《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。