Composer Classification with Score Data

Zach Sullivan

***Abstract*—Music theory gives us a large amount of higher order structures to construct pieces, or scores. I explore different sets of features extracted from these structures that to create of a successful composer classifier. I found that any information pulled from the scores would help classify it better than a majority classifier. The set of features that was most indicative of a composer was a combination of key signatures and time signatures together with notes and their duration.**

I. Introduction

Music informatics makes use of machine learning to answer several domain specific questions including music fingerprinting, song recognition, music recommendation, and even to assist in musical composition. A majority of the work makes use of audio and MIDI data as a sort of “bottom up” approach to learning.

A relatively untapped kind of musical information comes in the form of scores. Score data contains the higher level information of a piece and may be thought of as a “top down” approach to working with music. Composers in the past several centuries have recorded their musical pieces in this highly structured form. It is because of the structure that distinct differences between composers can be heard by the human ear when performed. Looking at scores as data in a machine learning problem may provide new approaches to labeling pieces by composers.

II. Background

A. Data

Musical scores pose several challenges for the data scientist. For example, its complex structure is not a simple spread sheet or string. Text data is naturally a sequence of words or characters. Score data is a sequence of *measures* (if there is only one part), which have a context that includes tempo, dynamics, key signatures, time signatures, and several different kinds of primitives. For this project, primitives are tuples which contain a note or a rest paired with a duration. Primitives occur in a musical context, which in this project are represented as a tuple of a key signature and time signature. A sequence of these make up a single voice or part in a piece. A set of voices played in parallel make up a score or a data point when paired with a composer.

The largest issue when working with score data is finding it. Western music notation has been around for hundreds years, but encoding it in a digestible data representation requires more effort to digitize than simply scanning old scores as images or recording MIDI/audio data from performances. Because of their abundance, for a data scientist working with MIDI or audio data is less of a headache.

Music data has a plethora of file types all with different attributes. There are audio files which contain sound waves over time. There are MIDI files which contain enharmonic pitches over time (this is often called symbolic music data). Data formats that represent complex symbolic data in the form of scores include ABC notation, MusicXML, `**kern` and Lilypond. For this project, we will be using in Lilypond (a markup language similar to LateX) and MusicXml (an xml based symbolic representation).

Composer	pieces	primitives
Bach	192	55,706
Beethoven	23	45,713
Horetzky	60	10,542

Fig. 1. Data Summary, primitives are pitches or rests paired with their duration

The dataset I use is comprised of pieces from three different composers: J.S. Bach, L.V. Beethoven, and F. Horetzky. The Bach pieces are his chorales, numbers 250-438 from *Bach-Werke-Verzeichnis*. These pieces were in MusicXML format and a publicly available dataset: `jshanley` [**?**]. The Beethoven pieces are from his piano sonatas where different movements are represented as different pieces. The Horetzky guitar pieces are from series called *60 National Airs of Different Nations*. The Beethoven and Horetzky pieces come from an open source Lilypond project: `MutopiaProject` [**?**].

My goal is to find a set of features extracted from scores of these three composers in order to guess the composer of an unknown piece from the set. This involves translating the data into the same representation and extracting features from it, and then training a model over these features.

Lebar et al. [**?**], approach the same problem discussed here, they work with a dataset in the format of `**kern`. They had a larger dataset with more composers which allowed them order to sharpen their classifiers more. The features they used were primarily chords, that is notes that sounded simultaneously.

III. Methods

A. Preprocessing

The Lilypond dataset did not come in a form that was easy to run through multi-class classifiers. The markup language is context-sensitive and can contain arbitrary Scheme code. Therefore, parsing it is a non-trivial task. Luckily, a tool exists `python-ly` that would convert Lilypond into MusicXML. The tool could not handle arbitrary Scheme code which prevented some of the Beethoven pieces from being included in the dataset. Given that the tool could not handle the Scheme code, it is possible that other parts of the score may have been corrupted by this incomplete tool.

Extracting a representation of a score from MusicXml was easier, but still difficult as things like the duration of note and chords were context-sensitive. Unlike the Lilypond scores though, the xml based format was easily lexed and parsed into a generic xml rose tree. Unfortunately, musical elements were still not parsed correctly, most notably parallel note compositions like chords and multiple voicings were turned into just sequential notes. Several part pieces (not to be confused with multiple voicings) were parsed successfully, i.e. the Bach chorales. Other information that was discarded at this step were accidentals and dynamics.

B. Feature Selection

Music as a piece of data has some similarities to text data. Like text data it comes in some arbitrary length, for text data it is strings. A common approach for handling the strings is a bag of words, that is, a count the number of times in which each word occurs in a piece of data. Words as part of a vocabulary which themselves are composed as small sequences of characters having semantic meaning is not the case in music. Instead, primitives composed of pitches or rests paired with durations are used as the elements in a bag.

We can represent the musical contexts, key signatures and time signatures, as binary values because there is only one for each piece. This is done out of convenience given the dataset, but it is not necessarily the case that there is only one per piece.

The total features extracted from the scores in this analysis are the following:

- *Key Signatures* and *Time Signatures*, these contextual elements of a score will be different for each composer. These features spaces for my dataset are $\{0,1\}^{17}$ and $\{0,1\}^{7}$ for key and time signatures respectively.
- *Bag of (Pitchclass + Rests)* $\times$ *Duration*, these primitive elements are the basic building blocks of the scores and the features we have the most data for. This feature space is represented in the set $\mathbb{N}^{536}$, where 536 is the number of unique primitives in the dataset.
- *Bag of (Pitchclass + Rests)* $\times$ *Duration* $\times$ *KeySignature* $\times$ *TimeSignature*, essentially this is a combination of all of the previous features. The key and time signatures are just turned from binary to natural numbers in this case, making the feature space $\mathbb{N}^{560}$.

C. Data Partitioning

Data partitions for training and testing where done with a 70/30 split. Partitioning was done after randomly permuting the data, which was necessary so that all of the labels would show up in both the training and test data.

D. Classifiers

Evaluation included testing several classifiers, but more importantly it involved checking which feature sets were the most effective for classifying composers. Each of the following classifiers was applied to each feature set:

- *Majority*, this simply picked the the most common composer in the test set. In every case, this was Bach. This was the baseline.
- *Logistic Regression*, a basic linear classifier.
- *SVM* was chosen to avoid overfitting, especially because data was scarce.
- *Multinomial Naive Bayes*, was chosen as a linear model that is also a generative model.
- *Random Forest*, in the case that domain specific divisions of features did not capture the separation in the data, or that it was not linearly separable, random forests would provide a good contrast from the linear classifiers.

Classifier	Correct	Incorrect	Percentage
Majority	61	21	74.4
Logistic Regression	82	0	100
SVM	81	1	98.8
Multinomial Naive Bayes	82	0	100
Random Forest	81	1	98.8

Fig. 2. Best results with different classifiers, all achieve using all of the features.

IV. Experiments

A. Classifier Evaluation

For running the experiments on the 275 data points with a 70/30 split, there were 193 points in the training set and 82 in the test set.

??, shows that the linear models were all effective at classification and that the data was linearly separable. Depicted in **??**-**??**, we see the confusion matrices for Logistic Regression applied to different feature spaces and in their caption we see the accuracy. The other linear classifiers performed similarly with each set of features, where performance is measured as accuracy.

B. Ablative Analysis

One of the goals is to determine which of the feature sets is most indicative of a composer. I found that key signatures and time signatures did not influence the selection of the piece by that much, but they were better by at least 4% accuracy than the majority classifier.

These classifiers were likely effected by the amount of data for each composer. Looking at **??** and **??**, Beethoven was *never* guessed as the composer and he had 40 less pieces than Horetzky. Bach was selected 64 more times than Horetzky in **??**, the time signature feature set. In **??** the key signature feature set, Bach was only chosen 44 more times than Horetzky despite there being the same number of each composer in the two feature spaces. This suggests that key signatures are a better indicator than time signatures, which is likely due to the number of unique key signatures being larger than time signatures in this dataset.

C. Handling Skewness

Several attempts were made to handle the skewness of the data. The skewness came in two forms: inequality in the number of musical primitives (Horetzky had far fewer than the

	Bach	Beethoven	Horetzky
Bach	61	3	9
Beethoven	0	0	0
Horetzky	0	2	7

Fig. 3. Logistic Regression Classifier on Time Signature features (columns are the actual value and rows are the predicted values). Correct: 68 out of 82; Accuracy: 0.829

	Bach	Beethoven	Horetzky
Bach	53	5	5
Beethoven	0	0	0
Horetzky	8	0	11

Fig. 4. Logistic Regression Classifier on Key Signature features (columns are the actual value and rows are the predicted values). Correct: 64 out of 82; Accuracy: 0.780

	Bach	Beethoven	Horetzky
Bach	60	1	1
Beethoven	0	3	0
Horetzky	1	1	15

Fig. 5. Logistic Regression Classifier on (Note,Duration) features (columns are the actual value and rows are the predicted values). Correct: 78 out of 82; Accuracy: 0.951

	Bach	Beethoven	Horetzky
Bach	61	0	0
Beethoven	0	5	0
Horetzky	0	0	16

Fig. 6. Logistic Regression Classifier on all features (columns are the actual value and rows are the predicted values). Correct: 82 out of 82; Accuracy: 1.0

others **??**) and inequality number of total pieces, which Bach had far more than Horetzky which had more than Beethoven.

To handle the first case, I included every Horetzky piece 5 times. With less variance in the number of musical primitives, LR's accuracy was 66.2 % on time signatures, 81.8 % on key signatures, 94.8 % on primitives, and 99.4 % on all of the features.

To handle the second case, I randomly removed two-thirds of Bach's pieces and included Beethoven's pieces twice. LR's accuracy on the resulting dataset was 50.9 % on time signatures, 78.6 % on key signatures, 91.2 % on primitives, and 98.1 % on all of the features.

These two attempts to reduce skewness decreased the performance of the classifier in terms of accuracy, for instance the complete features set with Logistic Regression dropped from 100 to 98.2 percent accuracy. Like with the vanilla dataset, the choice of which linear classifier did not have an impact of more than 5% accuracy.

V. Discussion

A shortcoming of the dataset used is that all of the pieces are played with different instrumentation. Bach's pieces are all multi-part string instrument arrangements. Beethoven's pieces are for piano and Horetzky's are for guitar. Therefore, I have created a successful instrumentation classifier as well as a composer classifier. The only remedy for this is more data.

A. Future Work

This was just a small subset of the pieces from the `MutopiaProject`. They were selected because there were a large collection of pieces from a single composer. It is probable that we will find overfitting with our model when we put more of the `MutopiaProject` through this feature extraction pipeline.

Not explored here are the effects of accidentals and chords in classification. They are relevant features especially in this dataset because each piece was written for different instrumentation and that effects chord usage. Accidentals are another indication of key and some composers were known for straying outside the key notated with the key signature. Exploring these features could lead to better classifiers, especially with more data. This would envolve expanding the MusicXml parser.

Music describes temporal information and therefore begs for a Markov model representation, which could capture information as the music develops over time.

Generative models for music are a popular research area for automatic music composition. With the infrastructure created for feature extraction in this paper as a base, it would not be difficult to try some generative models to produce some interesting music trained on different composers.

There is a lot of research on feature extraction and music fingerprinting for musical data in the form of audio and MIDI. A question that the success of the classifiers presented in this paper asks is "how does it compare to classifiers for the same task, but with different data types?" Answering this question would require a dataset that contains both score data and audio or MIDI data.

B. Conclusion

I have described a dataset for musical scores and produced a series of feature sets for analyzing score data. All of the features extracted had some impact on the predictive accuracy of a linear model. Increasing the feature set only increased the accuracy of classifier. More data from different composers will be needed to test the robustness of the features and models used here.