Mozart Simulator

Paul Smith, Luke Wright, Bryce Lunceford, Daniel Smith, Erika Ibarra

March 28, 2022

Abstract

Automatic music generation has an anxious market among movies and video games. We focus our work on pieces by Mozart because of their objectively structured compositions. Our data comes from *midi-files* which has the advantage of the data being structured for our methods. While there could be many approaches for generating music, we focus on utilizing *Hidden Markov Models* (HMMs). We attempt to generate additional notes in some of Mozart's work.

1 Problem Statement

Computer generated music was first seen in the late 1950s with the creation of, The Silver Scale, a melody produced at Bell Laboratories. The duration of this simple composition was 17 seconds (Briot 4). It was a breakthrough accomplishment, even if it did sound like a strange sci-fi jingle. With many notable advancements in the domain of machine learning, algorithm generated music has since become much more sophisticated. Today, one can easily peruse the internet and find modules like Musenet, developed from the research OpenAI, which can analyze anything from Beethoven to the Beatles, calculate the style, and synthesize a similar or complementary composition that utilizes sounds from a handful of instruments. The techniques most commonly used to support computer generated music come from Deep Learning and Deep Neural Networks. However, research suggests that simpler Markov models can be used as well. With that being said, there are trade-offs. On the one hand, Markov models are simple to use, they can learn from a small set of data, and they can often be implemented with a control. On the other hand, they may not be as good at capturing long term structures or generalizing as well as their Neural Network counterparts. The objective of this project is to test the hypothesis that Markov models can be used to to accurately synthesize music.

Altogether, the motivation for this project comes down to supporting art. Music is a huge part of human culture. It can influence feelings, encourage behaviors, and ultimately shape who we are. But music is also expensive to produce. Creating an algorithm that is capable of replicating the general style of Mozart piano sonatas could not only help individuals inherit the well-documented benefits of listening to his music, but it could provide such benefits cheaply, and in the form of new music each and every time.

2 Data

The data we used was a series of midi files with recordings of Mozart piano sonatas. These recordings were accurate renditions of Mozart's music, so we didn't have to worry about any missing data or innacuracies. Midi files are also constructed in a convenient manner that breaks the recording into individual timesteps and describes what happens to each note at each timestep. Using this information, we created an $n \times 88 \times 2$ array (where n is the number of timesteps in the piece). For each timestep, the first column describes which of the 88 possible notes are being played, and the second column describes the speed at which the corresponding notes are being played; this speed refers to the speed at which the action of the key is being pressed, not how long the note is played for. This array we created (hereafter called the Master Array), provided a simple way for us to work with out data. To train on multiple pieces of music, we simply concatenated the pieces' master arrays into a single array.

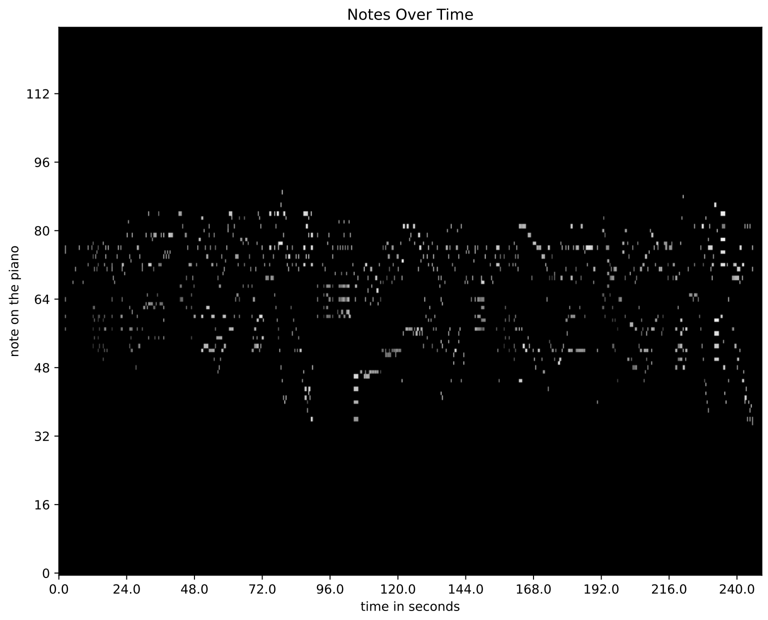


Figure 1: Representation of Mozart Piano Sonata No. 8 in A Minor, K.310, 1st Movement. As with all compositions by Mozart, musical patterns and structure are clearly evident.

3 Methods

3.1 State Space Dimension Reduction

Each note of the Master Array can take on three values for a given timestep: 1 signifies that the note is being pressed, -1 signifies that the note is being depressed, and 0 signifies that the state of the note has not changed. Since there are 88 possibe notes that could be played at any given timestep, and each note could take on 3 different values, each timestep has 3^{88} possible values. To decrease the dimension size of our data, we assigned a single number to each unique combination of notes played throughout the piece. This decreased the dimension size of our data from 3^{88} to only a few hundred. We then assigned each of these numbers to their corresponding timestep so that we had a single array of ints with length n. We used the same technique to also lower the dimension size of the note speeds.

3.2 Attempt 1: Hidden Markov Model

We used an *HMM* to create a model that tried to accurately simulate how the inputted music is construted using a given number of hidden states (which we intially tested with 20 states). We fit the model, and it gave us predicted notes to be played, but to determine the speeds of each note, we had to use a different approach. We decided to look at the hidden states the model generated, and then find the distribution of speeds that were associated with each hidden state. Recall that we lowered the dimension of the speed state space by assigning an integer to each unique value. Since our *HMM* gave us a hidden state for each timestep, we simply created a speed distribution for each hidden state, and then we took a random sample for each timestep and assigned it to that timestep. By this point, we had a new Master Array with new predicted notes and speeds. But before we could reassemble a midi file with our new song, we had to clean it.

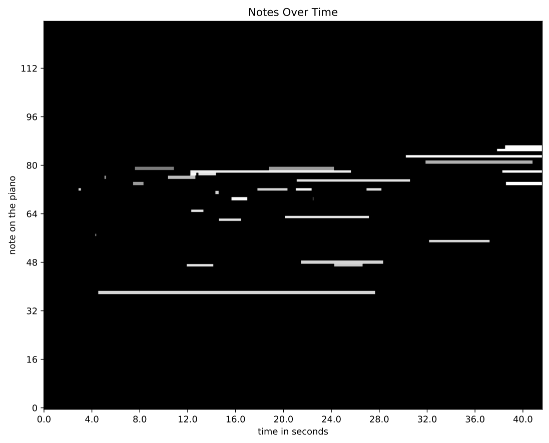
To clean our new song, we looked at the notes and the speeds individually. There were two problem cases with the notes: (1) the note was being depressed when it wasn't pressed before, and (2) the note was being pressed when it was already pressed before without being depressed. In both of these cases, we decided to set the note equal to 0 (i.e. keep the state of the note unchanged). At this point, the speeds associated to the notes at each timestep was an array of speeds we drew from the corresponding distribution of speeds. If the number of nonzero speeds in the speeds array was not equal to the number of notes being pressed in a given timestep, we simply assigned the mean of the speeds to each note being pressed. After this cleaning, we had a new Master Array where each note was pressed and depressed with an associated speed as expected. We then used this new Master Array to reconstruct a new midi file which could then be used to play our new song.

Unfortunately, the resulting music was much less satisfactory than we had hoped. The music was entirely unstructered and atonal, but our attempt to clean the music also very clearly left holes in the final product. We knew we could do better (almost anything would be better), so we decided to go back to the drawing board and try a different approach altogether.

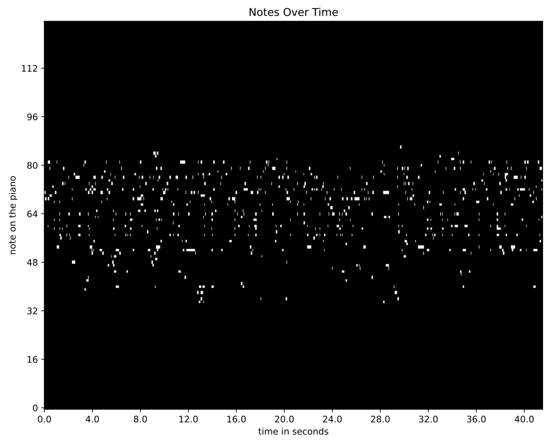
3.3 Attempt 2: Random Forest

Using our Master Array, we decided to create a new set of datapoints to train with, this time using a random forest. To create the new set of data, we iterated through the master array, and for each timestep we updated the state of the notes as follows. If a note was not being played at that timestep, we set its value to 0; if a note was being played at that timestep, we incremented its value by 1. This left us with an $n \times 88$ array which we then fed into our random forest to train on. Our hope was that the random forest would try to predict not only which notes to play, but also how long each note should be played for. Since we figured we had bigger fish to fry, we simply set each note (action) speed to a constant value (in this case, we set it to 120). While this didn't yield the most accurate results for the action speed, the results weren't too far off for a Mozart piano sonata, because as a Classical Composer, Mozart tended to keep his music within a relatively confined range of possible dynamics.

We also wanted to make sure our music was tonal this time around. In order to do that, we created a binary mask for each musical Key a piece could be written in. When we read in our data (a list of midi files), we made sure to generate a list of Keys that the music was written in. Then, when we began creating a new piece of music, the first thing we did was randomly sample from our distribution (list) of Keys we trained with. Then, to make sure our music would train in the right Key, we masked the entire training set with the Key we drew from our distribution. The result was that we trained on a dataset that only had notes play that were in the desired Key.



(a) Attempt 1 using Hidden Markov Model



(b) Attempt 2 using Random Forest

Figure 2: These visualizations clearly show that our initial attempt (left) had large holes and didn't accurately predict how long each note should be pressed. Our second attempt (right), though not free from error, yielded much better results for both note duration and overall structure. An attempt at musical patterns is also evident.

4 Results

In the end, the program we developed for this project succeeded in generating music. Our first attempt using a Hidden Markov Model proved quite unsuccessful, but our second attempt with a Random Forest yielded promising results. The melodies produced in our second attempt sounded quite cluttered, but they were obviously tonal thanks to our masking the data into a single key. In addition, the untrained ear could definitely detect patterns and general ideas develop as the music progressed. However, the music did not sound like Mozart. There is no way any comparison could be made between our machine-generated music and anything written by Mozart. Our final result was obviously producing music inspired by classical compositions, but it could not be mistaken for the real thing.

Samples of these musical compositions can be found at ****include link here*******.

5 Analysis

Classical music strove to maintain the rigorous musical structure developed during the Baroque Period, but it also sought to make music more enjoyable and pleasant to the ear than it was during much of the Baroque Period. While the first goal of Classical music is relatively quantifiable (that is, musical theory has mathematical properties), the second goal is not very quantifiable at all. It might be for this reason that our algorithm was incapable of generating enjoyable, beautiful music. This seems to further support the age-old consensus that human thought and emotion is virtually impossible to artificially replicate.

6 Ethical Considerations

On the surface, there are few ethical implications of our project regarding privacy. It is important to only train with music samples which have been either purchased by the user or are public domain. The analysis and results we conducted and collected do not have any ethical implications. However, our results could be misused or misunderstood. The goal of our project was not to eliminate original music production. Our goal was to construct an algorithm which could generate music that the entertainment industry can use in various circumstances as background music that most would consider inconsequential. While not yet possible, our analysis and results are not intended to remove original music production from the industry. After all, original composition is necessary to train from, and it would seem that the artist's touch has yet to be artifically replicated. Consequently, we do not consider the vital role of the composer to be jeopardized by an algorithm similar to our own, even if it could yield promising results.

7 Conclusion

In conclusion, it is difficult to create a model that will produce consistently useable music. However, we made progress in this area of research by using HMMs and Random Forests to generate unique musical content of a specific style. Our analysis and results stem from data that incorporates how and when a note is pressed and depressed. We simplified our models by organizing our data into a simplified, lower dimensional state space, but our simplified models still could not generate adequate music. We believe that data-predicting algorithms have a long way to go if they will ever be able to replicate the complexities of human emotion.