Reinforcement Learning HW 9

Mert Bilgin (7034879)
mert.bilgin@student.uni-tuebingen.de

Lalitha Sivakumar (6300674)
lalitha.sivakumar@student.uni-tuebingen.de

Kevin Van Le (7314700)
kevin-van.le@student.uni-tuebingen.de

December 17, 2025

1 Deep Deterministic Policy Gradient (DDPG)

(a)

The intuition is that the critic should evaluate the current policy of the actor. Therefore, the target value is computed using the actor's target network to estimate the action that would be taken in the next state. Using target networks makes this estimate change slowly, which stabilizes the TD update and prevents divergence.

(b)

In Deterministic Policy Gradient (DPG), target networks are updated using a *soft update* mechanism, where the target network parameters are slowly moved towards the online network parameters:

$$\theta_{\text{target}} \leftarrow \tau\theta + (1-\tau)\theta_{\text{target}}, \quad \tau \ll 1.$$

This gradual update ensures that the target values change smoothly over time. Compared to hard updates that copy parameters abruptly, soft updates reduce sudden shifts in the targets making them more stable and help prevent oscillations/divergence during learning.

(c)

During training, noise is added to the actor's actions so that the policy doesn't act in a purely deterministic way. This controlled randomness allows the agent to try out new actions and explore the continuous action space rather than repeatedly exploiting the same behavior. As learning progresses and the noise is reduced, the policy naturally shifts toward exploiting the learned actions, balancing exploration and exploitation and improving overall performance.

(d)

In DDPG, overestimation happens because the critic learns from its own predictions. When the target value is computed, the critic uses a maximization step over noisy value estimates, so random errors tend to push values upwards rather than cancel out. With function approximation, these

small errors accumulate, causing the critic to systematically overestimate how good actions are, which can mislead the actor during learning.

2 DDPG- Hands On

(a)

The updated code is in `DDPG.py`.

(b)

The plots for this task and the subsequent ones can be found in `Gym-DDPG-plots.ipynb`.

The episode rewards improve steadily from about -1500, with raw rewards fluctuating around the -500 range after roughly 1500 episodes. The running mean continues to improve and approaches -250 later in the training, indicating stable learning.

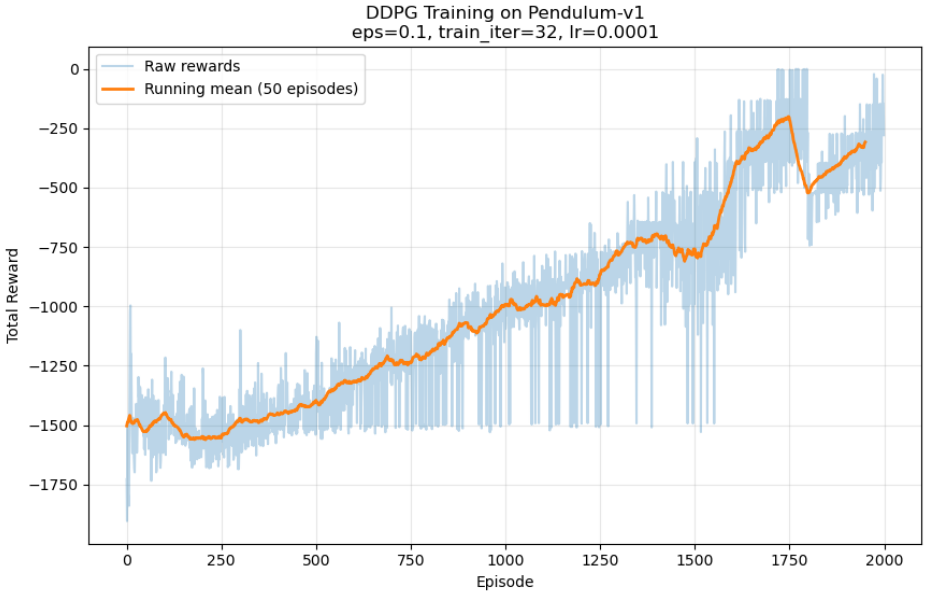

Figure 1: Pendulum-v1 Experiment

(c)

We collected 20,000 state–action pairs by running 100 episodes with exploration noise strength 0.2. The mean episode reward was -357.49 ± 160.73 so we can say that the policy is pretty stable under this noise level.

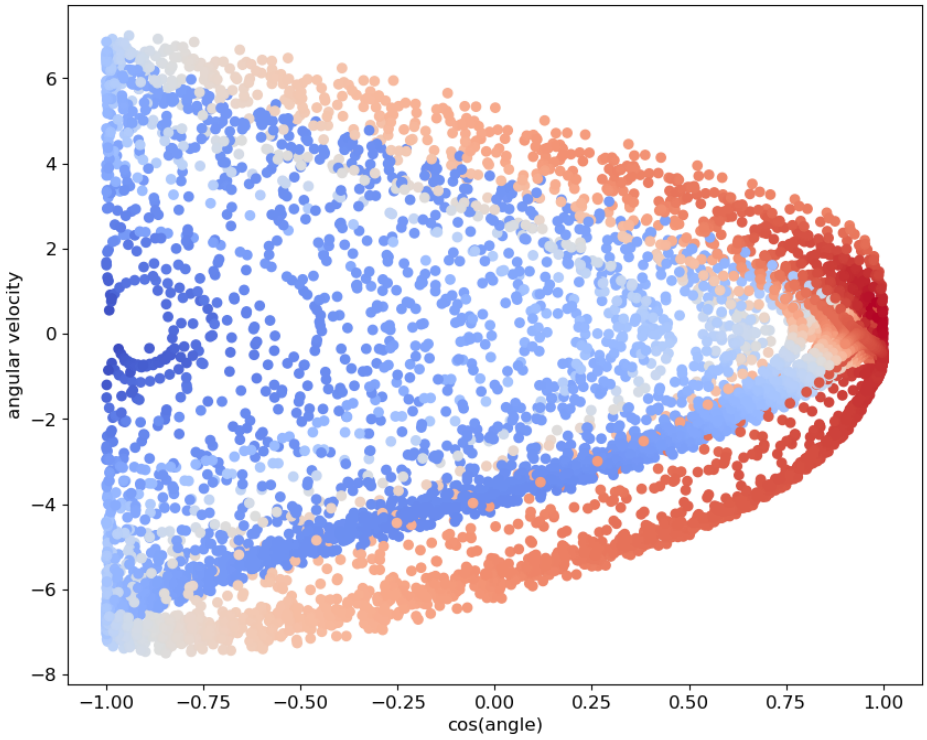

Figure 2: Q-function plot

(d)

For update frequency 20, a learning rate of 0.0001 converges fastest. By fixing this learning rate, we see that updating the target networks every 20 episodes leads to faster learning than updating every 100 episodes.

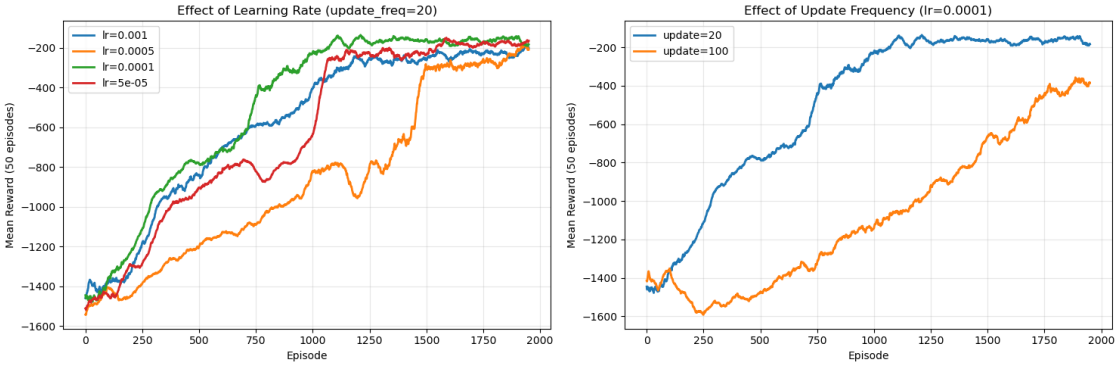

Figure 3: Experiments with varying learning rate and update frequency

(e)

Results can be found in the notebook.

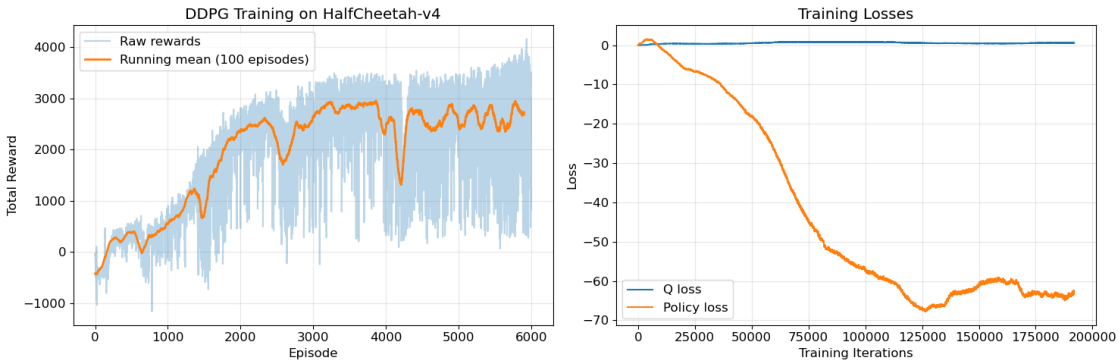

Figure 4: HalfCheetah-v4 Experiment

(f)

Yes, the half cheetah runs successfully. However, sometimes it falls and cannot recover the standing position. We got mean reward 3773.03 with maximum 4177.15 reward.