NFM:Neural Factorization Machines for Sparse Predictive Analytic

知乎 AI科技时讯 4天前

1、前言

FM能够有效的发现二阶组合特征，但存在的问题在于，FM捕获的二阶组合特征是线性组合的（其表达式就是线性组合），无法捕获非线性组合特征。现在深度神经网络可以发现非线性的组合特征，例如谷歌的Wide&Deep，微软的DeepCross，但对于这些深度网络，存在的缺点是很难训练。**本文提出NFM模型，其能将FM模型捕获的二阶线性组合特征以及神经网络捕获的高阶非线性组合特征组合起来。NFM比FM更具表现力，因为FM可以被看作是NFM不含隐藏层的特例。**

2、Introduction

组合特征(cross feature)，例如occupation = {banker,doctor}， gender = {M,F}则其组合特征为occupation_ender = {banker_M,banker_F,doctor_M,doctor_F}。
FM能够很好的捕获二阶组合特征，但其缺点是它毕竟还是属于线性模型，它的表达能力受限，而且它只能对二阶组合特征进行建模。**NFM针对FM的缺点，在二阶特征组合的隐向量空间中，引入了非线性变换来提升模型非线性表达能力；同时，也可以学习到高阶的组合特征。**

3、MODELLING FEATURE INTERACTIONS

最近几年，embedding-based方法越来越受欢迎，**通过把高维稀疏的输入embed到低维度的稠密的隐向量空间中，模型可以学习到训练集中没有出现过的特征组合。**大致分为两类：

factorization machine-based linear models
neural network-based non-linear models

3.1、FM

公式：

$$\hat{y}_{FM}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbf{v}_i^T \mathbf{v}_j \cdot x_i x_j,$$

FM很好的解决了高纬度高稀疏输入特征组合的问题。通过隐向量内积来建模权重，针对在训练集中没有出现过的特征组合或者出现次数很好的特征组合，也能有效的学习。
参数：

$$\theta \in \{w_0, \{w_i\}, \{v_{if}\}\}.$$

，表达式可写作：y(x) = g + hθ

3.2、DNN

业内大部分DNN的架构都是：把特征的嵌入向量简单拼接起来，输入到神经网络中学习。这样简单的拼接嵌入向量，因为缺失了很多组合特征的信息效果并不好，那么只能寄希望于后面的MLP可以弥补不足。但是为了提高NN的学习能力就需要增加网络层数，复杂的网络结构会收到诸如梯度消失/爆炸、过拟合、degradation(随着网络层数的增加，训练准确率不升反降，非常反常)等问题的困扰，网络的学习或者优化会非常困难。

如果不对嵌入层预训练，Wide&Deep和DeepCross的性能比FM还差，而且DeepCross严重过拟合，Wide&Deep遇到了degradation问题。
如果使用FM预训练初始化嵌入层，Wide&Deep和DeepCross性能都提升了，甚至超过了FM。Wide&Deep的degradation问题也解决了，因为训练集的性能得到了提升。但是两者依旧都有过拟合的问题。实验说明DNN的训练学习真的存在困难。

NFM丢弃了直接把embedding vector拼接输入到神经网络的做法，而是采用在embedding层后增加了Bi-Interaction操作来对二阶组合特征进行建模。这使得low level的输入表达的信息更加的丰富，极大的提高了后面隐藏层学习高阶非线性组合特征的能力。

4、Neural Factorization Machines

结构图：

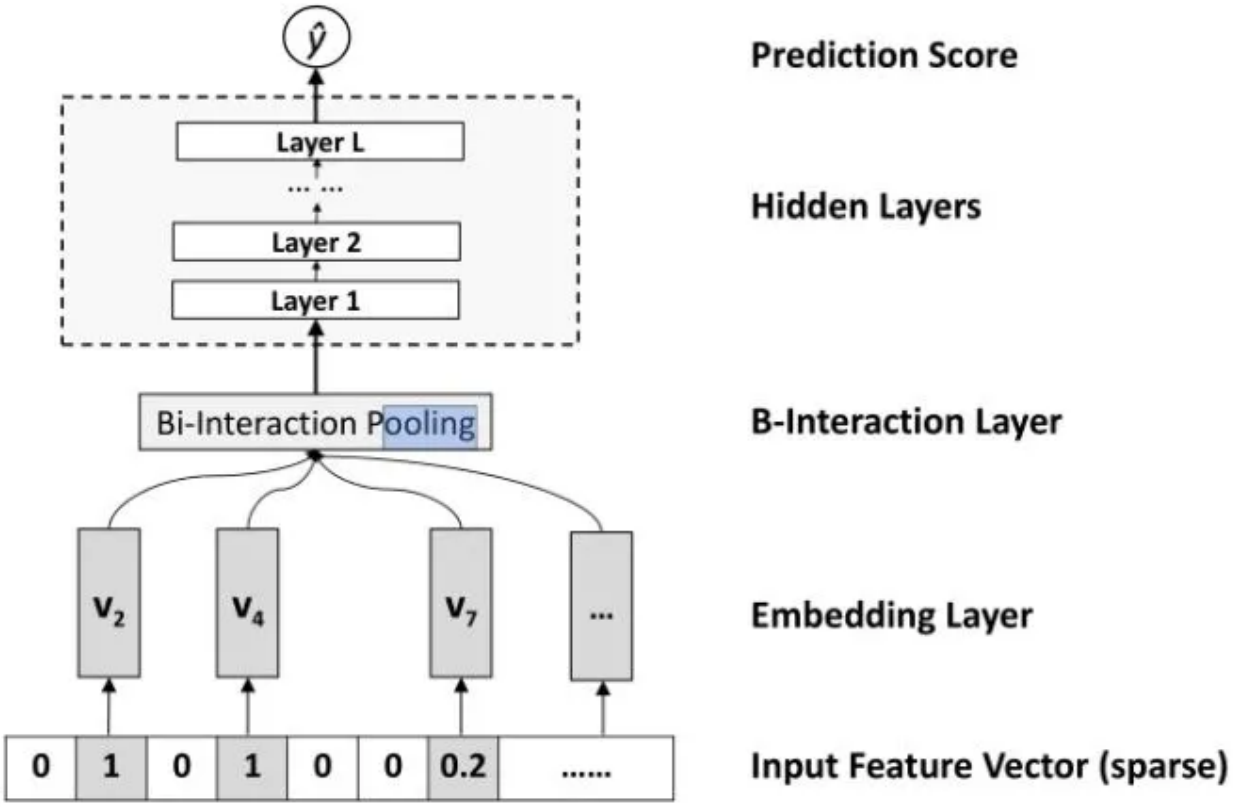

Figure 2: Neural Factorization Machines model (the first-order linear regression part is not shown for clarity).

公式：

$$\hat{y}_{NFM}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + f(\mathbf{x}),$$

其中f(x)是NFM的核心，用来学习二阶组合特征和高阶的组合特征模式。前面两项为线性回归部分，与FM相似。

4.1、Embedding Layer

Embedding将输入转换到低维度的稠密的嵌入空间中进行处理。然后与原始特征进行相乘得到Embedding vector，Vx = {x1**v**1,...,xn**v**n}，而不是简简单单的使用embedding_lookup。

4.2、Bi-Interaction Layer

把embedding vector喂给Bi-Interaction Layer，采用了类似pooling操作，把多个向量转换成一个向量，形式化如下：

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

对公式的理解：fBI的输入是整个的embedding向量，xi xj是特征取值，vi vj是特征对应的embedding向量。中间的操作表示对应位置相乘。所以原始的embedding向量任意两个都进行组合，对应位置相乘结果得到一个新向量；然后把这些新向量相加，就得到了Bi-Interaction的输出。这个输出只有一个向量。上式可参考FM的优化方法得到：

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

它的计算复杂度是O(NK)，其中k是嵌入向量的维度，N是输入x中非零特征的个数。

Bi-Interaction Layer实现了对二阶组合特征的建模，但是又没有引入额外的开销，包括参数数量和计算复杂度。

4.3、Hidden Layers

表达式：

$$\begin{aligned}
\mathbf{z}_1 &= \sigma_1(\mathbf{W}_1 f_{BI}(\mathcal{V}_x) + \mathbf{b}_1), \\
\mathbf{z}_2 &= \sigma_2(\mathbf{W}_2 \mathbf{z}_1 + \mathbf{b}_2), \\
&\cdots\cdots \\
\mathbf{z}_L &= \sigma_L(\mathbf{W}_L \mathbf{z}_{L-1} + \mathbf{b}_L),
\end{aligned}$$

其中Zi表示的是隐藏层，以此来学习高阶组合特征。

隐藏层结构图类型（每个隐藏层大小）有tower，constant，diamond等，一般选用constant的效果要好一些。**本文实验中NFM使用一个隐藏层得到了最好的效果。**

4.4、Prediction Layer

最后一层隐藏层zL到输出层最后预测结果公式如下：

$$f(\mathbf{x}) = \mathbf{h}^T \mathbf{z}_L,$$

其中h是中间的网络参数。

综上NFM的表达式总结如下：

$$\hat{y}_{NFM}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \mathbf{h}^T \sigma_L(\mathbf{W}_L(\ldots\sigma_1(\mathbf{W}_1 f_{BI}(\mathcal{V}_x) + \mathbf{b}_1)\ldots) + \mathbf{b}_L),$$

参数：

$$\Theta = \{w_0, \{w_i, \mathbf{v}_i\}, \mathbf{h}, \{\mathbf{W}_l, \mathbf{b}_l\}\}$$

对比FM，NFM模型的参数主要是{**Wl,bl**}，这个是学习高阶组合特征的参数。相比于FM其实多出的参数就是隐藏层的参数。**FM也可以看做是一个神经网络架构，就是去掉隐藏层的NFM**。我们把去掉隐藏层的NFM称为NFM-0，形式化如下：

$$\hat{y}_{NFM-0} = w_0 + \sum_{i=1}^{n} w_i x_i + \mathbf{h}^T \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i \mathbf{v}_i \odot x_j \mathbf{v}_j$$

$$= w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{f=1}^{k} h_f v_{if} v_{jf} \cdot x_i x_j.$$

如果h为全1向量，那么此时NFM就是FM。

这是第一次把FM看做是神经网络来处理，这样的观点对于优化FM提供了一些新的思路。同时，像NN中常用的技巧也可以应用到这里面来，比如Dropout，实验发现在正则化FM的时候，使用Dropout比传统的L2正则化还要有效。

5、NFM vs Wide&Deep、DeepCross

最重要的区别就在于Bi-Interaction Pooling。Wide&Deep和DeepCross都是用拼接操作(concatenation)替换了Bi-Interaction，并且隐藏层运用tower结构的MLP（或残差单元）。

> Concatenation操作的最大缺点就是它并没有考虑任何的特征组合信息，所以就全部依赖后面的MLP去学习特征组合，但是很不幸，MLP的学习优化非常困难。
> 使用Bi-Interaction考虑到了二阶特征组合，使得输入的表示包含更多的信息，减轻了后面MLP部分的学习压力，所以可以用更简单的模型，取得更好的成绩。

6、Learning

6.1、目标函数

NFM可以用于分类、回归、ranking问题，对应着不同的目标函数。

回归：square loss
分类：Hinge Loss 或 log loss
ranking：Contrastive max-margin loss

本中以回归问题为例，使用square loss，表达式如下：

$$L_{reg} = \sum_{\mathbf{x} \in \mathcal{X}} (\hat{y}(\mathbf{x}) - y(\mathbf{x}))^2,$$

这里并没有正则化项，因为作者发现在NFM中使用Dropout能够得到更好的效果。

6.2、参数估计

使用mini-batch Adagrad来进行参数估计，Adagrad是SGD的变体，特点是每个参数都有自己的学习速率。然后让参数沿着目标函数负梯度的方向进行更新，是下降最快的方向，形式化如下：

$$\theta = \theta - \eta \cdot 2(\hat{y}(\mathbf{x}) - y(\mathbf{x})) \frac{d\hat{y}(\mathbf{x})}{d\theta},$$

Bi-Interaction在求梯度的做法：

$$\frac{df_{BI}(\mathcal{V}_x)}{d\mathbf{v}_i} = (\sum_{j=1}^{n} x_j \mathbf{v}_j)x_i - x_i^2 \mathbf{v}_i = \sum_{j=1, j\neq i}^{n} x_i x_j \mathbf{v}_j.$$

所以NFM的训练依旧可以是端到端的训练，只需要把Bi-Interaction插入到网络中即可。

6.3、Dropout

在NFM中，Bi-Interaction的输出后就增加了Dropout操作，随机的丢弃了一部分的输出。随后的MLP同样应用了Dropout。**Dropout在NFM中可以有效的抑制过拟合。**

6.4、Batch Normalization

DNN的训练面临很多问题。其中一个就是协方差偏移(covariance shift)，意思就是：由于参数的更新，隐藏层的输入分布不断的在变化，那么模型参数就需要去学习这些变化，这减慢了模型的收敛速度。

$$BN(\mathbf{x}_i) = \gamma \odot (\frac{\mathbf{x}_i - \mu_B}{\sigma_B}) + \beta,$$

对于隐藏层的输入，BN在mini-batch数据上，把输入转换成均值为0，方差为1的高斯分布。其中的gamma、beta是两个超参数，为了扩大模型的表达能力，如果模型发现不应用BN操作更好，那么就可以通过学习这两个参数来消除BN的影响。NFM中Bi-Interaction Layer的输出就是MLP的第一个输出，包括后面所有隐藏层的输入都需要进行Batch Normalization。**Batch Normalization在NFM中可以加快训练速度。**

如果用FM来pre-train嵌入层，NFM会收敛的非常快，但是NFM最终的效果并没有变好。说明NFM对参数有很好的鲁棒性。

7、总结

NFM主要特点：

引入了Bi-Interaction Layer，采用Pooling操作将多个embedding vector转化成一个vector
加入了NN，用于发现高阶非线性组合特征