深入理解 Wide&Deep 模型 [1]

leolinuxer

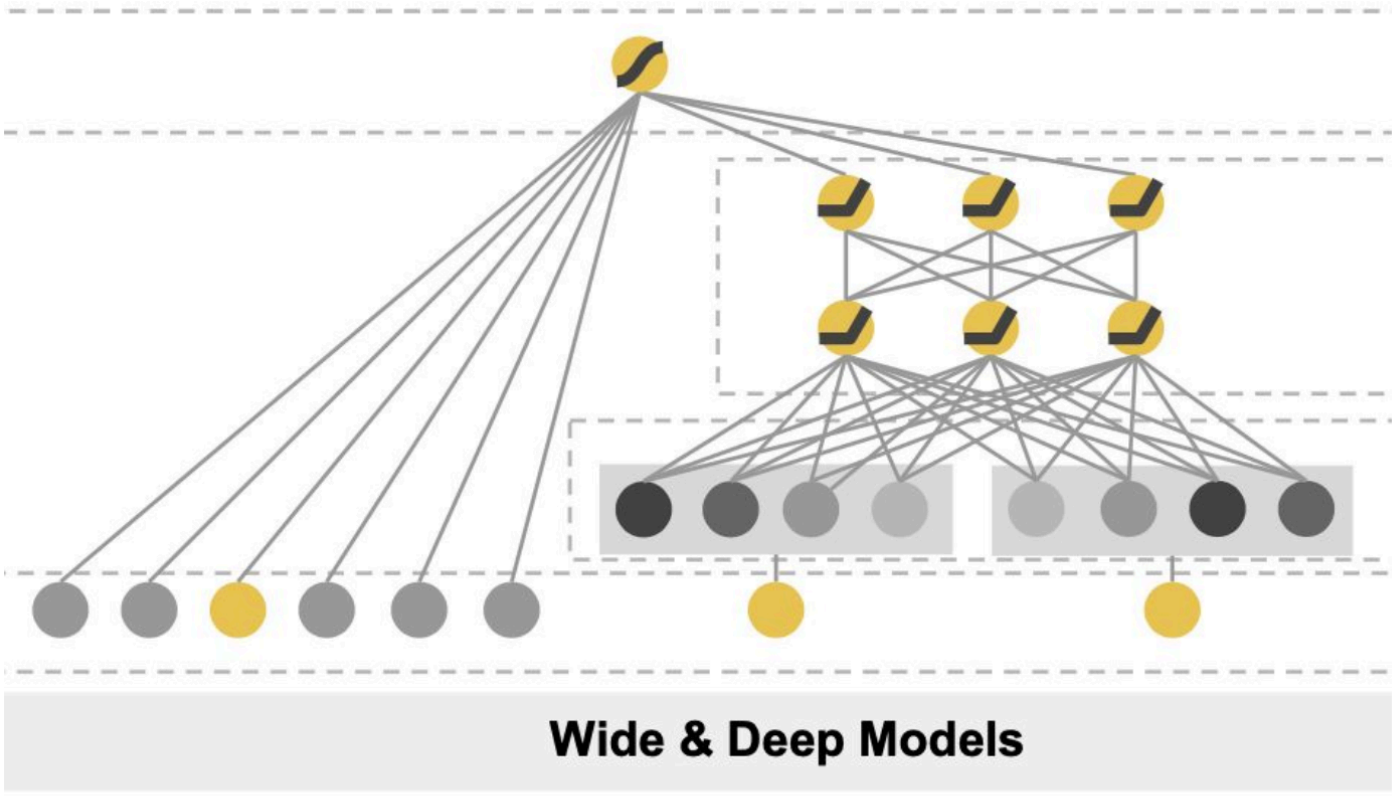
July 16, 2020

Contents

1 Wide&Deep 模型简介

1.1 模型结构

Wide&Deep 由浅层（或单层）的 Wide 部分神经网络和深层的 Deep 部分多层神经网络组成，输出层采用 softmax 或 logistics regression 综合 Wide 和 Deep 部分的输出。



Wide&Deep模型示意图

1.2 模型特点

Wide 部分有利于增强模型的“记忆能力”，Deep 部分有利于增强模型的“泛化能力”。

2 为什么 Wide&Deep 采用了不同的训练方法

为什么在 Google 的 Wide&Deep 模型中，要使用带 L1 正则化项的 FTRL 作为 wide 部分的优化方法，而使用 AdaGrad 作为 deep 部分的优化方法？

论文原文的描述是这样的：

In the experiments, we used Follow- the-regularized-leader (FTRL) algorithm with L1 regularization as the optimizer for the wide part of the model, and AdaGrad for the deep part.

2.1 为什么 Wide 部分要用 L1 FTRL 训练

这里简要介绍一下，可以把 FTRL 当作一个**稀疏性很好，精度又不错的随机梯度下降方法**。由于是随机梯度下降，当然可以做到来一个样本就训练一次，进而实现模型的在线更新。所以在四五年前，大部分公司还是线性模型为主的时代，FTRL 凭借非常好的在线学习能力成为主流。

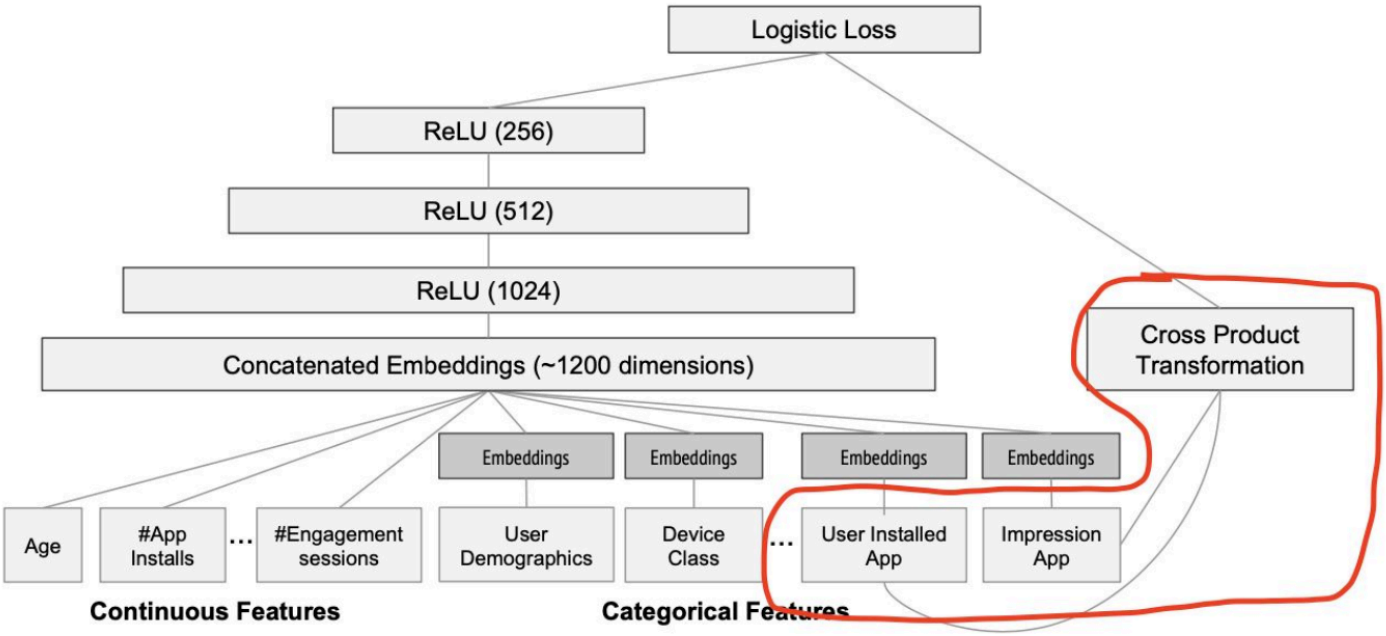
说完了 FTRL，再说 L1 正则化，参加过算法岗面试的同学可能都碰到过那个经典面试题“为什么 L1 正则化比 L2 正则化更容易产生稀疏解?”。问题的答案现在当然已经是显学了，但这里“稀疏”这个

性质又冒出来了。也就是说 FTRL with L1 非常注重模型的稀疏性。这也就是问题的答案，W&D 采用 L1 FTRL 是想让 Wide 部分变得更加稀疏。

再白话一点就是，L1 FTRL 会让 Wide 部分的大部分权重都为 0，我们准备特征的时候就不用准备那么多 0 权重的特征了，这大大压缩了模型权重，也压缩了特征向量的维度。

2.2 Wide 部分的稀疏性为什么这么关键

稀疏性不见得一直是一个好东西，它不管怎样都会让模型的精度有一定的损伤。肯定是特征向量维度过高导致“稀疏性”成为了关键的考量。这就涉及到 Google Wide 部分的特征选取了，到底 Google 选了什么特征需要这么注重稀疏性。我们回到他的业务场景中来。



大家可以看到红圈内的 Wide 部分采用了两个 id 类特征的乘积，这两个 id 类特征是：

- User Installed App；
- Impression App；

这篇文章是 Google 的应用商店团队 Google Play 发表的，我们不难猜测 Google 的工程师使用这个组合特征的意图，他们是想发现当前曝光 app 和用户安装 app 的关联关系，以此来直接影响最终的得分。

但是两个 id 类特征向量进行组合，在维度爆炸的同时，会让原本已经非常稀疏的 multihot 特征向量，变得更加稀疏。正因如此，wide 部分的权重数量其实是海量的。为了不把数量如此之巨的权重都搬到线上进行 model serving，采用 FTRL 过滤掉哪些稀疏特征无疑是非常好的工程经验。

2.3 为什么 Deep 部分不特别考虑稀疏性的问题

大家注意观察可以发现 Deep 部分的输入，要么是 Age，#App Installs 这些数值类特征，要么是已经降维并稠密化的 Embedding 向量，工程师们不会也不敢把过度稀疏的特征向量直接输入到 Deep 网络中。所以 Deep 部分不存在严重的特征稀疏问题，自然可以使用精度更好，更适用于深度学习训练的 AdaGrad 去训练。

2.4 再说回模型的泛化能力和记忆能力

再说回所谓 wide 部分的“记忆能力”。其实大家可以看到，所谓的“记忆能力”，可以简单理解为发现“直接的”、“暴力的”、“显然的”关联规则的能力。比如该问题中，Google W&D 期望在 wide 部分发现这样的规则：

用户安装了应用 A，此时曝光应用 B，用户安装的 B 概率大。

而 Deep 部分就更黑盒一些，它把能想到的所有特征扔进这个黑盒去做函数的拟合，显然这样的过程会“模糊”一些直接的因果关系，泛化成一些间接的，可能的相关性。

从这个角度来说，所谓“泛化能力”和“记忆能力”就更容易被直观的理解了。

3 一些思考

1. 如果 lr 时代特征工程做的很好，迁移到 deep，加 wide 部分收益不会太大，w&d 可能更多的是给出一个简单通用的 lr 到 deep 的迁移框架。

2. 在实际应用时，可以考虑将 wide 和 deep 分开，在 deep 部分做 batch update 保证准确性和充足表达能力，wide 部分做 online learning 保证实效性；特别是对时效性要求高的时间段或场景，deep 的效率跟不上，可以固定住 deep，对 wide 进行 online learning 来增强记忆性

3. 如果直接把 deep 部分的 embedding 输入到 wide 侧可行吗；那就相当于 embedding 层后的 mlp 换成一层 lr 了，本质就是 deep 侧的 dnn。

References

[1] “见微知著，你真的搞懂 google 的 widd deep 模型了吗？.” [Online]. Available: https://zhuanlan.zhihu.com/p/142958834