Why FM and FFM

创建关联特征，即特征组合

How

多项式模型

Difficulty

矩阵过于稀疏。

每个参数 Wij 的训练都需要大量的 Xi 和 Xj 都非零的样本。由于样本数据本来就稀疏，满足两者都非零的样本更加少。训练样本的不足，很容易导致参数不准确，最终将严重影响模型的性能。

FM (One parameter per feature)

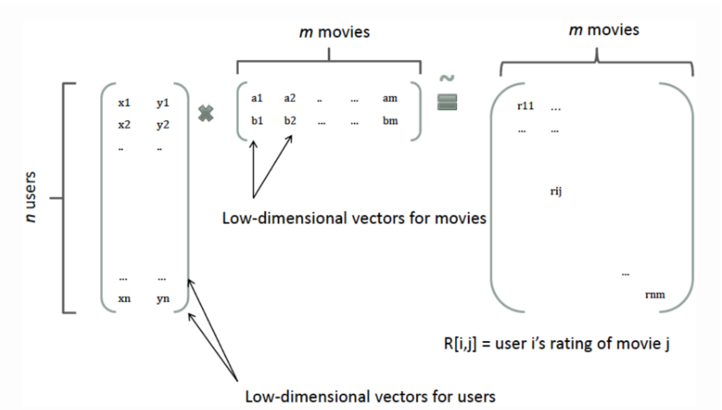
$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

- 其中，n 代表样本的特征数量，xi 是第 i 个特征的值，w0、wi、wij 是模型参数。
- 组合特征的参数一共有 n(n−1)/2 个，任意两个参数都是独立的。
- 每个参数 wij 的训练需要大量 xi 和 xj 都非零的样本；由于样本数据本来就比较稀疏，满足“xi 和 xj 都非零”的样本将会非常少

Idea

使用”矩阵分解” 解决多项式模型/二次项参数的训练问题。

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。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 wij 可以组成一个对称阵 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 W=VTV，V 的第 j 列便是第 j 维特征的隐向量。换句话说，每个参数 wij=⟨ vi,vj⟩，这就是 **FM 模型的核心思想**。

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

具体来说，xhxi 和 xixj 的系数分别为 ⟨vh,vi⟩和 ⟨vi,vj⟩，它们之间有共同项 vi。也就是说，所有包含“xi 的非零组合特征”（存在某个 j≠i，使得 xixj≠0）的样本都可以用来学习隐向量 vi，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，whi 和 wij 是相互独立的。

显而易见，y(x)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 MSE（Mean Square Error）求解回归问题，Hinge/Cross-Entropy 求解分类问题，Logistic 求解二元分类问题 (FM 的输出经过 sigmoid 变换)

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)。由此可见，FM 可以在线性时间对新样本作出预测。

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

模型训练

利用 SGD（Stochastic Gradient Descent）训练模型。模型各个参数的梯度如下

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

FFM

通过引入 field 的概念，FFM 把相同性质的特征归于同一个 field。

在 FFM 中，每一维特征 xi，针对其它特征的每一种 field fj, 都会学习一个隐向量 vi,fj。因此，隐向量不仅与特征相关，也与 field 相关。

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

FM: V_i 第 i 维特征的隐向量

FFM: V_i becomes $[V_{i,f1}, V_{i,f2}, \ldots, V_{i,fj}, \ldots]$ 第 i 维特征的隐向量对每个 field j 都有个单独的隐向量

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, 即 n 个隐向量。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。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。

Yu-Chin Juan 实现了一个 C++版的 FFM 模型。

这个版本的 FFM 省略了常数项和一次项，模型方程如下。

$$\phi(\mathbf{w}, \mathbf{x}) = \sum_{j_1, j_2 \in C_2} \langle \mathbf{w}_{j_1,f_2}, \mathbf{w}_{j_2,f_1} \rangle x_{j_1} x_{j_2}$$

其中，C2 是非零特征的二元组合，j1 是特征，属于 field f1，wj1,f2 是特征 j1 对 field f2 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，yi∈{−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)

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

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 $R[i]=1/\|X[i]\|$
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，即输出 ϕ；
     $$\phi(\mathbf{w}, \mathbf{x}) = \sum_{j_1, j_2 \in \mathcal{C}_2} \langle \mathbf{w}_{j_1, f_2}, \mathbf{w}_{j_2, f_1} \rangle x_{j_1} x_{j_2}$$
   - 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。

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

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

第三，OpenMP 多核并行计算

第四，SSE3 指令并行编程。

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。

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 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。**[FFM 默认处理]**
- 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 是非常必要的。**[把数据转换成 libffm 的数据格式前处理完毕]**

- 第三，**省略零值特征**。从 FFM 模型的表达式可以看出，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。**[把数据转换成 libffm 的数据格式前处理完毕]**