

**模式识别大作业**

题 目 Criteo广告预测

学 院 信息科学与工程

专 业 控制科学与控制工程

组 员 孙浩 张欣茹 毛盈

刘静博 陈显锋

指导教师 赵海涛

**完成日期： 2018 年 12 月16日**

1 问题描述

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。  
 本题我们使用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。请训练出合适的模型，预测用户在不同的特征下是否会点击广告。

2 问题分析

对于一个典型的二分类问题，并且在训练数据中是含有标签的，因此是一个有监督学习任务。有监督学习任务的过程如下图所示：



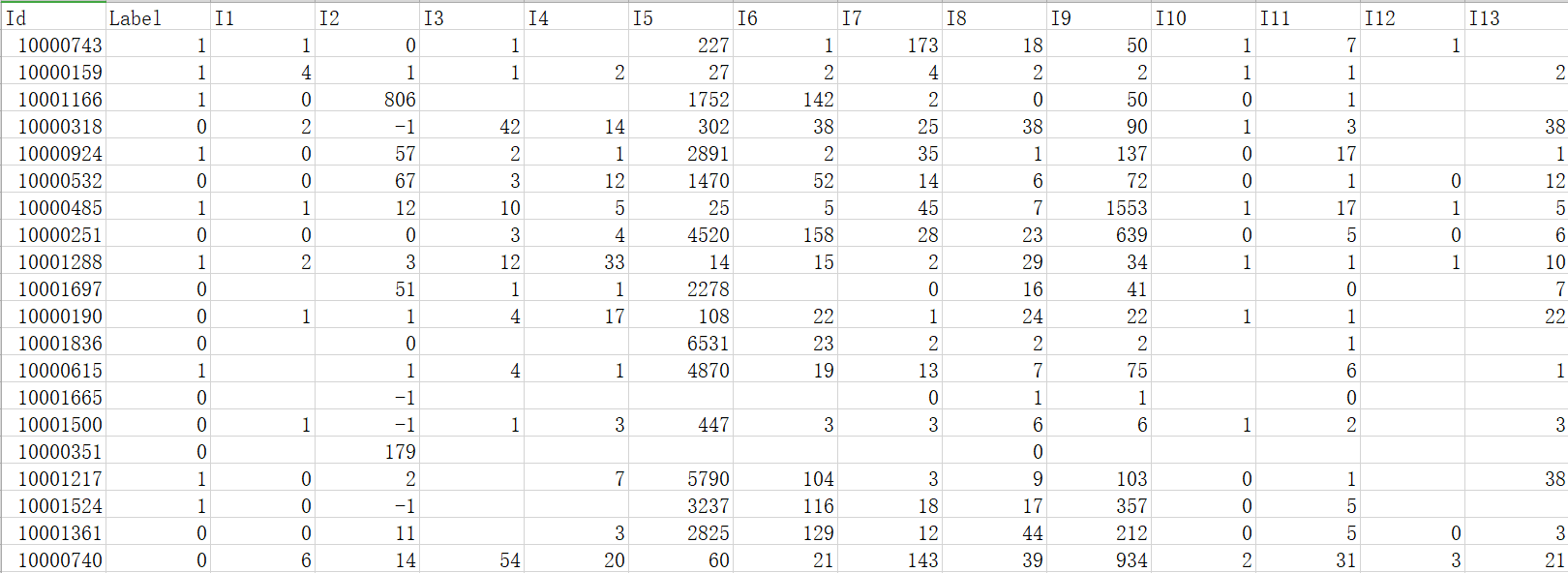
**图2.1 模型结构图**

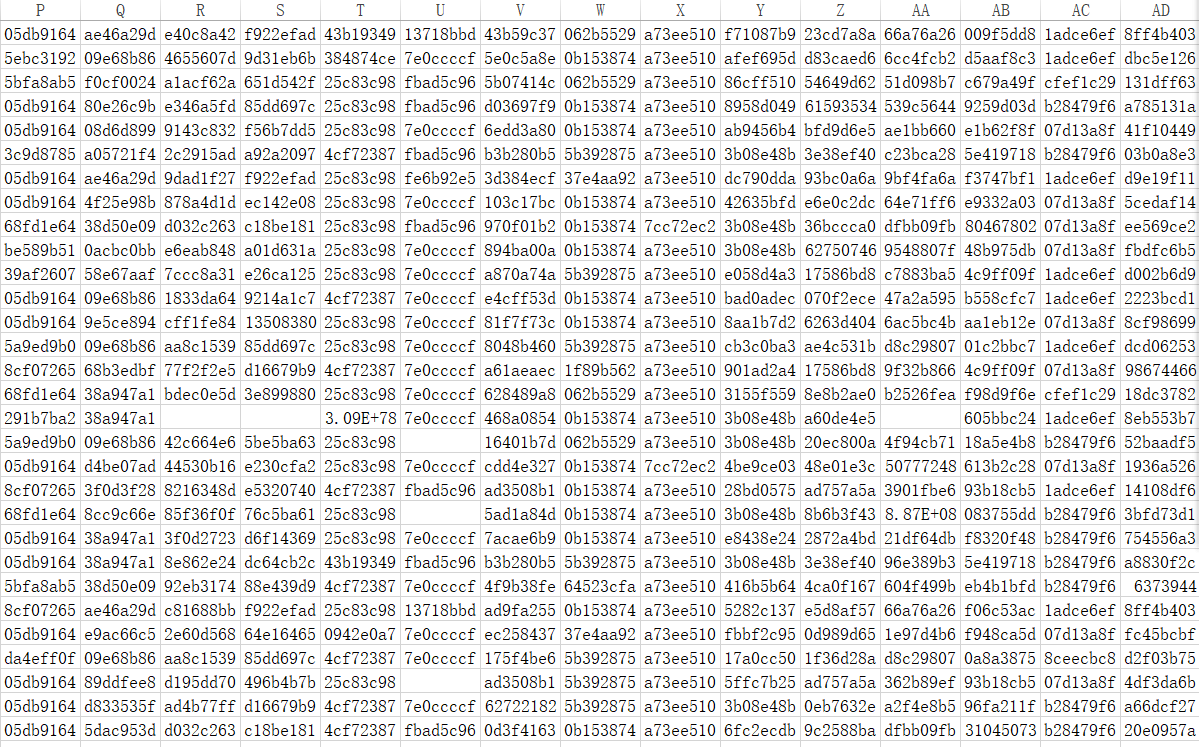
从上图中可以看出，有监督学习任务主要分为数据预处理，利用学习算法

训练模型，验证模型和预测新数据几个步骤。

3 数据预处理

3.1 数据分析





**图3.1 train.csv部分数据**

题目数据包含一个训练集（train.csv）、一个测试集（test.csv）以及预测结果(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。图2.1为原始数据对的部分截图，从图中可以很明确的看出，无论是计数特征还是类别特征都存在着缺失值，因此在选择训练数据的过程中需要进行取舍。

训练数据特征的选取我们采用了两种思路，第一种是仅选择前十三组的数值数据作为训练样本，因为前十三组的特征都为数字量，较为方便进行训练和处理；第二种思路是加入部分类别量进行训练，但是会涉及到非数字量的特征，因此需要将这些信息转化为0/1的量。

同时测试数据也存在着缺失的问题，所以需要对训练集和测试集都进行数据的补全，而由于各个特征所处的值域范围不同，为了确保这些不同的特征能够处在一个相近的范围内，考虑对各个特征进行特征缩放。

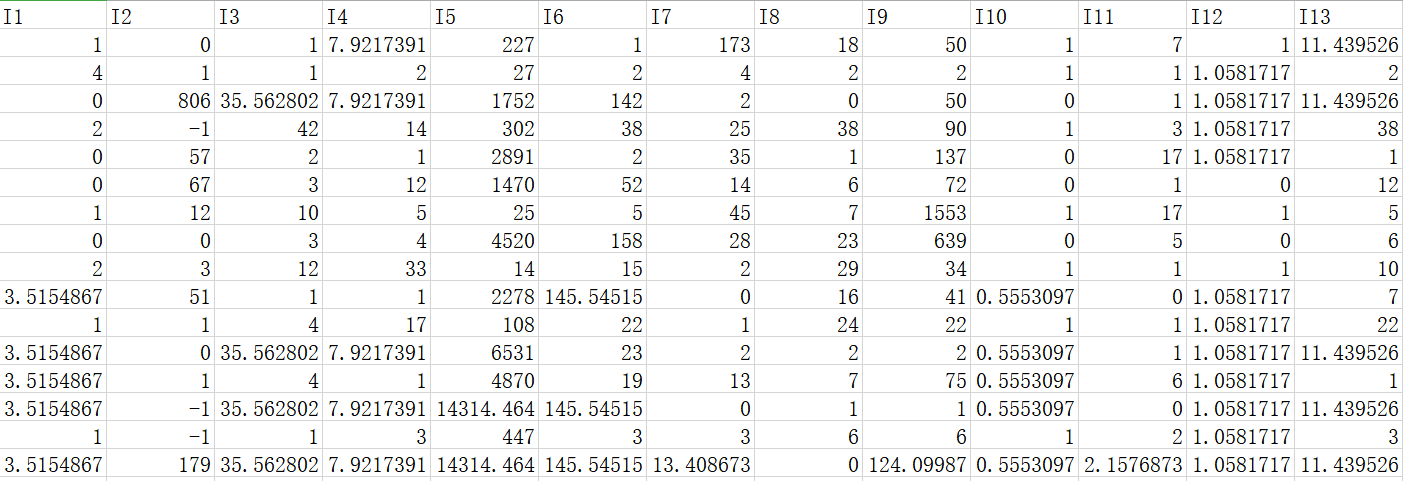
3.2 数据填充

### 3.2.1 平均值填充

平均值法填充数据相对简单，但是与真实数据存在的误差较大，但是操作起来较为简单。代码如下：

|  |
| --- |
| X\_train['I1'].fillna(X\_train['I1'].mean(),inplace = True)  X\_train['I2'].fillna(X\_train['I2'].mean(),inplace = True)  X\_train['I3'].fillna(X\_train['I3'].mean(),inplace = True)  X\_train['I4'].fillna(X\_train['I4'].mean(),inplace = True)  X\_train['I5'].fillna(X\_train['I5'].mean(),inplace = True)  X\_train['I6'].fillna(X\_train['I6'].mean(),inplace = True)  X\_train['I7'].fillna(X\_train['I7'].mean(),inplace = True)  X\_train['I8'].fillna(X\_train['I8'].mean(),inplace = True)  X\_train['I9'].fillna(X\_train['I9'].mean(),inplace = True)  X\_train['I10'].fillna(X\_train['I10'].mean(),inplace = True)  X\_train['I11'].fillna(X\_train['I11'].mean(),inplace = True)  X\_train['I12'].fillna(X\_train['I12'].mean(),inplace = True)  X\_train['I13'].fillna(X\_train['I13'].mean(),inplace = True) |

### 经过处理后的数据如图所示：



**图3.2 平均值填充数据处理结果**

### 3.2.2 随机森林填充

为了更加接近真实值，我们尝试采用随机森林的方法进行数据的填充。

其基本原理如下：

随机森林（Random Forest）顾名思义，是用随机的方式建立一个森林，森林里面有很多的决策树组成，随机森林的每一棵决策树之间是没有关联的。在得到森林之后，当有一个新的输入样本进入的时候，就让森林中的每一棵决策树分别进行判断，判断这个样本属于哪一类（对于分类算法），然后看看哪一类被选择最多，就预测这个样本为哪一类。它是一种统计学习理论，利用bootsrap重抽样方法从原始样本中抽取多个样本，对每个bootsrap样本进行决策树建模，然后组合多颗决策树的预测，通过投票得出最终预测结果。

### 随机森林分类（RFC）是由多种决策树分类模型组合而成的组合分类模型，且参数是独立同分布的随机变量。在给定自变量集合X下，每个决策树分类模型都有一票投票权来选择最优的分类结果。RFC的基本过程包括：利用bootstrap抽样从原始训练集抽取N个样本，且每个样本的样本容量与原始训练集一样；对N个样本分别建立N个决策树模型，得到N种分类结果；根据k种分类结果对每个记录进行投票表决决定其最终分类。

### RF通过构造不同的训练集增加分类模型间的差异，从而提高组合分类模型的外推预测能力。通过K轮训练，得到一个分类模型序列，再用它们构成一个多分类模型系统，该系统的最终分类结果采用简单多数投票法。最终的分类决策：

### 

### 其中，表示组合分类模型，是单个决策树分类模型，Y表示输出变量（或称目标变量），为示性函数。该公式说明了使用多数投票决策的方式来确定最终的分类。

从随机森林的原理可以看出，建立相互关联性小而强度高的随机森林，首先要解决的问题是：1：如何构建随机决策树？2：如何对每棵树进行有效地组合？

首先是随机样本的选取。随机森林是决策树的组合，用bagging的方法产生不同的训练集，也就是从原始训练集中利用bootstrap抽样生成新的训练集，对每个新的训练集利用随机特征选取方法生成决策树，且决策树在生长过程中不进行剪枝。用bagging方法生成训练集，原始训练集中接近37%的样本出现在bootstrap样本中，这些数据为袋外数据，使用这些数据来估计模型的性能称为OOB估计。因此随机森林本质上是Bagging的一个扩展变体，RF在以CART决策树为基学习器构建Bagging集成的基础上，进一步在决策树的训练过程中引入了随机属性选择。

其次随机特征选取。为了使得每棵树之间的关联尽可能地小，在构造树时要对它的特征进行恰当的选择，主要有两种方法：随机选取特征变量和随机寻去特征变量的线性组合。

最后随机决策树的构建：在构建随机决策树时，对采样后的数据使用完全分裂的方式建立决策树，这样决策树的某一个叶子节点要么是无法继续分裂的，要么里面的所有你样本都是指向同一个分类，具体构造过程为：

1：寻找一个最能区别样本的属性

2：根据这个属性创建一个树节点，然后根据这个属性来创建节点的子节点，每个子节点代表所选属性的一个唯一的取值。

3：重复步骤1、2，直到这样决策树的某一个叶子节点要么是无法继续分裂，要么里面的所有样本都指向同一个分类。

随机森林的训练过程可以总结如下：

1：给定训练集D，特征维数F。确定参数：随机森林中CART决策树的数量n，每棵树的深度d，每个节点使用到的特征数量k，终止条件：节点上最少样本数s，节点上最少的Gini Gain

2：对于第 i 棵树，从训练集D中有放回的抽取大小和D一样的训练集D(i)作为构造第 i 颗树的数据，并从根节点开始进行分支

3：如果当前节点上达到终止条件，则设置当前节点为叶子节点，如果是分类问题，该叶子节点的输出为当前节点样本集合中数量最多的那一类；如果是回归问题，输出为当前节点样本集各个样本值的平均值。然后继续训练其他节点。如果当前节点没有达到终止条件，则从F维特征中无放回的随机选取k维特征。选择k个特征中分类效果最好的一个特征

4：重复2，3步直到所有节点都训练过了或者被标记为叶子节点。

5：重复2，3，4步直到所有CART决策树都被训练过。

随机森林有很多的优点，在分类问题中有广泛地应用：在数据集上表现良好；在当前的很多数据集上，相对其他算法有着很大的优势；它能够处理很高维度（特征）的数据，并且不用做特征选择；在训练完后，它能够给出哪些特征比较重要；在创建随机森林的时候，对generlization error使用的是无偏估计；训练速度快；在训练过程中，能够检测到特征间的互相影响；容易做成并行化方法，实现比较简单。

但是通过仔细分析数据中的13列计数特征，我们发现I2列的数据是齐全的，I5,I6,I7,I8,I9,I11这5列数据缺失的较少，为了保证随机森林算法的实施，对这五列特征进行平均值法进行补全，而剩下的I1,I3,I4,I0,I12,I13这几列特征采用随机森林的方法进行特征值填充，同时每完成一次随机森林算法，下一次随机森林算法的数据都会增加一列，起到了逐渐完善数据的一个过程。算法流程图如下：



**图3.3 随机森林填充算法流程图**

具体程序如下：

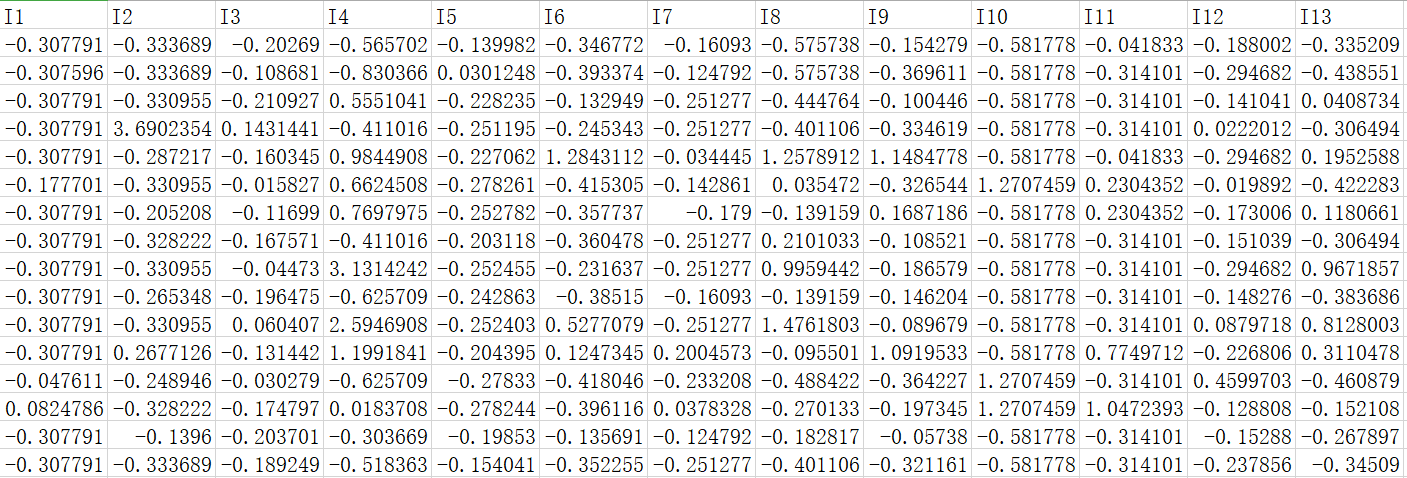
|  |
| --- |
| def set\_missing\_I1s(df): # 把已有的数值型特征取出来丢进Random Forest Regressor中  I1\_df = df[['I1','I2','I5','I6','I7','I8','I9','I11']]  known\_I1 = I1\_df[I1\_df.I1.notnull()].as\_matrix()  unknown\_I1 = I1\_df[I1\_df.I1.isnull()].as\_matrix()  # y即目标  y = known\_I1[:, 0]  # X即特征属性值  X = known\_I1[:, 1:]  # fit到RandomForestRegressor之中  rfr = RandomForestRegressor(random\_state=0, n\_estimators=2000, n\_jobs=-1)  rfr.fit(X, y)  # 用得到的模型进行未知结果预测  predictedI1s = rfr.predict(unknown\_I1[:, 1::])  # 用得到的预测结果填补原缺失数据  df.loc[ (df.I1.isnull()), 'I1' ] = predictedI1s  return df, rfr  X\_train ,rfr = set\_missing\_I1s(X\_train)  tmp\_df = X\_test[['I1','I2','I5','I6','I7','I8','I9','I11']]  null\_I1 = tmp\_df[X\_test.I1.isnull()].as\_matrix() # 根据特征属性X预测并补上  X = null\_I1[:, 1:]  predictedI1s = rfr.predict(X)  X\_test.loc[ (X\_test.I1.isnull()), 'I1' ] = predictedI1s |

3.3 特征缩放

如图2.3所示，对各个特征进行特征缩放。

|  |
| --- |
| [m,n]=size(P);  Pmean=mean(P);%均值  Pstd=std(P);%标准差  for i=1:m  P1(i,:)=(P(i,:)-Pmean)/(Pstd);%特征缩放  end |

经过预处理后的部分训练数据如图3.4所示。



**图3.4 预处理后的部分数据**

# 4 算法分析

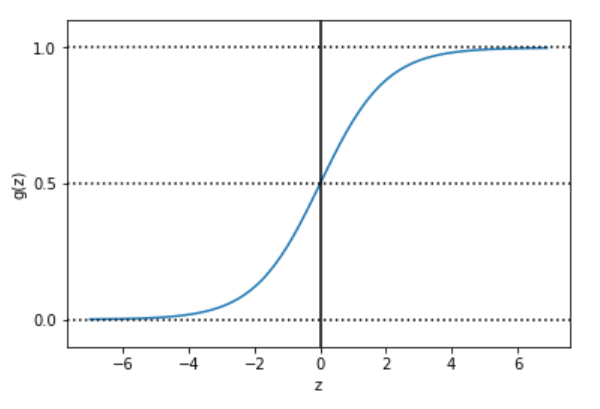
本次作业我们小组采用logistic回归算法和xgboost算法对广告点击率进行预测。

4.1 logistic算法

### 4.1.1 logistic原理

Logistic回归模型，虽然称为回归，但是却用来解决分类问题。从简单的二分类情况开始，可以扩展到多分类问题。类比于线性回归（基于吴恩达讲述的机器学习内容），我们希望logistic回归模型也有一个假设函数（Hypothesis），但是对于这个假设函数，我们希望它的取值范围为，（稍后作出解释），的形式如下：

只需要将z换掉，因此：



**图4.1 g(z)的图像**

从图2.5中可以看出，的取值范围是符合我们刚才的要求。因此，对于的输出值，我们将它类似的看成一种概率的估计，即输入为X，并且类别标签y=1时的概率。（对于二分类问题，只有两个类别，我们选择标签为0和1，0代表消极类别（negative class），1代表积极类别（positive class））。例如则表示在输入特征X（特征大多数是多维矩阵，所以此处采用大写X，在公式中为了与统一，采用小写x）的情况下，属于类别1的概率为70%。如果采用确切的概率形式表示：

含义是：给定X的情况下，参数的作用下，属于类别y=1的概率，其中x表示的是输入特征向量，为参数，y是类别标签。

对于分类问题来说，最重要的是构建出分类边界，也被称为决策边界。我们所构造的假设函数，通过观察g(z)>0.5时，z>0；g(z)<0.5时，z<0。可以很明显的看出g(z)=0.5是一个临界值，类似一个分界面，将这个映射到H函数上，因此构建如下策略：

即 此时判定y=1

即 此时判定y=0

从概率的角度来说，我们假设的是当前的目标属于类别1，因此当概率大于50%的情况下，判定属于类别1，当概率小于50%的情况下，判定为类别0。貌似在理论上行得通。

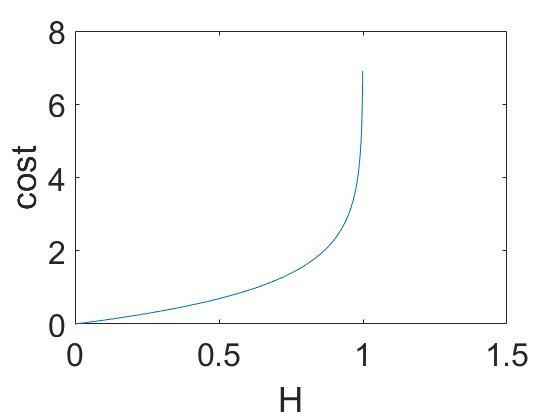
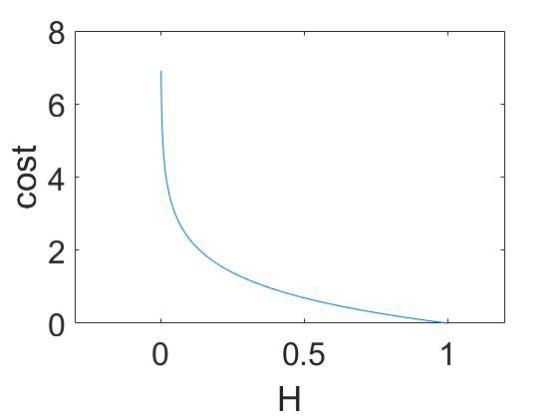
但是需要注意的是，我们所构建的这个决策边界实际上是假设函数的属性，决定的是分类面，跟数据集属性无关，与参数有关。

参数如何来获得，将是logistic回归模型中最重要的一个环节。模型的参数实际上是需要通过训练集的数据训练获得的。

假设有训练集，共有m组数据，每一组数据中包含特征x又是n维的列向量，，则：

想要确定参数的值，需要构建一个代价函数，优化的目标就是让代价函数最小，换句话说，在做决策的时候，希望决策者做的决策所付出的代价要尽可能小，对于线性回归来说，预测值和真实值之间的误差越小，决策者付出的代价也就越小，拟合的效果也就越好。因此对于线性回归来说，其代价函数如下：

但是对于logistic回归来说，是非线性函数，如果也采用上面线性回归的代价函数的话，会生成非凸形式的，采用梯度下降法不能收敛到全局最优，因此需要重新构建一个。构造如下的代价函数：



**图4.2 代价函数**

图2.6的图中描绘的分别是y=1的情况下（左侧的图），y=0的情况下的（右侧的图）代价函数的图形。横坐标H代表的就是，纵坐标cost代表的。

从左侧的图（对应的是y=1）中可以看出,在y=1时，同时，也就是说我们预测出的y=1的概率是100%（预测结果为y=1），此时代价函数为0；而如果我们在y=1时，，此时说明我们预测出的y=1的概率为0（也就是预测出结果为y=0），此时我们要付出的代价是无穷大的，说明错误的决策损失很严重。

同样的在右侧的图（对应的是y=0）中也可以看出，如果实际情况是y=0，而预测的结果，此时，预测结果和实际符合，代价为0；而如果预测出的结果为，此时预测结果和实际结果完全相反，则付出的代价是无穷大的。

通过以上的分析，构建的代价函数可以作为目标函数，作为优化的目标，将代价函数最小化，就可以求得较为理想的模型。为了方便后续的优化处理，将上面的代价函数写在一起：

其实上式是通过统计学中极大似然估计得出的一个公式，其实很容易理解，因为y的取值无非两种0或1，当y=1时，第二项中的1-y=0，就消去了第二项；当y=0时，自动消去了第一项。

由于训练集中有m组数据，因此构建优化目标函数：

上式的就是我们需要优化的目标函数，尽可能的让的小，从而得到的参数用于构建分类模型。

可以采用梯度下降法来将最小化：

需要注意的是，同时更新所有的。为方便运算可以采用向量化方法求解，赵老师在前两次课上详细讲述了向量化处理梯度下降法的公式。

在求出参数后，将参数带回到最初的，此时就完成了模型的训练，可以使用该模型对测试集数据进行预测。

### 4.1.2 logistic算法程序

根据上述原理进行logistic算法编程，正则化以及梯度迭代过程如图2.7所示。

|  |
| --- |
| %广告点击预测 逻辑回归 I12缺失数据过多不作为特征  A=[I1,I2,I3,I4,I5,I6,I7,I8,I9,I10,I11,I13];  [m,dim]=size(A);%特征维度  for i=1:m  A(i,dim+1)=1;  end  X=A(:,1:dim+1);%训练集数据  Y=Label;%训练集label  B=zeros(dim+1,1);%初始化参数矩阵  step=0;%迭代步数  Z=X\*B;  for j=1:m  H(j,:)=1/(1+exp(-Z(j,:)));%sigmiod函数  end  E(1,:)=(-1/m)\*(Y'\*log(H)+(1-Y')\*log(1-H));  J=X'\*(H-Y)/m;  a=0.05;%learning rate  lambda=10;%正则化系数  for i=1:10000  sum=0;%正则化项  Z=X\*B;%simoid自变量 m\*1维  for j=1:m  H(j,:)=1/(1+exp(-Z(j,:)));%sigmiod函数  end    for j=1:dim  sum=sum+B(j,:)\*B(j,:);  end  EC(i,:)=lambda\*sum/m;  E(i,:)=(-1/m)\*(Y'\*log(H)+(1-Y')\*log(1-H))+lambda\*sum/m;%Loss Function  J=X'\*(H-Y)/m+lambda\*B/m;%梯度  B=B-a\*J;%梯度迭代  end  disp('loss')  E(i)  figure(1);  plot(E);%绘制loss与迭代次数的关系图  figure(2);  plot(EC);%绘制正则化项与迭代次数的关系图 |

4.2 xgboost算法

4.2.1算法原理

在介绍XGboost的原理之前，首先介绍GBDT（梯度提升决策树），它是一种基于决策树(decision tree)实现的分类回归算法。xgboost算法比较复杂，针对传统GBDT算法进行改进，包括损失函数、正则化、切分点查找算法优化、稀疏感知算法、并行化算法设计等等。因此首先介绍传统GBDT算法。

GBDT模型全称Gradient Boosted Decision Trees，在1999年由Jerome Friedman提出，它是一个加性回归模型，通过 boosting 迭代构造的一组弱学习器，与逻辑回归相比有如下优势：不需要做特征的归一化、自动进行特征选择、模型可解释性较好，可以适应多种损失函数如：SquareLoss，LogLoss 等等。但GBDT模型是非线性模型，其相对线性模型的缺点也是显然的：boosting 是个串行的过程，不能并行化，计算复杂度较高，同时其不太适合高维稀疏特征，通常采用稠密的数值特征如点击率预估中的 COEC。

GBDT 模型的具体原理和推导如下：

分类的目标是寻找使得期望损失最小的决策函数 强分类器是一组弱学习器的加性组合，那么可以写成如下形式：



这里指代回归树是模型参数，这里指代每个节点的分裂特征（变量），最佳分割点，节点的预测值。M代表弱分类器的个数。

GBDT损失函数的数值优化可以看成是在函数空间，而不是在参数空间，因此考虑函数空间的优化问题。在函数空间上形式的使用梯度下降法求解，首先固定 x，对求解其最优解。将预测函数对应参数P，最优解变成了：



相当于在函数空间上作梯度下降。每一步梯度下降：



现在把这个思想代入到GBDT，上文已经得到预测函数为：



需要做的事情是得到预测函数的最优解，就是：



需要估计，这里采用决策树的实现去逼近函数，使得俩者之间的距离尽可能的近。距离的衡量方式有很多选择，比如均方误差。这里给出 LogLoss 损失函数下的具体推导过程， LogLoss 损失函数形式如下：



Step1：求解初始。令其偏导为 0；



Step2：估计 ，并用决策树对其进行拟合；



Step3：用 a single Newton-Raphson step 去近似求解下降方向步长，通常的实现中Step3 被省略，采用 shrinkage 的策略通过参数设置步长，避免过拟合。



xgboost 相比于GBDT改进在于：

1: 自定义损失函数，即对 loss function 做了二阶的泰勒展开，

2:规范化的正则项：叶子节点数目、叶子结点的分数。在目标函数之外加入了正则项整体求最优解，用以权衡目标函数的下降和模型的复杂程度，避免过拟合。

3：建树与剪枝：先建完全树再剪枝，包括：支持分裂点近似搜索、稀疏特征处理、缺失值处理

4：特征重要性与特征选择

5:并行计算

将目标函数做泰勒展开，并引入正则项：



除去常数项，求得每个样本的一阶导 g\_i 和二阶导 h\_i，将目标函数按叶子节点

规约分组，略去一些中间步骤:



在树结构是 fix 的时候，上式中叶子节点权重有闭式解，解和对应的目标函数值如下:



在目标函数是 LogLoss 损失函数下，这里给出一阶导和二阶导的推导：





### 4.2.2 xgboost算法程序

|  |
| --- |
| X\_train=X\_train1.join(X\_train2)  X\_test=X\_test1.join(X\_test2)  #DictVectorizer进行特征提取  dict\_vec = DictVectorizer(sparse=False)  X\_train = dict\_vec.fit\_transform(X\_train.to\_dict(orient='record'))  X\_test = dict\_vec.transform(X\_test.to\_dict(orient='record'))  #模型选择XGB  xgb\_model = xgb.XGBClassifier()    #设置参数  params = dict(booster='gbtree',  objective='multi:softmax',  num\_class=2,  learning\_rate=0.1,  max\_depth=2,  silent=0,)  # 设置迭代次数  plst = list(params.items())  num\_rounds = 2000    # sklearn.cross\_validation进行训练数据集划分，训练集和交叉验证集比例  train\_x, val\_X, train\_y, val\_y = train\_test\_split(X\_train, y\_train, test\_size=0.2, random\_state=1)    # xgb矩阵赋值  xgb\_val = xgb.DMatrix(val\_X, label=val\_y)  xgb\_train = xgb.DMatrix(train\_x, label=train\_y)  xgb\_test = xgb.DMatrix(X\_test)    #watchlist 方便查看运行情况  watchlist = [(xgb\_train, 'train'), (xgb\_val, 'val')]    # training model  # early\_stopping\_rounds 当设置的迭代次数较大时，early\_stopping\_rounds 可在一定的迭代次数内准确率没有提升就停止训练  model = xgb.train(plst, xgb\_train, num\_rounds, watchlist, early\_stopping\_rounds=100)    #测试集合预测值  preds = model.predict(xgb\_test, ntree\_limit=model.best\_ntree\_limit)  #结果输出  np.savetxt('/Users/Administrator/Desktop/CRITEO2.csv', np.c\_[range(1, len(X\_test) + 1), preds], delimiter=',', header='Label', comments='', fmt='%d') |

# 5 结果分析

将预测结果submission.csv上传至Lintcode。结果如下：

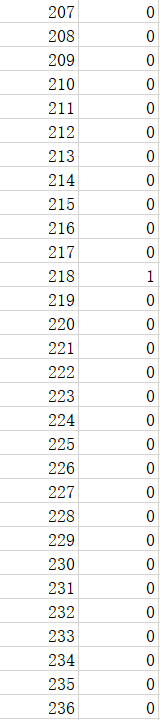
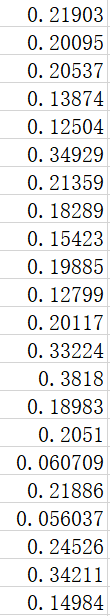


**图5.1 上传结果图**

图中前两个是用Logistic回归进行预测的结果，后面六个是XGboost算法预测的结果，比较可得，Logistic的分类结果优于XGboost算法，分析原因是Logistic的分类结果显示的是属于某一类的概率，即点击广告的概率，而XGboost算法是0、1分类器，所以产生结果不同。可以进行提高的地方是：加入数据中的离散型特征。我们在结果为7.16680的仿真中加入了这离散型文本特征，但是处理结果不佳。这一部分数据处理运用有待提高。

用Logistic回归进行预测第一个结果为0.46769，优于第二个结果0.46940。第一个中使用随机森林进行数据填充，第二个是用平均值进行填充，因为随机森林填充更加接近真实情况在数据集上表现良好；在当前的很多数据集上，随机森林相对其他算法有着很大的优势；它能够处理很高维度（特征）的数据，并且不用做特征选择，而且训练速度快；在训练过程中，能够检测到特征间的互相影响；容易做成并行化方法，实现比较简单。

XGboost算法及Logistic回归的预测输出结果如图5.2、5.3所示：

**图5.2 XGboost算法输出结果** **图5.3 Logistic回归输出结果**

**6 总结**

（1）孙浩 Y20180078

在完成大作业的过程中，我们小组的成员经过充分的讨论，确定了我们的解决策略，让我深刻的体会到了团队协作的力量。之前的那次个人完成的作业，只有自己完成，很多方面考虑的都不是很严谨，而在和组员的探讨中，发现了很多新的可以改进的方法和策略。团队协作，分工明确，较为顺利的完成了老师布置的大作业。经过这一段时间的模式识别课程的学习，为我以后的研究打下了基础，赵老师细心地讲解，让我在很多重要的理论和概念的理解上都有了突破，非常感谢赵老师在模式识别上的指导。

（2）毛盈 Y30180664

在考完试后我们小组成员就开始积极准备大作业，初步确定好要完成的题目（广告预测）后就开始工作起来，首先，我们确定使用的算法是logistic回归算法和xgboost算法，然后就进行了小组分工，两个人编写logistic算法程序运行结果并完成相应的算法原理，两个人xgboost，还有一个人负责报告的整理汇总。其中，我负责的是logistic回归算法部分。

一开始接触logistic回归觉得一头雾水，做起作业来也是束手无策，在经过几次的写笔记、大作业之后，现在对它已经是感觉亲切熟悉了。Logistic 回归以其高效、通俗易懂，不需要太大的计算量，很容易调整等优点，已经成为了一种被人们广泛使用的算法，而且它的输出也有很好好的预测概率。与线性回归一样，当去掉与输出变量无关的属性以及相似度高的属性时，logistic 回归效果会更好。因此特征处理在 Logistic 和线性回归的性能方面起着重要的作用。

在这次大作业中，我们遇到的一个比较大的问题是对数据的处理，我们一开始只选择了前面13个连续特征进行仿真，但是缺省的值很多，我们先用平均值替代法进行了填充，发现结果还可以，后来尝试用随机森林法进行迭代填充，想以此更接近真实值，验证出最后的结果确实更好了。之后我们又尝试加入后面的离散特征，发现结果并没有得到提高。

经过这次的团队作业，我深刻认识到了交流沟通的重要性，自己一个人想不到的东西往往在交流过程中可以有新的思路，别人的想法往往可以给自己启发；其次就是，在进行一个团队作业时，沟通交流是必须的，但是明确的分工也是必不可少的，只有每个人都清楚明白自己的任务和责任，一个团队作业才可以成功。

（3）张欣茹 Y30180697

这次模式识别大作业，我们组经过，最终选定题目：Criteo广告点击率预测。这个问题实质上是一个分类问题，在课上我们也了解学习到许多相关算法：逻辑回归、决策树等。经过多次尝试，我们最终用逻辑回归和XGboost进行广告点击率预测，同时对两种算法进行比较。然后进行小组分工，两个人编写logistic算法程序运行结果并完成相应的算法原理，两个人xgboost，还有一个人负责报告的整理汇总。其中，我负责的是部分logistic算法及报告整理汇总。

首先数据处理，我们用前面的13个数据特征进行分类，但是因为题目中给的数据有许多缺失，因此使用随机森林和平均值两种算法进行填充，随机森林算法因更加接近真实情况，效果更好。对接下来算法仿真，算法模型进行L1正则化，最后汇报总结。在这个过程中，对课上学习的知识进行应用，加深对知识点的理解，提高了实践应用能力；同时也了解到新的知识：GBDT、随机森林等的原理应用。编程一直以来都是我的短板，但是只要动起手来：已有程序上修改、模仿再到自己独立编写，不断积累，一定可以克服这一不足。

这次小组合作也让我深刻感觉到了团队合作以及团队意识的重要性，小组合作最重要的就是沟通交流，大家分工合作，有效交流可以大大提高工作的效率；而且合作中不同思想、火花的碰撞可以扩宽我们的思路。这个过程提高了我的团队合作能力以及责任意识，受益匪浅。

（4）刘静博 Y30180659

在完成这次大作业的过程中，通过小组间的讨论和自己认真查阅相关资料，使得我对logistic算法的数学推导和应用有了更加深层次的理解，对书本上的知识有了更加全面的认识。在算法实现的过程中，我自身也暴露出很多问题，如代码运行出现错误，代码运行结果和想法不一致等，但是在组员的协同努力下，成功解决了问题并最终实现了不错的预测率，在查错和改错的过程中也提高了自己算法实现的能力。在这里首先感谢我可爱的队友，是他们陪伴我完成这项工作并给予我可行性的建议，感谢赵老师在课堂上给与我们对基本算法知识的讲解，使得我们有了前行的方向与目标。在未来的日子里，我会对在这次完成作业的过程中暴露出的不足和问题，有针对性的进行弥补和训练，不断提高自己，从而更好地服务大家。

1. 陈显锋 Y30180636

这次的大作业增加了我对模式识别的兴趣，学习好这门课程即需要弄懂每种算法的推导过程，还要通过编程实现算法。通过这次作业使我对逻辑回归有了进一步的学习与了解，以及对编程有了进一步的学习与兴趣，由于编程能力的缺乏，导致我实现的效果可能不足。但是在小组成员的合作下我们还是完成了这个大作业。在这个过程中，我学习到了许多算法以及编程的知识，也认识到了自身的不足，在以后的学习中还有许多需要改进的地方。感谢赵老师的帮助，并且在作业过程中给了我们大家很多指导。