Author：Liu Jian

Time：2021-07-27

Multi-armed Bandits

1 k-摇臂赌博机问题描述

A k-armed bandit problem 是一个很简单的问题，存在很强的假设，因此不是一个 full reinforcement learning problem。但通过介绍该问题能很清晰的展示出 the need to balance exploration and exploitation is a distinctive challenge that arises in reinforcement learning. 此外，We use this problem to introduce a number of basic learning methods which we extend in later chapters to apply to the full reinforcementl earning problem.

我们先介绍 nonassociative 的摇臂赌博机，再介绍 associative (that is, when the best action depends on the situation) 的摇臂赌博机。nonassociative 的摇臂赌博机又可分为 stationary (静态) 和 nonstationary (非静态) 的情况，即选择某个 action 后产生的 reward 服从的概率分布是固定的 (静态，stationary problem: bandit problems in which the reward probabilities do not change over time) 还是会随时间变化的 (非静态，that is, the true values of the actions changed over time)；更特殊地，若每个时间步返回的 reward 只可能取一个值而不是随机的服从某一概率分布，则赌博机是 deterministic (或者说，k-臂赌博机的每个臂产生的 reward 的方差为 0)。而 associative 的情况就很接近 the full reinforcement learning problem 了。

最简单的 stationary k-armed 摇臂赌博机问题描述如下：

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.

A_t: the action selected on time step t

R_t: the corresponding reward of A_t

the value of an arbitrary action a: $q_\star(a) \triangleq \mathbb{E}[R_t|A_t = a]$

the estimated value of action a at time step t: $Q_t(a)$, We would like $Q_t(a)$ to be closet to $q_\star(a)$.

2 Action-value Methods

action-value methods: estimate the values of actions and use the estimates to make action selection decisions。这里，动作 a 的价值 $q_\star(a)$ 的估计 $Q_t(a)$ 由 a 所产生的 reward 取平均得到。针对静态和非静态问题，我们有不同的取平均方式；此外，价值函数也可采用不同的初始化策略。

各算法的实际效果 (the practical effectiveness of the methods) 由数值实验—— 10-armed testbed 给出：

- 2000 个随机生成的 10-armed bandits：对单个 10-armed bandit，每个臂的价值 $q_\star(a)$ $(a = 1, 2, \cdots, 10)$ 由 $\mathcal{N}(0, 1)$ 随机生成，而每个臂返回的价值服从均值为 $q_\star(a)$ 方差为 1 的高斯分布；
- 每个 10-armed bandit 的一轮 (run) 实验包含 1000 个时间步，一轮跑完就得到了被评估算法在该 10-armed bandit 上的实际效果；对 2000 个 10-armed bandits 每个都跑一轮 (共 2000 轮)，再对结果取平均，平均掉赌博机的影响，就得到了算法每步的平均效果。

---

greedy method: $A_t = \arg\max_a Q_t(a)$，select one of the actions with the highest estimated value。

ε-greedy method: 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.

- ε-greedy method 选择最优的 action 在理论上是 asymptotic guarantee 的: the probability of selecting
  the optimal action converges to greater than $1 - \varepsilon$, that is, to near certainty. 此外，由 10-armed testbed 也可验证，The $\varepsilon = 0.1$ method explored more, and usually found the optimal action earlier, but it never selected that action more than 91% of the time. The $\varepsilon = 0.01$ method improved more slowly, but eventually would perform better than the $\varepsilon = 0.1$
  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.

对于 stationary problem，$Q_t(a)$ 可直接采用 reward 的算术平均来计算：

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

进一步，我们可以采用增量的方式完成上述计算。**the incremental implementation of action-value methods**: the averages of rewards can be computed in a computationally efficient manner, in particular, with constant memory and constant per-time-step. To simplify notation we concentrate on a single action. Let R_i now denote the reward received after the ith selection of this action, and let Q_n denote the estimate of its action value after it has been selected $n - 1$ times, i.e. $Q_n = \frac{\sum_{i=1}^{n-1} R_i}{n-1}$. Then,

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

更一般的更新格式可总结为：

$$\text{NewEstimate} \leftarrow \text{OldEstimate} + \text{StepSize}(\text{Target} - \text{OldEstimate})$$

这里，StepSize 为 $\frac{1}{n}$，we denote the step-size parameter by α or, more generally, by $\alpha_t(a)$.

对于 nonstationary problem，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：

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

where the step-size parameter $\alpha \in (0, 1]$ is constant. This results in Q_{n+1} being a weighted average of past rewards and the initial estimate Q_1:

$$\begin{aligned} Q_{n+1} &= Q_n + \alpha(R_n - Q_n) \\ &= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i \end{aligned}$$

上式能被称为加权平均是因为 $(1-\alpha)^n + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} = 1$. Accordingly, this is sometimes called an exponential recency-weighted average.

关于收敛性的讨论：更一般地，Let $\alpha_n(a)$ denote the step-size parameter used to process the reward received after the nth selection of action a:

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

对于 sample-average method，$\alpha_n(a) = \frac{1}{n}$；对于 exponential recency-weighted average method，$\alpha_n(a) \equiv \alpha$。对于 stationary problem，sample-average method is guaranteed to converge to the true action values by the law of large numbers，因此是适用的。根据统计近似理论，依概率收敛的条件需满足如下两个条件：

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

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. 可以看到，对于 constant step-size parameter $\alpha_n(a) = \alpha$，第二个条件并不满足，indicating that the estimates never completely converge but continue to vary in response to the most recently received rewards. 因此，对 nonstationary problem，exponential recency-weighted average method 才适用!

下面讨论初始价值函数 $Q_1(a)$ 的设置对算法的影响。

All the methods we have discussed so far are dependent to some extent on the initial action-value estimates, $Q_1(a)$. In the language of statistics, these methods are biased by their initial estimates.

the sample-average methods 不受 $Q_1(a)$ 初始化的影响，但 methods with constant α 会受影响：For the sample-average methods, the bias disappears once all actions have been selected at least once (也就是说，当不存在 a 的样本时，其价值为设定的默认值，而一旦存在 a 的样本，则丢弃默认值，适用样本返回的 reward 的平均值), but for methods with constant α, the bias is permanent, though decreasing over time as given by $Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$ (可以看到，初始设定的 Q_1 总是参与价值函数的估计的).

The bias $Q_1(a)$ 的利弊：(1) The downside is that the initial estimates become, in effect, a set of parameters that must be picked by the user; (2) The upside is that they provide an easy way to supply some prior knowledge about what level of rewards can be expected. 比如 Initial action values can also be used as a simple way to encourage exploration (例子可参考Page 34 Fig 2.3). We call this technique for encouraging exploration optimistic initial values.

- 这一 trick 只适用于 stationary problem 而不适用于 nonstationary problem 等更一般的强化学习任务: We regard it as a simple trick that can be quite effective on stationary problems, but it is far from being a generally useful approach to encouraging exploration. For example, it is not well suited to nonstationary problems because its drive for exploration is inherently temporary. If the task changes, creating a renewed need for exploration, this method cannot help. **Indeed, any method that focuses on the initial conditions in any special way is unlikely to help with the general nonstationary case**. The beginning of time occurs only once, and thus we should not focus on it too much. (基于类似的原因，sample-average methods 不适用于 nonstationary problem: This criticism applies as well to the sample-

average methods, which also treat the beginning of time as a special event, averaging all subsequent rewards with equal weights.)

constant step size methods 会 produce initial bias，但能处理 nonstationary problem；sample-average methods 不会 produce initial bias ，但无法处理 nonstationary problem。为此，本小节最后还介绍了 Unbiased Constant-Step-Size Trick，这是一种能 avoid the bias of constant step sizes while retaining their advantages on nonstationary problems 的方法，即 Q_n is exponential recency-weighted average without initial bias。

ε-greedy method 引入随机性来兼顾 exploration 和 exploitation。进一步，Upper-Confidence-Bound Action Selection (UCB) 综合考虑 (1) 当前价值函数估计值的大小，以及 (2) 价值函数估计的不确定大小，来选择 action a 以兼顾 exploration 和 exploitation：

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

where $N_t(a)$ denotes the number of times that action a has been selected prior to time t, and the number $c > 0$ controls
the degree of exploration. If $N_t(a) = 0$, then a is considered to be a maximizing action.

The idea of this upper confidence bound (UCB) action selection is that the square-root term is a measure of the uncertainty or variance in the estimate of a's value. The quantity being max'ed over is thus a sort of upper bound on the possible true value of action a, with c determining the confidence level.

UCB often performs well, as shown here, but is more difficult than ε-greedy to extend beyond bandits to the more general reinforcement learning settings considered in the rest of this book.

3 Gradient Bandit Algorithms

不同于 action-value methods estimate action values and use those estimates to select actions. 这里，我们采用另外一种方法，即 考虑 learning a numerical preference for each action a, which we denote $H_t(a) \in \mathbb{R}$. The larger the preference, the more often that action is taken, but the preference has no interpretation in terms of reward. Actions are detetmined according to a soft-max distribution as follows:

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

where the new notation $\pi_t(a)$ denotes the probability of taking action a at time t.

preference $H_t(a)$ 的更新方式如下：On each step, after selecting action A_t and receiving the reward R_t, the action preferences are updated by:

$$\begin{aligned} H_{t+1}(A_t) &= H_t(A_t) + \alpha(R_t - \overline{R}_t)(1 - \pi_t(A_t)), \quad \text{and} \\ H_{t+1}(a) &= H_t(a) - \alpha(R_t - \overline{R}_t)\pi_t(a), \qquad \text{for all } a \neq A_t \end{aligned}$$

where $\alpha > 0$ is a step-size parameter, and $\overline{R}_t \in \mathbb{R}$ is the average of the rewards up to but not including time t (with $\overline{R}_1 = R_1$), which can be computed incrementally.

The $\overline{R}_t$ term serves as a baseline with which the reward is compared. If the reward is higher than the baseline, then the probability of taking A_t in the future is increased, and if the reward is below baseline, then the probability is decreased. The non-selected actions move in the opposite direction.

事实上，上述更新公式可视为 a stochastic approximation to gradient ascent of expected reward $\mathbb{E}[R_t] = \sum_x \pi_t(x) q_\star(x)$ (这里的 $\sum_x$ 对应积分)，This assures us that the algorithm has robust convergence properties.下面的推导比书中的更合理。

我们希望基于概率 $\pi_t(x)$ (由 preference $H_t(x)$ 给出) 选择 action 这一策略的期望回报 $\mathbb{E}[R_t] = \sum_x \pi_t(x) q_\star(x)$ 尽可能大，为此我们采用梯度上升法最大化 $\mathbb{E}[R_t]$，即对所有 a：

$$H_t(a) \leftarrow H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)}$$

进一步，我们可以作推导：

$$\begin{aligned} \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} &= \sum_x q_\star(x) \frac{\partial \pi_t(x)}{\partial H_t(a)} \\ &= \sum_x (q_\star(x) - B_t) \frac{\partial \pi_t(x)}{\partial H_t(a)} \end{aligned}$$

where B_t, called the baseline, can be any scalar that does not depend on x (能添加 B_t 的原因是 $\sum_x \frac{\partial \pi_t(x)}{\partial H_t(a)} = 0$).

将上式写成期望的形式以便于使用蒙特卡洛法近似计算梯度：

$$\begin{aligned} \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} &= \sum_x \pi_t(x) (q_\star(x) - B_t) \frac{\partial \pi_t(x)}{\partial H_t(a)} / \pi_t(x) \\ &= \mathbb{E}\left[(q_\star(x) - B_t) \frac{\partial \pi_t(x)}{\partial H_t(a)} / \pi_t(x) \right] \\ &= \mathbb{E}\left[(q_\star(x) - B_t)(\mathbb{1}_{a=x} - \pi_t(a)) \right] \end{aligned}$$

其中，$x \sim \pi_t(x)$，$\frac{\partial \pi_t(x)}{\partial H_t(a)} = \pi_t(x)(\mathbb{1}_{a=x} - \pi_t(a))$。对于上述期望，我们只选用一个样本点来近似计算，也就是第 t 个时间步选择的 action，即 $x = A_t$，由此：

$$\begin{aligned} \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} &= \mathbb{E}\left[(q_\star(x) - B_t)(\mathbb{1}_{a=x} - \pi_t(a)) \right] \\ &\approx (q_\star(A_t) - B_t)(\mathbb{1}_{a=A_t} - \pi_t(a)) \end{aligned}$$

上式中，但我们并不知道 $q_\star(x)$，因此需要作进一步的近似。同样地，由蒙特卡洛法 $q_\star(A_t) = \mathbb{E}[\text{reward} | A_t] \approx R_t$，则：

$$\begin{aligned} \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} &\approx (q_\star(A_t) - B_t)(\mathbb{1}_{a=A_t} - \pi_t(a)) \\ &\approx (R_t - B_t)(\mathbb{1}_{a=A_t} - \pi_t(a)) \end{aligned}$$

由此，随机梯度上升的更新公式：

$$H_t(a) \leftarrow H_t(a) + \alpha (R_t - B_t)(\mathbb{1}_{a=A_t} - \pi_t(a)), \quad \text{for all } a$$

进一步，取 $B_t = \overline{R}_t$，并使用更新后的 $H_t(a)$ 作为 $t+1$ 步决策的依据，即 preference $H_{t+1}(a)$，由此即得前文中的更新公式。可以看到，对于 baseline B_t，设置为任一数值都可以：For example, we could have set it to zero, or to 1000, and the algorithm would still be an instance of stochastic gradient ascent. 但是，选择合适的 B_t 能提高算法的数值稳定性：The choice of the baseline does not affect the expected update of the algorithm, but it does affect the variance of the update and thus the rate of convergence. 通常我们将其设为 $\overline{R}_t$：Choosing it as the average of the rewards may not be the very best, but it is simple and works well in practice.

可以看到，根据 $H_t(x)$ 所给出的概率分布进行决策既在 exploit 又在 explore，而决策反馈修正 $H_t(x)$ 并作为下一步决策的依据 $H_{t+1}(x)$ 则体现了强化学习通过与环境交互来进行学习的特点。

Gradient bandit algorithms are a special case of the gradient-based reinforcement learning algorithms introduced by Williams (1992) that later developed into the actor–critic and policy-gradient algorithms that we treat later in this book.

对前文几种方法的总结：

We have presented in this chapter several simple ways of balancing exploration and exploitation:

- The ε-greedy methods choose randomly a small fraction of the time, whereas UCB methods choose deterministically but achieve exploration by subtly favoring at each step the actions that have so far received fewer samples.
- Gradient bandit algorithms estimate not action values, but action preferences, and favor the more preferred actions in a graded, probabilistic manner using a soft-max distribution.
- The simple expedient of initializing estimates optimistically causes even greedy methods to explore significantly.

4 Associative case

At the end of this chapter, we take a step closer to the full reinforcement learning problem by discussing what happens when the bandit problem becomes associative, that is, when the best action depends on the situation (有些文献也称其为 contextual bandits).

associative case 与 nonstationary case 相同的点在于赌博机的性质 (也就是环境) 是不断变化的，不同点在于 associative case 会显示地告知变化的赌博机当前处于哪一种状态（相当于存在多个 k-armed bandits，associative case 会显示地告知当前步操作的是哪一个 k-armed bandits)，因此在决策时，需考虑这一因素，即 action 的选择需以 situation 为自变量 (associate different acitons with situations): the goal is to learn a policy, i.e. a mapping from situations to the actions that are best in those situations. 但对于 stationary/nonstationary case 来说: the learner either tries to find a single best action when the task is stationary, or tries to
track the best action as it changes over time when the task is nonstationary.

Suppose there are several different k-armed bandit tasks, and that on each step you confront one of these chosen at random. Thus, the bandit task changes randomly from step to step，则：

- If the probabilities with which each task is selected for you do not change over time 并且不被告知当前处理的是哪个 k-armed bandit task, this would appear as a single stationary k-armed bandit task.
- If the probabilities with which each task is selected for you change over time 并且不被告知当前处理的是哪个 k-armed bandit task, this would appear as a nonstationary k-armed bandit task.
- associative case: When a bandit task is selected for you, you are given some distinctive clue about its identity (but not its action values). Maybe you are facing an actual slot machine that changes the color of its display as it changes its action values. Now you can learn a policy associating each task, signaled by the color you see, with the best action to take when facing that task.

associative case 介于一般的 k-armed bandit problem 和 full reinforcement learning problem 之间，若进一步，action 还会影响下一步的 situation，则为 full reinforcement learning problem: Associative search tasks are intermediate between the k-armed bandit problem and the full reinforcement learning problem. They are like the full reinforcement learning problem in that they involve learning a policy, but they are also like our version of the k-armed bandit problem in that each action affects only the immediate reward. If actions are allowed to affect the next situation as well as the reward, then we have the full reinforcement learning problem.

> 最后，文中提到，基于 Bayesian method 对摇臂赌博机进行建模，得到的是一个 full reinforcement learning problem.

例子：Suppose you face a 2-armed bandit task whose true action values change randomly from time step to time step. Specifically, suppose that, for any time step, the true values of actions 1 and 2 are respectively 10 and 20 with probability 0.5 (case A), and 90 and 80 with probability 0.5 (case B). (虽然说是 face 一个 2-armed bandit，但实际上，case A 对应一个 2-armed bandit，而 case B 对应另一个 2-armed bandit)

1. If you are not able to tell which case you face at any step, what is the best expected reward you can achieve and how should you behave to achieve it?

   Answer：虽然存在 case A 和 case 2 两种 2-armed bandit，但我们没有 situation 信息 (不是 associative case)，此外，case A 和 case B 出现的概率是固定不变的 (不是 nonstationary case)，因此，问题实际上是一个 a single stationary k-armed bandit task。对于这个 stationary 2-armed bandit task，action 1 的期望 reward 为:

   $$q_\star(a=1) = \Pr\{\text{case A}\} \cdot 10 + \Pr\{\text{case B}\} \cdot 90 = 0.5 \cdot 10 + 0.5 \cdot 90 = 50$$

   action 2 的期望 reward 为:

   $$q_\star(a=2) = \Pr\{\text{case A}\} \cdot 20 + \Pr\{\text{case B}\} \cdot 80 = 0.5 \cdot 20 + 0.5 \cdot 80 = 50$$

   因此，采用前述 2、3 节中的策略，每步的期望 reward 最终收敛于 50，最优 action 为 $1, 2$ 均可。

2. Now suppose that on each step you are told whether you are facing case A or case B (although you still don't know the true action values). This is an associative search task. What is the best expected reward you can achieve in this task, and how should you behave to achieve it?

   Answer：若给出信号以区别 case A 和 case B (只知道 A,B case 不同，而不知道不同在哪里)，则学习到的策略与具体的 case 相关 (associate different acitons with situations): 通过学习，最终我们会在 case A 时选择 action 2，在 case B 时选择 action 1，则每步的期望 reward 收敛于 $\Pr\{\text{case A}\} \cdot 20 + \Pr\{\text{case B}\} \cdot 90 = 0.5 \cdot 20 + 0.5 \cdot 90 = 55$。可以看到，相比于不提供 situation 信息，association case 的策略更精细，能得到更高的收益。
