CS221 Fall 2017 Final Project Proposal

Automatic Music Instrument Fingering Arrangement using Reinforcement Learning

Ting-Wei Su Brandon Wu

Introduction

Automatic fingering arrangement (AFA) aims at finding the optimal fingering path for a song or a sequence of symbolic music notes (such as MIDI). For a whole automatic music transcription process, AFA is the very crucial final step, especially for instruments that use tablature as their musical notation such as guitar. It is also helpful for beginners to learn how to perform a new song.

Generally speaking, manually arranging fingerings is time-consuming and may be difficult for beginners. However, fingering of most instruments follows some rules and strategies, so it is possible to make use of these rules and consider AFA as a search problem. Burlet *et al.*[1] applied A* algorithm to their guitar tablature transcription framework, and Balliauw *et al.*[2] proposed a tabu search algorithm on piano fingering. Hori *et al.*[3] published the method of using HMM given manually adjusted parameters to generate guitar tablatures and fingering decisions. Other previous work includes using recurrent neural networks to predict the tablature[4]. Nevertheless, search problems are based on already known rules, and there might be some more latent and potential rules that we don't know. RNNs require a lot of training data, which are very hard to collect. Moreover, most of the guitar related work only generates guitar tablature without the fingering decision.

Therefore, in this project, we try to solve these problems using reinforcement learning, especially deep Q-network. By introducing the most basic rules in fingering to this task, such as the bio-mechanical constraints on human hands, we hope that the model will find a better solution and some potential strategies on AFA without learning from a large dataset. We will test our model on guitar and piano fingering arrangement and compare it with some approaches mentioned above.

System

The input of our system is **a sequence of symbolic music notes** with each note including the following information:

- Pitch (midi note number)
- Onset (s)
- Duration (s)

The output is **the position on instrument and the finger you use to play each input note**. Here is a concrete example of inputs and outputs on guitar problem. Given a sequence of notes

`[Note(pitch: 48, onset: 0.0, duration: 0.5),`
`Note(pitch: 52, onset: 0.6, duration: 0.5),`
`Note(pitch: 55, onset: 1.2, duration: 1.0)]`,
the output should be
`[(string: 5, fret: 3, finger: 4),`
`(string: 4, fret: 2, finger: 3),`
`(string: 3, fret: 0, finger: 0)]`
, where `string` (1 to 6) is the string number on guitar, `fret` (0 to 22 for electric guitar) is the fret number, and `finger` ($0\tilde{5}$) denotes which finger you should use to press the string (0 means no need to press the string).

We will apply deep Q-network to this problem, and the basic infrastructure will be created using PyTorch[1]. The possible features for DQN include **pitch, next pitch, duration, duration between this onset and the next onset, string, fret, finger**, etc.

Evaluation

We will test the model with small datasets, which contain about 100 songs with ground-truth labels, on both guitar and piano. The performance will be evaluated by its accuracy, precision, recall rate, and F-1 score. Then, we will compare our results to an A* algorithm system.

References

[1] Gregory Burlet and Ichiro Fujinaga. Robotaba guitar tablature transcription framework. In *ISMIR*, pages 517–522, 2013.

[2] Matteo Balliauw, Dorien Herremans, Daniel Palhazi Cuervo, and Kenneth Sörensen. Generating fingerings for polyphonic piano music with a tabu search algorithm. In *Mathematics and Computation in Music*, pages 149–160. Springer, 2015.

[3] Gen Hori, Hirokazu Kameoka, and Shigeki Sagayama. Input-output hmm applied to automatic arrangement for guitars. *Information and Media Technologies*, 8(2):477–484, 2013.

[4] Elias Mistler. Generating guitar tablatures with neural networks. Master's thesis, School of Informatics, University of Edinburgh, 2017.

[1] PyTorch: https://github.com/pytorch/pytorch