【论文导读】NFM---FM与DNN相结合，附TF2.0复现代码

原创 潜心 推荐算法的小齿轮 8月2日

收录于话题
#推荐 1352 #Github 84 #CTR 18 #推荐系统论文与复现 17

前言

本次分享一篇2017年由何向南教授发表的《Neural Factorization Machines for Sparse Predictive Analytics》。何向南教授的很多文章我都读过，NCF、ONCF等。本篇文章提出了一个模型---NFM。利用神经网络作为隐藏层代替了FM的特征二阶交互的部分，提高了模型的性能。文章末尾也给出了该模型的复现代码。

本文约1.5k字，预计阅读10分钟。

NFM

NFM（lNeural Factorization Machine）是2017年由新加坡国立大学的何向南教授等人在SIGIR会议上提出的模型。NFM是对FM的改进。作者认为FM或者其扩展，例如FFM，依旧只是一个二阶特征交叉的模型：

> 原文描述：However, these variants are all linear extensions of FM and model the second-order feature interactions only.

因此，作者将FM与深度神经网络相结合，利用FM的长处和DNN的特征交互的能力，来构建性能更为优越的模型---NFM。
即将FM模型的二阶交叉部分：

$$y_{FM} = \langle w, x \rangle + \sum_{j_1=1}^{d} \sum_{j_2=j_1+1}^{d} \langle V_i, V_j \rangle x_{j_1} \cdot x_{j_2}$$

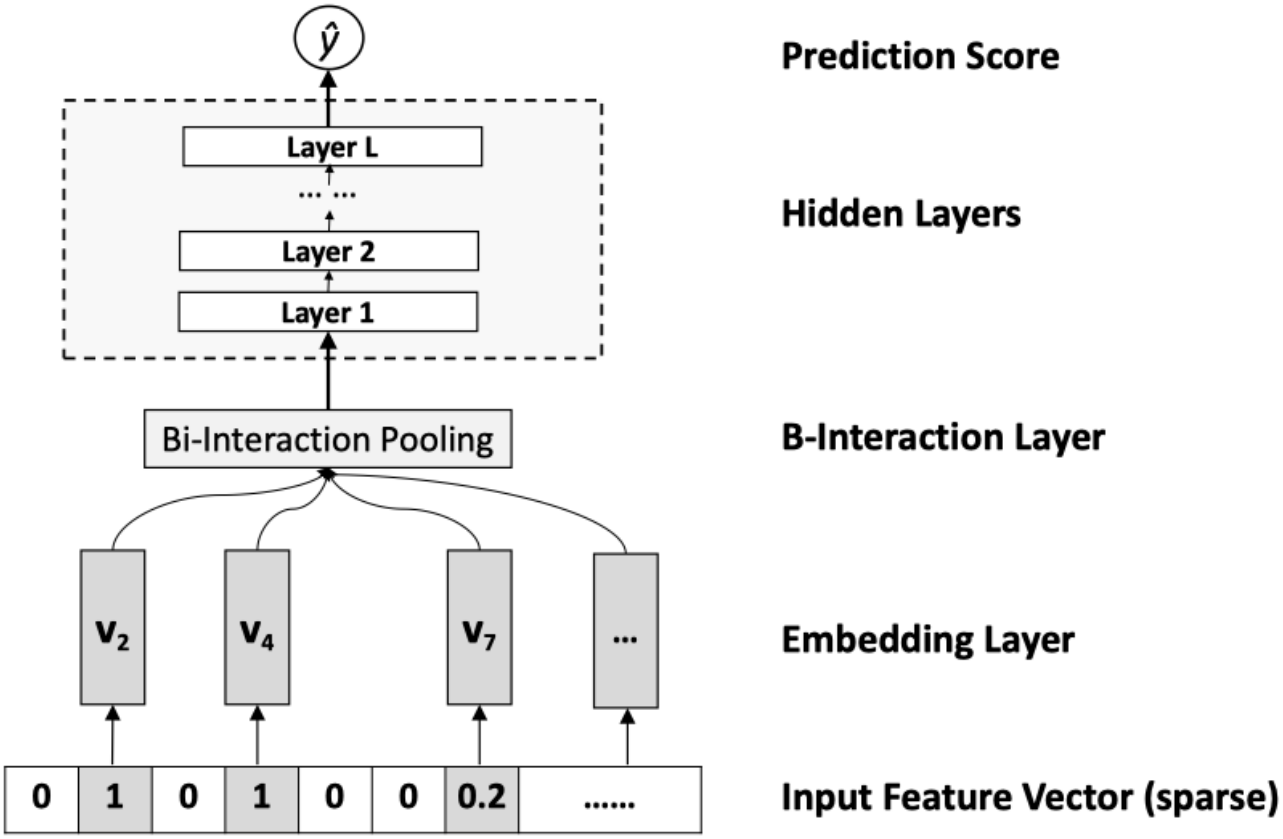
更改为：

$$y_{NFM} = \langle w, x \rangle + f(x)$$

其中$f(x)$可以由某个神经网络结构进行替代。

模型构建

模型整体结构如图所示：



包括：输入层、Embedding层、特征交叉池化层、隐藏层和输出预测层。最主要的层是特征交叉池化层。将多个embedding向量进行一个池化操作。

Input与Embedding层

输入层的特征为了方便，文章指定了稀疏离散特征（当然在实际场景的应用中，分为数值特征与分类特征）。Embedding层（在该模型中其实就是一个全连接层）将高维的稀疏特征转化为低维的密集特征表示。

此处指定$\mathbf{v_i} \in \mathbb{R}^k$为第$i$个特征的embedding向量。用$\mathcal{V}_x = \{x_1\mathbf{v}_1, \ldots, x_n\mathbf{v}_n\}$来表示输入特征。

Bi-Interaction 层

改层是将embedding集合$\mathcal{V}_x$通过池化操作转化为一个向量，即：

$$f_{BI}(\mathcal{V}_x) = \sum_{i=1}^{n}\sum_{j=i+1}^{n} x_i\mathbf{v}_i \odot x_j\mathbf{v}_j$$

其中x_i表示的是不为0的输入特征，⊙定义为两个向量对应元素相乘，这便定义了在embedding空间特征的二阶交互，因此最后的输出是一个k维向量。

作者指出，Bi-Interaction层不需要额外的模型学习参数，更重要的是它在一个线性的时间内完成计算，即时间复杂度为$O(kN_x)$，N_x为embedding向量的数量。参考FM，可以将上式转化为：

$$f_{BI}(\mathcal{V}_x) = \frac{1}{2}\left[\left(\sum_{i=1}^{n} x_i \mathbf{v}_i\right)^2 - \sum_{i=1}^{n} (x_i \mathbf{v}_i)^2\right]$$

所以复杂度为$O(kN_x)$。

隐藏层

即由多个全连接层构成。不过在此之前，作者还采用了**Dropout****和Batch Normalization**，原因如下：

- Dropout：是神经网络的正则化技术，为了防止过拟合；
- Batch Normalization：该层是对输入的每个小批量（min-batch）标准化为零均值的单位方差的高斯分布（zero-mean unit-variance Gaussian distribution）。**作者使用BN，是为了避免embedding向量的更新将输入层的分布更改为隐藏层或输出层；**

输出层

通过逻辑回归将隐藏层的向量转变为最后预测的结果。

创新点

文章最主要的创新点集中于**Bi-Interaction池化部分来代替其他模型使用的拼接**。对比于Wide&Deep模型与Deep Crossing模型完全依赖深度学习有意义的特征交互，NFM使用特征交叉池化的方式捕获了较低级别的二阶特征交互，这比拼接提供的信息更多。这极大地促进了NFM的后续隐藏层以更容易的方式学习有用的高阶特征交互。

代码复现

论文的模型结构较为简单，模型如下：

```
class NFM(keras.Model):
    def __init__(self, feature_columns, hidden_units, dropout_rate):
        super(NFM, self).__init__()
        self.dense_feature_columns, self.sparse_feature_columns = feature_columns
        self.embed_layers = {
            'embed_' + str(i): Embedding(input_dim=feat['feat_num'], input_length=1,
```

```
                                        output_dim=feat['embed_dim'], embeddings_initializer='rand
            for i, feat in enumerate(self.sparse_feature_columns)
        }
        self.dropout = Dropout(rate=dropout_rate)
        self.bn = BatchNormalization()
        self.concat = Concatenate(axis=-1)
        self.dnn_network = [Dense(units=unit, activation='relu') for unit in hidden_units]
        self.dense = Dense(1)

    def call(self, inputs):
        # Inputs layer
        dense_inputs, sparse_inputs = inputs
        # Embedding layer
        embed = [self.embed_layers['embed_{}'.format(i)](sparse_inputs[:, i]) for i in range(sparse
        embed = tf.transpose(tf.convert_to_tensor(embed), [1, 0, 2])
        # Bi-Interaction Layer
        embed = 0.5 * (tf.pow(tf.reduce_sum(embed, axis=1), 2) -
                       tf.reduce_sum(tf.pow(embed, 2), axis=1))
        # Concat
        x = self.concat([dense_inputs, embed])
        # Dropout
        x = self.dropout(x)
        # BatchNormalization
        x = self.bn(x)
        # Hidden Layers
        for dnn in self.dnn_network:
            x = dnn(x)
        outputs = tf.nn.sigmoid(self.dense(x))
        return outputs
```

模型具体代码见：https://github.com/BlackSpaceGZY

总结

NFM融入了深度学习的内容，对FM进行改进。特征交叉池化层对embedding向量进行低阶的特征交互，提高了神经网络的高阶特征交叉的能力。本人最近建立了一个开源项目： Recommender System with TF2.0 ---使用TF2.0对经典的推荐论文进行复现，欢迎大家star和fork。地址：https://github.com/BlackSpaceGZY