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

1. INTRODUCTION

2. BACKGROUND

We use this section to introduce important terminology for the rest of 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*

good music definitions, need to figure out how to work them into a coherent paragraph or 2, this is super clunky to me currently

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.

This may not seem very important now, but it will be important later when explaining how one of the programs tried to evaluate music automatically.

Chords used during music are when several notes are being played at the same time. This can have a pleasing affect known as harmony, otherwise known as consonace, or a displeasing effect known as dissonance. While most songs tend to be harmonic, dissonance gives the music drive and forward motion.

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 line 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 cache of previous notes. Meaning that every note is dependant upon a set of notes before it, whether it is only the previous note or a larger set of notes. This allows the program to determine a melodic progression for the music.

2.3 Machine Learning

Machine learning, a subcategory of artificial intelligence, is an area of computer science that focuses on programs able to systematically change, otherwise known as mutations, themselves based upon a learning period. These programs are designed to alter themselves without the need for human input, making them beneficial because they are not bound to a specific set of instructions. Instead, during the learning period, the programs are able to change themselves to best fit however the data demands that they work. This is the learning part of the program. The program will slightly alter how it interacts with the data, such as changing an equation or possibly an input. After the mutations, the program will iterate over the data again, if the output is considered better the changes are kept. 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 complex field of music.

2.3.1 *Decision Trees*

Decision trees are a structure that represent the tests and outcome of an algorithm. Every node on the decision tree represents a possible test or choice that an algorithm can make. The resulting branches coming from that node represent all of the possible outcomes of that test. This can easily be represented by flipping a coin multiple times in a row. Every time the coin is flipped, a node is created in the decision tree. In this instance, the two nodes for the coin flip are heads and tails. For every subsequent coin flip, a new set of nodes and leaves are generated. As a result the decision tree contains all possible outcomes for the Nth coin flip.

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

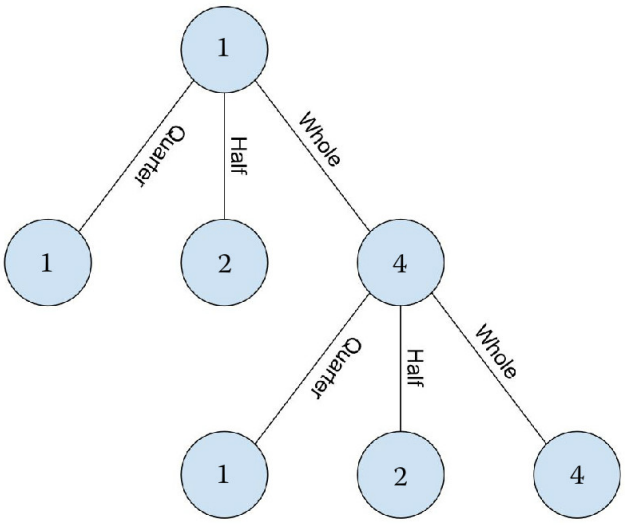

Figure 1: Decision tree example

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. Meaning that the program that has developed is not able to accurately work on data that it was not trained on. Overfitting results in a loss of accuracy due to program being so complex that it cannot accurately predict new results.

2.4 Finite Automata

Finite automata, otherwise known as finite state machines, are representations of a an abstract computational model that can exist in only one state at a time. The transition from one state to another is caused by external inputs to finite automata. Every finite automata has a start state, denoted by the arrow head, and an end state, denoted by the double circle. The finite automata listen below deviates from these notations, the start state is the state with the value of negative four, and the end state is the state with the value of four. The values one, two, and four represent quarter notes, half notes, and whole ntoes respectively.

3. METHODS

In this section, we will be describing three separate subcategories of machine learning. The three subcategories of machine learning described are *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 the random forest results from the average answer of all decision trees in the forest. 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 forest, each decision tree is made on a small part of the data set, with overlap among all of the trees. This overlap in the data that the trees are built on helps to reduce over fitting of the trees to the data. When it comes to music, a program known as ALYSIA, uses two random forests to generate music from a given set of lyrics.

In comparison to random forests, markov chains function differently. Markov chains use a state machine and a statistical models to predict the next value. As demonstrated in Figure 1, a simple finite automata generates the note length, but not pitch values for a song. 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 uses probabilistic determination to predict what 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 be a quarter note with an 80% chance of having a repeated quarter note, with only a 10% chance of a subsequent half note.

introduce the word trained instead of generated to make sense with the neural networks paragraph

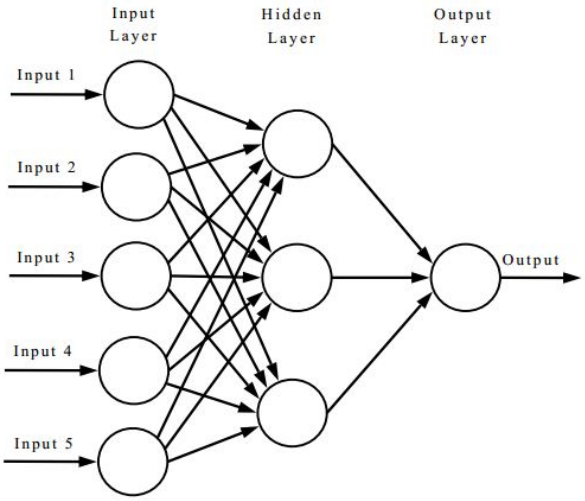

Figure 2: example of a neural network

Another approach to computer generation of music is neural networks, which is a deterministic style of algorithm. They are a style of machine learning that is based 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 produce a favorable. Unlike the previous two methods, neural networks are almost always determinisi. 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

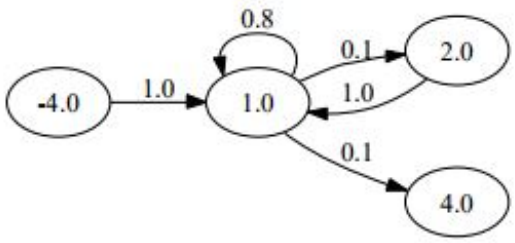

Figure 3: Finite automata detailing the Markov Generated in Automatic Composition of Music with Methods of Computational Intelligence

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

4.1 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] Random forests can provide some inherent benefits over Markov chains. Even though both models work from a cache of 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 to what the next note should 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 function known as feature extraction to determine the shape of the background music that the ALYSIA is trying to compose music to. 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 even word vowel strength (value extracted from CMUDict v0.07 database.) 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.

talk about the co creative nature of ALYSIA mentioned in the paper

4.2 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 several types of neural networks including *recurrent neural networks* (RNN) based upon the *long short-term memory* system (LSTM).

A recurrent neural network is similar to the neural networks described above. 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. This means 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 is a specific alteration to RNN style that allows the program to store dependencies, what can be considered as previous context, throughout iterations of the network. This type of model is particularly well suited for the generation of much because every note is dependent upon previous notes to make a current melody. The long term memory of the network determines context of the music, better allowing the program to produce a coherent subsequent note. 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.

4.3 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 3 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 stated 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 an A to a B transition, the relative markov chain would output a one-step transition. The benefit of the relative style is that it is less work to transpose the music from one to another; everything is based upon step distance rather than absolute note values.

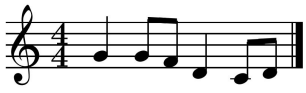

Figure 4: A single bar melody produced after the second iteration of markov chains

The last step to the markov model is labeled as post processing. The post processing of the music is a step that takes the already generated simplistic melody and then runs

it through a last markov chain in an attempt the 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 alteration through several methods: note splitting, one point mutation, and recombination.

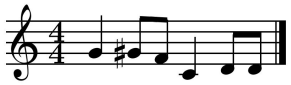

Figure 5: Previous melody after being put through one-point mutation

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 the 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 sixteenth notes.

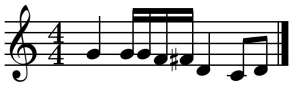

Figure 6: Previous melody after being put through a note-splitting mutation

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

4.4 Citations

Acknowledgments

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

the picture on descision trees needs to be smaller than it is currently

I know the markov chain graphic is in a very weird spot, but there was a weird spacing issue with it earlier so currently i have it placed there to get rid of the spacing problem