wide&deep论文学习笔记

原创 linger 数据挖掘菜鸟 2018-03-18

在大多数情况下，稀疏输入的大规模回归和分类问题都是通过线性模型和非线性特征来解决。通过特征交叉带来的Memorization（记忆能力）非常有效和可解释，然后generalization（泛化能力）需要更多的特征工程的成本。但是，如果用DNN的话，不需要那么多特征工程，就有足够的泛化能力，对稀疏特征进行embedding后得到的低维稠密特征隐含了一些特征组合。不过，DNN容易导致over-generalize（过度泛化），当user-item交互行为矩阵较为稀疏并且是high-rank的时候模型容易推荐出少部分一些不相关的item。在这篇论文中，作者提出wide&deep的模型，对线性模型和DNN进行jointly trained(共同训练)，取长补短获得较好的记忆能力和泛化能力，对推荐系统带来收益。该方法已在google play应用和验证过，Google play可是一款超过十亿活跃用户和超过百万app的商业化的手机应用市场。通过在线实验的结果发现，相对于wide-only和deep-only的模型，wide&deep模型能够显著地提升app的转化。另外，该模型的实现已经通过tensorflow进行了开源。

大多数的推荐系统的算法架构，都是分为retrieval和ranking两个部分。retrieval（叫检索或者召回）通过机器算法或者人工规则的方法把相关的item检索出来，然后ranking模块对候选item进行精准排序。Ranking问题可以抽象为P(y|x)这么一个数学表达，x为特征，包括user features（用户特征比如年龄）、contextual features（上下文特征比如设备、时间）、impression features（展示特征比如app主页、历史统计量）。通常会将前两者合并叫query，后者叫item，抽象为query-item的输入输出问题。论文主要讨论将wide&deep模型应用在ranking模块。

在工业界中大规模推荐系统的排序模块，广义线性模型比如逻辑回归被广泛应用，因为简单、可扩展、可解释。模型通常是通过one-hot编码的二值化特征输入进行训练。这种线性模型的记忆能力通常是通过特征交叉来表达，比如AND(user_installed_app=netflix, impression_app=pandora")表示的是用户安装过netflix这个app并且当前即将展示的app是pandora。这种特征设计的方式，其实是想通过模型学习到user特征和item特征的关联性，就用这例子来说，假如在实际的数据上，很多安装过netflix的用户喜欢点击pandora这个内推荐的app，那么模型就可以根据这个交叉特征AND(user_installed_app=netflix, impression_app=pandora")学习到内在的关联性，如果模型的输入没有这个交叉特征那么模型就没法学习到。在线性模型的这个前提下，模型的泛化能力需要更多粒度更为宽泛的特征作为输入，比如上面的交叉特征可以往上泛化衍生出一个新的交叉特征AND(user_installed_category=video,impression_category=music)，显然，这需要大量的特征工程，就这个例子来说，需要对netflix和pandora等等app进行分类或者打标签。特征交叉的方式能够使得线性模型具备捕获关联性的能力，但是泛化能力还需要输入的数据有具体的特征，假如query-item的feature-pair在训练数据中没有，模型就没法学习到，比如上面的例子，假如模型的特征设计中没有AND(user_installed_category=video,impression_category=music)这种更为高阶的泛化交叉特征，模型就没法学习到“安装过视频app的用户更喜欢点击推荐的音乐app”这么一个关联性的规律。

对于FM或者DNN这种embedded的模型，一般会对输入的query-item feature pair进行embedding，每个query和每个item都得到一个低维稠密的向量，这种情况下，不需要更多的特征工程就使得模型具备泛化能力。但是，如果输入的query-item的矩阵是稀疏和高秩的话，比如一些user有一些特殊罕见的特征和一些不热门的item，很难学习出有效的低维稠密向量表达。在这种情况下，大部分的query-item pair应该是不会有交互行为出现的，但是基于dense embedding的向量预测，会对所有的query-item产生很多非0的预测，从而导致过度泛化而推荐出一些不相关的item。另一方面，带交叉特征表达的线性模型可以通过少部分的参数记忆住这种特殊的规则，比如某些query-item pair并没有那么相关。

基于以上考虑，论文提出了一种wide&deep的学习框架，共同训练一个线性模型组件(wide组件)和一个神经网络模型组件（deep组件），在一个模型中学习到memorization（记忆能力）和generalization（泛化能力）。

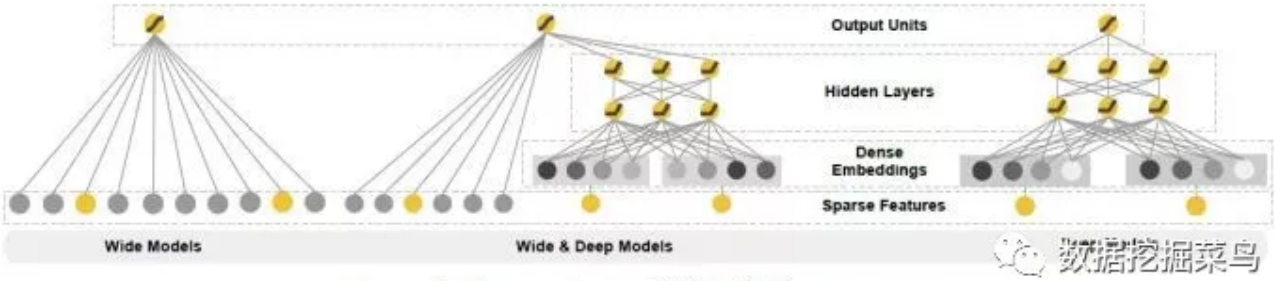

Figure 1: The spectrum of Wide & Deep models.

wide组件和deep组件的输出的对数几率加权求和后再输入到共同的一个logistic loss function。值得注意的是，jointly
training（共同训练）和ensemble（集成学习）的一个主要区别是，在ensemble中，模型之间是独立训练的，互相并不感知对方的存在，他们的预测值在推理阶段结合在一起而不是在训练阶段。而jointly training是同时优化所有参数，在训练阶段就考虑了wide组件和deep组件以及他们之间的加权求和的权重参数。另外在模型的大小也是有区别，对于ensemble，因为训练是互斥的，每个独立的模型为了达到有效的准确性通常导致模型很大（比如更多的特征和更多转换），这样ensemble才能有效。但是相反，对于jointly training，wide组件仅需要补充deep组件的弱点，通常是一小部分交叉特征的变换，而不是一个整个full-size的wide模型。

Jointly training一个wide&deep模型是可以通过从模型输出到wide组件和deep组件反向传播梯度利用mini-batch的随机梯度的方法优化完成。在Google的实验中，对于wide组件是通过带L1正则项的ftrl优化，对于deep组件通过AdaGrad优化完成。模型的预测输出如下，由wide组件和deep组件加起来套进一个sigmoid函数。

$$P(Y=1|\mathbf{x}) = \sigma(\mathbf{w}_{wide}^T[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^T a^{(l_f)} + b)$$

模型在Google Play的实现如下图。值得注意的是，对于连续值的特征，是这么做归一化的，先将原始值通过累计分布函数进行变换P(X <=x),然后用分位数表示，最后用分位数进行归一。训练阶段，输入层将训练数据和词表产生稀疏和稠密的特征以及label。Wide组件由用户安装app和展示app的交叉特征构成。对于deep组件，枚举类的稀疏特征embedding成为一个32维的向量，然后跟连续值特征拼接在一起成为一个1200维的向量。串联后的向量会灌进三层ReLU layer。最后deep组件和wide组件一起输入到logistic。

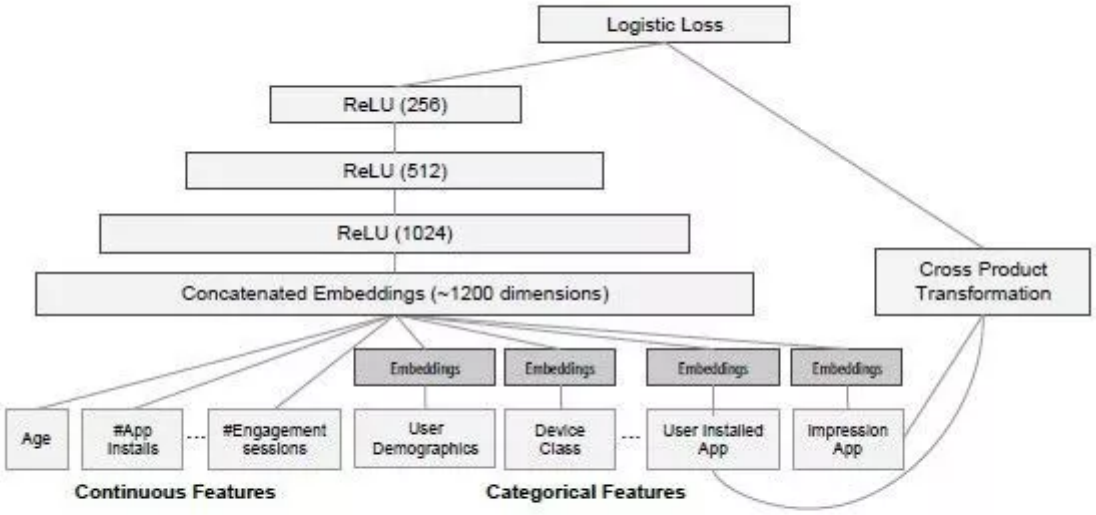

Figure 4: Wide & Deep model structure for apps recommendation.

整个wide&deep模型通过超过500billion的样本训练完成。每一次有新的训练数据，模型都需要重新训练，这样会造成极大的计算成本以及更新延迟。为了解决这个问题，系统实现了热启动的方式，利用上一次的模型参数来初始化embeddings的向量以及线性模型的参数。新的模型训练完，会跟之前模型对比进行测试检验后，再上线。

Google Play的实验分为三个组，以wide-only模型作为对照组，deep模型以及wide&deep模型作为测试组，每组分别1%的user流量。可以看到，相比于对照组，wide&deep模型获得3.9%的app转化率（从推荐列表到app落地页）。结合模型离线auc以及线上的转化率来看，可以看到两点有趣的现象。第一是，deep-only模型离线auc比wide-only低千分之四，但是转化率却提高了2.9%。第二是，wide&deep模型离线auc比wide-only提高仅千分之二，但是线上的转化率却提高了3.9%。对于第二点，论文中有这么一个解释，一个可能的原因是在离线数据中展示和label的数据都是固定的，而在线上系统，调配泛化能力和记忆能力能够展示新挖掘出来的候选item，然后在从新的用户反馈中学习。

Table 1: Offline & online metrics of different models. Online Acquisition Gain is relative to the control.

Model	Offline AUC	Online Acquisition Gain
Wide (control)	0.726	0%
Deep	0.722	+2.9%
Wide & Deep	0.728	数据挖掘菜鸟

本文作者:linger

本文链接：https://zhuanlan.zhihu.com/p/34676942