**Music and AI: Music Genre Classification using Machine Learning Methods**

Mitchell Peterson and John Mohoang

Prof. Susan Fox

Comp 484 - Intro to Artificial Intelligence

Macalester - Fall 2016

12/09/2016

**Final Paper**

Abstract

Many music classification systems today recognize sounds by finding matches in pre-existing databases. Shazam, Siri and Soundhound are all examples of this. While this is important, we wanted to find a way to classify music that isn’t in the database already. This is useful as it will give people more knowledge about the song so that they can make a more informed search of that song in future. In this project, we extracted frequency, the melody and average beats per minute (bpm) from 1000 songs from the Tzanetakis dataset. We analyzed our data using four machine learning algorithms, Decision Tree, Support Vector Machines, Linear Discriminant Analysis and Quadratic Discriminant Analysis, which we used to classify the songs by genre. We found out that Support Vector Machines (SVM) worked best with a classification accuracy of ~29%.

Introduction and Background

Our motivation for this project arises from an interest in the intersection of Music and Computer Science, specifically Artificial Intelligence (AI) and Machine Learning. Music is such a subjective art form by nature. We thought it would be really interesting to apply something “completely objective” (machine learning) to music and see what kind of trends can be exposed and what sort of patterns emerge.

Most music recognitions systems today work by collecting samples of sound and comparing them to a pre-established database built up of recognized and analyzed songs.[[1]](#footnote-0) A good example of this is Shazam, a popular music identification application which recognizes songs through audio fingerprinting. Youtube and other services use similar methods of music identification as well. When one gives an audio recognition system program a short sample of a song, it finds a match by through running some algorithms to find its audio fingerprint and then match it with one that already exists in its database. Due to the basis on which these music identification methods work, there is an inherent problem of not providing any information if the song does not already exist in the database. This poses a problem and may render an audio recognition system useless, especially to people who are interested in expanding their musical horizons beyond the scope of well known songs. As a result, in this project our goal is to build a machine learning model that can predict the genre of a song that isn’t already in the database - a complex and interesting task. The success of this project could help us better understand the niche categorization of musical genre, and help with thinking about how humans classify genre objectively, rather than subjectively like music is often thought to be. This project is a starting point for further work in music classification and music feature extraction.

Related Works (Literature)

We decided to do some further research to familiarize ourselves with work already done around this topic. We explored four papers that were all tangential to our interest area, and gleaned some important information for how to proceed with our project. The most important aspects were finding out how the researchers extracted data from each song, how they represented those extracted features, and what they did to process and analyze their results.

In the paper “Learning Features from Music Audio with Deep Belief Network” (2010), Hamel and Eck used the concepts of deep belief networks and discrete fourier transformations to process audio, then combined the “activations” of the network with a support vector machine to determine the genre of a specific type of music - similar in idea with what we wanted to do, but different in structure. Their method observed an accuracy of 84.3 percent on the Tzanetakis dataset (1000 songs split up into 10 folders, where each folder represents a different genre). This is the same dataset we ended up using. Hamel and Eck compared different music classifying methods to see how each of them performs. It is important to fit a model that can best learn from the music information. Though Hamel and Eck's work was graduate level work, it can give us a basis of comparison for our project.

Another way of classifying music is the usage of multiple classification methods at once. In the paper “A Machine Learning Approach to Automatic Music Genre Classification.” (2008) Silla Jr et al. classified music using multiple feature vectors and a pattern recognition ensemble approach according to space and time decomposition schemes. They also used Naive Bayes, Decision Trees, K-Nearest Neighbors, Support Vector Machines MultiLayer Perceptron Neural Nets to analyze the extracted data. The results from each of the classification methods were aggregated to provide a better classification result. This was inspiring for us as we wanted to potentially follow a similar approach. Looking at the different classification methods used here informed our choice of algorithms to use.

We then found a paper that dives deeper into the application of extracting useful information about a song. In their work "Automatic Identification of Music Performers with Learning Ensembles."(2005) Stamatatos and Widmer described using similar machine learning methods to identify the most likely pianist in a new musical piece given a set of data consisting of a single piece of music performed by different pianists. Rather than genre, they tried to get the algorithms to recognize patterns in individual player’s music, a step further than most music classification research goes. This project gave us ideas of what areas we would like to take the project in the future, and represented an interesting way to approach the problem. A likely performer is a very important cue to to trying to identify a song, and could be used as “a piece of the puzzle” for lack of a better word, when running algorithms to identify music.

Finally, in the paper "Music Genre Classification of Audio Signals", Tzanetakis and Cook explore the problem using multiple music classification methods , and even build their own dataset for analysis. Tzanetakis and Cook propose a model that to classify music genre based on timbral texture, rhythmic content and the pitch content of each song. Their model resulted in a 61% classification accuracy, which is in par with human classification accuracy. Tzanetakis and Cook's work is highly parallel to what our project aims to perform. The major difference between Tzanetakis and Cook's work and ours is the methods used. We used a combination of frequencies, estimated melodic pitch and beats per minute to help identify classify music into different genres. I will be interesting to compare the accuracy of Tzanetakis and Cook's model to those that we make.

Data

The dataset we used was collected by George Tzanetakis (2002) for his music classification thesis project. The dataset was collected between the year 2000 and 2001 from a variety of sources including CDs, radio and microphone recordings. A variation in the sources of the dataset induced some background noise to some songs which led to a variation in sound quality across the dataset. This is advantageous because it helps our classification models work with noisy data which is reflective to the real world scenarios they will be used in. The dataset itself consists of 1000 songs, each 30 seconds long, originally in .au format. We had to run a batch conversion to change them all to .wav format, which we are more comfortable using. This conversion allowed us to work with the python module of our choosing. The songs in the dataset are divided equally into 10 genres; blues, classical, country, disco, hiphop, jazz, metal, pop, reggae and rock, with 100 examples of each. While the songs are all given specific genres, the styles within those genres vary, so for example in Jazz there was a mix of ballads, hard bop, bebop, and other styles.

It took us a very long time to find an open source python module that would accomplish what we wanted. There are many custom built libraries out there meant for audio feature extraction, but when we installed them, there seemed to always be some incompatibility. This is a common trend when downloading repositories from Github. Finally, we were able to settle on one after trying about 4 or 5 others. To extract music features from the songs in our dataset we used a module called Aubio.[[2]](#footnote-1) The module was developed by Paul Brossier as part of his phD thesis (Brossier, 2006). Aubio has a host of various extraction and modification capabilities, but we narrowed our selection down to three main features. For each song, we extracted the sound frequencies used to build a continuous sound spectrogram, a generalized melody, and the estimated average beats per minute (bpm). While Aubio has many other options for feature extraction, we chose these three features because we felt they most accurately would represent the differences among music genre genre, and we hoped our algorithms would be able to recognize patterns among them. To extract the data, we built a python script that compiled all three feature extraction methods into one big method. We had to modify the Aubio code to allow us to extract the data from multiple songs at once, all within a given folder. This allowed us to run the program on all 10 genres instead of running it 1000 times. Once the data was extracted, we wrote a method to output the data to a .csv file.

Because continuous data can be hard to work with (especially in the context of music) we needed to figure out a way to condense our data. Without doing this, each song would have almost 600,000 data points. For the spectrogram only, with 30 seconds of music split up into 2586 frames based on the sample rate of the music, each frame represented the amplitudes of 257frequency bands**.**  If we add 2586 pitches (one for each frame) for the melody analysis and one for bpm for each song, then multiply that by 1000 songs, we would have significantly more data. Calculated out, that’s somewhere around half a billion data points. This is unnecessary, bloated, and will cause or algorithms to overfit the models to the training dataset. As a result, we minimized our dataset to the most informative condensed pieces.

To condense our data, we first compressed the frequency data of each song. Each frame, which is a fraction of a second long, is represented by an array of numbers each indicating the intensity/amplitude of a particular frequency range at that time point. We calculated the average of these arrays to get an array of the average frequency for each second interval. We chose to average over intervals of one second because music does not have significant changes in frequency over small time intervals. For comparison among the frames, we treated each frequency array as a multidimensional vector and calculated the difference between each consecutive vector as the angle between the two vectors. The larger the angle, the more different the arrays are, and the smaller the angle, the less different the arrays are. At the end, for each second, we just have one number representing the overall change in frequency as compared to the previous second. In music whose frequencies are less consistent, like Jazz, we expect the angles to be higher unlike in more consistent genres like Rock and Pop. See Figure 1 for a comparison of a Jazz spectrogram to a Rock one. In the Jazz (left) you can visibly see the variation in the sound, while in rock the consistency of the music is much more apparent. We hoped that our algorithms would be able to pick up on these qualities.

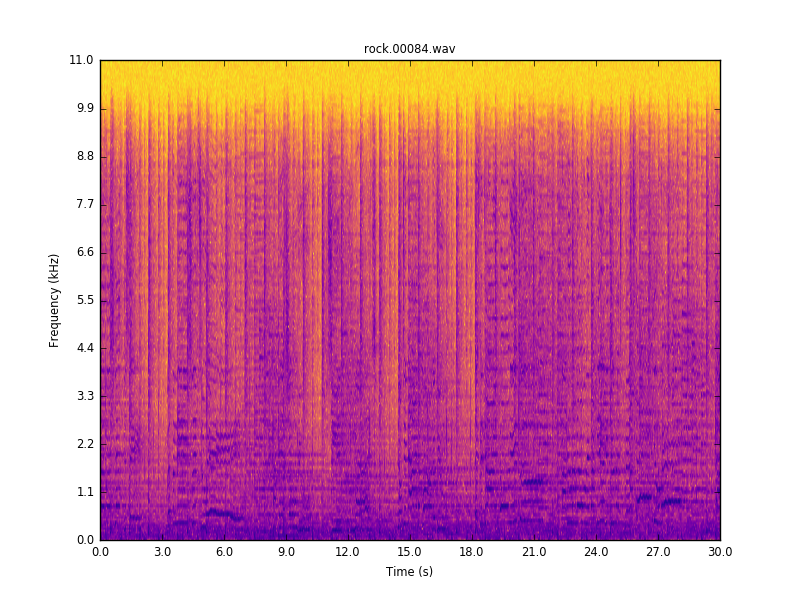
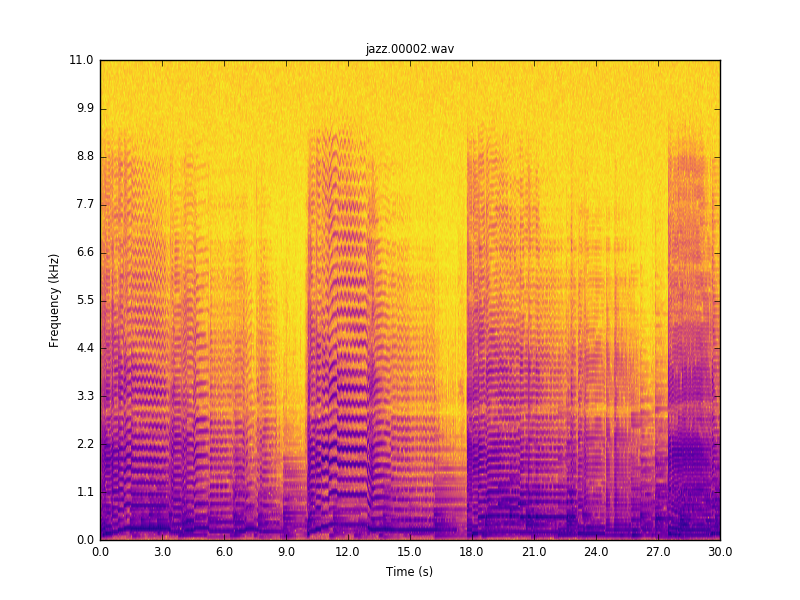


Figure 1. - Comparison of spectrograms between Jazz (left) and Rock (right). Frequency is mapped to the Y axis, while time is mapped to the X. When comparing the two, it is visually easy to see that Jazz has many more spectral changes as compared to rock.

For pitch, we used a more simple averaging method to condense our data. For each song, there were 2586 frames, but again because the melody does not change 80+ times over the course of a second, so we figured we could just average the pitched for each second. This gave us 30 concise numbers for pitch. The pitch is measured in Midi, which is simply a scale of 0 to 127 ( because 127 is the maximum positive value of an 8-bit signed integer). Each number is associated with a note starting at C-0 and going up to G-10.

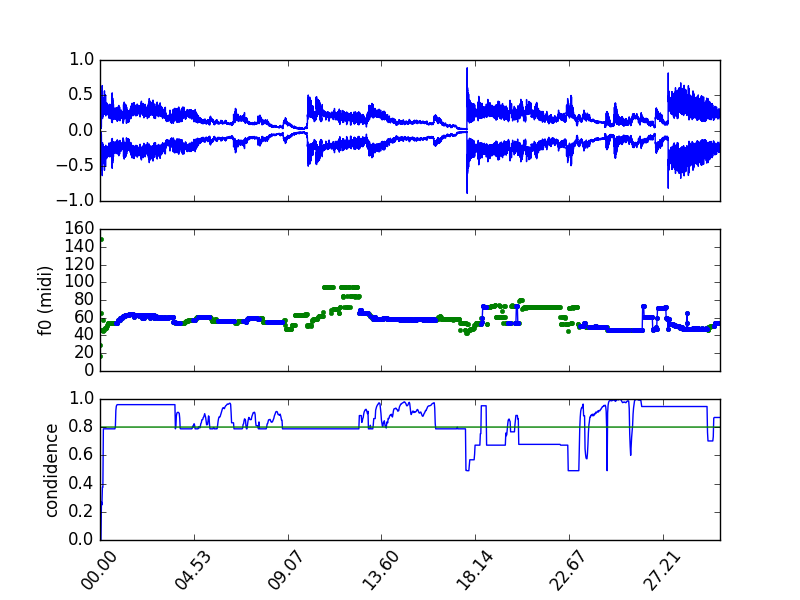


Figure 2. - This graph shows the pitch analysis for the same Jazz song used above. It shows a graph of the amplitude, a graph of the estimated Midi values, and the confidence, which overall lies around 80 percent.

For the BPM analysis, Aubio simply outputs a general estimate of what the average BPM is estimated to be for each song, so no condensing was needed there. Once all the data was cleaned up, the script would output the data for each song to the .csv file organized by genre.

Methods and Results

We built four machine learning classification models to predict the genres of songs using the R programming language. The software made it easy to compare four classification methods. In order to test how well each model works, we first divided our data into two data sets; 80% was the training set while 20% was the testing set. Each of the sets had equal representation of the music genres so as to avoid skewing our models towards certain genres. In order, our procedure for modelling was a) building a model based on the training data then, b) using the model to classify each of the songs in the testing data, and lastly, c) comparing what the model predicted to what the actual genres of the songs actually were. To find a comparison between the model results and the actual genres, we calculated the percentage error of the model.

1. **Classification Tree**

The first model we built was a decision tree model. The first output for the model is shown in Figure 3. Based on Figure 3, there is most gain when placing the freqDiff-12 (angle difference of frequencies between 11 and 12 seconds) at the root of the tree. This means that most genres differ greatly based on that variable. While this seems arbitrary, this must have been a pattern in our data, and thus was observed by this learning algorithm.

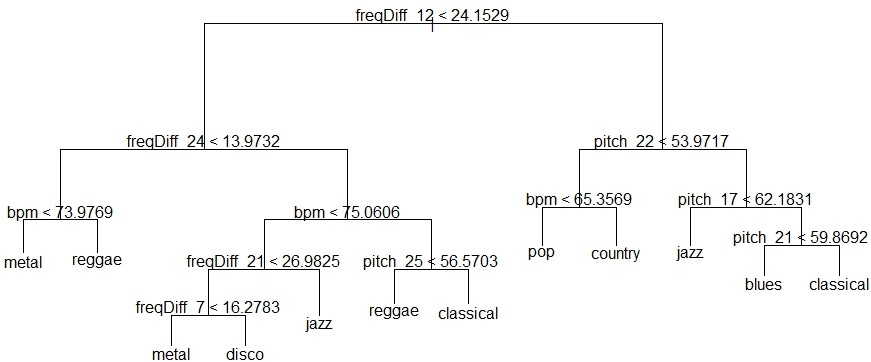


Figure 3. Decision Tree for the Training Data.

The model shown in Figure 3 gave an error rate of 81.05%. This means about 19 percent of the time the model will correctly identify the genre of a new piece of music. In order to try and reduce the error rate of the tree, we used cross validation of different sizes of trees to see what size of the tree resulted in best results. Figure 4 shows a plot of misclassifications against the best trees at each size.

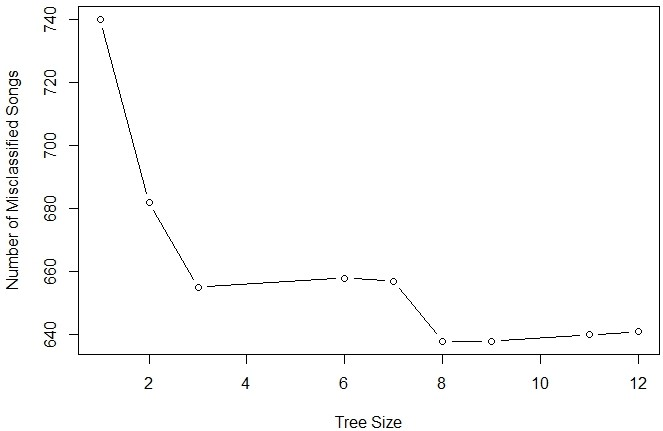


Figure 4. - a graph representing the most optimal tree size, which we used to prune our tree. The most optimal tree was at size 8.

Based on Figure 2, the best size of the tree that produces the least number of misclassification is at size 9. As such, we pruned our tree so that t is only of size 9. The resulting tree had an error rate of 78.95% which is a little lower than that of the original tree.

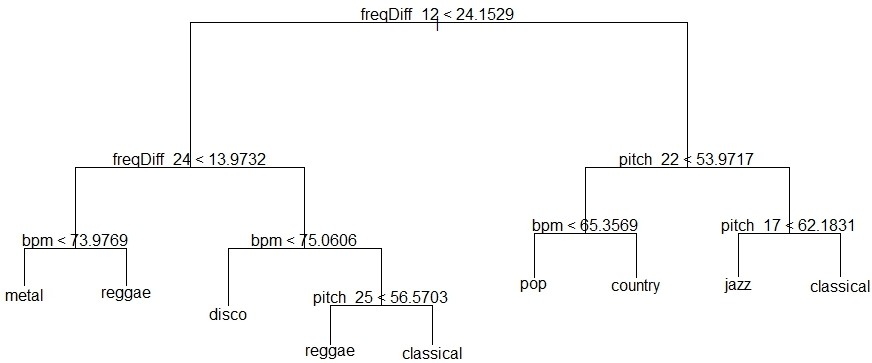


Figure 5. The Pruned Tree

1. **Support Vector Machines (SVM)**

Support vector machines use both classification and regression analysis to classify new data by using the optimal separating hyperplanes built using training data. (Gareth, 2013) We built an SVM model based on the songs in the testing data and then made tests on the training data. In comparison to what the the genres of the songs in the testing data actually were, the SVM model produced an error of 71.05%. This error rate is much much better that the one we got out of the tree model.

1. **Linear Discriminant Analysis (LDA)**

LDA creates classification by searching for a linear combination of predictors that best separate the data. It assumes that the observation of each category are drawn from a multivariable Gaussian distribution with a category specific mean vector and a covariance matrix that is common to all categories (Gareth, 2013). We built an LDA model help classify our music data into different genres. When we tested what the model produced in comparison to what the actual test data genres were, the model had a percentage error of 78.95%. The LDA model performed worse than the SVM model.

1. **Quadratic Discriminant Analysis (QDA)**

The QDA is very much like the LDA except in making a classification model, it assumes that each category has its own covariance matrix. As a result, QDA is a more flexible classifier in comparison to LDA though it has a higher variance (Gareth, 2013). Since we have a large enough dataset, variance was not a big concern. We therefore proceeded to make a model to predict music genres. When we tested what the model produced in comparison to what the actual test data genres were, the model had a percentage error of 84.74%. The percentage error rate our highest, thus making the QDA the worst model.

Discussion and Conclusion

To give a summary of our methods and results, Table 1 shows the all the models we created. The table also shows the percentage misclassification errors of each model.

**Table 1: Summary of the results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Error Rates of Classification models | | | | |
|  | Decision Tree | SVM | LDA | QDA |
| Percentage Misclassification Error | 78.95 | 71.05 | 78.95 | 84.74 |

Based on Table 1, it is obvious that the best model we built was the SVM model. We believe this is due to the overlapping nature of our data. Because there are not clear cut distinctions between genres, there tends to be a lot of overlap in the data points, causing a lot of confusion for the algorithms. This could also be what attributed to our error rates being so high. While ~30% accuracy isn’t very impressive at first glance, we took data that started at 600 million data points and condensed it by almost 10,000 percent. With this condensed data, that fact that we were able to get the algorithms to learn anything was impressive. As compared to the researchers we modeled our project after, we achieved approximately half the accuracy they did. In his thesis paper "Musical genre classification of audio signals.", Tzanetakis was only able to achieve 60 percent accuracy, and that was graduate level research, with hours of time dedicated to perfecting and optimizing his results. In “Learning Features from Music Audio with Deep Belief Network” (2010), Hamel and Eck were only able to achieve 84.3% accuracy. Additionally, we were only using three features to categorize our data, while the graduate level researchers used many more. The fact that we were at least able to achieve some learning in our algorithms made us proud and convinced us of the validity of the project.

Overall, this project gave us a hands on introduction to Machine learning and fueled our interest in the subject matter. We loved being able to fuse a hobby of ours (music) with this project and used it as an opportunity to explore something we weren’t very familiar with. It was a daunting task, but we are glad we were able to achieve at least some positive results, especially with no experience in signal processing.

Future Work

The features of music that we collected, frequencies, pitch, and bpm, seem to have a cap of 29% accuracy. In future, we would like to extract more features of the music that can be helpful in increasing our model accuracy. We would also like to explore the overlapping nature of information on between frequencies and pitch. This will be useful as it will improve our model predictions.

During our bpm calculations, we realized that Aubio did not do a very good job. We compared the estimated bpm to our hand calculated bpm and found the estimate to be significantly off. However, when we removed bpm from our dataset, all four algorithms performed worse. We determined that it was giving the algorithms something to work with, but would like to investigate the discrepancies further in the future and see if a more accurate bpm calculation can increase our accuracy.

In retrospect, if we had more time we would have loved to do some optimization. What other feature extraction techniques could we have used? What is the optimal time to average frequencies over that yields the best results without overfitting? What optimizations to our algorithms might we be able to make, and what other machine learning methods might we have attempted to achieve the most significant/accurate results? There are many things we would have liked to continue with, but we were constrained by time.

Bibliography

Brossier, Paul M. "Automatic annotation of musical audio for interactive applications". Diss. Queen Mary, University of London, 2006.

Hamel, Philippe, and Douglas Eck. "Learning Features from Music Audio with Deep Belief Network." International Society for Music Information Retrieval. (2010) .Web.

James, Gareth, et al. *An introduction to statistical learning*. Vol. 6. New York: springer, 2013.

Jr., Carlos N. Silla, Alessandro Koerich L., and Celso Kaestner A. A. "A Machine Learning Approach to Automatic Music Genre Classification." J. Braz. Comp. Soc. Journal of the Brazilian Computer Society 14.3 (2008): 7-18. Web.

Stamatatos, Efstathios, and Gerhard Widmer. "Automatic Identification of Music Performers with Learning Ensembles." *Artificial Intelligence* 165.1 (2005): 37-56. Web.

Tzanetakis, George, and Perry Cook. "Musical genre classification of audio signals." *IEEE Transactions on speech and audio processing* 10.5 (2002): 293-302.

Appendix

With this paper, we are submitting the following:

1. A file with the cleaned data, split into training and testing sets.
2. Python Code we used to extract and clean the data
3. Link to the R file of the Analysis

1. <https://www.google.com/patents/US6990453> A paper written by the Shazam founder. It is about audio fingerprinting. [↑](#footnote-ref-0)
2. <https://github.com/aubio/aubio> [↑](#footnote-ref-1)