

# 推荐系统遇上深度学习——FM模型理论和实践

文文 书圈 2019-01-07



本文转自：小小挖掘机(已获授权)

## 1、FM背景

在计算广告和推荐系统中，CTR预估(click-through rate)是非常重要的一个环节，判断一个商品的是否进行推荐需要根据CTR预估的点击率来进行。在进行CTR预估时，除了单特征外，往往要对特征进行组合。对于特征组合来说，业界现在通用的做法主要有两大类：FM系列与Tree系列。今天，我们就来讲讲FM算法。

## 2、one-hot编码带来的问题

FM(Factorization Machine)主要是为了解决数据稀疏的情况下，特征怎样组合的问题。已一个广告分类的问题为例，根据用户与广告位的一些特征，来预测用户是否会点击广告。数据如下：(本例来自美团技术团队分享的paper)

Clicked?	Country	Day	Ad_type
1	USA	26/11/15	Movie
0	China	1/7/14	Game
1	China	19/2/15	Game

clicked 是分类值，表明用户有没有点击该广告。1 表示点击，0 表示未点击。而 country,day,ad\_type则是对应的特征。对于这种categorical特征，一般都是进行one-hot编码处理。

将上面的数据进行one-hot编码以后，就变成了下面这样：

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

因为是categorical特征，所以经过one-hot编码以后，不可避免的样本的数据就变得很稀疏。举个非常简单的例子，假设淘宝或者京东上的item为100万，如果对item这个维度进行one-hot编码，光这一个维度数据的稀疏度就是百万分之一。由此可见，数据的稀疏性，是我们在实际应用场景中面临的一个非常常见的挑战与问题。

one-hot编码带来的另一个问题是特征空间变大。同样以上面淘宝上的item为例，将item进行one-hot编码以后，样本空间有一个categorical变为了百万维的数值特征，特征空间一下子暴增一百万。所以大厂动不动上亿维度，就是这么来的。

### 3、对特征进行组合

普通的线性模型，我们都是将各个特征独立考虑的，并没有考虑到特征与特征之间的相互关系。但实际上，大量的特征之间是有关联的。最简单的以电商为例，一般女性用户看化妆品服装之类的广告比较多，而男性更青睐各种球类装备。那很明显，女性这个特征与化妆品类服装类商品有很大的关联性，男性这个特征与球类装备的关联性更为密切。如果我们能将这些有关联的特征找出来，显然是很有意义的。

一般的线性模型为：

$$y = \omega_0 + \sum_{i=1}^n \omega_i x_i$$

从上面的式子很容易看出，一般的线性模型压根没有考虑特征间的关联。为了表述特征间的相关性，我们采用多项式模型。在多项式模型中，特征 $x_i$ 与 $x_j$ 的组合用 $x_i x_j$ 表示。为了简单起见，我们讨论二阶多项式模型。具体的模型表达式如下：

$$y = \omega_0 + \sum_{i=1}^n \omega_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^n \omega_{ij} x_i x_j$$

上式中， $n$ 表示样本的特征数量， $x_i$ 表示第 $i$ 个特征。

与线性模型相比，FM的模型就多了后面特征组合的部分。

### 4、FM求解

从上面的式子可以很容易看出，组合部分的特征相关参数共有 $n(n-1)/2$ 个。但是如第二部分所分析，在数据很稀疏的情况下，满足 $x_i, x_j$ 都不为0的情况非常少，这样将导致 $\omega_{ij}$ 无法通过训练得出。

为了求出 $\omega_{ij}$ ，我们对每一个特征分量 $x_i$ 引入辅助向量 $\mathbf{V}_i = (v_{i1}, v_{i2}, \dots, v_{ik})$ 。然后，利用 $\mathbf{V}_i \mathbf{V}_j^T$ 对 $\omega_{ij}$ 进行求解。

$$\mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1k} \\ v_{21} & v_{22} & \cdots & v_{2k} \\ \vdots & \vdots & & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nk} \end{pmatrix}_{n \times k} = \begin{pmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_n \end{pmatrix}$$

那么 $\omega_{ij}$ 组成的矩阵可以表示为：

$$\hat{\mathbf{W}} = \mathbf{V} \mathbf{V}^T = \begin{pmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_n \end{pmatrix} (\mathbf{V}_1^T \quad \mathbf{V}_2^T \quad \cdots \quad \mathbf{V}_n^T)$$

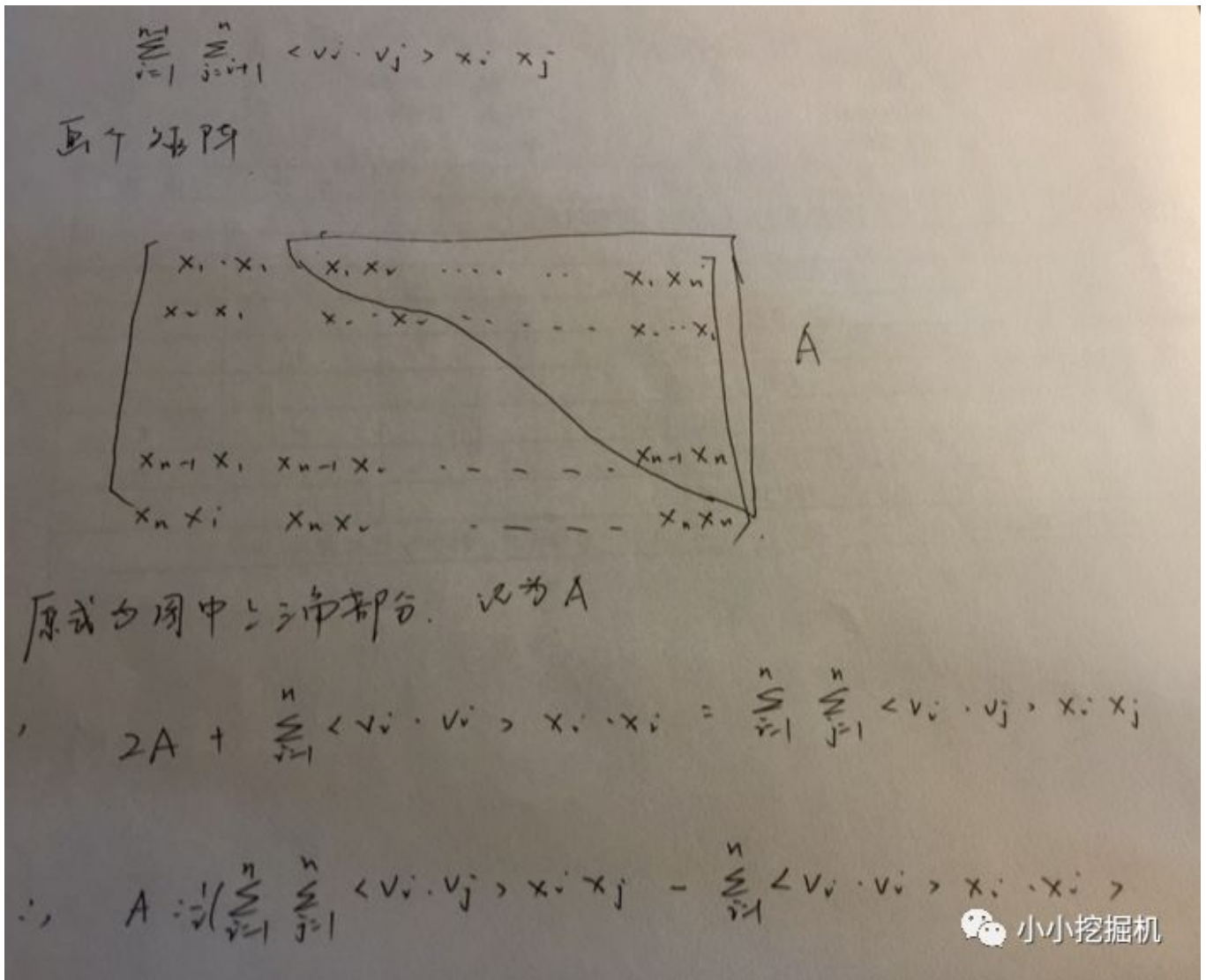
那么，如何求解 $v_i$ 和 $v_j$ 呢？主要采用了公式：

$$((a+b+c)^2 - a^2 - b^2 - c^2) / 2$$

具体过程如下：

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

上面的式子中有同学曾经问我第一步是怎么推导的，其实也不难，看下面的手写过程(大伙可不要嫌弃字丑哟)



经过这样的分解之后，我们就可以通过随机梯度下降SGD进行求解：

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

## 5、tensorflow代码详解

代 码 参 考 地 址 :

[https://github.com/babakx/fm\\_tensorflow/blob/master/fm\\_tensorflow.ipynb](https://github.com/babakx/fm_tensorflow/blob/master/fm_tensorflow.ipynb)

上面的代码使用的是python2编码，在python3下运行会出错，所以如果大家使用的是python3的话，可以参考我写的，其实就是修复了几个bug啦，哈哈。

我的github地址：

[https://github.com/princewen/tensorflow\\_practice/tree/master/recommendation-FM-demo](https://github.com/princewen/tensorflow_practice/tree/master/recommendation-FM-demo)。





```
csr_matrix((data, indices, indptr)
```

可以看到，函数接收三个参数，第一个参数是数值，第二个参数是每个数对应的列号，第三个参数是每行的起始的偏移量，举上图的例子来说，第0行的起始偏移是0，第0行有2个非0值，因此第一行的起始偏移是2，第1行有两个非0值，因此第二行的起始偏移是4，依次类推。

下面的代码是如何将原始的文件输入转换成我们的矩阵：

```
def vectorize_dic(dic, ix=None, p=None, n=0, g=0):
    """
    dic -- dictionary of feature lists. Keys are the name of features
    ix -- index generator (default None)
    p -- dimension of featrure space (number of columns in the sparse
    """
    if ix==None:
        ix = dict()

    nz = n * g

    col_ix = np.empty(nz, dtype = int)

    i = 0
    for k, lis in dic.items():
        for t in range(len(lis)):
            ix[str(lis[t]) + str(k)] = ix.get(str(lis[t]) + str(k), 0)
            col_ix[i+t*g] = ix[str(lis[t]) + str(k)]
        i += 1

    row_ix = np.repeat(np.arange(0, n), g)
    data = np.ones(nz)
    if p == None:
        p = len(ix)

    ixx = np.where(col_ix < p)
    return csr.csr_matrix((data[ixx], (row_ix[ixx], col_ix[ixx])), shape=(n, p))

cols = ['user', 'item', 'rating', 'timestamp']
```


```
train = pd.read_csv('data/ua.base', delimiter='\t', names = cols)
test = pd.read_csv('data/ua.test', delimiter='\t', names = cols)

x_train, ix = vectorize_dic({'users': train['user'].values,
                             'items': train['item'].values}, ix, 1)

x_test, ix = vectorize_dic({'users': test['user'].values,
                             'items': test['item'].values}, ix, 1)

y_train = train['rating'].values
y_test = test['rating'].values

x_train = x_train.todense()
x_test = x_test.todense()
```



如果不做处理，函数返回的矩阵是按如下的格式保存的：

```
(0, 1) 2.0
(1, 1) 1.0
(1, 2) 1.0
(2, 1) 1.0
(2, 3) 1.0
(3, 1) 1.0
(3, 4) 1.0
(4, 1) 1.0
(4, 5) 1.0
(5, 1) 1.0
(5, 6) 1.0
(6, 1) 1.0
(6, 7) 1.0
(7, 1) 1.0
(7, 8) 1.0
(8, 1) 1.0
(8, 9) 1.0
(9, 1) 1.0
```

使用todense变换后，变成如下样式：

```
[[ 0.  2.  0. ...,  0.  0.  0.]
 [ 0.  1.  1. ...,  0.  0.  0.]
 [ 0.  1.  0. ...,  0.  0.  0.]
 ...,
 [ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
```

## 估计值计算

得到我们的输入之后，我们使用tensorflow来设计我们的模型，其实很简单啦，我们模型的估计值由两部分构成，原始的可以理解为线性回归的部分，以及交叉特征的部分，交叉特征直接使用我们最后推导的形式即可，再回顾一遍：



$$= \frac{1}{2} \sum_{f=1}^k \left( \left( \sum_{i=1}^n v_{i,f} x_i \right)^2 - \sum_{i=1}^n v_{i,f}^2 x_i^2 \right)$$

因此，我们需要定义三个placeholder，分别是输入的x，输入的y，以及我们的 用户数\*电影数大小的待学习的fm矩阵：

```
n, p = x_train.shape
```

```
k = 10
```

```
x = tf.placeholder('float', [None, p])
```

```
y = tf.placeholder('float', [None, 1])
```

```
w0 = tf.Variable(tf.zeros([1]))
```

```
w = tf.Variable(tf.zeros([p]))
```

```
v = tf.Variable(tf.random_normal([k, p], mean=0, stddev=0.01))
```

```
#y_hat = tf.Variable(tf.zeros([n, 1]))
```

```
linear_terms = tf.add(w0, tf.reduce_sum(tf.multiply(w, x), 1, keep_dims=True
```

```
pair_interactions = 0.5 * tf.reduce_sum(
    tf.subtract(
        tf.pow(
            tf.matmul(x, tf.transpose(v)), 2),
        tf.matmul(tf.pow(x, 2), tf.transpose(tf.pow(v, 2)))
    ), axis = 1, keep_dims=True)
```

```
y_hat = tf.add(linear_terms, pair_interactions)
```

## 定义损失函数

这里我们定义的损失函数除了平方损失外，还加了l2正则项，并使用梯度下降法进行参数的更新：

```
lambda_w = tf.constant(0.001, name='lambda_w')
lambda_v = tf.constant(0.001, name='lambda_v')

l2_norm = tf.reduce_sum(
    tf.add(
        tf.multiply(lambda_w, tf.pow(w, 2)),
        tf.multiply(lambda_v, tf.pow(v, 2))
    )
)

error = tf.reduce_mean(tf.square(y-y_hat))
loss = tf.add(error, l2_norm)

train_op = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(loss)
```

## 模型训练

接下来就是训练啦，这段代码比较好理解：

```
epochs = 10
batch_size = 1000

# Launch the graph
init = tf.global_variables_initializer()
with tf.Session() as sess:
    sess.run(init)

    for epoch in tqdm(range(epochs), unit='epoch'):
        perm = np.random.permutation(x_train.shape[0])
        # iterate over batches
        for bX, bY in batcher(x_train[perm], y_train[perm], batch_size):
            _, t = sess.run([train_op, loss], feed_dict={x: bX.reshape(-1, 1), y: bY})
            print(t)
```

```
errors = []
for bX, bY in batcher(x_test, y_test):
    errors.append(sess.run(error, feed_dict={x: bX.reshape(-1,
    print(errors)
RMSE = np.sqrt(np.array(errors).mean()))
print (RMSE)
```

## 参考文章:

- 1、<http://blog.csdn.net/bitcarmanlee/article/details/52143909>
- 2、<https://blog.csdn.net/u012871493/article/details/51593451>

## 有关作者:

石晓文，中国人民大学信息学院在读研究生

简书ID：石晓文的学习日记(<https://www.jianshu.com/u/c5df9e229a67>)

天善社区：<https://www.hellobi.com/u/58654/articles>

腾讯云：<https://cloud.tencent.com/developer/user/1622140>



长按二维码 关注我们

京东首发热卖中