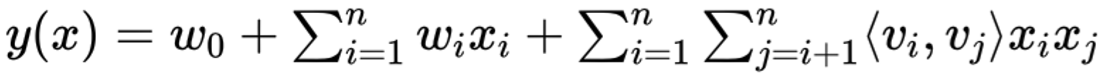
FFM理论与实践

# 背景

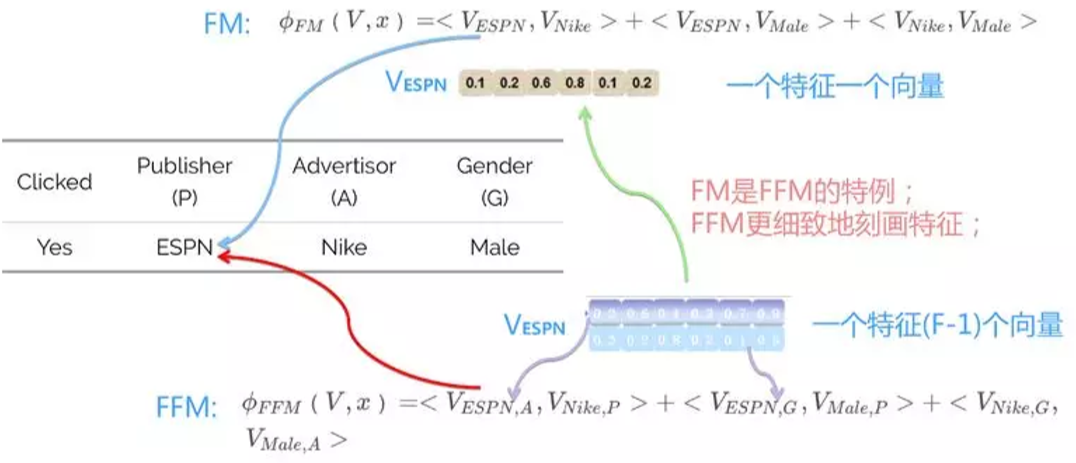
首先，回顾下FM算法，FM的公式如下：



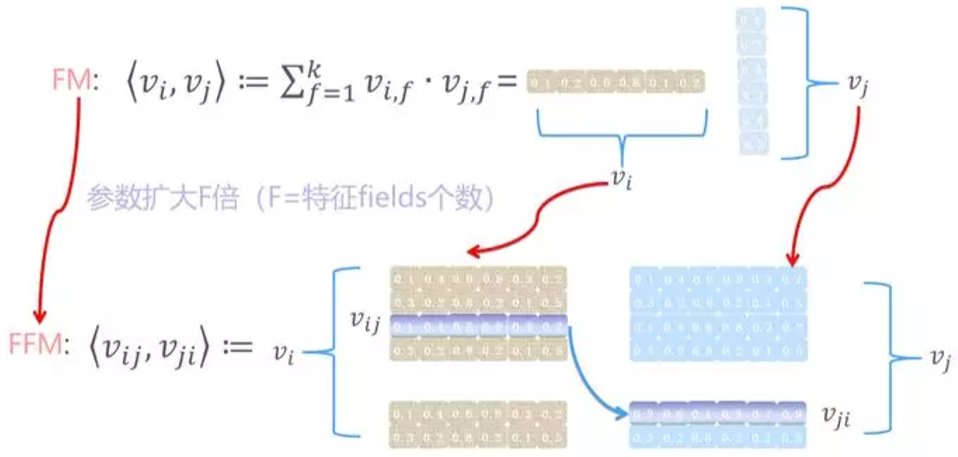
样本x是n维向量，xi是第i个维度上的值。vi是xi对应的长度为k的隐向量，V是模型的参数，所有的样本都使用同一个V，即x1,1和x2,1都使用向量v1。

这里先采用案例的方式介绍大致了解下FM与FFM的区别（也是本人看到这个案例比较详细，并且绘图精致，所以想记录下）

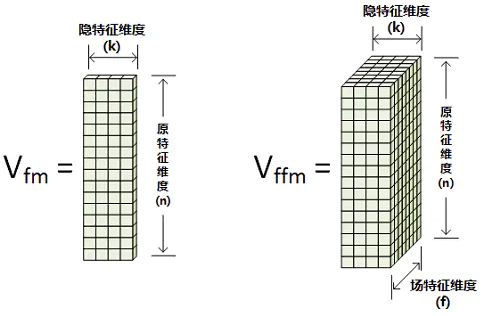
举例：假如有三个特征 fields，这是一个在线投广告的应用场景，比如说我要往 ESPN 这个网站投广告，第一个特征 Publisher是一个网站，第二个特征是广告主（Advertisor） 是谁？本例中假设是 Nike；第三个特征是阅读者（Gender）是男性还是女性？模型需要预测阅读者的性别是男性（male）时；他会不会点击。



从图中我们可以看出，在FM模型中，对任意两个特征进行了组合，它的本质是对把任意一个特征学出一个embedding，从而增强模型的泛化能力。



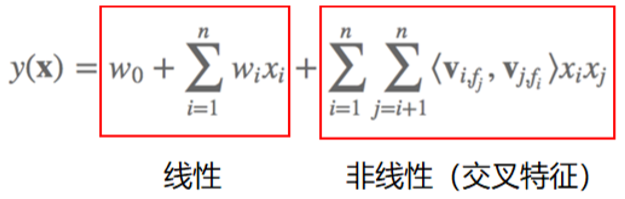
在FFM模型中，更精细的刻画了特征交叉的过程。FFM模型也是对任意两个特征组合，但是区别在于：FM是每个特征采用一个隐向量，但是在本案例中每个特征有两个隐向量，这意味着同一个特征，要和不同的fields进行组合的时候，是用不同的embedding向量去组合。FFM中采用的隐向量多了，就会导致FFM模型的参数量更多。对于一个特征来说，原先是一个 vector，现在会拓成F个vector，F是特征fields的个数，只要有跟它任意组合，就有一个 vector来代表，这就是FFM的基本思想。关于FM和FFM模型隐向量部分的对比可以参考下面的图：



# FFM算法

## FFM算法概述

FFM（Field-aware Factorization Machines），是 FM（Factorization Machines）的改进版。FFM 初的概念来自 Yu-Chin Juan（阮毓钦，毕业于中国台湾大学，现在美国 Criteo 工作）与其比赛队员。通过引入 field 的概念，FFM 把相同性质的特征归于同一个 field。在FFM中，每一维特征xi，针对其它特征的每一种field fj，都会学习一个隐向量Vi,fj。因此，在FFM中，隐向量不仅与特征相关，也与field相关，这也是FFM中“Field-aware”的由来。FFM的结构如下：



线性部分与LR相同，非线性部分是FFM的关键所在，并且与同样有非线性部分的FM算法相比，FFM引入了 field的概念。因此交叉项与FM算法的<vi , vj>不同，这里的是<vi,fj , vj,fi>。

## FFM中的Field

在FFM中，每一维特征feature都归属于一个特定的field，field和feature是一对多关系。下表是一个示例：

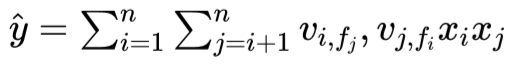
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Field1=age | Field2=city | | | Field3=sex | |
| Feature | x=age | x2=北京 | x3=上海 | x4=深圳 | x5=男 | x6=女 |
| user1 | 23 | 1 | 0 | 0 | 1 | 0 |
| user2 | 25 | 0 | 1 | 0 | 0 | 1 |

对于连续特征，一个特征就对应一个Field，或者对连续特征离散化，把特征分箱成为离散特征，比如对于特征age可以采用分区间的方式离散化（也就是分箱，也称为分桶）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Field1=age | | | |
| Feature | <20 | 20-30 | 30-40 | >40 |
| user1 | 0 | 1 | 0 | 0 |
| user2 | 0 | 0 | 0 | 0 |

对于离散特征，采用one-hot编码，一般把同一个特征one-hot之后的特征归属到一个field，不论是连续特征还是离散特征，都有一个共同点：同一个field下只有一个feature的值不为0，其他feature的值都为0。

FFM 模型认为vi不仅与xi有关系，还与xi相乘的xj所属的field有关系，即对应于FM模型中的vj在FFM模型中变换成了一个二维向量vF,K，其中F是field的总个数，若只保留FFM中的二次项，可以用公式表示如下：



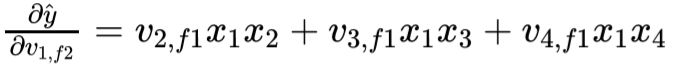
以上文表格中的数据为例，计算user1的y，如下：



由于x2,x3,x4属于同一个field，所以f2,f3,f4可以用同一个变量来代替，比如就用f2，则user1的y可以进一步写成：

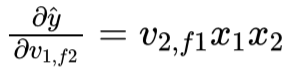


计算y对v1,f2的偏导，如下：

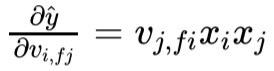


等式两边都是长度为K的向量

由于x2,x3,x4是同一个属性的one-hot表示，所以x2,x3,x4中只有一个是1，其它都是0（如：在本例中x3=x4=0,x2=1），所以



由此可以推广到一般情况：



𝑥𝑗属于Field 𝑓𝑗，且同一个Field中其它𝑥𝑚都等于0。实际项目中𝑥是非常高维的稀疏向量，求导时只关注那些非0项即可。

这里补充下一次项求导的结果：

当参数为w0时，

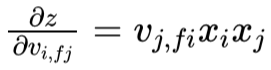
当参数为wi时，

你一定有个疑问：𝑣是模型参数，为了求𝑣我们采用梯度下降法时需要计算损失函数对𝑣的导数，为什么这里要计算𝑦̂对𝑣的导数？

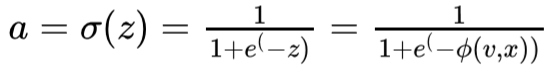
在实际预测点击率的项目中，我们不会直接使用公式的，通常会再套一层 sigmoid 函数，公式中的𝑦̂用z来取代，公式如下：



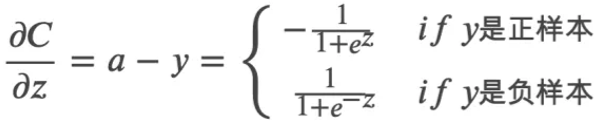
由得：

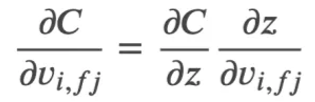


用𝑎表示对点击率的预测值，则：



令y=0表示负样本，y=1表示正样本，C表示交叉熵损失函数。根据神经网络调优公式可得：

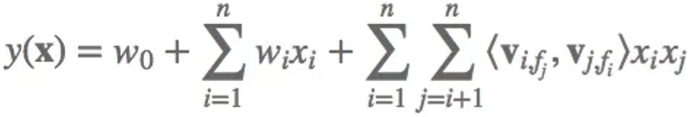




注意：如果以y=1代表正样本，y=-1代表负样本，则

有了上面的推导，我们再来看 FFM 的公式。

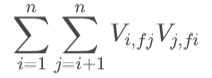
设样本一共有n个特征， f个field，那么FFM的二次项有nf个隐向量。而在FM模型中，每一维特征的隐向量只有一个。FM可以看做FFM的特例，是把所有特征都归属到一个 Field时的FFM模型。根据FFM的Field敏感特性，可以导出其模型方程：



其中， fj是第j个特征所属的字段，如果隐向量的长度为k，那么FFM的二次参数有nfk个，远远多于FM模型的nk个，此外，由于隐向量与Field相关，FFM二次项并不能够化简，时间复杂度是O(kn2)。需要注意的是由于FFM中的laten vector只需要学习特定的Field，所以通常：KFFM << KFM。FFM由于引入了Field，使得每两组特征交叉的隐向量都是独立的，可以取得更好的组合效果，FM可以看做只有一个Field的FFM。

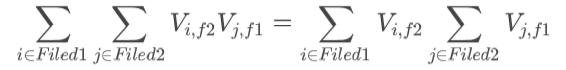
# 加速计算

对于FFM中交叉项部分，当所有的x都是one-hot时，交叉项可以写成：



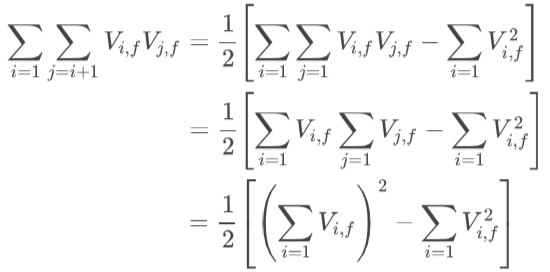
在上面变形的基础上，又需要分两种情况：

## 情况一：i和j不属于同一个field时



对上面的公式变形，可以采用类比的方式理解。比如：a,b,c属于field1，d,e属于field2，则ad + ae + bd + be + cd + ce = (a + b + c) + (d + e)

## 情况二：i和j属于同一个field时



上面的公式变换也可以类比下面这个例子理解。比如：a,b,c属于同一个field，则



此时，原来的乘法计算由原来的O(n2)下降为O(n)，n表示field内有几个特征。

# 实践tricks

## 对数据集的要求

1. 含有类别特征的数据集，且需要对特征进行二值化处理。
2. 越是稀疏的数据集表现效果相较于其他模型更优。
3. FM比较难应用于纯数值类型的数据集。

## 特征处理

1. 连续值离散化

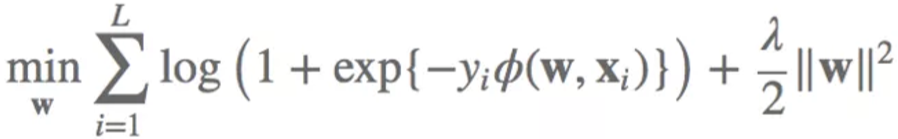
第一种方式是，采用等频分隔或者等间距分隔的方式，先把数据划分到其中某一个区间中，进而one-hot，即数据所在区间置为1，其它置为0（相当于分区间的之后的one-hot）。

第二种方式是，把所有的连续值都当成一个分类变量处理。例如：对于年龄age的连续变量，存在value为[10,19,20,22,22,34]，这种方法就生成了 age\_10,age\_19,age\_20,age\_22,age\_34这些变量，如果连续值一多，这个方法带来的计算量就直线上升。

第三种方式是，采用分箱的方式使连续值离散化。例如：对于一个年龄age的连续变量[10,19,20,22,22,34]，先建立一个map，[0,10):0,[10,20):1,[20,30):2,[30,100):3。原始的数据就变成了[1,1,2,2,2,3]，再进行2的连 续值离散化方法，生成了age\_1,age\_2,age\_3这几个变量，优化了计算量，而且使得结果更具有解释性。

## 损失函数logistic loss

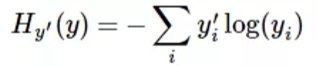
Logistic loss是label是-1/1做二分类的时候常用的loss计算方法，公式如下：



值得注意的是，需要把label拆分成-1/1而不是0/1，当我们预测正确的时候， predlabel>0且越大正确的程度越高，相应的log项是越小的，整体loss越小；相反，如果我们预测的越离 谱，predlabel<0且越小离谱的程度越高，相应的log项是越大的，整体loss越大。

## 交叉熵（tf.nn.softmax\_cross\_entropy\_with\_logits）

TensorFlow内部模块tf.nn.softmax\_cross\_entropy\_with\_logits，是多分类中的损失函数，公式如下：



需要注意的是，若选用这个函数作为损失函数，则需要把label拆分成[1,0]和[0,1]进行计算。

## 归一化

1. 样本归一化。在FFM中需要对样本进行归一化，否则容易造成数据溢出（inf），进而引起梯度计算出现nan。
2. 特征归一化。CTR/CVR模型采用了多种类型的源特征，包括数值型和categorical类型等。但是，categorical类编码后的特征取值只有0和1，较大的数值型特征会造成样本归一化后categorical类生成特征的值非常小，没有区分性。例如，一条用户-商品记录，用户为“男”性，商品的销量是5000个（假设其它特征的值为零），那么归一化后特征“sex=male”（性别为男）的值略小于0.0002，而“volume”（销量）的值近似为1。特征“sex=male”在这个样本中的作用几乎可以忽略不计，这是相当不合理的。因此，将源数值型特征的值归一化到[0, 1]是非常必要的。
3. 先进行特征归一化，再进行样本归一化

## Early stopping

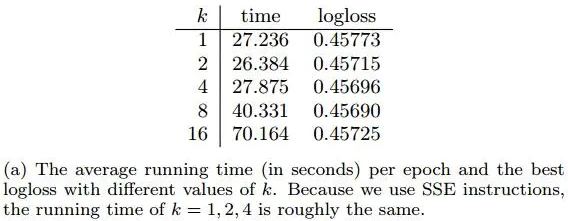
一定要设置早停策略，FFM很容易过拟合。

## 省略零值特征

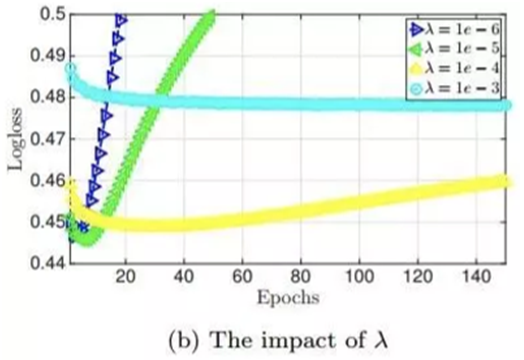
零值特征对模型没有任何贡献，无论是1次项还是2次项都为0

## 论文实践过程中得出的结论

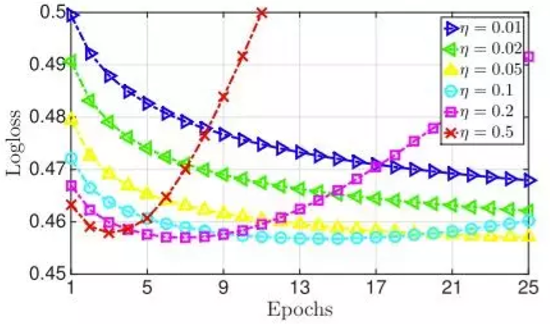
1. FFM的隐向量维度远小于FM的隐向量维。k值不用太大，没啥提升



1. 正则项lambda和学习率alpha需要着重调参。 正则化项系数lambda如果太大，容易导致模型欠拟合，反之，容易过拟合。



1. 在论文中，使用的是Adagrad优化器，全局学习率也是超参数。如果在一个较小的水平，则可以表现最佳。过大，容易导致过拟合。过小，容易欠拟合。如下图：



1. epoch别太大，既会拖慢速度，而且造成过拟合；在原论文中甚至要考虑用early-stopping避免过拟合，所以epoch=1，常规的来讲就可以了。

# FFM优缺点

可参考：<https://github.com/jpegbert/code_study/tree/master/FFM>

# FFM优缺点

FFM优点：

1. 在高维稀疏性数据集中表现很好
2. 相比FM，细化隐向量的表示，同一特征针对不同field使用不同隐向量，模型建模更加准确

FFM缺点：

1. 计算复杂度比较高，参数个数为nfk，时间复杂度为O(k\*n^2)，而FM是O(kn)
2. 参数多容易过拟合，必须设置正则化方法，以及早停的训练策略

# 后记

虽然FFM的效果比FM好，但是由于FFM模型的参数量扩大了F倍，把FFM模型应用于真实场景会由于参数量太大导致时间喝空间的耗费非常大，因此，业界提出了FFM模型的改进版：双线性FFM（Bilinear-FFM）模型。

参考：

<https://www.cnblogs.com/wkang/p/9788012.html>

<https://mp.weixin.qq.com/s/0MqsXXiBcmw9u8h43ooBYw>

<https://www.cnblogs.com/zhangchaoyang/articles/8157893.html>

<https://mp.weixin.qq.com/s/qynApDntA9xiGRBQpoIsjQ>

<https://mp.weixin.qq.com/s/E6EH6aJjzTwN2UZf_4nwoA>

<https://www.jianshu.com/p/8b57473e385a>

<https://mp.weixin.qq.com/s/azxKR2yAVkGqSwRmQ7mGRQ>

<https://mp.weixin.qq.com/s/-p9lTtjEf0B3fUk7MvhDGA>

<https://tech.meituan.com/2016/03/03/deep-understanding-of-ffm-principles-and-practices.html>

<https://mp.weixin.qq.com/s/bdBz1HChUNyLzSkcDISP9Q>