Deep Reinforcement Learning (Sp25)

Instructor: Dr. Mohammad Hossein Rohban

Quiz Solutions - Lecture 5

Solution by : Benyamin Naderi

Q1

Chek the drawbacks of Q-learning algorithm

A)Non-stationary environment
B)Noisy gradient
C)Non-iid samples in trajectory
D)Correlation between mini batches

Correct Answers: C

Explanation: samples come from sequential interactions with the environment so samples in a trajectory are still dependent, in Q-learning method we don't use any neural network and there is no need to gradients since it's a tabular method and Q values are updated using bellman equation so the second item is wrong and doesn't necessarily relate to Q learning its actually one of the DQN method drawbacks. by the same instruction it's not a learning method and there is no minibatch involved during the Q values update mechanism.

Q2

An engineer wants to design an arm for a robot. The robot must find the best permutation of at most 8 consecutive actions to unlock something, seeking a sparse reward and ever since it wants to decide it has only 32 possible actions. Which algorithm should they use for the design? And please state any well known algorithm that fits this situation. What are the drawbacks of your algorithm and support it by comparison. Hint: in terms of value based/gradient method/off policy/on policy

Answer: The best choice for this scenario is Deep Q-Networks (DQN) with enhancements like Prioritized Experience Replay or Double DQN since, The problem has a fixed, relatively small action space (32 actions), making Q-learning a good fit. The agent can explore different action sequences and still use past experiences to improve learning through experience replay buffer. Techniques like reward shaping or PER can improve learning efficiency in environments with sparse rewards. please note that action space in this problem is discrete and policy gradient methods are not suitable for this environments. this method is a **Value-based** and **Off-policy** method since it learns a Q-value function and guarantees policy improvement. it updates its Q-values using experiences collected from past policies, rather than requiring fresh on-policy data for every update.

Q3

Compare value-based methods and policy-based methods?

Value-based methods (e.g., DQN, Q-learning) learn a Q-value function to estimate the expected reward for each action and select actions greedily, making them well-suited for discrete action spaces and off-policy learning, which improves sample efficiency. However, they struggle with continuous actions and high-dimensional problems. In contrast, policy-based methods (e.g., PPO, REINFORCE) directly optimize the policy without needing Q-values, making them better for continuous action spaces and stochastic policies but less sample efficient due to on-policy updates and high variance in training. While value-based methods work well for structured decision-making, policy-based approaches are suitable when the problem is highly dynamic and requires smooth updates.

Q4

In solving an Atari game an agent is trained to reach the highest reward. Our algorithm must decide as fast as possible, but we have memory issues. Recommend a solution by making any changes in the Q-network or replay buffer. Can you make any changes in the architecture? What effects will happen by changing the buffer size?

Answer: To address memory issues, you can reduce the replay buffer size or use Prioritized Experience Replay (PER) to focus on more informative experiences. Reducing the buffer size lowers memory usage but may impact learning stability and diversity, while PER can improve efficiency by prioritizing more important experiences.

Q5

Consider the following scenarios:
i) An agent is learning to navigate a stochastic environment where the transition dynamics are highly uncertain.
ii) An agent is trained using a neural network to approximate the Q-function, and experience replay is employed to stabilize training.
iii) An agent updates its Q-values using the action actually taken during exploration, rather than the optimal action. Which of the following statements is true?

A) Scenario 1 favors SARSA over DQN because SARSA's on-policy nature accounts for environmental uncertainty more effectively.
B) Scenario 2 describes SARSA, as it inherently uses neural networks and experience replay.
C) Scenario 3 describes DQN, as it updates Q-values based on the optimal action rather than the exploratory action.

Correct Answers: A) Scenario 1 favors SARSA over DQN because SARSA's on-policy nature accounts for environmental uncertainty more effectively.

- **Statement A is true**: SARSA is an on-policy algorithm, meaning it updates Q-values based on the agent's actual exploratory actions rather than assuming optimal actions. This makes SARSA more suitable for highly stochastic environments where DQN's off-policy nature might lead to over-optimistic value estimates.
- **Statement B is false**: While Scenario 2 describes the use of neural networks and experience replay, these are key features of DQN, not SARSA. SARSA traditionally does not use experience replay or neural networks.
- **Statement C is false**: Scenario 3 actually describes SARSA, as SARSA updates Q-values based on the action actually taken during exploration. In contrast, DQN updates Q-values based on the maximum Q-value (optimal action) from the next state, which makes it an off-policy algorithm.

Q6

Compare SARSA and DQN in terms of sample efficiency? Support your answers with a proof.

1 SARSA (On-Policy TD Control)

SARSA updates Q-values using:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma Q(s',a') - Q(s,a) \right] \tag{1}$$

where the update is based on the **actual** action taken (a'), making it **on-policy**.

- SARSA learns using the same policy it follows, leading to more stable learning in stochastic environments.
- However, since it follows the ϵ-greedy policy, it may explore suboptimal actions longer, making it **slower to converge** in deterministic settings.

2 DQN (Deep Q-Network, Off-Policy)

DQN updates Q-values using:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a';\theta^-) - Q(s,a) \right] \tag{2}$$

where it selects the **max-Q** action from the target network (θ^-), making it **off-policy**.

- Experience replay allows DQN to **reuse past transitions**, improving sample efficiency.
- However, since DQN does not follow the same policy it learns from, it may take longer to adapt in highly stochastic environments.

Q7

Policy gradient methods, such as REINFORCE are inherently on-policy because they optimize the policy directly using trajectories generated by the current policy. This is the same as Q-learning and SRASA, which can learn from trajectories generated by any policy. Is this statement true or false?

The statement is false because Q-learning is an off-policy method that updates Q-values using the maximum action value, regardless of the behavior policy, while SARSA is on-policy, updating based on the action actually taken. Policy gradient methods like REINFORCE are also on-policy, as they optimize the policy using only trajectories generated by the current policy.