推荐系统应用多臂老虎机算法

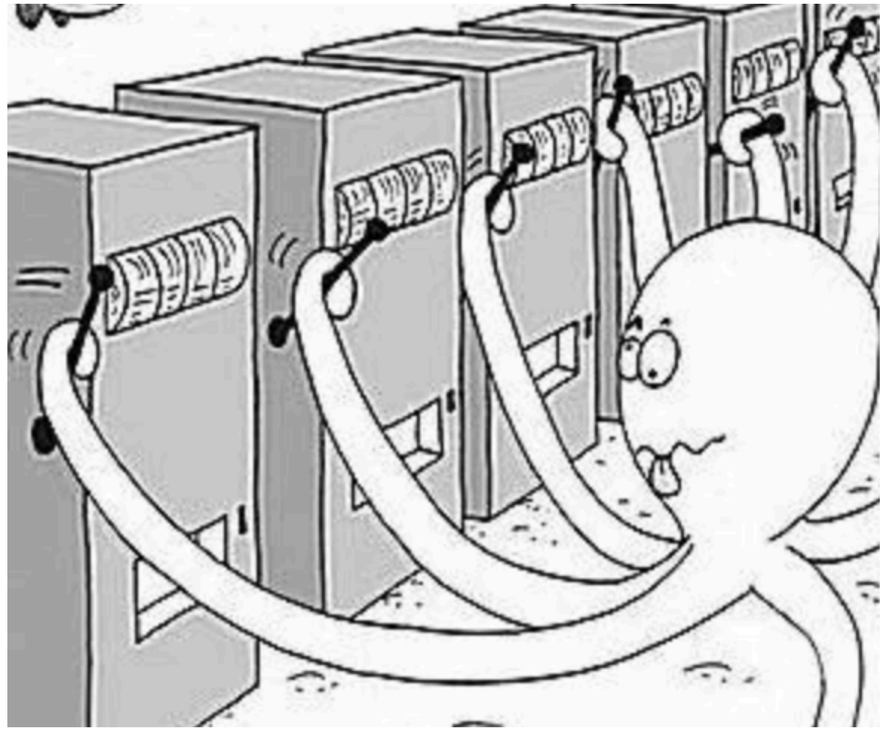
Bandit Algorithm in Recommendation

龙旷飞/李泽浩

推荐系统应用多臂老虎机算法

内容

- 推荐系统简介与新内容
- Bandit 算法
- LinUCB算法与改进算法
- 结果分析



推荐系统简介与新内容

推荐系统应用多臂老虎机算法

推荐系统简介与新内容

- 随着今天越来越多的直接面向消费者（DTC）平台的选择，大多数消费者无法订阅所有平台。订阅/购买决定是由内容和用户体验共同驱动的。今天的消费者在考虑、购买和接触内容时，期望获得实时、精心策划的体验。无论是提高点击率、增加观看次数、观看时间、订阅或购买优质内容，媒体公司都在努力寻找方法，以提供更好的客户体验并扩大盈利能力。

- 推荐系统是实现这些目标的一个重要工具。DTC平台提供的推荐可以最大限度地发挥深度内容目录的价值，在消费者观看了最初将他们带到平台的内容之后，还可以保持他们的参与。例如，视频点播（VOD）平台的良好推荐可以通过在基于消费者行为的推荐中浮现长尾内容而增加收入。我们首先回顾目前使用的常见的推荐系统的种类。然后，我们深入研究了这个领域中的一些最新发展。

推荐系统应用多臂老虎机算法

常见推荐系统

- 常见系统可以被归类为基于内容的过滤或协作式过滤。基于内容的过滤是最简单的系统之一，但有时仍然是有用的。它是基于明确或隐含地提供的已知的用户偏好，以及关于项目特征的数据（比如项目所属的类别）。虽然这些系统很容易实现，但它们的建议往往让人感觉是静态的，而且很难处理那些偏好未知的新用户。

AWS Media Blog

What's new in recommender systems

by Brent Rabowsky and Liam Morrison | on 17 NOV 2020 | in Announcements | Permalink | Share

With the ever-increasing selection of direct to consumer (DTC) platforms available today, most consumers cannot subscribe to all platforms. Subscription/purchase decisions are driven both by content (what shows/movies a platform has) and user experience (how easy a platform is to use). Consumers today expect real-time, curated experiences as they consider, purchase, and engage with content. Whether it's improving click-through rate, increasing views, view duration, subscriptions, or purchases of premium content, media companies are working hard to find ways to deliver a better customer experience and expand profitability.

Resources

AWS for M&E
AWS Media Services
AWS Answers for M&E
How To Guides for M&E
AWS Media Blog Home

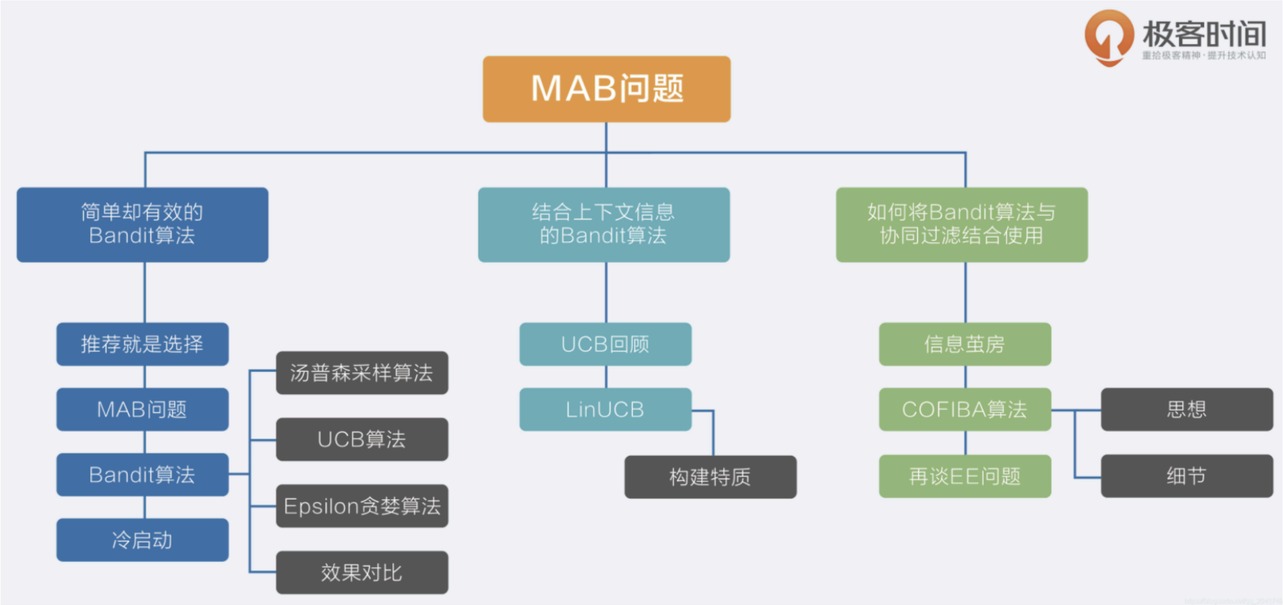
Follow

Bandit算法

推荐系统应用多臂老虎机算法

Bandit算法起源

- 在推荐系统领域里，有两个比较经典的问题常被人提起，一个是EE问题，另一个是用户冷启动问题。
- 这两个问题本质上都是如何选择用户感兴趣的主题进行推荐，比较符合Bandit算法背后的MAB问题。



推荐系统应用多臂老虎机算法

Bandit算法与遗憾

- 首先，这里我们讨论的每个臂的收益非0即1，也就是伯努利收益。然后，每次选择后，计算和最佳的选择差了多少，然后把差距累加起来就是总的遗憾。是第i次试验时被选中臂的期望收益， 是所有臂中的最佳那个，如果上帝提前告诉你，我们当然每次试验都选它，问题是上帝不告诉你，所以就有了Bandit算法。
- 这个公式可以用来对比不同Bandit算法的效果：对同样的多臂问题，用不同的Bandit算法试验相同次数，看看谁的regret增长得慢。接下来介绍不同的Bandit算法。

$$R_T = \sum_{i=1}^{T}(w_{opt} - w_{B(i)}) = Tw^* - \sum_{i=1}^{T} w_{B(i)}$$

推荐系统应用多臂老虎机算法

常用的Bandit算法——Thompson Sampling算法

- 假设每个臂是否产生收益，其背后有一个概率分布，产生收益的概率为p
- 我们不断实验，去估计一个置信度较高的概率p的概率分布就能近似解决这个问题了
- 估计概率p的概率分布的方法是假设概率p的概率分布符合beta(wins, lose)分布，它有两个参数: wins, lose
- 每个臂都维护一个beta分布的参数。每次试验后，选中一个臂，摇一下，有收益则该臂的wins增加1，否则该臂的lose增加1
- 每次选择臂的方式是用每个臂现有的beta分布产生一个随机数b，选择所有臂产生的随机数中最大的那个臂去摇

推荐系统应用多臂老虎机算法

常用的Bandit算法——UCB算法

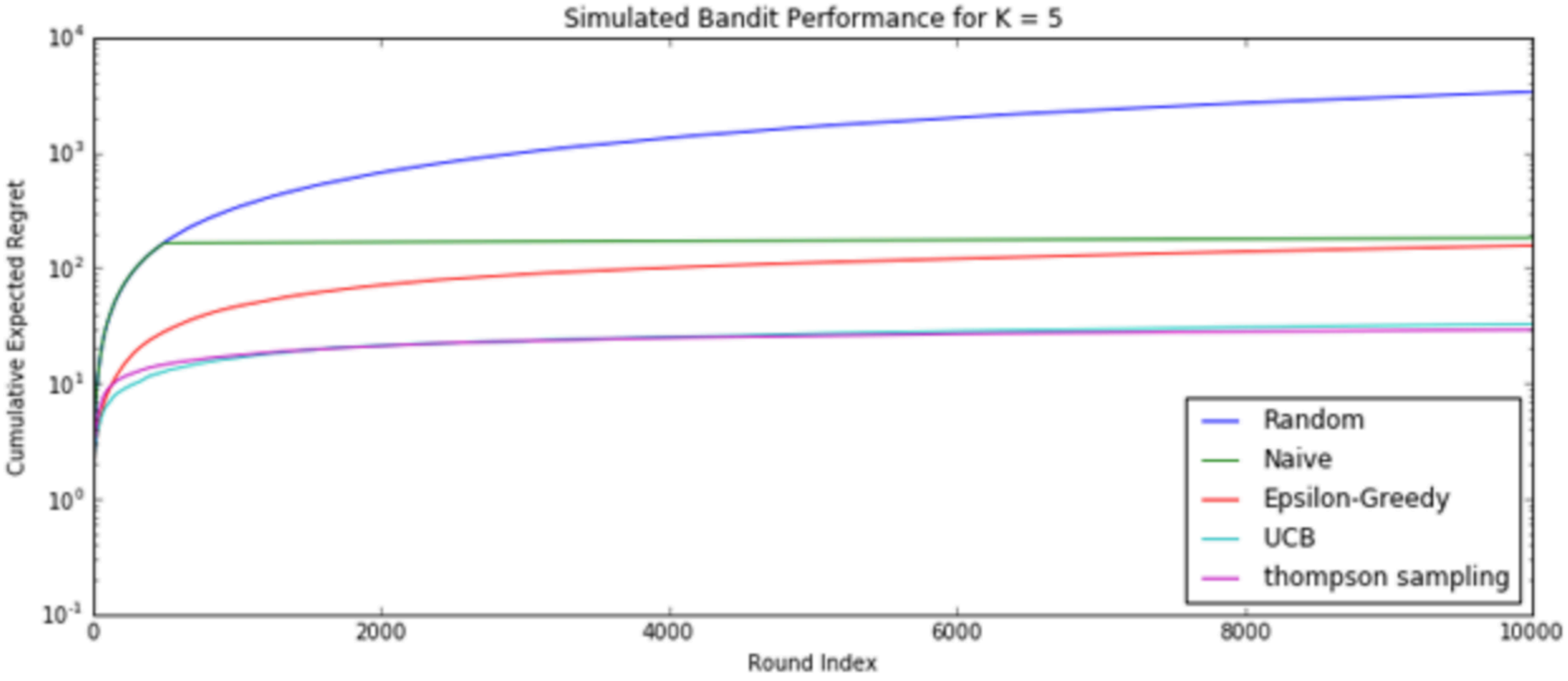
- UCB算法全称是Upper Confidence Bound（置信区间上界）
- 初始化：先对每一个臂都试一遍
- 按右式计算每个臂的分数，然后选择分数最大的臂作为选择
- 观察选择结果并更新，其中加号前面是这个臂到目前的收益均值，后面的叫做bonus，本质上是均值的标准差

$$\bar{x}_j(t) + \sqrt{\frac{2 \ln t}{T_{j,t}}}$$

推荐系统应用多臂老虎机算法

常用的Bandit算法——Epsilon – Greedy算法

- 选一个（0，1）之间较小的数作为Epsilon
- 每次以概率Epsilon做一件事：所有臂中随机选择一个
- 每次以概率1 – Epsilon选择平均收益最大的那个臂
- Epsilon的值可以控制对Exploit和Explore的偏好程度，越接近0，则越保守


Simulated Bandit Performance for K = 5
Cumulative Expected Regret
Round Index
10^4
10^3
10^2
10^1
10^0
10^{-1}
0
2000
4000
6000
8000
10000
Random
Naive
Epsilon-Greedy
UCB
thompson sampling

LinUCB算法与改进算法

推荐系统应用多臂老虎机算法

回顾UCB算法

- UCB解决Multi-armed bandit问题的思路是：用置信区间。置信区间可以简单地理解为不确定性的程度，区间越宽，越不确定。每个item的回报均值都有个置信区间，随着试验次数增加，置信区间会变窄（逐渐确定了到底回报丰厚还是可怜）。每次选择前，都根据已经试验的结果重新估计每个Item的均值及置信区间。选择置信区间上限最大的那个Item
- 如果Item置信区间很宽（被选次数很少并不确定），那么它会倾向于被多次选择，这个是算法冒风险的部分
- 如果Item置信区间很窄（被选次数很多，比较确定其权衡好坏了），那么均值达到倾向于被多次选择，这个是算法保守稳妥的部分
- UCB算法中选择置信区间的上界排序时，是一种乐观的算法。UCB算法中选择置信区间的下界排序时，是一种悲观保守的算法

推荐系统应用多臂老虎机算法

UCB算法加入特征信息

- 我们能用Feature来刻画User和Item，在每次选择Item之前，通过Feature预估每一个arm（item）的期望回报及置信区间，选择的收益就可以通过Feature泛化到不同的Item上
- LinUCB算法做了一个假设：一个Item被选择后推送给一个User，其回报和相关Feature成线性关系，这里的相关Feature就是context，也是实际项目中发挥空间最大的部分
- 用User和Item的特征预估回报及其置信区间，选择置信区间上界最大的Item推荐，观察回报后更新线性关系的参数

$$E[r_{t,a}|x_{t,a}] = x_{t,a}^T \theta_a^*$$

0: Inputs: $\alpha \in \mathbb{R}_+$
1: **for** $t = 1, 2, 3, \ldots, T$ **do**
2: Observe features of all arms $a \in \mathcal{A}_t$: $\mathbf{x}_{t,a} \in \mathbb{R}^d$
3: **for all** $a \in \mathcal{A}_t$ **do**
4: **if** a is new **then**
5: $\mathbf{A}_a \leftarrow \mathbf{I}_d$ (d-dimensional identity matrix)
6: $\mathbf{b}_a \leftarrow \mathbf{0}_{d \times 1}$ (d-dimensional zero vector)
7: **end if**
8: $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1} \mathbf{b}_a$
9: $p_{t,a} \leftarrow \hat{\boldsymbol{\theta}}_a^\top \mathbf{x}_{t,a} + \alpha \sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1} \mathbf{x}_{t,a}}$
10: **end for**
11: Choose arm $a_t = \arg\max_{a \in \mathcal{A}_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff r_t
12: $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t} \mathbf{x}_{t,a_t}^\top$
13: $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t \mathbf{x}_{t,a_t}$
14: **end for**

推荐系统应用多臂老虎机算法

尝试引入模拟退火

- 模拟退火的原理也和金属退火的原理近似：我们将热力学的理论套用到统计学上，将搜寻空间内每一点想像成空气内的分子；分子的能量，就是它本身的动能；而搜寻空间内的每一点，也像空气分子一样带有能量，以表示该点对命题的合适程度。算法先以搜寻空间内一个任意点作起始：每一步先选择一个邻居，然后再计算从现有位置到达邻居的概率

- 加入模型的想法是基于一种对解决EE（Exploit-Explore）问题的探索。我们可以在初期选择arm的时候加入模拟退火的思想，这样我们就不会每次都选择reward最大的臂并进行模型迭代。因为我们评判reward的累积遗憾是基于伯努利收益的设定，如果我们在迭代训练模型之后收敛的速度并没有得到很大的影响的话，我们在初期对于一个解决EE（Exploit-Explore）问题的方案引入则显然是值得的

推荐系统应用多臂老虎机算法

尝试混合系数模型

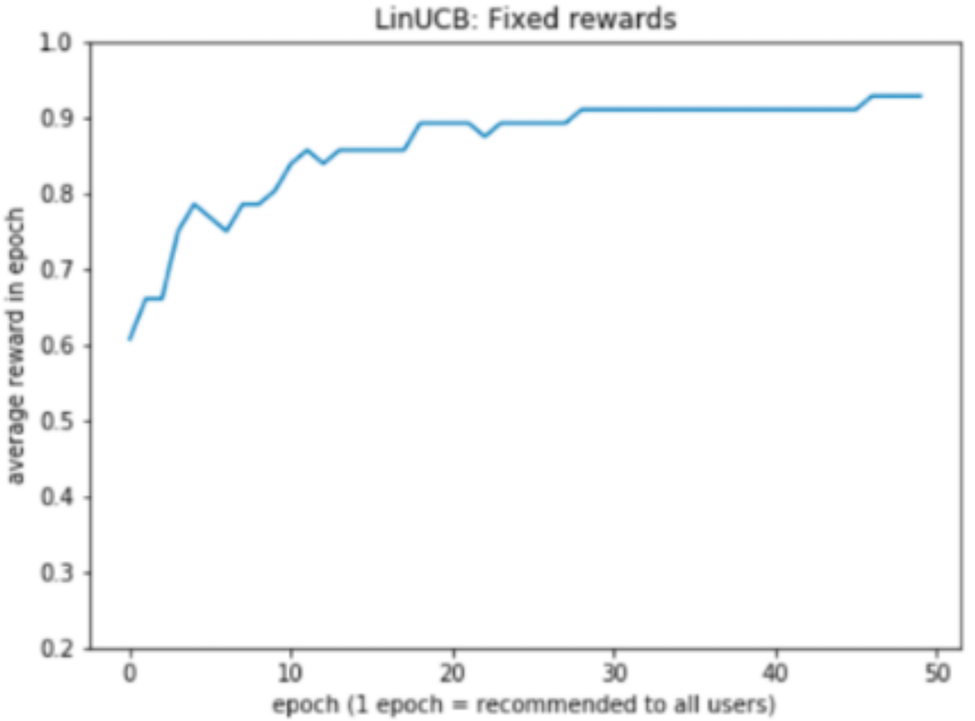
- 在许多应用中，除了特定arm的功能外，使用所有arm共享的功能也很有帮助。例如，在新闻文章推荐中，用户可能只喜欢有关政治的文章，这就提供了一个机制。因此，拥有共享和非共享部分的特征是有帮助的。从形式上看，我们通过在前面的方程中加入另一个线性项来采用以下的混合模型

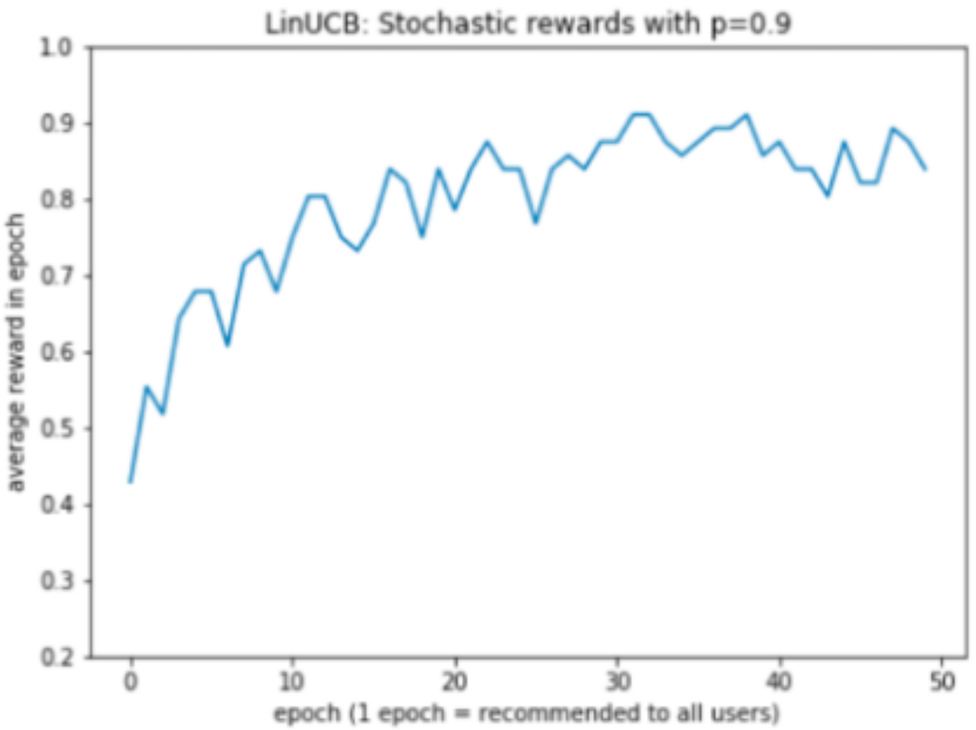
$$E[r_{t,a}|x_{t,a}] = z_{t,a}^T\beta^* + x_{t,a}^T\theta_a^*$$

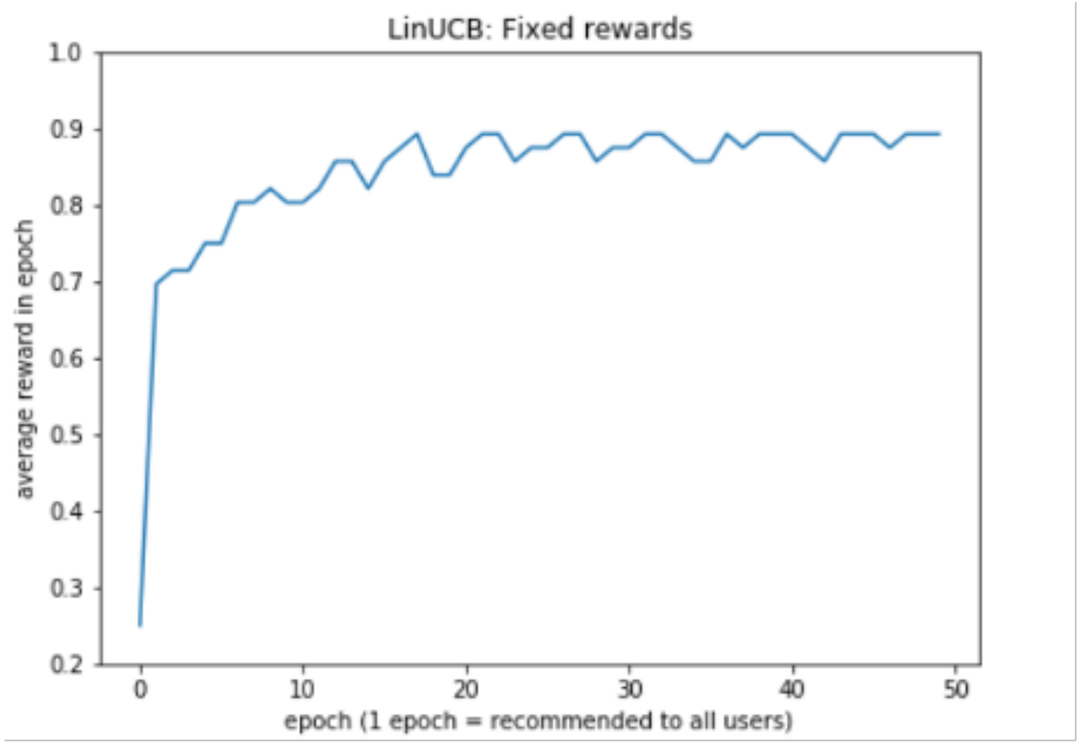
Algorithm 2 LinUCB with hybrid linear models.

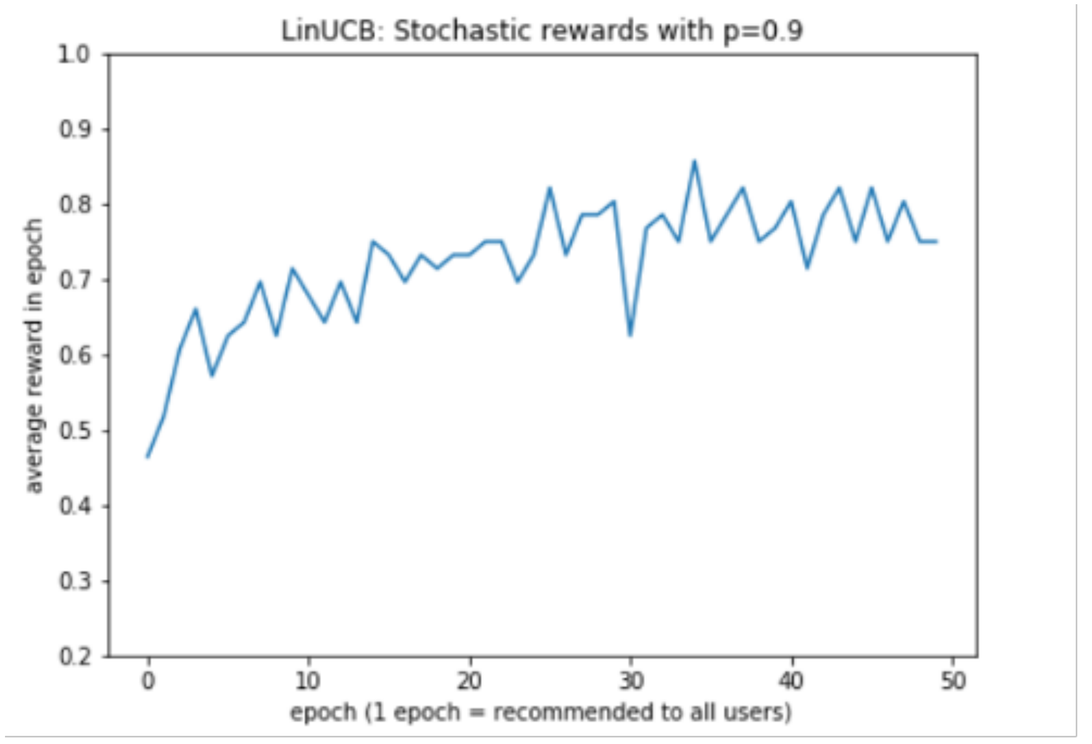
Inputs: $\alpha \in \mathbb{R}_+$
$\mathbf{A}_0 \leftarrow \mathbf{I}_k$ (k-dimensional identity matrix)
$\mathbf{b}_0 \leftarrow \mathbf{0}_k$ (k-dimensional zero vector)
for $t = 1, 2, 3, \ldots, T$ **do**
  Observe features of all arms $a \in \mathcal{A}_t$: $(\mathbf{z}_{t,a}, \mathbf{x}_{t,a}) \in \mathbb{R}^{k+d}$
  $\hat{\boldsymbol{\beta}} \leftarrow \mathbf{A}_0^{-1}\mathbf{b}_0$
  for all $a \in \mathcal{A}_t$ **do**
    if a is new **then**
      $\mathbf{A}_a \leftarrow \mathbf{I}_d$ (d-dimensional identity matrix)
      $\mathbf{B}_a \leftarrow \mathbf{0}_{d\times k}$ (d-by-k zero matrix)
      $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ (d-dimensional zero vector)
    end if
    $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1}\left(\mathbf{b}_a - \mathbf{B}_a\hat{\boldsymbol{\beta}}\right)$
    $s_{t,a} \leftarrow \mathbf{z}_{t,a}^\top\mathbf{A}_0^{-1}\mathbf{z}_{t,a} - 2\mathbf{z}_{t,a}^\top\mathbf{A}_0^{-1}\mathbf{B}_a^\top\mathbf{A}_a^{-1}\mathbf{x}_{t,a} + \mathbf{x}_{t,a}^\top\mathbf{A}_a^{-1}\mathbf{x}_{t,a} + \mathbf{x}_{t,a}^\top\mathbf{A}_a^{-1}\mathbf{B}_a\mathbf{A}_0^{-1}\mathbf{B}_a^\top\mathbf{A}_a^{-1}\mathbf{x}_{t,a}$
    $p_{t,a} \leftarrow \mathbf{z}_{t,a}^\top\hat{\boldsymbol{\beta}} + \mathbf{x}_{t,a}^\top\hat{\boldsymbol{\theta}}_a + \alpha\sqrt{s_{t,a}}$
  end for
  Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff r_t
  $\mathbf{A}_0 \leftarrow \mathbf{A}_0 + \mathbf{B}_{a_t}^\top\mathbf{A}_{a_t}^{-1}\mathbf{B}_{a_t}$
  $\mathbf{b}_0 \leftarrow \mathbf{b}_0 + \mathbf{B}_{a_t}^\top\mathbf{A}_{a_t}^{-1}\mathbf{b}_{a_t}$
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^\top$
  $\mathbf{B}_{a_t} \leftarrow \mathbf{B}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{z}_{t,a_t}^\top$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{t,a_t}$
  $\mathbf{A}_0 \leftarrow \mathbf{A}_0 + \mathbf{z}_{t,a_t}\mathbf{z}_{t,a_t}^\top - \mathbf{B}_{a_t}^\top\mathbf{A}_{a_t}^{-1}\mathbf{B}_{a_t}$
  $\mathbf{b}_0 \leftarrow \mathbf{b}_0 + r_t\mathbf{z}_{t,a_t} - \mathbf{B}_{a_t}^\top\mathbf{A}_{a_t}^{-1}\mathbf{b}_{a_t}$
end for

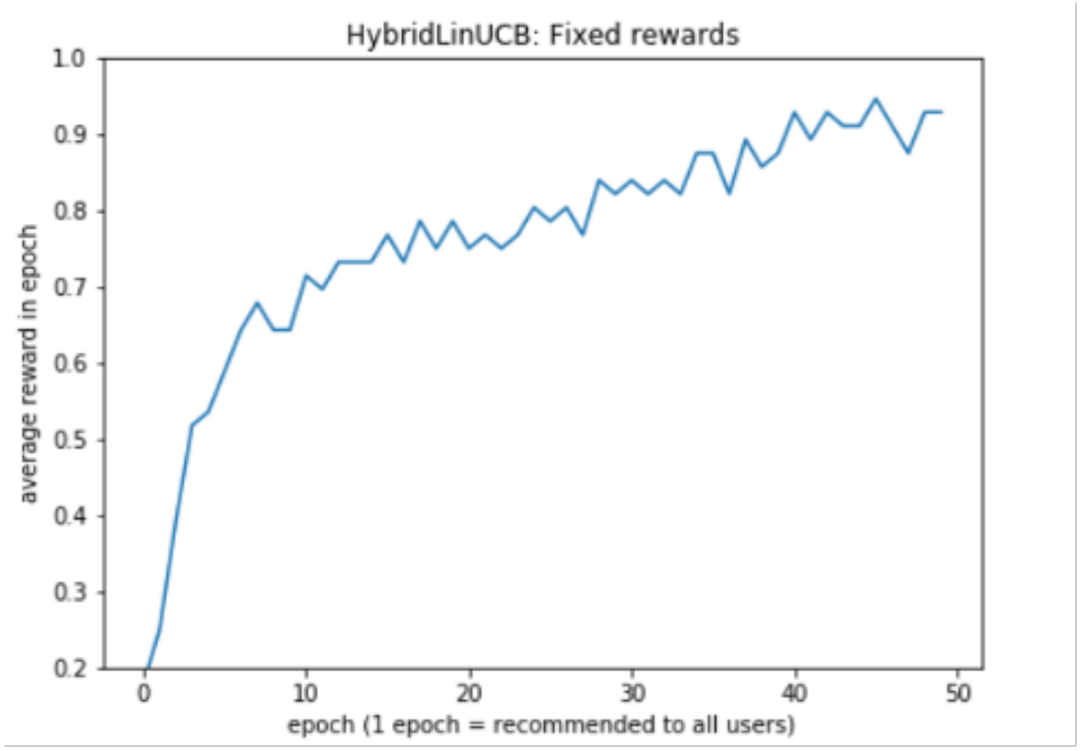
结果分析

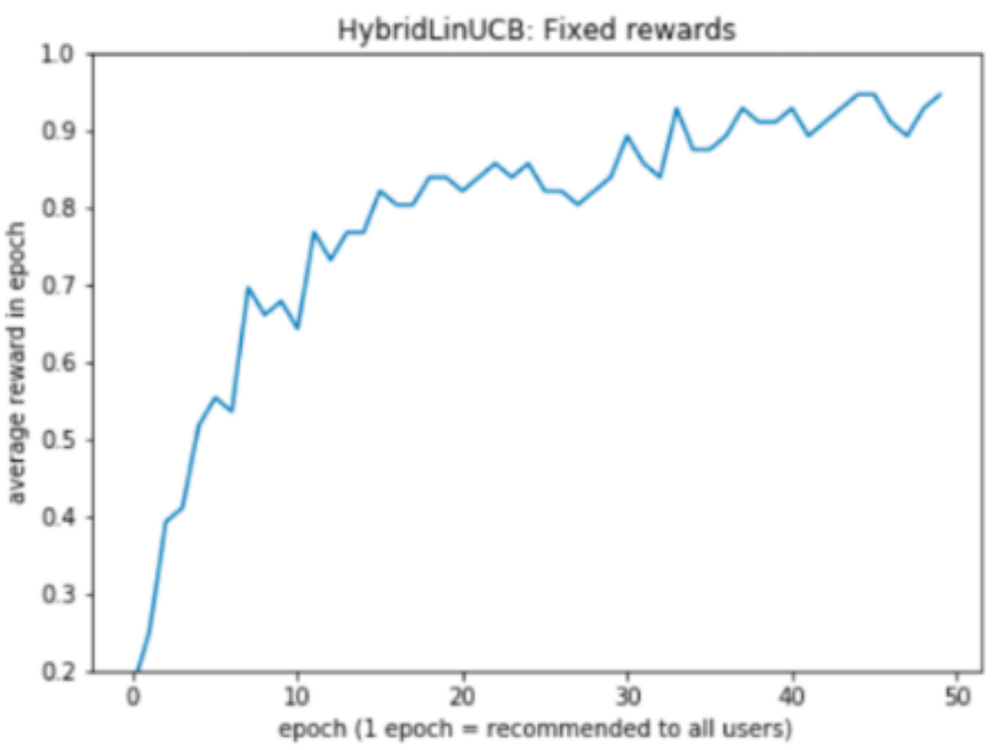

LinUCB: Fixed rewards
average reward in epoch
epoch (1 epoch = recommended to all users)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50


LinUCB: Stochastic rewards with p=0.9
average reward in epoch
epoch (1 epoch = recommended to all users)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50


LinUCB: Fixed rewards
average reward in epoch
epoch (1 epoch = recommended to all users)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50


LinUCB: Stochastic rewards with p=0.9
average reward in epoch
epoch (1 epoch = recommended to all users)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50


HybridLinUCB: Fixed rewards
average reward in epoch
epoch (1 epoch = recommended to all users)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50


HybridLinUCB: Fixed rewards
average reward in epoch
epoch (1 epoch = recommended to all users)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50

Thanks