arXiv:2302.01522v1 [math.NA] 3 Feb 2023

通过模拟用户兴趣提升推荐相关性

Alexander Kushkuley，kushkuley@gmail.com
Joshua Correa，jcorrea@salesforce.com

2023年2月6日

摘要

大多数（如果不是全部）在线物品到物品推荐系统都依赖于对商品之间某种距离度量（排名）的相似度进行估计。对于在线推荐系统而言，这种相似度度量的时间敏感性极为重要。我们观察到，通过迭代降低非活跃项目的排名，可以直接且透明地维护推荐的新近性。本文简要总结了基于这一不言自明的观察所得的算法发展。该工作的基本思想已在在线推荐系统的背景下获得专利。

1 引言

几乎所有（如果不是全部）在线物品到物品推荐系统（例如参见 [1], [2]）都依赖于产品相似度这一概念，因此，对于一个（锚点）商品 A，A 推荐的商品 B_1、B_2、$\cdots B_m$ 会显示在产品 A 网页上，并按照它们与 A 的相似度排序。为便于后续引用，将此类页面记作

$$A \to (B_1, r_1), (B_2, r_2) \cdots, (B_m, r_m) \tag{1}$$

其中 $\{r_1, r_2, \cdots, r_n\}$ 是与项目列表 $L = \{B_i,\ i = 1, \cdots, m\}$ 相关联的一些数字（排名）。通常，排名 r_i 是通过估算某种定义在所有产品对集合上的成对相似度度量 μ 得出的，因此根据定义（例如参见 [3]）

$$r_i = \mu(A, B_i),\ i = 1, \cdots, m \tag{2}$$

在线商品推荐具有很强的时效性，因此会采用各种技术来确保“距离”度量 μ 与时间相关（例如参见 [3]）。

例如，其中一些技术可能基于事件频率估计（例如参见 [5], [6], [9]）），另一些时间感知方法则采用随机游走（例如参见 [4]）。在某些情况下，还会实施将事件时间戳纳入考量的临时性流程。

我们提出一种简单的事件日志模拟方法，它

- 创建推荐列表 (1)，同时避免进行成对相似度度量估计 (2) - 通过迭代降低非活跃项目的排名来维持商品的“新近性”，从而避免频率计数 - 直接将现有的推荐点击纳入考虑 - 不使用任何时间相关逻辑来调整推荐排名

2 算法描述

Let

$$q_i = \mathbb{P}(B_i|\text{recommendation click on page A}),\ i = 1, \cdots, m \tag{3}$$

表示在用户访问与商品 A 对应的网页时，点击产品 B_i 推荐项的概率。

注释 1 *(1-2)* 中的排名 r_i 可视为概率分布 *(3)* 的估计，因为 *(1)* 中的排名列表可以转换为概率列表 $Q = \{q_1, q_2 \cdots, q_m\}$，例如，设置 $q_i = (1 - q'_i)/(m-1)$其中 $q'_i = r_i / \sum_{j=1}^{m} r_j$

当然，(3) 中的概率 q_i 是未知的（至少无法实时获知），而估计分布 (3) 的标准方法是统计事件频率（参见例如 [5], [6]）。然而，在在线推荐系统中单纯进行频率计数，往往会高估过去曾经热门的商品，并且需要复杂的逻辑规则来调整计数器（不过，参见 [9]）。因此，我们的目标是设计一种内置时间敏感性的简单在线点击概率估计算法。假设我们有一个分类分布

$$P = \{p_1, p_2 \cdots, p_m\} \tag{4}$$

它对应于（如式 (1) 和 (3) 中所示）一个项目列表

$$A \to L = \{B_1, B_2, \cdots, B_m\} \tag{5}$$

请注意，对推荐内容的一次点击 B_i 可以用狄拉克 delta 函数分布表示

$$\delta_i = (0, \cdots, 1, \cdots, 0) \tag{6}$$

在索引集合 $\{1, \cdots, m\}$ 上，除索引为 $i.$ 的元素外，所有元素的概率质量值均为 0。因此，很自然地可以假设，此类点击会按照某条规则对概率分布 P 进行凸混合更新

$$P \to \alpha P + (1-\alpha)\delta_i, \;\; 0 < \alpha < 1$$

换言之，点击推荐 B_i 会降低所有其他页面推荐的排名，同时以尽可能简单的方式使估计的排名分布倾向于推荐商品 B_i 。由此，我们得到以下在线推荐算法。

算法 1 固定一个接近 *1* 的实数 $0 < \alpha < 1$ （下文称为排名降低参数）和一个较小的阈值 ϵ （例如 $\epsilon = .001$）。点击带有锚点 A 的推荐 B，会对与列表 *(5)* 相关联的分类分布 *(4)* 进行如下变换：

1) 如果 $B = B_j \in L$，则设置 $p_i \to \alpha p_i$、 $i = 1$、$\cdots n$、 $i \neq j$，并设置 $p_j \to \alpha p_j + 1 - \alpha$。*2)* 如果 $P = \emptyset$ ，则将 A 推荐的项目列表设置为 $\{B\}$ ，并设置 $P = \{1\}$。*3)* 如果 $B \notin L$ ，则可以将其加入列表，加入概率小于 $min\{p_i, i = 1, \cdots, m\}$ ；或者，更好的策略似乎是按照附录 *1 (10-11)* 中所述，通过计算最大熵混合分布来添加新商品。*4)* 从 L 中移除所有满足 $p_i < \epsilon$ 的商品 $B_i \in L$ ，并重新缩放生成的列表，使剩余概率之和为 *1*。

值得一提的是，该算法有两项“内置”功能：

- 递减 α 会提高近期点击推荐事件的优先级
- 因此，只需调整一个参数，就能轻松控制该排名方案对新推荐事件的敏感度（甚至可以在运行时调整）

备注 2 （参见例如 *[7]*）。假设希望某个商品在处于闲置状态期间，其所属列表被更新 T 次后，排名降低一半。将参数 α 设置为 $\exp(-\log(2)/T)$ 即可实现这一点。例如，如果 $T = 10$ ，那么 $\alpha \approx .93$

备注 3 与在重击者（大象流）估算场景中采用的“指数衰减计数策略”有些相似的方法，最近在 *[9]* 中被提出。

关于支持算法 1 的一些直观“理论”考量，请参见[10] 和附录 2。附录3介绍了用于评估该算法的仿真模型。该算法的一些有用特性可概括如下：

- 不受欢迎的（空闲）推荐项的排名概率将以指数速率降低 α^t （对于任意 t 次未点击该商品的情况）
- 被点击的推荐项概率 p 会乘以一个乘数

$$\alpha + p^{-1}(1-\alpha) > 1$$

其幅度与该概率成反比。因此，低排名推荐项的相对点击后提升高于热门推荐项；换言之，该算法倾向于提高迄今为止访问次数较少的商品的可见性

- 如前所述，“空闲衰减”和“活跃度提升”的速率均仅由一个参数控制

- 任何地方都不会执行依赖时间的逻辑

- 该算法可以与部署在（网站）上的任何其他推荐系统并行运行，也可以作为其增强功能运行

2.1 事件日志仿真

遗憾的是，对于大型在线系统，算法 1 的实时实现可能会很复杂，因为需要同时更新许多异步变化的商品。因此，作为一种实用方案，该算法的半在线仿真版本按如下方式实现：

算法 2 按照固定的时间间隔（例如每小时），通过模拟最新未处理事件日志中的所有推荐点击事件，将算法 *1* 应用于一组列表 *(5)*。由此得到的概率分布集合会被记录下来，并在下一次模拟运行中重复使用。

2.2 秩降低参数选择

选择参数 α 有多种方法。目前采用的方法使用以下数据:

- 一个时间“兴趣周期” T （例如一周），其选择依据是：商品闲置时间达到 T 后，其排名会显著降低（例如减半）- 根据事件日志计算得出的当前平均单品点击率

有了这些数据，每次运行模拟时都可以（重新）计算参数 α （参见例如 [7] 和备注 2）

2.3 两项改进

我们的第一个观察是，同一产品可以在多个网页上获得推荐，也就是说，产品 X 可以由若干产品 A， B， C... 推荐。这带来了对算法 1 的如下修改

算法 3 当推荐 B 被点击时，对所有已记录的推荐列表 $A' \to L'$ 应用算法 *1*，使得 $B \in L'$。

备注 4 作为对上述方法的修改，可以考虑根据与 A 的接近度标准筛选锚点 A'

除了推荐点击之外，考虑其他事件也有意义。在线零售系统通常会跟踪的重要事件当然包括结账事件和“加入购物车”事件。当一个合理的推荐器运行时，这些事件的发生频率会低于推荐点击。在这种情况下，可以考虑对算法 1 和 2 进行如下修改

算法 4 。

1) 在模拟启动时，收集并维护最近已结账商品和已加入购物车的商品列表。*2)* 按照第 *2.2* 节所述，分别计算用于推荐点击、结账和加入购物车事件的排名降低参数 $\alpha_R, \alpha_C, \alpha_A$ （基于各自的点击率）。*3)* 当推荐商品最近已被结账或加入购物车时，使用步骤 *2)* 中计算出的相应排名降低参数。

3 结论

基于算法 2 及其变体实现的推荐系统，在几次 A/B 测试中显著提高了推荐点击率。最近的一次 A/B 测试显示，在参与测试的 22 个网站中，有 15 个网站的推荐点击率相较于基线显著提升。不过需要注意的是，算法 2 作为一种模拟技术，依赖于同一网站上至少有一个其他推荐系统运行，以便启动模拟。

4 附录 1. 分布混合与最大熵原理

对于任意两个大小相同的离散分布 P 和 Q，我们有一个由参数构成的凸分布混合族

$$P' = P'(\alpha) = (1-\alpha)P + \alpha Q, \;\; 0 \leq \alpha \leq 1$$

根据最大熵原理（例如参见 [8]），分布 P 和 Q 的无偏凸混合对应于这样一个参数 α 的取值：它使混合熵达到最大。将熵表示为 α 的函数，可写为

$$\mathbb{E}(P') = \mathbb{E}(P, Q, \alpha) = -[\sum_i (1-\alpha)P_i + \alpha Q_i] \log [\sum_i (1-\alpha)P_i + \alpha Q_i] \quad (7)$$

当 Q 是一个狄拉克分布（6）时，分布 P' 与 P 仅在第 i 个位置上有显著差异，而其在其他任意索引 $j \neq i$ 处的质量都只是αp_j。因此，在这种情况下，熵函数 $E(P') \equiv E(\alpha)$ 看起来特别简单

$$-E(\alpha) = (1-\alpha q)\log(1-\alpha q) + \alpha \sum_{j \neq i} p_j \log(p_j) + \alpha \log(\alpha) q$$
$$\text{where } q = 1 - p_i = \sum_{j \neq i} p_j$$

现在很明显，最大化问题 $\max_\alpha E(\alpha)$ 具有一个简单的闭式解。事实上，$-E(\alpha)$ 的导数（在 α 中）为

$$-q\log(1-\alpha q) - q + \sum_{j \neq i} p_j \log(p_j) + (1 + \log(\alpha))q$$

将该导数设为零可得到方程

$$q\log\left(\frac{\alpha}{1-\alpha q}\right) = -C_i \equiv -\sum_{j\neq i} p_j \log(p_j)$$

我们得到如下结果

引理 1 对于 δ 函数分布 *(6)*，当满足以下条件时，凸混合分布 *(7)* 具有最大熵

$$\alpha = \frac{1}{q + e^{C_i/q}}$$
$$p_i' = 1 - \alpha q = \frac{e^{C_i/q}}{q + e^{C_i/q}}$$

其中 $C_i = \sum_{j\neq i} p_j \log(p_j)$ 以及 $q = 1 - p_i = \sum_{j\neq i} p_j$。

当一个点击项不在列表 (1) 中时，我们可以按如下方式对我们的 Distributions$P$ 和 Q 进行填充

$$P = (0,\ p_1,\ \cdots,\ p_n) \tag{8}$$
$$Q = (1,\ \cdots,\ 0,\ \cdots, 0) \tag{9}$$

然后以与上文相同的方式计算混合参数 α，得到

推论 1 对迄今尚未出现的推荐点击进行无偏添加，是分布 *(8)* 和 *(9)* 的凸混合，其中混合参数的取值为

$$\alpha = \frac{1}{1 + e^{H}}$$
$$p_i' = 1 - \alpha = \frac{e^{H}}{1 + e^{H}}$$

其中，$-H$ 是 P的熵

备注 5 引理 *1* 中的计算主要基于以下考虑。一方面，有一些经验证据表明，在算法 *1* 的情形 *3)* 中，推论 *1* 提出的分布更新比将新商品放在列表底部的简单策略效果更好。另一方面，由于无法将迄今未出现过的商品的“正确排名”与已访问商品的排名进行比较，因此无偏的最大熵解似乎是一个不错的选择。

5 附录 2. 一些理论考量

本节主要沿用 [10]。一般而言，推荐点击的到达可视为离散时间依赖的类别分布 $P(\tau)$ 构成的随机过程（序列），其中时间 τ 以每个网页的点击次数计量。合理的假设是，在较短的时间间隔内（例如几百次点击），分布 $P(\tau)$ 不会发生太大变化。为了评估算法 1 跟踪近期事件频率的能力，考虑以下简化情境。假设迭代次数 t 对应于推荐相关性时间窗口。例如，如果只有上周的推荐排名非常重要，那么令 t 表示一周的点击次数。以点击计数器衡量时间，假设该时间段开始时估计的点击分布为 X，并且在时间 t_1 内，未知的输入点击分布 P_1 保持不变。再假设在时间 t_1 时，输入分布变为 P_2，并在剩余时间 $t_2 = t - t_1$ 内保持不变。不难看出（参见 [10]），在该时间段结束时，算法 1 计算得到的分布的期望为

$$\alpha^t X + \alpha^{t_2}(1-\alpha^{t_1})P_1 + (1-\alpha^{t_2})P_2$$

对于足够大的 t，只有最后两项显著，上述公式实质上表明，我们的模拟算法将输入点击分布近似为 P_1 和 P_2 的加权和。为了了解近期点击事件如何影响这一近似，我们来估算上述表达式中 P_2 和 P_1 的系数比。由于假设 $\beta = 1-\alpha$ 很小，我们有（参见 [10]）

$$\frac{1-\alpha^{t_2}}{\alpha^{t_2}(1-\alpha^{t_1})} = \alpha^{-t_2}\frac{1-(1-\beta)^{t_2}}{1-(1-\beta)^{t_1}} \approx \alpha^{-t_2}\frac{t_2}{t_1} \tag{10}$$

备注 6 如我们所见，算法 *1* 在每次迭代中为近期活跃的项目带来约 α^{-1} 的“速度提升”

备注 7 显然（参见 *[10]*），对于 m 输入点击分布 P_1、P_2、$\cdots$、P_m，若它们在时刻 $0 < t_1 < t_2 <$“切换”
$\cdots < t_m < t$ 算法 *1* 生成的分布的期望为

$$E(\alpha) = \alpha^t X + \alpha^{t-t_1}(1-\alpha^{t_1})P_1 + \alpha^{t-t_1-t_2}(1-\alpha^{t_2})P_2 + \cdots + (1-\alpha^{t_m})P_m$$

将两边除以 P_m 处的系数，舍去第一项（因为 $\alpha^t \ll 1-\alpha^{t_m}$），并应用与式 *(10)* 中相同的近似，可得

$$E(\alpha) \approx \alpha^{t-t_1}\frac{t_1}{t_m}P_1 + \alpha^{t-t_1-t_2}\frac{t_2}{t_m}P_2 + \cdots + P_m$$

如果对于所有 $i=1$、2、$\cdots$、m，比值 t_i/t_m 都不太大，那么最后一项将主导此表达式。换言之，在所述条件下，算法 *1* 近似的及时性很大程度上取决于输入分布的切换速率。

6 附录3：标准单纯形上的随机游走仿真

一般而言，对于一般的到达过程，要估计第 2 节所述算法产生的近似结果似乎很困难。因此，我们在此描述一个基于前文讨论的现实仿真模型。考虑所有离散的 n 维分布构成的空间，即标准 n维单纯形 $\Sigma^{n-1} \subset \mathbb{R}^n$。我们要仿真的随机游走从 Σ^{n-1} 的中心 $X_1 = (1/n, 1/n, \cdots, 1/n)$ 开始。此选择对应于这样一个假设：某个推荐商品最初有 n 个推荐，且这些推荐起初无法区分，排名相同，即 $R_1 = (1、1、\cdots、1)$。因此，如第 2 节所述（参见式 (6)），一次传入的点击可表示为对 Σ^{n-1} 某个顶点的选择。

6.1 入站点击 Distributions

仿真模型中的第一个动态部分是一系列 $C_1, \cdots, C_{m-1}$类别分布，由 Σ^n中的一系列点表示。假设模拟点击总数 T 被划分为若干长度随机的时间间隔（天），分别为 $t_1, t_2, \cdots, t_m$，$\sum_i^m t_i = T$。根据备注 7 中讨论的考虑因素，我们假设点击分布在 t_1，t_1+t_2，$\cdots$，$t_1+t_2\cdots+t_{m-1}$时刻发生变化。换言之，对于半开区间 $T_k \equiv (t_1+t_2+\cdots+t_k,\ t_1+t_2+\cdots+t_{k+1}]$内的任意 t，在“时间” t 被点击的商品 i （也就是说，选中Σ^{n-1} 的第 i 个顶点）的概率等于 C_k的第 i个坐标。为简单起见，并再次遵循备注 7 的做法，我们假设区间长度$|T_k|$取样自正态分布 $N(\mu, \sigma)$，且 $\sigma = \frac{1}{5}\mu$。此外，合理的假设是点 C_{i+1} 接近点 C_i，例如，C_{i+1} 与 C_i 仅相差一个较小的高斯噪声。以下 Julia 代码（参见

[11]）展示了满足上述假设的点击分布切换过程。

```
using Random, Distributions, LinearAlgebra
function nextDistribution(C::Distributions.Categorical; eps=.01 )
X = C.p                        # previous categorical distribution
n = size(X)[1]                 # dimension of the simplex
S = rand(n,n); S = S'S         # random symmetric positive definite matrix
I_n = Diagonal( ones(n) )S = I_n + eps * S
                                        # small positive definite perturbation of I_n
  G = MvNormal(X, S)                    # Gaussian with the mean X and covariance S
  V = abs.( rand(G, 1) )[:,1]           # "almost G-distributed" positive vector
  X = V/sum(V)                          # next categorical distribution
  return Categorical(X)
end
```

6.2 算法1的模拟

模拟的第二个组成部分是对算法 1 的直接实现，并且只应用于一个锚点。
这里假设商品点击来自以下分类分布序列 C_k， $k = 1$，2，$\cdots$令
$t+1 \in T_k$，令 i 为从时刻 $t+1$ 的分布 C_k 中采样得到的随机索引，并令
α 为算法 1 描述中的固定平滑参数（参见第 2.2 节）。那么

$$X_{t+1,j} = \alpha X_{t,j} \text{ for } j = 1, 2, \cdots n,\ j \neq i$$

and

$$X_{t+1,i} = \alpha X_{t,i} + 1 - \alpha$$
$$R_{t+1,i} = R_{t+1,i} + 1$$

对于所有变量，第一个索引均为模拟迭代编号（点击“时间”），第二个索引为向量坐标。
按照上述方式，模拟生成两个类别分布序列
序列

X_t 用于第 2 节中描述的算法

ˆ 通过对点击计数器取平均定义的点击频率分布 R_t

$$\hat{R}_t = \left(\frac{1}{\sum_{i=1}^{n} R_{t,i}} \right) R_t$$

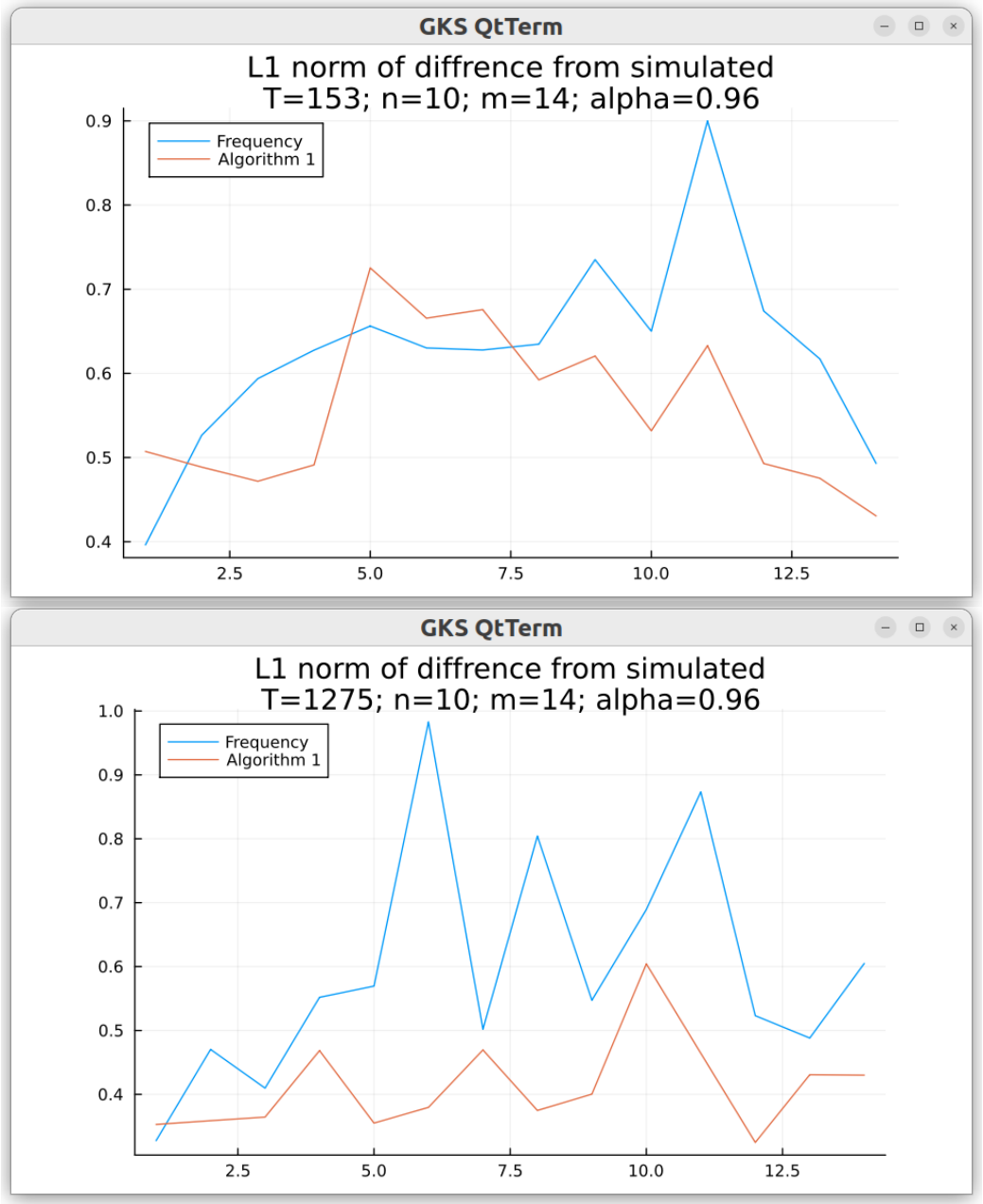

图1：上图显示了14个每日随机时间间隔的l_1-范数差异$\Delta_k(X)$和$\Delta_k(R)$，这些时间间隔的长度分别为14，11，12，14，11，12，9，12，7，9，11，10，9，12。算法 1 的平均l_1范数差异约为0。55，而频率计数器的约为0。62。

底部图表显示了 14 个每日随机时间间隔的长度分别为 114、89、104、94、114、115、88、99、76、37、78、72、75、120 时的 l_1-范数差异$\Delta_k(X)$ 和$\Delta_k(R)$。算法 1 的平均 l_1 范数 11 差异约为 0.40，而频率计数器的平均差异约为 0.59。

可通过比较分类分布序列 X_t， R_t 与“实际”输入的点击分布序列 C_k，k =1、2、$\cdots$，来评估算法 1。例如，可以比较平均 l_1-范数差异

$$\Delta_k(X) = \frac{1}{|T_k|} \sum_{t \in T_k} \parallel X_t - C_k \parallel_1$$

$$\Delta_k(R) = \frac{1}{|T_k|} \sum_{t \in T_k} \parallel \hat{R}_t - C_k \parallel_1$$

图 1 上方绘出了典型模拟运行中 $\Delta_k(X)$ $\Delta_k(R)$ 的序列（参见 [12]）。

参考文献

[1] https://en.wikipedia.org/wiki/Recommender_system

[2] https://en.wikipedia.org/wiki/Item-item_collaborative_filtering

[3] Brent Smith、Greg Linden，《Amazon.com 推荐系统的二十年》，IEEEINTERNET COMPUTING，由 IEEE计算机学会 出版，1089-7801/17/，2017[4] ATHANASIOS N. NIKOLAKOPOULOS、GEORGE KARYPIS，《基于近似非耦合随机游走的商品协同过滤》，arXiv:1909.03579v2，2020[5] Chen-Yu Hsu、Piotr Indyk、Dina Katabi 和 Ali Vakilian，《基于学习的频率估计算法》，ICLR 2019[6] Anshumali Shrivastava Arnd Christian K¨onig、Mikhail Bilenko，《用于汇总数据流的时间自适应草图（Ada-Sketches）》，SIG-MOD’16，2016年6月26日至7月1日，美国加利福尼亚州旧金山

[7] https://www.evanmiller.org/rank-hotness-with-newtons-law-of-cooling.html

[8] Ariel Caticha，《熵推断》，arXiv:1011.0723，2010

[9] Tong Yang、Haowei Zhang、Jinyang Li、Junzhi Gong、Steve Uhlig、Shigang Chen 和 Xiaoming Li，《HeavyKeeper：一种用于查找 Top-k 大象流的精确算法》，IEEE/ACM 网络学报，2019

[10]Alexander Kushkuley, 《重击者与伯努利卷积》,
arXiv:1905.08930v2（http://arxiv.org/abs/1905.08930v2）, 2019
[11] https://juliastats.org/Distributions.jl/stable/
[12] https://docs.juliaplots.org/stable/