RBE 595 — Reinforcement Learning
Week #2 Assignment

Arjan Gupta

Problem 1

What is the benefit of incremental update for action-value function, versus non-incremental?

Answer

The non-incremental implementation of the action-value function requires that we store all of the rewards that have been seen so far in each time-step. The formula for this would be,

$$Q_n = \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}$$

Where Q_n is the current estimation of the reward R after $n-1$ time-steps. The problem with this approach is that, as the number of time-steps increases, the amount of memory required to store all of the rewards increases linearly, i.e. the space complexity is $\mathcal{O}(n)$.

Instead, we use the incremental update approach, which uses $\mathcal{O}(1)$ space complexity (constant memory). The formula for this can be derived as follows,

$$\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}(R_n + \sum_{i=1}^{n-1} R_i) \\
&= \frac{1}{n}(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i) \\
&= \frac{1}{n}(R_n + (n-1)Q_n) \\
&= \frac{1}{n}(R_n + nQ_n - Q_n) \\
&= Q_n + \frac{1}{n}(R_n - Q_n)
\end{aligned}$$

Computationally, this incremental approach is better as well, because it only requires one addition, one subtraction, and one division per time-step. The non-incremental approach requires $n-1$ additions, 1 subtraction, and one division per time-step. Therefore, the incremental approach is $\mathcal{O}(1)$ in terms of time complexity, while the non-incremental approach is $\mathcal{O}(n)$.

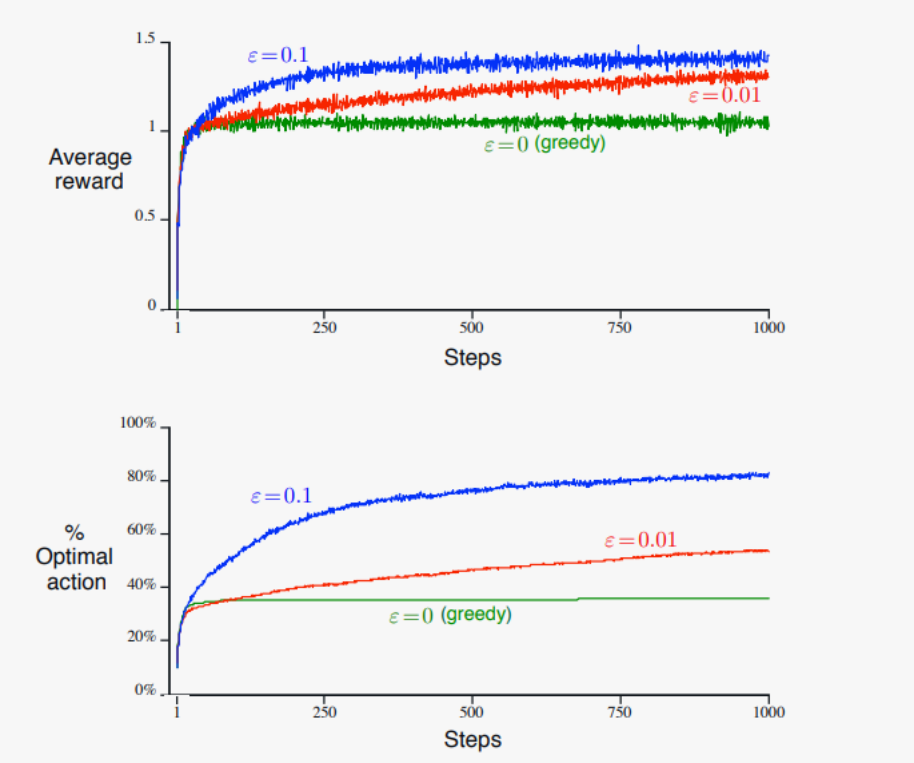
Problem 2

Why balancing exploration/exploitation is important? Name a few ways to balance exploration/exploitation in multi-armed bandits.

Answer

Balancing exploration and exploitation is important because we want to be able to eventually find the optimal action. If we only exploit, we will never find the optimal action. This is the the *greedy* approach, where we always choose the action with the highest reward so far. On the other hand, if we only explore, we will never be able to exploit the highest estimated action (the action we believe to be the optimal action).

When we give our action-value method the opportunity to explore some of the time, we find that we are able to estimate the highest known reward far better than the greedy approach. In fact, the greedy approach simply plateaus off after a while, attaining a roughly 0 slope. The approaches that have been given some opportunity to explore maintain a positive slope and continue to estimate the reward better. The following two graphs from the textbook summarize this behavior.



The following are a few ways to balance exploration/exploitation in multi-armed bandits.

- **Epsilon-greedy**: This is the simplest approach when the bandit has stationary probability distributions. We choose the greedy action with probability $1 - \epsilon$, and we choose a random action with probability ϵ.
- **Optimistic initial values**: We set high initial values for the actions in our action-value algorithm, so that we are encouraged to explore more. This is because we will always choose the action with the highest estimated reward, so with high initial values, the algorithm is more likely to choose various actions and therefore explore more.

- **Upper confidence bound (UCB)**: This approach helps choose actions according to their potential for actually being optimal (as against a random choice as in the case for epsilon-greedy). The UCB action selection uses the following formula (from the textbook):

$$UCB = Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}$$

Where $Q_t(a)$ is the current estimate of the value of action a. Also, c is a constant that determines the degree of exploration, t is the number of time-steps so far, and $N_t(a)$ is the number of times that action a has been chosen prior to time-step t.

Problem 3

In the equation,

$$NewEstimate = OldEstimate + StepSize \times [Target - OldEstimate]$$

what is the target?

Answer

In general, the target is the presumed desired value of the action-value function that we are trying to estimate, or the direction we are trying to move towards. That is why $[Target - Estimate]$ is known as the *error* in the estimate.

In the specific case of the multi-armed bandit problem, the target is the reward that we received after taking the action. With each time step, we attempt to get closer to the target value. For the sample-average method used in the multi-armed bandit problem, the equation in the problem takes on the following form,

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

where the target is the reward R_n that we received after taking the action.

Problem 4

What is the purpose of using Upper Confidence Bound (UCB)?

Answer

The purpose of using UCB is to provide exploratory behavior balanced with exploitation in a systematic manner.

The UCB formula is as follows,

$$A_t = argmax_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

Here, the square root term is a measure of the uncertainty in the estimate of the current action a, and c is used to control the confidence level of that uncertainty. The way this square root term works is that it increases if the action has not been chosen often (because denominator $N_t(a)$ is the number of times the action is chosen), and decreases if the action has been chosen often. This means that the action will be chosen more often if it has not been chosen much in the past, and less often if it has been chosen a lot in the past. The nature of the logarithm term is ideal because, in the beginning it favors exploration overall because of high slope, but then as all actions are tried, it flattens out and favors exploitation.

Therefore, UCB is used to approach the true value of an action in a more 'systematic' way than epsilon-greedy.

Problem 5

Why do you think in Gradient Bandit Algorithm, we defined a soft-max distribution to choose the actions from, rather than just choosing action?

Answer

In the Gradient Bandit Algorithm, we are looking to to create a **numerical preference** for each action. The ideal way to do this is by using the *soft-max distribution*, or the Gibbs/Boltzmann distribution. This is done by exponentiating the action-value function, and then normalizing it. In the form of a formula, this is,

$$\pi_t(a) \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}}$$

which is the probability of choosing action a at time-step t.

The reason we want to create a numerical preference for each action is because we want a snapshot in the current time-step of likely we are to select each action. This gives us a high degree of predictability in our method. This is a big improvement over the epsilon-greedy method, where we have no idea how likely we are to select each action (because with epsilon probability, it is random, and just chooses an action).