FM 总结

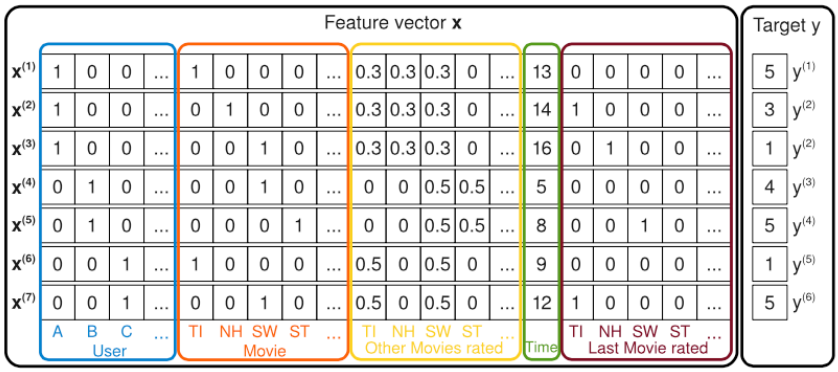
FM 背景

Factorization Machine (FM) 由 Steffen Rendle 在 2010 年提出，模型主要通过特征组合来解决大规模稀疏矩阵的分类问题，该模型在搜索推荐领域被广泛使用。

FM 将 SVM 的优势和 factorization models 结合起来。不同于 SVM 的是，FM 使用 factorized parameters 对变量间交互关系进行建模，因而可以对存在巨大稀疏性（比如推荐系统）的问题评估交互关系，这是 SVM 办不到的。另一方面，存在其他一些 factorization 模型，比如矩阵分解，parallel factor analysis，或者 SVD++，PITF, FPMC。这些模型的缺点是不能胜任一般预测任务，而只对特殊输入数据有效；此外，它们的模型方程和优化算法对每项任务是需要单独导出的，而通过指定输入数据，FM 可以模仿这些模型。

一般的预测任务是预估一个函数 $y: R^n \rightarrow T$，输入是实数特征向量 $x \in R^n$，输出是 T（对回归问题 $T = R$，对分类问题 $T = \{+, -\}$）。在监督训练中，训练数据集 $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots\}$，目标函数 y 已知。

经典的电影评分问题：



问题就是需要对电影进行评分(y 项)，而 x 都是特征，其中：

1. 第一部分蓝色为当前评分的用户（每行只有一个值为1）
2. 第二部分红色为被评分的电影（每行只有一个值为1）
3. 第三部分黄色为该用户曾经为其他电影的评分情况
4. 第四部分绿色为该用户当前评分的月数
5. 第五部分棕色为该用户最近一次评分的电影

这是一个回归问题，最简单粗暴的方法是线性回归，对绿色特征处理成 binary，计算公式为：

$$y = w_0 + \sum_{i}^{n} w_i \cdot x_i$$

为了更高级一些，对某些特征进行组合：

$$y = w_0 + \sum_{i}^{n} w_i \cdot x_i + \sum_{i}^{n} \sum_{j=i+1}^{n} w_{i,j} x_i \cdot x_j$$

这么做存在一些问题：

1. 参数空间过大（$O(n^2)$），在处理互联网数据时，特征两两级别可能是亿级别的；
2. 需要人工经验，这里一般会选择某些特征来组合，此时人工/专家经验就会很重要；
3. 样本量过于稀疏，在于在训练样本中未出现的组合该模型无能为力。

FM 方法：

定理：对于一个正定矩阵 W，始终存在一个矩阵 V 使得 $W = V \cdot V^T$ 成立（需要 V 的维数k足够大）

但是在巨大稀疏矩阵的情况下，当 k 并不是很大时，$V \cdot V^T$ 也可以很接近 W，因此可以用

$$w_{i,j} = \left\langle v_i, v_j \right\rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}$$

$\left\langle v_i, v_j \right\rangle$ 表示两个向量的点积，也称为隐向量，于是有：

$$y = w_0 + \sum_{i=1}^{n} w_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left\langle v_i, v_j \right\rangle x_i \cdot x_j$$

通过该式可以看出，FM 很像是计算每一个经过 One-Hot Encoding 变化的特征的 Embedding，然后学习不同特征之间 Embedding 相似度对于最后预测结果的影响。由于可以将上千万维的稀疏向量压缩为几十，或者几百维的 Embedding，极大地减少了模型的参数数量级，从而增强模型的泛化能力，得到更好的预测结果。

FM 训练：

FM 的训练就是训练参数 w_0, w, V，这里使用梯度下降法（比如 SGD）。FM 的梯度为：

$$\frac{\partial}{\partial \theta} \hat{y(x)} = \begin{cases} 1, & if\ \theta\ is\ w_0 \\ x_i, & if\ \theta\ is\ w_i \\ x_i \sum_{j=1}^{n} v_{j,f} x_j - v_{i,f} x_i^2, & if\ \theta\ is\ v_{i,f} \end{cases}$$

其中 $\sum_{j=1}^{n} v_{j,f} x_j$ 和 i 是独立的，可以预计算。FM 在过拟合方面可以使用 L2 正则项。

FFM (Field-aware Factorization Machines)

FFM 是在 FM 的基础上改进的，作者认为相同性质的特征归于同一 field，而当前特征在不同 field 上的表现应该是不同的。比如在广告领域中性别对于广告商（Advertiser）和投放地（Publisher）的作用就是不一样

的，比如：

Clicked	Publisher (P)	Advertiser (A)	Gender (G)
Yes	ESPN	Nike	Male

这里的特征被分成了三类，有投放地（Publisher），广告商（Advertiser）和性别（Gender），如果使用 FM 来预估这个点击率为：

$$\langle v_{ESPN}, v_{Nike} \rangle + \langle v_{ESPN}, v_{Male} \rangle + \langle v_{Nike}, v_{Male} \rangle$$

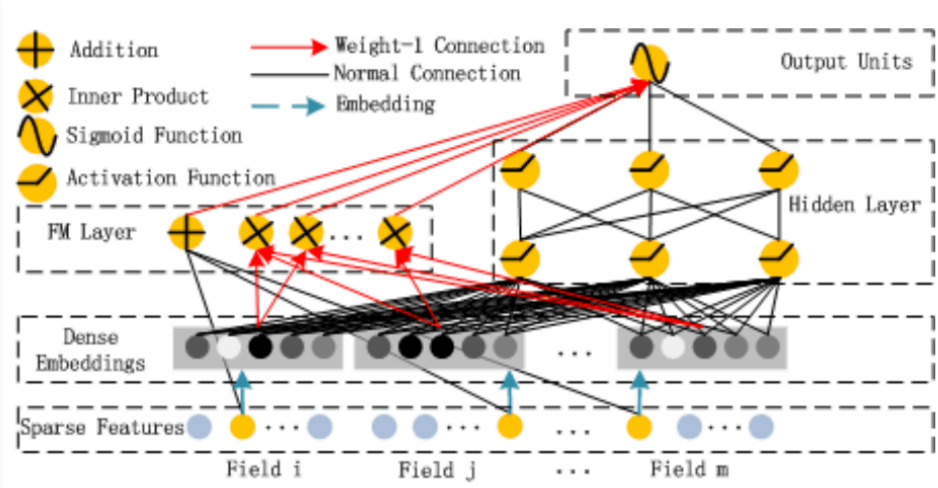
可以看出 FM 中隐向量对于不同类别的特征进行组合时都是使用同一个向量，而基于 Field-aware 的 FFM 认为当前向量对于每个类别都有一个不同的隐向量，比如性别和投放地进行组合使用的隐向量为 $v_{Male,P}$，这样推广开来之后这个问题中 FFM 的二阶项就可以表述为：

$$\langle v_{ESPN,A}, v_{Nike,P} \rangle + \langle v_{ESPN,G}, v_{Male,P} \rangle + \langle v_{Nike,G}, v_{Male,A} \rangle$$

$$y_{FFM} = w_0 + \sum_{i=1}^{n} w_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left\langle v_{i,field(j)}, v_{j,field(i)} \right\rangle x_i \cdot x_j$$

DeepFM

受 Wide&Deep 的启发，DeepFm 将 FM 和 Deep 深度学习结合起来，就是将 Wide 部分使用 FM 来代替，同时 FM 的隐向量可以充当 Feature 的 Embedding。



DeepFM 的流程为：

1. 输入的是稀疏特征的 id；
2. 进行一层 lookup 之后得到 id 的稠密 embedding；
3. 这个 embedding 一方面作为隐向量输入到 FM 层进行计算；
4. 同时该 embedding 进行聚合之后输入到一个 DNN 模型；
5. 然后将 FM 层和 DNN 层的输出求和之后进行训练

FM 部分

FM 部分是一个 factorization machine

$$y_{FM} = \langle w, x \rangle + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \left\langle V_i, V_j \right\rangle x_i \cdot x_j$$

其中，$w \in R^d, v_i \in R^k$ 。

Deep 部分

embedding 层的输出记为：

$$a^{(0)} = [e_1, e_2, \cdots, e_m]$$

其中，e_i 是第 i 个 field 的 embeding，m 是 field 的数目，$a^{(0)}$ 输入至 DNN，前向过程为：

$$a^{(l+1)} = \sigma(W^{(l)} a^{(l)} + b^{(l)})$$

最后一层的结果为：

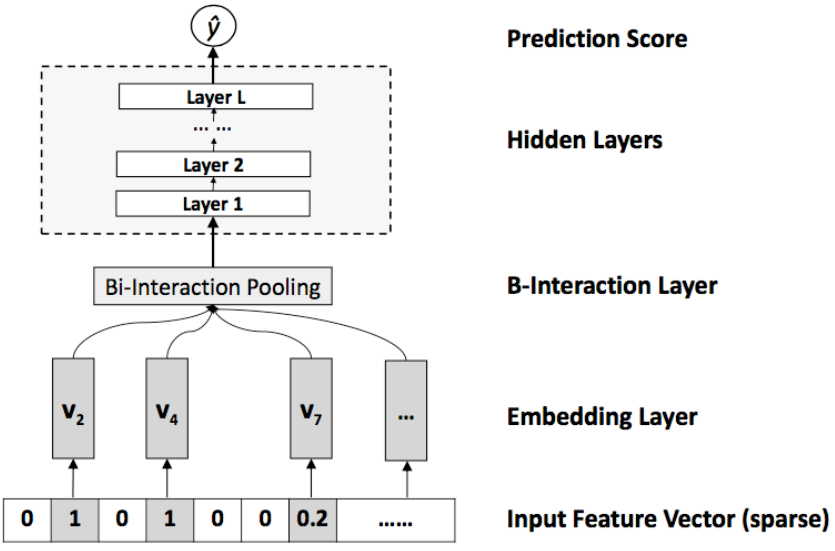
$$y_{DNN} = \sigma(W^{(H+1)} a^{H} + b^{(H+1)})$$

最终 DeepFM 可表示为：

$$y_{DeepFM} = sigmoid(y_{FM} + y_{DNN})$$

NFM

NFM (Neural Factorization Machines) 又是在 FM 上的改进工作，出发点是 FM 通过隐向量可以完成特征组合工作，还解决了稀疏的问题。但是 FM 对于 non-linear 和 higher-order 特征交叉能力不足，而 NFM 则结合了 FM 和 NN 来弥补这个不足。模型框架为：



其中：

1. Input Feature Vector 层是输入的稀疏向量，可以带权；
2. Embedding Layer 对输入的稀疏向量 lookup 成稠密的 embedding 向量；
3. Bi-Interaction Layer 将每个特征 embedding 两两做 element-wise product，Bi-Interaction 的输出是一个 k 维向量（隐向量的大小），这层负责了特征之间的 second-order 组合

$$f_{Bi}(v_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i v_i \odot x_j v_j$$

⊙ 表示向量的 element-wise product。 类似 FM 的式子转换，同样可以做如下转换将复杂度降低：

$$f_{Bi}(v_x) = \frac{1}{2} \left[(\sum_{i=1}^{n} x_i v_i)^2 - \sum_{i=1}^{n} (x_i v_i)^2 \right]$$

4. Hidden Layers 这是个全连接层，能够学习特征的高阶组合，定义如下：

$$z_1 = \sigma_1(W_1 f_{BI}(v_x) + b_1),$$

$$z_2 = \sigma_2(W_2 z_1 + b_2),$$

$$\cdots$$

$$z_L = \sigma_L(W_L z_{L-1} + b_L)$$

5. Prediction Layer 最后一层 hidden 层的输出向量转换为最终的预测 score：

$$f(x) = h^T z_L$$

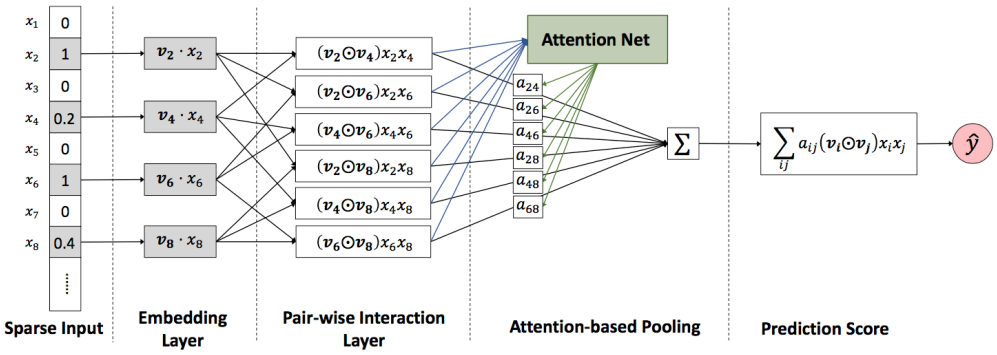
h 代表预测层的神经元权重。 综上，NFM 预测模型的公式为：

$$\hat{y_{NFM}}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + h^T \sigma_L(W_L(\cdots \sigma_1(W_1 f_{BI}(v_x) + b_1) \cdots) + b_L)$$

FM 可以看做是 NFM 模型 Hidden Layer 层数为 0 的一种特殊形式。

AFM

AFM (Attentional Factorization Machines) 的主要贡献是引入了 pair-wise interaction 层和 attention-based pooling 层。



Pair-wise Interaction Layer

将 m 个向量扩展为 $m(m-1)/2$ 个组合向量，每个组合向量是两个向量的 element-wise 乘积。假设特征向量中非零向量集合为 χ，embedding 层的输出为 $\varepsilon = \{v_i x_i\}_{i \in \chi}$，则 pair-wise 层的输出为向量集：

$$f_{PI}(\varepsilon) = \{(v_i \odot v_j) x_i x_j\}_{(i,j) \in R(x)},$$

其中，$\odot$ 表示两个向量的 element-wise 乘积，$R_x = \{(i,j)\}_{x \in \chi, j \in \chi, j > x}$，则 FM 可写为：

$$\hat{y} = \mathbf{p}^T \sum_{(i,j) \in R_x} (v_i \odot v_j) x_i x_j + b$$

其中 $\mathbf{p}$ 为 k 维向量，b 为实数，分别表示预测层的权重和 bias。

Attention-based Pooling Layer

通过 Attention 建立权重矩阵来学习两两向量组合时不同的权重。

$$f_{Att}(f_{PI}(\varepsilon)) = \sum_{(i,j) \in R_x} a_{ij} (v_i \odot v_j) x_i x_j,$$

其中，a_{ij} 是特征组合 $\hat{w_{ij}}$ 的 attention score，计算公式为：

$$a'_{ij} = \mathbf{h}^T Relu(\mathbf{W}(v_i \odot v_j) x_i x_j + \mathbf{b}),$$

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

其中，$\mathbf{W} \in R^{t \times k}, \mathbf{b} \in R^t, h \in R^t$ 为模型参数，t 为 attention network 的隐藏层大小。attention-based pooling 层的输出是 k 维向量，最终 AFM 的计算公式为：

$$\hat{y_{AFM}}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \mathbf{p}^T \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{ij} (v_i \odot v_j) x_i x_j$$

模型的参数为 $\theta = \{w_0, \{w_i\}_{i=1}^n, \{v_i\}_{i=1}^n, \mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{h}\}$。

总结

1. FMs 算法被广泛应用于 CTR(Click-Through-Rate 点击通过率)预测类问题中，可以取得不错的效果，最大特征是可以解决特征组合问题。
2. 原始 FM 算法的运行性能最快，可以达到 $O(k\bar{n})$，在工业中使用最广最简单，其他带神经网络的 FM 如果想在在线系统中使用得做很多离线和分解。