Open Program – Kaan Gögcay

Introduction

The initial goal of my Open Program was to make progress in the earlier dropped project Brawlhalla AI. Also did I try to use RL for a more basic game.

Brawlhalla AI

So the initial goal for open program was to make progress in the Brawlhalla AI project. This happened to be way harder than I thought. Earlier this semester I did generate the game information I wanted to feed to my model to for example create states and actions see Figure 1 and Figure 2.

Figure 1: Moving objects in game getting detected in real time

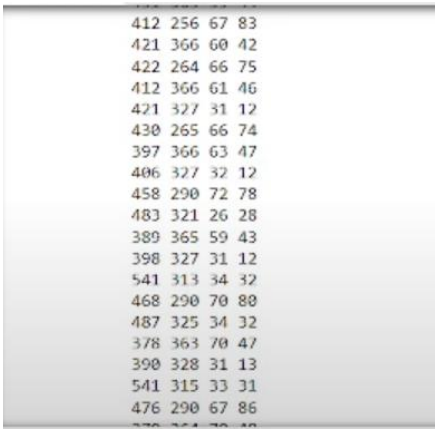

Figure 2: the coordinates of the detected objects

But not having the game information was just one problem I faced. I just couldn't figure out what to do. In theory I could explain how Reinforcement Learning works and the concept behind it. But I just didn't know how to actually do it. I tried reading and watching sources like (Firoiu, 2017) and (Renotte, 2022). But I always ended up with so much confusion. I couldn't just copy any other project since most of the projects I found had supporting libraries, such as Gym Retro or a library for web games. The game I wanted to implement reinforcement learning on is a game on Steam. So that also brought an extra layer of difficulty with it.

So I decided to have a talk with my ML teacher. I told her that I was stuck and explained my problems. She suggested me to try taking smaller steps. "*Instead of making an RL model for Brawlhalla, try making a RL model for a basic game*". This sounded like a good plan. Once I understand the basics of RL I could try to expand it to more complicated problems (games).

Cartpole Game – Attempt 1

The Cart Pole game is a very simplistic game where you have to balance a pole on a cart.

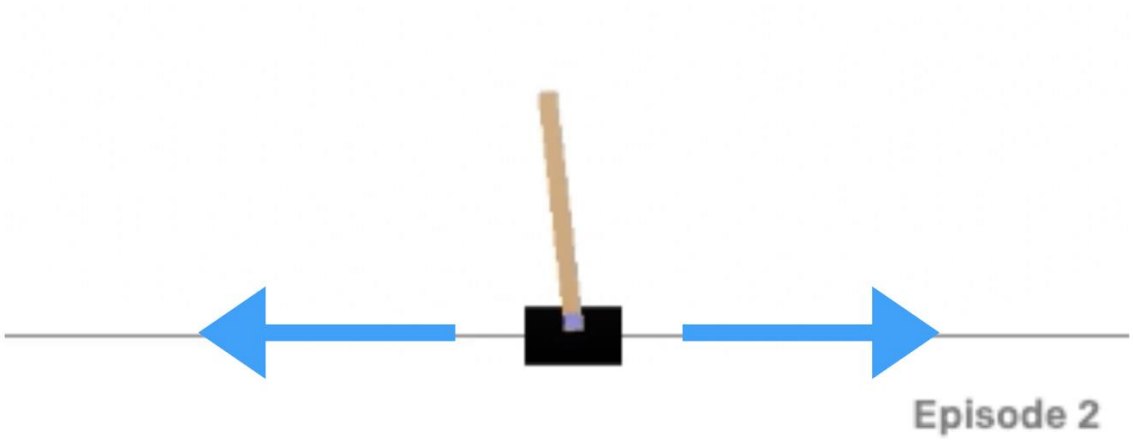

Figure 3: Cart Pole game

I did manage to get my hands on code where the game is played with random inputs (source: (Lessons, 2019)). This page also provided code to train a model using a Deep Q Network. But trying to make this code work I ran into error after error. This was very demotivating and suddenly I just had no time left. I did deliver the code I copied from the internet that did work but it's not that impressing.

Cartpole Game – Attempt 2

I was frustrated. Frustrated because I understood the theory behind RL. Knew what I had to know. But I couldn't make the code from internet work. I said to myself, *"If I understand the theory, then I should be able to make it myself… right?"*. This is how a new adventure started. I'd advise just going to the notebook itself. I will briefly summarise here what I have done but you can find everything in detail there.

Getting game information (states)

Unlike the Brawlhalla project. It was very easy to get the game state. It was as simple as typing give me the state and you got the state of the game. This state included cart position, cart speed, pole angle and pole speed. I know this because I used (OpenAI, Cart Pole, 2022) (the official documentation).

Defining Rewards

I started of by thinking how I can create rewards. Since if there are no rewards, the bot never knows when he does something good or bad. My idea was to give a higher reward the slower the pole moves. For example if the pole stands still we gave a reward of 5 and If the pole moves not slow not fast we give it a reward of 2. So now that we have defined a reward we can start collecting data.

Collecting Data (Training Model)

To train my model we needed one last thing. A way to save everything I just mentioned. We will save the state the game was in, the action it took, and the reward it gave all in an array. When we will test the model, we will make decisions using this data stored in the array. We can for example say "Hey, have you ever seen this state before?" and if it did see the state before, it can tell you the move it took and the reward it got. Now we can say, if the reward you took is for example lower than 2, do another action to see what reward that gives. If it indeed gives a better reward, we overwrite the data. This way we can optimise the model so that it always makes the best move in the given state.

Testing the Model

Testing the model is as easy as keeping track what score we get. I always played 100 games per agent/model and took the average of it. This way we get an accurate score for the model.

Scoreboard

Here are the scores I got per model/agent

Rank	Model/Agent	Method	Memory entries	Average score
1	1dec-model-both-v1	Pole Speed + Pole Angle	121.466	136,17
2	1dec-model-angle-v1	Pole Angle	13.121	107
3	1dec-model-v3	Pole Speed	11.846	99,89
4	Random Agent	Random Actions	-	23,4
5	1dec-model-v2	Pole Speed	3050	19,08

Conclusion

I observed the gameplay of the models on rank 1, 2 and 3. I'd say it doesn't matter too much which one you use, let's go over them

Rank 3 - 1dec-model-v3 (Pole Speed)

The main problem with this model was. The pole can have a speed of 0 and lean to a direction. For example if the pole leans left, the model learned to keep the pole speed at 0, so the cart will move left too. They will both move left. Till they hit a wall and game over.

Rank 2 - 1dec-model-angle-v1 (Pole Angle)

The problem with this model was that it would keep the pole very balanced but it would still lean to one direction very slightly. This meant that in the end it would hit a wall again.

Rank 1 - 1dec-model-both-v1 (Pole Speed + Pole Angle)

Same issue as 2

How can we fix this?

How can we solve the problems we're currently stuck at?

Look in the future

The proper way to fix this would be to make the model look multiple steps ahead. This way it would be able to predict it will die if it keeps moving the same direction all the time.

Look in the past

Another way to fix this problem is to instead of looking into the future, looking back in the past. After the first 10 moves, if were still alive the first move was good. Something like that. This solution sounds more probable to me just because I can already imagine how I could do this.

Change reward

The last and most lamest solution is to keep adding more methods. What I mean with this is instead of looking only at pole speed and pole angle that we also look at cart position and possibly also the cart speed. But this is very lame.

Cartpole Game – Attempt 3

In attempt 3 I tried to make a model play by looking multiple steps into the future. The notebook that belongs to this attempt is 10 times more readable than my first notebook. It's a prequel so check out both notebooks. If I explain everything here it would have no use since I would basically explain everything that has already been explained in the notebook, just check out the notebook.

I could at least put the results here but that would be a spoiler. You have to feel what I felt during the process.

Epilogue

I planned to work on Brawlhalla AI, and in the end I uh… made my own self-learning algorithm. Is this useful? Not really. Was it cool? Definitely. It tested my Reinforcement Learning theory. Since it some what worked out I'd say my theory is there.

The next step would be to try to make use of Deep Q-Learning. If I get the hang of that. I would be really powerful. The world would be unsafe.

Sources

Brooker, R. (2020, February 22). *OpenAI Gym: CartPole-v1 - Q-Learning*. Retrieved from Youtube: https://www.youtube.com/watch?v=JNKvJEzuNsc

Firoiu, V. (2017, May 8). *Beating the World's Best at Super Smash Bros. with Deep Reinforcement Learning.* Retrieved from arxiv: https://arxiv.org/abs/1702.06230

Lessons, P. (2019, September 22). *Introduction to Reinforcement Learning*. Retrieved from pylessons: https://pylessons.com/CartPole-reinforcement-learning

OpenAI. (2022). *Cart Pole*. Retrieved from gymlibrary: https://www.gymlibrary.dev/environments/classic_control/cart_pole/

OpenAI. (n.d.). *Gym Documentation (Cart Pole)*. Retrieved from gymlibrary: https://www.gymlibrary.dev/environments/classic_control/cart_pole/

Renotte, N. (2022, July 8). Reinforcement Learning for Gaming | Full Python Course in 9 Hours.

Surma, G. (2018, September 26). *Cartpole - Introduction to Reinforcement Learning (DQN - Deep Q-Learning)*. Retrieved from Medium: https://gsurma.medium.com/cartpole-introduction-to-reinforcement-learning-ed0eb5b58288#c876

Tiger767. (2020, December 30). *ML-Brawlhalla*. Retrieved from GitHub: https://github.com/Tiger767/ML-Brawlhalla