Exercise 2 - Robot Learning Course

Andrea Ongaro
Computer Engineering
Politecnico di Torino, Italy
s329706@studenti.polito.it

Abstract: This article is the second in a series of three for the Reinforcement Learning laboratory course at Politecnico di Torino. In this report, it's explored the potential of Q-learning within the OpenAI Gym environment, focusing on the CartPole problem. The analysis includes experiments with different epsilon values and various Q-table initialization strategies.

Keywords: Reinforcement Learning, Robots, AI

1 Introduction

1.1 The system - CartPole-v0

CartPole-v0 system from OpenAI is part of the classic control environment. The starting code it's been provided by Andrea Protopapa in the course of Reinforcement Learning. Information about this system is sourced from the official documentation [1] and its GitHub repository. Below is the definition provided by the authors for CartPole-v0

This environment corresponds to the version of the cart-pole problem described by Barto, Sutton, and Anderson in "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problem". A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The pendulum is placed upright on the cart and the goal is to balance the pole by applying forces in the left and right direction on the cart.[1]

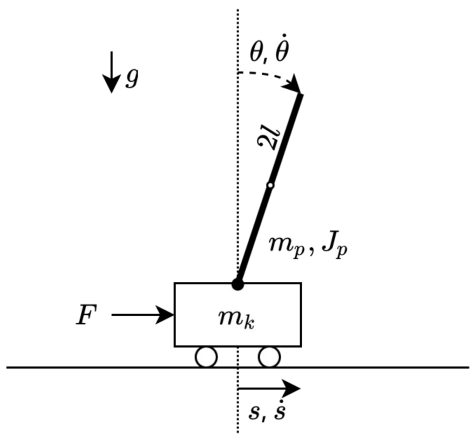

Figure 1: Graphical explation about the whole system

The observation is a four element vector:

$$s = \begin{pmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{pmatrix}$$

Figure 2: Observation vector

where

- x: the position of the cart
- $\dot{x}$: the velocity,
- θ: the angle of the pole
- $\dot{\theta}$: the angular velocity of the pole.,

1.2 Q-Learning

Q-learning is a model-free reinforcement learning algorithm that relies on trial-and-error methods. Unlike model-based approaches, it does not construct an internal model of the environment or explicitly use a Markov Decision Process. Instead, the agent learns directly from the outcomes of its actions by interacting with the environment.

Initially, the agent is aware of the set of possible states and actions within the environment but has no knowledge of how these states transition or what rewards they yield. Through exploration, the agent gradually discovers state transitions and rewards, enabling it to improve its policy over time.

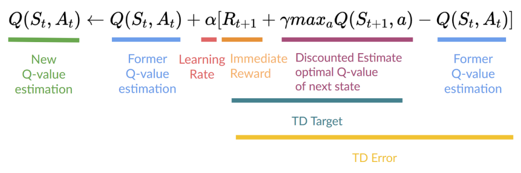
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$
New Q-value estimation
Former Q-value estimation
Learning Rate
Immediate Reward
Discounted Estimate optimal Q-value of next state
Former Q-value estimation
TD Target
TD Error

Figure 3: Q-value estimation

1.2.1 Exploration vs. Exploitation Tradeoff

An agent interacts with its environment by taking actions and observing the outcomes. During this process, the agent has a set of actions to choose from. Some actions may have been taken previously, allowing the agent to predict their outcomes, while others remain unexplored and unpredictable.

This balance between leveraging known actions and exploring new ones is called the **exploration-exploitation trade-off.**

- When the agent **explores**, it experiments with unfamiliar actions, expanding its knowledge and potentially improving long-term rewards.
- Conversely, when the agent **exploits**, it chooses actions based on what it already knows to maximize immediate rewards, even if this behavior might be suboptimal in the long run.

A **trade-off is necessary** since the agent cannot explore and exploit simultaneously . At the start of training, the agent has no knowledge about the consequences of its actions, so sufficient initial exploration is crucial. This allows the agent to identify actions that yield higher rewards. However, relying solely on exploitation can be risky, as it prevents the agent from discovering better strategies or optimal behaviors.

1.2.2 ϵ greedy

In Q-learning, the agent selects an action based on the reward it expects to receive, aiming to maximize the cumulative reward over time. It always chooses the action that leads to the highest estimated reward for the given state, which ensures that it generates the maximum possible reward in that state.

On the other hand, the epsilon-greedy strategy introduces a balance between exploitation (using prior knowledge to select the best action) and exploration (trying new actions to potentially discover better options). The agent typically selects the action with the highest estimated reward most of the time but occasionally explores other actions.

The strategy incorporates an exploration factor, denoted by ϵ (epsilon), which determines the probability of choosing a random action. With a small probability of ϵ, the agent will explore by selecting an action at random, regardless of its current knowledge of action values. The remaining time, with probability 1 - ϵ, the agent exploits its knowledge and chooses the action with the highest Q-value, thus maximizing the expected reward.

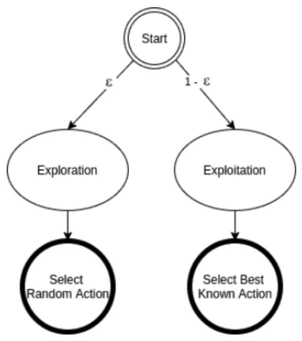

Figure 4: Exploitation vs Exploration [2]

1.2.3 GLIE - greedy in limit with infinite exploration

GLIE (Greedy in the Limit with Infinite Exploration) ensures that each action of a state that is visited often is, in turn, chosen infinitely often (with probabilty of 1) allowing for comprehensive exploration of the environment. Over time, the learning policy converges to a greedy approach with respect to the Q-value function.

1.2.4 Heatmap

A heatmap can be utilized to analyze the current state of the system under study. The heatmap depicts values for a main variable of interest across two axis variables as a grid of colored squares. The axis variables are divided into ranges like a bar chart or histogram, and each cell's color indicates the value of the main variable in the corresponding cell range [3].

Each cell can represent different states, ranging from high-value regions, which correspond to states where the agent expects to achieve high cumulative rewards (e.g., states near the upright pole position with low velocity and angular velocity), to low-value regions, which represent states likely to lead to termination or failure (e.g., the pole falling or the cart moving out of bounds).

2 Methodology

2.1 Fixed ϵ vs GLIE epsilon schedule

They're been developed three different strategies, two of which will be compared in the following paragraph. The first strategy uses a constant epsilon value of 0.2 ($\epsilon = 0.2$) while the second employs the following formula (Equation 2.1) where epsilon decreases to 0.1 after 20.000 episodes. To achieve this, we set the value of b to 2221, which ensures that at the end of the GLIE schedule, epsilon will be approximately equal to 0.09995049727735025

$$e_k = b/(b+k)$$

The reward trends for the two plots reveal distinct patterns. In the first plot (Figure 5), there is a rapid initial rise in rewards during the early episodes, indicating steep learning, followed by stabilization around an average of 150-175 with consistent fluctuations. The second plot (Figure 6) also shows a sharp initial rise, but the rewards exhibit greater variability in later episodes and seem to slightly decline after about episode 15,000.

In terms of variability, the first plot displays more stable reward trends after the initial learning phase, with spikes that are present but not extreme, whereas the second plot contains more pronounced and frequent fluctuations throughout training.

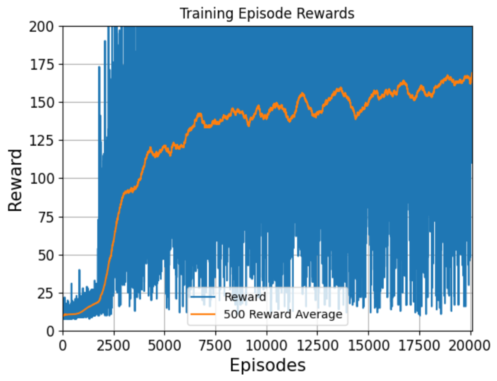

Figure 5: Reward obtained - $\epsilon = 0.2$

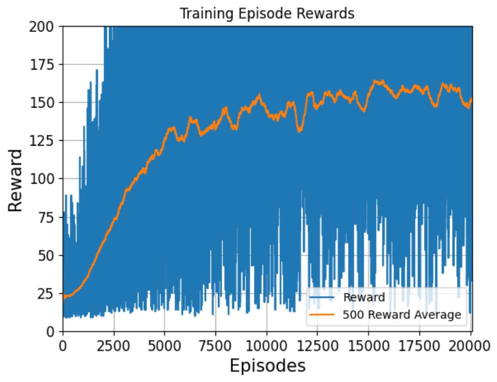

Figure 6: Reward obtained - Formula 2.1

2.1.1 Value function

After computing the associated value function, the heatmap of the computed value function in terms of x and θ,(the current position of the cart and the pole angle) was generated. To plot this, the values were averaged over $\dot{x}$ and $\dot{\theta}$.

The focus will be on the Q-value heatmap at different stages of the training: before training, after a single episode, after half of the episodes and at full training. The general trend is consistent between the two methodologies. These heatmaps can be found in the data/plots folder for further analysis.

2.1.2 Before training

Before training the heatmap is likely flat, with no clear structure. All state-action pairs have equal zero Q -values, meaning the value function is uniform across states. The agent has no prior knowledge of the environment. See Figure 7 and Figure 8.

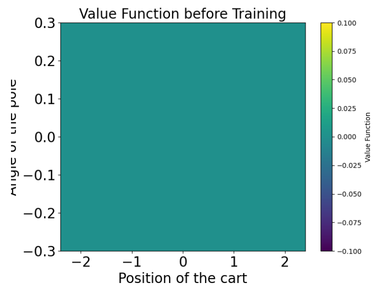

Figure 7: Heatmap in the initial state using $\epsilon = 0.2$

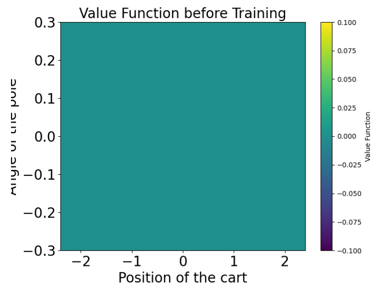

Figure 8: Heatmap in the initial state using Formula 2.1

2.1.3 After one episode

After one episode some regions of the heatmap may begin to show slightly higher values, particularly in states visited during the episode. The updates are sparse and localized around the trajectory the agent followed during the episode. Unvisited states retain their initial values, while the agent updates the Q-values for the specific states and actions it encountered, thereby improving its estimate of $V(s)$ in those regions. As a result, most of the state space remains unexplored, leading to minimal structure in the heatmap.

The second image (Figure 9) shows a wider range of values, suggesting the agent has learned a more diverse set of values across the different positions of the cart while the first one (Figure 10), with a narrower value range, suggests it has learned a more limited set of values after a single episode.

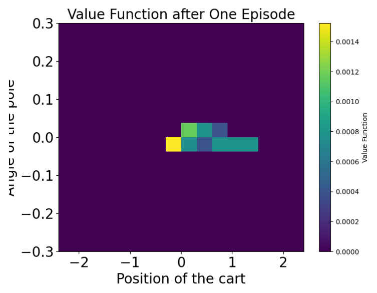

Figure 9: Heatmap after one single episode using $\epsilon = 0.2$

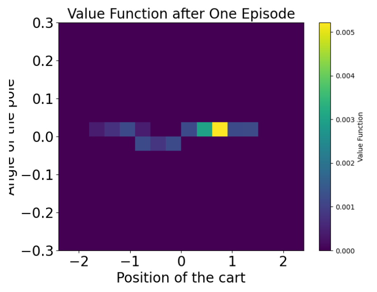

Figure 10: Heatmap after one single episode using Formula 2.1

2.1.4 Half-way training

The heatmap begins to take shape, revealing a clearer distinction between “good” and “bad” states. Some areas may still appear noisy or less structured, particularly in regions that have been underexplored. (Figure 11 and 12)

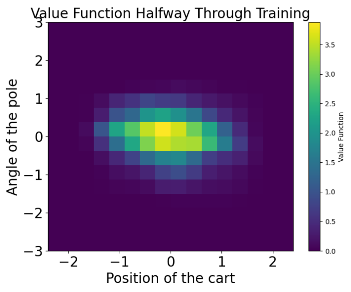

Figure 11: Halfway Training Heatmap using $\epsilon = 0.2$

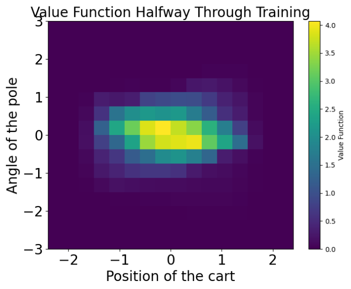

Figure 12: Halfway Training Heatmap using Formula 2.1

2.1.5 Full training

Analyzing the two full training heatmaps (Figure 13 and Figure 14), it is clear that the agent has effectively learned the optimal policy, even if they’re slighly different. The heatmaps are smooth and clear, with distinct high-value regions corresponding to optimal solution.

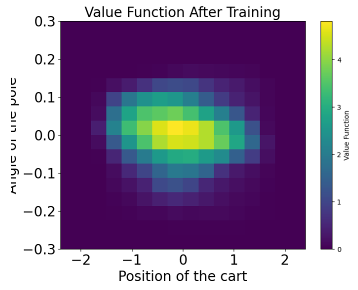

Figure 13: Heatmap image - $\epsilon = 0.2$

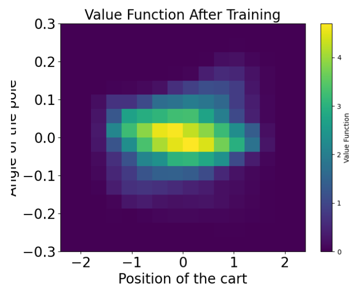

Figure 14: Heatmap image - Formula 2.1

2.2 Epsilon zero

The final strategy analyzed in this article involves setting epsilon to 0. With epsilon set to 0, there is no exploration, meaning the agent relies solely on its current knowledge without attempting to discover new states or actions. Without exploration, there is no guarantee that the agent can sufficiently learn the environment to identify the optimal policy and it may stuck in **sub-optimal policy**.
In this subsection, the scenario of epsilon equal to zero is examined under two different cases:

- keeping the initial Q-function values at zero;
- setting the initial Q-function values at 50 for all states and actions;

2.2.1 Comparison

Initializing the Q-function introduces a certain degree of bias. Initializing Q-values n to 50, the agent assumes it can achieve high rewards from all state-action pairs initially. As a result, convergence becomes slower because the updates to the Q-values take longer to overwrite these inflated initial values.

The second approach proves to be more effective, leading to a better estimation of the Q-function. This improvement stems from the nature of the greedy policy, which consistently selects actions that maximize the cumulative reward for a given state. When the Q-function is initialized to the same value for all state-action pairs, the greedy policy initially has no preference and is forced to randomly select actions. This randomness ensures that the agent begins to estimate the values of the actions it takes for each state.

Then the optimistic initialization encourages exploration. At first, the agent assumes all actions will yield high rewards (+50). However, as actions are taken, the agent receives rewards that are typically lower than this optimistic estimate. This “disappointment” motivates the agent to switch to other actions, resulting in all actions being explored multiple times before the Q-values converge. Consequently, this approach achieves a significant amount of exploration even when the policy remains greedy (i.e., always selecting the current best-known action).

As expected, instead, the first approach fails to discover new actions and results in suboptimal performance (Figure 15 and Figure 17)

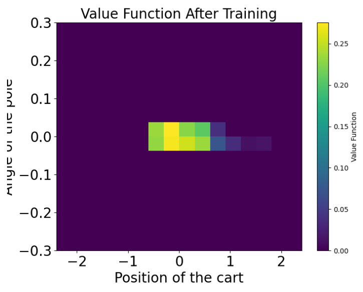

Figure 15: Full training Heatmap with $\epsilon = 0$

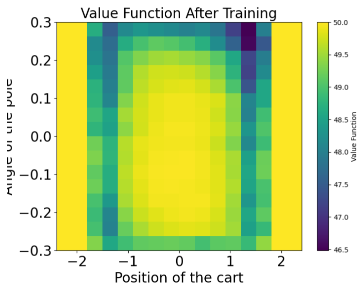

Figure 16: Reward obtained with $\epsilon = 0$ and $Qvalues = 50$

3 Q-Learning : Different applications

3.1 Continuous state spaces

Q-learning can be adapted for environments with continuous state spaces, but it requires some modifications because the algorithm inherently relies on a discrete state-action table. A common approaches for continuous state spaces is to use a state discretization, as it's been done in this article.

It is important to note that discretization can result in a loss of precision and is particularly affected by the "curse of dimensionality"[4] in high-dimensional spaces.

To overcome this problem there is an alternative that involves the use of the function approximators (e.g., neural networks, decision trees, or linear regression) to represent the Q-function Q(s, a) instead of a table. This permits to handles continuous state spaces directly without discretization and scales better to high dimensions and it's the foundation of Deep Q-Learning (DQN), where neural networks approximate $Q(s, a)$.

3.2 Continuous action spaces

The challenge of applying Q-learning to continuous action spaces becomes apparent when examining the TD Target (see Figure 3). The process of determining the maximum Q-value for the next state becomes increasingly inefficient and less accurate as the size of the action space grows.

In continuous action spaces, where the number of possible actions is infinite, it is impossible to simply scan all actions to identify the one that produces the maximum Q-value. This makes the traditional max operation computationally impractical.

One potential solution is to discretize the action space, reducing it to a finite set of actions. However, this approach can lead to a loss of precision and may not be suitable for all problems.

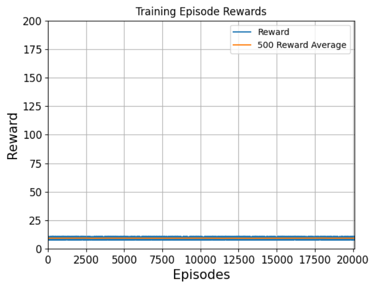

Figure 17: Full training Heatmap with $\epsilon = 0$

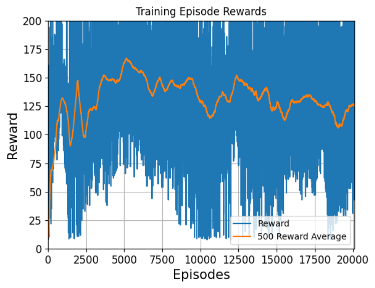

Figure 18: Reward obtained with $\epsilon = 0$ and $Qvalues = 50$

To address this issue, variants of Q-learning have been developed. These methods not only approximate the Q-value function but also approximate the max operation itself, enabling the algorithm to handle continuous action spaces more effectively.

References

[1] URL https://www.gymlibrary.dev/environments/classic_control/cart_pole/.

[2] URL https://www.baeldung.com/cs/epsilon-greedy-q-learning.

[3] URL https://www.atlassian.com/data/charts/heatmap-complete-guide.

[4] R. Bellman. Dynamic programming and stochastic control processes. *Information and Control*, 1(3):228–239, 1958. ISSN 0019-9958. doi:https://doi.org/10.1016/S0019-9958(58)80003-0. URL https://www.sciencedirect.com/science/article/pii/S0019995858800030.