Multi-Armed Bandit: epsilon-greedy

冯伟
Hulu推荐算法

智能决策系列的开篇当然要献给伟大的**M**ulti-**A**rmed **B**andit (以下简称MAB)问题。

背景

假设我们开了一家叫Surprise Me的饭馆，客人来了不用点餐，由算法来决定改做哪道菜，整个过程如下：

步骤 1: 客人 user = 1...T 依次到达餐馆

步骤 2: 给客人推荐一道菜，客人接受则留下吃饭(reward=1)，拒绝则离开(reward=0)

步骤 3: 记录选择接受的客人总数 total_reward += reward

整个过程的伪代码如下：

```
for t in range(0, T): # T个客人依次进入餐馆
    # 从N道菜中推荐一个，reward = 1 表示客人接受，reward = 0 表示客人拒绝并离开
    item, reward = pick_one(t, N)
    total_reward += reward # 一共有多少客人接受了推荐
```

假设

为了由浅入深地解决这个问题，我们先做两个假设：

1. 同一道菜，有时候会做的好吃一些 (概率＝p)，有时候会难吃一些 (概率 = 1-p)，但我们并不知

道概率p是多少，只能通过多次观测进行统计。
2. 菜做的好吃时 (概率=p)，客人一定会留下(reward=1)；菜不好吃时(概率 = 1- p)，客人一定会离开 (reward=0)。暂时先不考虑个人口味的差异 (后续会在Contextual Bandit中考虑)
3. 菜好吃不好吃只有客人才说的算，饭馆是事先不知道的（先验知识会在Bayesian Bandit中考虑）

解决思路

探索阶段 (Exploration)：通过多次观测推断出一道菜做的好吃的概率 – 如果一道菜已经推荐了k遍（获取了k次反馈），我们就可以算出菜做的好吃的概率：

$$\tilde{p} = \frac{\sum reward_i}{k}$$

如果推荐的次数足够多，k足够大，那么 $\tilde{p}$ 会趋近于真实的菜做的好吃的概率 p 。

利用阶段 (Exploitation)：已知所有的菜做的好吃的概率，该如何推荐？ – 如果每道菜都推荐了多遍，我们就可以计算出N道菜做的好吃的概率 { $\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_N$ }，那么我们就可以推荐 $\tilde{p}$ 最大的那道菜。

核心问题：什么时候探索(Exploration)，什么时候利用 (Exploitation)?

探索 (Exploration) v.s. 利用(Exploitation)，这是一个经久不衰的问题：

- Exploration的代价是要不停的拿用户去试菜，影响客户的体验，但有助于更加准确的估计每道菜好吃的概率
- Exploitation会基于目前的估计拿出“最好的”菜来服务客户，但目前的估计可能是不准的（因为试吃的人还不够多）

解决方法 $\epsilon-greedy$ ： 每当客人到来时:

- 以 ϵ 的概率选择探索 (Exploration)，从N道菜中随机选择(概率为 $\frac{\epsilon}{N}$)一个让客人试吃，根据客人的反馈更新菜的做的好吃的概率 { $\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_N$ }
- 以 $1-\epsilon$ 的概率选择利用 (Exploitation)，从N道菜{ $\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_N$ }中选择好吃的概率最高的菜推荐给用户

那么 $\epsilon-greedy$ 的缺点是什么呢：

- **在试吃次数相同的情况下，好吃和难吃的菜得到试吃的概率是一样的**：有一道菜持续能得到好吃的反馈，而另一道菜持续得到难吃的反馈，但在 $\epsilon-greedy$ 中，探索两道菜的概率是一样的（均为 $\frac{\epsilon}{N}$ ）。
- **在估计的成功概率相同的情况下，试吃次数多的和试吃次数少的菜得到再试吃的概率是一样的**：假设有两道菜，第一道菜50人当中30个人说好，第二道菜5个人当中3个人说好，虽然两道菜的成功概率都是60%(30/50 = 3/50)，但显然反馈的人越多，概率估计的越准。再探索时，应该把重心放在试吃次数少的菜上。

下一讲我们会介绍更加智能的探索方法 – Upper Confidence Bound (UCB) 算法，它会充分利用每道菜的历史信息进行探索，包括：

- 某道菜被多少人试吃过
- 某道菜目前为止做的好吃的概率是多少

最后附上 $\epsilon-greedy$ 的完整代码：

```
import numpy as np

T = 100000 # T个客人
N = 10 # N道菜

true_rewards = np.random.uniform(low=0, high=1, size=N) # N道菜好吃的概率
estimated_rewards = np.zeros(N)
number_of_trials = np.zeros(N)
total_reward = 0

def alpha_greedy(N, alpha=0.1):
    item = 0
    if np.random.random() < alpha:
        item = np.random.randint(low=0, high=N)
    else:
        item = np.argmax(estimated_rewards)
    reward = np.random.binomial(n=1, p=true_rewards[item])
    return item, reward

for t in range(1, T): # T个客人依次进入餐馆
   # 从N道菜中推荐一个，reward = 1 表示客人接受，reward = 0 表示客人拒绝并离开
   item, reward = alpha_greedy(N)
   total_reward += reward # 一共有多少客人接受了推荐

   # 更新菜的平均成功概率
   number_of_trials[item] += 1
   estimated_rewards[item] = ((number_of_trials[item] - 1) * estimated_rewards[it



print("total_reward=" + str(total_reward))
```

欢迎订阅**微信公众号 "零基础机器学习"**，搜索微信号：ml-explained

编辑于 2018-01-12

「真诚赞赏，手留余香」

赞赏

还没有人赞赏，快来当第一个赞赏的人吧！

机器学习　强化学习 (Reinforcement Learning)　在线机器学习

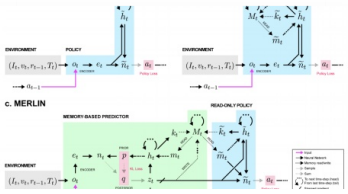
文章被以下专栏收录

零基础机器学习　已关注

推荐阅读

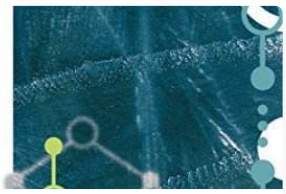
城·湛略|湾区01：世界级湾区的天时、地利与人和

九同堂堂主 发表于精明城市化...



除了DQN/A3C，还有哪些高级强化学习成果

论智 发表于论智



《Reinforcement Learning: An Introduction》第五章笔记

Kelvi... 发表于强化学习读...

8 条评论 切换为时间排序

写下你的评论...

Jiaxi Tang 1 年前

有一个问题：在更新成功概率的时候不应该是用number_of_rewards[item] / number_of_trails[item]吗?

赞

冯伟 (作者) 回复 **Jiaxi Tang** 1 年前

多谢斧正，已经修正了代码：）

赞

差剃须刀的火神 1 年前

更新菜的成功概率应该是错了

赞

冯伟 (作者) 回复 **差剃须刀的火神** 1 年前

多谢斧正，已经修正了代码：）

赞

desperado 1 年前

if np.random.random() < (alpha / N): 应该是if np.random.random() < (alpha): ？？

赞

desperado 1 年前

这样得到的概率是 epsilon/N^2

赞

冯伟 (作者) 回复 **desperado** 1 年前

多谢指正，已经进行了修正

赞

哈哈哒哒 4 个月前

reward = np.random.binomial(n=1, p=true_rewards[item])

在做选择探索 (Exploration) 时，这行代码也会执行？这样岂不是本身概率高的更有可能被选中?

赞