CTR

> Click-through rate (CTR) is the ratio of users who click on a specific link to the number of total users who view a page, email, or advertisement. It is commonly used to measure the success of an online advertising campaign for a particular website as well as the effectiveness of email campaigns.

在计算广告和推荐系统中，CTR预估(click-through rate)是非常重要的一个环节，判断一个商品的是否进行推荐需要根据CTR预估的点击率来进行。准确的估计CTR对于提高流量的价值，增加广告收入有重要的指导作用。在进行CTR预估时，除了单特征外，往往要对特征进行组合。预估CTR，业界常用的方法有人工特征工程 + LR(Logistic Regression)、GBDT(Gradient Boosting Decision Tree) + LR、FM（Factorization Machine）和 FFM（Field-aware Factorization Machine）模型。在这些模型中，FM和FFM近年来表现突出，分别在由Criteo和Avazu举办的CTR预测竞赛中夺得冠军

FM

FM(Factorization Machine)主要是为了解决数据稀疏的情况下，特征怎样组合的问题。

线性回归

在传统的线性回归模型中，对于一个给定的特征向量$X=(x_1,x_2,...,x_n)^T$，线性回归建模时采用的函数是

$$\begin{aligned}\hat{y}(x) &= w_0+w_1x_1+w_2x_2+...+w_nx_n \\ &= w_0+\sum_{i=1}^{n}w_ix_i\end{aligned}$$

可以看出，各特征分量x_i和x_j之间是相互孤立的，即$\hat{y}(x)$中仅考虑单个的特征分量，而没有考虑特征分量之间的相互关系（interaction）

特征组合

所以，我们可以把函数$\hat{y}$改写成

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

这样，便将任意两个（互异）特征分量之间的关系也考虑进来了

不过，这种直接在特征分量$x_i x_j$前直接分配系数w_{ij}的方式，在处理稀疏矩阵时有一个很大的缺陷。

稀疏矩阵即意味着大部分的特征分量是没有出现过交互的，即在矩阵中该位置为0（$x_i = 0$ or $x_j = 0$），这样就不能对相应的参数进行估计

而在高度稀疏的数据场景中，由于数据量的不足，样本中出现未交互的特征分量是很普遍的

辅助向量

因此针对每个维度的特征分量x_i，引入辅助向量

$$v_i = (v_i 1, v_i 2, ..., v_i k)^T \in R^k，\ i = 1, 2, ..., n$$

其中$k \in N^+$是超参数(hyperparameters)

并将w_{ij}改写成

$$\hat{w_{ij}} = {v_i}^T v_j := \sum_{l=1}^{k} v_{il} v_{jl}$$

于是，函数$\hat{y}$可以进一步改写为

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} < {v_i}^T v_j > x_i x_j$$

复杂度分析

对于模型方程，直观地计算复杂度可得

$$n + (n-1) + \{\frac{n(n-1)}{2}[k + (k-1) + 2] + \frac{n(n-1)}{2} - 1\} + 2 = O(kn^2)$$

但可以对方程改写，将其复杂度降为线性的O(kn)，具体推导过程如下：

$$
\begin{aligned}
&\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(\mathbf{v}_i^\top\mathbf{v}_j)x_ix_j \\
=\;& \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}(\mathbf{v}_i^\top\mathbf{v}_j)x_ix_j-\sum_{i=1}^{n}(\mathbf{v}_i^\top\mathbf{v}_i)x_ix_i\right) \\
=\;& \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{k}v_{il}v_{jl}x_ix_j-\sum_{i=1}^{n}\sum_{l=1}^{k}v_{il}^2x_i^2\right) \\
=\;& \frac{1}{2}\sum_{l=1}^{k}\left(\sum_{i=1}^{n}(v_{il}x_i)\sum_{j=1}^{n}(v_{jl}x_j)-\sum_{i=1}^{n}v_{il}^2x_i^2\right) \\
=\;& \frac{1}{2}\sum_{l=1}^{k}\left[\left(\sum_{i=1}^{n}(v_{il}x_i)\right)^2-\sum_{i=1}^{n}v_{il}^2x_i^2\right],
\end{aligned}
$$

于是复杂度变为

$$k\{[n+(n-1)+1]+[3n+(n-1)]+1\}+(k-1)+1=O(kn)$$

one-hot编码

举一个广告分类问题的例子

Clicked?	Country	Day	Ad_type
1	USA	26/11/15	Movie
0	China	1/7/14	Game
1	China	19/2/15	Game

clicked是分类值，表明用户有没有点击该广告。1表示点击，0表示未点击。而country,day,ad_type则是对应的特征。对于这种categorical特征，一般都是进行one-hot编码处理

将上面的数据进行one-hot编码以后，就变成了下面这样：

Clicked?	Country=USA	Country=China	Day=26/11/15	Day=1/7/14	Day=19/2/15	Ad_type=Movie	Ad_type=Game
1	1	0	1	0	0	1	0
0	0	1	0	1	0	0	1
1	0	1	0	0	1	0	1

因为是categorical特征，所以经过one-hot编码以后，不可避免的样本的数据就变得很稀疏。比如假设淘宝或者京东上的item为100万，如果对item这个维度进行one-hot编码，光这一个维度数据的稀疏度就是百万分之一

one-hot编码带来的另一个问题是特征空间变大。同样以上面淘宝上的item为例，将item进行one-hot编码以后，样本空间有一个categorical变为了百万维的数值特征，特征空间就一下子暴增一百万

FFM

field

FFM（Field-aware Factorization Machine）是在FM的基础上发展出来的算法。通过引入field的概念，FFM把相同性质的特征归于同一个field

在上面的广告分类问题中，“Day=26/11/15”、“Day=1/7/14”、“Day=19/2/15”这三个特征都是代表日期的，可以放到同一个field中。同理，Country也可以放到一个field中。简单来说，同一个categorical特征经过One-Hot编码生成的数值特征都可以放到同一个field，包括用户国籍，广告类型，日期等等

在FFM中，每一维特征x_i，针对其它特征的每一种field f_j，都会学习一个隐向量$v_{i,fj}$。因此，隐向量不仅与特征相关，也与field相关。也就是说，“Day=26/11/15”这个特征与“Country”特征和“Ad_type"特征进行关联的时候使用不同的隐向量，这与“Country”和“Ad_type”的内在差异相符，也是FFM中“field-aware”的由来

假设样本的n个特征属于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可以看作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

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} < v_{i,f_j}, v_{j,f_i} > x_i x_j$$

其中，f_j 是第 j 个特征所属的field。如果隐向量的长度为 k，那么FFM的二次参数有 nfk 个，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，其预测复杂度是 $O(kn^2)$

特征组合

下面以一个例子简单说明FFM的特征组合方式。输入记录如下：

User	Movie	Genre	Price
YuChin	3Idiots	Comedy, Drama	$9.99

这条记录可以编码成5个特征，其中“Genre=Comedy”和“Genre=Drama”属于同一个field，“Price”是数值型，不用One-Hot编码转换。为了方便说明FFM的样本格式，我们将所有的特征和对应的field映射成整数编号

Field name	Field index	Feature name	Feature index
User	1	User=YuChin	1
Movie	2	Movie=3Idiots	2
Genre	3	Genre=Comedy	3
Price	4	Genre=Drama	4
		Price	5

那么，FFM的组合特征有10项，如下图所示

$$
\begin{aligned}
&\langle \mathbf{v}_{1,2}, \mathbf{v}_{2,1} \rangle \cdot 1 \cdot 1 + \langle \mathbf{v}_{1,3}, \mathbf{v}_{3,1} \rangle \cdot 1 \cdot 1 + \langle \mathbf{v}_{1,3}, \mathbf{v}_{4,1} \rangle \cdot 1 \cdot 1 + \langle \mathbf{v}_{1,4}, \mathbf{v}_{5,1} \rangle \cdot 1 \cdot 9.99 \\
&+ \langle \mathbf{v}_{2,3}, \mathbf{v}_{3,2} \rangle \cdot 1 \cdot 1 + \langle \mathbf{v}_{2,3}, \mathbf{v}_{4,2} \rangle \cdot 1 \cdot 1 + \langle \mathbf{v}_{2,4}, \mathbf{v}_{5,2} \rangle \cdot 1 \cdot 9.99 \\
&+ \langle \mathbf{v}_{3,3}, \mathbf{v}_{4,3} \rangle \cdot 1 \cdot 1 + \langle \mathbf{v}_{3,4}, \mathbf{v}_{5,3} \rangle \cdot 1 \cdot 9.99 \\
&+ \langle \mathbf{v}_{4,4}, \mathbf{v}_{5,3} \rangle \cdot 1 \cdot 9.99
\end{aligned}
$$

其中，红色是field编号，蓝色是特征编号

与FM对比

FFM由于引入了field的概念细化了FM的隐向量的表示，因此设计合理的field之后，效果会比FM好

但是由于FM经过推导后，复杂度是线性的，而FFM的复杂度是二次的，在工业界的业务中，又常常会出现上亿维度的特征空间，如果使用FFM，会导致成本过高，所以如果使用FFM对于FM没有明显的提升效果，一般选用FM模型

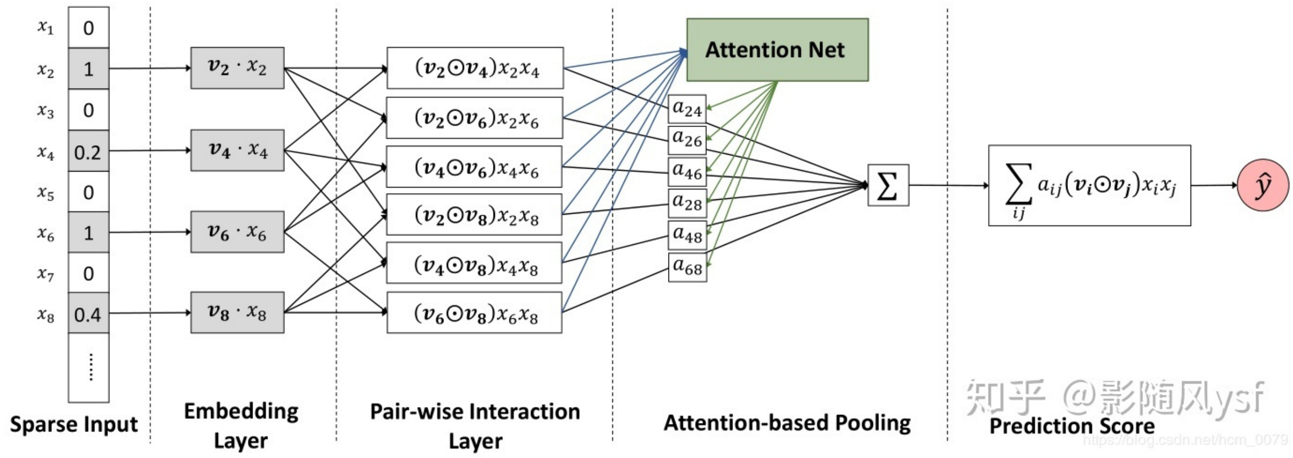
从FM和FFM在公开数据集上的表现来看，FM和FFM各有千秋，但相比于原始未进行特征交互的LM和直接计算特征交互矩阵W的Poly2方法还是有一定程度的提升

	statistics			logloss			
Data set	# instances	# features	# fields	LM	Poly2	FM	FFM
KDD2010-bridge	20,012,499	651,166	9	0.27947	0.2622	0.26372	0.25639
KDD2012	149,639,105	54,686,452	11	0.15069	0.15099	0.15004	0.14906
phishing	11,055	100	30	0.14211	0.11512	0.09229	0.1065
adult	48,842	308	14	0.3097	0.30655	0.30763	0.30565
cod-rna (dummy fields)	331,152	8	8	0.13829	0.12874	0.12580	0.12914
cod-rna (discretization)	331,152	2,296	8	0.16455	0.17576	0.16570	0.14993
ijcnn (dummy fields)	141,691	22	22	0.20093	0.08981	0.07087	0.0692
ijcnn (discretization)	141,691	69,867	22	0.21588	0.24578	0.20223	0.18608

AFM

FM能够发现二阶组合特征，但是所有特征的权重都是一样的，这会阻碍FM的效果，因为不是所有的特征都是有用的，例如有些无用的特征进行组合会引入噪声，降低FM的效果。因此AFM模型引入了attention机制。attention机制相当于一个加权平均，attention的值就是其中权重，用来判断不同特征之间交互的重要性

结构图



可以看出，AFM前三个部分，sparse input、embedding layer、pair-wise interaction layer其实和FM是一样的。

在pair-wise interaction layer中，AFM把embedding后的特征向量进行两两组合，得到：

$$f_{PI}(\mathcal{E}) = \{(\mathbf{v}_i \odot \mathbf{v}_j)x_i x_j\}_{(i,j)\in\mathcal{R}_x},$$

圆圈中有个点的符号代表的含义是element-wise product，即：

$$(x_{1,1}, x_{1,2}\ldots) \odot (x_{2,1}, x_{2,2}\ldots) = (x_{1,1}x_{2,1}, x_{1,2}x_{2,2}\ldots)$$

因此，我们在求和之后得到的是一个K维的向量，还需要跟一个向量p相乘，得到一个具体的数值。

最后经过attention-based pooling的加权后得到预测公式

$$y'_{AFM} = w_0 + \sum_{i=1}^{n} w_i x_i + p^T \sum_{i=1}^{n}\sum_{j=i+1}^{n} a_{ij}(v_i \odot v_j)x_i x_j$$

其中$a_{i,j}$的值是通过最小化损失函数得到的，所以还需要加入MLP，将此层称为attention network，现定义为：

$$a'_{ij} = \mathbf{h}^T ReLU(\mathbf{W}(\mathbf{v}_i \odot \mathbf{v}_j)x_i x_j + \mathbf{b}),$$

对score进行 softmax 的归一化：

$$a_{ij} = \frac{\exp(a'_{ij})}{\sum_{(i,j)\in\mathcal{R}_x}\exp(a'_{ij})},$$

另外，由于这种attention的方式对训练数据的拟合表达更充分，但也更容易过拟合，因此AFM的论文作者除了在loss中加入正则项之外，在attention部分加入了dropout

与FM的对比

作者在Frappe和MovieLens数据集上进行了实验，AFM的表现beat掉了所有fine tuned的对比模型（LibFm、HOFM、Wide&Deep、DeepCross）。而且，使用这套框架的传统FM也要比LibFM的效果好，原因之一在于dropout的引入降低了过拟合的风险，第二在于LibFM在迭代优化时使用的是SGD方法，对每个参数的学习步长一致，很容易使后期的学习很快的止步，而Adagrad的引入使得学习率可变，因此性能更加优良。另外，attention的引入，一方面加速了收敛速度，另一方面也具有对交叉特征的可解释性（因为就是权重），类似于特征重要性。