Machine Learning and Music Composition

Daniel Woeste
Division of Math and Sicence
University of Minnesota, Morris
Morris, Minnesota, USA 56267
woest015@morris.umn.edu

ABSTRACT

Throughout history, musical composition has been thought of as limited to those gifted with a higher insight. As computers have risen in prominence, they have also asserted themselves as an almost necessary tool used in the field of music. Programs such as Finale and Sibelius are used by a composer for music writing and recording. Recent advancements in machine learning technologies has possibly provided a new way for computers to be used in the field of music, as well as possibly bringing music composition to the masses. Machine learning offers the unique possibility of having a program generate the song, leaving the user to edit or pick the song closest to their musical tastes.

Keywords

Random Forests, Markov Chains, Neural Networks, Machine Learning

1. INTRODUCTION

Music has become an almost daily part of our lives, whether it is through background grounds in movies, commercials, or even listening to it for the sake of listening to a song. For many of us, our interactions with music are mostly a passive experience, listening rather than contributing. Recent advancements in machine learning may give people the chance to produce music tailored to their personal music tastes. These programs can be used as musical assistance to enhance or fill in gaps in musical knowledge. They can also be used to autonomously generate music without much input from the end user.

2. BACKGROUND

We use this section to introduce important terminology used to understand concepts throughout the paper.

2.1 Music

In this section we explain the terminology that is important to the music side of the paper.

2.1.1 Melody

A melodic progression is the intervall traveled when changing from one note to another. These intervals are broken down into whole steps and half steps. Combining several progressions together produces a melody. A melody is a combination of pitches, rhythms and note duration, considered pleasing to the ear of the listener. At any point during a piece of music, the most important part played is considered the melody.

Chords are when several notes are being played at the same time. This can have a pleasing effect known as harmony, otherwise known as consonace, or a clashing effect known as dissonance. While most songs tend to be harmonic, dissonance gives the music drive and forward motion. Without dissonance, music lacks a sense of progression. Music is given a sense of forward motion by the resolution from dissonant chords to harmonic chords. Dissonance, in a simple sense, gives the music a direct path to follow, moving from a clash to peace and harmony

Musical texture also plays an important role in the style of music generated by machine learning programs. Homophonic and polyphonic textures are two types of music textures, which will be discussed in greater detail later. A homophonic texture is a piece of music containing one melody played at a time. The homophonic melody can be played by more than one instrument simultaneously, but there is no competing melody being played at the same moment. Polyphonic means there is more than one melodic lines being played simultaneously. Polyphonic music is a more common style to jazz and contemporary pieces. Simpler classical music and pop music tend to favor homophonic texture due to its easier to understand nature.

2.2 General Framework

All of the methods described in this paper share a similar framework that would be useful to understand. The commonality between all of the methods is that every note is generated in relation to a sequence of previous notes. Sometimes the new note is only dependant on a single previous note, while in other circumstances it depends on a larger set of notes. This method of relying on previous notes allows the program to generate new notes that fit well with the current melodic progression. Similar to how a sentence is formed, each new word is dependent on the previous word to make a coherent thought. A coherent melody is created by using notes that blend well with the overall theme already being expressed.

2.3 Machine Learning

Machine learning, a subcategory of artificial intelligence, is an area of computer science that focuses on programs that are able to systematically change their behavior. These

UMM CSci Senior Seminar Conference, November 2017 Morris, MN.

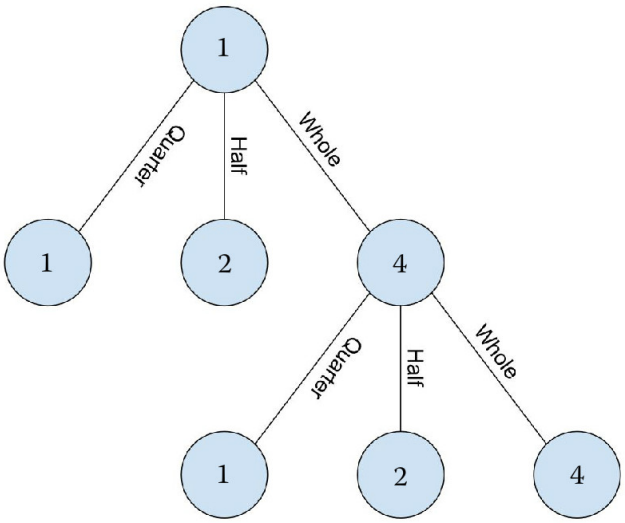

Figure 1: Decision tree example

learning systems are able to make alterations, or mutations, during a time of training. Alterations to the system are made by using existing musical melodies. These melodies are used as a "final answer" that the program can use to score itself against. Using multiple passes over the training melodies, the program slowly approaches a correct answer by comparing new changes to the answer. After an unknown period of time, the program is able to generate melodies that are similar in style, but not identical to the training data set. Using a machine learning method frees the system from possible biases of the programmer. This style of data driven learning allows machine learning programs to be well suited for applications such as self-driving cars, text prediction, online recommendations, and even the field of music composition.

In the remainder of this section we will describe other basics of machine learing including *decision trees*, *overfitting*, and *finite automata*.

2.3.1 Decision Trees

Decision trees are structures that represent choices and their possible outcomes in a branching style, see 1. Every node on a decision tree represents a choice, such as a note length to follow the previous note. In Figure 1 the nodes are represented by the blue circles representing a current note. The branches out from the nodes represent the possible choices that could be made for note lengths. The decision tree branches out to include all possible outcomes for every choice on the tree. Starting from the top of the tree would place us at the start of a rhythmic pattern. Progressing through the tree also progresses the rhythmic pattern note by note.

2.3.2 Overfitting

A possible side effect with decision trees and all types of machine learning is overfitting. Machine learning algorithms and programs, such as decision trees, develop overfitting because the algorithm itself becomes too complex. Through multiple iterations of training, the algorithm develops too many parameters that are overly specific to the training data set. When this happens the program begins to learn and integrate small fluctuations in the data that are inconsequential to the actual outcome, instead of learning the overall "big picture." A learning system that has developed overfitting cannot be accurately run on data that is not the training data. Overfitting can be prevented by taking a system that has already been trained, and manually altering the program to remove the level of detail that it has. Musically this means that a program developed for composition would only be able to produce the exactly, or almost identical, pieces of music to what it was trained on. It would have failed to grasp the style of music, but instead would only be able to repeat exactly the music it has learned.

2.3.3 Finite Automata

Finite automata, otherwise known as finite state machines, are representations of an abstract computational model, see Figure 2. These state machines can only exist in one position at a time. Finite state machines represent a stochastic model of the data. Every possible move is represented by a probability, but the path through the automata is chosen at random, following the probabilities. In Figure 2, the automata starts on the node with negative four, also denotes by the arrow head. From there it transitions to the next possible state, the node with a value of one, or a quarter note. It will continue these transistion until it reaches the last node with a value of four, the node with a double circle.These models work well for music. For instance, a developing music rhythm can be thought of stepping through an automata, giving each new note a specific chance to be a given value such as quarter or half note.

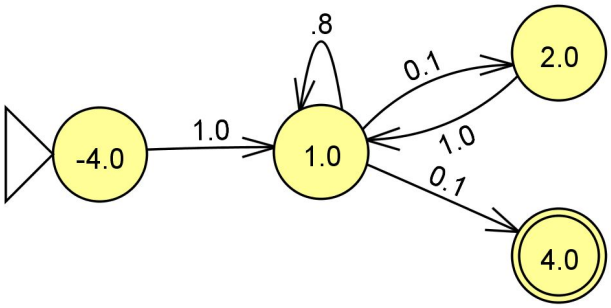

Figure 2: Finite automata detailing the Markov Generated in Automatic Composition of Music with Methods of Computational Intelligence

3. METHODS

In this section, we describe three separate subcategories of machine learning *random forests*, *markov chains*, and *neural networks*.

First, we discuss the use of random forests. Random forests operate by generating and altering a large collection of decision trees. The generated output of a random forests is formed through a method of consensus. Every decision tree in the forest will have it's own answer, the final output will be the most common answer generated. As with all machine learning a set of amount of time is required to teach the algorithm to properly generate answers. For the case of random forests, the training data is broken up into small

subsections. Each subsection of data will overlap with one or more of the other subsections of data. Then a decision tree is constructed for each subsection of the training data. The overlaps in the subsections of data are a way to try to combat overfitting. Using overlaps allows multiple decision trees to cover every possible choice. Not leaving a possible choice up to a single decision tree means that small fluctuations in the data are not made so important that they skew the outputs. Individual trees may have overfitting in a random forest, but to the nature of using consensus to produce an answer, those trees become outliers. For this style of machine learning we will discuss a program known as ALYSIA in section 4.2.

In comparison to random forests, markov chains function differently. Markov chains use a state machine and stochastic models to predict the next note. As demonstrated in Figure 1, a simple finite automata generates the note length. The values 1, 2, and 4 represent quarter, half, and whole notes respectively. The value of -4 is a starting state that holds no value musically, other than a place to start the song. The automata visualizes the way the markov chain decides the next note or value will be. In the terms of music, this means for each state in the automata, it has a percentage chance of moving to any of the other nodes in the automata. In this case, the first note will always be a quarter note with an 80% chance of having a repeated quarter note, with only a 10% chance of a subsequent half note, and a 10% chance of ending in a whole note.

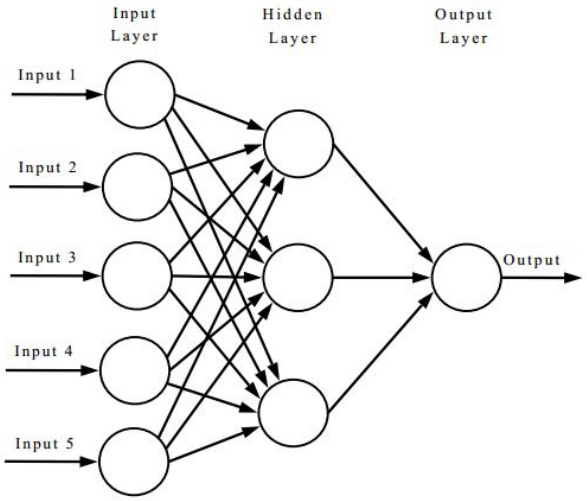

Figure 3: example of a neural network

Change picture to a more simple neural network

Another approach to computer generation of music is neural networks. They are a style of machine learning that is based loosely upon the functions of the human brain. This is accomplished by generating many highly interconnected nodes (neurons) that are working together to solve a single problem. Similar to random forests, neural networks are trained upon a given set of data to produce a network that will generate a melody. Unlike the previous two methods, neural networks are almost always deterministic. This means for the same input given, a neural network will always have the same output. This differs fundamentally from the other methods which rely heavily on elements of chance to determine an outcome.

4. METHODS AND MUSIC

In this section, we will discuss how the previously stated methods can be applied to the composition of music with specific examples.

4.1 Markov Chains

The program based upon markov chains in Klinger et al produces melodic lines by running the produced music through multiple iterations of the program. Each iteration of the music focuses on the generation of a new layer of music. The first iteration of the music is used to generate a simple rhythmic pattern for the rest of the program to alter. Figure 2 shows a simplistic version of a markov chain generated to produce simplistic patterns of only quarter and half notes, with a melody that always ends in a whole note.

The resulting output from the rhythmic markov chain is then piped into a new markov chain that sets the pitches from the rhythm. To accomplish this, the program could use two different styles of markov chains. They are labeled as an absolute or a relative markov chain in the Klinger paper. The difference between the two styles of markov chains is that for the absolute chain, it functions purely by choosing a note value to work with. In other words, it chooses one of the available notes in the octave (CDEFGABC) to use as the next note value. Conversely the relative markov chain uses interval length, or the distance from one pitch to the next to determine the next note. So instead of generating a transition from A to B, the relative markov chain would output a whole-step transition. The benefit of the relative style is that it is less work to transpose the music from one key to another; everything is based upon step distance rather than absolute note values.

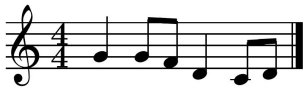

Figure 4: A single bar melody produced after the second iteration of markov chains

The last step to the markov model is post processing. The post processing of the music is a step that takes the already generated simple melody and then runs it through a last markov chain to complicate the melody. The complication of the melody is an attempt to make the music more interesting to the listener. The post processing can be run as many times as the user feels necessary. The post processing can achieve these alterations through several methods: note splitting, one-point mutation, and recombination.

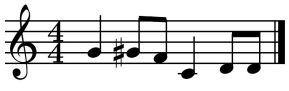

Figure 5: Previous melody after being put through one-point mutation

One-point mutation is process of taking the melody and

traversing over it with a low chance for each note to change pitch value.

Note splitting is similar to the one-point notation in the fact that it is traversal over the melody, but instead of the changing the pitch value of the note, it splits the note into an equal value of shorter duration notes. For example note splitting could change a quarter note into a run of sixteenth notes.

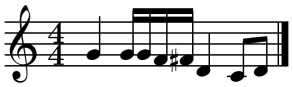

Figure 6: Previous melody after being put through a note-splitting mutation

Recombination works in an entirely different way than the previous two methods. The concept of recombination is taking two original parent melodies, generated from two different runs of the previously stated program, and combining them into a single child melody.

4.2 ALYSIA

ALYSIA, Automated LYrical Song-writing Application, is a program that uses random forests to generate songs and melodies for a given set of pop lyrics. ALYSIA contains two different random forests that work in tandem: one to produce rhythm, the other to produce pitch values. [ALYSIA] ALYSIA has mainly been used to write new melodies for an already existing pop song. It takes in both a set of lyrics and the already existing accompaniment, background music, to to the pop song. ALYSIA then outputs several possible new melodies for the song. Random forests can provide some inherent benefits over Markov chains. Even though both models work from previous notes, and both predict what the next note should be, random forests and markov chains function based on different models. Markov chains offer only a statistical probability of what the next note should be. While random forests include a reasoning behind what the next note will be.

ALYSIA is able to encode a reason for the next note; instead of relying on entirely random chance for the next note, it uses a system known as feature extraction to determine the shape of the background music that the ALYSIA is trying to compose music for. The feature extractions includes parameters such as key signature, time signature, note duration, and many others. It additionally includes feature extraction for the lyrics that ALYSIA is trying to match. The feature extraction includes parameters such as, word frequency, and syllable number (the syllable position within a word). This function allows ALYSIA to examine the previous few notes of the melody, if not the entire melody, in order to predict what the best next note will be.

A unique feature of the ALYSIA method is that it allows for a co-creative work flow. The ALYSIA program was developed in a way that would allow it to either fill in for missing musical knowledge of the user, or to output a multitude of melodic outputs that the user can choose from.

elaborate co creative

4.3 LSTM and RNN

Some neural networks that were used to generate an entirely different style of music: polyphonic music. Polyphonic music is a more complicated style of music, containing multiple melody lines played at once. This adds an additional level of complication to what the neural network has to keep track of due to complexities of music in relation to music patterns and harmonic relations between the multiple lines of music. The paper by Daniel D. Johnson covers a type of neural networks kown as *recurrent neural networks* (RNNs) based upon the *long short-term memory* system (LSTM).

A recurrent neural network (RNN) is similar to the neural networks described in section 3. The difference between the two is that instead of having several layers of nodes that always progress from one layer to next, in a forward motion, recurrent networks allow for backwards travel along the network, meaning that as a node completes an operation, the transition, as indicated by the arrow pointing to the next node, might not point at a node in the next layer. It can transition back to the node itself or to a node in the previous layers.

Long short-term memory (LSTM) is a specific alteration to RNNs that allows the program to store dependencies, that can be considered as previous context, throughout iterations of the network. This type of model is particularly well suited for the generation of music because every note is dependent upon previous notes to make a coherent melody. The long term memory of the network determines the context of the music. Using the context the RNN is able to produce the subsequent notes. For example, an RNN with LSTM could remember what key the music is in, for instance the music is in C major, and it would additionally be able to remember key musical features such an ascending melodic line. With this knowledge the network could determine that the appropriate following action might be a descending musical line.

5. EVALUATION

5.1 Markov chains

The markov method described above used several different methods to evaluate the output of their program. We will be going into detail about two of these evaluation methods *Feature Extraction* and *Decision Trees*.

The feature extraction used by the markov chain method is used to look for musical patterns with in the music that may lead to an understanding of the quality of the generated melody. Two examples of the used feature extraction include functions called *Rests on Downbeats* and *Repeated Pitch*. Both of these two features are trying to calculate a ratio to assign a numerical value to the melody [**?**].

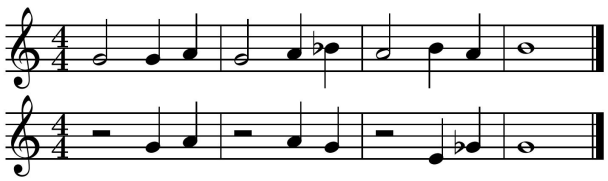

Figure 7: Rests on Downbeats

Rests on Downbeats is a function that calculates the ratio between the number of bars of music that begin with silence,

a rest, and the number of bars that begin with a note. The function is trying to determine whether or not there is too much blank space within the melody made by the markov chains. In Figure 7, there are two example of melodies that were generated through the markov chain method. The first example, the top line, when processed through the Rests on Downbeats has a ratio of 0 because every bar starts with a note. The second example, the second line in Figure 7, has a ratio of .75 because three quarters of the bars start with rests.

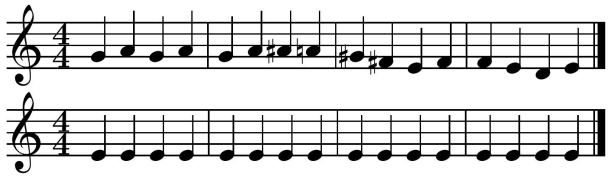

Figure 8: Repeated Pitch

The next function what we will be talking about is the called repeated pitch. This feature extraction is a function that is looking for stagnation within the music. It calculates the ratio between the number of notes with interval 0 over the number of notes in the piece [**?**].

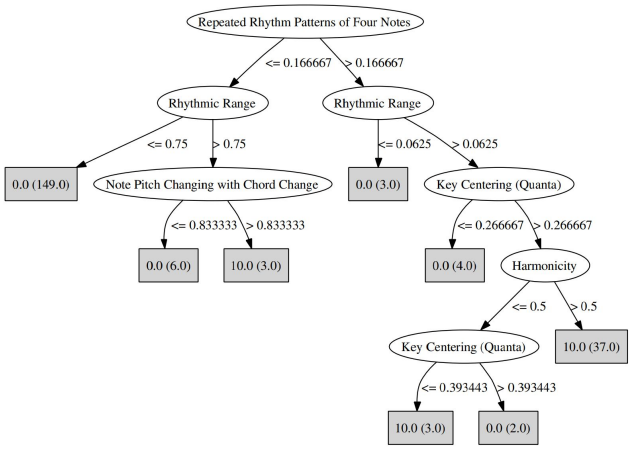

Figure 9: Decision Tree

Klinger et al. took the feature extraction one step further than what was described above [**?**]. After having made all of the functions to calculate values for the feature extraction, they then began to build decision trees using the feature extractions as the tests that the tree could ask.

6. RESULTS

6.1 Citations

Acknowledgments

Fix the section heads because currently they are not organized in a way that makes sense.

fix placement of the decision tree example because aahh!

....add citations and get bib working