Robot Learning Lab 3 - Q-learning and Deep-Q-learning

Borella Simone (S317774)
Computer engineering - Artificial Intelligence
Polytechinc University of Turin, Italy
s317774@studenti.polito.it

1 Abstarct

The objective of this laboratory is to develop a solid understanding of the Q-Learning algorithm, one of the well known Reinforcement Learning (RL) algorithms. The approach involves a comprehensive implementation of the Q-Learning algorithm applied to the Cartpole-v0 environment, as offered by the gym library. The subsequent sections of this report will delve into an analysis of both the strengths and weaknesses of the Q-Learning algorithm. The outcomes of the experiments conducted in the Cartpole-v0 environment will be presented, contributing to a comprehensive evaluation of the algorithm's performance.

2 Q-learning

Q-learning is a fundamental algorithm in the literature of Reinforcement Learning, and its key characteristics include:

- **Model-Free:** Q-learning is a model-free reinforcement learning algorithm, meaning it does not require knowledge of the underlying system dynamics or transition probabilities.
- **Action Value Function:** The algorithm learns a value function, denoted as Q, which represents the expected cumulative reward of taking an action in a particular state and following a certain policy.
- **Off-Policy:** Q-learning is an off-policy algorithm, meaning it can learn from experiences generated by a different policy. This characteristic makes it versatile in utilizing data collected from different exploration strategies.
- **Bellman Equation:** Q-learning is based on the Bellman equation, which expresses the relationship between the value of a state-action pair and the values of its successor states. The update rule is given by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot \Big[(R_{t+1} + \gamma \cdot \max_{a'} Q(s_{t+1}, a')) - Q(s_t, a_t)\Big]$$

  where $Q(s_t, a_t)$ is the value of taking action a in state s, R_{t+1} is the immediate reward, α is the learning rate, γ is the discount factor, and s_{t+1} is the next state.
- **Exploration-Exploitation tradeoff:** Q-learning balances exploration and exploitation to discover optimal policies. The algorithm uses an ϵ-greedy strategy to choose between exploration (random actions) and exploitation (choosing the action with the highest Q-value).
- **Convergence:** Under certain conditions, Q-learning is guaranteed to converge to the optimal Q-values, ensuring the agent learns an optimal policy over time.

3 Tabular Q-learning

3.1 State space and Action space

In many real-world scenarios, the state and action spaces are continuous and infinite. However, tabular Q-learning requires discrete spaces for practical implementation. Discretization involves dividing continuous spaces into a finite number of bins or intervals, thereby converting them into discrete values.

In the Cartpole-v0 the action space is discrete and can be represented as $\{0, 1\}$ where 0 means *push left* and 1 means *push right*. However the state space is composed by four continuous dimensions and must be discretized. In this experience the choice falls into the discretization of the continous spaces into 16 bins.

3.2 Exploration-Exploitation tradeoff

The exploration-exploitation tradeoff is a fundamental concept in reinforcement learning, referring to the approach taken by the agent during training. By exploring the environment more thoroughly, trying new actions that may not be the current best, the agent has the potential to discover better actions or previously unknown states and transitions. On the other hand, exploitation involves relying on the current best policy to maximize immediate rewards based on the knowledge acquired so far.

The challenge lies in finding the delicate balance between exploration and exploitation to achieve optimal long-term performance. If the agent overly focuses on exploitation, it risks getting stuck in a suboptimal policy, missing opportunities to discover superior actions or states. Conversely, excessive exploration may lead to inefficient decision-making, especially when the agent already possesses a good understanding of the environment.

The GLIE (Greedy in the Limit with Infinite Exploration) theorem is a theoretical result that outlines conditions under which a reinforcement learning algorithm converges to an optimal policy. It posits that the policy should gradually become more greedy (exploitative) as the number of time steps approaches infinity. However, the exploration parameter (commonly denoted as ϵ) should approach zero only as the number of episodes increases.

Two different experiments have been done to explore the behaviour of different $\epsilon-greedy$ strategies:

- **Constant epsilon:** Use of a constant $\epsilon = 0.2$. The agent does not shift its focus more towards exploitation as it gains experience. This might prevent the agent from fully exploiting its learned policy and maximizing cumulative rewards, especially in later stages of learning when it has acquired substantial knowledge about the environment. Nevertheless, the exploitation is still present, thus a near-optimal solution is expected.
- **GLIE decreasing epsilon:**
  This strategy decrease ϵ during the training procedure, as shown in Figure 1, and is given by:
  $$\epsilon_k = \frac{b}{b+k} \tag{1}$$
  Where k is the episode number and b is a parameter set the velocity of decrease. Setting $b = 2222.\overline{2}$ ensures that $\epsilon_k = 0.1$ when $k = 20000$.

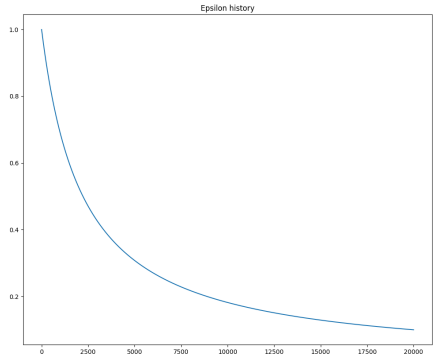

Figure 1: Epsilon function

Figure 2 and Figure 3 illustrate respectively the training procedure (episode reward in blue, moving average reward in orange) with constant-ϵ strategy and GLIE-ϵ strategy.

For these experiments, 20000 training episodes, each with a length of 500, were conducted along with 100 testing episodes, each having a length of 200.

The GLIE-ϵ strategy exhibits a decreasing exploration and increasing exploitation, contributing to the smoothness of the learning curve. Additionally, this strategy demonstrates improved performance, achieving a maximum moving average reward of 200 compared to the 196 reached by the constant-ϵ strategy as testing results (432 versus 317 during a 500-timestep episode test).

The heatmaps shown in Figure 4 and Figure 5 illustrate the value functions, in terms of the x (abscissa axis) and θ (ordinate axis) dimensions, obtained with respectively the constant-ϵ strategy and GLIE-ϵ strategy. The value function is obtained making the average over the $\dot{x}$ and $\dot{\theta}$ dimensions.

It is evident that the two policies are remarkably similar. Both policies assign high values to being in the center of the simulation with the pole upright. Additionally, they exhibit a subtle diagonal pattern, indicating a preference for positions at the edge of the simulation where the pole is slightly tilted towards the center over positions where the pole is tilted away from the center.

This behavior is a consequence of the exploration phase during the training procedure. The value function $v(s) = \max_a q(s, a)$ starts with all zeros. After each episode, it updates only the states it has visited using the next state estimate and the immediate reward. Thus, after the first episode, we can observe the actions taken by the agent during that episode. In the middle of training, the value function should be well-defined, especially in the more frequently visited states. Since there is little to no randomness in the initialization of the simulation, these frequently visited states are likely to be in the center of the simulation with the pole close to vertical (approximately $x = 0, \theta = 0$).

The Q values obtained with constant-ϵ strategy are saved as *q_values_eps_const.npy*

The Q values obtained with GLIE-ϵ strategy are saved as *q_values_eps_glie.npy*

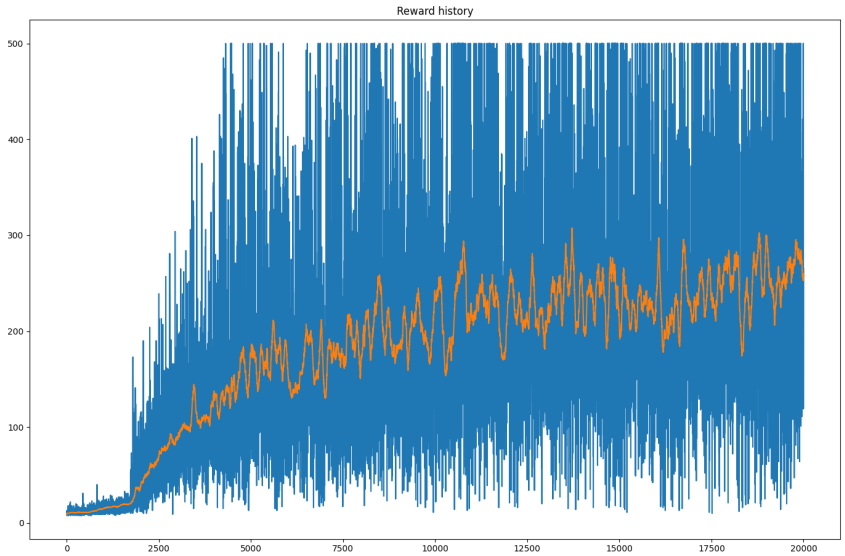

Figure 2: Training procedure with constant epsilon

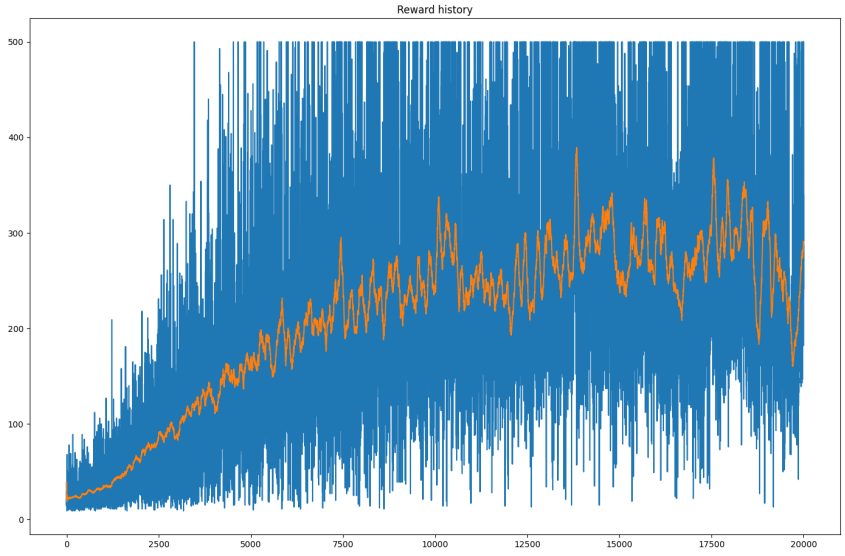

Figure 3: Training procedure with GLIE epsilon

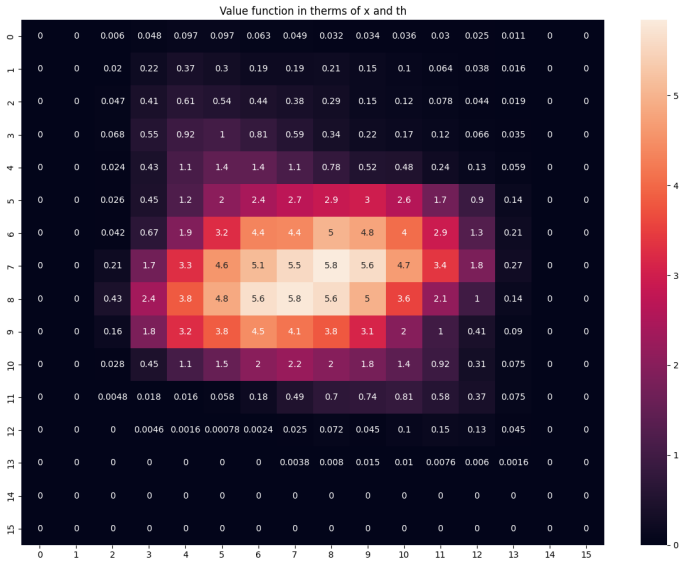

Figure 4: Value function heatmap for constant-ϵ strategy

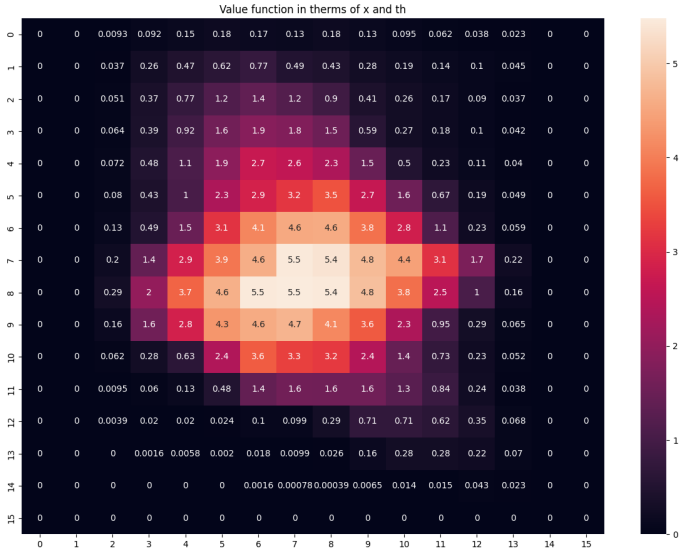

Figure 5: Value function heatmap for GLIE-ϵ strategy

3.3 Optimistic initial values

Optimistic initialization in Q-learning involves initializing the Q-values with optimistic estimates, typically set higher than the true maximum Q-values. This optimistic bias encourages the agent to explore the state-action space extensively in the initial learning stages, operating under the belief that all actions hold the potential for higher rewards. When the agent chooses an action it considers the best, the actual reward will be smaller than expected, intentionally set to be higher. Consequently, this prompts the agent to update the value of the chosen action, reducing it and creating room for exploration of other actions. Over time, as the agent interacts with the environment and updates Q-values, the initially optimistic estimates gradually converge towards more accurate values. To facilitate this convergence, a greedy policy must be adopted.

In our experiments, two different scenarios were explored: the first involved initializing values to zero (lower than the actual maximum value), and the second initialized values to fifty (higher than the actual maximum value). Both experiments were conducted with a greedy policy ($\epsilon = 0$).

Observing Figure 6 and Figure 7 some aspects could be considered. The results of testing the obtained Q values are significantly in favor of the 50-initialization with an average result over 100 episodes of 167 against 9 for the 0-initialization. While it is evident that neither policy has converged to the optimal policy, the one initialized with optimistic values performs notably better during training. Furthermore, examination of the heatmaps representing the final policies reveals that the policy initialized at fifty engaged in significantly more exploration. Although not all states deviate from the initialization, a majority exhibit values distinct from the initial estimates. In contrast, the policy initialized at zero predominantly maintains states with values at zero, indicating minimal exploration in this case.

The Q values obtained with initial value 0 are saved as *q_values_init_0.npy*

The Q values obtained with initial value 50 are saved as *q_values_init_50.npy*

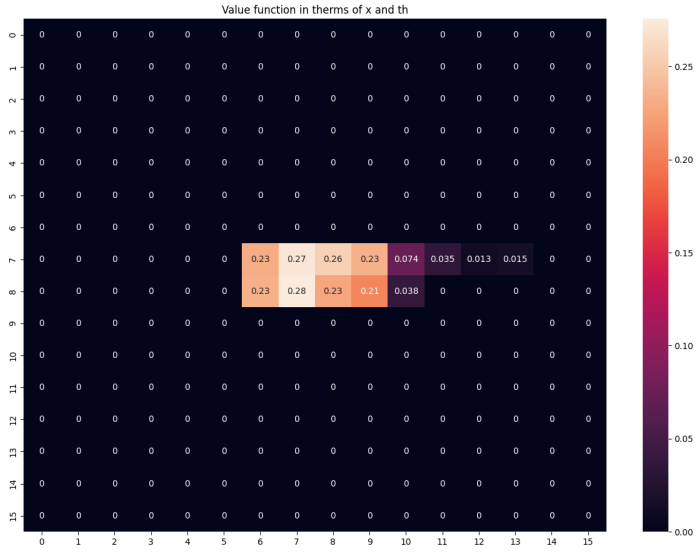

Figure 6: Value function heatmap with initial values set to 0

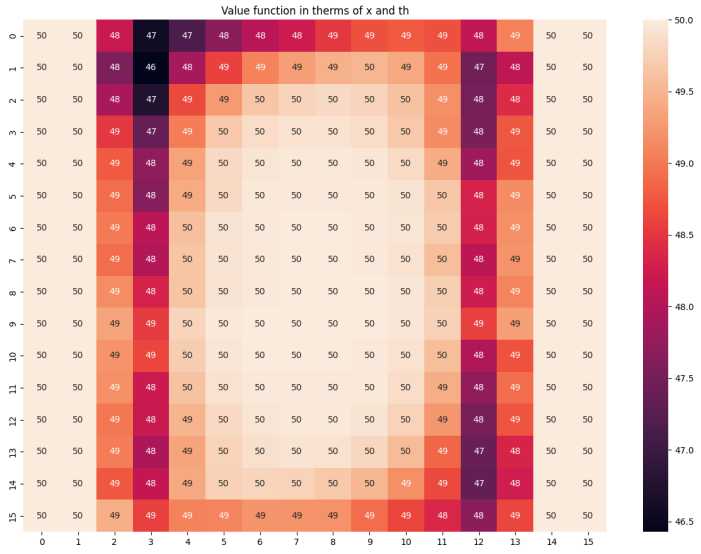

Figure 7: Value function heatmap with initial values set to 50

3.4 Q-learning in continuous state space and action space

Q-learning can be extended to address environments featuring continuous state spaces by employing function approximation techniques. In situations where the state space is continuous and cannot be explicitly represented or enumerated, the conventional tabular Q-learning approach becomes impractical due to the vast size of the state space.

To overcome this challenge, function approximation methods are employed to generalize Q-values across continuous state spaces. Function approximation enables the Q-function to be represented by a parameterized function, typically implemented using a neural network or other function approximators. Deep Q-learning (DQN) stands out as a prominent variant of Q-learning that leverages deep neural networks for function approximation, thereby facilitating the handling of continuous state spaces.

Other challenges are encountered dealing with continuous action spaces. The enumeration and comparison of an infinite set of actions becomes infeasible for continuous action spaces, presenting issues during both training and inference.

A potential solution to this problem involves discretizing the action space through binning. However, even with discretization, the action space may remain too large for practical computation.

Another approach is to employ a Policy Gradient Reinforcement Learning (RL) algorithm with a Gaussian policy. These methods aim to directly approximate the policy, generating the mean of a Gaussian distribution that represents the probability distribution over continuous actions given the state.

Recent advancements in Policy Gradient Reinforcement Learning (RL) algorithms aim to concurrently approximate both the policy and the Q-value function, referred to as the Actor and Critic, respectively. The Critic, representing the Q-value function, evaluates actions, providing a quantitative measure of their quality. Meanwhile, the Actor, embodying the policy, determines the strategy for selecting actions. This synergy between the Actor and Critic components contributes to more stable and efficient learning processes.

When both the Actor and Critic are realized as approximation functions based on neural networks, these algorithms are categorized as DDPG (Deep Deterministic Policy Gradients) algorithms.

4 Deep Q-learning

To update