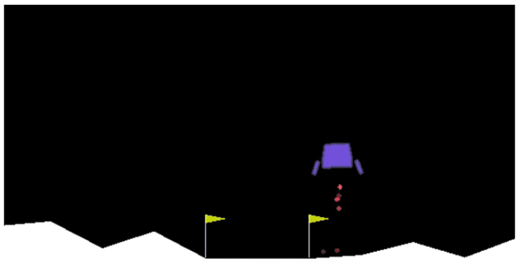
CS221 Fall 2018 Project
An AI agent for Lunar Lander

Collaborators: Amey Naik, Prabhjot Singh Rai, Abhishek Bharani
SU Net IDs: ameynaik, prabhjot, abharani

1 Introduction

The purpose of this project is to build an AI agent to play Lunar Lander game to safely land on a landing pad. We are making use of Box2D Lunar Lander available on OpenAI gym. Landing pad is always at coordinates (0,0). Reward for reaching the landing pad is about 100 to 140 points, depending on the speed. If lander moves away from landing pad it loses reward back. Episode finishes if the lander crashes or comes to rest, receiving additional -100 or +100 points. Each leg ground contact is +10. Firing main engine is -0.3 points each frame. Solved is 200 points.

Figure 1: Game Environment

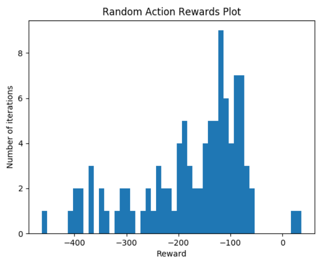


Landing outside landing pad is possible. Fuel is infinite. Four discrete actions available: do nothing, fire left orientation engine, fire main engine, fire right orientation engine. Our state at any given frame is defined by 8 parameters: horizontal position (x-axis), vertical position (y-axis), orientation (theta), linear velocity (v), angular velocity (w), state of each landing leg (left and right). Moreover, our environment, which is the surface of the moon, keeps changing with each episode.

2 Baseline

We define multiple baselines:

(a) Random Action Baseline: Taking random action at any given state out of the available actions. We see that in almost all the cases, we get negative rewards. In order to check if taking a single action may lead to the optimum, we saw that this policy performed even worse than taking random actions.



(b) Start with randomly picking an action for a new state that is explored, if the state gets explored in the episode ‘e1’, then in the next episode ‘e2’ if you encounter that state, pick the previously stored action for that state. Our baseline will be a greedy algorithm which will act to fetch maximum rewards but does not account for future states and early convergence.

3 Oracle

We define multiple oracles for this problem in the increasing order of their performance and complexity:

(a) Human playing score.

(b) Maximum score that one can achieve which is 140 [reach landing pad] + 100 [stopping without crashing] + 10*2 [each leg ground contact] + 200 [solving the episode]

4 Metrics

We can evaluate the performance of agent by comparing the score with the scores from the oracle and in least number of episodes. There is leader-board to compare how fast the learning is of our model against others. [1]

5 Challenges And Approach

Firstly, this problem has a very large state space, we can not expect a simple epsilon-greedy algorithm to explore everything in a reasonable amount of time. The next logical step is to apply Q-learning. This may help us take the best direction but may not provide a detailed planning. In general, it may not understand the game dynamics from a general point of view. By using function approximation, we can generalize well to unseen states. Reflex-based DNN(DQN) will be used by us to approach this problem, particularly different variants of DQN such as DQN with prioritize replay[2], Double DQN[3], and Dueling network architecture [4] to solve this problem.

References

[1] https://github.com/openai/gym/wiki/Leaderboard.

[2] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2015.

[3] H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015.

[4] Z. Wang, N. de Freitas, and M. Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015.