



一轮考核论文

# 二○二三年四月

**目 录**

**线性回归分析··············· 3**

**softmax回归分析············· 8**

**参考资料················· 17**

1. **线性回归分析**

**1.1 建模流程**

代价函数的实现

导入数据

判断相关性

梯度下降的实现并打印loss值

导入测试集，得到预测值

绘制测试值和实际值的对比图

**图1-1** 模型建立流程图

**1.2 各步骤实现与问题分析**

1.导入数据：

（1）代码实现：

#读取数据集

pf = pd.read\_csv(r"C:\Users\86189\Desktop\train\_.csv")

#处理空值

pf.dropna(inplace=True)

pff = pf.drop(['outcome','id'],axis = 1)

#获取数据集长度

m = len(pf) #m=342

（2）参数含义注释：

pf：读取到的训练集

pff：除去特征向量以外的其余列后所得的数据矩阵

m：数据集的长度

1. 判断相关性：

（1）代码实现：

#判断相关性

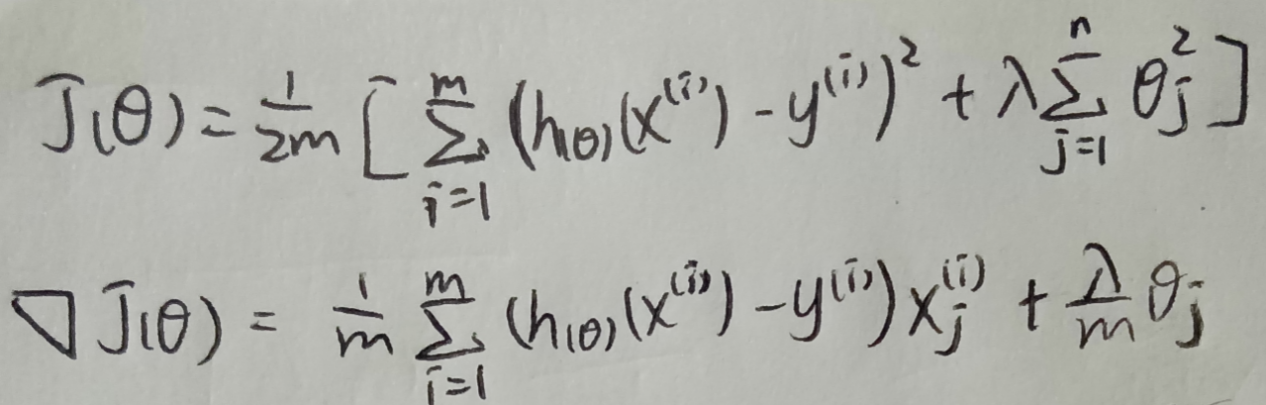
#for i in range(0,13):

# print(pf[[str(i),'outcome']].corr())

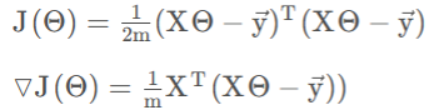
1. 代价函数的实现：

（1）原理：

1）原始形式：



2）向量化形式：



（2）代码实现：

#定义代价函数及其梯度

def J(x,theta,y,m):

return np.dot((np.dot(x,theta) - y).T,np.dot(x,theta) - y)/(2\*m)

def grad\_J(x,theta,y,m):

return np.dot(x.T,np.dot(x,theta) - y)/m

（3）参数含义注释：

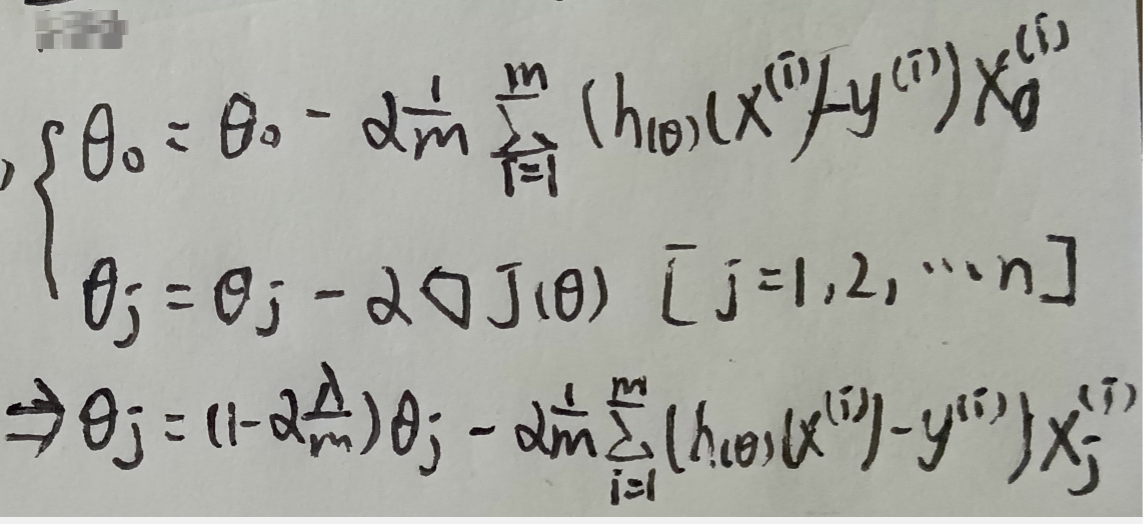
x: 特征向量矩阵

y: 标签向量

m: 数据的长度

theta: 用于与特征向量匹配，预测标签向量的参数向量

1. 梯度下降的实现并打印loss值：
2. 普通方法：
3. 原理：



2）代码实现：

#向量化x

x0 = np.ones(m).reshape(m,1)

xj = np.array(pff).reshape(m,13)

x = np.hstack((x0,xj))

#生成θ向量，以下θ为迭代一定次数后的，如此生成θ来节省时间

theta = np.array([[ 1.37951168e+00]

,[-1.20565819e-01]

,[ 6.77314704e-02]

,[-6.01090893e-02]

,[ 2.49781636e+00]

,[ 1.20147163e-01]

,[ 5.19517749e+00]

,[-4.49108833e-03]

,[-1.13399968e+00]

,[ 2.56038839e-01]

,[-1.07051830e-02]

,[-2.64260464e-01]

,[ 1.64098605e-02]

,[-4.65252149e-01]]

).reshape(14,1)

#生成y向量

y = np.array(pf['outcome']).reshape(m,1)

#定义学习率α与迭代次数step

alpha = 0.000006373829

step = 600000

for i in range(step):

theta = theta - np.multiply(alpha,grad\_J(x,theta,y,m))

print(J(x,theta,y,m))

3）参数含义注释：

alaph: 学习率

step: 迭代次数

xj：除去特征向量以外的其余列后所得的数据矩阵

x0：作为theta0的参数

x: 特征向量矩阵

y: 标签向量

m: 数据长度

J：代价函数

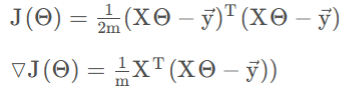
grad\_J：梯度函数

4）问题分析：

Q：预测值打印显示nan

A：学习率太大，导致下降时越过了极值点，从而产生振荡，以至于预测值越来 越大

1. 正规方程法：
2. 原理：



2）代码实现：

#生成正则化矩阵

E = np.eye(14)[0]

E[0] = 0

#生成正则化系数lamda

lamda = 0.02

#开始正则化标准方程计算

theta = np.dot(np.linalg.inv(np.dot(x.T,x) + lamda \* E),np.dot(x.T,y))

3）参数含义注释：

E：除了第一个元素是零，主对角线上全是1的对角矩阵

lamda：正则化系数

x：特征向量矩阵

y：标签向量

theta：用于与特征向量匹配，预测标签向量的参数向量

1. 绘制测试值和实际值的对比图：

（1）代码实现：

#plt.subplot(121)

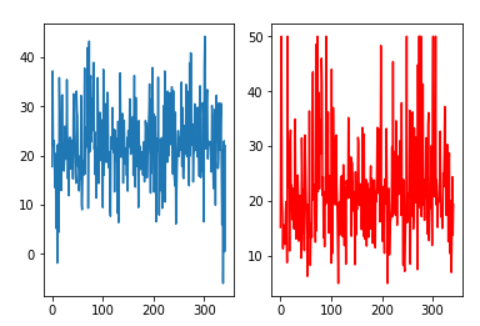
#plt.plot(np.dot(x,theta))

#plt.subplot(122)

#plt.plot(y,’r’)

#plt.show()

（2）对比图呈现：



注：左侧蓝色图像为预测值，右侧红色图像为真实值

1. 导入测试集，得到预测值：

（1）代码实现：

#开始测试

pp = pd.read\_csv(r"C:\Users\86189\Desktop\test\_.csv")

pp.fillna(0,inplace=True)

mp = len(pp)

pp=pp.drop('id',axis=1)

#向量化x

xp0 = np.ones(mp).reshape(mp,1)

xpj = np.array(pp).reshape(mp,13)

xp = np.hstack((xp0,xpj))

outcome = np.dot(xp,theta)

ps=pd.read\_csv(r"C:\Users\86189\Desktop\sample\_submission.csv")

ps['outcome'] = outcome

ps.to\_csv(r"C:\Users\86189\Desktop\my\_submission.csv")

（2）参数含义注释：

pp：读取到的测试集

mp：数据集的长度

xpj：除去特征向量以外的其余列后所得的数据矩阵

xp0：作为theta0的参数

xp: 特征向量矩阵

outcome：得到的预测向量

ps：读取到的提交模板文档

1. **softmax回归分析**
   1. **建模流程**

导入数据

原理引入

示性函数及概率函数的实现

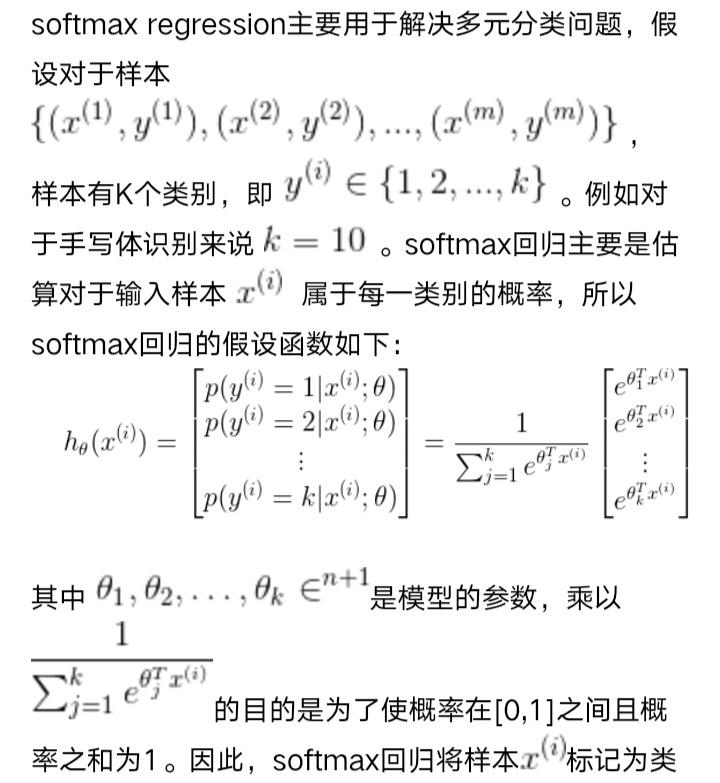
梯度函数的实现

根据所得预测概率映射出对应类别

导入测试集，得到预测值

**1.2 各步骤实现与问题分析**

1. 原理引入：



1. 导入数据：
2. 代码实现：

#读取数据集

pf = pd.read\_csv(r"C:\Users\86189\Desktop\train.csv")

#处理空值

pf.dropna(inplace=True)

#获取数据集的长度

m = len(pf)

#分割数据集，获得矩阵x的前身

pp = pf.drop('label',axis = 1)

#向量化操作

xj = np.array(pp).reshape(m,784)

x0 = np.ones(m).reshape(m,1)

x = np.hstack((x0,xj))

theta = np.zeros(7850).reshape(785,10)

y = np.array(pf['label'])

#k为类别个数

k = np.linspace(0,9,10)

1. 参数含义注释：

pf：读取到的训练集

m：数据长度

pp：除去label一列后得到的数据集，也是矩阵x的前身

xj：向量化的pp

x0：作为theta0的参数

x：特征向量矩阵

theta：用于与特征向量匹配，预测标签向量的参数向量

y：标签向量

k：类别向量

1. 示性函数及概率函数的实现：
2. 示性函数：
3. 含义：数学表达式为1·（），当括号中的式子为真时返回1，为假时返回0
4. 代码实现：

#示性函数

if\_one = []

for i in range(m):

for j in range(0,10):

if y[i] == k[j]:

if\_one.append(1)

else:

if\_one.append(0)

if\_one = np.array(if\_one).reshape(m,10)

1. 参数含义注释：

if\_one：存放示性函数返回值的示性向量

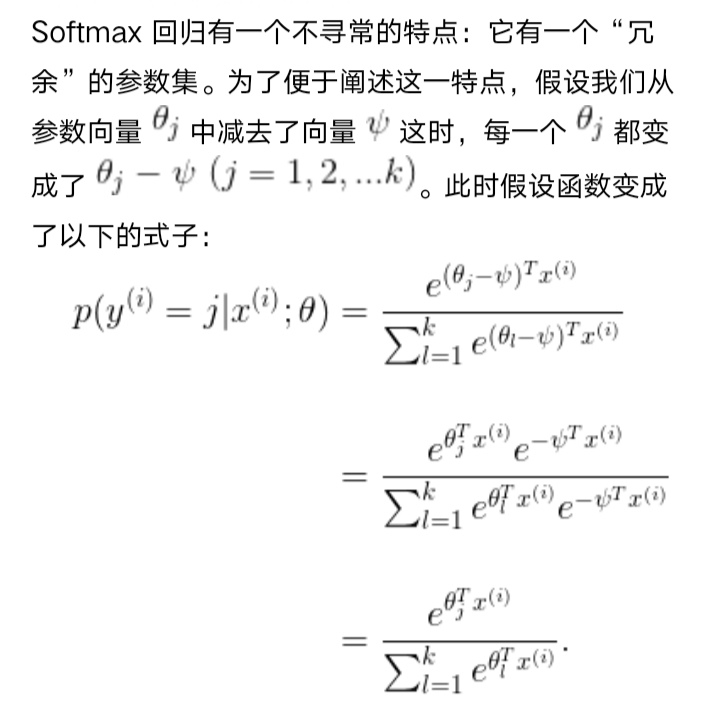
y：标签向量

k:类别向量

m：数据长度

1. 概率函数：

1）定义：



1. 代码实现：

#概率函数

def h(theta,x):

sum1 = np.sum(np.exp(np.dot(x,theta)),axis=1)

#这里需要把sum1改变一下形状，从（2000，）换成（2000，1），不然无法利用矩阵广播的性质进行相除

sum1 = sum1.reshape(sum1.shape[0],1)

return np.exp(np.dot(x,theta))/ (sum1)

1. 参数含义注释：

x：特征向量矩阵

theta：用于与特征向量匹配，预测标签向量的参数向量

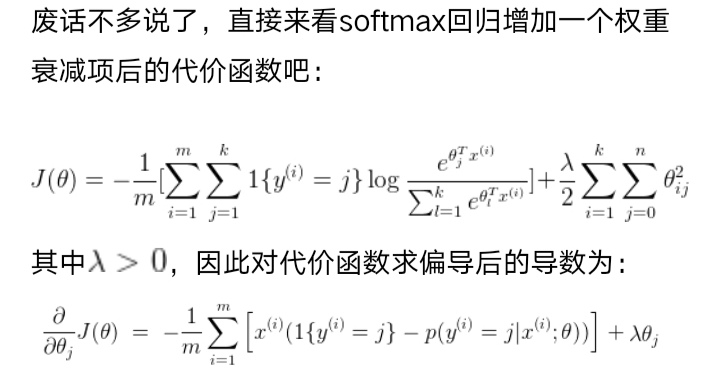
1. 问题分析：

Q：在运行概率函数时显示（2000，）与（2000，1）不能进行操作

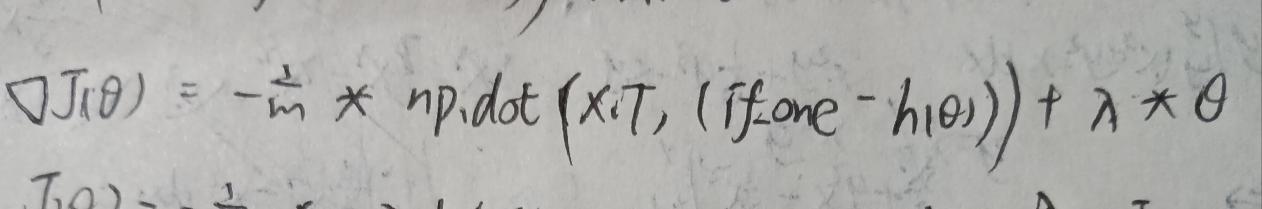
A：因为利用np.sum()所得的和是一个一维矩阵，无法与二维矩阵进行矩阵 广播操作，所以只需要改变sum1的形状即可

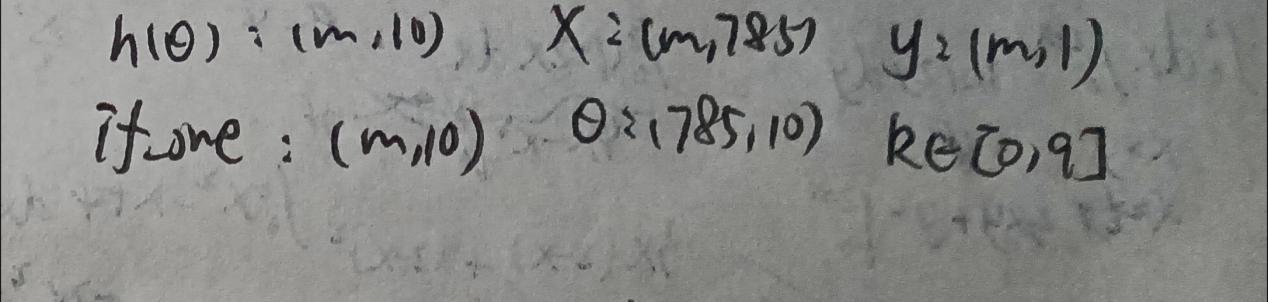
1. 梯度函数的实现：
2. 原理：

1）普通形式的梯度函数：



2）向量化的梯度函数及本实验中所用到的向量大小：





1. 代码实现：

#梯度函数

def grad\_J(lamda,theta,x,m,y,k):

return (-1/m) \* np.dot(x.T,(if\_one - h(theta,x))) + lamda \* theta

1. 参数含义注释：

grad\_J：梯度函数

lamda：正则化系数

theta：用于与特征向量匹配，预测标签向量的参数向量

x：特征向量矩阵

m：数据长度

y：标签向量

k：类别向量

if\_one：示性函数

1. 问题分析：

Q：在预测时运行速度极其缓慢

A：梯度函数的表达式出现错误，同时把示性函数写进了循环中，导 致每 一次循环都要重新计算一遍，解决方法是实现计算示性函数 并将其存储于 向量中

1. 根据所得预测概率映射出对应类别：
2. 代码实现：

#得出概率矩阵

h\_theta = h(theta,x)

#映射出label值

def map\_label(h\_theta,k,m):

label=[]

max\_probability = np.array(np.argmax(h\_theta,axis=1))

for i in range(0,len(max\_probability)):

label.append(k[max\_probability[i]])

label = np.array(label).reshape(m,1)

return label

1. 参数含义注释：

label：预测的类别向量

max\_probility：概率矩阵中每一行最大值所在的列索引所构成的向量

h\_theta：概率矩阵

k：类别向量

m：数据长度

1. 导入测试集，得到预测值：
2. 代码实现：

#开始测试

pr = pd.read\_csv(r"C:\Users\86189\Desktop\test\_no\_answer.csv")

pr.dropna(inplace=True)

mp = len(pr)

pr = pr.drop('sign',axis=1)

xrj = np.array(pr).reshape(mp,784)

xr0 = np.ones(mp).reshape(mp,1)

xr = np.hstack((xr0,xrj))

h\_theta\_prediction = h(theta,xr)

label\_prediction = map\_label(h\_theta\_prediction,k,mp)

ps = pd.read\_csv(r"C:\Users\86189\Desktop\sample\_submission.CSV")

ps['label'] = label\_prediction

ps.to\_csv(r"C:\Users\86189\Desktop\my\_submission.csv")

1. 参数含义注释：

pr：读取到的训练集

mp：数据长度

xrj：向量化的pp

xr0：作为theta0的参数

xr：特征向量矩阵

theta：用于与特征向量匹配，预测标签向量的参数向量

h\_theta\_prediction：概率矩阵

label\_prediction：预测向量

k：类别向量

1. **参考资料**

bilibili: 吴恩达——《机器学习系列课程》

CSDN： 天泽28——《softmax回归（Softmax Regression）》