Artificial Intelligence Lab Report 9

1st Dipean Dasgupta
202151188
BTech CSE
IIIT,Vadodara

2nd Shobhit Gupta
202151149
BTech CSE
IIIT,Vadodara

3rd Rahul Rathore
202151126
BTech CSE
IIIT,Vadodara

4th Rohan Deshpande
202151133
BTech CSE
IIIT,Vadodara

***Abstract*—Understanding exploitation and exploration in the context of reinforcement learning—more especially, the n-arm bandit problem—is the goal of this lab. The epsilon-greedy algorithm, a well-liked exploration-exploitation tactic in reinforcement learning, will be the main focus of the project.**

I. Introduction

The n-arm bandit problem is a traditional reinforcement learning problem in which n "one-armed bandits" or slot machines with varying payoff probabilities are used. By playing the machines frequently and taking note of the results, the objective is to identify the machine with the biggest expected payoff.
Selecting the machine that, given the available information, is expected to yield the most payoff is known as exploiting it. By selecting machines with lower predicted rewards, on the other hand, exploration aims to learn more about the true reward probabilities of these machines.
A straightforward method that strikes a balance between discovery and exploitation is the epsilon-greedy algorithm. With a probability of 1-epsilon, the algorithm chooses the machine with the highest estimated payoff, and with a probability of epsilon, it chooses a random machine.

1. Consider a binary bandit with two rewards 1-success, 0-failure. The bandit returns 1 or 0 for the action that you select, i.e. 1 or 2. The rewards are stochastic (but stationary). Use an epsilon-greedy algorithm discussed in class and decide upon the action to take for maximizing the expected reward. There are two binary bandits named binaryBanditA.m and binaryBanditB.m are waiting for you.

Ans. - To use an epsilon-greedy algorithm to decide which action to take for maximizing the expected reward from the two binary bandits, we can follow these steps:

1) Define the epsilon value, which determines the probability of taking a random action instead of the greedy action that maximizes the expected reward.
   For example, epsilon = 0.1 means that there is a 10% chance of taking a random action.
2) Initialize the expected rewards for each action in each bandit to zero. This can be done using a 2x2 matrix.
3) Repeat the following steps for a fixed number of iterations or until convergence:
   a) With probability epsilon, select a random action. Otherwise, select the action with the highest expected reward (the greedy action).
   b) Observe the reward from the selected action and update the expected reward for that action using a simple average of the previous reward and the new reward.
4) After the iterations are completed, select the bandit with the highest expected reward.

The following findings were derived from the application of the binaryBanditA and binaryBanditB reward functions:

Initially, the predicted payouts for bandit A were 0 when the reward probability was low. But the anticipated benefits likewise rose with the number of trials.
In contrast, the projected rewards for banditB were initially close to 1, even though the reward probability was high (0.8 and 0.9). Because of the averaging component, the projected rewards did not change as the number of steps rose.

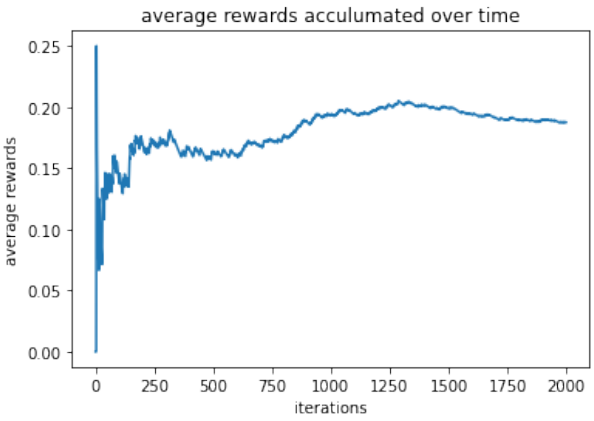

Fig1. : Expected rewards we get after using banditA

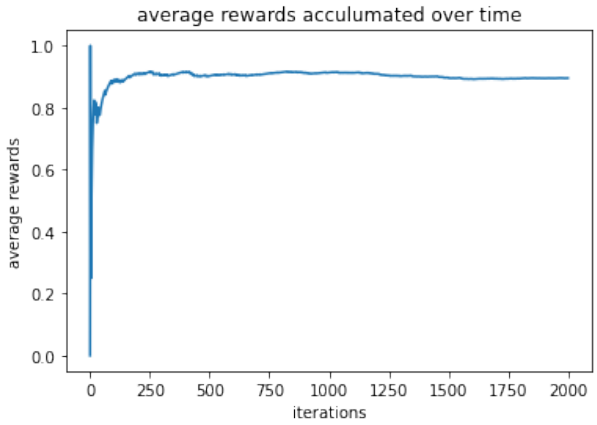

Fig2. : Expected rewards we get after using banditB

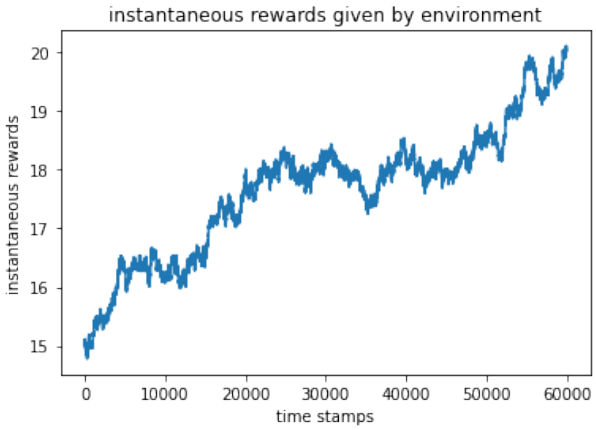

Fig3. : Ten Armed Bandit

We can see that initially, all rewards are equal to 0, but as the number of iterations increases, the rewarding policy of each action is affected, and the expected rewards increase at a rapid, non-zero rate.

2. Develop a 10-armed bandit in which all ten mean-rewards start out equal and then take independent random walks (by adding a normally distributed increment with mean zero and standard deviation 0.01 to all mean-rewards on each time step). {*function [value] = bandit_nonstat(action)*}

Ans. - The 10-armed bandit is represented by the Bandit class, which has a reward method that provides a reward for a given arm index and an incrementReward method that adds a normally distributed increment with mean zero and standard deviation 0.01 to update the mean rewards of each arm. A normal distribution with a mean equal to the arm's current mean reward and a standard deviation of one is used to determine the reward.

For a ten-arm bandit, the mean-rewards are initially set to a fixed array with a value of 1. The Epsilon Greedy Algorithm is used for exploration and exploitation. To decide whether to exploit or explore, a random number between 0 and 1 is generated, and if it exceeds epsilon, exploitation is carried out based on prior knowledge. At each iteration, a ten-value array is produced, which is normally distributed with a mean of zero and a standard deviation of 0.01. This array is added to the mean array, and the updated array is used for subsequent actions, which yield rewards based on the updated mean-rewards array. The rewards in this scenario are non-stationary.

3. The 10-armed bandit that you developed (bandit_nonstat) is difficult to crack with a standard epsilon-greedy algorithm since the rewards are non-stationary. We did discuss how to track non-stationary rewards in class. Write a modified epsilon-greedy agent and show whether it is able to latch onto correct actions or not. (Try at least 10000 time steps before commenting on results)

Ans. - In contrast to Problem 2, this problem involves updating the action reward estimation by assigning greater significance to the current earned reward through utilization of the alpha parameter, as opposed to the average method used in the former.

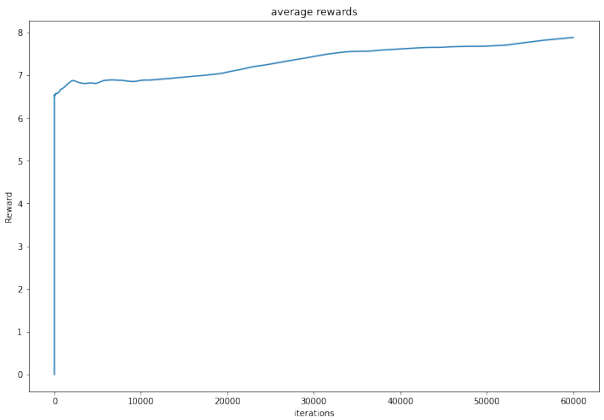

Fig.4 : Expected rewards we get when rewards are non stationary (averaging method)

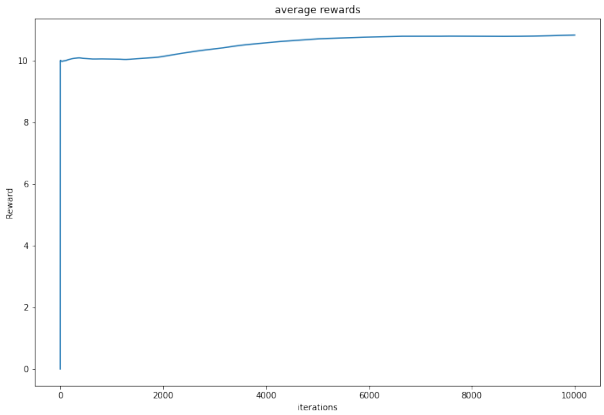

Fig.5 : Expected rewards we get when rewards are non stationary (non averaging method)

The graph's incline in question 3 was much steeper than that of question 1, and the expected rewards were significantly greater than question 2.

II. References

1) Reinforcement Learning: an introduction by R Sutton and A Barto (Second Edition) (Chapter 1-2)