Exercise 2 - Robot Learning Course

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

Abstract: This article is part of a series of three articles for the Reinforcement Learning course in Politecnico di torino. In this article, the second one, we analyze the potentiality of the q-learning in the context of open AI gym, specifically in the Cart Pole changing different value of epsilon and different initialization of the q-table.

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 and it's been used the sample code 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]

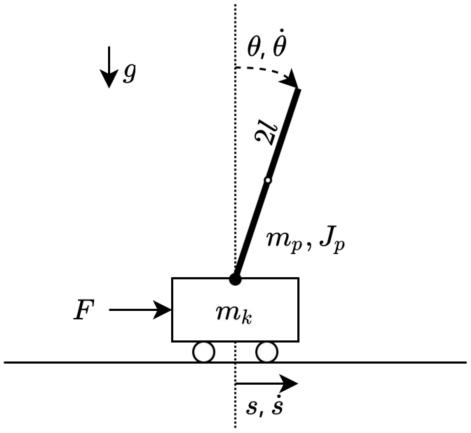

Figure 1: Graphical explation about the whole system

The observation is a four element vector:

where

- x: the position of the cart

7th Conference on Robot Learning (CoRL 2023), Atlanta, USA.

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

Figure 2: Observation vector

- $\dot{x}$: 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.

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 3.

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.

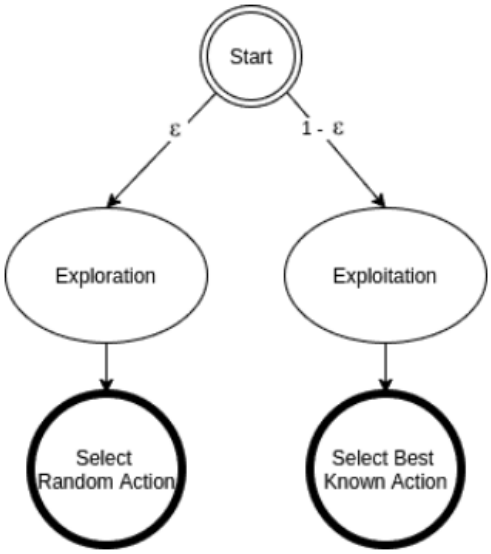

Figure 3: Exploitation vs Exploration [3]

1.2.3 Heatmap

To analyze the current state of the analyzed system it's possible to used an heatmap. 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 [2].

High-Value Regions: • Represent states where the agent expects to receive high cumulative rewards. • For example, in the cart-pole problem: • States near the upright position (balanced pole) with low velocity and angular velocity likely have higher values.

Low-Value Regions: • Represent states leading to termination or failure (e.g., the pole falling or the cart moving out of bounds). • For the cart-pole, states with extreme angles or high velocities might have low values.

1.2.4 GLIE - greedy in limit with infinite exploration

- Each action is executed infinitely often in every state that is visited infinitely often; - In the limit, the learning policy is greedy with respect to the Q-value function with probability 1

2 Process - Another name

2.1 Epsilon value

During this expirement we developed three different stategies that we are going to compare, in this subsection we are going to compare the first two. The first one with a costant epsilon ($\epsilon = 0.2$) while the second with the following formula 2.1 such that epsilon reaches 0.1 after k=20000 episodes (round b to nearest integer). In order to do this we have set the value of b as 2221 and in the end of the glie schedule we'll have epsilon exactly equal to 0.09995049727735025

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

2.1.1 Fixed ϵ vs GLIE epsilon schedule

Speaking about the performance between a fixed epsilon vs. the GLIE epsilon schedule

Plot the return obtained at each training episode, to view how the training process has progressed

Looking at the testing render you can see an improvement compared to the fixed epsilon so there is an higher change see the pole mantain the stability, but it's not perfect anyway.

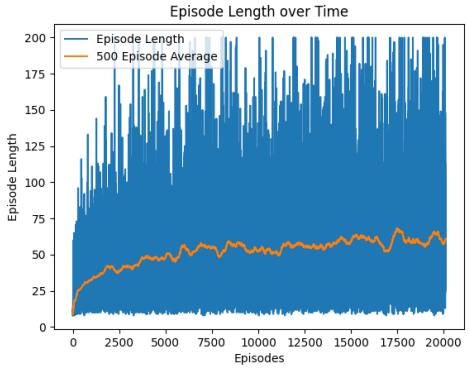

Figure 4: Return obtained at each training episode - $\epsilon = 2$

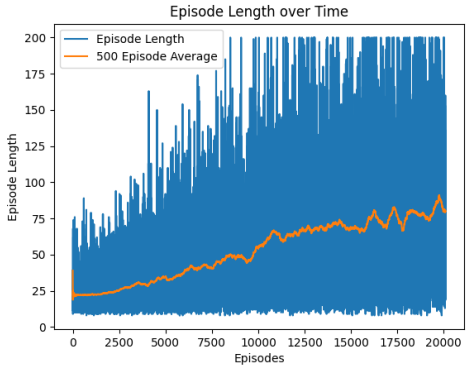

Figure 5: Return obtained at each training episode - ϵ get from the formula 2.1

2.2 Value function

After computing the associated value function d we generate the heatmap of the computed value function in terms of x and $\dot{\theta}$, (the current position of the cart and the pole angle) and therefore average the values over $\dot{x}$ and $\dot{\theta}$ for plotting.

After the full training The agent has effectively learned the optimal policy and accurately estimates V(s) and the heatmap is clear and smooth, with distinct high-value regions corresponding to optimal states or policies.

Now, we can take into consideration other states in which you can analyze the system.

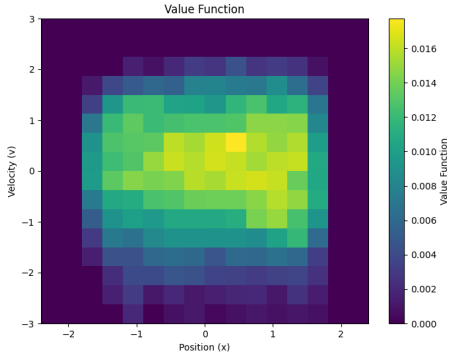

Figure 6: Heatmap image for q-learning with costant epsilon

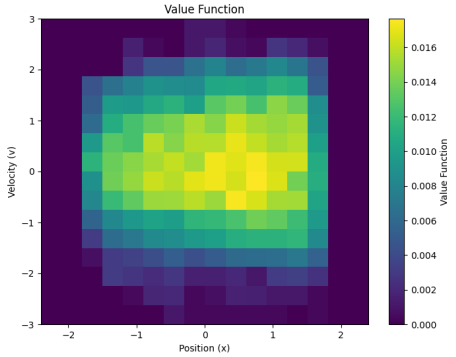

Figure 7: Heatmap image for q-learning with glie epsilon schedule

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.

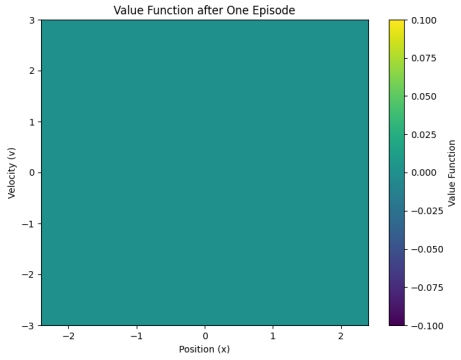

Figure 8: Heatmap in the initial state

2.2.2 After one episode

Some regions of the heatmap may start showing 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 still retain their initial values. the agent updates the Q-values for the specific states and actions it encountered, improving its estimate of V(s) in those regions. • Most of the state space remains unexplored, leading to minimal structure in the heatmap.

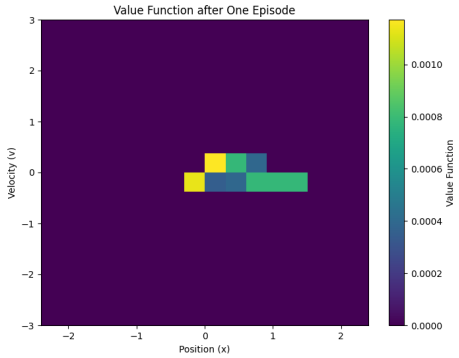

Figure 9: Heatmap in the initial state

2.2.3 Half-way training

The heatmap begins to take shape, showing a clearer distinction between “good” and “bad” states. • High-value regions correspond to states closer to the goal or those leading to higher rewards. • Low-value regions reflect states where the agent expects poor outcomes or failures. • Some areas might still be noisy or less structured, especially in underexplored regions.

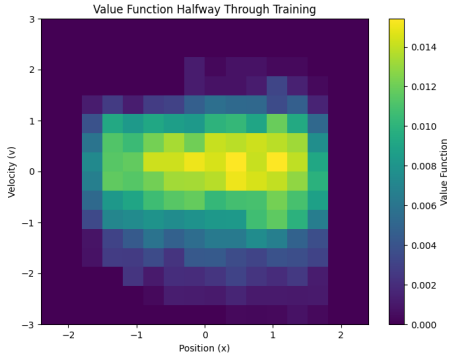

Figure 10: Halfway Training Heatmap

2.3 Epsilon zero

What’s the meaning in setting epsilon zero? There is not exploration stage, it will always use an known action. No reason to do this In this subsection we experiment epsilon equal to zero in two different casisticts:

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

- In which case does the model perform better? - Why is this the case? How does the initialization of Q values affect exploration?

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. • Divide the continuous state space into discrete bins (grids). • Each bin represents a discrete state, and standard Q-learning operates on this discretized state space. There is an important thing to notice that discretization can lead to loss of precision and suffers from the "curse of dimensionality"[**?**] 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. This is the foundation of Deep Q-Learning (DQN), where neural networks approximate Q(s, a).

3.2 Continuous action spaces

Risposta da dare studiando molto di più.

4 Conclusion

Domande: perchè faccio l'average sugli altri due dati. Non capisco. E' giusto che comunque faccia schifo i primi due, quello costante e quello con la formula. Con la formula migliora, ma non così tanto. Vuol dire che non sta trovando la policy ottimale?

Quale heatmap devo analizzare? E' uguale tanto il pattern dovrebbe essere sempre quello indipendente dall'epsilon impostato. Prima poi, poi sempre un po' di più.

References

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

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

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