推荐系统入门系列(七)-Deep&Cross Network理论与实战

何无涯 何无涯的技术小屋 7月15日

精诚所至,金石为开!

—— 无名

一、Deep&Cross Network算法思想

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的缺点在于隐式地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）。那么能不能用DNN显示地来做高阶的特征组合呢?

Deep&Cross Network就是第一个这么做的。Deep&Cross Network，简称为DCN，是谷歌和斯坦福大学在2017年提出的用于广告CTR的模型，为了简单，后面都是用DCN。

二、Deep&Cross Network模型结构

DCN的架构图如下图所示，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。其中最重要的是Cross Network这一部分，待会儿详细会说道。

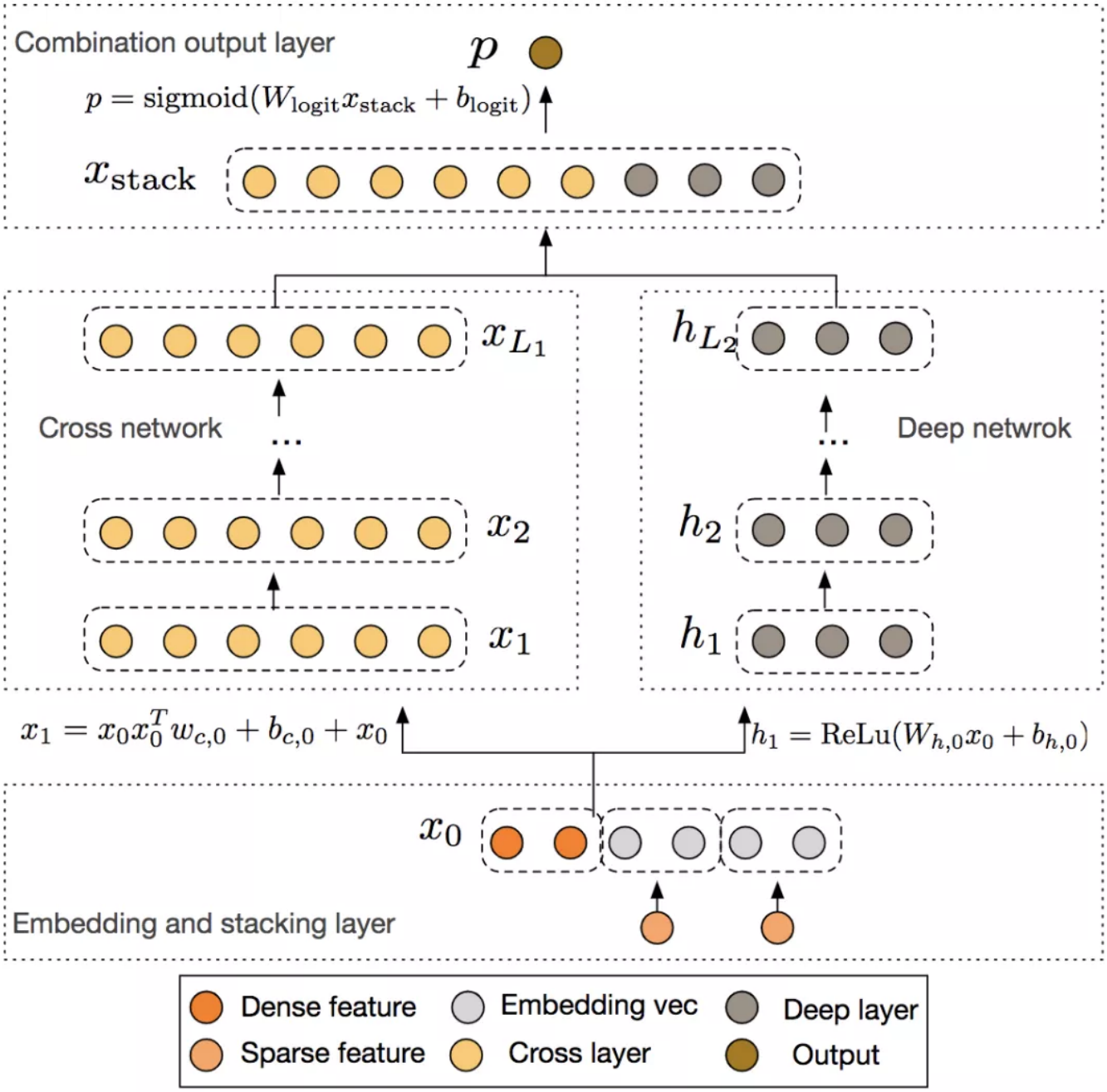

Figure 1: The Deep & Cross Network

接下来将详细看一下DCN每一层的具体做法。

假设CTR任务中的输入中有离散型特征和连续型特征，对于离散型的特征通常就是使用one-hot，但是one-hot之后输入的特征维度非常高而且非常稀疏。所以就有了Embedding来大大降低输入的维度，将这些稀疏特征编码为低维的稠密向量。

注意这里还有一个stacking layer，为什么要stack呢？因为输入中还有连续型特征，这里将连续型特征规范化后和离散型特征的嵌入向量stacking到一起，就得到了原始的输入x0。

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

Cross Network是DCN最为核心的部分。它被设计用来显示地高效地学习高阶的组合特征，它的形式化表示如下：

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

注意到上面式子中都是列向量，w也是列向量。这个公式是什么意思呢？其实很好理解，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可视化如下图所示：

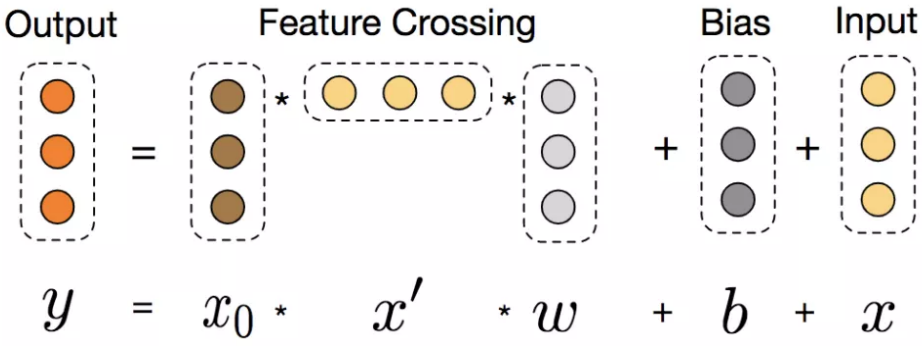

Figure 2: Visualization of a cross layer.

另外Cross Network的层数越深，能学习到的交叉组合特征的阶数也更高。相对于输入x0来说，一个l层的cross Network的交叉组合特征的阶数为l+1。

而且Cross Network的时间复杂度是线性的。假设一共有L层的Cross Layer，其中输入x0的维度为d，那么每一层的W和b也都是d维度的，所以整个cross network的参数个数为dxLx2！！！可以看到，Cross Network的时间复杂度是线性的，和DNN是一个级别的。

然而Cross Network也有其弊端，正是因为它的参数量比较少导致它的表达能力受限，为了能够学习高阶的非线性的组合特征，DCN并行引入了Deep Network。这一部分没啥好说的，就是一个前向传播的全连接神经网络。

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：

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

最后，损失函数使用的是带正则项的对数损失函数，形式化如下：

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

而且网络Cross Network和Deep Network是一起训练的。

其实这里可以看一下DCN和FM模型之间的一些联系，DCN其实和FM模型一样都使用了参数共享机制，只不过相比而言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，怎么理解？

> In a FM model, feature x_i is associated with a weight vector $\mathbf{v}_i$, and the weight of cross term $x_i x_j$ is computed by $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$. In DCN, x_i is associated with scalars $\{w_k^{(i)}\}_{k=1}^{l}$, and the weight of $x_i x_j$ is the multiplications of parameters from the sets $\{w_k^{(i)}\}_{k=0}^{l}$ and $\{w_k^{(j)}\}_{k=0}^{l}$. Both models have each feature learned some parameters independent from other features, and the weight of a cross term is a certain combination of corresponding parameters.

因此和FM模型一样，DCN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强。

三、Deep&Cross Network实现

DCN模型使用Pytroch实现的代码如下：

```
class DeepCrossNetworkModel(nn.Module):
    """
    A pytorch implementation of Deep & Cross Network.
    Reference:
        R Wang, et al. Deep & Cross Network for Ad Click Predictions, 2017.
    """

    def __init__(self, field_dims, embed_dim, num_layers, mlp_dims, dropout):
        super().__init__()
        self.embedding = FeaturesEmbedding(field_dims, embed_dim)
        self.embed_output_dim = len(field_dims) * embed_dim
        self.cn = CrossNetwork(self.embed_output_dim, num_layers)
        self.mlp = MultilayerPerception(self.embed_output_dim, mlp_dims, drop
        self.linear = torch.nn.Linear(mlp_dims[-1] + self.embed_output_dim, 1

    def forward(self, x):
        """
        :param x: Long tensor of size ``(batch_size, num_fields)``
        """
        embed_x = self.embedding(x).view(-1, self.embed_output_dim)
        x_l1 = self.cn(embed_x)
        h_l2 = self.mlp(embed_x)
```

```
        x_stack = torch.cat([x_l1, h_l2], dim=1)
        p = self.linear(x_stack)
        return torch.sigmoid(p.squeeze(1))
```

其中核心代码Cross Network如下：

```
class CrossNetwork(nn.Module):
    '''Cross Network'''
    def __init__(self, input_dim, num_layers):
        super().__init__()
        self.num_layers = num_layers
        self.w = nn.ModuleList([
            nn.Linear(input_dim, 1, bias=False) for _ in range(num_layers)
        ])
        self.b = torch.nn.ParameterList([
            nn.Parameter(torch.zeros((input_dim,))) for _ in range(num_layers
        ])

    def forward(self, x):
        """
        :param x: Float tensor of size ``(batch_size, num_fields, embed_dim)`
        """
        x0 = x
        for i in range(self.num_layers):
            xw = self.w[i](x)
            x = x0 * xw + self.b[i] + x
        return x
```

完整的代码可以参考我的github: https://github.com/yyHaker/RecommendationSystem。

四、小结

FM模型是一种非常浅的结构，并且限制在表达二阶组合特征上，DCN更多的可以看作是FM模型的一种泛化，将FM参数共享的思想从一层扩展到多层，并且学习到高阶的特征组合，而且参数是随着输入维度的增长是线性增长的，与网络的深度也是线性的。而且，DCN相比与传统的DNN，损失函数更低。

参考：

【1】Deep & Cross Network for Ad Click Predictions (2017)