Subject: Reinforcement Learning

AY: 2023 – 24

Experiment 1

Name: Kresha Shah
SAP ID: 60009220080

Exploration Exploitation Dilemma AIM:

a) To solve the exploration exploitation dilemma using epsilon greedy strategy
b) To understand the effect of epsilon by comparing the different values of epsilon

THEORY:

With partial knowledge about future states and future rewards, our reinforcement learning agent will be in a dilemma on whether to exploit the partial knowledge to receive some rewards or it should explore unknown actions which could result in much larger rewards.

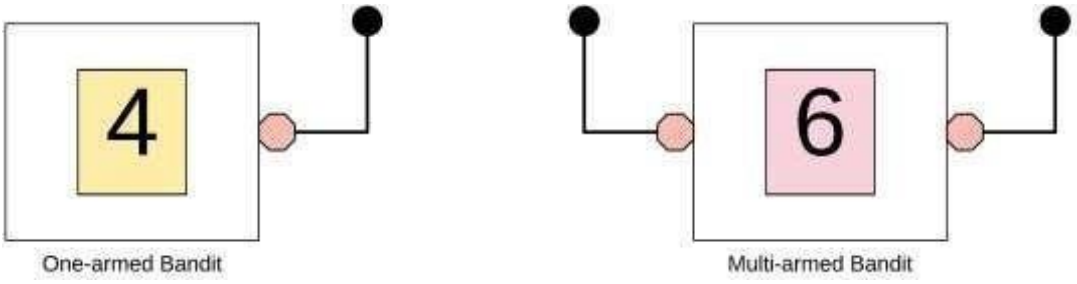
- Exploitation: Make the best decision given current information. The best long-term strategy may involve short-term sacrifices
- Exploration Gather more information. Gather enough information to make the best overall decisions

However, we cannot choose to explore and exploit simultaneously. In order to overcome the Exploration-Exploitation Dilemma, we use the Epsilon Greedy Policy.

MULTI ARMED BANDIT PROBLEM (MAB)

In a multi-armed bandit problem (MAB) (or n-armed bandits), an Agent makes a choice from a set of actions. This choice results in a numeric reward from the Environment based on the selected action. In this specific case, the nature of the Environment is a stationary probability distribution. By stationary, we mean that the probability distribution is constant (or independent) across all states of the Environment. In other words, the probability distribution is unchanged as the state of the Environment changes. The goal of the Agent in a MAB problem is to maximize the rewards received from the Environment over a specified period.

The MAB problem is an extension of the “one-armed bandit” problem, which is represented as a slot machine in a casino. In the MAB setting, instead of a slot machine with one-lever, we have multi-levers. Each lever corresponds to an action the Agent can play. The goal of the Agent is to make plays that maximize its winnings (i.e., rewards) from the machine. The Agent will have to figure out the best levers (exploration) and then concentrate on the levers (exploitation) that will maximize its returns (i.e., the sum of the rewards).



Left: One-armed bandit. The slot machine has one lever that returns a numerical reward when played.

Right: Multi-armed bandits. The slot machine has multiple (n) arms, each returning a numerical reward when played. In a MAB problem, the reinforcement agent must balance exploration and exploitation to maximize returns.

Epsilon-greedy

The agent does random exploration occasionally with probability ε and takes the optimal action most of the time with probability 1− ε.

Epsilon greedy method. At each step, a random number is generated by the model. If the number was lower than epsilon in that step (exploration area) the model chooses a random action and if it was higher than epsilon in that step (exploitation area) the model chooses an action based on what it learned.

Usually, epsilon is set to be around 10%. Epsilon-Greedy can be represented as follows:

$$A_t \leftarrow \begin{cases} argmax\ Q_t(a) & with\ probability\ 1-\epsilon \\ a \sim Uniform(\{a_1 ... a_k\}) & with\ probability\ \epsilon \end{cases}$$

The Action that the agent selects at time step t, will be a greedy action (exploit) with probability

(1-epsilon) or may be a random action (explore) with probability of epsilon.

ALGORITHM:

Algorithm 2: Epsilon-Greedy Action Selection

Data: Q: Q-table generated so far, : a small number, S: current state

Result: Selected action

Function *SELECT-ACTION(Q, S, ϵ)* **is**

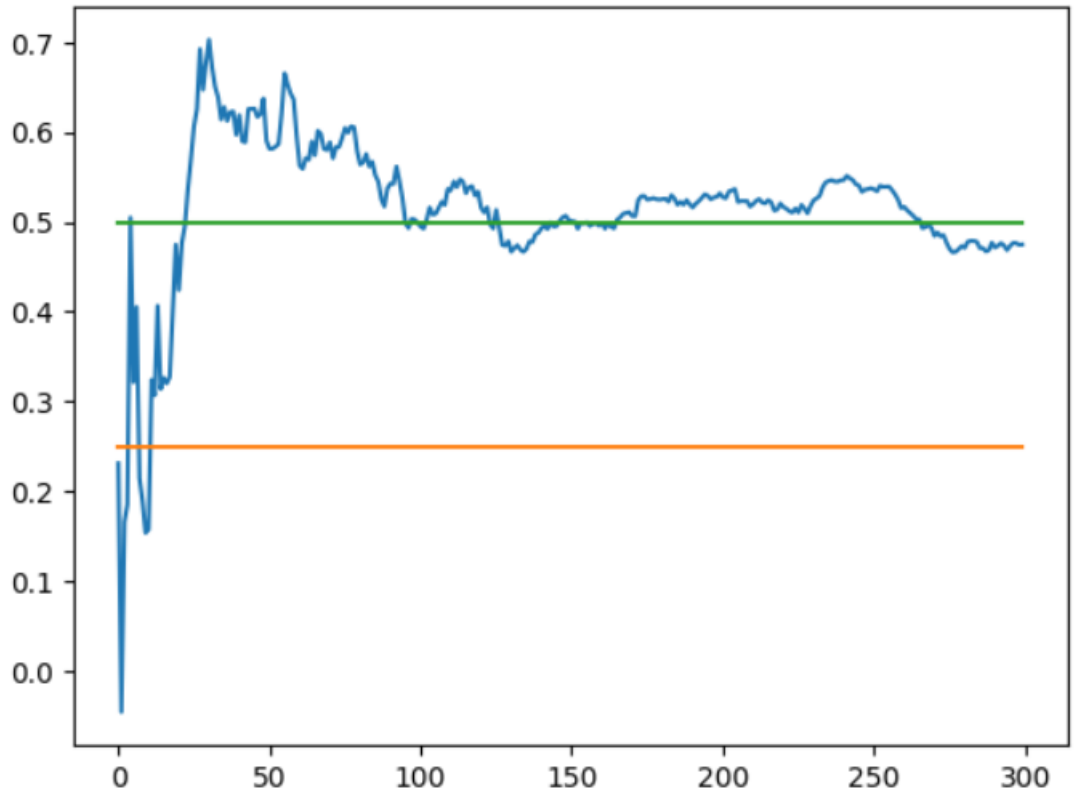
```
n ← uniform random number between 0 and 1;
if n < ϵ then
    A ← random action from the action space;
else
    A ← maxQ(S,.);
end
return selected action A;
```

end

LAB ASSIGNMENT TO DO:

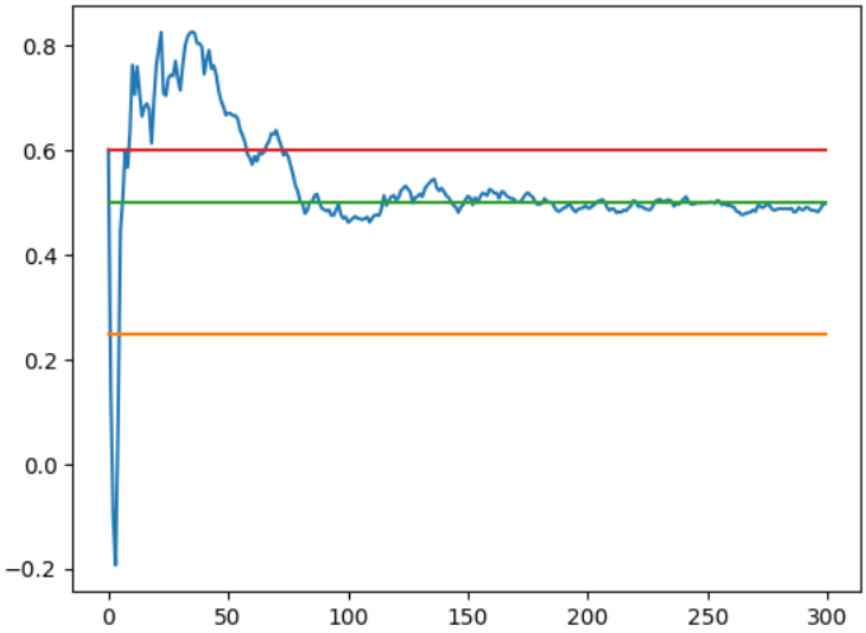
1. Create a multi armed bandit agent which would estimate the win rate using the epsilon -greedy strategy.

```
[7]  c2 = two_arm_bandit_epsilon_greedy(0.25, 0.5, 0.05, 300)
```



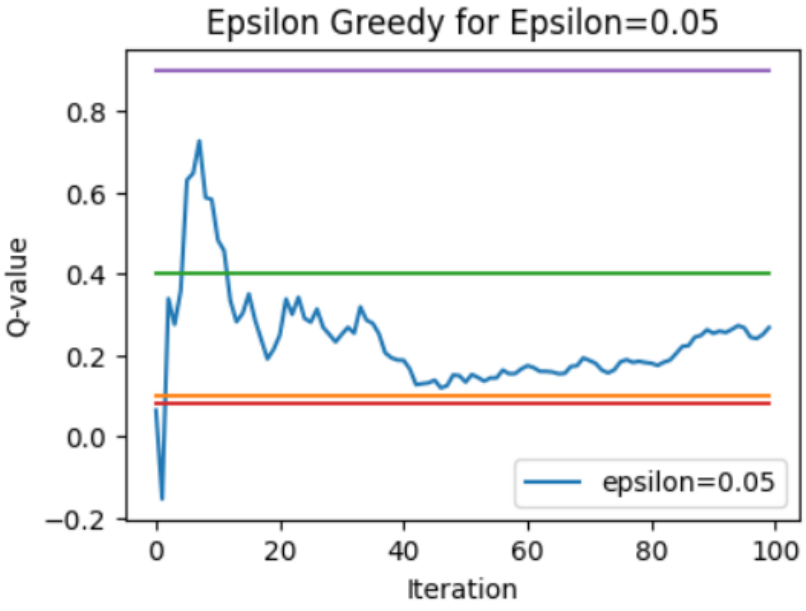

```
No of times explored:  24
No of times exploited:  276
Mean of rewards from arm 1: -0.3522636253800862
Mean of rewards from arm 2: 0.5611991674879656
```

```
[11] c1 = three_arm_bandit_epsilon_greedy(0.25, 0.5,0.6, 0.5, 300)
```




```
No of times explored:  136
No of times exploited:  164
Mean of rewards from arm 1: 0.22384377722667678
Mean of rewards from arm 2: 0.8700286481606592
Mean of rewards from arm 3: 0.2616494198101825
```

```
Enter the number of arms: 4
Enter the mean for arm 1: 0.1
Enter the mean for arm 2: 0.4
Enter the mean for arm 3: 0.08
Enter the mean for arm 4: 0.9
Enter the number of epsilon values: 4
Enter epsilon value 1: 0.05
Enter epsilon value 2: 0.7
Enter epsilon value 3: 0.001
Enter epsilon value 4: 0.5
Enter the number of iterations (N): 100
```



```
No of times explored:  57
No of times exploited:  43
Mean of rewards from arm 1: 0.07717758180422292
Mean of rewards from arm 2: 0.22843064502485255
Mean of rewards from arm 3: 0.0998080566627195
Mean of rewards from arm 4: 0.869048370839483
```

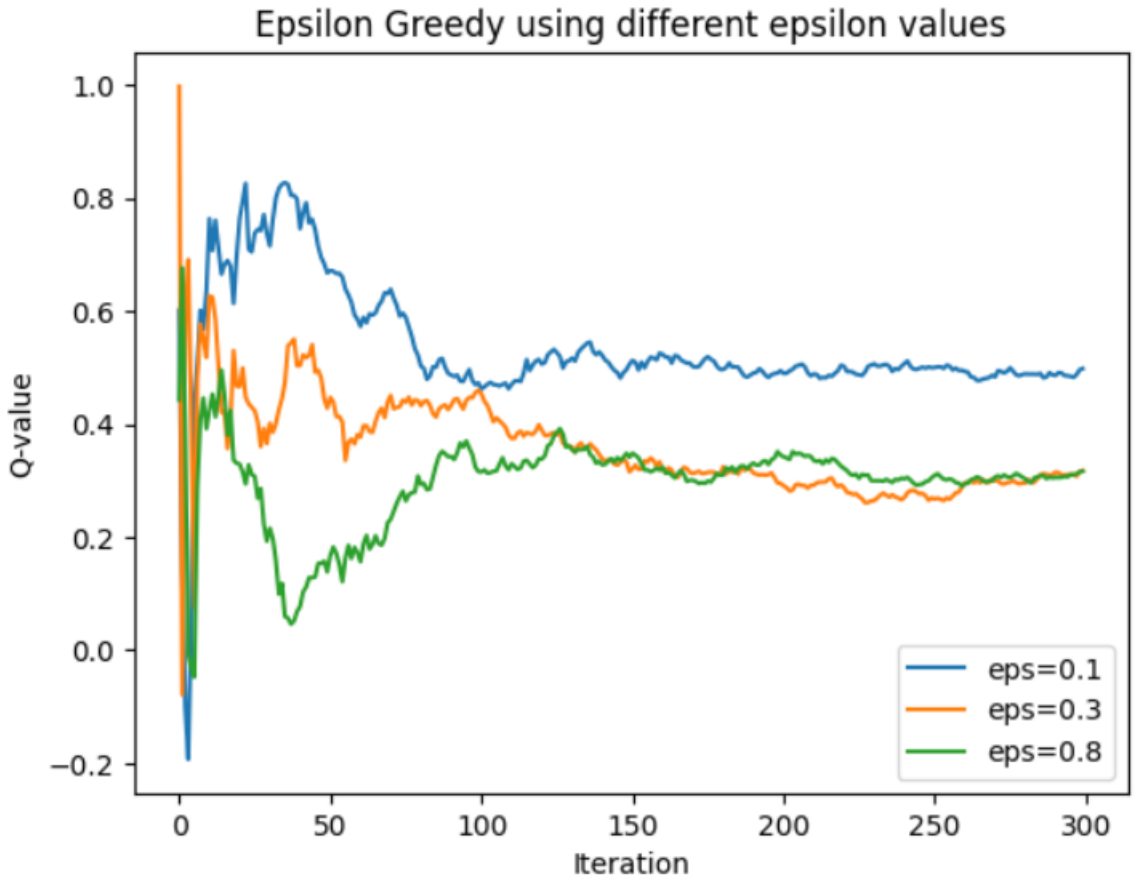
2. Understand the effect of the value of epsilon on the win rate by comparing the win rates corresponding to different values of epsilon. Hence draw conclusions.

```
<matplotlib.legend.Legend at 0x7d1b8429cb50>
```

Performance:

- The line with **epsilon 0.1** seems to achieve the **highest Q-value** throughout most of the training. This suggests that for this particular problem, a **lower epsilon value led to better performance**.
- The line with **epsilon 0.75** has the **lowest overall Q-value**. This is likely due to the **excessive exploration** preventing the agent from effectively exploiting the best arm.
- The line with **epsilon 0.5** shows a **balance between exploration and exploitation**, resulting in a **performance between the other two epsilon values**.

```
<matplotlib.legend.Legend at 0x7d1b8406f910>
```



Exploration vs. Exploitation:

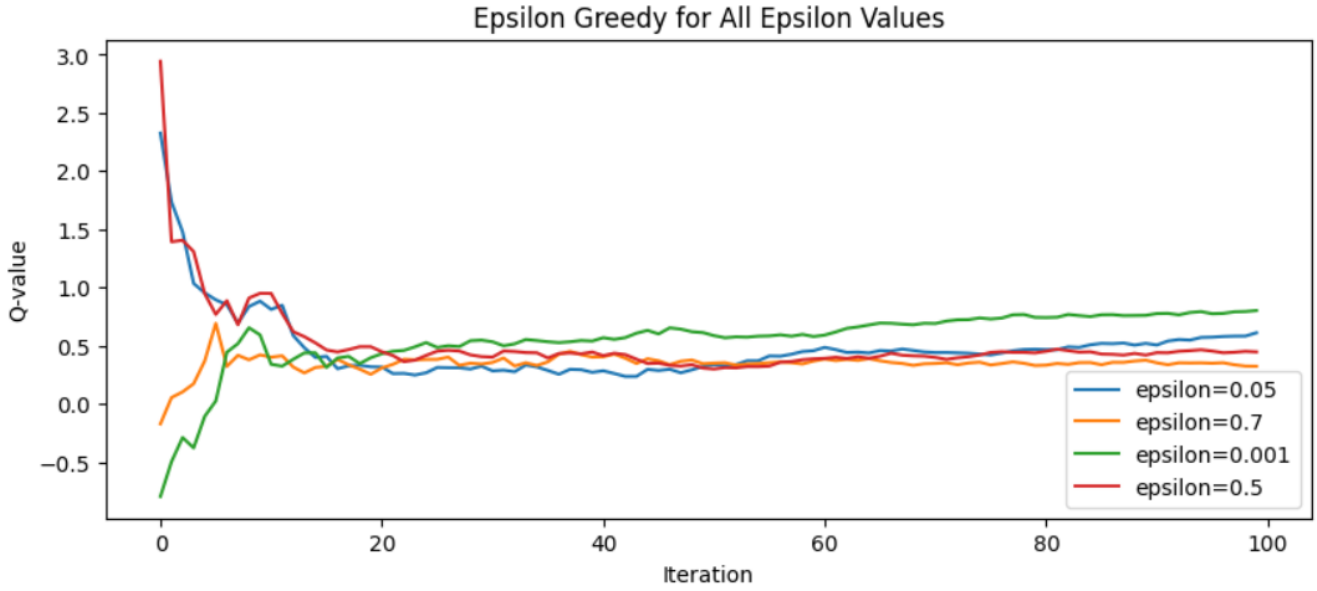
- As expected, **higher epsilon values lead to more exploration** as the agent is more likely to choose random actions. This can be seen in the graph by the **wider fluctuations** in the lines with higher epsilon values (0.3 and 0.8) in the early stages of the experiment.
- **Lower epsilon values lead to more exploitation** as the agent tends to stick with the arm that it believes to be the best. This can be seen in the graph by the **smoother lines** with lower epsilon values (0.1) and their **faster initial increase** in Q-value.

Performance:

The line with **epsilon 0.1** seems to achieve the **highest** Q-value throughout most of the training. This suggests that for this particular problem, a lower epsilon value led to better performance.

The line with **epsilon 0.8** has the **lowest** overall Q-value. This is likely due to the excessive exploration preventing the agent from effectively exploiting the best arm.

The line with **epsilon 0.3** shows a **balance** between exploration and exploitation, resulting in a performance between the other two epsilon values.



Conclusions that can be drawn from the graph:

- **The performance of the algorithm improves with more iterations.** This is because the algorithm is able to learn more about the rewards of each arm over time.
- **The higher the value of epsilon, the more exploration the algorithm does.** This can be seen in the fact that the lines with higher values of epsilon are more erratic in the early iterations.
- **The higher the value of epsilon, the slower the algorithm converges to the optimal solution.** This is because the algorithm is spending more time exploring suboptimal arms.