本文是一篇关于 MLP 在推荐系统的应用的汇总，包括两个部分 NCF，Factorization Machine。

一．NCF

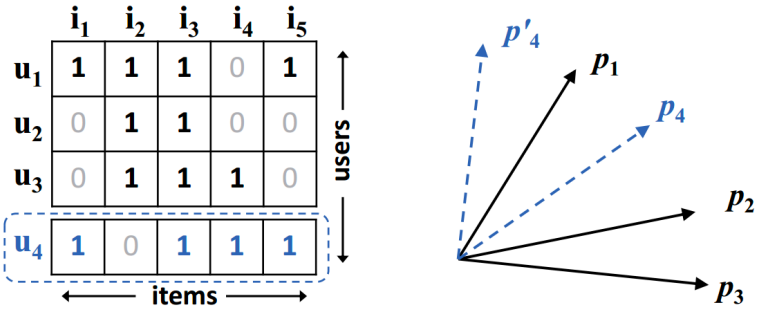
1. NCF 模型

许多利用 deep learning 来做推荐的工作 focus 在辅助信息的提取上，而对协同过滤最关键的元素---user 和 item 之间的交互作用，这些工作仍然利用的是矩阵分解模型，利用 latent feature 的内积进行推荐。Neural collaborative filtering（NCF）是一种用神经网络取代这个内积的部分的技术，以学到 latent feature 之间任意的函数关系。

MF 利用特征向量 pu、qi 的内积评估 u 对 i 的偏好：

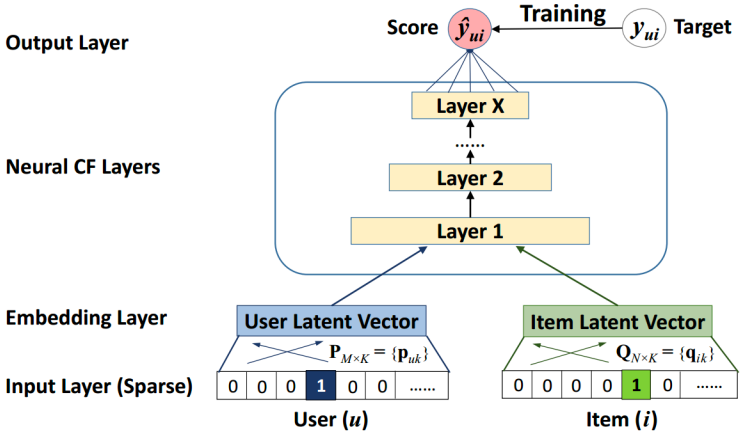
$$\hat{y}_{ui} = f(u, i|\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^T \mathbf{q}_i = \sum_{k=1}^{K} p_{uk} q_{ik},$$

由于将 user 和 item 映射到了同样的特征空间，然后使用内积也就是两个向量的 cosine 来衡量相似性；同样的，我们也可以用内积来衡量两个用户的相似性。使用 jaccard 系数（集合 A 与 B 的交集与并集的比值作为集合的相似度）来作为用户之间真实的相似程度，那么下面评分矩阵用户 123 之间 S23>S12>S13，在 latent space 中的几何关系如右图所示，加入用户 4 时，S41>S43>S42，于是我们让用户 4 的特征靠近 1，然而无论怎么放，都无法使用户 3 比用户 2 更接近用户 4。这就是使用内积描述相似度的局限性，我们可以增大 K 来解决这个问题，但是存在过拟合的风险。

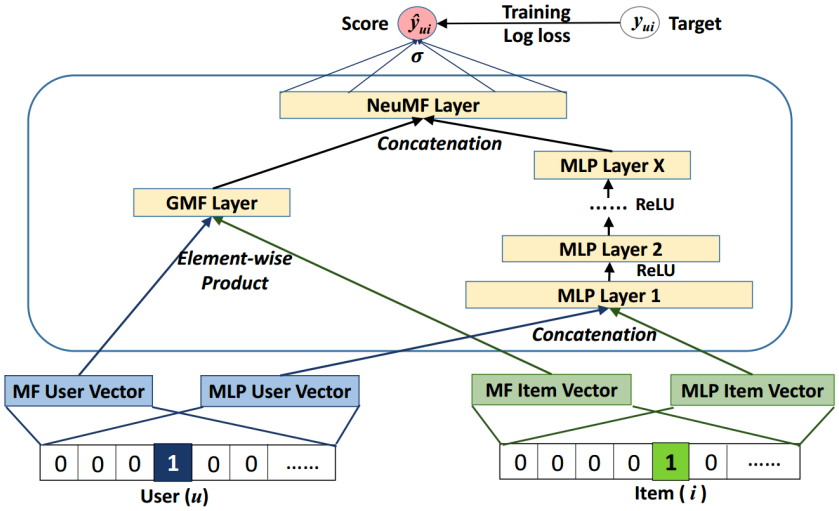


(a) user–item matrix　　(b) user latent space

下图是 NCF 的框架，user 和 item 的 id 先经过 embedding 层得到一个特征，然后输入到 MLP 中得到打分结果，用 pointwise 的目标函数进行训练：



其中 user 和 item 的 embedding 结果可以通过逐元素乘积的形式结合，然后作为 MLP 的输入，这种情况下 NCF 可以包含 MF，如果使用非线性的激活函数，那么比起线性 MF，这个设置使模型具有更强的表达能力；此外一个常规的思路就是讲两个 vector 拼接起来作为 MLP 的输入。本文也提出了一个将这两种方法融合起来的模型：



由于本模型使用的是隐式反馈（1/0），如果使用平方误差函数：

$$L_{sqr} = \sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} w_{ui}(y_{ui} - \hat{y}_{ui})^2$$

那么实际上相当于认为预测的 y_{ui} 是以$f(u,i|\Theta)$为均值的高斯分布，而这样的假设显然是不适合二值的隐式反馈。所以我们可以将待遇测的值看做是一个分类问题，即用户和项目是否有交互。使用逻辑回归进行训练，得到一个二元交叉熵损失，其中 Y-可以是全部或者部分的负样本（也会就是负采样方法）：

$$p(\mathcal{Y},\mathcal{Y}^-|\mathbf{P},\mathbf{Q},\Theta_f) = \prod_{(u,i)\in\mathcal{Y}} \hat{y}_{ui} \prod_{(u,j)\in\mathcal{Y}^-} (1-\hat{y}_{uj}).$$

$$\begin{aligned} L &= -\sum_{(u,i)\in\mathcal{Y}} \log \hat{y}_{ui} - \sum_{(u,j)\in\mathcal{Y}^-} \log(1-\hat{y}_{uj}) \\ &= -\sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} y_{ui}\log\hat{y}_{ui} + (1-y_{ui})\log(1-\hat{y}_{ui}). \end{aligned}$$

2. NMF 的应用：异构跨域推荐

在线平台一般分为两种，一种是信息导向的，比如一些电商的网站，强调 user-item 的相互作用；另一种是社交导向的，比如推特等提供社交网络服务，有丰富的 user-user 链接。虽然这两种 domain 是异构的，但是他们会共有一些用户，称为桥梁用户（bridge users），通过他们我们可以进行跨域社交推荐，也就是在社交网络中的一些潜在用户推荐信息 domain 中的相关 items。

当前大部分的 cross domain 的推荐方法都是对同构的域，而针对本文的 task，存在的困难有：

1. 数据集中桥梁用户的不足
2. 在信息域中的 attribute 充足，但是很少有人关注利用这些 attribute 来提高对社交网络中用户的推荐结果

本文提出了一种称为 Neural Social Collaborative Ranking (NSCR)的方法来利用信息域中 user-item connection 和社交域中的 user-user connection。在信息域，利用属性增强对 user 和 item embedding 的效果，在社交域中，将桥梁用户的 embedding 结果通过社交网络传播给非桥梁用户。

问题描述：在信息域中，用户集合 U_1，项目集合 I，用户给项目的打分信息为矩阵 Y，关于用户和项目的 attributes 分别为 G_u 和 G_i 表示；在社交域中，用户集合 U_2，社交关系为 S。两个域的桥梁用户为 $U=U_1 \cap U_2$。

输入：信息域$\{U_1,I,Y,G_u,G_i\}$，社交域$\{S,U_2\}$，$U_1 \cap U_2$ 非空。

输出：为社交域中每一个用户 u'确定一个对 items 的排名函数。

NSCR solution：矩阵分解模型（MF）是推荐系统重要的一个模型，这里先引入一个观点，即 CF 模型可以看做是一个浅层神经网络模型：如下图所示，我们输入用户\项目 ID 的 one-hot 的表示，然后将其映射到一个 embedding 层，将两者 embedding 向量进行逐元素相乘，得到向量 h（如果直接将 h 映射到一个打分值的话，那么这个模型就是 MF 模型）。本文认为，MF 的表现受限于使用内积来捕捉 user-item 交互作用；同样，在常规的对 attributes 的利用上，单纯的让用户 embedding 和 attribute embedding 相加，也不足以捕捉 user、item、attribute 之间的联系。

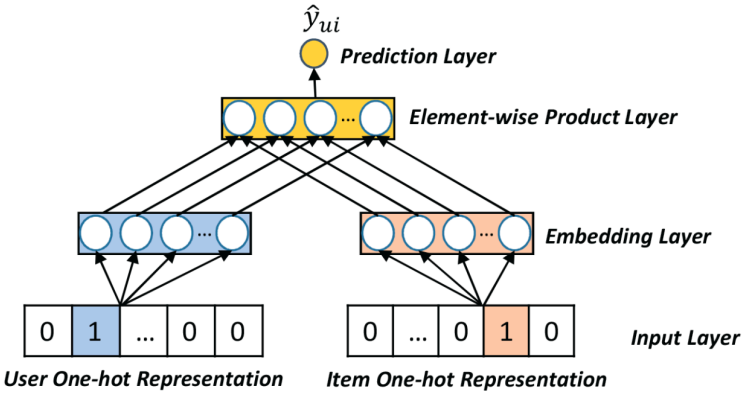


Figure 2: MF as a shallow neural network model.

由于 task 是为社交网络中的用户进行跨域推荐，本文使用了基于表示学习（embedding）的方法，认为问题的关键在于如何将 item 和来自社交网络的用户映射到相同的 embedding 空间。由于两个域的用户只有少量的重合，本文给出的解决方案是分开学习两个域的 embedding，而强迫两个学习过程共享相同桥梁用户的 embedding。优化目标为$\mathcal{L} = \mathcal{L}_I(\Theta_I) + \mathcal{L}_S(\Theta_S)$，等号右边分别是两个域各自的目标函数。

1. Learning of Information Domain

学习 cf 模型的参数，有两种目标函数：point-wise 和 pair-wise 目标函数，前者最小化预测分值和真实分值之间的损失，后者本质上是负采样，适合于本文中使用隐式反馈，同时要得到每个用户的个性化 item 排序的任务。首先取三元组(u,i,j)，其中 u 是一个用户，i 是该用户评分过的项目($y_{ui}=1$)，j 是用户没有评分过

的项目($y_{uj} = 0$)。目标函数想要学到(i,j)的正确顺序：

$$\mathcal{L}_I = \sum_{(u,i,j)\in O} (y_{uij} - \hat{y}_{uij})^2 = \sum_{(u,i,j)\in O} (\hat{y}_{ui} - \hat{y}_{uj} - 1)^2$$

其中 $y_{uij} = y_{ui} - y_{uj}$，$\hat{y}_{uij} = \hat{y}_{ui} - \hat{y}_{uj}$，其中$\hat{y}_{ui}$是预测打分值。

确定目标函数之后，我们来看预测值$\hat{y}_{ui}$是通过怎样的模型得到的。本文在 Neural Collaborative Filtering 模型的基础上，进一步加入了 attribute 的信息，结构如下图所示：

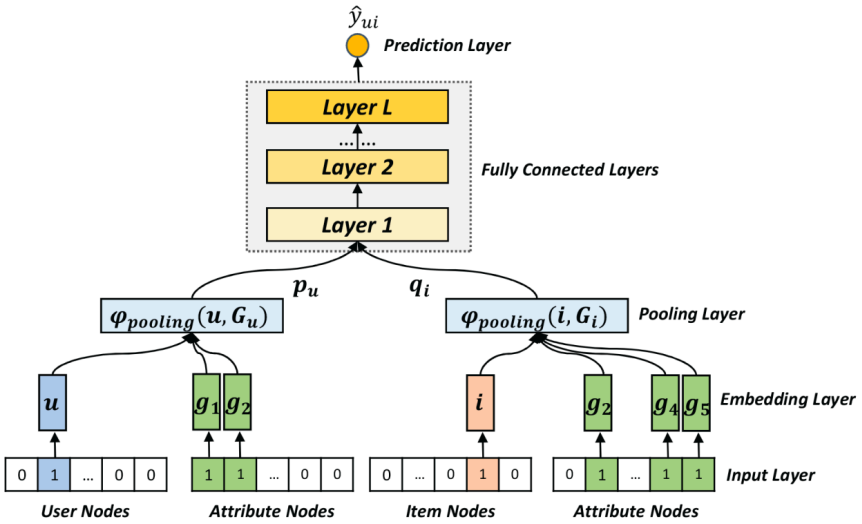


Figure 3: Illustration of our Attributed-aware Deep CF model for estimating an user-item interaction.

输入层：输入四种信息的 id，用 one-hot 向量表示

embedding：将四种信息分别进行 embedding

pooling 层：由于 attributes 的数量不确定，embedding 后的向量集大小不确定，为了给后面的 nn 一个定长的信息，进行 pooling 操作。由于最大\平均 pooling 不能捕捉用户和各 attributes 之间的交互作用，所以设计了一种 pairwise pooling 的方法：

$$\mathbf{q}_i = \varphi_{pairwise}(\mathbf{i}, \{\mathbf{g}_t^i\}) = \sum_{t=1}^{V_i} \mathbf{i} \odot \mathbf{g}_t^i + \sum_{t=1}^{V_i} \sum_{t'=t+1}^{V_i} \mathbf{g}_t^i \odot \mathbf{g}_{t'}^i.$$

对项目也做类似处理，最后将 pu⊙qi 的结果作为后面 MLP 的输入，MLP 输出预测结果。

2. Learning of Social Domain

在社交域中，本文使用了半监督学习的方法将信息域中用户 embedding 结果从桥梁用户传播到非桥梁用户。这基于这样的一个假设：如果两个用户有很强的社交关系，那么他们可能会有相似的偏好，从而在 latent space 有相似的特征表示。学习包括两部分：

平滑约束（smoothness constrain）：定义了结构一致性损失，希望相邻的用户的表示相似；$s_{u',u''}$是两个用户社交关系的强度，$d_{u'}$是节点 u'的出度，为每个用户的特征表示除以了出度的开方，进行了 normalization，如果没有这个处理，那么社交关系多的活跃用户将会产生更有效的传播。

$$\theta(\mathcal{U}_2) = \frac{1}{2}\sum_{u',u''\in\mathcal{U}_2} s_{u'u''}\left\|\frac{\mathbf{p}_{u'}}{\sqrt{d_{u'}}} - \frac{\mathbf{p}_{u''}}{\sqrt{d_{u''}}}\right\|^2$$

拟合约束：为了使两个域的 latent space 保持一致，迫使桥梁用户的两种表示接近，也就是拟合损失：

$$\theta(\mathcal{U}) = \frac{1}{2}\sum_{u'\in\mathcal{U}}\left\|\mathbf{p}_{u'} - \mathbf{p}_{u'}^{(0)}\right\|^2$$

$$\mathcal{L}_S = \theta(\mathcal{U}_2) + \mu\theta(\mathcal{U})$$

训练完成后，将 $p_{u'}$输入信息域中的预测框架，得到预测 item 的排名。

3. NMF 的应用：同构跨域推荐

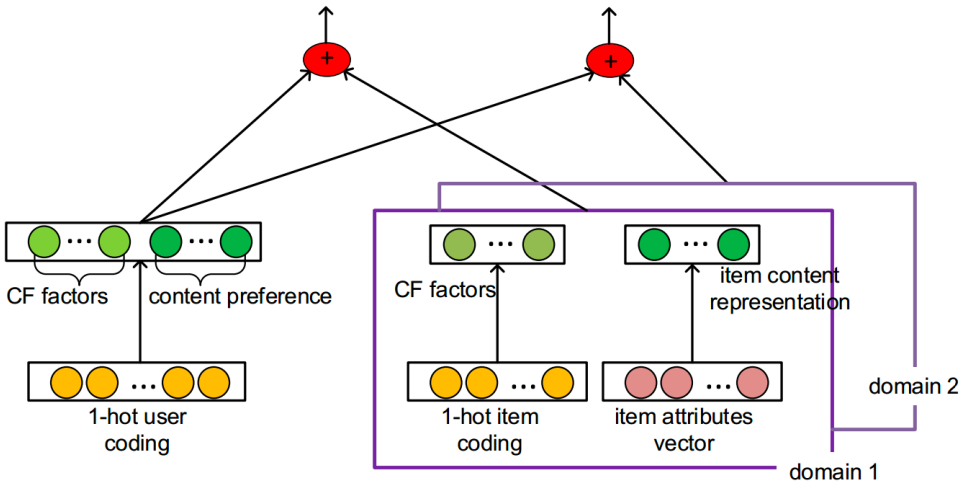
将协同过滤看作是浅层神经网络的观点同样可以拓展到两个同构的交叉域推荐，比如来自微软的 Content-Boosted Collaborative Filtering Neural Network（CCCFnet）方法。该方法认为，在不同的域中存在一些一致的模式，针对的的 task 是在两个用户完全重叠的域中，利用相对密集的辅助域的信息，提高相对稀疏的目标域的预测效果。

该方法与第二小节的 NSCR 方法相似，将 item、user 的 ID 以及 item 的 attribute 信息进行 embedding，与 NSCR 不同的是，本方法没有对 item 两部分的 embedding 结果进行 pooling，而是进行了拼接，将{Vi,Pi}作为描述 item j 的 vector，同时增加 user i 的 embedding 的维度{Uia,Uib}，Uia 是常规的 CF 部分，Uib 是基于内容的偏好。这样就结合了协同过滤和基于内容的规律。由于 user 是重合的，本方法通过共享 user 的 latent vector 实现跨域。

训练目标：

$$\mathcal{L} = \frac{1}{2}\sum_{i,j}(r_{ij}^{(1)} - U_{ia}\cdot V_j^{(1)} - U_{ib}\cdot\sum_k a_{jk}^{(1)}B_k^{(1)})^2$$
$$+\frac{\lambda_1}{2}\sum_{i,j}(r_{ij}^{(2)} - U_{ia}\cdot V_j^{(2)} - U_{ib}\cdot\sum_k a_{jk}^{(2)}B_k^{(2)})^2$$
$$+\frac{\lambda_*}{2}\sum_k(\|\Theta\|)^2$$

模型架构：



二．Factorization Machine

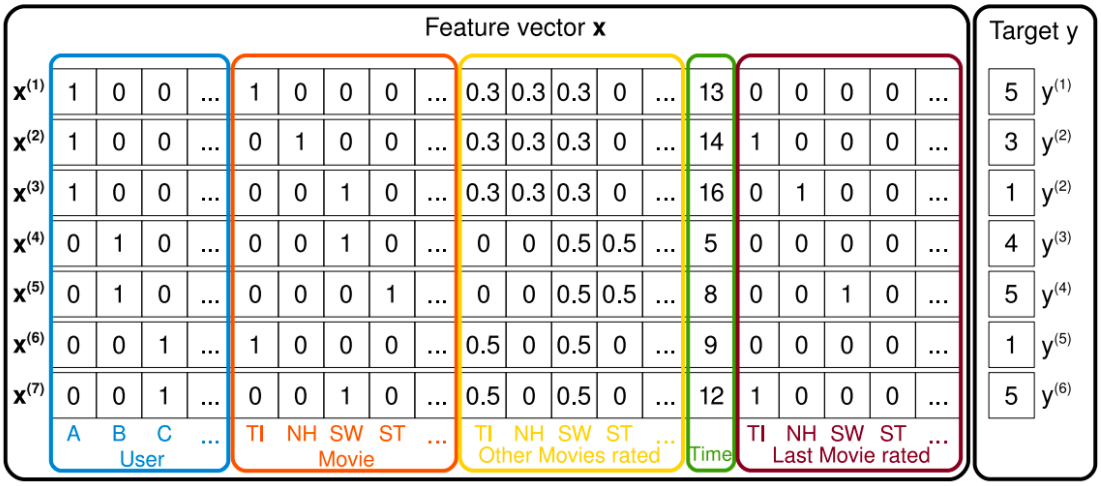
对于推荐任务，除了 MF 方法之外，还有一种称为因子分解机的方法适用于这种高度稀疏的数据，下面先对分解机模型进行介绍，然后展示两种利用神经网络对分解机进行改良的方法。

1. Factorization Machine 模型

有监督的预测任务是要找到一个函数 $y:R^n \to T$，其中特征向量 $x \in R^n$，T 是目标域。在很多现实的任务中，特征向量是高度稀疏的。比如存在一个电影评分系统，观测数据集是一些用户在某时刻对电影显式的评分(1~5 的整数)，要预测用户在某个时刻的打分行为，那么最直接的方式就是将每条评分行为的所有属性：用户、电影、用户评价过的所有电影、当前评分时间、用户评价的上一部电影等信息组合起来做为特征向量 x。对于观测数据 S

$$S = \{(\text{A}, \text{TI}, 2010\text{-}1, 5), (\text{A}, \text{NH}, 2010\text{-}2, 3), (\text{A}, \text{SW}, 2010\text{-}4, 1)$$
$$(\text{B}, \text{SW}, 2009\text{-}5, 4), (\text{B}, \text{ST}, 2009\text{-}8, 5),$$
$$(\text{C}, \text{TI}, 2009\text{-}9, 1), (\text{C}, \text{SW}, 2009\text{-}12, 5)\}$$

我们可以得到：



如果使用线性回归建模：

$$\begin{aligned}\hat{y}(\mathbf{x}) &= w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \\ &= w_0 + \sum_{i=1}^{n} w_i x_i,\end{aligned}$$

那么各个特征分量之间是孤立，没有考虑各个分量之间的相互联系。

二阶分解机模型定义如下：

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i\, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j$$

模型参数为：$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^n, \quad \mathbf{V} \in \mathbb{R}^{n \times k}$

其中 w_0 是全局的偏置；wi 建模第 i 个变量的权重；<vi,vj>是 V 中第 i 行和第 j 行的点积。比起用 wij 来建模变量 i 和 j 的相互作用，使用这个点积建模相当于

是对 W 矩阵的分解 VV^T。

那么这个模型的表达能力如何呢？由于当 k 足够大时，对于任何正定矩阵 W，都存在矩阵 V，使得 $W = VV^T$。而如何保证矩阵 W 正定呢，由于 i<j，所以起到作用的实际上是 W 中主对角线以上的元素，主对角线以下我们可以设置为对称的值，对角线元素可以取得足够大，那么可以保证 W 的正定。所以用这种方式建模具有和 W 一样的表达能力。在数据稀疏的场景中，为了避免过拟合，我们往往会限制 k 的大小，使模型具有更好的泛化性。

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ & & \ddots & \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{pmatrix}_{n \times n}$$

$$\widehat{W} = VV^\top = \begin{pmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_n^\top \end{pmatrix} (\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_n)$$

$$= \begin{pmatrix} \mathbf{v}_1^\top \mathbf{v}_1 & \mathbf{v}_1^\top \mathbf{v}_2 & \cdots & \mathbf{v}_1^\top \mathbf{v}_n \\ \mathbf{v}_2^\top \mathbf{v}_1 & \mathbf{v}_2^\top \mathbf{v}_2 & \cdots & \mathbf{v}_2^\top \mathbf{v}_n \\ & & \ddots & \\ \mathbf{v}_n^\top \mathbf{v}_1 & \mathbf{v}_n^\top \mathbf{v}_2 & \cdots & \mathbf{v}_n^\top \mathbf{v}_n \end{pmatrix}_{n \times n}$$

$$= \begin{pmatrix} \mathbf{v}_1^\top \mathbf{v}_1 & \widehat{w}_{12} & \cdots & \widehat{w}_{1n} \\ \widehat{w}_{21} & \mathbf{v}_2^\top \mathbf{v}_2 & \cdots & \widehat{w}_{2n} \\ & & \ddots & \\ \widehat{w}_{n1} & \widehat{w}_{n2} & \cdots & \mathbf{v}_n^\top \mathbf{v}_n \end{pmatrix}_{n \times n}$$

分解机方法在保持了表达能力的同时，具有两个优势：

1. 对于稀疏数据，我们没有充足的信息对每两个变量之间的交互建模，但是通过将这些交互参数分解成每个变量的“交互属性”，关于某两个变量交互的的数据可以帮助评估这两个变量其他相关的交互。比如用户 A 和 B 对电影 X 有相似的评分，那么$<V_A,V_X>$和$<V_B,V_X>$相似，从而 V_A 与 V_B 相似，对于电影 Y，只有用户 B 给出了评分，则用户 A 与电影 Y 的因子向量的点积$<V_A,V_Y>$与$<V_B,V_Y>$相似。
2. 模型的计算复杂度为 O(kn)，在稀疏空间中，复杂度为 $O(km_D)$，m_D 是数据中特征向量平均的非零元个数，比如在典型的推荐任务中，$m_D = 2$。

二阶因子分解机在特征只有用户和项目时相当于 funkSVD，但是因子分解机可以推广到多阶：

$$\hat{y}(x) := w_0 + \sum_{i=1}^{n} w_i x_i$$
$$+ \sum_{l=2}^{d} \sum_{i_1=1}^{n} \cdots \sum_{i_l=i_{l-1}+1}^{n} \left(\prod_{j=1}^{l} x_{i_j} \right) \left(\sum_{f=1}^{k_l} \prod_{j=1}^{l} v_{i_j,f}^{(l)} \right)$$

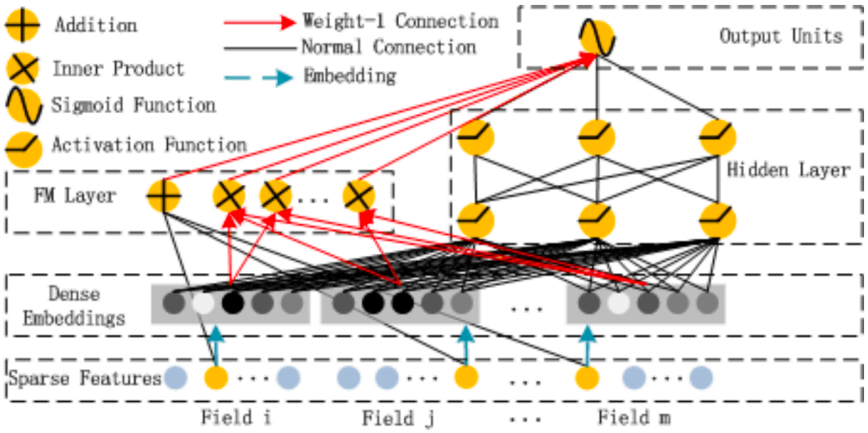
2. deepFM

deepFM 是一种基于因子分解机的用于点击率预测（CTR）的模型。在 CTR 这个任务中，学习到特征的复杂的相互作用（interaction）是预测点击率的关键，deepFM 的亮点在于用一个 end2end 的学习可以同时建模特征中低阶和高阶（low-order&high-order）的交互。

CTR 的训练集为多个用户行为的实例(x,y)，特征向量 x 中包含 m 个 field 的数据$x = [x_{field_1}, x_{field_2}, ..., x_{filed_j}, ..., x_{field_m}]$，比如用户 id，广告 id，用户性别，时间等，这些 field 用 one-hot 的形式表示，对于连续的 filed 则用数值本身，y 是一个 binary 的值点击/未点击（1/0）。

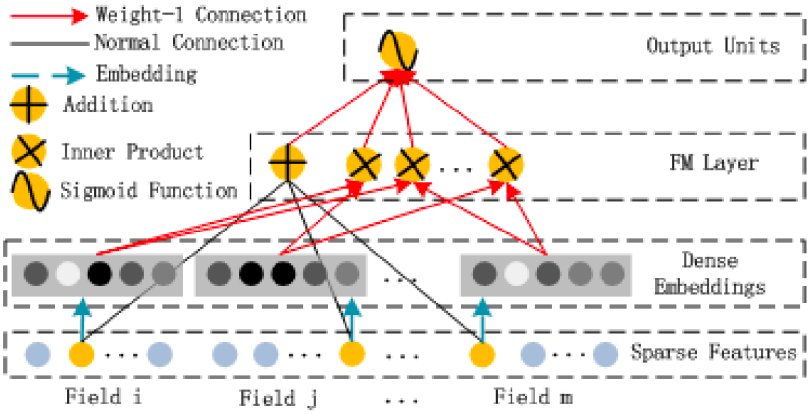
deepFM 的架构如下图所示，包括一个 FM 部分和一个 deep network 的部分，FM 中建模了特征向量中每一维的权重（order-1）和两两之间的交互作用（order-2），deepFM 建模了 high-order 的特征交互作用。

$$\hat{y} = sigmoid(y_{FM} + y_{DNN})$$



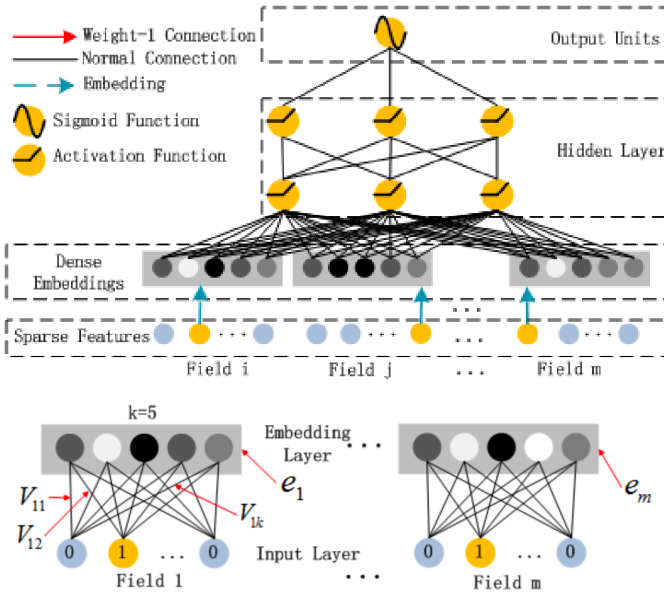
下图的 FM 部分的架构，FM 受益于将交互矩阵 W 分解为 VV^T，我们想要建模变量 i 和 j 的交互 w_{ij} 时，不需要观测数据中变量 i 和 j 同时出现（非零），只需要 i 和 j 分别出现在训练集中，就可以得到其 latent vector V_i 和 V_j，而用其内积建模。

$$y_{FM} = \langle w, x \rangle + \sum_{j_1=1}^{d} \sum_{j_2=j_1+1}^{d} \langle V_i, V_j \rangle x_{j_1} \cdot x_{j_2}$$



下图是 deep network 的部分，先经过 embedding 层，然后拼接作为 NN 的输入。需要注意的是，我们将每个 field 的 embedding 层的维度都设置为 k，同时将输入中每一个变量的 latent vector V 作为了 embedding 层的网络权重。在其他工作中，有人将 FM 作为 deep network 的预训练部分，而本文则选择 end-to-end 的同时训练这两个部分,也就是使 FM 和 DNN 共享了 embedding 层，这样可以同时建模原始输入数据中低阶和高阶的交互:

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



3. NFM

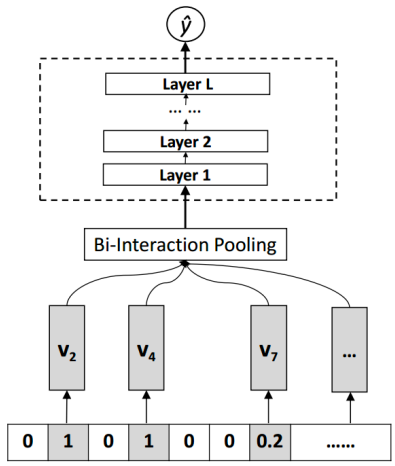
NFM 对 FM 的拓展与 NCF 对 CF 的拓展几乎完全相同，NFM 的目标函数为：

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

前半部分是 FM 常规的线性回归，后半部分使用了 NN 对特征交互部分进行拓展，首先同样是将矩阵 V 看作是一个 embedding 矩阵，经过 embedding 后得到了 embedding 向量 $Vx = \{x_1v_1,\ldots,x_nv_n\}$，然后经过 Bi-Interaction Layer：

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

将 embedding 向量两两进行逐元素乘积再求和作为 MLP 的输入：



NFM vs deepFM：

首先两种方法都将 V 看做 embedding 矩阵，如果输入数据都是 one-hot 的形式，那么两者 embedding 的过程相同，只不过在 embedding 后，deepFM 是将得到的向量进行拼接，而 NFM 是将各向量两两之间求逐元素乘积求和（类比 FM 的公式）。当输入中存在连续向量时（比如时间），deepFM 是以 field 为单位得到一个 embedding 向量，而 NFM 则是对 field 中的每一维得到一个 embedding 结果 $x_i v_i$，然后将这一维的值 x_i 看做是对 embedding 向量 v_i 的缩放。