

# 王耳学推荐 | (四) wide & deep 和 deep & cross

原创 王耳 sad tom cat 2019-12-24

收录于话题 #王耳学推荐

14个

前阵子在组里做过串讲，有关于工业界一些比较经典的神经网络的介绍。这里把它们整理成公众号。这里先将wide and deep和deep and cross写上。

## Dirty Deeds Done Dirt Cheap

The Backing Tracks - Play Bass with the Music of AC/DC



## 简易目录

- 引言
- wide and deep (以下简称wnd)
- deep and cross (以下简称dcn)
- 总结

## 引言

先说我当时接触到这两个神经网络结构的第一印象：**秀！** 对深度学习的理解还只停留在市面上工具书介绍内容上的我，在看到谷歌大佬们写的这两篇论文，真的有被惊艳到。因为我还没有想过特征还能这么玩，结构还可以这样设计。好的，不多吹了。首先放上论文的链接：

- wnd(2016) : <https://arxiv.org/pdf/1606.07792.pdf>
- dcn(2017) : <https://arxiv.org/pdf/1708.05123.pdf>

建议读原文，体味浓浓google工业风。

## wide and deep

wnd设计之初，是应用在推荐系统领域，是为了解决单纯的dnn模型在实际操作过程中过于泛化的问题。过于泛化这个现象，具体表现为：在对稀疏的特征进行embedding之后，得到是稠密的向量，这使得所有未在train中出现过的“用户--物品”对都得到了非零的预

测。这样为用户和物品强加联系的做法就会产生毫无关联的推荐结果。就个人理解，这算是“成也embedding，败也embedding”，尽管embedding技术将高维且稀疏的数据映射到一个低维流形上，这很方便，确实很方便，这样减少了在特征工程上的劳动力，但是带来的潜在问题（上述）也是需要引起重视的。而在google则提出wnd这样的网络结构，同时构建两个网络：liner model (wide) 和 deep nerual network (deep)，拼接两个网络的输出再进入sigmoid函数，使得参数更新时可以同时感知到这两个网络。

下图是wnd的整体结构对比：

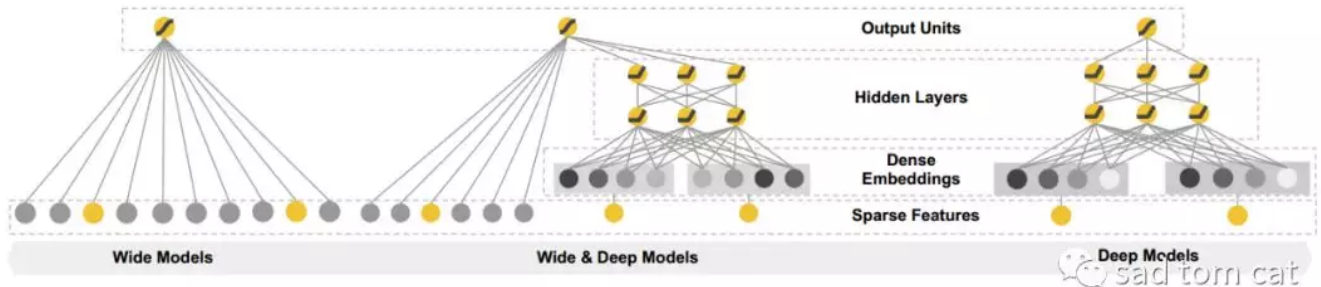


Figure 1: The spectrum of Wide & Deep models.

- wide model, 主memorization, 模型效果表现为可以从历史数据中挖掘出物品/特征之间的关联，记忆那些出现频繁的共现结构。那没有在历史数据中出现过呢，模型的效果如何呢？答：表现欠佳。
- deep model, 主generalization, 模型效果表现为可以产生更强的泛化能力，通过embedding技术得到未被观察到特征交互。

下图是wnd的特征输入和网络结构：

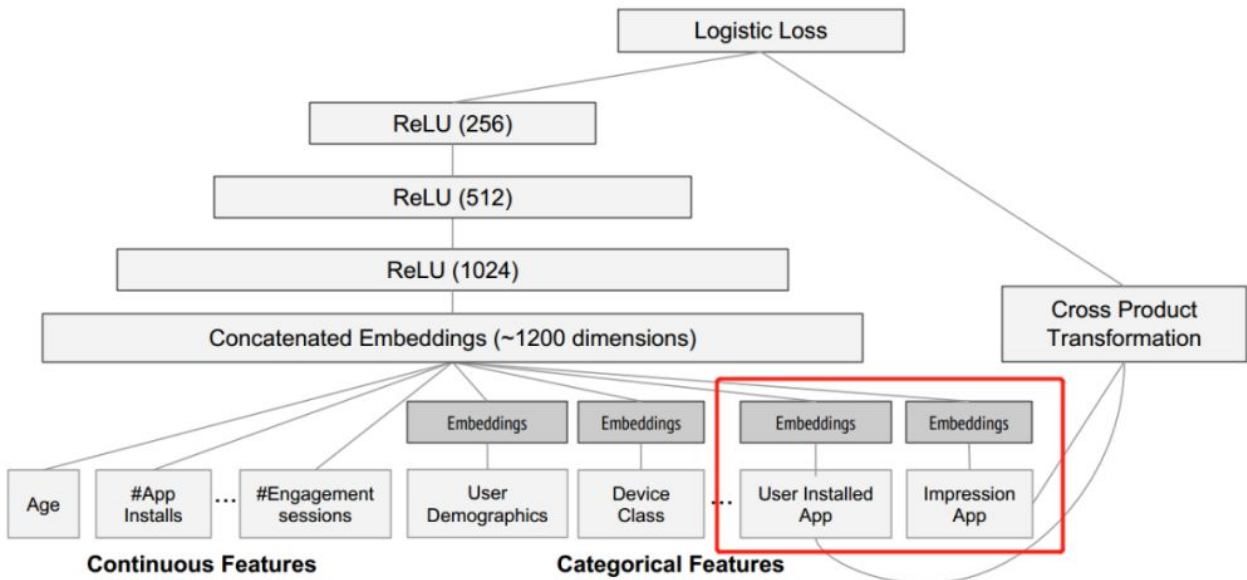




Figure 4: Wide & Deep model structure for apps recommendation.  sad torn cat

需要注意，图中的wide部分，经过了一个**cross product transformation**的过程。其实是这样的：


$$\phi_k(\mathbf{x}) = \prod_{i=1}^d x_i^{c_{ki}} \quad c_{ki} \in \{0, 1\}$$

 sad torn cat

个人认为，这是针对分类特征处理的一种比较好的手段，提高了wide模型的表达能力，论文的原话是 “adds nonlinearity to generalized linear model” 。

下式是最终的输出层形式：

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

 sad torn cat

值得一提的是，这里的embedding是跟随网络一起进行训练的，抛弃了以前事先得到embedding表示，再作为模型变量一起输入的老做法。个人认为，**这很符合实际场景的需求**，sparse feature的embedding结果也需要跟着场景调整更新的，并不是一份完成好的embedding就能一劳永逸的。但这也增加了额外的训练成本。有利有弊嘛。

另外，下图是一个成熟的搜索/推荐引擎的流程图，我觉得很棒，出自wnd原文论文：

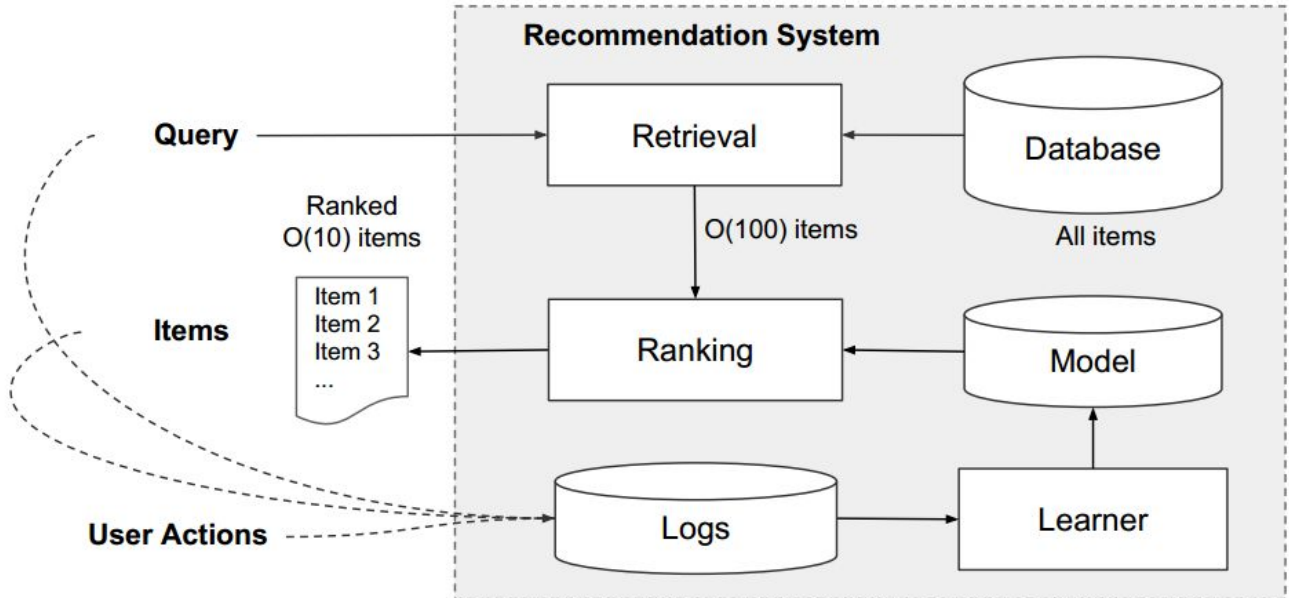



Figure 2: Overview of the recommender system.  ashitama cat

## deep and cross

dcn的诞生动机也很实在，就是**为了特征之间的交叉**，借以提升广告的点击率。特征交叉，获得更丰富，更多样，更合适的特征，以提升模型的表达能力，一直是特征工程的重要内容之一。（最近也在看一本《精通特征工程》的书）。比如因子分解机（Factorization Machine，简称FM），场感知因子分解机（Field-aware Factorization Machine，简称FFM）是建立在矩阵分解，这样的做法是建立在矩阵分解的基础上进行的特征分解，一旦需要获得高阶的交互特征，在实际场景中带来的计算成本是难以想象的。抑或是如DNN一类的神经网络结构，是依赖embedding技术和非线性变换的特点，达到特征交叉。但是DNN得到的高阶特征通常过于隐晦，且难以解释。在实际操作中，更倾向于设计一个网络，让它能更高效地生产显式而又不过于复杂的交叉特征。wnd就是一个很好的例子。

dcn也秉承着上述说到的特点，设计了一个比较新颖的cross结构，完成特征的交叉。总体结构如下图所示：

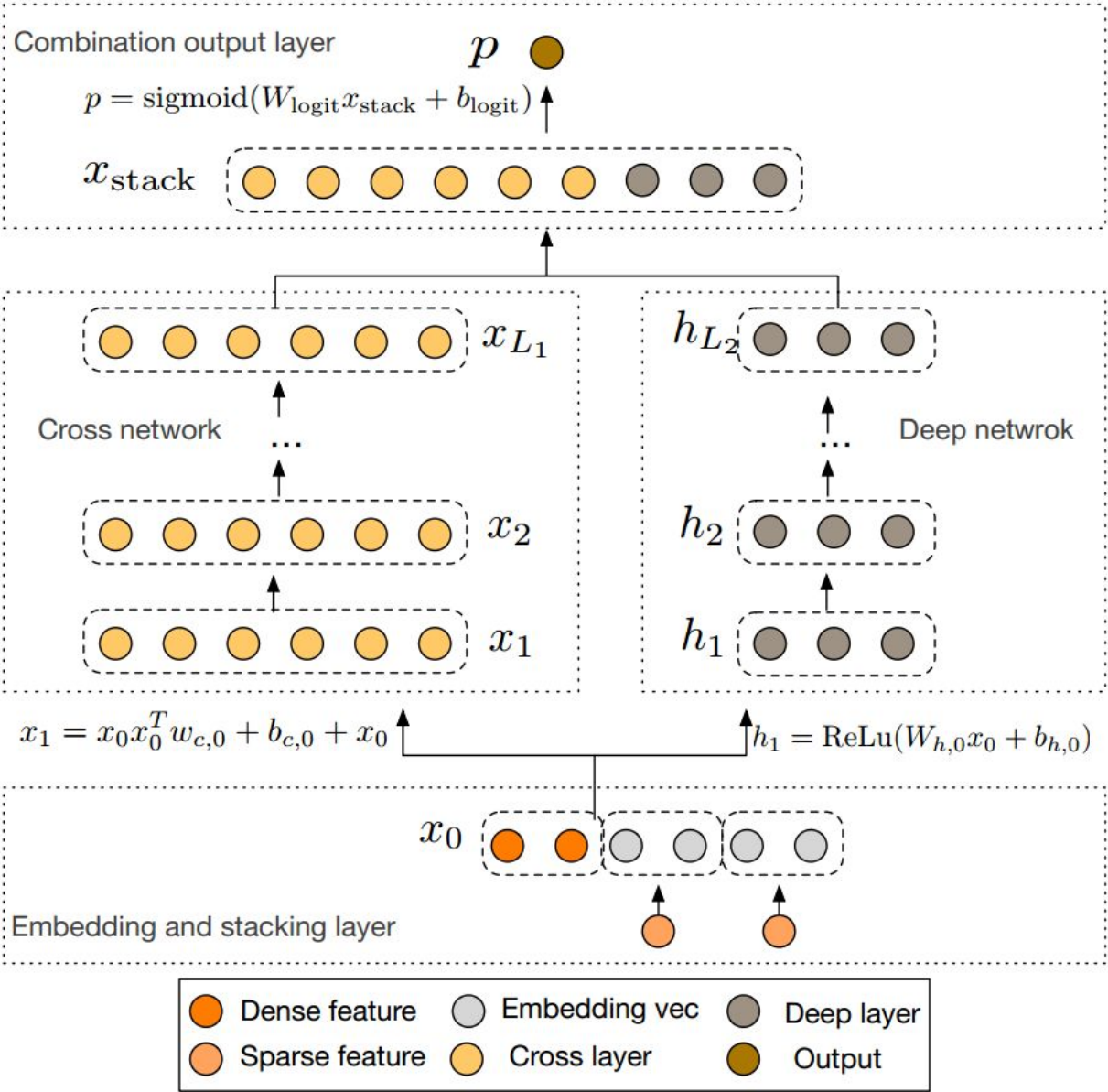
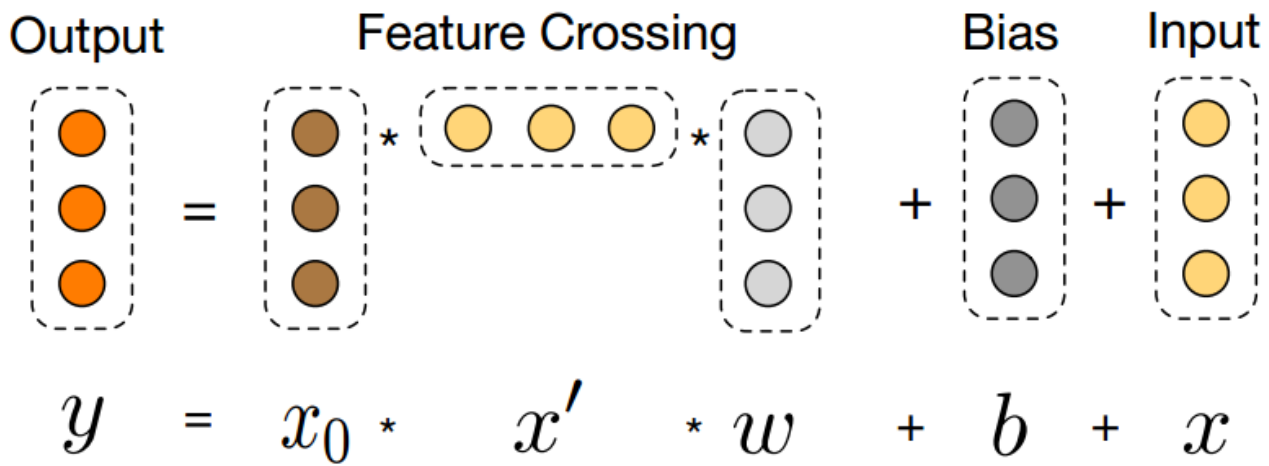


Figure 1: The Deep & Cross Network  sad torn cat

不难看出，有些地方还是和wnd有相似的地方，比如两个网络同时训练，再比如两个网络的输出，concat之后再输入到sigmoid函数中。下图详细展示cross部分的结构：





**Figure 2: Visualization of a cross layer.**

其中 $x'$ 是第 $l$ 层的结果， $y$ 是第 $(l+1)$ 的结果。后面的层的结果，不断地与输入层 $x_0$ 进行交叉，获得更复杂的交叉结果。**吐槽**:最后的 $x$ 其实也应该写成 $x'$ 吧，毕竟迭代公式是这样的：

$$\mathbf{x}_{l+1} = \mathbf{x}_0 \mathbf{x}_l^T \mathbf{w}_l + \mathbf{b}_l + \mathbf{x}_l = f(\mathbf{x}_l, \mathbf{w}_l, \mathbf{b}_l) + \mathbf{x}_l$$

文中还提到dcn在本质是一种FM的拓展，可以理解为**利用既有的特征，通过交叉产生独立于现有特征的高阶特征**。欲使用dcn获得高阶特征，仅需要增加cross结构的层数即可，而且需要的参数也是输入层维数 $d$ 的线性倍数。我认为这才是dcn的优势所在。

## 总结

wnd和dcn都是很出色的网络结构，比起照搬，更要深入理解他们这样设计网络的初衷和理论基础。wnd兼顾memorization和generalization，dcn侧重feature crossing。

<=== to be continued...