# EECS 126

Music Generation with Markov Chains

Sarika Madhvapathy, Hersh Godse, Sukrit Arora, Sushanth Varma

## 1 Introduction

Music is an art form and creative expression central to our human experience. Bach, Mozart, Michael Jackson and other music revolutionaries have all blessed us with their compositions, evolving their genres and inspiring new ones. While we were only fortunate enough to be exposed to their brilliance for dismally short amounts of time, what if we can reproduce a central repository of their musical prowess using their past productions? Better yet, what if we can use this repository to produce original music mimicking the sounds of the greats?

In our endeavor to reproduce musical genius and immortalize our favorite musicians and genres, we have captured several common stylistic elements and themes across compositions from the same artist and/or sub-genres. We analyze factors/variables of songs like musical notes, loudness, and duration; this information is encoded into a Markov Chain model to approximate the songs. A simple random walk along this Markov chain produces fascinating results that come close to passing off as original, human-made music!

## 2 Methods

We used MIDI files as a resource to build our Markov Chains (MC) by analyzing the aforementioned variables: the states of the MC were built using a combination of musical notes and their duration in different tracks on a single MIDI file, and the transition probabilities were determined based on the note progressions found in our input MIDI files. For example, if the following progression occurs frequently in a given song - note A plays for

100ms, and note B plays immediately after for 50ms - then we would have two states ('note A for 100ms', 'note B for 50ms'), and we'd have a high transition probability from the former to the latter. Thus, based on the input songs we use to build/train our MC, we can generate songs that are thematically similar.

We simplified the complexity of our state space by approximating note durations to the nearest multiple of 25ms (so if in the original input file, we have note A for 62ms, this would be treated as the occurrence of a 'note A for 50ms' state). This assumption greatly reduced the sparsity of our MC and improved the variety of chord progressions occuring as we took random walks, while losing a minimal amount of data.

## 3 Experiments

MIDI files are composed of several tracks, where each track represents a singular instrument in the musical piece. For example, a song could comprise of tracks with two pianos and a drum set, and in such a case, a MIDI file would have three tracks, one for each instrument. Initially, we only experimented on songs with a single track and built a standard MC (as mentioned above), where the states were represented as tuples of notes and their duration. In each experiment, the MC was only trained on one song (from either our classical or video game music library), and new songs were generated via random walks.

We added further depth and complexity to our productions by exploring songs with multiple tracks. To accommodate multiple tracks in our songs, we began representing our MC states as a tuple containing sub-tuples and a duration value, with one sub-tuple per track. Each sub-tuple contains all the notes from its track that are being played at this state. For example, if at a given state we have track 1 playing notes A and B, and track 2 playing notes C and D, all for a duration of 200 ms, we would have a state represented as ((A,B), (C,D), 200). We first trained the MCs on multi-track songs, one song at a time. We added yet further complexity by training on multiple multi-track songs belonging to the same genre, and to different genres.

## 4 Results/Analysis

For our simple initial experiments, where we trained on just one single-track song, we were quite successful at generating new stylistically similar music. The generated files clearly shared the stylistic elements of the songs they were based on (for example, a generation from a simple 'Row Your Boat' tune had the same notes and similar note progressions, mixed in a new order). Generations from longer and more complex single-track songs like 'Clair de Lune' came out significantly different from the source material, yet retained several style elements. Some generations sounded like little more than a random shuffle of notes, while others were actually surprisingly pleasing to listen to. These results were consistent across songs from both classical and video game genres.

Our generations from multi-track songs did indeed capture the increased sophistication of chord progressions present across the different tracks, but were certainly more prone to 'random' sounding note progressions. In all, we explored themes in Classical and Video game music and even gave EDM music a shot. EDM music was a little more complex than the rest as its songs had a very high number of tracks, which together produced a less harmonious MC that at times made generated songs sound quite disjoint. This could be attributed to the bass-heavy and curt nature of EDM music and the many different instruments present in these songs, whereas the other two genres/themes are more constrained in the variety of instruments used in the songs, making the transition probability matrix of the MC less complex and more harmonious. Finally, training our MC with just a single input song made significantly more pleasing music than when training on multiple songs. This is because the input songs were all already pleasing to listen to, so only training on one in isolation better fit the random walk to the patterns present in the song, while training on multiple songs mixed these patterns together to varying degrees and produced a less satisfying output. This phenomenon was especially apparent in our training on multiple songs across genres.

To summarize, there was a clear correlation between less complex songs and higher-quality generations. Our generations were at times genuinely quite pleasing, and certainly

captured several of the stylistic elements of the songs they were based on. As our training song complexity increased, the quality of the generations suffered, as the MC model was less able to capture this complexity. A major factor contributing to this was our design of the next note+duration state in the random walk being only determined on the immediately preceding note+duration state. By instead using a larger number of preceding states in both training and generation, we could potentially improve the quality of our generations. This could be implemented with a MC containing a much higher amount of states, where these states now represent a sequence of note+duration tuples (rather than just a single note+duration tuple in the current architecture).

## 5 Discussion/Limitations

There were several issues we ran into while implementing our project. The first was inconsistency in the structure of training data. While MIDI is a standard format, the notes in the MIDI file do not conform to a single format. As a result, we had to manually look through the contents of a set of MIDI files to look for patterns in the overall structure. Fortunately, we were able to classify most MIDI files into two groups: files that toggled notes using either on-off commands or using a velocity variable. Accordingly, we wrote two different file parsers, one to handle each note structure.

Another limitation we saw with our random song generation was that the random walk very clearly retained the characteristics of the data we trained on. This worked well for training on a single file, but produced muddled results when trained on multiple files; but this is understandable since each song works with different notes and durations. Ultimately, by mixing the composing styles and keys of different artists, our results were very interesting: not terrible, but not very organic either.

All this leads to a larger limitation of the overall project. Music is inherently not random. The first step a musician takes while composing music is picking a key and envisioning a general structure of chord progressions followed by considering melodies to use. These

melodies are the motifs and themes that turn up repeatedly in music and are inscribed into memory. Musicians then add all the other intricacies of music: volume, speed, style which are followed by another layer of uniqueness that is inherent to each composer. These layers are pretty hard to capture within one song. The markov chain does make an admirable approximation of it, but in our attempts to train on multiple songs, these intricacies are lost and the resulting random walks sound disjoint. Thus, while this project was fascinating to work on and a great approximation for a single training set and style, it doesn't work very well when generalized to multiple song training set with different styles.