Latin, Dubstep, and Gangta performed at about 86% on average. When compared to the 3 way classifier on Latin, Dubstep, and Jazz, which perfomed at about 76% on average, the effects are clear. Furthermore results are dropped dramatically when attempting to distinguish between Hip-Hop and Rap, as the distinction is not clear for human beings.

## Future & Improvements

As was previously speculated, we feel that our tests were far from conclusive. In order to produce more accurate results we will begin by taking steps to better encoding the data that is used for the algorithms. We found that the encodings were really low quality and this may have had an affect in the overall performance of our algorithms. We also utilizing Support Vector Machines to get a better, non-linear, decision boundary for our data set in order to improve boundaries between hard to distinguish genres. Memory usage also became a problem during our tests, as such we would search for more efficient implementations and hope to increase the amount of data that we can analyze through different methods of compression. We found that analyzing clips of different sizes had very little impact in overall performance so in order to reduce memory usage and increase number of training examples future studies should favor smaller clip sizes.

Our study may also benefit from using different features that are encoding independent, such as using the beats per minute, the Fourier transform of each song to compare each of the songs component frequencies. This will require a more sophisticated method of storage that will render the underlying bytes of the file more meaningless. Using these more sophisticated features we feel that we can get drastic improvements over the 79% accuracy using just the raw data of the encoded files. Implementation of using the Fourier transform of each song will require reading the encoded files into its component waves and encoding those waves as a vector over the songs duration at a particular sampling frequency. These improvements in algorithmic design would bolster the amount of work that we get out of a machines memory and processing power and as such would be the next logical choices for improvements in the future. However, when taking this approach

with these features the algorithm loses its format independency.