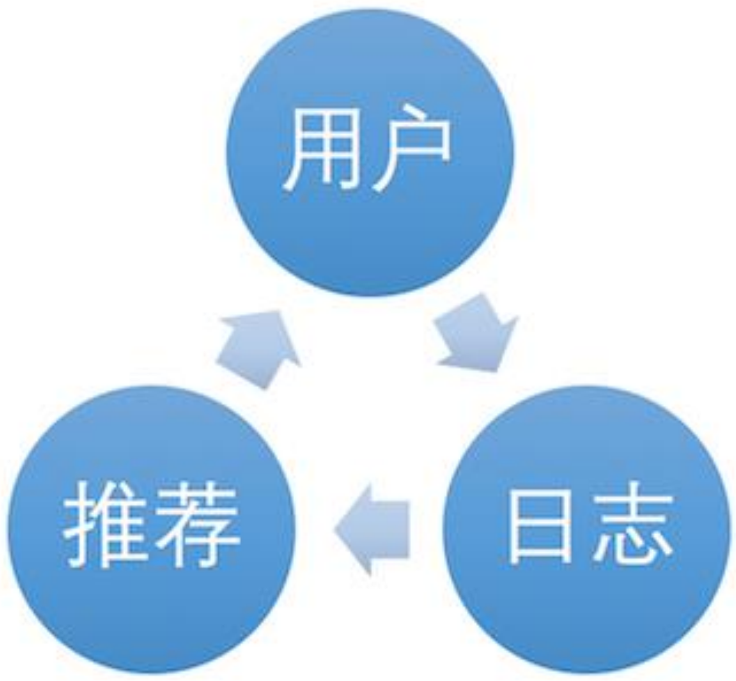
EE (Explore & Exploit) Introduction

--xpguo & lshen

数据闭环



推荐系统根据用户日志来进行建模推荐，即：

日志 -> 推荐算法 -> 用户

日志也是由用户产生的，即：

用户 -> 日志

两者拼成一个环状，我们称之为 “数据闭环”

先有鸡？先有蛋？

问：为什么给A推荐”摇滚”歌曲？

答：因为A过去听的都是”摇滚”歌曲，所以A喜欢”摇滚”。

问：推荐系统不给A用户推”非摇滚”，用户怎么能听到”非摇滚”？

在数据闭环中流转的都是”老Item”，新”Item”并没有多少展现机会，

推荐变得越来越窄

方案

- 越推越窄是典型的EE问题(explore & exploit)
- 解决方案有两类
  - Bandit: epsilon-greedy, thompson sampling, UCB, linUCB
  - Reinforcement Learning: Q-learning, Policy Gradients, ...

MAB (Multi-armed bandit)

How to play

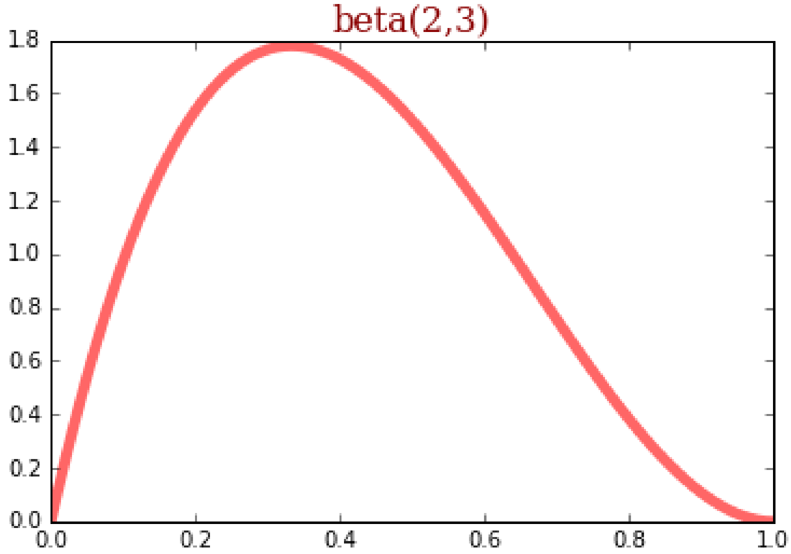
1. A row of slot machines
2. Choose one to play
3. Each machine provides a random reward from a probability distribution specific to that machine.

How to maximize the sum of rewards earned through a sequence of lever pulls?

Epsilon-greedy

- **epsilon** selected at random
- **1-epsilon** the best selected

Thompson Sampling—概率的概率



Thompson Sampling—统计估计

- 统计

正面：n^1次

反面：n^0次

$$p(head) = \frac{n^1}{n^1 + n^0}$$

$$p(tail) = \frac{n^0}{n^1 + n^0}$$

Thompson Sampling—似然估计

- 最大似然

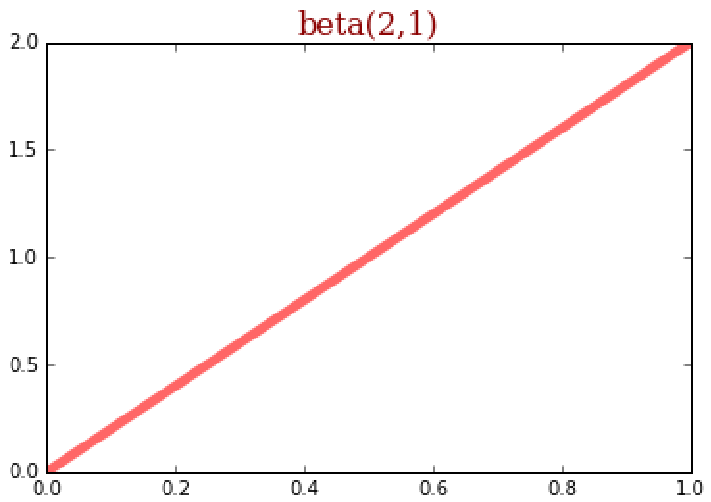
找一枚使该事件**最可能**发生的硬币

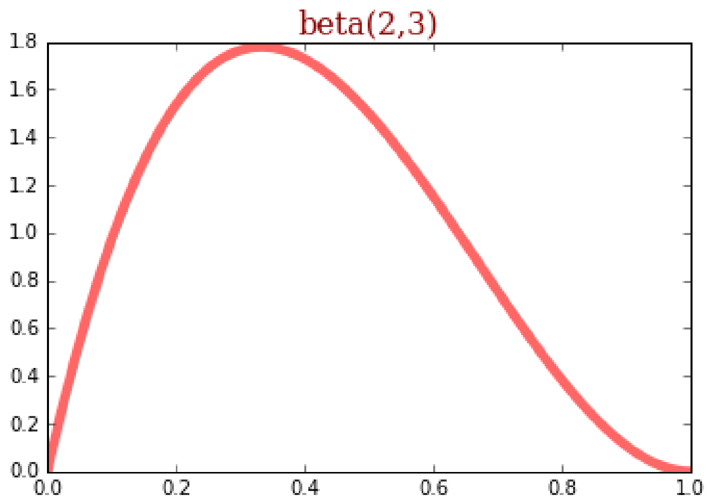
$$
\begin{aligned}
\vartheta &= \operatorname{argmax}_{\vartheta}\ L(\vartheta|X) \\
&= \operatorname{argmax}_{\vartheta}\ p(X|\vartheta) \\
&= \operatorname{argmax}_{\vartheta}\ \prod_{i=1}^{N} p(C = c_i|\vartheta) \\
&= \operatorname{argmax}_{\vartheta} \sum_{i=1}^{N} logp(C = c_i|\vartheta) \\
&= \operatorname{argmax}_{\vartheta} n^{(1)} logp(C = 1|\vartheta) + n^{(0)} logp(C = 0|\vartheta) \\
&= \operatorname{argmax}_{\vartheta} n^{(1)} log\vartheta + n^{(0)} \log(1-\vartheta)
\end{aligned}
$$

求导后，$\vartheta = \dfrac{n^1}{n^1 + n^0}$

最大似然和统计得到的值是一样的！

Thompson Sampling — 共轭





此处黑板

beta分布与二项分布共轭

Thompson Sampling—极大后验

- 极大后验

找一枚先验条件下（beta分布）使该事件最可能发生的硬币

极大后验=多几次先验实验的最大似然

$$
\begin{aligned}
\vartheta &= \operatorname{argmax}_{\vartheta}\ L(\vartheta|X) \\
&= \operatorname{argmax}_{\vartheta}\ \frac{p(X|\vartheta)p(\vartheta)}{p(X)} \\
&= \operatorname{argmax}_{\vartheta}\ p(X|\vartheta)p(\vartheta) \\
&= \operatorname{argmax}_{\vartheta} \sum_{i=1}^{N} logp(C=c_i|\vartheta) + logp(\vartheta) \\
&= \operatorname{argmax}_{\vartheta} n^{(1)} logp(C=1|\vartheta) + n^{(0)} logp(C=0|\vartheta) + logp(\vartheta) \\
&= \operatorname{argmax}_{\vartheta} n^{(1)} log\vartheta + n^{(0)} \log(1-\vartheta) + logBeta(\vartheta|\alpha,\beta)
\end{aligned}
$$

求导后，$\vartheta = \dfrac{n^1 + \alpha - 1}{n^1 + n^0 + \alpha + \beta - 2}$

相当于加正则，或平滑操作

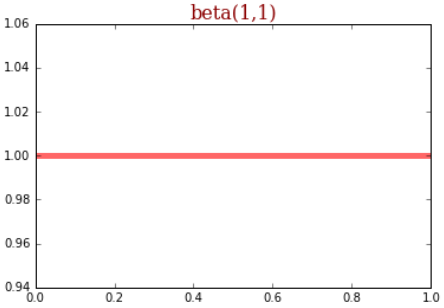
Thompson Sampling —贝叶斯估计

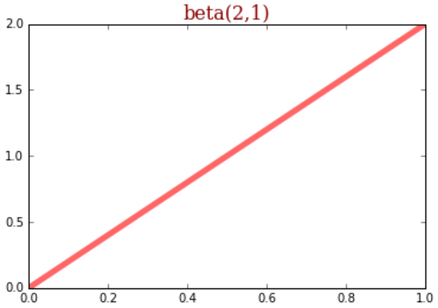
- 根据X从p的分布中采样

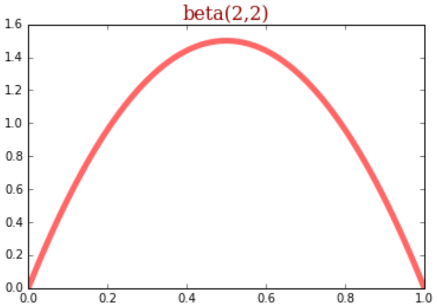
先验+事件=后验

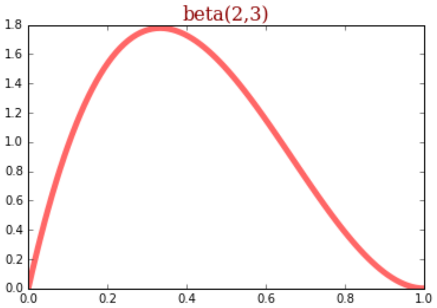
$$p(\vartheta|X) = \frac{p(X|\vartheta)p(\vartheta)}{p(X)}$$

$$p(p|C,\alpha,\beta) = \frac{\prod_{i=1}^{N} p(C = c_i|p)\, p(p|\alpha,\beta)}{\int_0^1 \prod_{i=1}^{N} p(C = c_i|p)\, p(p|\alpha,\beta)dp} = Beta(p|n^1 + \alpha, n^0 + \beta)$$









未抛币 → 抛正面 → 抛反面 → 抛反面

UCB (Upper Confidence Bound)

- Try the action that maximize $argmax_j\ (\bar{x}_j + \sqrt{\frac{2lnn}{n_j}})$

For each action j, Avg reward $\bar{x}_j$, Number of times tried nj

Total number tried n

$\bar{x}_j$ exploit

$\sqrt{\frac{2lnn}{n_j}}$ explore

- Proof

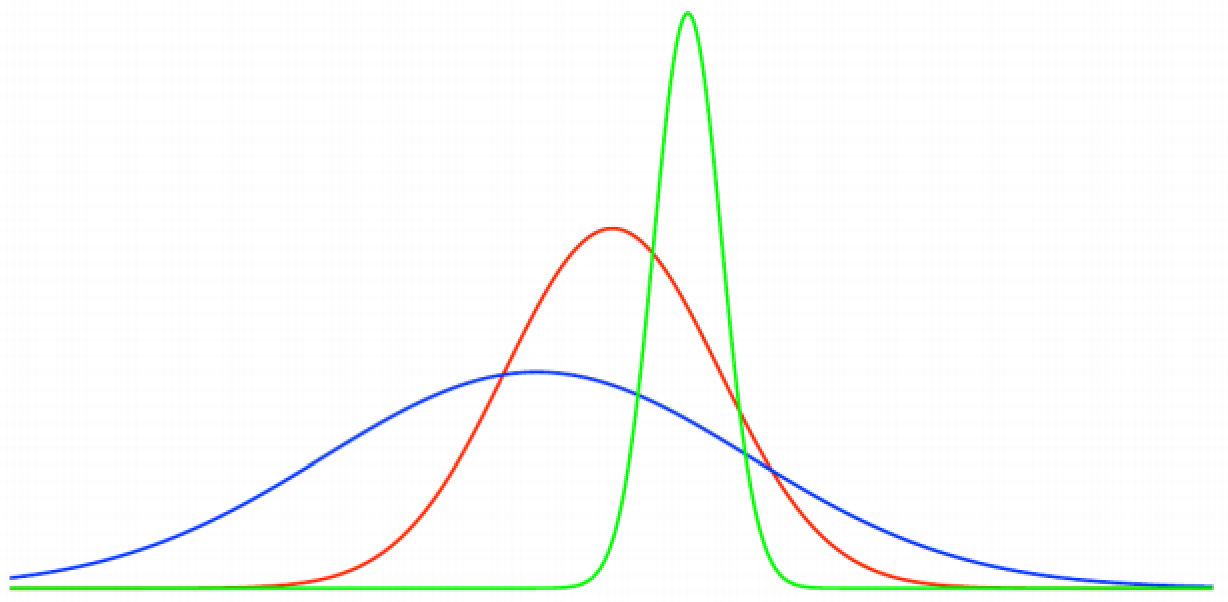
Recall that if $X_1, X_2, \ldots, X_n$ are independent and 1-subgaussian (which means that $\mathbb{E}[X_i] = 0$) and $\hat{\mu} = \sum_{t=1}^{n} X_t/n$, then

$$\mathbb{P}\left(\hat{\mu} \geq \varepsilon\right) \leq \exp\left(-n\varepsilon^2/2\right).$$

Equating the right-hand side with δ and solving for ε leads to

$$\mathbb{P}\left(\hat{\mu} \geq \sqrt{\frac{2}{n}\log\left(\frac{1}{\delta}\right)}\right) \leq \delta. \qquad (1)$$

UCB (Upper Confidence Bound)



- The more **uncertain** we are about an action–value
- The more important it is to **explore** that action
- It could turn out to be the **best action**

LinUCB (1)

Algorithm 1 LinUCB with disjoint linear models.

```
Inputs: α ∈ ℝ+
for t = 1, 2, 3, ..., T do
    Observe features of all arms a ∈ 𝒜_t: x_{t,a} ∈ ℝ^d
    for all a ∈ 𝒜_t do
        if a is new then
            A_a ← I_d (d-dimensional identity matrix)
            b_a ← 0_{d×1} (d-dimensional zero vector)
        end if
        θ̂_a ← A_a^{-1} b_a
        p_{t,a} ← θ̂_a^⊤ x_{t,a} + α sqrt(x_{t,a}^⊤ A_a^{-1} x_{t,a})
    end for
    Choose arm a_t = arg max_{a∈𝒜_t} p_{t,a} with ties broken arbitrarily, and observe a real-valued payoff r_t
    A_{a_t} ← A_{a_t} + x_{t,a_t} x_{t,a_t}^⊤
    b_{a_t} ← b_{a_t} + r_t x_{t,a_t}
end for
```

原始特征向量都要归一化成单位向量。

还要对原始特征降维，以及模型要能刻画一些非线性的关系。

用Logistic Regression去拟合用户对文章的点击历史，其中的线性回归部分为：

$$\boldsymbol{\phi}_u^\top \mathbf{W} \boldsymbol{\phi}_a$$

拟合得到参数矩阵W，可以将原始用户特征（1000多维）投射到文章的原始特征空间（80多维），投射计算方式：

$$\boldsymbol{\psi}_u \stackrel{\text{def}}{=} \boldsymbol{\phi}_u^\top \mathbf{W}.$$

这是第一次降维，把原始1000多维降到80多维。

然后，用投射后的80多维特征对用户聚类，得到5个类簇，文章页同样聚类成5个簇，再加上常数1，用户和文章各自被表示成6维向量。

Yahoo!的科学家们之所以选定为6维，因为数据表明它的效果最好[5]，并且这大大降低了计算复杂度和存储空间。

我们实际上可以考虑三类特征：U（用户），A（广告或文章），C（所在页面的一些信息）。

LinUCB (2)

```scala
object TestLinUCB extends App {

  var Aa: DenseMatrix[Double] = null
  var ba: DenseVector[Double] = null
    for( t <- 1 to 100) {

        if(t==1){
          Aa = DenseMatrix.eye[Double](3)
          ba = DenseVector.zeros[Double](3)
        }
      val Xt: DenseMatrix[Double] = new DenseMatrix[Double](3,1,Array(1.0,0,1.0))
      val theta: DenseVector[Double] = inv(Aa)*ba
      val alpha = 0.25
      val itemScore1 = theta dot Xt.toDenseVector
      val itemScore2 = alpha * Math.sqrt((Xt.t*inv(Aa)*Xt).data(0))
      val itemScore = itemScore1+itemScore2
      println("itemScore1="+itemScore1+";itemScore2="+itemScore2+";itemScore="+itemScore)
      Aa = Aa + Xt * (Xt.t)
      if(t%2==0)ba = ba + Xt.toDenseVector
    }
```

两天点一次

```
itemScore1=0.0;itemScore2=0.3535533905932738;itemScore=0.3535533905932738
itemScore1=0.0;itemScore2=0.2041241452319315;itemScore=0.2041241452319315
itemScore1=0.3999999999999999;itemScore2=0.15811388300841894;itemScore=0.5581138830084189
itemScore1=0.2857142857142857;itemScore2=0.1336306209562122;itemScore=0.4193449066704979
itemScore1=0.4444444444444444;itemScore2=0.11785113019775792;itemScore=0.5622955746422024
itemScore1=0.36363636363636376;itemScore2=0.10660035817780524;itemScore=0.470236721814169
itemScore1=0.4615384615384617;itemScore2=0.09805806756909202;itemScore=0.5595965291075538
itemScore1=0.40000000000000036;itemScore2=0.09128709291752768;itemScore=0.49128709291752803
```

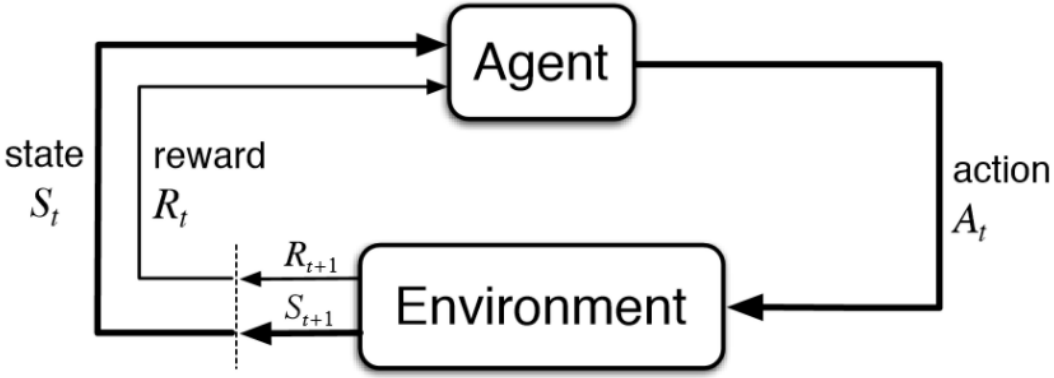
```
itemScore1=0.4923076923076941;itemScore2=0.02531848417709162;itemScore=0.5176261764847857
itemScore1=0.4974619289340083;itemScore2=0.02518963609299382;itemScore=0.5226515650270022
itemScore1=0.4924623115577873;itemScore2=0.025062735355854193;itemScore=0.5175250469136415
```

七天点一次

```
itemScore1=0.0;itemScore2=0.3535533905932738;itemScore=0.3535533905932738
itemScore1=0.0;itemScore2=0.2041241452319315;itemScore=0.2041241452319315
itemScore1=0.0;itemScore2=0.15811388300841894;itemScore=0.15811388300841894
itemScore1=0.0;itemScore2=0.1336306209562122;itemScore=0.1336306209562122
itemScore1=0.0;itemScore2=0.11785113019775792;itemScore=0.11785113019775792
itemScore1=0.0;itemScore2=0.10660035817780524;itemScore=0.10660035817780524
itemScore1=0.0;itemScore2=0.09805806756909202;itemScore=0.09805806756909202
itemScore1=0.1333333333333333;itemScore2=0.09128709291752768;itemScore=0.22462042625086098
itemScore1=0.11764705882352944;itemScore2=0.08574929257125442;itemScore=0.20339635139478385
itemScore1=0.10526315789473684;itemScore2=0.08111071056538127;itemScore=0.1863738684601181
```

```
itemScore1=0.13333333333333375;itemScore2=0.02531848417709162;itemScore=0.15865181751042537
itemScore1=0.14213197969542968;itemScore2=0.02518963609299382;itemScore=0.1673216157884235
itemScore1=0.14070351758793898;itemScore2=0.025062735355854193;itemScore=0.16576625294379316
```

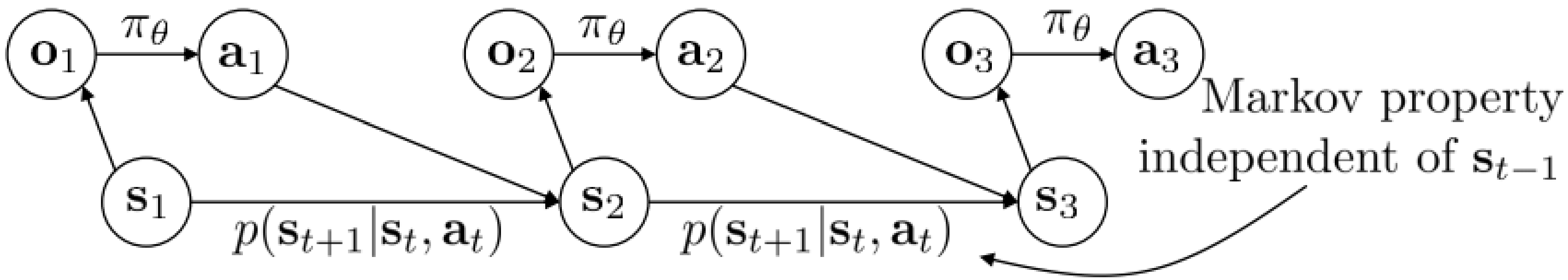
Reinforcement Learning (1)



两大实体

- Agent
  - agent可以从environment中得到reward
  - agent需要知道自己的state
  - agent可以选择自己的action，即是一个p(action|state)的求解过程
- Environment
  - environment需提供一个reward函数（可能需自定义设计）
  - environment需进行state的状态转移（可能是黑盒子）
  - environment需接收agent的action

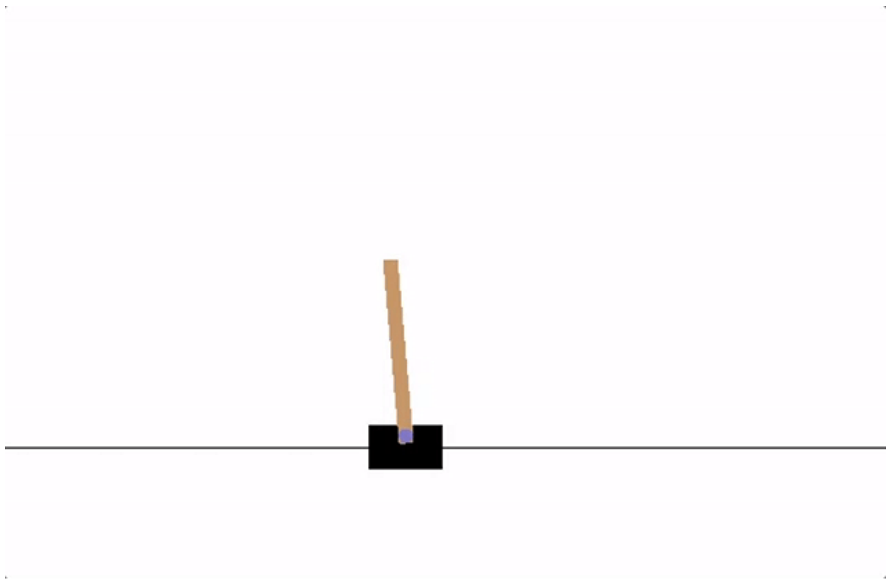
Reinforcement Learning (2)



两大实体互相作用，有几大重要的元素

- action: 由agent产生，作用于environment
- reward: environment针对agent的state+action产生的奖赏or惩罚
- state: agent的状态，由action实现状态转移，即p(state_x+1|state_x, action_x)的马尔科夫转移过程
- observation: 即state的外在表现

CartPole-v0



Observation

Type: Box(4)

Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -41.8°	~ 41.8°
3	Pole Velocity At Tip	-Inf	Inf

Actions

Type: Discrete(2)

Num	Action
0	Push cart to the left
1	Push cart to the right

Reward

Reward is 1 for every step taken, including the termination step

Episode Termination

1. Pole Angle is more than ±12°
2. Cart Position is more than ±2.4 (center of the cart reaches the edge of the display)
3. Episode length is greater than 200

Q Learning — Q Value

- Q Value

what our return would be, if we were to take an action in a given state

- Q Table

一个两维空间[observation, action]，表示在某个observation时执行某个action的总的reward和（立即的reward和之后的reward的discount）

Q Learning — Bellman Equation

- $Q(s, a) = r + \gamma(\max(Q(s', a'))$
- 其中,
  s表示state，也即observation
  a表示action
  r表示current reward
  s’表示next state，即state下做出action之后到达的new state
  a’表示next state后的策略，max(Q(s’,a’)表示s’后的最佳策略的Q值
  γ表示future reward的一个discount

Q Learning — Code

```
#The Q-Table learning algorithm
while j < 99:
    j+=1
    #Choose an action by greedily (with noise) picking from Q table
    a = np.argmax(Q[s,:] + np.random.randn(1,env.action_space.n)*(1./(i+1)))
    #Get new state and reward from environment
    s1,r,d,_ = env.step(a)
    #Update Q-Table with new knowledge
    Q[s,a] = Q[s,a] + lr*(r + y*np.max(Q[s1,:]) - Q[s,a])
```

Q Learning — Q Value

- Q Value

what our return would be, if we were to take an action in a given state

- Q Table

一个两维空间[observation, action]，表示在某个observation时执行某个action的总的reward和（立即的reward和之后的reward的discount）

Q Learning — 问题

- Table空间太大存不下
- 连续值问题

RL 解决的问题

- 延迟Reward
- 探索开采

Reference

- http://banditalgs.com/
- https://github.com/openai/gym/wiki/CartPole-v0
- http://web.stanford.edu/class/cs234/index.html
- https://jeremykun.com/2013/10/28/optimism-in-the-face-of-uncertainty-the-ucb1-algorithm/
- https://www.cs.bham.ac.uk/internal/courses/robotics/lectures/ucb1.pdf

Q&A