華東理工大學

模式识别大作业

题　　目　Criteo 展示广告

学　　院　信息科学与工程学院

专　　业　控制科学与工程

组　　员　岳丹阳（Y30180694）

指导教师　赵海涛

完成日期：　**2018** 年 **10** 月 **25** 日

模式识别作业报告——Criteo 展示广告预测用户是否会点击广告

组员：岳丹阳

一、题目介绍

Criteo 是一家第三方展示广告公司，与世界上超过 4000 家电子商务公司有合作关系。说到广告，关注的最多的就是点击率了。我们经常能听说某某科学家通过建立更好的点击率预测模型，为公司带来上亿的增量收入。

该题使用 Criteo 所共享的一周展示广告数据，在数据中提炼了 13 个连续特征、26 个离散特征和用户是否点击了该页面广告的标签。要求训练出合适的模型，预测用户在不同的特征下是否会点击广告。题目为我们提供了训练数据集 train.csv，测试数据集 test.csv 和一个提交的数据集 submission.csv。数据文件 train.csv 提供了 1599 条的用户访问网页和点击广告记录的对应特征，l1～l13 为计数特征，c1～c26 为类别特征。Label 表示用户是否点击广告，0 为未点击，1 为点击。数据文件 test.csv 与 train.csv 类似，提供了 train.csv 之后一段时间的用户访问网页和点击广告记录对应特征。我们先使用训练数据训练模型，再使用测试数据预测用户的点击概率。

二、整体解决方案

2.1 方案分析

最近几年广告系统成为很多公司的重要系统之一，点击率预估是定向广告技术中的重要组成部分，在解决广告点击概率方面常用 Logistic Regression 逻辑回归进行预测概率。Logistic Regression 逻辑回归主要与分用于类问题，常用来预测概率，如知道一个人的年龄、体重、身高、血压等信息，预测器患有心脏病的概率是多少。

然而逻辑回归于线性回归(Linear Regression)有什么区别？为什么采用 LR 回归？普通线性回归主要用于连续变量的预测，即线性回归的输出 y 的取值范围是整个实数区间；逻辑回归用于离散变量的分类，即它的输出 y 的取值范围是一个离散的集合，主要用于类的判别，而且其输出值 y 表示属于某一类的概。其次，选择 LogisticRegression 预估点击率主要是因为：

(1) 数据规模较大时，LR 在训练和预测方面的计算复杂度都很低

(2) LR 不仅可以预测一个样本属于哪一类的，还可以给出属于每一类概率大小

(3) LR 的模型简单，解决预测结果也相对容易

在使用 Logistic Regression 回归做出预测后，可能会发现结果不是太理想，可以考虑加入 L1 正则。正则化即惩罚函数，对模型向量进行“惩罚”，从而避免在回归过程中出现的过拟合现象。

最后，如果利用已有的特征采用上述方法仍然不能获得较好的结果，说明特征不能很好的表达数据，可以采用 GBDT(梯度提升决策树)构建新的特征使便于更好地表达数据。然后将 GBDT 与 LR 融合进行模型的预测。

2.2 数据结构分析

在进行数据集的训练之前首先要观察训练数据，观察数据的特征，Train 数据如下图所示：

Id	Label	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	I13
10000743	1	1	0	1		227	1	173	18	50	1	7	1	
10000159	1	4	1	1	2	27	2	4	2	2	1	1		2
10001166	1	0	806			1752	142	2	0	50	0	1		
10000318	0	2	-1	42	14	302	38	25	38	90	1	3		38
10000924	1	0	57	2	1	2891	2	35	1	137	0	17		1
10000532	0	0	67	3	12	1470	52	14	6	72	0	1	0	12
10000485	1	1	12	10	5	25	5	45	7	1553	1	17	1	5
10000251	0	0	0	3	4	4520	158	28	23	639	0	5	0	6
10001288	1	2	3	12	33	14	15	2	29	34	1	1	1	10
10001697	0		51	1	1	2278		0	16	41		0		7
10000190	0	1	1	4	17	108	22	1	24	22	1	1		22
10001836	0		0			6531	23	2	2	2		1		
10000615	1		1	4	1	4870	19	13	7	75		6		1
10001665	0		-1					0	1	1		0		
10001500	0	1	-1	1	3	447	3	3	6	6	1	2		3
10000351	0		179						0					
10001217	1	0	2		7	5790	104	3	9	103	0	1		38
10001524	1	0	-1			3237	116	18	17	357	0	5		
10001361	0	0	11		3	2825	129	12	44	212	0	5	0	3
10000740	0	6	14	54	20	60	21	143	39	934	2	31	3	21
10001087	0	2	1	1	12	31	12	40	40	124	1	6	1	12
10000253	0		508	6	6	14166	37	1	6	63		1		6

C1	C2	C3	C4	C5	C6	C7	C8	C9
75ac2fe6	1cfdf714	713fbe7c	aa65a61e	25c83c98	3bf701e7	7195046d	0b153874	a7
05db9164	6c9c9cf3	2730ec9c	5400db8b	25c83c98	7e0ccccf	8a6600b0	813607cc	a7
05db9164	207b2d81	19d853db	cc4ecbc8	4cf72387	7e0ccccf	b5043bcd	0b153874	a7
05db9164	09e68b86	983fc1f4	49045e9d	25c83c98	7e0ccccf	50b436c9	0b153874	a7
05db9164	2a69d406	76c78125	13508380	25c83c98	7e0ccccf	0433a7db	0fb392dd	a7
5a9ed9b0	421b43cd	c9c2ec4d	29998ed1	25c83c98	fe6b92e5	7076d935	0b153874	a7
5a9ed9b0	b0d4a6f6	73006735	97ceb02f	25c83c98	fbad5c96	407438c8	0b153874	a7
68fd1e64	b46aceb6	7fc6b941	8a8d0597	43b19349	fe6b92e5	4d90c4ed	6c41e35e	a7
05db9164	00ac063c	f483d069	a089e42f	25c83c98		d4ab9344	5b392875	a7
439a44a4	38a947a1	bc2aea05	ac975db6	25c83c98	7e0ccccf	7079e499	0b153874	a7

图 1 原始数据结果

该训练数据提供了 1599 条用户的访问网页和点击广告的记录，其中 I1~I13 为计数特征，C1~C26 为类别特征，Label 表示用户是否点击广告。ID 为用户身份。

考虑计算的复杂度和个人能力，我们采用计数特征用于训练数据，去除类别特征。观察数据可以看出在 I1~I13 计数特征中，存在了一些问题：

第一、数据集中很明显缺失了大量的数据，不利于数据集的训练需要我们进行预处理进行填充，填充数据特征的中位数或特征的平均值。

第二、在数据集中发现一些异常的数据，不符合实际要求，可以对一些异常数据进行处理。

第三、在某些特征数据中，用户的数据缺失较多，仅有少量的用户有计数信息，如 I12 特征缺失太多的数据，考虑该特征作用不大，可以去除该个特征。

第四、在数据集中发现各个特征的数据范围各不相同且差异较大，并且还存在着负数，因此需要对数据进行归一化处理。。

最终，考虑到算法的复杂度和实际需要以及编程复杂度问题，在原始数据表中只采用了 12 列数据的特征、用户 ID、Label。Test 数据集中于 Train 数据的特征一样。需要进行同样的处理。

2.2 数据读入与数据的预处理

编程工具我们采用 Python 语言，在 SublimeText3 编译器中进行代码的编写与调试。首先应该导入我们需要的数据，即训练集 train 和测试集 test。读取数据主要使用了 python 中的 Pandas 模块，可将数据集读取为数据框架，如图 3 所示：

	Id	Label	I1	...	C24	C25	C26
0	10000743	1	1.0	...	4d19a3eb	cb079c2d	456c12a0
1	10000159	1	4.0	...	72c78f11	NaN	NaN
2	10001166	1	0.0	...	8fc66e78	001f3601	f37f3967
3	10000318	0	2.0	...	1793a828	e8b83407	5cef228f
4	10000924	1	0.0	...	45ab94c8	2bf691b1	c84c4aec
5	10000532	0	0.0	...	b34f3128	NaN	NaN
6	10000485	1	1.0	...	732eaaee	NaN	NaN
7	10000251	0	0.0	...	38be899f	NaN	NaN
8	10001288	1	2.0	...	c2fe6ca4	001f3601	602f0609
9	10001697	0	NaN	...	d28d80ac	NaN	NaN
10	10000190	0	1.0	...	aee52b6f	NaN	NaN
11	10001836	0	NaN	...	b34f3128	NaN	NaN
12	10000615	1	NaN	...	0cae79aa	NaN	NaN
13	10001665	0	NaN	...	7fb4ff91	NaN	NaN
14	10001500	0	1.0	...	6e9a987f	NaN	NaN
15	10000351	0	NaN	...	88ed380d	NaN	NaN
16	10001217	1	0.0	...	aee52b6f	NaN	NaN

图 1 原始数据的读入

数据框架提取过后，还需要将所提取的数据转换为我们所需要的数据矩阵，提出我们我需要的数据特征和点击结果 Label。Train 数据集与 Test 数据集处理方法一样。

该部分 Python 程序代码如下所示：

```
# 训练集与测试集的提取
app_train = pd.read_csv('train.csv')
app_test = pd.read_csv('test.csv')

#对数据进行处理
#对train集进行处理：删除不需要的类别特征和I12特征,删除ID，提出Label
train_labels = app_train['Label']
train_del=[0,1,13,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36
,37,38,39,40]
app_train.drop(app_train.columns[train_del],axis=1,inplace=True)

#对test集进行处理：处理方法同上，提出ID
submit = app_test[['Id']]
test_del =
[0,12,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39
]
app_test.drop(app_test.columns[test_del],axis=1,inplace=True)

#得到训练测试数据集
train = app_train.copy()
test = app_test.copy()
```

提取所需要的数据之后便需要对数据进行预处理，填补缺失的数据，常用的填补方法有填补特征的平均值或特征的中位数。填充后的数据如下图所示：

```
train:
 [[  1.   0.   1. ...   1.   7.   5.]
 [  4.   1.   1. ...   1.   1.   2.]
 [  0. 806.   6. ...   0.   1.   5.]
 ...
 [  0.   1.   2. ...   0.   2.  12.]
 [  1.   2.   6. ...   0.   0.   1.]
 [  1.  34.   3. ...   0.   0.   4.]]
test:
 [[  1.  -1.   6. ...   0.   2.   5.]
 [  1.  -1.   6. ...   0.   1.   5.]
 [  0.   0.   2. ...   0.   1.  13.]
 ...
 [  0. 300.   4. ...   0.   2.   5.]
 [  1.   1.   2. ...   1.   1.   1.]
 [  1.   2.   6. ...   0.   1.   5.]]
```

图 2 填补后的数据

填充完数据之后，还需要对数据进行标准化处理，规范到 0-1 之间，归一化后的结果如下图所示：

```
normalize_train:
 [[1.05263158e-02 2.54258835e-04 1.18245241e-04 ... 2.50000000e-01
  1.89189189e-01 6.49350649e-03]
 [4.21052632e-02 3.81388253e-04 1.18245241e-04 ... 2.50000000e-01
  2.70270270e-02 2.59740260e-03]
 [0.00000000e+00 1.02720570e-01 7.09471444e-04 ... 0.00000000e+00
  2.70270270e-02 6.49350649e-03]
 ...
 [0.00000000e+00 3.81388253e-04 2.36490481e-04 ... 0.00000000e+00
  5.40540541e-02 1.55844156e-02]
 [1.05263158e-02 5.08517671e-04 7.09471444e-04 ... 0.00000000e+00
  0.00000000e+00 1.29870130e-03]
 [1.05263158e-02 4.57665904e-03 3.54735722e-04 ... 0.00000000e+00
  0.00000000e+00 5.19480519e-03]]
normalize_test:
 [[1.05263158e-02 1.27129418e-04 7.09471444e-04 ... 0.00000000e+00
  5.40540541e-02 6.49350649e-03]
 [1.05263158e-02 1.27129418e-04 7.09471444e-04 ... 0.00000000e+00
  2.70270270e-02 6.49350649e-03]
 [0.00000000e+00 2.54258835e-04 2.36490481e-04 ... 0.00000000e+00
  2.70270270e-02 1.68831169e-02]
 ...
 [0.00000000e+00 3.83930842e-02 4.72980963e-04 ... 0.00000000e+00
  5.40540541e-02 6.49350649e-03]
 [1.05263158e-02 3.81388253e-04 2.36490481e-04 ... 2.50000000e-01
  2.70270270e-02 1.29870130e-03]
 [1.05263158e-02 5.08517671e-04 7.09471444e-04 ... 0.00000000e+00
  2.70270270e-02 6.49350649e-03]]
Training data shape:  (1599, 12)
Testing data shape:  (400, 12)
```

图 3 （0，1）标准化后的数据

该部分数据处理调用 sklearn.preprocessing 中的 MinMaxScaler,Imputer 进行数据处理，该部分 Python 程序代码如下：

```
#进行数据填充，填充中位数/平均值
imputer = Imputer(strategy='median')
imputer.fit(train)
train = imputer.transform(train)
test = imputer.transform(test)

#进行数据标准化
scaler = MinMaxScaler(feature_range=(0, 1))
scaler.fit(train)
train = scaler.transform(train)
test = scaler.transform(test)
```

2.3 算法原理及程序实现

2.3.1 Logistics Regression

对于 Logistic Regression 来说，其基本思想也是基于线性回归再加个 sigmoid 函数，其公式如下所示：

$$h_\theta(x)=\frac{1}{1+e^{-z}}=\frac{1}{1+e^{-\theta^T x}}$$

其中 $\theta^T=[\theta_0,\theta_1,\theta_2,...,\theta_n]$，$\frac{1}{1+e^{-z}}$ 函数图像如下所示：

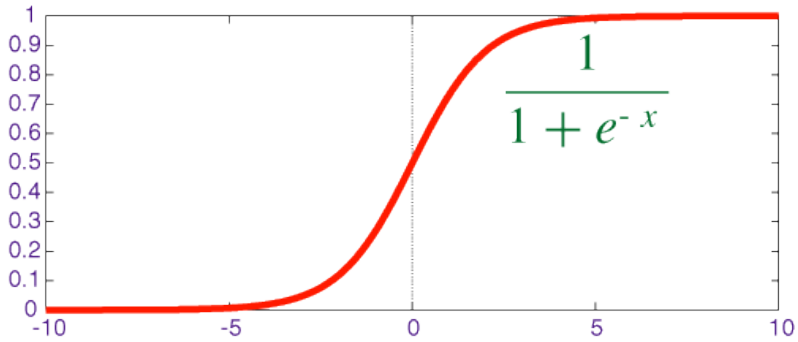

图 4　Sigmoid 函数图像

Logistic Regression 算法是将线性函数的结果 z 映射到 sigmoid 函数中，因此 $h_\theta(x)$ 的输出范围为（0，1），因此数据可以分成 0，1 类别的数据，也可以设置分类范围，不仅仅是 0，1 分类。因此可以将 Sigmoid 函数看成样本数据的概率密度函数。接着便需要我们估计模型的参数 θ，当 $h_\theta(x)$ 结果为取 1 的概率时，则对于输入 x 分类结果为 1 和 0 的概率分别为：

$$P(y=1\mid x;\theta)=h_\theta(x)$$
$$P(y=0\mid x;\theta)=1-h_\theta(x)$$

这样的模型就是 Logistic Regression 模型，对于给定的训练数据集 $T=\{(x_1,y_1),(x_2,y_2),...,(x_N,y_N)\}$，其中，$x_i\in R^n, y_i\in\{0,1\}$，$x^i$ 为 x 的第 i 个特征，可以应用极大似然估计法估计模型参数 θ，似然函数为：

$$L(\theta)=\prod_i^m\left(h_\theta(x^i)\right)^{y^i}*\left(1-h_\theta(x^i)\right)^{1-y^i}$$

取对数似然估计函数：

$$l(\theta)=\log(L(\theta))=\sum_{i=1}^m\left[y^i\log\left(h_\theta(x^i)\right)+(1-y^i)\log\left(1-h_\theta(x^i)\right)\right]$$

对$l(\theta)$求极大值，便可以得到参数的估计值，实际我们可以构建$J(\theta)=-\frac{1}{m}l(\theta)$函数，这样就转化为求函数$J(\theta)$取最小值时得到参数$\theta$的最佳估计。而求解$J(\theta)$的最小值可以使用梯度下降法，根据梯度下降法可得参数θ的更新公式：

$$\theta_{j+1}=\theta_j-\alpha\frac{\partial}{\partial\theta_j}J(\theta)$$

其中α为学习率，$J(\theta)$如下：

$$J(\theta)=-\frac{1}{m}\sum_{i=1}^{m}\left[y^i\log\left(h_\theta\left(x^i\right)\right)+\left(1-y^i\right)\log\left(1-h_\theta\left(x^i\right)\right)\right]$$

通过对$J(\theta)$求偏导：

$$\begin{aligned}\frac{\partial}{\partial\theta_j}J(\theta)&=-\frac{1}{m}\sum_{i=1}^{m}\left[y^i\frac{1}{h_\theta\left(x^i\right)}\frac{\partial}{\partial\theta_j}\left(h_\theta\left(x^i\right)\right)-\left(1-y^i\right)\frac{1}{1-h_\theta\left(x^i\right)}\frac{\partial}{\partial\theta_j}\left(h_\theta\left(x^i\right)\right)\right]\\&=-\frac{1}{m}\sum_{i=1}^{m}\left[y^i\frac{1}{h_\theta\left(x^i\right)}-\left(1-y^i\right)\frac{1}{1-h_\theta\left(x^i\right)}\right]\frac{\partial}{\partial\theta_j}\left(h_\theta\left(x^i\right)\right)\\&=-\frac{1}{m}\sum_{i=1}^{m}\left[y^i\frac{1}{h_\theta\left(x^i\right)\left(1-h_\theta\left(x^i\right)\right)}-\frac{1}{1-h_\theta\left(x^i\right)}\right]h_\theta\left(x^i\right)\left(1-h_\theta\left(x^i\right)\right)\frac{\partial}{\partial\theta_j}\theta^Tx^i\\&=\frac{1}{m}\sum_{i=1}^{m}\left[h_\theta\left(x^i\right)-y^i\right]*x_j^i\end{aligned}$$

将此式代入参数更新公式，省略 1/m 可得：

$$\theta_{i+1}=\theta_i-\alpha\sum_{i=1}^{m}\left[h_\theta\left(x^i\right)-y^i\right]*x_j^i,\left(j=1,2,...n\right)$$

综上可以得知 Logist Regression 算法推导过程如下：

1. 输入数据训练集$T=\{(x_1,y_1),(x_2,y_2),...,(x_N,y_N)\}$，构建 Logistic Regression 模型。
2. 构建$J(\theta)$函数，利用梯度下降法不断更新迭代使得$J(\theta)$最小，此时便可以得到训练完成的模型参数θ
3. 接着便可以利用模型参数构建的$h_\theta(x)=\frac{1}{1+e^{-\theta^Tx}}$公式来计算测试数据的结果，从而进行预测概率或这进行分类。

2.3.2 Gradient Boosting Decesion Tree

决策树是一种基本的分类与回归方法，决策树模型呈树型结构，在分类问题中，表示基于特征对实例进行分类的过程。其主要优点是模型具有可读性，分类速度快。

提升方法（boosting）在分类问题中，通过改变训练样本的权重（增加分错样本的权重，减小分对样本的的权重），学习多个分类器，并将这些分类器线性组合，提高分类器性能。boosting 数学表示为：

$$f\left(x\right)=w_0+\sum_{m=1}^{\mathrm{M}}w_m\phi_m\left(x\right)$$

其中 w 是权重，ϕ 是弱分类器的集合，可以看出最终就是基函数的线性组合，决策树与 boosting 结合可以产生 GBDT 算法。GBDT 的思想可以用一个通俗的例子解释，假如有个人 30 岁，我们首先用 20 岁去拟合，发现损失有 10 岁，这时我们用 6 岁去拟合剩下的损失，发现差距还有 4 岁，第三轮我们用 3 岁拟合剩下的差距，差距就只有一岁了。如果我们的迭代轮数还没有完，可以继续迭代下面，每一轮迭代，拟合的岁数误差都会减小。

假设输入训练数据集 $T=\{(x_1,y_1),(x_2,y_2),...,(x_N,y_N)\}$，其中，$x_i\in R^n,y_i\in\gamma=\{-1,,i=1\}$　　N，首先初始化一个弱的学习器：

$$f_0\left(x\right)=\arg\min_c\sum_{i=1}^{N}L\left(y_i,c\right)$$

对以下部分进行 M 次迭代：

⑴对于样本 $i=1,2,...,N$，计算损失函数的负梯度：

$$r_{mi}=-\left[\frac{\partial L\left(y,f\left(x_i\right)\right)}{\partial f\left(x_i\right)}\right]_{f(x)=f_{m-1}(x)}$$

⑵利用（x_i，r_{mi}）拟合一颗回归树，得到第 m 棵树的叶结点区域 R_{mj}，j=1，2，...J,即一颗由 J 个叶子节点组成的树。

⑶对 j=1，2，...J，计算最佳拟合值：

$$c_{mj} = \arg\min_{c} \sum_{x_i \in R_{mj}} L\left(y_i, f_{m-1}\left(x_i\right) + c\right)$$

(4)更新学习器：

$$f_m\left(x\right) = f_{m-1}\left(x\right) + \sum_{j=1}^{J} c_{mj} I, x_i \in R_{mj}$$

经过上述的 M 次迭代计算后，得到最终的回归树模型：

$$f_M\left(x\right) = \sum_{m=1}^{M} f_m\left(x\right) = \sum_{m=1}^{M} \sum_{j=1}^{J} c_{mj} I, x_i \in R_{mj}$$

在使用 Python 编程时可以从 sklearn.ensemble 中调取 GradientBoostingClassifier，先调取 GBDT 分类模型进行训练，构建分类特；然后利用 OneHot 编码将分类特征的每个元素转化为为可以用来计算的值作为新的特征数据输入到 Logistic 回归模型中进行训练，并用转化后的数据进行预测广告点击的概率。

该部分代码如下：

```
# 调用GBDT分类模型。
grd = GradientBoostingClassifier(n_estimators=10)
# 调用one-hot编码。
grd_enc = OneHotEncoder()
# 调用LR分类模型。
grd_lr = LogisticRegression(penalty='l1',C=0.2)
'''使用X_train训练GBDT模型，后面用此模型构造特征'''
grd.fit(train,train_labels)
# fit one-hot编码器
grd_enc.fit(grd.apply(train)[:, :, 0])

'''
使用训练好的GBDT模型构建特征，然后将特征经过one-hot编码作为新的特征输入到LR模型训练。
'''
grd_lr.fit(grd_enc.transform(grd.apply(train)[:, :, 0]), train_labels)

log_reg_prob = grd_lr.predict_proba(grd_enc.transform(grd.apply(test)[:, :, 0]))[:,1]

print(log_reg_prob)

#绘制预测概率图像
plt.plot(log_reg_prob,color='r')
plt.ylabel("Probility")
plt.show()
```

2.4 评定测试

根据 Criteo 广告点击的题目要求，使用 Logarithmic Loss 作为最后评判标准，公式如下：

$$\log\text{loss} = -\frac{1}{n}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log\left(p_{i,j}\right)$$

其中 N 代表测试数据集中访问记录的数量，其中 M 代表测试数据集中预测的分类数量（该题中为 2，代表预测点击与未点击），$y_{i,j}$ 代表其真实的点击情况（0 为未点击，1 为点击），$\log\left(p_{i,j}\right)$ 代表预测的点击概率。

由于点击结果只有{0，1}两类，则对数损失函数的公式简化为：

$$\log\text{loss} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i\log\left(p_i\right)+\left(1-y_i\right)\log\left(1-p_i\right)\right)$$

y_i 为真实的点击情况，p_i 为预测的点击概率。结果评估的 Python 程序代码如下：

```
#Logarithmic Loss评估函数
def logloss(y_true, y_pred, eps=1e-15):
    import numpy as np

    # Prepare numpy array data
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)
    assert (len(y_true) and len(y_true) == len(y_pred))

    # Clip y_pred between eps and 1-eps
    p = np.clip(y_pred, eps, 1-eps)
    loss = np.sum(- y_true * np.log(p) - (1 - y_true) * np.log(1-p))

    return loss / len(y_true)
```

2.5 调试及预测结果

为了得到较好的预测结果，在实验调试过程中对参数进行了多次调试，来观察评估结果。Logloss 获得结果越小越优。在未加入 GBDT 算法优化前，LogisticRegression 的参数 C=1，获得最优评估结果在 0.46 附近，预测结果的概率和 logloss 结果如下图所示：

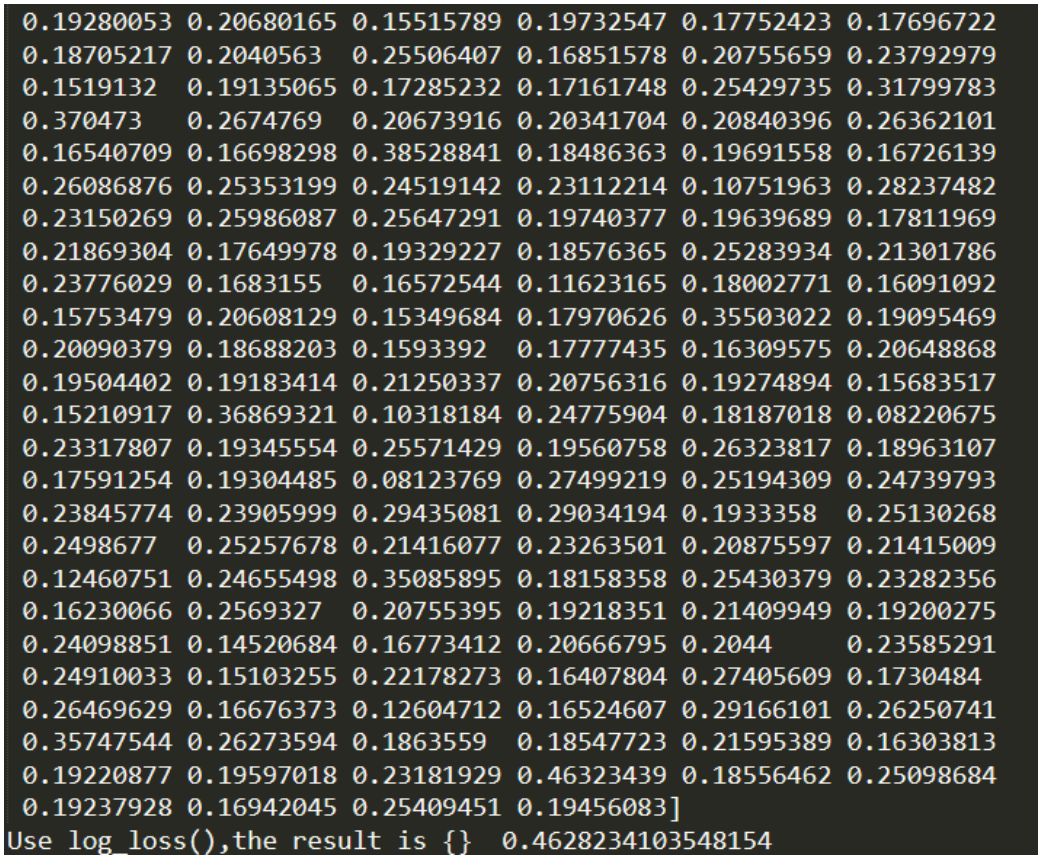

图 5　LR 预测的广告点击概率及 logloss 结果

增加 GBDT 分类模型之后，调试过程中主要调试的参数 LogisticRegression 的参数 C，C 在 1、0.1、0.01、0.001 下所获得 logloss 评估结果，如下表所示：

LogisticRegression 参数 C	填补平均值	填补中位数
1	0.43935	0.45122
0.1	0.42657	0.43711
0.01	0.48363	0.48532
0.001	0.69315	0.69315

由上表可以看出 C=0.1,对缺失的数据使用平均值补充时所得效果最好，实验所得的评估结果为 0.42657，点击率部分预测结果如下图所示：

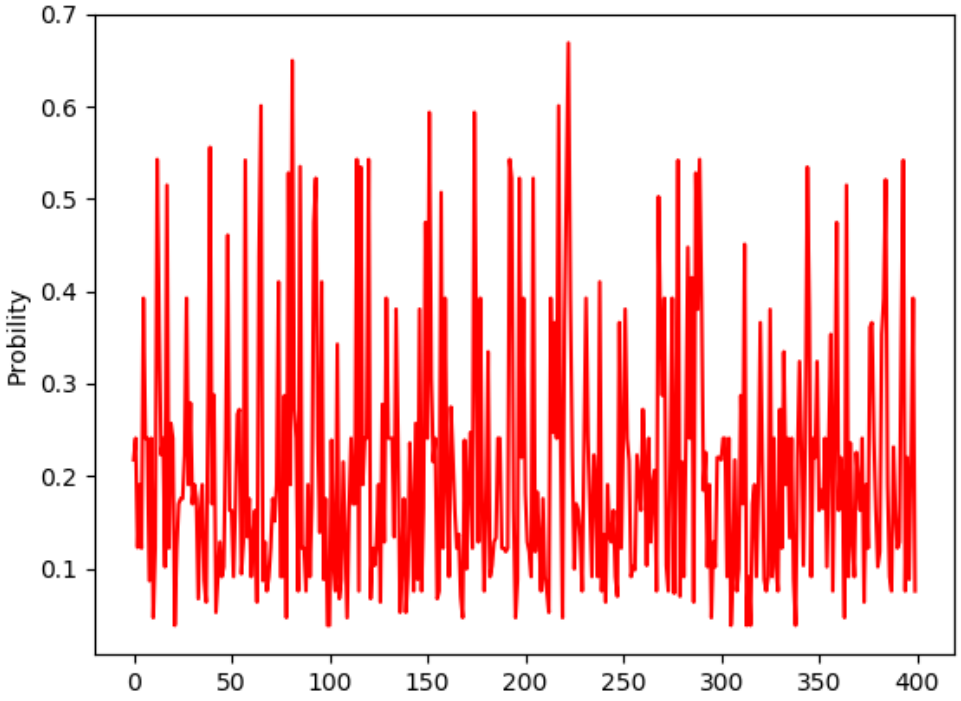

图 6　最终预测的广告点击概率

最终，将所得预测概率，到处到 submission.csv 的文件中，然后提交到 LintCode 中原题中，得到最终预测结果分数 0.43874，如图所示：

#	队伍名称	分数	成员
1	**seagullbird**	0.00000	
2	**HaoliZhang**	0.00000	
3	**SinceJune**	0.00000	
4	**ysf**	0.42957	
5	**yaohualiu2018**	0.43230	
6	**768219451**	0.43874	
7	**yinccc**	0.44461	
8	**wuwuwu123**	0.44899	
9	**shanshan13**	0.45488	
10	**__ww__**	0.45599	

图 4　提交 lintcode 后的分数

四、作业总结

Criteo 展示广告——预测用户是否会点击广告这道题，初始一看真是毫无头绪，这么大的数据集，这么多的特征，还缺失好多数据，真是难以处理。置于算法方面一开始想的是采用线性回归，但是结果实在不太好，还是采用了赵老师所说的 LogisticRegression 方法对模型进行学习预测。

此外，还要多亏了 Criteo 展示广告这题的题目概述部分，给了一些小提示和先修技能。小提示给了这样的两条建议：(1)数据预处理：本题可以考虑对连续特征做等频的离散化处理，对类目特征做 one hot encoding。(2) 模型：可以考虑主流的“logistic regression + L1 正则”。其中还告诉采用梯度提升决策树 GBDT 做回归预测，并且 GBDT 的思想具有天然优势可以发现多种有区分性的特征以及特征组合。从这些概述中真是收益颇多，于是我就相关的知识点查到资料理解这些算法的原理和使用方法。在数据预处理方面，首先提出需要的特征值，对缺失的数据进行补充，如补充平均值、中位值和众数。补全数据便进行了 0-1 之间的标准化处理。

在初始数据处理好后，便开始构建 Logistic Regression 回归模型，进行数据训练预测结果，一开始预测结果不是太好，于是对 Logistic Regression 的参数 C 进行调节，并考虑加入 L1 正则化防止过拟合现象。最终为了使得预测的结果进一步的优化，我调用了 GBDT 分类模型，然后使用训练好的 GBDT 模型构建特征，然后将此特征经过 one-hot 编码作为新的特征输入到 Logistic Regression 的模型中进行训练，得到的广告点击概率预测结果优于原来的单纯 Logistic Regression 的预测结果。

这次的大作业使得我收获颇丰，原来不太清晰的算法理论在做实验的过程中了解的更加清晰，并且也增加 python 编程的熟练度，学习到了更多。最后感谢赵老师的教学指导，使得我在解决问题的过程中少走了一些弯路。

附：文件说明

本次附件一共包含有：

1 大作业报告；

2 最终的 Python 实现程序源码：Criteo.py

3 导出了概率预测数据集：submit .csv