二分网络召回

来自于电子科技大学周涛教授的代表作
Bipartite network projection and personal recommendation

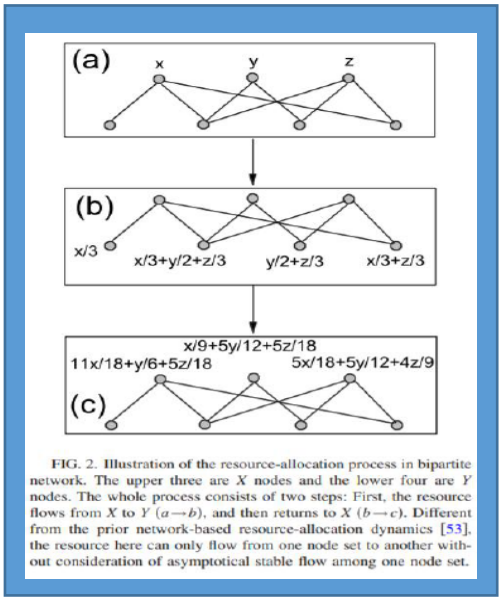


FIG. 2. Illustration of the resource-allocation process in bipartite network. The upper three are X nodes and the lower four are Y nodes. The whole process consists of two steps: First, the resource flows from X to Y $(a \rightarrow b)$, and then returns to X $(b \rightarrow c)$. Different from the prior network-based resource-allocation dynamics [53], the resource here can only flow from one node set to another without consideration of asymptotical stable flow among one node set.

多路召回

多路召回就是采取不同哦那过的策略特征或者简单的模型，分别召回一部分的候选集，然后把这些候选的集混合在一起后供排序模型的使用的策略.
考虑到召回率高的化，那么召回的速度肯定会相应的降低，所以一般采用多种简单的召回策略叠加的形式就是多路召回.
每一个召回策略之间好不相干，比如新闻推荐系统中，按照文章的类别，作者和热度等分别进行召回，召回更加贴切实际的要求.

推荐系统评价指标

TOP-N

1. HIT Rate（HR）
HR 是目前 TOP-N 推荐研究中十分流行的评价指标，其中@users 是用户总数，而@hits 是测试集中的 item 出现在 Top-N 推荐列表中的用户数量。

$$HR = \frac{@hits}{@users}$$

2. Average Reciprocal Hit Rank (ARHR)

ARHR 也是目前 Top-N 推荐中十分流行的指标，它是一种加权版本的 HR，它衡量一个 item 被推荐的强度

$$ARHR = \frac{1}{@users} \sum_{i=1}^{@hits} \frac{1}{p_i}$$

其中权重 pi 是推荐列表位置的倒数

3.quality of the list of recommendations

当推荐项目的数量很大时，用户会更加重视推荐列表中排在前面的 item。这些 item 中发生的错误比列表中排在后面的 item 中的错误更严重。按排名列表对推荐效果进行加权评估的方法考虑了这种情况。在最常用的排名度量指标中，有以下标准信息检索度量：

（a）半衰期（half-life），假设当用户远离顶部的推荐时，他们的兴趣指数下降；
（b）贴现累积增益（discounted cumulative gain），其中衰减函数是对数函数。
（c）排序偏差准确率(rank-biased precision，RBP)，以等比数列衰减

MRP&MAP

1.Mean Reciprocal Rank (MRR)

MRR 是把正确的 item 在推荐列表中的排序取倒数作为它的准确度，再对所有的问题取平均。相对简单

Query	Results	Correct response	Rank	Reciprocal rank
新闻3	新闻1，新闻2，新闻3	新闻3	3	1/3
新闻2	新闻1，新闻2，新闻3	新闻2	2	1/2
新闻1	新闻1，新闻2，新闻3	新闻1	1	1

MRP = (1/3+1/2+1)/3=11/18=0.61

2.Mean Average Precision (MAP)

平均准确率 MAP，假使当我们搜索某个关键词，返回了 10 个结果。当然最好的情况是这 10 个结果都是我们想要的相关信息。但是假如只有部分是相关的，比如 5 个，那么这 5 个结果如果被显示的比较靠前也是一个相对不错的结果。但是如果这个 5 个相关信息从第 6 个返回结果才开始出现，那么这种情况便是比较差的。这便是 AP 所反映的指标，与 recall 的概念有些类似，不过 AP 是“顺序敏感的 recall

$$AP_u = \frac{1}{\Omega_u} \sum_{i \in \Omega_u} \frac{\sum_{j \in \Omega_u} h(p_{uj} < p_{ui}) + 1}{p_{ui}}$$

其中 Ωu 表示 ground-truth 的结果, pui 表示 i 物品在推荐列表中的位置， puj <pui 表示物品 j 排在 i 前面。

MAP 表示所有用户 u 的 AP 取均值

$$MAP = \frac{\sum_{u \in U} AP_u}{|U|}$$

场景转化指标

- **访客数**（UV）就是指一天之内到底有多少不同的用户访问了你的网站。访客数要比 IP 数更能真实准确地反映用户数量。
- **浏览量**（PV)和访问次数是呼应的。用户访问网站时每打开一个页面，就记为 1 个 PV。同一个人浏览你网站同一个页面，不重复计算 pv 量。
- 潜在用户在我们的网站上完成一次我们期望的行为，就叫做一次**转化**。
- **平均访问时长**是用户访问网站的平均停留时间.

转化率= 转化次数/访问次数。

pv 点击率= pv 点击/pv

uv 点击率= 点击 uv/整个产品的 uv

曝光点击率= 点击量/曝光次数

消费满意度

1.留存率（x 日后仍活跃的用户数/x 日前的用户数）

2.停留时间长（实际播放时间 OR 进度条时长，一般前者）

3.播放完成率（播放时长/视频时长）

FM-FFM-DeepFM

因子分解机-背景 1

1.传统的线性模型如 LR 中，每个特征都是独立的，如果需要考虑特征与特征直接的交互作用，可能需要人工对特征进行交叉组合；

2.矩阵分解 MF、SVD++等，这些模型可以学习到特征之间的交互隐藏关系，但基本上每个模型都只适用于特定的输入和场景

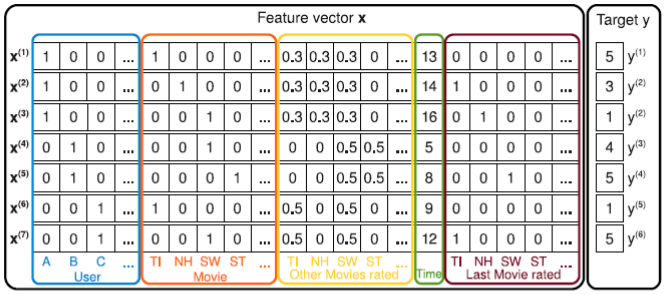


Fig. 1. Example for sparse real valued feature vectors $\mathbf{x}$ that are created from the transactions of example 1. Every row represents a feature vector $\mathbf{x}^{(i)}$ with its corresponding target $y^{(i)}$. The first 4 columns (blue) represent indicator variables for the active user; the next 5 (red) indicator variables for the active item. The next 5 columns (yellow) hold additional implicit indicators (i.e. other movies the user has rated). One feature (green) represents the time in months. The last 5 columns (brown) have indicators for the last movie the user has rated before the active one. The rightmost column is the target – here the rating.

因子分解机-背景 2

1. 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。目前主要应用于 CTR 预估以及推荐系统中的概率计算。下图是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。

Country	Day	Ad_type	Clicked?
USA	3/3/15	Movie	1
China	1/7/14	Game	0
China	3/3/15	Game	1

因子分解机-背景 3

2. 这样是否点击广告由 Country，Day，Ad_type 三列共同决定。由于此三列都为category类别特征，很多模型不能直接用其来预测，所以需要先转化为one-hot特征，即:

Countr y=USA	Country =China	Day= 3/3/15	Day= 1/7/14	Ad_type =Movie	Ad_type= Game	Clicked?
1	0	1	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1

对特征编码后的特征共有 6 维，平均只有 3 维为非零值。这样的意思是特征很样本数据很稀疏，当特征维度有几百维度甚至上万维的时候，特征将更稀疏，很可能几百列只有一列为非零。
但是特征和特征直接的关联有时候对 label 的重要性会提高，如 USA 和 Thanksgiving;China 与 Chinese New Year,所以引入特征和特征的组合作为新特征很有必要。

因子分解机-背景 4
特征中的前四列表示用户 u（one-hot 编码，稀疏），接着五列表示电影 i（ont-hot 编码，稀疏），再接下去五列表示用户 u 对电影 i 的打分（归一化特征），紧接着一列表示时间（连续特征），最后五列表示用户 u 对电影 i 打分前评价过的最近一部电影（one-hot 编码，稀疏）

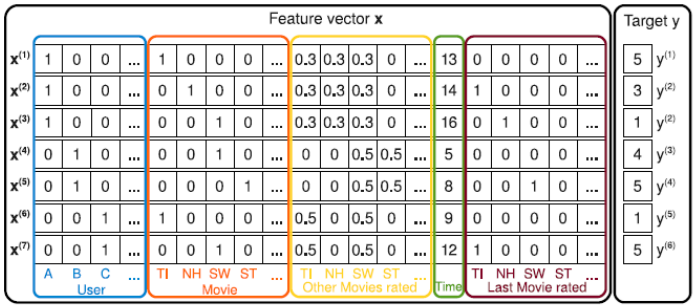


Fig. 1. Example for sparse real valued feature vectors $\mathbf{x}$ that are created from the transactions of example 1. Every row represents a feature vector $\mathbf{x}^{(i)}$ with its corresponding target $y^{(i)}$. The first 4 columns (blue) represent indicator variables for the active user; the next 5 (red) indicator variables for the active item. The next 5 columns (yellow) hold additional implicit indicators (i.e. other movies the user has rated). One feature (green) represents the time in months. The last 5 columns (brown) have indicators for the last movie the user has rated before the active one. The rightmost column is the target – here the rating.

因子分解机
为了表达原有特征的属性意义和特征组合的意义。FM 模型的表达式如下

$$y = w_0 + \sum_{i=1}^{n} w_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \cdot x_i \cdot x_j$$

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。

FM 求解

FM 模型表达式中，wij 表达如果单独为 0 则没有特别意义，所以将其分解成隐变量来表达。W=VVT,

$$W = VV^T = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \begin{pmatrix} v_1^T & v_2^T & \cdots & v_n^T \end{pmatrix} = \begin{pmatrix} v_1 v_1^T, v_1 v_2^T \cdots v_1 v_n^T \\ v_2 v_1^T, v_2 v_2^T \cdots v_2 v_n^T \\ \vdots \\ v_n v_1^T, v_n v_2^T \cdots v_n v_n^T \end{pmatrix} = \begin{pmatrix} v_1 v_1^T, w_{12} \cdots w_{1n} \\ w_{21}, v_2 v_2^T \cdots w_{1n} \\ \vdots \quad \vdots \quad \vdots \\ w_{n1}, w_{n2} \cdots v_n v_n^T \end{pmatrix}$$

这样，FM 模型应该为

$$y = w_0 + \sum_{i=1}^{n} w_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} < v_i, v_j > \cdot x_i \cdot x_j$$

这样，求解交叉项参数<vi,vj>项

$$\sum_{i=1}^{n} \sum_{j=\mathrm{i}+1}^{n} < v_i, v_j > \cdot x_i \cdot x_j$$

$$= \frac{1}{2} (\sum_{i=1}^{n} \sum_{j=1}^{n} < v_i, v_j > \cdot x_i \cdot x_j - \sum_{i=1}^{n} < v_i, v_j > \cdot x_i \cdot x_i)$$

$$= \frac{1}{2} (\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{f=1}^{k} v_{if} v_{jf} \cdot x_i \cdot x_j - \sum_{i=1}^{n} \sum_{f=1}^{k} v_{if} v_{jf} \cdot x_i \cdot x_i)$$

$$= \frac{1}{2} \sum_{f=1}^{k} (\sum_{i=1}^{n} v_{if} x_i \sum_{j=1}^{n} v_{jf} x_j - \sum_{i=1}^{n} v_{if}^2 x_i^2)$$

$$= \frac{1}{2} \sum_{f=1}^{k} [(\sum_{i=1}^{n} v_{if} x_i)^2 - \sum_{i=1}^{n} v_{if}^2 x_i^2]$$

FM 推导

假设有 3 个变量（特征）x1 x2 x3，每一个特征的隐变量分别为 v1=(1 2 3)、v2=(4 5 6)、v3=(1 2 1)（注：每个特征都有一个隐变量）则

$$V=\begin{bmatrix}V_1\\V_2\\V_3\end{bmatrix}=\begin{bmatrix}1&2&3\\4&5&6\\1&2&1\end{bmatrix}\quad V^T=\begin{bmatrix}V_1^T\\V_2^T\\V_3^T\end{bmatrix}=\begin{bmatrix}1&4&1\\2&5&2\\3&6&1\end{bmatrix}$$

$$VV^T=\begin{bmatrix}V_1\\V_2\\V_3\end{bmatrix}\times\begin{bmatrix}V_1^T\\V_2^T\\V_3^T\end{bmatrix}=\begin{bmatrix}1&2&3\\4&5&6\\1&2&1\end{bmatrix}\times\begin{bmatrix}1&4&1\\2&5&2\\3&6&1\end{bmatrix}=\begin{bmatrix}14&32&8\\32&77&20\\8&20&6\end{bmatrix}$$

$$\begin{aligned}\sum_{i=1}^{3}\sum_{i=1}^{3}w_{ij}x_ix_j=&14x_1^2+32x_1x_2+8x_1x_3\\&+32x_1x_2+77x_2^2+20x_2x_3\\&+8x_3x_1+20x_3x_2+6x_3^2\end{aligned}$$

其中，我们只需要获得 x1x2,x2x3,x1x3,x2x1,,x3x2,x3x1 其中 x2x1,,x3x2,x3x1 为重复项，所以只需要保留 x1x2,x2x3,x1x3，且其中 x1x1，x2x2，x3x3 不是交叉项，同时也需要去除

$$\begin{aligned}\sum_{i=1}^{3}\sum_{i=1}^{3}w_{ij}x_ix_j=&14x_1^2+32x_1x_2+8x_1x_3\\&+32x_1x_2+77x_2^2+20x_2x_3\\&+8x_3x_1+20x_3x_2+6x_3^2\end{aligned}$$

FFM

场感知分解机器（Field-aware Factorization Machine ，简称 FFM）

User	Movie	Genre	Price
YuChin	3Idiots	Comedy, Drama	$9.9

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。假设每条样本的 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。因此可以把 FM 看作是 FFM 的特例，即把所有的特征都归属到一个 field 是的 FFM 模型。根据 FFM 的 field 敏感特性，可以导出其模型表达式：

$$y = w_0 + \sum_{i=1}^{n} w_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} < v_{ifi} v_{jfi} > \cdot x_i \cdot x_j$$

该条记录可以编码为 5 个数值特征，即 User^YuChin, Movie^3Idiots, Genre^Comedy, Genre^Drama, Price^9.9 。其中 Genre^Comedy, Genre^Drama 属于同一个 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。假设每条样本的 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。因此可以把 FM 看作是 FFM 的特例，即把所有的特征都归属到一个 field 是的 FFM 模型。根据 FFM 的 field 敏感特性，可以导出其模型表达式：

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^9.9	5

那么，FFM 的组合如下：

<v1,2,v2,1>•1•1+<v1,3,v3,1>•1•1+<v1,3,v4,1>•1•1+<v1,4,v5,1>•1•9.9
+<v2,3,v3,2>•1•1+<v2,3,v4,2>•1•1+<v2,4,v5,2>•1•9.9
+<v3,3,v4,3>•1•1+<v3,4,v5,3>•1•9.9
+<v4,4,v5,3>•1•9.9

其中，红色表示 Field 编码，蓝色表示 Feature 编码，绿色表示样本的组合特征取值。二阶交叉项的系数是通过与 Field 相关的隐向量的内积得到的

FFM 注意点：

样本归一化。对样本进行归一化，否则容易造成数据溢出，梯度计算失败

•特征归一化。为了消除不同特征取值范围不同造成的问题，需要对特征进行归一化

•Early stopping。一定要设置该策略，FFM 很容易过拟合

•省略零值特征。零值特征对模型没有任何贡献，省略零值特征，可以提高 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

DeepFM

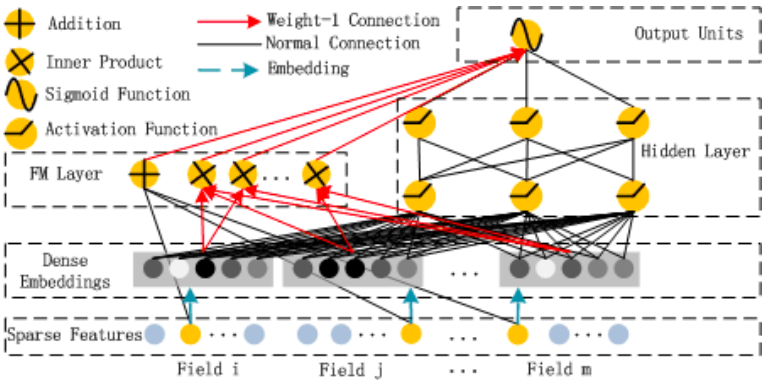
DeepFM 算法有效的结合了因子分解机与神经网络在特征学习中的优点：同时提取到低阶组合特征与高阶组合特征.在 DeepFM 中，FM 算法负责对一阶特征以及由一阶特征两两组合而成的二阶特征进行特征的提取；DNN 算法负责对由输入的一阶特征进行全连接等操作形成的高阶特征进行特征的提取。

1.结合了广度和深度模型的优点，联合训练 FM 模型和 DNN 模型，同时学习低阶特征组合和高阶特征组合。

2.端到端模型，无需特征工程。

3.DeepFM 共享相同的输入和 embedding vector，训练更高效。

4.评估模型时，用到了一个新的指标“Gini Normalization”



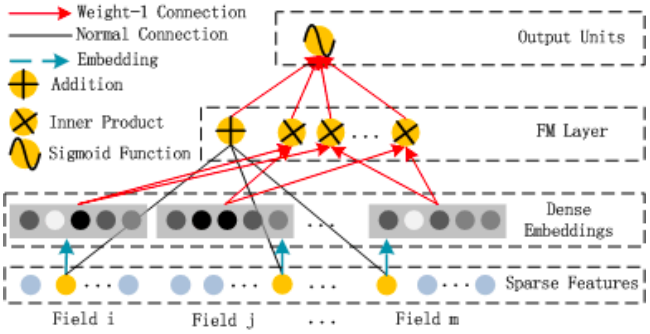
DeepFM 的输入可由连续型变量和类别型变量共同组成，且类别型变量需要进行 One-Hot 编码。而正由于 One-Hot 编码，导致了输入特征变得高维且稀疏。

应对的措施是：针对高维稀疏的输入特征，采用 Word2Vec 的词嵌入（WordEmbedding）思想，把高维稀疏的向量映射到相对低维且向量元素都不为零的空间向量中。

DeepFM 包括 FM 和 DNN 两部分，所以模型最终的输出也由这两部分组成：

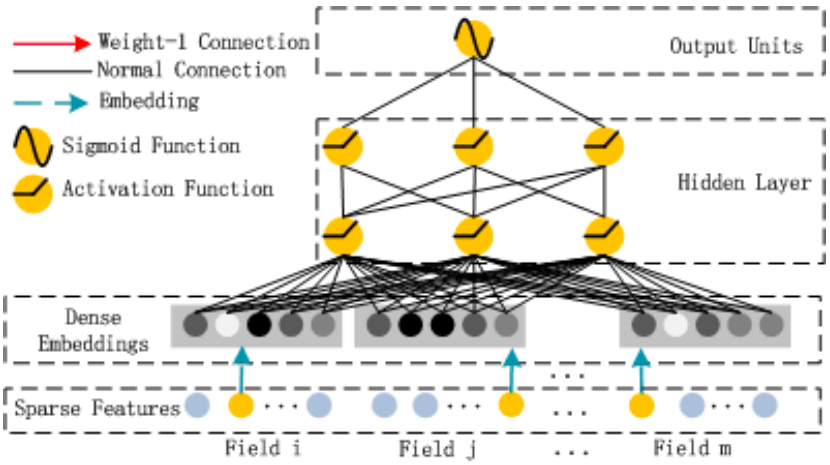
$$\hat{y} = sigmoid(output(FM) + output(DNN))$$

其中 FM 层为：

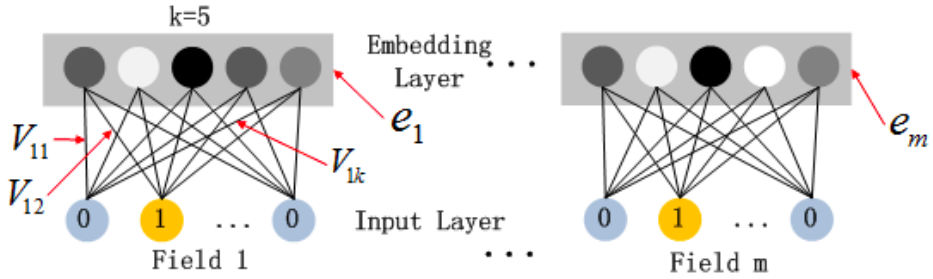


$$y = w_0 + \sum_{i=1}^{n} w_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} < v_i, v_j > \cdot x_i \cdot x_j$$

DeepFM 算法结构图-DNN 层：

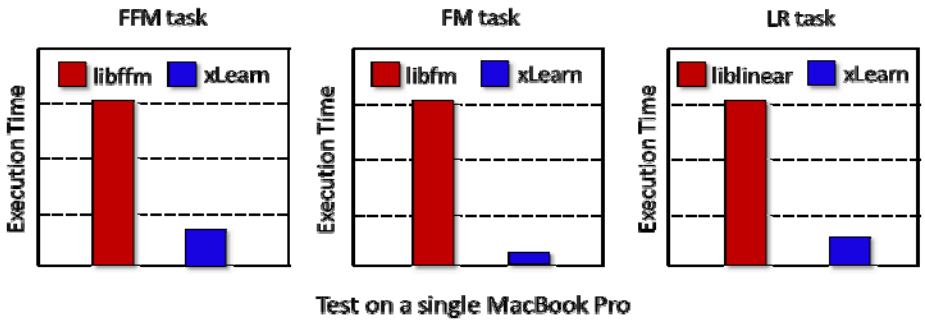


DNN 的作用是构造高维特征，且有一个特点：DNN 的输入也是 embedding vector



XLearn 代码的实现

xLearn 是一款高性能的，易用的，并且可扩展的机器学习算法库，你可以用它来解决大规模机器学习问题，尤其是大规模稀疏数据机器学习问题。在近年来，大规模稀疏数据机器学习算法被广泛应用在各种领域，例如广告点击率预测、推荐系统等。如果你是 liblinear、libfm、libffm 的用户，那么现在 xLearn 将会是你更好的选择，因为 xLearn 几乎囊括了这些系统的全部功能，并且具有更好的性能，易用性，以及可扩展性。



对于 LR 和 FM 算法而言，输入数据格式必须是 CSV 或者 libsvm. 对于 FFM 算法而言，我们的输入数据必须是 libffm 格式。

libsvm format:

y	index_1:value_1	index_2:value_2	...	index_n:value_n
0	0:0.1	1:0.5	3:0.2	...
0	0:0.2	2:0.3	5:0.1	...
1	0:0.2	2:0.3	5:0.1	...

CSV format:

y	value_1	value_2	..	value_n
0	0.1	0.2		0.2	...
1	0.2	0.3		0.1	...
0	0.1	0.2		0.4	...

libffm format:

y	field_1:index_1:value_1	field_2:index_2:value_2	...
0	0:0:0.1	1:1:0.5	2:3:0.2	...
0	0:0:0.2	1:2:0.3	2:5:0.1	...
1	0:0:0.2	1:2:0.3	2:5:0.1	...

xlearn-FM 实现-set 模式

1.初始化 xlearn fm 模型；
2.调用 fm 模型初始化训练集数据
3.设置 fm 模型参数
4.交叉验证

导入模型

[1]:
```
import xlearn as xl
```

初始化fm 模型

[2]:
```
fm_model = xl.create_fm()  # 使用xlearn自带的FM模型
```

直接设置训练集格式模型

[3]:
```
fm_model.setTrain("./titanic/titanic_train.txt")  # 训练数据
```

设置二分类的参数

[4]:
```
# 参数:
#  0. 二分类任务
#  1. learning rate: 0.2
#  2. lambda: 0.002
#  3. metric: accuracy
param = {'task':'binary', 'lr':0.2,
         'lambda':0.002, 'metric':'acc'}
```

使用交叉验证

[6]:
```
fm_model.cv(param)
```

[ ]:

导入模型

[2]:
```
import xlearn as xl
import numpy as np
import pandas as pd
```

初始化fm 模型

[3]:
```
fm_model = xl.create_fm()  # 使用xlearn自带的FM模型
```

DataFrame 格式设置第一列为标签

直接设置训练集格式模型

[4]:
```
titanic_train = pd.read_csv("./titanic/titanic_train.txt", header=None, sep="\t")
titanic_test = pd.read_csv("./titanic/titanic_test.txt", header=None, sep="\t")
```

[5]:
```
titanic_train.head()
```

[5]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	1	0	1	0	0	0	1	0	1	0	0	1	-0.561363	-0.502445
1	1	1	0	0	1	1	0	0	1	0	1	0	0	0.613182	0.786845
2	1	0	0	1	0	0	0	1	1	0	0	0	1	-0.267727	-0.488854
3	1	1	0	0	1	0	0	1	1	0	1	0	0	0.392955	0.420730
4	0	0	0	1	0	0	0	1	0	1	0	0	1	0.392955	-0.486337

[6]:
```
titanic_train[0].value_counts()
```

[6]:
```
0    549
1    342
Name: 0, dtype: int64
```

[7]:
```
titanic_test.head()
```

[7]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	1	0	0	1	0	0	1	0	0	1	0.307535	-0.496637
1	0	1	0	1	0	0	0	1	1	0	0	0	1	1.256230	-0.511497
2	0	0	0	1	0	0	1	0	0	1	0	1	0	2.394665	-0.463335
3	0	0	0	1	0	0	0	1	0	1	0	0	1	-0.261683	-0.481704
4	0	1	1	1	0	0	0	1	1	0	0	0	1	-0.641161	-0.416740

[8]:
```
len(titanic_test)
```

[8]:
```
418
```

[9]:
```
titanic_test[0].value_counts()
```

[9]:
```
0    418
Name: 0, dtype: int64
```

获取训练集的x 和 y

[10]:
```
X_train = titanic_train[titanic_train.columns[1:]]
y_train = titanic_train[0]
```

获取验证集的x 和 y

[11]:
```
X_test = titanic_test[titanic_test.columns[1:]]
y_test = titanic_test[0]
```

0 为标签第一列

设置DMatrix 格式

[12]:
```
xdm_train = xl.DMatrix(X_train, y_train)
xdm_test = xl.DMatrix(X_test, y_test)
```

初始化fm 模型

[13]:
```
fm_model = xl.create_fm()
```

设置训练集和验证集合

设置训练集格式

[14]:
```
fm_model.setTrain(xdm_train)    # Training data
fm_model.setValidate(xdm_test)  # Validation data
```

[15]:
```
# 参数:
#  0. 二分类任务
#  1. learning rate: 0.2
#  2. lambda: 0.002
#  3. metric: accuracy
param = {'task':'binary', 'lr':0.2,
         'lambda':0.002, 'metric':'acc'}
```

初始化后的模型导入参数

[16]:
```
fm_model.fit(param, './temp/model_dm.out')
```

设置测试集和输出分类 0-1

[17]:
```
#### 设置测试集数据并且把输出生成为二分概率
```

[18]:
```
fm_model.setTest(xdm_test)  # Test data
fm_model.setSigmoid()  # Convert output to 0-1
```

[19]:
```
res = fm_model.predict("./temp/model_dm.out")

print(res)
```

```
[0.1338213  0.36157772 0.22472496 0.12465371 0.48475164 0.18462412
 0.5019446  0.26714218 0.5698421  0.19003786 0.12160286 0.21920532
 0.8255089  0.20912342 0.72180945 0.682923   0.22287293 0.18839796
 0.43303072 0.45479497 0.30163878 0.23711687 0.55498034 0.45130998
 0.7577094  0.12662041 0.87881887 0.18111108 0.36309162 0.21678676
 0.19636144 0.26154596 0.47545007 0.5292689  0.5100306  0.20154886
 0.44465882 0.48877826 0.1316185  0.16385806 0.1807116  0.33502677
 0.10958893 0.6190658  0.7256025  0.13077226 0.41956092 0.13296005
```

保存结果

* 内容来自开课吧第十课课件