DeepFM在贝壳房源详情页推荐场景的实践

原创 刘敏 李鹏 袁彬 壳算子 2019-12-30

上一篇文章《**wide&deep 在贝壳推荐场景的实践**[1]》中，我们介绍了贝壳首页推荐展位使用的 Wide & Deep 模型，本文向大家介绍贝壳房源详情页推荐展位使用的 DeepFM 模型。

本文主要内容如下：

- DeepFM 模型介绍
- DeepFM 在贝壳房源详情页推荐展位的实践应用

DeepFM 模型介绍

Wide & Deep 是一个 Wide 侧使用 LR，Deep 侧使用 DNN 的联合学习模型（详情见《**wide&deep 在贝壳推荐场景的实践**》）。但在 Wide 侧 LR 一般需要大量的特征工程工作。华为的诺亚方舟实验室提出的 DeepFM 则使用 FM[3] 替换 Wide & Deep 模型中 Wide 部分的 LR，以实现 Wide 的自动特征交叉，降低人工特征工程的工作。
DeepFM 结构如图-1：

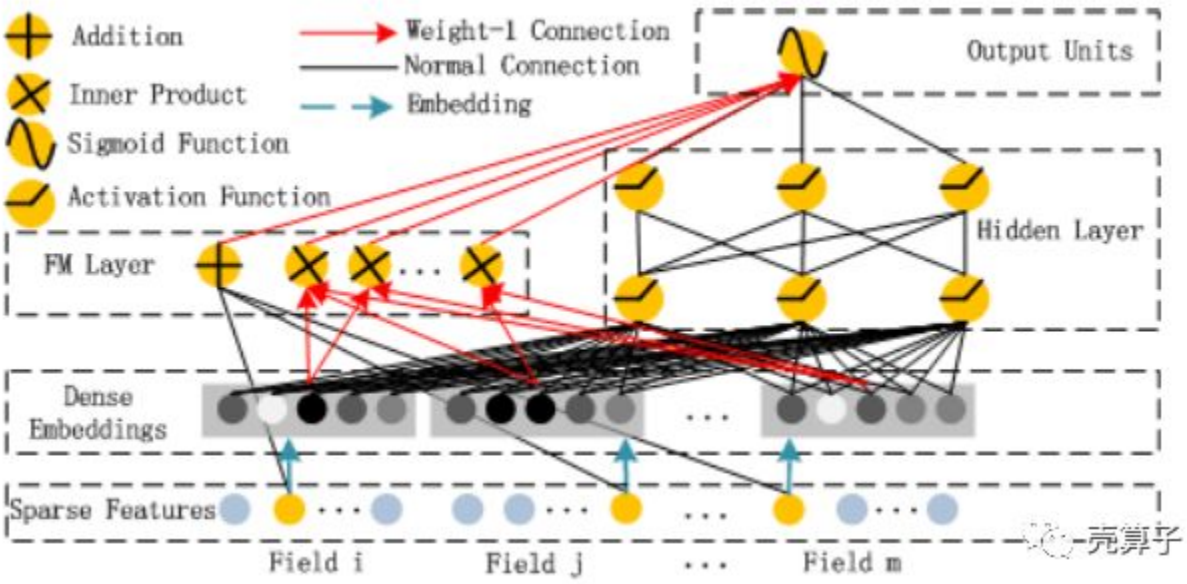

图-1 DeepFM 结构

左侧是一个 FM 结构，右侧是一个 DNN 结构。DeepFM 的预测公式因此为：

$$y = \text{sigmoid}(y_{FM} + y_{DNN})$$

要了解 DeepFM 模型，先要了解 FM 模型。

FM 模型

FM 原理介绍

FM 全称 Factorization Machines（因子分解机），是 2011 年在文献[3]中提出的一种具备特征自动交叉能力的模型。

LR（Logistic Regression，逻辑回归模型) 因其模型简单、可解释性强等优点被广泛应用，但其缺点也十分显著：模型简单，无法学习原生特征与最终目标间的非线性关系，需要人工设计高阶特征，对特征工程极其依赖。

为了解决这一缺点，让 LR 能够学习到原生特征同最终目标间的非线性关系，一个直观的方法就是就是引入原生特征间的自动交叉，比如两个原生特征间的交叉项——二阶交叉项：

$$y(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} v_{ij} x_i x_j$$

n为原生特征的数量，二阶交叉项共引入了 $\frac{n(n-1)}{2}$ 个参数。

特征经过离散化和 OneHot 编码后，特征数量n会十分巨大， 几万至几百万不等（依赖使用到的 ID 类特征数量），二阶交叉项的参数数量能够达到几亿至几千亿。巨大的参数数量，使得模型的 VC 维大，复杂度高，要充分训练如此复杂的模型，需要庞大的样本数量。一般我们认为，样本数量需要至少是参数数量的 10 倍，模型才不容易过拟合，才能学习到真正的知识。

由于交叉特征数量巨大，而样本数量有限，模型很难得到充分训练，必须对二阶交叉项进行参数精简。

如何对二阶交叉项进行精简呢？矩阵分解提供了一个很棒的精简思路：任何一个 n 阶实对称矩阵 $\mathbf{\Lambda}$， 一定存在一个 $n \times k$ 阶实矩阵 $\mathbf{V}$，使得 $\mathbf{\Lambda} = \mathbf{V}\mathbf{V}^{\mathrm{T}}$。我们可以通过约束$k$的大小，来实现精简二阶项系数的目的！

二阶项的参数可以写成矩阵形式：

$$\mathbf{\Lambda}=\begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{bmatrix}, \quad v_{ij}=v_{ji}$$

这是一个实对称矩阵，可以使用一个 $n \times k$ 阶矩阵 $\mathbf{V}$ 来拟合 $\mathbf{\Lambda}=\mathbf{V}\mathbf{V}^{\mathrm{T}}$，其中

$$\mathbf{V}=\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \cdots \\ \mathbf{v}_n \end{bmatrix}, \quad \mathbf{v}_i=\begin{bmatrix} v_i^1 & v_i^2 & \cdots & v_i^k \end{bmatrix}$$

因此，通过使用 $\mathbf{V}$ 来替代 $\mathbf{\Lambda}$ 的方式，使 $v_{ij}=<\mathbf{v}_i,\mathbf{v}_j>$，最终得到 FM 模型：

$$y(x)=w_0+\sum_{i=1}^{n}w_ix_i+\sum_{i=1}^{n}\sum_{j=i+1}^{n}<\mathbf{v}_i,\mathbf{v}_j>x_ix_j$$

其中，$\mathbf{v_i}$ 为 k 维向量（$k<<n$），通过约束 k的大小，可以有效精简二阶交叉项参数数量。一般 k 取值为几十或几百，这样二阶交叉项的参数个数从 n^2 降低到了 kn。

通过约束 k的大小，使得需要训练的参数大大减少。然而，二阶交叉项计算的时间复杂度仍然为 $O(kn^2)$，实际上，FM 的二阶交叉项计算过程可以化简，计算的时间复杂度可以降低到$O(kn)$，文献[3]中对二阶交叉项的计算过程做了推导：

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=i+1}^{n}<\mathbf{v_i},\mathbf{v_j}>x_ix_j&=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}<\mathbf{v_i},\mathbf{v_j}>x_ix_j-\frac{1}{2}\sum_{i=1}^{n}<\mathbf{v_i},\mathbf{v_i}>x_ix_i \\ &=\frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{f=1}^{k}v_{i,f}v_{j,f}x_ix_j-\sum_{i=1}^{n}\sum_{f=1}^{k}v_{i,f}v_{i,f}x_i,x_i\right) \\ &=\frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n}v_{i,f}x_i\right)\left(\sum_{j=1}^{n}v_{j,f}x_j\right)-\sum_{i=1}^{k}v_{i,f}^2x_i^2\right) \\ &=\frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n}v_{i,f}x_i\right)^2-\sum_{i=1}^{k}v_{i,f}^2x_i^2\right)\end{aligned}$$

通过上述推导，FM 二阶交叉项计算的时间复杂度由 $O(kn^2)$ 降低到了 $O(kn)$，变成了线性时间复杂度。

至此，我们得到了一个能够自动实现二阶特征交叉的 FM 模型，该模型的参数数量为 $O(kn)$，计算时间复杂度为$O(kn)$。

DeepFM 中的 FM 结构

在 DeepFM 中，FM 的结构见图-2：

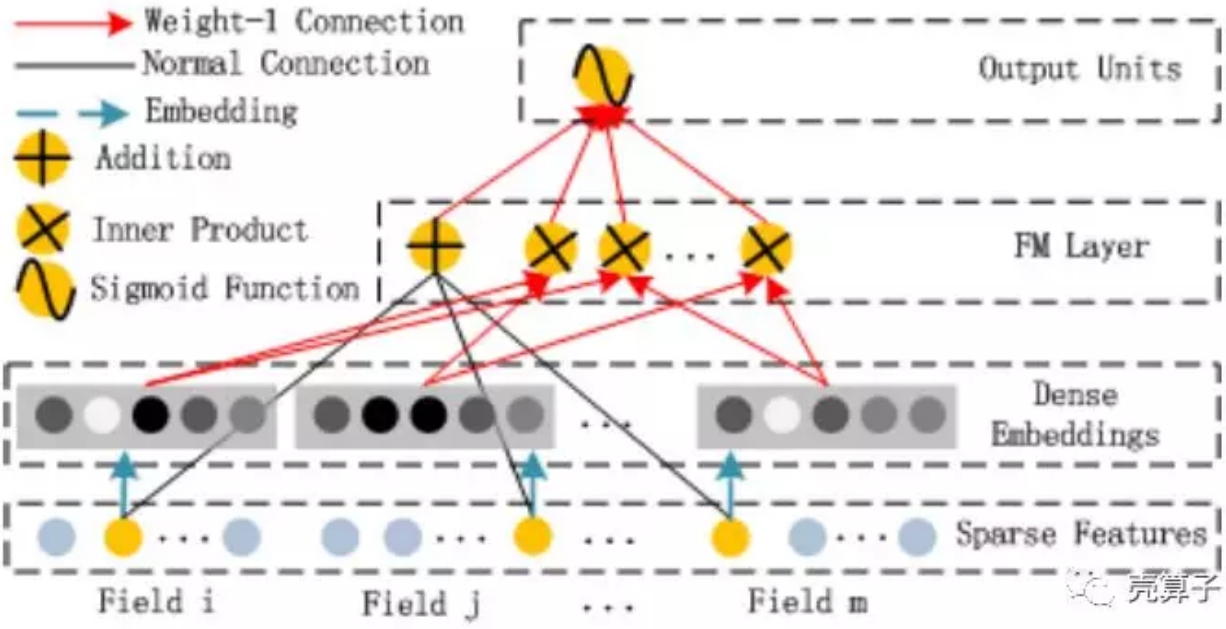

图-2 DeepFM 中的 FM 结构

从下往上看：

第一层 Sparse Features 表示的是做了 OneHot 编码的输入特征。其中简单理解可以认为 **Field i 表示某一维类别特征的 OneHot 编码向量**，其中黄色点代表 1，浅色点代表 0。
图-2 中只画出了类别特征。对于连续特征，我们可以简单理解为只有一个类别的类别特征，其只有黄色点，代表连续特征的值 x_i。

第二层 Dense Embeddings 表示抽取每个 Field 中为 1 的特征 x_i 的 Embedding 向量 $\mathbf{v}_i$（这里 Embedding 向量属于模型参数的一部分，一般采用随机初始化，模型训练时会进行参数更新）。用来计算二阶交叉项的权重 $v_{ij} = <\mathbf{v}_i, \mathbf{v}_j>$。
图-2 中只画出了对类别特征作 Embedding 处理。对于连续特征，只有一个 Embedding 向量，且最终使用的 Embedding 需要与连续特征值 x_i 相乘，即 $\mathbf{v}_i = x_i \cdot \mathbf{v}_i$。

第三层 FM Layer 中，左边的 Addition 符号表示计算 FM 的一阶项 $\sum_{i=1}^{n} w_i x_i$，可以看到只计算 Sparse Features 中为 1 特征；右边的 Inner Product 符号表示计算 FM 的二阶项 $\sum_{i=1}^{n} \sum_{j=i+1}^{n} <\mathbf{v}_i, \mathbf{v}_j> x_i x_j$，可以看到只计算都为 1 的特征的交叉项（其他交叉项 $x_i x_j = 0$ 可以不计算）。

在 DeepFM 中，FM 侧没有单独使用 sigmoid，而是将

$$y_{FM} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} <\mathbf{v}_i, \mathbf{v}_j> x_i x_j$$

与 DNN 输出结果一起输入到 sigmoid。

DNN

DNN 结构这里就不多介绍了，只是简单介绍 DeepFM 中使用的 DNN 结构：

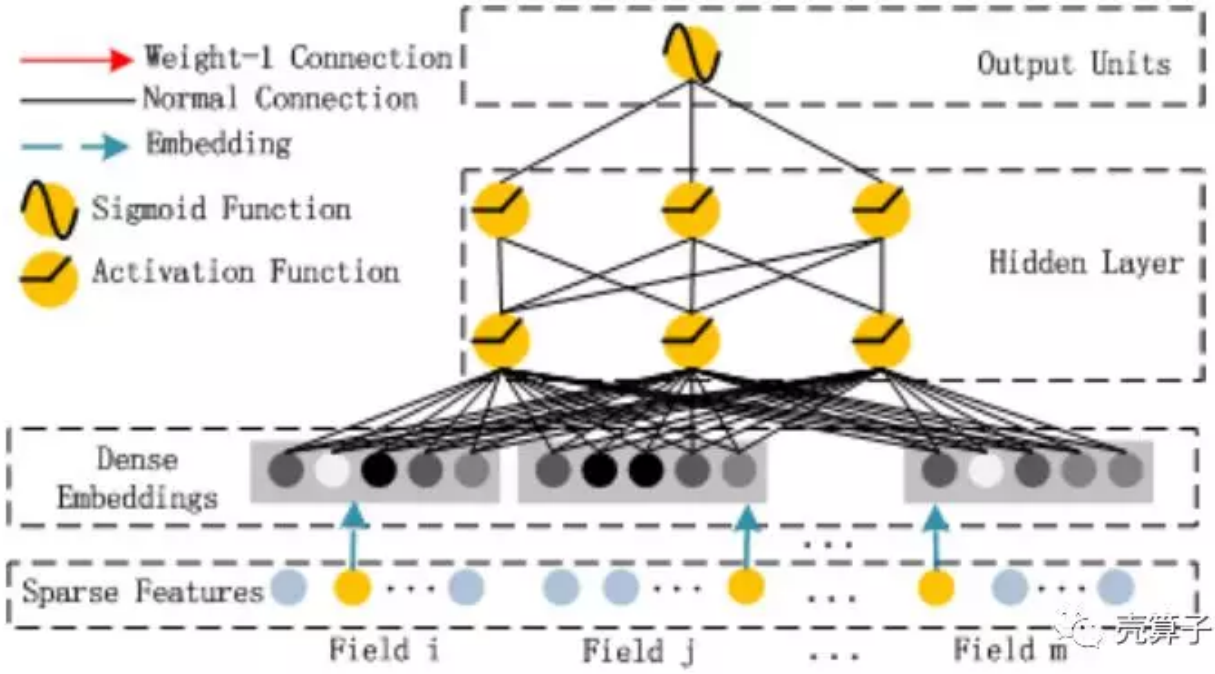

图-3 DeepFM 中的 DNN 结构

从下往上看：

第一层 Sparse Features 与 FM 同。

第二层 Dense Embeddings 表示抽取每个 Field 中为 1 的特征 x_i 的 Embedding 向量 $\mathbf{v}_i$（同 FM 第二层 Dense Embeddings）。然后首位相连得到 DNN 的输入。**这里使用的 $\mathbf{v}_i$ 与 FM 中的是同一个，也就是说 DNN 侧与 FM 侧对 x 对应的向量是共享的。**

第三层 使用了 2 层隐层。

DeepFM 中的 DNN 并没有单独使用 sigmoid，与 FM 输出结果一起输入到 sigmoid 最终得到 DeepFM 的输出：

$$y = \text{sigmoid}(y_{FM} + y_{DNN})$$

以上便是 DeepFM 模型的介绍，下面介绍 DeepFM 在贝壳详情页的应用。

DeepFM 在贝壳房源详情页推荐展位的实践应用

背景介绍

我们将 DeepFM 应用在了贝壳找房 APP 中房源详情介绍页的推荐展位上，见图-4（a）：

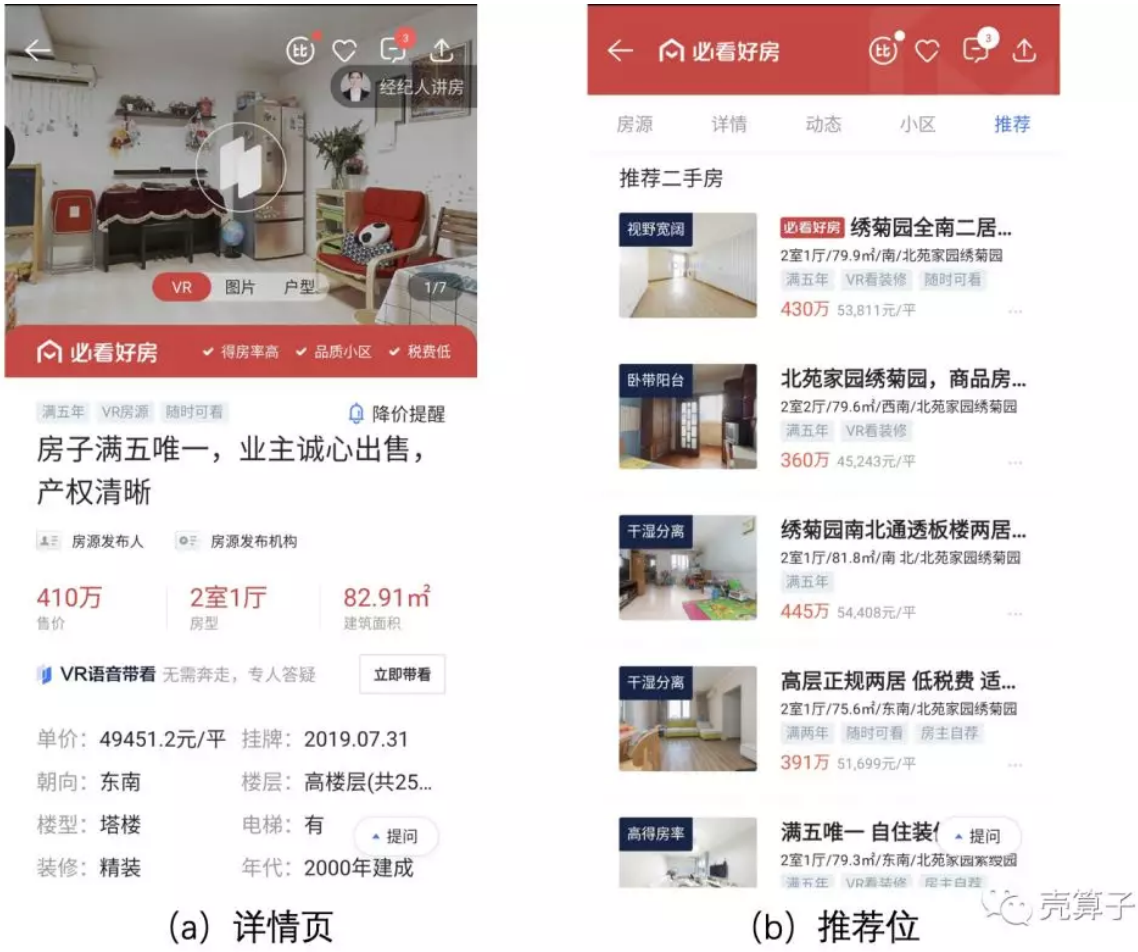

图-4 详情页推荐位介绍

这个推荐场景最显著的特点是有很强的上下文约束。很长一段时间， 这个场景一直使用 LR 模型作为一个 benchmark 的排序模型，2019 年我们开始精准优化这个核心场景的推荐效果。

在前一篇文章中我们提到，首页推荐场景采用了 Wide & Deep，这是因为首页推荐场景积累了很长时间的人工特征，且没有上下文约束，采用 Wide & Deep 可以很好的继承原

有的人工特征，实现无缝迁移。但在详情页推荐场景下，上下文约束很强，需要大量上下文交叉特征，因此，我们采用了 DeepFM，借助 FM 的上下文特征自动交叉减轻我们在特征工程方面的工作。

样本构建

一开始，我们使用的正负样本构建方式：将一次推荐列表中点击的房源（左边列表红框）作为正样本，最后一个点击位之前的曝光未点击的房源作为负样本。如图-5 所示（左边表示有点击行为的推荐列表，右边表示无点击行为的推荐列表）

图-5 样本构建1

且对矛盾样本去重——如果同一个用户在同一个上下文下对同一个推荐房源既有点击行为也有未点击的行为会合并成一条正样本。

但是这种方式构建的训练样本实际效果并不好，我们通过这种方式构建的样本训练的模型在训练集上的 AUC 都只有 0.566 左右，且伴随有过拟合的迹象。

随后我们猜测是因为样本的问题：

- 样本量不够。由于房产领域用户是低频行为，因此我们只有百万级别的训练数据（与 Paper 中动辄上亿的训练数据有很大差距），每个用户的行为数据也比较少；
- 样本存在偏差。我们只是将有点击行为的列表的曝光无点击数据作为负样本，对于那些只有曝光无点击的列表则直接抛弃了，使得模型学习不到这部分负样本的信息。

基于上述两个考虑，我们重新进行样本构造：

图-6 样本构建2

第一，我们将有点击（左边列表红框）的列表中点击之后的曝光未点击样本也采集作为负样本；
第二，我们将无点击的曝光列表（右边）中的第一个曝光未点击的样本也采集作为负样本。

这里有两个小细节需要注意：

1. 为什么都没有加入最后一个曝光的数据？
   这个主要是出于具体的产品情况的考虑，我们详情页中只要房源曝光了一点就算曝光，也就是说用户可能并没有看到这个房源，但是系统也算曝光了，因此我们将最后一个曝光的数据直接去掉了。
2. 为什么无点击的曝光列表中只采用第一个曝光的样本？
   主要是为了保证正负样本相对平衡，避免变成一个样本不平衡学习。我们当前的样本构建方式正负样本比约为 1: 5。如果将无点击的曝光列表的所有数据都作为负样本，则正负样本比为约为 1 : 18，正负样本严重失衡。

特征工程

特征选择

我们分析了我们的场景，详情页推荐的一个基本假设是：用户对详情页的房源感兴趣，推荐相似的房源用户也感兴趣。但是如果只关注上下文房源的特征容易造成推荐的房源都是极度相似的房源，不具备差异性，给用户造成视觉上的疲劳。但如果推荐的房源和上下文的房源差别较大，那么很容易让用户产生不信任。为了平衡这两者，我们决定引入用户画像特征，让模型自己学习个性化和上下文相似的平衡关系。

因此，我们总共选择了如下 6 大类特征：

- 上下文房源特征
- 推荐房源特征
- 用户特征
- 上下文房源与推荐房源匹配特征
- 上下文房源与用户画像匹配特征
- 推荐房源与用户画像匹配特征

每类的具体特征如图-7：

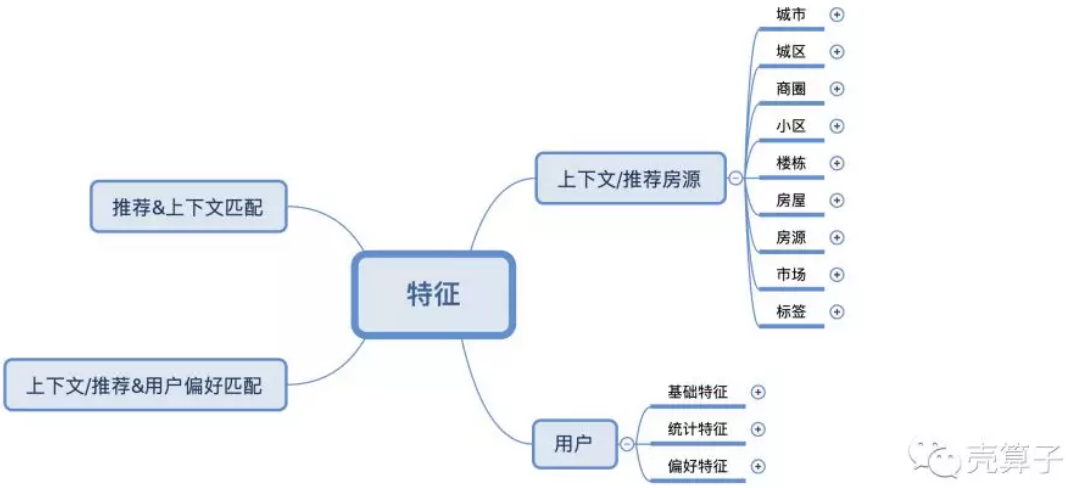

图-7 特征类别

在特征变换中：

- 离散特征
  - Embedding，离散特征都进行 Embedding 编码。
- 连续特征
  - 异常值填充。对于偏好类的特征我们采用 0 填充，对于房源属性特征（如价格，面积等）则使用均值填充；

- 去极端值，为了避免极端值或异常点对最后归一化造成影响（比如存在一个超大的异常点，最后做 MinMax 归一化则会造成这一维的值都会偏小甚至趋向于 0）。因此我们使用分位点的方式将大于 95% 分位点的值替换成 95% 分位点；
- 归一化，归一化我们使用了 MinMax 归一化方式。

此外，还对部分连续特征做了离散化，最终得到了 324 维特征，其中连续特征 143 维，离散特征 181 维。

模型构建

具体的 DeepFM 模型网上有很多优秀的开源项目，这里就不具体介绍。这里主要介绍实施过程中的一些细节。

离线评估

我们将数据划分为训练集、验证集和测试集三部分，其中前 N 天作为训练集、1 天作为验证集，1 天作为测试集。

图-8 训练集，验证集和测试集划分

训练集和验证集中的样本采用前面提到的样本构建方式进行构建。

测试集样本则使用点击为正样本，曝光未点击为负样本。这么构建的原因在于：

- 测试集的正负样本比例要同实际线上样本分布保持一致。原本离线评估使用的 AUC 指标与线上 AB 使用的 CTR 之间就存在 Bias，如果测试集还对测试样本进行选择会增加离线评估的 Bias；
- 而训练集为了解决数据不平衡，样本同实际分布不一致；
- 统一测试集。后续模型迭代都使用统一的测试集，保证模型之间的可比性。

参数调优

- batch_size

batch_size	AUC
64	-5.8%
512	+0.0%
2048	+0.6%

- 网络结构

> 我们使用了 2 层的全连接层。

神经元数	AUC
256, 128	+0%
128, 64	+0.17%

- BN 层

> 加入 BN 层，AUC 相对提升 0.68%。

- Embedding Size

> Embedding Size 从 32 到 64 没有带来明显的效果提升，因此最终使用 32 维。

在线评估

通过线上 AB 实验，**CTR 相对提升 12.65%**。

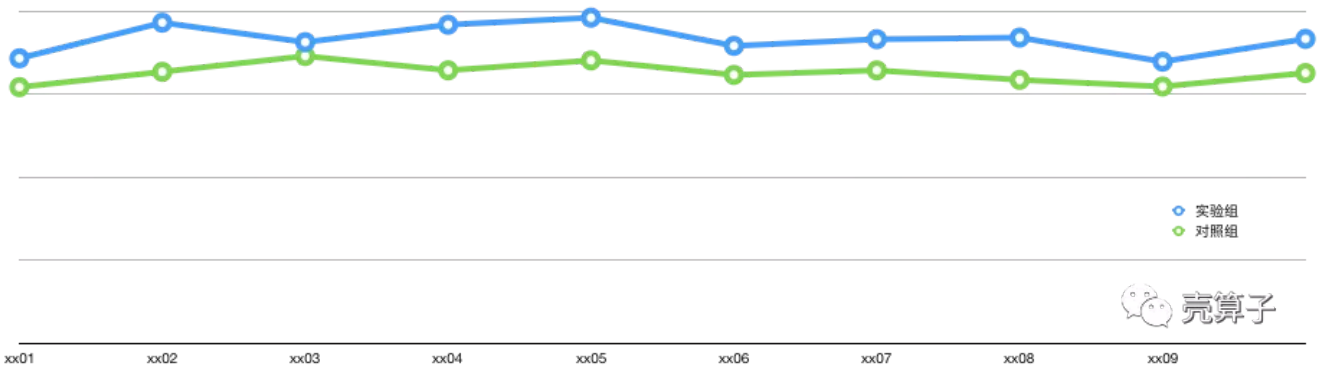

图-9 线上效果

参考资料

[1]wide & deep[1]
[2]deepfm[2]
[3]fm[3]
[4]深度学习在美团搜索广告排序的应用实践[4]
[5]from ranknet to lambdarank to lambdamart: an overview[5]

作者介绍

- 刘敏，2018 年校招加入贝壳找房，主要从事推荐算法相关工作。
- 李鹏，2019 年校招加入贝壳找房，主要从事推荐算法相关工作。
- 袁彬，之前在新浪从事推荐相关工作，现就职于贝壳找房数据智能中心，担任资深算法工程师，专注推荐算法相关工作。