Planning a path for autonomous mobile robots in an obstacle-free 2D environment with deep reinforcement learning

Nguyen Van Luong

20173249

Flow

Introduction

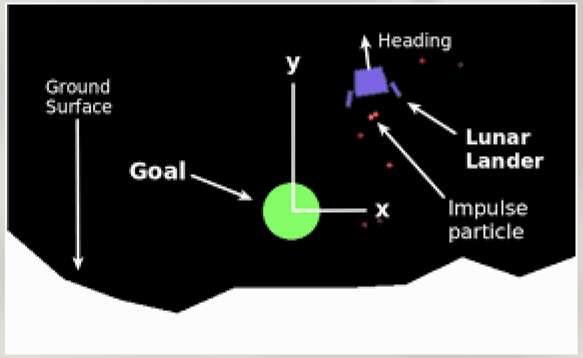
- Mobile robots are robots that can move from one place to another autonomously, that is, without assistance from external human operators. [1]
- Application of mobile robots:
  - In many sectors:
    - Exploration (planet, undersea, polar)
    - Aid
    - Army
    - Transportation (delivery, heavy cargo, ...)
    - Inspection (factories, bridges, etc.)
  - In many services
    - Indoor (vacuuming, lawn mowing, laundry, ...)
    - Medical
    - Transportation
    - Entertainment, pets
    - Customer care

[1] Tzafestas, S. G. (2013). Introduction to mobile robot control. Elsevier

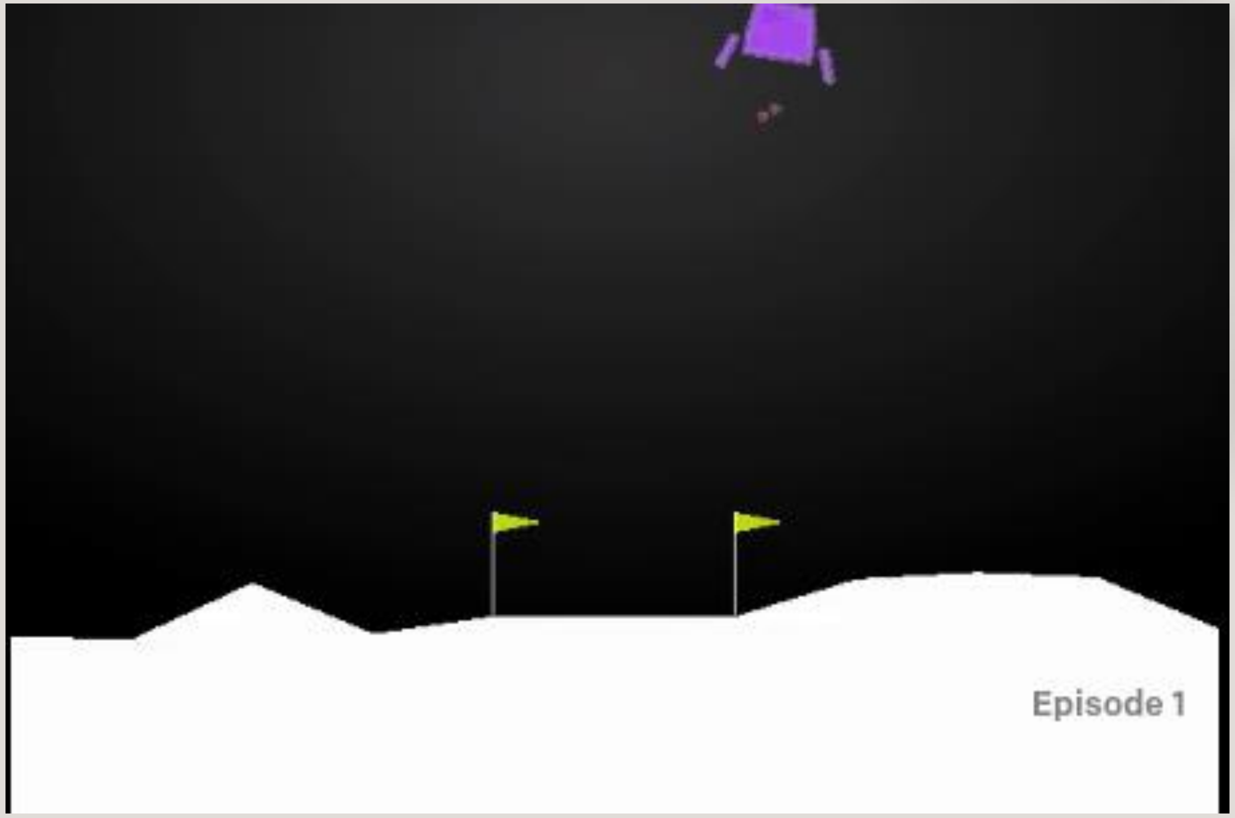
Introduction

- An ideal mobile robot would be a fully connected and closed set: power source, sensor and processor, but different degrees of autonomy:
    - Teleoperation (Viễn thông) (ROV of remote operating vehicles, eg robot war, mine clearance, surgical robot ...)
    - Semi-autonomous / Supervised (example: tracking on Mars ...)
    - **Fully autonomous** (for example: Robot vacuum cleaner, unmanned car ...)

Problem



- Lunar Lander Continuous [2]
  - The Lunar Lander example is an example available in the OpenAI Gym (Discrete) and OpenAI Gym (Continuous) where the goal is to land a Lunar Lander as close between 2 flag poles as possible, making sure that both side boosters are touching the ground.
  - We can land this Lunar Lander by utilizing actions and will get a reward in return - as is normal in Reinforcement Learning.



[2] OpenAI Gym – Lunar Lander v2 < https://gym.openai.com/envs/LunarLander-v2 >

Problem

Observation Space

Action Space

Reward Function

Constraint & Scope

Problem

- Observation Space:
  - The observation space is illustrated by a "Box" containing 8 values between [-inf, inf] these values are:
    - Position X
    - Position Y
    - Velocity X
    - Velocity Y
    - Angle
    - Angular Velocity
    - Is left leg touching the ground: 0 OR 1
    - Is right leg touching the ground: 0 OR 1

Problem

- Action Space:
  - Discrete (Discrete Action Space with 4 values):
    - 0 = Do Nothing
    - 1 = Fire Left Engine
    - 2 = Fire Main Engine
    - 3 = Fire Right Engine
  - Continuous (Box Action Space with 2 values between -1 and +1):
    - Value 1: [-1.0, +1.0] for main engine where [-1.0, 0.0] = Off and [0.0, +1.0] = On
    - Value 2:
      - [-1.0, -0.5]: Left Engine
      - [-0.5, 0.5]: Off
      - [0.5, 1.0]: Right Engine

Problem

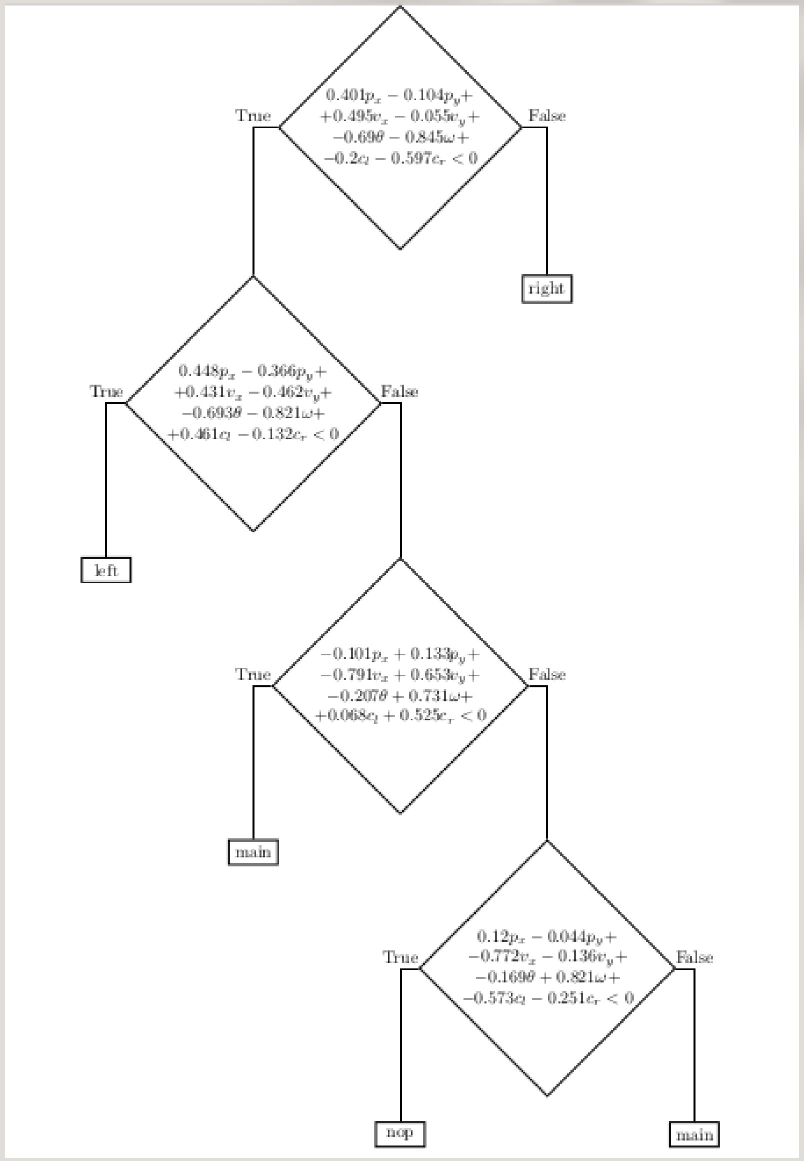
- Reward Function:
  - The Reward Function is a bit more complex and consists out of multiple components:
    - [100, 140] points for Moving to the landing pad and zero speed
    - Negative reward for moving away from the landing pad
    - If lander crashes or comes to rest it gets -100 or +100
    - Each leg with ground contact gets +10
    - Firing the main engine is -0.3 per frame
    - Firing the side engine is -0.03 per frame
    - Solved is 200 points

Problem

- Constraint
  - Timesteps
  - Map
  - Avoid collision
- Scope
  - Maximum Reward (closest to target coordinates, least energy consumed, avoid collision, grounding)

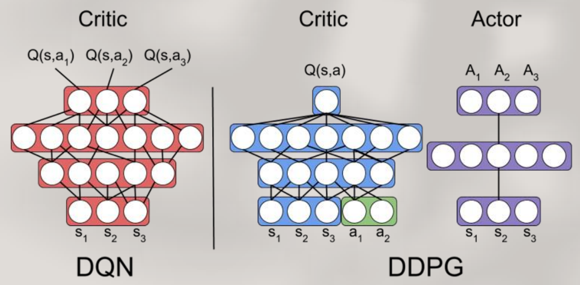
Related work

- Best oblique decision tree evolved in the LunarLander-v2 environment (Discrete). [3]



[3] Lucio Custode, Leonardo, and Giovanni Iacca. "Evolutionary learning of interpretable decision trees." arXiv e-prints (2020): arXiv-2012.

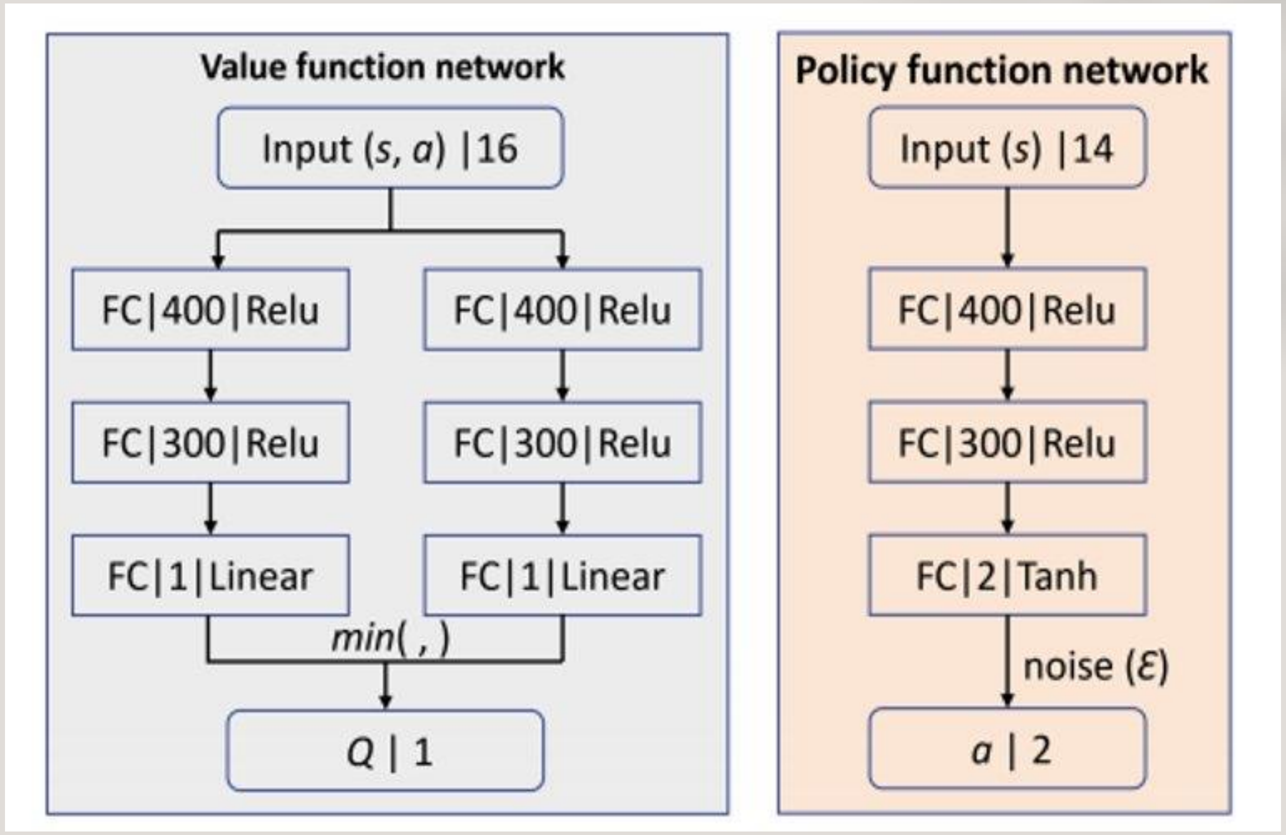
Algorithm



- TD3 Algorithms
  - Description:
    - Twin Delayed Deep Deterministic policy gradient -TD3
    - TD3 is the successor to the Deep Deterministic Policy Gradient (DDPG)(Lillicrap et al, 2016). Up until recently, DDPG was one of the most used algorithms for continuous control problems such as robotics and autonomous driving.
    - Although DDPG is capable of providing excellent results, it has its drawbacks. Like many RL algorithms training DDPG can be unstable and heavily reliant on finding the correct hyper parameters for the current task (OpenAI Spinning Up, 2018).
    - This is caused by the algorithm continuously over estimating the Q values of the critic (value) network. These estimation errors build up over time and can lead to the agent falling into a local optima or experience catastrophic forgetting.
    - TD3 addresses this issue by focusing on reducing the overestimation bias seen in previous algorithms. This is done with the addition of 3 key features:
      1. Using a pair of critic networks
      2. Delayed updates of the actor
      3. Action noise regularization

Algorithm

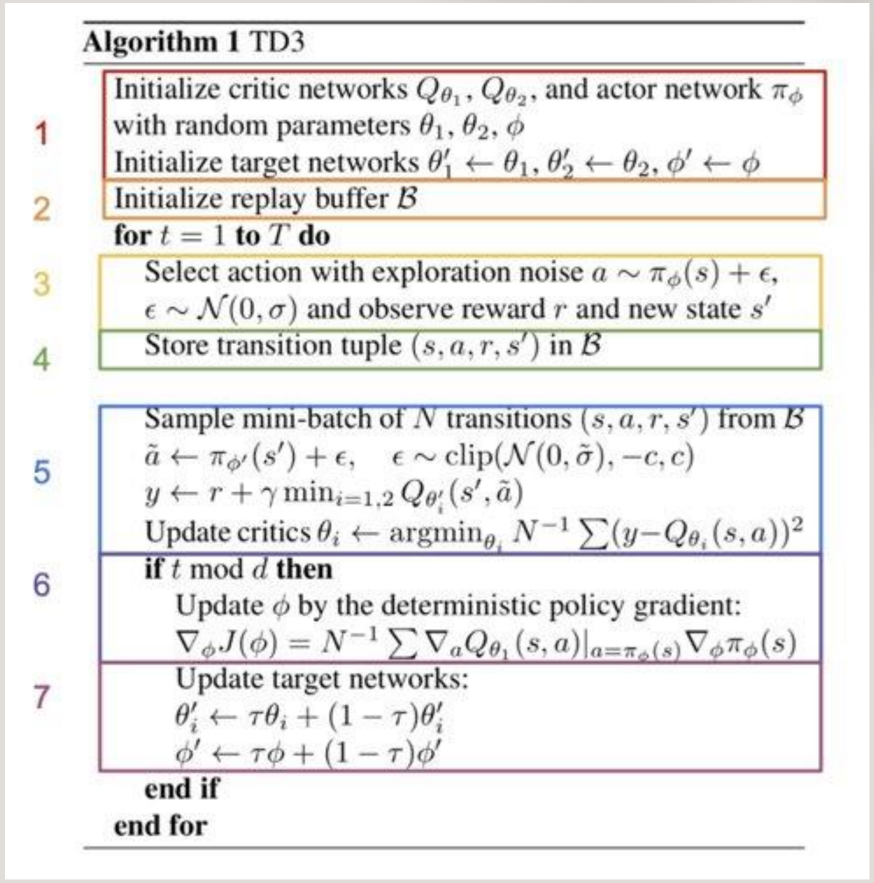
- Network Architecture [4]
  - Critic
    - Input Shape 16 --> 10
  - Actor
    - Input Shape 14 --> 8



[4] Gao, J., Ye, W., Guo, J., & Li, Z. (2020). "Deep Reinforcement Learning for Indoor Mobile Robot Path Planning". Sensors, 20(19), 5493.

Algorithm

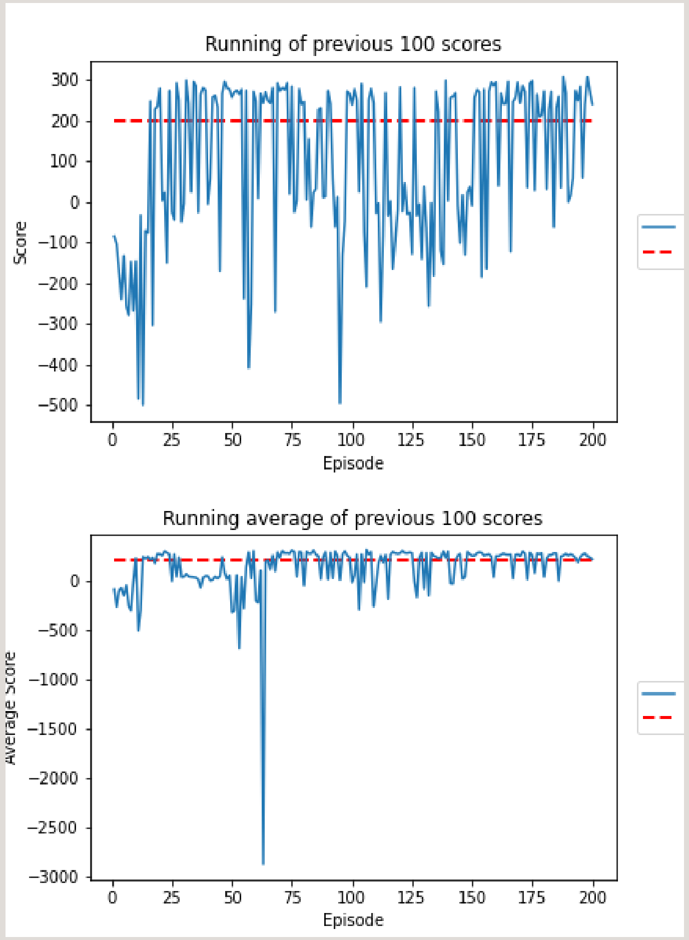
- Pseudo code [5]
  1. Initialize networks
  2. Initialize replay buffer
  3. Select and carry
     out action with exploration noise
  4. Store transitions
  5. Update critic
  6. Update actor
  7. Update target networks
  8. Repeat until sentient



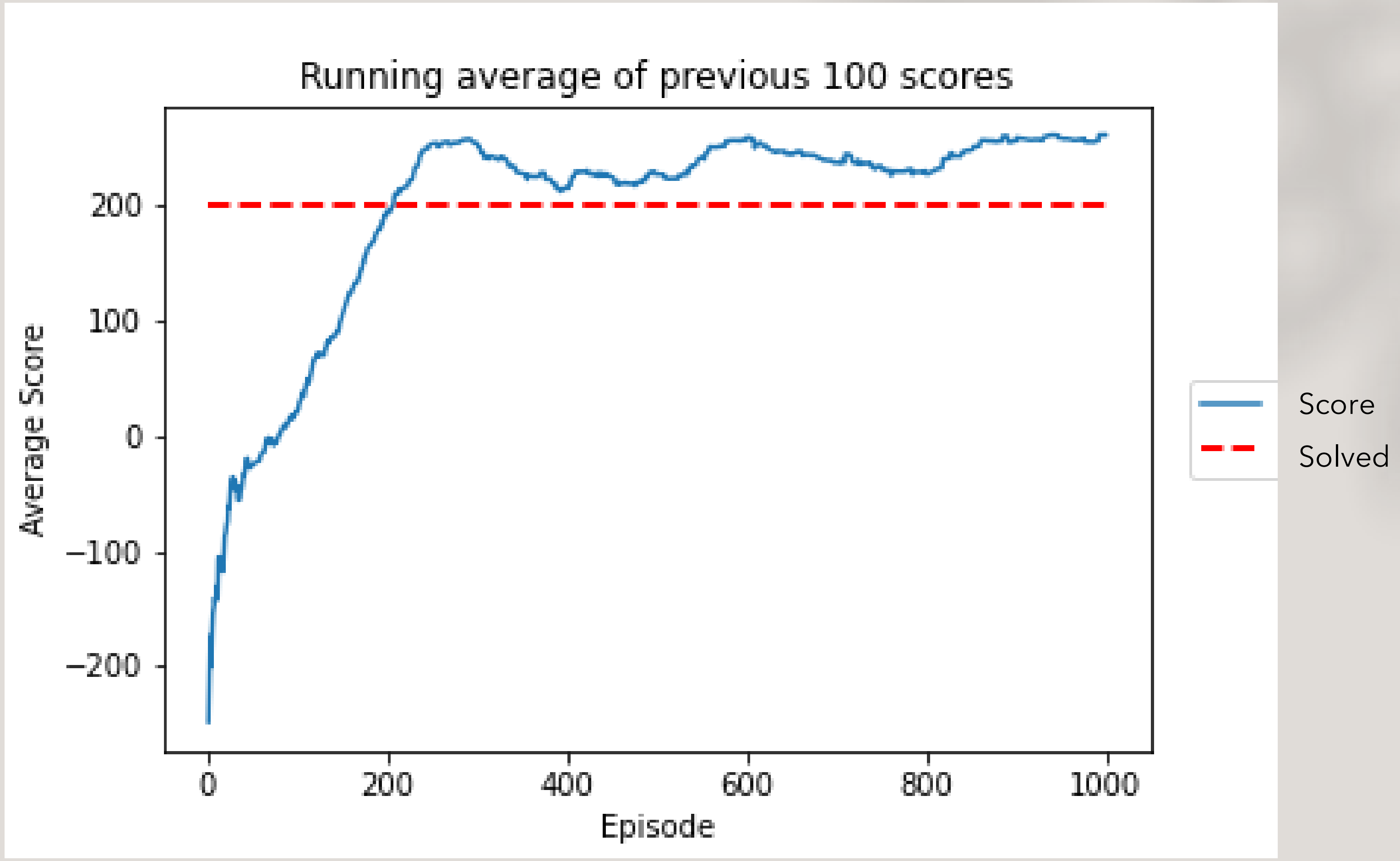
```
Algorithm 1 TD3
Initialize critic networks $Q_{\theta_1}$, $Q_{\theta_2}$, and actor network $\pi_\phi$
with random parameters $\theta_1, \theta_2, \phi$
Initialize target networks $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ to $T$ do
    Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$,
    $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward $r$ and new state $s'$
    Store transition tuple $(s, a, r, s')$ in $\mathcal{B}$

    Sample mini-batch of $N$ transitions $(s, a, r, s')$ from $\mathcal{B}$
    $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
    $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
    Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
    if $t$ mod $d$ then
        Update $\phi$ by the deterministic policy gradient:
        $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
        Update target networks:
        $\theta'_i \leftarrow \tau \theta_i + (1 - \tau)\theta'_i$
        $\phi' \leftarrow \tau \phi + (1 - \tau)\phi'$
    end if
end for
```

Results

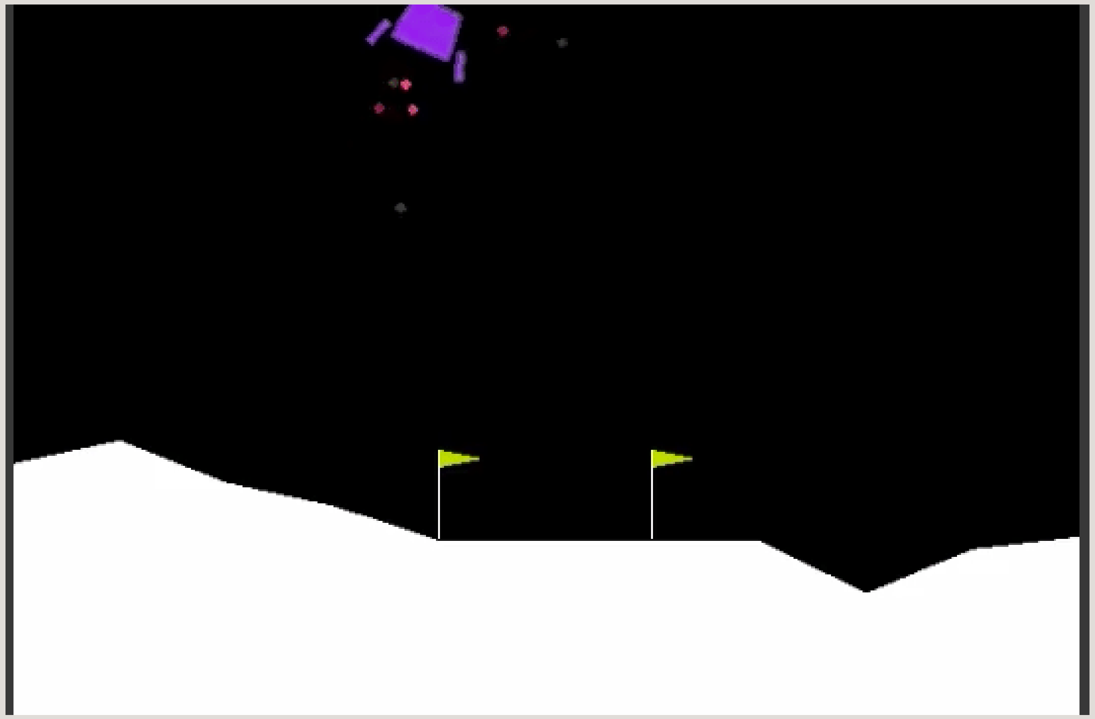


Running of previous 100 scores
Score
Episode
Running average of previous 100 scores
Average Score
Episode

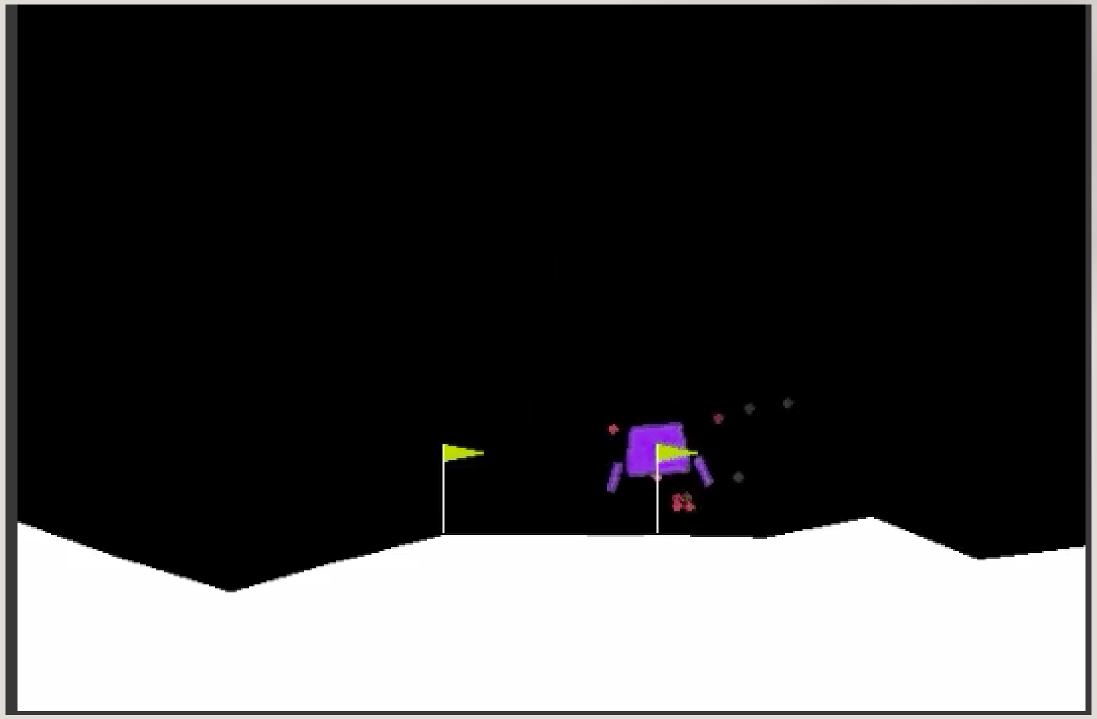


Running average of previous 100 scores
Average Score
Episode
Score
Solved

Simulation



Point: 276.74234249152335



Point: 297.34392408211403

References

- [1] Tzafestas, S. G. (2013). Introduction to mobile robot control. Elsevier
- [2] OpenAI Gym – Lunar Lander v2
- [3] Lucio Custode, Leonardo, and Giovanni Lacca. "Evolutionary learning of interpretable decision trees." arXiv e-prints (2020): arXiv-2012.
- [4] Gao, J., Ye, W., Guo, J., & Li, Z. (2020). "Deep Reinforcement Learning for Indoor Mobile Robot Path Planning". Sensors, 20(19), 5493.
- [5] Fujimoto, S., Van Hoof, H., & Meger, D. (2018). "Addressing function approximation error in actor-critic methods". arXiv preprint arXiv:1802.09477.

Happy Ending !

- MSO Lab