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 installment, we explore the potential of Q-learning within the OpenAI Gym environment, focusing on the CartPole problem. Our 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]

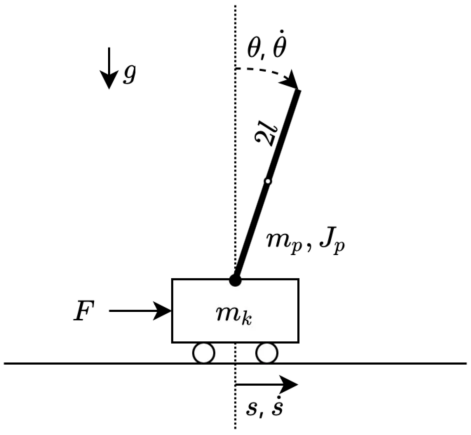

Figure 1: Graphical explation about the whole system

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 algorithm. We can think of model-free algorithms as trial-and-error methods. The agent explores the environment and learns from outcomes of the actions directly, without constructing an internal model or a Markov Decision Process. In the beginning, the agent knows the possible states and actions in an environment. Then the agent discovers the state transitions and rewards by exploration.

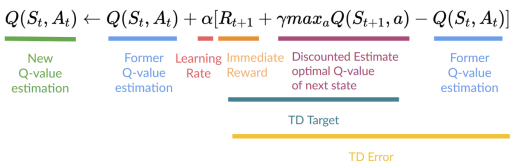

Figure 3: Q-value estimation

1.2.1 Exploration vs. Exploitation Tradeoff

The agent tries to discover its environment. During these trials, an agent has a set of actions to select from. Some of the actions are previously selected and the agent might guess the outcome. On the other hand, some actions are never taken before.

The concept of exploiting what the agent already knows versus exploring a random action is called the exploration-exploitation trade-off. When the agent **explores**, it can improve its current knowledge and gain better rewards in the long run. However, when it **exploits**, it gets more reward immediately, even if it is a sub-optimal behavior.

As the agent can't do both at the same time, there is a trade-off. Initially the agent doesn't know the outcomes of possible actions. Hence, sufficient initial exploration is required. If some actions lead to better rewards than others, we want the agent to select these options. However, only exploiting what the agent already knows is a dangerous approach.

1.2.2 ϵ greedy

In Q-learning, we select an action based on its reward. The agent always chooses the optimal action. Hence, it generates the maximum reward possible for the given state.

In epsilon-greedy policy, the agent uses both exploitations to take advantage of prior knowledge and exploration to look for new options. In the figure 4, it is possible to observe how the two actions are determined.

The epsilon-greedy approach selects the action with the highest estimated reward most of the time. The aim is to have a balance between exploration and exploitation. Exploration allows us to have some room for trying new things, sometimes contradicting what we have already learned.

With a small probability of ϵ, we choose to explore, i.e., not to exploit what we have learned so far. In this case, the action is selected randomly, independent of the action-value estimates. As we've discussed above, usually the optimal action, i.e., the action with the highest Q-value is selected. Otherwise, the algorithm explores a random action.

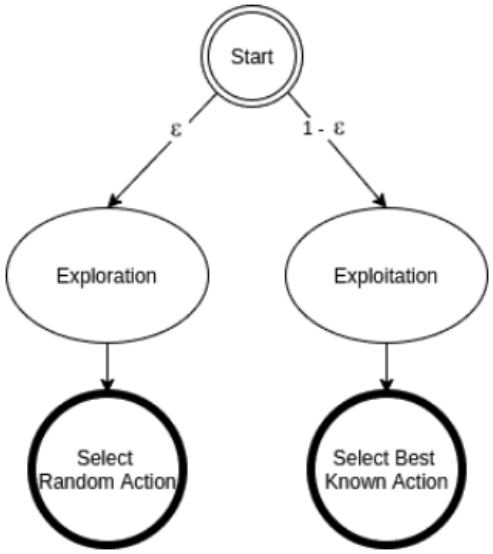

Figure 4: Exploitation vs Exploration [2]

1.2.3 GLIE - greedy in limit with infinite exploration

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

1.2.4 Heatmap

To analyze the current state of the system under study, a heatmap can be utilized. 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 Epsilon

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

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

2.1.1 Fixed ϵ vs GLIE epsilon schedule

When comparing the performance of a fixed epsilon (Figure 5) versus the GLIE epsilon schedule (Figure 6), there is not much difference between the two. Overall, the second strategy (with the GLIE epsilon schedule) starts with a lower reward, but ultimately achieves a higher one

Looking at the testing render, it is evident that the GLIE schedule leads to more frequent successful episodes compared to the fixed epsilon strategy, with the agent managing to keep the pole stable for longer durations. However, the performance is still not perfect, as the agent frequently fails to maintain stability.

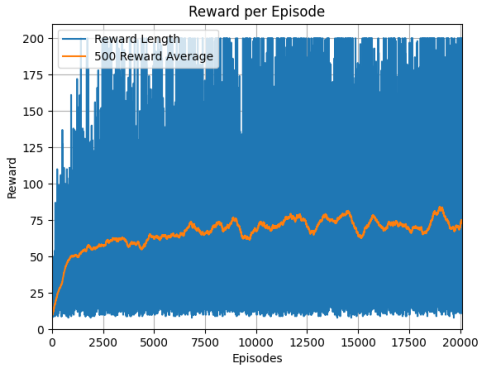

Figure 5: Reward obtained - $\epsilon = 2$

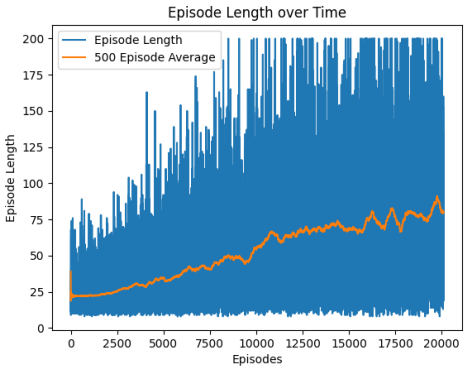

Figure 6: Reward obtained - Formula 2.1

2.2 Value function

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

Analyzing the full training heatmaps, one for the constant epsilon strategy (Figure 7) and the other for the GLIE schedule (Figure 8), it is clear that the agent has effectively learned the optimal policy. The heatmaps are smooth and clear, with distinct high-value regions corresponding to optimal states or policies.

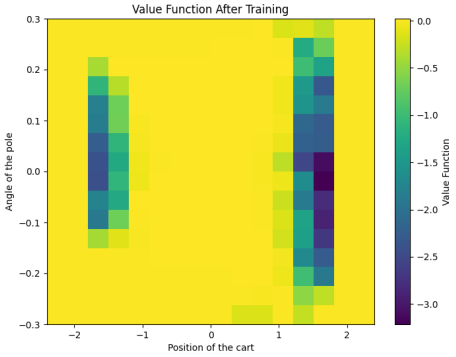

Figure 7: Heatmap image for q-learning with costant epsilon

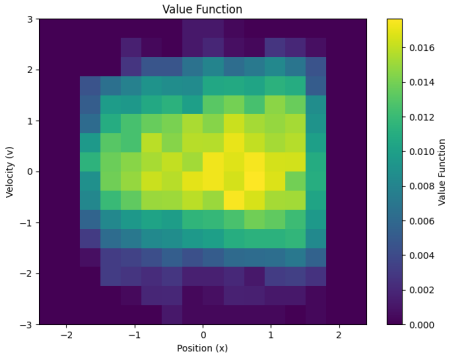

Figure 8: Heatmap image for q-learning with glie epsilon schedule

Next, the focus will be on analyzing the constant epsilon case, specifically examining the Q-value heatmap at different stages of the training: before training, after a single episode, and after half of the episodes.

The general trend is consistent. These heatmaps can be found in the data/plots folder for further analysis.

2.2.1 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.

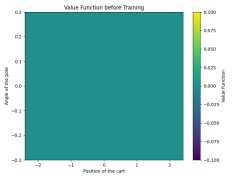

Figure 9: Heatmap in the initial state

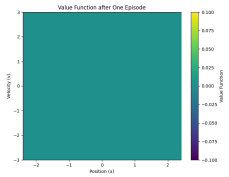

Figure 10: Heatmap in the initial state

2.2.2 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. See figure 12

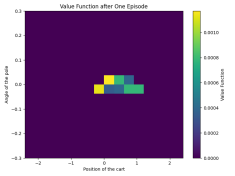

Figure 11: Heatmap after one single episode using $\epsilon = 0.2$

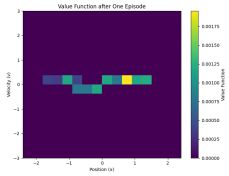

Figure 12: Heatmap after one single episode using Formula 2.1

2.2.3 Half-way training

The heatmap begins to take shape, revealing a clearer distinction between “good” and “bad” states. High-value regions correspond to states that are closer to the goal or lead to higher rewards, while low-value regions represent states where the agent anticipates poor outcomes or failures. Some areas may still appear noisy or less structured, particularly in regions that have been underexplored. See figure 14

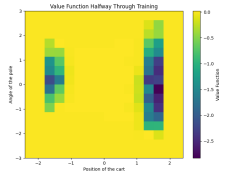

Figure 13: Halfway Training Heatmap using $\epsilon = 0.2$

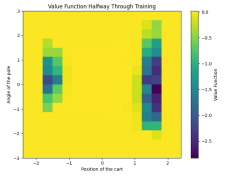

Figure 14: Halfway Training Heatmap using Formula 2.1

2.3 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;

2.3.1 Performance

Change the initialization introduce centainlty some bias. If Q-values are initialized to 50, the agent assumes it can achieve high rewards from all state-action pairs initially and the convergence will be slower because the updates of the Q-values will take longer to overwrite the inflated values.

The performance is poor in both cases; however, the strategy with higher initialization yields slightly better results, with longer episode durations.

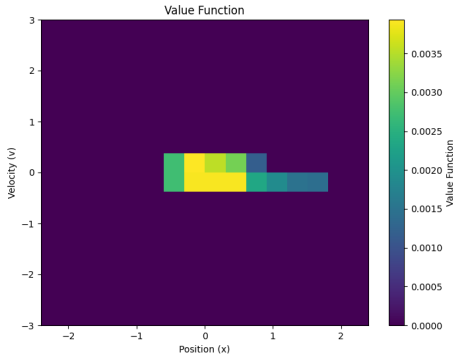

Figure 15: Full training Heatmap with epsilon zero

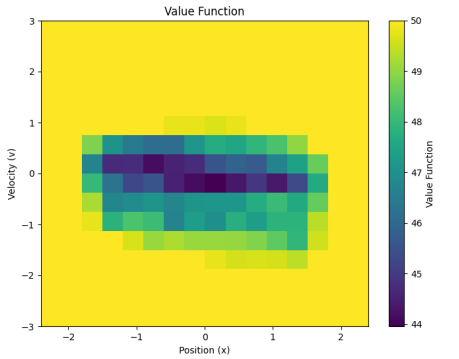

Figure 16: Full training Heatmap with epsilon zero and Q values initialized at 50

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 problem is can be understood easily looking the formula, the part about the TD Target (See Figure 3) The process for evaluating the maximum becomes less efficient and less accurate the larger the space that it needs to check.
If your actions are continuous, and therefore infinite, you can't just look up which action produces the maximum q-value because there are an infinite number of actions to scan. Unless you discretise the action space, then this becomes very unwieldy. There are variants of Q-learning which seek to solve this problem by not only approximating Q, but also approximating the max operation.

4 Conclusion

Domande non chieste: Standard deviation è molto alta sempre mi sa.

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.