CA2 - Reinforcement Learning: Lunar Lander v2

Name: Quah Johnnie
Admin No: 2007476

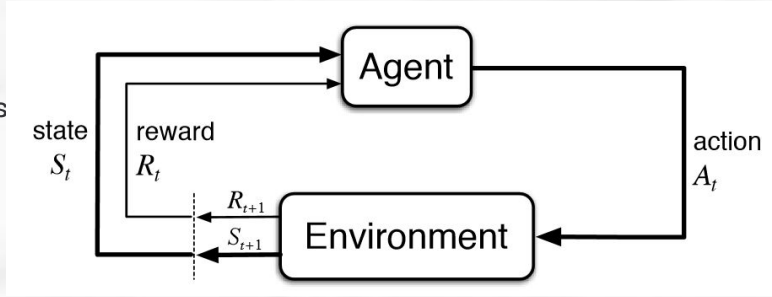
Partner: Bey Wee Loon

Contents (Slides)

1.0: RL & Lunar Lander	RL & Introduction to Lunar Lander Environment
2.0: Training Function & Metrics	DQN as example*
3.0: DDQN	Agent training using Double Deep Q-learning Network (DDQN)
4.0: Actor and Critic /w PPO	Agent training using Actor and Critic with Proximal Policy Optimization (PPO)
5.0: DQN + PER	Agent training using DQN with Prioritized Replay Buffer
6.0: Hyperparameter Tuning	Hyperparameter tuning 5 parameters + Improvement Results
7.0: Objective Final Testing	Final Evaluation & Conclusion

1.0) Reinforcement Learning

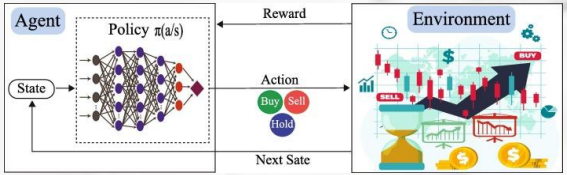
Reinforcement Learning is a type of machine learning where an agent learns to make decisions through trial and error. The agent takes actions in an environment and receives feedback in the form of a reward signal, indicating whether the actions taken were good or bad. Over time, the agent learns to maximize its reward by making better decisions.



Applications: Robotics, gaming, finance etc.

Ping pong robot playing against human

OpenAI agent dominating live Dota 2 environment



Ideology behind reinforcement learning with finance

1.0) RL in Lunar Lunar

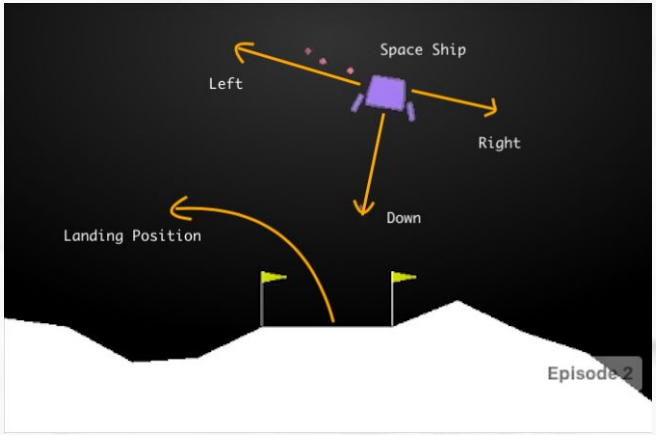
The lunar lander has limited fuel, and the agent must use it carefully to control the lander's position and velocity. The goal of the agent is to land the lunar lander at designated landing site with the least fuel.

There are a total of 4 discrete actions that the lander can take:

0. Do nothing
1. Fire left orientation engine
2. Fire main engine
3. Fire right orientation engine

There are a total of 8 observation spaces:

- The coordinates of the lander in x and y (2)
- Its linear velocities in x and y (2)
- Its angle and angular velocity (2)
- Whether left or right left is in contact with ground (2)



Reward system of Lunar Lander:

- Start with 100 reward
- Firing main engine -0.3 reward per step.
- Firing side engine -0.03 reward per step.
- Crashing -100 reward
- Landing +100 reward
- Landing within flags additional +100 reward
- 1 Leg contact +10 reward (Both legs +20)

2.0) DQN

$$Q(s,a) = E[R_t + \gamma * R_{t+1} + \gamma^2 * R_{t+2} + \ldots | s_t = s, a_t = a]$$

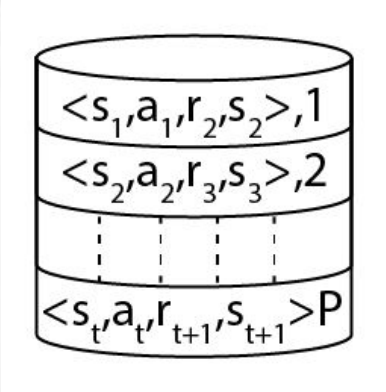
Symbol	Meaning
s_t	represents the state at time t
a_t	represents the action taken at time t
R_t	represents the reward at time t
γ	represents the discount factor, which determines the importance of future rewards relative to immediate rewards ($0 < \gamma \leq 1$)

One important fundamentals of DQN is the Q-function (also known as the state-action value function) is a mapping from states and actions to expected returns.

Important Metrics recorded for all networks:

- Landing Rate: reward >= 120
- Success Rate: reward >= 200
- Average Scores
- Epoch to solve environment

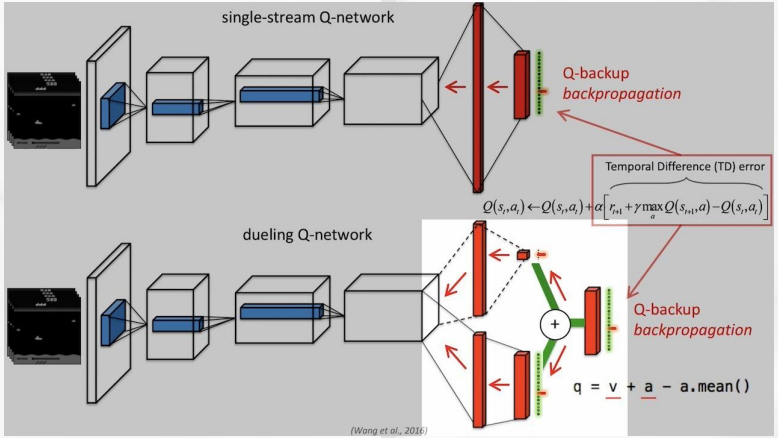
Experience replay is used to stabilize the learning process. The agent learns from the sequence of experiences it encounters as it interacts with its environment



3.0) DDQN

DDQN is an extension of DQN, it addresses a common problem known as overestimation bias. In DQN, the Q-network is used both to select actions and to evaluate the expected future rewards for each action, which can lead to overstimation of Q-values.

DDQN builds upon the basic idea of DQN by using the online network to select the best action, and then using the target network to estimate the corresponding Q-value for that action. This helps to mitigate the overestimation problem in DQN, and leads to improved performance

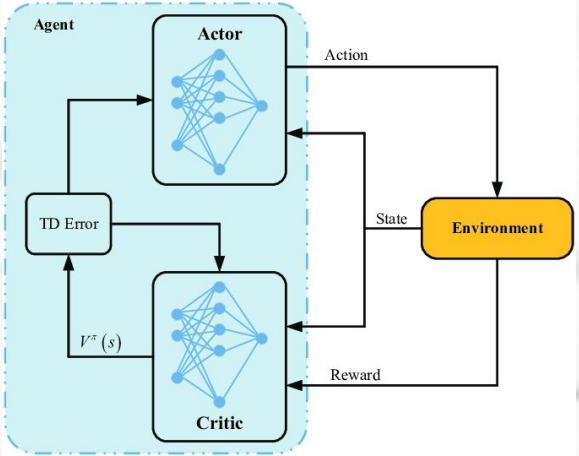


4.0) Actor & Critic /w PPO

The actor network, also known as the policy network, is used to select actions in the environment. It maps the state of the environment to a probability distribution over the available actions. The policy network is trained to maximize the expected reward by choosing actions that are likely to lead to high rewards.

The critic network, on the other hand, is used to estimate the value of being in a particular state, or the value of taking a particular action in a state. The critic network provides a baseline for the actor network to improve its policy. The critic network is trained to minimize the error between its estimated value and the actual return received after taking an action in the environment

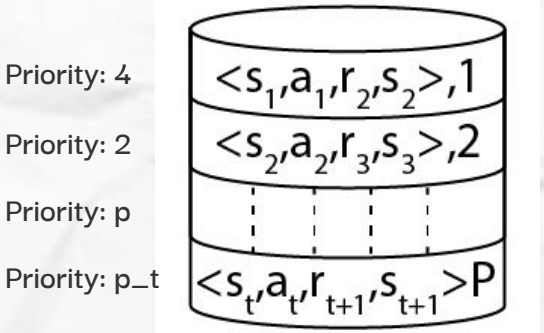
In PPO, Proximal Policy Optimization, the policy network is updated by maximizing a modified objective function that trades off between the improvement in expected reward and the deviation of the new policy from the previous policy.



5.0) DQN /w PER (Prioritized Replay Buffer)

Prioritized Replay Buffer is an extension of the Replay Buffer that assigns a priority to each transition, based on the magnitude of its temporal difference error. The temporal difference error measures the difference between the expected return and the observed return for a given transition. The idea behind prioritized replay is to sample transitions with high priority more frequently, as these transitions are likely to be more informative for the learning process.

Prioritized Replay Buffer can significantly improve the performance of DQN and other deep reinforcement learning algorithms by prioritizing the samples that are likely to lead to faster and more stable convergence. This is because high priority transitions are likely to contain rare or unusual events that can help to correct the neural network's predictions, leading to more robust policies.



6.0) Hyperparameter Tuning

Metrics for choosing hyperparameters: Lowest episodes needed to solve an environment

Hyperparameters Tuned:
- Discount Factor (gamma)
- Maximum Time Alive (each episode)
- Learning Rate (LR)
- Weight Decay (Noise)
- Complexity (Hidden Layers)

Random hyperparameter tuned.

Total trials combination: 960
Trials ran: 30

Total Percentage: 3.125%

```
Trial #30 Finished - Search Time 7.85 Mins
Total Time Elapsed: 611.33 Mins

Hyperparameters         |Trial Values: #30      |Best Trial Values: #29

Learning Rate           |0.0005000              |0.0001077
Max Time Alive          |600                    |800
Discount Factor         |0.9925                 |0.9875
Model Layers            |32                     |32
Weight Decay            |0.00000010             |0.00000100
Epoch To Solve          |915                    |840


Next trial:  [0.00010772173450159416, 1000, 0.985, 64, 1e-08]


Trial ended at trial #30
```

7.0) More Objective Testing

- **How to be more objective when it comes to evaluation besides using the metrics `Lowest Epochs To Solve Environment` ?**

Our approach:

- Train our models on the same training environment `(eg. env seed 1)`
- Test all models on the same testing environment `(eg. env seed 2)`
- All models will only have the chance to be trained for 1000 epochs in the same trianing environment.
- All models will go through 500 episodes of testing in the same testing environment and metrics will be recorded.
- Scores will then be able to be more objectively comparable since all models are trained the in same training environment and tested with the same testing environment. The only different being the Network used to train our agent.

Trained models (All 1000 episodes):

- DQN
- DDQN
- AC /w PPO
- DQN /w PER (Hyperparameter Tuned)

7.0) Conclusion

In this project, we successfully improved our reinforcement learning implementation by constructing and testing a few networks available for reinforcement learning. After experimenting with various methods including DQN, DDQN, and Actor-Critic with PPO, we found that the optimal solution was a DQN with a Prioritized Replay Buffer. To optimize our model further, we conducted hyperparameter tuning on five key variables. Furthermore, to objectively evaluate our models, we used the same training environment (seed: 1) for all networks and tested them on a separate testing environment (seed: 2).

Points in history where the best humans cannot beat AI anymore.
Top image (DeepBlue; 1997)
Btm image (DeepMind; 2016)
(DeepBlue is not trained with RL)

Thanks!

Fin. 完。

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon** and infographics & images by **Freepik**