

**模式识别大作业**

**题目 广告点击预测**

**院系 信息科学与工程学院**

**专业 控制科学与工程**

**组员 应桃**

**指导老师**   **赵海涛**

**2018 年 10 月25日**

**一、广告点击预测简介**

说到广告，关注的最多的就是点击率了。在网络广告中，点击率(click rate)是在HTML网页上的一条广告打开后被点击的次数百分比。如果10个人中有一个人点击了打开的页面上的一条广告，这条广告的点击率就是10%。通过对点击率的分析，可以清晰的展示用户的使用偏好，进而针对性的投放，给企业带来收益。因此我们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也就不足为奇了。

在本题中，我们使用Criteo所共享的一周展示广告数据，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。由于能力所限，所以只考虑其中13个连续的数字特征确定出合适的模型，并预测用户在不同的特征下是否会点击广告。

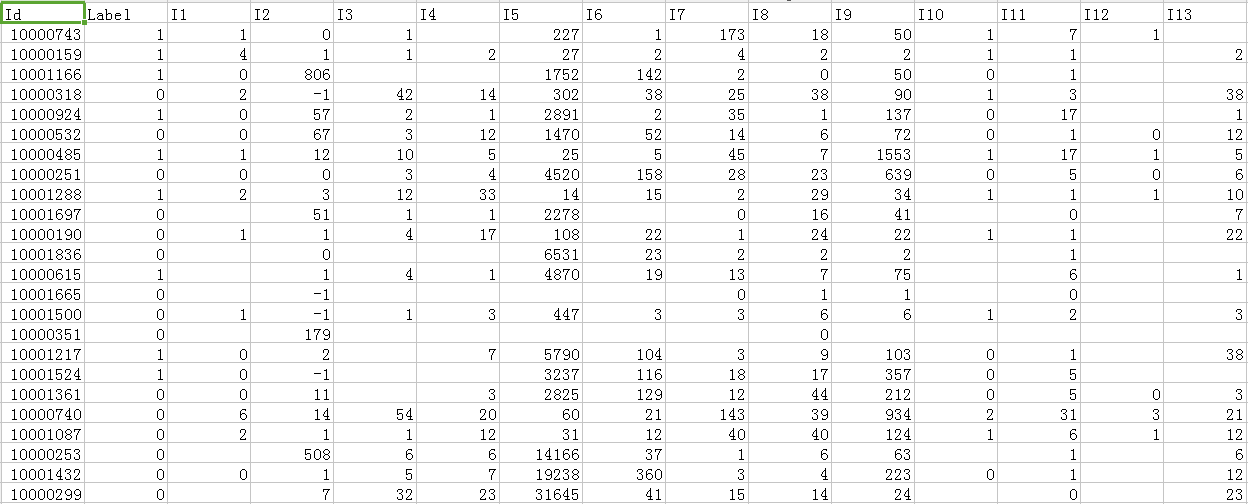
**二、整体解决方案**

将最终广告的点击与否抽象为一个0-1分类的logistic regression问题，在仔细思考与对原始数据研究的基础上，我首先确定了一个大的求解的框架，就是把这个问题看成一个分类问题，利用数据，找出影响广告点击的因素，并且对这些“因素”与“是否点击”这两者关系进行建模。

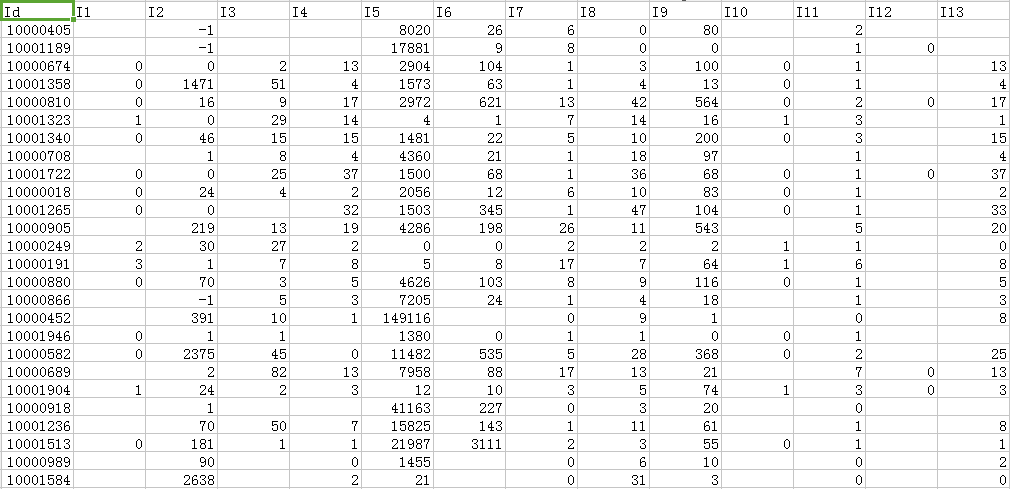
建立模型后，在训练数据上反复调试模型中的部分参数，使得分类器对于训练数据达到较好的分类效果后，将该模型用于测试数据的预测，并分析其预测效果。

**2.1 数据读入**

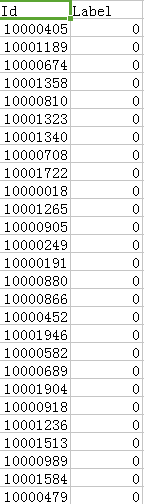
* 数据从可从Lintcode网站直接下载。数据包括训练数据train.csv，测试数据test.csv和提交集submission.csv。内容依次如图所示：



**图1训练集样本信息**



**图2测试集信息**



**图3所需提交集信息**

其中训练数据一共有1599组，测试数据有400组。由于数据标签未给定具体含义，故无法确定各标签数据重要程度，因此将所有数据读取并进行处理，其标签依次为“I1”至“I13”。本次程序主要用Python编写，为进行数据训练及测试，必须读取对应的数据。本次实验读取数据主要使用了Pandas模块，可将数据集读取为数据框架。其python代码如下：

train = pd.read\_csv('C:\\Users\\Administrator\\Desktop\\train.csv') #训练集读入

indata = train.ix[0:,2:15] #选择其中所需I1-I13的数据

test = pd.read\_csv('C:\\Users\\Administrator\\Desktop\\test.csv') #测试集读入

intest = test.ix[0:,1:14] #选择其中所需I1-I13的数据

output = pd.read\_csv('C:\\Users\\Administrator\\Desktop\\submission.csv')

**2.2 数据预处理**

* 数据中明显存在数据缺失等情况。

当有数据缺失的记录在整个数据中只占一个很小比例时，可以直接删除缺失记录，对余下的完全数据进行处理。但是在实际数据中，往往缺失数据占有相当大的比重，这样做不仅会产生偏差，甚至会得出有误导性的结论，同时丢失大量信息，造成浪费。因此我们需要使用一种新的方法来进行处理。填补法是处理数据缺失时普遍使用的一种技术，就是说给各个缺失数据找一个填充值，用这样的方法得到“完整数据”，然后用标准正常的完整数据的统计方法进行数据分析和推断。包括平均值填充法、热卡填充法（或就近补齐）、使用任何可能的值填充等方法。

* 在本次代码中，采取了中位值填充法。其python代码如下：

#依次将所需I1-I13的均值赋给其缺失值

imputer = Imputer(strategy='median') #赋中位值

imputer.fit(indata)

indata = imputer.transform(indata) #训练集I1-I13数据

intest = imputer.transform(intest) #测试集I1-I13数据

* 由于数据间的差距较大，故需将其进行标准化处理。其python代码如下：

minmax\_data\_train = MinMaxScaler(feature\_range=(0, 1))

data\_train\_std = minmax\_data\_train.fit\_transform(data\_train).T

minmax\_data\_test = MinMaxScaler(feature\_range=(0, 1))

data\_test\_std = minmax\_data\_test.fit\_transform(intest).T

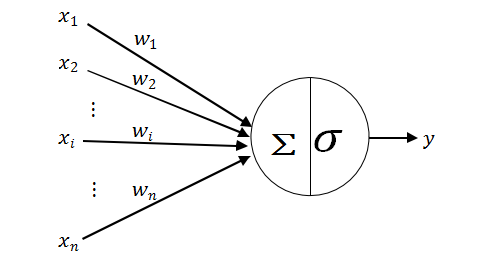
**2.3 算法原理及程序实现**

Logistic Regression和线性回归一样，是回归中常见的算法。其实用最小二乘法求解线性回归方程，就是我们最早接触到的回归。在二维平面上有很多的点，从这些点中选出一条直线，来很好的拟合这些点。通过求解，最后得到的回归方程形式是，然后来一个新的系数，通过这个系数，能够计算得到对应的输出的值。所以一种常见的回归就是通过一系列的点，计算得到一条合适的线。当有新的输入时，可以直接计算得到输出。简单来说，回归就是根据输入预测一个值。

单层的神经网络在应用sigmoid函数后，通过梯度下降法对系数进行求解，就是Logistic Regression函数，因此在本题中，使用神经网络进行模型建立。

**2.3.1 神经元模型**

神经网路的大致模型为神经元模型，其基本结构如图4所示



**图4 神经元模型**

该神经元的输入输出关系为：



其中为神经元的输入，为各神经元之间的权重，表示前一个神经元对后一个神经元的影响程度，为神经元的阈值（这里约定阈值写成与权值相似的形式，即把阈值看成样本输入为 1 的随机数），为神经元的输出，为神经元的激活函数，本文中所使用的激活函数为：

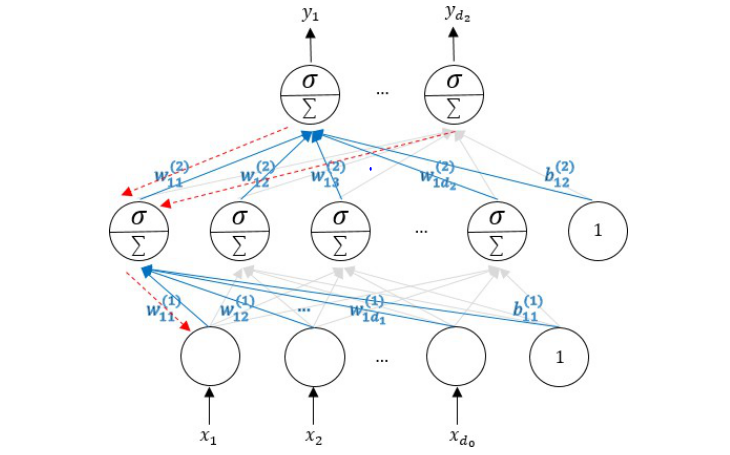


**2.3.2 BP算法**

BP 神经网络因其学习算法采用反向误差传播算法（BP 算法）而得名。BP 算法的学习过程由信息正向传播和误差反向传播两部分组成。如图5所示是神经网络的结构示意图，其中蓝色实线为正向传播，红色虚线为反向传播。对于次网络，符号约定如表1所示。

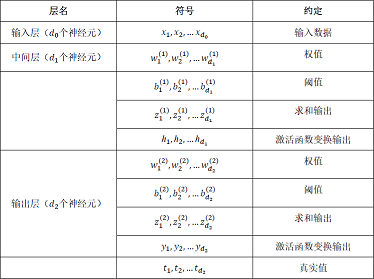
在正向传播过程中，计算各层神经元的状态。输入信息从输入层经各隐含层逐层处理，并传向输出层，每层神经元（节点）的状态只影响下一层节点的状态。如果输出层的状态与期望的输出不一致，则转入反向传播过程，将误差信号沿原来的连接通路返回，同时修改各层神经元的权值，权值的不断调整会使网络误差越来越小。

BP 算法的核心是数学中的“梯度下降法”，即 BP 网络的误差调整方向总是沿着误差下降最快的方向进行。



**图5 BP神经网络拓扑图**

**表1 神经网络符号约定**



在正向传播过程中，隐含层或输出层任一神经元将来自前一神经元传来的信息进行加权整合，通常还会在整合过的信息中添加一个阈值，这主要是模仿生物学中神经元必须达到一定的阈值才会触发的原理。再经激活函数变换后传到下一层神经元。

1. 从输入层到中间层

求和：

变换：

其中：

1. 从中间层到输出层

求和：

变换：

其中：

在反向传播过程中，构造关于输出值与真实值的目标函数，按照减少输出值与真实值之间误差的方向，依托梯度下降法，从输出层反向经过各中间层回到输入层，逐步修正各连接权值与阈值。

构造目标函数（目标函数应为多个样本的和，下面为简化的方式）：



1. 从输出层到中间层



其中，为便于书写，令

修正权值：



修正阈值：



其中

1. 从中间层到输入层

因为和每个都有关系，因此在求导时对每个都进行：



其中，为书写方便，令

修正权值：



修正阈值：



其中，

其python代码如下：

weight\_init\_std = 0.01

W1 = weight\_init\_std \* np.random.randn(hidden\_size, input\_size)

b1 = np.zeros([hidden\_size,1])

W2 = weight\_init\_std \* np.random.randn(output\_size, hidden\_size)

b2 = np.zeros([output\_size,1])

for i in range(iters\_num):

z1 = np.dot(W1,data\_train\_std) + b1

h = sigmoid(z1)

z2 = np.dot(W2, h) + b2

y = sigmoid(z2)

delta2 = result\_train - y

dW2 = np.dot(delta2,np.transpose(h))

db2 = np.dot(delta2,np.ones([train\_size,1]))

delta1 = np.dot(np.transpose(W2), delta2) \* h \* (1-h)

dW1 = np.dot(delta1,np.transpose(data\_train\_std))

db1 = np.dot(delta1,np.ones([train\_size,1]))

W2 += learning\_rate \* dW2

b2 += learning\_rate \* db2

W1 += learning\_rate \* dW1

b1 += learning\_rate \* db1

**2.4 算法结果评估**

Logistic regression用来实现0-1问题求解，通过训练的模型进行结果的预测。这里假设0-1值满足伯努利分布，即





可简化为：



由于上式不便计算，可对其取对数，得到：



最终我们的目标是求解最大似然估计函数，即：

Logistic regression的代价函数如下所示：



其中，

**三、实验结论**

通过将测试集数据经模型训练过的系数进行运算后，得出该模型针对其的预测结果，将其与提交集信息进行比对，得出预测结果，计算其准确率为60%（略有波动）。其python代码如下：

minmax\_data\_test = MinMaxScaler()

minmax\_test\_test = MinMaxScaler()

data\_test\_std = minmax\_data\_test.fit\_transform(data\_test)

result\_test\_std= minmax\_test\_test.fit\_transform(result\_test)

z1 = np.dot(W1,data\_test\_std.T) + b1

h = sigmoid(z1)

z2 = np.dot(W2, h) + b2

y\_test = z2

minmax\_data\_test = MinMaxScaler(feature\_range=(0, 1))

data\_test\_std = minmax\_data\_test.fit\_transform(intest).T

test\_z1 = np.dot(W1,data\_test\_std) + b1

test\_h = sigmoid(test\_z1)

test\_z2 = np.dot(W2, test\_h) + b2

test\_y = test\_z2

**四、小组分工**

程序：应桃

参数调试：应桃

报告撰写：应桃

1. **作业总结**

在本次的模式识别大作业中，由于之前没有python基础，使得代码的编写尤其艰难，于是参考了张贤益同学发在QQ群中的神经网络中的相关知识和代码。同时，自己对数据进行了相关的处理和研究，但最后所完成作业还是存在一定的问题。这也让自己知道了只知道理论基础，却不注重相关的代码编写在模式识别的学习道路上是远远不够的。

因此，在接下来的研究生学习生涯中，一定注意对代码编写能力的培养和提高，可以熟练掌握python，同时也保持对模式识别的兴趣，争取有所成绩。