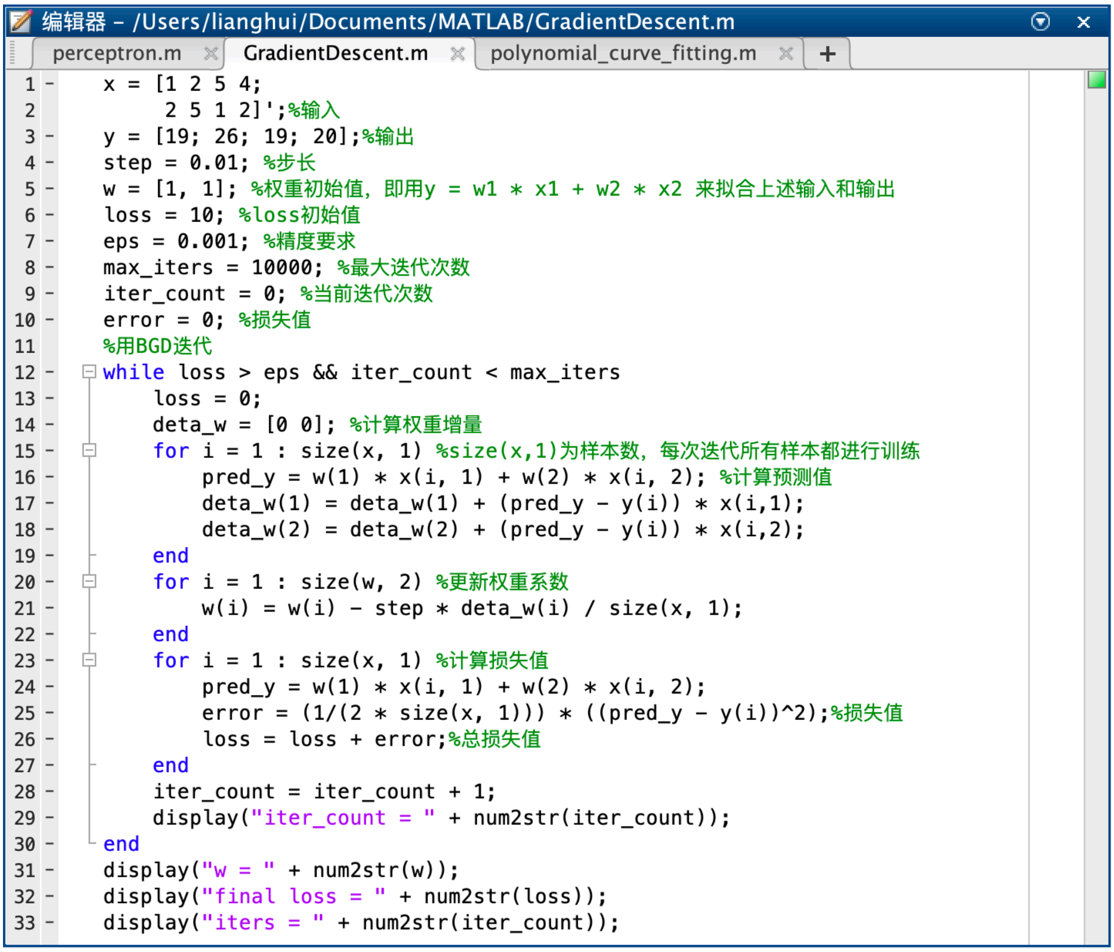
第六章选做作业答案

一、

- X=$\begin{pmatrix}1 & 2 & 5 & 4\\ 2 & 5 & 1 & 2\end{pmatrix}$, Y=$\begin{pmatrix}19\\26\\19\\20\end{pmatrix}$, step =0.001, w_0=[1 1]^T
- Implement SGD and GD to solution for w

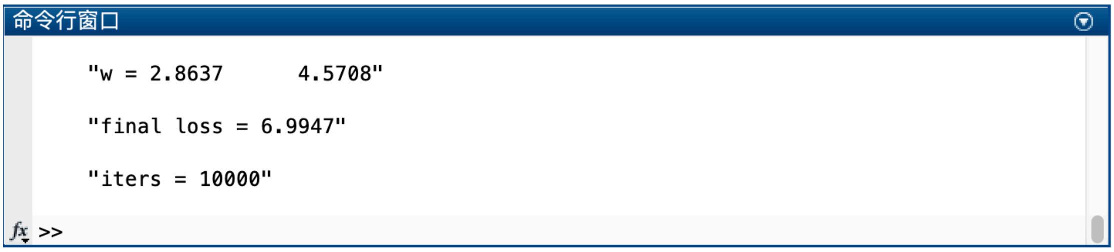
答：

用 GD 进行权重更新，w = [2.8737, 4.5708]，代码如下：


编辑器 - /Users/lianghui/Documents/MATLAB/GradientDescent.m
perceptron.m
GradientDescent.m
polynomial_curve_fitting.m
+

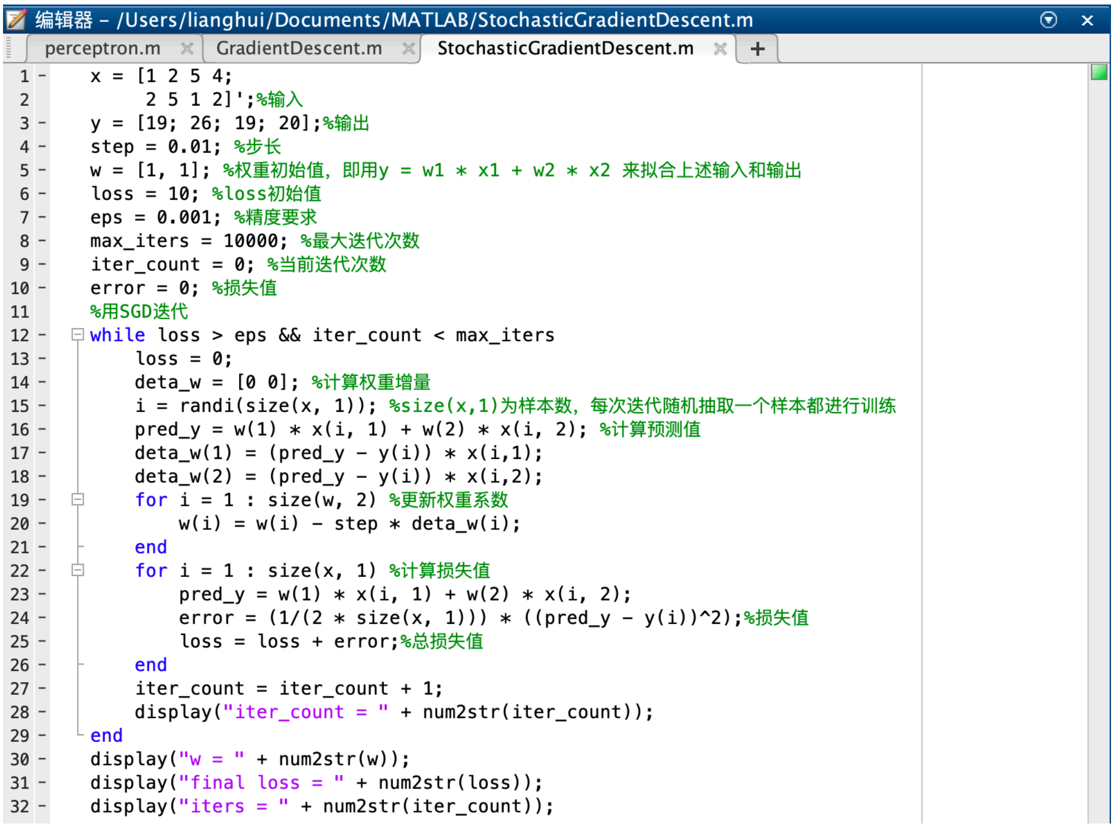
```matlab
x = [1 2 5 4;
     2 5 1 2]';%输入
y = [19; 26; 19; 20];%输出
step = 0.01; %步长
w = [1, 1]; %权重初始值，即用y = w1 * x1 + w2 * x2 来拟合上述输入和输出
loss = 10; %loss初始值
eps = 0.001; %精度要求
max_iters = 10000; %最大迭代次数
iter_count = 0; %当前迭代次数
error = 0; %损失值
%用BGD迭代
while loss > eps && iter_count < max_iters
    loss = 0;
    deta_w = [0 0]; %计算权重增量
    for i = 1 : size(x, 1) %size(x,1)为样本数，每次迭代所有样本都进行训练
        pred_y = w(1) * x(i, 1) + w(2) * x(i, 2); %计算预测值
        deta_w(1) = deta_w(1) + (pred_y - y(i)) * x(i,1);
        deta_w(2) = deta_w(2) + (pred_y - y(i)) * x(i,2);
    end
    for i = 1 : size(w, 2) %更新权重系数
        w(i) = w(i) - step * deta_w(i) / size(x, 1);
    end
    for i = 1 : size(x, 1) %计算损失值
        pred_y = w(1) * x(i, 1) + w(2) * x(i, 2);
        error = (1/(2 * size(x, 1))) * ((pred_y - y(i))^2);%损失值
        loss = loss + error;%总损失值
    end
    iter_count = iter_count + 1;
    display("iter_count = " + num2str(iter_count));
end
display("w = " + num2str(w));
display("final loss = " + num2str(loss));
display("iters = " + num2str(iter_count));
```

输出结果如下：



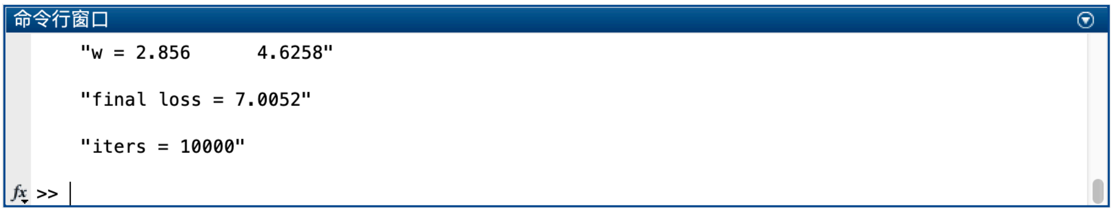
```
    "w = 2.8637      4.5708"

    "final loss = 6.9947"

    "iters = 10000"

fx >>
```

用 SGD 进行权重更新，w = [2.856,4.6258]，代码如下：


编辑器 – /Users/lianghui/Documents/MATLAB/StochasticGradientDescent.m
perceptron.m
GradientDescent.m
StochasticGradientDescent.m
+

```
x = [1 2 5 4;
     2 5 1 2]';%输入
y = [19; 26; 19; 20];%输出
step = 0.01; %步长
w = [1, 1]; %权重初始值，即用y = w1 * x1 + w2 * x2 来拟合上述输入和输出
loss = 10; %loss初始值
eps = 0.001; %精度要求
max_iters = 10000; %最大迭代次数
iter_count = 0; %当前迭代次数
error = 0; %损失值
%用SGD迭代
while loss > eps && iter_count < max_iters
    loss = 0;
    deta_w = [0 0]; %计算权重增量
    i = randi(size(x, 1)); %size(x,1)为样本数，每次迭代随机抽取一个样本都进行训练
    pred_y = w(1) * x(i, 1) + w(2) * x(i, 2); %计算预测值
    deta_w(1) = (pred_y - y(i)) * x(i,1);
    deta_w(2) = (pred_y - y(i)) * x(i,2);
    for i = 1 : size(w, 2) %更新权重系数
        w(i) = w(i) - step * deta_w(i);
    end
    for i = 1 : size(x, 1) %计算损失值
        pred_y = w(1) * x(i, 1) + w(2) * x(i, 2);
        error = (1/(2 * size(x, 1))) * ((pred_y - y(i))^2);%损失值
        loss = loss + error;%总损失值
    end
    iter_count = iter_count + 1;
    display("iter_count = " + num2str(iter_count));
end
display("w = " + num2str(w));
display("final loss = " + num2str(loss));
display("iters = " + num2str(iter_count));
```

输出结果如下：



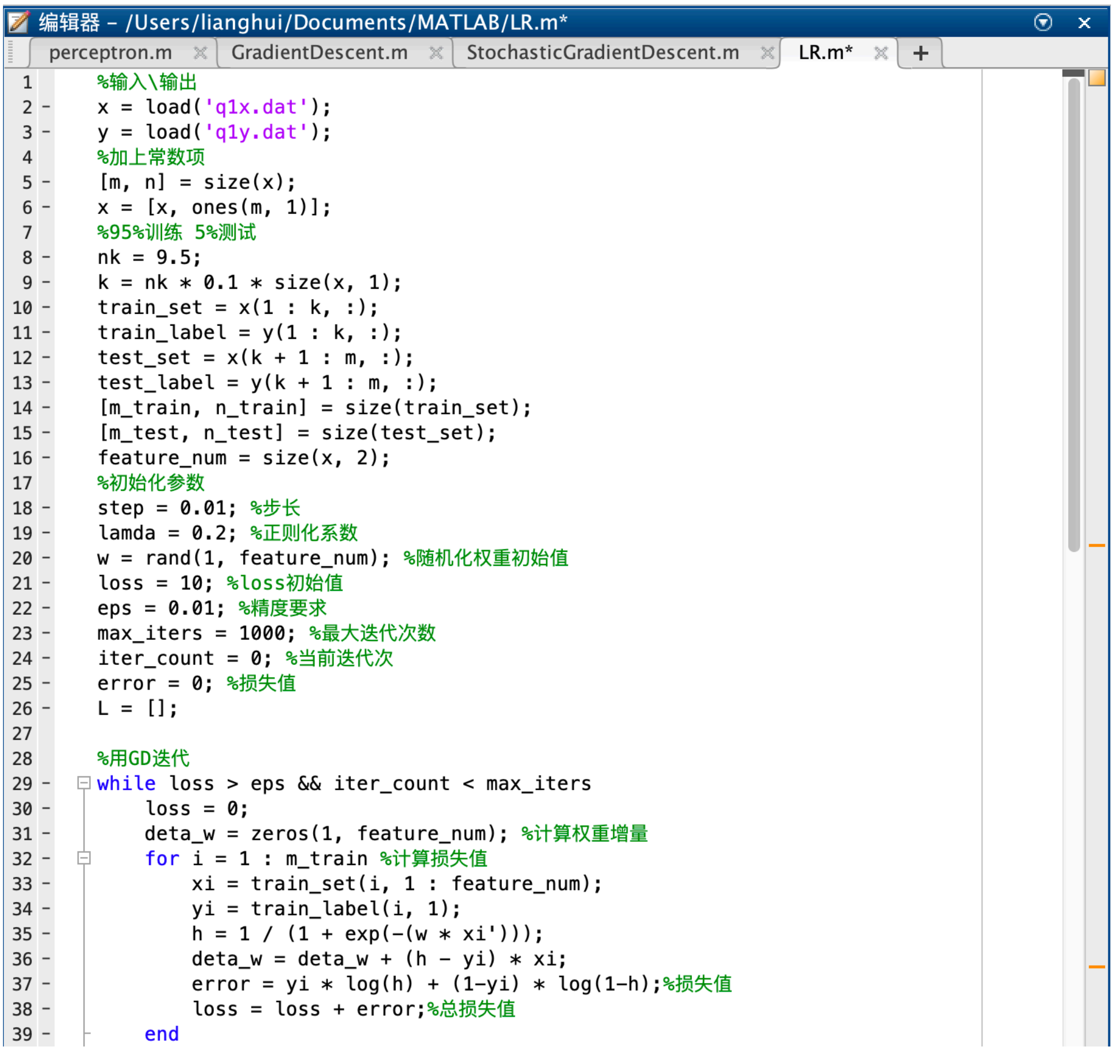
```
    "w = 2.856       4.6258"

    "final loss = 7.0052"

    "iters = 10000"

fx >> 
```

二、Practice of Logistic Regression:

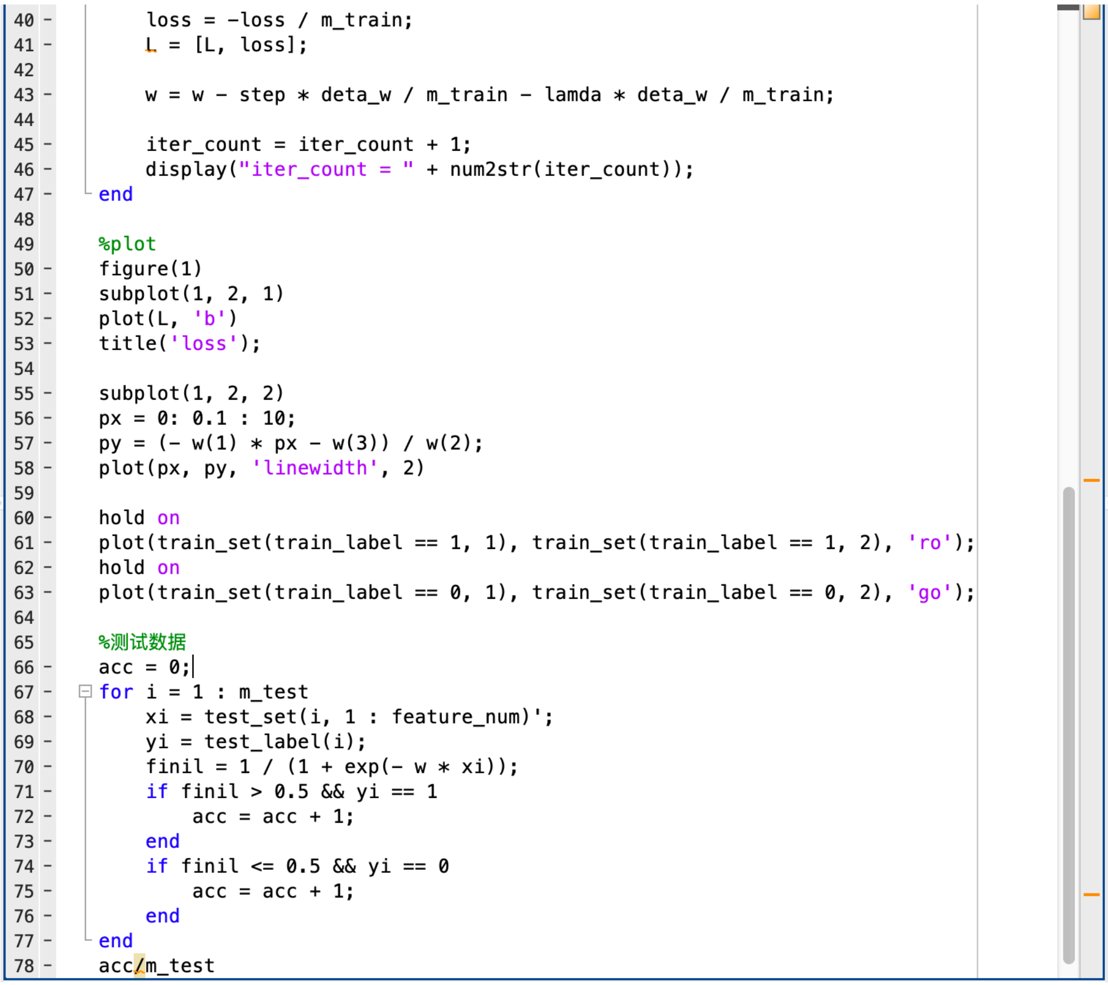
The files /afs/ir/class/cs229/ps/ps1/q1x.dat and /afs/ir/class/cs229/ps/ps1/q1y.dat contain the inputs (x(i) ∈ R2) and outputs (y(i) ∈ {0, 1}) respectively for a binary classification problem, with one training example per row. Implement Gradient Descent method for optimizing ℓ(θ), and apply it to fit a logistic regression model to the data.

Data: q1x.dat, q1y.dat, q2x.dat, q2y.dat

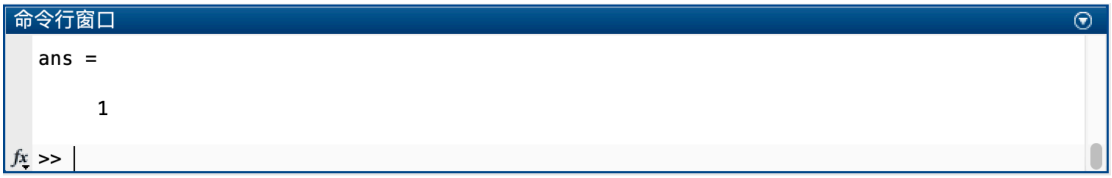
Binary Classification


编辑器 – /Users/lianghui/Documents/MATLAB/LR.m*
perceptron.m
GradientDescent.m
StochasticGradientDescent.m
LR.m*
+

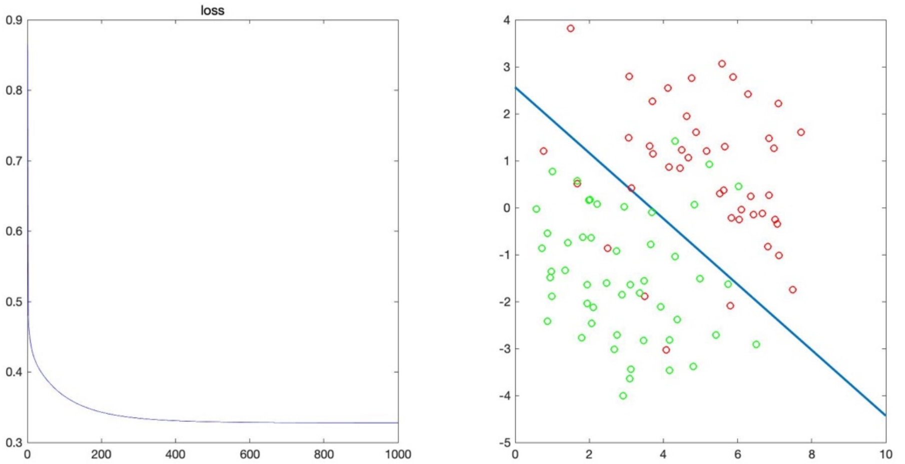
```
%输入\输出
x = load('q1x.dat');
y = load('q1y.dat');
%加上常数项
[m, n] = size(x);
x = [x, ones(m, 1)];
%95%训练 5%测试
nk = 9.5;
k = nk * 0.1 * size(x, 1);
train_set = x(1 : k, :);
train_label = y(1 : k, :);
test_set = x(k + 1 : m, :);
test_label = y(k + 1 : m, :);
[m_train, n_train] = size(train_set);
[m_test, n_test] = size(test_set);
feature_num = size(x, 2);
%初始化参数
step = 0.01; %步长
lamda = 0.2; %正则化系数
w = rand(1, feature_num); %随机化权重初始值
loss = 10; %loss初始值
eps = 0.01; %精度要求
max_iters = 1000; %最大迭代次数
iter_count = 0; %当前迭代次
error = 0; %损失值
L = [];

%用GD迭代
while loss > eps && iter_count < max_iters
    loss = 0;
    deta_w = zeros(1, feature_num); %计算权重增量
    for i = 1 : m_train %计算损失值
        xi = train_set(i, 1 : feature_num);
        yi = train_label(i, 1);
        h = 1 / (1 + exp(-(w * xi')));
        deta_w = deta_w + (h - yi) * xi;
        error = yi * log(h) + (1-yi) * log(1-h);%损失值
        loss = loss + error;%总损失值
    end
```



```
    loss = -loss / m_train;
    L = [L, loss];

    w = w - step * deta_w / m_train - lamda * deta_w / m_train;

    iter_count = iter_count + 1;
    display("iter_count = " + num2str(iter_count));
end

%plot
figure(1)
subplot(1, 2, 1)
plot(L, 'b')
title('loss');

subplot(1, 2, 2)
px = 0: 0.1 : 10;
py = (- w(1) * px - w(3)) / w(2);
plot(px, py, 'linewidth', 2)

hold on
plot(train_set(train_label == 1, 1), train_set(train_label == 1, 2), 'ro');
hold on
plot(train_set(train_label == 0, 1), train_set(train_label == 0, 2), 'go');

%测试数据
acc = 0;
for i = 1 : m_test
    xi = test_set(i, 1 : feature_num)';
    yi = test_label(i);
    finil = 1 / (1 + exp(- w * xi));
    if finil > 0.5 && yi == 1
        acc = acc + 1;
    end
    if finil <= 0.5 && yi == 0
        acc = acc + 1;
    end
end
acc/m_test
```



```
ans =

     1

>>
```


loss