CSCE 790-002: Deep Reinforcement Learning and Search

Musical Deep Reinforcement Learning

Francisco Vilchez

November 4, 2020

Introduction

In this project we are trying to solve the automated musical composition problem using Deep Reinforcement Learning. Solving this problem with the usage of Reinforcement Learning becomes an interesting approach since it enables the interaction between the computer and humans for training which compositions sound good and that way improve the computer's compositional skills.

In this project report we will present the results that we have obtained so far by applying *q-learning*. We have been able to teach the computer which notes are the best to be played after other notes according to our criteria, and that way generate a melody that match our preferences. We will present the approach followed, examples on how we tested it, and the future steps that we will take in the project.

Related Work

The first approach analyzed for making this project was the one by Jaques et al. They proposed the usage of multiple deep Q-networks to store the rewards for each note and also for musical theory. That way, their algorithm will have the freedom to create melodies which are based on its experience, but will also take into consideration the different musical rules that will "ensure" that their compositions have a pleasant sound. Because of that, the reward function takes into consideration both, the rewards based on experience and the value according the musical rules. For both of them, and also for the target deep Q-network, *Recurrent Neural Networks (RNN)* were used [1]. The usage of *RNN*s for the musical composition problem has been corroborated by Eck and Schmidhuber, specifically with the usage of *Long Short Term Memory (LSTM)* RRNs. As they mentioned, *LSTM* is able to keep characteristics such as timing and musical structure while training, which was not possible with other type of models. They applied the solution to the *blues* musical genre and obtained pleasant results according to their work [2]. Both proposals based their work on optimizing the best note that should follow a previous sequence of notes, which is considered the main goal in the musical composition problem [3]. Our project differs from these two principal approaches in three aspects. Firstly, the usage of musical theory is completely ignored, since we consider that, even though musical theory can explain why a melody sounds nice, it should not constraint the set of notes that can be used [4] [5]. Secondly, we will try to introduce a novel data structure which is based on the note's grades instead of storing a note itself (more details in the next section). Finally, the principal goal of our project is to enable the human-computer interaction for the training process and that way we expect the computer to develop a more human-like musical skill.

Approach

As mentioned before, *q-learning* has been applied so far in this project. We were able to teach the computer what sequence of notes would have a pleasant sound according to our criteria, and by applying a greedy search the computer was able to generate those results successfully. The original approach from our project proposal has been followed; however, some constraints were applied. We will keep improving them in order to get better results and get to the expected result. Some of the constraints applied are detailed below:

Note representation A specific number for each note is being used at the moment instead of the note's grade, which will be changed in the next releases.

State Even though we have the intention to include more musical information in the state, e.g. duration of the note and tonality, we reduced this to just the note as a starting point for the project. Because of that, all the "melodies" generated so far have the same duration. Additionally, we have constrained the amount of notes that can be generated, so our "melodies" always have the same amount of notes for now.

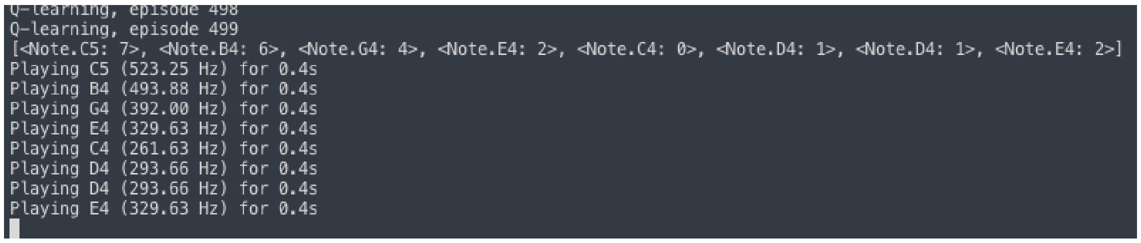
RNN A recurrent neural network have not been used to store the *(action, state)* values. Instead, they have been stored in a matrix. Future releases will try to include this requirement in order to get better results.

Monte Carlo As mentioned, *q-learning* had been used so far in our algorithm. We expect to change this in the future in order to improve our results.

Transition reward The interaction between human-computer for giving the rewards have been implemented. However, for testing purposes, we have created automated rewards in order to accelerate the learning process during testing. Additionally, the learning values of our model are not being persisted yet, which is still pending to implement.

Experimental Results

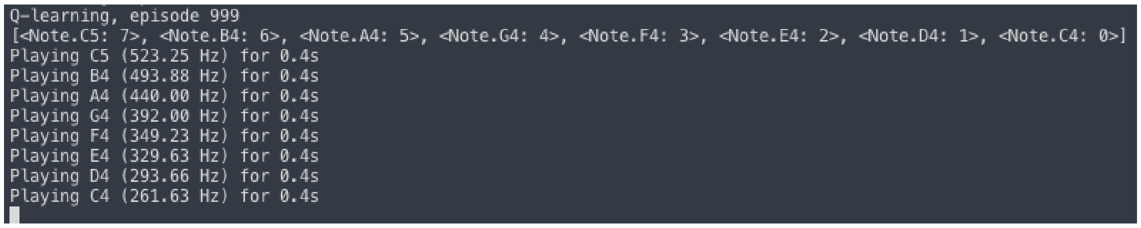
Since the learning process that involves the human-computer interaction takes a regular amount of episodes in order to generate results, automated policies where used during training in order to test the functionality of our algorithm. For example, one of the policies that we applied in order to test if our algorithm was learning correctly, was to tell it *not to play any note with a frequency higher than the previous note*. In other words, always play a note with lower pitch that the previous one. In musical terms, what this is trying to teach is a descendant scale. For that, we gave a -100 reward in case the algorithm played a note higher than the previous one, and a $+10$ if it played a note lower than the previous one. We could see that our algorithm was able to learn the optimal policy after 1500 iterations in average.



```
Q-learning, episode 498
Q-learning, episode 499
[<Note.C5: 7>, <Note.B4: 6>, <Note.G4: 4>, <Note.E4: 2>, <Note.C4: 0>, <Note.D4: 1>, <Note.D4: 1>, <Note.E4: 2>]
Playing C5 (523.25 Hz) for 0.4s
Playing B4 (493.88 Hz) for 0.4s
Playing G4 (392.00 Hz) for 0.4s
Playing E4 (329.63 Hz) for 0.4s
Playing C4 (261.63 Hz) for 0.4s
Playing D4 (293.66 Hz) for 0.4s
Playing D4 (293.66 Hz) for 0.4s
Playing E4 (329.63 Hz) for 0.4s
```

Figure 1: Iteration 500: Not efficient policy due to note transitions like C4 to D4 in position 5

As we can see in Figure 1, the optimal policy was not found yet, since the algorithm is still deciding to use a higher note in the position 6 than the previous note (position 5).



```
Q-learning, episode 999
[<Note.C5: 7>, <Note.B4: 6>, <Note.A4: 5>, <Note.G4: 4>, <Note.F4: 3>, <Note.E4: 2>, <Note.D4: 1>, <Note.C4: 0>]
Playing C5 (523.25 Hz) for 0.4s
Playing B4 (493.88 Hz) for 0.4s
Playing A4 (440.00 Hz) for 0.4s
Playing G4 (392.00 Hz) for 0.4s
Playing F4 (349.23 Hz) for 0.4s
Playing E4 (329.63 Hz) for 0.4s
Playing D4 (293.66 Hz) for 0.4s
Playing C4 (261.63 Hz) for 0.4s
```

Figure 2: Iteration 1000: Optimal policy. Each note is lower than the previous one.

In Figure 2 we notice that the algorithm finally found the optimal policy for the scenario that we proposed (each note should be lower than the previous one). Of course, this condition does not have any musical validity, it was only used for testing the learning capability of our algorithm.

Future Milestones

We are expecting to complete the following changes in the future releases in order to improve our results:

- 11/8: Persist the *(state, action)* values for future training. The matrix of values could be stored in a file for continuing the training later.
- 11/15: Enable different durations for notes. We plan to include the duration of the note in the state.
- 11/25: Finish project presentation and report.
- Desirable: Start using relative values instead of the notes itself. We need to modify the values for our notes in our state based on a melodies tonality. The tonality initially will be an hyperparameter in our algorithm.
- Desirable: Include *chords* in the composition process. This will enable us to translate the notes based on the chord and not in the tonality hyperparameter. We are planning to include the chord as part of the state.
- Desirable: Include Neural Network with our q-learning learning algorithm. The approach will follow the process from the assignments using PyTorch.
- Desirable: Start using RNN and Monte Carlo for the learning process. More research will be necessary in order to determine the effort needed for this task.

Conclusion

Based on the progress done so far, we have been able to adapt the musical composition problem to a Reinforcement Learning scenario. We were able to use *q-learning* in order to learn *musical policies* that allow us to know which note should follow a previous sequence of notes. We were able to test how our algorithm learn based on the input provided by the user, and we included an automated test in order to verify that the algorithm was adequately learning the appropriate values for each *(state, action)*. Finally, we detailed the future enhancements that we are planing to include for the future releases of our project with its respective methodology and that way improve the results that we are getting in order to get more pleasant melodies.

References

[1] Natasha Jaques, Shixiang Gu, Richard E. Turner, and Douglas Eck. Generating music by fine-tuning recurrent neural networks with reinforcement learning. In *Deep Reinforcement Learning Workshop, NIPS*, 2016.

[2] Douglas Eck and Juergen 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*, pages 747–756. IEEE, 2002.

[3] P. Todd and G. Loy. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13:173–194, 1989.

[4] Francisco Vílchez, Jose Astuvilca Fuster, and César Beltrán Castanón. Artificial musical pattern generation with genetic algorithms. In *2015 Latin America Congress on Computational Intelligence (LA-CCI)*, pages 1–5. IEEE, 2015.

[5] John A Biles. Performing with technology: Lessons learned from the genjam project. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*. Citeseer, 2013.