山东大学 计算机科学与技术 学院

机器学习与模式识别 课程实验报告

学号：201600301304	姓名：贾乘兴	班级： 人工智能 16
实验题目：基于牛顿法迭代的逻辑回归		
实验学时：2 小时	实验日期： 2018/10/22	
实验目的：本实验是在给定的两个 feature（两门考试成绩）下，预测是否被通过（y=0 或 y=1）的问题。在给定的数据集 x（两个特征）与标签 y 下，实现逻辑回归，并牛顿法的数值方法进行参数的更新，最终得到参数，并进行预测		
硬件环境： 16 GB 内存		
软件环境：mac os，matlab 2017b		

实验步骤与内容：

一．逻辑回归

1. 在线性回归的假设中，我们令 y=wx+b，对应 y 的值域在本问题下显然不合适，本问题是通过或者不通过的两个标签的问题，是个离散的结果，我们的假设的函数是连续的，所以可以将该假设变为预测问题，得到的结果为概率。
2. 适合作为概率预测的函数，我们选择了 sigmoid 函数，sigmoid 函数如下：

$$y=\frac{1}{1+e^{-z}}$$

   sigmoid 函数具有对称性等良好性质，可以将 z=wx+b 映射在 0 到 1 的区间内，然后我们将其二值化，大于 0.5 为 1，小于 0.5 为 0，0.5 自由处理，这样就实现了逻辑回归。
3. 逻辑回归问题中，我们定义的目标函数如下：

$$J(\theta)=\prod_{i=1}^{m}\left[h_\theta(x_i)^{y_i}\left(1-h_\theta(x_i)\right)^{1-y_i}\right]$$

   观察可知，该函数在 h 与 y 最接近时是最大的，所以我们的目的是最大化该函数，得到的 h 才能接近真实的 y。但该函数在求导和计算上有一些问题和困难，容易出现向下溢出等等问题，而且也与一般问题中最小化目标函数不同，所以我们对其取对数并取反，得到了我们定义的损失函数如下：

$$Loss(\theta)=-\frac{1}{m}\sum_{i=1}^{m}\left[y_i\log\left(h_\theta(x_i)\right)+(1-y_i)\log\left(1-h_\theta(x_i)\right)\right]$$

   对损失函数求导可得：

$$\nabla_\theta L=\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x_i)-y_i\right)x_i$$

二．牛顿法参数更新

1. 在之前梯度下降中，我们参数的更新公式为：

$$\theta:=\theta-\eta\nabla_\theta L$$

其中埃塔为学习率，是超参数，我们设置了迭代停止的条件如下：

$$\left|L_{i+1}-L_i\right|<\varepsilon$$

其中埃普西隆为设置的阈值，该问题设置的阈值为 1e-9，在学习过程中我们发现最好的学习率下都需要几百次迭代，我们将数据标准化后迭代次数达到了 10 次（5.83 的学习率下）。

2. 本实验我们使用牛顿法的数值方法进行参数更新，更新公式如下：

$$\theta := \theta - H^{-1}\nabla_{\theta} L$$

其中 H 为 hession 矩阵，对 L 各个参数的二阶求导组成的矩阵，hession 矩阵公式如下：

$$H=\frac{1}{m}\sum_{i=1}^{m}\left[h_{\theta}\left(x_i\right)\left(1-h_{\theta}\left(x_i\right)\right)x_i x_i^T\right]$$

在该方法下，我们只需要 8 次就可以达到收敛。

三．边界确定

在逻辑回归中分界线是 h=0.5，对应的是 theta*x=0，得到参数后将直线 theta*x=0 在图像上表示出来

四．实验内容与部分代码

1. 数据读取、分类与显示：

```
%% data load
x=load('ex4x.dat');
y=load('ex4y.dat');
m=length(y);
X=[ones(m,1),x];

%% show the data
pos=find(y==1);
neg=find(y==0);
figure;
plot(X(pos,2),X(pos,3),'+');
hold on
plot(X(neg,2),X(neg,3),'o');
xlabel('exam 1 score');
ylabel('exam 2 scroe');
```

2. 牛顿法迭代代码如下：

```
%% train(newton's method)
e=1e-9;
theta=zeros(3,1);
loss=zeros(1,5);
num=1;
loss(1,num)=inf;
% repeat
num=num+1;
h=1./(1+exp(-X*theta));
H=(1/m)*X'*(h.*(1-h).*X);
```

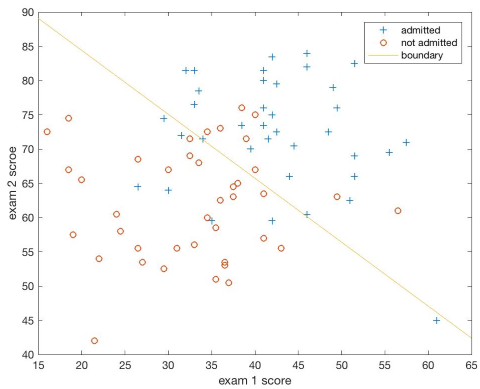
```
hinv=inv(H);
loss(1,num)=-(1/m)*sum((1-y).*log(1-h)+y.*log(h));
theta=theta-hinv*(1/m)*(X'*(h-y));
while abs(loss(1,num)-loss(1,num-1))>e
    num=num+1;
    h=1./(1+exp(-X*theta));
    H=(1/m)*X'*(h.*(1-h).*X);
    hinv=inv(H);
    loss(1,num)=-(1/m)*sum((1-y).*log(1-h)+y.*log(h));
    theta=theta-hinv*(1/m)*(X'*(h-y));
end
```

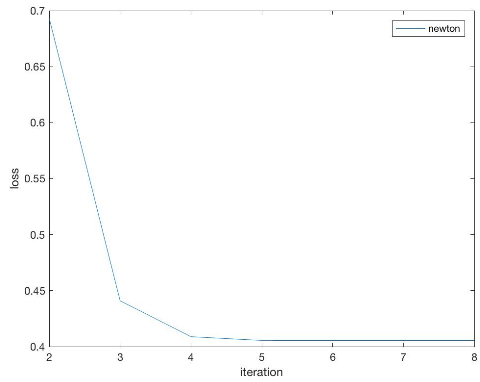
3. 训练结果图像绘制：

```
% theta*X=0,show the decision boundary(newton's method)
x1=[15:0.1:65];
x2=(theta(1)+theta(2)*x1)/(-theta(3));
hold on
plot(x1,x2,'-');
legend('admitted','not admitted','boundary');

%% plot
figure
plot([1:num],loss,'-');
xlabel('iteration');
ylabel('loss');
legend('newton');
```

得到关于边界和迭代过程中误差变化的图像如下：





4. 问题求解：

```
%% pre
x0=[1,20,80];
y0=1/(1+exp(-x0*theta));

%% show the result
theta
```

```
1-y0
num
```

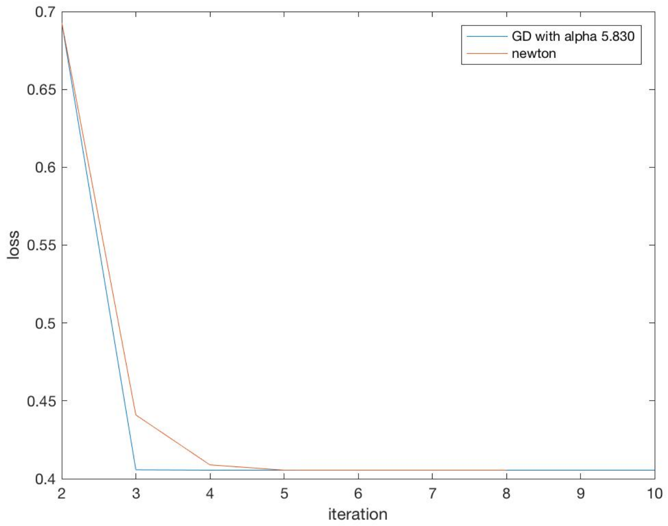
结果保留四位小数，最终得到的 theta 为：[-16.3787,0.1483,0.1589]

跌代次数为 8

不被通过的概率为 0.6680

五. 问题改进与探究

我们对原数据进行标准化之后，设置学习率为 5.830，与牛顿法收敛速度进行对比如下：



可见只有在最好的学习率与良好的数据条件下，手动设置学习率的梯度下降方法才能略差与牛顿法（迭代 10 次与迭代 8 次），而复杂的问题下找到最好的学习率是很困难的，所以牛顿法是效率极高的一种方法。

结论分析与体会： 本次实验实现了逻辑回归问题，通过对牛顿法的梯度下降的实现，我们对优化逻辑回归有了更好的掌握

附录：程序源代码

ex4.m

```
clear,clc;
%% data load
x=load('ex4x.dat');
y=load('ex4y.dat');
m=length(y);
X=[ones(m,1),x];
```

```matlab
%% show the data
pos=find(y==1);
neg=find(y==0);
figure;
plot(X(pos,2),X(pos,3),'+');
hold on
plot(X(neg,2),X(neg,3),'o');
xlabel('exam 1 score');
ylabel('exam 2 scroe');

%% train(newton's method)
e=1e-9;
theta=zeros(3,1);
loss=zeros(1,5);
num=1;
loss(1,num)=inf;
% repeat
num=num+1;
h=1./(1+exp(-X*theta));
H=(1/m)*X'*(h.*(1-h).*X);
hinv=inv(H);
loss(1,num)=-(1/m)*sum((1-y).*log(1-h)+y.*log(h));
theta=theta-hinv*(1/m)*(X'*(h-y));
while abs(loss(1,num)-loss(1,num-1))>e
    num=num+1;
    h=1./(1+exp(-X*theta));
    H=(1/m)*X'*(h.*(1-h).*X);
    hinv=inv(H);
    loss(1,num)=-(1/m)*sum((1-y).*log(1-h)+y.*log(h));
    theta=theta-hinv*(1/m)*(X'*(h-y));
end
% theta*X=0,show the decision boundary(newton's method)
x1=[15:0.1:65];
x2=(theta(1)+theta(2)*x1)/(-theta(3));
hold on
plot(x1,x2,'-');
legend('admitted','not admitted','boundary');

%% plot
figure
plot([1:num],loss,'-');
xlabel('iteration');
ylabel('loss');
legend('newton');

%% pre
```

```
x0=[1,20,80];
y0=1/(1+exp(-x0*theta));

%% show the result
theta
1-y0
num
```