《Wide & Deep Learning for Recommender System》翻译

概要：

使用非线性特征的广义线性模型（GLM）广泛应用在大规模，输入变量稀疏的回归和分类问题中。其中，通过关于交叉特征的wide模型，对特征间关系的memorization达到了有效和可解释的结果，但同时，也需要很多精力投入在特征工程上。与之对应的，deep模型不需要很多特征工程，通过从系数特征中学习到的低纬稠密特征，可以更好的对没见过的特征组合进行泛化。但是，deep的NN模型很容易过拟合，在user-item的关联很稀疏高维的情况向下，会推荐出不太相关的内容。在本文中，我们提出了Wide & Deep learning，结合了wide线性模型和deep的NN模型，从而获得了二者的memorization和generalization的优点，来应用到推荐系统中。我们在Google Play中实现并评测了这个系统。在线实验表示，与单一的wide模型或者deep模型相比，Wide & Deep可以显著的提高app下载量。我们也在TensorFlow中开源了我们的实现。

1 Introduction

推荐系统可以看做是搜索排序系统，其中，输入是一个用户和上下文信息的集合，输出是一个推荐item的list。给出一个query，推荐系统的任务就是从数据库中找到相关的item，然后根据特性，如点击或购买，来排序。

我们做推荐系统的主要挑战，和一般的搜索排序问题相似，主要是memorization和generalization。memorization可以被大概定义为，从历史数据中学习item或特征的共现信息，探索相关性。generalization，则是基于相关性，来探索在历史数据中没有或者很少出现的新的特征的结合。基于memorization的推荐系统，更倾向于用户已经有过行为的item，更加有局部性和相关性。与memorization相对，泛化性能则倾向于提高推荐item的多样性。本论文中，我们着眼于app推荐，但这个方法应该可以用在更加普遍的推荐系统中。

对于工业界的大规模线上推荐系统和排序系统而言，广义线性模型GLM（如LR等）由于其简单、可扩展和可解释而应用广泛。模型通常使用one-hot编码为binary的稀疏特征来进行训练。比如，XX=A特征一类的。通过对系数特征的cross-product交叉转换，和类似AND这样的逻辑运算，可以有效的获得memorization特性。这也解释了特征pair交叉与目标label之间的关系。generalization通常使用更加粗粒度的特征组合，如A类和B类这样的来获得，但通常需要人工来做特征工程。交叉特征的一个限制是，他们对于没有在query-item中出现过的特征组合表现力不足。

embedding-based模型，如FM或deep NN，可以通过学习低维度稠密特征vector，来泛化一些没有见过的query-item pair特征，并且不需要很多人工特征工程。然而，在query-item的矩阵稀疏高维，用户有特殊的偏好，没太大的扩展空间的情况下，很难学习到有效的低维度表达。这种情况下，在大部分query-item pair之间，应该没有联系，但是，dense embedding会训练得到非0的预测权重，从而过拟合导致推荐出一些没太相关的内容。另一方面，对特征进行cross-product的线性模型可以memorize这些特殊rule，并且所需要的参数很少。

本文中，我们提出Wide & Deep模型框架，通过结合一个线性模型和一个NN模型，来在同一个模型中获取memorization和generalization，见图F1。

本文主要贡献如下：

提出了Wide & Deep学习框架，结合前向NN网络，并嵌入了线性模型，可以用于通用稀疏输入的推荐系统。

在Google Play上实现并且评测了Wide & Deep推荐系统。

在TF上开源了我们的实现。

尽管这个idea很简单，但我们展示了，Wide & Deep框架可以显著的提升app获取率，同时，也满足了训练和服务的效率需求。

2 推荐系统概览

图2显示了app推荐系统的概况。当一个用户访问app store的时候，就会产生一个包含各种用户和上下文的特征的query。推荐系统会返回一个app列表（即，app被展示），用户可以对列表中的app执行操作，如点击或者购买。这些用户的行为，与query和展示一起，记录在log中，作为学习的训练数据。

由于在数据库中有100万多个app，因此，在每个query下，对所有app都出一个分是很难的满足服务延迟要求的（一般要求O(10)毫秒）。因此，第一步是做召回。召回系统会返回一个与query匹配的较短的item的list，一般是机器学习模型和人工定义规则的共同影响的结果。在降低了候选池中item的数量之后，评分系统来对所有item评个分，然后进行排序。这个评分一般含义是P(y|x)，即，给定用户和上下文特征x，label y出现的概率。在本文中，我们主要集中在基于Wide & Deep学习框架的评分系统。

3 Wide & Deep Learning

3.1 Wide部分

Wide部分是一个广义线性模型，形式是：y=wx + b，如图F1（左）所示。y是预测结果，x是d个特征组成的向量，w是模型参数，b是偏置。特征集合中，包含了基础的输入特征和转换后的特征。其中，最重要的转换是cross-product（交叉）转换，定义如下：

（见论文公式）

其中，c\_ki是一个布尔值，如果第i个特征是转换的一部分，就为1，否则为0。

对于二进制的特征，一个交叉转换是（如，AND(性别=女，语言=英语））当且仅当特征（性别=女）和特征（语言=英语）都为1的时候，才为1，否则为0。这样就能抓取到了两个二进制特征的交叉因素，并且为广义线性模型中增加了非线性。

3.2 Deep部分

Deep部分是一个前向反馈NN网络，如图F1（右）所示。对于分类特征，初始输入是特征的字符串（如，语言=英语）。这些稀疏高维特征先转换为低维稠密特征向量，如，embedding vector，一般是O(10)到O(100)维度。这些embedding vector先随机初始化，然后训练模型，降低最终的损失函数的结果。这些低维稠密的embedding vector之后会作为输入进入到NN的隐藏层。其中，每个隐藏层都会做如下计算：

（见式子2）

其中，l是隐藏层数，f是激活函数，经常使用ReLU。a,b和W是第l层的激活，偏执，和模型权重。

3.3 Wide & Deep模型的组合训练（joint training）

Wide部分和Deep部分通过一个加权他们的输出的log odd，一般再用一个普通的LR来做组合训练。注意：组合训练与ensemble是有区别的。在ensemble（集成学习）中，不同模型是独立训练，相互之间不影响的，只有在预测的时候，才把他们的结果集合起来，而不是在训练时候就结合起来。与此相比，组合训练在训练的时候，就把wide和deep部分中的参数和他们结果组合的时候的加权参数都一起考虑进来做优化了。他们的模型大小也不同：对于ensemble来说，由于不同模型是分开训练的，所以每个模型都需要更大一点（如，更多特征和转换）来获取较好的准确率。但组合训练只需要为了弥补deep部分的弱点，来用很少一些特征交叉就可以了，而不是用所有的全量的wide模型。

Wide & Deep模型的组合训练可以通过mini-batch的随即梯度优化，用BP算法对wide和deep部分的模型输出进行同时计算。在实验中，我们对wide部分的优化用了FTRL算法，并使用了L1正则化，对deep部分用了AdaGrad优化。

组合后的模型，如图F1所示（中）。对于一个LR问题，模型预测如下：

（式3）

其中，Y是分类标签，δ（·）是sigmoid函数，X(x)是x转换的交叉特征。b是偏置。w\_wide是wide部分的参数全红，w\_deep是在最终的激活中用到的权重。

4 系统实现

app推荐系统的实现包含三个阶段：数据生成，模型训练和模型服务。见图F3.

4.1 数据生成

在本阶段，一段时间内的用户和app的展示信息被用来生成训练数据。每个样本对应一条展示。标签是app下载：如果展示app被安装则为1，否则为0。

vocabulary，即类别字符串特征转换为数字id特征的映射表，也在本阶段生成。对于所有的出现次数超过一个最低限制次数的字符串特征，系统会用来计算id的特征空间。连续特征会正则化到[0,1]区间，计算CDF的P(X≤x），分割成为n\_q个部分。对于第i个，计算值为：(i-1)/(n\_q - 1)。他们的边界会在数据生成的时候就计算出来。

4.2 模型训练

我们在实验中用到的模型结构如图F4所示。在训练中，输入层使用了训练数据，vocabulary，一起生产了系数和稠密特征。wide部分包含用户安装的app和展示给用户的app的交叉特征。对于deep部分来说，对于每个类别特征，都使用了一个32维的embedding vector来转换为低维稠密特征。我们将所有的embedding vector得到的稠密特征连接起来，输入到一个稠密的vector中到下一层，接近1200维度。之后，向量进入3个ReLU层，最后，进入LR中。

这个Wide & Deep模型训练集合为5000亿个样本。每次有新的训练数据，模型都需要重新训练。然而，每次从头开始算很消耗时间，影响线上模型更新。为了解决这个问题，我们实现了热启动，即，每次用上一个模型的embedding和线性模型的权重来初始化新模型。

在每次上线新模型之前，会dry run一下模型，来确保在线上实时流量中不会出现问题。我们通过经验来验证模型质量，与上一个模型做对比。

4.3 模型服务

在模型训练和验证后，就加载到线上服务器里。对于每次请求，服务器从召回系统获取到一列app候选集，根据用户特征来对每个app进行打分。然后，将app根据评分从高到低排序，按照此排序向用户展示。这个评分是通过在Wide & Deep模型中，前向反馈计算得到的。

为了保证服务延迟在10ms以内，我们使用多线程优化了下，每次并行线程各跑一个小batch，而不是在一个预测评分中跑所有的候选app。

5 实验结果

为了在真实推荐系统中评价Wide & Deep模型的有效性，我们上线了实验并且评测了系统。主要在两个方面：app获取和服务性能。

5.1 app获取

我们上线了3周的线上AB实验。对于控制组，随机选了1%的用户，展示内容结果为之前的高度优化的wide-only的LR模型，其中包含了很多交叉特征。对于实验组的1%的用户，展示的是Wide & Deep模型使用同样数据训练得到的模型推荐的结果。如表1所示，在main landing page上，与控制组相比，Wide & Deep模型的app获取率提高了3.9%。实验结果还与另一个只用了deep模型的1%的用户的结果对比，该deep模型使用了相同的特征和NN结构，而Wide & Deep比它提升了1%。

除了线上实现，我们也在线下holdout数据中测试了AUC。Wide & Deep在线下AUC中只有少许提升，但在线上实验中却提升很多。一个可能的原因是，线下数据中的展示和标签label是固定的，然后，在线上系统中，可以通过模型的memorization和泛化，探索新的推荐item，并且从新用户的反馈中继续学习。

5.2 服务性能

对于大流量商业移动app store来说，高吞吐量和低延迟是一个巨大挑战。我们的推荐系统在峰值时，每秒钟对1千万个app进行评分。单线程中，一个batch来对所有候选集评分耗时31ms。我们使用多线程技术，每个batch降低大小，成功将延迟降低到14ms，如表2所示。

6 相关工作

将包含交叉特征的wide线性模型与包含稠密embedding特诊的深度NN模型结合起来的idea是由之前工作引发的思考，如，FM中，通过将两个变量之间的关系向量化为两个低维embedding vector的dot product。在本文中，我们使用NN模型，通过使用embedding vector而不是dot product来计算高度非线性联系，扩展了模型的容量。

在语言模型中，将RNN与n-gram的最大熵模型结合，通过直接学习输入和输出之间的权重的方法已经被提出来过，用来显著降低RNN的复杂度。在计算机视觉领域，deep residual learning被提出用来降低训练深度模型的难度，通过跳过一层或多层来建立跨层的快捷连接来提高准确度。NN模型和图模型一起结合，被用在图像中的人类姿势识别。在本文中，我们探索了在常见的具有稀疏输入特征的推荐和排序系统中，通过稀疏特征和输出unit之间的直接连接，使用前向神经网络与线性模型进行结合训练。

在推荐系统中，已经有人通过将对内容信息的深度学习和对评分矩阵的CF相结合，探索了协作深度学习。之前也有在对移动app推荐系统的实践，如AppJoy中，针对用户对app使用记录来使用CF。与之前的基于CF的或基于内容的算法不同，我们在app推荐系统上，使用用户和展示信息结合训练了Wide & Deep模型。

7 结论

对于推荐系统而言，memorization和generalization都很重要。Wide线性模型可以通过交叉特征转换来有效的记忆稀疏特征之间的关联，而deep NN模型可以通过低维embedding来泛化出之前没见过的特征之间的关系。我们推出了Wide & Deep模型框架，来结合二者的长处。我们实现了模型，并在Google Play上评测了模型结果。线上实验结果表示，与wide-only和deep-only模型对比，Wide & Deep模型可以显著提升app下载。