Lecture 2: Multi-armed Bandit Problem (MAB)

Tianpei Xie

Dec 27th., 2017

Contents

1 Multi-armed Bandits

Compared to supervised learning where the training samples are used to **instruct** by giving the correct actions, in reinforcement learning, training samples are used to **evaluate** the actions taken. Purely ***evaluative feedback*** indicates how good the actions taken was, but not whether or not it was the best or worst action possible, and purely ***instructive feedback***, on the other hand, indicates *the correct action to take*, independently of the action actually taken. In their pure forms, these two kinds of feedback are quite distinct: *evaluative feedback depends entirely on the action taken*, whereas *instructive feedback is independent of the action taken*. In other words, supervised learning enforce each step to be as instructed whereas the reinforcement learning only cares if the end goal is reached, and does not care the process.

The multi-armed bandit problem is seen as a reinforcement learning problem with ***nonassociative*** setting, one that does not involve learning to act in more than one situation (**only a single state**).

A ***k*-armed bandit problem** is stated as follows: Consider the following learning 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.

1.1 Exploration and Exploitation

Consider a k-armed bandit problem, each of the k actions has an expected or mean reward given that action is selected. We refer it as the *value* of that action. Denote action selected at time t as A_t, and the corresponding reward is R_t. The value then of an arbitrary action a_t denoted $q_*(a)$ is the ***expected reward*** given that a is selected.

$$q_*(a) := \mathbb{E}\left[R_t \mid A_t = a\right].$$

The problem is trivial if we know the value of each action: the optimal solution is to select the action with highest value. Assume that we do not know the action values with certainty. 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 $q_*(a)$.

At any time step, given estimates of values for all actions, we can choose one with highest estimated value. The corresponding action is called ***greedy action***, and the strategy we use is referred as *exploiting* your current knowledge of the values of actions. Instead, if an non-greedy action is selected, we can improve the estimate of value for that action. This strategy is called *exploration.*

Exploitation is the right thing to do to maximize the expected reward on the one step, but exploration may produce the greater total reward in the long run. Because it is not possible both to explore and to exploit with any single action selection, one often refers to the "tradeoff" between exploration and exploitation. In theory, many sophisticated methods have been proposed to balance the exploration and exploitation, under strong assumptions that may not holds in practice.

```
A simple bandit algorithm

Initialize, for $a = 1$ to $k$:
    $Q(a) \leftarrow 0$
    $N(a) \leftarrow 0$

Loop forever:
    $A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$
    $R \leftarrow bandit(A)$
    $N(A) \leftarrow N(A) + 1$
    $Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\big[R - Q(A)\big]$
```

Figure 1: Multi-armed Bandit with ϵ-greedy. [?]

2 Methods

2.1 Action-value Methods

The natural way to estimate the expected rewards is to average the rewards actually received:

$$Q_t(a) := \frac{\sum_i^{t-1} R_i \cdot \mathbb{1}\{A_i = a\}}{\sum_i^{t-1} \mathbb{1}\{A_i = a\}} \tag{1}$$

and $Q_t(a) = 0$ if the denominator is zero. By Law of large numbers, $Q_t(a) \to q_*(a)$ as $t \to \infty$. This is the sample-average method for estimating action values because each estimate is an average of the sample of relevant rewards.

The simplest strategy is: *greedy action selection* method

$$A_t := \text{argmax}_a\, Q_t(a) \tag{2}$$

Greedy action selection always exploits current knowledge to maximize immediate reward; it spends no time at all sampling apparently inferior actions to see if they might really be better.

Another strategy is ϵ-greedy action selection:

$$A_t = \begin{cases} \text{argmax}_a\, Q_t(a) & \text{w.p. } 1-\epsilon \\ \text{uniform}(\mathcal{A}) & \text{w.p. } \epsilon \end{cases} \tag{3}$$

where $\mathcal{A} := \{\text{all actions available}\}$. The benefit for ϵ-greedy action selection 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 $q_*(a)$.

2.1.1 Incremental update

We can compute $\boldsymbol{Q}_n := (Q_n(a) : a \in \mathcal{A})$ via incremental updates

$$\begin{aligned}\boldsymbol{Q}_{n+1} &:= \frac{\sum_i^n \boldsymbol{R}_i}{n} \\ &= \frac{1}{n}\left(\boldsymbol{R}_n + (n-1)\boldsymbol{Q}_n\right) \\ &= \boldsymbol{Q}_n + \alpha_n(\boldsymbol{R}_n - \boldsymbol{Q}_n) \quad \text{where } \alpha_n := \frac{1}{n} \qquad (4) \\ \Rightarrow \textbf{new estimate} &= \textbf{old estimate} + \text{stepsize} \cdot (\text{target} - \text{old estimate}) \qquad (5)\end{aligned}$$

where $\boldsymbol{R}_i := (R_{a_i} \text{ if } a = a_i \text{ otherwise } 0, \text{for all } a \in \mathcal{A})$. Note that $(\boldsymbol{R}_n - \boldsymbol{Q}_n)$ serves as an *error* term in the estimate. By reducing the error, the estimate is approaching the target, which is presumably a desirable direction in which to move, although it may be noisy.

2.1.2 Tracking nonstationary problem

The averaging method in (4) does not fit if the MAB is non-stationary, e,g, if the reward probabilities change over time. In that case, it makes sense to give more weight to recent rewards than to long-past rewards. One of the most popular ways of doing this is to use a constant step-size parameter $\alpha_n := \mathit{const} \in (0, 1]$.

$$\begin{aligned}\boldsymbol{Q}_{n+1} &:= \boldsymbol{Q}_n + \alpha(\boldsymbol{R}_n - \boldsymbol{Q}_n) \\ &= \alpha\boldsymbol{R}_n + (1-\alpha)\boldsymbol{Q}_n \\ &= (1-\alpha)^n\boldsymbol{Q}_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i}\boldsymbol{R}_i \qquad (6)\end{aligned}$$

This results in $\boldsymbol{Q}_{n+1}$ being a *weighted average* of past rewards and the initial estimate $\boldsymbol{Q}_1$. Also see that $(1-\alpha)^n + \sum_{i=1}^n \alpha(1-\alpha)^{n-i} = 1$. And the weight depends on $n - i$, i.e. how many rewards ago was observed and it decays *exponentially* as the number of intervening rewards increases. This is also called *exponential recency-weighted average.*

We can also choose the stepsize $\alpha_n(a)$ as a function of n. It assure convergence of equation (4) w.p. 1 if

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

The first condition is required to guarantee that the steps are large enough to eventually overcome any initial conditions or random fluctuations. The second condition guarantees that eventually the steps become small enough to assure convergence. Note that for $\alpha_n = \frac{1}{n}$, both conditions are satisfied, but for $\alpha_n = \alpha$ const. the second condition is not satisfied, meaning that the process do not converge, which is expected for non-stationary MAB.