因子机深入解析

Mar 10, 2017

在组内做过一次因子机的技术分享，这里将内容以及自己的思考记录如下。

综述

2010年，日本大阪大学(Osaka University)的 Steffen Rendle 在矩阵分解(MF)、SVD++[2]、PITF[3]、FPMC[4] 等基础之上，归纳出针对高维稀疏数据的因子机(Factorization Machine, FM)模型[11]。因子机模型可以将上述模型全部纳入一个统一的框架进行分析。并且，Steffen Rendle 实现了一个单机多线程版本的 libFM。在随后的 KDD Cup 2012, track2 广告点击率预估(pCTR)中，国立台湾大学[4]和 Opera Solutions[5] 两只队伍都采用了 FM，并获得大赛的冠亚军而使得 FM 名声大噪。随后，台湾大学的 Yuchin Juan 等人在总结自己在两次比赛中的经验以及 Opera Solutions 队伍中使用的 FM 模型的总结，提出了一般化的 FFM 模型[6]，并实现了单机多线程版的 libFFM，并做了深入的试验研究。事实上，Opera Solutions 在比赛中用的 FM 就是FFM。

将 FM 向更高阶推广的工作也有一些，例如 Steffen Rendle 在论文[11]中提出一般化的 d-way FM，他将二阶组合的FM中的二阶项替换成d阶组合项，可以利用 FM 相同的处理技巧将计算时间复杂度降低为线性时间复杂度。这个模型的缺点在于只考虑d阶组合，而实际上，低阶组合模式更有意义，因此到目前为止也没有看到谁在实际中使用。针对这种不足，2016年日本NTT通信科学实验室的 Mathieu Blondel 等人在NIPS2016上提出了一般意

义上的高阶 FM 模型，它保留了所有高阶项，并利用 ANOVA kernel 和动态规划算法，将计算时间复杂度降低到线性时间复杂度[12]!

什么是因子机

因子机的定义

机器学习中的建模问题可以归纳为从数据中学习一个函数 $f : R^n \to T$，它将实值的特征向量 $x \in R^n$ 映射到一个特定的集合中。例如，对于回归问题，集合 T 就是实数集 R，对于二分类问题，这个集合可以是 $\{+1, -1\}$. 对于监督学习，通常有一标注的训练样本集合 $D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$。

线性函数是最简单的建模函数，它假定这个函数可以用参数 w 来刻画，

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

对于回归问题，$y = \phi(x)$；而对于二分类问题，需要做对数几率函数变换（逻辑回归）

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

线性模型的缺点是无法学到模型之间的交互，而这在推荐和CTR预估中是比较关键的。例如，CTR预估中常将用户id和广告id onehot 编码后作为特征向量的一部分。

为了学习特征间的交叉，SVM通过多项式核函数来实现特征的交叉，实际上和多项式模型是一样的，这里以二阶多项式模型为例

$$\begin{aligned}\phi(x) &= w_0 + \sum_i w_i x_i + \sum_i \sum_{j<i} w_{ij} x_i x_j \\ &= w_0 + \mathbf{w_1}^T \mathbf{x} + \mathbf{x}^T \mathbf{W_2} \mathbf{x}\end{aligned}$$

多项式模型的问题在于二阶项的参数过多，设特征维数为 n，那么二阶项的参数数目为 $n(n+1)/2$。 对于广告点击率预估问题，由于存在大量id特征，导致 n 可能为 10^7 维，这样一来，模型参数的 量级为 10^{14}，这比样本量 4×10^7 多得多[6]！这导致只有极少数的二阶组合模式才能在样本中找到， 而绝大多数模式在样本中找不到，因而模型无法学出对应的权重。例如，对于某个w_{ij}，样本中找不到$x_i = 1, x_j = 1$ （这里假定所有的特征都是离散的特征，只取0和1两个值）这种样本，那么w_{ij}的梯度恒为0，从而导致参数学习失败!

很容易想到，可以对二阶项参数施加某种限制，减少模型参数的自由度。FM 施加的限制是要求二阶项系数矩阵是低秩的，能够分解为低秩矩阵的乘积

$$\mathbf{W_2} = \mathbf{V}^T\mathbf{V}, V \in \mathbb{R}^{k \times n}$$
$$w_{ij} = \mathbf{v_i}^T\mathbf{v_j}, \mathbf{v_i} \in \mathbb{R}^k$$
$$\mathbf{V} = [\mathbf{v_1}, \ldots, \mathbf{v_n}]$$

这样一来，就将参数个数减少到 kn，可以设置较少的k值（一般设置在100以内，$k << n$），极大地减少模型参数，增强模型泛化能力，这跟矩阵分解的方法是一样的。向量 $\mathbf{v_i}$ 可以解释为第i个特征对应的隐因子或隐向量。 以user和item的推荐问题为例，如果该特征是user，可以解释为用户向量，如果是item，可以解释为物品向量。

计算复杂度

因为引入和二阶项，如果直接计算，时间复杂度将是 $O(\bar{n}^2)$，$\bar{n}$ 是特征非零特征数目， 可以通过简单的数学技巧将时间复杂度减少到线性时间复杂度。

基于一个基本的观察，齐二次交叉项之和可以表达为平方和之差

$$\sum_i \sum_{j<i} z_i z_j = \frac{1}{2}\left(\left(\sum_i z_i\right)^2 - \sum_i z_i^2\right)$$

上式左边计算复杂度为$O(n^2)$，而右边是$O(n)$。根据上式，可以将原表达式中二次项化简为

$$\sum_i \sum_{j<i} w_{ij} x_i x_j = \sum_i \sum_{j<i} \sum_k v_{ik} v_{jk} x_i x_j$$
$$= \frac{1}{2} \sum_k \left(\left(\sum_i v_{ik} x_i\right)^2 - \sum_i v_{ik} x_i^2\right)$$

上式计算时间复杂度是$O(\bar{n})$。

基于梯度的优化都需要计算目标函数对参数的梯度，对FM而言，目标函数对参数的梯度可以利用链式求导法则分解为目标函数对ϕ的梯度和$\frac{\partial \phi}{\partial \theta}$ 的乘积。前者依赖于具体任务，后者可以简单的求得

$$\frac{\partial \phi}{\partial \theta} = \begin{cases} 1, & \text{if } \theta \text{ is } w_0 \\ x_i, & \text{if } \theta \text{ is } w_i \\ x_i \sum_j v_{jk} x_j - v_{ik} x_i^2, & \text{if } \theta \text{ is } w_{ik} \end{cases}$$

优化方案

论文[2]中给出了三种优化方案，它们分别是

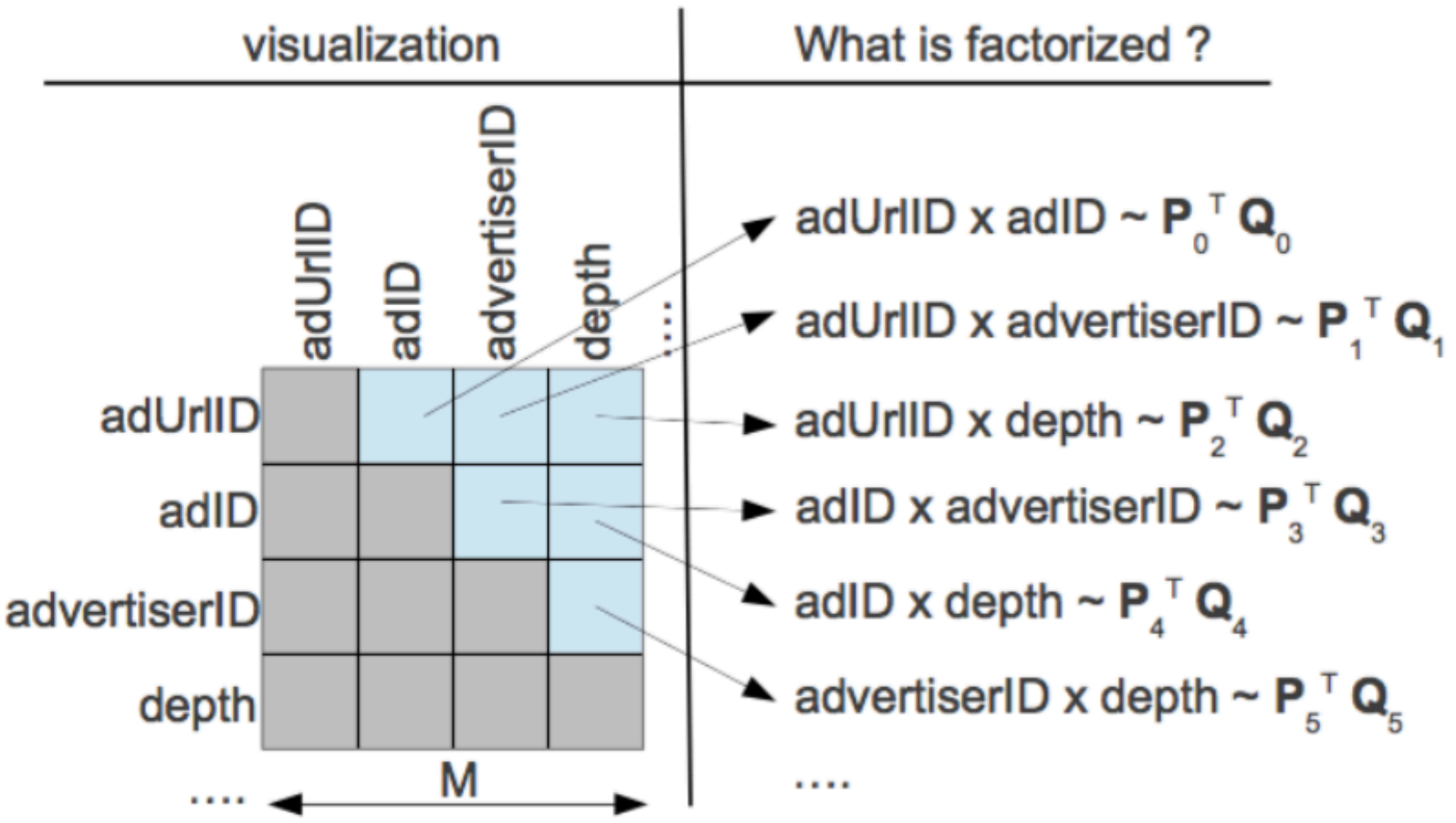
1. 随机梯度下降，这种方案收敛慢而且非常敏感，可以利用现代的一些trick，例如采用AdaGrad 算法，采用自适应学习率，效果相对比较好，论文[6]对FFM就采用这种方案。
2. 交替方向乘子(ALS)，这种方案只适用于回归问题，它每次优化一个参数，把其他参数固定，好处是每次都是一个最小二乘问题，有解析解。
3. 基于蒙特卡罗马尔科夫链的优化方案，论文中效果最好的方案，细节可以参考原文。

FFM

在实际预测任务中，特征往往包含多种id，如果不同id组合时采用不同的隐向量，那么这就是 FFM(Field Factorization Machine) 模型[6]。它将特征按照事先的规则分为多个场(Field)，特征 x_i 属于某个特定的场 f。每个特征将被映射为多个隐向量 $\mathbf{v}_{i1}, \ldots, \mathbf{v}_{if}$，每个隐向量对应一个场。当两个特征 x_i, x_j 组合时，用对方对应的场对应的隐向量做内积！

$$w_{ij} = \mathbf{v}_{i,f_j}^T \mathbf{v}_{j,f_i}$$

f_i, f_j 分别是特征 x_i, x_j 对应的场编号。FFM 由于引入了场，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，但是使得计算复杂度无法通过优化变成线性时间复杂度，每个样本预测的时间复杂度为 $O(\bar{n}^2 k)$，不过FFM的k值通常远小于FM的k值。论文[6]对FFM在Criteo和Avazu两个任务（Kaggle上的两个CTR预估比赛）上进行了试验，结果表明 FFM 的成绩优于 FM。事实上，FM 可以看做只有一个场的 FFM。



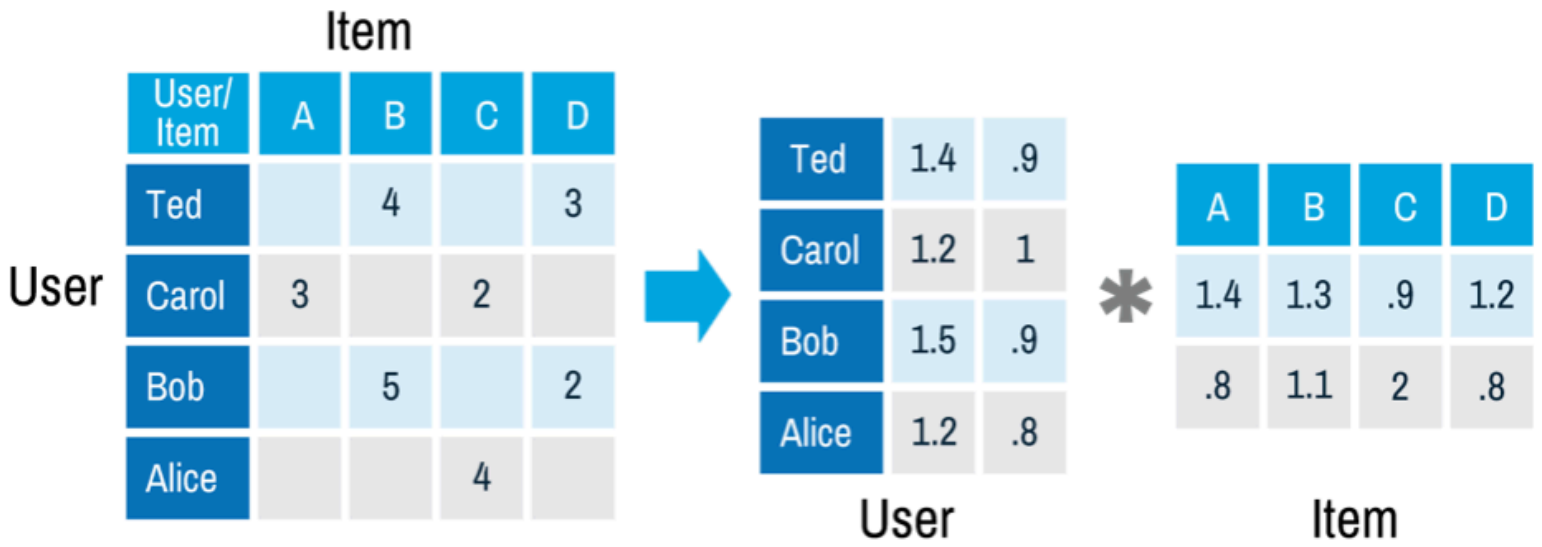
Model and implementation	parameters	training time (seconds)	public set logloss	public set rank	private set logloss	private set rank
LM-SG	$\eta = 0.2, \lambda = 0, t = 13$	527	0.46262	93	0.46224	91
LM-LIBLINEAR-CD	$s = 7, c = 2$	1,417	0.46239	91	0.46201	89
LM-LIBLINEAR-Newton	$s = 0, c = 2$	7,164	0.46602	225	0.46581	222
Poly2-SG	$\eta = 0.2, \lambda = 0, B = 10^7, t = 10$	12,064	0.44973	14	0.44956	14
Poly2-LIBLINEAR-Hash-CD	$s = 7, c = 2$	24,771	0.44893	13	0.44873	13
FM	$\eta = 0.05, \lambda = 2 \times 10^{-5}, k = 40, t = 8$	2,022	0.44930	14	0.44922	14
FM	$\eta = 0.05, \lambda = 2 \times 10^{-5}, k = 100, t = 9$	4,020	0.44867	11	0.44847	11
LIBFM	$\lambda = 40, k = 40, t = 20$	23,700	0.45012	14	0.45000	15
LIBFM	$\lambda = 40, k = 40, t = 50$	131,000	0.44904	14	0.44887	14
LIBFM	$\lambda = 40, k = 100, t = 20$	54,320	0.44853	11	0.44834	11
LIBFM	$\lambda = 40, k = 100, t = 50$	398,800	0.44794	9	0.44778	8
FFM	$\eta = 0.2, \lambda = 2 \times 10^{-5}, k = 4, t = 9$	6,587	0.44612	3	0.44603	3

(a) `Criteo`

Model and implementation	parameters	training time (seconds)	public set logloss	public set rank	private set logloss	private set rank
LM-SG	$\eta = 0.2, \lambda = 0, t = 10$	164	0.39018	57	0.38833	64
LM-LIBLINEAR-CD	$s = 7, c = 1$	417	0.39131	115	0.38944	119
LM-LIBLINEAR-Newton	$s = 0, c = 1$	650	0.39269	182	0.39079	183
Poly2-SG	$\eta = 0.2, \lambda = 0, B = 10^7, t = 10$	911	0.38554	10	0.38347	10
Poly2-LIBLINEAR-Hash-CD	$s = 7, c = 1$	1,756	0.38516	10	0.38303	9
Poly2-LIBLINEAR-Hash-Newton	$s = 0, c = 1$	27,292	0.38598	11	0.38393	11
FM	$\eta = 0.05, \lambda = 2 \times 10^{-5}, k = 40, t = 8$	574	0.38621	11	0.38407	11
FM	$\eta = 0.05, \lambda = 2 \times 10^{-5}, k = 100, t = 9$	1,277	0.38740	17	0.38531	15
LIBFM	$\lambda = 40, k = 40, t = 20$	18,712	0.39137	122	0.38963	127
LIBFM	$\lambda = 40, k = 40, t = 50$	41,720	0.39786	935	0.39635	943
LIBFM	$\lambda = 40, k = 100, t = 20$	39,719	0.39644	747	0.39470	755
LIBFM	$\lambda = 40, k = 100, t = 50$	91,210	0.40740	1,129	0.40585	1,126
FFM	$\eta = 0.2, \lambda = 2 \times 10^{-5}, k = 4, t = 4$	340	0.38411	6	0.38223	6

(b) `Avazu`

FM与矩阵分解



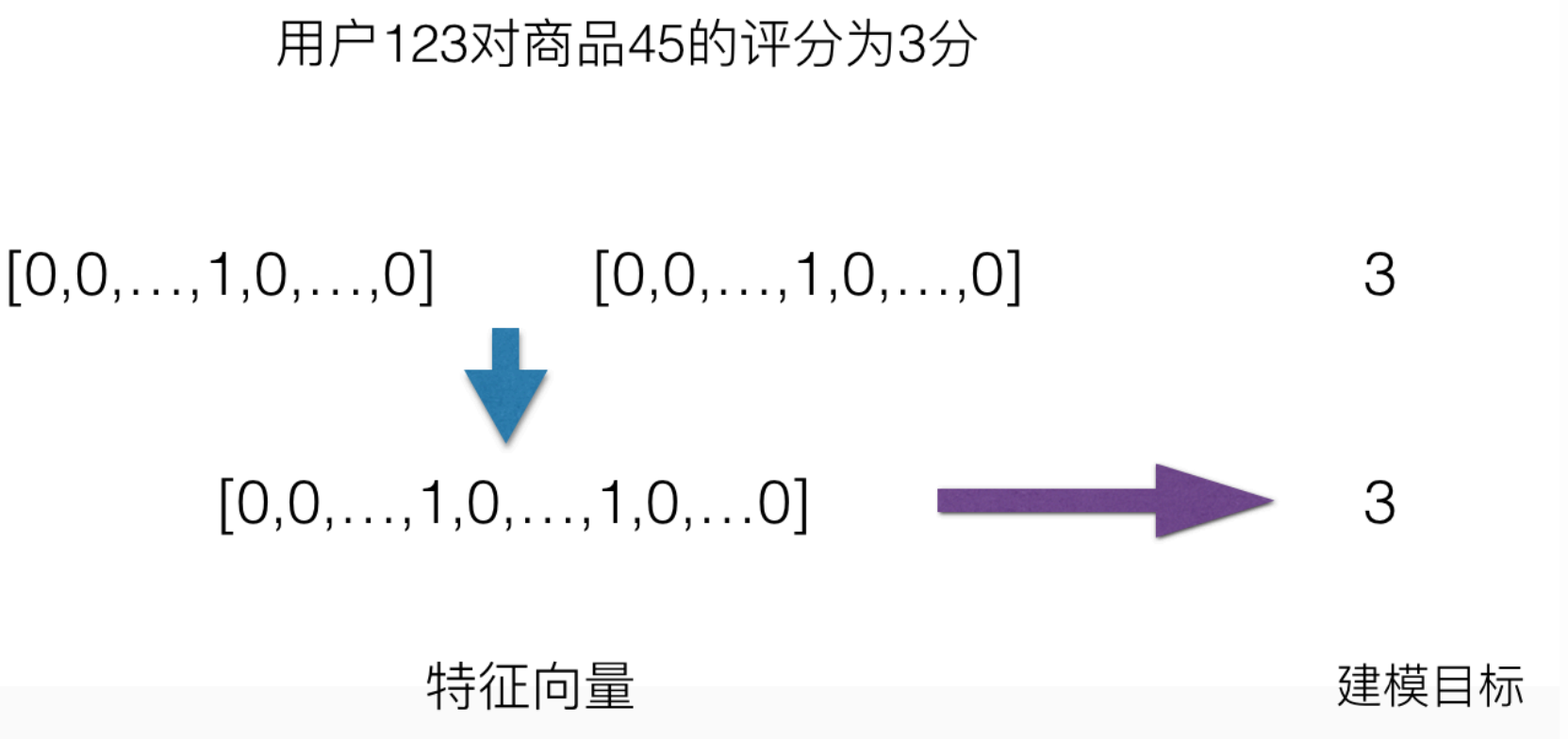
基于矩阵分解的协同过滤是推荐系统中常用的一种推荐方案[7]，从历史数据中收集user对item的评分，可以是显式的打分，也可以是用户的隐式反馈计算的得分。由于user和item数量非常多，有过打分的user和item对通常是十分稀少的，基于矩阵分解的协同过滤是来预测那些没有过行为的user对item的打分，实际上是一个评分预测问题。矩阵分解的方法假设user对item的打分$\hat{r_{ui}}$由下式决定

$$\hat{r_{ui}} = q_i^T p_u + \mu + b_i + b_u$$

其中q_i是第i个item对相应的隐向量，p_u是第u个user对应的隐向量，b_i代表item的偏置，用于解释商品本身的热门和冷门，b_u代表user的偏置，用于解释用户本身的打分偏好（例如

有些人喜欢打低分），μ是常数。即将评分矩阵分解为user矩阵和item矩阵的乘积加上线性项和常数项，而这两个矩阵是低秩的！这些参数通过对最小化经验误差得到。

$$\min_{p,q,b} \sum_{(u,i)\in K} (r_{ui} - \hat{r_{ui}})^2 + \lambda(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2)$$



从上面的叙述来看，FM的二阶矩阵也用了矩阵分解的技巧，那么基于矩阵分解的协同过滤和FM是什么关系呢？以user对item评分预测问题为例，基于矩阵分解的协同过滤可以看做FM的一个特殊例子，对于每一个样本，FM可以看做特征只有userid和itemid的onehot编码后的向量连接而成的向量。从FM和MFCF公式来看，MFCF的用户向量p_u和item向量q_i可以看做FM中的隐向量，用户和item的bias向量b_u, b_i就是FM中的一次项系数，常数μ也和FM中的常数w_0相对应，可以看到，**MFCF就是FM的一个特例**！另外，FM可以采用更多的特征，学习更多的组合模式，这是单个矩阵分解的模型所做不到的！因此，FM比矩阵分解的方法更具普遍性！事实上，现在能用矩阵分解的方法做的方案都直接上FM了！

FM

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

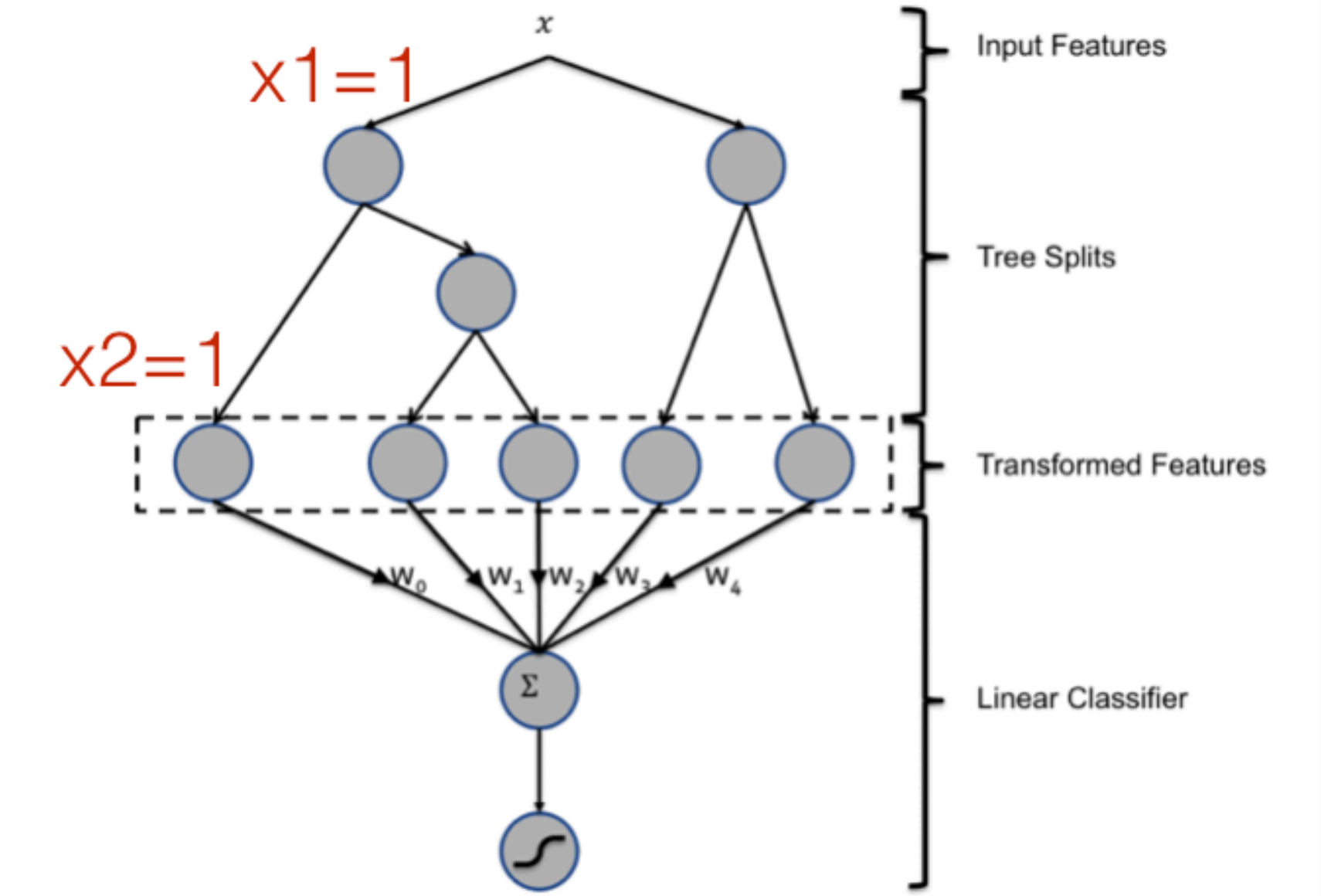
协同过滤

$$\min_{p,q,b} \sum_{(u,i)\in K} (r_{ui} - \hat{r}_{ui})^2 + \lambda(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2)$$

$$s.t. \quad \hat{r}_{ui} = q_i^T p_u + \mu + b_i + b_u$$

FM与决策树

FM和决策树都可以做特征组合，Facebook就用GBDT学习特征的自动组合[8]，决策树可以非常方便地对特征做高阶组合。如图所示，一棵决策的叶子节点实际上就对应一条特征规则，例如最左边的叶子节点就代表一条特征组合规则$x_1 = 1, x_2 = 1$。通过增加树的深度，可以很方便的学习到更高级的非线性组合模式。通过增加树的棵树，可以学习到很多这样的模式，论文[8]采用GBDT来建立决策树，使得新增的决策树能够拟合损失函数的残差。

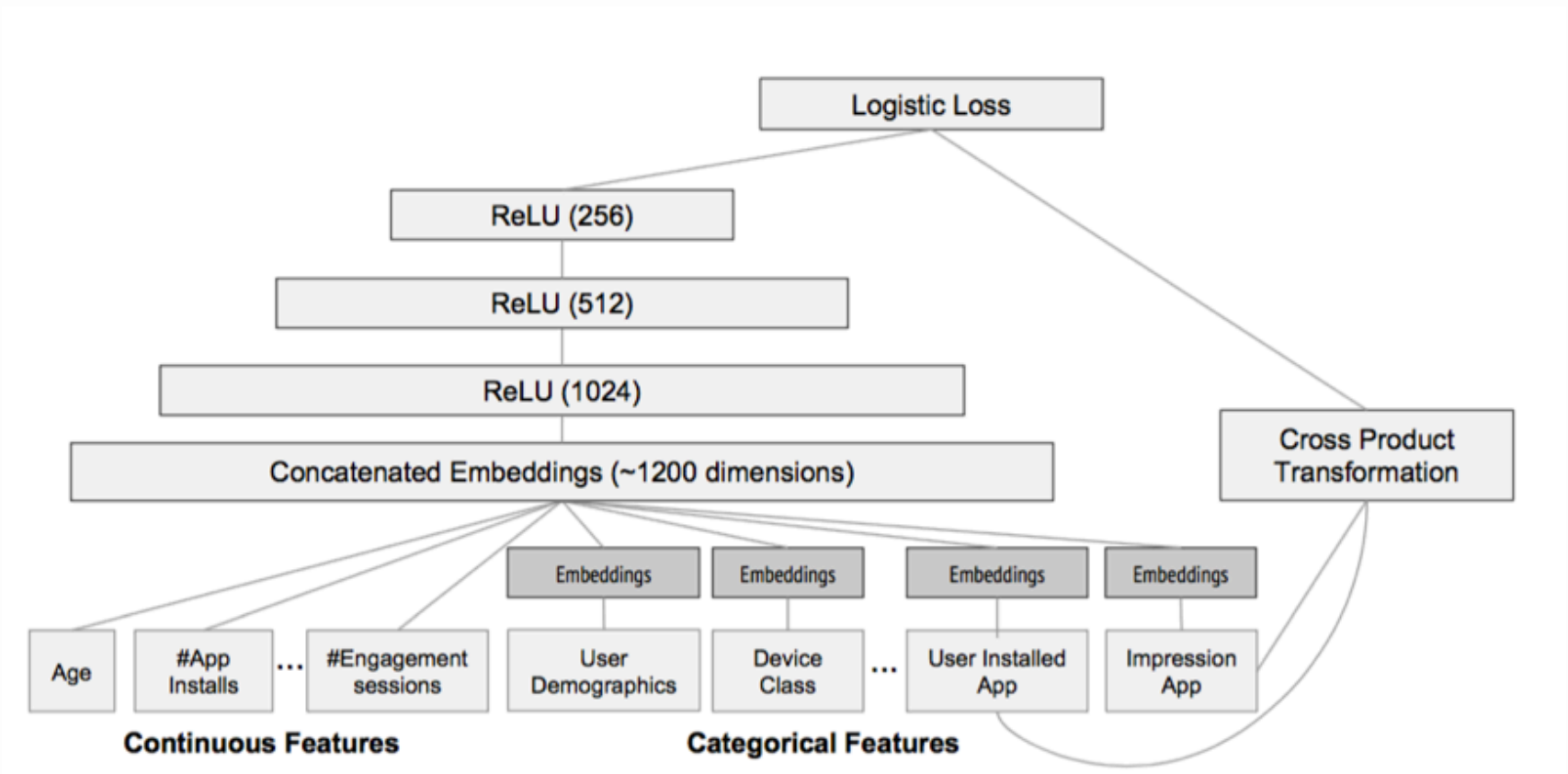


但是，**决策树和二项式模型有一个共同的问题，那就是无法学习到数据中不存在的模式**。例如，对于模式$x_1 = 1, x_2 = 1$，如果这种模式在数据中不存在，或者数量特别少，那么决策树在对特征x_1分裂后，就不会再对x_2分裂了。当数据不是高度稀疏的，特征间的组合模式基本上都能够找到足够的样本，那么决策树能够有效地学习到比较复杂的特征组合；但是在高度稀疏的数据中，二阶组合的数量就足以让绝大多数模式找不到样本，这使得决策树无法学到这些简单的二阶模式，更不用说更高阶的组合了。

FM与神经网络

神经网络天然地难以直接处理高维稀疏的离散特征，因为这将导致神经元的连接参数太多。但是低维嵌入（embedding）技巧可以解决这个问题，词的分布式表达就是一个很好的例子。事实上 **FM就可以看做对高维稀疏的离散特征做 embedding**。上面举的例子其实也可以看做将每一个user和每一个item嵌入到一个低维连续的 embedding 空间中，然后在这个 embedding 空间中比较用户和item的相似性来学习到用户对item的偏好。这跟word2vec[9]词向量学习类似，word2vec 将词 embedding 到一个低维连续空间，词的相似性通过两个词向量的相似性来度量。神经网络对稀疏离散特征做 embedding 后，可以做更复杂的非线性变换，具有比FM跟大的潜力学习到更深层的非线性关系！基于这个思想，2016年，Google提出 wide and deep 模型用作 Google Play的app推荐[10]，它利用神经网络做离散特征和连续特征之间的交叉，神经网络的输出和人工组合较低维度的离散特征一起预测，并且采用端到端的学习，联合优化DNN和LR。如图所示，Catergorial 特征 embedding 到低维连续空间，并和连续特征拼接，构成了1200维的向量，作为神经网络的输入，神经网络采用三隐层结构，激活函数都是采用 ReLU，最后一层神经元的输出 $a^{(lf)}$ 和离散特征 $\mathbf{x}$ 及人工叉积变换后的特征 $\phi(\mathbf{x})$，一起预测

$$P(Y=1|\mathbf{x}) = \sigma\left(\mathbf{w}_{wide}^T[\mathbf{x};\phi(\mathbf{x})] + \mathbf{w}_{deep}^T a^{(lf)} + b\right)$$



注意到，在 wide and deep 模型中，wide部分是通过对用户安装过的APP id和用户Impression App id做叉积变换，解决 embedding 的过泛化问题。 所谓的过泛化，实际上是因为用户的偏好本身就很集中，即使相似的一些 item，用户也只偏好其中一部分，使得query-item矩阵稀疏但是高秩。 而这些信息实际上已经反映在用户已有的行为当中了，因此可以利用这部分信息，单独建立wide部分，解决deep部分的过泛化。

从另一个角度来看，wide和deep部分分别在学习不同阶的特征交叉，deep部分学到高阶交叉，而wide部分学到的是二阶交叉。 后来，有人用FM替换了这里wide部分的二阶交叉，是得模型对高度稀疏的特征的建模更加有效，因为高度稀疏特征简单的叉积变换也难以有效地学到二阶交叉， 这在前面已经叙述过了。因此，很自然的想法就是，用FM替换这里的

二阶交叉，得到DeepFM模型[13]。

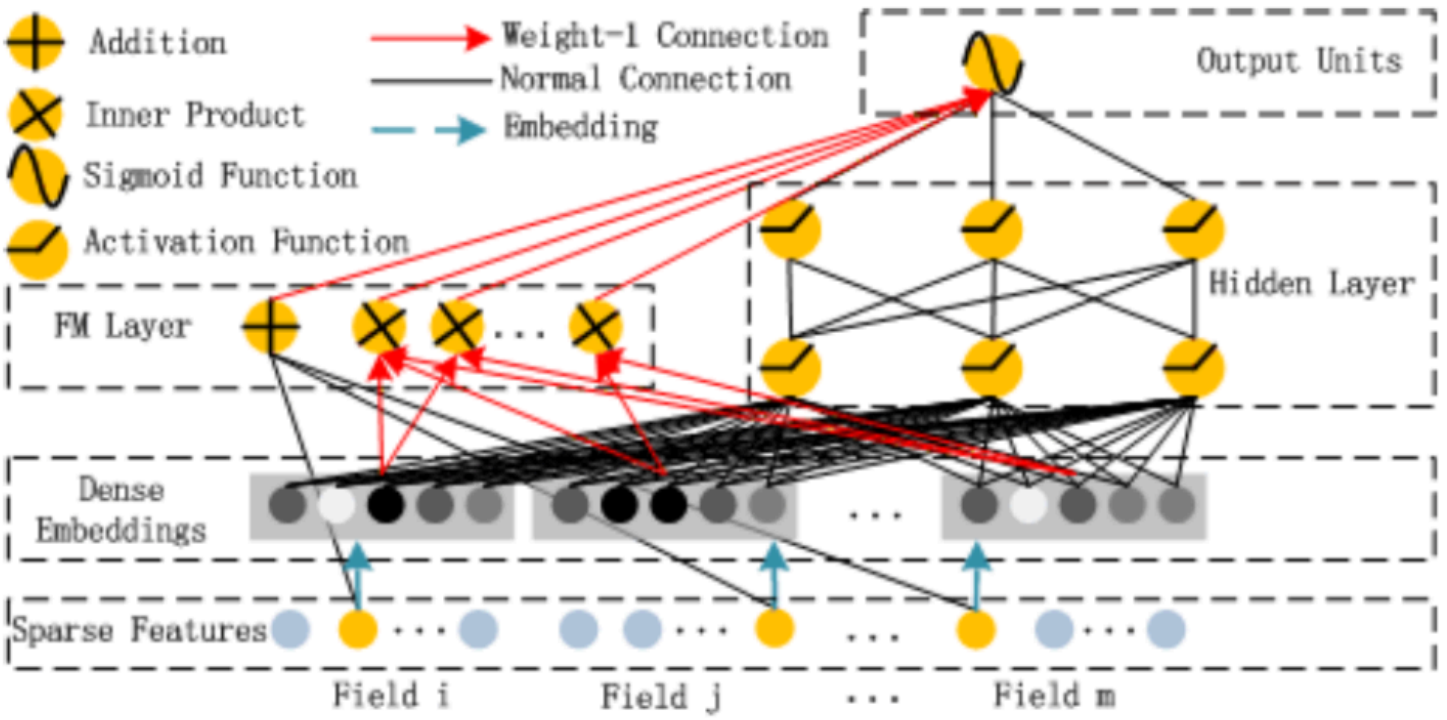


Figure 1: Wide & deep architecture of DeepFM. The wide and deep component share the same input raw feature vector, which enables DeepFM to learn low- and high-order feature interactions simultaneously from the input raw features.

事实上，对于连续特征和非高度稀疏特征的高阶交叉，决策树似乎更加擅长。因此，很自然的想法是将GBDT也加到模型中。 但是问题是，决策树的优化方法和神经网络之类的不兼容，因此无法直接端到端学习。一种解决方案是，利用Boost融合的方案， 将神经网络、FM、LR当做一个模型，先训练一个初步模型，然后在残差方向上建立GBDT模型，实现融合。 微软的一篇文献[14]也证实，Boost方式融合DNN和GBDT方案相比其他融合方案更优，因此这也不失为一种可行的探索方向！

Cross Net

为了将FM推广到高阶组合，一系列的变体被研究人员提出，例如 d-way FM[11], 高阶FM[12]，但是应用到实际数据中的工作一直未见报道。 2017年，Google的研究人员从另一种思路触发，融合了残差网络的思想，设计出叉积网络Cross Net[15]，实现起来简单，可以通过加层的方式方便地扩展到任意阶数。具体来说，首先通过 embedding 层将稀疏特征转换成低维向量表示，将这些向量和连续值特征拼成一个大的d维向量 $\mathbf{x_0} = [\mathbf{x_{e,1}^T}, \mathbf{x_{e,2}^T}, \ldots, \mathbf{x_{e,k}^T}, \mathbf{x_{dense}^T}]$，作为网络的输入。

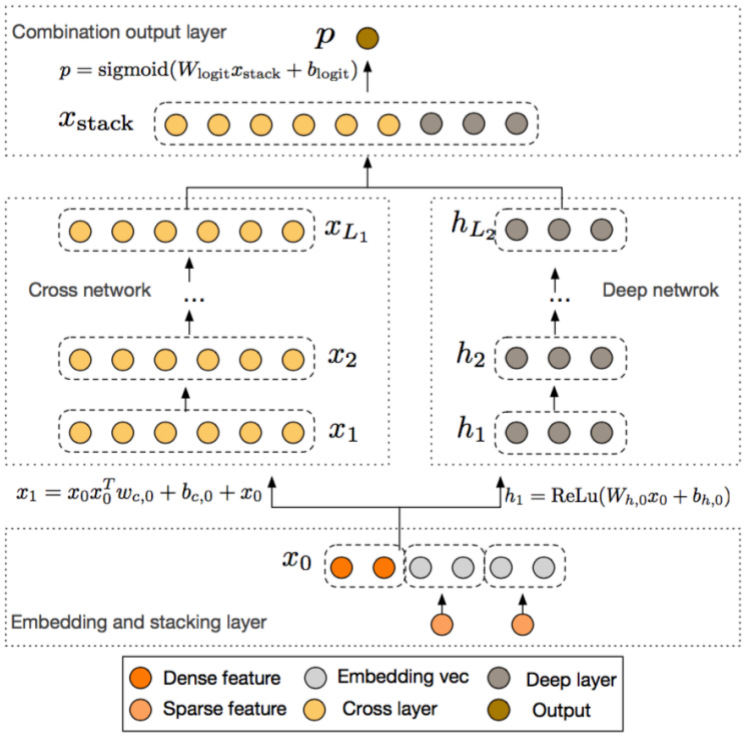


Figure 1: The Deep & Cross Network

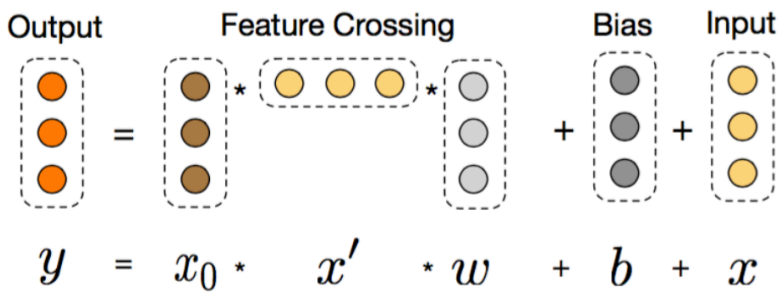


Figure 2: Visualization of a cross layer.

用$\mathbf{x_l}$表示Cross net的的l层的输出，那么cross net的第l层的转换可以表示为

$$\mathbf{x_{l+1}} = \mathbf{x_0 x_l^T w_l} + \mathbf{b_l} + \mathbf{x_l}$$

这里$\mathbf{w_l}, \mathbf{b_l} \in \mathbb{R}^d$是第l层的参数，注意最后一项的存在，这一项是残差链接项，因此前面两项拟合的是残差！ 第一项可以参考图2，实际上只是在x_0的基础上乘上了一个系数！

为了理解这个表达式，我们用$\mathbf{w_{l+1}}$乘以上式，可以理解为最后一层输出的得分（是一个标量）

$$\mathbf{x_{l+1}^T w_{l+1}} = (\mathbf{x_l^T w_l})(\mathbf{x_0^T w_{l+1}}) + \mathbf{b_l^T w_{l+1}} + \mathbf{x_l^T w_{l+1}}$$

如果不考虑常数项和残差项，只保留第一项，并不断的递归会有

$$\mathbf{x_{l+1}^T w_{l+1}} = (\mathbf{x_0^T w_0}) \ldots (\mathbf{x_0^T w_l})(\mathbf{x_0^T w_{l+1}})$$

这表明，Cross Net可以看做FM的直接推广，FM是Cross Net的特例，当l=1且$w_0 = w_1$时，就可以看做是FM！

$$x_1^T w_1 = (x^T W_e^T w_0)(w_0^T W_e x) = x^T W x$$
$$W = W_e^T w_0 w_0^T W_e$$

W_e是 embedding 等效矩阵，x是原始稀疏高维特征向量！

FM 的实现

libFM是FM的最初实现，利用OpenMP实现单机多核的并行！目前，FM已有多种实现

- libFM 单机多线程并行

- difacto 基于ps-lite分布式实现
- fast_tffm 基于 tensorflow 的分布式实现
- xlearn

参考文献

1. Y. Koren, “Factorization meets the neighborhood: a multifaceted collabo- rative filtering model,” in KDD ’08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. New York, NY, USA: ACM, 2008, pp. 426–434.
2. S. Rendle and L. Schmidt-Thieme, “Pairwise interaction tensor factorization for personalized tag recommendation,” in WSDM ’10: Proceedings of the third ACM international conference on Web search and data mining. New York, NY, USA: ACM, 2010, pp. 81–90.
3. S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, “Factorizing personalized markov chains for next-basket recommendation,” in WWW ’10: Proceedings of the 19th international conference on World wide web. New York, NY, USA: ACM, 2010, pp. 811–820.
4. A two-stage ensemble of diverse models for advertisement ranking in KDD Cup 2012[C]//KDDCup. 2012.
5. Jahrer M, Toscher A, Lee J Y, et al. Ensemble of collaborative filtering and feature engineered models for click through rate prediction[C]//KDDCup Workshop. 2012.
6. Juan Y, Zhuang Y, Chin W, et al. Field-aware Factorization Machines for CTR Prediction[C]. conference on recommender systems, 2016: 43-50.
7. Koren Y, Bell R M, Volinsky C, et al. Matrix Factorization Techniques for Recommender Systems[J]. IEEE Computer, 2009, 42(8): 30-37.
8. He X, Pan J, Jin O, et al. Practical Lessons from Predicting Clicks on Ads at Facebook[C]. knowledge discovery and data mining, 2014: 1-9.
9. Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]. neural information processing systems, 2013: 3111-3119.
10. Cheng H T, Koc L, Harmsen J, et al. Wide & Deep Learning for Recommender Systems[C]// The Workshop on Deep Learning for Recommender Systems. ACM, 2016:7-10.
11. Rendle S. Factorization Machines[C]. international conference on data mining, 2010.
12. Blondel M, Fujino A, Ueda N, et al. Higher-Order Factorization Machines[C]. neural information processing systems, 2016: 3351-3359.
13. Guo H, Tang R, Ye Y, et al. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction[J]. arXiv preprint arXiv:1703.04247, 2017.
14. Ling X, Deng W, Gu C, et al. Model Ensemble for Click Prediction in Bing Search

Ads[C]//Proceedings of the 26th International Conference on World Wide Web Companion. International World Wide Web Conferences Steering Committee, 2017: 689-698.
15. Wang R, Fu B, Fu G, et al. Deep & Cross Network for Ad Click Predictions[J]. arXiv preprint arXiv:1708.05123, 2017. MLA

Like

Issue Page

Loading comments...

Write Preview Login with GitHub

Leave a comment

Styling with Markdown is supported

Comment

Powered by Gitment

(c) Copyright all right reserved. 本站所有内容的版权归作者所有，如需转载和使用请与作者联系， 请尊重知识，尊重版权。

tracholar
tracholar

记录每天的心情和收获，不积跬步无以至千里！