

**机器学习作业**

**题 目: 三种方法求解Logistic回归**

**学 号: 919106840333**

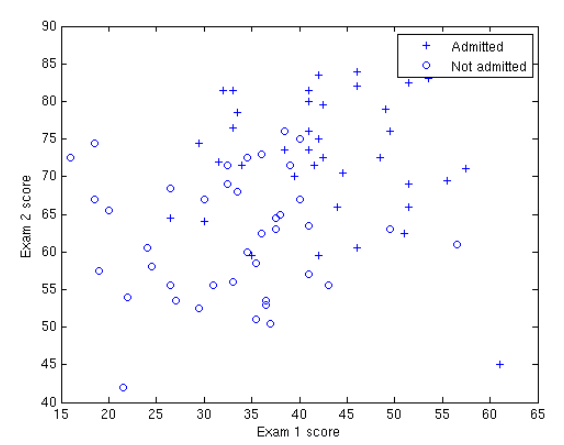
**姓 名: 孙傲歆**

**指导教师: 潘金山**

**2021年11月**

**一、题目要求**

根据下面的训练数据:



– 实现 1) 梯度下降; 2) 随机梯度下降; 3) Logistic回归的牛顿法, 设置初始参

数𝜃 = 0.

– 决定迭代的次数,计算每次迭代并绘制结果

**二、算法思路**

**1.梯度下降法求解logistics回归:**

首先我们根据题目所给数据，可以确定目标函数形式为。且y只在1和0之间取值。我们先将x数据进行标准化处理，公式为：(X-X\_mean)/X\_std。我们取𝜃向量为[0;0;0]（即全0列向量）。并设置学习率a=0.1，按照梯度下降法的公式进行迭代。当两次迭代结果之差小于0.001，我们认为结果收敛，停止迭代，并且在图中画出所有数据点，并将结果以直线形式显示出来，直线以上表示一类数据，直线以下表示另一类数据。同时我们对每次迭代计算结果进行记录，绘制每次迭代计算结果的折线图，观察其变化趋势。

**2.随机梯度下降法求解logistics回归:**

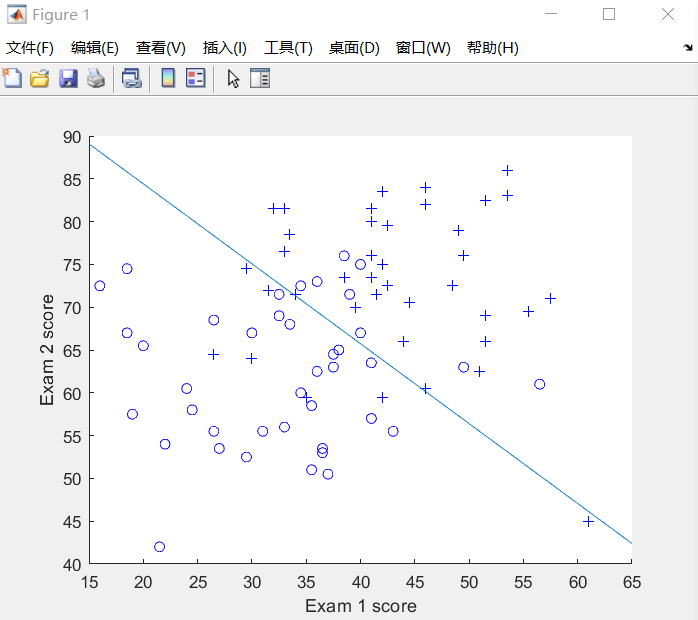
思路与梯度下降法类似，只是在每次迭代时，并不是选择全部样本进行计算，而是随机选择其中一小部分进行计算。这里我们在每次迭代时随机取1-80之间的5个数。其余部分与普通的梯度下降法几乎没有区别。同样的，最后我们将计算结果以直线形式显示出来，直线以上表示一类数据，直线以下表示另一类数据。同时我们对每次迭代计算结果进行记录，绘制每次迭代计算结果的折线图，观察其变化趋势。

**3.牛顿法求解logistics回归:**

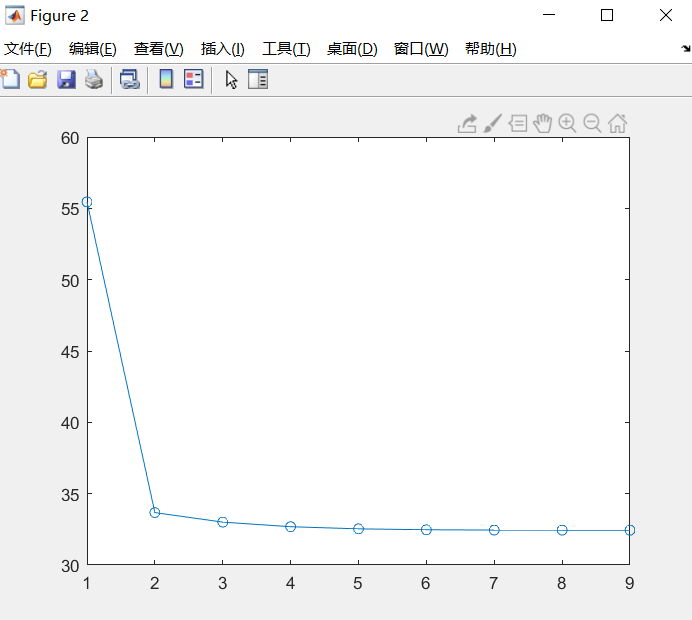
牛顿法虽然思路上较为简便，但实际代码实现上有一定难度。为了方便计算，我们编写get\_loss函数来计算每次迭代的目标函数结果。主要是Hessian矩阵的计算比较麻烦。由于牛顿法是二阶收敛，所以其结果收敛速度十分快，这里我们选择设置两次迭代结果只差小于0.000001，认为结果收敛。数据处理和图形绘制方面的思路与梯度下降法一致，在此便不再赘述。

**三、运行结果**

**1.梯度下降法求解logistics回归:**



**图1.1**

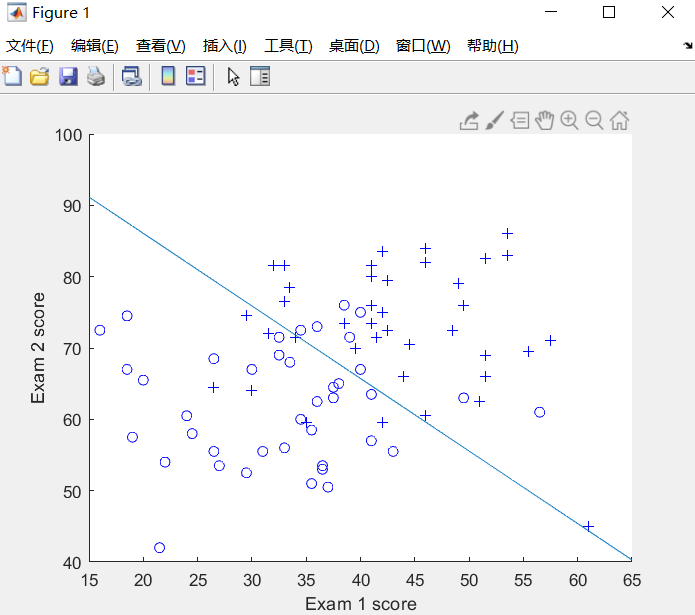


**图1.2**

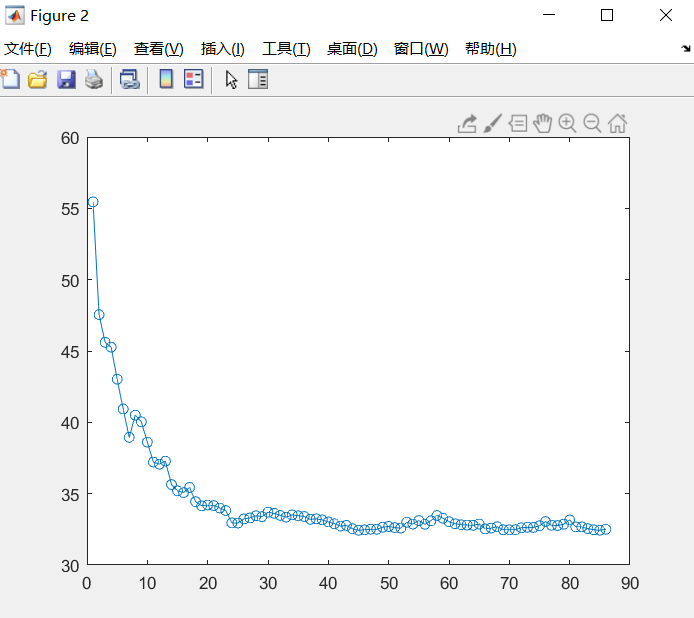
上图为代码运行结果。**图1.1**是数据显示，我们将所有点的信息绘制在图上并分类显示，o表示y值为1，+表示y值为0。直线是计算结果的显示，直线之上表示y=1的一类，直线之下表示y=0的一类。最终我们得到的函数为：。**图1.2**是每次迭代结果的折线图（不包含初始值），从图中可以看出，仅通过9次迭代，结果便达到了收敛。

**2.随机梯度下降法求解logistics回归:**

由于随机梯度下降法是随机取样本，所以每次代码运行的结果都不一样，这里我们记录两次运行结果。

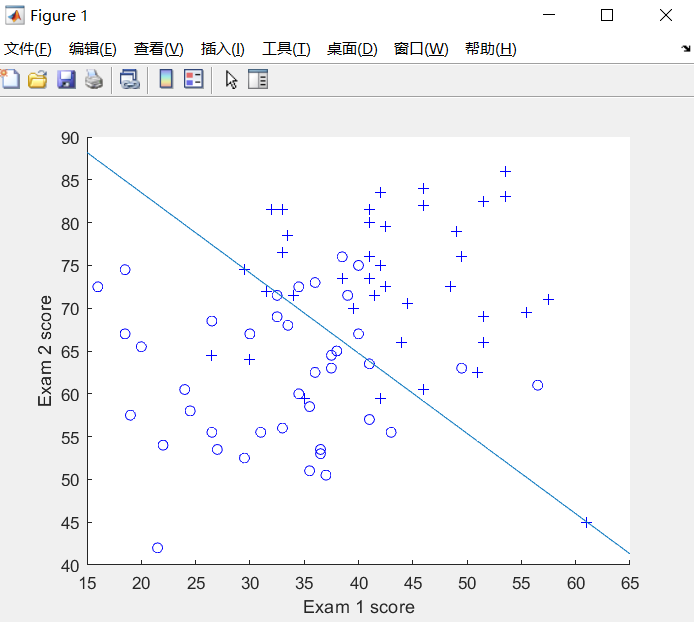


**图2.1**

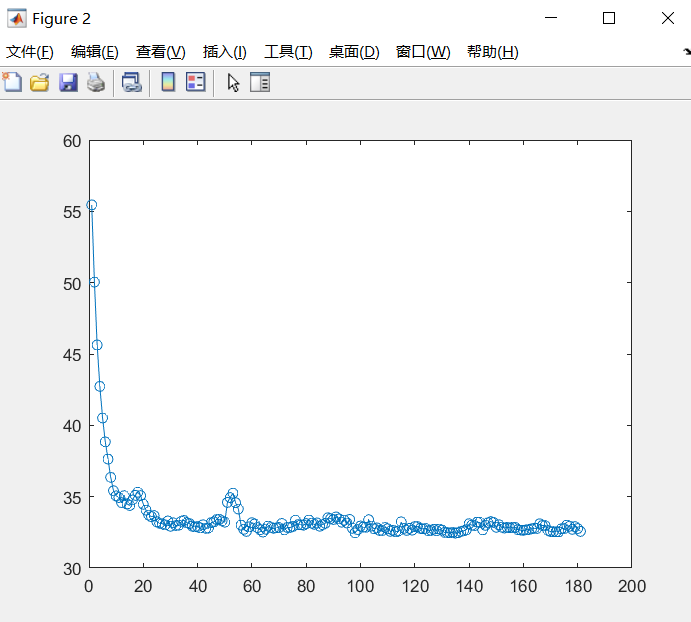


**图2.2**

首先是**图2.1、图2.2，**是第一次代码运行结果。**图2.1**同样是数据显示结果，在此便不再解释，最终表达式为：。**图2.2**是每次迭代结果的折线图（不包含初始值），由于是随机取样本，所以迭代次数相较于普通的梯度下降法较多，进行了86次迭代。且迭代过程中的结果忽高忽低，但是整体还是下降趋势。



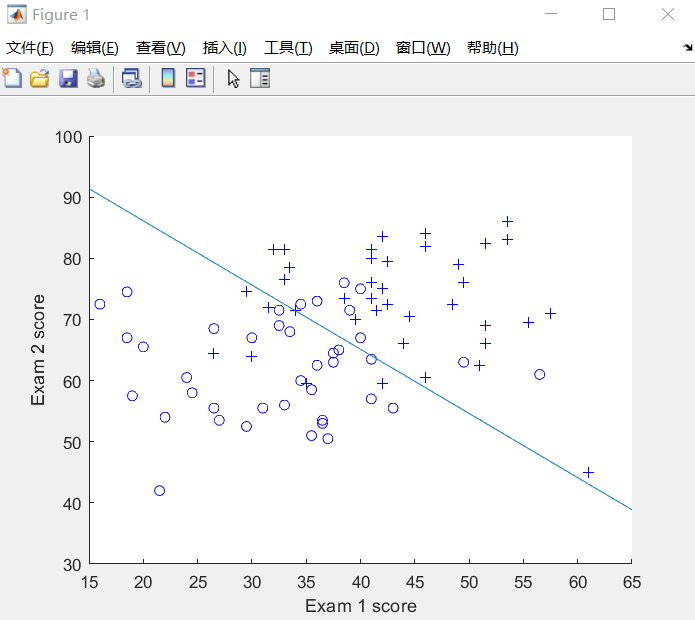
**图2.3**



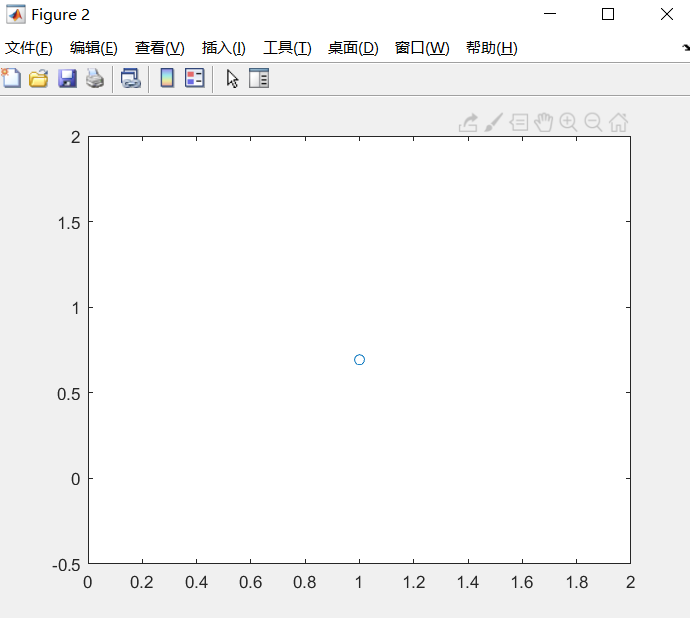
**图2.4**

其次是**图2.3**、**图2.4**，是第二次代码运行结果。最终结果为：。从**图2.4**我们可以看出，此次运行代码的迭代次数更大了，足足有181次。所以随机梯度下降法可能在迭代次数上不太稳定，这可能与随机到的数据样本的质量有关。

**3.牛顿法求解logistics回归:**



**图3.1**



**图3.2**

上图为代码运行结果，**图3.1**结果数据显示，不再加以解释，最终结果为：。可见在结果上牛顿法与梯度下降法还是有一定差异的。而从**图3.2**可以看出，牛顿法仅仅通过一次迭代就收敛了，所以牛顿法相较于梯度下降法其收敛速度十分之快。

**四、可运行代码**

**1.sigmoid函数:**

function y = sigmoid(x)

y=1/(exp(-x)+1);

end

**2.get\_loss函数：**

function y = get\_loss(data\_y,h)

loss1 = data\_y'\*log(h);

loss2 = (1-data\_y)'\*log((1-h));

y=(1.0/80)\*(-loss1-loss2);

end

**3.梯度下降法求解logistics回归:**

clc;

clear;

x=load("ex4Data/ex4x.dat");

y=load("ex4Data/ex4y.dat");

[m,n]=size(y);

figure(1);

xlabel('Exam 1 score');

ylabel('Exam 2 score');

for i=1:m

hold on;

if(y(i)==1)

plot(x(i,1),x(i,2),'b+');

else

plot(x(i,1),x(i,2),'bo');

end

end

mean=mean(x);

std=std(x);

for i=1:m

for j=1:2

x(i,j)=(x(i,j)-mean(j))/std(j);

end

end

temp(1:80,1)=1;

x=[x,temp];

a = 0.1;

theta =[0;0;0];

loss = 0;

old\_loss = 0;

temp\_loss=[];

for i=1:m

if (y(i) == 1)

loss=loss+log(sigmoid(x(i,:)\*theta));

else

loss=loss+log(1-sigmoid(x(i,:)\*theta));

end

end

while abs(loss-old\_loss) > 0.001

temp =x\*theta;

dew = [0,0,0];

for i=1:m

dew=dew+(y(i)-sigmoid(temp(i)))\*x(i,:);

end

theta = theta+a\*dew';

old\_loss = loss;

loss = 0;

for i=1:m

if (y(i) == 1)

loss=loss+log(sigmoid(x(i,:)\*theta));

else

loss=loss+log(1-sigmoid(x(i,:)\*theta));

end

end

disp(-old\_loss);

temp\_loss=[temp\_loss,-old\_loss];

end

plot\_x2 =zeros(1,51);

plot\_x1 =15:1:65;

for i=15:65

plot\_x2(i-14) = -(theta(3)+theta(1)\*((i-mean(1))/std(1)))/theta(2);

plot\_x2(i-14) = plot\_x2(i-14)\*std(2)+mean(2);

end

hold on;

plot(plot\_x1,plot\_x2,'-');

figure(2);

plot(temp\_loss,'-o');

**4.随机梯度下降法求解logistics回归:**

clc;

clear;

x=load("ex4Data/ex4x.dat");

y=load("ex4Data/ex4y.dat");

[m,n]=size(y);

figure(1);

xlabel('Exam 1 score');

ylabel('Exam 2 score');

for i=1:m

hold on;

if(y(i)==1)

plot(x(i,1),x(i,2),'b+');

else

plot(x(i,1),x(i,2),'bo');

end

end

mean=mean(x);

std=std(x);

for i=1:m

for z=1:2

x(i,z)=(x(i,z)-mean(z))/std(z);

end

end

temp(1:80,1)=1;

x=[x,temp];

a = 0.1;

theta =[0;0;0];

loss = 0;

old\_loss = 0;

temp\_loss=[];

for i=1:m

if (y(i) == 1)

loss=loss+log(sigmoid(x(i,:)\*theta));

else

loss=loss+log(1-sigmoid(x(i,:)\*theta));

end

end

while abs(loss-old\_loss) > 0.001

temp =x\*theta;

dew = [0,0,0];

z = round(rand(1,5)\*79)+1;

for i=1:5

dew=dew+(y(z(i))-sigmoid(temp(z(i))))\*x(z(i),:);

end

theta = theta+a\*dew';

old\_loss = loss;

loss = 0;

for i=1:m

if(y(i) == 1)

loss=loss+log(sigmoid(x(i,:)\*theta));

else

loss=loss+log(1-sigmoid(x(i,:)\*theta));

end

end

disp(-old\_loss);

temp\_loss=[temp\_loss,-old\_loss];

end

plot\_x2 =zeros(1,51);

plot\_x1 =15:1:65;

for i=15:65

plot\_x2(i-14) = -(theta(3)+theta(1)\*((i-mean(1))/std(1)))/theta(2);

plot\_x2(i-14) = plot\_x2(i-14)\*std(2)+mean(2);

end

hold on;

plot(plot\_x1,plot\_x2,'-');

figure(2);

plot(temp\_loss,'-o');

**5.牛顿法求解logistics回归:**

clc;

clear;

x=load("ex4Data/ex4x.dat");

y=load("ex4Data/ex4y.dat");

[m,n]=size(y);

figure(1);

xlabel('Exam 1 score');

ylabel('Exam 2 score');

for i=1:m

hold on;

if(y(i)==1)

plot(x(i,1),x(i,2),'b+');

else

plot(x(i,1),x(i,2),'bo');

end

end

mean=mean(x);

std=std(x);

for i=1:m

for z=1:2

x(i,z)=(x(i,z)-mean(z))/std(z);

end

end

temp(1:80,1)=1;

x=[temp,x];

a = 0.1;

theta =[0;0;0];

loss = 0;

old\_loss = 0;

temp\_loss=[];

temp=-(x\*theta);

result1(1:80,1)=0;

for i=1:m

result1(i)=sigmoid(temp(i));

end

loss = get\_loss(y,result1);

while (abs(old\_loss-loss) > 0.000001)

%h = 1.0/(1+np.exp(-(data\_x\*theda)))

temp=-(x\*theta);

h(1:80,1)=0;

for i=1:m

h(i)=sigmoid(temp(i));

end

J = (1.0/m).\*(x'\*(h-y));

temp=h.\*(1-h);

result2(80:80,1)=0;

for i=1:m

result2(i,i)=temp(i);

end

H = (1.0/m).\*(x'\*result2\*x);

theta = theta-inv(H)\*J;

old\_loss = loss;

temp\_loss=[temp\_loss,old\_loss];

loss = 0;

loss = get\_loss(y,h);

end

plot\_x2 =zeros(1,51);

plot\_x1 =15:1:65;

for i=15:65

plot\_x2(i-14) = -(theta(1)+theta(3)\*((i-mean(1))/std(1)))/theta(2);

plot\_x2(i-14) = plot\_x2(i-14)\*std(2)+mean(2);

end

hold on;

plot(plot\_x1,plot\_x2,'-');

figure(2);

plot(temp\_loss,'-o');