Stochastic Models for Procedural Music Generation

Brian Standage
Email: bstandag@iu.edu

1. Introduction

This paper presents an approach to simulating the composition of a fugue through stochastic processes aided by rules of counterpoint. Particularly, I will investigate methods to emulate the style of J.S. Bach (1685-1750), using the fugues from the Well-Tempered Clavier, BWV 846–893.

There have been a number of different approaches to simulate the process of contrapuntal based music composition. The two main methods can be generalized to be driven by either stochastic models or deep learning. Within these methods, there are both completely autonomous music composition systems and human assisted semi-autonomous systems.

Stochastic models are widely used in procedural content generation for their efficiency and simplicity. These models can be effectively trained over large data sets and quickly generate output. Music composition can be more easily customized and controlled through stochastic processes because the model is operational and has modifiable states. Limitations of stochastic models are their linear generation style and inability to preserve themes throughout a piece. In response to the short memory of stochastic models, some approaches to procedural content generation utilize stochastic models of order n. Using models of order n is advantageous in capturing longer musical themes. However, other papers have shown that models of order n quickly leads to verbatim repetition of the training data while the state space becomes increasingly massive as n increases [5].

Using stochastic models for music composition reveals a fundamental difference between humans and machines in the process of composing music. While stochastic models generate music linearly, humans compose music in an anachronic fashion. Recent work in building deep learning models for procedural music composition attempts to generate music in a manner that mimics the human process. The goal of mimicking the composition style of humans has motivated the use of convolutional neural networks to compose partial music scores that are independent of the direction of composition [3]. Deep learning models require a much larger amount of data, though, and take significantly longer to train. Nonetheless, the thorough training of deep learning models results in higher order dependencies to be formed and better preserves long term themes and context.

1.1. Motivation

Although I do not consider myself a serious musician, there are several motivations to this project that are founded in a goal of producing a deliverable result. Instead of doing some sort of classification or analysis task, I thought it would be very interesting to perform content generation as it produces results that are enjoyable to analyze. By doing so, this allows one to develop a good appreciation for big data and its applications. I find it very motivating and interesting to work with a very large dataset consisting of data constructed in the 1700's. This gives a very interesting perspective on the procedural content generation pipeline, as you can use data that is hundreds of years old to mimic the artistic styles of historic data sources.

Furthermore, this immediately seems like a task that would be better suited for deep learning. However, it is interesting to approach the problem using stochastic processes where we can create strict and rigid definitions for the model and better understand it's decision making process. Some criticisms of deep learning models is the "black box" problem where it isn't always clear why a model made the choices it did. Approaching the problem with stochastic models gives us a better understanding of how a somewhat random walk about a particularly crafted state space can produce desirable results. With a background in mathematics, I find it motivating to explore how many natural phenomena can be model by random walks on various state spaces.

1.2. What is a Fugue?

A fugue is a composition which uses techniques from both counterpoint and voice leading that involve two or more voices. A fugue is centered on a subject that is introduced by one voice, then adopted by the second voice at a different pitch while the first voice plays (or sings) a different melody that is referred to as the countersubject. When the subject reenters in new voices, it is often not an exact replica of the first voice transposed to a different pitch, because such a reentry may violate the rules that have been established in counterpoint. This structure is very rigid and methodical, making it much easier to simulate with more basic methods like stochastic models. Comparatively, we can expect music like jazz to be more difficult to simulate given the lack of rigid structure.

2. Preliminaries

2.1. Data Collection

To collect a large amount of relevant data, I began by downloading the MIDI files for each of the 48 pieces of music in the Well-Tempered Clavier, BWV 846–893. This includes two sets of 24 preludes and fugues composed in every major and minor key. To fulfill the assumptions of the model it is important that we parse these pieces, removing each of the preludes, to keep only the fugues. One approach is to train the model on one fugue at a time. Another approach is to train the model on two fugues of the same key. Both approaches of data selection will ensure that the data is from a single musical key.

Once the fugues are selected the files must be parsed for the relevant data. The related data can be read using a Python library called Mido. Mido extracts the data into individual "tracks". Each track contains "messages" which store the time signature, each note that is played, the velocity at which the notes are played, and the time elapsed since the previous note was played. In order to keep the state space at a computationally feasible size, I collect only the notes that are played and the time elapsed since the previous note was played.

2.2. Data Structure

The data structure used to represent the stochastic process is a dictionary of dictionaries. In Python, dictionaries are represented as sets of key-value pairs, where a value can be accessed by referencing its unique key. This is advantageous when building a stochastic model from MIDI data because each note in the set of all notes in the piece can be represented as a key in the outer dictionary. Additionally, each inner dictionary can hold key-value pairs that represent a note and its duration. Such keys of the outer dictionary represent the initial states of the stochastic process. The transition states of the stochastic model are represented by both the key and the value of the inner dictionary.

A key will be added to the outer dictionary for each note that is played in the musical piece. By calling and accessing a key one can see the value that is associated with it. In this dictionary of dictionaries, the value associated to each key in the outer dictionary will be another dictionary. Like the outer dictionary, each inner dictionary is also composed of key-value pairs. The keys of the inner dictionaries are the notes that were directly transitioned to by the other notes and their values are the each note's respective duration.

3. The Model

3.1. Defining the State Space

We begin by defining a state space of MIDI notes called N.

$$N = \{0, 2, 4, 5, 7, 9, 11\} \bmod 12$$

$$N^c = \{1, 3, 6, 8, 10\} \bmod 12$$

Furthermore,

$$\forall n \in N, 21 \leq n \leq 108$$

Next we define a sate space of time denoted by τ.

$$\tau = 0 \bmod 125$$

3.2. Building Transition Probability Matrix

With the state space defined, we can analyze a piece or a collection of pieces of music and generate a matrix which represents the frequency of transitions between every combination of tuples in the training set of music.

Let $F_{i,j}$ be a transition frequency matrix generated from the data. $F_{i,j}$ is generated dynamically using the following logic:

1) Read the data into a list, D, of Python objects called Mido(citation) messages.

2) Iterate through D until a unique note n^* is found. (i.e. $n^* \notin F_i$)

3) If a unique note n^* is found, append it to F_i and continue to iterate through the D. label=()

   a) Let all notes directly transitioned to from n^* be called n' with durations d'. Append all vectors (n', d') to the columns of F_j and increment $F_{n^*,(n',d')}$ by 1.

4) Continue iterating through D until no unique notes are found.

A transition probability matrix $P_{i,j}$ can now be constructed by diving elements from $F_{i,j}$ in every row by the sum of their respective rows.

$$F_{i,j} \rightarrow P_{i,j}$$

3.3. Defining the Stochastic Models

From the definitions above, it follows that each vector generated by the stochastic process belongs to the Cartesian product $N \times \tau$. In other words, every vector that is generated by the stochastic process is a tuple containing a note's pitch and its duration.

$$(x_i, t_i^x)_{i \in Z_+} \in N \times \tau$$

Where $(x_i, t_i^x)_{i \in Z_+}$ is generated via the transition probability matrix $P_{i,j}$.

From $(x_i, t_i^x)_{i \in Z_+}$ we have a continuous time processes $(X_t)_{t \in R_+} \in N \cup \{\Delta\}$.

$$(x_i, t_i^x)_{i \in Z_+} \Leftrightarrow (X_t)_{t \in R_+}$$

Also, from $(y_j, t_j^y)_{i \in Z_+}$ we have a continuous time processes $(Y_t)_{t \in R_+} \in N \cup \{\Delta\}$.

$$(y_j, t_j^y)_{i \in Z_+} \Leftrightarrow (Y_t)_{t \in R_+}$$

We have interestingly composed two definitions of essentially the same model to be run side by side. Really, we can think of the model as having two different concurrent time spaces per voice. The key distinction is that one model, a Markov Chain, has discrete time and discrete state space. The other representation of the model, has continuous time and discrete state space. Defining the model in two different ways is advantageous in that it allows us to create rules based on both discrete and continuous time values.

This is helpful because we know that the first voice X will begin playing notes before the second voice Y. Using discrete time is not a problem when making rules within a single voice, because our definition $(x_i, t_i^x)_{i \in Z_+}$ lets us view the current state of the stochastic process after $i \in Z_+$ notes have elapsed, as well as what the possible states are for (x_{i+1}, t_{i+1}^x).

However, if we want to make a restriction on the possible notes another voice Y can generate at the same time as X in the piece, the discrete number of notes which have elapsed will represent different points in the output for each voice. Given so, we can represent the time as a continuous space of the number of milliseconds which have elapsed in the piece. Therefore, at time $t \in R_+$, we know $(X_t)_{t \in R_+}$ and $(Y_t)_{t \in R_+}$ are at the same location in the output vector. Thus, we can check what X_t is and restrict the possible values the process can move to from Y_{t-1}.

3.4. Restrictions on x_i

3.4.1. Leaps can not be larger than a sixth.

$$|x_{i+1} - x_i| \leq 9$$

3.4.2. Consecutive alternations between two notes within a step apart are not permitted. If:

$$x_i = x_{i+2} \text{ and } (|x_{i+1} - x_i| = 1 \text{ or } |x_{i+1} - x_i| = 2)$$

Then:

$$x_{i+3} \neq x_{i+1}$$

3.4.3. Leaps greater than a third must be succeeded by step-wise motion in the opposite direction of the leap. If an upward leap occurs:

$$x_{i+1} - x_i \geq 4$$

Then the following note must move a step down in the state space:

$$x_{i+2} = \begin{cases} x_{i+1} - 2, & x_{i+1} - 2 \in N \\ x_{i+1} - 1, & x_{i+1} - 2 \notin N \end{cases}$$

If a downward leap occurs:

$$x_{i+1} - x_i \leq 4$$

Then the following note must move a step up in the state space:

$$x_{i+2} = \begin{cases} x_{i+1} + 2, & x_{i+1} + 2 \in N \\ x_{i+1} + 1, & x_{i+1} + 2 \notin N \end{cases}$$

3.4.4. The piece should conclude following a resolving note. It is important that the model is conscious of when to end the simulation of the stochastic process. In order to ensure that the simulation ends on a note which sounds like the piece is ending, I created a rule to check for any form of "resolving" note.

Once a specified length of notes has been generated, the model will begin checking if the piece "wants" to end. In this case, after 16 notes generated the model begins checking for a resolving note.

- If $x_i = 67$ let the final note be a random variable following the discrete uniform distribution over the set {60, 72}.
- If $x_i = 79$ let the final note be a random variable following the discrete uniform distribution over the set {72, 84}.
- If $x_i = 71$ the final note is 72.
- If $x_i = 83$ the final note is 84.

3.4.5. The piece should conclude on a downbeat. If $t = 0 \mod 1000$
Then the following note is either the first or the third beat of the measure.

3.5. Joint Restrictions on X_t and Y_j

3.5.1. Parallel octaves are not permitted.

$$\forall t, \text{ s.t. } X_t = Y_t + 12n \text{ for some } n \in Z$$

$$\text{Let } i \text{ be s.t. } t \in [t_i^X, t_{i+1}^X) \text{ and let } j \text{ be s.t. } t \in [t_j^Y, t_{j+1}^Y).$$

$$\text{Additionally, suppose } X_{i+1} \neq X_i.$$

$$\Downarrow$$

$$Y_{j+1} \neq X_{i+1} (\text{mod } 12)$$

3.5.2. Parallel fifths are not permitted.

$$\forall t, \text{ s.t. } X_t = Y_t \pm 7 (\text{mod } 12) \text{ for some } n \in Z$$

$$\text{Let } i \text{ be s.t. } t \in [t_i^X, t_{i+1}^X) \text{ and let } j \text{ be s.t. } t \in [t_j^Y, t_{j+1}^Y).$$

$$\text{Additionally, suppose } X_{i+1} \neq X_i.$$

$$\Downarrow$$

$$Y_{j+1} \neq X_{i+1} \pm 7 (\text{mod } 12)$$

4. Evaluating Results

Evaluating the output of the model is somewhat subjective since there is no definite way to judge if a piece of music is good. Someone who is more experience with music theory could perhaps compare the sheet music of the output piece to the required structure of a fugue and judge how well the model meets the expected requirements. For the sake of the class project, it is fine to simply listen and make a decision if the music sounds pleasing or not.

It turns out that the model is quite good at generating the subject of the fugue. The subject of the fugue nearly always sounds pleasing on its own. However, the model begins to struggle when generating the countersubject. This deficiency in the model is directly from a lack of music theory knowledge on my part. In order to improve the model's ability to compose a countersubject, it would be necessary to increase the number of joint restrictions on X_t and Y_t to closer follow the musical rules of counterpoint. Regardless, the model does quite a good job at producing pleasing music when compared to current papers using similar methods for generating music via stochastic processes. Specifically, I was impressed with my model's ability to sound much less random and for each piece it generates to have it's own defined style compared to other papers which used stochastic models.

5. Conclusion and Future Work

I am very satisfied with the results of this project and the ability for my model to generate musically pleasing subjects and countersubjects of a fugue. Simply by training the model on Bach's previous work and simulating a stochastic process we can generate unique music in a style that is completely contrary to that of humans. The resulting music may not be the best, but the project highlights a unique difference in the way that humans and machines formulate and solve problems. It is very interesting to compare the processes of humans and machines in the creation of art.

While this project did a good job of displaying the difference in content generation style, there are certainly other methods which may have yielded in better results in terms of the final music. In the future, if I were to approach this problem differently I would use deep learning. This paper has revealed that generating music via stochastic processes can be done to a certain level, but it becomes inefficient in preserving long lasting themes throughout a musical piece. Approaching this problem with a deep learning model would definitely result in music which sounds more fluid and interconnected. In conclusion, we see that stochastic methods for music generation can create very promising results, but only opens the door for more sophisticated models to truly produce meaningful results. This findings seem to mirror the results found when applying stochastic models to other domains such as procedural text generation and other content generation tasks.

References

[1] Ames, C. (1989). The Markov Process as a Compositional Model: A Survey and Tutorial. Leonardo 22(2), 175-187. https://www.muse.jhu.edu/article/602251.

[2] Briot, J. P., Hadjeres, G., & Pachet, F. D. (2017). Deep learning techniques for music generation–a survey. arXiv preprint arXiv:1709.01620.

[3] Huang, C. Z. A., Cooijmans, T., Roberts, A., Courville, A., & Eck, D. (2019). Counterpoint by convolution. arXiv preprint arXiv:1903.07227.

[4] José M. Iñesta, Darrell Conklin & Rafael Ramírez (2016) Machine learning and music generation, Journal of Mathematics and Music, 10:2, 87-91, DOI: 10.1080/17459737.2016.1216369

[5] Volchenkov, D., & Dawin, J. R. (2012). Musical Markov Chains. In International Journal of Modern Physics: Conference Series (Vol. 16, pp. 116-135). World Scientific Publishing Company.