【模型融合】一网打尽协同过滤、矩阵分解和线性模型

在上一篇文章中，我讲到了使用逻辑回归和梯度提升决策树组合的模型融合办法，用于CTR预估，我还满怀爱意地给这对组合起了个名字，叫做辑度组合，因为这对组合的确可以在很多地方帮到我们。

这对组合中，梯度提升决策树，也就是人们常说的GBDT，所起的作用就是对原始的特征做各种有效的组合，一棵树一个叶子节点就是一种特征组合。

这大概就是逻辑回归的宿命吧，作为一个广义线性模型，在这个由非线性组成的世界里，唯有与各种特征组合办法精诚合作，才能活下去。

从特征组合说起

对逻辑回归最朴素的特征组合就是二阶笛卡尔乘积，但是你有没有想过这样暴力组合的问题所在。

1. 两两组合导致特征维度灾难；
2. 组合后的特征不见得都有效，事实上大部分可能无效；
3. 组合后的特征样本非常稀疏，意思就是组合容易，但是并不能在样本中找到对应的组合出现，也就没办法在训练时更新参数。

如果把包含了特征两两组合的逻辑回归线性部分写出来，就是：

$$\hat{y} = \omega_{0} + \sum_{i=1}^{n}{\omega_{i}x_{i}} + \sum_{i=1}^{n}{\sum_{j=i+1}^{n}{\omega_{ij}x_{i}x_{j}}}$$

这和原始的逻辑回归相比，就多出来了后面那一大坨，特征两两组合那部分，也需要去学习对应的参数权重。

问题就是两两组合后非常有可能没有样本能够学习到 $w_{ij}，不但没有样本可以用来学习到参数，而且在应用时，如果遇到了这样的组合，也就只能放弃，因为没有学到权重。

针对这个问题，就有了一个新的算法模型：因子分解机模型，也叫做FM，即Factorization Machine。因子分解机也常常用来

做模型融合，今天就和你聊聊因子分解机的来龙去脉。

FM模型

1.原理

因子分解机模型是在2010年被提出来的。因为逻辑回归在做特征组合时样本稀疏，从而无法学到很多特征组合的权重，所以因子分解机的提出者就想，能不能对上面那个公式中的$w _{ij}$做解耦，让每一个特征学习一个隐因子向量出来。

就好像前面讲矩阵分解时，为每一个用户和每一个物品各自都学习一个隐因子向量一样，这样一来，任何两个特征不小心在实际使用时相遇了，需要组合，那么各自掏出自己随身携带的隐因子变量做一个向量点积，就是两者组合特征的权重了。

还是针对逻辑回归的线性部分，用公式写一下更清楚：

$$\hat{y} = \omega_{0} + \sum_{i=1}^{n}{\omega_{i}x_{i}} + \sum_{i=1}^{n}{\sum_{j=i+1}^{n}{<V_{i}, V_{j}>x_{i}x_{j}}}$$

这个公式和前面特征组合的公式相比，不同之处就是原来有个 w_{ij}，变成了这里的两个隐因子向量的点积 $<V_{i}, V_{j}>$。

不要小看这个变化。它其实认为两个特征之间，即使没有共同出现在一条样本中，也是有间接联系的。比如说特征A和特征B曾在一些样本中一起出现过，特征B和特征C曾在一些样本中出现过，那么特征A和特征C无论是否在样本中一起出现过，仍然是有些联系的。

如果在实际预测CTR时，特征A和特征C真的在一起出现了，如果你用的是因子分解机模型，这时候你的预测程序就不慌不忙走向数据库，从中取出早已准备好的特征A和特征C的隐因子向量，拿出来做一个点积运算，就得到了两者组合的权重。

也许逻辑回归见到这一切不禁要问：居然还有这种操作？是的，因子分解机的先进之处就在于此。

现在聪明如你，一定也想到了，既然二阶特征组合可以学到隐因子向量，那么三阶特征组合也可以加进来，四阶，五阶……想一想是不是有点小激动？不要急，组合越多，计算复杂度就会陡增，所以一般在实际使用中，因子分解机就表演到二阶特征组合就OK了。

2模型训练

因子分解机的参数学习并无特别之处，看目标函数，我在这里是把它当作融合模型来看的，用来做CTR预估，因此预测目标是一个二分类，因子分解机的输出还需要经过sigmoid函数变换：

$$\sigma(\hat{y}) = \frac{1}{1+e^{-\hat{y}}}$$

因此，损失目标函数也就是常用的logistic loss：

$$ loss(\theta)= -\frac{1}{m}\sum_{i=1}^{m}{[y^{(i)}log(\sigma(\hat{y}))+(1-y^{(i)})log(1-\sigma(\hat{y}))]} $$

公式中 $\sigma(\hat{y})$ 是因子分解机的预测输出后经过sigmoid函数变换得到的预估CTR， $y^{(i)}$ 是真实样本的类别标记，正样本是1，负样本是0，m是样本总数。

对这个损失目标函数使用梯度下降或者随机梯度下降就可以得到模型的参数，和前面文章里的没有区别，注意损失函数实际上还需要加上正则项，这在上一篇专栏中已经总结过机器学习损失函数的两板斧，就是偏差和方差。

3预测阶段

假如现在已经得到了因子分解机的模型参数，你忍不住跃跃欲试想端着它冲上战场，且慢，因子分解机中二阶特征组合那一坨，在实际计算时，复杂度有点高，如果隐因子向量的维度是k，特征维度是n，那这个复杂度就是$O(kn^2)$，其中n方是特

征要两两组合，k是每次组合都要对k维向量计算点积。需要对此稍微做一下改造，改造过程如下：

$$
\begin{aligned}
\sum_{i=1}^{n}\sum_{j=i+1}^{n}{<V_{i}, V_{j}>x_{i}x_{j}}={}& \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}{<V_{i}, V_{j}>x_{i}x_{j}} - \frac{1}{2}\sum_{i=1}^{n}{<V_{i}, V_{j}>x_{i}x_{i}} \\
&=\frac{1}{2}(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{f=1}^{k}{v_{i,f}v_{j,f}x_ix_j}-\sum_{i=1}^{n}\sum_{f=1}^{k}{v_{i,f}v_{i,f}x_ix_i}) \\
&=\frac{1}{2}\sum_{f=1}^{k}((\sum_{i=1}^{n}{v_{i,f}x_i})(\sum_{j=1}^{n}v_{j,f}x_j)-\sum_{i=1}^{n}{v_{i,f}^2x_i^2}) \\
&=\frac{1}{2}\sum_{f=1}^{k}((\sum_{i=1}^{n}{v_{i,f}x_i})^2-\sum_{i=1}^{n}{v_{i,f}^2x_i^2})}
\end{aligned}
$$

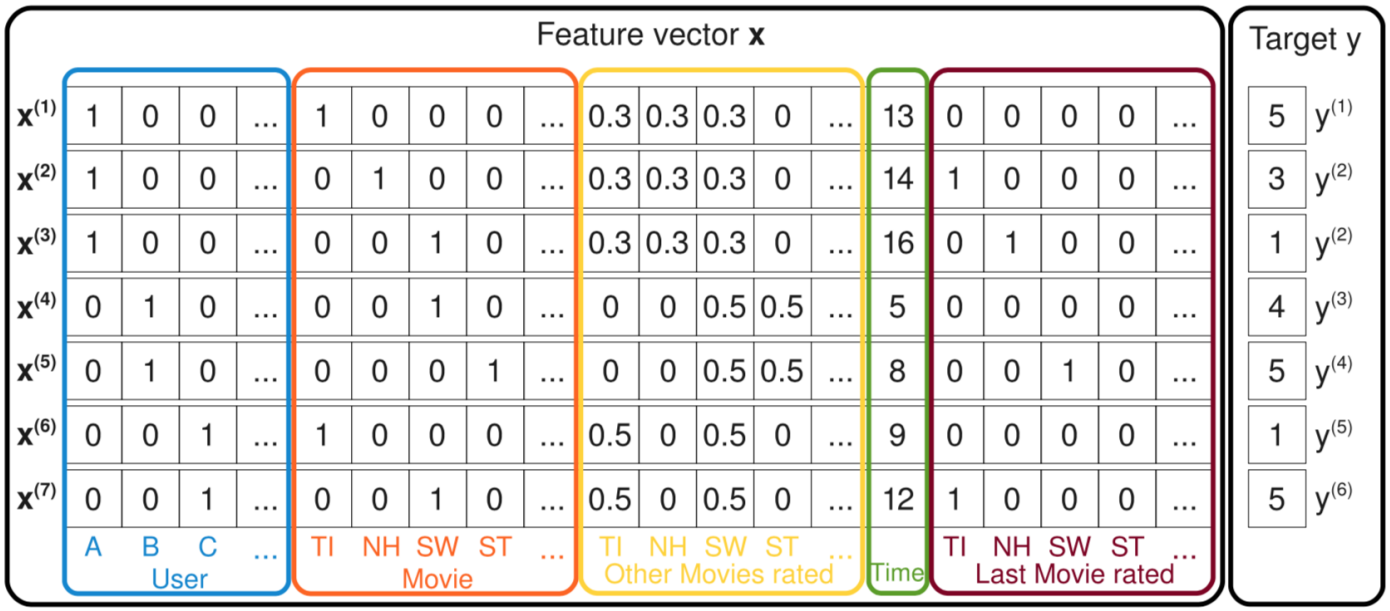
看上去这个有点复杂，你如果不想理解也没关系，我直接告诉你最后该怎么算。

```
loop1 begin: 循环k次，k就是隐因子向量的维度，其中，循环到第f次时做以下事情

 loop2 begin:循环n个特征，第i次循环时做这样的事情

  1. 从第i个特征的隐因子向量中拿出第f维的值
  2. 计算两个值：A是特征值和f维的值相乘，B是A的平方

 loop2 end

 把n个A累加起来，并平方得到C，把n个B也累加起来，得到D

 用C减D，得到E

loop1 end

把k次循环得到的k个E累加起来，除以2
```

这就是因子分解机中，二阶组合部分的实际计算方法，现在这样做的复杂度只是O(kn)，原来的平方复杂度不见了。

4.一网打尽其他模型

下面继续带你见识一些因子分解机的神奇之处。看下面这张图，这里示意了一批样本。



这张图中每一条样本都记录了用户对电影的评分，最右边的y是评分，也就是预测目标；左边的特征有五种，用户ID、当前评分的电影ID、曾经评过的其他分、评分时间、上一次评分的电影。

好，现在我们来看因子分解机如何一网打尽其他模型的，注意，这里说的一网打尽并不是打败，而是说模型可以变形成其他模型。

前面已经说了因子分解机可以实现带有特征组合的逻辑回归。

现在假设图中的样本特征只留下用户ID和电影ID，因子分解机模型就变成：

$$\hat{y} = \omega_{0} + \omega_{u} + \omega_{i} + <V_{u}, V_{i}>$$

解释一下如何变成这样的。因为用户ID和电影ID，在一条样本中，各自都只有一个维度是1，其他都是0，所以在一阶部分就没有了求和符合，直接是wu和wi，二阶部分特征乘积也只剩下了一个1，其他都为0了。你瞧，这不就是带有偏置信息的SVD吗?

现在继续，在SVD基础上把样本中的特征加上用户历史评过分的电影ID，再求隐因子向量，这就是SVD++呀！

再加上时间信息，就变成了time-SVD。

所以因子分解机是把我之前讲过的矩阵分解一网打尽了，顺便还干起了逻辑回归的工作，也正因如此，因子分解机常常用来做模型融合，在推荐系统的排序阶段肩负起对召回结果做重排序的任务。

5.FFM

因子分解机的故事已经讲完，但我后来在因子分解机基础上改进了一下。改进的思路是：不但认为特征和特征之间潜藏着一些不可告人的关系，还认为特征和特征类型有着千丝万缕的关系。

这个特征类型，就是某些特征实际上是来自数据的同一个字段，比如用户ID，占据了很多维度，变成了很多特征，但他们都属于同一个类型，都叫“用户ID”。这个特征类型就是字段，即Field。这种改进叫做Field-aware Factorization Machines，简称FFM。

再回顾一下，因子分解机模型的样子是这样：

$$\hat{y} = \omega_{0} + \sum_{i=1}^{n}{\omega_{i}x_{i}} + \sum_{i=1}^{n}{\sum_{j=i+1}^{n}{<V_{i}, V_{j}>x_{i}x_{j}}}$$

之前因子分解机认为每个特征有一个隐因子向量，FFM改进的是二阶组合那部分，改进的模型认为每个特征有f个隐因子向量，这里的f就是特征一共来自多少个字段（Field），二阶组合部分改进后如下：

$$ \sum_{i=1}^{n}{\sum_{j=i+1}^{n}{<V_{i,fj}, V_{j,fi}>x_{i}x_{j}}} $$

FFM模型也常用来做CTR预估。在FM和FFM事件过程中，记得要对样本和特征都做归一化。

总结

今天，我给你介绍了另一种常用来做CTR预估的模型，因子分解机。因子分解机最早提出在2010年，在一些数据挖掘比赛中都取得了很好的乘积，后来被引入工业界做模型融合，也表现不俗。严格来说，因子分解机也算是矩阵分解算法的一种，因为它的学习结果也是隐因子向量，也是用过隐因子向量的点积代替原来的单个权重参数。

最后，由于不断提到特征组合的重要性，前有GBDT，现有FM，都是在特征组合上花功夫，你能不能在这里分享一下，你所用过的特征组合办法有哪些呢？欢迎留言一起讨论。

精选留言

qi
感觉越来越不理解了，只怪自己太浅了，学识不够！
2018-04-08 07:59

上个纪元的赵天师
跪求老师出版实体书，感觉太有收获了
2018-04-04 17:43

作者回复

会有的。
2018-04-04 19:28

mervynlh
老师，现在项目中用的gbdt还是fm,两者比较呢
2018-04-04 07:56

帅帅
目前看起来，模型从简单到强大，一次是LR、GBDT+LR、GBDT+FM、DNN；
那是不是直接上DNN最好呢?

我的理解并不是，如果数据量很小使用DNN会容易过拟合；
因此，简单的就选GBDT+LR、复杂的就选DNN；
2018-09-25 20:54

愚公移山
老师，使用了两两特征组合后，逻辑回归从线性模型变成了非线性模型，因此模型表现的更好，可以这样理解吗?
2018-04-05 17:56

您的好友William
DNN虽然可以自动做一些feature engineering的工作，但是对于大型系统来讲，还是规定一些feature，将这一部分单独拿出来做之后共享给其他组，之后各个组的工作才能对接，对接之后fine-tune的可解释性也强，如果大家都用DNN，那么就是一个黑盒子加一个黑盒子，有可能输入输出还不一样，到时候融合对接都成问题。所以DNN作为一个超级function approximator在工业界还是应该比较适用于小型独立的项目，项目组之前各个组之间feature的统一提取，或者是之后作为项目最后的决策层。
2018-10-01 15:16

林彦
感觉现在周围一般的机器学习实践GBDT用的更多一点。没和实践过推荐系统的人直接交流过，不知道因子分解机除了预测点击率外，对什么场景效果优于其他的特征组合方法。现在陈老师的理论讲得通俗易懂，不过自己编程和工程实践训练不够，实践还不知道如何入手。用哪套数据，哪套来源工具包，阅读哪套源码来学习实践还没有认知。
2018-04-05 22:49

作者回复
如果找不到实践机会，就去kaggle刷比赛吧。如果你想实习，也可以给我发简历：chenkaijiang001@lianjia.com
2018-04-07 10:56

Classtag
后边会说到deepfm fnn 这些模型吗?
2018-04-04 23:26

作者回复
会说到相似的模型。
2018-04-10 10:07