# lec05 详细版

# 多臂赌博机问题

*$\doteq$ 在本文中表示 equality relationship that is true by definition，根据定义为真的平等关系*

## 一个k-臂赌博机问题

考虑以下的学习问题：我们将反复面临 $k$ 个不同的行动中的选择，根据选择的不同，每次选择后，都会收到从固定概率分布中选择的数字奖励。我们的目标是在一段时间内最大化预期总奖励，比如 1000 个操作或 time steps

### k-臂赌博机问题

这个赌博机拥有 $k$ 个杠杆，每个动作就像选择赌博机的杠杆之一，各个杠杆都有各自的回报概率分布。$k$ 个动作中每一个都有平均奖励，这个奖励叫做动作的 *value*。我们将

- time step $t$ 上选择的动作表示为 $A_t$ (Action t)
- 相应的奖励表示为 $R_t$ (Reward t)
- 那么任意动作 $a$ 的 value 表示为 $q_*(a)$，这是选择 $a$ 时的**预期奖励** (为什么要加个"*"啊)
- $Q_t(a)$ 表示在 $t$ 处选择动作 $a$ 的**估计奖励** (为什么不用 $\hat{q}_t(a)$ 啊)

有：

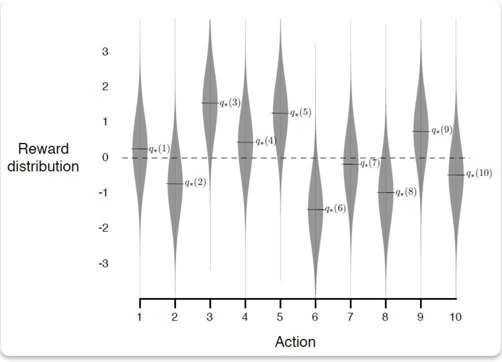
$$q_*(a) \doteq \mathbb{E}[R_t | A_t = a] \tag{1}$$

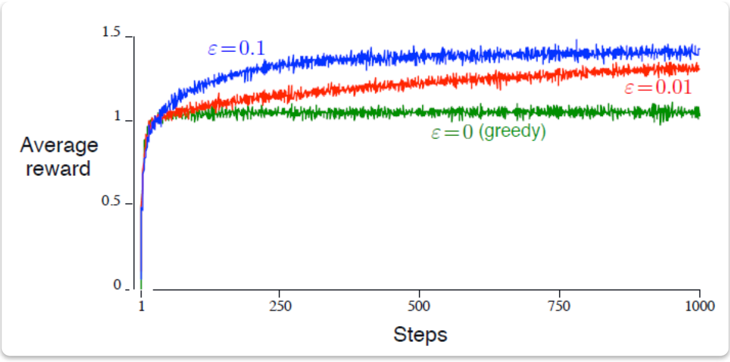
如果我们知道每个动作的 value，那么我们总是选择 value 最高的动作就可以了。
但是在真实的连续多选择问题中，我们不知道这个 value，当然，我们希望 $Q_t(a)$ 尽可能接近 $q_*(a)$

### Greedy, Exploit, Explore

对于任意的 $t$，至少存在一个 $a$ 使得 $Q_t(a)$ 最大，我们可以称这些使 $Q_t(a)$ 最大的动作为 *greedy* 动作。当我们选择 greedy 动作时，被称为 *exploiting*；当我们选择非 greedy 动作时，被称为 *exploring*。

Exploiting 每次选择收益最大的动作，但是在实际情况中，我们的尝试次数往往是有限的。下面的两张图片较好地解释了这一点。
Greedy 方法往往会选择 action-value 最大的动作，在之前的少量样本中 (进行了一定数量的尝试)，算法发现动作 5 的 action-value 最大，因此之后不断重复动作 5，最后的收益均值落在 1 出头，这正好是动作 5 的回报均值。
然而实际上，收益最高的是动作 3，greedy 没有进行 explore，无法发现实际上动作 3 的回报是最大的。
对于 $\epsilon = 0.01$，尝试 explore 的概率太小，寻找最佳动作的速度太慢




# Action-value 方法

## Sample-average

某个行动的真正价值是选择该行动时的平均奖励，估计这一点的一种自然方法是将实际收到的奖励平均：

$$Q_t(a) \doteq \frac{t\text{之前 } a\text{的奖励总和}}{t\text{之前 } a\text{发生的次数}} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \cdot \mathbb{1}_{A_i=a}} \tag{2}$$

ppt 中的公式：

$$Q_t(a) = \frac{r_1 + r_2 + \ldots + r_{k_a}}{k_a} \tag{3}$$

其中，$\mathbb{1}$ 与其角标表示随机变量，如果其角标为 true，则为 1；如果其角标为 false，则为 0。如果分母为 0，则我们将 $Q_t(a)$ 定义为某个默认值，比如 0；$k_a$ 表示动作 $a$ 的次数
当分母趋近于无穷时，根据大数定律，$Q_t(a)$ 将会收敛于 $q_*(a)$

$$\lim_{k_a \to \infty} Q_t(a) = q_*(a) \tag{4}$$

这个方法被称为 *sample-average*，其可能并非最好的方法，但是非常容易理解

对于 greedy 行动，我们定义

$$A_t \doteq \underset{a}{\operatorname{argmax}} \left(Q_t(a)\right) \tag{5}$$

其中 $\operatorname{argmax}_a(\cdot)$ 表示使得括号内表达式最大时的 $a$；当有多个动作都能使得括号内的表达式取得最大时，我们可以通过某种规则选择一个动作 (比如随机)。但就像之前说的那样，greedy 行动无法得知自己的选择是不是最佳选择。

## ε-Greedy

在 greedy 的基础上，我们每步都以 $\epsilon$ 的概率进行 explore，即，每步都有 $\epsilon$ 的概率等可能地随机选择其他的行动。这样的行动被称为 near-greedy，这样的方法被称为 ε-greedy 方法

## 方法的选择

在图2中，发现 ε-greedy 方法远远优于简单的 greedy 方法，实际上在大多数情况下也的确如此。
但是对于 (1) 方差极小，甚至为 0 (2) 一个 run 中能进行的 action 极为有限。基础的 greedy 方法可能更胜一筹

对于非平稳序列，则更加鼓励探索

# The 10-armed Testbed

以10臂赌博机为例，参考*图一*，每个赌博机问题中，动作的价值 $q_*(a), a=1,...,10$ 服从标准正态分布。
在时间步 $t$ 选择动作 $A_t$，其实际收益 $R_t$ 服从均值为 $q_*(A_t)$ 标准差为 1 的正态分布。我们将这套测试任务称为 10-armed testbed。
对于书中的例子，一个 *run* 进行了 1000 个时间步，而我们一共要进行 2000 个 run，每次运行都是一个新的实验，其目的是评估算法的平均行为和性能

# Incremental Implementation (增量实现)

在恒定的内存与恒定的时间步内高效计算这些平均值

我们用 $R_i$ 表示在第 $i$ 次选择这个动作后的收到的回报，$Q_n$ 表示该动作被选择 $n-1$ 次后的估计值，有

$$Q_n \doteq \frac{R_1+R_2+\cdots+R_{n-1}}{n-1} \tag{6}$$

一种显然的做法就是记录所有的回报，再在需要 $Q_n$ 时计算，很明显，这并不是一个好算法

我们可以通过处理每个新回报所需的小而恒定的计算，很容易地设计出用于更新平均值的增量公式。
给定 $Q_n$ 与第 $n$ 次的回报 $R_n$，有

$$\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}(R_n + (n-1)Q_n) \\
&= \frac{1}{n}R_n + nQ_n - Q_n \\
&= Q_n + \frac{1}{n}(R_n - Q_n)
\end{aligned} \tag{7}$$

(7) 的更新规则非常常用，其被称为**增量法**，它更一般的结构可以被描述为下式：

$$\text{NewEstimate} \leftarrow \text{OldEstimeate} + \text{StepSize} \times \left(\text{Target} - \text{OldEstimate}\right) \tag{8}$$

其中，$[\text{Target} - \text{OldEstimate}]$ 就是估计的 *error*。
对于 (7) 中的 $\text{StepSize}$ 部分，它会随着时间步的改变而改变。在处理动作 $a$ 的第 $n$ 次回报时，次方法使用了 step-size 参数 $\frac{1}{n}$，接下来我们会使用 $\alpha$，或者更一般地，$\alpha_t(a)$，作为 step-size 参数

使用增量计算样本均值和 ε-greedy 的伪代码如下所示，我们假设函数 $\text{bandit}(A)$ 会进行动作并返回相应的回报

$$\begin{aligned}
&\text{Initialize, for } a = 1 \text{ to } k: \\
&\quad Q(a) \leftarrow 0 \\
&\quad N(a) \leftarrow 0 \\
&\text{Loop forever}: \\
&\quad A \leftarrow \begin{cases} \text{argmax}_a Q(a) & \text{with } \Pr(1-\epsilon) \quad \text{(breaking ties randomly)} \\ \text{random action} & \text{with } \Pr(\epsilon) \end{cases} \\
&\quad R \leftarrow \text{bandit}(a) \\
&\quad N(A) \leftarrow N(A) + 1 \\
&\quad Q(A) \leftarrow Q(A) + \frac{1}{N(A)}(R - Q(A))
\end{aligned}$$

# Tracking a Non-stationary Problem (非平稳问题)

之前我们讨论的都是平稳状态下 (stationary) 的赌博机问题，平稳状态就是说赌博机的回报概率不会随着时间步改变而改变。但是在相当数量的强化学习任务中，我们会面对非平稳问题。在这种情况下，我们应该*给予最近回报更多的权重*。
对于 (7) 中的公式，我们改写如下：

$$Q_{n+1} \doteq Q_n + \alpha(R_n - Q_n) \tag{9}$$

其中，步长参数 $\alpha \in (0, 1]$ 是一个常数 (固定步长)，因此，$Q_{n+1}$ 是过去回报 $R_i$ 和初始估计值 $Q_1$ 的加权平均值：

$$\begin{aligned}
Q_{n+1} &= Q_n + \alpha(R_n - Q_n) \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)(\alpha R_{n-1} + (1-\alpha)Q_{n-1}) \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-1} + \cdots \\
&\quad + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i
\end{aligned} \tag{10}$$

显然我们有 $(1-\alpha)^n + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} = 1$
对于回报 $R_i$，它出现地越早，权重 $\alpha(1-\alpha)^{n-i}$ 衰减地越多；而当 $\alpha = 1$ 时，所有权中都在最后一个回报 $R_n$ 上。
这个方法有时被称为 *exponential recency-weighted average (指数近期加权平均)*

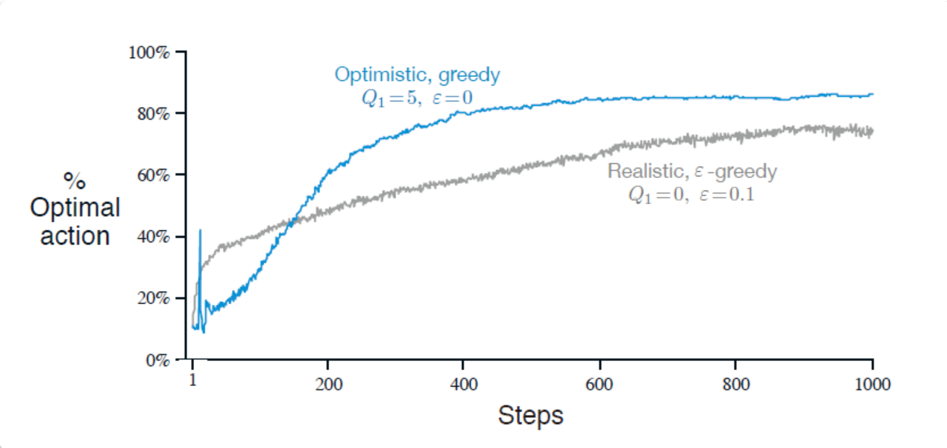
Sample-average 中，$\alpha$ 取 $\frac{1}{n}$ 可以保证估计值 $Q_n$ 收敛于行动的真实价值 $q_*$
(https://zhuanlan.zhihu.com/p/411829189 有详细说明)

$$\begin{aligned}
\sum_{n=1}^{\infty} \alpha_n(a) &= \infty \\
\sum_{n=1}^{\infty} \alpha_n^2(a) &< \infty
\end{aligned} \tag{11}$$

但是指数近期加权平均并不能保证 $Q_n$ 在 $n$ 趋近于无穷时收敛于 $q_*$，不过这正是我们恰恰希望的，跟踪非平稳问题时就该有这样的性质

# Optimistic Initial Values (乐观的初始值)

对于通常的 ε-greedy 算法，初始值通常设置为 0；我们不妨设置为 5 或其他的初始值，使得算法在开始的时候更倾向于探索




这是一种简单的技巧，在处理平稳问题时非常有效，但它远不是鼓励探索的一种普遍有用的方法；它不适用于非平稳问题，因为它的探索动力本质上是暂时的

## 尖峰

*注意到，开头有一个神秘的尖峰，由于这个图像是根据 2000 次重复实验后的平均数据绘制的，因此这个尖峰几乎不可能是偶然现象*

这是因为：前十次我们采样了所有的摇臂，不难计算有 40% 以上的情况最佳摇臂拥有最大的回报，所以在第十一次我们会有 40% 以上的概率选择最佳摇臂；在选择了最佳摇臂之后，由于我们的初始值很大，第 11 次的回报几乎不可能超过之前的回报均值，因此我们转而选择其他摇臂，曲线陡然下降

# Upper-Confidence-Bound Action Selection (置信度上界)

ε-greedy 并没有对非贪婪行动进行区分，它平等地对每个动作进行 explore，而一个很显然的改进思路便是选择那些*具有更大潜力的行动*

对于每个行动，如果

- 行动的估计值与最大值接近
- 行动的不确定性大

那么这个行动具有更大的潜力，因此有下式：

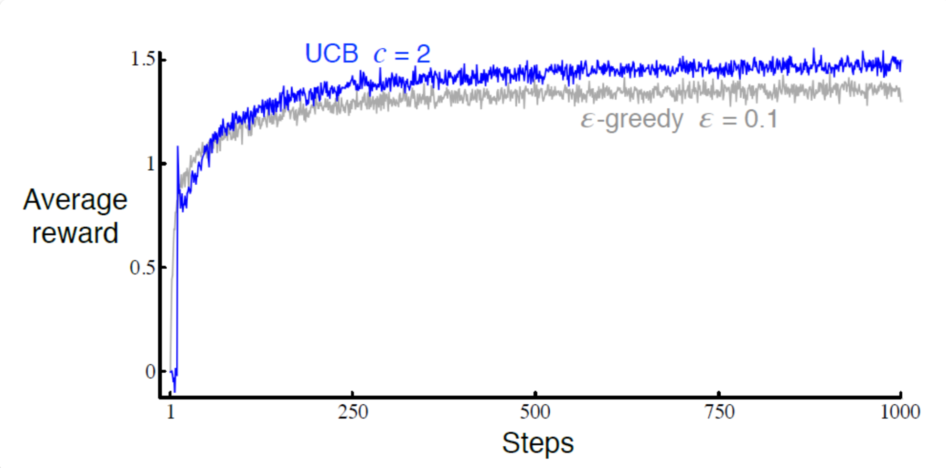
$$A_t \doteq \underset{a}{\operatorname{argmax}} \left[ Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}} \right] \tag{12}$$

其中

- $N_t(a)$ 表示行动 $a$ 在 $t$ 之前被选择的次数 (式 (1) 中的分母)，如果 $N_t(a)=0$，那么 $a$ 会被认为是最大化动作
- $c$ 控制探索的程度
- $\sqrt{\frac{\log t}{N_t(a)}}$ 是不确定性的度量或 $a$ 值估计的方差

一方面说，最大数量是动作 $a$ 的可能真实值的一种上限，$c$ 确定置信度；每次 $a$ 被选择后，不确定性都会减少 (平方根项的大小随着分母 $N_t(a)$ 增加而减小)
另一方面考虑，每次我们选择 $a$ 以外的动作时，$t$ 增加而 $N_t(a)$ 不变，不确定性对数增加
所有动作最终都会被选择，但是那些具有较低估计值的动作，或是那些已经被频繁选择的动作，他们被选择的概率将会随着时间的推移下降

UCB 在多臂赌博机问题上的表现非常好




但是 UCB 方法更难向更为一般的强化学习问题上扩展：

- 难以处理非平稳问题
- 难以处理较大的状态空间，特别是想要利用一些函数进行逼近时

## 尖峰

*像乐观的初始值中的图像一样，UCB 方法的图像也有一个尖峰（提示，$c=1$ 时，尖峰不明显）*

由于 $N_t(a)=0$，则 $a$ 是满足最大化条件的动作，所以第一步到第十步间 10 个动作每个都被选择了一次 $N_t(a)=1$
记置信估计为：

$$C_t(a)=c\sqrt{\frac{\log t}{N_t(a)}}$$

$t=11$，$N_t(a)=1$ 且 $c=2$ 的置信估计为：

$$C_t(a)=2\sqrt{\frac{\log 11}{1}}\approx 3.1$$

此时，10 个动作的置信估计都为 3.1，收益估计最大的动作不妨记为 $a'$，$a'$ 被选择，曲线上升

在第 12 步，其余 9 个动作被选择第二次之前的置信估计都为：

$$C_t(a)=2\sqrt{\frac{\log 12}{1}}\approx 3.15$$

只有 $a'$ 的置信估计是：

$$C_t(a')=2\sqrt{\frac{\log 12}{2}}\approx 1.11$$

此时 $a'$ 比其他动作的置信估计小约 2.04，只有当 $a'$ 之前的收益估计比其余动作的收益估计都大 2.04 才会继续选择 $a'$。而每个动作的收益期望采样自均值为 0 方差为 1 的高斯分布，2.04 相当于两个标准差

标准正态分布的函数曲线下 $68.268949\%$ 的面积在均值左右一个标准差内，$a'$ 的收益期望比其余动作的期望都大两个标准差的概率为 $((1-P)/2)^{10}\approx 1e^{-8}$。因此，$a'$ 的收益期望比其余动作的期望都大两个标准差的事件几乎不能发生。所以12步时会选择 $a'$ 以外的动作，曲线陡然下降。同理在 13-20 步，会按照收益估计从高往低，选择 $N_t(a)=1$ 的其他动作，所以后续若干步曲线震荡

对于 $c=1$：

$t=11$，$C_t(a)\approx 1.55$

$t=12$，$C_t(a)\approx 1.58$，$C_t(a')\approx 1.11$

根据上述思路，第 12 步时选择 $a'$ 的概率并不小，因此曲线并不会有骤降，尖峰不明显