深入FFM原理与实践

del2z, 大龙 ・2016-03-03 09:00

FM和FFM模型是最近几年提出的模型，凭借其在数据量比较大并且特征稀疏的情况下，仍然能够得到优秀的性能和效果的特性，屡次在各大公司举办的CTR预估比赛中获得不错的战绩。美团点评技术团队在搭建DSP的过程中，探索并使用了FM和FFM模型进行CTR和CVR预估，并且取得了不错的效果。本文旨在把我们对FM和FFM原理的探索和应用的经验介绍给有兴趣的读者。

前言

在计算广告领域，点击率CTR（click-through rate）和转化率CVR（conversion rate）是衡量广告流量的两个关键指标。准确的估计CTR、CVR对于提高流量的价值，增加广告收入有重要的指导作用。预估CTR/CVR，业界常用的方法有人工特征工程 + LR(Logistic Regression)、GBDT(Gradient Boosting Decision Tree) + LR[1] (http://blog.csdn.net/lilyth_lilyth/article/details/48032119)[2] (http://www.cnblogs.com/Matrix_Yao/p/4773221.html)[3] (http://blog.csdn.net/lilyth_lilyth/article/details/48032119)、FM（Factorization Machine）[2] (http://www.cnblogs.com/Matrix_Yao/p/4773221.html)[7] (http://www.algo.uni-konstanz.de/members/rendle/pdf/Rendle2010FM.pdf)和FFM（Field-aware Factorization Machine）[9] (http://www.csie.ntu.edu.tw/~r01922136/slides/ffm.pdf)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[4] (https://www.kaggle.com/c/criteo-display-ad-challenge)[5] (https://www.kaggle.com/c/avazu-ctr-prediction)。

考虑到FFM模型在CTR预估比赛中的不俗战绩，美团点评技术团队在搭建DSP（Demand Side Platform）[6] (https://en.wikipedia.org/wiki/Demand-side_platform)平台时，在站内CTR/CVR的预估上使用了该模型，取得了不错的效果。本文是基于对FFM模型的深度调研和使用经验，从原理、实现和应用几个方面对FFM进行探讨，希望能够从原理上解释FFM模型在点击率预估上取得优秀效果的原因。因为FFM是在FM的基础上改进得来的，所以我们首先引入FM模型，本文章节组织方式如下：

1. 首先介绍FM的原理。
2. 其次介绍FFM对FM的改进。
3. 然后介绍FFM的实现细节。
4. 最后介绍模型在DSP场景的应用。

FM原理

FM（Factorization Machine）是由Konstanz大学Steffen Rendle（现任职于Google）于2010年最早提出的，旨在解决稀疏数据下的特征组合问题[7] (http://www.algo.uni-konstanz.de/members/rendle/pdf/Rendle2010FM.pdf)。下面以一个示例引入FM模型。假设一个广告分类的问题，根据用户和广告位相关的特征，预测用户是否点击了广告。源数据如下[8] (http://www.cs.cmu.edu/~wcohen/10-605/2015-guest-lecture/FM.pdf)

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label，Country、Day、Ad_type是特征。由于三种特征都是categorical类型的，需要经过独热编码（One-Hot Encoding）转换成数值型特征。

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

由上表可以看出，经过One-Hot编码之后，大部分样本数据特征是比较稀疏的。上面的样例中，每个样本有7维特征，但平均仅有3维特征具有非零值。实际上，这种情况并不是此例独有的，在真实应用场景中这种情况普遍存在。例如，CTR/CVR预测时，用户的性

别、职业、教育水平、品类偏好，商品的品类等，经过One-Hot编码转换后都会导致样本数据的稀疏性。特别是商品品类这种类型的特征，如商品的末级品类约有550个，采用One-Hot编码生成550个数值特征，但每个样本的这550个特征，有且仅有一个是有效的（非零）。由此可见，数据稀疏性是实际问题中不可避免的挑战。

One-Hot编码的另一个特点就是导致特征空间大。例如，商品品类有550维特征，一个categorical特征转换为550维数值特征，特征空间剧增。

同时通过观察大量的样本数据可以发现，某些特征经过关联之后，与label之间的相关性就会提高。例如，“USA”与“Thanksgiving”、“China”与“Chinese New Year”这样的关联特征，对用户的点击有着正向的影响。换句话说，来自“China”的用户很可能会在“Chinese New Year”有大量的浏览、购买行为，而在“Thanksgiving”却不会有特别的消费行为。这种关联特征与label的正向相关性在实际问题中是普遍存在的，如“化妆品”类商品与“女”性，“球类运动配件”的商品与“男”性，“电影票”的商品与“电影”品类偏好等。因此，引入两个特征的组合是非常有意义的。

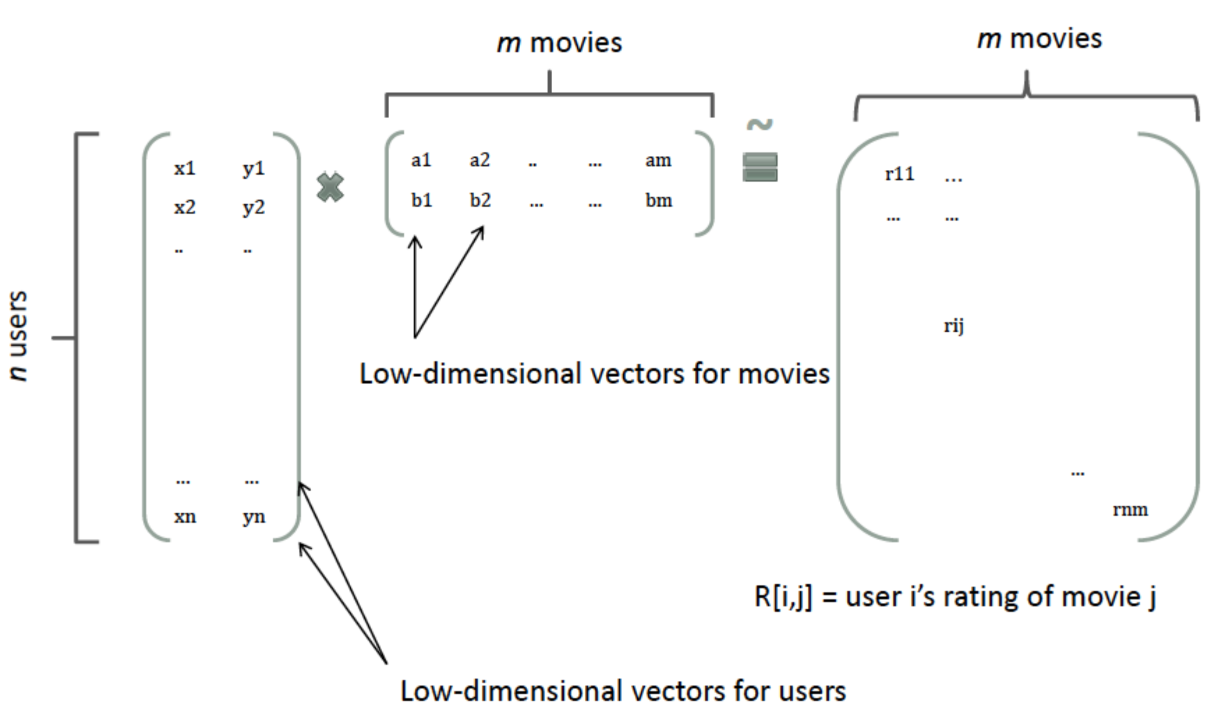
多项式模型是包含特征组合的最直观的模型。在多项式模型中，特征 x_i 和 x_j 的组合采用 $x_i x_j$ 表示，即 x_i 和 x_j 都非零时，组合特征 $x_i x_j$ 才有意义。从对比的角度，本文只讨论二阶多项式模型。模型的表达式如下

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

其中，n 代表样本的特征数量，x_i 是第 i 个特征的值，w_0、w_i、w_{ij} 是模型参数。

从公式(1)可以看出，组合特征的参数一共有 $\frac{n(n-1)}{2}$ 个，任意两个参数都是独立的。然而，在数据稀疏性普遍存在的实际应用场景中，二次项参数的训练是很困难的。其原因是，每个参数 w_{ij} 的训练需要大量 x_i 和 x_j 都非零的样本；由于样本数据本来就比较稀疏，满足“x_i 和 x_j 都非零”的样本将会非常少。训练样本的不足，很容易导致参数 w_{ij} 不准确，最终将严重影响模型的性能。

那么，如何解决二次项参数的训练问题呢？矩阵分解提供了一种解决思路。在model-based的协同过滤中，一个rating矩阵可以分解为user矩阵和item矩阵，每个user和item都可以采用一个隐向量表示[8] (http://www.cs.cmu.edu/~wcohen/10-605/2015-guest-lecture/FM.pdf)。比如在下图中的例子中，我们把每个user表示成一个二维向量，同时把每个item表示成一个二维向量，两个向量的点积就是矩阵中user对item的打分。



类似地，所有二次项参数 w_{ij} 可以组成一个对称阵 $\mathbf{W}$（为了方便说明FM的由来，对角元素可以设置为正实数），那么这个矩阵就可以分解为 $\mathbf{W} = \mathbf{V}^T \mathbf{V}$，$\mathbf{V}$ 的第 j 列便是第 j 维特征的隐向量。换句话说，每个参数 $w_{ij} = \langle \mathbf{v}_i, \mathbf{v}_j \rangle$，这就是FM模型的核心思想。因此，FM的模型方程为（本文不讨论FM的高阶形式）

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \tag{2}$$

其中，$\mathbf{v}_i$ 是第 i 维特征的隐向量，$\langle \cdot, \cdot \rangle$ 代表向量点积。隐向量的长度为 k（$k << n$），包含 k 个描述特征的因子。根据公式(2)，二次项的参数数量减少为 kn个，远少于多项式模型的参数数量。另外，参数因子化使得 $x_h x_i$ 的参数和 $x_i x_j$ 的参数不再是相互独立的，因此我们可以在样本稀疏的情况下相对合理地估计FM的二次项参数。具体来说，$x_h x_i$ 和 $x_i x_j$ 的系数分别为 $\langle \mathbf{v}_h, \mathbf{v}_i \rangle$ 和 $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$，它们之间有共同项 $\mathbf{v}_i$。也就是说，所有包含“x_i 的非零组合特征”（存在某个 $j \neq i$，使得 $x_i x_j \neq 0$）的样本都可以用来学习隐向量 $\mathbf{v}_i$，这很大程度上避免了数据稀疏性造成的影响。而在多项式模型中，w_{hi} 和 w_{ij} 是相互独立的。

显而易见，公式(2)是一个通用的拟合方程，可以采用不同的损失函数用于解决回归、二元分类等问题，比如可以采用MSE（Mean Square Error）损失函数来求解回归问题，也可以采用Hinge/Cross-Entropy损失来求解分类问题。当然，在进行二元分类时，FM的输出需要经过sigmoid变换，这与Logistic回归是一样的。直观上看，FM的复杂度是 $O(kn^2)$。但是，通过公式(3)的等式，FM的

二次项可以化简，其复杂度可以优化到 $O(kn)$[7] (http://www.algo.uni-konstanz.de/members/rendle/pdf/Rendle2010FM.pdf)。由此可见，FM可以在线性时间对新样本作出预测。

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j = \frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} x_i \right)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2 \right) \tag{3}$$

我们再来看一下FM的训练复杂度，利用SGD（Stochastic Gradient Descent）训练模型。模型各个参数的梯度如下

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

其中，$v_{j,f}$ 是隐向量 $\mathbf{v}_j$ 的第 f 个元素。由于 $\sum_{j=1}^{n} v_{j,f} x_j$ 只与 f 有关，而与 i 无关，在每次迭代过程中，只需计算一次所有 f 的 $\sum_{j=1}^{n} v_{j,f} x_j$，就能够方便地得到所有 $v_{i,f}$ 的梯度。显然，计算所有 f 的 $\sum_{j=1}^{n} v_{j,f} x_j$ 的复杂度是 $O(kn)$；已知 $\sum_{j=1}^{n} v_{j,f} x_j$ 时，计算每个参数梯度的复杂度是 $O(1)$；得到梯度后，更新每个参数的复杂度是 $O(1)$；模型参数一共有 $nk+n+1$ 个。因此，FM参数训练的复杂度也是 $O(kn)$。综上可知，FM可以在线性时间训练和预测，是一种非常高效的模型。

FM与其他模型的对比

FM是一种比较灵活的模型，通过合适的特征变换方式，FM可以模拟二阶多项式核的SVM模型、MF模型、SVD++模型等[7] (http://www.algo.uni-konstanz.de/members/rendle/pdf/Rendle2010FM.pdf)。

相比SVM的二阶多项式核而言，FM在样本稀疏的情况下是有优势的；而且，FM的训练/预测复杂度是线性的，而二项多项式核SVM需要计算核矩阵，核矩阵复杂度就是N平方。

相比MF而言，我们把MF中每一项的rating分改写为 $r_{ui} \sim \beta_u + \gamma_i + x_u^T y_i$，从公式(2)中可以看出，这相当于只有两类特征 u 和 i 的FM模型。对于FM而言，我们可以加任意多的特征，比如user的历史购买平均值，item的历史购买平均值等，但是MF只能局限在两类特征。SVD++与MF类似，在特征的扩展性上都不如FM，在此不再赘述。

FFM原理

FFM（Field-aware Factorization Machine）最初的概念来自Yu-Chin Juan（阮毓钦，毕业于中国台湾大学，现在美国Criteo工作）与其比赛队员，是他们借鉴了来自Michael Jahrer的论文[14] (https://kaggle2.blob.core.windows.net/competitions/kddcup2012/2748/media/Opera.pdf)中的field概念提出了FM的升级版模型。通过引入field的概念，FFM把相同性质的特征归于同一个field。以上面的广告分类为例，“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，商品的末级品类编码生成了550个特征，这550个特征都是说明商品所属的品类，因此它们也可以放到同一个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，包括用户性别、职业、品类偏好等。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 $\mathbf{v}_{i,f_j}$。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。

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_{i,f_j}, \mathbf{v}_{j,f_i} \rangle x_i x_j \tag{4}$$

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 $O(kn^2)$。

下面以一个例子简单说明FFM的特征组合方式[9] (http://www.csie.ntu.edu.tw/~r01922136/slides/ffm.pdf)。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，绿色是此样本的特征取值。二次项的系数是通过与特征field相关的隐向量点积得到的，二次项共有 $\frac{n(n-1)}{2}$ 个。

FFM实现

Yu-Chin Juan实现了一个C++版的FFM模型，源码可从Github下载[10] (https://github.com/guestwalk/libffm)。这个版本的FFM省略了常数项和一次项，模型方程如下。

$$\phi(\mathbf{w}, \mathbf{x}) = \sum_{j_1, j_2 \in \mathcal{C}_2} \langle \mathbf{w}_{j_1, f_2}, \mathbf{w}_{j_2, f_1} \rangle x_{j_1} x_{j_2} \tag{5}$$

其中，$\mathcal{C}_2$ 是非零特征的二元组合，j_1 是特征，属于field f_1，$\mathbf{w}_{j_1,f_2}$ 是特征 j_1 对field f_2 的隐向量。此FFM模型采用logistic loss作为损失函数，和L2惩罚项，因此只能用于二元分类问题。

$$\min_{\mathbf{w}} \sum_{i=1}^{L} \log\left(1 + \exp\{-y_i \phi(\mathbf{w}, \mathbf{x}_i)\}\right) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

其中，$y_i \in \{-1, 1\}$ 是第 i 个样本的label，L 是训练样本数量，λ 是惩罚项系数。模型采用SGD优化，优化流程如下。

Algorithm 1 SGD(tr, va, pa)

$model = \mathbf{init}(tr.n, tr.m, pa)$
$R_{tr} = \mathbf{1}, R_{va} = \mathbf{1}$
if $pa.norm$ **then**
  $R_{tr} = \mathbf{norm}(tr), R_{va} = \mathbf{norm}(va)$
end if
for $it = 1, \cdots, pa.itr$ **do**
  if $pa.rand$ **then**
    $tr.X = \mathbf{shuffle}(tr.X)$
  end if
  for $i = 1, \cdots, tr.l$ **do**
    $\phi = \mathbf{calc\Phi}(tr.X[i], R_{tr}[i], model)$
    $e\phi = \exp\{-tr.Y[i] * \phi\}$
    $L_{tr} = L_{tr} + \log\{1 + e\phi\}$
    $g_\Phi = -tr.Y[i] * e\phi/(1 + e\phi)$
    $model = \mathbf{update}(tr.X[i], R_{tr}[i], model, g_\Phi)$
  end for
  for $i = 1, \cdots, va.l$ **do**
    $\phi = \mathbf{calc\Phi}(va.X[i], R_{va}[i], model)$
    $L_{va} = L_{va} + \log\{1 + \exp\{-va.Y[i] * \phi\}\}$
  end for
end for

参考 $Algorithm$ 1, 下面简单解释一下FFM的SGD优化过程。
算法的输入 tr、va、pa 分别是训练样本集、验证样本集和训练参数设置。

1. 根据样本特征数量（$tr.n$）、field的个数（$tr.m$）和训练参数（pa），生成初始化模型，即随机生成模型的参数；
2. 如果归一化参数 $pa.norm$ 为真，计算训练和验证样本的归一化系数，样本 i 的归一化系数为

$$R[i] = \frac{1}{\|\mathbf{X}[i]\|}$$

3. 对每一轮迭代，如果随机更新参数 $pa.rand$ 为真，随机打乱训练样本的顺序；
4. 对每一个训练样本，执行如下操作
    - 计算每一个样本的FFM项，即公式(5)中的输出 ϕ；
    - 计算每一个样本的训练误差，如算法所示，这里采用的是交叉熵损失函数 $\log(1+e\phi)$；
    - 利用单个样本的损失函数计算梯度 g_Φ，再根据梯度更新模型参数；
5. 对每一个验证样本，计算样本的FFM输出，计算验证误差；
6. 重复步骤3~5，直到迭代结束或验证误差达到最小。

在SGD寻优时，代码采用了一些小技巧，对于提升计算效率是非常有效的。

第一，梯度分步计算。采用SGD训练FFM模型时，只采用单个样本的损失函数来计算模型参数的梯度。

$$\mathcal{L} = \mathcal{L}_{err} + \mathcal{L}_{reg} = \log\left(1 + \exp\{-y_i \phi(\mathbf{w}, \mathbf{x}_i)\}\right) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}_{err}}{\partial \phi} \cdot \frac{\partial \phi}{\partial \mathbf{w}} + \frac{\partial \mathcal{L}_{reg}}{\partial \mathbf{w}}$$

上面的公式表明，$\frac{\partial \mathcal{L}_{err}}{\partial \phi}$ 与具体的模型参数无关。因此，每次更新模型时，只需计算一次，之后直接调用 $\frac{\partial \mathcal{L}_{err}}{\partial \phi}$ 的值即可。对于更新 nfk 个模型参数，这种方式能够极大提升运算效率。

第二，自适应学习率。此版本的FFM实现没有采用常用的指数递减的学习率更新策略，而是利用 nfk 个浮点数的临时空间，自适应地更新学习率。学习率是参考AdaGrad算法计算的[11] (https://en.wikipedia.org/wiki/Stochastic_gradient_descent#AdaGrad)，按如下方式更新

$$w'_{j_1,f_2} = w_{j_1,f_2} - \frac{\eta}{\sqrt{1 + \sum_t (g^t_{w_{j_1,f_2}})^2}} \cdot g_{w_{j_1,f_2}}$$

其中，w_{j_1,f_2} 是特征 j_1 对field f_2 隐向量的一个元素，元素下标未标出；$g_{w_{j_1,f_2}}$ 是损失函数对参数 w_{j_1,f_2} 的梯度；$g^t_{w_{j_1,f_2}}$ 是第 t 次迭代的梯度；η 是初始学习率。可以看出，随着迭代的进行，每个参数的历史梯度会慢慢累加，导致每个参数的学习率逐渐减小。另外，每个参数的学习率更新速度是不同的，与其历史梯度有关，根据AdaGrad的特点，对于样本比较稀疏的特征，学习率高于样本比较密集的特征，因此每个参数既可以比较快速达到最优，也不会导致验证误差出现很大的震荡。

第三，OpenMP多核并行计算。OpenMP是用于共享内存并行系统的多处理器程序设计的编译方案，便于移植和多核扩展[12] (http://openmp.org/wp/openmp-specifications/)。FFM的源码采用了OpenMP的API，对参数训练过程SGD进行了多线程扩展，支持多线程编译。因此，OpenMP技术极大地提高了FFM的训练效率和多核CPU的利用率。在训练模型时，输入的训练参数ns_threads指定了线程数量，一般设定为CPU的核心数，便于完全利用CPU资源。

第四，SSE3指令并行编程。SSE3全称为数据流单指令多数据扩展指令集3，是CPU对数据层并行的关键指令，主要用于多媒体和游戏的应用程序中[13] (http://blog.csdn.net/gengshenghong/article/details/7008704)。SSE3指令采用128位的寄存器，同时操作4个单精度浮点数或整数。SSE3指令的功能非常类似于向量运算。例如，a 和 b 采用SSE3指令相加（a 和 b 分别包含4个数据），其功能是 a 中的4个元素与 b 中4个元素对应相加，得到4个相加后的值。采用SSE3指令后，向量运算的速度更加快捷，这对包含大量向量运算的FFM模型是非常有利的。

除了上面的技巧之外，FFM的实现中还有很多调优技巧需要探索。例如，代码是按field和特征的编号申请参数空间的，如果选取了非连续或过大的编号，就会造成大量的内存浪费；在每个样本中加入值为1的新特征，相当于引入了因子化的一次项，避免了缺少一次项带来的模型偏差等。

FFM应用

在DSP的场景中，FFM主要用来预估站内的CTR和CVR，即一个用户对一个商品的潜在点击率和点击后的转化率。

CTR和CVR预估模型都是在线下训练，然后用于线上预测。两个模型采用的特征大同小异，主要有三类：用户相关的特征、商品相关的特征、以及用户-商品匹配特征。用户相关的特征包括年龄、性别、职业、兴趣、品类偏好、浏览/购买品类等基本信息，以及用户近期点击量、购买量、消费额等统计信息。商品相关的特征包括所属品类、销量、价格、评分、历史CTR/CVR等信息。用户-商品匹

配特征主要有浏览/购买品类匹配、浏览/购买商家匹配、兴趣偏好匹配等几个维度。

为了使用FFM方法，所有的特征必须转换成“field_id:feat_id:value”格式，field_id代表特征所属field的编号，feat_id是特征编号，value是特征的值。数值型的特征比较容易处理，只需分配单独的field编号，如用户评论得分、商品的历史CTR/CVR等。categorical特征需要经过One-Hot编码成数值型，编码产生的所有特征同属于一个field，而特征的值只能是0或1，如用户的性别、年龄段，商品的品类id等。除此之外，还有第三类特征，如用户浏览/购买品类，有多个品类id且用一个数值衡量用户浏览或购买每个品类商品的数量。这类特征按照categorical特征处理，不同的只是特征的值不是0或1，而是代表用户浏览或购买数量的数值。按前述方法得到field_id之后，再对转换后特征顺序编号，得到feat_id，特征的值也可以按照之前的方法获得。

CTR、CVR预估样本的类别是按不同方式获取的。CTR预估的正样本是站内点击的用户-商品记录，负样本是展现但未点击的记录；CVR预估的正样本是站内支付（发生转化）的用户-商品记录，负样本是点击但未支付的记录。构建出样本数据后，采用FFM训练预估模型，并测试模型的性能。

	#(field)	#(feature)	AUC	Logloss
站内CTR	39	2456	0.77	0.38
站内CVR	67	2441	0.92	0.13

由于模型是按天训练的，每天的性能指标可能会有些波动，但变化幅度不是很大。这个表的结果说明，站内CTR/CVR预估模型是非常有效的。

在训练FFM的过程中，有许多小细节值得特别关注。

第一，样本归一化。FFM默认是进行样本数据的归一化，即 $pa.norm$ 为真；若此参数设置为假，很容易造成数据inf溢出，进而引起梯度计算的nan错误。因此，样本层面的数据是推荐进行归一化的。

第二，特征归一化。CTR/CVR模型采用了多种类型的源特征，包括数值型和categorical类型等。但是，categorical类编码后的特征取值只有0或1，较大的数值型特征会造成样本归一化后categorical类生成特征的值非常小，没有区分性。例如，一条用户-商品记录，用户为“男”性，商品的销量是5000个（假设其它特征的值为零），那么归一化后特征“sex=male”（性别为男）的值略小于0.0002，而“volume”（销量）的值近似为1。特征“sex=male”在这个样本中的作用几乎可以忽略不计，这是相当不合理的。因此，将源数值型特征的值归一化到 $[0, 1]$ 是非常必要的。

第三，省略零值特征。从FFM模型的表达式(4)可以看出，零值特征对模型完全没有贡献。包含零值特征的一次项和组合项均为零，对于训练模型参数或者目标值预估是没有作用的。因此，可以省去零值特征，提高FFM模型训练和预测的速度，这也是稀疏样本采用FFM的显著优势。

后记

本文主要介绍了FFM的思路来源和理论原理，并结合源码说明FFM的实际应用和一些小细节。从理论上分析，FFM的参数因子化方式具有一些显著的优势，特别适合处理样本稀疏性问题，且确保了较好的性能；从应用结果来看，站内CTR/CVR预估采用FFM是非常合理的，各项指标都说明了FFM在点击率预估方面的卓越表现。当然，FFM不一定适用于所有场景且具有超越其他模型的性能，合适的应用场景才能成就FFM的“威名”。

参考文献

1. http://blog.csdn.net/lilyth_lilyth/article/details/48032119 (http://blog.csdn.net/lilyth_lilyth/article/details/48032119)
2. http://www.cnblogs.com/Matrix_Yao/p/4773221.html (http://www.cnblogs.com/Matrix_Yao/p/4773221.html)
3. http://www.herbrich.me/papers/adclicksfacebook.pdf (http://www.herbrich.me/papers/adclicksfacebook.pdf)
4. https://www.kaggle.com/c/criteo-display-ad-challenge (https://www.kaggle.com/c/criteo-display-ad-challenge)
5. https://www.kaggle.com/c/avazu-ctr-prediction (https://www.kaggle.com/c/avazu-ctr-prediction)
6. https://en.wikipedia.org/wiki/Demand-side_platform (https://en.wikipedia.org/wiki/Demand-side_platform)
7. http://www.algo.uni-konstanz.de/members/rendle/pdf/Rendle2010FM.pdf (http://www.algo.uni-konstanz.de/members/rendle/pdf/Rendle2010FM.pdf)
8. http://www.cs.cmu.edu/~wcohen/10-605/2015-guest-lecture/FM.pdf (http://www.cs.cmu.edu/~wcohen/10-605/2015-guest-lecture/FM.pdf)
9. http://www.csie.ntu.edu.tw/~r01922136/slides/ffm.pdf (http://www.csie.ntu.edu.tw/~r01922136/slides/ffm.pdf)
10. https://github.com/guestwalk/libffm (https://github.com/guestwalk/libffm)
11. https://en.wikipedia.org/wiki/Stochastic_gradient_descent#AdaGrad (https://en.wikipedia.org/wiki/Stochastic_gradient_descent#AdaGrad)
12. http://openmp.org/wp/openmp-specifications/ (http://openmp.org/wp/openmp-specifications/)