Music Composition Using Markov Chain Model

Research Query

How can a computer compose music that sounds good, and if so, what is the most efficient way it can be done?

Abstract

Music composition offers interesting challenges when being applied to computers or other machines. Musical composition can be approached by using a variety of methods. Some of these paths include using a probabilistic analysis and utilizing neural networks that can be used to feed data to a computer. This data being fed into the computer effectively trains the computers and hones its skill in a certain skill set. In my case this skill set is the ability to compose music. I intend to use music composition as a means of comparing the growth of a human's skills compared to that of a machine's or a computer. I want this computer program to be used, not only by people who have a serious goal of composing unique music, but also people who are looking to explore their creative strong points to see what they can produce. This program can be incorporated into a psychological method that helps certain patients recover from traumatic and difficult situations. I believe this program has the power to revolutionize the way we create music.

Background

Since a young age, music has been something that intrigued me in its different styles and the different instruments that are used to compose and create it. All of these different styles invoke different sensations and emotions. I have also been programming in Python for about two years and I've explored many different libraries and versions of the language. I look at programming as the way a person expresses his or her unique approach to a problem. Then the thought came to me; " What if I were to combine these two aspects, music and programming, to create a program that would create music on its own. With my knowledge of music theory I have gathered through playing violin and piano, I began to explore many different approaches that would allow a computer program to create music such as ANNs (Artificial Neural Networks), a probabilistic analysis, algorithm based composition, etc. I eventually opted to go for a probabilistic analysis because it was a unique approach to this query. I will explain in another slide why I didn't choose to use ANNs.

Proposed Approach

Music can be composed by humans by taking a melody that one may like and then improvising upon it. Computers can be programmed to compose music in this way. There are many possible approaches to composing music such as: ANNs (Artificial Neural Networks), probabilistic analysis, and algorithmic composition. All three of these methods have their pros and cons but ultimately, a probabilistic analysis seems to be the a more efficient method. ANNs are inconvenient and algorithmic composition can restrict creativity. In a later section, all three of these methods will be compared. The probabilistic approach in this research is based upon Markov Chain Model. Essentially, the Markov Chain takes a "training" melody in as input and uses it to first "train" an empty 12 by 12 Adjacency Matrix. After being "trained", the Adjacency Matrix has various weightages of different note transitions within the "training" melody/melodies. The Markov Chain allows a transition from one note to another in the order of probabilities i.e. for a given note, the note with the greatest weightage is chosen as a next note. This process continues to repeat as long as the user specified length of the phrase is created. The python program, upon user inputs, can train the matrix with existing melodies or use an existing trained matrix to generate a melodic phrase. This phase is then improvised with various musical transformations to create more phases. Such improvisations are currently performed with randomness but such choices can also employ the next level of Markov Chain model.

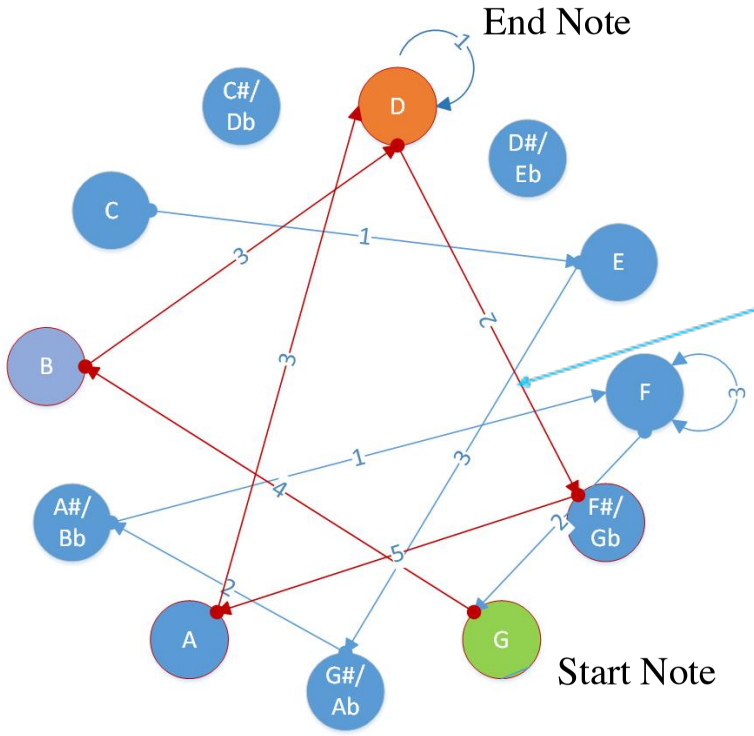
Mathematics and Research Background

To understand how my project functions better, I will explain the mathematics behind some of the different aspects of my program. To start off, a Markov Chain, is a sequence of probability based choices that only depends on the current state and not any of the states that preceded it. They deal with the transitions from one state to another based on the probability of making a given transition. In my case, the state referred to is any note being played. In my program, before a sequence of different transitions or notes can be created, the different probabilities or weightages of varying transitions is organized into a matrix called a transition or adjacency matrix. My program first takes in an input melody and takes note of each of the transitions between consecutive notes in the melody. These transitions are then represented by the matrix by taking a given note's relative position within an octave as an integer from 0 to 11 (0 being the note C and 11 being the note B). This first number given will determine the notes row number within the matrix. This octave can be chosen arbitrarily but an octave starting from C is the most simplistic. Then the program takes account of the next note in the melody's sequence and finds its relative position within a C octave. Now you take the relative row and column positions of these two consecutive notes and increment that position by one. This note transition now has a weightage of one. The program continues through the rest of the melody, making the previous column element the new row element. Another way of explaining this is that the initial output values will become the next input values. In this case, the rows represents the inputs and the columns represent the outputs. These note transitions can also be represented as a graph: Nodes or vertices and edges representing note transitions. For a given row element, the columns element that has the greatest weightage will be the next note. This describes the functionality of the Markov Chain. Going back to adjacency matrices; these are matrices, in this case, that represent a directed cyclic graph. This matrix is not symmetrical because if a note transition is from note 2 (D) to note 0 (C) it is not the same thing as going from note 0 (C) to note 2 (D).

Mathematical Model For Melody Generation

- Set of notes: N = { Set of notes}
- E = {A pair of notes representing not
- Directed Cyclic Graph (DCG) is repre
- DCG with weights is modeled using A where w represents weight of transi

Note transitions using Directed Cyclic Graph (DCG) – like Markov chain model



High Probability Note transitions denote an example melodic phrase

Adjacency Matrix Corresponding to Directed Cyclic Graph

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 3 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 \\
0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Comparison Between Artificial Neural Networks, Probabilistic Analysis and Algorithmic Composition

Artificial Neural Networks are Neural Networks that are created to mimic the way the Neural Networks in human brains function. This method however has two major flaws. For one, it requires obscene amounts of data to make it higher functioning than a human. The second flaw, being a result of the first, is that to be able to train these neural networks, you need access to huge servers that have incredible computational capabilities. A probabilistic analysis that is used to predict a certain outcome based on the input data. This type of method is quite efficient because in comparison to ANNs, it can start creating music based off the base melodies right away because the Markov Chain allows the program to choose note transitions that have the greatest weightage or probability. Therefore there is no real "training" required with a probabilistic approach to this query. Lastly, algorithmic composition is the composition of music by the means of an algorithm. To musicians, this method may seem quite strange, because there is no set algorithm to composing music. If there were, then all the music in the world would sound quite similar and it would be boring.

Tools for programming

For this project, I decided to use Python with a library called MIDI-Util. This library can be used within Python to create MIDI files as an output. I can play these MIDI files outputted through an app called Garageband that is already pre-installed with the Mac. I decided to use Python for my project because I've been using it for two years now and is, personally, the most familiar and convenient language for me. Another reason I chose Python is that there are many libraries that are available for it where as some other languages may not have as many options. Also, I may have to learn a lot more to get my coding ability in another language to the level that my ability in Python is currently. Before I opted for using MIDI-Util I was exploring some entirely different Python based languages such as Jython. With MIDI-Util, I was able to output the MIDI files containing the music created quite easily. Convenience was also a strong point of this library because I didn't have to keep on creating new files to store the music that was created. The old music that preexisted on the MIDI file would be overwritten with the new music. I have extensively used the Indian Classical Music melodies with their ascending and descending scale notes. I was fascinated by the fact that there are melodies suggested for various times of the day. I did not want to necessarily use that observation to restrict the creativity of my program but it can be explored later.

Application of my work

My goal with this project is to make a simple program available to serious composers who are looking to create unique music by using some training melodies. They can explore their favorite melodies and see what different types of music they are able to produce with those melodies using my program. In fact, anyone can create music with help of this program and get inspired. I also want this to be made available to the general public, because in some cases, music may not be readily accessible to them. With my program, they can create their own music that can be personalized to their individual's taste. Music's applications is not only limited to entertainment, but it can also be used to heal patients who may be suffering from difficult times. These patients or the patients' aids can easily use this program to create music that will soothe themselves or their patients so that they can heal and recover from any difficult situations they may be going through. I also aim to be able to make an app that has the same functionality as the program running on a computer. This would increase the ease of portability and would be much more convenient to use. Simplicity of the approach and reduced computing requirements can allow very simple Raspberry Pi like device without any connectivity to create music.

Observations & Results

During the beginning stages of my project, I explored the idea of using neural networks as a possible medium for creating music. After doing some further research into it, I realized that neural networks are very cumbersome and difficult to work with. From there onwards, I shifted over to a completely probabilistic approach to creating music with my program. This method is very easy to work because it doesn't require a huge amount of data as input. It can in fact work well with only one base melody as an input as well as a large number of melodies. However, with a small number of inputs, this methods ran into some problems such as getting stuck into a loop of notes. This is basically saying that, if I were to graph the note transitions of a melody, there would be some sections of the graph that were in a loop, of say, two or three notes. To work around these so-called "self loops", I had to zero out the diagonal on the trained matrix. This is because if the row element and the column element for the greatest element happens to be the same, it will continuously loop only on those elements because the initial columns element will become the new row element. After I was able to work out this problem in the creation of a single phrase, I added different improvisations to create the final melody product. I also added ornamentation of notes by algorithmically modulating the volume. The Midiutil library did not allow me any more ornamentation than this. I have tried various Ragas (Indian Classical Melodies) to train the matrix to generate music. Due to probabilistic nature of the algorithm, there is always a slight variation of music that is created i.e. single phrase as well as phrase improvisations.

Conclusion

My original thought of using some sort of algorithmic composition technique or neural networks didn't seem like the right route to take, after some research and consideration of these techniques' effectiveness. Neural Networks require a huge amount of compute power and data storage that wasn't suitable to be used in a portable application of my program. I also decided not to use an algorithmic based approach to composing music through my program because of the creativity that it may restrict. After conducting research on my initial ideas and using my intuition I developed on using a probabilistic approach which I later on found similar to Markov Chain Model. Although this method may lack in some places where the neural networks may have performed better, it is definitely more efficient. I strive to continue working on this project in the future and improving upon its versatility and performance.

A note of Gratitude:

I really want to thank my music teachers Ms. Rei(Violin), Mrs. Robin(Piano) and Mrs. Moon(School String Orchestra). I also want thank my science teacher Ms. Mohler for encouraging me in STEM. Lastly, I would like to thank my dad for introducing me to programming and inspiring me to remain passionate in everything I do.

References

- Music and Artificial Intelligence (1993) by Chris Dobrian: music.arts.uci.edu/dobrian/CD.music.ai.htm
- Neural Networks by Christos Stergiou and Dimitrios Siganos: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html
- AI Methods in Algorithmic Composition: A Comprehensive Survey: https://www.jair.org/media/3908/live-3908-7454-jair.pdf
- Making Music with AI: Some examples: http://www.iiia.csic.es/files/pdfs/1265.pdf
- AI and Music Composition to Expressive Performance: http://www.iiia.csic.es/~mantaras/AIMag23-03-006.pdf
- Cognitive Computing by Peter Fingar
- Music for Geeks & Nerds by Pedro Kroger
- MIDIUtil: https://github.com/duggan/midiutil
- PyBrain: http://pybrain.org/
- Digital Music Programming 2: Markov Chains: http://peabody.sapp.org/class/dmp2/lab/markov1/
- Markov Chains: https://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/Chapter11.pdf

Future Directions

- Create an app for both Android and IOS so that the program can be personalized and portable
- If enough resources and computational ability available, I will implement ANNs to explore music generation.
- If I figure out the proper method of doing so, I will harmonize my melodies by adding left hand notes or bass clef notes.
- Collaborate with music and computer science professors to learn more about their perspective topics to further the complexity of the program.
- Make same program in different programming language to measure the speed of music generation.
- Currently my program only takes txt files for melody inputs but I like to devise a way to sample music and extract melodies.
- I also plan to extend Markov chain model to phrase improvisations.