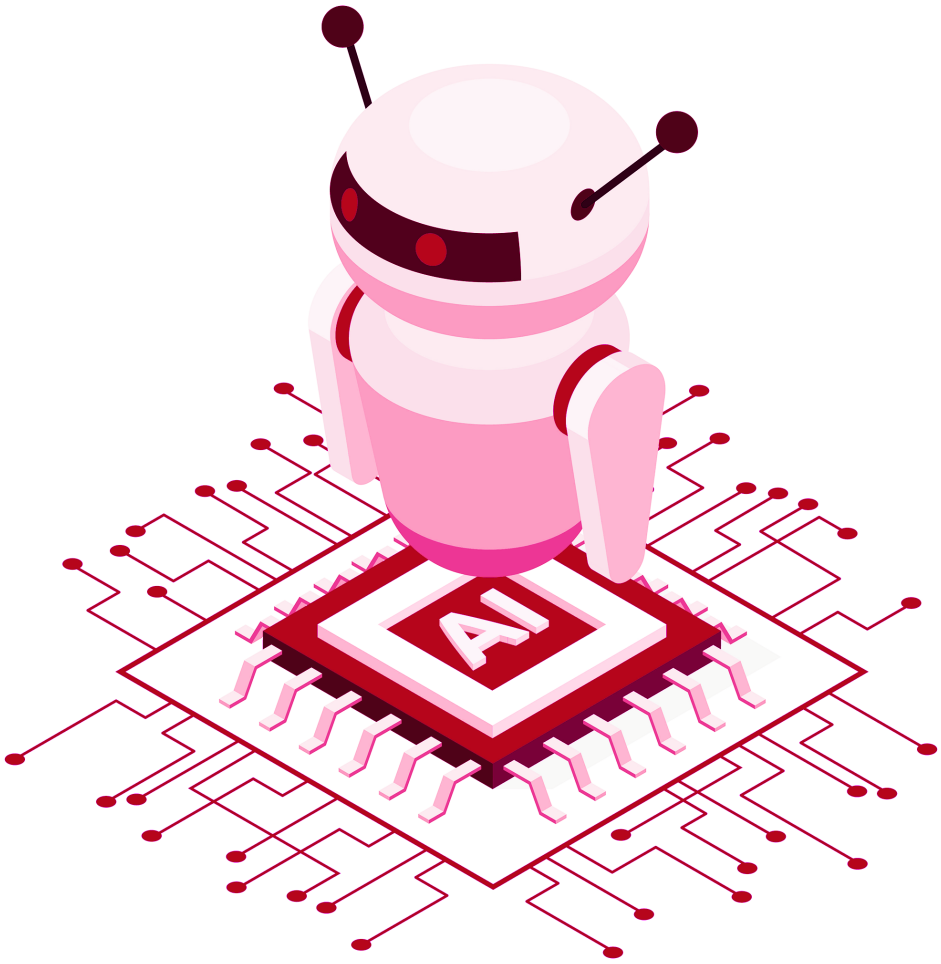

Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 2:

Introduction to RL

By:

Parsa Ghezelbash

401110437

RIML

Spring 2025

Contents

1 Epsilon Greedy N-step Sarsa/Q-learning

Question:

Epsilon 0.1 initially has a high regret rate but decreases quickly. Why is that?

Answer:

Epsilon (ε) = 0.1 means the agent explores 10% of the time and exploits 90% of the time. Initially, the Q-values are uninformative, so the agents policy is essentially random, leading to high regret since it frequently falls off the cliff or takes suboptimal actions.

Question:

Both epsilon 0.1 and 0.5 show jumps. What is the reason for this?

Answer:

The jumps in regret are caused by exploratory moves that lead to significantly worse outcomes, especially when the agent falls into the cliff. The higher the epsilon, the more frequently these jumps occur because the agent explores more often and deviates from the optimal path.

Question:

Epsilon 0.9 almost changes linearly. Why?

Ansewr: Epsilon (ε) = 0.9 means the agent chooses a random action 90% of the time and follows the greedy policy only 10% of the time. This high level of exploration results in near random behavior Since the agent is mostly choosing random actions, it does not follow a structured or optimized policy. Instead, it frequently steps into suboptimal states, including falling into the cliff, leading to a steady accumulation of regret over time.

Question:

Compare the optimal policy for epsilon values 0.1 and 0.9. How do they differ, and why do they look different?

Answer:

In the $\varepsilon = 0.1$ the optimal policy is to go all the way up and after reaching the top rightmost point going down to reach G but with the $\varepsilon = 0.9$ the optimal policy goes through mid level cells to reach the goal and this is due to high exploration this agent had through mid cells of the grid.

Question:

n the epsilon decay section, analyze the optimal policy for the row adjacent to the cliff(lowest row). Then, compare the different learned policies and their corresponding rewards.

Answer:

In my experiment all three epsilon decay agents didn't converge to a correct policy.

Question:

What is the difference between Q-learning and sarsa?

Answer:

In sarsa we choose the next state action based on our current policy but in Q-learning we just take the best action based on the Q value learned and doesn't use the agents policy.

Question: Compare how different values of n affect each algorithms performance separately.

Answer:

In my observation the rewards earned by the agent is more stable and smoother in low n for sarsa and it becomes noisy. But for Q-learning increasing n doesn't destroy or change the reward plot as much it does for sarsa.

2 DQN vs. DDQN

It should be said that I used target net to create target Q values in my DQN algorithm too.

Question:

Which algorithm performs better and why?

Answer:

In my experience both models did reach the 500 goal but DDQN performed better in number of successful agents. That is because the targets generated for the deep model are decided by the policy net which is some steps ahead of the target net, this way the target values become better estimation of the real ones.

Question:

Which algorithm has a tighter upper and lower bound for rewards.

Answer:

It seems like DDQN has tighter upper and lower bound in it's rewards.

Question:

Based on your previous answer, can we conclude that this algorithm exhibits greater stability in learning? Explain your reasoning.

Answer:

Yes. DDQN is more stable in learning and DQN exhibits more fluctuation in rewards.

Question:

What are the general issues with DQN?

Answer:

DQN uses the same network to select and evaluate the next action. This can lead to overestimation of Q-values, causing instability and suboptimal policies.

If the Q-network incorrectly assigns a high Q-value to a bad action, DQN may prioritize this action incorrectly. This leads to an unstable and less reliable learning process.

Question:

How can some of these issues be solved? (You may refer to external sources such as research papers and blog posts be sure to cite them properly.)

Answer:

The standard ε-greedy exploration strategy may not effectively balance exploration and exploitation, especially in complex environments.

Solution: Integrate Noisy Networks, which add parameterized noise to the network weights, enabling more efficient exploration by allowing the agent to learn which actions to explore.

Question:

Based on the plotted values in the notebook, can the main purpose of DDQN be observed in the results?

Answer:

Yes the results of DDQN were quite smoother than DQN.

Question:

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 DDQNs performance. Then, compare them to the CartPole environment. Does CartPole exhibit these characteristics or not?

Answer:

In complex environments with large state-action spaces and numerous obstacles, DDQNs may require more training data and time to learn effective policies.

In highly dynamic environments where conditions change rapidly and unpredictably, DDQNs may struggle to adapt, leading to suboptimal performance.

In CartPole The state space is relatively small, typically comprising four variables: cart position, cart velocity, pole angle, and pole angular velocity. The action space is also limited to two actions: applying a force to the left or right. and the environment's dynamics are predictable and do not change over time, providing a stable platform for learning.

The absence of dynamic changes and obstacles means that once the DDQN learns an effective policy, it remains applicable without the need for frequent adjustments.

Question:

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 postsbe sure to cite them properly.)

Answer:

This approach integrates multiple improvementssuch as DDQN, Dueling DQN, Prioritized Experience Replay, and Noisy Networksinto a single architecture. By combining these methods, Rainbow DQN achieves superior performance compared to using any single enhancement alone.

References

[1] M. Fortunato, M. Gheshlaghi Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, "Noisy Networks for Exploration," 2017. Available online: https://arxiv.org/abs/1706.10295

[2] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining Improvements in Deep Reinforcement Learning," 2017. Available online: https://arxiv.org/abs/1710.02298