Music Harmonization Using Reinforcement Learning

Lakshya
2017eeb1149@iitrpr.ac.in
Indian Institute of Technology Ropar

Figure 1. The left hand is generally responsible for harmony whereas the right hand is responsible for melody

Abstract

Application of Learning Algorithms to creative processes has long been discussed among the AI community. However, it has been in the recent years that these applications have began delivering significant results. One of such areas has been that of generation of harmonies. In this application the algorithms trains for generating musically sound harmonies to accompany a given bar of melody.
In this paper, we'll introduce yet another effort to train an algorithm using reinforcement learning for this specific task. We'll discuss on the reward modelling, feature engineering, value approximation model and data cleaning for this approach.

Keywords: music, neural networks, harmonization, reinforcement learning

1 Introduction

There are two kinds of approaches broadly followed by musicians. One is a strict rule based theory intensive approach followed in Classical Composition, and the other is an improvisation based approach followed by jazz musicians. The latter approach is much more nuanced than the former one and thus cannot be expressed in terms of a rigid rule set, hence, can take advantage of decision based learning approaches of Reinforcement Learning. In this application, we'll follow the approach of modelling the environment as a partially observable MDP and the problem as an ”optimal control problem”. The concept of what makes sound music and what makes music sound ”good” is highly subjective and thus an approach that can learn theory followed in a particular set of music compositions can really help customize the harmony generation to a particular musicians style. [5]

2 Preliminaries

2.1 Basic Music Theory

In total there are 12 notes spanning an octave. The same 12 notes repeat in other octaves as twice of their frequencies.
A melody is a sparse set of notes that typically represent the voicing in a piece of music. It is that part of a song that can be hummed.
Chords are a set of notes played together. These notes are part of a bigger set of notes that represent the "mood" of a piece of music. In our project, we are only considering triads. Triads are chords made up of exactly three notes. Chords represent the harmony of a piece of music.

2.2 MIDI

MIDI - Musical Instrument Digital Interface, is one of the technologies used to represent music in digital format. A typical MIDI file has multiple `tracks` that represent notes played by different instruments. Under the hood, each track comprises of a series of `Messages`. The messages of type "note_on" represents a pressed key on a synthesizer and has the attributes `note, velocity and time`. The note value is a 7 bit integer representing the note played, the velocity represents approximately how loudly the note is played and the time attribute represents the wait time after this note

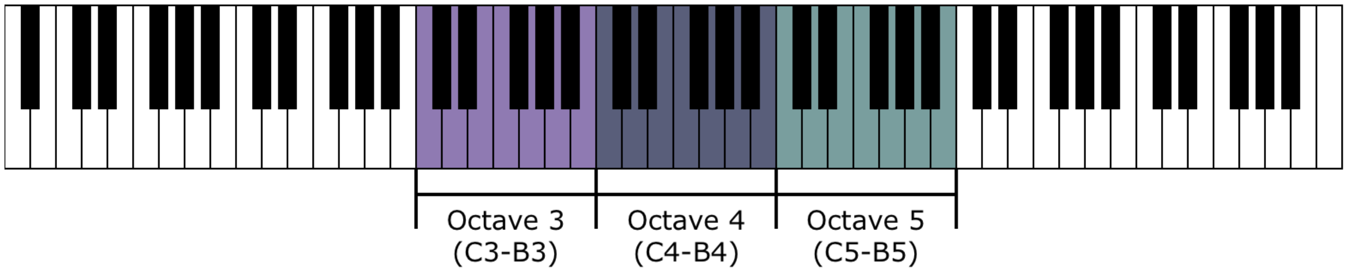

Figure 2. Each octave has 12 notes and they repeat with doubling of respective of frequencies each higher octave[2]

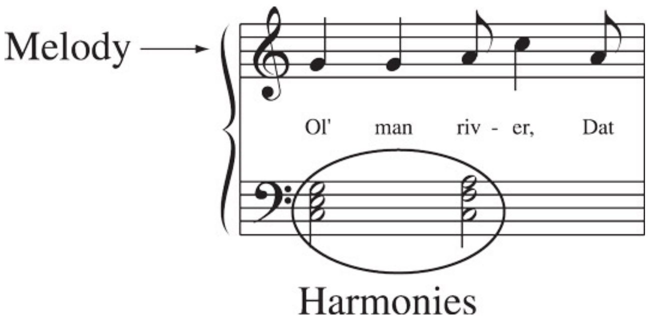

Figure 3. A sheet music representation of Melody and Harmony[1]

is played in `ticks`. In our case, we decided on 480 ticks per beat.

2.3 Artificial Neural Networks

Artificial Neural Networks have been widely in use recently for their ability to generate representations of complex data. In our project, we use a relatively small neural network to approximate the value function.

3 Methodology

3.1 Data Acquisition

For our method, we required segregated sets of MIDI files for the melody and harmony. Most MIDI files available in the public domain are highly complex and carry with them the nuances of varied tempo, multiple tracks and voicings. We eventually settled on 2 sets of MIDI files by Daniel Bonetti on YouTube[4][3]. Each of these MIDI files provide a compilation of 25 songs from popular music with separated MIDI files for the melody and harmony.

3.2 State and Action

Modelling this problem in a reinforcement learning framework requires the definition of states and action.
For our purpose, we consider 1 bar as a state, each bar containing 4 beats(quarter notes) in accordance with 4/4 time signature, each beat being able to represent 8 notes. This brings the total number of notes in each state to 32. Constricting the notes to a singular octave, there are 12 possibilities for each of the 32 notes. This brings the number of unique states in our state space S to $|S| = 12^{32}$
Our actions will be predictions of chords which match the melody. In this work, we're only considering triad chords, that is chords made up of exactly three notes. There are 4 types of triadic chords in music theory.

- Major chord
- Minor chord
- Diminished chord
- Augmented chord

Each chord has a root note. Hence, for each of the 12 notes in an octave, we can have 4 triadic chords. This brings the number of possible actions in our action space S to $|A| = 48$.

3.3 Data Cleaning

All note messages from the MIDI files were quantized to 32nd notes, that is the minimum note length that we have assumed in our state.
All notes were aligned with exact precision to their respective time divisions to reduce any form of humanization from the MIDI. Humanization refers to the process of purposefully mis-aligning MIDI objects to emulate human lack of precision. This process greatly aids in parsing the MIDI file for note messages and binning then to the bars that they span over.
We combine all the melody MIDI files into a single file and similarly a corresponding harmony MIDI file. All Data cleaning and pre-processing mentioned was carried out using LogicProX.

3.4 Parsing and Modelling MIDI

We parse the MIDI files using the **mido** open source library. State at time $= t$ is represented by $s_t \in \Re^{12}$ where the i_{th} element is 1.0 if a note of value $12n + i$ is present in the bar that is represented by the state $s_t \ \forall n \in W$.
The same method is followed for representing action a_t, although due to our definition of the state, exactly three elements of a_t will be 1.0

3.5 Features

We compute a set of 3 distinct features for each state-action pair.

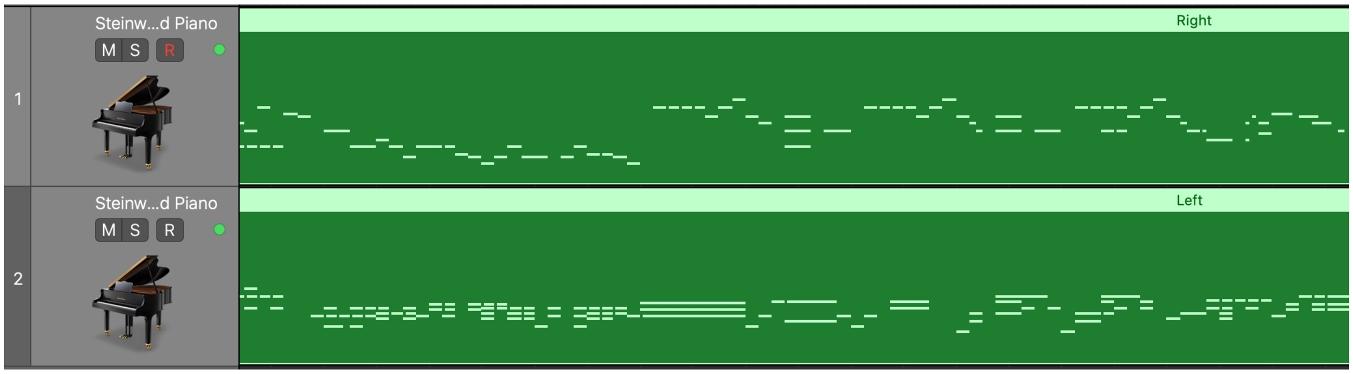

Figure 4. A piano role representation of the midi files used for training. The Right hand generally plays melody while the Left hand generally plays harmony which is also evident from the distinct pattern in the two tracks

- $f1(s, a)$: fraction of common notes between the state and action over total notes in the state.
- $f2(s, a)$: fraction of common notes between the state and action over total notes in the action.
- $f3(s, a)$: 1.0 if no tritone exists in the action with respect to any of the notes in the state.

A tritone refers to an interval that is least harmonic with the root. It divides the octave into two parts and is considered one of the most dissonant sounds. It's existence in the genre of popular music is extremely rare and hence can be a good feature to consider.

3.6 Model Architecture

We utilize features from the current state s_t and last state s_{t-1} to compute the input vector for our value function approximation. We model our value function in form of a simple 3 layer neural network $A : \Re^6 \to \Re$. Each layer is followed by a leaky ReLU activation with $\alpha = 0.1$

$$Q(s_t, s_{t-1}, a_t) = A([f1(s_t, a_t), f2(s_t, a_t), f3(s_t, a_t), f1(s_{t-1}, a_t), f2(s_{t-1}, a_t), f3(s_{t-1}, a_t)]) \quad (1)$$

3.7 Reward

Modelling the reward is one of the most crucial part of the Reinforcement Learning Framework. It is essential to note that in our problem, the choice of action does not affect the next state that we encounter. Hence a binary form of reward will be proper. We compare the predicted action(chord) with the harmony. Since the harmony data need not conform to our definition of chord which is only limited to a triad, We devised a reward metric to deal with this dissimilarity.

3.8 Updates

We use a SARSA[6] like algorithm to solve our control problem. the difference compared to normal SARSA is that here, we know what the next state is going to be without any dependence on the action taken. The other difference is utilization of more than 2 state in our algorithm.

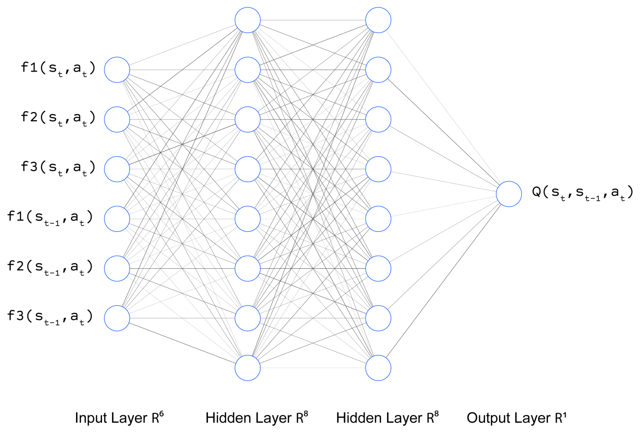

Figure 5. The architecture of our Neural Network that represents our value function

Code 1 The reward function defined to accommodate for different voicings of harmonies in the true harmony data

```
def get_reward(chord, harmony):
    if np.sum(harmony) > np.sum(chord):
        if np.array_equal(chord,np.multiply(
            ↪ chord, harmony)):
            return 1.0
        else:
            return 0.0
    else:
        if np.array_equal(harmony,np.multiply(
            ↪ chord, harmony)):
            return 1.0
        else:
            return 0.0
```

Code 2 Slightly modified SARSA algorithm for our control problem

```
for episode in range(episodes):
        melody1 = melo_episode[0]
        melody0 = melo_episode[-1]
        chord1 = get_action(melody0, melody1)
        for index in range(melo_episode.shape[0]
            ↪  - 1):
            reward = get_reward(chord1,
                ↪ harm_episode[index])
            melody1 = melo_episode[index]
            melody2 = melo_episode[index + 1]
            chord2 = get_action(melody1, melody2)
            update(melody0, melody1, melody2,
                ↪ chord1, chord2, reward, gamma)
            melody0 = melody1
            chord1 = chord2
```

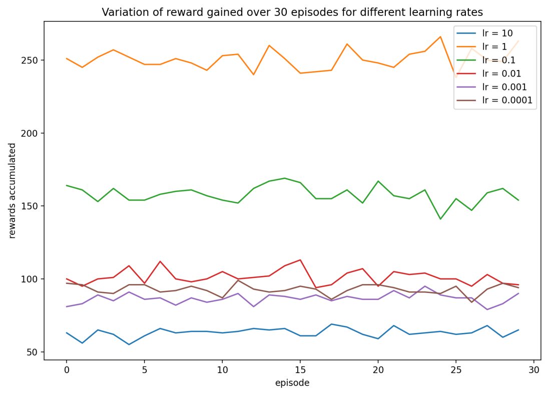

Figure 6. Variation of accumulated rewards with different learning rates over 30 episodes

We use the $TD(0)$ update to update the weights for our value function network. The standard gradient descent optimizer is used to update the weights.

$$\Delta w = \alpha(R_{t+1} + \gamma Q(s_{t+1}, s_t, a_{t+1}) - Q(s_t, s_{t-1}, a_t))\nabla_w Q(s_t, s_{t-1}, a_t) \quad (2)$$

4 Experimentation

We experiment with different learning rates for our optimizer:
learning rates = [10, 1, 0.1, 0.01, 0.001, 0.0001]

5 Results and Discussions

From the experiments, we can observe that a learning rate of 1 seems to deliver the best result out of our limited number of episodes. Since rewards are sparse in our problem, we see almost no improvement in the performance of the agent in 30 episodes. The learning seemed to be extremely slow, a possible explanation for this can lie in the inability for the tensor computation to take advantage of significant vectorization of loops for a Reinforcement Learning setting.

On playback of the generated harmonies, certain bars seem to have appropriate harmonies. Each episode has 1151 bars. It is essential to note here that, the reward expected over one episode for a random choice of action is $\frac{1}{48} \times 1151 = 23.97$. Our best result $(reward = 266)$ is 11.09 times the reward for random. The code for implementation can be found `in this colab notebook`

6 Conclusions and future work

Reinforcement Learning is a probable candidate for generating harmonies for melodies. However, RL frameworks are very time intensive in terms of training and it is difficult to make inferences about the roles that different hyperparameters play.

In future work, better feature engineering, more able value function approximations and better, faster hardware can drastically improve the results of RL algorithms on such creative tasks.

References

[1] [n.d.]. Getting Two-Dimensional in Classical Music: Piece and Harmony. https://www.dummies.com/art-center/music/getting-two-dimensional-in-classical-music-piece-and-harmony/
[2] [n.d.]. Octave Registers. https://www.allaboutmusictheory.com/piano-keyboard/octave-registers/
[3] 2015. 25 Pop Songs Piano Medley 2014 Synthesia (MIDI + Sheet) - YouTube. https://www.youtube.com/watch?v=FI1FVDH_AqQ
[4] 2016. 25 Pop Songs Piano Medley 2015 Synthesia (MIDI + Sheet) - YouTube. https://www.youtube.com/watch?v=7BYDBTTfP4o
[5] T. Klimek and Ben Machlin. [n.d.]. Teaching Music Harmonization With Reinforcement Learning.
[6] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning: an introduction*. MIT Press, Cambridge, Mass.