CS230

Classical Composer Identification on Interval Features for CS230-Spring 2018

Andrew Davis
Department of Computer Science
Stanford University
daandre@stanford.edu

Abstract

This paper examines the ability of a neural network to classify Western Classical composers given only information about a work's musical intervals. For this paper, five composers were chosen: Johann Sebastian Bach, Joseph Haydn, Ludwig van Beethoven, Arcangelo Corelli, and Frédéric Chopin. The dataset was statistical information about a given work's interval profile that was extracted from MIDI representations of musical works. This was fed into a three layer neural network with a "softmax" classifier. The results showed that the neural network could correctly identify with 83% accuracy. The results showed that Classical and Baroque composers can be distinguished solely on the basis of how composers treat musical intervals in their works.

1 Introduction

Identifying Classical composers in and of itself is not a problem that needs to be solved. The corpus of work from these composers is finite and is known by most musicologists and scholars. The trained musical ear is quite good at picking apart famous composers from other famous composers. A more important question is what are the essential characteristics that make them distinct. Is it types of instruments? Is it choice of melodic phrase? Is it harmonic motion? The answer is likely a combination of many different characteristics. Music theorists and musicologists have theorized different reasons as to the importance of specific features but these are qualitative assessments. Neural networks can provide quantitative analysis of musical works and help buffer or detract from various theories as to what defines a composer's style.

To narrow the scope of the project, this paper examines the extent to which knowledge of a work's musical intervals aids our ability to identify composers. To rephrase, if simply given information about a work's interval structure, could a neural network sufficiently learn to distinguish major Western Classical composers from each other?

To directly answer this question, a neural network was given seventeen statistical features about a MIDI file of a composer's work. Some examples of the seventeen features include the fraction of perfect fifths and most common melodic interval. (For a complete list of features, please see section 3 on dataset and features). The output to the neural network was a softmax classifier for five classes. In this project, five composers were selected. They are as follows: Johann Sebastian Bach, Joseph Haydn, Ludwig van Beethoven, Arcangelo Corelli, and Frederic Chopin. The output predicts which of the five composers was most likely to have written the work given the seventeen features provided to the network.

CS230: Deep Learning, Spring 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

With the rise of Artificial Intelligence over the last few years, musicologists, theorists, and computer scientists have begun to explore the relationship between neural networks and music. Common approaches often investigate music information retrieval, algorithmic composition, and classification. Since this paper explores classification, the related work section will focus primarily on this subsection of research.

Instrument identification is a typical avenue of exploration. This area of classification often deals with raw audio that undergoes significant preprocessing. Oftentimes, this involves using Fast Fourier Transforms to create spectrograms of the audio files. In Han et. al's paper "Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music," the authors convert polyphonic music into spectrograms whose image is then processed through a convolutional neural network[1]. Others have used the raw audio as the basis for exploration in a convolutional neural network and achieved similar results[2] but using an "end-to-end" approach. In some instances, researchers have been even more involved in the feature extraction. In Babak Toghiani-Rizi and Marcus Windmark's "Musical Instrument Recognition Using Their Distinctive Characteristics", the authors use various hands-on preprocessing to extract information about audio attack and decay to ascertain an instrument in a monophonic setting (i.e., only one instrument playing at a time).[3] This paper attempted to answer questions about the important of attack in identifying an instrument and favored a hands-on approach to test it, similar to what this paper seeks to do with musical intervals and composers. The aforementioned papers as well as Shelar and Bhalke's "Musical Instrument Recognition and Transcription using Neural Network"[4] seek to address instrument identification in a polyphonic setting (i.e., multiple instruments playing at once).

In the field of composer identification, researchers have used probabilistic neural networks and feedforward neural networks. In Kaliakatsos-Papakostas et al.'s "Musical Composer Identification through Probabilistic and Feedforward Neural Networks," the authors found that "as the similarity between two composers increases, the identification effectiveness between those two decreases."[5] In 2015 in Herreman et al's "Classification and Generation of Composer-Specific Music Using Global Feature Models and Variable Neighborhood Search", the authors attempted composer identification of three composers (Haydn, Beethoven, and Bach) using a small subset of features extracted through the jSymbolic library[6][9]. This paper is most similar to Herreman's research which tested on three different models: RIPPER ruleset, C4.5 Decision tree, and logistic regression. Logistic Regression was the best performer for their features and datasets at 81.7%. This paper focuses on different features and expands on the number of classification and attempts to solve a similar problem with feedforward neural networks.

3 Dataset and Features

The dataset was obtained through the Kern Scores library at the Center Computer Assisted Research in the Humanities at Stanford University[10]. Kern contains MIDI files of works from composers primarily from the 16th to the 18th century. The composers chosen for this research were Bach, Beethoven, Corelli, Haydn, and Chopin. In sum, 1341 total examples were collected (Bach: 563, Beethoven: 188, Chopin: 92, Corelli: 245, Haydn: 253) and 150 each were randomly selected for the development set and training set. Each MIDI file was preprocessed to extract seventeen features relating to music intervals using the open source tool jSymbolic[9].

The seventeen features are as follows: most common melodic interval, mean melodic interval, number of common melodic intervals, distance between most prevalent melodic interval, prevalence of most common melodic interval, relative prevalence of most common melodic intervals, chromatic motion, stepwise motion, melodic thirds, melodic perfect fourths, melodic tritones, melodic perfect fifths, melodic sixths, melodic sevenths, melodic octaves, melodic large intervals, and minor major melodic third ratio. Most of these features are evaluated on a percentage basis. For example, a feature such as melodic perfect fifths contains a real number $[0, 1]$ that represents the percentage of perfect fifths with respect to all the intervals in the entire work. Other features like most common melodic interval is represented as a natural number from one to twelve where each number represents one of the twelve possible intervals. All features were normalized across the data set.

4 Methods

Several models were tried, all using fully connected feedforward neural networks. The first was logistic regression which was used as a baseline model. It used a single layer with five nodes representing the output for each possible composer and then used a softmax classifier to pick the composer. Please note that all models trained in this research used a softmax output. The loss function for softmax is as follows:

$$L = -ylog(\hat{y})$$

The second model added another layer of 5 nodes. No other hyperparameter tuning was used. This model also functioned as a baseline model.

The third model was more sophisticated. It was a three layer model where the first two layers were tuned with various different number of nodes. It used L2 regularization to combat overfitting the training set. For L2 regularization, the following was appended to the overall cost function:

$$\frac{\lambda}{2m} \sum_{l=1}^{L} ||w^{[l]}||_F^2$$

Adding this cost penalizes strong weights and reduces the propensity of the weights to mold to only the training data. Additionally, the model was trained with an Adam optimizer which combines momentum and RMSprop to more quickly reach the global minimum. The model was trained using various different mini-batch sizes in conjunction with Adam optimization.

The fourth model added another layer and used similar techniques to the three layer model above. As will be shown in the results section, the fourth model was prone to overfitting so dropout was used to help assuage overfitting though with limited success. Dropout is another technique to combat overfitting where various nodes of a given layer are randomly "zero-ed" out at a prescribed rate. This prevents the model from relying on any one node or series of nodes to make its prediction.

5 Experiments/Results/Discussion

The results of the different models showed that the three layer model with L2 regularization and Adam optimization was the best performer. Logistic regression was the worst performer with an accuracy of approximately 73% on the testing set. The four layer model was prone to overfitting and attempts at regularization through dropout and L2 regularization compromised training accuracy too severely. The two layer model that was also used as a baseline was a better performer than logistic regression but not quite as good as the three layer model. A summary of performance can be found in Figure 1 below.

Figure 1: Generalized overview of performance by the four models assessed.

Model Performance		
Model	Training Accuracy	Testing Accuracy
Logistic Regression	73%	73%
Two Layer Model	82%	79%
Three Layer Model	87%	83%
Four Layer Model	84%	77%

Given that the training set was small, there were plenty of opportunities for hyperparameter tuning for the various models. Most of that energy was devoted to the three layer and four layer model. In initial training with the three layer model a learning rate of .001 was used with no attempts at regularization. Through experimentation it was quickly discovered that a small number of nodes was needed for each layer. Attempts to raise the number of nodes for each layer above ten proved ineffective. The final number of nodes in order from the first layer to the softmax layer was eight, six and five, respectively.

The three layer model was trained on a mini-batch of size 64, though 32 and 128 were also tested and found to be not as effective. Tuning of the epochs showed the best numbers were in the high hundreds. Too small of epochs did not effectively train the model.

Initially the model had a training accuracy of 89% but a development set accuracy of roughly 79% which was in line with the two layer model. To combat the overfitting, L2 regularization was added. The optimal λ rate found was .005. Additionally, dropout was initially used to combat overfitting. But dropout was too severe of a compromise to training accuracy given the small number of nodes used in each layer and the depth of the layers.

Figure 2: Examination of precision, recall, F1-score for the three layer model.

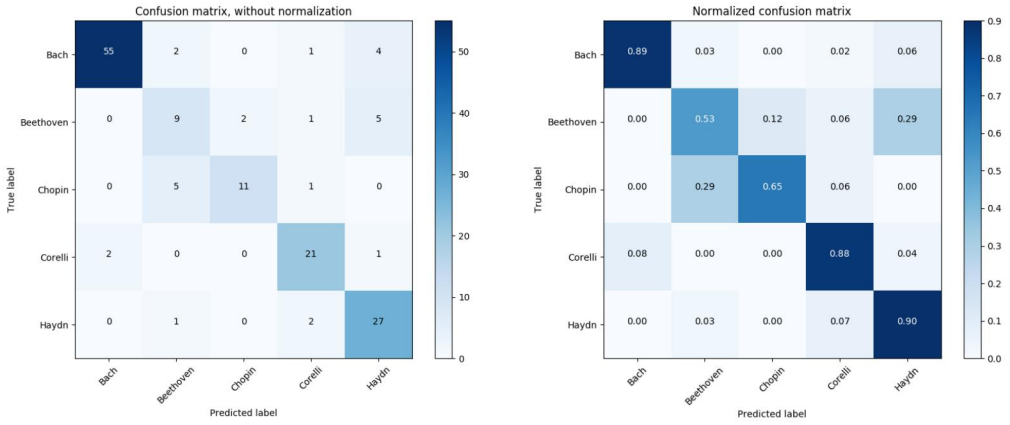
Three Layer Class Performance				
Composer	Precision	Recall	F1-score	Number of True Examples
Bach	.97	.92	.94	62
Beethoven	.59	.59	.59	17
Chopin	.92	.65	.76	17
Corelli	.83	.79	.81	24
Haydn	.69	.90	.78	30

Figure 2 above shows the results of the three layer model based on precision, recall and F1 score. Bach was the best performer of the group. The model was successful in both precision and recall in identifying his works. From a statistical perspective, this result was expected given that there was more data for Bach. From a musical perspective, this was also expected. Bach's distinctive and consistent use of musical intervals is well-known to music theorists and should be capable of being learned by the model.

The model had the most difficulty identifying the works of Beethoven. The model performed poorly both in precision and recall. There are two possible explanations for this: 1) the number of Beethoven examples was comparatively small (the total data collected included only 188 examples, the second smallest after Chopin), and 2) Beethoven is known for his change in style over the course of his life, which could create an inconsistent profile in regard to his use of musical intervals. Beethoven's fame as a composer stems not only for the great works he produced but also his evolving style. Successive works showed an evolution and maturity in musical chromaticism, which would in turn affect his use of musical intervals. Given that the data spanned works from his entire life, the model would have a difficult time distinguishing his evolution in musical intervals. Indeed, other researchers have confirmed this hypothesis such as Herremans et al. stating "Beethoven typically does not focus on using one particular interval, in contrast to Haydn or Bach, who have a higher prevalence of the most common melodic interval[6]."

Figure 3: Confusion Matrix

A confusion matrix of the three layer model showing the performance of the three layer model on the test set. Note: graphs were made using scikit-learn and matplotlib[11][12]



The four layer model was an improvement in training accuracy before regularization but the model overfitted the train data significantly. Initially the model had a training accuracy of 91% but a development set accuracy of around 70%. Similar attempts to the three layer model were made to the four layer model. It used a small number of nodes for each layer of 7, 6, 5, and 5, respectively. It

also used L2 regularization and Adam optimization and included a fractional dropout rate of 0.005. Nevertheless, it was not as successful as the three layer model. Attempts at larger models also proved ineffective.

6 Conclusion/Future Work

A shallow neural network was the best fit for this specific problem. Multiple rounds of parameter tuning showed that the optimal network was a three layer neural network of eight nodes, six nodes, and five nodes for the softmax classification. The optimal hyperparameters used to train the model can be found in Figure 4 below.

Figure 4: Optimal parameters for the three layer model.

Number of Epochs	900
Mini-batch size	64
λ for L2 Regularization	.005
Learning Rate	.001
β_1 for Adam Optimization	.9
β_2 for Adam Optimization	.999

The model was trained on 1041 examples with a development set of 150 and a test set of 150. In summary, the model was able to achieve 83% accuracy on the test set, an improvement of about 10% on the baseline logistic regression model. When the data were trained on deeper models, the model suffered from overfitting. In the optimal model, this was mitigated with L2 regularization. Ultimately, the best balance was achieved with a shallow neural network, an appropriate model given the complexity of the problem.

In the future, more data could be collected. The Kern database that housed the MIDI files only contains a subset of each composer's works, primarily keyboard music. But each of the composers listed wrote significantly more music than what was passed into the neural network. Furthermore, it would be interesting to expand the classifier to more composers or to train the models on composers from different musical eras. This model was trained on composers from the 17th and 18th century.

Additionally, this paper tackled the issue of composer identification by examining a small subset of the features in a given musical work. One could attempt to classify a musical work based on instrumentation, melodic profile, rhythm, or any number of different features. In fact, a preliminary model in the early stages of this research was trained on 141 features that included all these statistics and showed an accuracy of up to 96% on a supplied test set. However, for the purposes of musicologists and theorists, subsets of features provide more information because they can ask more focused questions about key aspects of style. Indeed, this paper has shown that the success of the model on just information about a work's interval structure supports the notion that musical intervals play an important role in identifying musical style.

7 Contributions

As I was the only team member for the project, all work and information found in this report and presentation is solely my own work. Inspiration was taken from the references below.

Code

Code and data for the project can be found here: https://github.com/andrewdavis33/CS230FinalProject

References

[1] Y. Han, J. Kim and K. Lee, "Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 25, no. 1, pp. 208-221, Jan. 2017.

[2] Li, Peter et al. “Automatic Instrument Recognition in Polyphonic Music Using Convolutional Neural Networks.” CoRR abs/1511.05520 (2015): n. pag.

[3] Toghiani-Rizi, Babak & Windmark, Marcus. (2017). Musical Instrument Recognition Using Their Distinctive Characteristics in Artificial Neural Networks.

[4] V S Shelar and D G Bhalke. Article: Musical Instrument Recognition and Transcription using Neural Network. IJCA Proceedings on Emerging Trends in Electronics and Telecommunication Engineering 2013 NCET:31-36, March 2014.

[5] Kaliakatsos-Papakostas, Maximos & Epitropakis, Michael & Vrahatis, Michael. (2010). Musical Composer Identification through Probabilistic and Feedforward Neural Networks. Lecture Notes in Computer Science.6025. 411-420. 10.1007/978-3-642-12242-2_42.

[6] D. Herremans, K. Sörensen and D. Martens, "Classification and Generation of Composer-Specific Music Using Global Feature Models and Variable Neighborhood Search," in Computer Music Journal, vol. 39, no. 3, pp. 71-91, Sept. 2015. doi: 10.1162/COMJ_a_00316

[7] Lebar, J., Chang, G., Yu, D.: Classifying musical score by composer: A machine learning approach, http://www.stanford.edu/class/cs229/projects2008.html

[8] Francois Chollet. Comma.ai, 2016. URL http://keras.io/.

[9] McKay, C., and I. Fujinaga. 2006. jSymbolic: A feature extractor for MIDI files. Proceedings of the International Computer Music Conference. 302–5. http://jmir.sourceforge.net/jSymbolic.html

[10] CCARH. 1984. Kern Scores. http://kern.ccarh.org/

[11] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[12] John D. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95 (2007), DOI:10.1109/MCSE.2007.55

[13] Travis E, Oliphant. A guide to NumPy, USA: Trelgol Publishing, (2006).