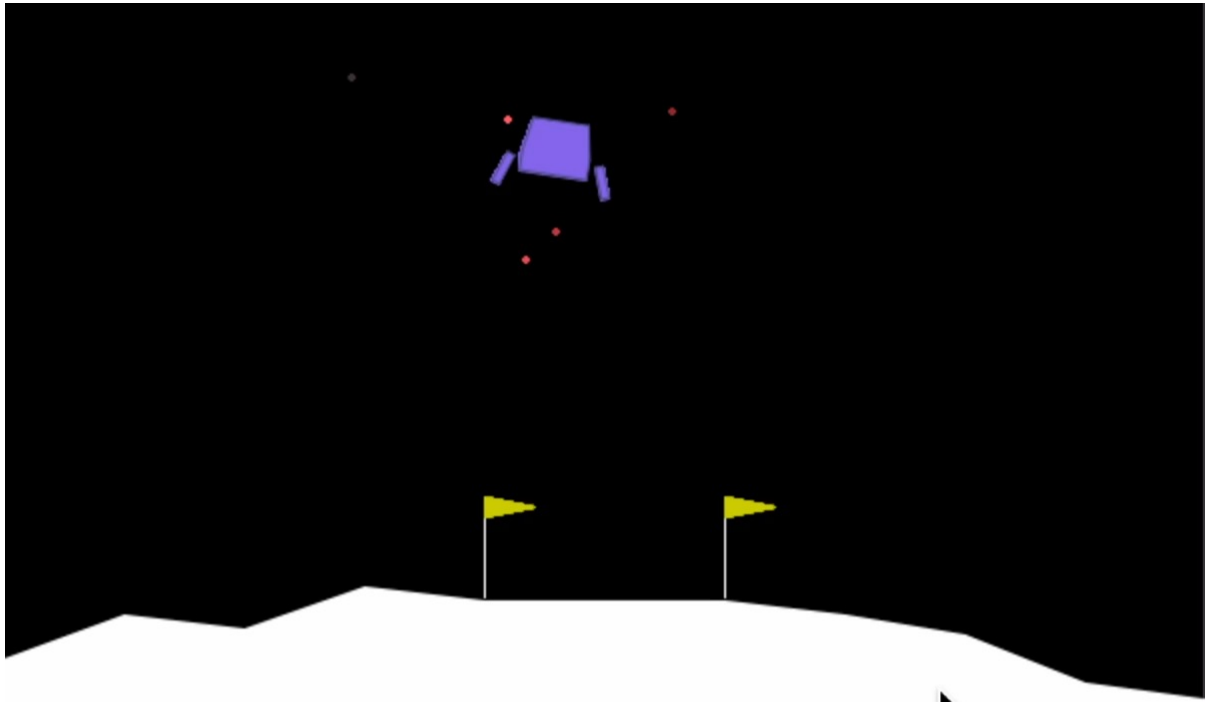
Lunar Lander

Krishan Subudhi

Final Project: University of Washington

Goal

Direct the agent to the landing pad as softly and fuel-efficiently as possible.



gym.openai.com

Algorithms

Atari

Box2D

Classic control

MuJoCo

Robotics

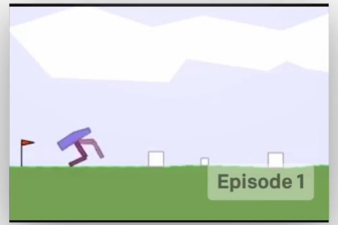
Toy text EASY

Third party environments

Box2D

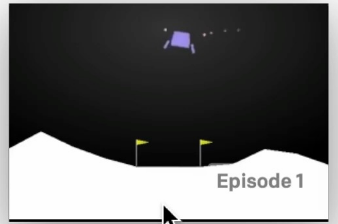
Continuous control tasks in the Box2D simulator.

BipedalWalker-v2
Train a bipedal robot to walk.

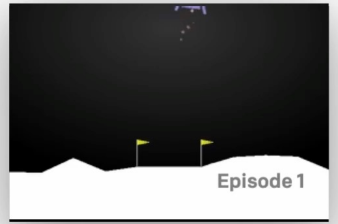


BipedalWalkerHardcore-v2
Train a bipedal robot to walk over rough terrain.

CarRacing-v0
Race a car around a track.



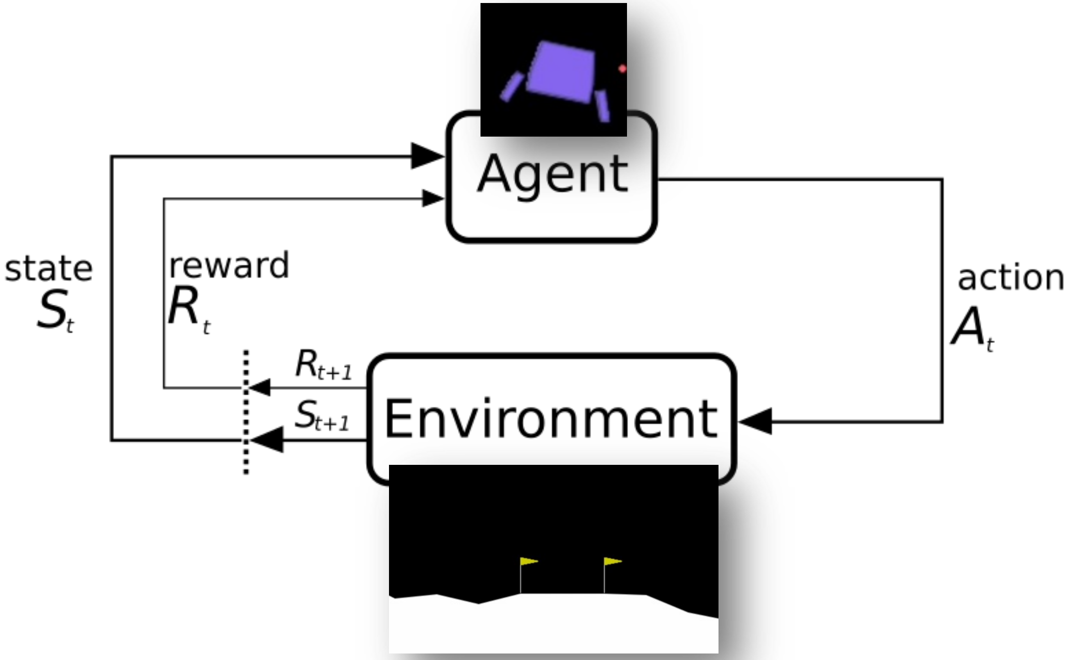
LunarLander-v2
Navigate a lander to its landing pad.



LunarLanderContinuous-v2
Navigate a lander to its landing pad.

Setup

- x coordinate
- y coordinate
- horizontal velocity
- Vertical velocity
- angle
- angular velocity
- Is Left leg on ground?
- Is right leg on ground?



1. do nothing
2. fire left orientation engine
3. fire main engine
4. fire right orientation engine

Rewards

- Moving from the top of the screen to the landing pad and coming to rest : +100 to +140
- If the lander crashes : -100
- If it comes to rest : +100
- Each leg with ground contact : +10
- Firing the main engine : -0.3
- Firing the side engine : -0.03

Approach

Why Reinforcement Learning?

We are treating the environment as a blackbox.

We don't know T(s,a,s') and R(s,a,s')

Algorithms tried

Random baseline

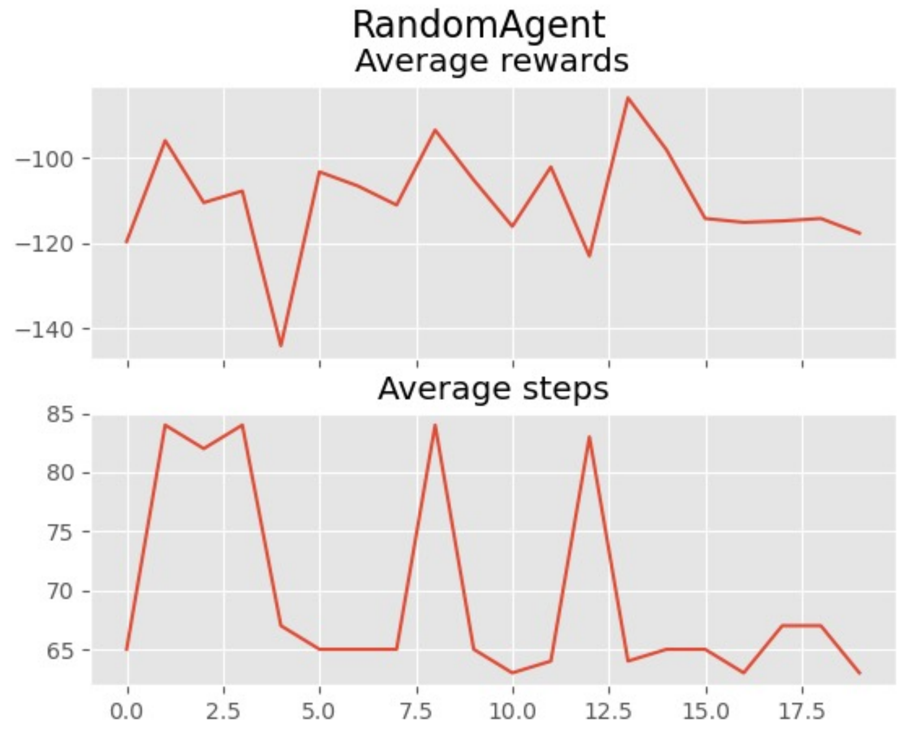
Q Learning

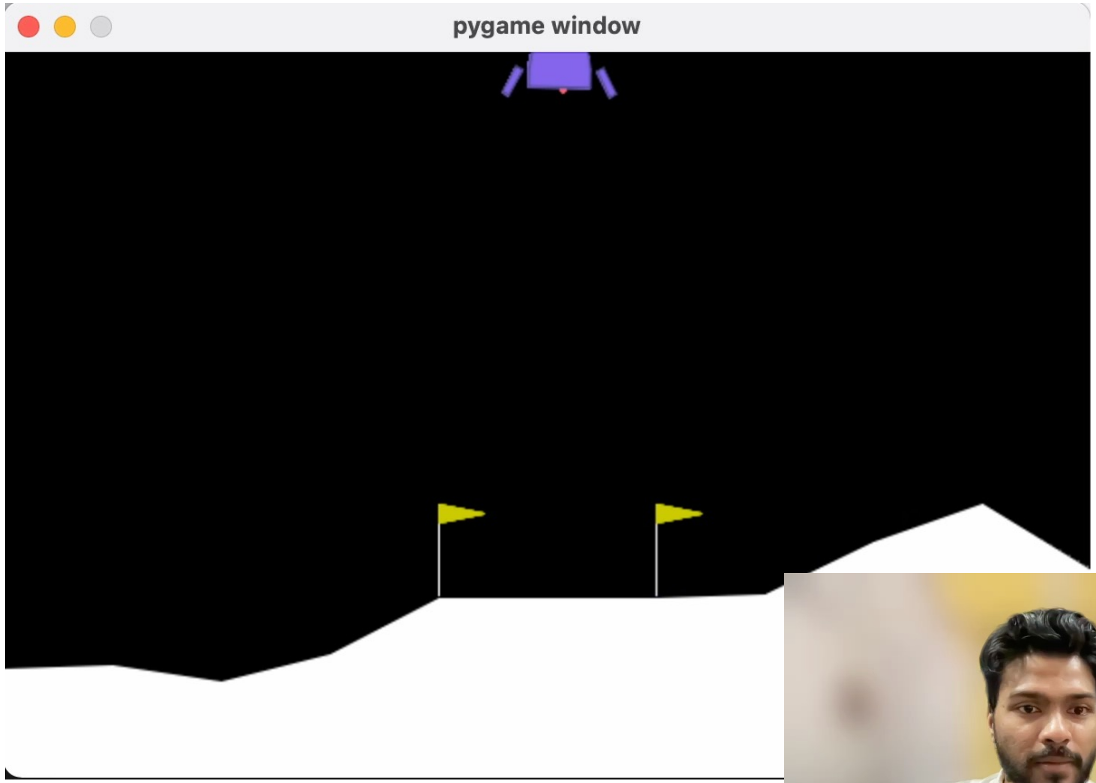
Approximate Q Learning

Deep Q learning

Random Agent

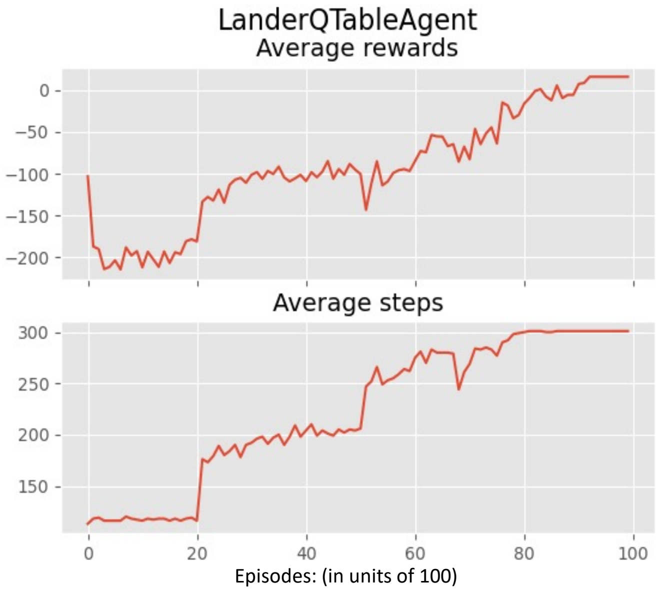
- Chooses action randomly
- Baseline





Q Learning

- Maintains a dictionary of
  - $\{(\text{state}, \text{action})\} : QValue$
- Our states are continuous. Need to discretize.
- Learning
  - $Q_{sample} = R(s,a,s') + \gamma \max_{a'} Q(s',a')$
  - $Q_{t+1} = (1-\alpha)Q_t + \alpha\, Q_{sample}$
- Hyper parameters
  - episodes = 10000
  - α = 0.3
  - γ = 0.95
  - ϵ = step function from 0.5 to 0



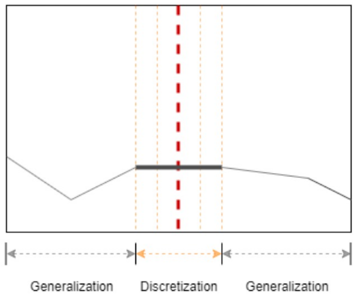
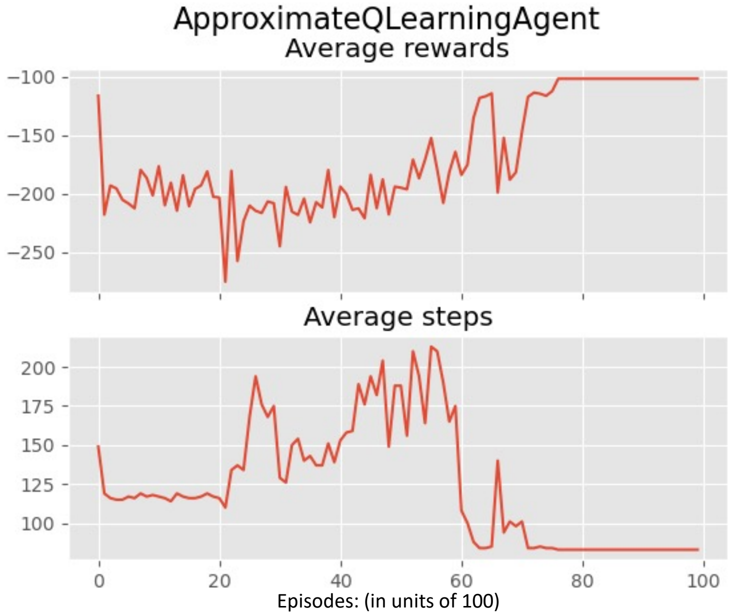
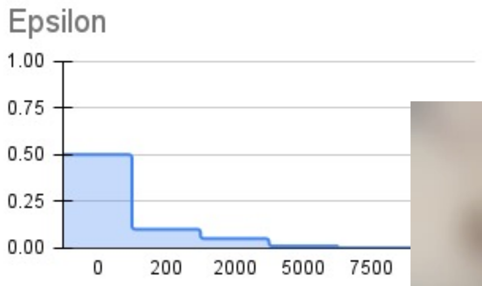


Image from arXiv:2011.11850v1 **[cs.LG]**

Approximate Q Learning

- Q table had a lot of (state, action) pairs.
- Approximate Q Functions
  - $Q(s,a) = \sum_i w_i \; f_i(s,a) + b\,ias$
- Learning after transition (s,a,r,s')
  - $diff \; = [R(s,a,s') + \gamma \max_{a'} Q(s',a') \; - Q(s,a)$
  - $w_i \; = w_i + \; \alpha \; * \; diff \; * f_i \; (s,a)$
- Hyper parameters
  - episodes = 10000
  - α = 0.3
  - γ = 0.95
  - ϵ = step function from 0.5 to 0
- Findings
  - Converges to lower average reward.
  - Increasing number of exploration steps does not help.
  - Probably underfitting issue.
  - How to verify that algorithm/code is accurate?





Custom game : 1Dtarget

- Goal
  - given a point as target (100) , and a starting state (1) , agent should reach the target as quickly as possible.
- State space
  - agent x position
- Action space
  - Left
  - right
- Reward
  - 1 if agent reaches target

```
[ 0 0 0 0 0 100 0 0 1 0 0]
```

Approximate Q Learner on 1DTarget

[0 0 1 0 0 100 0 0 0 0 0]

[0 0 0 1 0 100 0 0 0 0 0]

[0 0 0 0 1 100 0 0 0 0 0]

[0 0 0 0 0 100 0 0 0 0 0]

Struggle at the edges

[0 0 0 0 0 100 0 1 0 0 0]

[0 0 0 0 0 100 1 0 0 0 0]

[0 0 0 0 0 100 0 1 0 0 0]

Learned Policy:

0: right

1: right

2: right

3: right

4: right

5: right

6: right

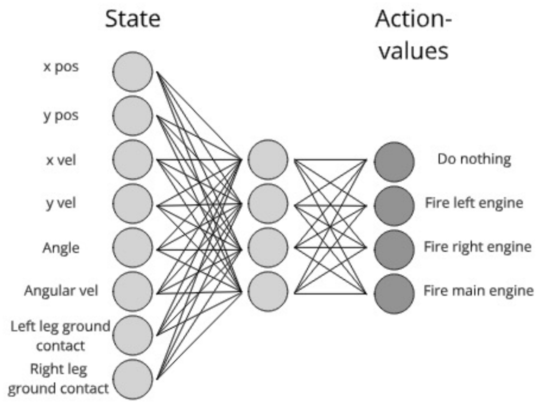
7: left

8: left

9: left

10: left

Deep Q Learning



Neural network

Fetches training samples from past memory

Advantages

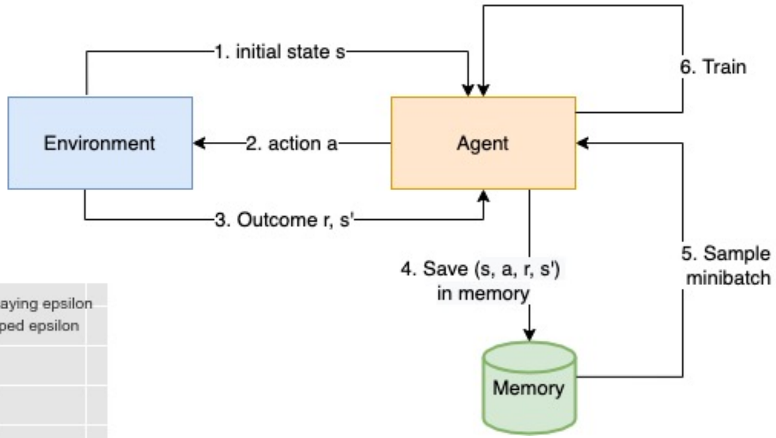
Non-linear QValue approximation.

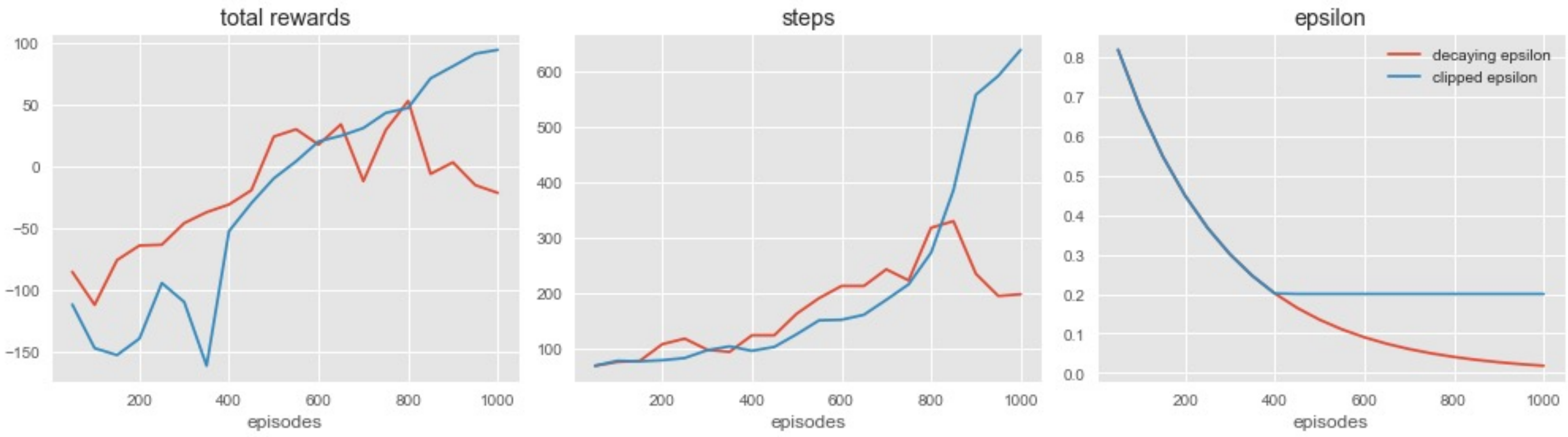
No feature engineering

Challenges

samples are not iid.

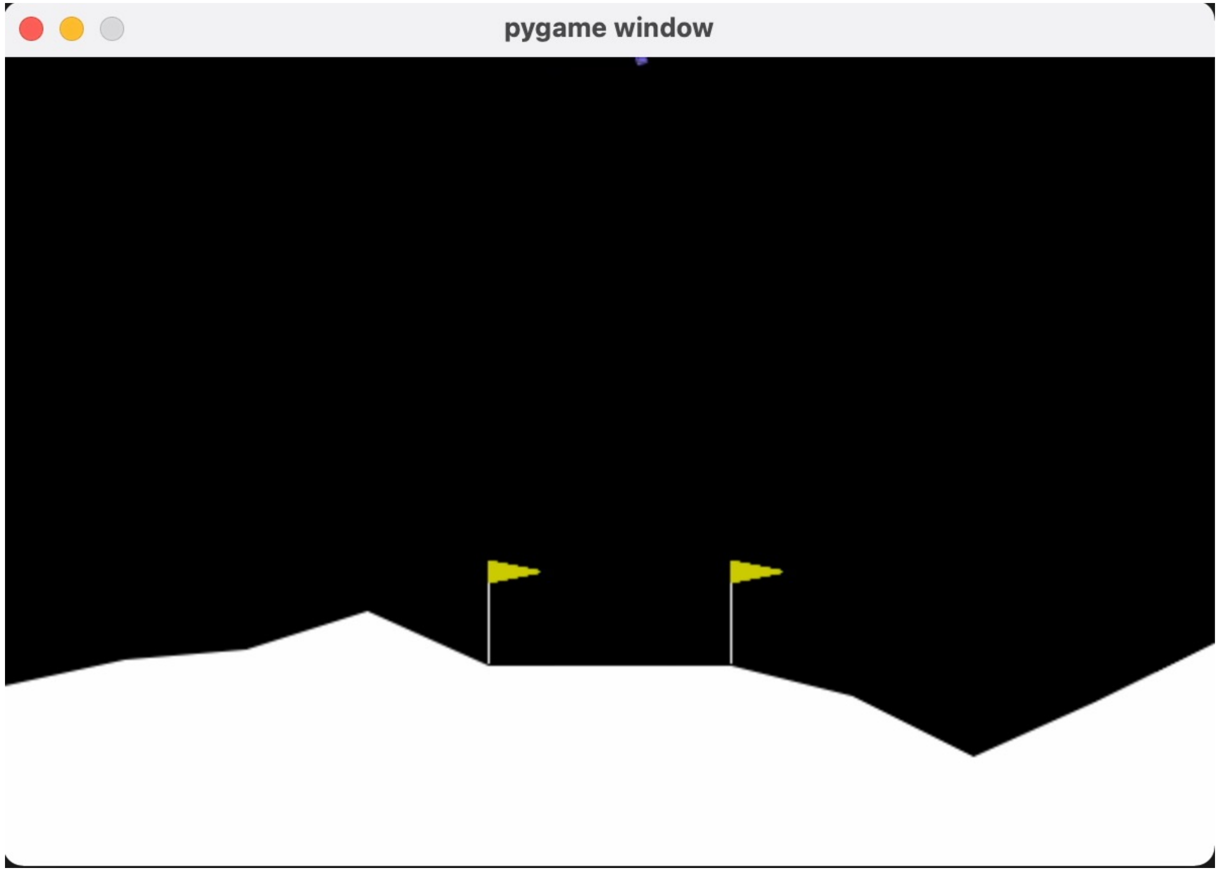
Label are derived bellman's equation.

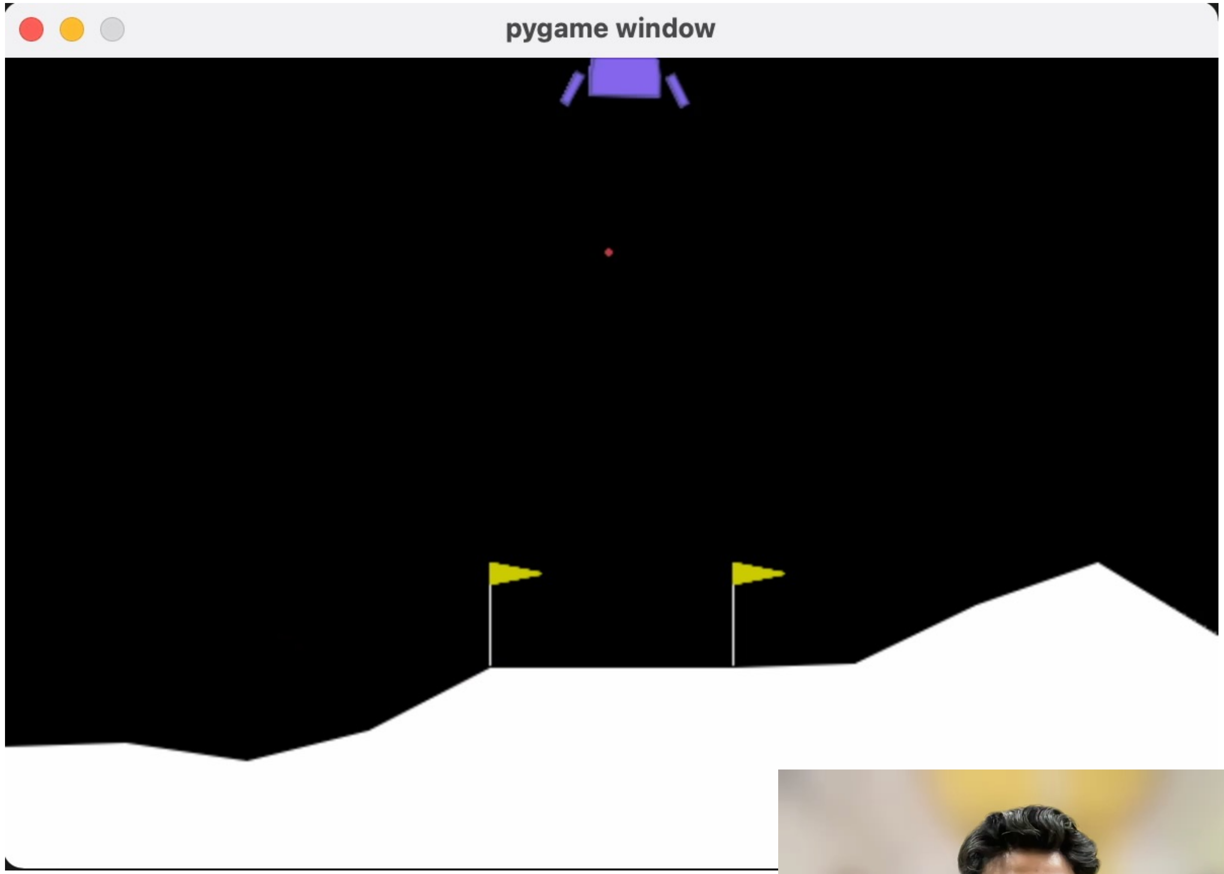




- Maximum posit
- Exploration is in
- Replay quality is
- Low exploration

Before and after training with DQL

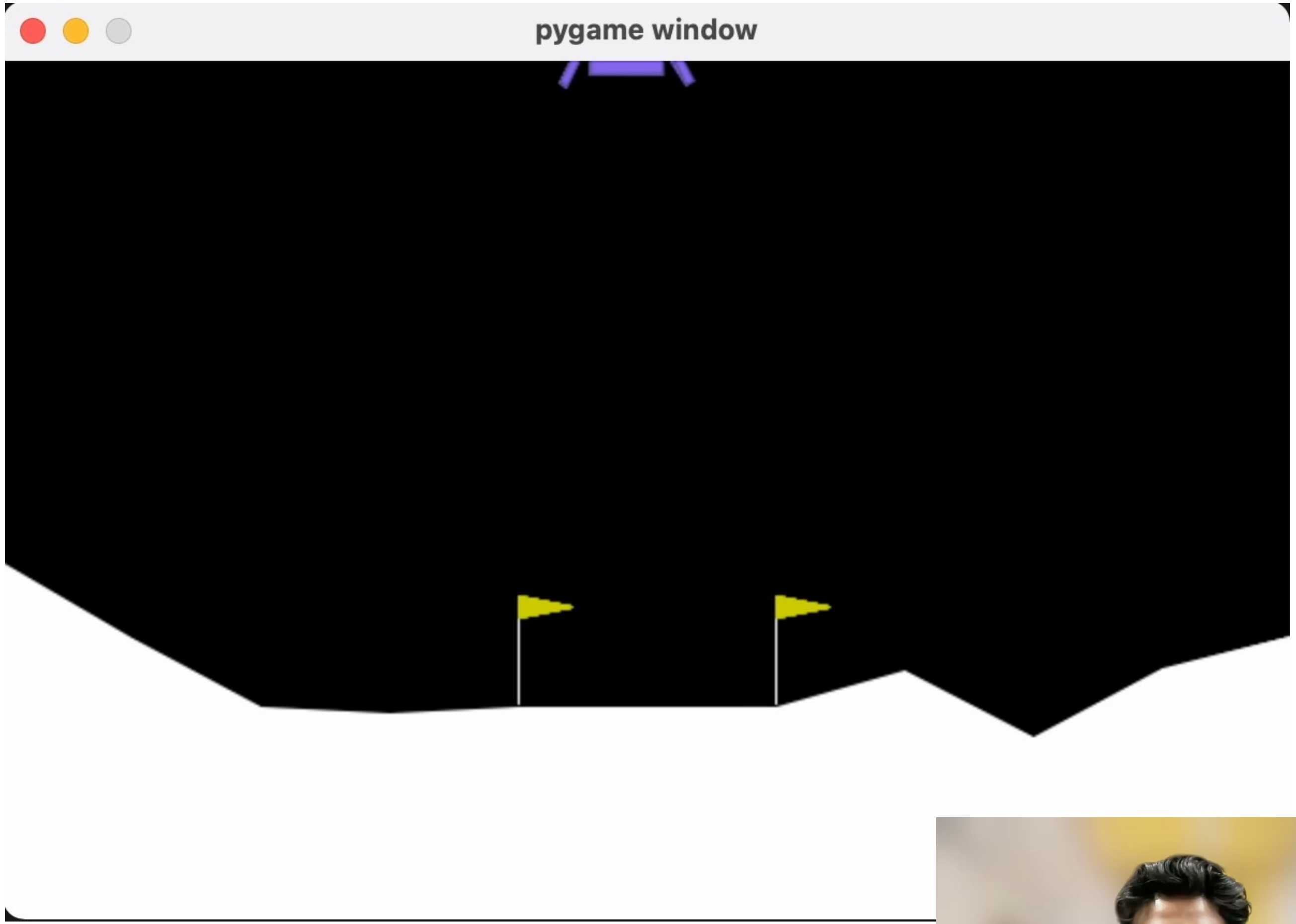




steps 328
reward +272.64
action_counts = [14, 83, 110, 121

Evaluation under different conditions

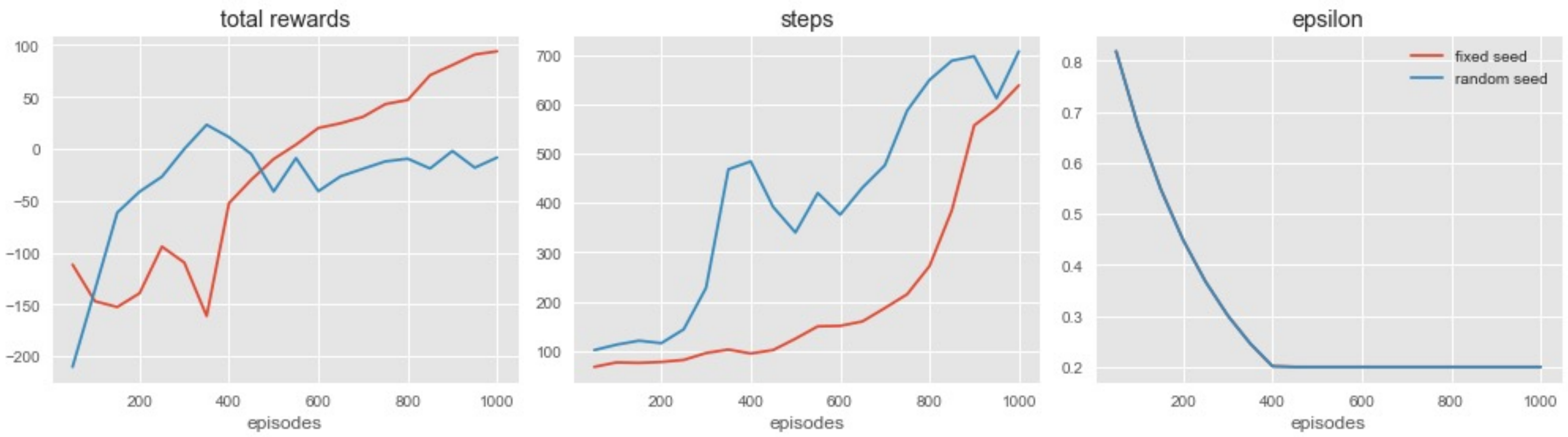
- `s = env.reset(seed=seed)`
- Seed determines the initial force and terrain.
- Agent was trained with fixed seed.
- Agent performs poorly for random seeds.
- Overfitting?

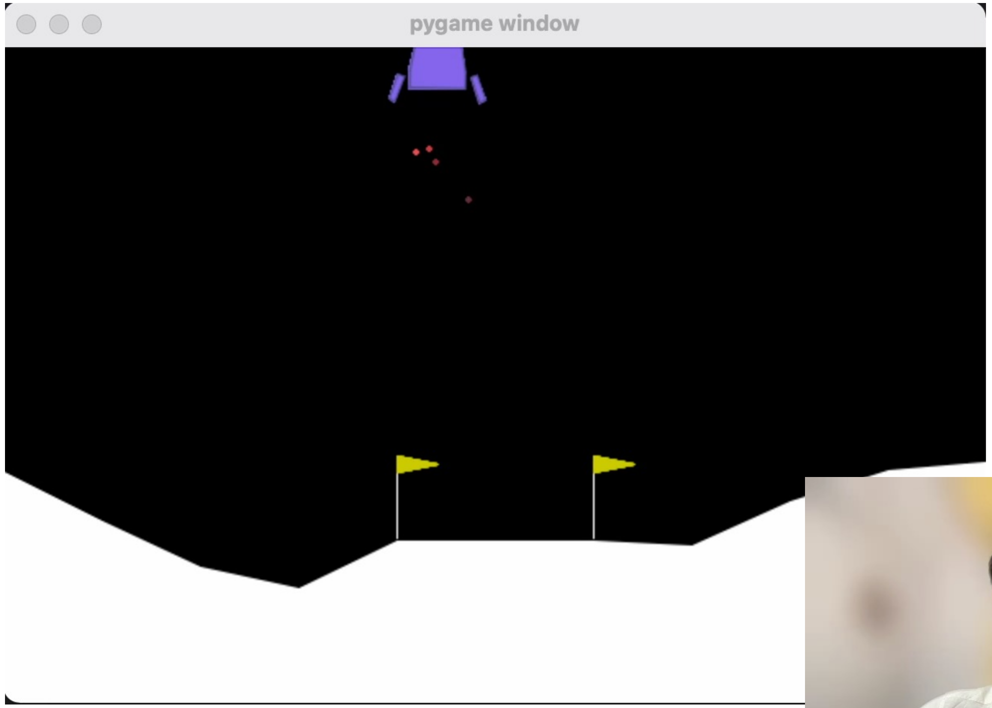


ep 0 steps 108 reward +6.31, action_counts = [0, 28,
ep 1 steps 581 reward +190.21, action_counts = [17,
ep 2 steps 224 reward -57.21, action_counts = [6, 44,
ep 3 steps 215 reward -69.31, action_counts = [7, 39,
ep 4 steps 193 reward +1.95, action_counts = [6, 52,

Training with Random seed

- More diverse states encountered.
- Average reward is low. Needs more hyperparameter tuning.
- ideas?





Learnings

- RL is difficult to debug. Start simple.
  - 1DTarget, Q tables
- Hyper parameter tuning is important.
  - Most important : ϵ
  - Other important hyper params : α, number of episodes, max steps per episode, replay memory size, γ
- Visualization is important
  - Renders, graphs – gather as much info as possible during training
- GPUs don't help in Deep Q Learning. Each step requires inferencing the NN.
- RL is fun!

References

Project source : https://github.com/krishansubudhi/lunarlander

Reference paper: arXiv:2011.11850v1 **[cs.LG]**

Environment: OpenAI Gym LunarLanderV2