Performance Exploration of Reinforcement Learning Algorithms in Discrete and Continuous Action Spaces

Tang Yingwei
Sun Kuan
Tang Zihan
Zhang Boyuan
Wang Zhonghan

18 Mar 2025

Content

- Introduction and background
- Q-learning and optimization
- DQN and REINFORCE
- Hyperparameter optimization and PPO
- Conclusion, limitations and future work

Introduction and background

Lunar Lander

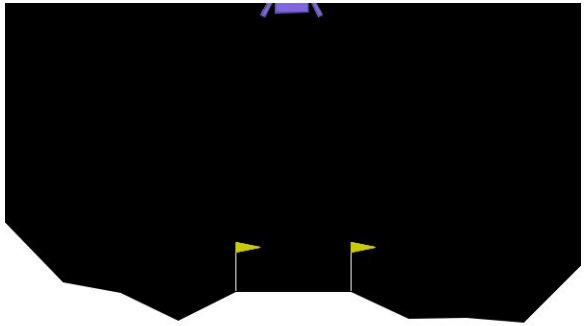
Target: Make the spacecraft land smoothly on a landing pad

State Space (8 dimension):

- Position (x, y)
- Speed (vx, vy)
- Angle (θ)
- Angular velocity (ω)
- Contact status(left/right leg)

Action Space:

- Discrete: Main engine, left/right thrust, no movement
- Continuous: Main Engine Thrust & Side Thrust

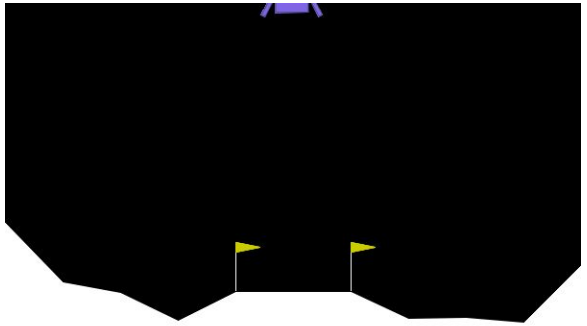


Introduction and background

Lunar Lander

Our study focuses on:

- Compare the performance of different RL algorithms
- Perform hyperparameter optimization



Q-learning and optimization

Q-learning Basics

Storing State-Action Value Functions with Q-table:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

Limitations:

- Need to discretize the state space, number of states grows too fast
- Inefficient exploration, slow convergence
- Existence of overestimation bias (due to max operation)

Q-learning and optimization

Q-learning Optimization strategy

Non-Uniform State Discretization:
Adaptive binning techniques are used to improve accuracy in critical areas (e.g., near landing sites)

Double Q-learning:
Solve the Q-learning overestimation problem by using two Q-tables to update alternately:

$$Q_A(s,a) \leftarrow r + \gamma Q_B(s', \arg\max_{a'} Q_A(s',a'))$$

Prioritized Experience Replay:
Empirical sampling weights were set based on TD errors (Temporal Difference Error):

$$P(i) = \frac{(|\delta_i| + \epsilon)^\alpha}{\sum_j (|\delta_j| + \epsilon)^\alpha}$$

DQN and REINFORCE

DQN (Deep Q-Network)

Q-approximation using neural networks instead of Q-table

Objective function:

$$L(\theta) = \mathbb{E}\left[(y_t - Q(s_t, a_t; \theta))^2\right]$$

where the target value:

$$y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-)$$

Optimization methods

- Replay Buffer
- Target Network
- Epsilon-greedy Exploration
- Learning-rate Scheduler

DQN and REINFORCE

REINFORCE (Policy Gradient)

Direct Optimization Strategy $\pi_\theta(a \mid s)$

Objective function:

$$J(\theta) = \mathbb{E}\left[\sum_t G_t \log \pi_\theta(a_t|s_t)\right]$$

Updated rules:

$$\theta \leftarrow \theta + \alpha \sum_t G_t \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Advantages:

- Suitable for high-dimensional state spaces
- Can handle continuous action spaces

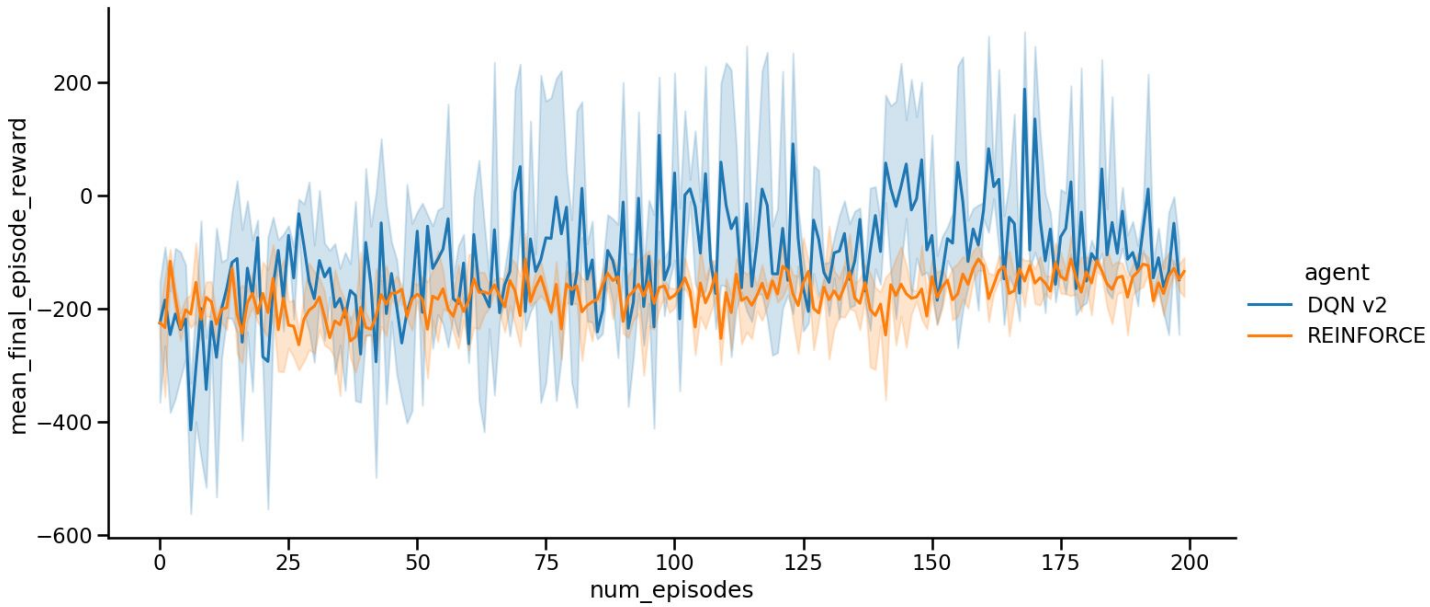
Disadvantages:

- Training is unstable and has high variance

DQN and REINFORCE

DQN vs REINFORCE:

- DQN has better performance, but REINFORCE is more flexible in policy optimization
- REINFORCE requires more complex tuning (e.g. Advantage Actor-Critic)



DQN and REINFORCE comparison

Hyperparameter optimization

Hyperparameter Optimization

Auto-Tuning with Optuna

Optimization goals:

$$\min -\text{mean}(\text{reward}) + \lambda_1 \text{std}(\text{reward}) - \lambda_2 \text{min}(\text{reward})$$

Key hyperparameters:

- Learning rate (lr)
- Exploration rate decay (ϵ-decay)
- Target network update frequency (target-update-period)

Optimization results:

- DQN optimized for faster training convergence
- REINFORCE Optimization does not improve significantly (may require more complex network)

PPO

PPO (Proximal Policy Optimization)

Suitable for continuous motion space

Objective function:

$$J_{clip}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right]$$

Features:

- Limit policy update steps by clipping
- It extends Actor-Critic architecture for improved stability

Results:

- Average reward = -366.5 during training and -398.8 during testing

AC(Actor-Critic)

Actor: Sample actions | **Critic**: Estimate states' values

Used to compute advantage and updating in two sides

Conclusions

Limitations of discrete Q - learning

Q-learning:

Metric	Training Phase	Test Phase
Episodes	40,000	5
Visited States	3,412 (0.02%)	-
Avg. Reward	-98.7 ± 12.3	-116.7 ± 9.1
Max Reward	-47.2	-110.1
Training Time	120mins (GPU)	-

TABLE I: Performance Metrics of Discrete Q-Learning Implementation

optimized Q-learning:

Implementation	Avg. Reward	Max Reward	Training Episodes
Basic Q-learning	-187.35	35.74	10,000
+ Prioritized Replay	-166.35	11.93	10,000
+ Full Optimization	-143.72	-0.83	10,000
DQN Baseline	-79.9	287.5	600

TABLE II: Comparison of Optimized Q-learning Variants

Discrete control tasks:

DQN out performs Q-learning, can be effective in solving Lunar Lander tasks.

DQN in continuous environment

DQN:

Metric	Training Phase	Test Phase	Baseline
Episodes	$3 * 200$	200	200
Visited States	3,000($< 0.02\%$)	-	-
Avg. Reward	-108.6	-79.9	-183.8
Max Reward	188.8(Avg.)	287.5	11.4
Training Time	18mins (RTX2070s)	-	-

TABLE III: Performance Metrics of Deep Q-Network

Metric	Old-Training	Old-Test	New-Training	New-Test
Episodes	$3 * 200$	200	$3 * 200$	200
Visited States	3,000	-	6500	-
Avg. Reward	-108.6	-79.9	-72.7	-104.2
Max Reward	188.8(Avg.)	287.5	123.7(Avg.)	217.6
Training Time	18mins	-	22mins	-

TABLE V: Performance Metrics of Non- / Optimized DeepQ-Network

Hyperparameter Optimization:

Optuna was used to tune hyperparameter searches for both DQN and REINFORCE

REINFORCE AND PPO

REINFORCE:

Metric	Training Phase	Test Phase	Baseline
Episodes	$3 * 200$	200	200
Avg. Reward	-175.2	-136.1	-183.8
Training Time	9.5mins (RTX2070s)	-	-

TABLE IV: Performance Metrics of REINFORCE Algorithm

PPO:

Metric	Training	Test
Episodes	$3 * 200$	200
Avg. Reward	-366.5	-398.8
Training Time	5.5mins	-

TABLE VII: Performance Metrics of PPO Algorithm

Policy Gradient Methods:
REINFORCE is stable in training and performs well

Future work

Explore more advanced methods of intensive learning:

- Try DQN s variants
- Better hyperparameters And larger training episodes
- Naive PPO performs poorly
- Algorithms for continuous control

Improved environmental design and training methods:

- Tune the **reward** function
- Introduction of more complex environments

Thanks For Listening

Tang Yingwei
Sun Kuan
Tang Zihan
Zhang Boyuan
Wang Zhonghan