基于跳过程与统计学习的普适排行榜类产品度量模型

程晨[†]　　陈子恒[‡]　　郝天泽[§]

摘要

关键字：

[†]School of Mathematical Sciences, Peking University, Beijing 100871, China. Email address: moriartycc@pku.edu.cn, ID: 1500010714
[‡]School of Mathematical Sciences, Peking University, Beijing 100871, China.
[§]School of Mathematical Sciences, Peking University, Beijing 100871, China.

目录

1 引言

2 假设

在开始阐述我们的模型之前，我们先描述这些模型所依赖的一些基本假设，和所建立的概率模型的空间。

排行榜 我们所讨论的排行榜为产品状态集合 $\mathcal{P} = \{1, 2, \cdots, N\}$ 的一个排序，我们将所有排行榜产品构成的集合记为 Ω_P，则 $\forall \omega \in \Omega_P$，我们就有一个排行榜

$$\omega = (p_1, p_2, \cdots, p_N)$$

产品固有性质 为了能够度量排行榜类产品的“优劣”，我们假设产品本身有一些固有性质。如果直接定义一个“好坏”的程度，既难以描述和精确定义，又不适合在现实中使用。因此，我们考虑一些能够准确观察到和用统计方法推断的性质，且所有数据都能够用**用户停留时间**τ 和**用户决策**a 来确定。对于任意一个产品 $i \in \mathcal{P}$，我们假设

- **用户停留时间 τ 的分布**。用户停留时间服从一个仅与 i 有关的分布 f_i，或

  $$\tau \sim f_i$$

  我们说明这样的假设是合理的。尽管对于每一个不同的用户 u，其所对应的 f_i^u 可能不同（我们认为一个确定的用户使用同一份产品所用的时间也是一个随机变量）。但考虑所有用户集合 $\mathcal{U}$，以及每一个用户 $u \in \mathcal{U}$ 被抽样的概率 p_u，我们就能得到

  $$f_i = \sum_{u \in \mathcal{U}} f_i^u p_u$$

- **用户的决策 a 的可能性**。对于排行榜类产品，用户本身可能有多种感官或心理上的体验，但这些体验并不能被开发者观察和收集到，因此并不是我们所关心的。我们关心的是用户所做的决策，而这些决策将被用来衡量排行榜类产品的好坏。具体来说，根据我们研究的不同排行榜类模型，我们将用户的决策分为
  - **接受（Accepted）**：对于选择型的排行榜，接受表示用户阅读该产品后选择了该产品。
  - **放弃（Rejected）**：对于选择型的排行榜，放弃表示用户阅读该产品后放弃了该产品。
  - **跳过（Skipped）**：对于选择、投票型的排行榜，跳过表示用户没有阅读该产品就直接跳过。
  - **结束（Terminated）**：对于投票型的排行榜，结束表示用户结束了他的阅读过程。
  - **投票（Voted）**：对于投票型的排行榜，投票表示用户阅读该产品后进行的决策，具体来说可以有三种
    * **支持（Up vote)**
    * **反对（Down vote)**
    * **不表态（No vote）**

用户 对于用户来说，我们关心他们的停留时间 τ 和决策 a。相应的，我们做出这些假设

- τ 为指数分布。即 $f_i = \mathrm{Exp}(\lambda_i)$。这样的假设是常用且合理的。
- 用户群体对于接受、放弃、跳过、结束、支持、反对和不表态分别有基于产品的固有概率 $a_i, r_i, s_i, t_i, u_i, d_i, n_i$。
- 产品独立性。每个用户使用当前产品时不会受到之前产品的影响，且不考虑个人喜好的因素，即其所做决策的概率完全有产品本身决定。

3 无跳过的顺序选择模型——Steam 探索队列

3.1 模型概述 & 符号约定

我们首先考虑一种最为简单，受约束较多的排行榜类推荐模型——**无跳过的顺序选择模型。在这个模型中，每一个用户必须依照排行榜的顺序依次阅读有关每一个产品的信息，不能跳过，且必须作出一个选择（我们假设如果到了排行榜队列底端就必须选择最后一个产品）**。这样的模型是基于著名的游戏品台 Steam 中的探索队列 (1) 功能简化得到的。

图 1: Steam 探索队列

Steam 探索队列是 Steam 游戏平台为用户推荐产品的一种方式。该队列通过分析用户的喜好设定、浏览记录以及购买记录等信息为用户推荐一个产品队列（可视为一种产品排行榜）。当用户浏览此队列时，需要依次浏览队列中每一个产品的相关信息并进行是否感兴趣的选择（即接受或放弃的用户决策）。当我们假定现在客户有购买一个产品的需求，其浏览 Steam 探索队列的方式及可抽象为这里的无跳过的顺序选择模型。而往往这样的队列足够长，且是通过分析用户喜好得来，所以不妨假设用户可以从这一队列中挑选出一个其接受的产品，即前面谈到的如果到了排行榜队列底端就必须选择最后一个产品的假设。

具体来说，我们给出无跳过的顺序选择模型的定义。

定义 3.1. （**无跳过的顺序选择模型**）该模型是一个建立在状态空间为 $\mathcal{P} \cup \{A\}$ 上的跳过程。其中 A 表示接受状态。其中状态 i 处的指数闹钟参数为 λ_i，且其嵌入链的转移概率为

$$\begin{aligned} a_{p_N} &= 1 \\ \boldsymbol{P}_{p_i,A} &= a_{p_i} \\ \boldsymbol{P}_{p_i,p_{i+1}} &= r_{p_i} = 1 - a_{p_i} \end{aligned} \tag{1}$$

相应地，该跳过程的速率矩阵可以描述为

$$\begin{aligned} \boldsymbol{Q}_{p_i,A} &= \lambda_{p_i} a_{p_i} \\ \boldsymbol{Q}_{p_i,p_{i+1}} &= \lambda_{p_i} r_{p_i} = \lambda_{p_i}(1 - a_{p_i}) \end{aligned} \tag{2}$$

在图（2）中展示了具体的模型。该模型可表述为 $\mathfrak{A}(\boldsymbol{\lambda}, \boldsymbol{a}, \omega)$。

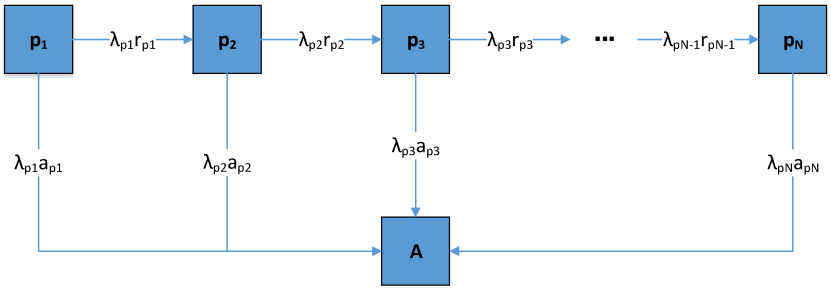

图 2: 无跳过的顺序选择模型结构

直观上看，此模型实际上就是我们上面文字描述的严格定义。A 表示接受状态，即用户决定接受某一产品，a_{p_i} 表示决定接受产品 p_i 的概率，$r_{p_i}=1-a_{p_i}$ 表示拒绝产品 p_i 从而继续浏览 p_{i+1} 的概率，于是 $\lambda_{p_i}a_{p_i}$ 即表示决定接受产品 p_i 的速率，$\lambda_{p_i}r_{p_i}=\lambda_{p_i}(1-a_{p_i})$ 即表示拒绝产品 p_i 从而继续浏览 p_{i+1} 的速率。从而构建出了此跳过程的转移速率矩阵。

3.2 度量模型

在这样的模型中，我们有理由设置用户找到其接受的产品的时间 τ_E 为衡量产品优劣的标准。时间越短，那么产品就越有效。而 τ_E 在对应的随机过程中即为到达状态 E 的首达时，其分布函数可以表示为

$$F_{\tau_E}(t)=e_{p_1}^T\exp(\boldsymbol{Q}t)e_E,\quad e_i\in\mathbb{R}^{N\times 1},i\in\mathcal{P}$$

其方差和期望分别为（约定 $a_{p_0}=1$）

$$\begin{aligned}\mathbb{E}\tau_E&=\sum_{k=1}^{N}\frac{\prod_{j=0}^{k-1}a_{p_j}}{\lambda_{p_i}}\\ \mathrm{Var}\tau_E&=\sum_{k=1}^{N}\left(\frac{\prod_{j=0}^{k-1}a_{p_j}}{\lambda_{p_i}}\right)^2\end{aligned}\tag{3}$$

证明请参见附录。为了综合考虑期望和方差，以及展现出用户对于长队列的不满程度，我们将衡量函数定义为

$$\mathrm{Score}_{\mathfrak{A}(\boldsymbol{\lambda},\boldsymbol{a},\omega)}(x,\alpha)=\sum_{k=1}^{N}\left(\frac{\prod_{j=0}^{k-1}a_{p_j}}{\lambda_{p_i}}x^k\right)^{\alpha},\quad x\in[1,\infty),\alpha\in[1,2]\tag{4}$$

其中的 $x\in[1,\infty)$ 可以理解为某种“不满值”，即随着浏览队列的加长，用户的耐心渐渐减少，每浏览队列中越靠后的一个商品用户增加的不满值越大，因而其属于 $[1,\infty)$。$\alpha\in[1,2]$ 则是综合考虑用户所需时间的期望与方差。对于一个优秀的排行方式，其应该尽量使得用户平均等待时间较少，所积累不满较少的同时保证用户“一致地”满意。于是我们得到了这里的衡量函数 $\mathrm{Score}_{\mathfrak{A}(\boldsymbol{\lambda},\boldsymbol{a},\omega)}(x,\alpha)$。注意到 $\alpha=1,x=1$ 时衡量函数就等于等待时间的期望，而 $\alpha=2,x=1$ 时衡量函数就等于等待时间的方差，而这里“不满值” x 也可以理解成一种“心理时间”，即在较长时间的等待后，对用户而言等待时间似乎被加长，这说明了衡量函数的合理性。

当然，由于参数 $\boldsymbol{\lambda},\boldsymbol{a}$ 是隐含的，我们需要通过统计推断的方法得到它们，即通过数据得到推断的参数 $\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{a}}$，然后得到我们的衡量函数

$$\mathrm{Score}_{\mathfrak{A}(\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{a}},\omega)}(x,\alpha)$$

且 $\text{Score}_{\mathfrak{A}(\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{a}},\omega)}(x,\alpha)$ 越小，则排行榜产品越好。

3.3 统计推断 & 优化算法

对于参数 $\boldsymbol{\lambda},\boldsymbol{a}$ 的推断我们使用极大似然估计，即假设第 m 个用户接受了第 p_{k_m} 个产品，前 k_m 个产品的浏览时间分别为 $t_{m,i}, 1\le i\le m$，总共有 M 个用户数据被收集，那么似然函数为

$$\mathcal{L}(\boldsymbol{\lambda},\boldsymbol{a},\omega)=\sum_{m=1}^{M}\ln p_k(\boldsymbol{\lambda},\boldsymbol{a})=\sum_{m=1}^{M}\ln\left(\left(\prod_{i=1}^{k_m-1}(1-a_{p_i})\right)a_{p_{k_m}}\left(\prod_{i=1}^{k_m}\lambda_{p_i}\right)\exp\left(-\sum_{i=1}^{k_m}\lambda_i t_{m,i}\right)\right) \tag{5}$$

容易看出此对数似然函数为凸函数，我们使用成熟的凸优化工具（CVX toolbox）可以得到极大似然估计

$$\left(\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{a}}\right)=\underset{(\boldsymbol{\lambda},\boldsymbol{a})}{\operatorname{argmin}}\,\mathcal{L}(\boldsymbol{\lambda},\boldsymbol{a},\omega)$$

利用推断得到的参数就可以计算衡量函数了。

3.4 实验结果

4 带跳过的顺序选择模型——美团、饿了吗外卖

4.1 模型修正 & 符号约定

在这一节中，我们考虑对第一个模型的修正——**带跳过的顺序选择模型。在这个模型中，每一个用户不必详细查看排行榜中的每一个产品，可以选择跳过其中的某些条目，只选择性的详细阅读某些条目。同无跳过的顺序选择模型一样，用户也必须且仅需做出一个选择**。这样的模型是基于美团、饿了吗外卖（3）以及滴滴打车等一次性服务项目的功能简化得到的。

图 3: 饿了吗外卖

例如上图，在使用饿了吗外卖时，用户会得到一个仅包含产品名的排行榜（我们认为得分也是排行榜的一部分，因此不予考虑）。用户会根据自己的经验或兴趣，顺着排行榜选择是否仔细阅读相关产品的信息；在发现自己想要的产品之后就做出选择。具体来说，我们给出带跳过的顺序模型修正后的定义。

定义 4.1.（**带跳过的顺序选择模型**）该模型是一个建立在状态空间为 $\mathcal{P} \cup \tilde{\mathcal{P}} \cup \{A\}$ 上的跳过程。其中 A 表示接受状态，$\tilde{\mathcal{P}}$ 表示判断是否要跳过状态 $i \in \mathcal{P}$ 的状态。其中状态 i 处的指数闹钟参数为 λ_i，在 $\tilde{i}$ 处的指数闹钟参数为 ∞，即瞬间跳跃；跳过状态 i 的概率为 s_i，且其嵌入链的转移概率为

$$
\begin{aligned}
a_{p_N} &= 1 \\
s_{p_N} &= 0 \\
\boldsymbol{P}_{\tilde{p}_i, \tilde{p}_{i+1}} &= s_{p_i} \\
\boldsymbol{P}_{\tilde{p}_i, p_i} &= 1 - s_{p_i} \\
\boldsymbol{P}_{p_i, A} &= a_{p_i} \\
\boldsymbol{P}_{p_i, \tilde{p}_{i+1}} &= r_{p_i} = 1 - a_{p_i}
\end{aligned}
\tag{6}
$$

在图（4）中展示了具体的模型。该模型可表述为 $\mathfrak{B}(\boldsymbol{\lambda}, \boldsymbol{a}, \boldsymbol{s}, \omega)$。

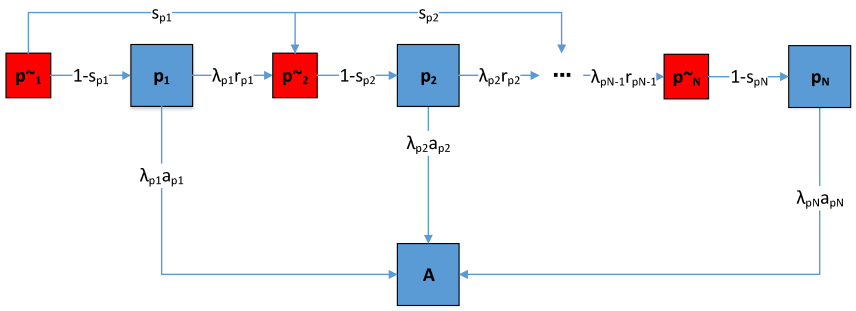

图 4: 带跳过的顺序选择模型结构

4.2 度量模型 & 统计推断 & 优化算法

与不带跳过的选择模型相同，我们依然选择用户找到其接受产品所花的时间 τ_E 来衡量排行榜算法的优劣，并认为时间越短就越有效。同样的，我们可以得到

$$
\begin{aligned}
\mathbb{E}\tau_E &= \sum_{k=1}^{N} \frac{(1 - s_{p_j}) \prod_{j=0}^{k-1} a_{p_j}}{\lambda_{p_i}} \\
\mathrm{Var}\tau_E &= \sum_{k=1}^{N} (1 - s_{p_j}^2) \left(\frac{\prod_{j=0}^{k-1} a_{p_j}}{\lambda_{p_i}} \right)^2
\end{aligned}
\tag{7}
$$

证明请参见附录。为了综合考虑期望和方差，我们用参数 α 进行连续过渡；同样地，我们也使用指标 x 表示用户对于长队列的不满。综上，我们将衡量函数定义为

$$
\mathrm{Score}_{\mathfrak{B}(\boldsymbol{\lambda}, \boldsymbol{a}, \boldsymbol{s}, \omega)} = \sum_{k=1}^{N} (1 - s_{p_j})(1 - s_{p_j} + \alpha s_{p_j}) \left(\frac{\prod_{j=0}^{k-1} a_{p_j}}{\lambda_{p_i}} x^k \right)^{\alpha}, \quad x \in [1, \infty], \alpha \in [1, 2] \tag{8}
$$

相应的，参数 $\boldsymbol{\lambda}, \boldsymbol{a}, \boldsymbol{s}$ 均是隐含的，我们需要从数据中学习出这些参数的推断值 $\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{s}}$，然后再代入上式进行计算。依然使用似然函数的办法和优化策略解决这个问题。

4.3 实验结果

5 带跳过的投票模型——豆瓣、知乎推荐

5.1 模型修正 & 符号约定

定义 5.1. （**带跳过的投票模型**）该模型是一个建立在状态空间为 $\mathcal{P} \cup \tilde{\mathcal{P}} \cup \mathcal{U} \cup \mathcal{D} \cup \mathcal{N} \cup \{T\}$ 上的跳过程。其中 T 表示结束状态，$\tilde{\mathcal{P}}$ 表示判断是否要跳过状态 $i \in \mathcal{P}$ 的状态，$\mathcal{U}$ 表示第 i 个状态的结果为**赞成**的状态，$\mathcal{D}$ 表示第 i 个状态的结果为**反对**的状态，$\mathcal{N}$ 表示第 i 个状态的结果为**不予评价**的状态。其中状态 i 处的指数闹钟参数为 λ_i，在 $\tilde{i}$ 以及 $\mathcal{U}$，$\mathcal{D}$，$\mathcal{N}$ 所对应状态处的指数闹钟参数为 ∞，即瞬间跳跃；跳过状态 i 的概率为 s_i，若不跳过则赞成的概率为 u_i，反对的概率为 d_i，不予评价的概率为 n_i；且其嵌入链的转移概率为

$$
\begin{aligned}
a_{p_N} &= 1 \\
s_{p_N} &= 0 \\
\boldsymbol{P}_{\tilde{p}_i, \tilde{p}_{i+1}} &= s_{p_i} \\
\boldsymbol{P}_{\tilde{p}_i, p_i} &= 1 - s_{p_i} \\
\boldsymbol{P}_{p_i, T} &= t_{p_i} \\
\boldsymbol{P}_{p_i, u_i} &= u_i \\
\boldsymbol{P}_{p_i, d_i} &= d_i \\
\boldsymbol{P}_{p_i, n_i} &= n_i \\
u_i + d_i + n_i &= 1 \\
\boldsymbol{P}_{u_i, \tilde{p}_{i+1}} &= \boldsymbol{P}_{d_i, \tilde{p}_{i+1}} = \boldsymbol{P}_{n_i, \tilde{p}_{i+1}} = 1 \\
\boldsymbol{P}_{u_N, T} &= \boldsymbol{P}_{d_N, T} = \boldsymbol{P}_{n_N, T} = 1
\end{aligned}
\tag{9}
$$

这里和后面都会用 u_i, d_i, n_i 同时表示概率和状态的含义，在表达式中不会产生歧义。在图（5）中展示了具体的模型。该模型可表述为 $\mathfrak{C}(\boldsymbol{\lambda}, \boldsymbol{t}, \boldsymbol{s}, \boldsymbol{u}, \boldsymbol{d}, \boldsymbol{n}, \omega)$。

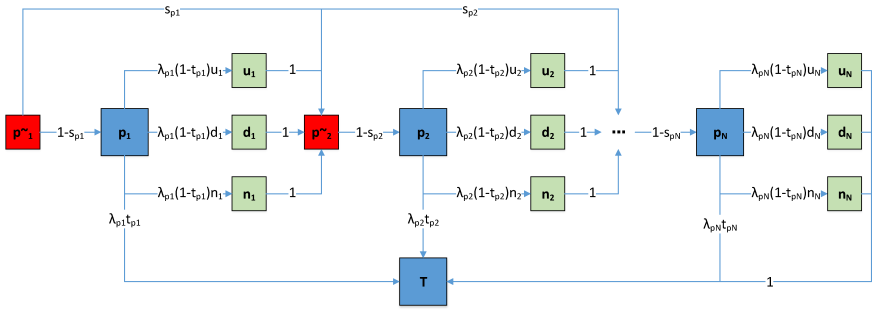

图 5: 带跳过的投票模型结构

5.2 度量模型 & 统计推断 & 优化算法

在这里，我们用一种新定义的**有效体验时间**τ_T 来衡量排行榜的好坏。具体来说 τ_T 是**所有做出了赞成或反对的阅读时间的总和**。对于豆瓣、知乎这种阅读体验累的产品来说，用户能够对阅读的产品更长时间的保持兴趣则产品设计的越好，而在阅读完之后做出了赞成或反对的选择就表示用户对于这个条目很感兴趣，这段时间可以被看作是有效的。具体来说，我们可以计算出 τ_T 的期望和方差为

$$\begin{aligned}\varphi_{p_j} &= s_{p_j} + n_{p_j} - s_{p_j} n_{p_j} \\ \mathbb{E}\tau_T &= \sum_{k=1}^{N} \frac{(1-\varphi_{p_j}) \prod_{j=0}^{k-1}(1-t_{p_j})}{\lambda_{p_i}} \\ \mathrm{Var}\tau_T &= \sum_{k=1}^{N}(1-\varphi_{p_j}^2) \left(\frac{\prod_{j=0}^{k-1}(1-t_{p_j})}{\lambda_{p_i}} \right)^2\end{aligned} \tag{10}$$

证明请参见附录。为了综合考虑期望和方差，我们用参数 α 进行连续过渡；同样地，我们也使用指标 x 表示用户对于后面才出现的感兴趣的话题不满。综上，我们将衡量函数定义为

$$\mathrm{Score}_{\mathfrak{C}(\boldsymbol{\lambda},\boldsymbol{t},\boldsymbol{s},\boldsymbol{u},\boldsymbol{d},\boldsymbol{n},\omega)} = \sum_{k=1}^{N}(1-\varphi_{p_j})(1-\varphi_{p_j}+\alpha\varphi_{p_j}) \left(\frac{\prod_{j=0}^{k-1}(1-t_{p_j})}{\lambda_{p_i}} x^k \right)^{\alpha}, \quad x \in (0,1], \alpha \in [1,2] \tag{11}$$

相应的，参数 $\boldsymbol{\lambda}, \boldsymbol{t}, \boldsymbol{s}, \boldsymbol{u}, \boldsymbol{d}, \boldsymbol{n}$ 均是隐含的，我们需要从数据中学习出这些参数的推断值 $\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{s}}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{d}}, \hat{\boldsymbol{n}}$，然后再代入上式进行计算。依然使用似然函数的办法和优化策略解决这个问题。

注意到，与上面的情况不同，我们的这个衡量函数实际上是在测量用户的**有效体验时间**，因此 $\mathrm{Score}_{\mathfrak{C}(\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{t}},\hat{\boldsymbol{s}},\hat{\boldsymbol{u}},\hat{\boldsymbol{d}},\hat{\boldsymbol{n}},\omega)}$ 的越大我们认为排行榜越好。

5.3 实验结果

6 实时计算

由于排行榜具有很强的时效性和话题性，除非是一些长久建立的口碑排行榜（如旅游景点排行榜），我们在推断产品固有性质中的 $a_i, r_i, s_i, t_i, u_i, d_i, n_i$ 时都不能把他们当做常数看待，而应该把它们都看做关于时间 t 的函数。因此，在推断某个时刻 t 的产品固有性质的值时，我们可以将所有 $[t-\Delta t, t)$ 内的用户作为有效数据点来进行推测。Δt 的选择应该适中，使得用户样本量足够大且间隔不会太长使得推断的结果不够准。

7 实际产品评价

7.1 两种著名的排行榜产品

7.2 不同情形下的实验比较

8 由此启发得到的基于优化的排行榜算法

9 总结