National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Faculty of Computing

CS 368: Reinforcement Learning

Lab 09: Model Free Reinforcement Learning - Control

(Q-Learning and Double Q-Learning)

Date: 22 November 2024

Time: 02:00 PM – 5:00 PM

Instructor: Dr. Zuhair Zafar

Lab 09: Model Free Control (Q-Learning & Double Q-Learning)

Objectives

Given a simulated environment, find the optimal policy using Q-Learning & Double Q-Learning Algorithm.

Tools/Software Requirement:

Google Colab, Python, Gymnasium Library

Introduction

Tabular Q-learning is a fundamental method in **reinforcement learning** (RL) that is used to train an agent to make decisions in an environment. It is based on estimating the optimal action-value function $Q^*(s, a)$, which represents the maximum expected cumulative reward achievable from a given state *s* and action *a*, following an optimal policy.

Q-Learning Algorithm

1. **Q-table**:
   In tabular Q-learning, a table (or matrix) is maintained where:
   - Rows correspond to states *s* in the environment.
   - Columns correspond to possible actions *a* the agent can take in each state.
   - The entries in the table represent the $Q(s, a)$ value, which is the agent's estimate of the expected cumulative reward for taking action *a* in state *s*.
2. **Bellman Equation**:
   Tabular Q-learning updates the Q-table using the Bellman equation:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a' \in A} Q(s', a') - Q(s, a))$$

   - α: Learning rate, controls how much new information updates the Q-value.
   - r: Immediate reward received after taking action aaa.
   - γ: Discount factor, determines the importance of future rewards.
   - $\max_{a' \in A} Q(s', a')$: The maximum Q-value for the next state s', assuming the agent acts optimally.
3. **Exploration-Exploitation Trade-off**:
   The agent balances:
   - **Exploration**: Trying new actions to discover their rewards.
   - **Exploitation**: Choosing actions with the highest known $Q(s, a)$ to maximize rewards.

Common strategies include ϵ −greedy, where the agent chooses a random action with probability ϵ and the best-known action otherwise.

4. **Iteration Process**:
   - The agent interacts with the environment repeatedly.
   - Observes the current state *s*, takes an action *a*, and receives a reward *r*.
   - Moves to the next state s' and updates the $Q(s,a)$ value in the Q-table using the update rule.
5. **Convergence**:
   If every state-action pair is visited infinitely often and α is appropriately reduced over time, tabular Q-learning is guaranteed to converge to the optimal $Q^*(s,a)$.

Q learning Algorithm Pseudocode:

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal,\cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

Double Q-Learning Algorithm

Double Q-Learning is an improvement over standard **Q-Learning** that addresses the issue of **overestimation bias** in the estimation of Q-values. This bias occurs because, in standard Q-Learning, the same Q-function is used both to select the best action (via $\max_{a' \in A} Q(s',a')$) and to evaluate the value of that action.

Motivation

In standard Q-Learning, the target update rule for a Q-value is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a))$$

Here, the term $\max_{a' \in A} Q(s',a')$ is used both to:

1. **Select** the action $\operatorname{argmax}_{a' \in A} Q(s',a')$.
2. **Evaluate** the value of that selected action.

This dual use leads to an overestimation of the true Q-values because random noise in the estimates can result in the maximum operator amplifying the errors.

Key Idea of Double Q-Learning

Double Q-Learning addresses overestimation by maintaining **two separate Q-value functions**, Q_1 and Q_2. These two functions are updated independently and are used alternately for **action selection** and **action evaluation**.

The update rule is modified as follows:

1. **Target Update with Decoupled Selection and Evaluation**:

$$y_{DoubleQ} = r + \gamma\, Q_2(s', \underset{a' \in A}{argmax}\, Q_1(s', a'))$$

or symmetrically:

$$y_{DoubleQ} = r + \gamma\, Q_1(s', \underset{a' \in A}{argmax}\, Q_2(s', a'))$$

2. **Updating One Q-Function at a Time**:
   - With some probability (e.g., 50%), update Q_1 using the target from Q_2:
   $$Q_1(s,a) \leftarrow Q_1(s,a) + \alpha \left(y_{DoubleQ} - Q_1(s,a)\right)$$
   - Otherwise, update Q_2 using the target from Q_1:
   $$Q_2(s,a) \leftarrow Q_2(s,a) + \alpha \left(y_{DoubleQ} - Q_2(s,a)\right)$$

Advantages of Double Q-Learning:

1. **Reduces Overestimation**:
   - By decoupling action selection and evaluation, Double Q-Learning reduces overoptimistic estimates of Q-values.
2. **Improved Stability**:
   - Training dynamics are more stable compared to standard Q-Learning.
3. **Better Policies**:
   - The reduced bias often leads to improved performance in many environments.

Limitations of Double Q-Learning:

1. **Increased Computation**:
   - Maintaining and updating two Q-functions doubles the memory and computational requirements.
2. **Performance Depends on Environment**:
   - While Double Q-Learning often outperforms standard Q-Learning, its benefits are more pronounced in environments with high stochasticity or noise.

Double Q Learning Algorithm Pseudocode:

Initialize $Q_1(s,a)$ and $Q_2(s,a), \forall s \in S, a \in A$ $t = 0$, initial state $s_t = s_0$
loop
  Select a_t using ϵ-greedy $\pi(s) = \arg\max_a Q_1(s_t,a) + Q_2(s_t,a)$
  Observe (r_t, s_{t+1})
  if (with 0.5 probability True) **then**

$$Q_1(s_t,a_t) \leftarrow Q_1(s_t,a_t) + \alpha(r_t + Q_1(s_{t+1}, \arg\max_{a'} Q_2(s_{t+1},a')) - Q_1(s_t,a_t))$$

  else

$$Q_2(s_t,a_t) \leftarrow Q_2(s_t,a_t) + \alpha(r_t + Q_2(s_{t+1}, \arg\max_{a'} Q_1(s_{t+1},a')) - Q_2(s_t,a_t))$$

  end if
  $t = t + 1$
end loop

Lab Tasks

- Learning rate, alpha, α = 0.1
- Discount factor, gamma, γ = 0.9
- Exploration factor, epsilon, ε = 0.2
- Number of Episodes = 20000

1. **Solve the Cart Pole problem given in the Gymnasium library using the Q-Learning algorithm. You have to simulate the episodes using Q-Learning algorithm and see its convergence. Plot the returns for episodes. Simulate the learned policy at the end to visualize the performance.**

2. **Solve the Acrobat problem given in the Gymnasium library using the Q-Learning algorithm. You have to simulate the episodes using Q-Learning algorithm and see its convergence. Plot the returns for episodes. Simulate the learned policy at the end to visualize the performance.**

3. **Solve the Cart Pole problem given in the Gymnasium library using the Double Q-Learning algorithm. You have to simulate the episodes using Double Q-Learning and see its convergence. Plot the returns for episodes. Compare it with Q-Learning algorithm**

4. **Solve the Acrobat problem given in the Gymnasium library using the Double Q-Learning algorithm. You have to simulate the episodes using Double Q-Learning and see its convergence. Plot the returns for episodes. Compare it with Q-Learning algorithm**

Deliverable:

Please submit your notebook on LMS before the deadline **(25th November 2024, 11:59pm).**

Lab Rubrics

Assessment	Does not meet expectation (1/2 marks)	Meets expectation (3/4 marks)	Exceeds expectation (5 marks)
Software Problem Realization **(CLO1, PLO1)**	The student struggles to formulate the problem as RL and does not apply Q-Learning/DQL algorithm to solve it. There is a lack of understanding of the problem's requirements and no attempt to find the optimal policy effectively.	The student formulates the problem as RL with some guidance, applies Q-Learning and DQL algorithms with hints, and shows it's working. However, the approach might not be fully optimized or lacks a thorough justification.	The student independently formulates the given problem as RL, applies Q-Learning and DQL algorithms without guidance, and effectively finds an optimal policy. The approach is fully optimized and can be applied to different similar problems.
Software Tool Usage **(CLO4, PLO5)**	Code has syntax errors, and the implementation of the Q-Learning/DQL algorithm is incorrect or incomplete. The code is not modular and lacks comments for readability and reuse. The student shows limited ability to use gymnasium library functions where required.	The student has implemented the Q-Learning and DQL algorithms correctly for the given problem with minor mistakes. The code is mostly correct in terms of syntax and functionality but might not be optimized or well-structured for reuse. Some documentation is provided. The student also shows working knowledge of gymnasium library where required.	The student has implemented the Q-Learning and DQL algorithms efficiently and correctly. The code is clean, modular, well-documented, and follows best practices for reproducibility and reuse. The student demonstrates full command of the gymnasium library and its functions.