Exercise 2.1: In epsilon-greedy action selection, for the case of two actions and epsilon = 0.5, what is the probability that the greedy action is selected?

Solution

p(greedy) = p(greedy|exploitation)*p(exploitation) + p(greedy|exploration)*p(exploration)
= 1*0.5 + 0.5*0.5
= 0.75

Exercise 2.2: Bandit example. Consider a k-armed bandit problem with k = 4 actions, denoted 1, 2, 3, and 4. Consider applying to this problem a bandit algorithm using epsilon-greedy action selection, sample-average action value estimates, and initial estimates of $Q_1(a) = 0$, for all a. Suppose the initial sequence of actions and rewards is $A_1 = 1$, $R_1 = -1$, $A_2 = 2$, $R_2 = 1$, $A_3 = 2$, $R_3 = -2$, $A_4 = 2$, $R_4 = 2$, $A_5 = 3$, $R_5 = 0$. On some of these time steps the epsilon case may have occurred, causing an action to be selected at random. On which time steps did this definitely occur? On which time steps could this possibly have occurred?

Solution

At time step 1, the quality of each action is:

$Q_1(1) = 0$
$Q_1(2) = 0$
$Q_1(3) = 0$
$Q_1(4) = 0$

the action and reward was:

$A_1 = 1$, $R_1 = -1$

This same decision, $A_1 = 1$ can be reached either through exploration or exploitation. Therefore time step 1 decision could possibly be an epsilon case.

At time step 2, the quality of each action is:

$Q_2(1) = -1$
$Q_2(2) = 0$
$Q_2(3) = 0$
$Q_2(4) = 0$

the action and reward is:

$A_2 = 2$, $R_2 = 1$

This same decision, $A_2 = 2$ can be reached either through exploration or exploitation. Therefore time step 2 decision could possibly be an epsilon case.

At time step 3, the quality of each action is:

$Q_3(1) = -1$
$Q_3(2) = 1$
$Q_3(3) = 0$
$Q_3(4) = 0$

the action and reward is:

$A_3 = 2$, $R_3 = -2$

This same decision, $A_3 = 2$ can be reached either through exploration or exploitation. Therefore time step 3 decision could possibly be an epsilon case.

At time step 4, the quality of each action is:

$Q_4(1) = -1$
$Q_4(2) = -0.5$
$Q_4(3) = 0$
$Q_4(4) = 0$

the action and reward is:

$A_4 = 2$, $R_4 = 2$

This same decision, $A_4 = 2$ can only be reached through exploration. According to the estimated value of each action, exploitation should select one of actions 3 or 4 at random uniformly. Therefore time step 4 decision is definitely an epsilon case.

At time step 5, the value of each action is:

$Q_5(1) = -1$
$Q_5(2) = 1/3$
$Q_5(3) = 0$
$Q_5(4) = 0$

the action and reward is:

$A_5 = 3$, $R_5 = 0$

This same decision, $A_5 = 3$ can only be reached through exploration. According to the estimated value of each action, exploitation should select action 2. Therefore the time step 5 decision is definitely an epsilon case.

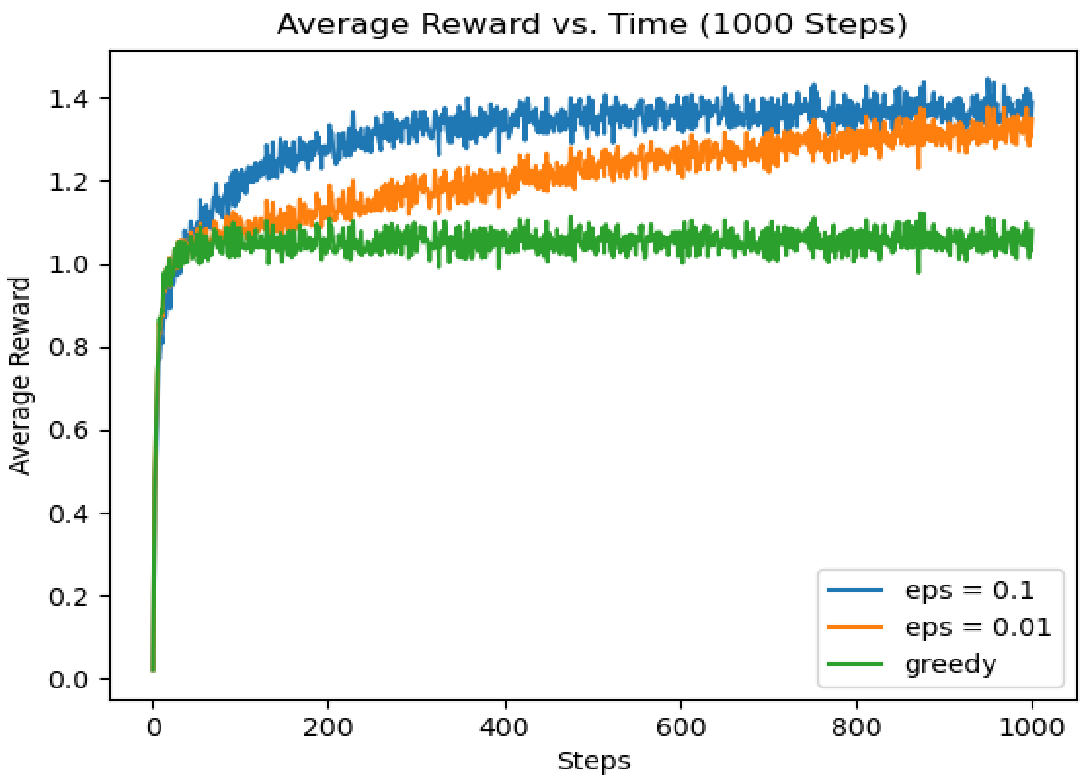
Exercise 2.3 In the comparison shown in Figure 2.2, which method will perform best in the long run in terms of cumulative reward and probability of selecting the best action? How much better will it be? Express your answer quantitatively.

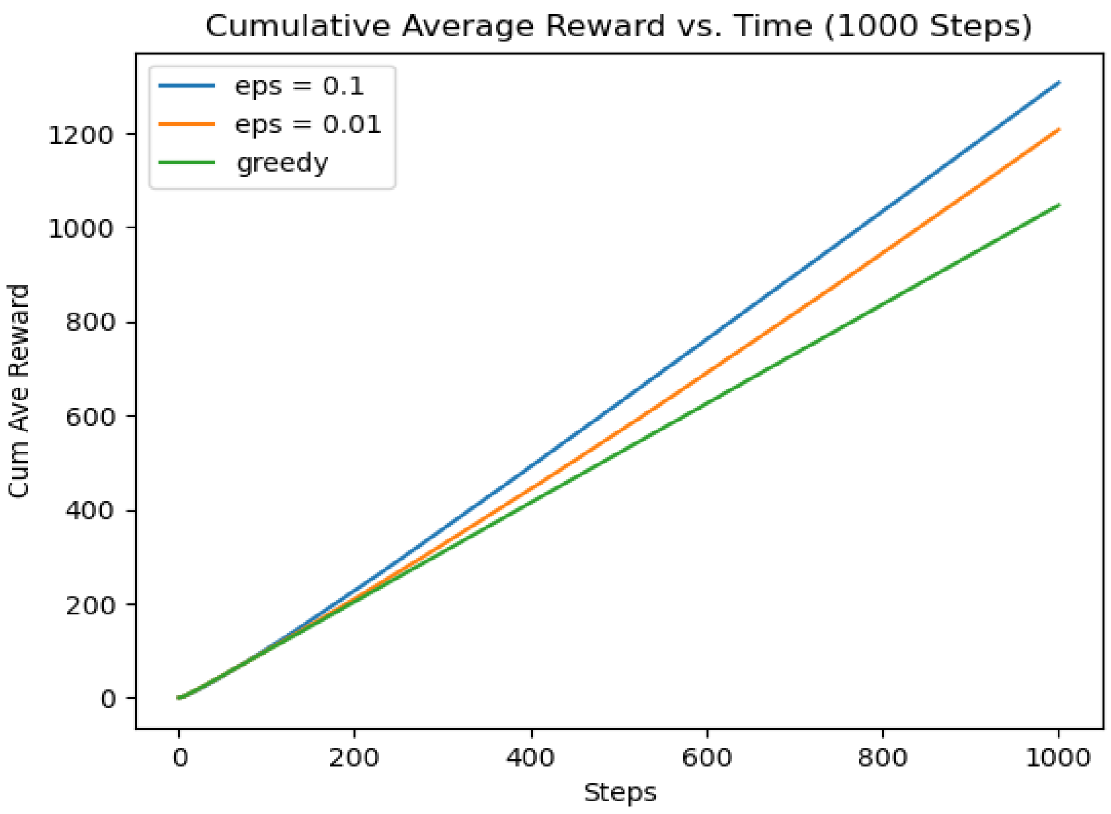
Solution

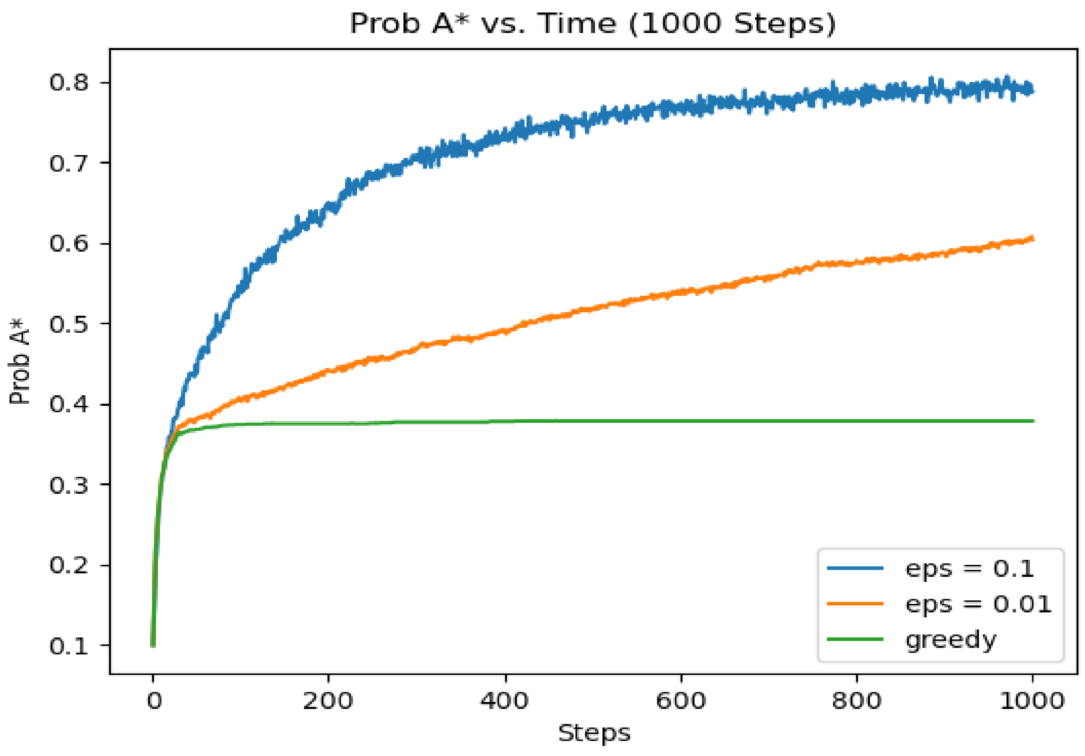
Epsilon-greedy method with epsilon = 0.01 will perform best in the long run in terms of average reward, cumulative reward, and probability of selecting the best action.

To illustrate this visually I ran the testbed with time steps of 1000, 10000, and 20000, and graphed the average reward, cumulative average reward and probability of selecting the best action against time steps.

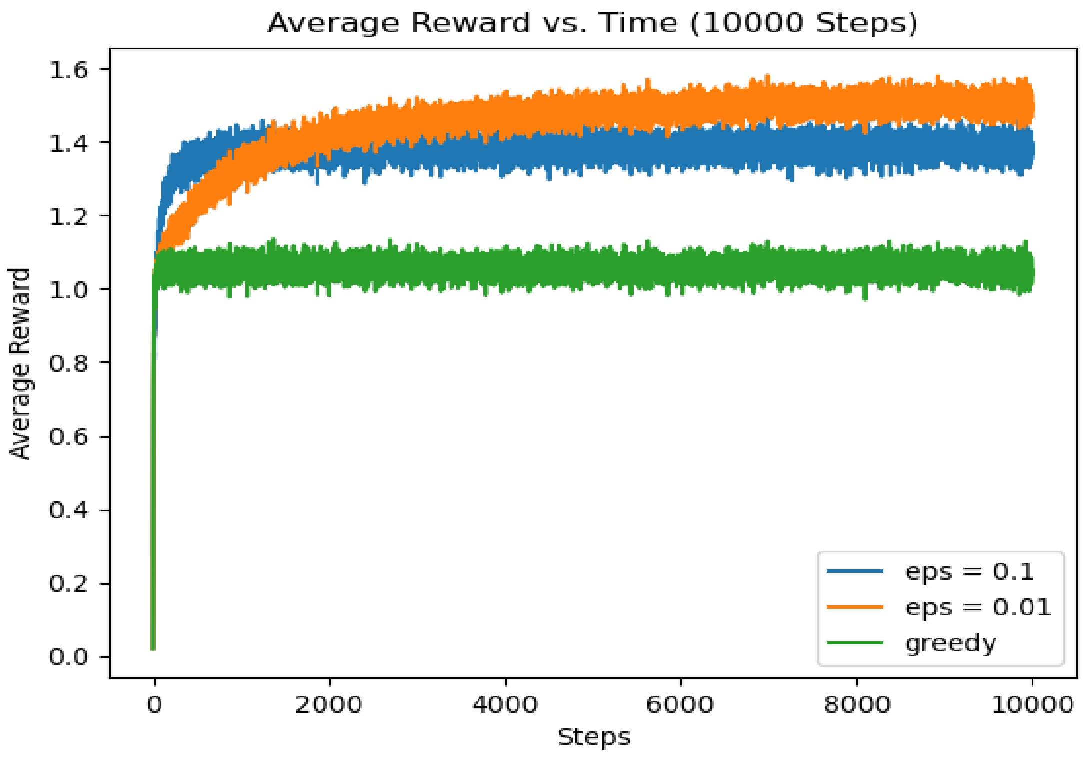
Time step 1000

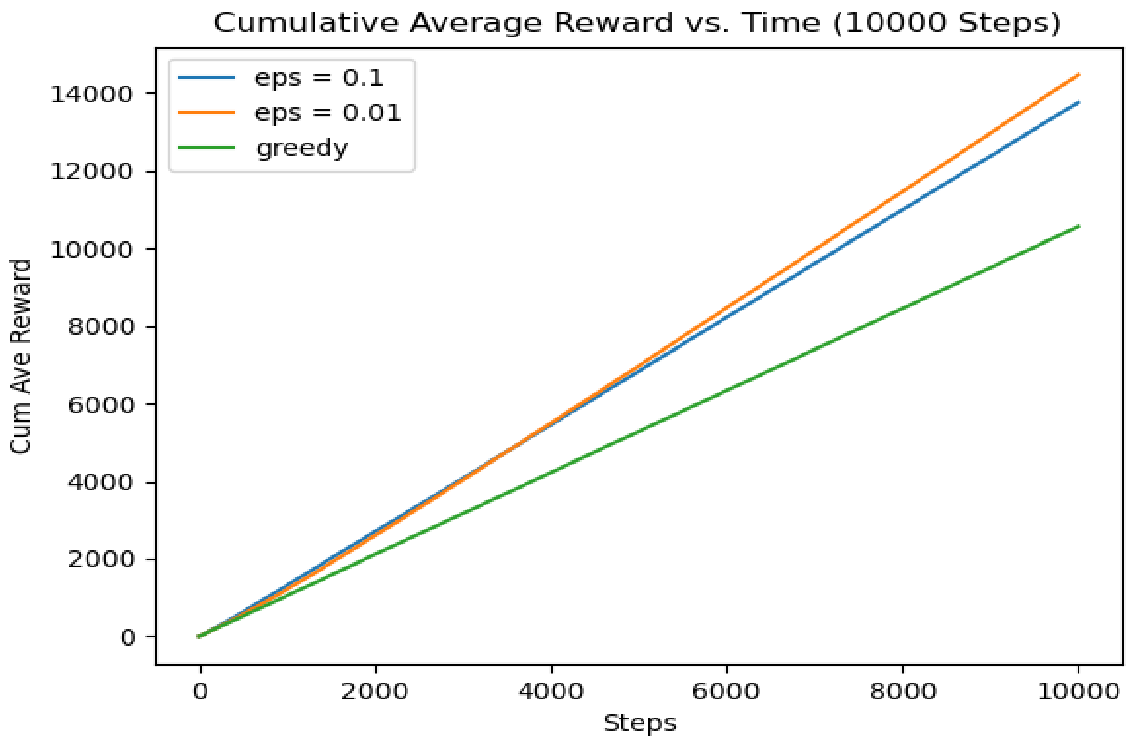


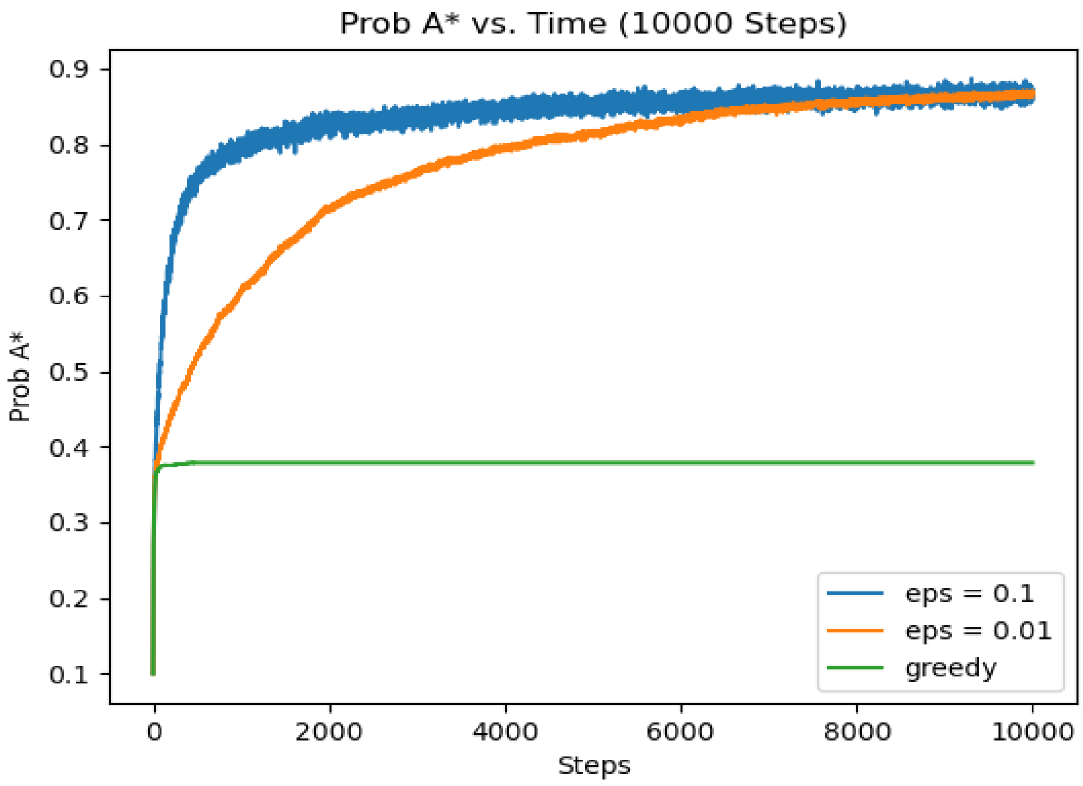

Cumulative Average Reward vs. Time (1000 Steps)
eps = 0.1
eps = 0.01
greedy
Cum Ave Reward
0
200
400
600
800
1000
1200
Steps


Prob A* vs. Time (1000 Steps)
eps = 0.1
eps = 0.01
greedy
Prob A*
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
200
400
600
800
1000
Steps

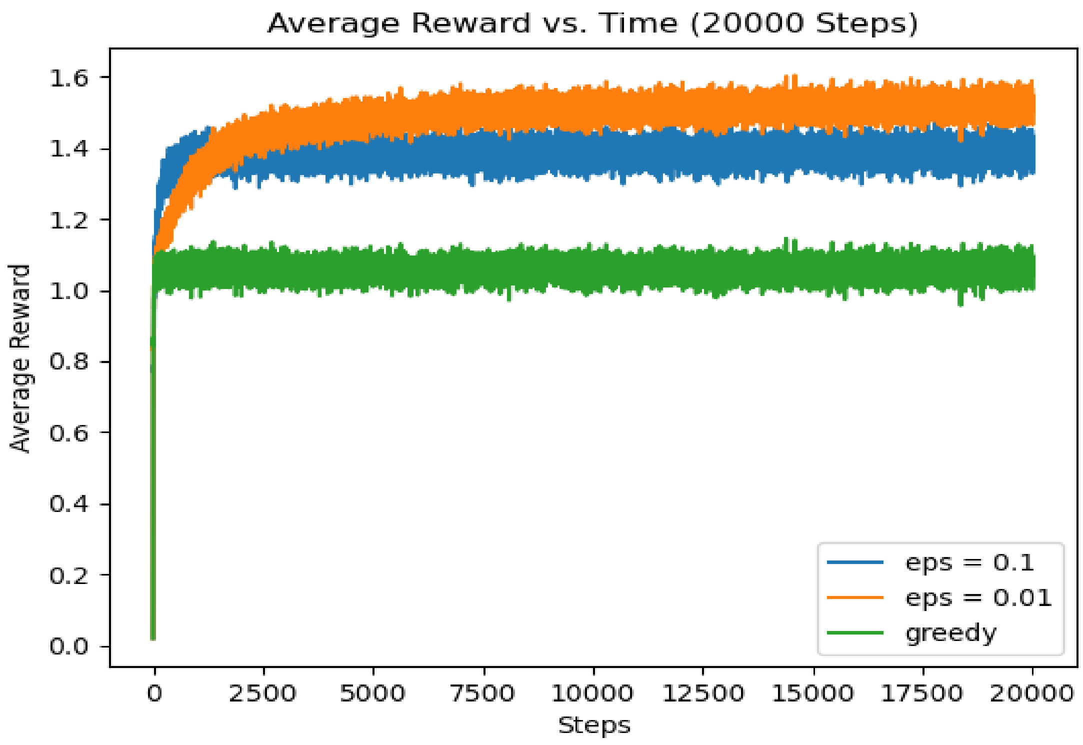
Time step 10000

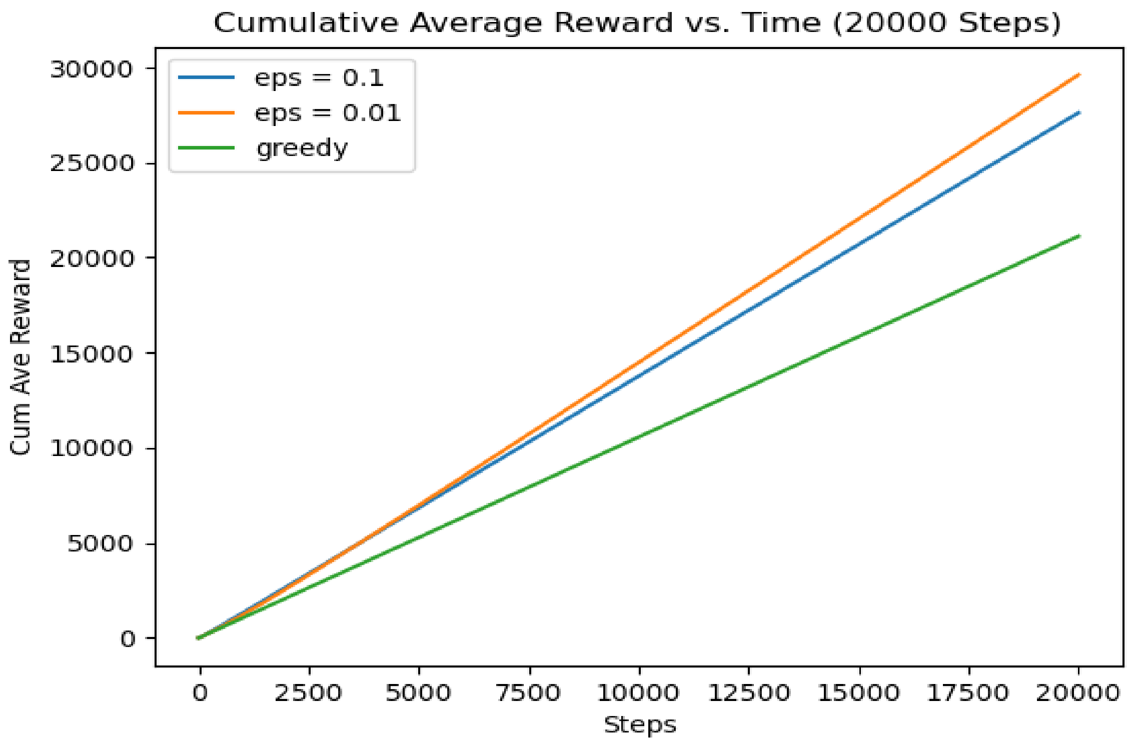


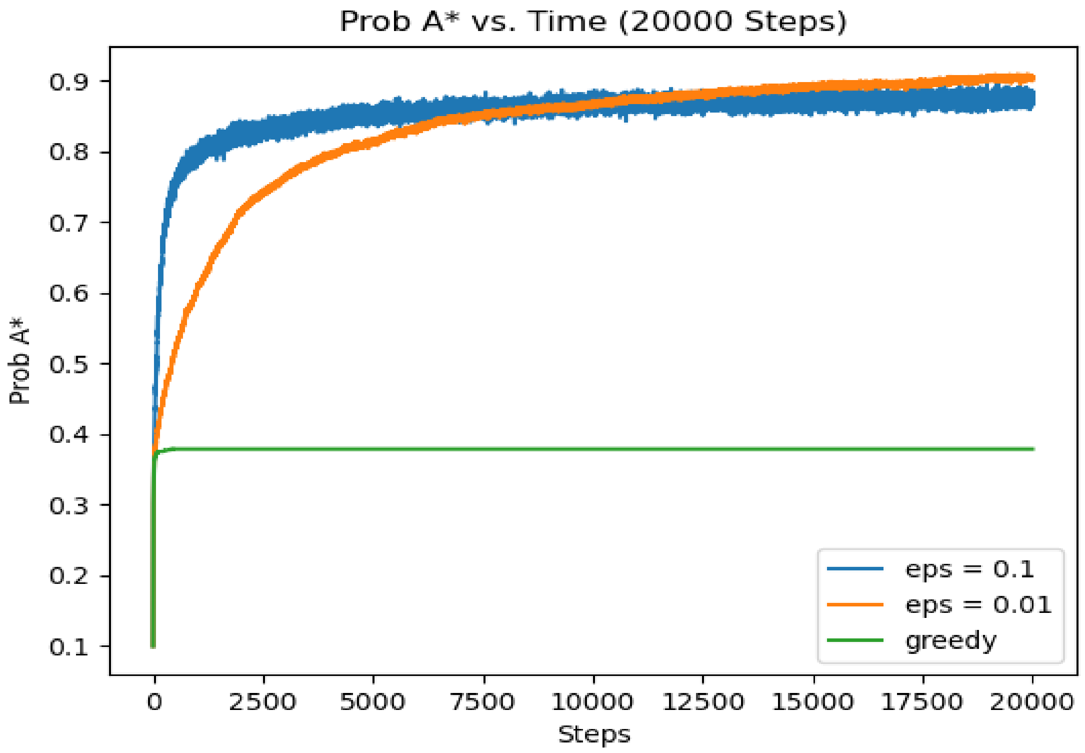




Time Step 20,000




Cumulative Average Reward vs. Time (20000 Steps)
Cum Ave Reward
Steps
eps = 0.1
eps = 0.01
greedy
0
5000
10000
15000
20000
25000
30000
0
2500
5000
7500
10000
12500
15000
17500
20000


Prob A* vs. Time (20000 Steps)
Prob A*
Steps
eps = 0.1
eps = 0.01
greedy
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
2500
5000
7500
10000
12500
15000
17500
20000

Quantitatively I present the cumulative average reward achieved by each of the three methods after 1000, 10000, and 20000 time steps in the table below.

	Time Step 1000	**Time Step 10000**	**Time Step 20000**
Epsilon-greedy 0.1	1307.74	13757.99	27622.69
Epsilon-greedy 0.01	1207.84	14470.99	29614.17
Greedy	1046.99	10558.99	21130.02

At 1000 time steps, the epsilon-greedy 0.1 method leads with epsilon-greedy 0.01 method following closely behind and greedy method weighing in last. By 10000 time steps, the epsilon-greedy 0.01 method has surpassed epsilon-greedy 0.1 to take the lead, with epsilon-greedy 0.1 dropping to second and greedy method last. At 20000 time steps, epsilon-greedy 0.01 maintains its lead, with epsilon-greedy 0.1 still second and greedy method last. So quantitatively as shown in the table above, epsilon-greedy 0.01 method is the best performer in terms of cumulative average reward.

Again quantitatively i present the probability of finding the optimal action achieved by each of the three methods after 1000, 10000, and 20000 time steps in the table below.

	Time Step 1000	**Time Step 10000**	**Time Step 20000**
Epsilon-greedy 0.1	0.788	0.865	0.871
Epsilon-greedy 0.01	0.6035	0.863	0.902
Greedy	0.3785	0.3785	0.3785

At 1000 time steps, the epsilon-greedy 0.1 method has the highest probability of finding the optimal action. However, by 20000 time steps, the epsilon-greedy 0.01 method has surpassed all other methods to take the leading spot with a very high probability (0.902) of finding the optimal action.

Exercise 2.4. If the step-size parameters, α_n, are not constant, then the estimate Q_n is a weighted average of previously received rewards with a weighting different from that given by (2.6). What is the weighting on each prior reward for the general case, analogous to (2.6), in terms of the sequence of step-size parameters?

Solution

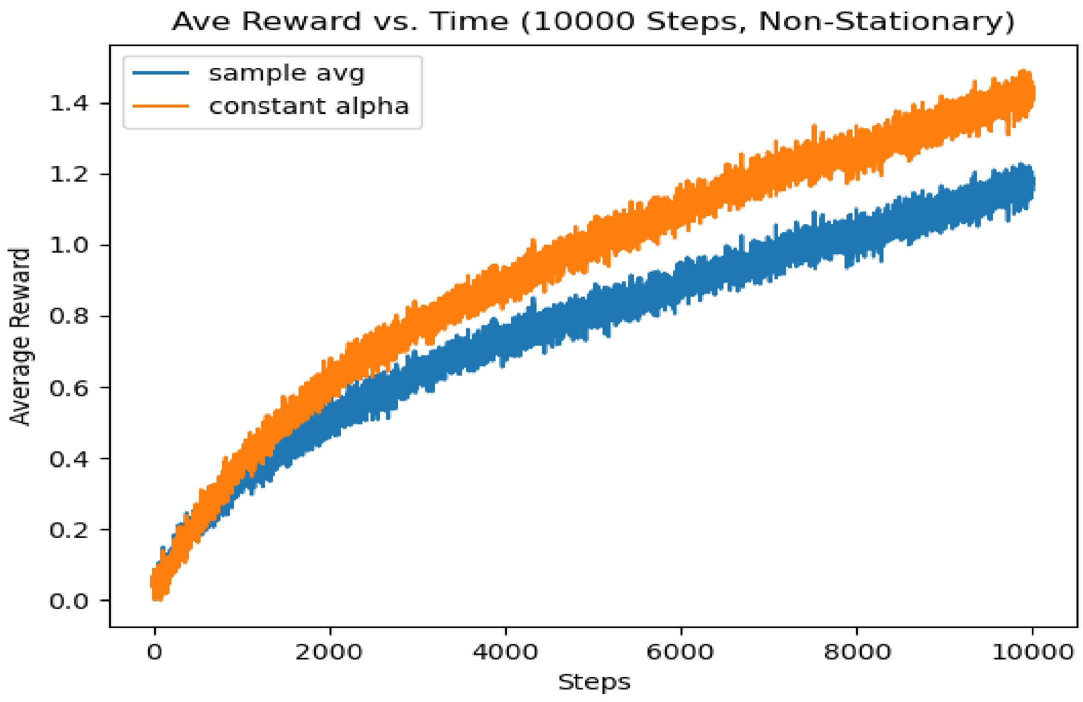
$$Q_{n+1} = (\prod_{j=1}^{n}(1-\alpha_j))Q_1 + \sum_{i=1}^{n}(\prod_{j=i+1}^{n}(1-\alpha_j))\alpha_i R_i$$

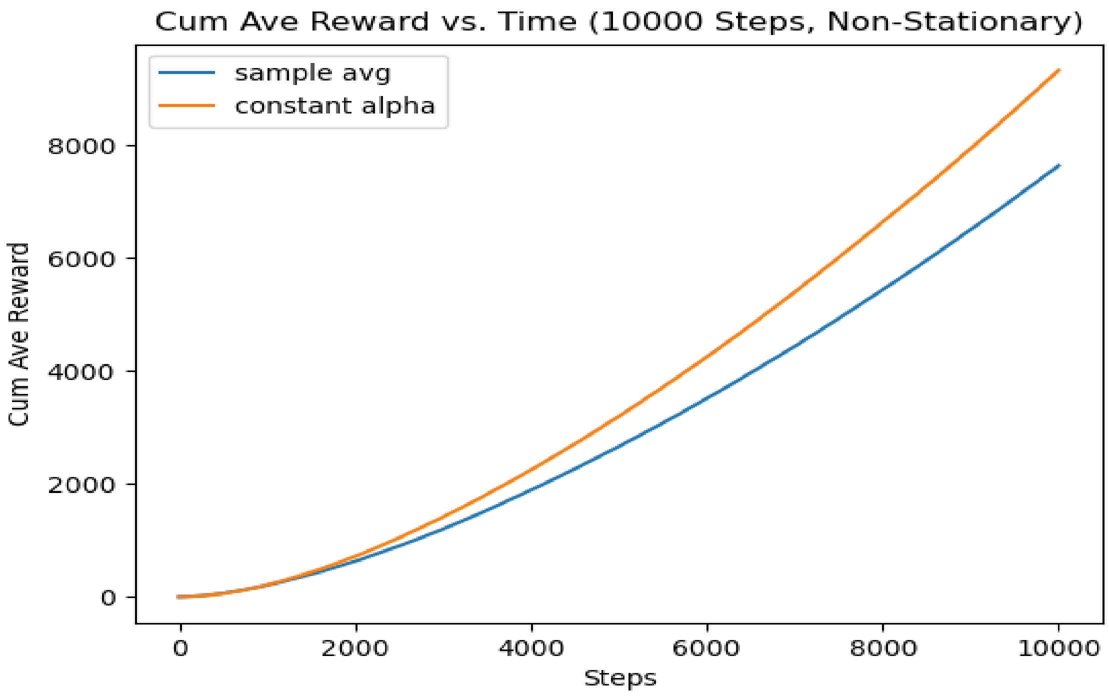
Exercise 2.5 (programming). Design and conduct an experiment to demonstrate the difficulties that sample-average methods have for nonstationary problems. Use a modified version of the 10-armed testbed in which all the $q_*(a)$ start out equal and then take independent random walks (say by adding a normally distributed increment with mean zero and standard deviation 0.01 to all the $q_*(a)$ on each step). Prepare plots like Figure 2.2 for an action-value method using sample averages, incrementally computed, and another action-value method using a constant step-size parameter, α = 0.1. Use ε = 0.1 and longer runs, say of 10,000 steps.

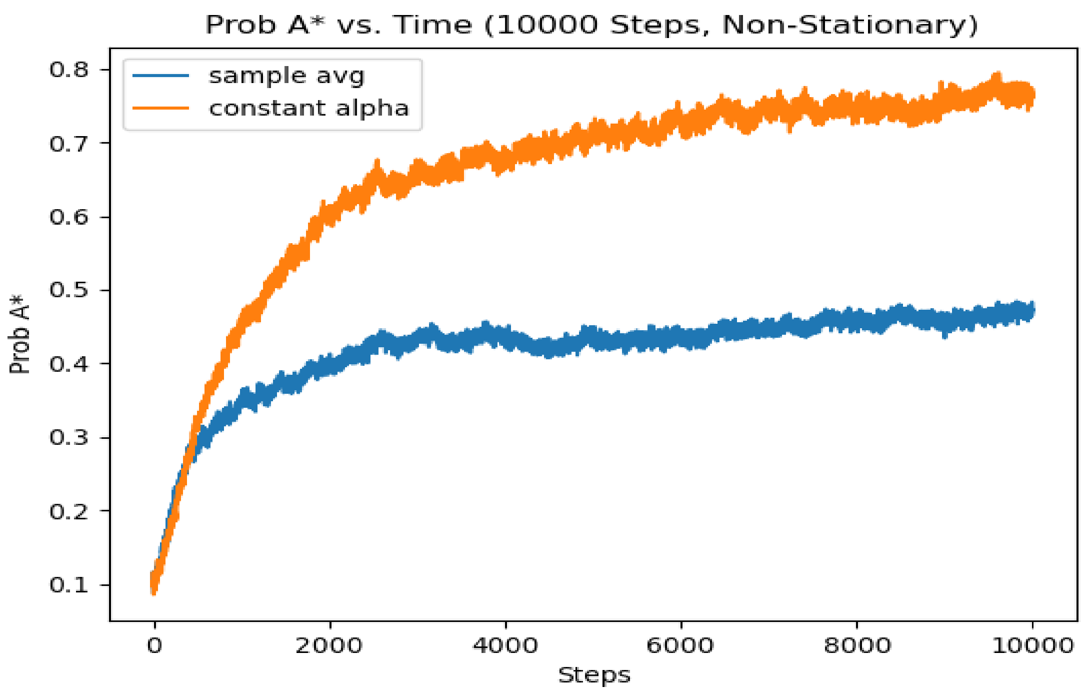
Solution

As per requirement, I designed a variant of the testbed experiment in which the rewards distribution are different from one time step to another (non-stationary) using the distributions recommended in the problem statement. Also the experiment compares the sample average method versus the constant alpha method.

Plots of average reward, cumulative average reward, and probability of finding the optimal action for the experiment are given below.


Ave Reward vs. Time (10000 Steps, Non-Stationary)
sample avg
constant alpha
Average Reward
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
2000
4000
6000
8000
10000
Steps


Cum Ave Reward vs. Time (10000 Steps, Non-Stationary)
sample avg
constant alpha
Cum Ave Reward
0
2000
4000
6000
8000
0
2000
4000
6000
8000
10000
Steps



The plots above clearly demonstrate the fact that the sample average method has problems for a non-stationary reinforcement learning problem when compared to some other learning method like for example the constant alpha method that was used in this experiment.

Quantitatively, after averaging 2000 different 10-armed bandit tasks with a time step of 10000 each, we get the following figures.

	Average reward	**Cumulative average reward**	**Probability of optimal action**
Sample average	1.17	7635.20	0.47
Constant alpha	1.43	9325.90	0.76

It is very clear from looking at the table above that the sample average method has difficulties coping with the non-stationarity of the rewards vis-a-vis the method that uses a constant alpha which clearly outperforms it according to my experiment..

The constant alpha method achieves an average reward of 1.43 by the 10000th time step, compared to the 1.17 achieved by the sample average method. Also the constant alpha method achieves a higher cumulative average reward of 9325.90 versus 7635.20 for sample average method and a higher probability of finding the optimal action of 0.76 versus 0.47 for sample average method.

Exercise 2.6: Mysterious Spikes. The results shown in Figure 2.3 should be quite reliable because they are averages over 2000 individual, randomly chosen 10-armed bandit tasks. Why, then, are there oscillations and spikes in the early part of the curve for the optimistic method? In other words, what might make this method perform particularly better or worse, on average, on particular early steps?

Solution

For a k-armed bandit problem, being solved using the optimistic initial values method, the spike occurs on the k+1 time step. This is because the method (which encourages exploration) would have explored each of the k different actions of the k-armed bandit problem at least once, such that on the k+1 time step in a greedy method there is a probability of finding the optimal action of 40% all of a sudden, on the k+1 time step, while there was a probability less than 1% of finding the optimal action in the first k time steps. Hence the spike noticed on the graph.

$Q_{n+1} = Q_n + \alpha (R_n - Q_n)$

Exercise 2.7: Unbiased Constant-Step-Size Trick. In most of this chapter we have used sample averages to estimate action values because sample averages do not produce the initial bias that constant step sizes do (see the analysis leading to (2.6)). However, sample averages are not a completely satisfactory solution because they may perform poorly on nonstationary problems. Is it possible to avoid the bias of constant step sizes while retaining their advantages on nonstationary problems? One way is to use a step size of

$\text{Beta}_n = \text{alpha} / o_n,$ (2.8)

To process the nth reward for a particular action, where alpha > 0 is a conventional constant step size, and on is a trace of one that starts at 0:

$O_n = o_{n-1} + \text{alpha}(1 - o_{n-1}),$ for n >= 0, with $o_0 = 0$. (2.9)

Carry out an analysis like that in (2.6) to show that Q_n is an exponential recency-weighted average without initial bias.

Solution

Todo

Exercise 2.8: UCB Spikes. In Figure 2.4 the UCB algorithm shows a distinct spike in performance on the 11th step. Why is this? Note that for your answer to be fully satisfactory it must explain both why the reward increases on the 11th step and why it decreases on the subsequent steps. Hint: if c = 1, then the spike is less prominent.

Solution

Time Step	**Average Reward**	**Cumulative Average Reward**	**Probability of finding the Optimal Action**
1	0.0609	0.0609	0.11
2	-0.0072	0.0537	0.09
3	-0.0586	-0.0048	0.08
4	-0.0253	-0.0302	0.11
5	0.0164	-0.0138	0.10
6	0.0070	-0.0068	0.09
7	0.0274	0.0206	0.11
8	0.0158	0.0364	0.11
9	0.0003	0.0367	0.10
10	0.0423	0.0791	0.10
11	1.1526	1.2317	0.45
12	0.9429	2.1747	0.33
13	0.8797	3.0544	0.32
14	0.8273	3.8817	0.31
15	0.7926	4.6742	0.31

Between steps 1 and 10, the algorithm selects randomly amongst actions, selecting each action exactly once. There is exactly a 0.1 probability of finding the optimal action. This results in low average reward betwen steps 1 and 10.

The ucb selection kicks in at step 11 exactly and bumps up the probability of finding the optimal action in direct proportionality with c, the confidence term. Higher values of c should increase the probability of finding the optimal action at step 11, while lower values of c should decrease the probability of finding the optimal action at step 11.

When the probability of finding the optimal action at step 11 is high (high c), then more tasks find the optimal action at step 11, hence a high spike in average reward at step 11. When the probability of finding the optimal action at step 11 is low (low c), then less tasks find the optimal action at step 11, hence a lower spike in average reward at step 11.

As per ucb mechanics, for those tasks that find the optimal action at step 11, each time the action is selected, its uncertainty term decreases while the uncertainty term increases for other actions in the same time step such that some other action soon becomes the ucb optimal action although it is not in fact the optimal action. Thus average reward drops again.

Exercise 2.9. Show that in the case of two actions, the soft-max distribution is the same as that given by the logistic, or sigmoid, function often used in statistics and artificial neural networks.

Solution

According to wikipedia (https://en.wikipedia.org/wiki/Softmax_function):

The standard logistic function is the special case for a 1-dimensional axis in 2-dimensional space, say the x-axis in the (x, y) plane. One variable is fixed at 0 (say $z_2 = 0$), so $e^0 = 1$, and the other variable can vary, denote it $z_1 = x$, so

$$\frac{e^{z_1}}{\sum_{k=1}^{2} e^{z_k}} = \frac{e^x}{(e^x + 1)}$$

the standard logistic function, and

$$\frac{e^{z_2}}{\sum_{k=1}^{2} e^{z_k}} = \frac{1}{(e^x + 1)}$$

its complement (meaning they add up to 1).

Exercise 2.10. Suppose you face a 2-armed bandit task whose true action values change randomly from time step to time step. Specifically, suppose that, for any time step, the true values of actions 1 and 2 are respectively 0.1 and 0.2 with probability 0.5 (case A), and 0.9 and 0.8 with probability 0.5 (case B). if you are not able to tell which case you face at any step, what is the best expectation of success you can achieve and how should you behave to achieve it?

Now suppose that on each step you are told whether you are facing case A or case B (although you still don't know the true action values). This is an associative search task. What is the best expectation of success you can achieve in this task, and how should you behave to achieve it?

Solution

We derive a model of sorts from the law of total expectation,

E[success] = P(A) * E[success A] + P(B) * E[success B]
E[success A] = P(0.1) * E[0.1] + P(0.2) * E[0.2]
E[success B] = P(0.9) * E[0.9] + P(0.8) * E[0.8]

If we are not able to tell which case we face at any step, we select either value of action 1 and 2 randomly uniformly with equal probability of

P(0.1) = 0.5, P(0.2) = 0.5
P(0.9) = 0.9, P(0.8) = 0.8

Therefore

E[success A] = 0.5 * 0.1 + 0.5 * 0.2 = 0.15
E[success B] = 0.5 * 0.9 + 0.5 * 0.8 = 0.85
E[success] = 0.5 * 0.15 + 0.5 * 0.85 = 0.50

Therefore the best expectation of success if we are not able to tell which case we face at any step is 0.50.

If on the other hand we are told whether we are facing case A or case B at any step, we can learn to select the best action in each case as follows

P(0.1) = 0.0, P(0.2) = 1.0
P(0.9) = 1.0, P(0.8) = 0.0

Therefore

E[success A] = 0.0 * 0.1 + 1.0 * 0.2 = 0.2
E[success B] = 1.0 * 0.9 + 0.0 * 0.8 = 0.9
E[success] = 0.5 * 0.2 + 0.5 * 0.9 = 0.55

Therefore the best expectation of success if we are told which case (A or B) we face at any step is 0.55.

Exercise 2.11 (programming). Make a figure analogous to Figure 2.6 for the nonstationary case outlined in Exercise 2.5. Include the constant-step-size epsilon-greedy algorithm with alpha=0.1.

Use runs of 200,000 steps and, and as a performance measure for each algorithm and parameter setting, use the average reward over the last 100,000 steps.

Solution

Todo