概述

首先需要了解推荐系统的一个痛点：E&E 问题，即开采（Exploitation）和探索（Exploration）问题。

E&E 问题可以分为两个子问题：

1) 探索问题：推荐系统需要不断探索用户的新兴趣，给用户以新颖的体验。
2) 开采问题：推荐系统需要不断拓宽用户的已有兴趣，给用户以精准的感受。

具有探索性的推荐是新颖的，但也是冒险的，会推荐给用户一些新的 Item 去试验用户的反馈，因此是转化率低的，是对未来有益对当前有害的。

具有开采性的推荐是精准的，但同时也是竭泽而渔的，会推荐给用户一些已知的兴趣点，长久之后，用户会觉得无趣，是对当前有益但对未来有害的。

探索和开采问题是互相矛盾的，如何权衡两者的指标，是面临的挑战。

本方案通过改进的 Bayesian Bandit 算法，在功能推荐场景取得了不错的效果，具体的创新点概括为：

1) 场景选择的要点：可重复推荐
2) 先验生成的方法：基于统计的先验
3) 原算法的优化：时间衰减和加速收敛技术
4) 线上实现方案：基于 hive 的 udf 实现的 bayesian bandit 算法和一整套工作流程

本专利申请对以上四点创新点进行保护，下文会对创新点详述。

1 现有技术

1.1 现有的技术方案

为了解决 E&E 问题，有很多方案可以选择，下面介绍几种经典的做法：

1) 轮播技术

为了保持用户的新鲜感，对推荐列表进行轮播，但是这种方案是原始的，而且轮播到尾部列表之后转化率会很低，属于最基础的技术。

2) e-greedy

为了增加新颖性，对原有的列表以 e 的概率随机穿插新的 Item，此种方案比较暴力，不能够及时的开采用户的已知兴趣，不能智能识别此时此刻探索和开采各应该占的比重是多少，也属于比较简单的做法。

3) UCB

选择置信区间的上限，作为 pCTR 的值，是 Bayesian Bandit 算法的主要竞争者。

在实践上，其缺点是因为取的是置信度上限，所以初期的转化率往往很低，初期对用户伤害很大，而且缺少随机性，如果用户没有反馈数据，列表会一成不变。

在理论上，没有 Bayesian Bandit 的理论那么优雅，UCB 更偏向于为解决问题而产生的算法。

1.2 现有技术的缺点

与上述的算法相比，Bayesian Bandit 的优势在于：

1) 智能平衡探索和开采：算法初期偏向于探索，算法后期偏向于开采，在整个迭代过程中，最大化用户的收益价值。
2) 理论完备优雅：利用伯努利分布和 Beta 分布的共轭性，进行随机采样，理论完备。
3) 无前期伤害：此主要是相对于 UCB 而言，Bayesian Bandit 并不会明显偏向于新 Item 的尝试，所以对于海量的 Item 推荐，有 UCB 无法相比的优势。
4) 具有随机性：具有随机性，及时用户无任何反馈行为，列表也可能发生变化，解决了新颖性的问题。
5) 参数可调&会遗忘：改进的 Bayesian Bandit 算法将更多的参数暴露出来，方便切合线上场景；加入时间衰减系数，会更加切合用户当前的状态。

2 本技术方案的详细阐述

2.1 技术侧

（1） 场景选择要点：可重复推荐

使用 Bayesian Bandit 之前，首先要理解，什么样的场景是适合的，什么样的场景是不适合的。典型的推荐场景有：

1) 手管软件管理的 APP 推荐
2) 手管广告推荐
3) 手管功能推荐

在以上场景中，可以分为两类推荐：可重复推荐和不可重复推荐。

可重复推荐指的是，转化之后的用户还可以转化，比如上述的“手管功能推荐”中的“拉活推荐”，以及“天猫店铺推荐”等等用户正向操作会加大用户选择该 Item 的概率的场景。

相反，不可重复推荐的场景，转化之后不能或者很难继续转化，比如上述的“小管推荐”中的“拉新推荐”，以及“游戏道具推荐”等等用户正向操作后会明显减小或者不可能继续选择该 Item 的场景。

（2） 先验的生成：基于统计的先验

Bayesian Bandit 算法的初始化，是通过统计的方法，将用户无差别的点击行为转化为 Item 自身的属性进行初始化的，其结果类似于转化率最高的“非个性化推荐”——CTR 热门推荐。

手管的功能项有 40+个，将随机模型回收到的反馈数据进行整理，计算 CTR 就可以得到“CTR 热门模型”非个性化推荐。

功能项 ID	点击	曝光	CTR
38	177	437	0.41
53	119	303	0.39
52	Click	Explode	Click/Explode
…			

表 3.2.1 CTR 热门表

其中，

功能项 ID 表示待推荐的功能标识；
点击表示日志回收后统计的点击次数；
曝光表示日志回收后统计的曝光次数；
CTR 表示统计得到的 ctr，其中$ctr = click/explode$。

因为篇幅限制，只展示了 3 个功能项的计算方法，将功能项按照点击率排序，可以表达为下图：

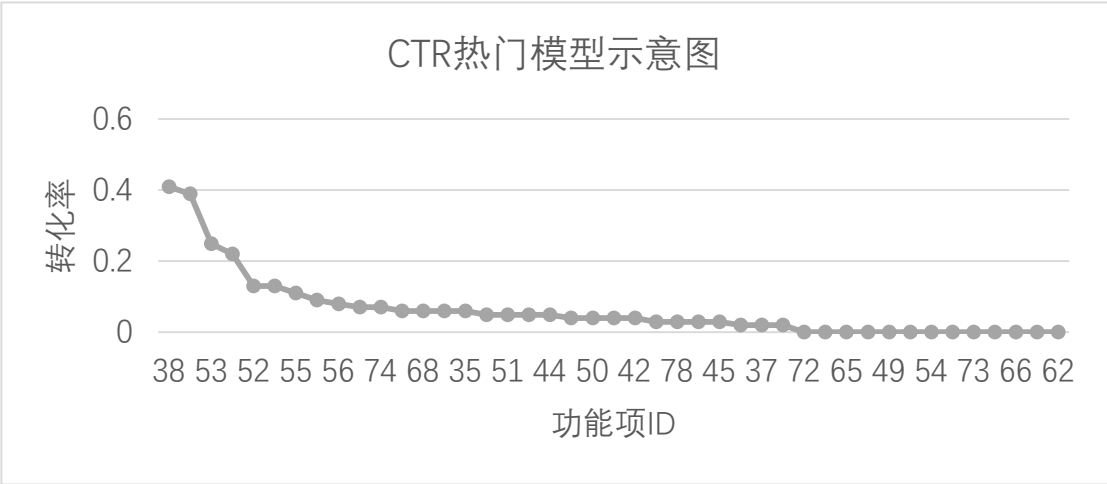


图 3.2.1 CTR 热门模型排序图

其中，
横轴表示功能项 ID，即 Item 的 Key；
纵轴表示 CTR，即点击除以曝光；
整个图表按照 CTR 降序排列，优先把 CTR 高的功能项 ID 推荐给用户。

在热门模型中，默认把 CTR 高的 Item 推荐给用户，相当于将 pCTR 和 CTR 设置为相同，即用户无差别的进行推荐，此即为非个性化模型中，点击率最高的模型。

$$pCTR_{u,i} = CTR_i \qquad \text{公式 } 3.2.1$$

其中，
$pCTR_{u,i}$表示对用户 u 和功能项 I 的 CTR 预估；
CTR_i表示功能项 I 的统计 CTR。

CTR 热门模型往往是初期最简单，最有效果的模型；最简单值得是计算简单，上线简单；最有效果指的是在非个性化模型中，CTR 热门的转化率最高。

有 CTR 热门模型，可以得到一个功能项的基本属性，该基本属性可以用来初始化 Bayesian Bandit 算法先验参数。

功能项 ID	CTR	初始化 A	初始化 B
38	0.41	41	59
53	0.39	39	61
52	Click/Explode	Round(Click/Explode)*S	S-Round(Click/Explode)*S
…			

表 3.2.2 Bayesian Bandit 模型初始化参数

其中，
初始化参数 A 等于 CTR 四舍五入取整后乘以一个参数 S（表中 S=100）；
初始化参数 B 等于 100-A；
初始化的参数 A 和参数 B 是 Beta 分布的参数 A 和 B，有 Beta（A，B）分布可以采样得到一个[0, 1]的值，该值作为 pCTR 的值，在后续会有介绍。

（3） Beta 分布的特性

Bayesian Bandit 算法基于一个 Beta 分布的采样，首先了解一下 Beta 分布的形态，以 Beta(3, 2)为例来解释，如下图：

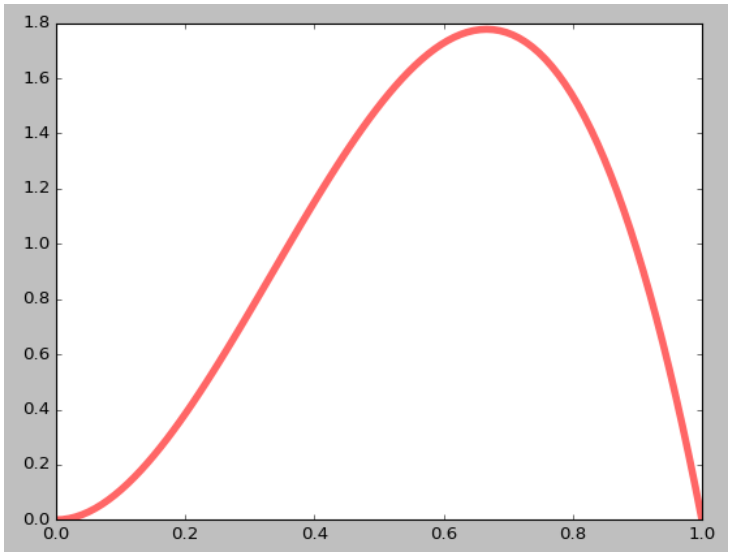


图 3.2.2 Beta(3, 2)分布

其中，

A=3，是 Beta 分布中的第一个参数，代表正样本出现的次数+1；

B=2，是 Beta 分布中的第二个参数，代表负样本出现的次数+1；

横轴的区间为[0, 1]，代表 pCTR 的预估值；

纵轴表示 pCTR 值的概率密度，其值越大，表示采样到该值的几率越大。

上图总体展示了 Beta 分布，下面对 Beta 分布进行不同参数的对比，来了解 A 和 B 参数对 beta 分布形态的影响，如下图：

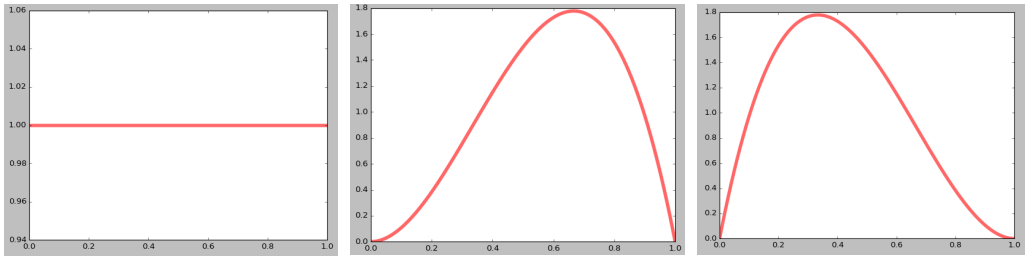


图 3.2.3 分布对比图 1(左到右分别是 Beta(1, 1)，Beta(3, 2)和 Beta(2, 3))

其中，

Beta(1, 1)表示为一条直线，是均匀分布，采样到每个点的概率相同，相当于随机模型；

Beta(3, 2)表示为顶峰在左侧的一个山峰状分布，其极值点为 0.67，采样到 0.67 附近的概率比较大；

Beta(2, 3)表示为顶峰在右侧的一个山峰状分布，其极值点为 0.33，采样到 0.33 附近的概率比较大。

说明在 Beta(A, B) 分布中，A 值越大，越有可能采样到大的值；B 值越大，越可能采样到小的值。

上图展示了 A 和 B 的相对值对 Beta 分布形态的影响，下面讨论 A+B=S 中，S 值对 Beta 分布形态的影响，如下图：

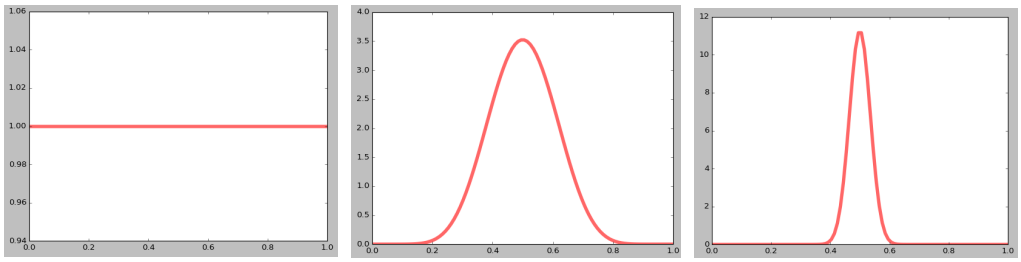


图 3.2.4 对比图 2（左到右分别是：Beta(1, 1)，Beta(10, 10)和 Beta(100, 100)）

其中，

Beta(1, 1)分布没有呈现山峰状，采样随机；

Beta(10, 10)分布呈现较低矮的山峰状，采样集中在 0.5 附近，波动比较大；

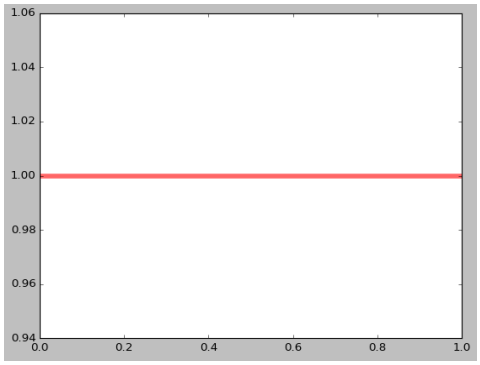
Beta(100, 100)分布呈现较高耸的山峰状，采样集中在 0.5 附近，波动比较小；

说明 A+B=S 中，S 值越大，随机性越差，采样越接近极值点。

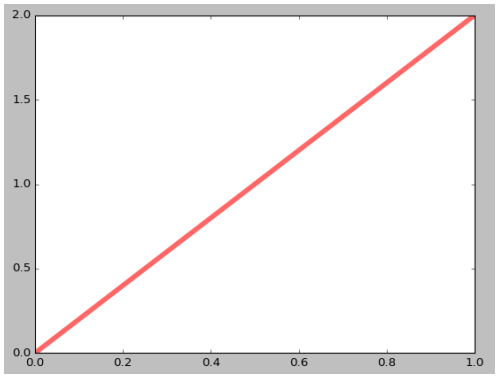
在伯努利实验中，独立重复试验的结果呈现伯努利分布，而一个有趣的性质是伯努利分布和 Beta 分布是共轭分布，如下：

$$Beta分布先验 + 伯努利事件 = Beta分布后验$$ 公式 3.2.2

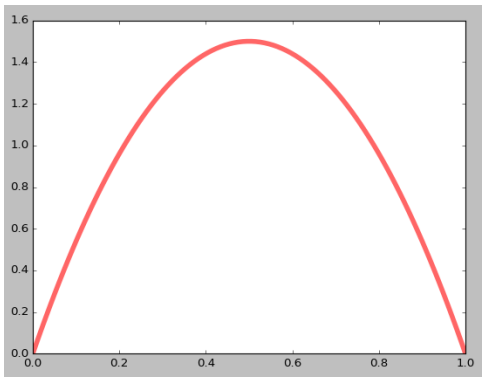
在此处不做证明，仅对这个分布的性质做一下说明，如下图：



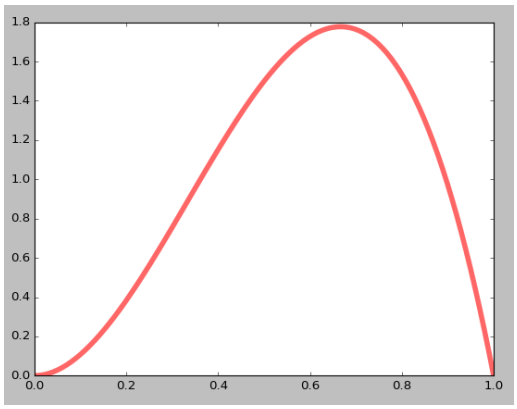
子图 1-Beta（1,1）分布



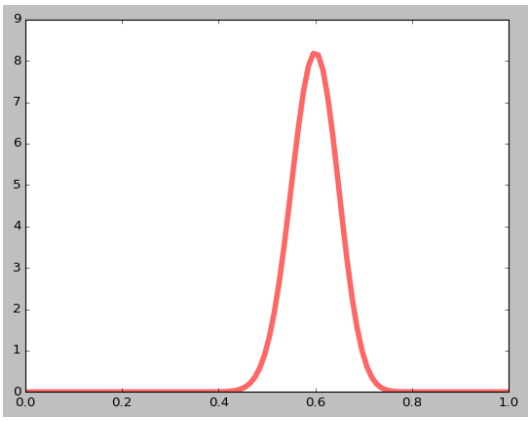
子图 2-Beta（2,1）分布



子图 3-Beta（2,2）分布



子图 4-Beta（3,2）分布



子图 5-Beta（61,41）分布

图 3.2.5 Beta 分布变迁图

其中，

子图 1 表示，在最开时的情况，并不知道概率需要以什么分布存在，依据最大熵的原理，pCTR 符合 Beta(1, 1) 分布；

子图 2 表示，经过了一次曝光点击事件，验证该用户偏好该功能项，因此 A 加 1，变为 Beta(2, 1)分布；

子图 3 表示，经过了一次曝光未点击事件，验证该用户不偏好该功能项，因此 B 加 1，变为 Beta(2, 2)分布；

子图 4 表示，经过了一次曝光点击事件，A 加 1，变为 Beta(3, 2)分布，在 0.67 处取得极值点，山峰较平缓，采样波动大；

子图 5 表示，经过了一系列曝光点击和曝光未点击事件，A 加 58，B 加 39，变为 Beta（61，41）分布，在 0.6 处取得极值点，山峰较陡峭，采样波动小。

此即为 Beta 分布的特性，用户的曝光和点击操作导致的 pCTR 的分布非常类似于上述的分布变化，Bayesian Bandit 算法即为模拟以上过程的一种采样算法。

（4） Bayesian Bandit 的算法和 Hive 实现

通过对上述过程的理解，可以把 Bayesian Bandit 算法做以下简要的描述：

Bayesian Bandit 算法
1. 初始化-1：计算统计 CTR，根据参数 S，计算每个待推荐项 i 的参数A_i和B_i，并对所有用户 u 设置$A_{(u,i)} = A_i$，$B_{(u,i)} = B_i$
2. 初始化-2：对每个用户 u 和每个待推荐项 i 进行$Beta(A_i, B_i)$的采样，得到p$CTR_{(u,i)}$的值
3. 初始化-3：对每个用户 u 的功能项p$CTR_{(u,i)}$进行排序，做初始化推荐
4. For u 对 i 发生的每个事件 $e_{(u,i)}$（曝光点击 e=1，曝光未点击 e=0）：
5. $A_{(u,i)} = A_{(u,i)} + e_{(u,i)}$
6. $A_{(u,i)} = A_{(u,i)} + 1 - e_{(u,i)}$
7. 对每个用户 u 的功能项p$CTR_{(u,i)}$根据 $Beta(A_{(u,i)}, B_{(u,i)})$进行采样，并更新推荐列表

表 3.2.3 Bayesian Bandit 算法

该算法的关键是 Beta 分布的采样，有两种实现方法：

1) 使用 python 中的 numpy 库
   numpy 库中有 Beta 分布的采样函数，可以通过 numpy.random.beta(A, B)得到。
2) 使用 java 库并包装称为 Hive 的 udf
   Randoms r = new Randoms()
   r.next.Beta(A, B)

第二种方法更能有效使用 hive 处理结构化数据，线上采用第二种方案。

（5） 时间衰减因子和参数选择：可调节的 A，B，S，K 和半衰期 P

为了更好的适应线上环境，对原 Bayesian Bandit 算法进行了更改，形成了改进的 Bayesian Bandit 算法，具体如下：

1) 加入参数 K：原算法收敛过慢，因此我们将用户每次更新的事件 e 扩大 K 倍，能有效加速收敛。K 值不能设置过大，过大容易导致收敛过快，探索的性能降低。
2) 加入天级别半衰期 p：原算法收敛后，探索性能逐渐降低，为了保证探索性能，加入半衰期参数 p，每天衰减为昨天的 p 倍，能够忘掉时间过于久远的数据，更符合现在的用户状体，同时也保证了 A 和 B 值不至于过大，保留了探索性能。
3) 天级别更新：原有算法是在线更新的，能够随时更新 A 和 B 的值，为了适应目前的天级别 Hive 计算，将算法按照天级别更新。

改进后的 Bayesian Bandit 算法如下：

改进的 Bayesian Bandit 算法
1. 初始化-1：计算统计 CTR，根据参数 S，计算每个待推荐项 i 的参数A_i和B_i，并对所有用户 u 设置$A_{(u,i)} = A_i$，$B_{(u,i)} = B_i$
2. 初始化-2：对每个用户 u 和每个待推荐项 i 进行$Beta(A_i, B_i)$的采样，得到p$CTR_{(u,i)}$的值
3. 初始化-3：对每个用户 u 的功能项p$CTR_{(u,i)}$进行排序，做初始化推荐
4. For 每天 u 对 i 发生的每个事件 $e_{(u,i)}$（曝光点击 e=k，曝光未点击 e=0）：
5. $A_{(u,i)} = \left(A_{(u,i)} + e_{(u,i)}\right) * p$
6. $B_{(u,i)} = \left(B_{(u,i)} + K - e_{(u,i)}\right) * p$

7. 每天，对每个用户 u 的功能项$pCTR_{(u,i)}$根据 $Beta(A_{(u,i)}, B_{(u,i)})$进行采样，并更新推荐列表

表 3.2.4 改进的 Bayesian Bandit 算法

（6） 作业处理工作流和评估：从数据采集，模型生成，分布采样，离线评估到在线评估介绍完成核心算法原理之后，整体的部署流程是：

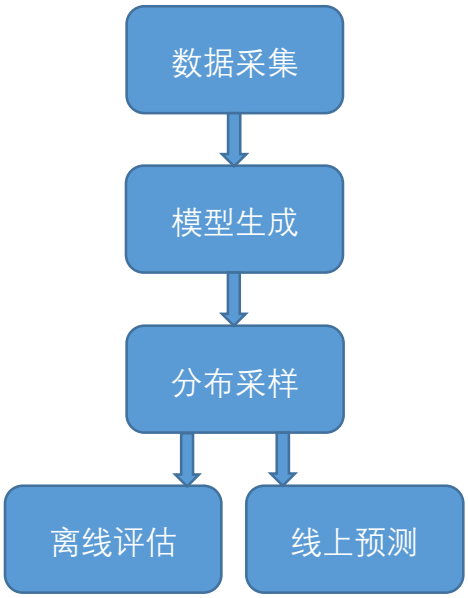


图 3.2.6 改进的 Bayesian Bandit 算法部署流程

其中线上使用的参数是：

参数	值	含义
S	100	A+B 和，初始化的收敛程度
K	50	加速收敛的速度
P	0.97	每日衰减系数

表 3.2.5 线上应用参数表

其中，离线评估使用 AUC 进行评估，计算得到结果评估表：

算法模型	离线 AUC	线上点击率
随机模型	0.50	5.53%
CTR 热门模型	0.68	9.73%
改进的 Bayesian Bandit 模型	0.69	9.99%

表 3.2.6 Bayesian Bandit 算法评估结果

因此，改进的 Bayesian Bandit 算法在解决重复推荐和 E&E 的问题上，在小管推荐的实践中，转化率可以提升 2.6%，是一种行之有效的方法。

3 在线化

第一，对于重复性推荐项，可以直接在线化，实时收集事件和实时更新 beta 参数即可。

第二，对于 app 推荐，因为不可重复，所以没有正向作用，只有负向反馈，所以相当于负向模型。（展示多次未点击排后面）

第三，在 pCTR 的最后一层，做一层 bandit，参数固定，用来动态解决 EE 问题。

第四，从预估层面就直接 bandit 的 linUCB