因子分解机

原创 gongyouliu 大数据与人工智能 2019-09-04

作者丨gongyouliu

编辑丨Zandy

这是作者的第17篇文章，约1万字，阅读需70分钟

以下为正文：

作者在上篇文章中讲解了《矩阵分解推荐算法》，我们知道了矩阵分解是一类高效的嵌入算法，通过将用户和标的物嵌入低维空间，再利用用户和标的物嵌入向量的内积来预测用户对标的物的偏好得分。本篇文章我们会讲解一类新的算法：**因子分解机(Factorization Machine**，简称**FM**，为了后面书写简单起见，中文简称为**分解机**)，该算法的核心思路来源于矩阵分解算法，矩阵分解算法可以看成是分解机的特例(我们在第三节1中会详细说明)。分解机自从2010年被提出后，由于易于整合交叉特征、可以处理高度稀疏数据，并且效果不错，在推荐系统及广告CTR预估等领域得到了大规模使用，国内很多大厂(如美团、头条等)都用它来做推荐及CTR预估。

本篇文章我们会从**分解机简单介绍、分解机的参数估计与模型价值、分解机与其他模型的关系、分解机的工程实现、分解机的拓展、近实时分解机、分解机在推荐上的应用、分解机的优势**等8个方面来

讲解分解机相关的知识点。期望本文的梳理可以让读者更好地了解分解机的原理和应用价值，并且尝试将分解机算法应用到自己的业务中。

一、分解机简单介绍

分解机最早由Steffen Rendle于2010年在ICDM会议(Industrial Conference on Data Mining)上提出，它是一种通用的预测方法，即使在数据非常稀疏的情况下，依然能估计出可靠的参数，并进行预测。与传统的简单线性模型不同的是，因子分解机考虑了特征间的交叉，对所有特征变量交互进行建模（类似于SVM中的核函数），因此在推荐系统和计算广告领域关注的点击率CTR（click-through rate）和转化率CVR（conversion rate）两项指标上有着良好的表现。此外，FM模型还具备可以用线性时间来计算，可以整合多种信息，以及能够与许多其他模型相融合等优点。

我们常用的简单模型有线性回归模型及logistic回归模型(**LR**)(见下面两个公式)，它们都是简单的线性模型，原理简单，易于理解，并且非常容易训练，对于一般的分类及预测问题，可以提供简单的解决方案。但是，特征之间是彼此独立的，无法拟合特征之间的非线性关系，而现实生活中往往特征之间不是独立的而是存在一定的内在联系，以新闻推荐为例，一般男性用户看军事新闻多，而女性用户喜欢情感类新闻，那么可以看出性别与新闻的类别有一定的关联性，如果能找出这类相关的特征，是非常有意义的，可以显著提升模型预测的准确度。

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

$$\hat{y}(x) = \frac{1}{1 + exp(w_0 + \sum_{i=1}^{n} w_i x_i)}$$

LR模型是CTR预估领域早期最成功的模型，也大量用于推荐算法排序阶段，大多工业推荐排序系统通过整合人工非线性特征，最终采用这种“线性模型+人工特征组合引入非线性”的模式来训练LR模型。因为LR模型具有简单、方便易用、解释强、易于分布式实现等诸多好处，所以目前工业上仍然有不少算法系统采取这种模式。但是，LR模型最大的缺陷就是人工特征工程，耗时费力，浪费大量人力资源来筛选组合非线性特征，那么能否将特征组合的能力体现在模型层面呢？也即，是否有一种模型可以自动化地组合筛选交叉特征呢？答案是肯定的。

其实想达到这一点并不难，如上图在线性模型的计算公式里加入二阶特征组合即可，任意两个特征进行两两组合，可以将这些组合出的特征看作一个新特征，加入线性模型中。而组合特征的权重和一阶特征权重

一样，在训练阶段学习获得。其实这种二阶特征组合的使用方式，和多项式核SVM是等价的(我们在第三节2中会介绍)。借助SVM中核函数的思路，我们可以在线性模型中整合二阶交叉特征，得到如下的模型。

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

虽然这个模型看上去貌似解决了二阶特征组合问题，但是它有个潜在的缺陷：它对组合特征建模，泛化能力比较弱，尤其是在大规模稀疏特征存在的场景下，这个毛病尤其严重。在数据稀疏的情况下，满足交叉项不为0的样本将非常少(非常少的主要原因有，有些特征本来就是稀疏的，很多样本在该特征上是无值的，有些是由于收集该特征成本过大或者由于监管、隐私等原因无法收集到)，当训练样本不足时，很容易导致参数w_{ij}训练不充分而不准确，最终影响模型的效果。特别是对于推荐、广告等这类数据非常稀疏的业务场景来说(这些场景的最大特点就是特征非常稀疏，推荐是由于标的物是海量的，每个用户只对很少的标的物有操作，因此很稀疏，广告是由于很少有用户去点击广告，点击率很低，导致收集的数据量很少，因此也很稀疏)，很多特征之间交叉是没有(或者没有足够多)训练数据支撑的，因此无法很好地学习出对应的模型参数。因此上述整合二阶两两交叉特征的模型并未在工业界得到广泛采用。

那么我们有办法解决该问题吗？其实是有的，我们可以借助矩阵分解的思路，对二阶交叉特征的系数进行调整，让系数不在是独立无关的，从而减少模型独立系数的数量，解决由于数据稀疏导致无法训练出参数的问题，具体是将上面的模型修改为

$$\hat{y}(x) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < v_i, v_j > x_i x_j$$

公式1：分解机模型

其中我们需要估计的模型参数是w_0、$w \in \mathbb{R}^n$、$V \in \mathbb{R}^{n \times k}$。
其中$w = (w_1, w_2, ..., w_n)$，是n维向量。
v_i、v_j是低维向量(k维)，类似矩阵分解中的用户或者标的物特征向量表示。V是由 v_i组成的矩阵。
< , > 是两个k维向量的内积：

$$< v_i, v_j >= \sum_{f=1}^{k} v_{i,f} v_{j,f}$$

v_i就是我们的FM模型核心的分解向量，k是超参数，一般取值较小(100左右)。

利用线性代数的知识，我们知道对于任意对称的正半定矩阵W，只要k足够大，一定存在矩阵V使得$W = V \cdot V^T$(**Cholesky decomposition**)。这说明，FM这种通过分解的方式基本可以拟合任意的二阶交叉特征，只要分解的维度k足够大（首先，W的每个元素都是两个向量的内积，所以一定是对称的，另外，分解机的公式中不包含x_i与x_i自身的交叉，这对应矩阵W的对角元素，所以我们可以任意选择W对角元素足够大，保证W是半正定的)。由于在稀疏情况下，没有足够的训练数据来支撑模型训练，一般选择较小的k，虽然模型表达空间变小了，但是在稀疏情况下可以达到较好的效果，并且有很好的拓展性。

上面我们对分解机产生的背景、具体的模型公式及特点做了简单介绍，下面一节我们来讲解怎么估计分解机的参数以及简单介绍一下分解机可以用于哪些机器学习任务。

二、分解机参数预估与模型价值

本节我们从分解机为什么可以解决稀疏场景下参数估计、计算复杂度、模型求解、分解机可以解决哪几类学习任务四个方面来简要说明。

2.1 分解机的参数估计

对于稀疏数据场景，一般没有足够的数据来直接估计变量之间的交互，但是分解机可以很好地解决这个问题。通过将交叉特征系数做分解，让不同的交叉项之间不再独立，因此一个交叉项的数据可以辅助来估计(训练)另一个交叉项(只要这两个交叉项有一个变量是相同的，比如$x_i x_j$与$x_i x_k$，它们的系数$< v_i, v_j >$和$< v_i, v_k >$共用一个相同的向量v_i)。

分解机模型通过将二阶交叉特征系数做分解，让二阶交叉项的系数不再独立，因此系数数量是远远小于直接在线性模型中整合二阶交叉特征的。分解机的系数个数为$1 + n + kn$，而整合两两二阶交叉的线性模型的系数个数为$1 + n + n^2$。分解机的系数个数是n的线性函数，而整合交叉项的线性模型系数个数是n的指数函数，当n非常大时，训练分解机模型在存储空间及迭代速度上是非常有优势的。

2.2 分解机的计算复杂度

直接从公式1来看，因为我们需要处理所有特征交叉，所以计算复杂度是$O(kn^2)$。但是我们可以通过适当的公式变换与数学计算，将模型复杂度降低到$O(kn)$，变成线性复杂度的预测模型，具体推导过程如下：

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

公式2：FM交叉项的计算可以简化为线性复杂度的计算

从上面公式最后一步可以看到，括号里面复杂度是$O(n)$，加上外层的$\sum_{f=1}^{k}$，最终的时间复杂度是$O(kn)$。进一步地，在数据稀疏情况下，大多数特征x为0，我们只需要对非零的x求和，因此，时间复杂度其实是$O(k\bar{m}_D)$，$\bar{m}_D$是训练样本中平均非零的特征个数。后面我们会说明对于矩阵分解算法来说$\bar{m}_D$=2，因此对于矩阵分解来说，计算量是非常小的。

由于分解机模型可以在线性时间下计算出，对于我们做预测是非常有价值的，特别是对互联网产品有海量用户的情况。拿推荐来说，我们每天需要为每个用户计算推荐(这是离线推荐，实时推荐计算量会更大)，线性时间复杂度可以让整个计算过程更加高效，可以在更短的时间完成计算，节省服务器资源。

2.3 分解机模型求解

分解机模型公式相对简单，完全可导，我们可以用平方损失函数、logit损失函数或者hinge损失函数来学习FM模型。从2的介绍我们知道分解机模型的值可以在线性时间复杂度计算出来，因此FM的模型参数($w_0, \mathbf{w}, \mathbf{V}$)可以在工程实现上高效地利用梯度下降算法(SGD、ALS等)来训练(即我们可以

线性时间复杂度求出下面的e_x，所以在迭代更新参数时是非常高效的，见下面的迭代更新参数的公式)。结合公式1和公式2，我们很容易计算出FM模型的梯度如下：

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

我们记$e_x \overset{def}{=} y - \hat{y}(\mathbf{x})$，针对平方损失函数，具体的参数更新公式如下(未增加正则项，其他损失函数的迭代更新公式类似，也可以很容易推导出)：

$$w_0 \leftarrow w_0 - \gamma e_x$$
$$w_i \leftarrow w_i - \gamma x_i e_x$$
$$v_{i,f} \leftarrow v_{i,f} - \gamma(x_i \sum_{j=1}^{n} v_{j,f} x_j - v_{i,f} x_i^2) e_x$$

其中，$\sum_{j=1}^{n} v_{j,f} x_j$与i无关，因此可以事先计算出来(在做预测求$\hat{y}(x)$或者在更新参数前，这两步都需要计算该量)。上面的梯度计算可以在常数时间复杂度$O(1)$下计算出来。在模型训练更新时，在$O(kn)$时间复杂度下完成对样本$(\mathbf{x}, y)$的更新(如果是稀疏情况下，更新时间复杂度是$O(km(\mathbf{x}))$, $m(\mathbf{x})$是特征$\mathbf{x}$非零元素个数)。我们在第4节工程实现中会细讲怎么进行模型训练。

2.4 模型预测

FM可以用于各类预测任务中，主要包括如下3类：

(1) 回归问题(Regression)

$\hat{y}(x)$直接作为预测项，可以转化为求最小值的最优化问题，具体如下：

$$\min_{w_0^*, w_i^*, v_i^*} \sum_{x \in D} (y - \hat{y}(x))^2$$

其中D是训练数据集，y是$x \in D$对应的真实值。

(2) 二元分类问题 (Binary Classification)

$Sgn(\hat{y}(x))$最为最终的分类，我们可以通过hinge loss或者logit loss来训练二元分类问题。

(3) 排序问题(Ranking)

对于排序学习问题(如pairwise)，我们可以利用排序算法相关的loss函数来训练FM模型，利用FM模型来做排序学习。

在所有上面3类问题中，我们都可以通过增加正则项到目标函数中，避免模型过拟合。

三、分解机与其他模型的关系

分解机的思想是从线性模型中通过增加二阶交叉项来拟合特征之间的交叉，为了拓展到数据稀疏场景并便于计算，吸收了矩阵分解的思想。在这一节中我们来简单讲解一下分解机与其他模型之间的关系，特别是与矩阵分解和SVM之间的关系，让读者更好地了解他们之间的区别与联系，从而更好地理解分解机。

3.1 FM与矩阵分解（MF）的联系

对于隐式协同过滤来说，我们用用户和标的物两类特征来作为FM的特征，特征的维度为$n := |U \cup I|$(用户数+标的物数)，其中U, I分别是用户集和标的物集，我们可以将矩阵分解看成两个类别变量U和I之间的交互(交叉)。显然我们有下面的关系，其中δ是指标变量(indicator variable)。

$$x_j := \delta\left(j = i \vee j = u\right)$$

只有当特征为u或者i时，x_u=1或者x_i=1，这即是用户u对标的物i进行了一次隐式反馈，每个样本中有且只有2个特征不为零(=1)，如下图。

$$(u, i) \rightarrow \mathbf{x} = (\underbrace{0, \ldots, 0, 1, 0, \ldots, 0}_{|U|}, \underbrace{0, \ldots, 0, 1, 0, \ldots, 0}_{|I|}),$$

这时，FM模型可以表示为

$$\hat{y}(\mathbf{x}) = \hat{y}(u, i) = w_0 + \ w_u \ + \ w_i \ + \ < \mathbf{v}_u, \mathbf{v}_i >$$

上面的公式跟包含用户和标的物bias的矩阵分解算法是一样的，所以可以说矩阵分解算法是FM的一种特例。

3.2 FM与SVM的联系

SVM可以表示为输入$\mathbf{x}$变换后的向量与模型参数$\mathbf{w}$的内积：$\hat{y}(\mathbf{x}) =< \phi(\mathbf{x}), \mathbf{w} >$，这里$\phi : \mathbb{R}^n \to \Omega$将输入向量映射到一个复杂的空间，$\phi$通过内积与核函数建立关联：

$$K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, K(\mathbf{x}, \mathbf{z}) =< \phi(\mathbf{x}), \phi(\mathbf{z}) >$$

下面我们分线性核函数和多项式核函数两种情况来说明SVM与FM之间的关系。

(1) 线性核

最简单的线性核函数是：$K_l(\mathbf{x}, \mathbf{z}) := 1+ < \mathbf{x}, \mathbf{z} >$，该核函数对应的映射为

$$\phi(\mathbf{x}) := (1, x_1, x_2, ..., x_n)$$

线性核的SVM模型方程可以写为

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i, \qquad w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^n$$

这对应无二阶交叉项的FM，也即是第五节1高阶分解机中的一阶分解机。

(2) 多项式核

多项式核的SVM可以建模自变量的高阶交叉特征，它的核函数是

$$K(\mathbf{x}, \mathbf{z}) := (< \mathbf{x}, \mathbf{z} > +1)^d$$

当d=2(二阶交叉特征)，对应的映射(其中可行的一个)为

$$\phi(\mathbf{x}) := (1, \sqrt{2}x_1, .., \sqrt{2}x_n, x_1^2, ..., x_n^2, \sqrt{2}x_1x_2, ..., \sqrt{2}x_1x_n, \sqrt{2}x_2x_3, ..., \sqrt{2}x_{n-1}x_n)$$

这时二阶多项式核的SVM模型方程为

$$\hat{y}(\mathbf{x}) = w_0 + \sqrt{2}\sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} w_{i,i}^{(2)} x_i^2 + \sqrt{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n} w_{i,j}^{(2)} x_i x_j$$

其中，

$w_0 \in \mathbb{R},\ \ \mathbf{w} \in \mathbb{R}^n,\ \ \mathbf{W}^2 \in \mathbb{R}^{n\times n}\ (symmetric\ matrix)$。

从2阶多项式核的SVM的模型方程来看，它与FM模型方程(公式1)的主要区别有2点，一是SVM二阶交叉项的参数$w_{i,j}^{(2)}$是独立的(如$w_{i,j}^{(2)}$与$w_{i,k}^{(2)}$是独立的)，而FM中二阶交叉项是有关联的，$< \mathbf{v}_i, \mathbf{v}_j >$与$< \mathbf{v}_i, \mathbf{v}_k >$之间相关，都依赖$\mathbf{v}_i$。二是SVM有变量与自己的交叉(如$x_i^2$等)而FM没有。

(3) 线性核和多项式核下SVM存在的问题

下面我们来说明在数据稀疏情况下，SVM无法很好地学习模型，我们拿隐式反馈的协同过滤(特征的值为0或者1)来说，利用用户特征和标的物特征两类特征来训练模型，预估用户对标的物的偏好。用户特征是n(用户数)维的，标的物特征是m(标的物数)维的，这时每一个样本的特征只有两个特征非零，其他都为零(该用户所在的列及该用户有过行为的标的物这列非零)。数据是相当稀疏的($\frac{2}{m+n}$的数据非零，一般m、n是非常大的，所以非常稀疏)。

针对上面提到的协同过滤数据，线性SVM模型等价为$\hat{y}(\mathbf{x}) = w_0 + w_u + w_i$。从作者的上篇文章《协同过滤推荐系统》，可以知道，该模型非常简单，仅仅整合了用户和标的物的bias，没有用户和标的物嵌入向量的内积项，因此非常容易训练，但是效果不会很好。

同样地，针对上面的协同过滤案例，二阶多项式核的SVM的模型方程现在变为

$$\hat{y}(\mathbf{x}) = w_0 + \sqrt{2}(w_u + w_i) + w_{u,u}^2 + w_{i,i}^{(2)} + \sqrt{2}w_{u,i}^{(2)}$$

从上式可以看到，$w_u, w_{u,u}^{(2)}$其实表达的都是用户相关的特征，我们可以将它们合并为一项，同理，$w_i, w_{i,i}^{(2)}$也可以归并在一起。通过归并，最终模型变为

$$\hat{y}(\mathbf{x}) = w_0 + \sqrt{2}(w_u + w_i) + \sqrt{2}w_{u,i}^{(2)}$$

一般来说，大多数情况一个用户只对某个标的物进行一次隐式反馈，因此针对划分在测试集中的(u,i)对，在训练集中没有数据与之对应，这时，我们无数据用于训练求参数$w_{u,i}^{(2)}$。因此在隐式协同过滤场景下，二阶多项式核的SVM无法很好利用二阶交叉特征，只能训练出用户和标的物的bias(w_u, w_i)，最终效果其实跟线性SVM是一样的。

通过上面的讲解及案例介绍，下面总结一下FM与SVM的主要差异点：

(1) 在稀疏数据情况下，SVM模型相互独立的高阶交叉参数无法得到很好的训练，而FM由于二阶交叉项的参数是通过分解得到的，参数之间是有关联的，所以更容易训练出，特别是在稀疏数据情况下，其他交叉项的训练数据可以用于训练，因此效果相对更好。
(2) FM模型可以直接学习，而SVM往往通过它的对偶形式进行学习；
(3) FM的模型方程与训练数据无关，而SVM在预测时依赖部分训练数据(支持向量)；

到此，我们讲完了FM与矩阵分解及SVM之间的关系及区别，参考文献2中有关于FM与其他更多模型之间的关系介绍，感兴趣的读者可以阅读进行深入学习。

四、分解机的工程实现

前面几节讲解了FM的原理、参数估计方法、与其他模型之间的关系，我们知道FM是一个表达能力很强的模型，并且在线性时间复杂度下可求解，那么具体在工程上怎么训练FM模型呢？本节我们试图讲解一般的方法，本节的思路来源于参考文献1，FM的提出者Rendle给出了FM的工程实现，并且基于该文章的思路，Rendle开源了一个求解FM的高效C++库：libFM，读者可以参考http://www.libfm.org/。另外，参考文献12提供了FM的一种实现，可以利用分解机解决各类回归、分类、排序问题，14、17讲解了怎么分布式训练FM模型，也是比较好的参考文献。

libFM通过SGD、ALS、MCMC三种方法来训练FM模型，下面我们介绍利用SGD来求解FM模型，其他算法读者可以参考该论文，这里不再讲解。在讲解具体的方法之前，我们先来统一符号化FM模型，将该模型的求解转化为求极值的最优化问题，方面后面讲解怎么迭代训练。

通过定义训练集S的损失函数l，一般优化问题可以转化为求所有损失的和的最小值问题，定义如下：

$$OPT(S) := \underset{\Theta}{argmin} \sum_{(\mathbf{x},y)\in S} l(\hat{y}(\mathbf{x}|\Theta), y)$$

对于回归问题，一般用最小平方损失，对应的损失函数为 $l^{LS}(y_1, y_2) := (y_1 - y_2)^2$，对于二分类问题，损失函数可以定义为:$l^{C}(y_1, y_2) := -ln\sigma(y_1 y_2)$，其中$\sigma(x) = \frac{1}{1+e^{-x}}$是logistic函数。

直接学习上述最优化问题容易产生过拟合，一般会加入L_2正则项，增加了正则化的最优化问题转化为：

$$OPT^{REG}(S) := \underset{\Theta}{argmin}\left(\sum_{(\mathbf{x},y)\in S} l(\hat{y}(\mathbf{x}|\Theta), y) + \sum_{\theta\in\Theta} \lambda_\theta \theta^2\right)$$

上述增加了正则项的函数就是我们优化的目标函数，下面的讲解都基于该目标函数。为了后面的算法讲解方便，下面我们先来求目标函数的导数，对于回归问题(最小平方损失)来说，导数为

$$\frac{\partial}{\partial\theta} l^{\mathrm{LS}}(\hat{y}(\mathbf{x}|\Theta), y) = \frac{\partial}{\partial\theta}\left(\hat{y}(\mathbf{x}|\Theta) - y\right)^2 = 2\left(\hat{y}(\mathbf{x}|\Theta) - y\right)\frac{\partial}{\partial\theta}\hat{y}(\mathbf{x}|\Theta),$$

对于分类问题(logit损失)，导数为

$$\frac{\partial}{\partial\theta} l^{\mathrm{C}}(\hat{y}(\mathbf{x}|\Theta), y) = \frac{\partial}{\partial\theta} - \ln\sigma\left(\hat{y}(\mathbf{x}|\Theta)\, y\right) = \left(\sigma\left(\hat{y}(\mathbf{x}|\Theta)\, y\right) - 1\right) y \frac{\partial}{\partial\theta}\hat{y}(\mathbf{x}|\Theta)$$

有了导数，下面我们来讲解怎么用SGD优化方法来求解FM模型。随机梯度下降算法(SGD)是分解类模型的常用迭代求解算法，该方法简单易懂，对各种损失函数效果都不错，并且计算和存储成本相对较低。下面的算法1就是优化FM模型的SGD算法。

算法1：用SGD来求解FM模型

Input：训练集S，正则参数λ，学习率η，方差σ

Output：模型参数$\Theta = (w_0, \mathbf{w}, \mathbf{V})$

初始化：$w_0 \leftarrow 0;\ \mathbf{w} \leftarrow (0, 0, ..., 0);\ \mathbf{V} \sim N(0, \sigma)$

repeat

$w_0 \leftarrow w_0 - \eta\left(\frac{\partial}{\partial w_0} l(\hat{y}(\mathbf{x}|\Theta), y) + 2\lambda_0 w_0\right)$

for $i \in \{1, 2, ..., p\} \wedge x_i \neq 0$ do

$w_i \leftarrow w_i - \eta\left(\frac{\partial}{\partial w_i} l(\hat{y}(\mathbf{x}|\Theta), y) + 2\lambda_{w_i} w_i\right)$

for $f \in \{1, 2, ..., k\}$ do

$v_{i,f} \leftarrow v_{i,f} - \eta\left(\frac{\partial}{\partial v_{i,f}} l(\hat{y}(\mathbf{x}|\Theta), y) + 2\lambda_{v_{i,f}} v_{i,f}\right)$

end

end

until 迭代停止条件达到

上述算法中可以针对不同的参数设置不同的正则化因子$\lambda_0, \lambda_{w_i}, \lambda_{v_{i,f}}$。对于一个训练样本，SGD算法的时间复杂度是

$$O(k\sum_{i=1}^{p}\delta(x_i \neq 0)) := O(kN_z(\mathbf{x}))$$

其中$N_z(\mathbf{x})$是特征向量$\mathbf{x}$中非零元素个数。

从应用的角度来说，读者没必要对求解FM的原理非常清楚，只要会用就可以了。libFM库提供了一个方便的求解FM的工具，该库在单机下运行，对于数据量大一次无法放入内存的情况，可以利用libFM二进制的数据格式，它可以更快地读取数据，并且一批只放部分数据到内存中进行训练。如果数据量非常大，那么我们就需要采用分布式FM模型了。

业界有很多这样的开源工具的，这里推荐一个腾讯的Angel框架，它内置了很多FM算法(及该算法的变种)的实现，并且可以跟Spark配合使用，非常适合工业级FM模型的训练。今年8月22日Angel发布了全新的3.0版本，整合了PyTorch，它在 PyTorch On Angel 上实现了许多算法：包括推荐领域常见的算法（FM，DeepFM，Wide & Deep，xDeepFM，AttentionFM， DCN 和 PNN 等），Angel 擅长推荐模型和图网络模型相关领域（如社交网络分析）。在腾讯内部，腾讯视频，腾讯新闻和微信等都在使用 Angel ，我们公司也在尝试使用Angel。

五、分解机的拓展

前面对分解机的算原理、工程实现等进行了介绍，我们知道分解机是一类非常有价值的模型，在工业界有大量应用，所以有很多人从各个维度对分解机进行了拓展与优化，让分解机预测更加准确从而发挥更大的商业价值，本节我们就来讲解分解机的各种拓展与变式，让大家对分解机有更进一步的认识与了解。

5.1 高阶分解机

传统意义上讲FM都是二阶交叉，计算复杂度可通过数学变换将时间复杂度改进到线性时间，在实际应用中一般也只用到二阶交叉。所谓高阶分解机就是将交叉项拓展到最多d(d>2)个特征的交叉，具

体的模型如下：

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

参考文献2中有对高阶分解机做简单介绍，通过类似2阶分解机的方法也可以将预测计算复杂度降低到线性时间复杂度，但是文章没有细说怎么做。参考文献16对高阶分解机进行了非常深入的介绍。这篇文章发表在NIPS 2016，它解决了三阶甚至更高阶的特征交叉问题。有兴趣的读者可以参考阅读。

5.2 FFM（Field-aware Factoruzation Machine）

在FM的基础上，FFM提出field的概念。一般来说，同一个ID类特征(如推荐中的用户和标的物特征)进行One-hot而产生的所有特征都可以归为同一个 field。在FFM中，对每一个特征x_i，每一个field f_j，学习一个隐向量v_{i,f_j}，不同的特征跟同一个 field 进行关联时使用不同的隐向量。假设总共有 n 个特征，属于 f个field，那么每个特征都用 f 个隐向量来描述，所以总共有n×f 个隐向量。而FM中，一个特征只有一个隐向量，所以FM可以看成FFM中所有特征都属于同一个 field 的特例。相对FM来说，FFM有更多的二阶交叉参数(FFM有$n \times k \times f$个参数，而FM只有$n \times k$)，训练时间会更长，但是在很多情况下效果会更好，具体的模型公式见下图。

$$\hat{y}_{ffm} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < v_{i,f_j}, v_{j,f_i} > x_i x_j$$

内积$< v_{i,f_j}, v_{j,f_i} >$表示让特征i与 特征 j 的 field 关联，同时让特征 j 与 i 的 field 关联，由此可见，FM的交叉是针对特征之间的，而FFM是针对特征与 field 之间的交叉。

感兴趣的读者可以阅读参考文献22了解更多细节，并且论文作者也提供了一个基于C++的FFM的开源实现 https://github.com/ycjuan/libffm 。另外一个基于python的实现参考 https://github.com/aksnzhy/xlearn，包括LR、FM、FFM等常用机器学习模型。

5.3 DeepFM

DeepFM是2017年华为若亚方舟团队提出的一个将FM与DNN有效结合的模型，主要借鉴Google的Wide&Deep论文的思想并进行适当改进，将其中wide部分(logistic回归)换成FM与DNN进特征交叉。wide和deep部分共享原始输入特征向量，这让DeepFM可以直接从原始特征中同时学习低阶和高阶特征交叉，因此不像Wide&Deep模型那样，需要进行复杂的人工特征工程(logistic回归部分需要人工特征工程)，同时训练效率会更高(DeepFM的网络结构参考下面图1)。参考文献中的15、20、28也是关于FM与深度学习结合的拓展。

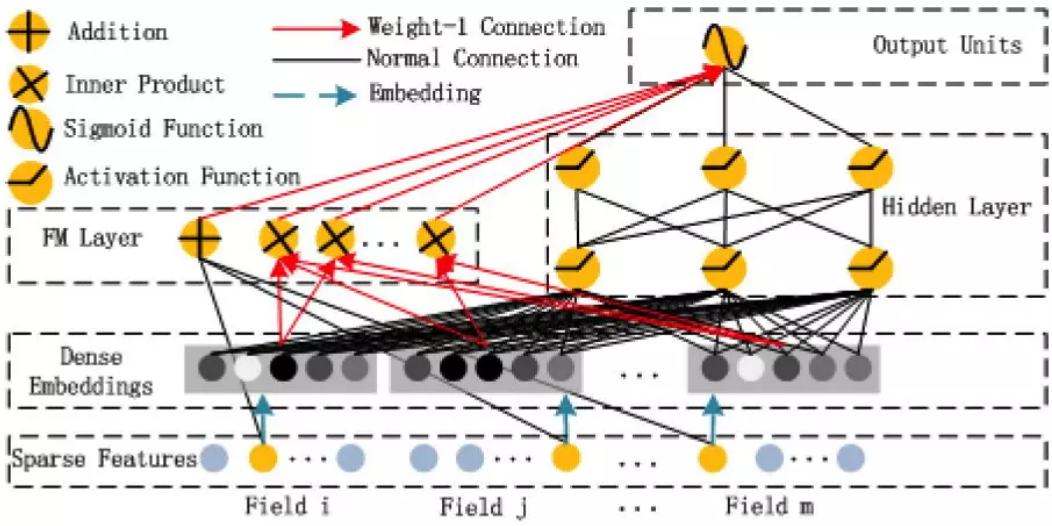

图1：DeepFM的网络结构(图片来源于参考文献4)

该算法从提出后，被工业界大量用于广告点击预估和推荐系统(如美团将DeepFM用于CTR预估)，有非常不错的效果。该团队在参考文献5中进一步对DeepFM两种变种进行了比较，并在华为的应用市场APP推荐真实业务场景中做AB测试，发现比原来的LR算法有近10%的点击率提升。

腾讯开源的Angel(Spark on Angel)中有DeepFM的实现，读者可以尝试将DeepFM应用到自己的推荐或者CTR预估业务中。

六、近实时分解机

Google在2013年提出了**FTRL**(**F**ollow-**T**he-**R**egularized-**L**eader)用于广告点击率预估，该方法可以高效地在线训练LR模型，在Google及国内公司得到大规模工业应用，广泛用于广告点击率预估和推荐排序。想了解的读者可以阅读参考文献25，Tensorflow上有FTRL算法的具体实现。

基于FTRL的思路，可以对FM离线训练算法进行改造，让FM具备在线学习的能力，参考文献3、21、26、27中有关于利用FTRL技术对FM进行在线训练的介绍，这几篇文章涉及到很多数学理论的证明和推导，感兴趣的读者可以自行学习。

上述论文读不懂也没关系，也不要读者自己重复造轮子了，很多机器学习平台上是有FM算法的FTRL实现的，腾讯开源的Angel平台上可以直接利用Spark On Angel(构建在Spark平台上的，利用Angel参数服务器的技术，让Spark具备提供参数服务能力)中的FTRL-FM算法，训练上百亿维特征的在线FM模型。

七、分解机在推荐上的应用

分解机可以作为一般的预测模型，用于回归和分类，特别是用在推荐系统和广告点击率预估等商业场景。在本节我们简单讲讲分解机在推荐系统上的应用。

分解机在推荐系统上的应用，可以采用回归和预测两类方法。当我们预测用户对标的物的评分时，就是回归问题，当我们预测用户对标的物是否点击时，可以看成是一个二分类问题，这时可以通过增加一个logit变换，转化为预测用户点击的概率问题。

下面我们讲讲怎么构建FM需要的特征，有哪些信息可以作为模型的输入。构建FM模型的特征主要分为如下4大类，我们来分别介绍。

7.1 分解机在推荐上的应用

包括用户的点击、播放、收藏、搜索、点赞等各种(隐式)行为。这些行为可以通过平展化的方式整合为特征。比如有n个用户，m个标的物。那么用户对某个标的物的行为平展化为n+m维的特征子向量，其中用户和标的物所在的列非零，其他为零，如下图。

$$(u,i) \rightarrow \mathbf{x} = (\underbrace{0,\ldots,0,1,0,\ldots,0}_{|U|},\underbrace{0,\ldots,0,1,0,\ldots,0}_{|I|}),$$

上图是用户行为的隐式操作转化为特征子向量，特征用0或者1表示，有隐式操作则为1，否则为0。

也可以采用将用户的每一种操作作为一个维度，每个维度的值代表用户是否操作过或者对应的操作得分(比如用户播放时长占视频总时长的比例作为得分)，如下图。

$$(u,i) \rightarrow (\underbrace{0}_{click}, \underbrace{1}_{search}, \underbrace{0}_{collect}, ..., \underbrace{1}_{thumps-up})$$

7.2 用户相关信息

用户相关的信息有很多，包括人口统计学信息，如年龄、性别、职业、收入、地域、受教育程度等等。另外还可以包括用户行为信息，用户是否是会员、什么时候注册过、最后一次登录时间、最后一次付费时间等。这些信息都可以作为某一个维度的特征灌入到FM模型中。

7.3 标的物相关信息

标的物的metadata信息都可以作为FM的特征，拿电影推荐来说，电影的评分、年代、标签、演职员、是否获奖、是否高清、地区、语言、是否是付费节目等都可以作为特征。其中评分、年代是数值特征，标签、演职员是类别特征，可以采用n-hot编码(每个电影有多个标签，对应标签上的值为1，所以这里叫做n-hot编码，而不是one-hot)。另外该节目的用户行为也可以作为特征，比如节目播放次数、节目平均播放完成度等。

7.4 上下文信息

用户在操作标的物时，是包含上下文信息的，这类上下文主要有时间、地点、上一步操作、所在路径、甚至是天气、心情等。时间可以是操作时间、是否是节日、是否是工作日、特殊事件(如双十一)、操作系统、版本等。地域对于LBS类应用是非常重要的。对于像购物等具备漏斗行为转化的产品或业务，用户的上一步操作及所在路径对训练模型非常关键。

整合了上述4大类信息，我们可以构建如下的训练集，利用FM模型来训练，求得参数，最终获得训练好的模型，用于线上预测。

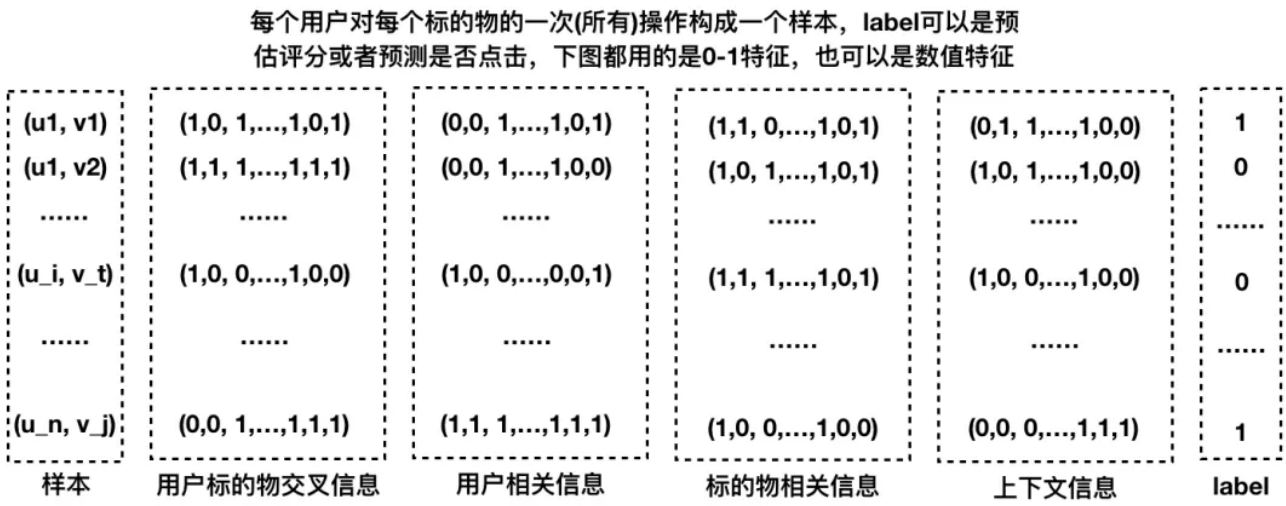

图2：基于4类信息构建分解机的训练数据集

我们可以根据不同的业务形态、当前业务具备的数据等来按照上面的方式获得模型特征，最终构建分解机模型。参考文献10、11、13、18中有怎么利用FM来进行推荐的介绍，另外参考文献23是一篇很长的博士毕业论文，综述利用分解模型(主要是FM)做推荐的各种方法及细节介绍，值得大家学习参考。

七、分解机的优势

在前面章节介绍FM时，我们零星提到了FM的各种特点。在本节，我们梳理一下FM的优势，让大家对FM的特性有一个更具体直观的了解。FM之所以在学术界和工业界受到追捧，得益于它各种优点，这些优点主要体现在如下几个方面。

(1) 可以整合交叉特征，效果不错

在真实业务场景中，往往特征之间的交叉对模型预测是非常有帮助的，而分解机可以自动整合二阶(高阶)交叉特征，免去了人工特征工程(如logistic回归需要大量的人工特征工程)的工作，从而(相比矩阵分解及logistic回归等模型)可以达到更好的训练效果。

(2) 线性时间复杂度

FM通过将预测函数做数学变换，将二阶交叉特征的计算从二阶多项式的复杂度降低到线性复杂度，方便模型预测和通过SGD等迭代方法估计参数， 从而让FM在工业界的大规模数据场景下的应用(推荐系统、CTR预估等)变得可行。

(3) 可以应对稀疏数据情况

FM通过分解二阶交叉特征的系数到低维空间，避免了交叉特征的系数独立的情况，减少了参数空间，并且由于不同交叉项之间的系数是有关联的，在高度稀疏的情况下，也可以容易估计模型系数，模型泛化能力强。

(4) 模型相对简单，易于工程实现

FM模型原理非常简单，思想也很朴素，并且预测过程可以降低到线性时间复杂度，可以采用SGD等常用算法来进行训练，在工程实现上是相对容易的，有很多开源的工具都有FM的实现，我们可以直接拿来用。正因为工程实现简单，才在工业界得到大规模的推广和应用。

总结

本文对分解机的算法原理、参数估计、跟其他模型之间的关系、工程实现、分解机的拓展、近实时分解机、分解机在推荐上的应用、分解机优点等各个方面进行了综合介绍。分解机类似SVM，是一个通用的预测器，适用于任何实值特征向量的预测问题，不仅仅应用于推荐算法，在广告点击率预估等其他方面都有很大的商业应用价值。鉴于FM模型的巨大优势和商业价值，自从FM被提出后，基于FM模型在学术界的研究和工业的实践从未止步过，FM模型值得每一位做算法的从业者研究、学习、实践。

参考文献

1. Factorization Machines with libFM
2. factorization machines
3. Factorization Machines with Follow-The-Regularized-Leader for CTR prediction in Display Advertising
4. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction
5. DeepFM: An End-to-End Wide & Deep Learning Framework for CTR Prediction
6. https://github.com/srendle/libfm
7. http://www.libfm.org/
8. *Scaling Factorization Machines to Relational Data*
9. *Bayesian Factorization Machines*

10. *Learning Recommender Systems with Adaptive Regularization*
11. *Fast Context-aware Recommendations with Factorization Machines*
12. fastFM: A Library for Factorization Machines
13. Optimizing Factorization Machines for Top-N Context-Aware Recommendations
14. DiFacto — Distributed Factorization Machines
15. Attentional Factorization Machines- Learning the Weight of Feature Interactions via Attention Networks
16. Higher-Order Factorization Machines
17. F2M- Scalable Field-Aware Factorization Machines
18. Discrete Factorization Machines for Fast Feature-based Recommendation
19. Field-weighted Factorization Machines for Click-Through Rate Prediction in Display Advertising
20. Neural Factorization Machines for Sparse Predictive Analytics
21. Sketched Follow-The-Regularized-Leader for Online Factorization Machine
22. Field-aware Factorization Machines for CTR Prediction
23. Advanced Factorization Models for Recommender Systems
24. https://github.com/Angel-ML/angel
25. Ad Click Prediction- a View from the Trenches
26. Compact convexified factorization machine: formulation and online algorithms
27. Online Compact Convexified Factorization Machine
28. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems

-end-

相关内容阅读

1.矩阵分解推荐算法
2.协同过滤推荐算法
3.基于标签的实时短视频推荐系统
4.基于Erlang语言的视频相似推荐系统
5.基于内容的推荐算法
6.构建可解释的推荐系统
7.推荐系统产品与算法概述