A PAPER REVIEW: AN EFFICIENT BANDIT ALGORITHM FOR REALTIME MULTIVARIATE OPTIMIZATION

FEI WU

参数以及系统模型

参数及其意义

- A：网页（APP）布局 (layout)：有D个变量（widget），每个变量有N种选择， 共N^D种布局 $A \in \{1, 2, \ldots, N\}^D$
- X：环境变量：包括用户信息，系统时间，设备类型等等
- $B_{A,X}$: A与X交互后的特征向量（feature vector），共m位
- R：奖赏（reward），包括conversion rate, click through，点击率等等
- W： $B_{A,X}$ 中每个元素的权重

注：

- A与X交互方式可能非线性，在之后会有具体讨论
- A在multi-armed bandit中被看作不同的arm
- W 和 $B_{A,X}$ 是两个m*1的列向量， 其中的值是一一对应的， W中每个值是$B_{A,X}$对应值的权重

参数以及系统模型

系统模型

- R：随机变量，其概率模型将在下一页给出
- W：W 中的每个元素都是一个随机变量，初始化为N~(0,1)

从离散时间 t = 1,2,3…T, 每一个时间点都是一次独立实验。对每一个时间点t，系统会生成一个环境变量X_t， 我们的算法会挑选一个布局A_t。奖励会基于A_t，X_t 生成， 记为 $E[R_{A_t,X_t}]$。

假设A_t^*代表在t时间最好的A（arm），即使奖励最大化：$$A_t^* = \underset{A}{argmax}\ E[R_{A,X_t}].$$

那么我们定义在t时间的悔恨值（regret）为：$$\Delta_t = E[R_{A_t^*,X_t}] - E[R_{A_t,X_t}].$$

注：即为我们算法挑选的arm得到的reward比最优的差多少

我们bandit算法的最终目标就是最小化所有的regret之和：$$\Delta^T = \sum_{t=1}^{T} \Delta_t.$$

概率模型以及特征向量的构成

概率模型

- R：假设reward R 是binary的，即只取1：成功，-1：失败
- 那么given A, X, R的概率分布定义为：

$$P(R|A,X) = \Phi\left(R * B_{A,X}^{\top} W\right)$$

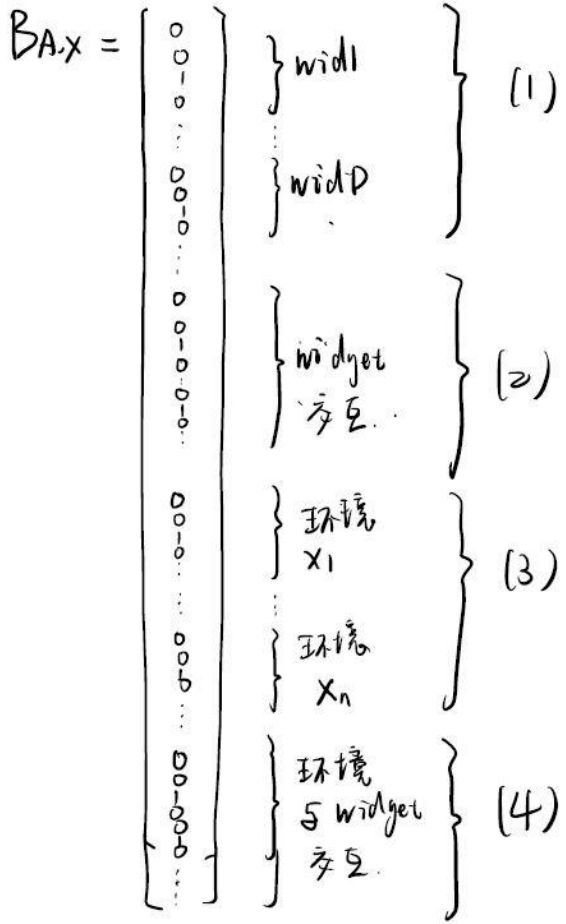
注：

- Φ 是标准正态分布的CDF
- $B_{A,X}$ 是一个m*1的列向量， W 是一个m*1的列向量， 因此$B_{A,X}^{T}$ $*$ W 变成一个标量。可以看作$B_{A,X}$ 中的值乘以权重的求和。
- W 中的每个元素都是相互独立的随机变量。初始化为iid~N(0,1)。根据每次实验的结果-R和$B_{A,X}$-更新其高斯后验概率，具体更新方法将在之后详细叙述

概率模型以及特征向量的构成

特征向量:

对于$B_{A,X}$而言，每个元素都是binary的，即0 或 1: 1代表选取而0代表没有选取.



(1) 代表每个widget自己。每段wid1 ~widD 长度均为N， 代表D 个Widget， 每个widget 有N 种选择。在对应Widget被选中的位置为1，其余位置均为0， 代表没被选中；

(2) 代表widget之间的相互影响，只有在（1）中被选中的2个feature交互处为1， 其余处为0. 长度为： N^2 * D*(D-1) / 2, 其中1的个数为：D*(D-1) / 2 , 代表widget的相互交互；

(3) 环境变量，与（1）类似；

(4) 环境变量与widget交互，与（2）类似。

注：
W 和$B_{A,X}$ 是一一对应的， 因此W中的元素也有四种， 代表这四种的权重（weight）。
在实际情况下，有可能为了简化而只用其中一部分（比如不考虑环境变量， （3）和（4）就没有了）， 下一页会有细节描述

$W_i^1(A)$	Impact of content in i^{th} widget
$W_{i,j}^2(A)$	Interaction of content in i^{th} widget with content in j^{th} widget
$W_i^c(X)$	Impact of i^{th} contextual feature
$W_{i,j}^{1c}(A,X)$	Interaction of content in i^{th} widget with the j^{th} contextual feature

概率模型以及特征向量的构成

特征向量

- 假设只考虑布局而不考虑环境变量。

如果要考虑所有变量的相互影响，一共会有N^D种布局。为了简化计算量，这里我们只考虑成对的相互影响（pair-wise interactions）,即每两个变量所产生的相互影响有多大。那么，

$$B_A^{\top} W = W^0 + \sum_{i=1}^{D} W_i^1(A) + \sum_{j=1}^{D} \sum_{k=j+1}^{D} W_{j,k}^2(A)$$

其中W^0是 common bias weight, W_{i}^1是第i个变量的权重， $W_{\mathrm{j},k}^2$是第j, k个变量相互影响的权重

- 考虑环境变量：

$$B_{A,X}^{\top} W = W^0 + \sum_{i=1}^{D} W_i^1(A) + \sum_{j=1}^{D} \sum_{k=j+1}^{D} W_{j,k}^2(A) + \sum_{l=1}^{L} W_l^c(X) + \sum_{m=1}^{D} \sum_{n=1}^{L} W_{m,n}^{1c}(A, X)$$

基本算法：Thompson Sampling

- 基本的算法思路使用了Thompson Sampling的概念，其基本思路为利用观察到的奖赏reward来更新每个feature $B_{A,X}$的权重W的后验分布。W 和 $B_{A,X}$ 之后可以用来推断奖赏的概率分布。 具体可以概括为4部分

1. 从**W** 的后验分布中sample 出当前的**W**值；
2. 利用当前的**W**值选出最优的Widget A (arm)；
3. 利用当前的**A**, **X** 获取reward R
4. 利用reward R 更新 **W** 的后验概率分布

Algorithm 1 Thompson Sampling for Contextual Bandits

1: **for all** $t = 1, \ldots, T$ **do**
2: Receive context X_t
3: Sample $\tilde{W}_t$ from the posterior $P(W|\mathcal{H}_{t-1})$
4: Select $A_t = argmax_A \, B_{A,X_t}^{\top} \tilde{W}_t$
5: Display layout A_t and observe reward R_t
6: Update $\mathcal{H}_t = \mathcal{H}_{t-1} \cup (A_t, R_t, X_t)$

注：

- 第一步中因为W的后验分布是高斯分布，Sample并不困难。 Simple 可以简单理解为生成对应分布的随机变量。
- 第三步换句话讲就是知道了环境和系统的layout，将其展示给用户以获得用户的反应
- 之后的篇幅将着重讲解如何实现第二步和第四步

基本算法：Hill Climbing Optimization

- 在算法的第二部中，已知当前的X_t,$\mathbf{W}_t$， 想要找到A_t使B^T_{A,X_t}最大， 其实是一个NP-hard的问题。 因为A 有O（N^D）种选法。为了简化这个流程，现在介绍一种Hill Climbing Optimization的方法：

1. 从一个随机layout A_0起始；
2. 随机选择一个widget i 进行优化 : 对 ith widget的 N种选择，找到第j种使 $j^* = \underset{j}{argmax} B^\top_{A^{k-1} \leftarrow (A[i]=j), X} \tilde{W}$ （$B^T_{A,X_t} W$最大）
3. 使用第j种选择更新A的widget i
4. 重复2，3直到对每个widget而言都无法进行优化或一共进行K次 （K为预设的最大运行次数）

注：

- Hill Climbing Optimization方法很有可能converge到一个局部最优解，而无法获得global optimum. 一个解决方法是尝试多种不同的初始layout， 然后从中选出最优的一个。

Algorithm 2 Hill climbing with random restarts

```
function HILL CLIMBING SEARCH(W̃, X)
    for s = 1, ..., S do
        Pick a layout A_s^0 randomly
        for k = 1, ..., K do
            Randomly choose a widget i to optimize
            Find j* = argmax_j B^⊤_{A_s^{k-1} ← (A[i]=j), X} W̃
            A_s^k = A_s^{k-1} ← (A[i] = j*)
    s* = argmax_s B^⊤_{A_s^K} W̃
    return A^K_{s*}
```

基本算法：Update $P(W|H_t)$

- 让我们重新回顾算法第四部，当我们使用选取的最优layout(arm) A_t 进行测试，我们会获得这个arm带来的reward R。这里给出如何使用R 来更新系统参数**W** 的后验分布。
- 整体的流程可以看作是这样的：$P(\mathbf{W}|H_{t-1})$ 可以看作是W 的前验分布，我们sample 出 向量 $\mathbf{W}_{t-1}$ （高斯随机变量），利用$\mathbf{W}_{t-1}$选取layout（使用之前的Hill Climbing算法）进行测试，最后使用测试的结果R 更新$P(\mathbf{W}|H_t)$ 。
- 在这里我们假设$\mathbf{W} = (w_1, w_2, w_3, \dots, w_m)$ 是一个m 维向量，其中每一个元素都是一个高斯随机变量。那么，其期望和方差的向量可以表示为：

$$\boldsymbol{\mu}_W = (\mu_{w1}, \mu_{w2}, \mu_{w3}, \dots \mu_{wm})$$

$$\boldsymbol{\sigma}_W^2 = (\sigma_{w1}, \sigma_{w2}, \sigma_{w3}, \dots \sigma_{wm})$$

基本算法：Update $P(W|H_t)$

- 我们要做的就是更新W的期望和方差向量，这里给出更新公式：

$$\tilde{\mu}_i = \mu_i + R \cdot b_i \cdot \frac{\sigma_i^2}{\Sigma} \cdot V\left(\frac{R \cdot B_{A,X}^T \bar{\mu}}{\Sigma}\right)$$

$$\tilde{\sigma}_i^2 = \sigma_i^2 \left[1 - b_i \frac{\sigma_i^2}{\Sigma} \cdot W\left(\frac{R \cdot B_{A,X}^T \bar{\mu}}{\Sigma}\right)\right]$$

where: b_i 是 $B_{A,X}$ 中对应 w_i in 元素，值为0或1

$$\Sigma = 1 + B_{A,X}^T \bar{\sigma}^2$$

$\bar{\mu}, \bar{\sigma}$ 为 期望，方差向量

$$V(t) = \frac{\mathcal{N}(t;0,1)}{\Phi(t;0,1)} \quad \mathcal{N}(t;0,1) \to \text{pdf}, \quad \Phi(t;0,1) \to \text{cdf}$$

$$W(t) = V(t) \cdot [V(t) + t]$$

简略的推导过程在附录

其中V(t)函数是标准高斯分布的pdf除以cdf

带有~的值是更新后的期望和方差，不带的为更新前的

最终奖赏预测

- 当我们做了足够多的实验， 我们或许想知道的是给定一个layout A, 其reward R的分布是什么，这里给出reward的分布:

$$P(R|\bar{A},\bar{x}) = \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} P(R|\bar{A},\bar{x},\bar{w})\, p(\bar{w})\, d\bar{w}$$

$\bar{w}$ 是 weight vector，其元素均为高斯随机变量

利用Bayes：

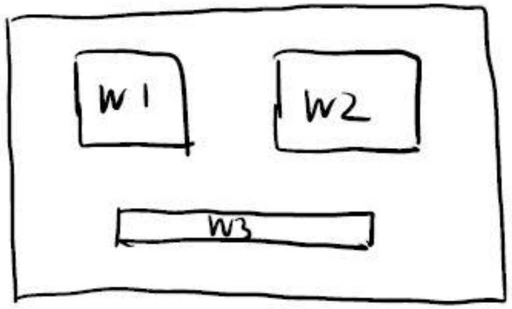
$$P(R|\bar{A},\bar{x}) = \Phi\left(\frac{R \cdot B_A^T \bar{x}\, \bar{m}}{\Sigma}\right)$$

where $\Sigma = 1 + B_A^T \bar{x} \cdot \bar{\sigma}^2$

实例讲解

实例讲解：

widget (D) = 3, N=3, MVT2：只有widget，没有环境变量

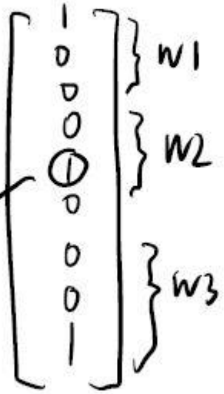


W1, W2, W3 各有3种选择，分别为 $W_{1\sim3,1\sim3}$

假设 W1 取其第一种，W2取第二种，W3取第三种.

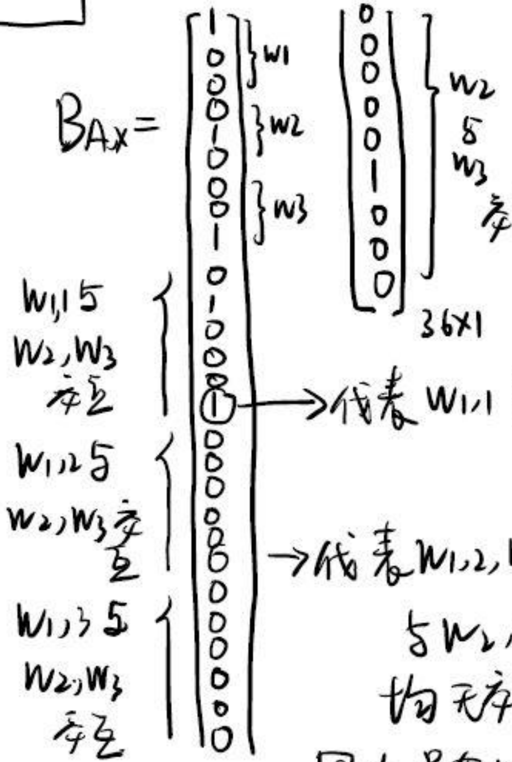
↓ $W_{1,1}$ 代表 W1选第一种

$A=$



代表widget2取了第二种选择.

$B_{Ax}=$



→代表 $W_{1,1}$ 与 $W_{3,3}$ 有交互.

→代表 $W_{1,2}$, $W_{1,3}$ 与 W_2, W_3 均无交互，因为只有 $W_{1,1}$ 被选中

$$B_{Ax}^{T}\cdot W = W_0^1 + W_{1,1}^1 + W_{2,2}^1 + W_{3,3}^1 + W^2_{1,1\leftrightarrow2,2} + W^2_{1,1\leftrightarrow3,3} + W^2_{2,2\leftrightarrow3,3}$$

其中 W^1 代表widget的，W^2 代表widget交互的权重(weight)

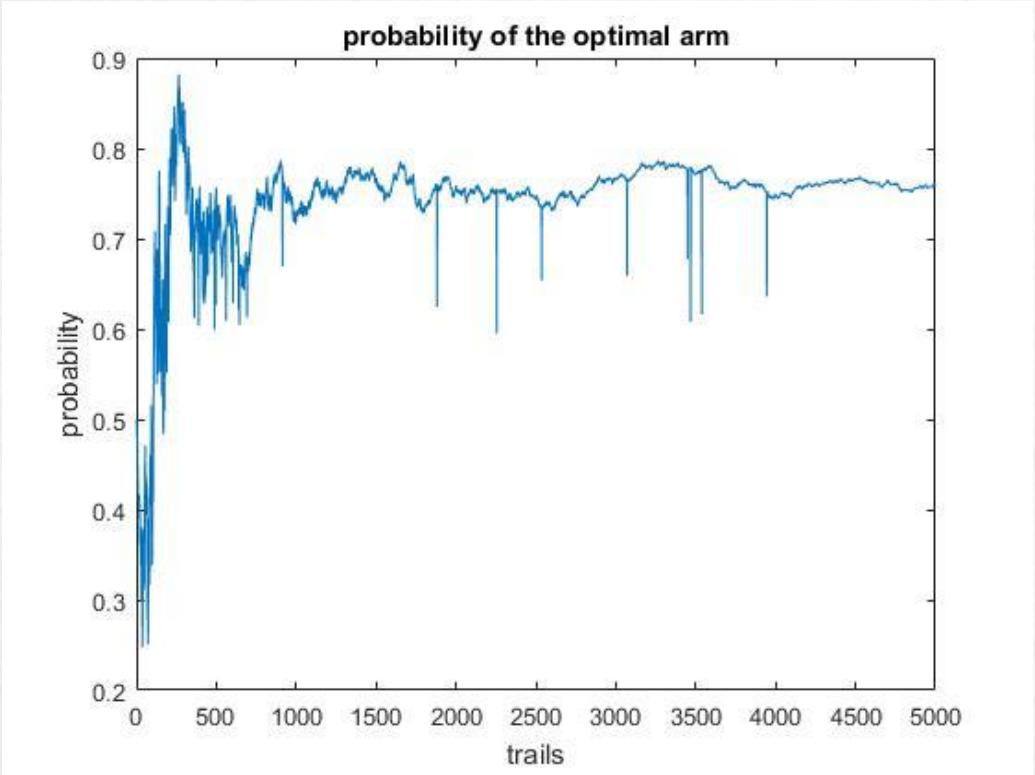
仿真

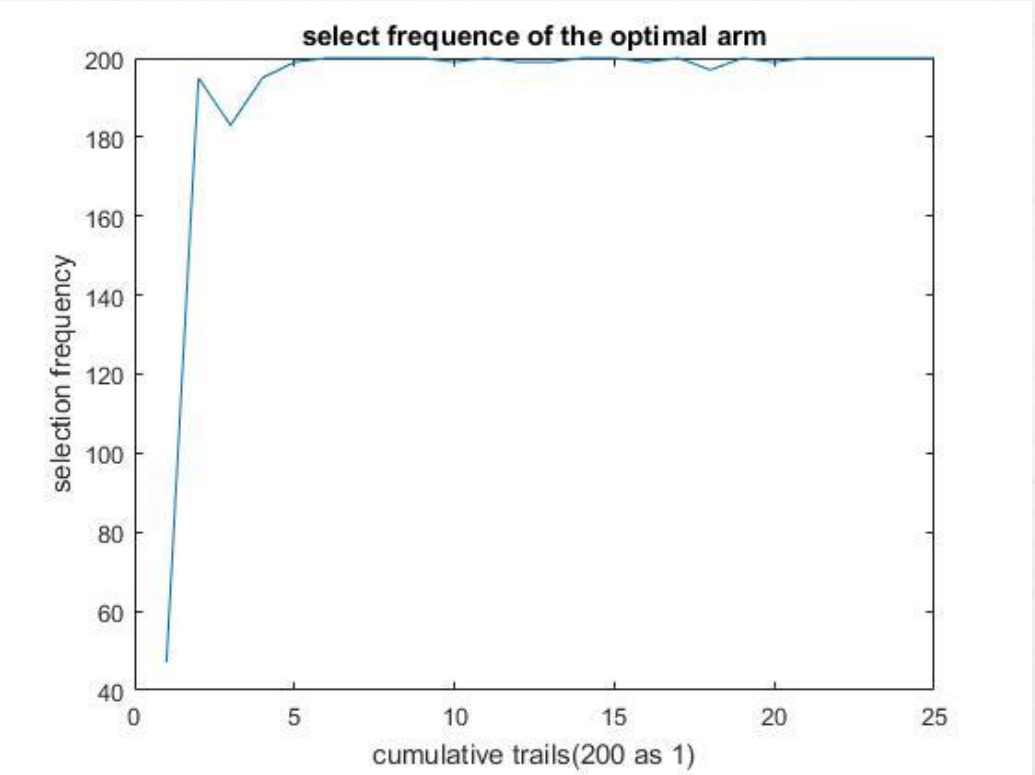
仿真参数:

- 采取MVT2 进行仿真，即只有widget 的权重而没有环境变量
- 选取layout时使用brute force方法
- 小数据N = 3， widget =3
- 大数据N=4， widget = 8
- 最优奖赏概率 = 0.75
- 其余奖赏概率uniformly distributed ~(0,0.7)

仿真

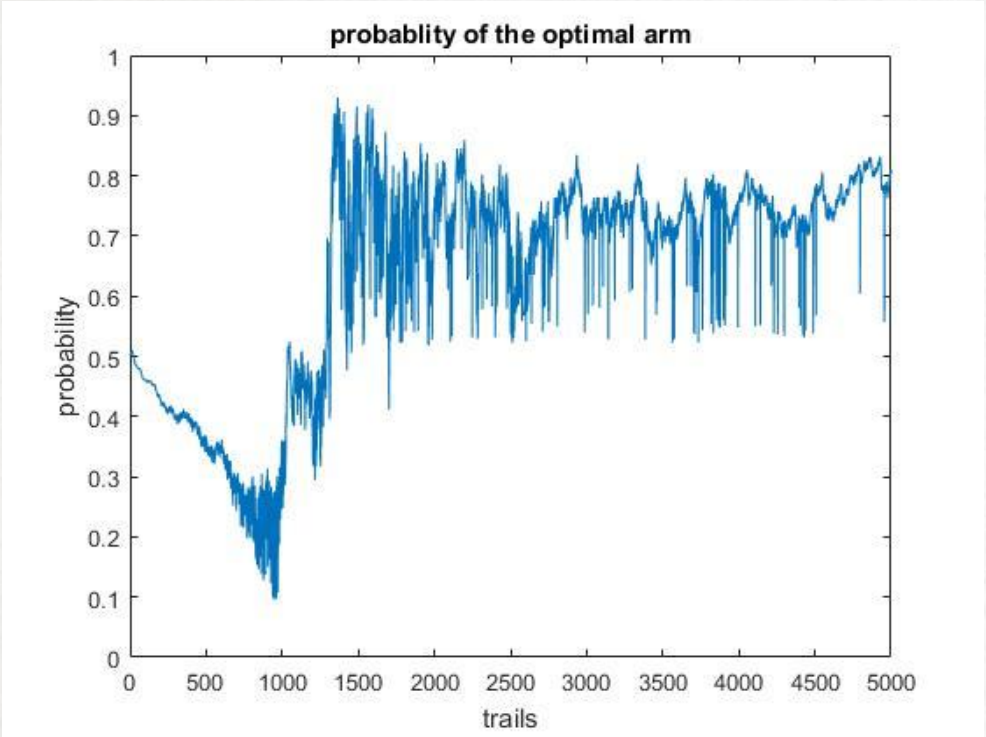
小数据N = 3， widget =3

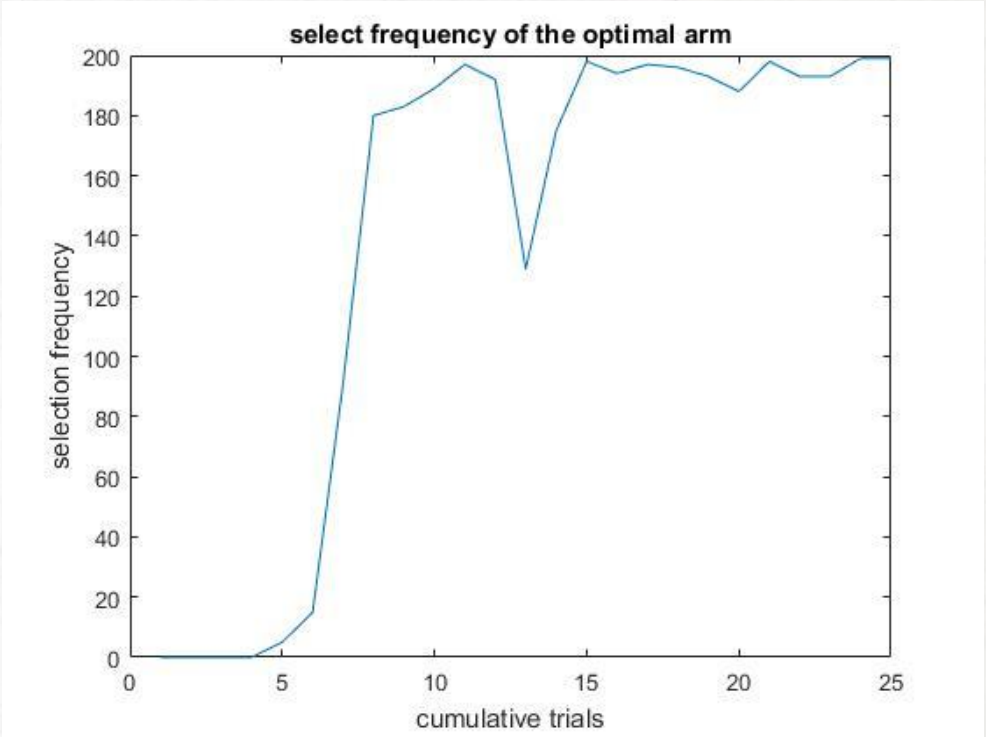




仿真

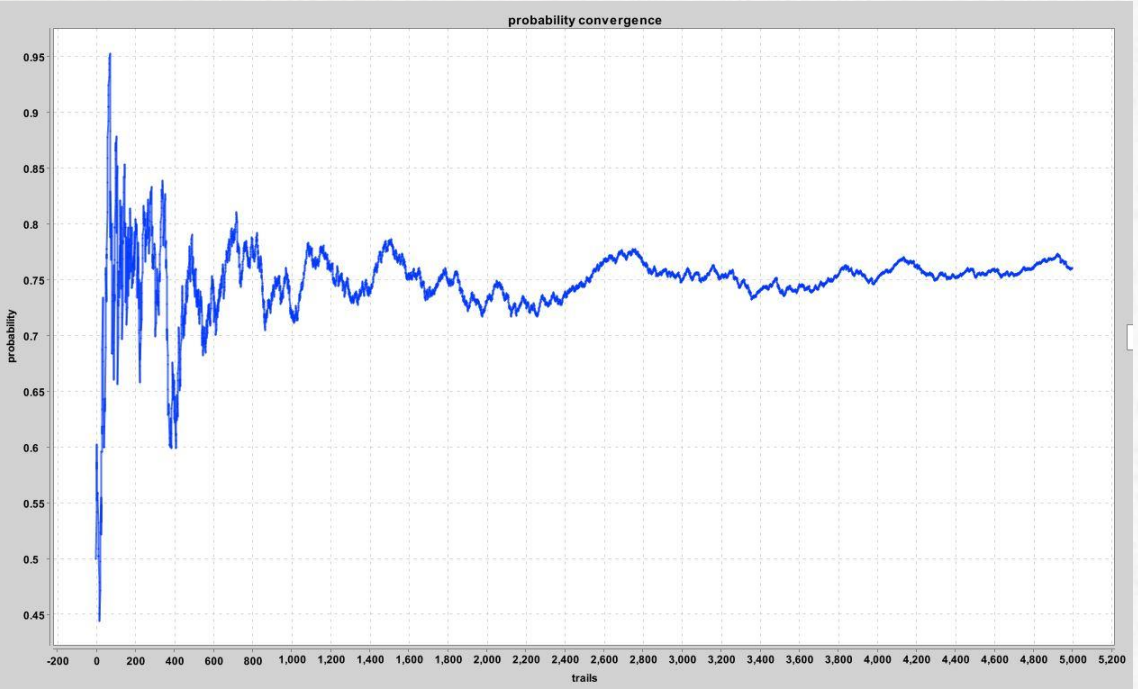
大数据N=4， widget = 8

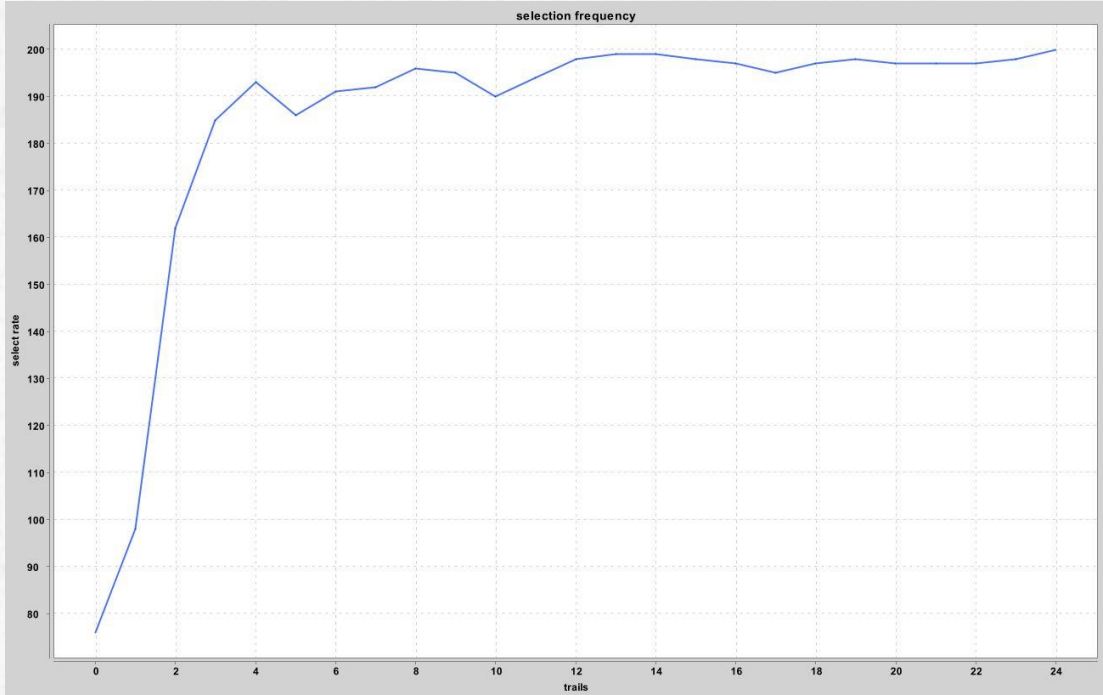




仿真 java code

小数据N=3， widget = 3





讨论

- 加入新的widget

本篇文章及其相关文章并没有讨论现有算法能否动态的加入新的widget。

就机器学习的角度来看。假设在模型已经训练好的情况下，新加入一个（或多个）widget， 模型要经过重新训练。

就人工智能而言，用人脑模仿选择过程，在已经建模好的情况下，新加入widget，因为对新加入的widget没有任何prior information，在训练中仍应优先选择原来的最优解， 继续经过一段时间训练会得到model的更新。

因此，我认为在原有算法实施中是可以加入新的widget。

讨论

- 更加开放 （generalized）的layout设计

在本篇文章中，layout的设计是由用户预先定好widget的数目，以及每个widget的选择个数。就算可以实现在算法过程中动态的加入新的widget，仍然无法实现完全的AI layout设计。

事实上，有另一篇文章有谈及到这方面的设计[1]。该文章主要讲述如何构建搜索结果页。在一个用户对关键字搜索后，得到的或许会包含新闻，图片，视频，地图等等完全不同属性的结果，如何在搜索结果页中最优的展示这些结果是这篇文章的主要研究方向。该文章加入了position features。

1. Flavio Chierichetti, Ravi Kumar, and Prabhakar Raghavan. 2011. Optimizing two-dimensional search results presentation. In Proceedings of the fourth ACM international conference on Web search and data mining. ACM, 257–266.

讨论

- 更加复杂的交互设计

在本篇文章中，无论是widget交互还是widget-environment交互均使用了简单的两两交互原则（pairwise）。 这种交互方式的确能节省时间，但是相比于更复杂的交互（nxn交互）是否能高效的达到最优解是一个问题。

文章介绍了一些相关研究。

在[1]中更复杂的交互通过factorization来实现

在[2]中则使用神经网络来模拟

1. Steffen Rendle. 2010. Factorization machines. In Data Mining (ICDM), 2010 IEEE 10th International Conference on. IEEE, 995–1000.
2. Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence

引用

- Amazon : “An Efficient Bandit Algorithm for Realtime Multivariate Optimization”
- Microsoft : “Web-Scale Bayesian Click-Through Rate Prediction for Sponsored Search Advertising in Microsoft’s Bing Search Engine”
- Microsoft： ”TrueSkill™ : A Bayesian Skill-Rating System”