An AI Agent for Lunar Lander

CS 221, Fall 2018, Category: Reinforcement Learning

Prabhjot Singh Rai (prabhjot), Abhishek Bharani (abharani), Amey Naik (ameynaik)

Stanford University

Goal

- Design an AI agent that learns to safely land on a landing pad. A successful solution is expected to consistently land safely on the target area with both legs touching the ground.
- Getting average reward of 200 over 100 consecutive episodes is considered solved.

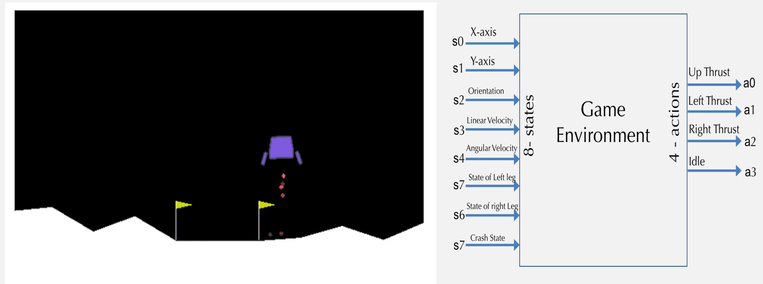


Figure 1: Lunar Lander V2

Approach & Infrastructure

- We modelled the Box2D lunar lander problem in Open AI gym framework as a Markov Decision Process and looked to solve it through variants of Q-Learning.
- Starting with a linear approximation for each state variable, we moved to advanced Q-learning methods, such as Full DQN, Double DQN and Dueling Network Architecture.
- We tuned the hyper parameters to solve the game in least number of episodes and then fixed the hyper parameters across different variants for performance comparison.

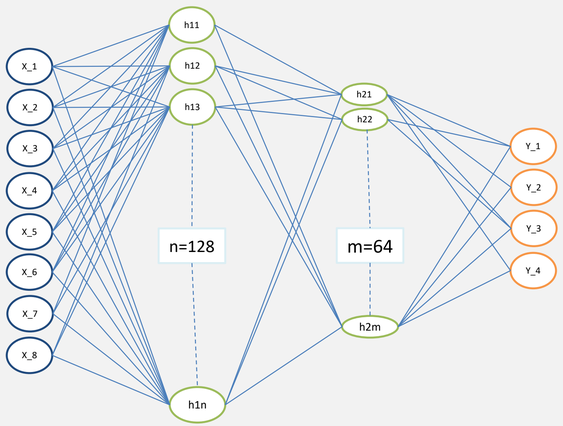


Figure 2: Deep Neural Network used for DQN variants

Optimization Strategy

We used DQN and it's different variants to solve this problem in least number of episodes. The algorithm incorporated :

- MSE loss to train DNN, since input to our NN is 8 dimensional state vector (not an image)
- Hyper-parameter tuning specifically for Lunar Lander game env.
- Epsilon-Greedy policy

DQN variants Explored -

1. **Full DQN:** Separate Target network $\tilde{Q}$ provides stable values and allows the algorithm to converge to the specified target:
$$Q(s,a) \rightarrow r + \gamma \, max_a \tilde{Q}(s',a)$$

2. **Double DQN:** Because of the max in the above formula, the network suffers from maximization bias, possibly leading to overestimation of the Q function's value and poor performance.
$$Q(s,a) \rightarrow r + \gamma \, \tilde{Q}(s', argmax_a Q(s',a))$$

3. **Dueling DQN:** Separate the estimators into the one that estimates the state value V(s) and other estimates the advantage for each action A(s,a).
$$Q(s,a) \rightarrow A(s,a) + V(s)$$
$$Q(s,a;\theta,\alpha,\beta) \rightarrow V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{A}\sum_{a'} A(s,a';\theta,\alpha)\right)$$

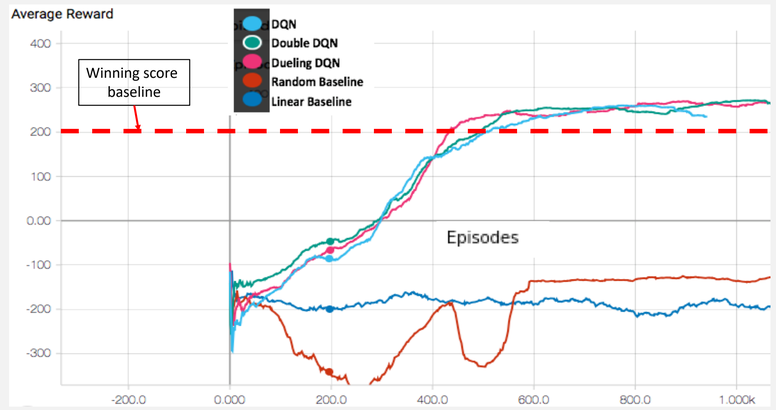
Hyperparameter	Set 1	Set 2	Set 3	Set 4(best)
gamma	0.99	0.99	0.99	0.99
Epsilon(max, min, decay)	(1, 0, 0.998)	(1,.01,0.995)	(1,.01,0.998)	(1,.01,0.995)
Learning Rate	0.0001	0.0001	0.0001	0.0001
DNN layers	[32,32]	[128,32]	[128,64]	[128,64]
Loss function	MSE	MSE	MSE	MSE
Batch Size	32	32	64	64
Replay Memory Size	2^16	2^16	2^16	2^16

Figure 3: Different set of hyper-parameters were tried to get best performance

Discussion

- Dueling DQN Networks outperforms the baseline and other variants of DQN.
- Initially, it was a challenge to get an agent to learn faster, increasing replay memory size helped a lot. DQN network started giving decent score after 900 episodes. After that we tuned the other hyperparameters to get better performance.
- The drop in the rewards after model learns is because after many consecutive successes, the replay buffer won't have many failure cases to train on. So, it will 'forget' how to recover from many failure cases.

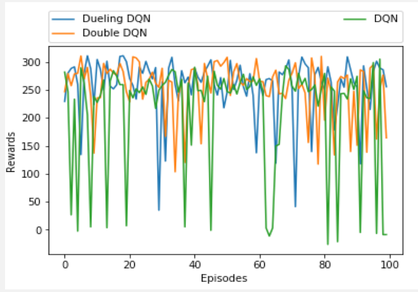
Implementation & Results



Plotting Avg. Rewards for different approaches on tensor board

Model\Metric	Average Score	No. of episodes to reach winning score of 200
Baseline	-200	Never
Linear Optimizer	-150	Never
DQN	123	525
Double DQN	220	500
Dueling DQN	225	435

Comparison of different models



Evaluating Performance of different DQN networks

Future Work

- Currently our algorithm takes 450 Episodes to learn and consistently win the game.
- We plan to try Prioritized Experience Replay and A3C algorithm to improve the DQN convergence speed.
- Conduct more extensive Hyper-parameter tuning.

References

[1] https://github.com/openai/gym/wiki/leaderboard#lunarlander-v2
[2] https://jaromiru.com/2016/10/03/lets-make-a-dqn-implementation/
[3] Human-level control through deep reinforcement learning MnihEtAlHassibis
[4] Dueling Network Architectures for Deep Reinforcement Learning by Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, Nando de Freitas