

第2章 推荐系统中的特征工程

第2章 推荐系统中的特征工程

笔者经常将算法工程师与厨师做类比：

- 厨师的工作是将不同口感、味道的食材，利用适当的烹饪手法（或大火快炒，或小火慢炖），使各种滋味充分融合，制作出美味佳肴。
- 我们算法工程师的工作，是将各种来源、各种形式（e.g.,数字、文本、声音、图像）的原始数据，利用恰当的模型（或简单如LR、GBDT，或复杂如含有几亿参数的超大DNN），使输入的数据充分交叉融合，挖掘出有意义的模式，制作出有用的模型，服务用户。

根据这一比喻，特征工程对于算法工程师，就好比刀功之于厨师，其重要性是再怎么强调也不过分的：

- 再好的食材，不切不洗，一古脑地扔下锅，即便用再大的火力，最后熟不熟都会成问题，更甭提味道了。好的刀功，就是将食材加工成合适的形状，方便吸收火力和其他食材的滋味。这样在上锅烹饪时，无需烈火烹油，也能释放出好的味道与营养。
- 同理，好的特征工程就是为了将原始数据加工成合适的形式，方便模型发挥威力。特征工程做得好，简单模型也能做出不错的效果。当然有了复杂、强大的模型，“或许”更能够锦上添花。

因此，本章将从以下4个方面详细讨论一下推荐系统中的特征工程：

- 有的读者可能不同意我的以上观点。他们受一种流行观点的影响，认为深度学习能够进行“自动化特征工程”，传统的“人工特征工程”已经落伍、过时。在2.1节，笔者首先要批判这种错误观点，再次强调特征工程对于推荐系统的重要性，希望读者能够树立正确的认识。
- 明确意义后，接下来的问题是，从推荐系统的原始数据中，我们需要提取出哪些特征？提取特征是一件需要创意的工作，虽没有一定之规，但是也需要一定的章法，否则东一榔头，西一棒子，提取出来的特征也不好管理与维护。在2.2节，笔者为大家梳理一个提取特征的框架，使提取特征的过程有章可循，提取出来的特征不重复、不遗漏。
- 提取出来的特征，不能直接扔进模型，还需要清洗加工后，特征中的信息才能更好地为模型所吸收。2.3节介绍针对数值特征进行清洗、加工特征的一些方法。
- 2.4节讨论一下推荐系统中的类别特征。推荐算法作为一类特殊的机器学习算法，特点之一就是它的特征空间主要由高维、稀疏的类别特征构成，类别特征是推荐算法的一等公民，享受VIP服务。充分理解

这一特点，是理解其他推荐算法的基础。否则，很多推荐算法精妙的设计，在习惯了低维、稠密特征的算法同行看来，纯属“无病呻吟”。

其实特征工程中还有两个重要话题没出现在本章中：

- 第一个是特征重要性分析和筛选，这项工作不仅能帮助模型瘦身，还能为改进模型提供方向与思路。由于特征评估一般发生在模型训练之后，我们将把它放进第9章“模型的评估与调试”的9.3节加以介绍。
- 第二个是特征快照，使特征工程在线上线下保持严格一致。这部分内容，我们将在第9章“模型的评估与调试”的第9.4.1节再详细讨论。

2.1 批判“特征工程过时”的错误论调

在讲解特征工程的“术”与“道”之前，作者首先要批判一种流行但错误的观点，即“深度学习使特征工程过时”。这种观点认为，深度神经网络（Deep Neural Network，DNN）的功能超级强大，是一个“万能函数模拟器”，只要层数足够多，DNN能够模拟模型的输入与输出之间的任何复杂的函数关系。既然“特征工程”也相当于施加在原始数据上的转换函数，自然也在DNN的模拟范畴之内。这样一来，在前深度学习时代那些让算法工程师费尽心思、绞尽脑汁的手动、显式的特征工程方法就都可以“光荣下岗”了，被DNN的自动化、隐式特征工程所取代。

之所以说这种论调是荒谬的，一是因为它的基础，即“DNN是万能函数模拟器”，已经被越来越多的实践证明是站不住脚的。Deep Cross Network的作者在她的论文[1]里，就直接指出了DNN有时候连简单的二阶、三阶特征交叉都模拟不好。所以DNN的“火力”没有吹得那么强，没经加工的原始食材扔进去，一样可能做不熟。造成神话破灭的一个原因就是，所谓的“万能函数模拟”只是一种理论，而现实训练中的种种问题，比如梯度消失、梯度爆炸、不同特征受训机会不均衡等，都会影响DNN的性能发挥。

二是因为DNN的“自动化特征工程”也不是无代价的。比如，接下来我们会讲到，阿里的DIN模型[2]能够从用户行为序列中挖掘出用户的短期兴趣，SIM模型[3]能够从中挖掘出用户的长期兴趣。既然如此，那么未来是不是没必要再对用户的历史行为做任何特征工程了，把一堆用户交互（e.g.,观看、购买、点赞、转发）过的物料ID直接扔进DIN+SIM不就行了，省时省力，效果也好？答案是否定的。

DIN中的Attention、SIM中的搜索，都属于比较复杂的计算，更要命的是它们的耗时与候选集规模成正比。做做候选集只有几百个物料的精排倒还可以，但这让候选集成千上万的粗排和召回，情何以堪？难道不用DIN+SIM就不能刻画用户的长短期兴趣了吗？当然不是，2.2.1节与2.2.2节介绍的几个特征工程上的技巧就能够派上用场，这几个技巧将计算压力从线上转移到线下，离线挖掘出用户的长短期兴趣，供召回、粗排环节的在线训练与预测使用。

2.2 特征提取

特征工程的第一步是提取特征。前面讲过了，提取特征不能随心所欲，否则难免重复或遗漏。本章将为读者梳理出一套特征提取的框架，使我们的特征提取过程有章可循，提取出来的特征不重不漏。

在正式开始介绍之前，先介绍两个接下来要遇到的概念，Field（特征域）和Feature（特征）。Field是同一类Feature的集合，比如：

- "手机品牌"是一个Field。具体品牌，比如苹果、华为、小米、OPPO、一加等都是这个Field中的Feature。
- 一篇文章的标题与正文是两个Field，某个具体的单词是Feature，在这两个Field中都会出现。

2.2.1 物料画像

不同场景要推荐不同的物品，我们接下来用"物料"（Item）来统一称呼它们。

物料自身属性

这些都是最简单、直接的信息，是物料入库的时候就能够获取到的信息。比如：

- 在视频推荐场景下，视频的作者、作者等级、作者粉丝数、投稿栏目、视频标题与简介、上传时间、时长、清晰度等信息，都属于物料属性。
- 在电商场景下，商品标题与简介、封面图片、所属商铺、商铺等级、品牌、价格、折扣、物料方式、上架时间等信息，都属于物料属性。

值得注意的是，在大型推荐系统中，物料的唯一标识（Item ID）也是重要的特征。可能有些读者不理解，觉得Item ID就是一串无意义的字母数字组合，里面能有啥信息？而且Item ID作为一个类别（Categorical）特征，可能包含有几十万、上百万的特征值，特征空间膨胀得厉害，模型学得出来吗？

- 首先，模型无须理解Item ID那串字符的含义，只要记住就好。比如，模型通过学习历史数据，发现来了一个数码爱好者，只要一推 iPhone 13 (A2634) 5G手机 午夜色 128GB 这个商品，点击率和购买率就非常高。因此，那些销量好的商品的Item ID本身就是非常强的信号，模型只需要把它牢牢记住，就能取得不错的效果。
- Item ID作为一个类别特征，的确是高维、稀疏的，如果傻傻用One Hot Encoding[4]来描述的话，一个几十万长的向量中，只有一个位置是1，其他地方都是0。如果训练数据比较少，的确没必要拿来当特征，

反正也学不出来。但是对互联网于大厂来讲，最不缺的就是训练数据，这时将Item ID当成特征，还是非常有必要的，能够在物料侧提供最个性化的信息。

物料的类别与标签

所谓物料的静态画像，是指不依赖用户反馈，只通过分析物料内容就能获得的物料的类别、标签等信息。简单来说，就是回答“物料是什么”的问题。

内容分析的专业性非常强，所使用的算法也与推荐算法有很大不同，因此一般由专门技术、人工团队来负责。在这里，我只做一个简单的概述，感兴趣的读者可以找专门资料来了解。

- 我们可以用“自然语言处理”（Natural Language Processing, NLP）算法，比如BERT[5]，分析物料的标题、摘要、评论等。如果是文章还可以分析正文，如果是视频还可以分析字幕。
- 我们可以用“计算机视觉”（Computer Vision, CV）算法，比如CNN模型[6]，分析物料的封面，或者视频的关键帧。

内容分析的结果就构成了物料的静态画像，一般包括以下几个方面：

- 一级分类是一个Field，比如“体育”、“电影”、“音乐”、“历史”、“军事”等是其中的Feature。
- 二级分类：比如“体育”又可以细分为“足球”、“篮球”等子类别；“军事”又可以细分为“一战”、“二战”、“战机”、“战舰”等子类别。
- 标签：更细粒度地刻画物料的某个方面。比如“篮球”类别下又可以包含“NBA”、“乔丹”等标签；“战机”类别下又可以包含“歼20”、“F-16”等标签。标签没必要从属于某一具体类别，比如某个明星的名字作为标签，既可能打在“电影”类别的文章上，也可能打在“音乐”类别的文章上。

静态画像可以表示成一个列表（List）。比如一篇娱乐报道，可能既属于“电影”类别，同时也属于“音乐”类别，所以它的二级分类Field可以表示成["电影"、"音乐"]这样的List。

另外，内容理解算法一般不会直接判断一个物料属于哪个类别或标签，而是给出它从属某个类别或标签的概率。因此，我们可以将这个概率与类别、标签一起加入画像，用“映射表”（Map）来表示。比如{"电影": 0.9, "音乐": 0.3}，表示打标算法有90%的信心认为这篇文章是在讲电影，只有30%的信心认为这篇文章在讲音乐。在特征中引入置信度能够给模型提供额外信息，有助于模型做判断。

基于内容的Embedding

以上打标方法，往往是利用CNN或BERT之类的模型，从一篇文章、一个视频中提炼出几个标签，其结果是超级稀疏的。试想一下，整个标签空间可能有几万个标签，而一篇文章往往只含有其中的两三个。

随着深度学习的发展，另一种从物料内容中提取信息的思路是，同样还是用CNN或BERT模型，这回拿模型的某一层的输出，当成物料特征，喂入上层模型。尽管这个向量不如那几个标签好理解，但是它有32位或64位那么长，里面蕴含的信息要比几个标签丰富一些。

物料的动态画像

物料的动态画像，指它们的后验统计数据，反映了物料的受欢迎程度，是物料侧最最重要的特征。

物料的动态画像可以从以下两个维度来进行刻画：

- 时间粒度：全生命周期、过去一周、过去1天、过去1小时、.....；
- 统计对象：CTR、平均播放进度、平均消费时长、排名、.....；

通过以上两个维度的组合，我们可以构造出一批统计指标作为物料的动态画像，比如：

- 文章A在过去6小时的CTR；
- 视频B在过去1天的平均播放时长；
- 商品C在过去1个月的销售额；
-

但是也请读者注意两点：

- 我们需要辩证地看待这些后验统计数据。一方面，它们肯定是有偏的，一个物料的后验指标好，只能说明推荐系统把它推荐给了对的人，并不意味着把它推给任何人都能取得这么好的效果，这里面存在着"幸存者偏差"。另一方面，如果这些后验指标参与精排，"幸存者偏差"的影响还不至于那么严重，毕竟交给精排模型的物料都已经通过了召回、粗排两环节的筛选，多多少少是和当前用户相关的，它们之前的后验指标还是有参考意义。
- 利用这些后验统计数据做特征，多少有些纵容马太效应，不利于新物料的冷启。后验指标好的物料会被排得更靠前，获得更多曝光与点击的机会，后验指标会更好，形成正向循环；而新物料的后验指标不好甚至没有，排名靠后而较少获得曝光机会，后验指标迟迟得不到改善，形成负向循环。我们将在2.3.1节讨论解决这个问题的方法。

用户给物料反向打标签

推荐系统构造特征的一般流程，都是先有物料画像，再将用户消费过的物料的标签积累在用户身上，就形成用户画像。反向打标是把以上流程逆转过来，将消费过某个物料的用户身上的标签，传递积累到这个物料身上，丰富物料画像。

比如一篇关于某足球明星八卦绯闻的文章，由于该球星的名字出现频繁，NLP算法可能会给它打上“体育”标签。但是后验数据显示，带“体育”标签的用户不太喜欢这篇文章，反而带“娱乐”标签的用户更喜欢，显然这篇文章也应该被打上“娱乐”的标签。类似的，给物料打上诸如“文青喜欢的电影榜第3名”、或者“数码迷最喜欢的手机”这样的反向标签，都包含了非常重要的信息，能够帮助提升模型性能。

2.2.2 用户画像

用户的静态画像

静态画像无非就是人口属性（e.g., 性别、年龄、职业、籍贯）、用户安装的APP列表、……等比较稳定的数据信息。获得这些信息的难点主要体现在产品设计层面，即如何在守法合规的前提下使用户心甘情愿地分享个人信息，算法工程师的作用比较有限，这里就不过多展开了。

不展开的另外一个原因是，根据笔者个人观点，人口属性等信息对推荐算法的作用并不大，纯属“食之无味，弃之可惜”的鸡肋。对于老用户，他们丰富的历史行为已经足够反映他们的兴趣爱好，自然轮不上人口属性等静态画像发挥重要使用。对于没啥历史行为的新用户，既然主力信息来源失效，很多人都寄希望于通过知道用户的性别、年龄、安装了哪些APP等信息猜测出用户的兴趣爱好，进而进行个性化推荐。但是仅从笔者个人的经验来看，这么做纯属徒劳，效果并不明显。笔者总结这其中的两方面原因：

- 如果新老用户共用一个模型，比如精排，因为老用户贡献的样本多，从而主导了训练过程，导致训练出来的模型不会重视静态画像等对新用户“友好”的特征。
- 如果新用户使用单独模型，比如我们可以基于静态画像单独为新用户添加一路召回，但是由于新用户的行为少，噪声多，我们又没有足够、可靠的数据训练出对新用户有效的模型。

不过与Item ID类似，用户唯一标识（User ID）倒是一个非常重要特征，因为它提供了用户侧最细粒度的个性化信息。但是它的缺点也很明显：

- 首先，User ID的取值要覆盖上亿用户，使特征空间膨胀得厉害。好在现代大型推荐系统都引入了Parameter Server分布式存储模型参数，User ID增加了一些参数量，问题倒是不大。
- 其次，User ID的功效也是因人而异。活跃用户贡献的样本多，足以将他们的User ID对应的模型参数（e.g., 一阶权重或Embedding）训练好。但是对于那些不活跃用户甚至新用户，他们的行为有限，贡献不了几条样本，他们的User ID对模型效果就纯属“打酱油”了。由于互联网大厂的主打APP的存量老用户居多，行为丰富，能够提供足够的训练数据，因此在大厂的实践中还是非常喜欢拿User ID当特征的。

用户的动态画像

用户的动态画像就是从用户的历史行为中提取出来的他的兴趣爱好。说它是动态的，是因为相比于稳定的人口属性信息，这部分信息变化频繁，能够及时反映出用户的兴趣迁移。

最简单直接的动态画像就是，将用户一段时间内用户交互过的物料的Item ID按时间顺序组成的集合。将这个集合扔进模型，让模型自动从中提取出用户兴趣。最简单的提取方式无非就是将每个Item ID先 Embedding，再把多个Embedding聚合（也称“池化”，Pooling，比如采用加和或平均）成一个向量，这个向量就是用户兴趣的抽象表达。更复杂一点，可以像DIN[2]或SIM[3]那样在Pooling时引入Attention机制，根据候选物料的不同，模型从相同的行为序列中提取出不同的用户兴趣向量，做到“千物千面”，具体细节将在本书的4.3.3节加以详细介绍。整个过程如图2-1所示。

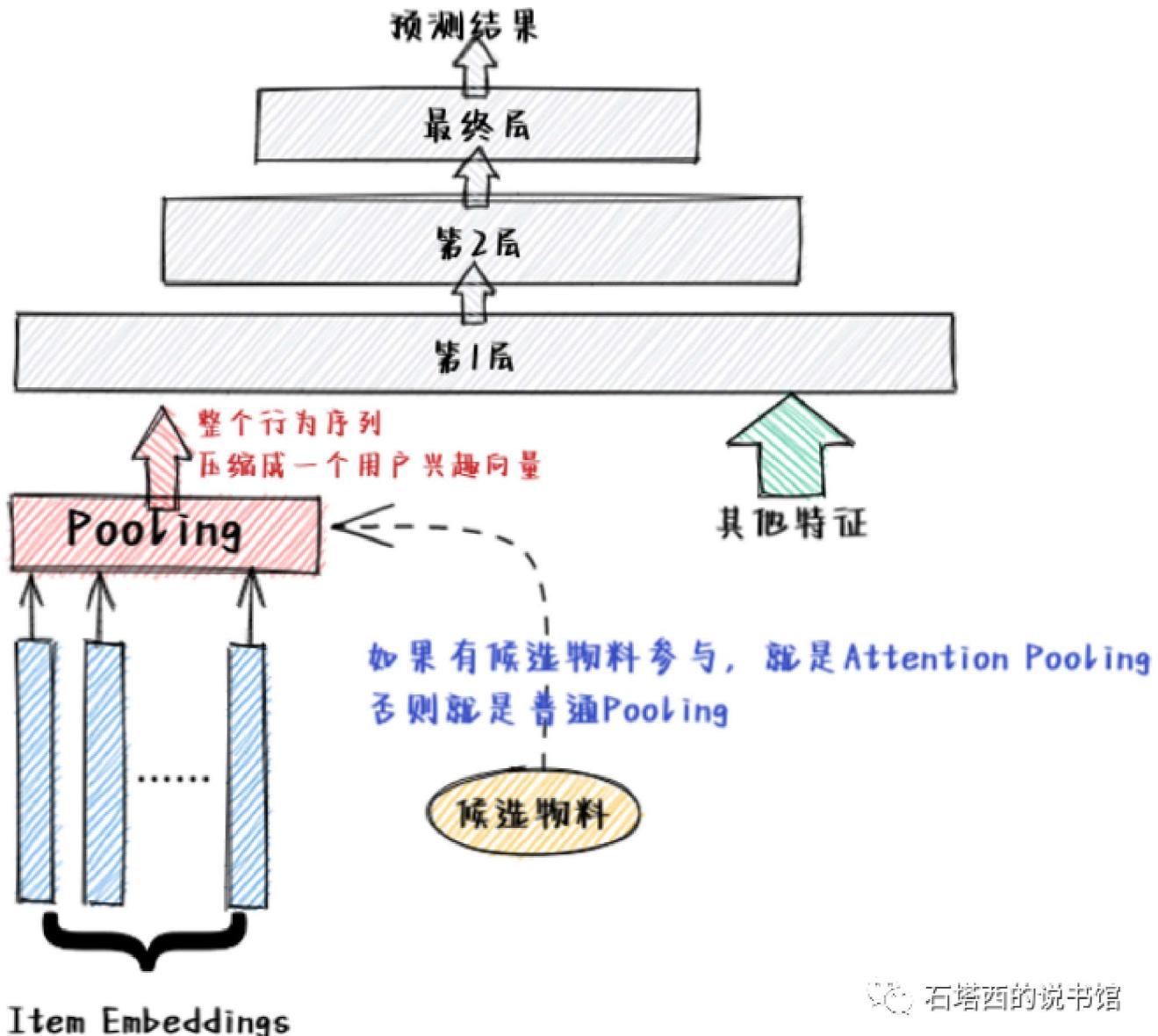


图 2-1 将行为序列喂入模型

这种将用户行为序列直接扔进模型的作法，优点是简单直接，无需过多的特征处理工作。它的缺点是，抽取用户兴趣与CTR建模合为一体，都必须在线上完成。特别是在使用DIN或SIM这种“强大但复杂”的模型提取兴趣时，耗时与“序列长度×候选集规模”成正比。所以，当我们希望从历史更久远、长度更长的行为序列中

提取用户兴趣，或者将其应用于召回、粗排等候选集规模很大的场景，这种作法根本就无法满足在线预测与训练的实时性要求。另外，这种作法抽取出来的用户兴趣是抽象的向量，解释性很差。

为克服以上缺点，我们可以将“提取用户兴趣”与“CTR建模”解耦，将提取兴趣的工作从线上转移到线下，通过Hadoop/Spark这样的大数据平台，从用户行为历史提取各类统计指标来描述用户兴趣，并且灌到数据库中。在线预测与训练时，只需要根据User ID去数据库中查询相应指标即可，耗时极少。由于计算主要发生在时间充裕的离线环境，我们在统计用户兴趣的时候，可以使用更复杂的算法，回溯更长的历史，关联更多的数据源。

这种“离线抽取，在线查询”的抽取方法，其优缺点与“在线抽取”正好相反：

- 优点：线上获取用户兴趣，只是一个查询操作，耗时低，非常适用于召回、粗排这种候选集庞大的任务。另外，用户兴趣是用各种统计指标来表示，相比于抽象的向量，简单直白，易于理解，而且也能够为运营、用户增长等非算法团队提供帮助。
- 缺点：抽取出来的兴趣不会随候选物料而改变，针对性不强，无法做到“万物千面”。另外，兴趣抽取主要靠离线定时进行，不能像在线模式那样及时捕捉到用户的兴趣迁移。

至于可以统计哪些指标来反映用户兴趣，我们可以从以下6个维度展开，做到不重不漏：

- 用户粒度，可以是单个用户，也可以是一群用户。针对一个用户群体的统计，有利于新用户冷启。
- 时间粒度，比如最近的100次曝光，再比如过去1小时、1周、1月。
- 物料属性，比如视频的一二级类别、标签、作者，再比如商品的分类、品牌、店铺、价位。
- 动作类型，可以是正向的，比如点击、点赞、转发等，也可以是负向的，比如忽略、点踩。
- 统计对象，比如次数、时长、金额等。
- 统计方法，比如加和、求平均、计算各种比例等。

通过以上6个维度的交叉，我们可以构造出一系列的统计指标来反映用户在各个时间跨度、各个维度上的兴趣爱好，比如表 2-1。

表 2-1 基于6个维度构造用户动态画像的示例

用户粒度	时间粒度	物料属性	动作类型	统计对象	统计方法
用户A	过去1天	"坦克"标签	点击	次数	CTR (比如给该用户推了10篇带"坦克"标签的文章，用户点击了其中的6篇，CTR=0.6)
用户B	过去1周	"坦克"标签	点击	次数	点击占比 (比如用户一共点击了10篇文章，其中6篇带"坦克"标签，点击占比就是0.6)

用户粒度	时间粒度	物料属性	动作类型	统计对象	统计方法
用户C	过去1周	"坦克"标签	观看	时长	时长占比（比如用户一共看了100分钟视频，其中60分钟看了带"坦克"标签的视频，时长占比是0.6）
用户D	最近100次购买	"食品"分类	购买	金额	加总
男性用户	过去1周	"军事"分类	点击	次数	CTR

2.2.3 交叉特征

构建交叉特征，在前深度学习时代是提升推荐模型性能的非常重要的手段。近年来，有一种说法认为既然DNN能够自动完成特征交叉，就没必要劳神费力地进行人工特征交叉了。这种说法是非常片面的，一方面，DNN并非万能，其交叉能力也有限；另一方面，手动交叉的特征犹如加工好的食材，个中信息更容易被模型消化吸收。因此，在推荐模型的深度学习时代，交叉特征依然大有可为，值得重视。

在具体交叉方式上，又有做“笛卡尔积”与做“内积”两种方式。

笛卡尔积交叉

笛卡尔交叉就是将两个Field内的Feature两两组合，组成一个新的Field。比如用户感兴趣的电影类别有{"动作片"、"科幻片"}，而当前候选物料的标签是{"施瓦辛格"、"终结者"、"机器人"}，这两个Field做笛卡尔交叉的结果就是{"动作片+施瓦辛格", "动作片+终结者", "动作片+机器人", "科幻片+施瓦辛格", "科幻片+终结者", "科幻片+机器人"}，显然"动作片+施瓦辛格"、"科幻片+机器人"都是非常强烈的信号，有助于模型判断用户与物料间的匹配程度。

至于交叉后的特征如何喂入模型，可以参考将用户行为序列喂入模型的方法，如图 2-1所示。

内积交叉

另一种特征交叉的方法是做点积，即选定一个画像维度（比如标签、分类），将用户在这个维度上的兴趣，和物料在这个维度上的属性，想像成两个稀疏向量，这两个向量做点积结果反映出用户和物料在这个画像维度上的匹配程度。点积结果越大，说明用户与候选物料在这个维度上越匹配。

比如用户感兴趣的标签是 $Tags_{user} = \{"坦克": 0.8, "足球": 0.4, "二战": 0.6, "台球": -0.3\}$, 每个标签后面的数字表示用户对这个标签的喜爱程度, 可以拿用户在这个标签上的后验指标来表示, 具体计算方法可以参考表 2-1。而当前候选物料的标签是 $Tags_{item} = \{"坦克": 1, "二战": 0.5, "一战": 0.8\}$ 。将 $Tags_{user}$ 与 $Tags_{item}$ 做点积, 也就是将共同标签对应的分数相乘再相加, 结果是1.3, 表示用户与当前候选物料在“标签”这个维度上的匹配程度。

2.2.4 偏差特征

众所周知, 推荐模型是根据用户的曝光点击记录训练出来的。这其中蕴含的前提假设是, 我们认为点击与否反映了用户的真实兴趣爱好, 但是严格来讲, 以上假设并不成立。具体实践中, 我们无法做到让所有候选物料在一个绝对公平的环境中供用户挑选, 这也就意味着用户的选择并非完全出于他的兴趣爱好, 用户点击的未必是他喜欢的, 没点击的也不代表用户就一定不喜欢。这种不可避免地引入的不公平因素就叫作“偏差”(Bias)。

推荐系统中最常见的是“位置偏差”(Position Bias), 如图 2-2 所示。

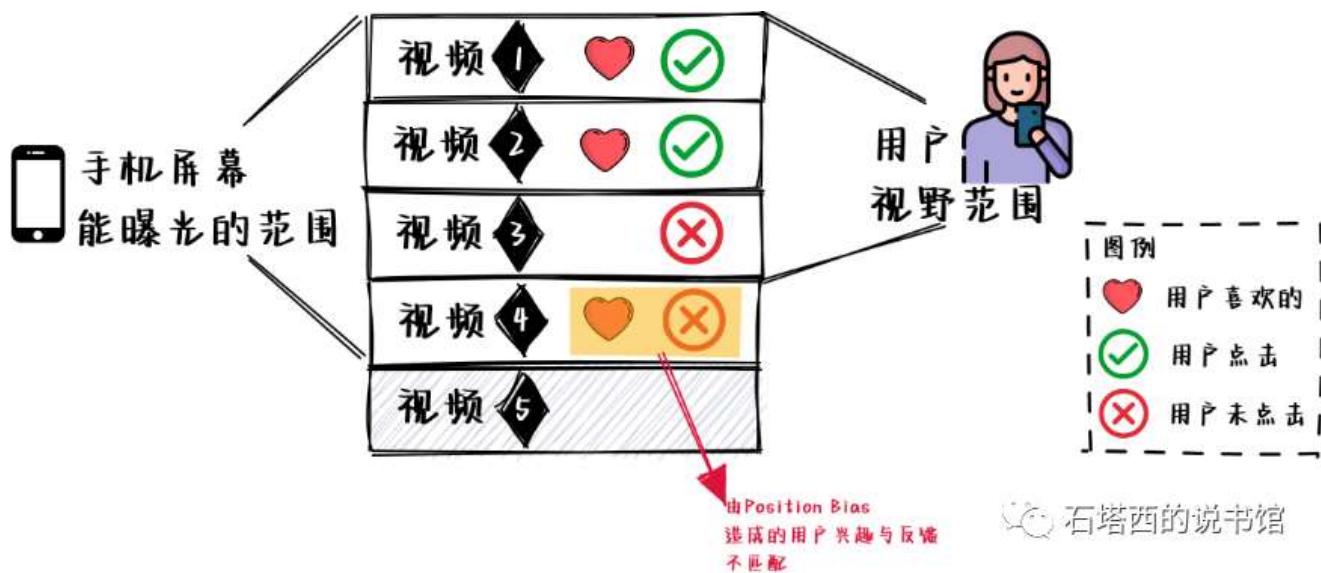


图 2-2 Position Bias示意

图 2-2 中视频1和2, 都是用户喜欢的, 并且占据了靠前的醒目位置。用户注意到了它们, 并通过点击给出了正面反馈。视频4是一个体育视频, 其实也是用户喜欢的, 如果用户看到了, 是一定会点击的。但是由于视频4的曝光位置太偏, 落在了用户的视线之外, 所以未发生点击, 训练时会被当成负样本, 显示用户不喜欢视频4。这就是由“位置偏差”引发的用户兴趣与反馈之间的脱节。这种偏差会误导模型, 因为模型看到视频4的负样本时, 可不知道它是由于位置太偏造成的, 反而会猜测这名用户可能不喜欢“体育”这个类别, 未来减少给他推荐“体育”类视频, 从而降低了用户体验。

至于如何解决“位置偏差”，一种方法是从数据入手，比如更加严格地定义正负样本。比如有一种Above Click的作法规定，只有在点击物料上方的未点击物料，才能被纳入训练数据当成负样本，如图2-3所示。

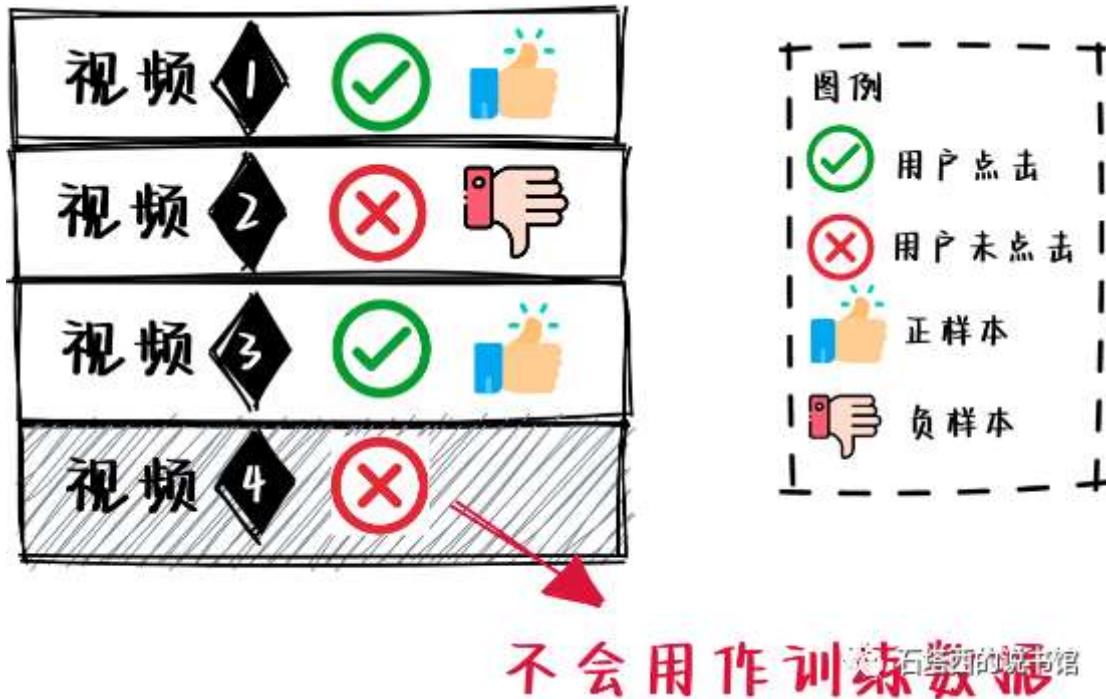


图 2-3 Above Click示意

图 2-3 中视频2和4都未被点击。根据Above Click规则，由于视频3是最下边一条被点击的视频，视频2被认为是真正被用户看到了却未点击，说明用户真心不喜欢视频2，应该当成一条负样本。而视频4的未点击可能是因为未被用户看到，谨慎起见，就不被参模型训练了。

另一种解决方法就是从模型入手。前面讲到了，模型之所以被误导，是因为它不知道这个负样本是由于偏差造成的，还是由于其他别的原因。纠正的方法就是将偏差因素当成特征也喂进模型，使模型有足够的信息来给用户反馈找出合理解释。但是这里又出现了一个新问题，“曝光位置”在训练时倒是能够拿得到，但是在在线预测时，“曝光位置”是模型预测的“果”，怎么可能作为“因”喂入模型呢？

对于以上这种“本末倒置”的问题，我们的解决方案是，在预测时，将所有候选物料的曝光位置一律填写成0，也就是假设所有候选物料都展示在最醒目的位置上，让模型根据其他因素给候选物料打分，并根据打分形成最终真实的展示顺序。像“0号展示位置”这种因为预测时拿不到而统一填充的特征值，我们称它们为“伪特征值”。

读到这里，细心的读者可能提出疑问：为什么在预测时要将所有展示位置填成0，填成1或2不行吗？如果不这样的话，岂不是选择不同的“伪特征值”，排序结果也会发生变化？那就说明排序过程深受一个未知因素的影响，听着都不靠谱。

这就牵扯到另外一个重要问题，就是这个Dummy Feature应该加在模型的什么位置上？答案是：Dummy Feature只能通过一个线性层加入模型，绝对不能和其他正常特征一起喂入DNN，如图2-4所示。只有这样接入，才能保证预测时无论“伪特征值”的取值如何，都不会改变排序结果，也就回答了细心读者的问题。

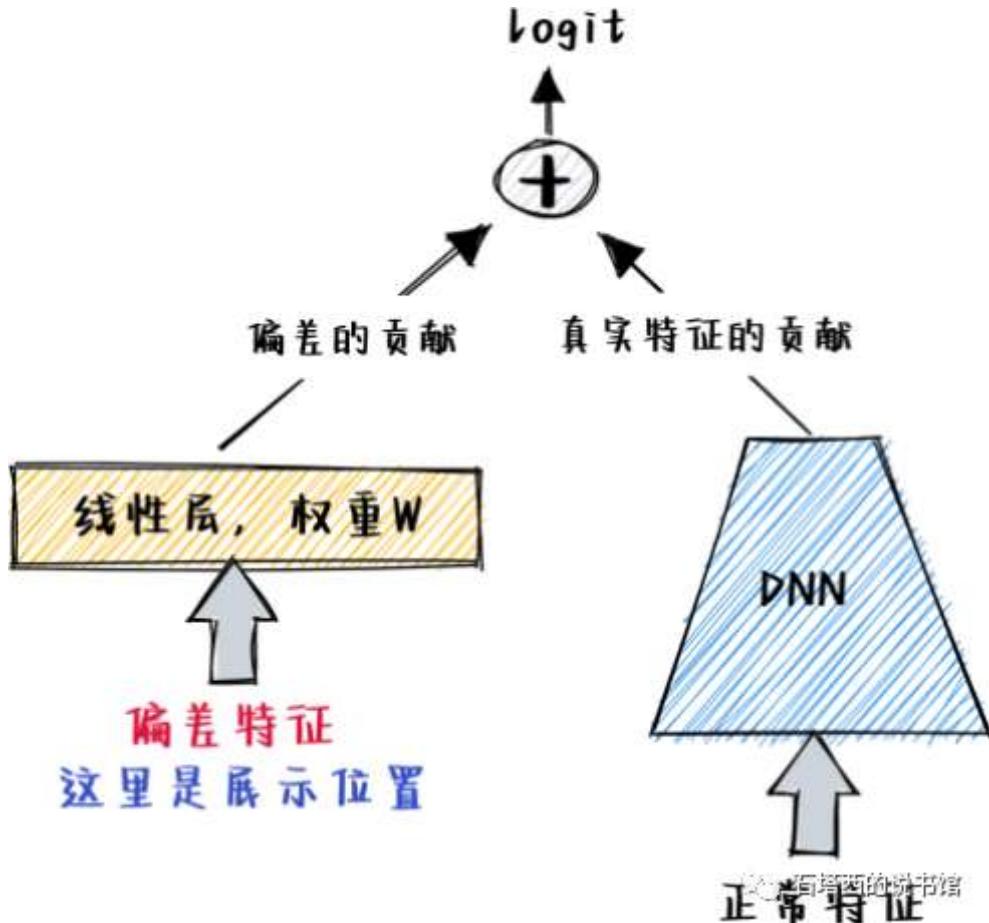


图 2-4 Dummy Feature只能通过线性层插入模型

简单分析其中的原因，按照图 2-4，最终打分logit如公式(2-1)所示。

$$\text{logit} = \text{DNN}(u, t) + w^T b \quad (2-1)$$

- DNN是推荐模型的深层网络部分
- u 和 t 表示来自用户与候选物料的真实特征。
- w 是负责接入偏差特征的线性层的权重。
- b 代表偏差特征向量，这个例子中只有"展示位置"一个值。训练的时候， b 取真实的展示位置。预测的时候， b 统一填成某个"伪特征值"。

如果按照图2-4那样接入"偏差特征"，排序结果能够满足公式(2-2)，即无论"伪特征值"取多少，都不影响我们得到真实排序。

$$\begin{aligned}
 & \text{SORT}(\text{DNN}(u, t_1) + \mathbf{w}^T \mathbf{b}_1, \text{DNN}(u, t_2) + \mathbf{w}^T \mathbf{b}_1) \\
 &= \text{SORT}(\text{DNN}(u, t_1) + \mathbf{w}^T \mathbf{b}_2, \text{DNN}(u, t_2) + \mathbf{w}^T \mathbf{b}_2) \\
 &= \text{SORT}(\text{DNN}(u, t_1), \text{DNN}(u, t_2))
 \end{aligned} \quad (2-2)$$

- DNN是推荐模型的深层网络， w 是负责接入偏差特征的线性层的权重。
- u 代表用户的真实特征， t_1 和 t_2 代表两个候选物料的真实特征。
- \mathbf{b}_1 和 \mathbf{b}_2 表示给偏差特征，先后使用两个不同的"伪特征值"。

- $SORT$ 代表排序函数。
- $SORT(DNN(u, t_1), DNN(u, t_2))$ 表示只基于真实特征的排序结果，也是我们希望得到的排序结果。

反之，如果将偏差特征和其他真实特征一同喂入DNN，则排序结果满足公式(2-3)。

$$\begin{aligned} & SORT(DNN(u, t_1, \mathbf{b}_1), DNN(u, t_2, \mathbf{b}_1)) \\ & \neq SORT(DNN(u, t_1, \mathbf{b}_2), DNN(u, t_2, \mathbf{b}_2)) \end{aligned} \quad (2-3)$$

这是因为DNN的高度非线性，使“伪特征值”与真实特征值产生深度交叉，预测时采用不同的“伪特征值”，将产生完全不同的排序结果。这就使排序结果严重依赖于一个在预测时的未知因素，是绝不可用的。

除了“位置偏差”，Youtube[7]还发现视频的“年龄”（当前时间减去上传时间）也会造成偏差。推荐模型会用视频的各种后验消费指标（比如点击率、人均观看时长等）来衡量物料的受欢迎程度。上传早的视频有足够长的时间来积累人气，所以后验指标更好，模型排名更高；反之，新上传的视频还没有积累起好看后的后验数据，模型排名较低，不利于新视频的冷启动。为了减轻这一偏差，Youtube在训练时将视频年龄作为偏差特征喂入模型，而在预测时统一设置成0。

2.3 数值特征的处理

2.3.1 处理缺失值

如果某条样本 x 中某个实数特征 F 的取值缺失，最常规的作法就是拿所有样本在特征 F 上的均值（Mean）或中位数（Median）代替。当然我们可以做得更精细一些。比如来了一个男性新用户，他对“体育”类视频的CTR未知，我们可以拿所有男性用户对“体育”视频的CTR来填充这一缺失值，相比于用全体用户的均值来填充更有针对性。

但是如果我们又知道了该新用户的年龄，难道又要根据“性别+年龄”的组合重新划分人群计算均值？那样岂不是太麻烦了。一个更合理的作法是训练一个模型来预测缺失值。比如对于新用户，我们可以构建一个模型，利用比较容易获得的人口属性（比如性别、年龄等）预测新用户对某个内容分类、标签的喜爱程度（比如对某类内容的CTR）。再比如对于新物料，我们可以训练一个模型，利用物料的静态画像（比如分类、标签、品牌、价位）预测它的动态画像（CTR、平均观看时长、平均销售额等）。

当然，如果我们选择对数值特征离散化，处理缺失值就更容易了。我们可以在分桶时，为每个特征都增加一个叫“未知”的桶，专门用来映射该特征的缺失值。

2.3.2 标准化

标准化的目的是将不同量纲、不同取值范围的数值特征都压缩到同一个数值范围内，使它们彼此可比。最常用的标准化是z-score标准化，如公式(2-4)所示。

$$x^* = \frac{x - \mu}{\sigma} \quad (2-4)$$

- x 是某条样本在特征F的原始取值。
- μ 、 σ 分别是特征F在训练数据集上的均值和标准差。
- x^* 是某条样本在特征F的标准化结果，称为z-score。

值得一提的是，推荐系统中经常会用到一些长尾分布的特征，比如观看次数，大多数用户一天观看不到100个视频，但是也有个别重度用户一天就要刷1000个视频。这种情况下，直接统计出来的 μ 和 σ 都会被长尾数据带偏，从而根据公式(2-4)计算出来的z-score区分度下降。

解决方法就是对原始数据做开方、取对数等非线性变换，先将原来的长尾分布压缩成接近正态分布，再在变换后的数据调用公式进行z-score标准化，能够取得更好的效果，如(2-5)所示。



图 2-5 对长尾分布的数据先压缩成接近正态分布

2.3.3 数据平滑与消偏

推荐系统中经常要计算各种比率作为特征，比如点击率、点赞率、购买率、复购率等。计算这些比率时，我们经常遇到的一个问题就是样本太少，导致计算结果不可信。比如计算一件商品，只被曝光了一次并被购买了，由此我们就说它的购买率是100%，从而认定它是爆款，应该大力推荐，显然这是站不住脚的。为克服小样本的负面影响，提高计算结果的置信水平，我们可以采用“威尔逊区间平滑”，如公式(2-5)所示。

$$p^* = \frac{p + \frac{z^2}{2n} - z\sqrt{\frac{p(1-p)}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}} \quad (2-5)$$

- z 是一个超参，代表对应某个置信水平的z-score。比如当我们希望计算结果有95%的置信水平时， z 应该等于1.96。

- p 是用简单方法计算出的比率。比如当 p 代表点击率时，就是拿点击样本数除以曝光样本数。
- p^* 是平滑后的比率。
- n 是样本数量。从公式中可以看出，当 n 非常大时， p^* 趋近于 p ，这也符合统计学中的大数定理与我们的直觉。而当 n 比较小时， p^* 会远小于 p ，从而能够避免出现"曝光1次，点击1次，CTR等于100%"的不靠谱结论。

另一种消偏是为了抹平不同细分领域天然存在的偏差，使不同细分领域得到的统计指标尽量公平可比。假设有A、B两篇文章，经过充分曝光后，A的点击率是10%，B的点击率是8%，是否一定说明A比B更受欢迎，接下来应该大力推荐？答案是否定的。仔细观察我们发现，文章A一般都展示在比较靠前、显眼的位置，而文章B的展示位置一般都在不醒目的边边角角。众所周知，展示位置对用户是否点击起着至关重要的作用。尽管文章B的位置不好，但是其点击率与占据好位置的文章A不相上下，这不正好说明B更受欢迎吗？下次如果我们能将B放在更引人注目的位置上，它的消费数据会更加出色。

至于为什么B那么优秀却未能显示在好位置，其中一个可能的原因就是使用CTR来衡量物料的受欢迎程度，而CTR没有考虑不同显示位置的吸引眼球的能力不同，从而使展示在较差位置上的优质物料没有出头之日。至于解决方法，我们在2.2.4节提到过可以将"展示位置"也作为特征喂入模型，这里提出另一种消偏的思路，就是用CoEC (Click Over Expected Click) 代替CTR衡量物料的受欢迎程度。CoEC的计算如公式(2-6)所示。

$$CoEC = \frac{\sum_{i=1}^N c_i}{\sum_{i=1}^N ec_{p_i}} \quad (2-6)$$

- i 表示第 i 次曝光样本， N 是样本总数。
- c_i 表示第 i 次曝光是否发生点击， $c_i = 1$ 表示发生点击， $c_i = 0$ 表示未发生点击。
- p_i 是第 i 次曝光的位置。
- ec_{p_i} 是位置 p_i 的期望点击，即在这个位置曝光一次，平均能产生多少次点击，其实就是这个位置上的CTR，可以离线通过大数据运算提前统计好。

CoEC消除了不同展示位置带来的偏差因素，从而更能公平地反映各物料的受欢迎程度。

2.3.4 分桶离散化

接下来在2.4节中，我们会讲到，在推荐模型中，使用类别特征具有能更好反映非线性关系、便于存储与计算等多方面优势，因此在实践中，我们更喜欢将实数特征离散成类别特征。离散方法就是分桶，即将实数特征的值域划分为若干区间，又称为"桶"，看实数特征落进哪个桶，就以那个桶的桶号作为类别特征值。比如，某用户在最近一小时看了5个视频，如果用实数特征描述，特征是"最近1小时看的视频数"，特征值是5。而如果离散成类别特征，整个特征可以表示成"last1hour_0_10"这个字符串，表示该用户在最近1小时看的视频数在0~10之间。至于如何将这个字符串传入模型，我们将在2.4.4节予以介绍。

分桶有三种实现方式：

- 等宽分桶：即将特征值域平均划分为了 N 等份，每份算一个桶。
- 等频分桶：将整个值域的 N 个分位数（Percentile）作为各桶的边界，保证落入各个桶的样本个数大致要相等。
- 模型分桶：对实数特征 F 分桶，要分为两个阶段。第1阶段，单独拿特征 F 与目标值拟合一棵简单的决策树[8]。第2阶段才进行分桶，将某个特征中 F 的实数取值喂进决策树，最终落进的那个叶子节点的编号就是 F 的离散化结果。

2.4 类别特征的处理

推荐算法作为一类特殊的机器学习算法，特点之一就是它的特征空间主要由高维、稀疏的类别特征构成，类别特征是推荐算法的一等公民，享受VIP服务。充分认识这一特点至关重要，它是理解推荐算法的基础。否则，很多推荐算法的技巧，在习惯了稠密数值特征的算法同行看来，都是“无病呻吟”。

推荐算法作为一类特殊的机器学习算法，特点之一就是它的特征空间主要由高维、稀疏的类别特征构成，类别特征是推荐算法的一等公民，享受VIP服务。充分理解这一特点，是理解其他推荐算法的基础。否则，很多推荐算法精妙的设计，在习惯了低维、稠密特征的算法同行看来，纯属“无病呻吟”。

2.4.1 类别特征更受欢迎

说高维、稀疏的类别特征是推荐系统的一等公民，因为它们更加契合推荐系统的特点。

首先，推荐系统的基础是用户、物料画像。画像中的一二级分类、标签都是类别特征，并且高维（比如一个推荐系统中有几万个标签是小意思）、稀疏（比如一篇文章至多包含那几万个标签中的十几个）。而且，为了增强推荐结果的个性化成份，互联网大厂都喜欢将User ID、Item ID这种最细粒度的特征加入模型，显然这些特征都是类别特征，并且将特征空间的维度和稀疏性都提高了许多。

其次，推荐模型的输入特征与要拟合的目标之间鲜有线性关系，更多的是量变引起质变。举例来说，在电商场景下，我们希望建模用户年龄与其购买意愿、兴趣之间的关系。我们可以用数值特征来描述，特征叫“用户年龄”，A用户的特征值是20，B用户的特征值是40。“用户年龄”在模型中对应一个Embedding，代表它对目标的贡献。使用时，拿特征值当成权重与特征Embedding相乘，表示对其贡献进行缩放。照这么说，年龄对B用户的购买意愿的影响是对A用户的两倍？这个结论肯定是不能成立的。

显而易见的，不同年龄段（少年、青年、中年、老年）对购买意愿、兴趣的影响绝非线性，而是拥有各自不同的内含。所以正确的方式应该是将每个年龄段视为独立的类别特征，要学习出自己的权重或 Embedding。比如A用户使用的"青年"Embedding在向量空间中可能与一些价格适中、时尚爆款的商品接近，而B用户使用的"中年"Embedding可能与一些经典、单价高的商品接近，二者绝非向量长度相差一倍那种简单关系。

线上工程实现更偏爱类别特征，因为推荐系统中的类别特征超级稀疏，可以实现非零存储、排零计算，减少线上开销，提升在线预测与训练的实时性。以Logistic Regression (LR) 为例，如公式(2-7)所示。

$$\begin{aligned} \text{logit} &= \mathbf{w}^T \mathbf{x} + b & (a) \\ \text{logit} &= b + \sum_{j \in \{i | x_i \neq 0\}} w_j & (b) \end{aligned} \quad (2-7)$$

- 公式(2-7)(a)使用了实数特征，公式(2-7)(b)使用了类别特征。
- x 是一条样本的特征向量， w 是LR模型学习出来的权重向量， b 是偏置项（bias）。
- 如果使用实数特征，如公式(2-7)(a)所示，需要进行很多乘法运算。
- 如果使用类别特征，如公式(2-7)(b)所示，只需要将非零特征对应的权重相加即可，节省了乘法计算。而且由于特征空间的稀疏性，一条样本中的非零特征并不多，运算速度更快。

2.4.2 类别特征享受VIP服务

说高维、稀疏的类别特征是一等公民，因为推荐系统中的很多技术都是为了更好地服务它们而专门设计的。

第一，单个类别特征的表达能力弱，为了增强其表达能力，业界想出了两个办法：

- 通过Embedding自动扩展其内涵。比如"用户年龄在20~30之间"这一个类别特征，既可能反映出用户经济实力有限，又可能反映出用户审美风格年轻、时尚。这一系列的潜台词，偏学术一点叫"隐语义"，都可以借助Embedding自动学习出来，扩展了单个特征的内涵。
- 多特征交叉。比如单拿"用户年龄在20-30之间"这一个特征，推荐模型可能还猜不透用户的喜好。再交叉上"工作"特征，比如"用户年龄在20-30之间、工作是程序员"，推荐模型立刻就明白了，"格子衬衫"对该用户或许是一个不错的选择。

第二，类别特征的维度特别高，几万个标签是小意思，再加上实数特征分桶、多维特征交叉，特征空间的维度轻轻松松就上亿，如果把User ID、Item ID也用作特征，特征空间上十亿都有可能打不住。要存储这么多特征的权重和Embedding向量，单机是容纳不下。于是，Parameter Server这样的架构应运而生，一方面 Parameter Server利用分布式集群分散了参数存储、检索的压力，另一方面它利用了推荐系统的特征空间超

级稀疏这一特点，每次计算时无须同步整个特征空间上亿特征的参数，大大降低了带宽资源与时间开销。关于Parameter Server的技术细节，我们将在3.3节加以详细讲解。

第三，类别特征本来就是稀疏的，“实数特征离散化”和“多特征交叉”进一步提高了特征空间的稀疏程度，从而降低了罕见特征的受训机会，导致模型训练不充分。为了解决这一问题，业界也想出了很多办法：

- FTRL这样的优化算法为每维特征自适应地调节学习率[9]。常见特征受训机会多，步长小一些，防止因为某个异常样本的梯度一下子冲过头。罕见特征的受训机会少，步长可以大一些，允许利用有限的受训机会，快速收敛。
- 阿里的DIN[2]为每个特征自适应地调节正则系数。
- 对第*i*个特征 x_i 与第*j*个特征 x_j 的交叉特征前面的权重 w_{ij} ，LR只能拿 x_i 与 x_j 都不为0的样本来训练，在推荐系统这种特征超级稀疏的环境下，这样的样本少之又少，导致交叉特征的权重不能充分训练。FM[10]解决了这一问题，使只要 $x_i \neq 0$ 或 $x_j \neq 0$ 的样本都能参与训练 w_{ij} ，大大提升了训练效率。关于FM的技术细节，我们将在4.2.2节加以详细介绍。

2.4.3 映射

前面列举了推荐系统中很多的类别特征，它们可以是

- User ID、Item ID、Shop ID、Author ID。
- 画像中的类别、标签。
- 实数分桶离散化后的结果，比如“用户观看了3~10个视频”。
- 交叉结果，比如“用户喜欢军事，候选物料带坦克标签”。

为了方便讲解，以上类别的值都是字符串。但是，大家都知道文字是无法直接喂进模型当特征的，我们必须先将它们数字化。

最简单的方法就是建立一张字符串到数字的映射表，如图 2-6所示。

- 我们收集常见标签组成映射表，将标签映射成一个整数。
- 映射成的整数对应Embedding矩阵中的行号。通过这种形式，我们为每个标签查询得到它对应的Embedding向量。
- 接下来就是常规操作了，这些标签的Embedding向量Pooling成一个向量，喂入上层的DNN。

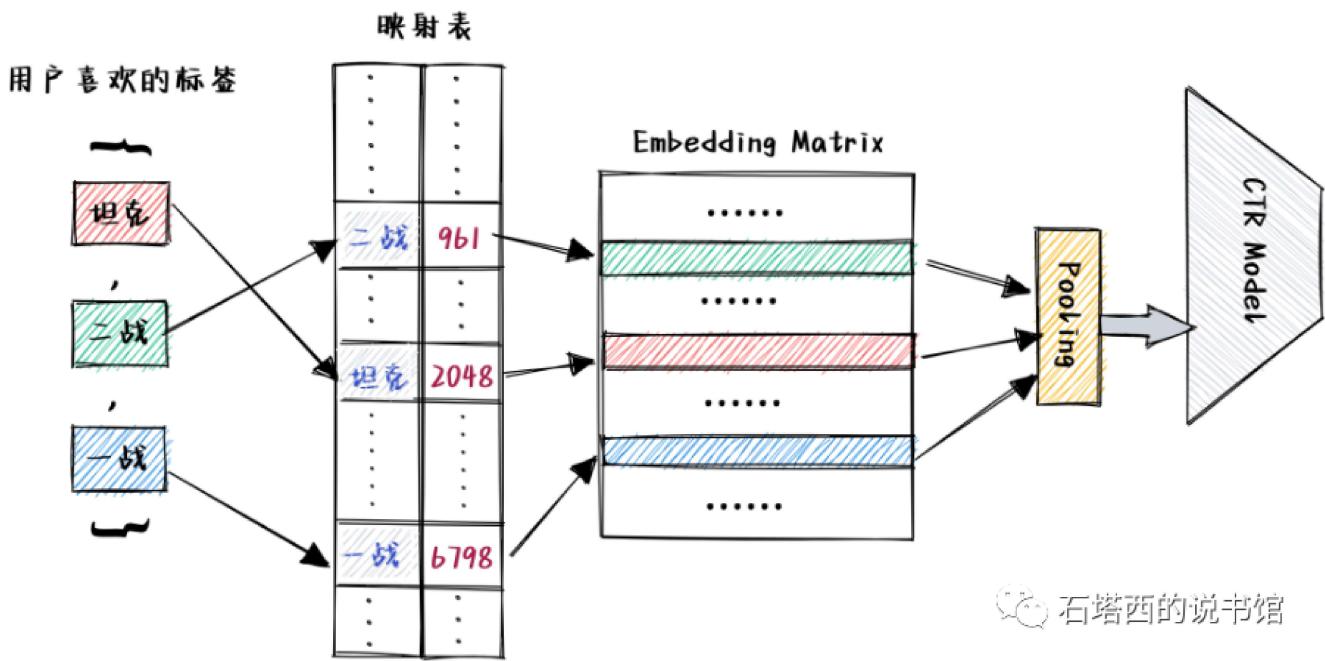


图 2-6 映射类别特征示意

这种方式最大的缺点就是要额外维护一张映射表。就拿标签的映射表来说，随着时间推移，映射表要剔除老旧标签，添加新出现的标签，而且还要保证维护前后，相同的标签要能够映射得到相同的Embedding向量。其实标签还相对稳定，维护起来还比较容易，而对于User ID、Item ID、交叉特征这些变化频繁的类别特征，维护映射表就成为极其繁重的负担。所以，映射表模式用在小型推荐系统中还可以，在大型推荐系统中就要被下面介绍的“特征哈希”模式所取代。

2.4.4 特征哈希

为了规避维护映射表的麻烦，大型推荐模型通常采用“特征哈希”（Feature Hashing，又称Hash Trick）的方法来映射类别特征，如图 2-7所示。

- Feature Hashing负责将输入的字符串映射成一个 $[0, N)$ 之间的整数， N 是Embedding矩阵的总行数，映射得到的整数代表该类别特征的Embedding在Embedding矩阵中的行号。
- Feature Hashing可以简单理解为，先计算输入的字符串的哈希值，再拿哈希值对Embedding矩阵行数 N 取余数。当然实际实现[11]要更复杂一些，以减少发生“哈希冲突”（Hash Collision）的可能性。
- 注意在图2-7中，只要Embedding的长度相同，若干Field可以共享一个Feature Hash模块与背后的Embedding矩阵。相比于让各个Field拥有独立的Embedding矩阵，这种共享方式对空间的利用率更高，是大型推荐模型的主流作法。

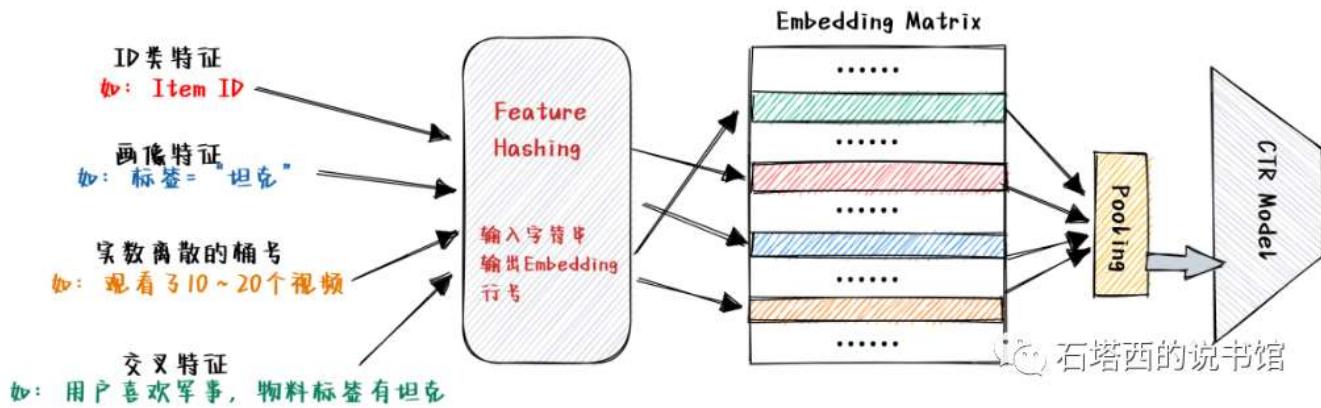


图 2-7 特征哈希示意

细心的读者可能还是会担心发生"哈希冲突"的问题，也就是Feature Hashing将两个不同的特征映射到 Embedding矩阵的同一行，从而这两个矩阵在训练与预测时被混淆，相互干扰。这种情况并不可能完全避免，但是问题也不大。

首先，哈希函数本身能够产生比较均匀的分布，冲突的可能性不大，冲突主要发生在第二步将哈希值压缩到 $[0, N)$ 的时候。因此，只要我们将 N 选得大一些，就能够将发生冲突的概率控制住。

其次，由于推荐系统的特征空间是超级稀疏的，因此Feature Hashing造成的冲突是偶发的，而且影响有限。

- 如果"哈希冲突"发生在两个罕见特征之间，毕竟影响的样本有限，冲突也就冲突了。
- 如果"哈希冲突"发生在一个常见特征与一个罕见特征之间，那个共享的特征Embedding大多数时间是被常见特征训练的，偶尔会被罕见特征带偏一下，但是也不会偏离太多。有点类似Dropout那种随机干扰，反而能够增强模型的鲁棒性。
- 如果"哈希冲突"发生在两个常见特征之间，一来这种可能性极低，二来只要我们加强对冲突指标的监控，这种冲突很容易被发现，而且只要我们在其中一个特征的字符串中添加一些前后缀，冲突就能解决。

总之，Feature Hashing简单易行，可扩展性好，尽管有发生冲突导致特征混淆的可能，但是这种可能性不大且负面影响可控，因此仍然是互联网大厂映射类别特征的标准手段。

2.5 本章小结

本章介绍推荐系统中的特征工程。

首先，作者在2.1节批判了“深度学习使特征工程过时”的错误论调，帮助读者树立正确意识，认清特征工程的重要性。在数据科学的所有领域，“垃圾进，垃圾出”（Garbage in, Garbage out）的理念永不过时。

接下来在2.2节讨论了应该为推荐系统提取哪些特征，从物料特征、用户特征、交叉特征、偏差特征这4个方面讨论了提取特征的一般方法。特别是在2.2.2节为读者梳理了提取用户画像的框架，确保提取出的特征不重不漏。

提取出来的特征，不能直接扔进模型，还需要清洗加工后，特征中的信息才能更好地为模型所吸收。2.3节从缺失值、标准化、平滑与消偏、分桶离散化这4个方面讨论了对数值特征进行清洗、预处理的方法。

推荐算法作为一类特殊的机器学习算法，特点之一就是它的特征空间主要由高维、稀疏的类别特征构成，类别特征是推荐算法的一等公民，享受VIP服务。充分认识这一特点至关重要，它是理解推荐算法的基础。2.4节重点介绍了如何将类别特征映射成数字喂进模型，其中“特征哈希”因为简单易行，可扩展性优异，是互联网大厂处理类别特征时的标准操作。

2.6 本章参考文献

- [1] WANG R, SHIVANNA R, CHENG D Z, 等. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems[J/OL]. 2020[2022-03-12]. <https://arxiv.org/abs/2008.13535v2>. DOI:10.1145/3442381.3450078.
- [2] ZHOU G, SONG C, ZHU X, 等. Deep Interest Network for Click-Through Rate Prediction[J/OL]. arXiv:1706.06978 [cs, stat], 2018[2022-04-23]. <http://arxiv.org/abs/1706.06978>.
- [3] QI P, ZHU X, ZHOU G, 等. Search-based User Interest Modeling with Lifelong Sequential Behavior Data for Click-Through Rate Prediction[J/OL]. arXiv:2006.05639 [cs, stat], 2020[2022-04-04]. <http://arxiv.org/abs/2006.05639>.
- [4] sklearn.preprocessing.OneHotEncoder[CP/OL]. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html>.
- [5] DEVLIN J, CHANG M W, LEE K, 等. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[M/OL]. arXiv, 2019[2023-01-16]. <http://arxiv.org/abs/1810.04805>. DOI:10.48550/arXiv.1810.04805.
- [6] Convolutional neural network[Z/OL]. https://en.wikipedia.org/wiki/Convolutional_neural_network.

[7] COVINGTON P, ADAMS J, SARGIN E. Deep Neural Networks for YouTube

Recommendations[C]//Proceedings of the 10th ACM Conference on Recommender Systems. New York, NY, USA, 2016.

[8] sklearn.tree.DecisionTreeClassifier[CP/OL]. <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>.

[9] MCMAHAN H B, HOLT G, SCULLEY D, 等. Ad click prediction: a view from the trenches[C/OL]//Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. Chicago Illinois USA: ACM, 2013: 1222-1230[2022-11-11]. <https://dl.acm.org/doi/10.1145/2487575.2488200>. DOI:10.1145/2487575.2488200.

[10] RENDLE S. Factorization Machines[C/OL]//2010 IEEE International Conference on Data Mining. 2010: 995-1000. DOI:10.1109/ICDM.2010.127.

[11] WEINBERGER K, DASGUPTA A, ATTENBERG J, 等. Feature Hashing for Large Scale Multitask Learning[J/OL]. 2009[2023-01-30]. <https://arxiv.org/abs/0902.2206v5>. DOI:10.48550/arXiv.0902.2206.