Reading Summary for Reinforcement Learning: An Introduction Chapter 2

Yufeng Yuan(Arthur)

This chapter introduces solution methods for bandit problems, which consists only one state and are the simplest form of reinforcement learning. Exploration is the most important feature that distinguishes reinforcement learning from other machine learning algorithms because training information is used to evaluate the actions taken rather than instructs by giving labels. There are two kinds of feedback in machine learning. Evaluative feedback indicates how good the action taken was, but not whether it was the best or not; instructive feedback indicates the correct action, no matter what the action actually taken. Evaluative feedback is discussed in this chapter in a nonassociative setting, where the agent will not act in more than one state.

In a k-armed bandit Problem, the agent can choose from K different actions and each leads to a numerical reward drawn from an unknown probability distribution. If we denote the action taken on time step t as A_t, and the corresponding reward as R_t, we can denote the value of action a as $q^*(a)$, which is the expected reward given a: $q^*(a) = E[R_t|A_t = a]$. Since $q*$ is unknown, we denote Q as its estimate. In this case, at any time step, there is at least one greedy action with the greatest estimated value. Once we have Q, it will be trivial to derive our greedy policy by setting: $A_t = argmax_a Q_t(a)$. However, totally dependent on greedy actions may not lead to good performance, because initially we do not have accurate Q. This leads to the exploration and exploitation problems in reinforcement learning and we should well balance them in the training phase. The simplest technique used for exploration is ϵ-greedy method, with which the agent will pick the greedy action mostly, but with a small probability ϵ, it will pick the random actions.

There are two more ways to encourage exploration discussed in this chapter. One way is optimistic initial value, which initializes Q with values higher than true values. In this case, the agent will actively pick different actions at the beginning. Another way is upper-confidence-bound action selection, which gives bonus to actions rarely selected. In this case, the action is selected according to:

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

where constant c controls the degree of exploration and $N_t(a)$ denotes the number of times action a has been selected. UBC usually achieves better performance than simple

ϵ-greedy.

To obtain the estimate Q, one natural way is to calculate it as the sample average of observed rewards. However, such average can be computed more efficiently with incremental implementation: $Q_{n+1} = Q_n + 1/n[R_n - Q_n]$. This estimate is unbiased to problems with stationary distribution. However, the typical problems in reinforcement learning usually consists of non-stationary distribution where using a fixed step size will be more effective: $Q_{n+1} = Q_n + \alpha[R_n - Q_n]$. In this case, recent samples will be given more weights but the convergence is usually not guaranteed. According to the stochastic approximation theory, the conditions required to assure convergence with probability 1 are:

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \text{ and } \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty$$

where $\alpha_n(a)$ is the step-size for nth selection of action α.

Rather than focusing on the value-based approach, policy can be derived by explicitly modeling the policy distribution. If we denote preference as $H_t(a)$, the policy distribution $\pi_t(a)$ can be written as:

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

On each step, after selecting action A_t and receiving the reward R_t, the update rule for action preference can be written as:

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \hat{R}_t)(1 - \pi_t(A_t))$$
$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \hat{R}_t)\pi_t(a), a \neq A_t$$

where α is the step-size parameter and $\hat{R}_t$ is the average of all the rewards so far.

A more complex setting for bandit problems is the contextual bandits where the player confronts a randomly-picked bandit at each time step. In this case, the bandit problems will become non-stationary which is challenging to all the algorithms discussed so far. Contextual bandits are the intermediate between the simplest bandit problems and full reinforcement learning where actions taken by the agent will affect next state.