

**机器学习作业**

**题 目: Softmax回归**

**学 号: 919106840333**

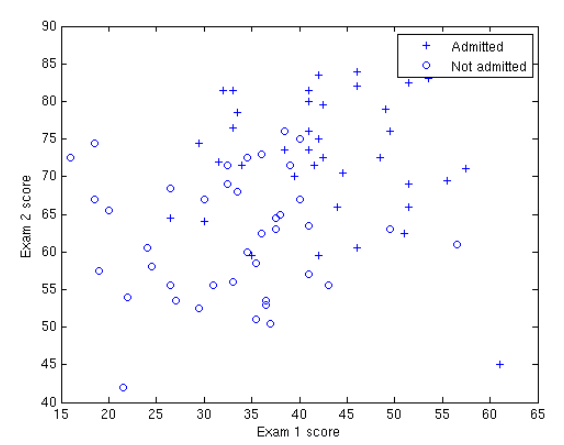
**姓 名: 孙傲歆**

**指导教师: 潘金山**

**2021年11月**

**一、题目要求**

• 给出下列训练数据:



• 实现Logistic回归：1) GD; 2) SGD

• 实现Softmax回归：1) GD; 2) SGD

• 比较Logistic回归和softmax回归

**二、算法思路**

**1.GD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.SGD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.GD求解softmax回归:**

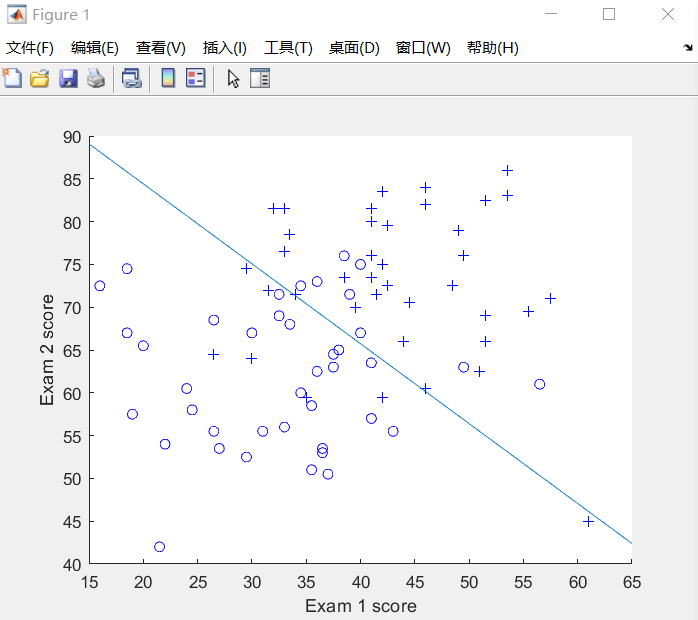
实际上softmax回归可以说是logistics回归的扩展，而logistics回归正是softmax回归当分类数为2时的特殊情况。同样地，我们先将x数据进行标准化处理，公式为：(X-X\_mean)/X\_std。我们取𝜃向量为[0,0;0,0;0,0]（即3\*2全0矩阵）。并设置学习率a=0.1，按照梯度下降法的更新公式进行迭代。当两次迭代结果之差小于0.001，我们认为结果收敛，停止迭代，并且在图中画出所有数据点，并将结果以直线形式显示出来。由于设置了两组𝜃，理论上应能画出两条直线。同时我们对每次迭代计算结果进行记录，绘制每次迭代计算结果的折线图，观察其变化趋势。

**4.SGD求解softmax回归:**

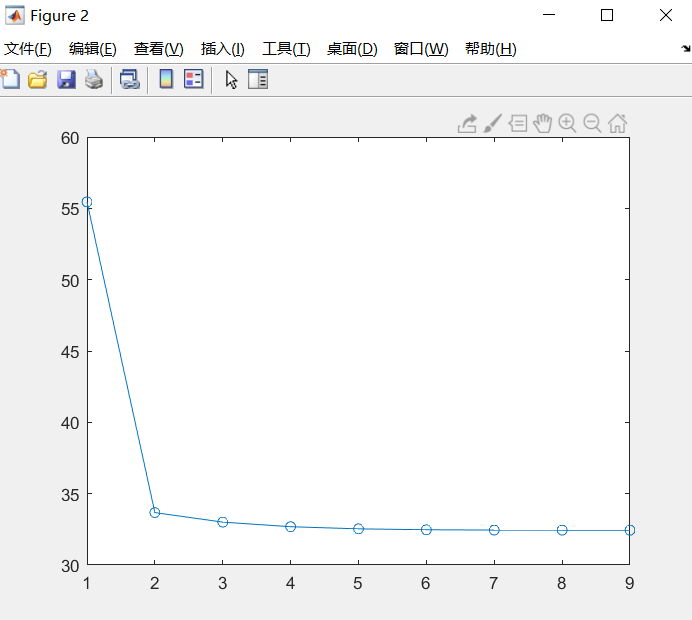
思路与梯度下降法类似，只是在每次迭代时，并不是选择全部样本进行计算，而是随机选择其中一小部分进行计算。这里我们在每次迭代时仅取1-80之间的1个数。其余部分与普通的梯度下降法几乎没有区别。同样地，我们会绘制出两条直线以显示结果。同时对每次迭代计算结果进行记录，绘制每次迭代计算结果的折线图，观察其变化趋势。

**三、运行结果**

**1.GD求解logistics回归:**



**图1.1**

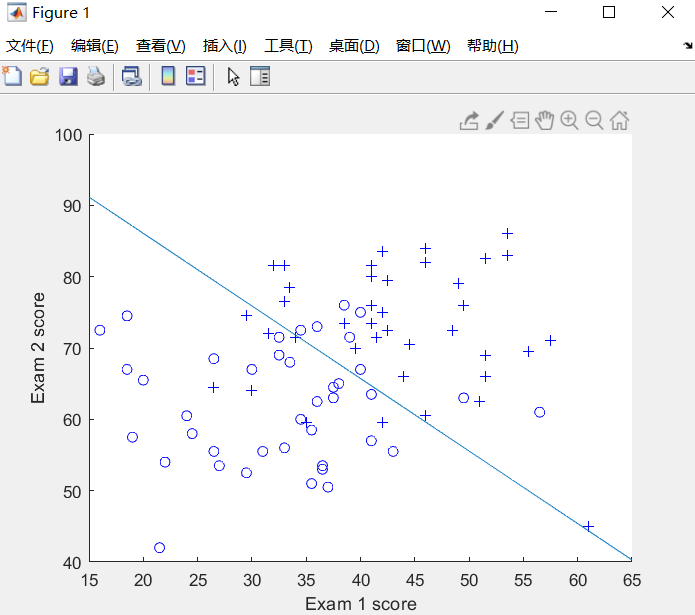


**图1.2**

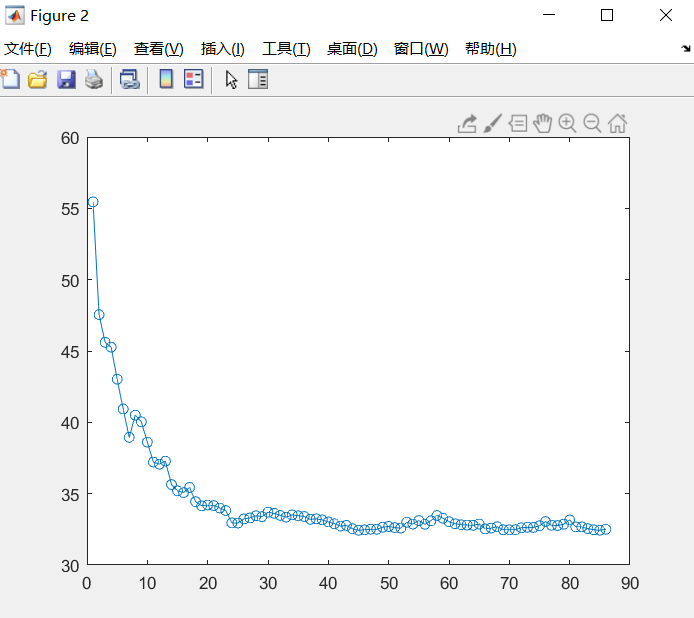
上图为代码运行结果。**图1.1**是数据显示，我们将所有点的信息绘制在图上并分类显示，o表示y值为1，+表示y值为0。直线是计算结果的显示，直线之上表示y=1的一类，直线之下表示y=0的一类。最终我们得到的函数为：。**图1.2**是每次迭代结果的折线图（不包含初始值），从图中可以看出，仅通过9次迭代，结果便达到了收敛。

**2.SGD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。

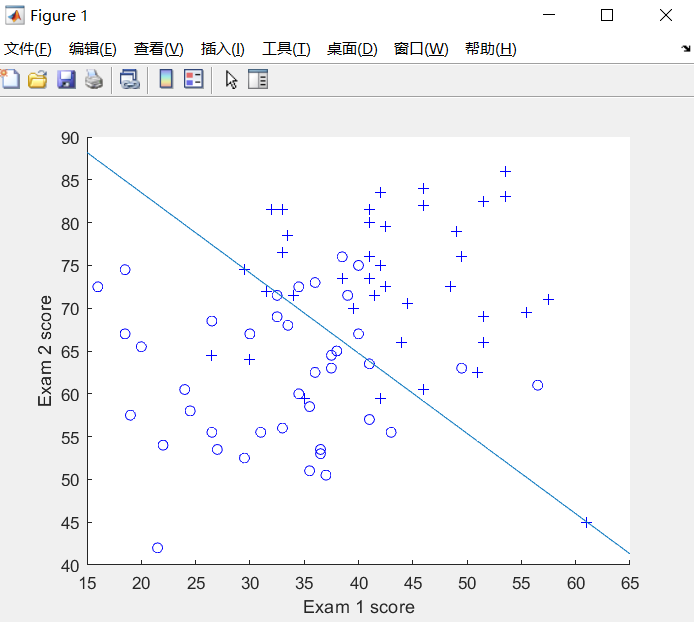


**图2.1**

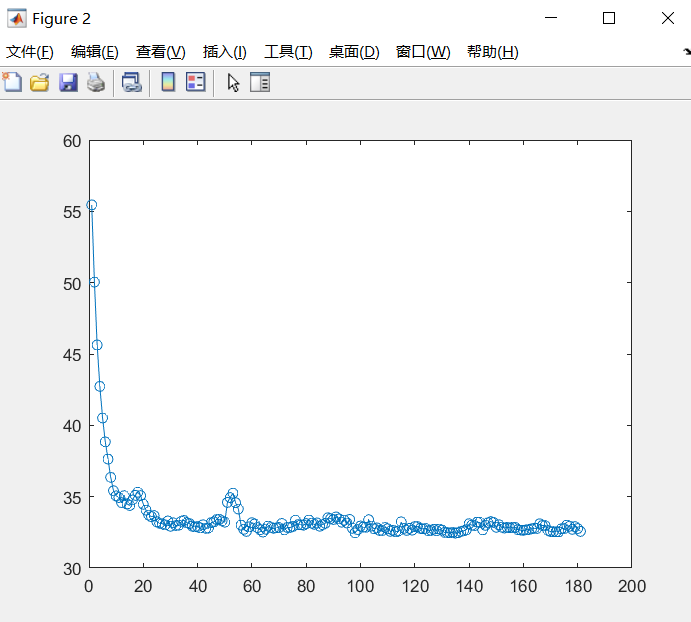


**图2.2**

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。



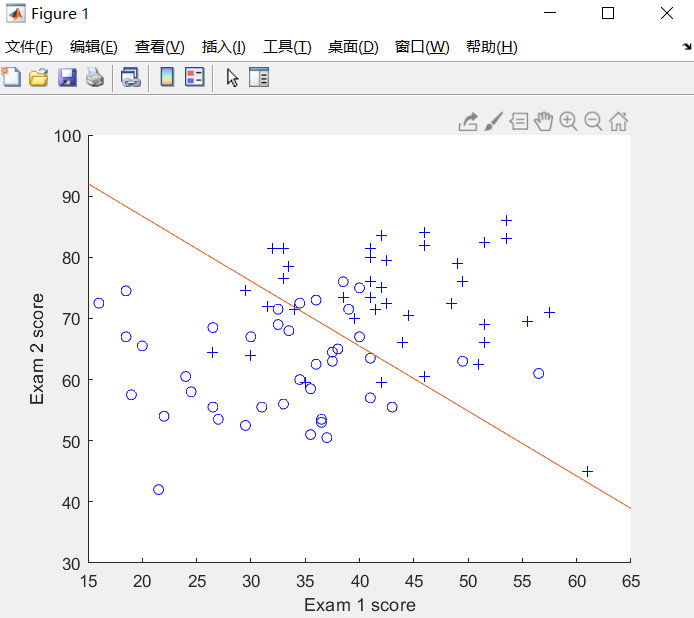
**图2.3**



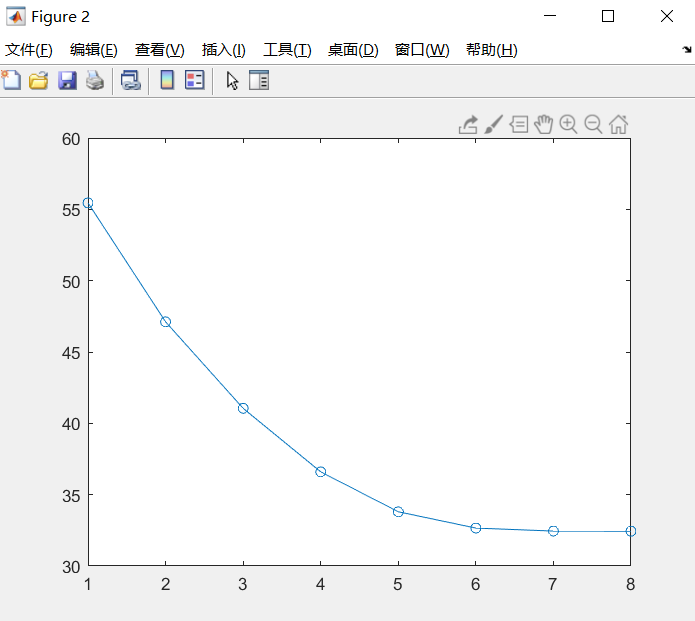
**图2.4**

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。

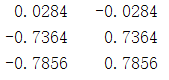
**3.GD求解softmax回归:**



**图3.1**



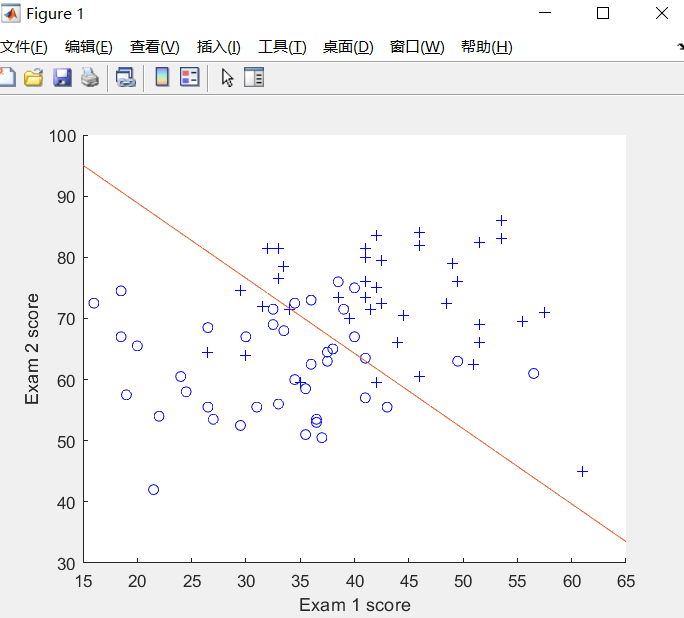
**图3.2**



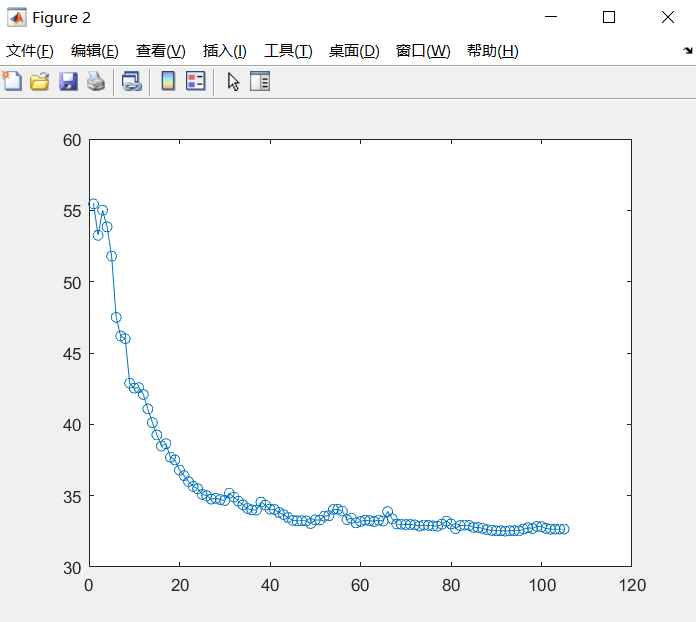
**图3.3**

**图3.1**和**图3.2**是代码运行结果，**图3.3**是最终的𝜃值。我们可以看到，虽然𝜃设置了两组，但是第一组数据和第二组数据之间仅差了一个负号，而且实际上用两组数据画出的直线是重合的。所以说虽然我们没有强制第C个𝜃为全0向量，实际上对应二分类问题，最终还是只会得到一条分界线。而且观察**图3.2**似乎softman回归在相同学习率与数据集情况下，其结果收敛相较于logistics回归似乎更为“平缓”。

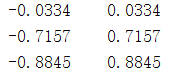
**4.SGD求解softmax回归：**



**图4.1**



**图4.2**



**图4.3**

**图4.1**和**图4.2**是代码运行结果，**图4.3**是最终的𝜃值。与递归下降法求解softmax回归一样，虽然设置了两组𝜃，但最终的分界线实际上是重合的。由于这里我们每次迭代只随机选取1个样本进行计算，所以总的迭代次数较大，上述运行结果一共进行了105次迭代，结果才最终收敛。

**5.logistics回归与softmax回归的比较**

我们可以看到，实际上logistics回归就是softmax回归的一个特例，当解决的问题是二分类问题时，两者得到的结果实际上相差不大。

值得一提的是，通过观察使用梯度下降法解决logistics回归与softmax回归的收敛曲线，发现似乎softmax回归的收敛曲线显得更加“平滑”，之后我又尝试了将学习率更改为0.01,0.001。发现同样的数据集以及同样的学习率条件下，貌似softmax回归的收敛曲线显得就是要比logistics回归的收敛曲线更加“平滑”。

**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：**

**1）用于logistics回归：**

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

**2）用于softmax回归**

function k = get\_loss(x,y,w)

temps=exp(x\*w);

tempsum=sum(temps,2);

for i=1:80

temps(i,1)=temps(i,1)/tempsum(i);

temps(i,2)=temps(i,2)/tempsum(i);

end

temps=log(temps);

k=0;

for i=1:80

k=k+temps(i,round(y(i))+1);

end

end

**3.GD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.SGD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.GD求解softmax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 =zeros(3,2);

const=zeros(80,2);

for i=1:80

const(i,round(y(i)+1)) = 1;

end

old\_loss = 0;

temp\_loss=[];

loss=get\_loss(x,y,theta);

while abs(old\_loss-loss) > 0.001

temp = exp(x\*theta);

%temp = temp / np.sum(temp, axis=1)

tempsum=sum(temp,2);

for i=1:80

temp(i,1)=temp(i,1)/tempsum(i);

temp(i,2)=temp(i,2)/tempsum(i);

end

temps = const-temp;

theta=theta+a\*(temps'\*x)';

old\_loss = loss;

loss=get\_loss(x,y,theta);

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

disp(old\_loss);

end

plot\_x2 =zeros(1,51);

plot\_x1 =15:1:65;

for i=15:65

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

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

end

hold on;

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

plot\_x2 =zeros(1,51);

plot\_x1 =15:1:65;

for i=15:65

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

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');

disp(theta);

**6.SGD求解softmax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 =zeros(3,2);

const=zeros(80,2);

for i=1:80

const(i,round(y(i)+1)) = 1;

end

old\_loss = 0;

temp\_loss=[];

loss=get\_loss(x,y,theta);

while abs(old\_loss-loss) > 0.001

temp = exp(x\*theta);

tempsum=sum(temp,2);

for i=1:80

temp(i,1)=temp(i,1)/tempsum(i);

temp(i,2)=temp(i,2)/tempsum(i);

end

temps =const-temp;

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

tempx = x(z,:);

temps=temps(z,:);

theta=theta+a\*(temps'\*tempx)';

old\_loss = loss;

loss=get\_loss(x,y,theta);

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

disp(old\_loss);

end

plot\_x2 =zeros(1,51);

plot\_x1 =15:1:65;

for i=15:65

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

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

end

hold on;

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

plot\_x2 =zeros(1,51);

plot\_x1 =15:1:65;

for i=15:65

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

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');

disp(theta);