Chapter 2

Exercise 2.1

Let the two actions be A and B. Assume $Q(A) > Q(B)$. The greedy Action is therefore action A.

$\varepsilon = 0.5$
$N = 2$ (Since there are only 2 actions)

$\pi(A) = (1 - \varepsilon) + \varepsilon * 1/N(a) = 0.5 + 0.5 * 0.5 = 0.75$
$\pi(B) = \varepsilon * 1/N(a) = 0.5 * 0.5 = 0.25$

Exercise 2.2:

Initial sequence of actions and rewards is $A_1 = 1$, $R_1 = 1$, $A_2 = 2$, $R_2 = 1$, $A_3 = 2$, $R_3 = -2$, $A_4 = 2$, $R_4 = 2$, $A_5 = 3$, $R_5 = 0$.

At $t = 2$, $Q_1 = 1$ and all other values are still zero. The greedy choice would then be $A = 1$. However, A_2=2 therefore at $t = 2$, exploration definitely took place. Similar analysis shows that exploration also definitely occurred at $t = 5$.

Other than that, if exploration means that we select uniformly from all actions, it could have occured at any time step.

Exercise 2.3:

The method with $\varepsilon = 0.01$ will perform the best in the long run. The greedy method will not explore and hence will be stuck at a suboptimal choice, while the $\varepsilon - greedy$ with $\varepsilon = 0.1$ will rapidly explore and find the optimal action but in the long run it will keep exploring at a high rate leading to lower cumulative rewards than the method with $\varepsilon = 0.01$.

Let us consider the situation after sufficient time has passed.
Let us assume that any suboptimal choice gives average value x while the optimal average value is y . Obviously, $x > y$.

The $\varepsilon - greedy$ method with $\varepsilon = 0.1$ will explore 100 out of 1000 times and therefore choose suboptimal values 100 times out of 1000.

Total reward in 1000 time steps is approximately $100 * x + 900 * y$

While the $\varepsilon - greedy$ method with $\varepsilon = 0.01$ will explore only 10 times out of 1000 and therefore will choose suboptimal values only 10 times.

Total reward in 1000 time steps is approximately $10 * x + 990 * y$

Given that $x > y$, the reward with $\varepsilon = 0.01$ is higher.

Exercise 2.4

If α is not constant, then we need to denote it by α_t .

$$Q_{n+1} = Q_n + \alpha_n * [R_n - Q_n]$$
$$Q_{n+1} = \alpha_n * R_n + [1 - \alpha_n] * Q_n$$
$$Q_{n+1} = \alpha_n * R_n + (1 - \alpha_n) * [\alpha_{n-1} * R_{n-1} + (1 - \alpha_{n-1}) * Q_{n-1}]$$

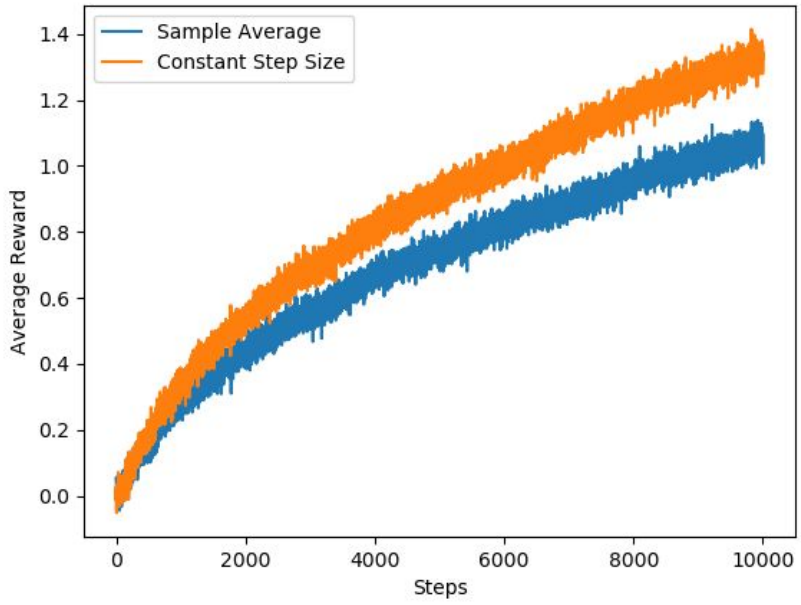
Therefore,

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

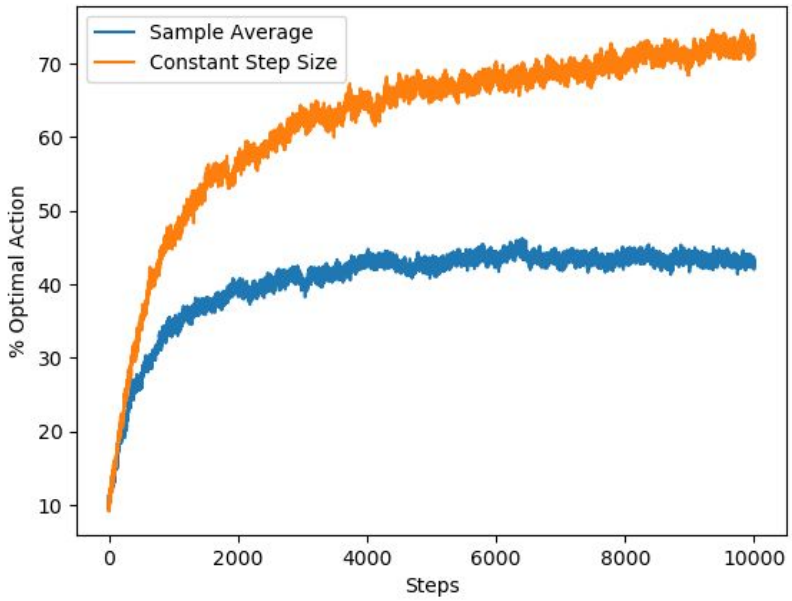
Exercise 2.5 (Programming):

Code: Check Github

Results:



Steps vs Average Reward



Steps vs % Optimal Action

Exercise 2.6:

With optimistic initial values, the agent initially chooses some action and gets low reward and decreases its value. It then selects another action (since all other actions are guaranteed to have higher value at this stage because of optimistic initialization) which is different from the one chosen. This continues for 10 steps. After 10 steps the agent has tried each action exactly once. At the 11th step the agent will choose the action with the highest value. At this point, there is a very high probability that the value of the optimal action is also the highest (because it would have produced a higher reward than other actions and hence its decrement must have been lower). Thus almost all the agents tend to go with the optimal action on step number 11. This is the reason for the spike in the graph. The agent then selects the optimal action till

its value gets lower than some other action. That other action is then chosen in future time steps until it's estimated value is the highest. This results in a dip after the spike in the graph. This process continues for a while till the effect of optimistic initialisation dies off. This means that optimistic initialisation encourages exploration in the beginning.

Exercise 2.7:

$$step\ size = \beta = \alpha/O_n$$
$$O_n = O_{n-1} + \alpha * (1 - O_{n-1})$$

Therefore,

$$Q_{n+1} = Q_n + \alpha/O_n * (R_n - Q_n)$$
$$Q_{n+1} = \alpha/O_n * R_n + (1 - \alpha/O_n) * Q_n$$
$$Q_{n+1} = \alpha/O_n * R_n + (1 - \alpha/O_n) * [\alpha/O_{n-1} * R_{n-1} + (1 - \alpha/O_{n-1}) * Q_{n-1}]$$

$$1 - \alpha/O_n = (O_n - \alpha)/O_n = (1 - \alpha) * O_{n-1}/O_n$$

$$Q_{n+1} = R_n * \alpha/O_n + \sum_{i=1}^{n-1} [R_i * \alpha/O_i * \prod_{j=i+1}^{n} (1 - \alpha) * O_{j-1}/O_j] + Q_1 * \prod_{i=1}^{n} (1 - \alpha) * O_{i-1}/O_i$$

$$\prod_{j=i+1}^{n} (1 - \alpha) * O_{j-1}/O_j = (1 - \alpha)^{n-i} * O_i/O_n$$
$$\prod_{j=1}^{n} (1 - \alpha) * O_{j-1}/O_j = (1 - \alpha)^{n-i} * O_0/O_n = 0 \ (\text{since } O_0 = 0)$$

$$Q_{n+1} = R_n * \alpha/O_n + 1/O_n * \sum_{i=1}^{n-1} [R_i * \alpha * (1-\alpha)^{n-i}]$$

$$Q_{n+1} = 1/O_n * \sum_{i=1}^{n} [R_i * \alpha * (1-\alpha)^{n-i}]$$

Which has the exact same format as Eq 2.6 and Q_{n+1} does not depend on Q_1 therefore Q_{n+1} is an exponential recency-weighted average without initial bias.

Exercise 2.8:

The answer to this is quite similar to the answer to Exercise 2.6. In the initial 10 steps the agent will try out all different actions. However, in this case it is not because of how Q values are initialised but because of how the action is chosen. At the 11th step all the actions have been chosen once and thus the value of $c * \sqrt{ln(t)/N_t(a)}$ is the same for all actions. Thus the action selection at 11th step depends on the Q value estimates of the actions. The estimate is expected to be higher for the optimal action and therefore almost all agents (across different runs) select the optimal action at step 11 leading to a spike in the graph. After the 11th step this behaviour may not be seen as different agents select different actions and thus there is a dip.

Exercise 2.9:

Sigmoid function is basically a special case of Soft-Max function.

Let the two actions be numbered by 0 and 1.

$$P(A = k|X) = e^{H(k)*X}/(e^{H(0)*X} + e^{H(1)*X})$$
$$P(A = k|X) = 1/(e^{(H(0)-H(k))*X} + e^{(H(1)-H(k))*X})$$
$$P(A = 1|X) = 1/(1 + e^{-(H(0)-H(1))*X}) = 1/(1 + e^{-H^1*X})$$
$$P(A = 0|X) = 1 - P(A = 1|X)$$

Which is the same as the Sigmoid Function.

Exercise 2.10:

If we do not know which case we are encountering at every time step but we know what the two cases are and what are their actual probabilities of occurring then we should pick the action whose weighted value is the highest among all actions. In the exercise we are given the probabilities of both cases occurring which we can use as weights to find which action is most suitable. On the other hand if we are given which case we are encountering at every time step, we can simply maintain 2 different tables for Q values and pick the action looking at the appropriate table. This is just like solving 2 completely independent Bandit problems.

Exercise 2.11:

My system is not powerful enough to complete this exercise in a reasonable amount of time. I have however replicated Figure 2.6 whose code needs only minor modifications to run for this exercise.