Classifying Musical Scores by Composer:
A machine learning approach

Justin Lebar Gary Chang David Yu
{jlebar, gwchang, yud}@stanford.edu

November 14, 2008

Abstract

We apply various machine learning algorithms to classify musical scores in the `**kern` format by their composer. Our algorithms were able to distinguish some composers well, but had difficulty distinguishing between composers from similar time periods. We conclude that categorization problems among certain sets of composers likely have a higher inherent error rate than others.

1 Introduction

Understanding the features that demarcate musical genres and distinguish the works of various composers is important for a number of applications, including organizing musical databases and building music recommendation engines. Previous work has shown that classifying audio recordings of music is a difficult machine learning problem, not in the least because one must employ sophisticated audio processing techniques to extract features from a noisy audio waveform (see examples of such complexity at [1]).

We avoided the challenges of audio processing by taking a different approach: Instead of analyzing audio files, we used musical scores in the plain-text `**kern` format (full specification available at [2]).

Classifying music by its waveform remains the primal task in this field, but we hope that our analysis of features and algorithms for classifying scores may suggest which techniques for analyzing audio might be more effective and may hint at the inherent error associated with distinguishing between certain composers.

2 Process

We used the The Humdrum Project's library of `**kern`-encoded classical scores, available at `http://kern.humdrum.org`, as our source of training and testing data. `**kern` scores are essentially a lossless representation of the original printed music and include information about articulations, dynamics, and even note stem directions in addition to the notes themselves. For simplicity, our model ignores everything except pitches and their durations.

Each line in a `**kern` file lists one or more notes which begin simultaneously. We call each of these lines a "chord", and they form the basic token for our analysis of scores.

We run each score through a `**kern` parser written in Java before applying our machine learning algorithms to the scores. The parser transposes the score from its original key to the key of C major and generates features from each score's chords.

3 Data

Figure 1 lists the composers we analyzed. For each composer except Beethoven, we were only able to access significant numbers of pieces of one type—either pieces for a keyboard instrument or pieces for string quartets. To maintain this symmetry, we consider Beethoven's string quartets and piano sonatas as being written by two different composers. For pieces with multiple movements, we analyzed each movement as a separate piece.

Key	*Composer*	*Lived*	*Type*	*n*
JSB	Bach, J. S.	1685-1750	kbd	111
SCA	Scarlatti, D.	1685-1757	kbd	58
HDN	Haydn	1732-1809	sqt	212
MOZ	Mozart	1756-1791	sqt	82
LVBp	Beethoven (kbd)	1770-1827	kbd	78
LVBq	Beethoven (sqt)	1770-1827	sqt	70
CHO	Chopin	1810-1849	kbd	88
JOP	Joplin, Scott	1867-1917	kbd	46

Figure 1: Composers analyzed. Type is either "kbd" for keyboard music or "sqt" for string quartets. n is the number of pieces by each composer in our training set.

4 Methodologies

4.1 Naive Bayes

Naive Bayes showed little success when we considered two chords equal only if they contained the same pitches for the same durations, but we improved upon this result by relaxing the conditions under which two chords were considered equal. The best equality function we tested is *pitch-count*. Two chords are equal under *pitch-count* if they both contain the same number of notes of each pitch, ignoring octave and duration. That is, two chords each containing two Cs and a D in some octave have the same *pitch-count*, but a chord containing one C and two Ds has a different *pitch-count* than the other two.

Requiring that two chords have similar rhythms in order to be considered equal decreased the efficacy of our classifier dramatically. If we ignored the number of times a note appeared in a chord and considered a chord containing three Cs and a D to be equal to a chord containing one C and four Ds, our Naive Bayes implementation performed with only slightly less accuracy than *pitch-count*.

If we created n-chord tokens by grouping together the first n chords into one token, chords 2 through $n+1$ into a second token, etc., our model's train error fell to nearly zero for all composers and the testing error increased, even for $n = 2$. This indicates that even 2-chord tokens cause our algorithm to overfit the data.

4.2 Support Vector Machines

Our second approach to the problem was to apply a standard Support Vector Machine using the same features we had extracted for Naive Bayes. To this end, we used the libSVM library, available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

The accuracy of our SVM varied greatly depending on the kernel we used and the parameters we passed to the SVM. We tried three kernels:

1. linear: $k(u, v) = u^T v$
2. sigmoid: $k(u, v) = \tanh(\gamma u^T v + c_0)$
3. Gaussian radial basis function: $k(u, v) = e^{-\gamma|u-v|^2}$

and found that the last significantly outperformed the first two. As a result, we spent most of our efforts tuning and trying different features for the Gaussian radial basis function kernel.

4.3 Linear and Quadratic Discriminant Analysis

We performed LDA classification using the same *pitch-count* features as Naive Bayes and SVM since it was shown it give better accuracy. LDA assumes that the input features are distributed according to a multivariate Gaussian; if this assumption holds, then LDA should require fewer training samples to reach performance similar to NB and SVM. Although we didn't observe that our data was closely distributed according to a multivariate Gaussian, we hoped that our LDA might nevertheless perform well given a limited training set. To maintain symmetry with our other algorithms, we trained on 360 samples overall (45 per composer), and thus we rank-reduced our data by selecting the 360 largest principal components and performed LDA on those.

Supposing that the linearity of the classifiers in LDA might be a limitation to its performance, we also performed quadratic discriminant analysis on our data, using the same features as LDA. To maintain the nonsingularity of the class-specific covariance matrices, we classified on the 45 largest principal components.

4.4 K-Nearest Neighbors

Our implementations of Naive Bayes, SVM, LDA, and QDA include no metric for measuring the similarity of two tokens aside from strict equality. As a result, we could not train on full chords—instead, we reduced chords to their *pitch-count* and learned the frequency that a composer wrote chords with various *pitch-count* values. This reduction throws out a great deal of information—in particular, it ignores the notes' octaves and gives long and short notes equal weight. We devised a scheme for classifying scores using nearest-neighbor techniques which attempts to overcome these deficiencies.

Our algorithm works as follows: For every measure m in every score, we create the set N_m containing the k measures in other pieces which are most similar to m, as measured by some distance function d. To classify a score S, we compute for each measure $m \in S$ a function $C(m, N_m)$ of m's neighbors which classifies m as being most similar to one composer. We then take a majority vote of all the measures to classify S.

We tried a number of different parameters to this algorithm. In the end, we represented each measure as a vector where each element of the vector was a weighted sum of the notes sounding at a given pitch in that measure. Longer notes received greater weight, in proportion to their length. We found that taking octaves into account did improve the accuracy of our algorithm, as we'd hoped.

We found that the algorithm was not very sensitive to the distance function used, but we had best performance when using

$$d(x, y) = \left\| \frac{x}{\|x\|} - \frac{y}{\|y\|} \right\|$$

as opposed to $d(x, y) = \|x-y\|$ or $d(x, y) = \|x-y\|/(\|x\|+\|y\|)$. Our algorithm was somewhat sensitive to our choice of $C(m, N_m)$, the function mapping a measure m and its neighbors N_m to a composer. The KNN algorithm performed best overall for $C(m, N_m)$ defined as

$$\underset{\text{composer } c}{\arg\max} \sum_{n \in N_m} \mathbf{1}\{c \text{ composed } n\} \exp\left(\frac{1}{d(n, m)^2}\right).$$

This choice of C decreases very quickly as $d(m, n)$ increases, suggesting that our KNN classification performs best when $C(m, N_m)$ outputs the single nearest neighbor of m, using a majority vote when m has many neighbors at approximately the same distance.

5 Results

We see the following consistencies across classifiers: First, all classification methods had difficulty distinguishing be-

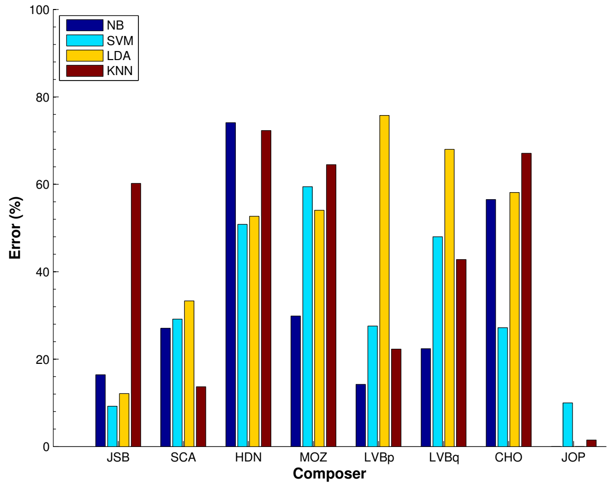

Figure 2: Test error for the four main algorithms used. In each case, we trained on 45 scores from each composer and tested on the remaining scores. For NB, SVM, and LDA, we used the *pitch-count* feature. For KNN, we used the *measure-count* feature. See Figure 1 for the full names of the composers listed.

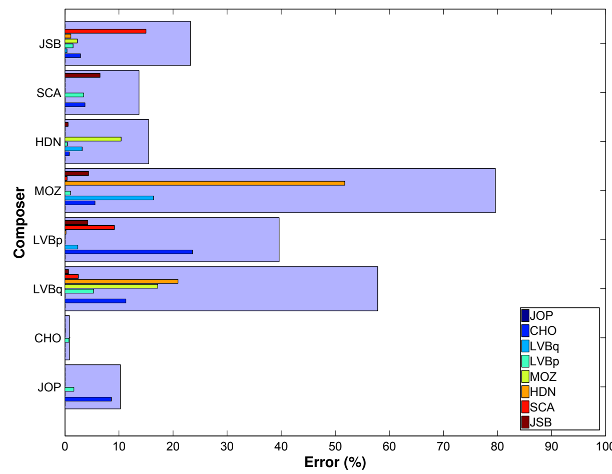

Figure 3: Confusion plot for Naive Bayes classification. The small bars represent the percentage of a composer's works which were misclassified as the composer listed on the vertical axis. The large bars represent the sum of the small bars they contain.

tween composers belonging to the same musical period. For example, in the SVM confusion graph (Figure 4), we see that most pieces misclassified as Bach were by Scarlatti and vice versa. This is unsurprising, since Bach and Scarlatti were contemporaries and both wrote in the Baroque style. We see the same pattern can be across Classical (Haydn, Mozart, and Beethoven) and Romantic (Beethoven piano and Chopin) composers in all our confusion plots.

Second, all classifiers were able to distinguish Joplin's works to a high degree of accuracy. This is consistent with the fact that Joplin's ragtime style that was a distinct departure from the European classical traditions.

5.1 Naive Bayes

We initially conducted our Naive Bayes modeling using 70% of each composer's scores for training and the remaining for cross-validation testing, but we found that this method biased the model towards composers with more scores. To remove this bias, we trained on a constant number of scores from each composer (45) and tested on the remaining scores; doing so, we obtained much more balanced results. We used this cross-validation scheme for all of our other classifiers as well.

5.2 Support Vector Machine

Using the same training and cross validation scheme as Naive Bayes, we tried features including *pitch-count*, *note-count* (which reports only the number notes in a chord and ignores the notes' pitches) and *pitch-present* (which is similar to *pitch-count*, except that for each pitch, the feature only reports a 1 if it appeared once or more, or a 0 if it never appeared). For each set of features, we performed a grid search to approximate the optimal values for the SVM parameters: We measured the performance of the classifier for values of the slack constant $C \in \{2^{-5}, 2^{-4}, ..., 2^{15}\}$ and the radial basis function parameter $\gamma \in \{2^{-15}, 2^{-14}, ..., 2^{3}\}$ by performing 10-fold cross validation on the training set. We found that the *pitch-count* feature set yielded the highest cross-validation accuracy of 67.5%, followed by *pitch-present* at 63.9% and *note-count* at 60.6%. It is interesting to note that *pitch-count* was also the feature set that yielded the highest accuracy in the Naive Bayes classifier. However, we still achieved a reasonably high level of accuracy when given only boolean values for each pitch (in the case of *pitch-present*) or when given only the number of notes in each chord (in the case of *note-count*).

Our last step was to perform a fine-grained search for C and γ around the local maximum found previously. With these parameters and the *pitch-count* feature, our cross-validation accuracy increased from 67.5% to 68.3%. Our overall accuracy on the test set using these optimal parameters is shown in Figure 2, and the confusion matrix is plotted in Figure 4.

5.3 K-Nearest Neighbors

Even with a high degree of tuning, our KNN algorithm performed on par with LDA and worse NB and SVM. We had hoped that the larger features we used would allow us to extract more information from the scores, but this turned out not to be the case. We suspect that the critical piece of information lost in the *measure-count* feature

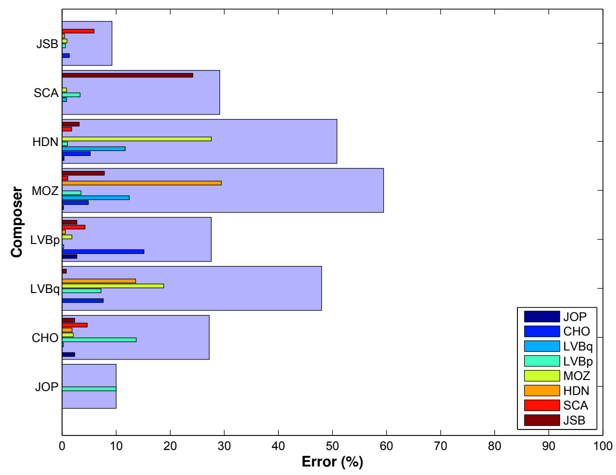

Figure 4: Confusion plot for SVM classification.

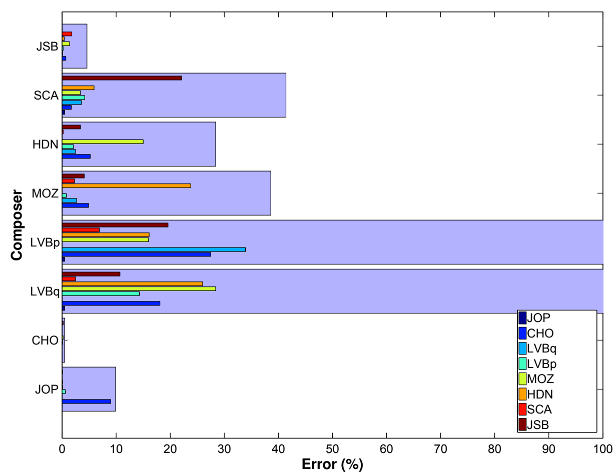

Figure 5: Confusion plot for KNN classification.

as compared to *pitch-count* is the number of notes being played at once. This information is likely key to successful differentiation between string quartets and keyboard pieces, something which our KNN algorithm had difficulty doing (see Figure 5).

5.4 Linear and Quadratic Discriminant Analysis

We achieved training error of 0.8% and cross-validation accuracy of 51.3% using linear discriminant analysis. While this seems substantially lower than SVM, we see from Figure 2 that LDA is quite comparable to SVM for all composers except for the Romantics (Beethoven and Chopin). The difficulty LDA had classifying the Romantics is likely due to the linearity of its decision boundary or and the fact that our data is not distributed according to a multivariate Gaussian.

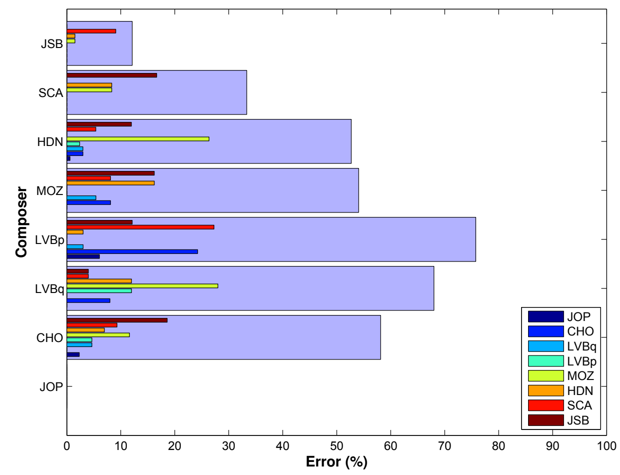

Figure 6: Confusion plot for LDA classification.

Meanwhile, QDA had an error rate upwards of 75%. QDA requires significantly more samples than LDA to classify data of the same dimension, so we had to run QDA on a lower-dimensional vector space than LDA, essentially giving QDA less training data. Furthermore, the fact that our data is not multivariate Gaussian (see Figure 8) made errors even more pronounced in QDA.

6 Analysis

In order to understand how well our classification methods have performed, we need to estimate the Bayes error that is inherent to our classification problem. The Bayes classifier is ambiguous when there is overlap in the posterior probability distributions of two or more distinct classes. Therefore, we can estimate the Bayes error by approximating the degree of overlap between the posterior distributions. We do this through PCA and a historical analysis.

6.1 Principal Component Analysis

Our first approach to estimating the degree of overlap between features of distinct classes was to create a 2-dimensional visualization of the data. This was done by mapping the *pitch-count* features to their first two principal components. We observed that composers belonging to the same musical period exhibit significant overlap in the first two principal components (see 8). On the other hand, composers belonging to distinct musical period show a high degree of separability (see 7). These observations help explain the patterns in confusion errors discussed above.

6.2 Historical Analysis

The degree of similarity in the music of distinct composers can also be traced historically. Haydn and Mozart

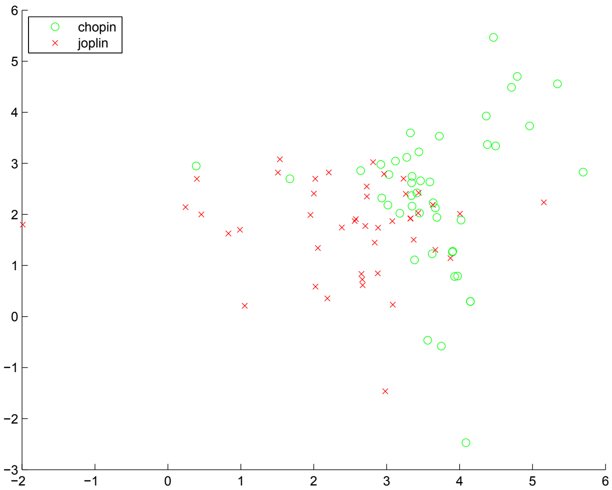

Figure 7: PCA of Romantic vs. Ragtime composers.

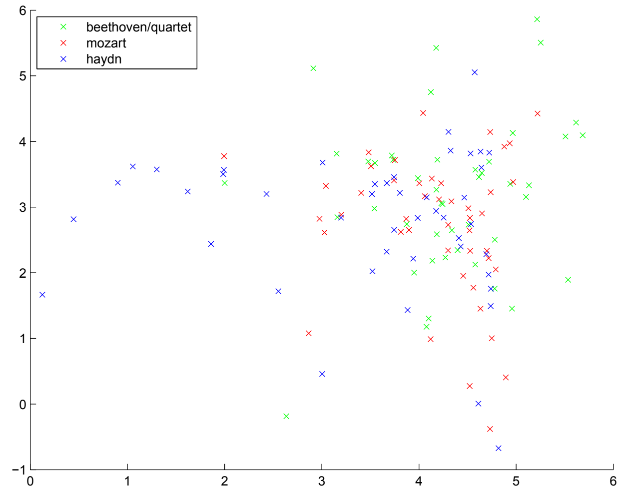

Figure 8: PCA of Classical composers.

were both in Vienna from 1781–1784 and were admirers of each other's works. Haydn's Opus 20 string quartets are believed to have been inspired by Mozart's K168–173, while Mozart's K387, 421, 428, 458, 464, and 465 are widely known as the "Haydn quartets" and were inspired by Haydn's Opus 33 [4]. Similarly, Beethoven was Haydn's pupil in Vienna from 1792–1795 and Chopin's late contemporary. Beethoven's later works (after 1815) are widely considered to be the beginning of the Romantic period, a canon to which Chopin belonged.

The similarities in Haydn and Mozart string quartets have been quantitatively accessed by Carig Sapp and Yi-Wen Liu of Stanford University. In an online quiz that asks listeners to distinguish between movements from Mozart and Haydn string quartets, the accuracy rates ranged from 52 to 66% for self-identified novices and experts respectively [5]. In light of the difficulty even experts have with this classification problem, our algorithms' difficulty here is not surprising.

7 Conclusions

We see that SVM is consistently the more accurate classifier across all composers we analyzed. This is likely because SVM makes no assumptions on the probabilistic distribution of features, allows for nonlinear decision boundaries, and can train on a sparse yet high-dimensional data. These properties make the SVM the superior classifier for this particular problem. Although our error rates for classifying some composers were large, errors we saw are in line with what we'd expect from a historical analysis, PCA, and human trials.

8 Further work

In this paper, we laid a framework for applying machine learning techniques to `**kern` scores. Our analysis here has been context-free: Our NB, SVM, and LDA algorithms consider only how many times a given chord appears in a score, and our KNN algorithm classifies measures with no consideration given to the surrounding measures. Future work might involve attempting to add more context to these models, either by adding context to the features used, or by using a context-full model, such as an HMM.

9 Acknowledgments

We'd like to thank Prof. Jonathan Berger for his valuable guidance at the beginning of this project. We'd also like to thank the Humdrum Project and the Center for Computer Assisted Research in the Humanities for making their collections of `**kern` scores available online for free.

References

[1] "MIREX 2008," International Music Information Retrieval Systems Evaluation Laboratory, University of Illinois at Urbana-Champaign, `http://www.music-ir.org/mirex/2008/index.php`

[2] "Everything You Need to Know About the Humdrum '`**kern`' Representation," Ohio State University School of Music, `http://dactyl.som.ohio-state.edu/Humdrum/representations/kern.html`

[3] Saunders, C., Hardoon, D. R., Shawe-Taylor, J. and Widmer, G. (2004) Using String Kernels to Identify Famous Performers from their Playing Style. In: The 15th European Conference on Machine Learning (ECML) and the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), 20-24 September, Pisa, Italy.

[4] Rosen, Charles. "The Classical Style: Haydn, Mozart, Beethoven." W. W. Norton Company, 1998.

[5] Sapp, Craig and Yi-Wen Liu. "The Haydn Mozart String Quartet Quiz." Stanford University, `http://qq.themefinder.org/`.