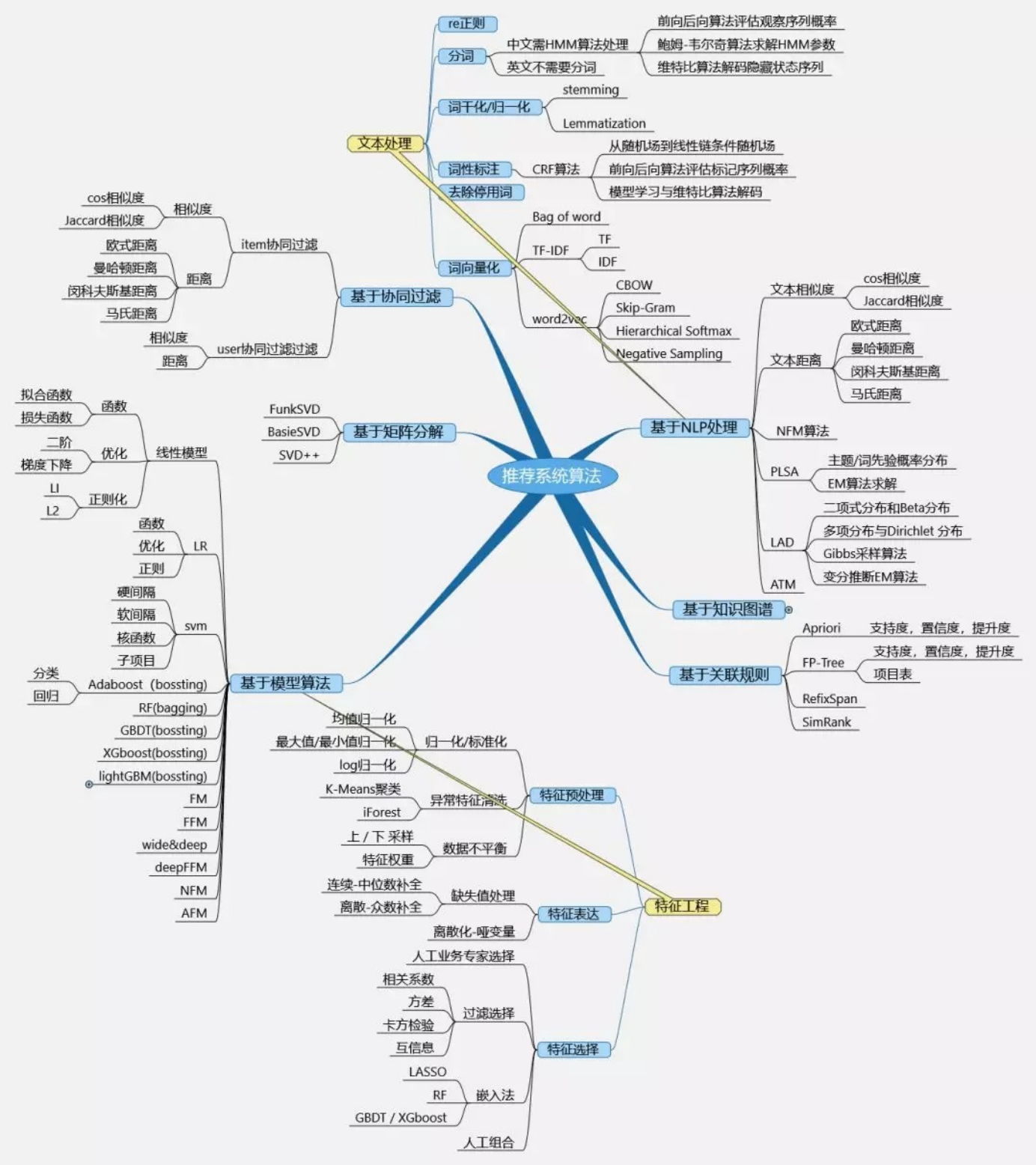
常用推荐算法模型整理



In [ ]:

1.基于关联规则

关联规则（association rules）挖掘：无监督机器学习方法；利用特定的度量指标，来发现大数据中存在的强规则。用于知识发现，而非预测。

支持度（Support）

支持度：项集（X,Y）在数据T中联合出现的概率，大于最小支持度的项集称为频繁项集。

$$Sup(X,Y) = \frac{Count(X,Y)}{Count(T)}$$

置信度（Confidence）

置信度：规则X⇒Y,表示包含项集X的数据中，也包含Y项集的条件概率。

$$Conf(X \Rightarrow Y) = \frac{Sup(X,Y)}{Sup(X)}$$

提升度（Lift）

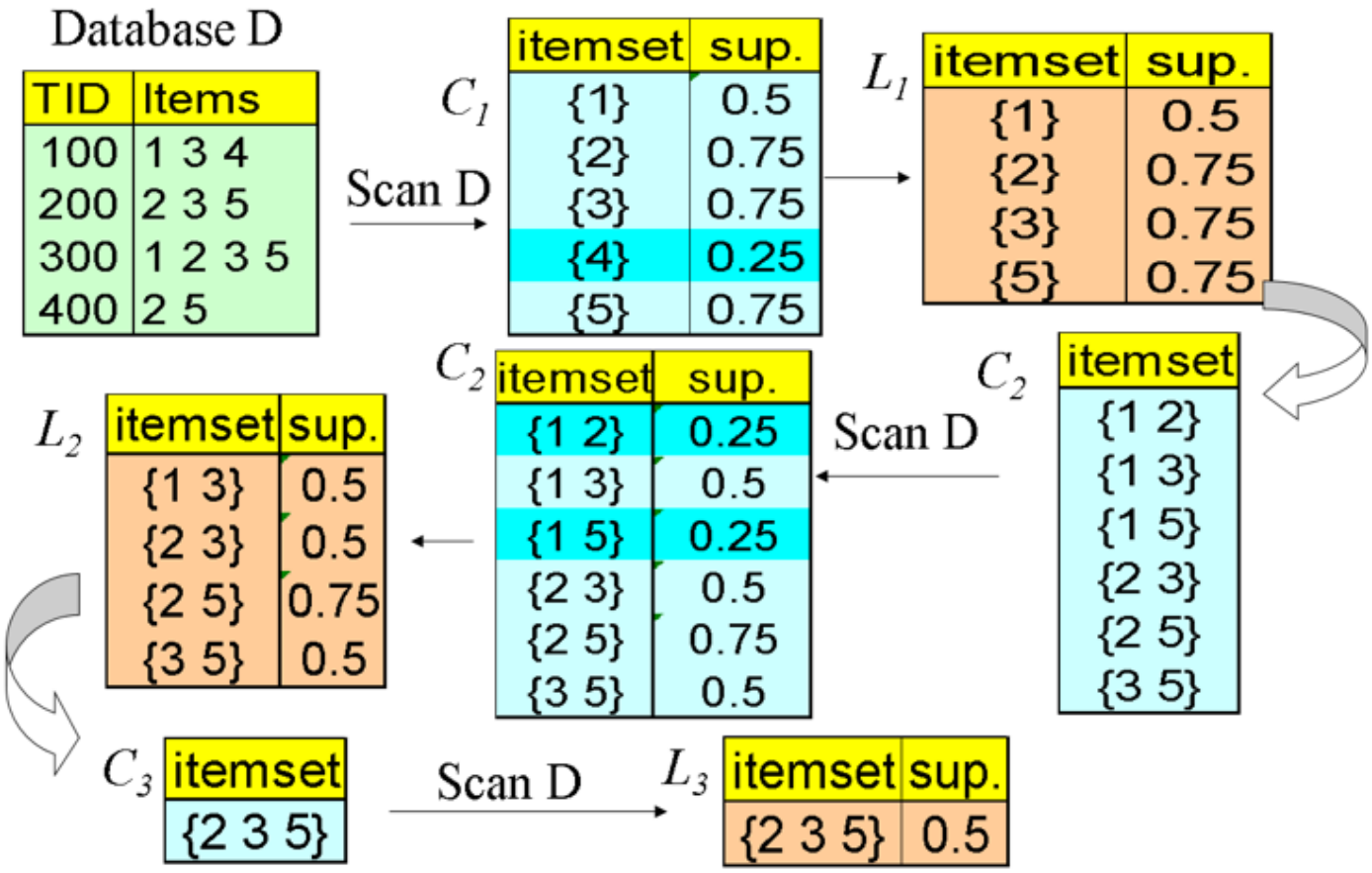
提升度:表示X出现，提升Y出现的程度。可以用来判断规则 X ⇒ Y 中的 X 和 Y 是否独立，如果独立，那么这个规则是无效的。

$$Lift(X \Rightarrow Y) = \frac{Sup(X,Y)}{Sup(X) * Sup(Y)} = \frac{Conf(X \Rightarrow Y)}{Sup(Y)}$$

如果Lift(X⇒Y)=1，说明项集X和Y独立，无任何关联。若小于1，说明X与Y是负相关，X出现可能导致Y不出现。大于1表示具有正相关，一般数据挖掘中，提升度大于3时，即认为挖据出的关联规则有效。

1.1 Apriori 算法

生成关联规则一般被划分为如下两个步骤：
1、利用最小支持度从数据库中找到频繁项集。 给定一个数据库 D ，寻找频繁项集流程如下图所示



从图中我们可以看到 itemset 中所包含的 item 是从 1 增长到 3 的。并且每次增长都是利用上一个 itemset 中满足支持度的 item 来生成的，这个过程称之为候选集生成（candidate generation）
2、利用最小置信度从频繁项集中找到关联规则。
同样假定最小的置信度为 0.5 ，从频繁项集 {2 3 5} 中我们可以发现规则 {2 3} ⇒ {5} 的置信度为 1 > 0.5 ，所以我们可以说 {2 3} ⇒ {5} 是一个可能感兴趣的规则。

1.2 FP-Growth

在Apriori算法中，寻找频繁项集，需要对每一个可能的项集遍历一遍数据，计算量较大。
FP-Growth通过将数据存储在FP树的结构上，寻找频繁项集只需要遍历两次数据。

构建FP树

第一遍扫描，计算每个单个元素的频率，并根据最小支持度，滤除不满足的元素。 第二遍扫描，首先对数据集进行处理，每一条数据按照元素的绝对出现频率排序，并滤除不满足最小支持度的元素。 例如根据上述的头指针表，元素排序为 { z:5, x:4, y:3, s:3, r:3, t:3 } ，所以处理后的数据为：

表12-2 将非频繁项移除并且重排序后的事务数据集

事务ID	事务中的元素项	过滤及重排序后的事务
001	r, z, h, j, p	z, r
002	z, y, x, w, v, u, t, s	z, x, y, s, t
003	z	z
004	r, x, n, o, s	x, s, r
005	y, r, x, z, q, t, p	z, x, y, r, t
006	y, z, x, e, q, s, t, m	z, x, y, s, t

处理后，遍历数据集，将每一条数据插入FP树中，从根节点开始递归添加路径，存在则将数值增加，不存在则创建新的节点。 例如下图所示，① 根节点不存在子节点 { z } ，所以创建新的子节点 { z } ，递归节点 { z } ，因不存在子节点 { r } ，所以创建新的子节点 { r } ，② 根节点存在子节点 { z } ，所以数值增加，递归节点 { z } ，因不存在子节点 { x } ，所以创建新的子节点 { x } ，递归节点 { x } ，.....，如此递归。

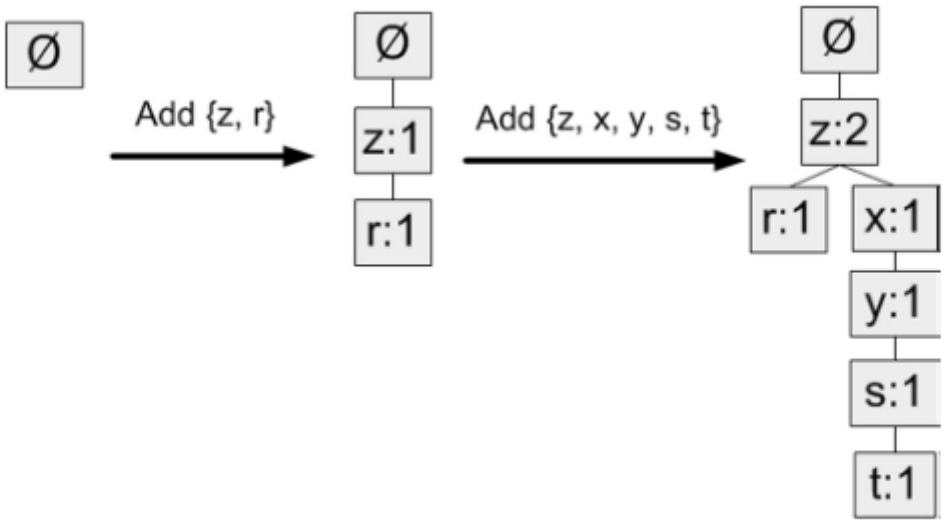


图12-3 FP树构建过程的一个示意图，图中给出了使用表12-2中数据构建FP树的前两步

构建好的完整FP-Tree如图：

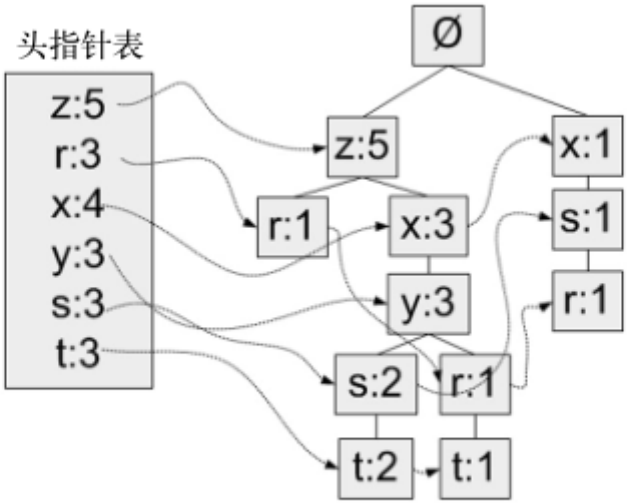


图12-2 带头指针表的FP树，头指针表作为一个起始指针来发现相似元素项

通过FP-tree挖掘频繁项集

- 对于头指针表中的每一个元素，获取其条件模式基（所有前缀路径（不包括当前元素和根节点）的集合）。
- 根据条件模式基，构建条件FP树（即，将条件模式基当作新的数据集，按照建树的流程，重新构建一棵FP树）
- 继续对于条件FP树的头指针表中的每一个元素，获取其条件模式基
- 继续根据条件模式基，构建条件FP树
- 如此递归过程，直到无法构建出FP树为止

实现代码与工具包

python实现FP-Growth (https://github.com/enaeseth/python-fp-growth)
python关联规则工具包 (https://github.com/evandempsey/fp-growth)

In [ ]:

```
```

2.基于协同过滤

协同过滤推荐方法包括：启发式规则(基于用户的推荐，基于物品的推荐)，基于模型的推荐。

2.1 User-based CF

对用户属性信息，历史行为信息进行统计，找到与目标用户相似的用户，根据相似用户的喜好产生向目标用户的推荐内容。

- 优点：在数据集完整，内容丰富的情况下能够获得较高的准确率，且能对物品间的关联性和用户的偏好进行隐式透明的挖掘。
- 缺点：当用户量不断增大时，计算用户相似度的代价也会增加，限制了算法的课扩展性。在线计算量大，实时性较低。
- 对于历史信息较少的新用户，无法准确计算与之相似的用户。

2.2 Item-based CF

通过用户对物品的历史评分记录，来计算物品之间的相似度，根据目标用户喜欢的物品的相似物品来产生推荐内容。

- 优点：系统的物品数量相对稳定，可离线计算物品的相似性，定期更新，减少线上计算复杂度，提高实时响应速度。尤其用户数远大于物品数的情况下效果更为显著。可解释性也较高。
- 缺点：基于物品相似性的推荐较少考虑用户之间的差异性，因此推荐准确率稍低。

UserCF和ItemCF代码 (https://github.com/xingzhexiaozhu/MovieRecommendation,'%E4%BB%A3%E7%A0%81')

用户对物品的评分预测

根据目标用户相似用户集（相似度大于阈值）U中所有用户u对物品i的评分以及用户对所有物品平均评分来计算目标用户对物品i的评分。计算公式如下：

$$p_{u,i} = \overline{r}_u + \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \overline{r}_{u'})}{\sum_{u' \in N} |s(u,u')|}$$

其中，s（u，u'）表示用户u和用户u'的相似度。

相似度指标

- 皮尔孙相关系数

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - \overline{r}_v)^2}}$$

- 余弦相似度

$$s(u,v) = \frac{r_u \cdot r_v}{\|r_u\|_2 \|r_v\|_2} = \frac{\sum_i r_{u,i} r_{v,i}}{\sqrt{\sum_i r_{u,i}^2} \sqrt{\sum_i r_{v,i}^2}}$$

- jaccard相似系数：用于比较有限样本集之间的相似性与差异性

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

距离度量指标

- 欧几里得距离（Euclidean Distance）

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- 曼哈顿距离（Manhattan Distance）

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

- 闵可夫斯基距离（Minkowski Distance）

$$d(x,y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p}$$

2.3 基于模型的协同过滤

基于用户和基于物品的协同过滤推荐算法共有缺点是计算量较大，难以处理大数据下的实时结果。基于模型的协同过滤方法则 对用户属性和物品属性进行统计与学习，利用历史数据训练好模型，用于新数据预测。（可以是对评分的回归模型，也可以是多分类模型）

- 优点：只要训练好了模型，就可以对新用户或新物品进行快速计算，实时响应。且准确率较高。
- 缺点：对于新增数据，模型要进行增量训练。

3.基于矩阵分解

矩阵分解：是指将一个矩阵分解成两个或者多个矩阵的乘积。 在推荐系统中，有很多用户和物品，以及部分用户对商品的评分矩阵，我们希望通过对评分矩阵的分解，从而预测目标用户对其他未评分物品的评分，进而将评分高的物品推荐给用户。

	D1	D2	D3	D4
U1	5	3	-	1
U2	4	-	-	1
U3	1	1	-	5
U4	1	-	-	4
U5	-	1	5	4

3.1 SVD (奇异值分解: Singular Value Decomposition)

传统SVD，要求矩阵稠密，且不能有缺失值。对于评分矩阵，需要对缺失值进行补全，如全局平均值补全等。

$$R_{m\times n} = U_{m\times m} S_{m\times n} V_{n\times n}$$

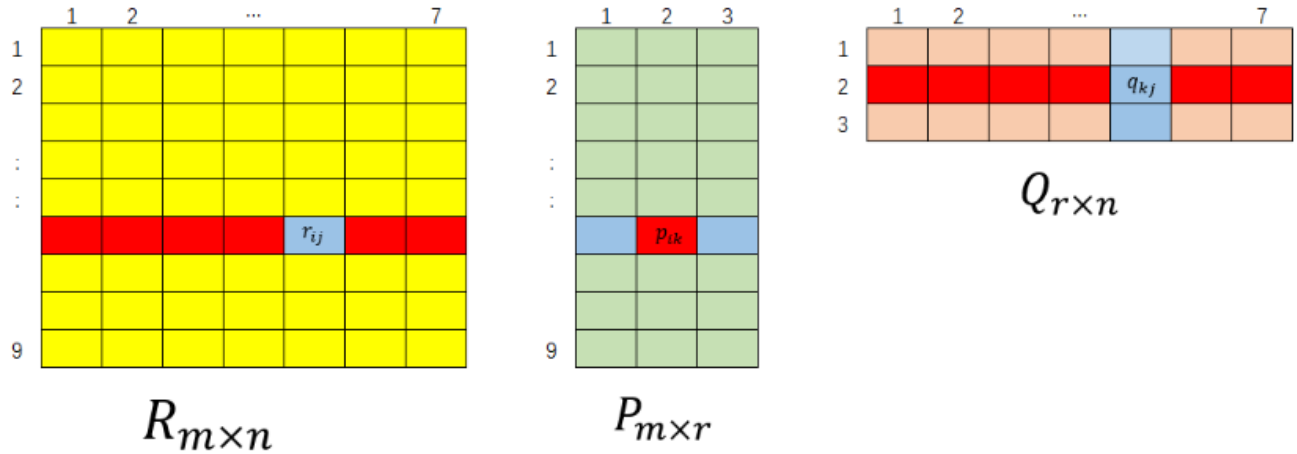
其中，U和V都是正交矩阵（转置等于逆），S为对角矩阵（对角线之外的元素值为0），对角线上的每一个元素都是矩阵的奇异值。 缺点：SVD矩阵分解计算量大。

3.2 FunkSVD

FunkSVD解决了传统SVD矩阵分解的计算效率问题，它用两个低维矩阵P和Q的乘积去逼近评分矩阵R的值：

$$R_{m\times n} \approx P_{m\times r} \times Q_{r\times n} = \hat{R}_{m\times n}$$

通过r个隐含特征将用户与物品联系起来。



$P_{m\times r}$代表用户对r个隐含特征的偏好程度，$Q_{r\times n}$代表r个隐含特征与商品的相关性。

$$\hat{r}_{ij} = \sum_{k=1}^{r} p_{ik} * q_{kj}$$

最终用户对物品的打分r取决于用户对所有隐含特征的偏好程度p以及隐含特征与物品的相关程度q之和。

FunkSVD求解

a. 损失函数

FunkSVD的求解同线性回归求解方法：最小二乘法（最小平方误差），以实际评分矩阵与分解重构后矩阵之间误差的平方作为损失函数。

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{r} p_{ik} * q_{kj})^2$$

最终，需要求解所有的非“-”项的损失之和的最小值。

$$minloss = \sum_{rij \neq -} e_{ij}^2$$

b. 损失函数求解

对于上述的平方损失函数，可以通过梯度下降法求解，梯度下降法的核心步骤是:

- 求解损失函数的负梯度

$$\frac{\partial}{\partial p_{i,k}} e_{i,j}^2 = -2\left(r_{i,j} - \sum_{k=1}^{K} p_{i,k} q_{k,j}\right) q_{k,j} = -2 e_{i,j} q_{k,j}$$

$$\frac{\partial}{\partial q_{k,j}} e_{i,j}^2 = -2\left(r_{i,j} - \sum_{k=1}^{K} p_{i,k} q_{k,j}\right) p_{i,k} = -2 e_{i,j} p_{i,k}$$

- 根据负梯度的方向更新变量

$$p_{i,k}' = p_{i,k} - \alpha \frac{\partial}{\partial p_{i,k}} e_{i,j}^2 = p_{i,k} + 2\alpha e_{i,j} q_{k,j}$$

$$q_{k,j}' = q_{k,j} - \alpha \frac{\partial}{\partial q_{k,j}} e_{i,j}^2 = q_{k,j} + 2\alpha e_{i,j} p_{i,k}$$

通过迭代，直到算法最终收敛。

Re：FunkSVD矩阵分解求解 (https://www.jianshu.com/p/812234c0da87) 矩阵分解 (https://www.jianshu.com/p/d69ab10d3d63)

3.3 BiasSVD (带偏置SVD)

FunkSVD改进算法。BiasSVD假设评分系统包括三部分的偏置因素：

- 全局偏置μ：样本集合中所有评分的平均，反映和用户物品无关的评分因素。如不同的电影评分平台因素（猫眼，豆瓣等）
- 用户偏置b_u：每个用户平均打分相对于μ的偏差，反映用户自带的和物品无关的评分因素，比如天生愿意给别人好评，心慈手软，比较好说话，有的人就比较苛刻，总是评分不超过3分。
- 物品偏置b_i：每个物品平均打分对于μ的偏差，反映物品自带的和用户无关的评分因素，如一个自带垃圾山寨货属性的商品评分不可能高，由于这个因素会直接导致用户评分低，与用户无关。

因此，用户对物品的评分可以用如下公式表示：

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T * q_i$$

最终BiasSVD的优化目标可以表示为：

$$minloss = \sum_{u \in m, i \in n} (r_{ui} - \hat{r}_{ui})^2 + \lambda(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2)$$

L2正则用以防止模型过拟合。

Re:SVD相关算法 (https://zhuanlan.zhihu.com/p/68896831)
Re:推荐算法入门 (https://www.jianshu.com/p/71bcad876c05)

3.4 SVD++

SVD：在模型中引入用户与物品交互行为的隐式反馈。 FunkSVD和biasSVD模型的训练都是在用户评分的基础上进行的，但实际场景中用户的评分行为通常较少，则需要通过引入其他维度的信息来提升模型性能。
SVD++算法在BiasSVD算法上进一步引入隐式反馈的信息。其中显式反馈指的是用户的评分行为，隐式反馈指的是用户与物品交互的其它行为（如浏览物品等）。
假设在隐式反馈中，和每个用户有过隐式交互的物品集合中，也都有一个k维隐向量与之相对应。这样一来就可以将与用户有过隐式交互的物品对应的隐向量都加起来，从一个新的维度来表示用户的兴趣偏好。 因此SVD++的优化目标公式如下：

$$\hat{r}_{ui} = \mu + b_u + b_i + q_u^T * p_i + q_i^T \frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} x_i$$

$$minloss = \sum_{u \in m, i \in n} (r_{ui} - \hat{r}_{ui})^2$$

- x_j隐式反馈对应的物品向量
- N(u):和用户u有个隐式交互的物品集合

4.基于模型算法

前言

在计算广告和推荐系统中，通常会利用机器学习模型算法来进行CTR预估。判断是否对一个商品进行推荐，是根据CTR预估的点击率来进行。
进行CTR预估除了单特征外，组合特征也是非常重要的。组合特征的方式：人工组合，Tree系列或者FM系列。
以一个广告分类的问题为例，根据用户与广告位的一些特征，来预测用户是否会点击广告。数据如下：

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：

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，且特征空间变大。

LR(Logistic Regression)

在拥有大规模离散稀疏特征的CTR预估应用场景中，最经典的是线性模型:LR模型,它简单，快速可解释性强，有很好的拟合能力。 LR模型可以用如下线性函数刻画：

$$y(x) = w_0 + \sum_i w_i x_i$$

在线下函数基础上，做变换后：

$$f(x) = \frac{1}{1 + exp^{-y(x)}}$$

缺点 泛化能力弱，且无法学到模型中特征之间的交互信息（即非线性特征），而这在推荐和CTR预估中是比较关键的。 因此，在线性回归的基础上增加特征的交互信息：

$$y(x) = w_0 + \sum_i w_i x_i + \sum_i \sum_{j<i} w_{ij} x_i x_j$$

缺点 二阶项的参数过多，如果特征个数为n，则二阶项的参数为$\frac{n(n+1)}{2}$，对于特征较多的模型，参数个数可能远远大于样本数量。这导致有绝大多数特征交互信息在样本中找不到，因而模型无法学出对应的权重参数。

4.1 因子分解机(Factorization Machine, FM)

FM是在矩阵分解(MF)、SVD++、PITF、FPMC等基础之上，归纳出针对高维稀疏数据的模型。
FM的思想是给每个特征找到一个k维的隐向量v_i，当特征进行组合时，对应特征隐向量的乘积则表示二阶组合特征的权重。
则n个特征的隐向量矩阵为：

$$\hat{\mathbf{W}} = \mathbf{V}\mathbf{V}^T = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^T & \mathbf{v}_2^T & \cdots & \mathbf{v}_n^T \end{pmatrix}$$

组合特征的权重矩阵W为：

$$\mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1k} \\ v_{21} & v_{22} & \cdots & v_{2k} \\ \vdots & \vdots & & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nk} \end{pmatrix}_{n \times k} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{pmatrix}$$

W为对角矩阵。
如此，参数个数从$\frac{n(n+1)}{2}$个减少到了kn个。 反过来看，FM是通过矩阵分解的方式，将组合特征参数矩阵分解为两个相同的低秩矩阵的乘积。
只要根据训练样本数据求得这个矩阵，那么即能求得每个特征的隐向量，从而求得组合特征的权重。这样即便样本中缺少某种组合特征也能计算出组合特征的权重，从而在后续全量数据的预测中起作用。

FM与决策树

FM和决策树都可以做特征组合，决策树可以非常方便地对特征做高阶组合。但是，当样本数据是高度稀疏的时，简单的线性回归和决策树都无法学习到特征的二阶或高阶组合参数。但是FM可以实现。

FM的实现

libFM 单机多线程并行 (http://www.libfm.org/)
difacto 基于ps-lite分布式实现 (https://github.com/dmlc/difacto)
fast_tffm 基于tensorflow的分布式实现 (https://github.com/kopopt/fast_tffm)
xlearn (https://github.com/aksnzhy/xlearn)

4.2 FFM(Field Factorization Machine)

FFM模型在FM模型的基础上，引入了域(Field)的概念。在实际预测任务中，往往包含多种类别特征，如果不同类别特征组合时采用不同的隐向量，那么这就是 **FFM(Field Factorization Machine) 模型**。

在FFM模型中，每个特征将被映射为多个隐向量$v_{i1}, \ldots, v_{if}$，每个隐向量对应一个域。当两个特征x_i, x_j组合时，用对方对应的域对应的隐向量做内积:

$$w_{ij} = v_{i,f_j}^T v_{j,f_i}$$

f_i, f_j分别是特征x_i, x_j对应的域编号。FFM 由于引入了域，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。

- 对于连续特征，一个特征就对应着一个域。由同一连续特征离散化后产生的特征，对应着相同的一个域。
- 对于离散特征，采用one-hot编码后，也都对应着同一个域。
- 同一个域内特征不可相互组合
  缺点：
  参数由FM的$k \times n$变成了$f \times k \times n$(f为场的个数。这样增加了计算量，模型也容易过拟合，在工业应用中通常无法直接采用其进行ctr预估。

优点： FM与FFM可以进行有效的二阶特征组合，且能很好的对高维稀疏特征进行学习。事实上FM可以看做对高维稀疏的离散特征做embedding。
缺点： FM与FFM无法进行高阶特征组合建模。

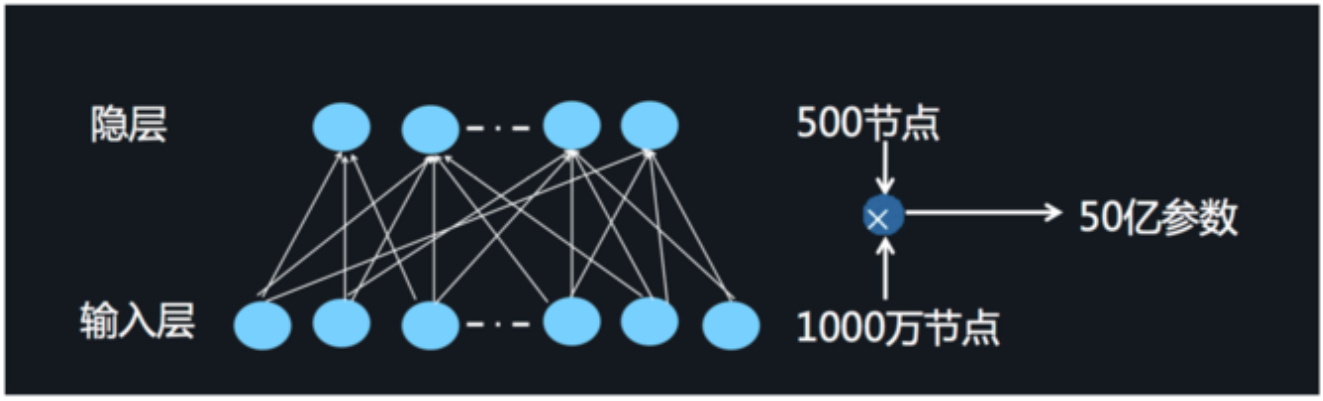
4.3 Wide&Deep

- Memorization根据历史行为数据，产生的推荐通常和用户已有行为的物品直接相关的物品。
- Generalization会学习新的特征组合，提高推荐物品的多样性。

wide&deep模型：核心思想是结合线性模型的拟合能力(记忆能力Memorization)和 DNN 模型的泛化能力（Generalization），从而提升整体模型性能。

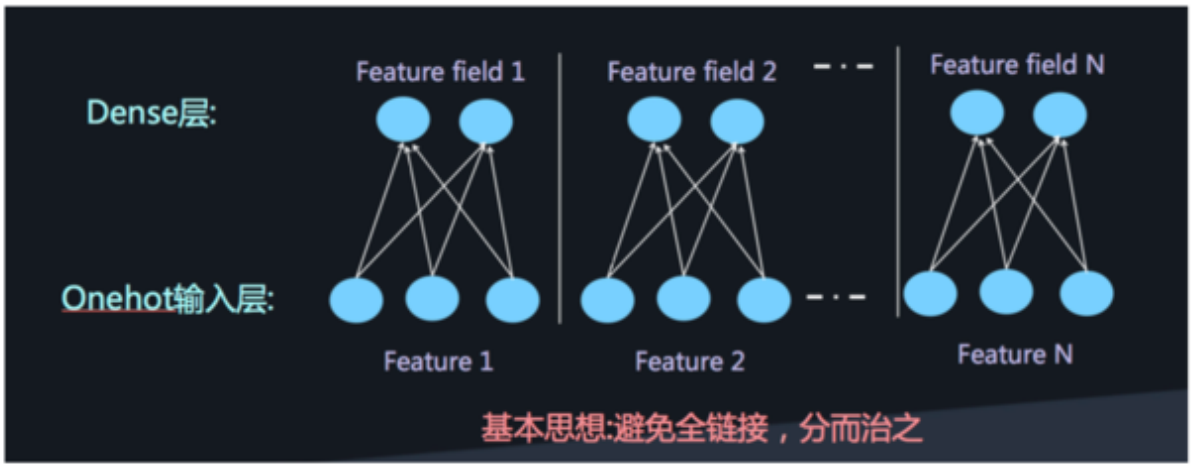
DNN模型可以通过**低维嵌入（embedding）**技巧解决高维稀疏特征问题，同时可以做更复杂的非线性变换，能学习**更高阶的特征组合**，并且具有很好的泛化能力。 对于离散特征的处理，将特征转换为one-hot形式，但输入DNN中会导致参数过多：

- Onehot作为DNN输入的问题：CTR预估任务里不可行

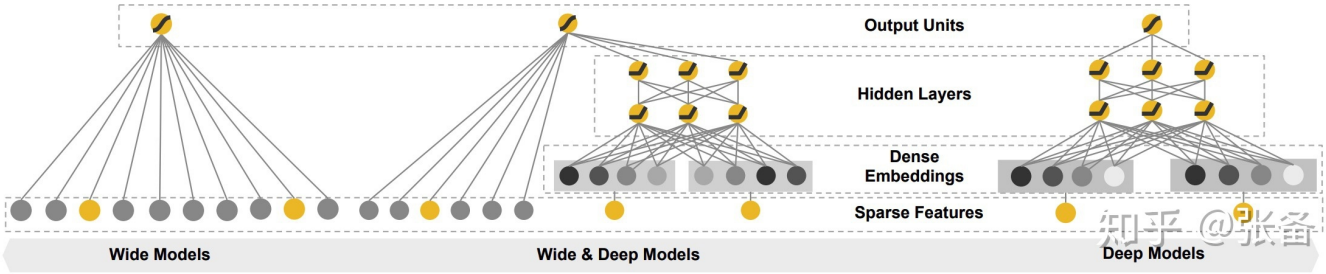


解决这个问题，可以引用类似FFM的思想，将特征分为不同的Field：

- 解决思路：从OneHot到Dense Vector



再加两层的全链接层，让Dense Vector进行组合，得到高阶特征：



1. Wide 该部分是广义**线性模型**，即：

$$y = w^T[x, \phi(x)] + b$$

其中x和$\phi(x)$分别表示原始特征和组合特征。

2. Deep 部分是普通神经网络，只不过在网络中增加了**embedding**层，用于将一些稀疏、高维的特征（如ID类特征）转换为低维稠密的特征，然后和一些原始dense特征一起作为网络的输入。每一层隐层计算：

$$y_{i+1} = F(w_i y_i + b_i)$$

其中，w_i，b_i，y_i分别是第i层的权重，偏置和激活值。F是激活函数。

3. 损失函数 模型选取logistic loss作为损失函数，此时Wide & Deep最后的预测输出为：

$$p(y=1|x) = \sigma(w_{wide}^T[x, \phi(x)] + w_{deep}^T y_{out} + b)$$

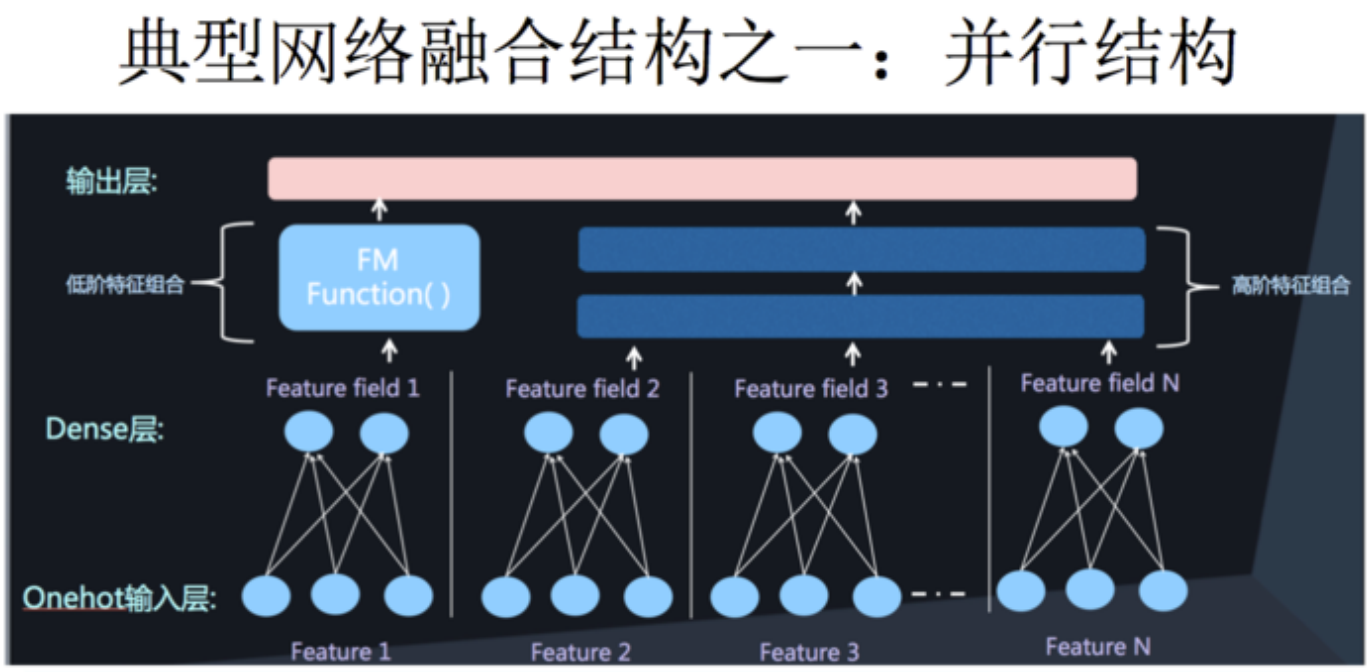
其中σ表示sigmoid函数w_{wide}, w_{deep}分别表示wide部分和deep部分特征的权重，y_{out}表示NN最后一层激活值。

4. 联合训练

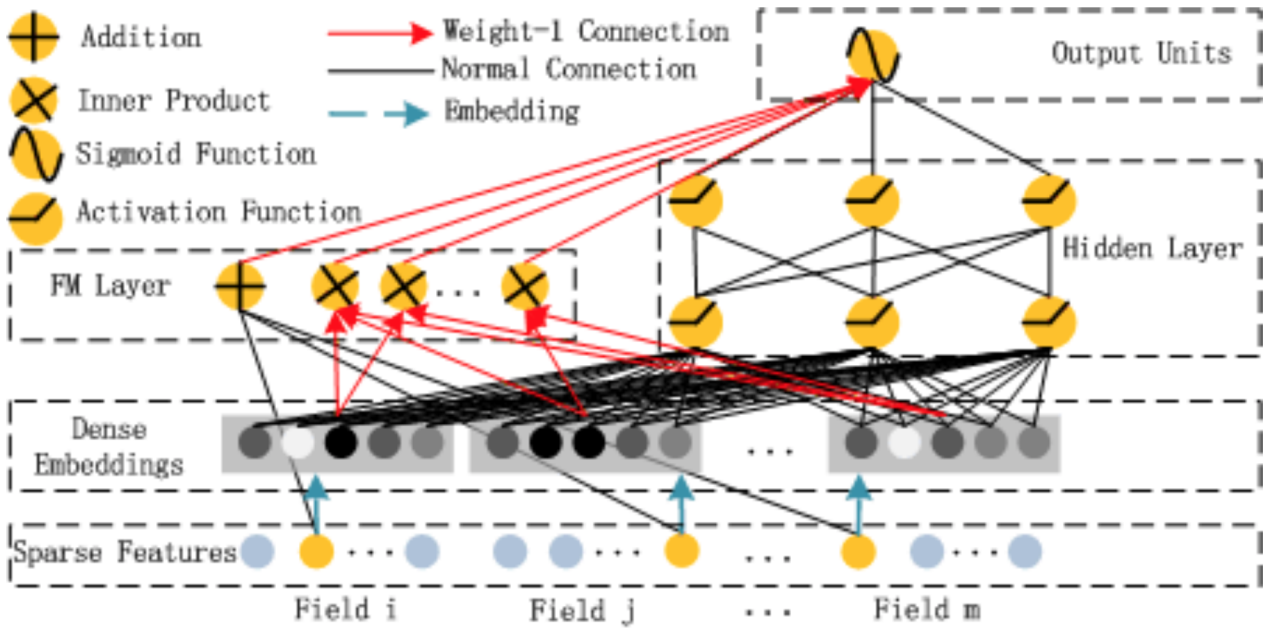
联合训练（Joint Training）和集成（Ensemble）是不同的，集成是每个模型单独训练，再将模型的结果汇合。相比联合训练，集成的每个独立模型都得学得足够好才有利于随后的汇合，因此每个模型的model size也相对更大。而联合训练的wide部分只需要作一小部分的特征叉乘来弥补deep部分的不足，不需要一个full-size的wide模型。

4.4 DeepFM

DeepFM就是把Wide&Deep模型的wide部分改为了FM：神经网络部分与因子分解机部分，分别负责高阶和低阶特征的提取。

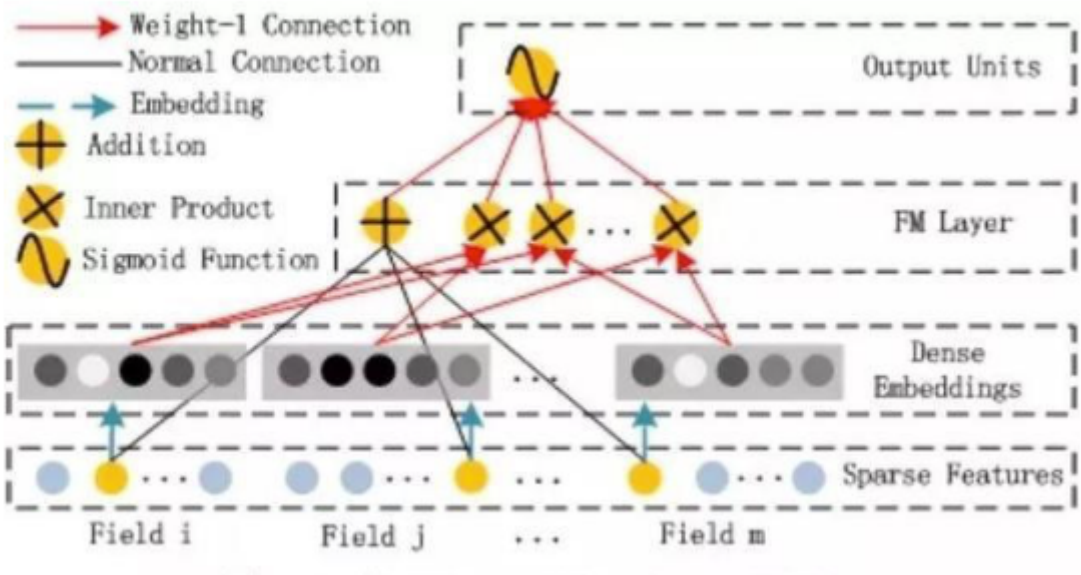


1. DeepFM的模型结构：



- 红色线---带权重的连接
- 黑色线---不带权重的连接
- 蓝色线---稀疏特征向稠密特征转换的embedding向量。
- 第一层：Sparse Feature稀疏特征，每个field下有一个onehot编码的向量，其长度等于此filed下特征类别数。这一层共m个field。
- 第二层：m个Dense向量拼接而成的mk维向量x。

2. FM层部分的详细结构如下：

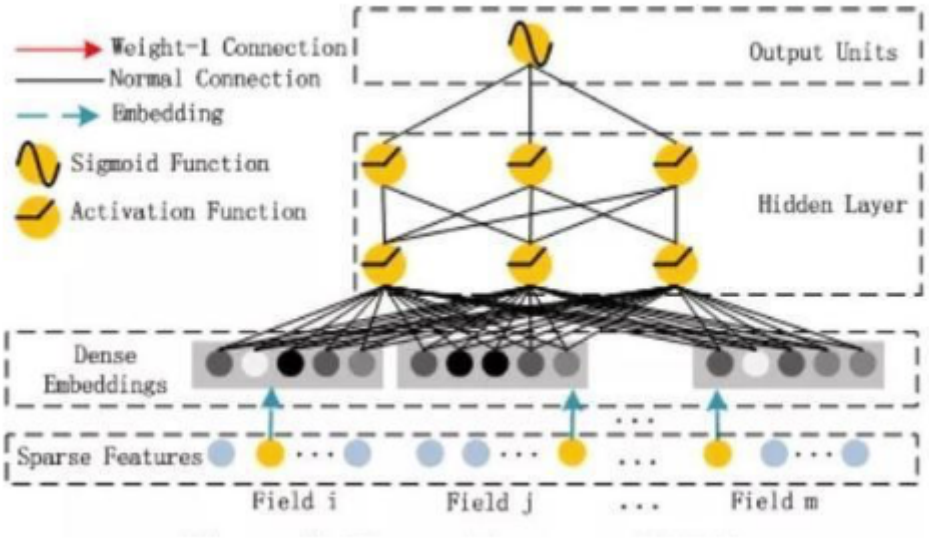


FM因为引入了隐变量的原因，对于几乎不出现或者很少出现的隐变量，FM也可以很好的学习。 输出公式为：

$$y_{FM} = w_0 + \sum_i w_i x_i + \sum_i \sum_{j<i} v_i v_j x_i x_j$$

- 黑色连线部分的Addition，代表FM的一阶特征$\sum_i w_i x_i$
- 红色连线部分的Inner Product，代表FM的二阶特征组合$\sum_i \sum_{j<i} v_i v_j x_i x_j$

3. deep部分结构如下：



- 尽管不同field的输入长度不同，但是embedding之后向量的长度均为K。
- Hidden Layer层为简单的全连接，输入是m个拼接后的Dense向量，输出公式为：

$$y_{DNN} = F(W^T X + b)$$

4. DeepFM预测结果为：

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

tensorFlow-deepFM (https://github.com/ChenglongChen/tensorflow-DeepFM) DeepFM模型理论和实践转载 (https://www.jianshu.com/p/bf07f73986a6)

模型对比

- **LR**：缺乏特征组合能力，需人工做特征工程
- **GBDT+LR**：Facebook提出的CTR预估方法，特种组合能力不强，对高维的稀疏特征应对乏力
- **FM**：台湾大学，具有较强的二阶特征组合能力，高阶特征组合应对乏力
- **Wide&Deep**：Google 2016年提出的CTR预估方法，较好地利用了低阶和高阶组合特征。但是wide部分依旧需要特征工程，其二阶特征组合需要人工挑选
- **DeepFM** ：华为诺亚方舟实验室和哈工大 2017年提出的CTR预估方法，其实是Wide&Deep的变体，把wide部分由LR转变为FM。所以更好地利用了低阶和高阶的组合特征。而且免去了人工特种组合。

Re链接 (https://www.jianshu.com/p/0de1c2714ffb)
推荐系统遇上深度学习(三)--DeepFM模型理论和实践转载 (https://www.jianshu.com/p/bf07f73986a6)

In [ ]:

线下指标

评分预测评价指标

- RMSE 根均方差（Root Mean Square Error）:对大误差更敏感，对预测算法评价较为严格

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(observed_i - predicted_i)^2}{N}}$$

- MAE 平均绝对误差（Mean Absolute Error）

$$MAE = \frac{\sum_{i=1}^{n} |observed_i - predicted_i|}{N}$$

推荐列表评价指标

只考虑推荐集合正确性，未考虑推荐顺序。 设：推荐系统推荐的购买列表：S_{pred}，用户实际购买列表：S_{ture}，则：

- 准确率(Precision)

$$P = \frac{|S_{pred} \cap S_{ture}|}{|S_{ture}|}$$

- 召回率(Recall)

$$R = \frac{|S_{pred} \cap S_{ture}|}{|S_{pred}|}$$

- F1值(F1)

$$F_1 = \frac{2 * P * R}{P + R}$$

- AUC，ROC

推荐顺序评价指标

- 平均准确率 MAP
- 归一化折扣增益值 NDCG

线上评估指标

- 点击率 CTR