Music Classification by Composer

Janice Lan
janlan@stanford.edu

Armon Saied
armons@stanford.edu

CS 229, Andrew Ng
December 14, 2012

Abstract

Music classification by a computer has been an interesting subject of machine learning research. We implemented a variety of different classification algorithms with the goal of identifying 9 classical composers from various time periods. To improve the performance of our learning models and optimize our feature selection we executed some diagnostic tests, including the leave-one-out cross validation method. Finally we analyzed and assessed the performance of our learning algorithms and remarked on some possible areas for further improvement.

1 Introduction

The ability to classify musical compositions by composer is useful for several applications, such as organizing musical databases, learning preferences and patterns of music listeners, and building music recommendation systems. Many musicians can usually differentiate compositions of different composers even if they have never heard those particular pieces before. Because of the way humans perceive music, people focus on the tone and color of the song, the emotional responses generated from listening to the music, general characteristics such as the level of dissonance, recurrent features such as trills or arpeggios, and other overarching qualities to distinguish the "style" of a composer. When listening to music, humans care much less about theoretical rules and numeric patterns of the notes. However, many of the overarching qualities are difficult to quantify, and thus unlikely to be used by computers. In this project, we attempt to differentiate various styles of music (in this case, composers) by applying machine learning to numeric data extracted from patterns in music.

Previously, Lebar, Chang, and Yu worked on a similar project where they attempted to classify music by composers. However, they used scores instead of audio files as their source of data [1]. They used Naïve Bayes, SVM, LDA, and k-NN. There have also been numerous projects of classifying music from audio files, though most classify by genre and not by composer. For instance, Haggblade, Hong, and Kao used KL Divergence, k-NN, k-means, SVM, and neural networks to classify songs into classical, jazz, metal, and pop [2].

Building off of previous ideas, we hoped to explore different techniques for successfully classifying music by composer. Therefore, our primary challenges were to choose effective machine learning algorithms to implement and to ascertain the most appropriate quantifiable musical features.

2 Data

To limit the scope of this project to a domain less broad than "music," we restricted our music data to solo piano compositions. We also limited the file format to MIDI files, in order to keep consistent our methods of parsing music for numeric data. MIDI files were obtained from several websites, mostly from www.piano-midi.de and www.classicalmidiconnection.com.

To select the musicians to classify, we decided to choose famous composers from different eras of classical music who composed large collections of piano compositions. We settled with nine

Composer	Lived	# Files
Bach, J.S.	1685-1750	136
Scarlatti	1685-1757	555
Mozart	1756-1791	56
Beethoven	1770-1827	78
Schubert	1797-1828	58
Chopin	1810-1849	102
Liszt	1811-1886	45
Debussy	1862-1918	52
Prokofiev	1891-1953	25

composers to analyze, ranging from the baroque era to the modern era.

We noted the relatively large amount of data for Scarlatti, especially because in a previous music classification study, a cross-validated Naïve Bayes model biased the classification towards composers with more data [1]. However, we later tested our learning algorithms with reduced Scarlatti data and found that there were no significant problems with the disparate amounts of data between composers.

3 Methodologies

3.1 Feature extraction

We used music21, a Python package, and miditoolbox, a Matlab package, to extract quantifiable features from the MIDI files. Initially, we started classifying solely using the mean and standard deviation of pitches, note durations, and inter-onset interval (time between attack-points of consecutive notes). Later on, we added pitch intervals, song tempo, pitch range, and the divergence from original key (the fraction of notes that do not belong in the major or minor scale of the original key). For most of these features we observed the mean, standard deviation, and the range. In addition we also analyzed the note sequence, transposed to C major/minor. This was used in the Bayesian Network classifier in an attempt to analyze the melodies, and to determine which composers they are the most characteristic of.

3.2 Preliminary Algorithms

We started out with classifying between 2 composers, Chopin and Bach, as a simplified test between a relatively distinct pair of composers. We used the original set of features as described above in three main learning algorithms: Naïve Bayes, perceptron learning algorithm (stochastic gradient descent), and logistic regression (stochastic gradient descent).

3.3 Support Vector Machine

We implemented the Support Vector Machine model as a binary classifier using "sequential minimal optimization." Later, implementing a "1 vs. all" scheme for classification, we extended the classifier to have multi-class capabilities in order to classify all 9 composers.

3.4 K-Nearest Neighbors

We implemented a simple K-Nearest Neighbors Algorithm using a Euclidean definition of the distance function. To improve the algorithms performance, we tested it for numerous k values, tried several different distance functions, and finally tried weighting the features in a manner that agreed with our feature analysis tests.

3.5 Bayesian Network

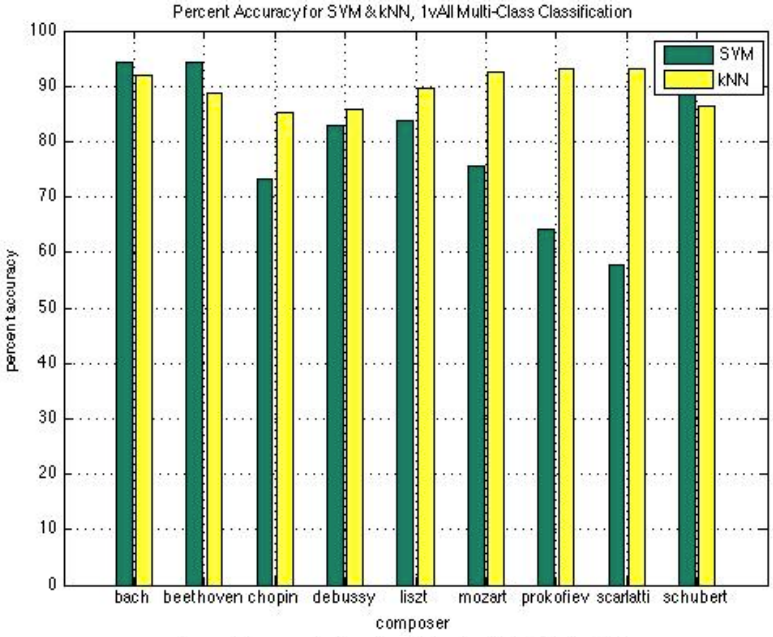
The Bayesian Network classifier is based off of the probability distribution of the next note given the current note and, optionally, previous note(s). Classification of a song was based on how likely it is to have its sequence of notes, i.e. the product of P(note i+1 | note i) for all notes in the composition.

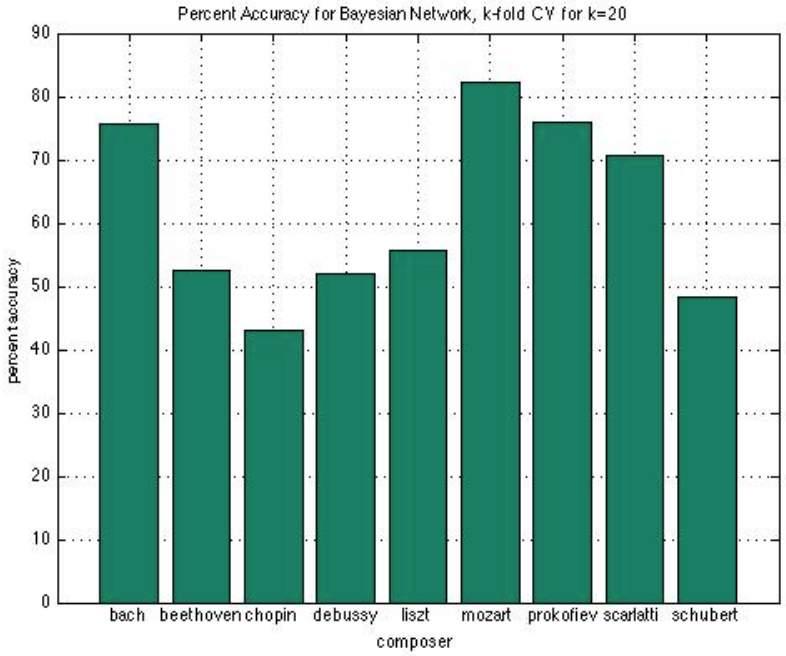
4 Results

The following are two graphs of our results for SVM, k-NN, and Bayesian Network. The first figure shows the percentage of songs that were correctly classified by the multi-class SVM and k-NN. The second figure similarly shows the percent accuracy for Bayesian Network, where a k-fold cross validation was also used.

4.1 Preliminary Algorithms

Our three preliminary algorithms were tested with the original training data. The Naïve Bayes reported a training error of 9%, the perceptron a training error of 21%, and the logistic regression a training error of 28%.





4.2 Support Vector Machine

The first implementation was very successful for most composers. Using binary classification, all of the composers, except for Prokofiev and Scarlatti were identified with less than 20% testing error. Using a "1 vs. All" multi-class implementation, we achieved the level of classification accuracy shown in the graph above. The following is a brief description of "1 vs. All": for each composer, mark its train data as class label positive. Run the SVM algorithm where the remaining composers are all marked class label negative. Then as long as the algorithm returns negative for the current training data, cycle the current composer into the subset of negative composers, classify, and then repeat the process.

4.3 K-Nearest Neighbors

The final results of this algorithm are shown in the graph above, for k =5. Again we used the "1 vs. All" scheme for multi-class classification. Our initial implementation of k-NN performed at a 78% total classifying accuracy for all of the composers. Adjusting the value of k yielded at most a 10% change in

testing error. Furthermore, using different distance functions did not show a consistent improvement in classification, so we settled on Euclidean distance: $d(q,p) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$.

4.4 Bayesian Network

The first attempt at a Bayesian Network classifier used P(next note | current note). This did not work well, as the training error was over 50% for all composers except for Bach, Mozart, and Schubert. It appears that the probability distribution of two consecutive notes was very characteristic of different composers. To improve on this, we adjusted the number of notes used in the conditional probability. Using two notes instead of one decreased the train error to 23%. Using k-fold cross validation, with k=20, to run this learning algorithm on test data we got the values shown in the graph above.

To see if using more notes would generate better models, we also tried using 3 notes as the conditional probability. The result of the k-fold cross validation was significantly worse, except for Scarlatti, which had a 93% accuracy. This shows that the faults in the learning algorithm are probably due to lack of data: there are 12^3 combinations of 3 notes, and we do not have enough notes and songs to give a reasonable probability distribution for all the triplets.

5 Analysis

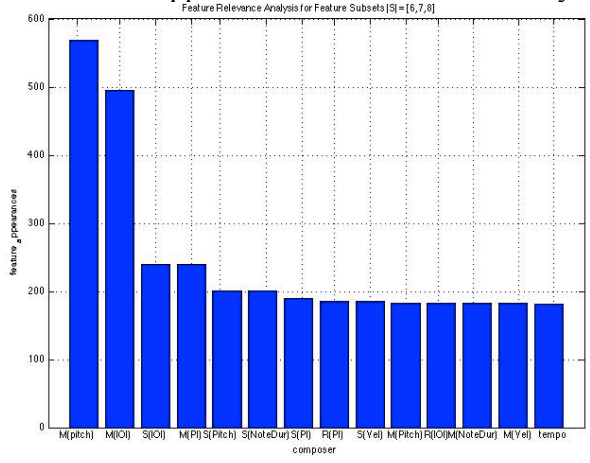
5.1 Algorithm Evaluation

We evaluated our algorithms with k-fold cross validation, using k = 20 because we did not have much data and did not want to leave much out of the training set. We experimented with adjustments of k. For the Bayesian Network classifier, k=10 gave the same results, except Debussy was 7% worse. With k=30, the results were the same except that Beethoven improved by 2% and Liszt improved by 4%. Moreover, we also conducted the same tests with reduced Scarlatti data to address concerns with the imbalance in data. When using only a randomly selected 150 compositions, there was almost no difference, except that Bach was a few percent less accurate. We can conclude that the exceptional amount of data for Scarlatti does not bias the classifications, and in fact, is beneficial for training the difference between Scarlatti and Bach.

For k-NN we achieved the highest classification accuracies for each composer by weighting the features. The weights were constants proportional to the values we determined for each feature's measured relevance discovered in the analysis described below.

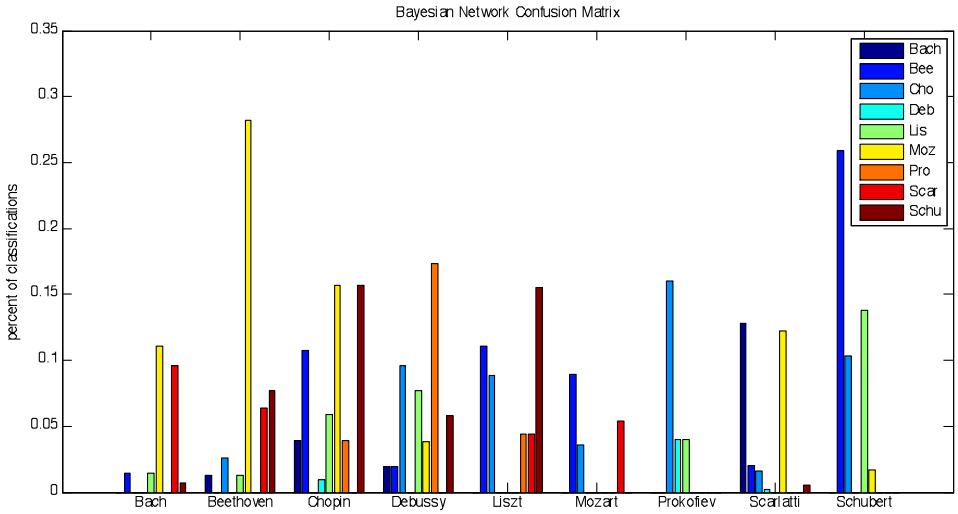
5.2 Feature Evaluation

We performed two distinct diagnostics in order to evaluate our features: feature relevance analysis and variance/bias analysis using learning curves. In order to assess the relevance of our chosen features, we wrote a program that executed our multi-class SVM for each possible n-sized subset chosen from the total features available. Given our time and computational limitations we only ran this program for n values of {6, 7, 8}, out of our total of 14 possible features. Below is a graph, showing the frequencies with which each feature appeared in a feature subset that yielded a testing error less than 5%.



5.3 Composer similarity

An interesting analysis of the classifiers' results is the similarity between each combination of composers. This can be seen in the confusion matrix of the results of k-fold cross validation with k=20. Each bar represents the percentage of wrong classifications; for instance, 28% of Beethoven compositions were classified as Mozart, but only 1% was mistakenly classified as Bach:



Higher bars represent a greater similarity between two composers. From this we can see that, as expected, composers of similar time periods are more easily mistaken for each other. Similar results can be seen when we remove one composer from the classification and see which composers improve in accuracy, and by how much. For instance, removing Bach increases the accuracy of Scarlatti by 12%, and removing Mozart increases the accuracy of Beethoven by 24%.

6 Further Work

Given more time, there are several other music features we would have liked to try. We would use chord progressions and note progressions, which are commonly used in musical analysis. One machine learning paper on musical style examined silence durations [3]. Furthermore, if we extracted features from songs in the wav file format we could look at song dynamics and tempo changes. Due to time constraints, we were unable to successfully implement the softmax regression; however, we believe that softmax is very appropriate for our project, and should be the next algorithm to test. In addition, we would attempt using Principal Component Analysis (PCA), neural networks, and discriminant analysis. Future extensions worthy of consideration are looking at songs with multiple instruments and trying additional genres of music.

7 References

[1] http://cs229.stanford.edu/proj2008/LebarChangYu-ClassifyingMusicalScoresByComposer.pdf
[2] http://cs229.stanford.edu/proj2011/HaggbladeHongKao-MusicGenreClassification.pdf
[3] http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4106034