Reinforcement learning (2)

policy search

motivation & goal:

learn policies that maximize rewards, not the values that predict them

目的不是精确拟合Q值，而是确保action的排序是正确的；（q-value是手段，不是目的）

solution:

start with an ok solution (e.g. Q-learning) then fine-tune by hill climbing on feature weights

- policy search

  - For this reason, policy search methods often use a stochastic policy representation $\pi_\theta(s, a)$, which specifies the probability of selecting action a in state s.

  - One popular representation is the **softmax function**:

  $$\pi_\theta(s, a) = e^{\hat{Q}_\theta(s,a)} / \sum_{a'} e^{\hat{Q}_\theta(s,a')}.$$

  then to get the best policy, we would want to optimize

  $$\arg\max_\theta E[\sum_{t=0}^{H} R(s_t) \mid \pi_\theta]$$

  将reward重新表示为：

  $$R(\tau) = \sum_{i=0}^{i=H} R(s_i, a_i, s_{i+1})$$

  - So we want to find :

  $$\arg\max_\theta V(\theta) = \sum_\tau P(\tau; \theta) R(\tau)$$

  注意 θ是策略参数，可以理解为就是前面的 feature的权重w；

  stop criterion就是 V(θ) 导数=0时，只需下面的式子 = 0：

$$
\begin{aligned}
\nabla_\theta V(\theta) &= \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau) \\
&= \sum_\tau \nabla_\theta P(\tau;\theta)R(\tau) \\
&= \sum_\tau \frac{P(\tau;\theta)}{P(\tau;\theta)} \nabla_\theta P(\tau;\theta)R(\tau) \\
&= \sum_\tau P(\tau;\theta) \frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)} R(\tau) \\
&= \sum_\tau P(\tau;\theta) \nabla_\theta \log P(\tau;\theta)R(\tau)
\end{aligned}
$$

这个形式是期望，可以用平均值来（无偏, unbiased）估计：

$$
\nabla_\theta V(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau^i;\theta)R(\tau^i)
$$

very good result, NOTE:

1. this is valid even when:

   R is discontinous or unknown (R也是通过采样在与环境的交互中获得的)

   sample space (of paths) is discrete (只需要该策略下采样几个路径，无论路径空间是离散的还是连续的)

2. the gradient tries to:

   increase probability of paths with positive R

   decrease probability of paths with negative R

   策略会朝着能够带来高回报的方向调整

进一步展开：

$$
\begin{aligned}
\nabla_\theta \log P(\tau;\theta) &= \nabla_\theta \log \prod_{t=0}^{H} P(s_{t+1}|s_t,a_t)\cdot\pi_\theta(s_t,a_t) \\
&= \nabla_\theta\Big[\sum_{t=0}^{H}\log P(s_{t+1}|s_t,a_t) + \sum_{t=0}^{H}\log\pi_\theta(s_t,a_t)\Big] \\
&= \nabla_\theta \sum_{t=0}^{H}\log\pi_\theta(s_t,a_t) \\
&= \sum_{t=0}^{H}\nabla_\theta \log\pi_\theta(s_t,a_t)
\end{aligned}
$$

发现连transition函数P也不需要知道，π 是 Q-value 求softmax

- 以上所有推导的summary（actually, the REINFORCE algorithm）：

Unbiased estimate of the gradient

$$
\nabla_\theta V(\theta) \approx \hat{g} = \frac{1}{m}\sum_{i=1}^{m}\nabla_\theta \log P(\tau^i;\theta)R(\tau^i)
$$

where

$$
\nabla_\theta \log P(\tau^i;\theta) = \sum_{t=0}^{H}\nabla_\theta \log\pi_\theta(s_t^i,a_t^i)
$$

Unbiased means

$$
E(\hat{g}) = \nabla_\theta V(\theta)
$$

这个用平均值估计期望的方法可能方差很大，解决的方式包括：

REINFORCE算法，

REINFORCE (Williams)

1.Choose an initial policy π_θ, e.g. a neural network

2.Gather a collection of trajectories τ—sequences of states, actions, and rewards —by executing the present policy in the environment.

3.Determine Returns $R(\tau)$, the total of the discounted rewards for each trajectory.

4.Calculate the Policy Gradients $\nabla_\theta \log P(\tau;\theta)$

5.Calculate $\hat{g} \approx \nabla_\theta V(\theta)$

6.Update the policy parameters θ in π_θ

7.Repeat from step 2.

这样的估计方法可能导致方差很大，solution是使用更稳定的优化算法如 PPO，等。

partially observable MDPs

partial observable environment 部分可观测环境

无法直接观测到自己的状态/位置，维护一个belief state 信念状态，表示自己目前状态的概率分布；

filtering：观测到一个事实之后对belief state进行更新

Partially Observable Markov Decision Processes

- A Markov decision process (MDP):
  - A set of states s ∈ S
  - A set of actions (per state) A
  - A model T(s,a,s')
  - A reward function R(s,a,s')
  - An observation function P(o | s) - sensor model
- Looking for an optimal policy π(b) where b is the belief state

belief state更新函数：

$$b''(s') = \alpha P(e|s') \sum_s P(s'|s,a) b'(s)$$

其中 $\sum_s P(s'|s,a)b'(s)$ 是通过action a 能够转移到 s' 的概率，α 是归一化系数；

- POMDP can be turned into MDPs, where the states of the MDP are the belief states

  具体转化的formula：

- Transition function

$$\begin{aligned} P(b'|b,a) &= \sum_e P(b'|e,a,b)P(e|a,b) \\ &= \sum_e P(b'|e,a,b)\sum_{s'} P(e|s')\sum_s P(s'|s,a)b(s) \end{aligned}$$

- (Expected) Reward function

$$\rho(b,a) = \sum_s b(s)\sum_{s'} P(s'|s,a)R(s,a,s')$$

bandits

- what is the bandit problem

  多臂老虎机问题：需要在有限时间内选择多臂中的一个，每个臂都有一个未知的概率分布，代理的目标是最终最大化收益；

  trade-off between exploration & exploitation （探索vs.利用）

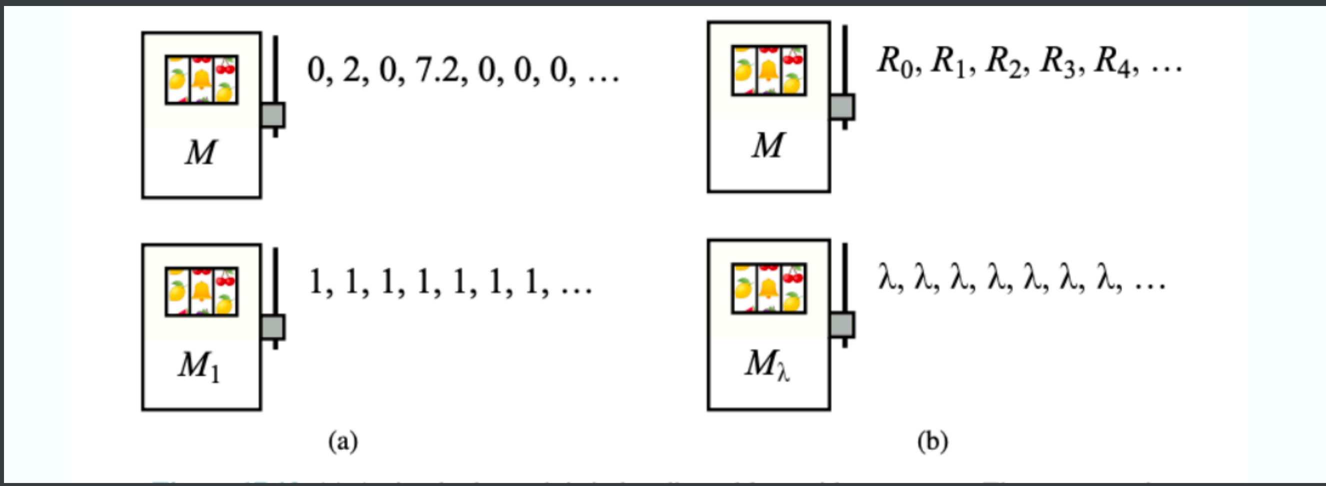
- application：推荐系统（未知点击率分布），药物临床试验（未知药效分布）
- bandits -> MDP which consists of several MRPs

  MRP是一个简化的MDP，每次只有一个动作可选（即“马尔可夫链”）

  1. each arm i is an MRP M_i
  2. then overall bandit problem is an MDP where the state space is $S = S_1 \times S_2 \times \ldots \times S_n$ (笛卡尔积)，每次的action只能选一个arm操作（对于一个特定的arm，action总是固定的），transition model对于被操作的bandit进行状态更新。并有 reward discount γ

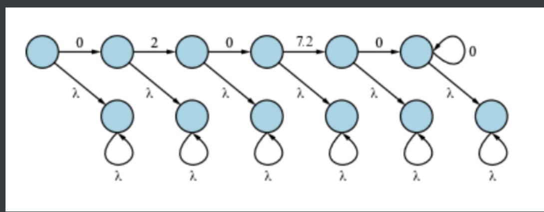
- gittins index

  两个臂 M 和 M_λ，奖励序列如下所示：



那么根据上面的建模，这个问题可以等价成如下的MDP：

（注意，这个例子很特殊，对一个arm执行了其相应的action后，一定从前一个状态转移到后一个状态；而对一般的bandit problem是不一定的，有可能还在当前状态）



对于这个模型，可由gittins index（每单位时间内可获得最大收益）决定如何切换arm，

- Gittins index $\lambda = \max\limits_{T>0} \dfrac{E(\sum_{t=0}^{T-1} \gamma^t R_t)}{E(\sum_{t=0}^{T-1} \gamma^t)}$ being indifferent to M and Mλ

每次选择gittins index最大的arm。

- bernoulli bandit

Bernouilli Bandit

arm i reward 0 or 1 with unknown probability μ_i
number of success and failures, $s_i + f_i$

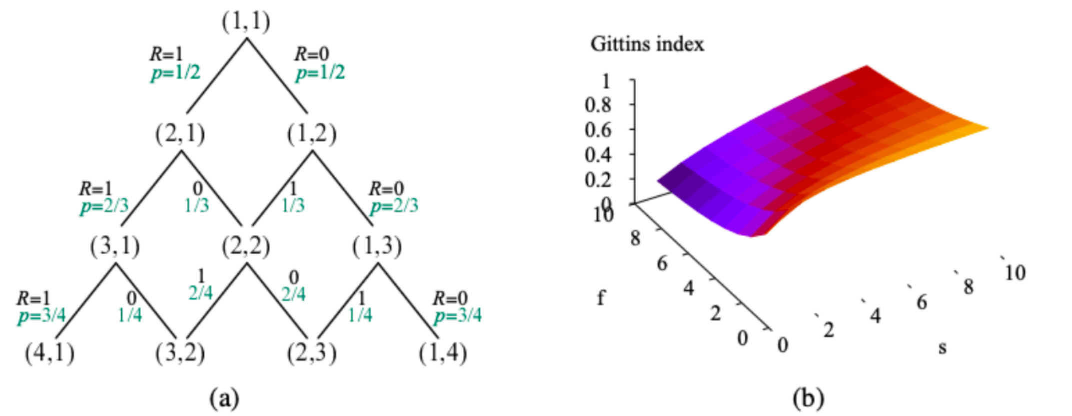


Figure 17.14 (a) States, rewards, and transition probabilities for the Bernoulli bandit. (b) Gittins indices for the states of the Bernoulli bandit process.

每个状态的reward服从二项分布；

- upper confidence bounds (UCB)

- $$UCB(M_i) = \hat{\mu}_i + \frac{g(N)}{\sqrt{N_i}}$$
- For instance, $g(N) = 2\log(1 + N\log^2 N))^{1/2}$

μ_i hat 表示当前观测到的 臂i 的平均reward，N_i 是臂i 被拉动的次数，UCB表达式的第二项是 confidence interval 的宽度；

UCB 的策略可以总结为：

- 如果一个臂的平均奖励较高，**exploit这个臂**。
- 如果一个臂的置信区间较宽，说明我们对这个臂的了解不足，**explore这个臂**。
- **总是选择 UCB 值最高的臂**，这样可以在 **explore和exploit** 之间取得平衡。