COMP9444 Project Summary

Building Cart Pole game using RL

Term 2, 2022

Sun Yuan Lum, z5289055

Jiawei Deng, z5305322

Yaxin Yuan, z5366243

Tianwei Mo, z5305298

Zebin Chen, z5237137

I. Introduction

The Cart Pole game is a game where a cart is attached to one end of a pole that is initially upright. The aim of the game is to move the cart left and right, trying to keep the pole upright for as long as possible.

In this report, we will discuss and compare several reinforcement learning methods for effectively learning the Cart Pole game, such as the REINFORCE method, Deep Q-Learning and Double Deep Q-Learning. Since this paper will go through the theoretical and practical aspect of implementing reinforcement learning methods on the Cart Pole game, this paper provides a template and guide for which people can learn how to implement these reinforcement learning methods to learn how an agent behaves in an environment. In doing so, this acts as an introduction for people to develop interest in reinforcement learning. This is important as the applications of reinforcement learning in the current world we live in are continuously and rapidly growing, expanding to the biggest industries in the world, such as healthcare systems, transportation, financial markets and manufacturing automation.

II. Literature Review

In RL regarding cart-pole problem, model-free learning which aim at solve MDPs without knowing reward functions mainly include Q-learning, Actor-critic policy gradient method and TD with Value function approximation.[9]

In addition to this, to overcome discrete representation in Q-table for the cart-pole, researchers introduce neural networks to approximate Q-value function. Followed by DQN, double DQN which contains a target network for separating action choosing and evaluation has shown to perform well in complex systems. Dueling DQN could learn the value of states while skipping the learning of each action's effect in states. [2]

Features like memory replay have been introduced for agents to learn both on current states and previous experiences.

III. Methods

This project uses OpenAI Gym to simulate the Cart-Pole system, using mainly REINFORCE , Deep Q-Learning and Double Deep Q-Learning methods to implement the Cart Pole game. The Cart Pole game can be seen as a problem about solving stochastic optimal control, so the REINFORCE method was tried in the initial stages of the project. This method expects to learn the strategy directly from the optimal distribution of the function. However, due to some shortcomings of the algorithm, which resulted in a long time to obtain results and less than optimal task scores, several other approaches were tried. As a reinforcement learning task, the Cart Pole game itself is set up to have some reward mechanism. Using the Deep Q-Learning method, the agent can be made aware of the corresponding reward that the action will generate. And the DQN algorithm introduces experience replay and target network mechanisms based on the use of deep neural network approximation functions, with the aim of enhancing the stability of the network and improving training efficiency. Despite some advantages of the DQN algorithm, the method also leads to unstable training and lower quality strategies due to its tendency to overestimate rewards. So, we further experimented with the Double Deep Q-Learning method. Double Deep Q-Learning, in which the agent uses two neural networks to learn and predict the action to be taken at each step, is suitable for problems that overestimate Q. The performance of Double Deep Q-Learning is more stable compared to DQN, and more efficient.

IV. Experimental Setup

Since CartPole is a game in the Reinforcement Learning environment, we do not need any dataset for training. For that reason, we chose to use the CartPole environment in OpenAI gym.
We start by analysing the initial state of the CartPole by either calling env.render() to visualise it or checking the returned tuple of env.reset(). Below is an example tuple returned by env.reset(). Based on the wiki page of CartPole-v0 environment, we get to know the meaning and range of each value:

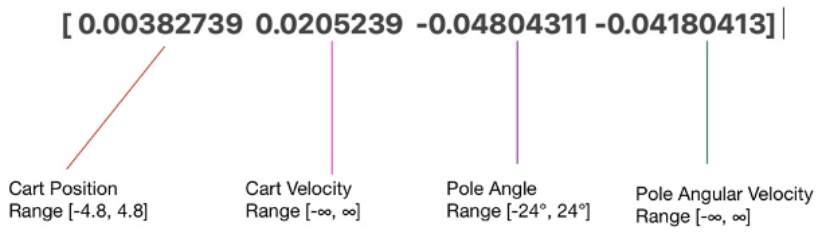

Figure 1 returned tuple of env.reset()

The values for the initial state are randomly assigned in range (-0.05, 0.05). Therefore, the values differ every time env.reset() returns.

To analyse CartPole's each movement step in the environment, we first need to know all the possible actions of the CartPole. Based on the wiki page of CartPole-v0 environment, we get to know that the gym environment uses values {0, 1} to indicate direction of the force that the cart is being pushed with (0 left, 1 right). The action (value 0 or 1) is determined by the training network, then it is passed into the env.step(action) function. env.step(action) will perform the action and return some information:

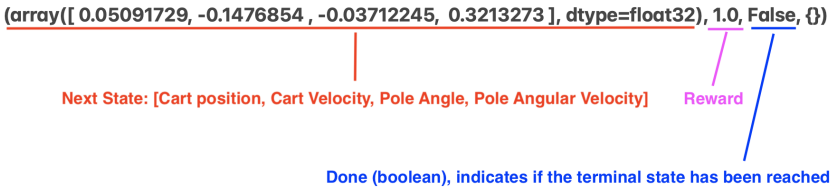

Figure 2 returned tuple of env.step(action)

Therefore, we can keep track of the movement and status of CartPole throughout the game and use the new state of the CartPole after one movement to figure out the next movement with the training network. We then analyse the terminal state of CartPole since the goal of CartPole is to maximise the rewards. An episode in the CartPole environment is the time duration between the initial state and terminal state.

Based on the wiki page of CartPole-v0 environment, the conditions for the terminal state of CartPole are: 1. Pole Angle is greater than ±12°; 2. Cart Position is greater than ±2.4 (centre of the cart reaches the edge of the display); 3. Episode length is greater than 500 for CartPole_v1, 200 for v0.

If any of the conditions above is reached, CartPole will reach the terminal state and terminate the game.

We evaluate the training model by checking the number of episodes the model trains to reach the target average score. The less episodes it takes, the better the training model is. Based on our testing results of the training model, we found out that among the hyperparameters we use in the Deep Q-learning Network, *batch_size, replay_capacity, hid* and *target_update* are the key model hyperparameters that will affect the learning performance significantly with different values assigned.

V. Results

Alternative RL methods are provided to see the difference compared to the reward at training and the average reward. An average duration above 195 would be considered as success in training. Comparing DQN and DDQN and we can see the improvement is little as the system is a simple one dimension problem. The results also show that increasing the hidden units number and layers could speed up the learning process as it could better describe the environments. Dense layers work well in DDQN.

VI. Conclusions

Our project focuses on reinforcement learning, specifically training a cartpole agent to play the game. We implemented and tested three methods. All of them can be successfully trained. The agent using the REINFORCE algorithm takes the longest time to be trained. DQN and Double DQN agents have similar performance that can reach our threshold in much shorter time. By fine tuning, DDQN can outperform DQN. However, all of the algorithms cannot converge after they reach a good performance. They will fall into a "converge then diverge" loop. To further improve our model, how to let the algorithm converge need to be discussed.

References:

1. Kang, C. (2021, May 12). REINFORCE on CartPole-v0. Chan`s Jupyter. https://goodboychan.github.io/python/reinforcement_learning/pytorch/udacity/2021/05/12/REINFORCE-CartPole.html?fbclid=IwAR2XAqLl7M1hSBw7-OYMVVfpNv49R14K_WQvDyRsp8lCOqgXSlG6WWkKgtg
2. Kumar, S. (2020). Balancing a CartPole System with Reinforcement Learning-A Tutorial. *arXiv preprint arXiv:2006.04938*.
3. Kurban, R. (2020, April 28). *Deep Q Learning for the CartPole*. Medium. https://towardsdatascience.com/deep-q-learning-for-the-cartpole-44d761085c2f
4. P. Akshay, D. Vrushabh, K. Sonam, S. Wagh and N. Singh, "Hamiltonian Monte Carlo based Path Integral for Stochastic Optimal Control," 2020 28th Mediterranean Conference on Control and Automation (MED), 2020, pp. 254-259, doi: 10.1109/MED48518.2020.9183150.
5. *PyLessons*. (n.d.). Pylessons.com. Retrieved August 2, 2022, from https://pylessons.com/CartPole-DDQN
6. Hu, R. (2022, June 27). *Apprenticeship Learning with Inverse Reinforcement Learning*. GitHub. https://github.com/rhklite/apprenticeship_inverse_RL
7. *Cart Pole - Gym Documentation*. (n.d.). Www.gymlibrary.ml. https://www.gymlibrary.ml/environments/classic_control/cart_pole/
8. www.youtube.com. (n.d.). *OpenAI Gym: How to Observe the Environment*. [online] Available at: https://www.youtube.com/watch?v=jY9smFvq0i0&ab_channel=DibyaChakravorty [Accessed 2 Aug. 2022].
9. Nagendra, S., Podila, N., Ugarakhod, R., & George, K. (2017, September). Comparison of reinforcement learning algorithms applied to the cart-pole problem. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 26-32). IEEE.