推荐算法的"五环之歌"

Original 石塔西 推荐道 2020-12-14

收录于话题

#深度学习 8 #推荐算法 11 #Embedding 2 #搜索算法 1

前言

研究推荐算法的一大痛点是什么？不是文章太少，而是文章太多，信息爆炸。每年KDD, SIGIR, CIKM上有那么多中外的王婆一起卖瓜，各种各样的NN、FM、Attention满天飞，其中不乏实打实的干货，更不缺乏湿漉漉的灌水文，让人不知道哪个方法才是解决自己问题的灵丹妙药（当然抱着找银弹的想法来读论文，也是too young, too naive）。

造成这种“永远追新、无所适从”的原因是，有些同学孤立地读论文，结果只能是一叶障目，只见树木，不见森林。正确的姿势应该是，**梳理一门学问的脉络**，然后在读论文的时候，就根据这个脉络分门别类，比如某篇文章到底是在哪个分支上进行了改进。等日后，你遇到实际问题，先拆解问题，再到问题涵盖的分支上去寻找合适的解决方案。只有这样，才能真正将各篇论文中的观点**融汇贯通**，读的论文越多，对学问掌握得越清晰，而不是“狗熊掰棒子”，读得越多反而越糊涂。

本文介绍我自己为“推荐算法”梳理出来的脉络。我把推荐算法总结成5个关键词，戏称"五环"。当然这种梳理方法，只是我的一家之言，或许有其他的角度。“横看成岭侧成峰”，只要能够构建出属于你自己的、清晰的知识体系，就没有高低对错之分。

第1环：记忆与扩展

记忆

我们希望推荐系统记住什么？能够记住的肯定是那些常见、高频的模式。举个简单的例子：

- 到了春节，来了中国人，电商网站给他推饺子，大概率能够成交
- 到了感恩节，来了美国人，电商网站给他推火鸡，大概率也能成交

为什么？因为<春节，中国人，饺子>的模式、<感恩节、美国人、火鸡>的模式**在训练样本中出现得太多太多了，推荐系统只需要记得住，下次遇到同样的场景，“照方扒抓药”，就能“药到病除”。**

怎么记住？上“评分卡”

Logistic Regression就是一个非常擅于记忆的模型。说是模型，其实就是一个**超大规模的“评分卡”。**

- 一个特征（中国、美国），或特征组合（<春节、中国人、饺子>）占据“推荐评分卡”中的一项。可想而知，一个工业级的推荐LR的评分卡里面，条目会有上亿项。
- 每项（i.e., 特征或特征组合）都对应一个分数
- 这个分数是由LR学习出来的，有正有负，代表对最终目标（比如成交，即label=1）的贡献。比如 `SCORE(<春节，中国人，饺子>)=5` ，代表这种组合非常容易成交；反之 `SCORE(<中国人、鲱鱼罐头>)=-100` ，代表这个组合极不容易成交
  - 简单理解，可以认为在正样本中出现越多的特征（组合）得分越高，反之在负样本中出现越多的特征（组合）得分越低
- 最终给一个<user, context, item>的打分是其命中的评分卡中所有条目的得分总和。比如当一个中国客户来了，预测他对一款“榴莲馅水饺”的购买欲望= `SCORE(<春节、中国`

人、饺子>)+SCORE(<中国人，榴莲>)=5-3.5=1.5 ，即推荐系统猜他有可能会购买，但是欲望并不那么强烈。

LR("评分卡")模型的特点

- LR的特点就是**强于记忆**，只要评分卡足够大（比如几千亿项），它能够记住历史上的发生过的所有模式（i.e., 特征及其组合）。
- 所有的模式，都依赖人工输入。
- LR本身**并不能够发掘出新模式**，它只负责评估各模式的重要性。（通过Cross Entropy Loss + SGD实现）
- LR不发掘新模式，反之它能够通过regularization，能够**剔除一些罕见模式**（比如<中国人，于谦在非洲吃的同款恩希玛>），即避免过拟合，又减少评分卡的规模

LR("评分卡")模型的缺陷

LR**强于记忆，弱于扩展。**还举刚才的例子

- 中国人来了推饺子，美国人来了推火鸡，都效果不错，毕竟LR记性好。
- 但是，当一个中国人来了，你的推荐系统会给他推荐一只火鸡吗？
- 假设是几前年，当时中国人对洋节接受度不高。如果你的推荐系统只有LR，只有记忆功能，答案是：**不会。因为<中国人，火鸡>属于小众模式，在历史样本罕有出现，LR的L1正则直接将<中国人火鸡>打分置0，从而被从评分卡中剔除**

不要小看这个问题，它关乎到企业的生死，也就关系到你老板和你的腰包

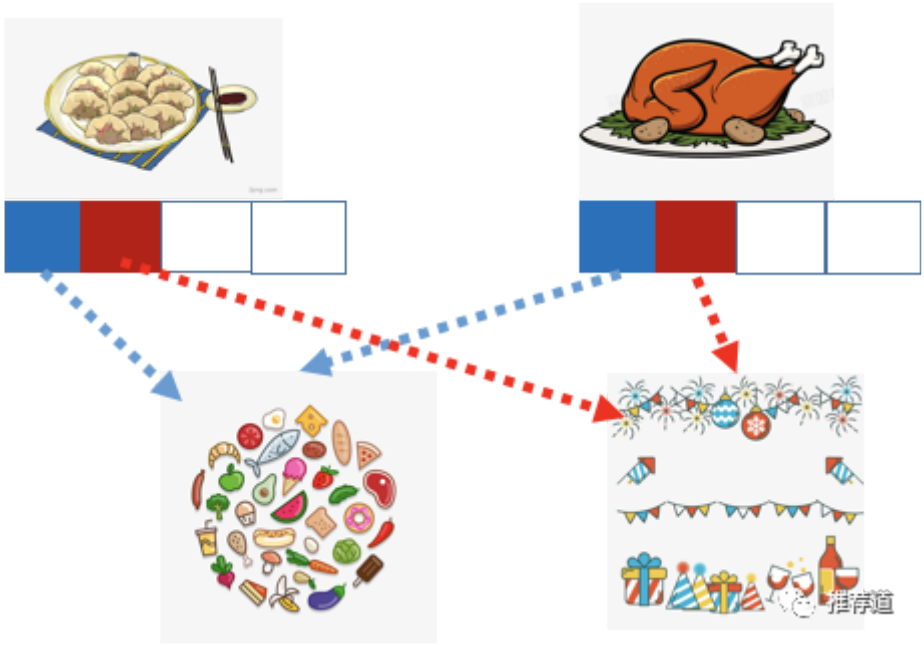
- 记住的肯定是那些常见、高频、大众的模式，能够handle住80%用户的80%的日常需求，但是对小众用户的小众需求呢（某些中国人喜欢开洋荤的需求、于老师的超级粉丝希望和偶像体验相同美食的需求）？无能为力，因为缺乏历史样本的支持，换句话说，**推荐的个性化太弱**。
- 另一个问题是，大众的需求，你能记住，别家电商也能记住。所以你和你的同行，只能在“满足大众需求”的这一片红海里相互厮杀。套用如今最时髦的词，“**内卷**”。

扩展

综上所述，**为了避开“大众推荐”这一片内卷严重的红海，而拥抱“个性化精准推荐”的蓝海**，推荐算法不能只满足于记住“常见、高频”的模式（训练数据中频繁出现的），而必须能够自动挖掘出**“低频、长尾”（训练数据中罕见的）**模式。

如何扩展?

看似神秘，其实就是将粗粒度的概念，拆解成一系列细粒度的特征，从而“**看山非山、看水非水**”。还举饺子、火鸡的例子



- 在之前讲记忆的时候，饺子、火鸡都是独立的概念，看似无什么相似性
- 但是，如果我们根据业务知识，将概念拆解，如上图所示。两个特征向量的第一位表示“是否是食物”，从这个角度来看，饺子、火鸡非常相似；两个特征的第二位是“是否和节日相关”，从这个角度来看，饺子、火鸡也非常相似。
- 喂入LR (评分卡)的除了粗粒度模式，<春节，中国人，饺子> 和 <感恩节，美国人，火鸡>，还有细粒度的模式，比如 <节日，节日相关的食物>。这样一来，<春节，中国人，火鸡> 这样的小众模式，也能够命中评分卡，并获得一个中等分数（因为 <节日，节日相关的食物> 在正负样本中都有出现，所以得分中等）。**相比于原来被L1正则优化掉，小众模式也有了出头之日，获得了曝光的机会。**

这样看来，只要我们喂入算法的，不是粗粒度的概念，而是细粒度的特征向量，即便是LR这样强记忆的算法，也能够具备扩展能力。

有没有自动扩展的方法?

但是，上述方法依赖于人工拆解，也就是所谓的“特征工程”，有两方面的缺点：

- 工作量大，劳神费力
- 人的理解毕竟有局限性。比如饺子、火鸡，拆解到食物、和节日相关这个级别，就已经算是细粒度了吗？还能不能从其他角度继续拆解？

既然人工拆解有困难、受局限，那能不能**让算法自动将概念拆解成特征向量**？如果你能够想到这一步，恭喜你，你一只脚已经迈入了深度学习的大门。你已经悟到了“道”，剩下的只是“技”而已。

第2环：Embedding

我将深度学习形象地总结为“**无中生有**”：

- 当你需要用到一个概念的特征 `v` （比如前面例子里的饺子、火鸡），或者一个函数 `f` （比如阿里Deep Interest Network中的“注意力”函数、CNN中的filter），但是却不知道如何定义它们。
- 没关系，先将 `v` 声明为特征向量，将 `f` 声明为一个小的神经网络，并随机初始化。
- 然后让 `v` 和 `f` ，随着主目标（最终的分类或回归loss），一同被SGD所优化。
- 当主目标被成功优化之后，我们也就获得了重要的副产品，i.e., 有意义的 `v` 和 `f` 。

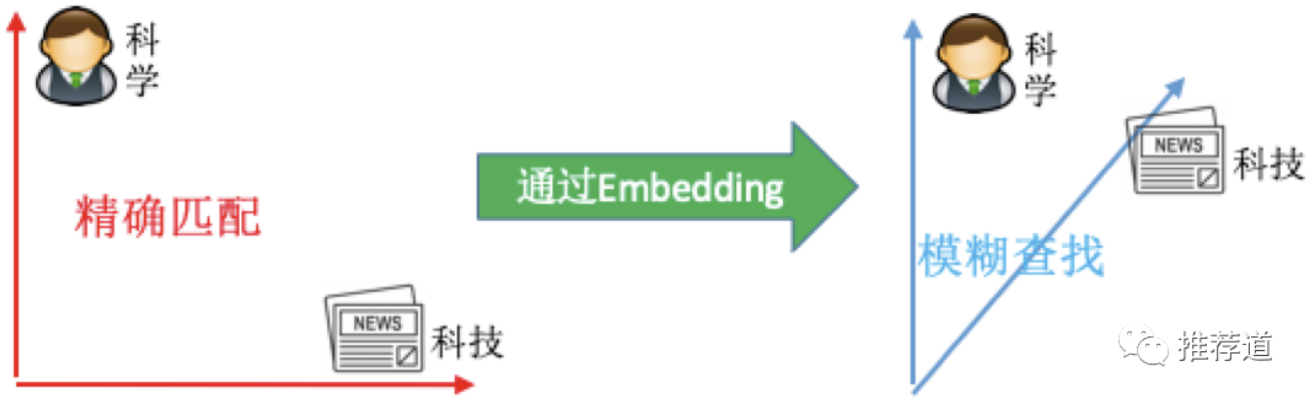
这种“无中生有”的套路，好似“上帝说，要有光，于是便有了光”的神迹。以讹传讹，后来就变成了初学者口中“深度学习不需要特征工程”，给了某些人“我只做深度学习，不做机器学习”的盲目自信。其实这种“**将特征、函数转化为待优化变量**”的思想，并不是深度学习发明的，早在用矩阵分解推荐的“古代”就已经存在了，只不过那时候，它不叫Embedding，而叫“隐向量”。

Embedding变“精确匹配”为“模糊查找”

深度学习对于推荐算法的贡献与提升，其核心就在于Embedding。如前文所述，Embedding是一门自动将概念拆解为特征向量的技术，目标是提升推荐算法的扩展能力，从而能够自动挖掘那些低频、长尾、小众的模式，拥抱“个性化推荐”的“蓝海”。

Embedding到底是如何提升“扩展”能力的？简单来说，**Embedding将推荐算法从“精确匹配”转化为“模糊查找”，从而能够“举一反三”。**

比如在使用倒排索引的召回中，是无法给一个喜欢“科学”的用户，推出一篇带“科技”标签的文章的（不考虑近义词扩展），因为“科学”与“科技”是两个完全独立的词。但是经过Embedding，我们发现“科学”与“科技”两个向量，并不是正交的，而是有很小的夹角。设想一个极其简化的场景，用户向量就用“科学”向量来表示，文章的向量只用其标签的向量来表示，那么用“科学”向量在所有标签向量里做Top-K近邻搜索，一篇带“科技”标签的文章就有机会呈现在用户眼前，从而破除之前“只能精确匹配‘科学’标签”带来的“**信息茧房**”



再回到原来饺子、火鸡的例子里，借助Embedding，算法能够自动学习到火鸡与饺子的相似性，从而给<中国人，火鸡>的组合打一个不低的分数，从而能更好地给那些喜欢过洋节的中国人提供更好的个性化服务。

第3环：高维、稀疏的类别特征

和机器学习的其他领域一样，推荐算法中所使用的特征主要分为两大类：

- 实数型特征：比如用户在过去1小时、6小时、1天之内点击的文章数

- 类别特征：比如文章的tag(e.g., 二战、德国、坦克)，或者更细粒度的ID（e.g., UserId, DocId, AuthorId, ……）

这两类特征中，后者才是推荐算法的“**一等公民**”，按郭老师的话说，就是VIP中P，需要我们这群打工人小心伺候着。

"类别特征"更受欢迎

说类别特征是“一等公民”，一是因为它们更受欢迎，在推荐算法中无处不在：

- 推荐算法的基础是画像。无论是物料画像还是用户画像，都是高维、稀疏、离散的。
  - 比如以最常见的标签为例，文章标签(e.g., 二战、德国、坦克)是物料画像的一部分，用户过去1小时、6小时、1天点击文章所携带的标签是不同时间粒度的用户画像的一部分
  - 高维：一个内容推荐系统中，有几万个标签是小意思
  - 稀疏：尽管系统中有几万个标签，但是具体到某篇文章，某个用户，其携带的标签最多几十个而已。
- 现实场景中，“目标~特征”之间鲜有线性关系
  - 比如，在电商场景下，客户年龄对于其购买欲望的影响肯定不是线性的，而是各个不同年龄段（少年、青年、中年、老年）对购买欲望的影响因子截然不同
  - 所以，**即便是实数特征，我们也经常将其分桶，离散化成类别特征**
  - 比如，在我们的实践中，不是将“ `用户过去1小时观看的视频数` ”当特征，然后其数值是 `3` 。因为这样一来，这个特征只能有一个影响因子（权重），显然无法兼顾"用户无论过去1小时看了3个"和"30个视频"这两类情况（前者可能因为用户喜欢看长视频，后者可能因为用户喜欢看短视频）。
  - 而是拿“ `用户过去1小时观看小于10个视频` ”当特征，其数值是1。另外，还有“ `用户过去1小时观10~50个视频` ”等其他特征，来应对其他情形。这样才更符合”目标~特征“非线性关系的本质。
- 线上工程实现，更偏爱高维、稀疏、离散的类别特征
  - 稀疏意味着我们可以**排零存储、排零计算**，减少线上开销，保证线上预测的实时性。所以，有时候，我看一些公司的宣传材料，声称其算法的特征空间有几百亿，我就会心一笑。这种数字都是哄哄小白的，高维背后的潜台词一定是稀疏，否则你很难想像几百亿维度的稠密矩阵运算具备线上实时预测的实战价值。

- 举刚才的例子，为什么不用“特征是 用户过去1小时观看的视频数 ，数值是 3 ”这个方案，而是采用“特征是 用户过去1小时观看小于10个视频 ，数值是 1 ”的方案？除了为体现“目标~特征”之间的非线性关系，还有一个重要原因就是后者的计算开销更小。以LR为例，$logit = \sum w_i x_i$，如果将所有实数特征都离散化，那么x_i只能是0或1，则LR在线上预测时简化为$\sum w_i$，即**找到非零特征对应的权重并累加，避免了乘法运算**，计算速度更快。

"类别特征"享受VIP服务

说类别特征是“一等公民”，二是因为推荐系统中的很多技术都是为了更好地服务这些VIP而专门设计的

- 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”这一个特征，即可能反映出用户经济实力不强，消费能力有限，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”一个特征，推荐算法还看不懂用户。再加一个特征，比如“用户年龄在20~30之间、工作是程度员”，推荐算法就明白了，“格子衬衫”或许是一个不错的选择。
- 前面已经说了，类别特征的维度特别高，几万个tag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。要存储这么多特征的权重和embedding向量，也是一笔不小的开销。
  - 所以常见的应对策略是通过hashing trick限制最大特征数，可能会因为hash collision带来一些损失，但是在实践中影响并不是太大。
  - 如果特征及其组合已经大到单机容纳不下，**Parameter Server**这样的架构应运而生。Parameter Server也正是利用了推荐、搜索中特征空间超级稀疏这一特点，从而在worker与server同步状态时，**无须同步上亿级别的整个特征空间，而只需要同步batch中所覆盖的极少数特征的状态**，通信开销大大降低。
- 类别特征本来就是稀疏的，“实数特征离散化”和“多特征交叉”使特征空间更加稀疏，而稀疏导致罕见特征（组合）受训机会降低。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，DIN中还为每维特征自适应地调节正则系数

- 普通LR只能拿x_i, x_j都不为0的样本才能训练<x_i, x_j>组合特征的系数w_{ij}，FM借助矩阵分解的思想巧妙解决了这一难题，使得只有$x_i \neq 0$与$x_j \neq 0$的样本也能够参与训练w_{ij}，对组合特征的训练更加充分

第4环：特征交叉

刚才在介绍第3环时已经说了，单个特征的表达能力太弱，所以需要交叉多个特征来增强模型的表达能力。

- 一阶手工交叉：LR。你没看错，一阶也可以交叉。FM之类的高阶自动交叉往往指的是多个特征的共现，而我们可以在预处理阶段计算一些**统计意义上的交叉**，比如用户喜欢的tag与物料所携带tag之间的重合度，然后将这些统计意义上的user/item交叉喂入LR，实践证明对模型效果提升明显。
- 自动二阶交叉：FM。
- 高阶交叉：DNN
- 混合交叉：Wide & Deep。回到第一环介绍的推荐算法的两大永恒主题，**Wide侧其实就是一个LR，负责记忆；Deep侧先经过Embedding，再输入DNN，负责扩展。**

其实目前主流的基于深度学习的排序算法都衍生自Wide & Deep，比如DeepFM或DCN。

- **都有一个浅层模型负责记忆，再有DNN进行高阶交叉，负责扩展。**
- **这也是我对DCN不是非常感冒的原因。**因为扩展功能已经由DNN负责了，所以**另一个模块的任务只负责记忆**，所以浅层模型如LR或FM足矣，DCN中crossing layer声称的“任意阶交叉”完全没有实现的必要。而且**浅层模型必须简单，好起到了类似“正则”的作用，防止DNN过分扩展。**因此在我看来，浅层模型实现超过3层的交叉，完全是不务正业。

第5环：Field & Pooling

什么是Field?

在不同的文章中，有不同的叫法，有的叫Field，有的叫Slot，还有的叫Feature Group，但是含义是相同的，都是若干关联特征的集合。

- 举个App的例子，用户安装、启动、卸载的App是三个Field；微信、支付宝、抖音、快手等都是Feature；三个Field共享一份App列表，可以说共享一份Vocabulary。
- 举用户历史的例子，“用户观看历史”是Field，看过的每个视频的DocId是Feature

原来我们的LR、FM都是只有Feature的概念，不涉及Field，不也干得好好的，怎么现在凭空多出来一个Field的概念？这还是与推荐系统“高维、稀疏的特征空间”这一特点分不开：

- 为了增强推荐算法的扩展性，我们需要将类别特征先进行Embedding，再接入DNN，进行高阶特征交叉。但是怎么接入DNN，变成了一个问题。
- 推荐算法的特征空间有上亿级别，每维特征再embedding成一个向量。如果将这些向量拼接起来接入DNN，DNN的输入层恐怕就要上十亿、百亿的规模，对于存储、计算都会造成不可想像的压力。

所以正确的姿势是，

- 将相关Feature组织成Field，**同一个Field的Feature Embedding需要Pooling成一个向量**，即Field Embedding
- 多个Field Embedding再拼接起来，喂入DNN
- 因为Field的数目要少得多（按我的经验，少则几十，多则几百），DNN的输入层的规模大大降低，连带整个DNN的参数数量也大大减少。

怎么Pooling?

刚才说了，Pooling是将一个Field下的多个Feature Embedding压缩成一个向量的过程。而不同论文在压缩方法上也是各有千秋

- 普通的Mean/Max Pooling，代表算法YoutubeNet。在Youtube的召回、排序模型中，是将用户过去看过的视频、搜索过关键词，先经过embedding，再分别取平均，代表用户的观看偏好和搜索偏好
- Neural FM中，让属于同一field的feature embedding两两交叉，完成所谓的Bi-Interaction Pooling，用v_i表示feature embedding，则Field Embedding=$\sum_{i=1}^{n}\sum_{j=i+1}^{n} v_i \odot v_j$

$$= \frac{1}{2}[(\sum_{i=1}^{n} v_i)^2 - \sum_{i=1}^{n} v_i^2]$$

- 有的人认为“普通平均”中信息损失得太厉害了，所以要引入**加权平均**，**而计算权重则是Attention最擅长的**。比如阿里的Deep Intereset Network (DIN)，在将用户过去购买过的item向量pooling成一个向量时，就通过计算candidate item与用户各历史item的attention score充当权重，然后将各历史item embedding加权平均成一个向量，以表达用户的历史购买偏好。这种加权平均的方式使用户的向量表达随不同的candidate item而变化，实现“千物千面”。
- 有一些Field，比如用户购买历史，其中的Feature存在时序关系。阿里的Deep Interest Evolution Network (DIEN)在Pooling时将时序关系也考虑进去。DIEN将用户历史上购买的item喂入一个RNN，则RNN中最后一步的隐层输出，就是能代表整个用户历史的压缩向量，从而完成了Pooling。这也是借鉴了RNN用于文本分类时的经典套路。但是，用户历史未必是等时间间隔的，这也就违反了RNN的使用前提，具体详情见我在《也评Deep Interest Evolution Network》一文中的讨论。

推荐算法的经典套路

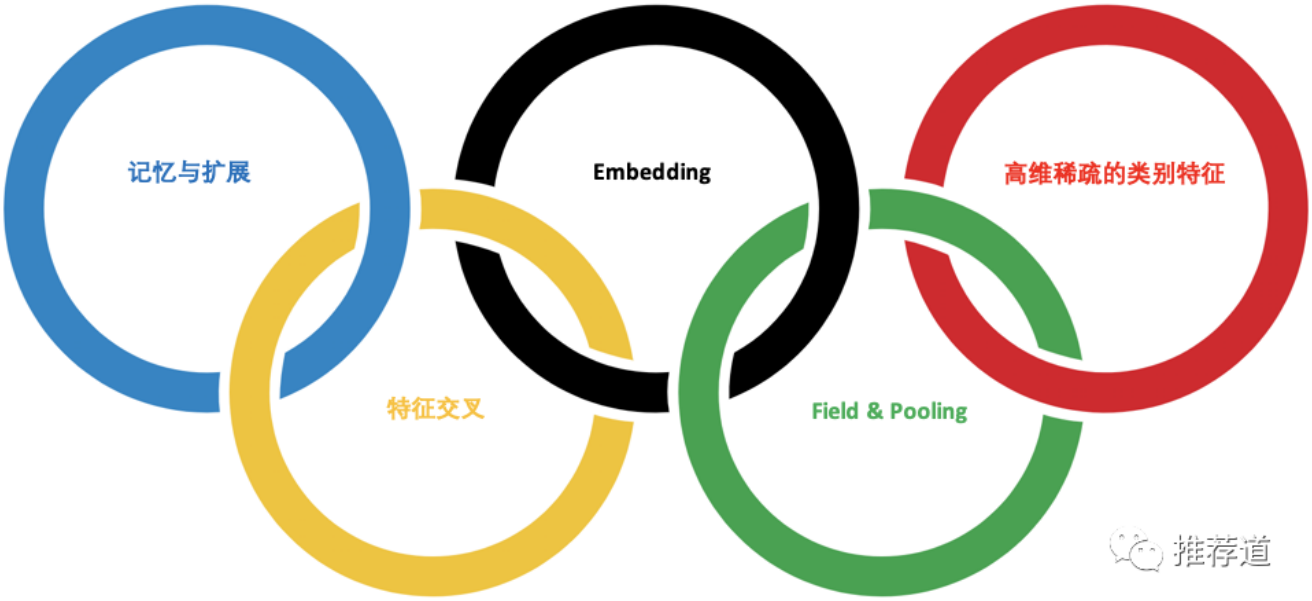
充分理解了上面的5环，你就不难理解推荐算法中的经典套路

- 排序模型一般都衍生自Google的Wide & Deep模型，有一个浅层模型（LR或FM）负责记忆，DNN负责扩展
- 特征一般都采用类别特征。画像、用户历史天然就是高维、稀疏的类别特征。对于实数型特征，比如用户、物料的一些统计指标，在我的实践中，也通过分桶，先离散化成类别特征，再接入模型
- 每个类别特征经过Embedding变成一个向量，以扩展其内涵。
- 属于一个Field的各Feature Embedding需要通过Pooling压缩成一个向量，以减少DNN的规模
- 多个Field Embedding拼接在一起，喂入DNN
- DNN通过多层Fully Connection Layer (FC)完成特征之间的高阶交叉，增强模型的扩展能力。

总结

至此，“推荐5环”梳理完毕。尽管给这5个关键词，起名“五环”有凑梗之嫌，但是也还算贴切，因为它们之间环环相扣

- 记忆与扩展是推荐算法两大经典、永恒的主题。如何实现扩展？靠的是Embedding和特征之间的交叉。
- **Embedding化“精确匹配”为“模糊查找“，大大提升了推荐算法的扩展能力，是”深度学习应用于推荐系统“的基石。**
- **高维、稀疏的类别特征是推荐系统中的一等公民。**为了弥补单个类别特征表达能力弱的问题，需要Embedding扩展其内涵，需要交叉扩展其外延。
- 高维特征空间直接接入DNN，会引发参数规模的膨胀。为解决这一难题，Field & Pooling应运而生。



推荐五环

通过将推荐算法梳理成这5环，再读论文，你会发现某些文章吹嘘的“显著提升、巨大进展”只不过是在某一环上进行的小小改进，而它们在其他环上的所采用的方法可能还有瑕疵，不值得借鉴。

而当你面临实际问题时，可以先将问题的难点拆解到五环中的某些环上，然后从那些环的研究成果中汲取解决问题的灵感，而不是胡子眉毛一把抓，急病乱投医。总之，有了这5环组成的知识体系，你头脑中的推荐算法就变得更加清晰，就可以吃着火锅，唱着歌，在你日常的调参、炼丹生活中谈笑风生，“啊啊，五环，……”