COSE474-2024F: Final Project Proposal “Develop game AI agent for Lunar Lander using Deep Reinforcement Learning Algorithms”

Kim JinHyeong

1. Introduction

When video games first emerged, people in the world were captivated by them. However, there was another big event making people more captivated, that was the advent of Computer Bot. Before Bot, players were limited to play in fixed environments or comparing their scores with other players individually. The feeling of playing with or against other players simultaneously didn’t exist. With the advent of Bot, even in its simplest form, players experienced a new feeling of competition against a virtual opponent. Yet, these early Bots, just following pre-programmed patterns without considering player actions, eventually lost their appeals due to their lack of adaptability and novelty.

As technology advanced, Game Bot also evolved. Game bots have come to be called Game AI Agents by having AI capabilities. Modern Game AI Agent, hereafter referred to as ’Agent’, can now observe player actions, formulate strategies, and effortlessly achieve their objectives using various algorithms. One of the most famous examples of Agent is “AlphaGo”, developed by DeepMind. AlphaGo’s ability to analyze its opponent’s moves and calculate its next-step made it to win the world champion Lee Sedol. Like this achievement, there is immense potential for further AI agent innovation. And here, this project aims to figure out Agent models by developing and estimating a new Agent.

The goal of this project is to design and develop a Game AI Agent using deep reinforcement learning(DRL) for the game “Lunar Lander”. The objective of the game is to safely land a Lander on the moon’s surface. Although it seems simple, it has various challenges, including precise control of the Lander and efficient fuel use, and time management. The Agent’s performance will be estimated by reward system such as smooth landing, fuel efficiency, and landing time, etc. Through this project, the goal is aim to enhance our understanding of how DRL can be applied to game environments and contribute to the development of Agent decision-making.

2. Problem definition & challenges

The primary objective of this project is to develop AI Agent that can efficiently explore and execute landing trajectories in the Lunar Lander game. During the development process, several challenges must be addressed:

2.1. Complexity of Lander condition

The Lander has various, continuous variables, such as position, velocity, angle, and fuel levels. Because of the complexity of these variables, the Agent must generalize learning effectively during multiple iterations. Achieving this generalization can be difficult with traditional reinforcement learning methods(RL), so DRL methods would be required.

2.2. Continuous decision

The Agent must decide the direction of Lander and whether or not to turn on the engine in that direction. These decisions must be made correctly at the right time and always until the Lander lands. To learn and make these decisions properly is a difficult challenge of this project.

2.3. Reward system

A clear reward system is required to guide the Agent’s behavior and estimate whether it has successfully landed. The Agent should receive positive rewards for safe landings, less fuel consumption and short landing time. Balancing this reward structure is crucial to encouraging the desired behaviors while discouraging suboptimal strategies.

2.4. Stability and Convergence

DRL algorithms can face instability and slow convergence due to the complexity of high-dimensional input spaces and the learning process. Ensuring the Agent continuously improves without oscillation or divergence is a key challenge.

2.5. Exploration vs Exploitation

The Agent must balance between Exploration(new action) and Exploitation(previous best action). Maintaining this balance is critical for the Agent to effectively learn optimal strategies without getting stuck in local minima.

By addressing these challenges, this project aims to contribute to the development of a Lunar Lander AI Agent and advance the understanding of game AI Agent development using DRL.

3. Related Works

Gymnasium is a platform provided by OpenAI where various reinforcement learning problems can be tested. Gymnasium also includes a pre-built Lunar Lander agent. In Gymnasium, the Lander's action space consists of four actions:

- do nothing(0)
- fire left orientation engine(1)
- fire main engine(2)
- fire right orientation engine(3).

Also, the lunar environment includes gravity and wind, and at the beginning, random forces affects the lander.

In Gymnasium, the Agent's reward structure is as follows:

- is increased/decreased the closer/further the lander is to the landing pad.
- is increased/decreased the slower/faster the lander is moving.
- is decreased the more the lander is tilted (angle not horizontal).
- is increased by 10 points for each leg that is in contact with the ground.
- is decreased by 0.03 points each frame a side engine is firing.
- is decreased by 0.3 points each frame the main engine is firing.
- The episode receive an additional reward of -100 or +100 points for crashing or landing safely respectively.
- An episode is considered a solution if it scores at least 200 points.

For this project, additional complexity will be added such as speed, total fuel, and time. The lander's directional control will also be expanded from 3 directions to 8. Based on these modifications, the agent will be improved and its performance will be evaluated, comparing it with the original agent from Gymnasium.

4. Datasets

The original Lunar Lander game environment can serve as a benchmark dataset for training agents using DRL. However, the game is quite simple and lacks some appealing elements that make it more like game. To address this, additional objectives such as fuel limits, landing speed restrictions, structured landing zones, landing angle constraints, and time limits will be added to make it more challenging and enjoyable. The agent will be trained in this more complex environment.

5. State-of-the-art methods and baselines

In traditional Reinforcement Learning (RL), the agent's learning process was based on the interaction between environment perception, actions, and reward systems. After that, the results were recorded in a table, allowing the agent to adjust its behavior in order to maximize future rewards. However, as environments became more complex, the number of states increased, and the variety of actions expanded, making it difficult to store all data in a table.

To address this, **Deep Reinforcement Learning (DRL)** emerged. Unlike the traditional method, DRL does not store information in a table, but instead selects only important features from the expanded state space. Therefore, DRL can address more complex and various states than classic RL.

Gymnasium uses RL models to train AI agent. And another some individual projects use DRL model. This project also use DRL instead of RL, because of a more complex environment than the original Lunar Lander. The performance of the agent trained by DRL will be compared with that of an agent trained by classic RL methods.

6. Schedule & Roles

- 2024.11.01 $\sim$ 2024.11.07 Create a more complex Lunar Lander game.
- 2024.11.08 $\sim$ 2024.11.25 Develop and train the Agent.
- 2024.11.26 $\sim$ 2024.12.05 Measure, compare Performance and modify Agent.

References

Gymnasium. An api standard for reinforcement learning with a diverse collection of reference environments, 2024. https://gymnasium.farama.org/ [Accessed: 2024-10-10].

Kwiatkowski, A., Towers, M., Terry, J., Balis, J. U., Cola, G. D., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Tan, H., and Younis, O. G. *Computational*

Complexity of Machine Learning. PhD thesis, Cornell University, 2024.

TensorFlow. Introduction to rl and deep q networks, 2023. `https://www.tensorflow.org/agents/tutorials/0_intro_rl?hl=en` [Accessed: 2024-10-15].