Comparing Q-Learning with Epsilon Greedy Exploration on the Cart-Pole Problem

Linda Motejl, guided under Petr Švarný

***Abstract*—This research paper will explore the implementation of using Q-Learning with Epsilon greedy exploration reinforcement learning performed on the Cart-Pole problem, provided by Open AI Gym**

***Index Terms*—Reinforcement Learning, Q-Learning, Cart-Pole Problem, Epsilon Greedy.**

I. Introduction

REINFORCEMENT learning is the study of teaching an agent to maximize reward in a particular situation. The agent or learner is immersed in an environment where it must solve a problem, but does not know how to solve it. The learner therefore is forced to discover how to solve the problem based on taking actions. Through trial and error, the agent learns how to solve the problem through experience. [1]

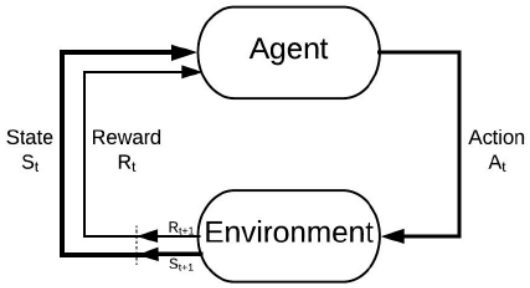

Fig. 1. Markov Decision Process

Fig 1. Displays the Markov Decision Process which mathematically presents a formalized way of displaying decision making in reinforcement problems. The agent takes one of the possible actions that is allowed in the environment based on the state that was given by the environment. Then based on the action the agent performed in some state, it receives either a reward or punishment. If the action that the agent performs is on target of the goal of the problem, it receives a reward. However, if the action did not facilitate in solving the problem, the agent will receive a punishment. The episode ends when it reaches a certain set of parameters which are unique to each problem.

II. Q-Learning

Q-Learning is an algorithm that seeks to learn a policy that maximizes the total reward. It is considered an off policy reinforcement learning algorithm because it learns from actions that are outside the current policy, by taking random actions. The 'Q' in Q-Learning stands for quality, which represents how useful a given action is in gaining some future reward. (cite)

Q-Learning utilizes a Q-Table which is initially set to be empty. This table will be updated after the end of each episode containing the Q-Values that were obtained by the agent. The Q-value is the optimal sum of discounted rewards associated with a state-action pair, representing how well the action was performed.

The Q function defines the value of taking action a in state s under some policy π. (cite)

III. Exploration and Exploitation

Before taking an action, the agent must decide whether it will choose to either explore or to exploit the environment. To explore denotes as to gather more information by taking a random action. To exploit an environment signifies to reference the Q-table, where in previous episodes Q-values have been inserted by exploration. The agent picks the highest Q-value that is provided as its next move.

An agent must perform actions based from its experience that consists of actions taken from the past. These are actions that have proved to be effective for obtaining rewards. But in order to exploit those actions, an agent must try actions that are random in order to explore if better actions may exist that the agent is unaware of. Since the action is purely random, it may happen that the agent picks an action that has already been explored given that state space. [2]

It is important to be able to balance the two. If an algorithm depends too much on exploration, the agent may face huge loss because it had not explored more optimal solutions, however if the agent spends too much time exploring, the solution will take very long to find, or maybe not at all. Therefore it is important to find the right balance between exploration and exploitation. Using Epsilon denoted as ε, the problem of being able to balance the trade between exploring and exploiting. It is responsible of deciding whether the agent shall explore or exploit based on its epsilon rate. An epsilon of 1 would signify the agent would take only random actions, an epsilon of 0 mean that the agent would only exploit its knowledge from past experience.

IV. Q-Learning with Epsilon Greedy exploration

A simple strategy is to use Q-Learning with Epsilon Greedy Exploration. It offers high exploration in the beginning, and through its knowledge and experience, high exploitation at the end. The goal is to maximize the total amount of reward in the long-term.

Algorithm 1: Epsilon-Greedy Q-Learning Algorithm

Data: α: learning rate, γ: discount factor, ϵ: a small number
Result: A Q-table containing Q(S,A) pairs defining estimated optimal policy π^*

```
/* Initialization                                          */
Initialize Q(s,a) arbitrarily, except Q(terminal,.);
Q(terminal,.) ← 0;
/* For each step in each episode, we calculate the
   Q-value and update the Q-table                          */
for each episode do
    /* Initialize state S, usually by resetting the
       environment                                         */
    Initialize state S;
    for each step in episode do
        do
            /* Choose action A from S using epsilon-greedy
               policy derived from Q                       */
            A ← SELECT-ACTION(Q, S, ϵ);
            Take action A, then observe reward R and next state S';
            Q(S, A) ← Q(S, A) + α [ R + γ max_a Q(S', a) - Q(S, A)];
            S ← S';
        while S is not terminal;
    end
end
```

Fig. 2. Pseudocode of E Greedy

The pseudo code states that the epsilon value is first initialized to 1, meaning that all actions are random. Through an epsilon decay value such as 0.99995, the epsilon number will decay ever so slightly at the end of every epoch (batches of episodes). A random number is generated, and if this number is higher than the epsilon value, then it will reference the q-table and choose the action that will produce the highest reward, else it will choose a random action and thus explore. The result of this algorithm is a Q-Table containing Q(S,A) pairs defining the estimated optimal policy π^*.

This algorithm is highly efficient because it chooses to explore in the beginning in order to update the q-table with valuable q-tables. As the Q-table is more and more filled, the agent will then choose to exploit on that information.

V. Q-Learning

In standard Q-learning, all of the concepts are the same as previously mentioned above, however the key difference is that the epsilon value is set to unalterable value such as 0.3. This means that 30% the agent will be taking a random action and exploring different state action pairs, while 60% of the time the agent will reference the Q-table for the best action to take given the state that it is in. Just as in the E greedy algorithm, the Q-table is updated at the end of each episode if the agent is tasked to explore.

```
define epsilon, alpha, gamma
initiate Q-table with zeros
observe initial state s
repeat:
      select an action a
         with probability ε select random action a
         else select action a = argmax(Q(s,a'))
      carry out action a
      observe next state s' and reward r
      calculate Q_target = r + gamma*max[Q(s',A)]
      Calculate Q_delta = Q_target - Q(s,a)
      add alpha*Q_delta to Q(s,a)
      s = s'
until terminated
```

Fig. 3. Pseudo code of Q-Learning

VI. Cart Pole Problem

A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every time-step that the pole remains upright. The cart must move in a systematic way in order to keep it in a range theta. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center. The game is considered to be solved if the average reward is greater than 195 over 100 consecutive trials. [3]

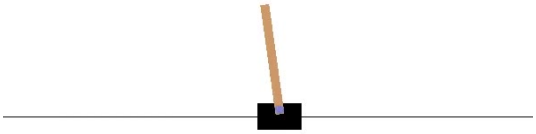

Fig. 4. Cart Pole Problem

The environment contains 4 values which describe the state that include:

1) Cart Position
2) Cart Velocity
3) Pull Angle
4) Pull Angular Velocity

The environment further on includes an Action Space that includes :

1) Left
2) Right

The action space describes the possible actions the agent can take.

Using reinforcement learning, the agent is capable of exploring several different state action pairs in order to find the best policy.

VII. Hypothesis on performance of Q-Learning vs E-greedy

As a result from the previous research on both algorithms, it is speculated that the ε-greedy algorithm will perform better than the Q-Learning algorithm when tested under the same parameters. This is because ε-greedy prioritizes exploration from the start, meaning that the agent will test many more state-action pairs than Q-Learning. It is more logical to put priority in exploration rather than to exploit what little knowledge the agent may have gained from the beginning. ε-greedy is as well better in producing better refinement due to the fact that an agent that never explores enough, may never observe an optimal trajectory even after using the best actions it has learnt, if not given enough time.

Both algorithms will be measured under the same circumstances which means that they will be tested in the same environment and use the same measure of success. Each algorithm shall run 60,000 episodes in order to measure the rate of improvement.

VIII. Results of E-Greedy

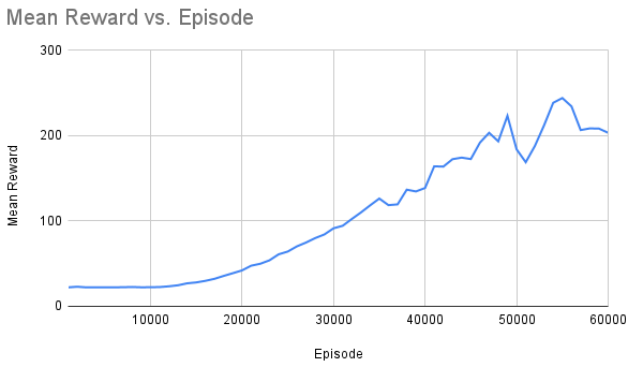

Fig. 5. Mean Reward vs Episode Result using E greedy

Figure 5 displays the data of a trial result of how the ε-greedy algorithm performed in 60,000 episodes. Data may vary differently after rerunning the experiment. The Y axis represents the mean reward of the agent against the number of episodes found on the X axis. Learning does not improve for the first 10,000 episodes and the average reward is 20, however it then improves slowly for the next 10,000 episodes. It may be observed that the rate of learning grows closely in a linear fashion. The agent reaches the benchmark of winning the game after 47,000 episodes. The Mean reward rise is stable until it dips at 49,000 episodes where the mean reward declines to a score of 168. It however quickly improves and peaks at a score of 244 on the 55,000th episode. For the remainder of the experiment, the mean score stabilizes at 210, which is comfortably above the winning bench mark.

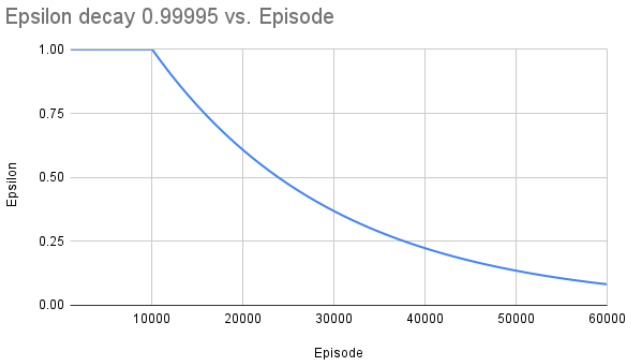

Fig. 6. Epsilon value vs Episode

The εis initially set to 1, which means that the probability that the agent will explore will be 100%, signifying that all actions that are taken are random. For the first 1000 episodes, the agent purely only explores, since εdecreases at the end of each epoch. However with a epsilon decay set at 0.99995, the value of epsilon decreases, allowing the agent to exploit progressively it's gained knowledge. As Epsilon decreases, it is evident that the mean reward increases in figure 5.

IX. Results of Q-Learning

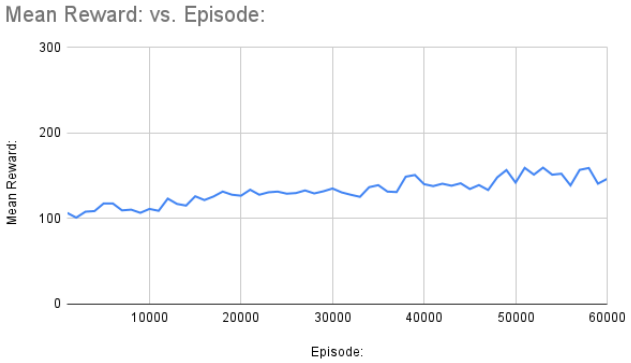

Fig. 7. Mean Reward vs Episode result using Q-Learning

Immediately it may be observed that the average mean reward is much higher, starting at a little over 100. This is due to the fact that the epsilon value is set to 0.3. Unlike in Figure 6 where it can be seen that the epsilon is constantly decreasing, the epsilon in Q-learning methodology stays stagnant. Random actions are not taken as much like in ε-greedy. Although an impressive start, the learner is no longer able to grow at a quick rate. The mean reward increases only slimly, however consistently. The reward peaks at 160. After 60,000 episodes, the algorithm is never able to solve the Cart-Pole problem. It is likely that the algorithm would be eventually able to reach a score of 195, however the amount of time it would take makes this algorithm unsuitable, and it is recommended to use a different approach such as the ε-greedy.

X. Conclusion

ε-greedy produces worse results after the few first epochs, but then produces higher quality results in the long-term compared to Q-Learning. It is therefore concluded that ε-greedy has been a more successful algorithm to use in this simulation on the Cart-Pole Problem.

REFERENCES

[1] R. S. Sutton and A. G. Barto. *Reinforcement Learning An Introduction*. MIT Press Ltd., 1998.
[2] Csaba Szepesvari. *Algorithms for Reinforcement Learning*. Morgan Claypool Publishers, 2010.
[3] Open AI Gym. CartPole-v1, 2022.