计算广告CTR预估系列(十一)--谷歌DCN模型理论与实践

原创 可爱又迷人的反派角色宁宁 机器学习荐货情报局 2018-09-12

计算广告CTR预估系列(十一)--谷歌DCN模型理论与实践

目录

一、介绍

> DCN全称Deep & Cross Network。

CTR预估全称是*Click Through Rate*，就是展示给用户的广告或者商品，估计用户点击的概率。公司规模较大的时候，CTR直接影响的价值在数十亿美元的级别。广告支付一个非常流行的模型就是CPC(cost-per-click)，就是按照用户的点击来付钱。那么准确的进行CTR预估，展现给用户他们最可能点击的广告就非常重要了。

传统的CTR预估模型需要大量的特征工程，耗时耗力；引入DNN之后，依靠神经网络强大的学习能力，可以一定程度上实现自动学习特征组合。但是DNN的缺点在于隐式的学习特征组合带来的不可解释性，以及低效率的学习(并不是所有的特征组合都是有用的)。

DCN全称*Deep & Cross Network*，是谷歌和斯坦福大学在2017年提出的用于Ad Click Prediction的模型。DCN(Deep Cross Network)在学习特定阶数组合特征的时候效率非常高，而且同样不需要特征工程，引入的额外的复杂度也是微乎其微的。

跟着小编一起走进DCN的世界吧！

二、相关工作

最开始FM使用隐向量的内积来建模组合特征；FFM在此基础上引入field的概念，针对不同的field上使用不同隐向量。但是，这两者都是针对低阶的特征组合进行建模的。
随着DNN在计算机视觉、自然语言处理、语音识别等领域取得重要进展，DNN几乎无限的表达能力被广泛的研究。同样也尝试被用来解决web产品中输入高维高稀疏的问题。DNN可以对高维组合特征进行建模，但是DNN是否就是针对此类问题最高效的建模方式那？直到现在，业界也没有一个准确的答案。
在Kaggle上的很多比赛中，大部分的获胜方案都是使用的人工特征工程，构造低阶的组合特征，这些特征意义明确且高效。而DNN学习到的特征都是高度非线性的高阶组合特征，含义非常难以解释。那么是否能设计一种DNN的特定网络结构来改善DNN，使得其学习起来更加高效那？

业内进行了很多探索，DCN就是其中一个。

三、DCN特点

DCN特点如下：

1. 使用cross network，在每一层都应用feature crossing。高效的学习了*bounded degree*组合特征。不需要人工特征工程。
2. 网络结构简单且高效。多项式复杂度由*layer depth*决定。
3. 相比于DNN，DCN的logloss更低，而且参数的数量将近少了一个数量级。

四、DCN

还记得DCN的全称是什么吗？*Deep & Cross Network*，聪明的你一定答对了吧！下面就跟着小编一起就进入到DCN里面一探究竟吧。

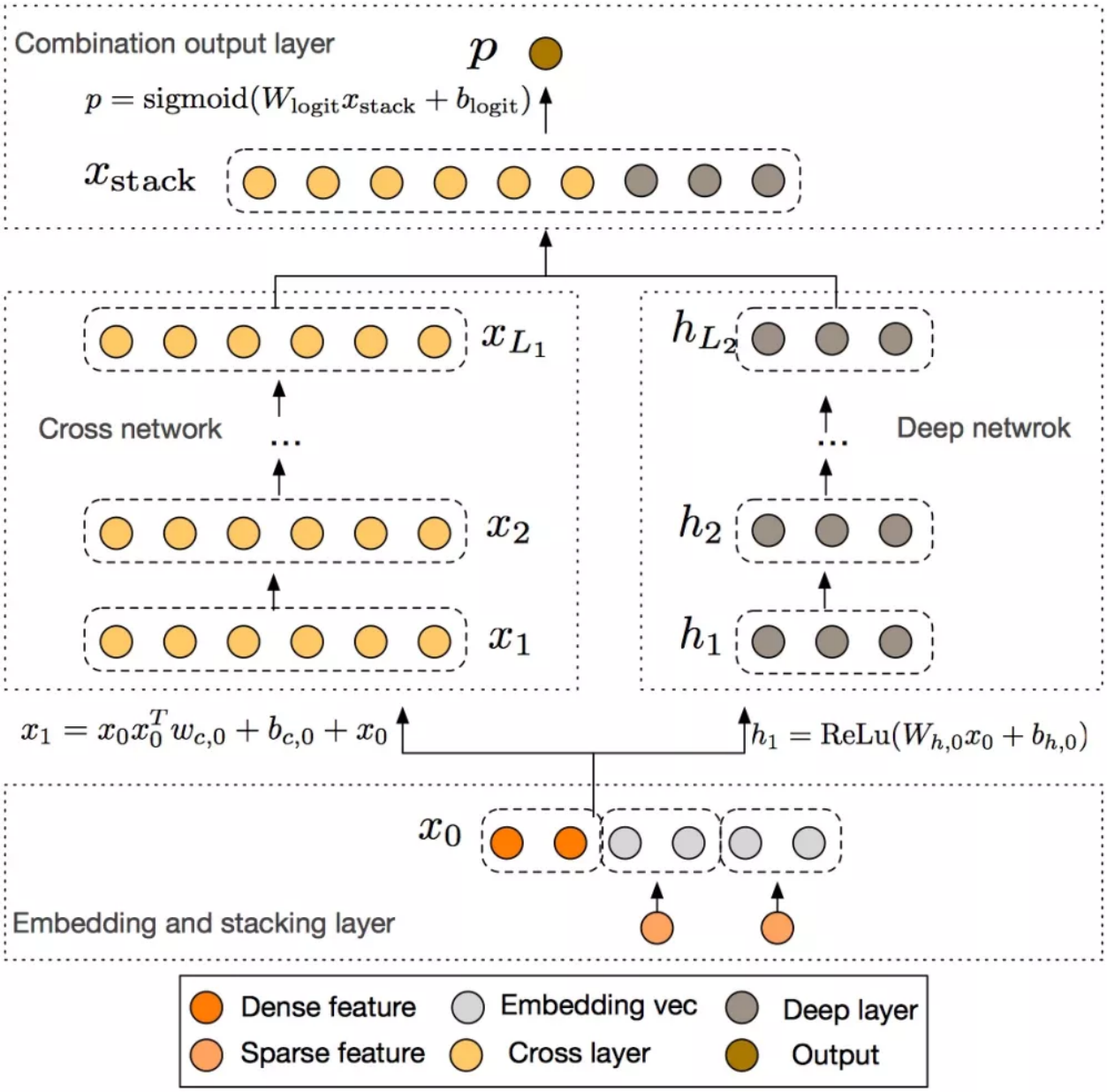

Figure 1: The Deep & Cross Network

DCN架构图

DCN架构图如上图所示：最开始是*Embedding and stacking layer*，然后是并行的*Cross Network*和*Deep Network*，最后是*Combination Layer*把*Cross Network*和*Deep Network*的结果组合得到*Output*。

下面详细解析下每一层具体怎么回事。

4.1 Embedding and Stacking Layer

这一层说起来其实非常的简单，就两个功能*Embed*和*Stack*。

为什么要Embed那？
在web-scale的推荐系统比如CTR预估中，输入的大部分特征都是类别型特征，通常的处理办法就是*one-hot*，但是one-hot之后输入特征维度非常高非常系数。
所以有了*Embedding*来大大的降低输入的维度，*就是把这些binary features转换成dense vectors with real values。*

Embedding操作其实就是用一个矩阵和one-hot之后的输入相乘，也可以看成是一次查询（lookup）。这个Embedding矩阵跟网络中的其他参数是一样的，是需要随着网络一起学习的。

为什么要Stack那？
处理完了类别型特征，还有连续型特征没有处理那。所以我们把连续型特征规范化之后，和嵌入向量*stacking*到一起，就得到了原始的输入：

$$\mathbf{x}_0 = \left[\mathbf{x}_{\text{embed},1}^T, \cdots, \mathbf{x}_{\text{embed},k}^T, \mathbf{x}_{\text{dense}}^T\right]$$

Embedding and Stacking

4.2 Cross Network

*Cross Network*是整篇论文的核心。它被设计来**高效的学习**组合特征，关键在于如何高效的进行*feature crossing*。形式化如下：

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

Cross Network

xl和xl+1 分别是第l层和第l+1层cross layer的输出，wl和bl是这两层之间的连接参数。注意上式中所有的变量均是**列向量**，W也是列向量，并不是矩阵。

该怎么理解那？

其实也不难，`xl+1 = f(xl, wl, bl) + xl`．每一层的输出，都是上一层的输出加上*feature crossing* f。而f就是在**拟合该层输出和上一层输出的残差。** 针对*one cross layer*可视化如下：

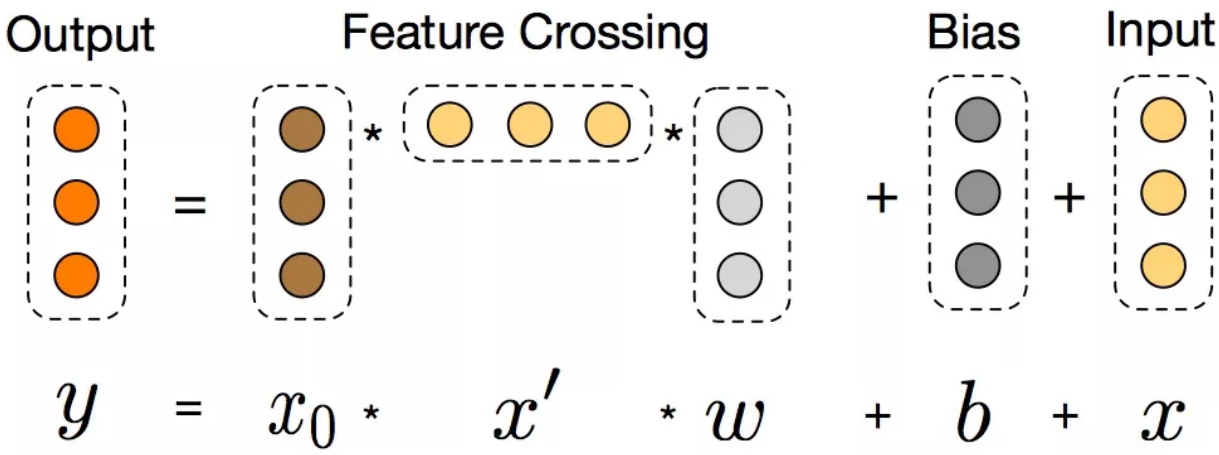

Figure 2: Visualization of a cross layer.

one cross layer

High-degree Interaction Across Features:

Cross Network特殊的网络结构使得cross feature的阶数随着layer depth的增加而增加。相对于输入x0来说，一个l层的cross network的cross feature的阶数为l+1。

复杂度分析：

假设一共有Lc层cross layer，起始输入x0的维度为d。那么整个cross network的参数个数为：

$$d \times L_c \times 2.$$

Number of Cross Network Parameters

因为每一层的W和b都是d维度的。从上式可以发现，复杂度是输入维度d的线性函数。所以相比于deep network，cross network引入的复杂度微不足道。这样就保证了DCN的复杂度和DNN是一个级别的。**论文中表示，Cross Network之所以能够高效的学习组合特征，就是因为 x0 * xT 的秩为1，使得我们不用计算并存储整个的矩阵就可以得到所有的cross terms。**

但是，正是因为cross network的参数比较少导致它的表达能力受限，为了能够学习高度非线性的组合特征，DCN并行的引入了Deep Network。

4.3 Deep Network

这一部分没什么特别的，就是一个前向传播的全连接神经网络，我们可以计算一下参数的数量来估计下复杂度。假设输入x0维度为d，一共有Lc层神经网络，每一层的神经元个数都是m个。那么总的参数或者复杂度为：

$$d \times m + m + (m^2 + m) \times (L_d - 1).$$

Number of Deep Network Parameters

4.4 Combination Layer

*Combination Layer*把Cross Network和Deep Network的输出拼接起来，然后经过一个加权求和后得到logits，然后经过sigmoid函数得到最终的预测概率。形式化如下：

$$p = \sigma\left([\mathbf{x}_{L_1}^T, \mathbf{h}_{L_2}^T]\mathbf{w}_{\text{logits}}\right)$$

Combination Layer

p是最终的预测概率；XL1是d维的，表示Cross Network的最终输出；hL2是m维的，表示Deep Network的最终输出；Wlogits是Combination Layer的权重；最后经过sigmoid函数，得到最终预测概率。

损失函数使用*带正则项的log loss*，形式化如下：

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p_i) + (1-y_i)\log(1-p_i) + \lambda \sum_{l} \|\mathbf{w}_l\|^2$$

loss function

另外，针对Cross Network和Deep Network，DCN是**一起训练的**，这样网络可以知道另外一个网络的存在。

五、泛化FM

跟FM一样，DCN同样也是基于参数共享机制的，参数共享不仅仅使得模型更加高效而且使得模型可以泛化到之前没有出现过的特征组合，并且对噪声的抵抗性更加强。
FM是一个非常浅的结构，并且限制在表达二阶组合特征上，DeepCrossNetwork(DCN)把这种参数共享的思想从一层扩展到多层，并且可以学习高阶的特征组合。但是和FM的高阶版本的变体不同，DCN的参数随着输入维度的增长是线性增长的。

六、实验

使用的是Criteo Display Ads Data. 特征如下：

- 输入有13个integer feature、26个categorical feature
- 一共是7天4亿条样本
- 训练集取前6天的数据，第七天的数据随机划分成相等的两部分分别作为测试集和验证集
- 针对这么大的用户量，logloss上0.001的提升都会在实际生产中带来巨大的影响

七、代码实战

核心部分代码如下：

Embedding部分：

```
    with tf.variable_scope("Embedding-layer"):
        embeddings = tf.nn.embedding_lookup(Feat_Emb, feat_ids)            # None
        feat_vals = tf.reshape(feat_vals, shape=[-1, field_size, 1])
        embeddings = tf.multiply(embeddings, feat_vals)                    # None
        x0 = tf.reshape(embeddings,shape=[-1,field_size*embedding_size])   # None
```

Cross Network部分：

```
    with tf.variable_scope("Cross-Network"):
        xl = x0
        for l in range(cross_layers):
            wl = tf.reshape(Cross_W[l],shape=[-1,1])                       # (F*K)
            xlw = tf.matmul(xl, wl)                                        # None
            xl = x0 * xlw + xl + Cross_B[l]                                # None
```

Deep Network部分：

```
     with tf.variable_scope("Deep-Network"):
         if mode == tf.estimator.ModeKeys.TRAIN:
             train_phase = True
         else:
             train_phase = False

         x_deep = x0
         for i in range(len(deep_layers)):
             x_deep = tf.contrib.layers.fully_connected(inputs=x_deep, num_outputs=
                 weights_regularizer=tf.contrib.layers.l2_regularizer(l2_reg), scop
             if mode == tf.estimator.ModeKeys.TRAIN:
                 x_deep = tf.nn.dropout(x_deep, keep_prob=dropout[i])
```

最后拼接两部分得到最后的输出：

```
with tf.variable_scope("DCN-out"):
        x_stack = tf.concat([xl, x_deep], 1)    # None * ( F*K+ deep_layers[i])
        y = tf.contrib.layers.fully_connected(inputs=x_stack, num_outputs=1, activa
        y = tf.reshape(y,shape=[-1])
        pred = tf.sigmoid(y)
```

完整的代码与论文见github： https://github.com/gutouyu/ML_CIA

记得star哦~