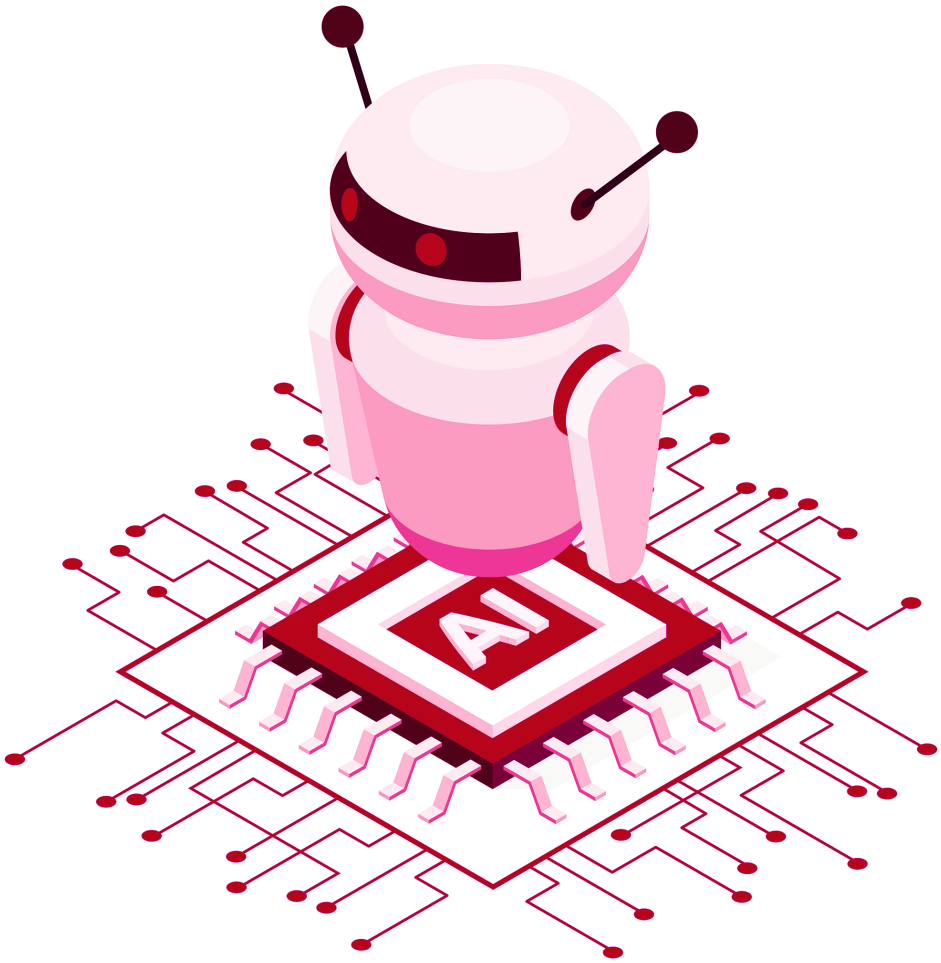

Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 2:

Value-Based Methods

By:

Ali Ghasemzadeh

401106339

RIML

Deep RL Course

Spring 2025

Contents

Grading

The grading will be based on the following criteria, with a total of 100 points:

Task	Points
Task 1: Epsilon Greedy & N-step Sarsa/Q-learning	40
Jupyter Notebook	25
Analysis and Deduction	15
Task 2: DQN vs. DDQN	50
Jupyter Notebook	30
Analysis and Deduction	20
Clarity and Quality of Code	5
Clarity and Quality of Report	5
Bonus 1: Writing your report in Latex	10

Notes:

- Include well-commented code and relevant plots in your notebook.
- Clearly present all comparisons and analyses in your report.
- Ensure reproducibility by specifying all dependencies and configurations.

1 Epsilon Greedy

1.1 Epsilon 0.1 initially has a high regret rate but decreases quickly. Why is that? [2.5-points]

At first the agent didn't have a good policy and should explore to learn but it explores with rate 0.1 and exploit mostly so at first it has a high regret but after some time it learns to play good and don't need to explore and it mostly act with its good policy and so we the regret rate decreases.

1.2 Both epsilon 0.1 and 0.5 show jumps. What is the reason for this? [2.5-points]

Jumps occur due to policy shifts or environmental stochasticity. exploration events can lead to sudden performance drops or improvments. so for $\epsilon = 0.5$ we have more jumps because making random choice more happens than $\epsilon = 0.1$ and so policy updates volatile.

1.3 Epsilon 0.9 changes linearly. Why? [2.5-points]

As you know with this ϵ we 90% of time choose the random action and it means we just 10% of the time use the policy that we learn so we have a slow convergence and since learning progresses in small increments happens then the performance curve is smooth and linear.

1.4 Compare the policy for epsilon values 0.1 and 0.9. How do they differ, and why do they look different? [2.5-points]

As it can be seen from the plots we understand that if we have $\epsilon = 0.9$ then we do random actions mostly so the arrows near the bad areas is up to minimize the risk of falling but with $\epsilon = 0.1$ the arrows are right because most of the time it goes respect to our policy.

1.5 In the epsilon decay section, analyze the optimal policy for the row adjacent to the cliff (the lowest row). Then, compare the different learned policies and their corresponding rewards. [2.5-points]

Fast Decay: Learns quickly but might play it too safe, avoiding the cliff aggressively. It reaches decent rewards fast but could settle for a suboptimal strategy.
Medium Decay: Strikes a balance—explores enough to find a smart path while still learning efficiently. Likely the best trade-off.
Slow Decay: Takes longer to learn but explores more thoroughly, potentially finding the absolute best strategy in the long run.

2 N-step Sarsa and N-step Q-learning

2.1 What is the difference between Q-learning and sarsa? [2.5-points]

Q-learning Uses max(Q(s', a')) for update(it is off-policy), it is also more greedy and favors exploration, it may converge faster but can be unstable.
Sarsa uses actual next action Q(s', a')(it is on-polic0y), it is more conservative and follows ϵ-greedy policy and it is more stable but slower convergence.

2.2 Compare how different values of n affect each algorithm's performance separately. [2.5-points]

small n make the updates fast but can be short-sighted and it is good for dynamic environments.
large n is more accurate value estimates but slower updates. it works in deterministic environments.
Sarsa
low n causes stable but slow learning.
high n causes better long-term planning but sensitive to randomness.
Q-learning
low n causes fast updates and good for changing environments.
high n causes better long-term planning and more accurate updates.

2.3 Is a Higher or Lower n Always Better? Explain the advantages and disadvantages of both low and high n values. [2.5-points]

no there is a trade-off and for unstable environments, low n is better, for stable, deterministic environments, high n can be better.

3 DQN vs. DDQN

3.1 Which algorithm performs better and why? [3-points]

respect to the plots DQN works better becuase it has more moving average.

3.2 Which algorithm has a tighter upper and lower bound for rewards. [2-points]

DDQN has a tighter bound because it mitigates overestimation, reducing fluctuations in rewards. DQN, on the other hand, often exhibits higher variance due to overestimated Q-values.

3.3 Based on your previous answer, can we conclude that this algorithm exhibits greater stability in learning? Explain your reasoning. [2-points]

Yes, A tighter bound means that DDQN's learning process is more stable and less prone to extrem fluctuations, Since the overestimation of DQN it leads to inconsistent updates and DDQN should provide more smoother and reliable convergence.

3.4 What are the general issues with DQN? [2-points]

it tends to overestimates Q-value leading to unstable learning.
it updates Q-values aggressively and sometimes forgetting past experiences.
it is also alow if we use a large replay buffer.
high variance in Q-value estimates can lead to unstable training.

3.5 How can some of these issues be mitigated? (You may refer to external sources such as research papers and blog posts be sure to cite them properly.) [3-points]

DDQN reduces overestimation by decoupling action selection from Q-value updatesVan Hasselt et al., 2016).
Prioritized Experience Replay: Samples more important transitions more frequently (Schaul et al., 2015).
Dueling DQN: Separates state-value and advantage functions for better value estimation (Wang et al., 2016).
Target Networks: Keeps a delayed copy of Q-values to stabilize learning. Reward Clipping: Helps prevent large, unstable updates in environments with high reward variance.

3.6 Based on the plotted values in the notebook, can the main purpose of DDQN be observed in the results? [2-points]

No it can't be seen in the results the DQN works better than DDQN in the same number of epochs.

3.7 The DDQN paper states that different environments influence the algorithm in various ways. Explain these characteristics (e.g., complexity, dynamics of the environment) and their impact on DDQN's performance. Then, compare them to the CartPole environment. Does CartPole exhibit these characteristics or not? [4-points]

Complexity: In highly complex environments, DDQN helps prevent misleading Q-values, leading to more stable learning.
Dynamics: If the environment changes frequently, DDQN adapts better than DQN by making more accurate updates.
I think that the number of episodes are not enough to show that the DDQN is working better or maybe the hyperparameters can make the convergence better but as can be seen at the end of the plot the moving average of the DDQN is near moving average of DQN and so I think in the future episodes it will act better.

3.8 How do you think DQN can be further improved? (This question is for your own analysis, but you may refer to external sources such as research papers and blog posts be sure to cite them properly.) [2-points]

Distributional RL: Instead of learning a single Q-value, it learns a distribution over Q-values (Bellemare et al., 2017).
Noisy Networks: Uses stochastic noise in network weights for better exploration (Fortunato et al., 2018).
Meta-learning Techniques: Helps the agent adapt faster to new environments.
Multi-step Learning: Uses multi-step returns to improve credit assignment (Hessel et al., 2018).
another idea that is from my own is using local search way and between the some best actions we choose one of them.

References

[1] R. Sutton and A. Barto, Reinforcement Learning: An Introduction, 2nd Edition, 2020. Available: http://incompleteideas.net/book/the-book-2nd.html.

[2] Gymnasium Documentation. Available: https://gymnasium.farama.org/

[3] Grokking Deep Reinforcement Learning. Available: https://www.manning.com/books/grokking-deep-reinforcement-learning

[4] Deep Reinforcement Learning with Double Q-learning. Available: https://arxiv.org/abs/1509.06461

[5] Cover image designed by freepik