【MAB问题】如何将Bandit算法与协同过滤结合使用

2018-04-13 刑无刀

推荐系统三十六式 进入课程 >

讲述：黄洲君
时长 11:44 大小 5.38M

推荐系统中最经典的算法是什么？对，是协同过滤，你已经学会抢答了。

是的，协同过滤是推荐系统发展史上浓墨重彩的一笔，其背后的思想简单深刻，在万物互联的今天，协同过滤的威力更加强大。与其说协同过滤是一门技术，不如说是一种方法论，不是机器在为你推荐，而是“集体智慧”在为你推荐。

协同过滤生动地诠释了什么是“物以类聚，人以群分”，你的圈子决定了你能见到的物品，这一点在前面的专栏中已经详细讲过了。但是这背后隐藏了一个重要的问题：是不是会存在信息茧房的问题？

信息茧房

其实作为一名对推荐系统略懂一二的普通海淀群众，我个人就会时常担心，是不是还能看到新的东西，是不是有惊喜。时不时乱点一通，是不是叉掉所有的推荐，让物品的推荐系统崩溃一下，这一切就是为了避免进入信息茧房，在眼前的圈子里苟且。

那么作为推荐系统的开发者，是不是应该做点什么呢？是的，在技术上，Bandit 算法就是一个权衡探索和利用的好方法。如果把它结合传统的协同过滤来做推荐，那么在一定程度上就可以延缓信息茧房的到来，偶遇诗和远方。

我已经和你聊了两篇关于 Bandit 算法的内容，我介绍过普通的 Bandit 算法，也介绍过加入特征信息的 LinUCB 算法，今天，我要介绍的是一个新方法，如何结合协同过滤的群体智慧，与 Bandit 的走一步看一步一起，让两种思想碰撞，也许可以让你的推荐系统与众不同。

这就是 2016 年有人提出的 COFIBA 算法，下面我就开始与你聊聊这种算法。

COFIBA 算法

1 思想

很多的推荐场景中都有两个规律。

1. 相似的用户对同一个物品的反馈可能是一样的。也就是对一个聚类用户群体推荐同一个 item，他们可能都会喜欢，也可能都不喜欢，同样的，同一个用户会对相似的物品反馈也会相同。这实际上就是基于用户的协同过滤基本思想。
2. 在使用推荐系统过程中，用户的决策是动态进行的，尤其是新用户。这就导致无法提前为用户准备好推荐候选，只能“走一步看一步”，是一个动态的推荐过程。这是 Bandit 的算法基本思想。

每一个推荐候选物品，都可以根据用户对其偏好的不同，将用户分成不同的群体。

然后下一次，由用户所在的群体集体帮他预估可能的收益及置信区间，这个集体就有了协同的效果，然后再实时观察真实反馈，回来更新用户的个人参数用于下次调整收益和置信区间，这就有了 Bandit 的思想在里面。

举个例子，如果你的父母给你安排了很多相亲对象，要不要见面去相一下？那需要提前看看每一个相亲对象的资料，每次大家都分成好几派，有说好的，有说再看看的，也有说不行

的。

你自己也会是其中一派的一员，每次都是你所属的那一派给你集体打分，因为他们是和你“三观一致的人”“诚不欺我”；这样从一堆资料中挑出分数最高的那个人，你出去见TA，回来后把实际感觉说给大家听，同时自己心里的标准也有些调整，重新再给剩下的其它对象打分，打完分再去见，

如果要推荐的候选物品较多，需要对物品聚类，就不用按照每一个物品对用户聚类，而是按照每一个物品所属的类簇对用户聚类，如此一来，物品的类簇数目相对于物品数就要大大减少。

2. 细节

基于上述的思想，COFIBA 算法要点摘要如下。

1. 在时刻 t，有一个用户来访问推荐系统，推荐系统需要从已有的候选池子中挑一个最佳的物品推荐给他，然后观察他的反馈，用观察到的反馈来更新挑选策略。
2. 这里的每个物品都有一个特征向量，所以这里的 Bandit 算法是 context 相关的，只不过这里虽然是给每个用户维护一套参数，但实际上是由用户所在的聚类类簇一起决定结果的。
3. 这里依然是用岭回归去拟合用户的权重向量，用于预测用户对每个物品的可能反馈（payoff），这一点和我们上一次介绍的 LinUCB 算法是一样的。

对比上一次介绍的 LinUCB 算法，COFIBA 的不同有两个：

1. 基于用户聚类挑选最佳的物品，即相似用户集体动态决策；
2. 基于用户的反馈情况调整用户和物品的聚类结果。

整体算法过程如下。

在针对某个用户 i，在每一次推荐时做以下事情。

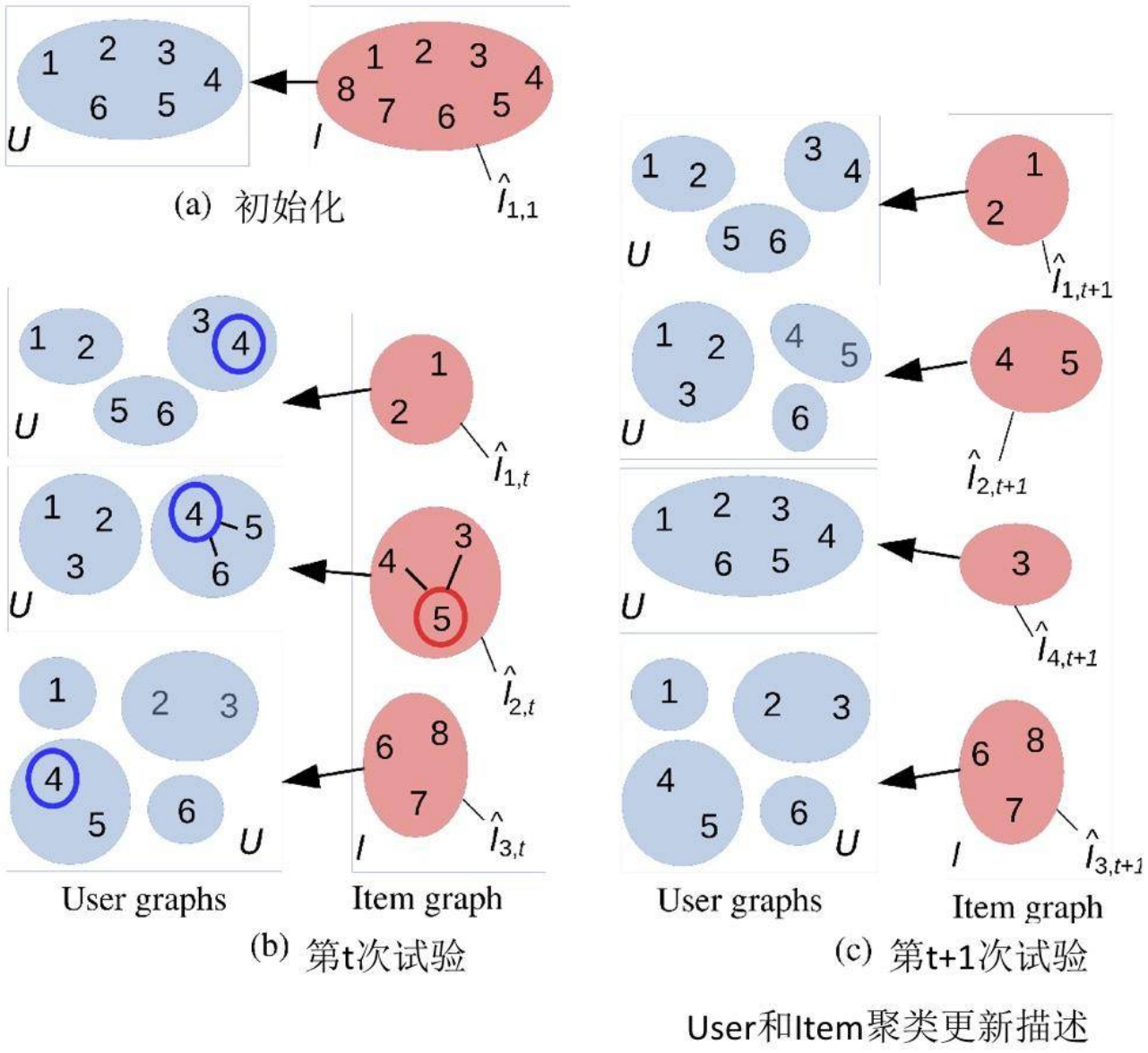
1. 首先计算用户 i 的 Bandit 参数 W，做法和 LinUCB 算法相同，但是这个参数并不直接参与到选择决策中，注意这和 LinUCB 不同，只是用来更新用户聚类。
2. 遍历候选物品，每一个物品已经表示成一个向量 x 了。

3. 每一个物品都对应一个物品聚类类簇，每一个物品类簇对应一个全量用户聚类结果，所以遍历到每一个物品时，就可以判断出当前用户在当前物品面前，自己属于哪个用户聚类类簇，然后把对应类簇中每个用户的 M 矩阵 (对应 LinUCB 里面的 A 矩阵)，b 向量（表示收益向量，对应 LinUCB 里面的 b 向量）加起来，从而针对这个类簇求解一个岭回归参数（类似 LinUCB 里面单独针对每个用户所做），同时计算其收益预测值和置信区间上边界。
4. 每个待推荐的物品都得到一个预测值及置信区间上界，挑出那个上边界最大的物品作为推荐结果。
5. 观察用户的真实反馈，然后更新用户自己的 M 矩阵和 b 向量，只更新每个用户，对应类簇里其他的不更新。

以上是 COFIBA 算法的一次决策过程。在收到用户真实反馈之后，还有两个计算过程：

1. 更新 user 聚类；
2. 更新 item 聚类。

如何更新 user 和 item 的聚类呢？我在这里给出了一个示意图。



解释一下这个图。

(a) 示意图中有 6 个用户，8 个物品，初始化时，用户和物品的类簇个数都是 1。

(b) 在某一轮推荐时，推荐系统面对的用户是 4。推荐过程就是遍历 1～8 每个物品，然后在面对每个物品时，用户 4 在哪个类簇中，把对应类簇中的用户聚合起来为这个物品集体预测收益值置信上边界。这里假设最终物品 5 胜出，被推荐出去了。

在时刻 t，物品一共有 3 个聚类类簇，需要更新的用户聚类是物品 5 对应的用户 4 所在类簇。

更新方式：看看该类簇里面除了用户 4 之外的用户，对物品 5 的预期收益是不是和用户 4 相近，如果是，则保持原来的连接边，否则删除原来的连接边。删除边之后相当于就重新构

建了聚类结果。

这里假设新的聚类结果由原来用户 4 所在的类簇分裂成了两个类簇：4 和 5 成一类，6 单独自成一类。

(c) 更新完用户类簇后，被推荐出去的物品 5，它对应的类簇也要更新。

更新方式是：对于每一个和物品 5 还存在连接边的物品，假如叫做物品 j，都有一个对这个物品 j 有相近收益预估值的近邻用户集合，然后看看近邻用户集合是不是和刚刚更新后的用户 4 所在的类簇相同。

是的话，保留物品 5 和物品 j 之间的连接边，否则删除。这里示意图中是物品 3 和物品 5 之间的连接边被删除。

物品 3 变成了孤家寡人一个，不再和任何物品有链接，独立后就给他初始化了一个全新的用户聚类结果：所有用户是一个类簇。

简单来说就是这样：

1. 用协同过滤来少选可以参与决策的用户代表，用 LinUCB 算法来实际执行选择；
2. 根据用户的反馈，调整基于用户和基于物品的聚类结果，即对物品和用户的群体代表做换届选举；
3. 基于物品的聚类如果变化，又进一步改变了用户的聚类结果；
4. 不断根据用户实时动态的反馈来调整用户决策参数，从而重新划分聚类结果矩阵。

COFIBA 算法也很容易实现，GitHub 上就有。原始论文也从理论和实验两方面都证明了它的有效性。

再谈 EE 问题

整个专栏的 Bandit 算法系列，主要是解决推荐系统中的冷启动和 EE 问题。探索和利用这一对矛盾一直客观存在，而 Bandit 算法是公认的一种比较好的解决 EE 问题的方案。

除了 Bandit 算法之外，还有一些其他的探索兴趣的办法，比如在推荐时，随机地去掉一些用户历史行为（特征）。

解决兴趣探索，势必要冒险，势必要面对用户的未知，而这显然就是可能会伤害当前用户价值的：明知道用户肯定喜欢 A，你还偏偏以某个小概率给推荐非 A。

实际上，很少有公司会采用这些理性的办法做探索，反而更愿意用一些盲目主观的方式。究其原因，可能是因为：

1. 互联网产品生命周期短，而探索又是为了提升长期利益的，所以没有动力做；
2. 用户使用互联网产品时间越来越碎片化，探索的时间长，难以体现出探索的价值；
3. 同质化互联网产品多，用户选择多，稍有不慎，用户用脚投票，分分钟弃你于不顾；
4. 已经成规模的平台，红利杠杠的，其实是没有动力做探索的。

基于这些，我们如果想在自己的推荐系统中引入探索机制，需要注意以下几点：

1. 用于探索兴趣的物品，要保证其本身质量，纵使用户不感兴趣，也不至于引起其反感，损失平台品牌价值；
2. 探索兴趣的地方需要产品精心设计，让用户有耐心陪你玩儿；
3. 深度思考，这样才不会做出脑残的产品，产品不会早早夭折，才有可能让探索机制有用武之地。

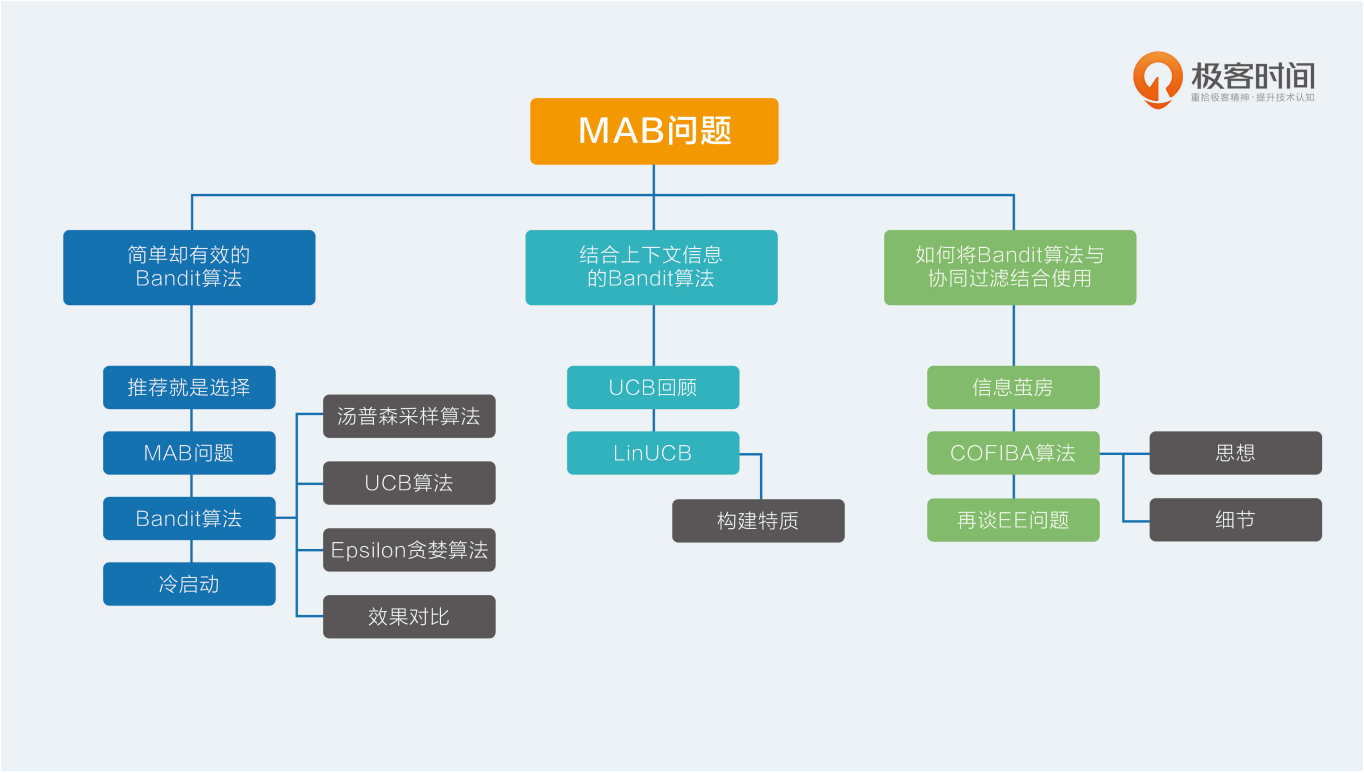
总结

今天，我介绍完成了 MAB 问题和推荐系统之间的千丝万缕联系。Bandit 算法是一种不太常用在推荐系统的算法，究其原因，是它能同时处理的物品数量不能太多。

但是，在冷启动和处理 EE 问题时，Bandit 算法简单好用，值得一试。当然，这个专栏介绍的所有推荐算法都不是单打独斗最好，而是与其他算法结合使用才能相映生辉，Bandit 算法亦是如此。

今天介绍的 COFIOBA 算法，原理很简单，就是把协同过滤思想引入到了 Bandit 算法中，不再是用户独立决策，而是用户所在的群体共同决策推荐结果。

这样比较问题，也可以加速收敛。不知道对你有没有启发呢？欢迎留言一起讨论。感谢你的收听，我们下次再见。



上一篇 【MAB问题】结合上下文信息的Bandit算法

下一篇 【深度学习】深度学习在推荐系统中的应用有哪些?

精选留言 (4)

写留言

林彦 2

2018-04-13

1. 看了下周三的LinUCB文章。这么理解的，COFIBA算法中的M矩阵相当于LinUCB算法中的D矩阵，2个维度都等于内容空间的特征维度数（之前文章中的选择次数m被特征维度数d说取代）。COFIBA算法中的W矩阵相当于LinUCB算法中的西塔θ^。COFIBA算法中的b向量相当于LinUCB算法中的C向量。

...

展开

zgl 2

2018-04-13

请问下，除了推荐算法介绍，有没有实际推荐架构的讲解和分析?

展开

作者回复: 过几天就有了。

虎虎
2018-07-01

1

有个地方没搞懂，推荐的时候做物品遍历，那么物品聚类的作用是什么呢?

展开

shangqiu86
2019-05-05

感觉confiba算法实质上就是把linUCB算法中的当个用户，单个物品全部变成了用户簇，物品簇；信息茧房的问题，感觉真的是理论派的论文中比较多的提到，实际中好像并没有过多的去关注这个问题。不过我觉得要想不断的发展壮大，是需要对EE问题认真对待的。