Summary of Several Recent Year Papers Concerning CTR Prediction

1 DeepFM (2017)

1.1 Source

DeepFM: A Factorization-Machine based Neural Network

IJCAI 2017, HIT & Huawei

- DeepFM 可以看成是把 Wide & Deep 的 LR 换成了 FM，并且修改了 Feature Embedding 变成了 Wide 和 Deep 的底层。

1.2 Code

https://github.com/ChenglongChen/tensorflow-DeepFM

Implemented by **Python** with **Tensorflow** framework

1.3 Overview

- ✓ End-to-end。

  训练的人工工作量小；

- ✓ Able to capture both **High- and Low- order** feature interactions；

- ✓ Having a share input to its **Wide and Deep** part。

  和 Wide and Deep 模型相比，DeepFM 的 Wide 和 Deep 部分共享相同的输入，不需要额外的特征工程；

- ✓ DeepFM 的 Wide 部分是 FM，Deep 部分是 DNN；

- ✓ Architecture

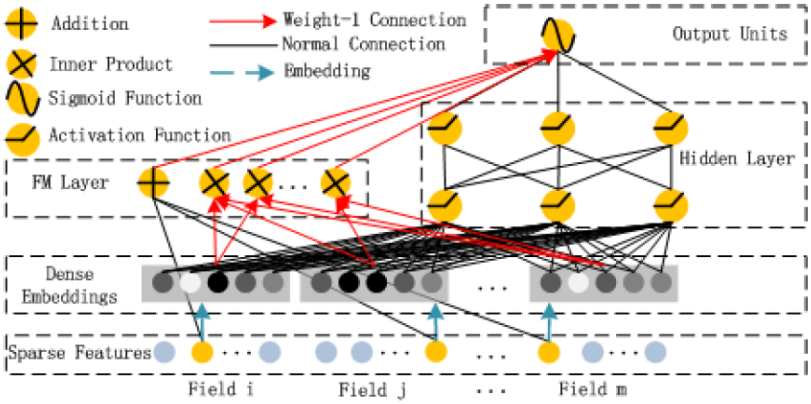


Figure 1: Wide & deep architecture of DeepFM. The wide and deep component share the same input raw feature vector, which enables DeepFM to learn low- and high-order feature interactions simultaneously from the input raw features.

✓ Details

FM Component

Nothing special…

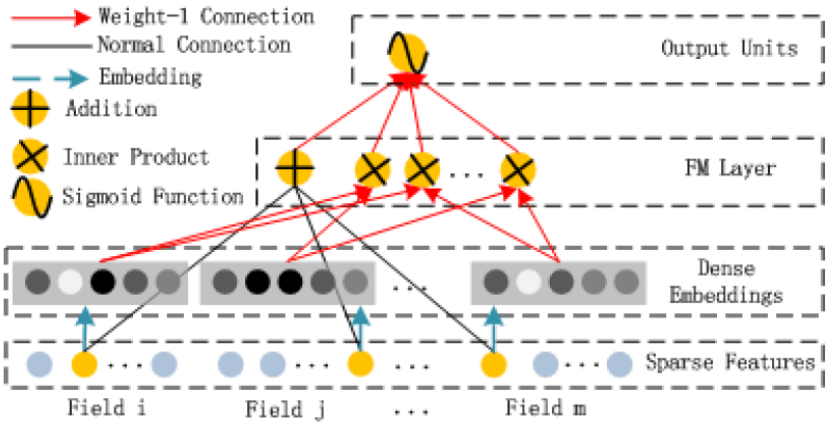


Figure 2: The architecture of FM.

Deep Component

这里的结构也比较简单，FC(200-200-200) +ReLU +Dropout

我们也可以进行一些其他的尝试，毕竟往深了去提 Feature 的网络结构很多，ResNet、DenseNet…

他们的数据量是 1 Billion，估计是机器跑不动……

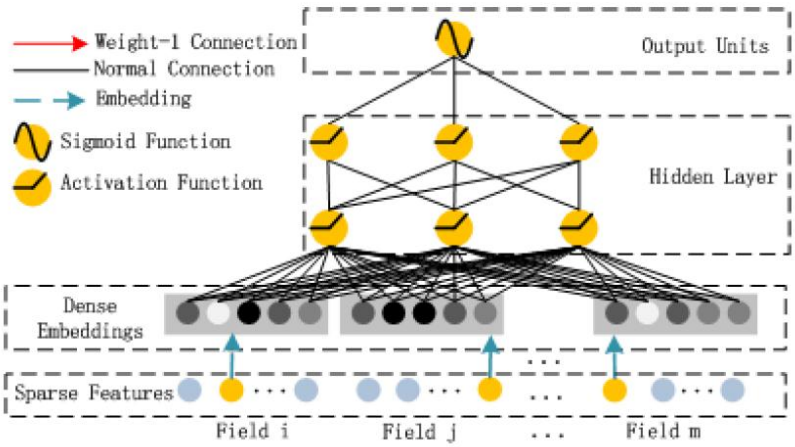


Figure 3: The architecture of DNN.

Details and **Tricks:**

a) ReLU as activation function

b) Dropout applied with keep ratio of 0.9

c) Constant layer shape 200-200-200

2 Deep & Cross (2017)

2.1 Source

Deep & Cross Network for Ad Click Predictions

ADKDD 2017, Stanford University

2.2 Code

https://github.com/Nirvanada/Deep-and-Cross-Keras

Implemented by **Python** with **Keras** framework

1.3 Overview

✓ Feature Cross 的概念都在网络结构中去实现的，不需要在 Feature Engineering 阶段去做；

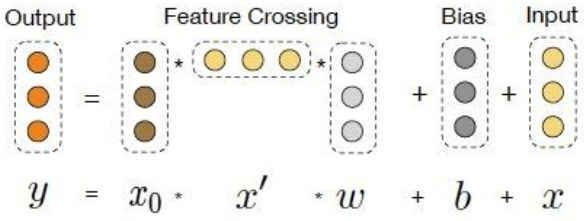
✓ Cross Network 部分：

这部分网上有段讲得很有意思，author 可能在 Cross Network 这部分借鉴了 ResNet: 如下图：

将输入的embedding column + continous column定义为x_0（$x_0 \in R^d$），第$l+1$层的cross layer为

$$x_{l+1} = x_0 x_l^T w_l + b_l + x_l$$

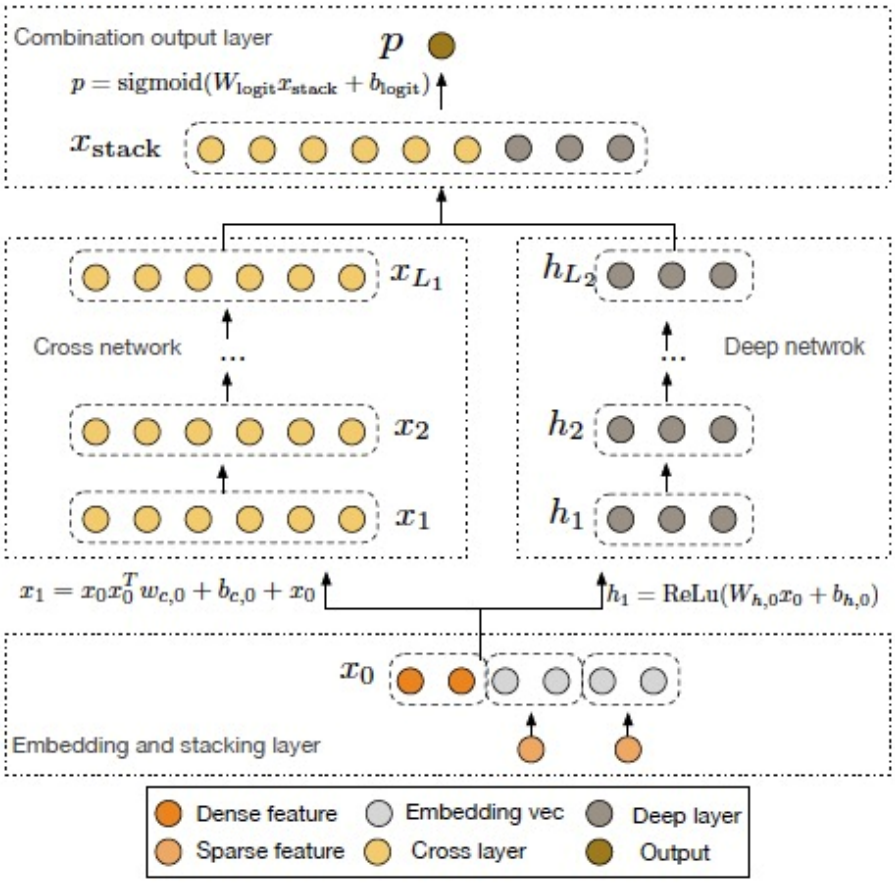
其中$w_l (w_l \in R^d)$和$b_l (b_l \in R^d)$为第l层的参数。这么看来，Cross这部分网络的总参数量非常少，仅仅为$layers * d * 2$，每一层的维度也都保持一致，最后的output依然与input维度相等。另一方面，特征交叉的概念体现在每一层，当前层的输出的higher-represented特征都要与第一层输入的原始特征做一次两两交叉。至于为什么要再最后又把x_l给加上，我想是借鉴了ResNet的思想，最终模型要去拟合的是$x_{l+1} - x_l$这一项残差。



✓ Deep Network 部分：

简单的 FC…然后在 Output 阶段过一层 Softmax

✓ Architecture：



Details and **Tricks:**

a) Hidden layer size: 32-1024

b) Hidden layer range from 2-5

c) Early stopping at 150,000

d) Not using L2 and Dropout.

e) Batch size: 512

f) Log transform for real-valued features

3 Wide & Deep (2016)

3.1 Source

Wide & Deep Learning for Recommender Systems

DLRS 2016, Google

3.2 Code

Tensorflow Implementation:

https://github.com/ichuang/tflearn_wide_and_deep

Keras Implementation(简单):

https://github.com/jrzaurin/Wide-and-Deep-Keras

Wide and Deep 的名气比较大，一个典型的应用是【美团的排序模型】:

https://mp.weixin.qq.com/s/847h4ITQMtUlZcurJ9Vlvg?scene=25##

3.3 Overview

✓ Architecture

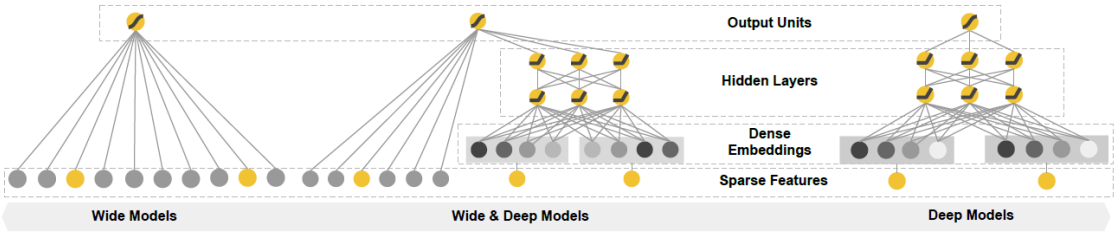


Figure 1: The spectrum of Wide & Deep models.

✓ Features

1. user features (e.g., country, language, demographics),
2. contextual features (e.g., device, hour of the day, day of the week)
3. impression features (e.g., app age, historical statistics of an app).

- ✓ Feature Engineering for Wide Part needs human expertise

- ✓ **Deep Part**: DNN
  Wide Part: LR

4 PNN (2016)

4.1 Source

Product-based Neural Networks for User Response Prediction

2016，Shanghai Jiao Tong University

4.2 Code

https://github.com/Atomu2014/product-nets

Implemented by **Python** with **Tensorflow** framework

4.3 Overview

✓ Architecture

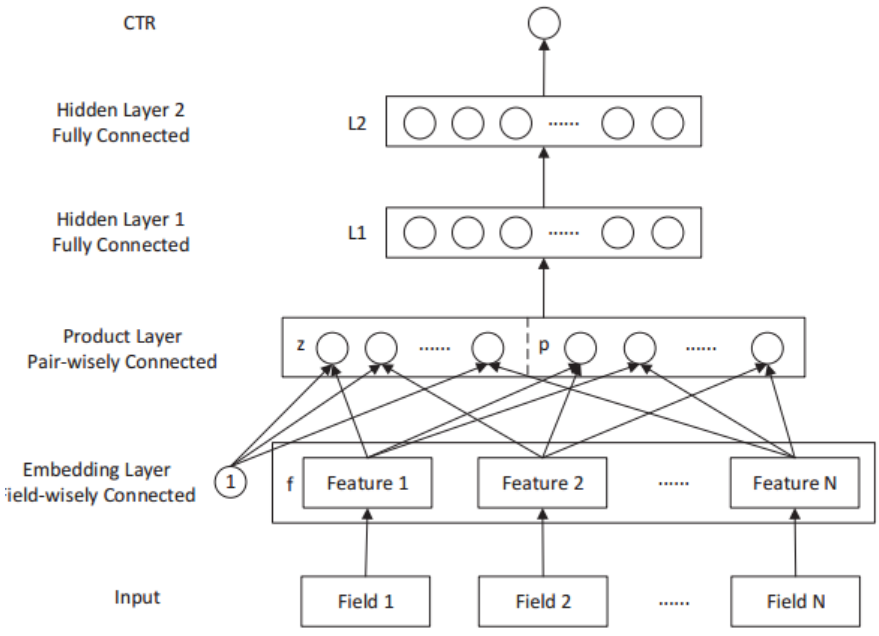


Fig. 1: Product-based Neural Network Architecture.

✓ 一句话概括就是 Embedding + Product Layer + DNN

✓ Some Variants: IPNN、OPNN、PNN*(按照 Product Layer 的不同分类)

5 结论

不同的模型，其实着眼于这样几个问题：

1. 如何提升模型的 Wide，如何提升模型的 Deep？
   Deep 通常用 DNN，但是用什么样的结构以及使用什么 Trick？
   Wide 有 LR，也有 FM；

2. Wide 维度和 Deep 维度应该以什么样的结构整合；

3. 考虑 Feature 的 2 阶交互项，结构应当如何设计；

4. Sparse Feature => Embedding