白话推荐系统(一)：一文看懂Wide & Deep深度推荐开山之作

陈壮实的搬砖日记
熟悉推荐系统，爱好AIGC、OCR，软工硕士

推荐系列源码：

EasyDeepRecommand
github.com/lamctb/EasyDeepRecommand

如果觉得不错的话，帮忙star一下，感谢!

1. 简介

Wide & Deep是由谷歌APP Stroe团队在2016年提出的关于CTR预测✦的经典模型，该模型实现简单，效果却非常好，因而在各大公司中得到了广泛应用，是推荐系统领域的经典模型!

> 原文地址：Wide & Deep Learning for Recommender Systems
> 团队：Google APP Store
> 发表时间：2016年

2 基本概念

2.1 线性特征✦和非线性特征✦

（1）线性特征

线性特征是指特征之间的关系可以通过一条直线来表示的特征。在数学上，如果一个特征空间中的特征可以表示为输入特征的**线性组合**，那么这些特征就是线性特征。

如：人的身高h和体重w两个特征，可以使用线性模型来进行拟合，即： $w = a * h + b$

常用的线性模型有： 线性回归、岭回归、套索回归、逻辑回归和线性 SVM

（2）非线性特征

非线性特征是指特征之间的关系**无法通过一条直线来表示的特征**。在数学上，如果特征空间中的特征无法通过输入特征的线性组合来表示，那么这些特征就是非线性特征。

如：比如对于二维特征空间，特征(x, y)的关系如果是圆形、椭圆形或其他曲线形状，则这些特征就是非线性特征。

常用的非线性模型有： 决策树、随机森林、支持向量机（通过核技巧来解决非线性问题）、神经网络

2.2 稀疏向量✦和稠密向量✦

（1）稀疏向量

稀疏向量是指在向量中大部分元素为零（或接近零）的向量。换句话说，稀疏向量中只有少数几个非零元素，而其他元素都是零。

示例：中文字符的一级字库共3755个字，使用one-hot向量编码，则每个字都会使用一个3755维度的向量表示，假设第1个字是"中"，则”中“的向量就表示为：[1, 0, 0, 0, 0, ... , 0]，除了第一个位置为1，其余位置全为0。这种向量就是一个稀疏向量！

（2）稠密向量

稠密向量是指向量中大部分或所有元素都有非零值的向量。换句话说，稠密向量中的元素数量与向量的维度相当，没有或只有少量的零元素。

示例： 使用Embedding编码的词向量

2.3 模型的记忆能力和泛化能力

（1）记忆能力

模型的记忆力可以理解为items(特征or商品)之间成对出现的一种学习能力，更为专业地说：记忆力是直接学习并利用历史数据中物品和特征的”共现频率“的能力！

特征间的共现频率越高，则这对组合特征与标签结果的关联性就越大，则应该特征这种组合特征的推荐权重。

举例说明：有这样一种情况，安装了netflix APP的用户，如果浏览到了pandora APP，那么这个用户安装pandora APP的比例是1/10，此时记忆力强的模型就应该提高安装了netflix APP用户的pandora APP的曝光率！

记忆力强的网络往往层数不多，因为深层网络，特征就会被深层处理，不断与其他特征进行交叉，从而导致深层网络的记忆力反而没有简单模型的记忆力强！

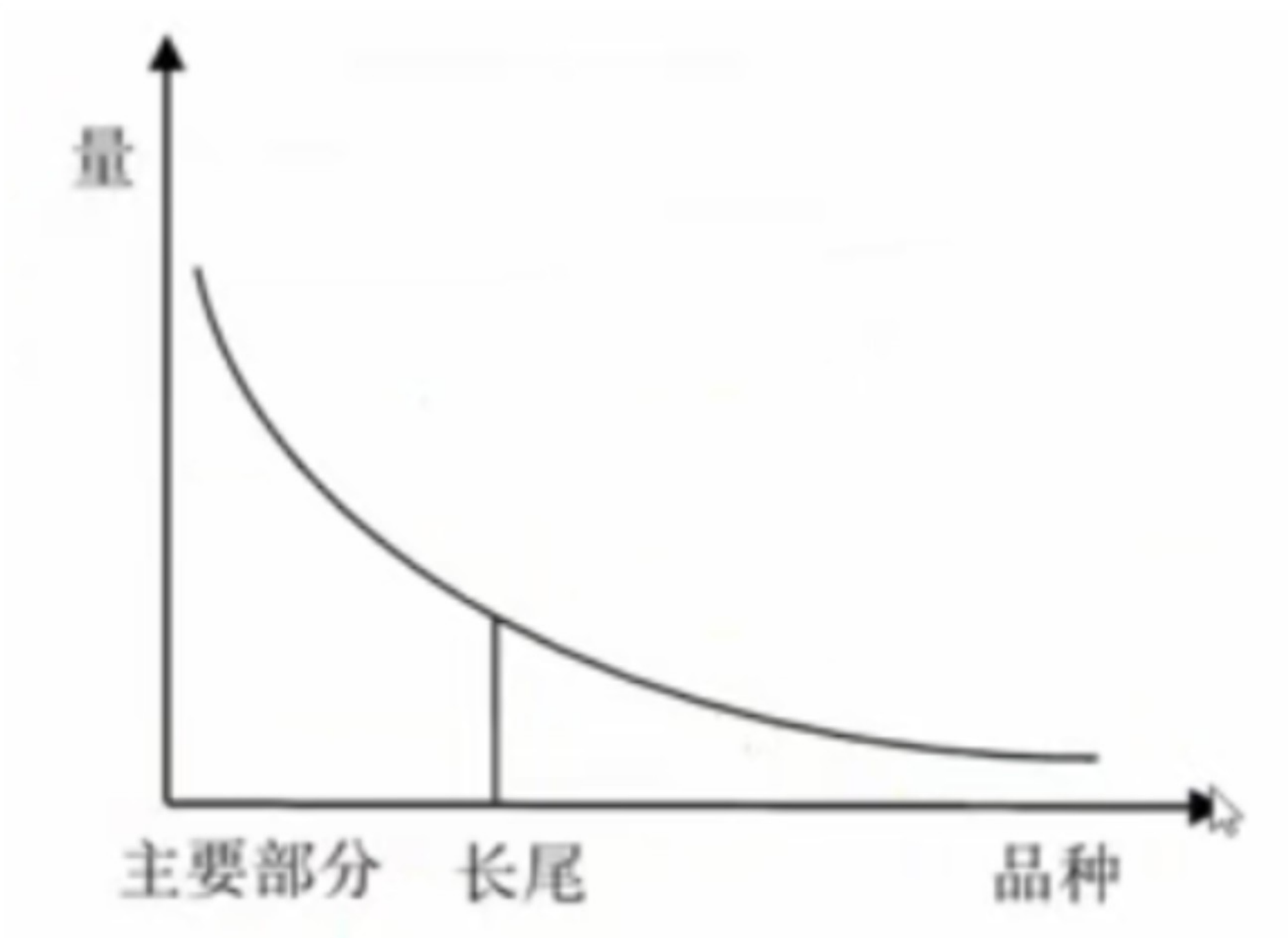
（2）泛化能力

它的主要来源是特征之间的相关性以及传递性。有可能特征A和B直接和label相关，也可能特征A与特征B相关，特征B与label相关，这种就称为传递性。利用特征之间的传递性， 我们就可以探索一些历史数据当中很少出现的特征组合，从而获得很强的泛化能力。

寻找特征间的隐形关系是深层神经网络的强项，神经网络可以通过多层特征交叉来**挖掘各种特征间的各种关系**，所以深层神经网络更利于提高模型泛化性。

Embedding也能提高模型的泛化能力，因为它将特征抽象成隐向量，这些隐向量在训练过程中往往是可学习的。但Embedding也存在一些缺点，它可能因为 "数据长尾分布"，导致长尾的一些特征值无法被充分学习，器对应的Embedding向量就不太准备，这会造成模型的泛化过度！

> 长尾效应：在经济学中表示20%的产品带来了80%的收益；在推荐场景中，可以理解为20%的产品包含了80%的点击



3. 提出Wide & Deep模型的背景

（1）在此之前，如何提高模型的记忆能力呢?

通常是人观察到哪些特征具有强关联，然后组合这些特征，使模型具有"记忆性“。这种方式具有很大的缺点：

a. 这种特征工程需要耗费太多精力；

b. 种是强行让模型记忆这些组合特征的，所以对于未出现过的特征组合，则无法进行泛化。

（2）不同模型的优缺点

1）简单模型

如：协同过滤、逻辑回归等简单模型能从历史数据中挖掘除 "共现频率" 高的组合特征，具有较强的记忆能力；

缺点就是：泛化能力不足。

2）矩阵分解、基于embedding+深度神经网络的模型

能够利用特征间的相关性和传递性去探索历史数据中未出现的特征组合，挖掘数据潜在的关联信息。优点就是：泛化能力强；

缺点就是：对于某些特定场景（数据长尾分布，贡献矩阵稀疏高秩）很难学习低纬表示，从而造成过度泛化。

4. Wide & Deep模型结构

模型的整体架构如下：

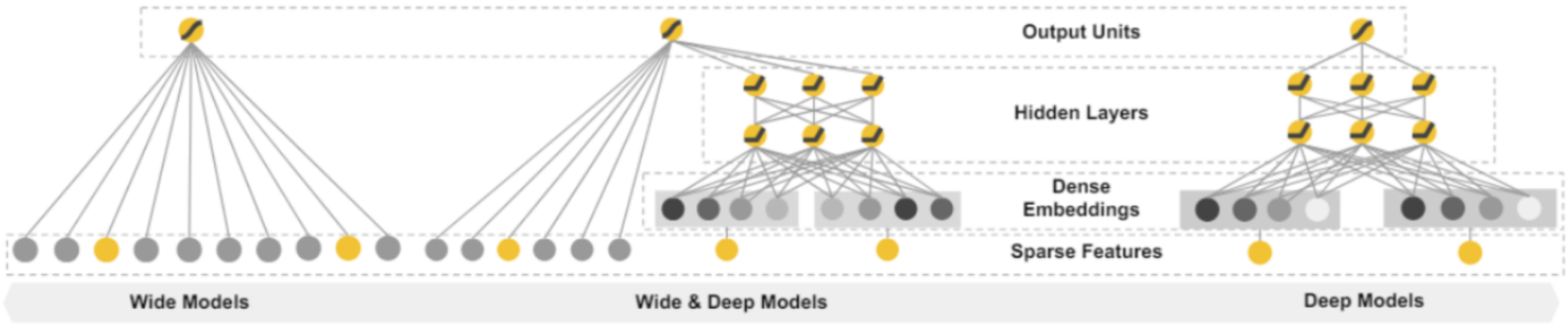


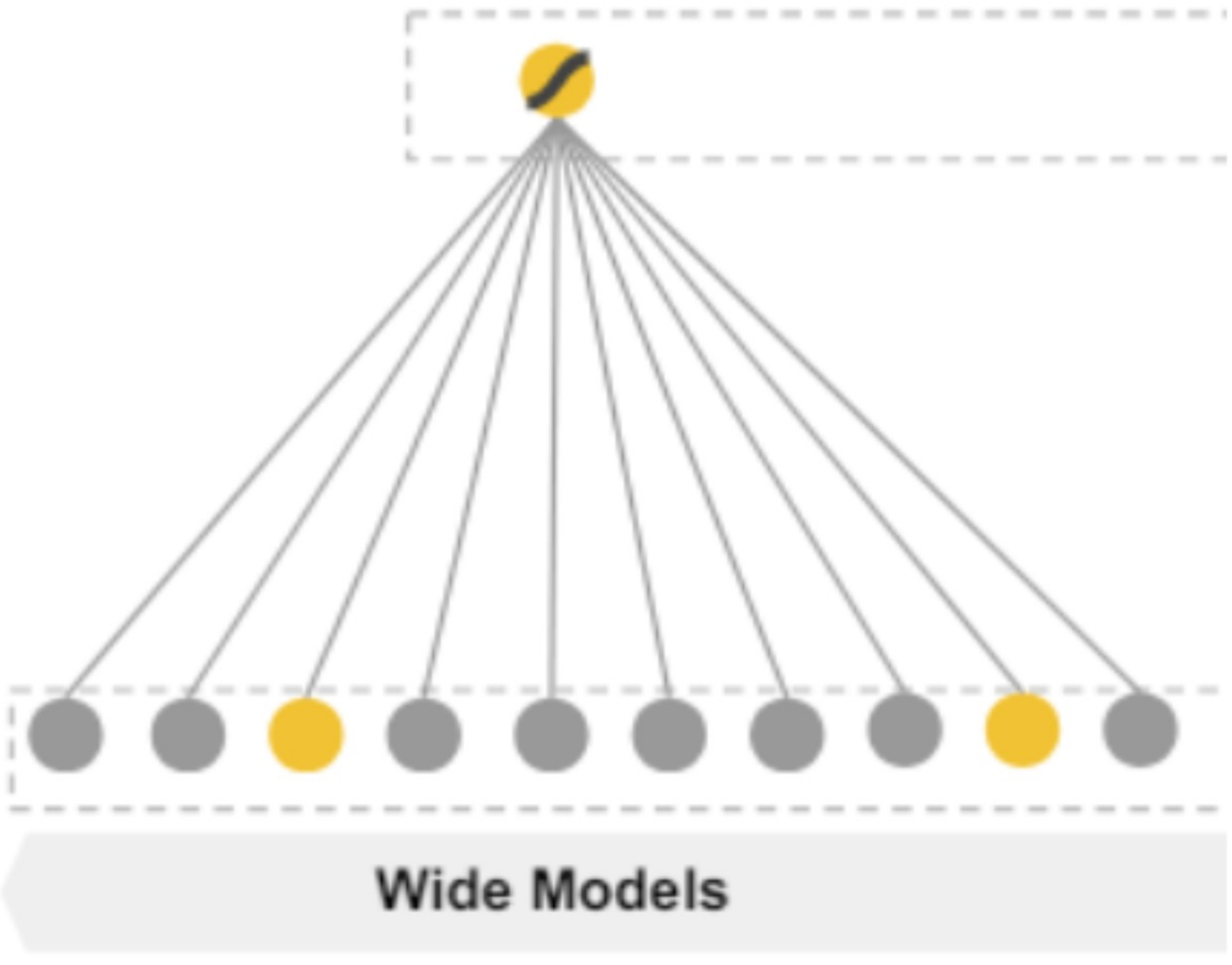
Figure 1: The spectrum of Wide & Deep models.

左侧是Wide部分，右侧是Deep部分，中间是两者结合的部分！

该模型通过简单的wide部分增强模型的记忆能力，通过Deep部分提高模型的泛化能力！

4.1 Wide模块

（1）模型前向传播部分



Wide部分是一个广义的线性模型，其数学含义如下： $y = W^T X + b$ 其中的 y 是输出的结果，W是权重向量， b 是偏置，这两个都是可学习的。 最为重要的是： X 是特征，它是一个 d 维向量 $X = [x_1, x_2, \ldots, x_d]$ ，这里的 d 是特征数量 ，它主要包含两种特征（其实还有一些离散的id类特征数据，因为神经网络不喜欢这种高稀疏的离散id特征，而Wide部分擅长处理这种）：一种是原始数据；另外一种是经过特征转化之后的特征。其中最为重要的一种转化形式就是交叉组合，交叉组合可以定义为如下形式：

$$\phi(X) = \prod_{i=1}^{d} x_i^{c_{ki}} \qquad c_{ki} \in \{0, 1\}$$

其中 X 表示所有的特征， d 表示所有特征的个数， i 表示第i个特征， k 表示第 k 个特征组合， c_{ki} 表示当第 i 个特征 x_i 属于第 k 个特征组合时，则为1；反之，为0。

举例说明： 假设我们的特征有：gender、age、name、address、language等，现有一个特征组合k=And(gender=female, language=en)，即：当gender=female，且language=en时候，才为1。

（2）优化器

Wide部分的优化器不是传统的SGD梯度下降算法，而是采用 ***FTRL+L1正则化***。FRPL的理论是非常复杂的，如果想详细了解，可以去搜索 **"冯扬 在线最优化求解"**。简而言之，***FTRL是一种稀疏性好、精度有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FTRL后，再说L1正则化，L1和L正则化都是用来控制模型复杂度和防止过拟合的常见手段，而L1正则化有利于模型稀疏，至于为什么能使得模型稀疏，可详见博客：[算法面试]_01_L1和L2正则化，为什么L1正则化更容易导致稀疏?

综上，**使用 "ETRL+L1" 是为了让模型更为稀疏，直观地说，就是让大部分权重都为0**，这样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。

（3）Wide部分为什么要稀疏

稀疏性往往都会使得模型精度降低，所以Wide部分为什么要稀疏呢？**肯定是特征向量维度太高，从而导致稀疏性成为了关键的考量。**

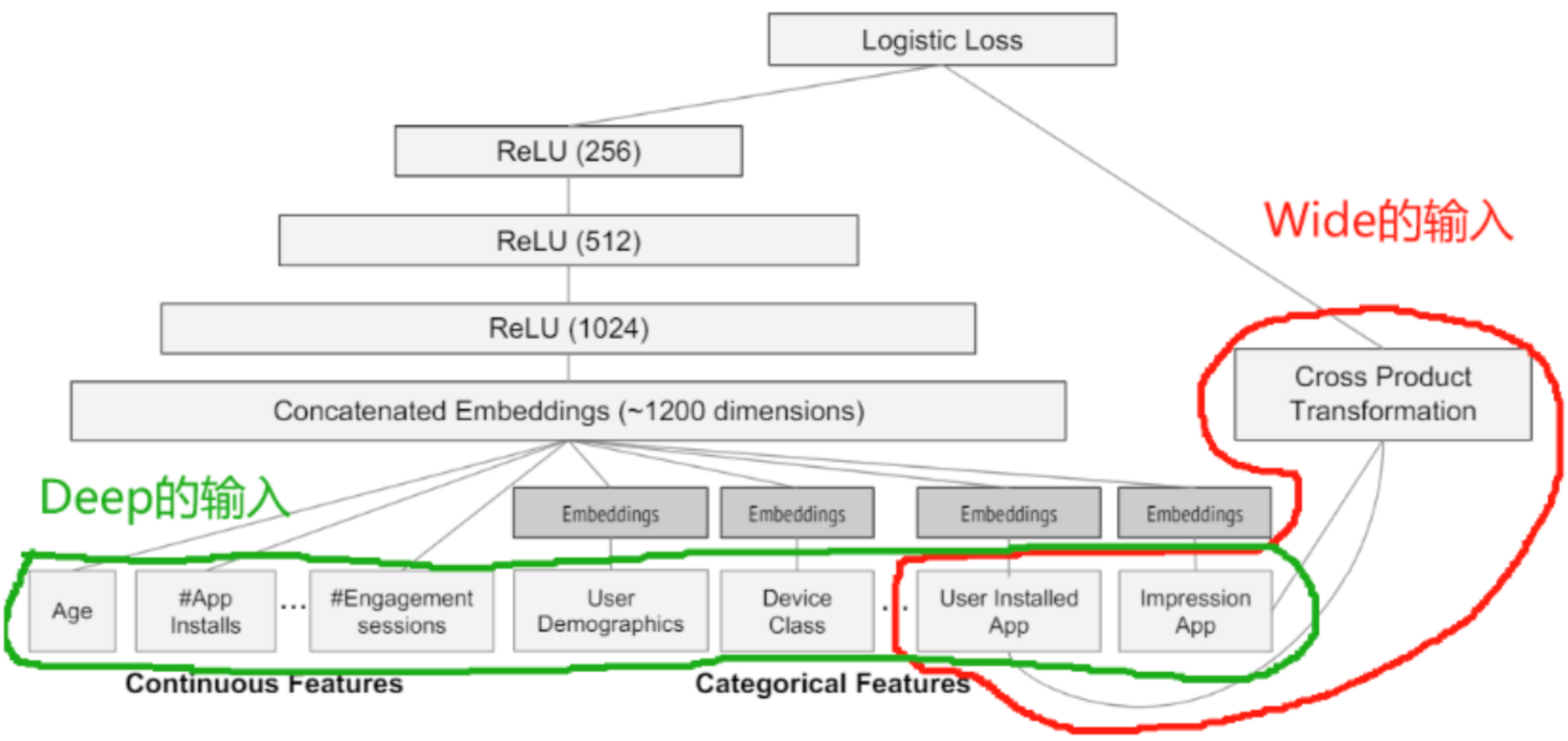


Figure 4: Wide & Deep model structure for apps recommendation.

上图是原文中的特征选取图，采用了User Installed App 和ImpressionAPP两个id类特征进行交叉积转换！

这篇文章是Google的应用商店团队Google Play发表的，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，以充分发挥Wide部分的记忆能力，以此来直接影响最终的得分。

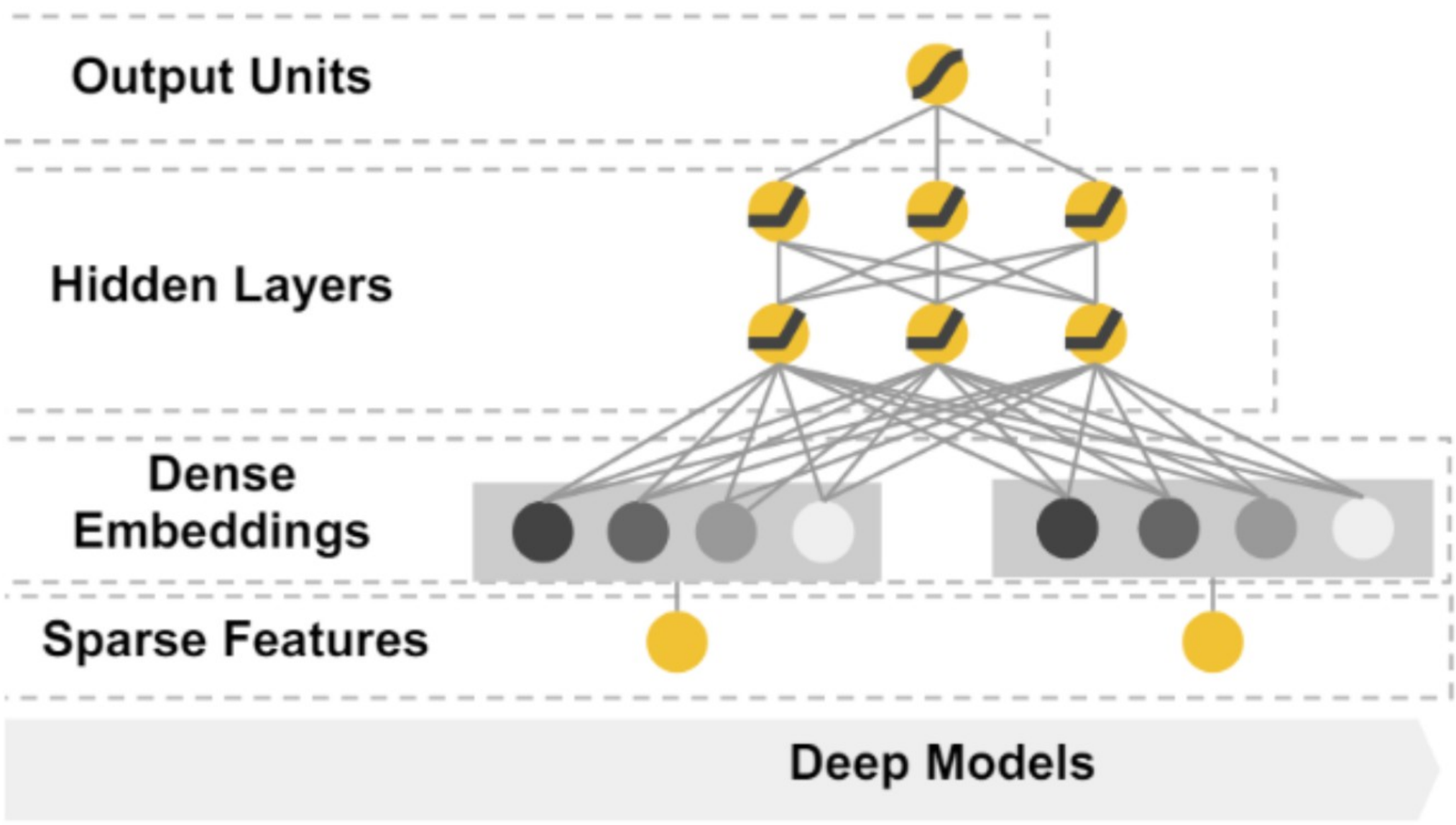
但是两个id类特征向量进行组合，在维度爆炸的同时，会让原本已经非常稀疏的multihot特征向量，变得更加稀疏。**正因如此，wide部分的权重数量其实是海量的。为了不把数量如此之巨的权重都搬到线上进行model serving，采用FTRL过滤掉哪些稀疏特征无疑是非常好的工程经验。**

（4）为什么后面的Deep部分不用稀疏性

从上图可以看出，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。

4.2 Deep模块

Deep部分就是简单的DNN网络，如图所示：



它的输入是一个sparse的feature，可以简单理解成multihot的数组。这个输入会在神经网络的第一层转化成一个低维度的embedding，然后神经网络训练的是这个embedding。这个模块主要是被设计用来处理一些类别特征，比如说item的类目，用户的性别等等。

和传统意义上的one-hot方法相比，embedding的方式用一个向量来表示一个离散型的变量，它的表达能力更强，并且这个向量的值是让模型自己学习的，因此泛化能力也大大提升。这也是深度神经网络当中常见的做法。

Deep部分在训练时是使用AdaGrad优化算法。

4.3 Wide & Deep 联合（joint）训练

单独讲了各个模块之后，合起来如下：

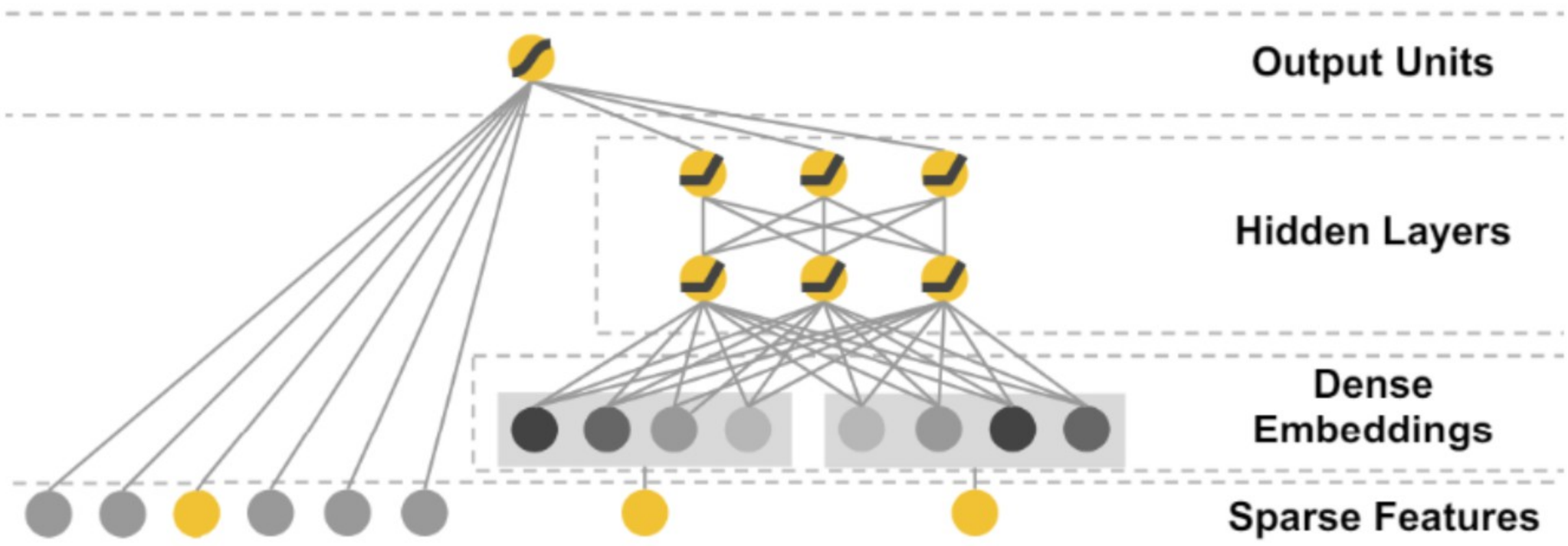


Figure 1: The spectrum of Wide & Deep models.

它的数学表现形式如下：

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

很多细节从图难以看出来，但从公司看起来就很清晰了，Wide和Deep的合并是采用相加的方式进行合并的，然后输入到激活函数中，就可以得出预测结果了。

另外，值得一提的是：论文特意**强调了Wide模型和Deep模型是联合(Joint)训练的**，与集成(Ensemble)是不同的，集成训练是每个模型单独训练，再将模型结果汇总。因此每个模型都会学的足够好的时候才会进行汇总，故每个模型相对较大。而对于Wide&Deep的联合训练而言，Wide部分只是为了补偿Deep部分缺失的记忆能力，它只需要使用一小部分的叉乘特征，故相对较小，**联合训练是同时训练的。**

自此，全文结束！

5. 本人疑惑-讨论点

（1）关于训练

原文说了在wide模块采用的是 “FTRL+L1” 优化的，但我看网络上面的代码都是采用的Adam。

6. EasDeepRecommand个人推荐系统开源项目介绍

更多推荐算法源码：

EasyDeepRecommand
github.com/lamctb/EasyDeepRecommand

一个通俗易懂的开源推荐系统（A user-friendly open-source project for recommendation systems）.

本项目将结合：代码、数据流转图、博客、模型发展史等多个方面通俗易懂地讲解经典推荐模型，让读者通过一个项目了解推荐系统概况！

持续更新中..., 欢迎star , 第一时间获取更新，感谢!!!

参考资料

> [1]【推荐算法】Wide&Deep模型 —— 谷歌曾经的主流推荐模型，在业界影响力巨大，综合记忆能力与泛化能力 bilibili.com/video/BV1U... [2] zhihu.com/tardis/zm/art...

编辑于 2025-03-03 14:13 · IP 属地北京