计算广告CTR预估系列(九)--NFM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 2018-06-25

计算广告CTR预估系列(九)--NFM模型理论与实践

一、引言

NFM全称Neural Factorization Machine，它用来解决的问题是Sparse Prediction。也就是说，当模型输入特别稀疏而且特征组合对于预测结果非常重要的时候，就可以考虑是用NFM模型。像CTR预估、推荐系统都属于这类问题。另外，公众号整个系列里介绍的模型都是用来处理这样的输入数据的，所以大家要时刻注意各个模型的适用场景、侧重点以及优缺点，活学活用。

一直关注公众号的同学可能已经听烦了，每篇文章开头都是在介绍模型输入特征。针对Sparse Prediction最开始有了FM来对低阶的二阶特征组合建模，后来又引入了DNN来对高阶的非线性组合特征建模，比如Wide&Deep(Google)、DeepCross(微软)、DeepFM(华为+哈工大)、DIN(阿里)、PNN(上交)。当然，还有通过不是DNN的其他方法来对非线性高阶组合特征建模的，比如LR+GBDT(Facebook)、MLR(阿里)。

NFM也是用FM+DNN来对问题建模的，相比于上面提到的这些DNN模型，NFM的特别之处在哪那？

> NFM相比于上面提到的DNN模型，模型结构更浅、更简单(shallower structure)，但是性能更好，训练和调整参数更加容易。
> 上面提到的所有模型，都可以在公众号的历史文章中找到！主页君倾尽心血针对模型使用场景、提出背景、侧重问题特点、模型优缺点以及代码实践进行了总结整理。特别有料！

二、Model Feature Interaction

2.1 介绍

这么多类别特征，特征组合也是非常多的。传统的做法是通过人工特征工程或者利用决策树来进行特征选择，选择出比较重要的特征。但是这样的做法都有一个缺点，就是：**无法学习训练集中没有出现的特征组合**。

最近几年，*Embedding-based*方法开始成为主流，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。

*Embedding-based*大致可以分为两类：

1. factorization machine-based linear models
2. neural network-based non-linear models

下面分别简单介绍下

2.2 FM

FM全称Factorization Machine通过隐向量内积来对每一对特征组合进行建模。形式化为：

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

FM公式

w0是全局偏置；wi是单特征的权重；vi vj代表特征xi xj的隐向量。xi表示一个小特征，原始的类别型特征经过one-hot得到二值特征。比如gender经过one-hot之后，*gender=male*是一个xi，*gender=female*是另一个xi。

2.2.1 优点

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。

2.2.2 缺点

FM的缺点就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。

解释下什么是*线性模型*:
待预测的target是输入参数的线性组合。

形式化如下:
假设输入参数为 $\theta = \{w_0, \{w_i\}, \{v_{if}\}\}$ ，待预测target为y，那么有 $y = g + h\theta$ ，其中g h是theta无关的表达式。

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。

2.3 Deep Neural Network

DNN模型已经在计算机视觉、自然语言处理、语音识别领域取得成功。但是DNN在IR(信息检索)领域的应用并不是很多。原因是IR中的输入太稀疏了。而(1)DNN如果处理稀疏数据研究的并不多；(2)稀疏数据如何在DNN中处理特征组合也不是很清楚。

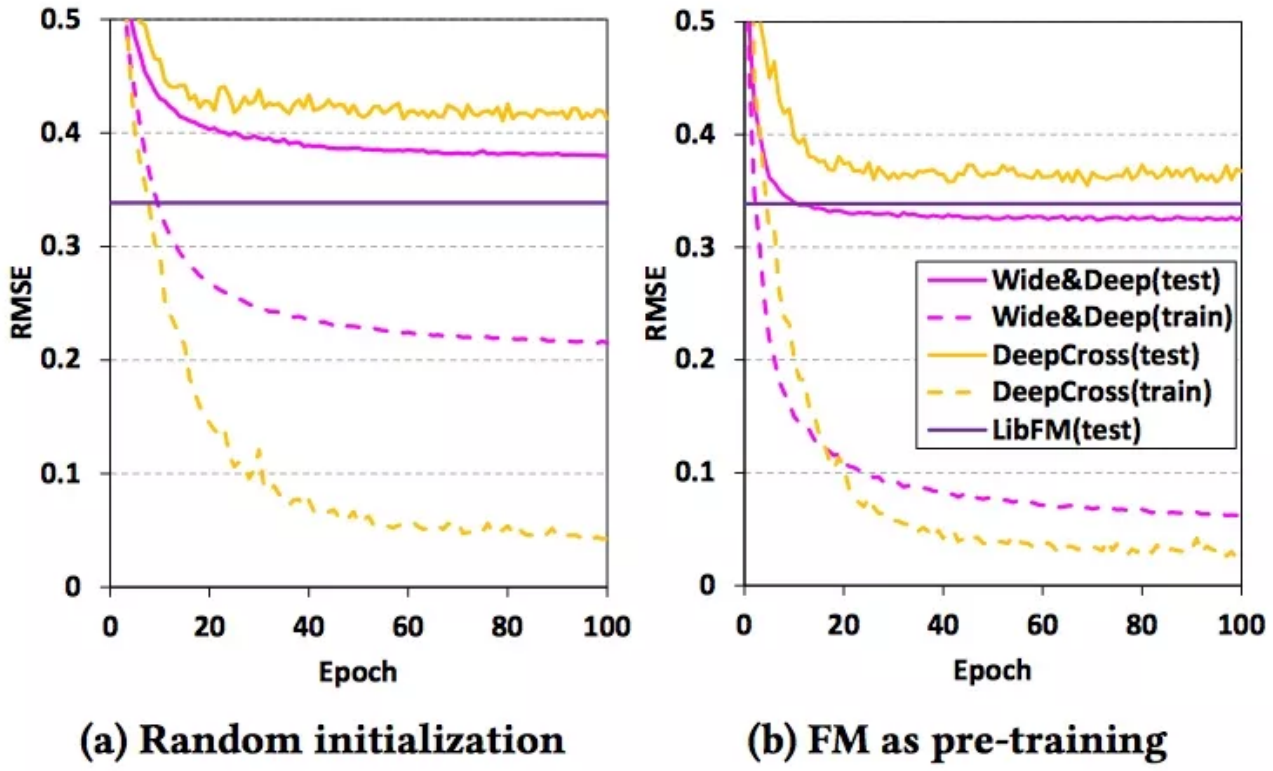
但是，最近业内也开始了各种尝试。比如Google的Wide&Deep，微软的DeepCross，以及FNN、PNN等。Wide&Deep的Deep部分是一个MLP，输入是把特征Embedding vector拼接起来得到的；DeepCross的区别在于NN部分不是MLP，而是使用了*state-of-the-art residual network*。

2.3.1 DNN优化困难

虽然多层神经网络已经被证明可以有效的学习高阶特征组合。但是DNN的缺点也很明显：网络优化或者说网络学习比较困难。
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(*carry too little information about feature interactions in low level*)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*等问题的困扰，网络的学习或者优化会非常困难。

> 解释下degradation problem。概念是Resident Neural Network中作者通过观察提出的，简单说就是：随着网络层数的增加，训练准确率不升反降，非常反常。

为了证明这一点，作者进行了实验：



(a) Random initialization (b) FM as pre-training

Figure 1: Training and test error of each epoch of Wide&Deep and DeepCross on Frappe. Random initialization of embeddings leads to poor performance, worse than LibFM (a). Initializing using the embeddings learned by FM improves the performance significantly (b). This reveals optimization difficulties for training deep neural networks.

DNN训练具有挑战

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摒弃了直接把嵌入向量拼接输入到神经网络的做法，在嵌入层之后增加了*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。

三、NFM

3.1 NFM模型

架构图如下：

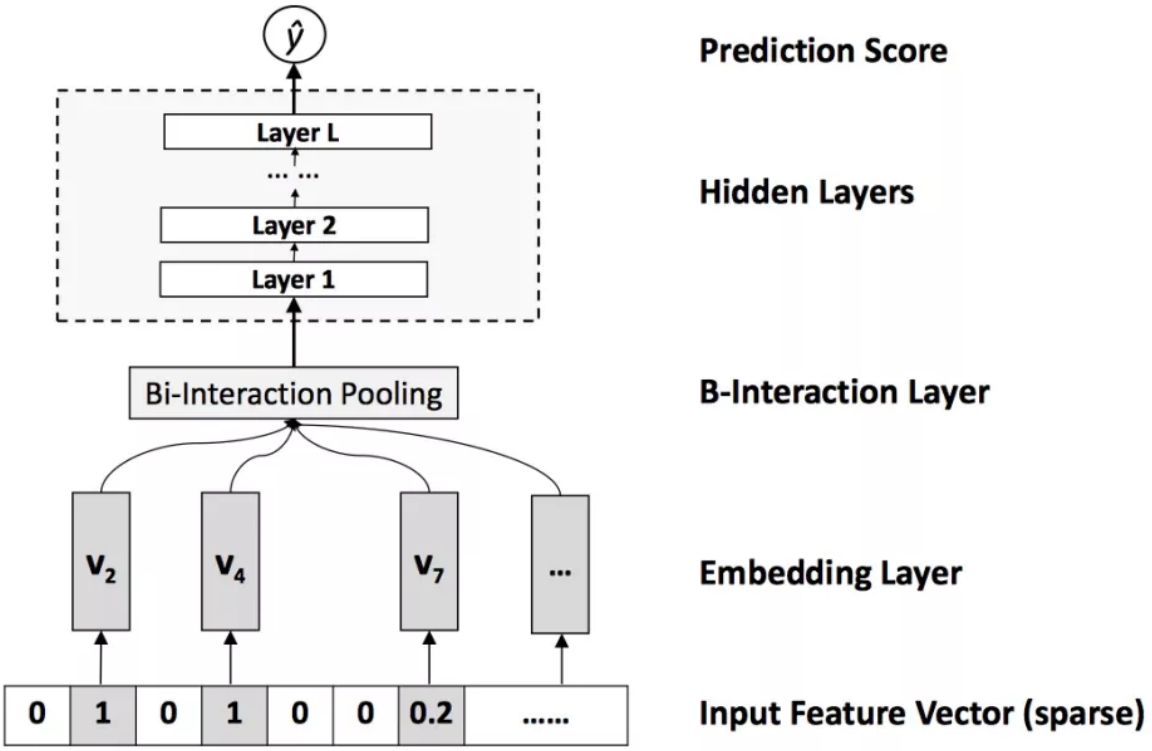


Figure 2: Neural Factorization Machines model (the first-order linear regression part is not shown for clarity).

NFM架构图

NFM可以形式化如下：

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

NFM公式

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。

3.1.1 Embedding Layer

和其他的DNN模型处理稀疏输入一样，Embedding将输入转换到低维度的稠密的嵌入空间中进行处理。作者稍微不同的处理是，*使用原始的特征值乘以Embedding vector*，使得模型也可以处理real valued feature。

3.1.2 Bi-Interaction Layer

Bi是Bi-linear的缩写，这一层其实是一个*pooling层操作，它把很多个向量转换成一个向量*，形式化如下：

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

Bi-linear Interaction

fBI的输入是整个的嵌入向量，xi xj是特征取值，vi vj是特征对应的嵌入向量。中间的操作表示*对应位置相乘*。所以原始的嵌入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。

需要说明的是：Bi-Interaction并没有引入额外的参数，而且它的计算复杂度也是线性的，和max/min pooling以及原来的拼接操作复杂度都是相同的。因为上式可以参考FM的优化方法，化简如下(想想一个矩阵)：

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

BI化简公式

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。这个公式大家要搞懂是怎么回事哦，代码里面就是按照这个来写的。

总结，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。

3.1.3 Hidden Layer

这个跟其他的模型基本一样，堆积隐藏层以期待来学习高阶组合特征。模型结构可以参考Wide&Deep论文的结论，一般选用constant的效果要好一些。

3.3.4 Prediction Layer

最后一层隐藏层zL到输出层最后预测结果形式化如下：

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

DNN最后的输出

其中h是中间的网络参数。考虑到前面的各层隐藏层权重矩阵，f(x)形式化如下：

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

NFM形式化公式

这里的参数为 $\theta = \{w_0, \{w_i, v_i\}, h, \{W_l, b_l\}\}$ ，相比于FM其实多出的参数就是隐藏层的参数，所以说*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：

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

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

0-Hidden Layer NFM

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。

3.2 NFM vs Wide&Deep、DeepCross

> 如果要求简明扼要说明NFM到底做了什么，就是这一段！

最重要的区别就在于Bi-Interaction Layer。Wide&Deep和DeepCross都是用拼接操作(concatenation)替换了Bi-Interaction。

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。

3.3 复杂度分析

上面提到过，NFM相比于FM，复杂度增加在MLP部分。所以NFM的复杂度和Wide&Deep、DeepCross是相同的，形式化如如下：

$$O(kN_x + \sum_{l=1}^{L} d_{l-1}d_l)$$

NFM复杂度

3.4 训练

3.4.1 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

论文中以回归问题为例，使用square loss，形式化如下。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。

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

L2正则化

3.4.2 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：

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

参数学习

这里唯一需要指出的是Bi-Interaction在求梯度时是怎么做的：

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

Bi-Interaction导数

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。

3.4.3 Dropout

Dropout在训练过程中随机丢掉一些神经元，那么再一次参数更新中也就只会更新部分参数。可以理解成是相当于很多个小的NN取平均值。增加了模型的抗过拟合能力。*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。
需要注意的是，*在测试阶段，Dropout是不使用的，所有的神经元都会激活。*

3.4.4 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。
Batch Normalization就是为了解决这个问题的，形式化如下：

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

Batch Normalization

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。*
注意，*在测试预测的时候，BN操作同样有效，这时的均值和方差在整个测试集上来进行计算。*

四、实验结果分析

4.1 数据集

使用了两份公开的数据集：

- Frappe。http://baltrunas.info/research-menu/frappe
- MovieLens。https://grouplens.org/datasets/movielens/latest/

随机的取70%数据作训练集，20%数据作验证集，10%数据作测试集。

4.2 实验结果

FM相当于是去掉DNN的NFM，论文中给出的数据是只用了一个隐藏层的NFM，相比于FM性能提升了7.3%；NFM只用了一个隐藏层，相比于3个隐藏层的Wide&Deep，和10个隐藏层的DeepCross，NFM用更简单的模型，更少的参数得到了性能的提升。

论文中对比了FM、Wide&Deep模型，效果不用说肯定是NFM最好，这里就不贴图了，感兴趣的小伙伴可以去原论文中查看。此处，只给出一些重要的结论：

1. Dropout在NFM中可以有效的抑制过拟合
2. Batch Normalization在NFM中可以加快训练速度
3. NFM使用一个隐藏层得到了最好的效果
4. 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。
5. 模型性能基本上随着Factor的增加而提升。Factor指Embedding向量的维度。

五、总结

NFM主要的特点如下：

1. NFM核心就是在NN中引入了Bilinear Interaction(Bi-Interaction) pooling操作。基于此，NN可以在low level就学习到包含更多信息的组合特征。
2. 通过deepen FM来学校高阶的非线性的组合特征

所以，依旧是FM+DNN的组合套路，不同之处在于如何处理Embedding向量，这也是各个模型重点关注的地方。现在来看，如何用DNN来处理高维稀疏的数据并没有一个统一普适的方法，业内依旧在摸索中。

六、代码实战

完整代码参考Github: https://github.com/gutouyu/ML_CIA
不要忘记Star哦~

核心代码如下：

FM中原始特征部分：

```
# FM部分
with tf.variable_scope("Linear-part"):
    feat_wgts = tf.nn.embedding_lookup(Feat_Wgts, feat_ids) # None * F * 1
    y_linear = tf.reduce_sum(tf.multiply(feat_wgts, feat_vals), 1)  # None * 1
```

NFM的核心Bilinear Interaction部分。先得到嵌入向量，然后两两组合得到BI输出结果：

```
with tf.variable_scope("BiInter-part"):
    embeddings = tf.nn.embedding_lookup(Feat_Emb, feat_ids) # None * F * k
    feat_vals = tf.reshape(feat_vals, shape=[-1, field_size, 1]) # None * F * 1
    embeddings = tf.multiply(embeddings, feat_vals) # vi * xi
    sum_square_emb = tf.square(tf.reduce_sum(embeddings, 1))
    square_sum_emb = tf.reduce_sum(tf.square(embeddings), 1)
    deep_inputs = 0.5 * tf.subtract(sum_square_emb, square_sum_emb) # None * k
```

下面的都是MLP的Deep-part的内容：
Bi-Inter的输出是第一个隐藏层的输入，所以需要进行Batch Norm与Dropout操作。需要注意的有两点：

1. 先进行Batch Normalization再进行Dropout
2. Dropout在训练的时候需要，在测试的时候是不用的

```
# BI的输出需要进行Batch Normalization
    if mode == tf.estimator.ModeKeys.TRAIN: # batch norm at bilinear interaction la
        deep_inputs = tf.contrib.layers.batch_norm(deep_inputs, decay=0.9, center=T
    else:
        deep_inputs = tf.contrib.layers.batch_norm(deep_inputs, decay=0.9, center=T
# Dropout
    if mode == tf.estimator.ModeKeys.TRAIN:
        deep_inputs = tf.nn.dropout(deep_inputs, keep_prob=dropout[-1]) # dropout a
```

连接各个隐藏层：

```
for i in range(len(layers)):
    deep_inputs = tf.contrib.layers.fully_connected(inputs=deep_inputs, num_outputs
```

得到MLP最终输出：

```
# Output
    y_deep = tf.contrib.layers.fully_connected(inputs=deep_inputs, num_outputs=1, a
    y_d = tf.reshape(y_deep, shape=[-1])
```

组合FM得到最终结果：

```
with tf.variable_scope("NFM-out"):
    y_bias = Global_Bias * tf.ones_like(y_d, dtype=tf.float32)
```

```
3       y = y_bias + y_linear + y_d
4       pred = tf.sigmoid(y)
```

Reference

1. Neural Factorization Machines for Sparse Predictive Analytics
2. https://zhuanlan.zhihu.com/p/33699909

计算广告CTR预估系列往期回顾

计算广告CTR预估系列(一)--DeepFM理论
计算广告CTR预估系列(二)--DeepFM实践
计算广告CTR预估系列(三)--FFM理论与实践
计算广告CTR预估系列(四)--Wide&Deep理论与实践
计算广告CTR预估系列(五)--阿里Deep Interest Network理论
计算广告CTR预估系列(六)--阿里Mixed Logistic Regression
计算广告CTR预估系列(七)--Facebook经典模型LR+GBDT理论与实践
计算广告CTR预估系列(八)--PNN模型理论与实践

获取更多机器学习干货，关注**机器学习荐货情报局**!

公众号