探索与利用（Exploration and Exploitation）

吉建民

USTC
jianmin@ustc.edu.cn

2023 年 11 月 27 日

Used Materials

Disclaimer: 本课件大量采用了 Rich Sutton's RL class, David Silver's Deep RL tutorial 和其他网络课程课件，也采用了 GitHub 中开源代码，以及部分网络博客内容

Table of Contents

Model-based RL

- 目标：从经历 $\{S_1, A_1, R_2, ..., S_T\}$ 中估计一个模型 M_η
- 监督式学习的训练集：(X, y)

$$
\begin{aligned}
S_1, A_1 &\rightarrow R_2, S_2 \\
S_2, A_2 &\rightarrow R_3, S_3 \\
&\vdots \\
S_{T-1}, A_{T-1} &\rightarrow R_T, S_T
\end{aligned}
$$

- 从 s, a 学习 r 的过程是一个回归问题 (regression problem)
- 从 s, a 学习 s' 的过程是一个密度估计问题 (density estimation problem)
- 查表模型

$$
\hat{P}^a_{s,s'} = \frac{1}{N(s,a)} \sum_{t=1}^{T} \mathbf{1}(S_t, A_t, S_{t+1} = s, a, s')
$$

$$
\hat{R}^a_s = \frac{1}{N(s,a)} \sum_{t=1}^{T} \mathbf{1}(S_t, A_t = s, a) R_t
$$

Sample-based Planning

- 模型 M_η 只用来提供 sample（虚拟采样）

$$S_{t+1} \sim P_\eta(S_{t+1} \mid S_t, A_t)$$
$$R_{t+1} = R_\eta(R_{t+1} \mid S_t, A_t)$$

- 基于这些 sample 再采用 Model-free RL 来求解，如：
  - Monte-Carlo Control
  - Sarsa
  - Q-learning

Dyna 算法

- Real experience Sampled from environment (true MDP)

$$S' \sim P^a_{s,s'}$$
$$R = R^a_s$$

- Simulated experience Sampled from model (approximate MDP)

$$S' \sim P_\eta(S' \mid S, A)$$
$$R = R_\eta(R \mid S, A)$$

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Do forever:
(a) $s \leftarrow$ current (nonterminal) state
(b) $a \leftarrow \epsilon$-greedy(s,Q)
(c) Execute action a; observe resultant state, s', and reward, r
(d) $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$ ← direct RL
(e) $Model(s,a) \leftarrow s', r$ (assuming deterministic environment) ← model learning
(f) Repeat N times:
$s \leftarrow$ random previously observed state
$a \leftarrow$ random action previously taken in s
$s', r \leftarrow Model(s,a)$
$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$ ← planning

Simple Monte-Carlo Search

1. 给定一个模型 M_η 和一个模拟策略（simulation policy）π
2. 针对行为空间里的每一个行为 $a \in A$
   2.1 Simulate 从这个当前（实际）状态 S_t 开始模拟出 K 个 Episodes:
   $$\{s_t, a, R^k_{t+1}, \ldots, S^k_T\}^K_{k=1} \sim M_{\eta,\pi}$$
   2.2 Evaluate 使用平均收获（蒙特卡罗评估）来评估当前行为 a 的行为价值 $Q(s_t, a)$
   $$Q(s_t, a) = \frac{1}{K}\sum_{k=1}^{K} G_t$$
3. 选择一个有最大 $Q(s_t, a)$ 的行为作为实际要采取的行为 a_t：
   $$a_t = argmax_{a \in A} Q(s_t, a)$$

Monte-Carlo Tree Search: Evaluation

1. 给定一个模型 M_η 和一个初始的模拟策略（simulation policy）π
2. Simulate 从当前状态 s_t 开始，使用当前的模拟策略 π，模拟生成 K 个 Episodes：

$$\{s_t, A_t^k, R_{t+1}^k, \ldots, S_T^k\}_{k=1}^K \sim M_{\eta,\pi}$$

3. 构建一个包括所有个体经历过的状态和行为的搜索树
4. Evaluate (Every Visit) 对于搜索树种每一个状态行为对 (s, a)，计算从该 (s, a) 开始的所有完整 Episode 收获的平均值，以此来估算该状态行为对的价值 $Q(s, a)$

$$Q(s, a) = \frac{1}{N(s, a)} \sum_{k=1}^{K} \sum_{u=t}^{T} \mathbf{1}(S_u, A_u = s, a) G_u$$

5. 当搜索过程结束，也就是所有 s, a 的 Q 值得到更新后，选择搜索树中状态 s_t 对应最大 Q 值的行为

$$a_t = argmax_{a \in A} Q(s_t, a)$$

Monte-Carlo Tree Search: Simulation

- 在 MCTS 中，simulation policy π improves
- 搜索树并不包括整个状态行为对空间的 Q 值，因此在改进策略时要分情况对待
- 每个 simulation，更新 simulation policy π 分为两个阶段 (in-tree, out-of-tree)：
  - 树内确定性策略（Tree Policy）(improves): pick actions to maximise $Q(S, A)$
  - 树外默认策略（Default Policy）: pick actions randomly
- Repeat (each simulation)
  - Evaluate states $Q(S, A)$ by Monte-Carlo evaluation
  - Improve tree policy, e.g. by ϵ—greedy(Q)
- Monte-Carlo control applied to simulated experience
- Converges on the optimal search tree, $Q(S, A) \rightarrow q^*(S, A)$

TD Search

1. 从当前实际状态 s_t 开始，模拟一系列 Episodes，在这一过程中使用状态行为价值作为节点录入搜索树
2. 对搜索树内的每一个节点（状态行为对），估计其价值 $Q(s, a)$
3. 对于模拟过程中的每一步，使用 Sarsa 学习更新行为价值：

$$\Delta Q(S, A) = \alpha(R + \gamma Q(S', A') - Q(S, A))$$

4. 基于 Q 值，使用 ϵ-greedy 或其他探索方法来生成行为。

Dyna-2 算法

Algorithm 1 Episodic Dyna-2

procedure LEARN
  Initialise A, B ▷ Transition and reward models
  $\theta \leftarrow 0$ ▷ Clear permanent memory
  loop
    $s \leftarrow s_0$ ▷ Start new episode
    $\bar{\theta} \leftarrow 0$ ▷ Clear transient memory
    $z \leftarrow 0$ ▷ Clear eligibility trace
    SEARCH(s)
    $a \leftarrow \pi(s; \bar{Q})$ ▷ e.g. ϵ-greedy
    while s is not terminal **do**
      Execute a, observe reward r, state s'
      $(A, B) \leftarrow$ UPDATEMODEL(s, a, r, s')
      SEARCH(s')
      $a' \leftarrow \pi(s'; \bar{Q})$
      $\delta \leftarrow r + Q(s', a') - Q(s, a)$ ▷ TD-error
      $\theta \leftarrow \theta + \alpha(s, a)\delta z$ ▷ Update weights
      $z \leftarrow \lambda z + \phi$ ▷ Update eligibility trace
      $s \leftarrow s', a \leftarrow a'$
    end while
  end loop
end procedure

Dyna-2 算法

22: **procedure** SEARCH(s)
23: **while** time available **do**
24: $\bar{z} \leftarrow 0$ ▷ Clear eligibility trace
25: $a \leftarrow \bar{\pi}(s; \bar{Q})$ ▷ e.g. ϵ-greedy
26: **while** s is not terminal **do**
27: $s' \leftarrow A(s, a)$ ▷ Sample transition
28: $r \leftarrow B(s, a)$ ▷ Sample reward
29: $a' \leftarrow \bar{\pi}(s'; \bar{Q})$
30: $\bar{\delta} \leftarrow r + \bar{Q}(s', a') - \bar{Q}(s, a)$ ▷ TD-error
31: $\bar{\theta} \leftarrow \bar{\theta} + \bar{\alpha}(s, a)\bar{\delta}\bar{z}$ ▷ Update weights
32: $\bar{z} \leftarrow \bar{\lambda}\bar{z} + \bar{\phi}$ ▷ Update eligibility trace
33: $s \leftarrow s', a \leftarrow a'$
34: **end while**
35: **end while**
36: **end procedure**

$$Q(s, a) = \phi(s, a)^\top \theta$$

$$\bar{Q}(s, a) = \phi(s, a)^\top \theta + \bar{\phi}(s, a)^\top \bar{\theta}$$

Table of Contents

简介

- 探索（Exploration）和利用（Exploitation）的困局：Exploitation 是做出当前信息下的最佳决定，Exploration 则是尝试不同的行为继而收集更多的信息
- 最好的长期战略通常包含一些牺牲短期利益举措
- 通过搜集更多或者说足够多的信息使得个体能够达到宏观上的最佳策略。因此探索和利用是一对矛盾。
- Online decision-making involves a fundamental choice:
  - Exploitation: Make the best decision given current information
  - Exploration: Gather more information

Examples

- Restaurant Selection
  - Exploitation: Go to your favourite restaurant
  - Exploration: Try a new restaurant
- Online Banner Advertisements
  - Exploitation: Show the most successful advert
  - Exploration: Show a different advert
- Oil Drilling
  - Exploitation: Drill at the best known location
  - Exploration: Drill at a new location
- Game Playing
  - Exploitation: Play the move you believe is best
  - Exploration: Play an experimental move

Principles

几种基本的探索方法：

- 朴素探索 (Naive Exploration)：在 Greedy 搜索的基础上增加噪音以实现朴素探索（如 ϵ-greedy）
  - Add noise to greedy policy (e.g. ϵ-greedy)
- 乐观初始估计 (Optimistic Initialization)：优先选择当前被认为是最高价值的行为，除非新信息的获取推翻了该行为具有最高价值这一认知
  - Assume the best until proven otherwise
- 不确定优先 (Optimism in the Face of Uncertainty)：优先尝试不确定价值的行为
  - Prefer actions with uncertain values
- 概率匹配（Probability Matching)：根据当前估计的概率分布采样行为
  - Select actions according to probability they are best
- 信息状态搜索 (Information State Search)：将已探索的信息作为状态的一部分联合个体的状态组成新的状态，以新状态为基础进行前向探索
  - Lookahead search incorporating value of information

搜索分类

- 根据搜索过程中使用的数据结构，可以将搜索分为：
  - 依据状态行为空间的探索 (State-Action Exploration)：针对每一个当前的状态，以一定的算法尝试之前该状态下没有尝试过的行为
  - 参数化搜索（Parameter Exploration)：直接针对策略的函数近似，此时策略用各种形式的参数表达，探索即表现为尝试不同的参数设置
- 参数化搜索的优缺点：
  - 优点：得到基于某一策略的一段持续性的行为
  - 缺点：对个体曾经到过的状态空间毫无记忆，也就是个体也许会进入一个之前曾经进入过的状态而并不知道其曾到过该状态，不能利用已经到过这个状态这个信息

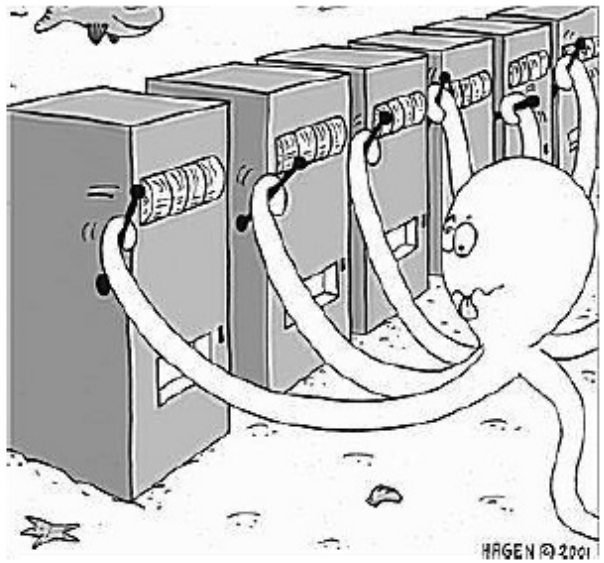
Table of Contents

与状态无关的多臂赌博机 (Multi-Armed Bandits)

- 多臂赌博机（下图）是由多个独立的单臂赌博机构成，赌博机相当于环境，个体拉下某一单臂赌博机的拉杆表示选择该赌博机，随后该赌博机会给出一个即时奖励 R
- 各个单臂赌博机之间是独立无关的，这等于说各单臂赌博机所给出的即时奖励值无关且各自的奖励分布也不同
- 此外，同一时刻只能拉下其中一个赌博机的拉杆，在某一个赌博机上先前拉杆得到的奖励与随后拉杆所能得到的奖励无关
- 另外，多臂赌博机每给出一个即时奖励，该 Episode 随即结束，因此一个 Episode 就是由一个行为和一个即时奖励构成，与状态无关

与状态无关的多臂赌博机 (Multi-Armed Bandits)

- A multi-armed bandit is a tuple $\langle \mathcal{A}, \mathcal{R} \rangle$
- $\mathcal{A}$ is a known set of m actions (or "arms")
- $\mathcal{R}^a(r) = \mathbb{P}[r|a]$ is an unknown probability distribution over rewards
- At each step t the agent selects an action $a_t \in \mathcal{A}$
- The environment generates a reward $r_t \sim \mathcal{R}^{a_t}$
- The goal is to maximise cumulative reward $\sum_{\tau=1}^{t} r_\tau$



Regret

为了方便描述问题，我们先给出几个定义：

- 这个问题不涉及状态本身，我们仿照状态价值（或状态行为价值）的定义来定义一个仅针对某一行为的价值 $Q(a)$：

$$Q(a) = E[r \mid a]$$

它的意思是一个行为的价值等于该行为能得到的即时奖励期望，即该行为得到的所有即时奖励的平均值

- 假如我们能够事先知道哪一个单臂赌博机能够给出最大即时奖励，那我们可以每次只选择对应的那个拉杆。如果用 V^* 表示这个最优价值，a^* 表示能够带来最优价值的行为，那么：

$$V^* = Q(a^*) = \max_{a \in A} Q(a)$$

Regret

- 事实上我们不可能事先知道拉下哪个拉杆能带来最高奖励，因此每一次拉杆获得的即时奖励可能都与最优价值 $V*$ 存在一定的差距，我们定义这个差距为后悔值 (regret):

$$l_t = E[V^* - Q(a_t)]$$

- 每做出一个行为，都会产生一个后悔值 l_t ，因此随着持续的拉杆行为，将所有的后悔值加起来，形成总后悔值 (total regret):

$$L_t = E\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right]$$

- 这样，最大化累计奖励的问题就可以转化为最小化总后悔值了: Maximise cumulative reward $\equiv$ minimise total regret

Counting Regret

现在我们从另一个角度重写总后悔值

- 定义计数（count）$N_t(a)$ 为到 t 时刻时已执行行动 a 的总次数
- 定义差距（gap）Δ_a 为最优价值 a^* 与行为 a 的价值之间的差

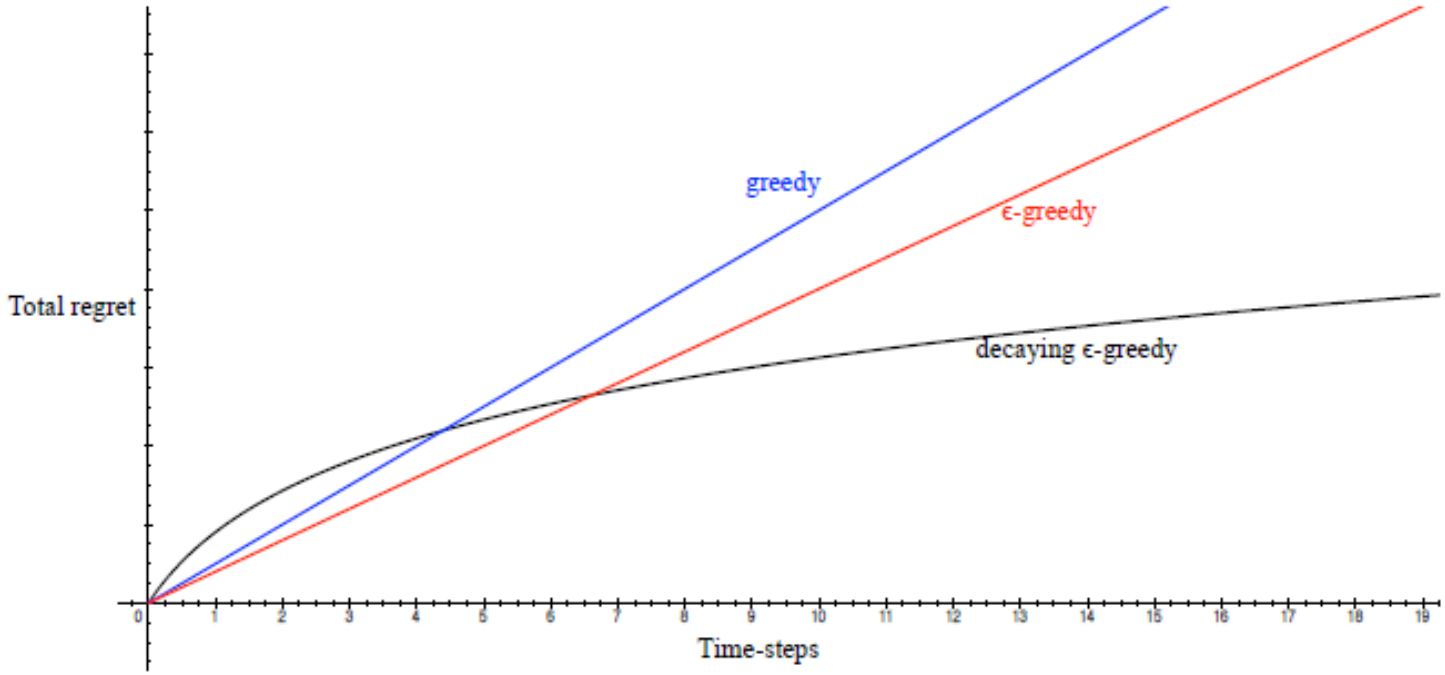
$$\Delta_a = V^* - Q(a)$$

- 那么总后悔值可以推导成（关于 gaps 和 counts 的函数）：

$$L_t = E\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right] = \sum_{a \in A} E[N_t(a)](V^* - Q(a)) = \sum_{a \in A} E[N_t(a)]\Delta_a$$

- 把总后悔值用计数和差距描述可以使我们理解到一个好的算法应该尽量减少那些差距较大的行为的次数。不过我们并不知道这个差距具体是多少，因为根据定义虽然最优价值 V^* 和每个行为的差距 Δ_a 都是静态的，但我们并不清楚这两者的具体数值，我们所能使用的信息就是每次行为带来的即时奖励 r
- A good algorithm ensures small counts for large gaps
- Problem: gaps are not known!

Linear or Sublinear Regret



- If an algorithm forever explores it will have linear total regret
- If an algorithm never explores it will have linear total regret
- Is it possible to achieve sublinear total regret?

$$y = a + bx^n, \qquad 0 < n < 1$$

Greedy Algorithm

- 我们使用每次的即时奖励来计算得到 t 时刻止某一行为的平均价值:

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t \mathbf{1}(a_t = a)$$

- 这个方法也叫蒙特卡罗评估，以此来近似该行为的实际价值 $Q(a)$:

$$\hat{Q}_t(a) \approx Q(a)$$

- The greedy algorithm selects action with highest value

$$a_t^* = argmax_{a \in A} \hat{Q}_t(a)$$

- Greedy can lock onto a suboptimal action forever
- Greedy has linear total regret

ϵ-Greedy Algorithm

- The ϵ-greedy algorithm continues to explore forever
  - With probability $1 - \epsilon$ select $a = argmax_{a \in A} \hat{Q}(a)$
  - With probability ϵ select a random action
- Constant ϵ ensures minimum regret

$$l_t = E[V^* - Q(a_t)] = (1 - \epsilon)(V^* - \hat{Q}(a)) + \frac{\epsilon}{|A|} \sum_{a \in A} \Delta_a$$

$$\geq \frac{\epsilon}{|A|} \sum_{a \in A} \Delta_a$$

- ϵ-greedy has linear total regret

Linear or Sublinear Regret

- 对于 ϵ-greedy 探索方法，总后悔值会呈线性增长，这是一个好的算法所不能接受的
- 这是因为每一个时间步，该探索方法有一定的几率选择最优行为，但同样也有一个固定小的几率采取完全随机的行为，如采取随机行为，那将一直会带来一定后悔值，如果持续以虽小但却固定的几率采取随机行为，那么总的后悔值会一直递增，导致呈现与时间之间的线性关系。类似的 softmax 探索方法与此类似
- 对于 greedy 探索方法，其总后悔值也是线性的，这是因为该探索方法的行为选择可能会锁死在一个不是最佳的行为上
- 现在问题是，能否找到一种探索方法，使用该探索方法时随着时间的推移其总后悔值增加得越来越少呢？答案是肯定的。下文将依次介绍几种较好的探索方法

乐观初始估计 (Optimistic Initialization)

- 理论上，这仍是总后悔值线性增加的探索方法，但是实际应用效果却非常好，因此放在这里介绍
- 其主要思想是在初始时给 $Q(a)$ 一个较高的价值，随后使用递增蒙特卡罗评估来更新该行为的价值（$N(a) > 0$）：

$$\hat{Q}_t(a_t) = \hat{Q}_{t-1} + \frac{1}{N_t(a_t)}(r_t - \hat{Q}_{t-1})$$

- 可以看出，某行为的价值会随着实际获得的即时奖励在初始设置的较高价值基础上不断得到更新，这在一定程度上达到了尽可能尝试所有可能的行为
- 但是该方法仍然可能锁死在次优行为上
- greedy + optimistic initialization has linear total regret
- ϵ-greedy + optimistic initialisation has linear total regret

衰减 ϵ-greedy (Decaying ϵ-greedy)

- 这是在 ϵ-greedy 的基础上做细小的修改：随着时间的延长，ϵ 值越来越小，采用 decay schedule $\epsilon_1, \epsilon_2, \ldots$
- 假设我们现在知道所有行为的差距 Δ_a。可设置为：如果一个行为的差距越小，则尝试该行为的机会越多；如果一个行为的差距越大，则尝试该行为的几率越小。数学表达如下：

$$c > 0$$

$$d = \min_{a|\Delta_a > 0} \Delta_a$$

$$\epsilon_t = \min\left\{1, \frac{c|A|}{d^2 t}\right\}$$

- 衰减 ϵ-greedy 能使得总的后悔值呈现出与时间步长的次线性（sublinear）关系：logarithmic asymptotic（对数渐进）total regret!
- 不过该方法需要事先知道每个行为的差距 Δ_a，实际上不可行
- Goal: find an algorithm with sublinear regret for any multi-armed bandit (without knowledge of R)

Lower Bound

- 在展开该方法之前，先思考一个问题：试想一下怎样的多臂赌博机问题容易选择最好行为？
- 如果一个多臂赌博机其中某个单臂一直给以较高的奖励，而另一个（或其它）单臂则一直给出相对较低的奖励，那么选择起来就容易得多了
- 相反，如果多个单臂给出的奖励变异程度较大，忽高忽低，而且多个单臂给出的奖励值有很多时候非常接近，那么选择一个价值高的行为可能就要花费很长时间了，也就是说这些单臂给出的奖励虽然类似，但其均值却差距较大
- 因此，可以通过比较两个单臂价值（均值）的差距 Δ_a 以及描述其奖励分布的相似程度的 KL 散度（相对熵）$KL(R_a||R_{a^*})$ 来判断总的后悔值下限

KL 散度 (Kullback–Leibler Divergence)

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \qquad \text{Entropy for } X = \{x_1, x_2, \ldots, x_n\}$$

$$H(x) = -\int p(x) \log p(x) \, \mathrm{d}x \qquad \text{Entropy for } x$$

$$KL(p||q) = -\sum_{i=1}^{n} p(x_i) \log \frac{q(x_i)}{p(x_i)}$$

$$= \sum_{i=1}^{n} p(x_i) \log \frac{p(x_i)}{q(x_i)} \qquad \text{KL divergence for } p(x) \text{ and } q(x)$$

$$KL(p||q) = \int p(x) \log \frac{p(x)}{q(x)} \, \mathrm{d}x \qquad \text{KL divergence for } p(x) \text{ and } q(x)$$

- KL 散度（相对熵，KL 距离）衡量的是相同事件空间里的两个概率分布的差异（相似）程度：$KL(p||q) \geq 0$，两分布相似度越高，KL 散度值越小
- 其物理意义是：在相同事件空间里，概率分布 $p(x)$ 的事件空间，若用概率分布 $q(x)$ 编码时，平均每个基本事件（符号）编码长度增加了多少比特
- 不存在其他比按照本身概率分布更好的编码方式了，所以 $KL(p||q)$ 始终大于等于 0 的

Lower Bound

- 针对多臂赌博机，有一个定理是这样说的：存在一个总后悔值的下限，没有哪一个算法能够做得比这个下限更好：

$$\lim_{t\to\infty} L_t \geq \log t \sum_{a|\Delta_a>0} \frac{\Delta_a}{KL(R_a||R_{a^*})}$$

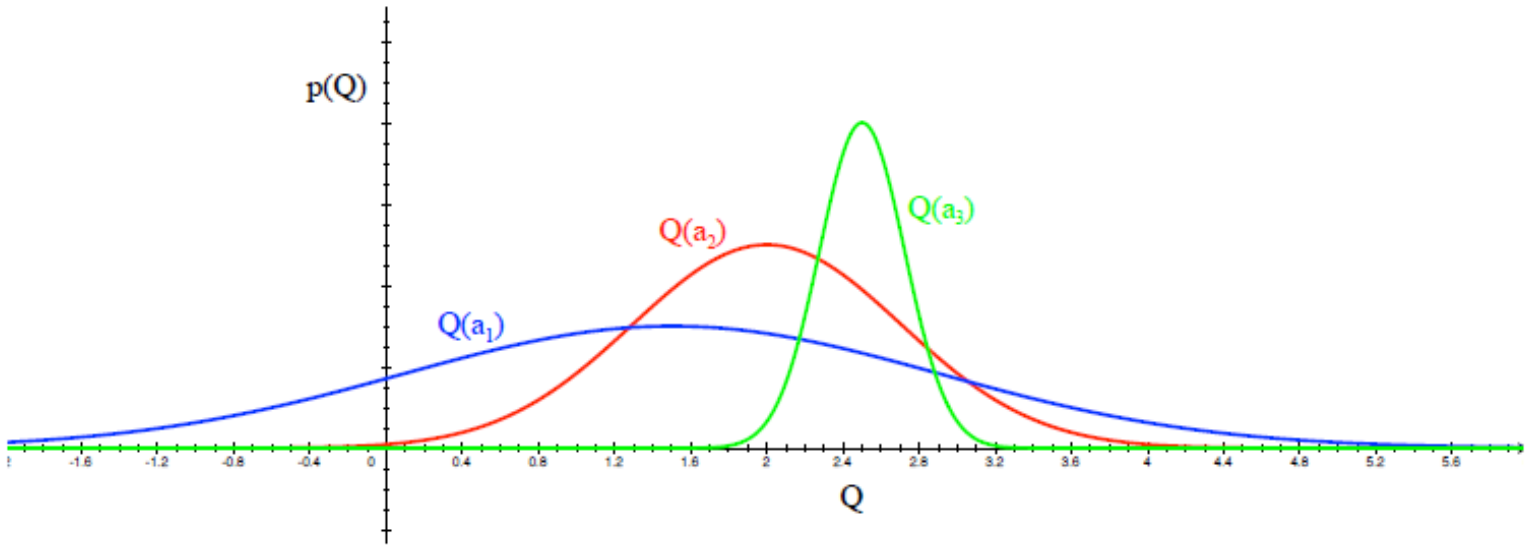
Theorem (Lai and Robbins)

Asymptotic total regret is at least logarithmic in number of steps

$$\lim_{t\to\infty} L_t \geq \log t \sum_{a|\Delta_a>0} \frac{\Delta_a}{KL(\mathcal{R}^a||\mathcal{R}^{a^*})}$$

不确定行为优先探索 (Optimism in the Face of Uncertainty)

现在我们来看看不确定行为优先搜索具体是怎么回事。想象一下现在由 3 个不同的单臂组成的多臂赌博机，现根据历史行为和奖励信息，绘制它们当前的奖励分布图



- The more uncertain we are about an action-value
- The more important it is to explore that action

不确定行为优先探索

- 特别注意这并不一定是它们真正服从的奖励分布，而是个体根据历史信息构建的一个经验分布
- 现在问题是，接下来该如何选择哪一个行为？正确的行为是采取蓝色的单臂，而不是绿色的单臂
- 理由是，蓝色的分布虽然其奖励均值比绿色的低（图中相应曲线最大 P 值对应的 Q 值），但其实际奖励分布范围较广，由于探索次数的限制，蓝色单臂对应的行为价值有不少的几率要比分布较窄的绿色单臂要高，也就是说蓝色单臂的行为价值具有较高的不确定性
- 因此我们需要优先尝试更多的蓝色单臂，以更准确地估计其行为价值，即尽可能缩小其奖励分布的方差

Upper Confidence Bounds

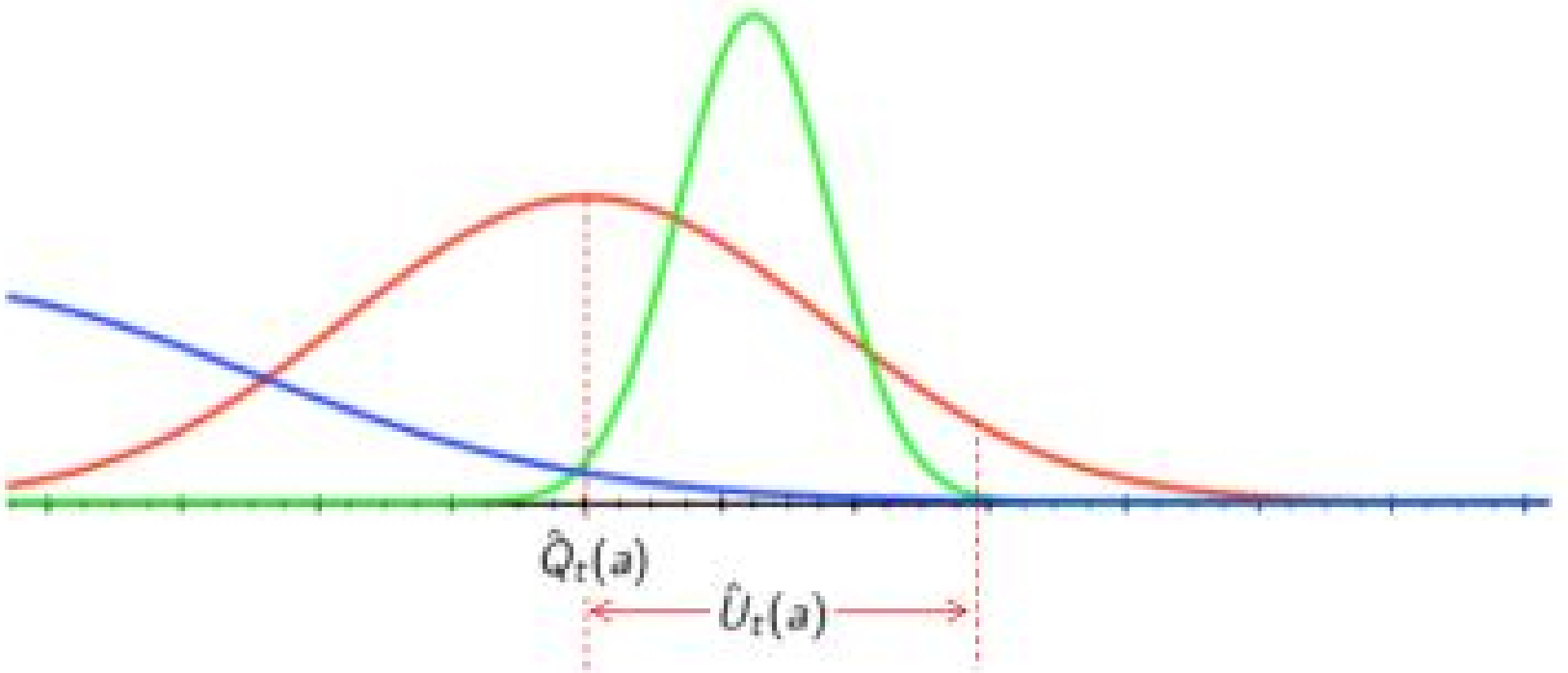
- 从上面的分析可以看出，单纯用行为的奖励均值作为行为价值的估计进而知道后续行为的选择因为采样数量的原因可能会不够准确，更加准确的办法是估计行为价值在一定可信度上的价值上限，比如可以设置一个行为价值的可信区间上限，将其作为指导后续行为的参考
- Estimate an upper confidence $\hat{U}_t(a)$ for each action value
- 如此一个行为的价值将有较高的可信度不高于某一个值 (with high probability)：

$$Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$$

- This depends on the number of times $N(a)$ has been selected
  - Small $N_t(a)$ implies large $\hat{U}_t(a)$ (estimated value is uncertain)
  - Large $N_t(a)$ implies small $\hat{U}_t(a)$ (estimated value is accurate)

Upper Confidence Bounds

下图针对红色行为的价值曲线标示了该不等式右侧的两个横坐标变量的位置（或范围）:



Upper Confidence Bounds

- 由此可以理解，当某一行为的计数较少时，该行为价值某一可信度上的价值上限将偏离均值较多；随着针对某一行为的奖励数据越来越多，该行为价值在某一可信度的上限将越来越接近均值
- 我们可以用置信区间上界 (Upper Confidence Bound, UCB) 来指导行为的选择，令：

$$a_t = argmax_{a \in A} \left(\hat{Q}_t(a) + \hat{U}_t(a) \right)$$

- 如果即时奖励分布是明确可知的，那么置信区间上界将比较容易根据均值进行求解。例如对于高斯分布 95% 的置信区间上界是均值与两倍标准差的和。而一般的对于分布未知的执行区间上界如何得到呢？

霍夫丁不等式（Hoeffding's Inequality）

Theorem (Hoeffding's Inequality)

Let $X_1, ..., X_t$ be i.i.d. random variables in [0,1], and let $\overline{X}_t = \frac{1}{\tau}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$\mathbb{P}\left[\mathbb{E}[X] > \overline{X}_t + u\right] \leq e^{-2tu^2}$$

- 该定理给出了位于区间 [0,1] 的两两随机变量其期望与均值之间满足的关系。结合该不等式，很容易得到:

$$P\left[Q(a) > \hat{Q}_t(a) + U_t(a)\right] \leq e^{-2N_t(a)U_t(a)^2}$$

该不等式描述的置信区间上限较之前描述的置信区间上限较弱，但也是实际可用的不等式

Calculating Upper Confidence Bounds

- 利用上面不等式得到一个特定可信度的置信区间上限就比较容易了。假定我们设定行为的价值有 p 的概率超过我们设置的可信区间上界，即令：

$$e^{-2N_t(a)U_t(a)^2} = p$$

- 那么可以得到：

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

- 随着时间步长的增加，我们逐渐减少 p 值，比如 $p = t^{-4}$，那么随着时间步长趋向无穷大，我们据此可以得到最佳行为
- 给出实际应用时 $U_t(a)$ 的公式：

$$U_t(a) = \sqrt{\frac{2\log t}{N_t(a)}}$$

UCB1

UCB1 Algorithm

$$a_t = argmax_{a \in A} \left(Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}} \right)$$

结论：由 UCB 算法设计的探索方法可以使得总后悔值满足对数渐进关系

Theorem

The UCB algorithm achieves logarithmic asymptotic total regret

$$\lim_{t \to \infty} L_t \leq 8 \log t \sum_{a | \Delta_a > 0} \Delta_a$$

Example: UCB vs. ϵ-Greedy On 10-armed Bandit

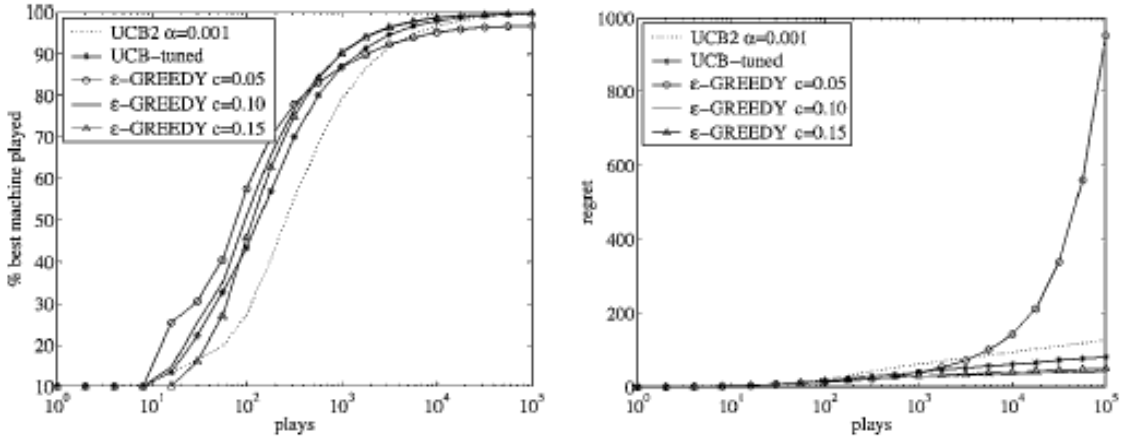


Figure 9. Comparison on distribution 11 (10 machines with parameters 0.9, 0.6, . . . , 0.6).

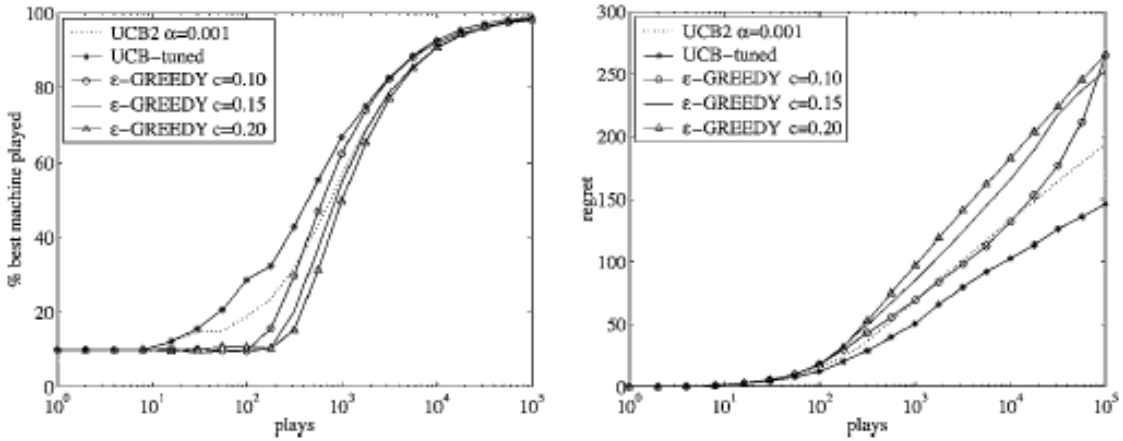


Figure 10. Comparison on distribution 12 (10 machines with parameters 0.9, 0.8, 0.8, 0.8, 0.7, 0.7, 0.7, 0.6, 0.6, 0.6).

Example: UCB vs. ϵ-Greedy On 10-armed Bandit

- 上页图展示了不同设置的 UCB 算法和不同设置的 ϵ-greedy 算法在 10 臂赌博机上的表现情况
- 该图由四个分图组成，上方与下方的不同在于 10-臂赌博机各臂参数设置不同，相当于描述了各个单臂的即时奖励的分布
- 左侧图描述的是各个算法随时间表现，右侧图描述的是总后悔值随时间的变化
- 事实表明，ϵ-greedy 算法如果参数调整得当，可以表现的很好，反之则可能是灾难
- UCB 在没有掌握任何信息的前提下也能做得很好

Bayesian Bandits

- 前面讲到 UCB 算法可以得到较好的效果，UCB 算法仅仅依靠统计各个行为的奖励均值和出现的次数，并在此基础上设定一定的置信区间上界，指导行为选择
- 如果已知先验的 reward distribution，则可以计算出更好的 bound estimation
- Bayesian bandits: 利用先验概率 $P[Q]$
  - 例如 $P[Q \mid w]$ 为独立高斯分布，$w = [\mu_1, \sigma_1^2, \ldots, \mu_k, \sigma_k^2]$，每个 action a_i 对应一个高斯分布 $\mathcal{N}(\mu_i, \sigma_i^2)$
- 给定历史信息 h_t

$$h_t = a_1, r_1, \ldots, a_{t-1}, r_{t-1}$$

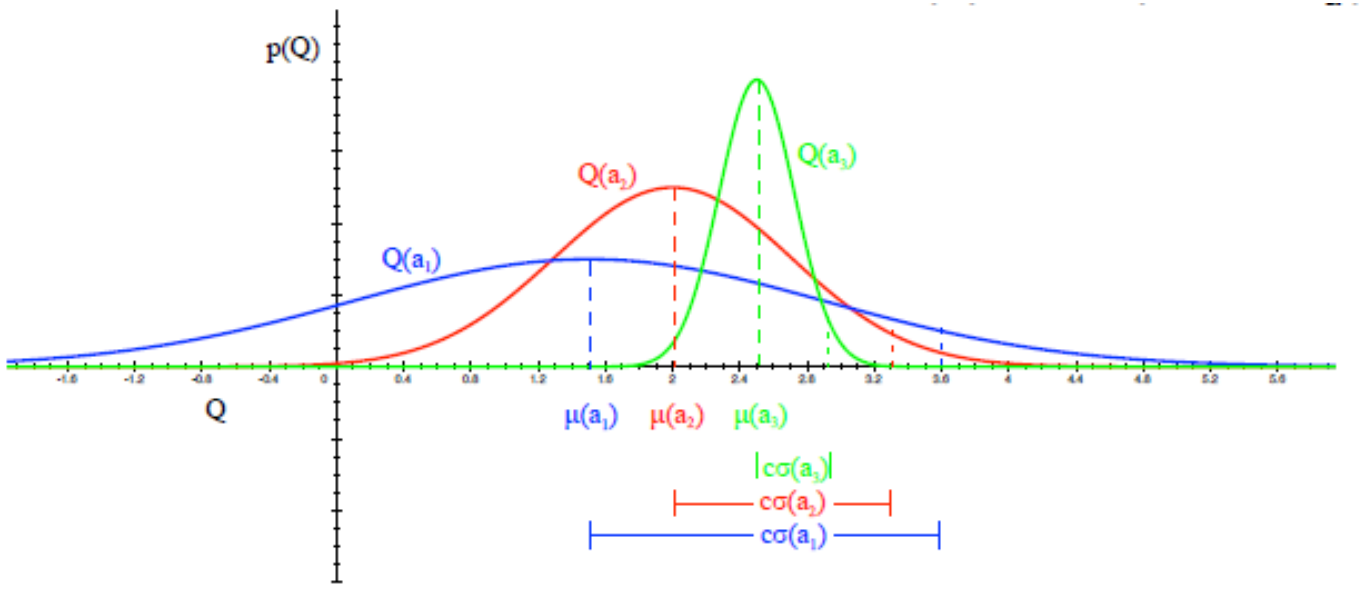
计算后验概率 $P[Q \mid h_t]$

$$P[Q \mid h_t] = P[Q \mid w] P[w \mid h_t]$$

- 利用后验概率来指导 exploration
  - Upper confidence bounds (Bayesian UCB)
  - Probability matching (Thompson sampling)
- 先验概率越准确，效果越好

Bayesian UCB Example: Independent Gaussians

- 假设各单臂赌博机服从相互独立的高斯分布，可以用每一个单臂赌博机的均值和标准差参数化整体奖励分布：



- 使用 Bayesian law 来计算 $P[w \mid h_t]$
- 再计算 action-values 后验概率
  $P[Q(a) \mid h_t] = P[Q(a) \mid w]P[w \mid h_t]$
- 估计后验概率的 upper confidence $U_t(a) = c\sigma(a)$，其中 $\sigma(a)$ 为 $P(Q(a) \mid w)$ 的标准差
- 依据最大的 $Q_t(a) + c\sigma(a)/\sqrt{N(a)}$ 选择行动

Bayesian UCB Example: Independent Gaussians

- 关于 $Q(a)$ 的先验知识 $P[\mu_a, \sigma_a^2]$，给定历史 h_t，计算后验概率 $P[\mu_a, \sigma_a^2 \mid h_t]$
- Compute Gaussian posterior over μ_a and σ_a^2 (by Bayes law)

$$P[\mu_a, \sigma_a^2 \mid h_t] \propto P[\mu_a, \sigma_a^2] \prod_{t|a_t=a} \mathcal{N}(r_t; \mu_a, \sigma_a^2)$$

注：

$$P[\mu_a, \sigma_a^2 \mid h_t] = \frac{P[\mu_a, \sigma_a^2] P[h_t \mid \mu_a, \sigma_a^2]}{\int P[\mu_a, \sigma_a^2] P[h_t \mid \mu_a, \sigma_a^2] \, \mathrm{d}\mu_a \, \mathrm{d}\sigma_a^2}$$

- 选择均值和一定比例的标准差之和来作为 UCB 算法中的置信区间上限，即依据下式选择后续行为：

$$a_t = argmax_{a \in A} \left(\mu_a + \frac{c\sigma_a}{\sqrt{N(a)}} \right)$$

注：GP-UCB (Gaussian Processes) selection rule: β_t 与 t 和 $|X|$ 有关

$$x_t = argmax_{x \in X} \left(\mu_{t-1}(x) + \sqrt{\beta_t} \sigma_{t-1}(x) \right)$$

概率匹配 (Probability Matching)

- 概率匹配的想法先估计每一个行为可能是最佳行为的概率，然后依据这个概率来选择后续行为

$$\pi(a \mid h_t) = P\left[Q(a) > Q(a'), \forall a' \neq a \mid h_t\right]$$

- 该算法背后隐藏的思想是：越不确定价值的行为有着越高的几率被选择，这种被选择的目的是通过采样减少其不确定性，进而调整后续策略
- Can be difficult to compute analytically from posterior

$$\begin{aligned} P(\theta \mid x) &= \frac{P(x \mid \theta) P(\theta)}{P(x)} \\ &= \frac{P(x \mid \theta) P(\theta)}{\int P(x \mid \theta') P(\theta') \, \mathrm{d}\theta'}. \end{aligned}$$

汤普森采样（Thompson Sampling）

- Thompson sampling 算法是基于 Probability Matching 的一种实际可行的算法，该算法实现起来非常简单，同时也是一个非常接近总后悔值对数关系的一个算法

$$\begin{aligned}\pi(a \mid h_t) &= P[Q(a) > Q(a'), \forall a' \neq a \mid h_t] \\ &= E[\mathbf{1}(a = argmax_{a \in A} Q(a)) \mid h_t]\end{aligned}$$

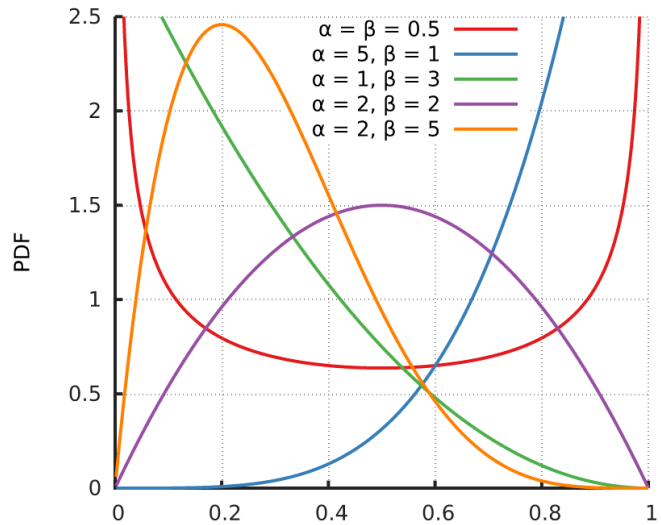
- 该算法的步骤如下：
  1. 基于历史信息构建各单臂的奖励分布估计，利用 Bayes Law 计算后验概率 $P[Q \mid h_t]$
     - Use Bayes law to compute posterior distribution $P[Q \mid h_t]$
  2. 依次从每一个分布中采样得到所有行为对应即时奖励的采样值
     - Sample an action-value function $Q(a)$ from posterior
  3. Select action maximising value on sample, $a_t = argmax_{a \in A} Q(a)$
- 该算法的采样过程中利用到了历史信息得到的分布，同时行为得到的真实奖励值将更新该行为的分布估计
- Thompson sampling achieves Lai and Robbins lower bound!

Beta 分布

- 介绍汤普森采样之前，先介绍一个分布：beta 分布。beta 分布可以看作一个概率的概率分布，当你不知道一个东西的具体概率是多少时，它可以给出了所有概率出现的可能性大小

$$f(x;\alpha,\beta) = \text{constant}\cdot x^{\alpha-1}(1-x)^{\beta-1} = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}\,\mathrm{d}u}$$

- beta 分布有两个控制参数：α 和 β 。先来看下几个 beta 分布的概率密度函数的图形：



Beta 分布

- beta 分布图形中的 x 轴取值范围是 (0,1)，可以看成是概率值，参数 α 和 β 可以控制图形的形状和位置：
  - $\alpha + \beta$ 的值越大，分布曲线越窄，也就是越集中
  - $\frac{\alpha}{\alpha+\beta}$ 的值是 beta 分布的均值（期望值），它的值越大，beta 分布的中心越靠近 1，否则越靠近 0
- beta 分布和 Bandit 算法有什么关联呢？实际上，每个臂是否产生收益的概率 P 的背后都对应一个 beta 分布。我们将 beta 分布的 α 参数看成是推荐后用户的点击次数，β 参数看成是推荐后用户未点击的次数。
- 一个臂选 n 次（伯努利试验），其中成功 s 次，它成功的概率是某个未知的数 $p \in [0,1]$; s 发生的概率为一个二项分布（Binomial distribution，单次为伯努利分布）

$$f(x) = p^x(1-p)^{1-x},\ x \in \{0,1\}, \qquad P(s) = \binom{n}{s} p^s (1-p)^{n-s}$$

相应的共轭先验（conjugate prior）为 beta 分布，其中参数 α 和 β 反映已知信息

$$P(p) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}\,\mathrm{d}u}$$

汤普森采样

- 汤普森算法的流程：
  - 每个臂都维护一个 beta 分布的参数，获取每个臂对应的参数 α 和 β，然后使用 beta 分布生成随机数
  - 选取生成随机数最大的那个臂作为本次结果，进行摇臂
  - 摇臂以后，如果有收益将对应臂的 α 加 1，否则 β 加 1
- 在实际的推荐系统中，需要为每个用户保存一套参数，假设有 m 个用户，n 个臂（选项，可以是物品，可以是策略），每个臂包含 α 和 β 两个参数，所以最后保存的参数的总个数是 $2mn$
- 汤普森采样算法有效性的直观解释：
  - 当尝试的次数较多时，即每个臂的 $\alpha+\beta$ 的值都很大，这时候每个臂对应的 beta 分布都会很窄，也就是说，生成的随机数都非常接近中心位置，每个臂的收益基本确定了
  - 当尝试的次数较少时，即每个臂的 $\alpha+\beta$ 的值都很小，这时候每个臂对应的 beta 分布都会很宽，生成的随机数有可能会比较大，增加被选中的机会
  - 当一个臂的 $\alpha+\beta$ 的值很大，并且 $\alpha/(\alpha+\beta)$ 的值也很大，那么这个臂对应的 beta 分布会很窄，并且中心位置接近 1，那么这个臂每次选择时都很占优势

信息价值 (Value of Information)

- 本小节尝试从信息的角度来讲解另外一种探索方法
- 探索之所以有价值是因为它会带来更多的信息，那么能否量化被探索信息的价值和探索本身的开销，以此来决定是否有探索该信息的必要呢？这就涉及到信息本身的价值
- Can we quantify the value of information?
  - How much reward a decision-maker would be prepared to pay in order to have that information, prior to making a decision
  - Long-term reward after getting information - immediate reward
- Information gain is higher in uncertain situations
- Therefore it makes sense to explore uncertain situations more
- If we know value of information, we can trade-off exploration and exploitation optimally

信息价值

- 先打个比方，对于一个 2 臂赌博机。假如个体当前对行为 a_1 的价值有一个较为准确的估计，比如是 100 镑，这意味着执行行为 a_1 可以得到的即时奖励的期望，个体虽然对于行为 a_2 的价值也有一个估计，假如说是 70 镑，但这个数字非常不准确，因为个体仅执行了非常少次的行为 a_2。那么获取“较为准确的行为 a_2 的价值”这条信息的价值有多少呢？这取决于很多因素，其中之一就是个体有没有足够多的行为次数来获取累计奖励，假如个体只有非常有限的行为次数，那么个体可能会倾向于保守的选择 a_1 而不去通过探索行为 a_2 而得到较为准确的行为 a_2 的价值。因为探索本身会带来一定几率的后悔。相反如果个体有数千次的行为次数，那么得到一个更准确的行为 a_2 的价值就显得非常必要了，因为即使通过一定次数的探索 a_2，后悔值也是可控的。而一旦得到的行为 a_2 的价值超过 a_1，则将影响后续数千次的行为的选择

Information State Space

- 为了能够确定信息本身的价值，可以设计一个 MDP，将信息作为 MDP 的状态构建对其价值的估计：

$$\tilde{M} = \langle \tilde{S}, A, \tilde{P}, R, \gamma \rangle$$

- At each step there is an information state $\tilde{s}$
  - $\tilde{s}$ is a statistic of the history, $\tilde{s} = f(h_t)$
  - summarising all information accumulated so far
- Each action a causes a transition to a new information state $\tilde{s}'$ (by adding information), with probability $\tilde{P}^a_{\tilde{s},\tilde{s}'}$

例子：Bernoulli Bandits

- 现在考虑一个即时奖励服从伯努利分布的赌博机 (奖励结果要么 1 要么 0)

$$R_a = \mathcal{B}(\mu_a)$$

上式中 μ_a 是拉下一个拉杆能够获得奖励 1 的概率，$1 - \mu_a$ 为奖励为 0 的概率

$$f(k; p) = p^k(1-p)^{(1-k)} \quad k \in \{0, 1\}$$

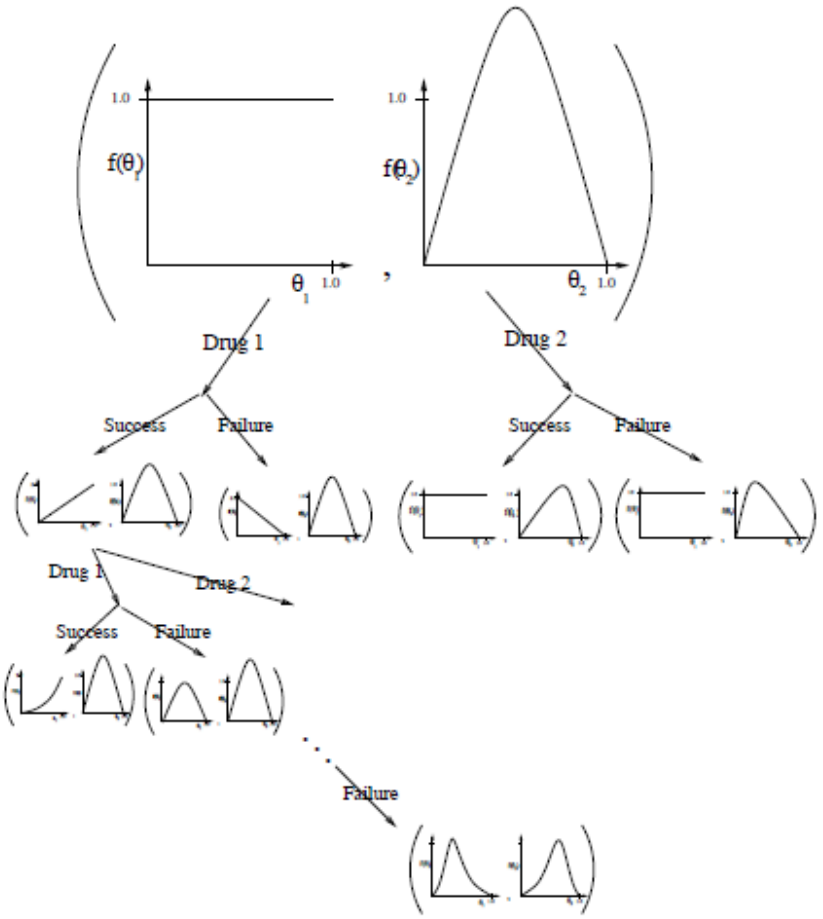
- 对于由服从该分布的单臂赌博机构成的 2 臂赌博机，我们的目标就是找到哪一个赌博机有较高的 μ_a ，这样就能每次选择那个较高 μ_a 的赌博机，以此来最小化总后悔值
- 对于这样由 2 个行为组成的行为空间，可以将信息状态描述为 $s\langle\alpha, \beta\rangle$，其中 α_a 为执行行为 a 得到奖励为 0 的次数，其中 β_a 执行行为 a 得到奖励为 1 的次数。该信息状态的内容记载了所有历史信息
- 事实上这样的信息状态构建得到的 MDP 是一个规模非常庞大的 MDP，解决它需要使用之前介绍的函数近似

Solving Information State Space Bandits

- We now have an infinite MDP over information states
- This MDP can be solved by reinforcement learning
- Model-free reinforcement learning
- Bayesian model-based reinforcement learning
  - This approach is known as Bayes-adaptive RL
  - Finds Bayes-optimal exploration/exploitation trade-off with respect to prior distribution

Bayes-Adaptive Bernoulli Bandits

- Start with $Beta(\alpha_a, \beta_a)$ prior over reward function $\mathcal{R}^a$
- Each time a is selected, update posterior for $\mathcal{R}^a$
  - $Beta(\alpha_a + 1, \beta_a)$ if $r = 0$
  - $Beta(\alpha_a, \beta_a + 1)$ if $r = 1$
- This defines transition function $\tilde{\mathcal{P}}$ for the Bayes-adaptive MDP
- Information state $\langle \alpha, \beta \rangle$ corresponds to reward model $Beta(\alpha, \beta)$
- Each state transition corresponds to a Bayesian model update



Bayes-Adaptive Bernoulli Bandits

- 右图是较为直观的信息状态转移图
- 我们从药物实验的角度来解释，行为 a_1 表示个体对某一患者使用药物 a_1，假设疗效服从伯努利分布，它与一个“金标准”比较可以是只有 2 个：优于金标准和不优于金标准
- 最初个体对于药物 a_1 的疗效一无所知，即它并不清楚药物 a_1 优于金标准疗效的概率分布，也就是说从 0-1 之间任意一个概率都是均一的（图中上方括号左侧的水平线图）
- 而对于药物 a_2，个体通过实验了解到其平均有 50% 的概率优于金标准疗效（途中上方括号右侧的曲线图）
- 每一个信息状态对应的两个曲线图都是对当前状态所有历史信息的描述，也就是各药物疗效优于或不优于金标准的计数
- 现在前向观察这个搜索树，比如现在选择了行为 a_1，发现行为 a_1 的疗效要优于金标准，那么我们修改对行为 a_1 疗效的计数，同时反映在对应的图上。如果随后继续使用 a_1 但发现疗效不好，则目前对于 a_1 来说一次疗效好，一次不好。则总体呈现出 50% 类似于药物 a_2 的疗效

Bayes-Adaptive MDP for Bernoulli Bandits

- 下图是一个纯粹数字形式描述的信息状态和转移图，对于每一个信息状态，要统计两种药物疗效分别优于和不优于金标准的次数。这就相当于是一个 Bayes-Adaptive MDP
- 理论上由于信息状态空间规模太大，精确的求解是很难的。一个解决方案是利用 Simulation-Based Search

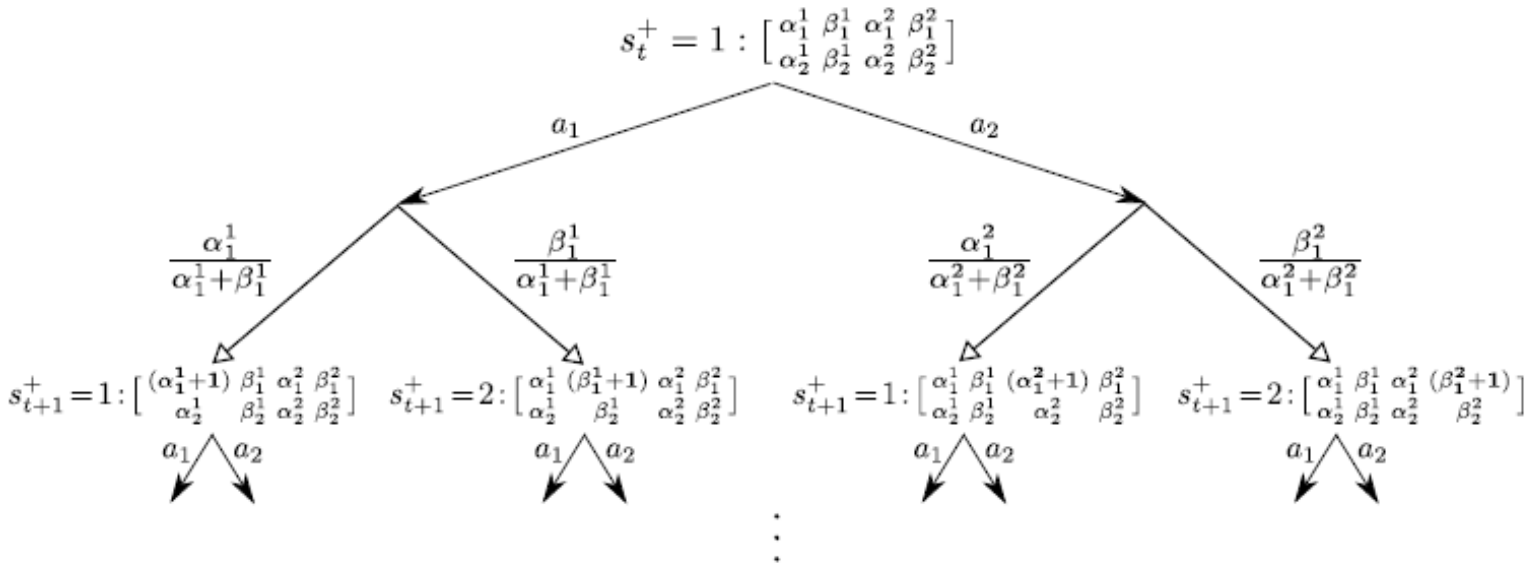


Table of Contents

Contextual Bandits

- A contextual bandit is a tuple $\langle \mathcal{A}, \mathcal{S}, \mathcal{R} \rangle$
- $\mathcal{A}$ is a known set of actions (or "arms")
- $\mathcal{S} = \mathbb{P}[s]$ is an unknown distribution over states (or "contexts")
- $\mathcal{R}_s^a(r) = \mathbb{P}[r|s,a]$ is an unknown probability distribution over rewards
- At each step t
  - Environment generates state $s_t \sim \mathcal{S}$
  - Agent selects action $a_t \in \mathcal{A}$
  - Environment generates reward $r_t \sim \mathcal{R}_{s_t}^{a_t}$
- Goal is to maximise cumulative reward $\sum_{\tau=1}^{t} r_\tau$

考虑状态的影响，初始状态分布 $P[s]$，不涉及状态转移

Linear Regression

- Action-value function is expected reward for state s and action a

$$Q(s, a) = E[r \mid s, a]$$

- Estimate value function with a linear function approximator

$$Q_\theta(s, a) = \phi(s, a)^\top \theta \approx Q(s, a)$$

- Estimate parameters by least squares regression

$$A_t = \sum_{\tau=1}^{t} \phi(s_\tau, a_\tau)\phi(s_\tau, a_\tau)^\top$$

$$b_t = \sum_{\tau=1}^{t} \phi(s_\tau, a_\tau) r_\tau$$

$$\theta_t = A_t^{-1} b_t$$

Linear Upper Confidence Bounds

- Least squares regression estimates the mean action-value $Q_\theta(s, a)$
- But it can also estimate the variance of the action-value $\sigma^2_\theta(s, a)$
  - the uncertainty due to parameter estimation error
- Add on a bonus for uncertainty, $U_\theta(s, a) = c\sigma$
  - define UCB to be c standard deviations above the mean
- 最小二乘本身就是对 θ 的高斯估计，利用 $\theta_t = A_t^{-1} b_t$ 计算出对均值的估计 θ_t；θ_t 的协方差矩阵为 $A_t^{-1}\sigma^2$，σ^2 为噪音的方差，假设为 I
- $Q_\theta(s, a) = \phi(s, a)^\top \theta$ 是线性变换，$Q_\theta(s, a)$ 也是高斯分布，其均值为 $Q_\theta(s, a)$，方差为 $\sigma^2_\theta(s, a) = \phi(s, a)^\top A^{-1} \phi(s, a)$
- Upper confidence bound is

$$Q_\theta(s, a) + c\sqrt{\phi(s, a)^\top A^{-1} \phi(s, a)}$$

- Select action maximising upper confidence bound

$$a_t = argmax_{a \in A} \left(Q_\theta(s_t, a) + c\sqrt{\phi(s_t, a)^\top A_t^{-1} \phi(s_t, a)} \right)$$

Properties of Gaussians

- Multivariate case

$$\left.\begin{array}{l} X \sim N(\mu, \Sigma) \\ Y = AX + B \end{array}\right\} \Rightarrow \quad Y \sim N(A\mu + B, A\Sigma A^T)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \Sigma_1) \\ X_2 \sim N(\mu_2, \Sigma_2) \end{array}\right\} \Rightarrow p(X_1) \cdot p(X_2) \sim N\left(\frac{\Sigma_2}{\Sigma_1 + \Sigma_2}\mu_1 + \frac{\Sigma_1}{\Sigma_1 + \Sigma_2}\mu_2, \quad \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}}\right)$$

(where division "–" denotes matrix inversion)

- We **stay Gaussian** as long as we start with Gaussians and perform only **linear transformations**

Table of Contents

Exploration/Exploitation Principles to MDPs

The same principles for exploration/exploitation apply to MDPs

- Naive Exploration
- Optimistic Initialization
- Optimism in the Face of Uncertainty
  - Upper Confidence Bounds: Model-Free RL
  - Bayesian Model-Based RL
- Probability Matching
  - Thompson Sampling: Model-Based RL
- Information State Search
  - Bayes Adaptive MDPs

Optimistic Initialisation: Model-Free and Model-Based RL

- Model-Free RL
  - 初始化 $Q(s, a)$ 为 $\frac{r_{max}}{1-\gamma}$
  - 之后再用 model-free RL 算法，如：Monte-Carlo control, Sarsa, Q-learning
- Model-Based RL
  - Construct an optimistic model of the MDP
  - Initialise transitions to go to heaven
    - 直接后继状态就是最终状态，同时 reward 为 r_{max}
  - 之后再用 planning 算法求解 optimistic model，如：policy iteration, value iteration, tree search
- 系统性的鼓励探索

小结

- Several principles for exploration/exploitation
  - Naive methods such as ϵ-greedy
  - Optimistic initialisation
  - Upper confidence bounds
  - Probability matching
  - Information state search
- Each principle was developed in bandit setting
- Same principles also apply to MDP setting