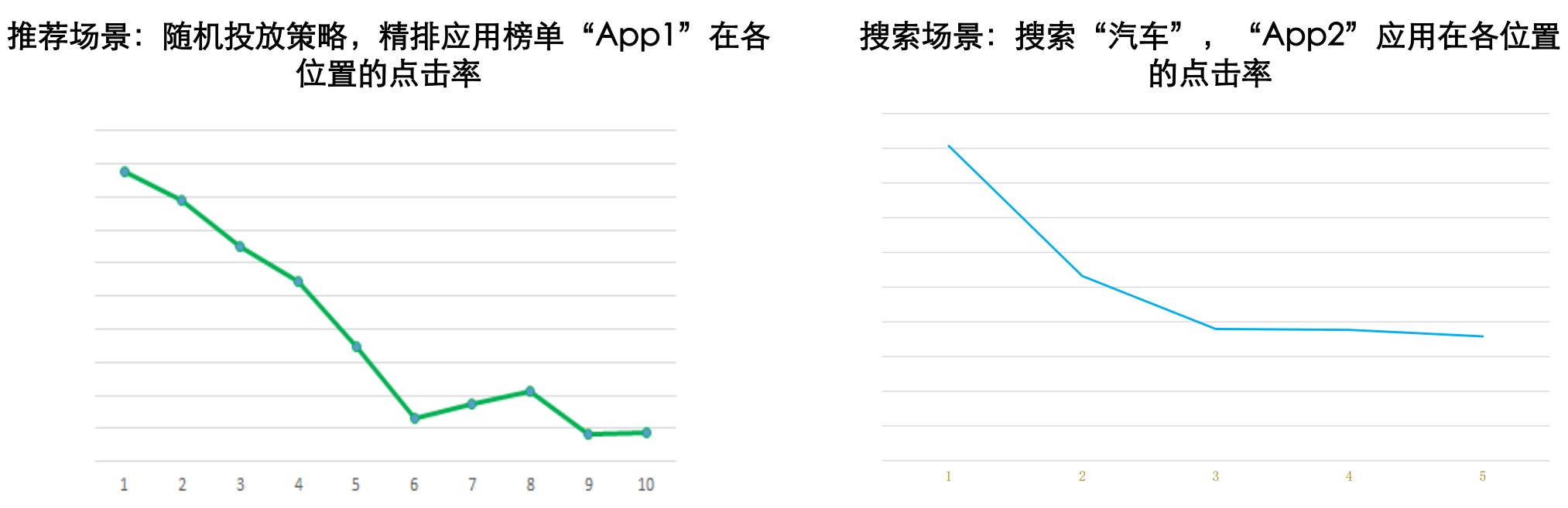
**问题的引出：某客户想调研因果推断causal inference在推荐系统中的落地，目的是为了对推荐系统中存在的各种bias（偏差/偏置）进行缓解或者消除即所谓的debias。**

**上下文知识：**

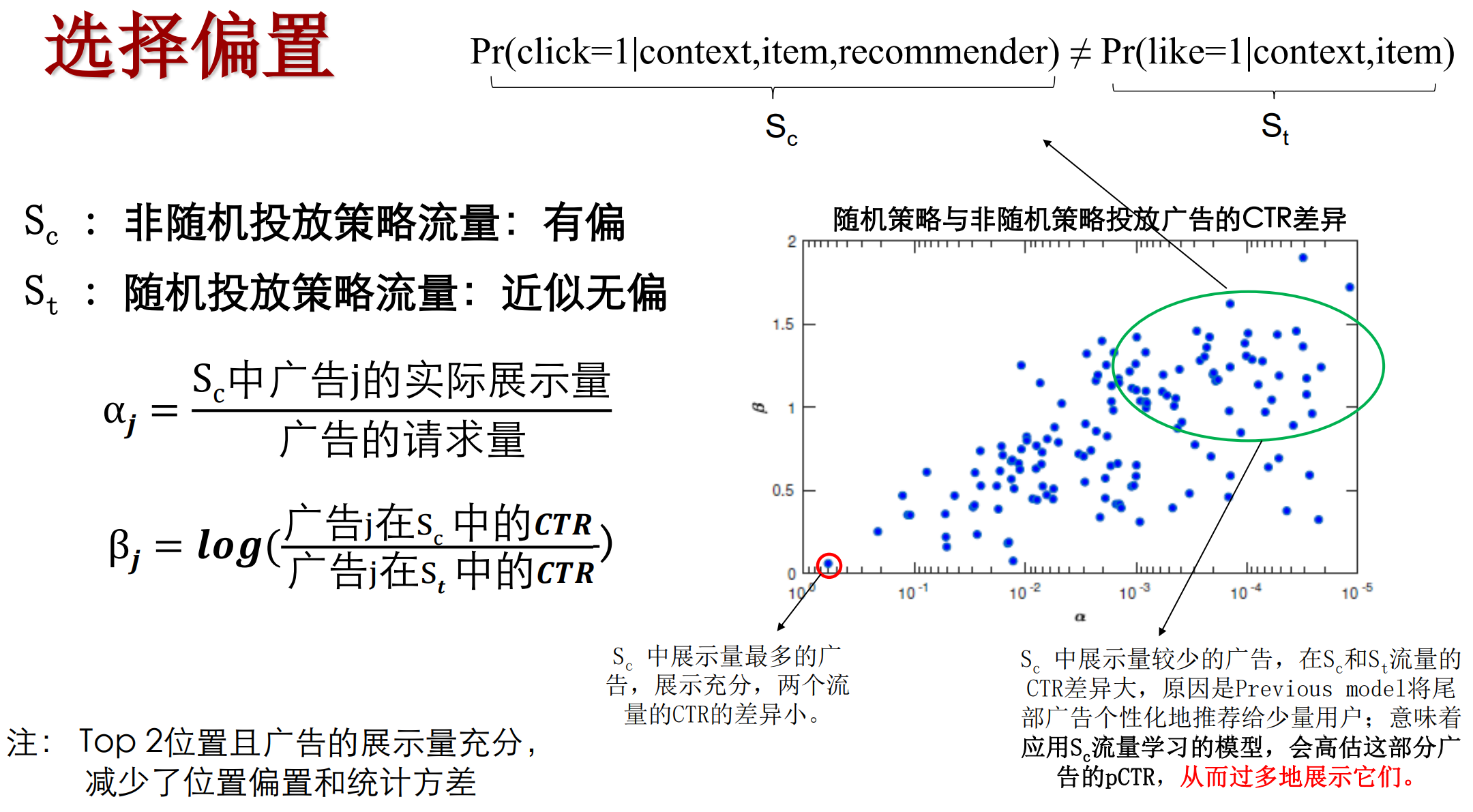
推荐系统中存在非常多的bias问题也就是所谓的偏置/偏差问题，比如位置偏置，选择偏置，流行度偏置等等。

**位置偏置**：描述用户倾向性地选择处于更好位置（比如更靠前的位置）的物品进行交互，这种倾向性与物品是否满足用户的实际需求无关（参考下图）。



**位置偏置导致好位置的item的CTR高，使用该数据建模型会加剧马太效应**（也就是强者越强，弱者越弱）。

**选择偏置**：简单说就是，推荐给用户的item是根据系统来选择的，而不是所有候选item集合，真实的候选item集合大大超过了推荐给用户的曝光item列表，导致训练数据带有选择性偏差。如下图所示。



**流行度偏置**：简单说，就是终端用户比较喜欢点击最近比较流行的item，一种常见的从众心理。

**Debias的方案调研：**

Debias现在在搜推广中应用比较多。主要借鉴causal Inference中的IPW(Inverse Probability Weighting)和Counterfactualinference（反事实推断）思想进行Debias。

1. 因果推断在推荐系统中的落地可以参考快手的一片博客(**推荐阅读**) <https://www.modb.pro/db/331248>

里面具体有三个场景：流行度的debias，视频完播率的debias(**这个是通过IPW来做样本权重的**)，因果表征解耦分离出用户感兴趣的embedding向量和从众的embedding向量。

里面提到了对视频完播率做debias的做法就是利用IPW做样本权重，权重公式没有很明确，**我猜测的如下**：

Smaple\_weight = finish\_playing\_ratio /item\_history\_CTR + (1 - finish\_playing\_ratio) / (1 - item\_history\_CTR)

另外，这篇文章中还提到如下：

Q：**Debias会不会在一定程度上对某些业务指标产生负作用**？

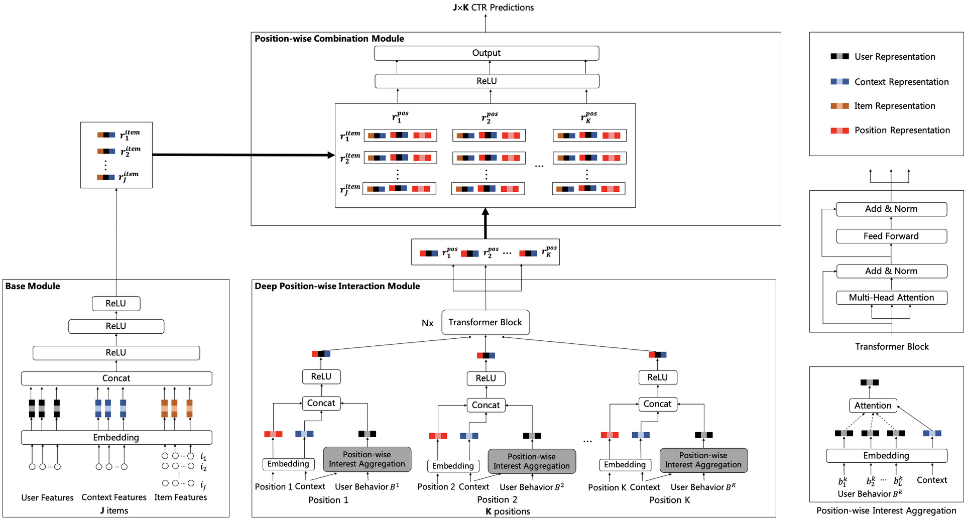
A：Bias不一定全是对模型有害的，例如流行度其实对推荐是有收益的。假如你纠偏过度，那么实际上你会降低系统的整体收益的；与之相反，如果你纠偏不够的话有无法解决bias问题。**这就是为什么因果推断很难在推荐中落地，因为这需要根据具体业务不断地对纠偏程度进行调整。**

B. 美团在2020年的KDD CUP推荐系统debias赛题中获得冠军（https://tech.meituan.com/2020/08/20/kdd-cup-debiasing-practice.html ），他们这个方案针对的是搜推广中的选择偏差和流行度偏差问题。

方案最终使用了i2i的思路通过基于图的多跳游走算法来做正负样本的生成。（看起来，我觉得这里并没有把正样本扩充，扩充的是负样本）；另外，这个博客还提到了在他们自己的搜索广告系统中落地debias的时候，是通过某种EE算法来扩充样本的（这个样本扩充肯定是包括正负样本的）

C. 美团在2021年发表了一片paper（https://tech.meituan.com/2021/06/10/deep-position-wise-interaction-network-for-ctr-prediction.html ），利用所谓的深度位置交叉网络DPIN来对整个候选广告(**就是对召回的广告集做排序任务**)的**所有曝光位置**做建模（所以这个模型非常的复杂！！）。

DPIN在评估模型的时候除了使用AUC，还是用了基于位置的AUC即所谓的PAUC，最终的排序依据是每个位置的ctr概率乘以出价的累积和。这个DPIN模型是为了对位置偏差做消除/缓解而产生的。



**它在建模的时候，会对每个用户的所有曝光位置的行为序列来建模，会导致数据更加稀疏；而且需要在线上开一个随机流量探索的桶作为位置无偏流量积累一些样本**（比如策略是随机挑选广告+随机给选出来的广告做无放回的分配位置，这样就需要宝贵的一些流量，因此这个做法可能在实际中比较难落地！）。除了上面这个因素，在实际落地的时候还有下面的因素需要考虑：

1）**召回的item集/广告集的长度是多少？**

长度太大的话，线上serving的时候无法满足。

注意这里美团的DPIN模型提出的时候是针对的他的站内广告的排序任务，一般来说召回的广告数量不多；但是对于常见的推荐系统的召回结果集一般都很多，很轻松就超过1000，把召回的1000+的itemid放入模型的base模块做前向计算非常耗时。

2）**推荐列表的位置长度是多少？**

由于DPIN模型本质上是对J\*K的矩阵打分（J是候选item的个数，K是推荐列表的长度），因此如果推荐列表比较长，比如100+，那么同样线上serving的latency是个问题。

一个workaround就是：一般来说不会对所有的推荐列表位置都考虑，会根据百分位统计来把比较靠后的所有位置用一个other位置来表示。

3）**每个终端用户的推荐列表的每个位置的数据是否足够dense？**

即使用了一些位置无偏流量来积累一些样本，并且使用了上面提到的合并推荐列表位置的方法，具体到单个用户的在每个合并后的推荐位置的数据仍然是稀疏的。

因此，**在把DPIN模型用到你的具体业务场景的时候要仔细思考，他是否适合你**！！

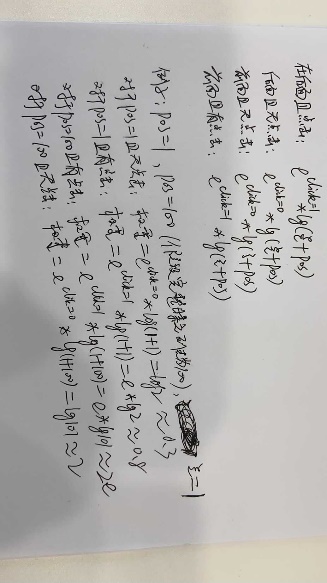
1. **关于位置偏置，常见的一个简单的缓解的方式就是给每个曝光样本和点击样本根据位置设置样本权重(其实算是一种IPW方法)**：

**原则上需要满足**：对于位置越靠后的item并且还点击了的话其样本权重越高，位置靠前并且还没有点击的话样本权重越低；位置很靠后没有被点击的话权重不要大，位置很靠前并且被点击的话权重也不要大。

**综合考虑是否点击以及曝光item的位置**，**我想到的一个我认为还算合理的计算样本权重的公式如下**：

sample\_weight = exp(isclick) \* log(a + position)

上式中的position是曝光的item的位置，a是用来平滑的（为了让position为1的时候，样本权重不为0），log是以10为底的对数函数。另外，**这个公式也考虑了即使以后的推荐列表的总长度发生变化（这个很可能会根据运营策略来修改的），对于老的样本的sample\_weight不需要重新修改**。可以参考下面的手算图：



从上面的计算结果可以看出，我认为还算是合理。另外，可以在样本权重中加入考虑了流行度偏置后的计算（就是对越流行的item加入惩罚，权重变低），比如

sample\_weight = exp(isclick) \* log(a + position)\* exp(-popu)

popu是用百分比表示的。

**可能的结论：**

目前看，使用因果推断在推荐系统中落地的情况还是比较少（大厂在做一些尝试）。即使想用这个方法的话，可以考虑从IPW方法来做尝试。