CSCE 790-002: Deep Reinforcement Learning and Search

Musical Deep Reinforcement Learning

Francisco Vilchez

October 5, 2020

Introduction

In this project we will try to solve the automated musical composition problem using Deep Reinforcement Learning. Musical composition is an interesting problem in Computer Science since it deals with a big challenge, which is trying to teach a computer how to evaluate if a composition sounds good, and maybe even a more difficult situation, which is to compare musical melodies and decide which of them has a better sound. Because of that, automated musical composition requires the usage of innovative methods that try to simulate the way humans (or musicians) take decisions. Additionally, solving this problem with the usage of Reinforcement Learning will become an interesting approach since it will enable the interaction between the computer and humans for training which compositions sound good and that way improve the computer's compositional skills. Finally, the broad amount of music genres, measures, melodic and harmonic combinations in a single instrument, and the combination of multiple instruments gives this problem a big scope for continuos improvement and the pursuit of a general model that can handle of the different scenarios that music usually has.

Approach

Our solution will be based in the usage of Reinforcement Learning and Recurrent Neural Networks for solving the automated musical composition problem. The details of each component of the algorithm are defined as follow:

States Each state will be represented by the sequence of notes that have been played so far. Each note is a represented by the note itself, its duration, the chord and the compositions tonality. Since we will be focused on generating melodies, the chord and composition tonality will be defined as an input. The notes may not be represented by its value itself (i.e C, D, E, etc), instead we may be using a relative representation based on the tonality or current chord (e.g. D will be a 1 if we are in a D chord, either major or minor, E will be a 2, F will be a 2.5, $F\sharp$ will be a 3, and so on) [1].

End state Since we don't want our algorithm to generate infinite amount of notes, we will have an *hyperparameter* that will be used to know when it has reached the limit amount of notes required. In other words, it will tell it when it reached the *end state*.

Actions We can see that the possible notes that can follow another note is very broad, because of that, the generation of the next actions will probably include some special criteria, for example, a note can only be followed by a note within an octave up and down of it. Additionally, the *hyperparameter* for the end state will be taken into consideration in order to decide the durations for the notes.

Transition sampling The reward generated by the transition will be zero by default unless the human decides that the melody (or part of it) deserves a reward higher than zero. Possible automated methods for this can be implemented in the future, however, one of the goals of this project is to allow the human-computer interaction during the learning process.

Monte Carlo Since the amount of states that we can explore is too big, we are considering to use Monte Carlo as our algorithm for reaching the goal state and learn from the experience. Since the a melody may only have a value after a set of notes are generated, we believe this method will be the most suitable for the situation.

RNN Since the next note that we want to use depends a lot on the previous notes generated, we consider that the usage of a *Recurrent Neural Network* will allow us to get better results than other type of models.

An attempt to apply Deep Reinforcement Learning to the automatic musical composition problem has been tried in different researches. One of them applied Deep Q-Learning, a Long Short-Term Memory (LSTM) network, a dataset of MIDI compositions and Music Theory Rules for the creation of melodies [2]. Our approach will take into consideration the work done on that paper but it will be focused on allowing the human-computer interaction for the creativity process of the computer. The usage of LSTM networks has considered effective for different researches [2] [3], showing even better results than the usage of other neural networks architectures [4]. One of them, is even an open source product which is currently available for usage [5]. However, for our case, the experience learned from the human-computer interaction will have more weight that any possible training performed with datasets in the LSTM, which will mean that the algorithm is learning based on the user's particular musical taste.

Evaluation

Measuring the performance of how well is our model is generating a musical melody can be a little complicated, since the quality of a musical composition is relative to each person's preference [6]. For our personal measurement, we will qualify the different set of compositions generated by individually listening to them and assign a good or bad label, and request other people to assign those labels according to their criteria. In order to compare our melodies with the results of other works that use the same methods, we will do a musical theoretical analysis of how many notes generated belong to the key and compare the results with metrics from previous works [2].

It is important to emphasize that the metrics based on music theory will not necessarily illustrate the performance of our algorithm since the learning process depends on the qualification that the user is given to the algorithm, which is not focused on following the theoretical rules, but satisfy its personal musical taste. Because of that, our personal measurement will provide a more interesting guidance for this type of project although it may not be used to compare it with the performance from other models.

Conclusion

We will be solving the automated musical composition problem by applying Deep Reinforcement Learning. Our solution will use Monte Carlo for learning from experience and a Recurrent Neural Network for storing the different notes combinations approximate values. For measuring our results we will evaluate if a composition from our model can be considered pleasant or not, and we will request the opinion from other people. However, for comparing the results with other works, we will use music theory in order to measure how many notes fit in the key and compare it with the performance obtained by previous works.

References

[1] F. Vílchez, J. A. Fuster, and C. B. Castanón, "Artificial musical pattern generation with genetic algorithms," in *2015 Latin America Congress on Computational Intelligence (LA-CCI)*, pp. 1–5, IEEE, 2015.

[2] N. Jaques, S. Gu, R. E. Turner, and D. Eck, "Generating music by fine-tuning recurrent neural networks with reinforcement learning," in *Deep Reinforcement Learning Workshop, NIPS*, 2016.

[3] D. Eck and J. Schmidhuber, "Finding temporal structure in music: Blues improvisation with lstm recurrent networks," in *Proceedings of the 12th IEEE workshop on neural networks for signal processing*, pp. 747–756, IEEE, 2002.

[4] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, "Deep learning techniques for music generation – a survey," 2019.

[5] A. Roberts, C. Hawthorne, and I. Simon, "Magenta.js: A javascript api for augmenting creativity with deep learning," in *Joint Workshop on Machine Learning for Music (ICML)*, 2018.

[6] J. A. Biles, "Performing with technology: Lessons learned from the genjam project," in *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, Citeseer, 2013.