Chapter 2: Multi-armed Bandits - Highlights

Jacob Taylor Cassady

November 27, 2022

- The most important feature distinguishing reinforcement learning from other types of learning is that it uses training information that *evaluates* the actions taken rather than *instructs* by giving correct actions.
- Purely evaluative feedback indicates how good the action taken was, but not whether it was the best or the worst action possible.
- one that does not involve learning to act in more than one situation. This *nonassociative* setting
- *associative*, that is, when actions are taken in more than one situation.

1 A k-armed Bandit Problem

- You are faced repeatedly with a choice among k different options, or actions. After each choice you receive a numerical reward chosen from a stationary probability distribution that depends on the action you selected. Your objective is to maximize the expected total reward over some time period, for example, over 1000 action selections, or *time steps.*
- This is the original form of the *k-armed bandit problem*, so named by analogy to a slot machine, or "one-armed bandit," except that it has k levers instead of one.
- Through repeated action selections you are to maximize your winnings by concentrating your actions on the best levers.
- Today the term "bandit problem" is sometimes used for a generalization of the problem described above, but in this book we use it to refer just to this simple case.
- In our k-armed bandit problem, each of the k actions has an expected or mean reward given that action is selected; let us call this the *value* of that action. We denote the action selected on time step t as A_t, and the corresponding reward as R_t. The value then of an arbitrary action a, denoted $q_*(a)$, is the expected reward given that a is selected:

$$q_*(a) \doteq \mathbb{E}[R_t|A_t = a]$$

- We assume that you do not know the action values with certainty, although you may have estimates. We denote the estimated value of action a at time step t as $Q_t(a)$. We would like $Q_t(a)$ to be close to $q_*(a)$.

- If you maintain estimates of the action values, then at any time step there is at least one action whose estimated value is greatest. We call these the *greedy* actions. When you select one of these actions, we say that you are *exploiting* your current knowledge of the values of the actions. If instead you select one of the nongreedy actions, then we say you are *exploring*, because this enables you to improve your estimate of the nongreedy action's value.

- Reward is lower in the short run, during exploration, but higher in the long run because after you have discovered the better actions, you can exploit them many times.

2 Action-value Methods

- methods for estimating the values of actions and for using the estimates to make action selection decisions, which we collectively call *action-value methods.*

-
$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}} \tag{2.1}$$

  where $\mathbb{1}_{predicate}$ denotes the random variable that is 1 if *predicate* is true and 0 if it is not. If the denominator is zero, then we instead define $Q_t(a)$ as some default value, such as 0. As the denominator goes to infinity, by the law of large numbers, $Q_t(a)$ converges to $q_*(a)$. We call this the *sample-average* method for estimating action values because each estimate is an average of the sample of relevant rewards.

- The simplest action selection rule is to select one of the actions with the highest estimated value, that is, one of the greedy actions as defined in the previous section.

- We write this greedy action selection method as

$$A_t \doteq \underset{a}{argmax}\ Q_t(a)$$

- A simple alternative is to behave greedily most of the time, but every once in a while, say with small probability ϵ, instead select randomly from among all the actions with equal probability, independently of the action-value estimates. We call methods using this near-greedy action selection rule ϵ-greedy methods.

- An advantage of these methods is that, in the limit as the number of steps increases, every action will be sampled an infinite number of times, thus ensuring that all the $Q_t(a)$ converge to $q_*(a)$. This of course implies that the probability of selecting the optimal action converges to greater than 1-ε, that is, to near certainty.

3 The 10-armed Testbed

- a set of 2000 randomly generated k-armed bandit problems with k = 10. For each bandit problem, the action values, $q_*(a)$, a = 1, . . . , 10, were selected according to a normal (Gaussian) distribution with mean 0 and variance 1. Then, when a learning method applied to that problem selected action A_t at time step t, the actual reward, R_t, was selected from a normal distribution with mean $q_*(A_t)$ and variance 1. We call this suite of test tasks the *10-armed testbed.*

- The greedy method improved slightly faster than the other methods at the very beginning, but then leveled off at a lower level.

- The greedy method performed significantly worse in the long run because it often got stuck performing suboptimal actions.

- The ε-greedy methods eventually performed better because they continued to explore and to improve their chances of recognizing the optimal action. The ε= 0.1 method explored more, and usually found the optimal action earlier, but it never selected that action more than 91% of the time. The ε= 0.01 method improved more slowly, but eventually would perform better than the ε= 0.1 method on both performance measures shown in the figure. It is also possible to reduce ε over time to try to get the best of both high and low values.

- suppose the reward variance had been larger, say 10 instead of 1. With noisier rewards it takes more exploration to find the optimal action, and ε-greedy methods should fare even better relative to the greedy method. On the other hand, if the reward variances were zero, then the greedy method would know the true value of each action after trying it once. In this case the greedy method might actually perform best because it would soon find the optimal action and then never explore.

- Even if the underlying task is stationary and deterministic, the learner [typically] faces a set of banditlike decision tasks each of which changes over time as learning proceeds and the agent's decision-making policy changes.

4 Incremental Implementation

- Incremental sample averages estimate

$$Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$$

5 Tracking a Nonstationary Problem

- .

6 Optimistic Initial Values

- .

7 Upper-Confidence-Bound Action Selection

- .

8 Gradient Bandit Algorithms

- .

9 Associative Search (Contextual Bandits)

- .

10 Summary

- .