Safak Ozkan

https://github.com/ozkansafak

June 13, 2016

SPURIOUS FUGUE GENERATOR / ARTIFICIAL CLASSICAL MUSIC COMPOSER

ABSTRACT

In my Capstone Project, I will train a Recurrent Neural Network (RNN) Model on MIDI files of Fugue compositions of Baroque composers such as J.S. Bach and possibly G. F. Handel and J. Pachelbel.

PROJECT RELATED TERMINOLOGY

- A **MIDI FILE** consists of the most basic information regarding the composition of a musical piece. It contains the note sequences and their durations. However, it doesn't contain the information regarding the audio characteristics of the musical piece or the instrument the file represents.
- A **FUGUE** is a contrapuntal compositional form in two or more voices. It is typically written to be performed on a harpsichord, a piano or a pipe organ or a string quartet. It is characterized by simultaneously occurring melody lines. The development of a fugue follows a set of complex rules and it's commonly divided into 3 main sections: *expositions, divertimenti* (episodes), and *stretto* (narrow).
- **MELODY** is a linear succession of single notes that's intended to be perceived as a single entity.
- **NOTE** describes the relative pitch of a musical sound.
- **NOTE VALUE** indicates the relative duration of a note. They're typically equal to an integer power of 1/2 of the length of 4 beats in standard time signature. (eg. a whole note, an 8th note, a 16th note)
- **COUNTERPOINT** originates from the Latin *punctus contra punctum* meaning 'point against point'. In music, counterpoint is the relationship between two melodies that are played simultaneously. It's a common characteristic of classical music in Renaissance and Baroque periods and in fugues.
- A **CHORD** is a group of typically three or more notes played together. It implies a harmonic environment where certain notes are allowed to be played while certain others are not. Chords form the basis of harmony.

- A **CHORD-SEQUENCE** is a series of musical chords that are directly played or implied by the melodic elements, that aims for a definite goal of establishing a tonality founded on a tonic chord and that is based upon a succession of root relationships.

DATA

Amongst JS Bach's approximately 170 fugal works that include fugues, canons, preludes, and fughettas, I have downloaded more than 50 MIDI files. These will be examined by use of an open source python library `python-midi` (https://github.com/vishnubob/python-midi). Firstly, the melody lines will be extracted and loaded into a time-series style numpy array of shape n-by-3 where each row will correspond to when the note will be played, the pitch of the note and the note value (duration).

METHODOLOGY

The RNN Model

The RNN Model I'm going to implement is developed by Andrej Karpathy for creating words and sentences to compose a new article or a poem based on the training data such as Shakespeare's poems or a journal article. The RNN model is observed to perform remarkably well in making up words, sequence of words, and whole sentences including the period at the end and some punctuation like comma, and the general format of the text—separate document sections in the case of a journal article. A compelling reason in using Karpathy's RNN model is that it bears a strong resemblance to the framework and the end goal of Spurious Fugue Composition project.

The RNN model has a simple API. It only calls a single function `step()` that takes a single vector `x` and outputs a single vector `y`.

```
rnn = RNN()

y = rnn.step(x)
```

The model will have a prescribed number of recursions on updating the value of the hidden layer. Each layer h_t will take the value of the hidden layer on the previous recursion h_{t-1} and the current input vector x as follows:

$$h_t = \tanh(W_{hh} * h_{t-1} + W_{xh} * x)$$

$$y = W_{hy} * h_t$$

where the entries of the coefficient matrices W_{hh}, W_{xh} and W_{hy} are the unknown parameters of the model which will be tweaked to optimize a loss function.

I'm expecting RNN model to learn everything from the training data including, the melodic patterns, chord progressions, the development of sections within the fugue, melodic development within a section.

Pre-processing

1. **The chord progression:** A separate model will extract the chord information implied buy the melodic development of the fugues. The MIDI files will be processed to decipher the chord-sequence. (This might require a substantial amount of work to model. Alternatively, I might supply a pre-described chord-sequence and have the RNN model compose the rest of the fugue while staying loyal to the chord-sequence)
2. **Extracting the melody lines:** All melody lines will be extracted from the MIDI file and will be treated as separate inputs that make up the fugue. However, the MVP will focus on building a single melody ignoring the contrapuntal melody lines.

Training the Model

1. **Harmonic landscape:** Firstly, a *Chord-Sequence Building Module* will produce the chord-sequence and the specific sections of the fugue. This will provide the model with a harmonic backbone to navigate through with the melodies.
2. **Start building a melody**: The *Melody Builder Module* will start with a random note or degree of the chord. The RNN model will build the melody sequentially.
3. **Progress:** The *Melody Builder Module* will produce a probability distribution for the next note degree within that chord taking into account how far along the section it's at, how far it is in the corresponding bar and the duration of the preceding notes. The next note will be picked by a random number generator based on the probabilities of the output of the RNN model[1].

FEATURES

The Vocabulary of a melody is made up of all possible notes –where there's 88 of them on a modern piano– and the durations of each note. However a note can be described by one of the 12 distinct notes and the one of the 7 octaves it belongs to.

The core of the model will be formed by identifying the following.

1. the chords and chordal movement,
2. the patterns in melody lines, and
3. the interaction between contrapuntal melodies.

I'm expecting the RNN to be able to handle all of the above items.

[1] Ideally, the model should assign a probability to the entire melody instead of a "next note".

References:

1. San Francisco Bach Choir, http://www.sfbach.org/what-fugue
2. Anatomy of a Fugue, Northern Arizona University, http://www2.nau.edu/tas3/fugueanatomy.html
3. Visualizing and Understanding Recurrent Networks, Talk by Andrej Karpathy, 9/10/2015, London, CodeNode, https://skillsmatter.com/skillscasts/6611-visualizing-and-understanding-recurrent-networks
4. The Unreasonable Effectiveness of Recurrent Neural Networks, Andrej Karpathy Blog, http://karpathy.github.io/2015/05/21/rnn-effectiveness/
5. Bach MIDI Files, http://www.midiworld.com/bach.htm
6. vishnubob/python-midi library, https://github.com/vishnubob/python-midi
7. J.S. Bach Page, http://www.jsbach.net/midi/midi_artoffugue.html