Homework 1

15331191 廖颖泓

1. 线性回归

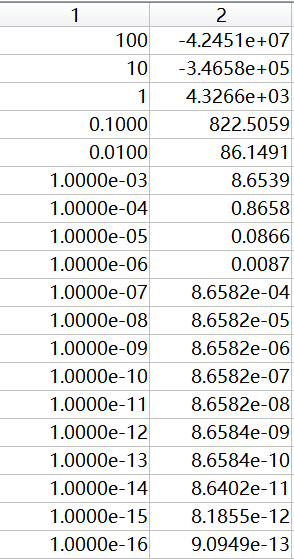
（1）因为讨论4门课分数和1门课分数的关系，所以θ0 = [θ0, θ1, θ2, θ3, θ4] = [0, 0, 0, 0, 0]。所以hθ(x) = θ0 + θ1x1 + θ2x2 + θ3x3 + θ4x4，其中m, x1, x2, x3, x4分别指数学、物理、语文、英语和化学5门课的分数。由此可以推出代价函数为 = =, = = 。由于学习率α= 1，则 = = = 。 经过一轮迭代以后，可以得到θ1 = [θ0, θ1, θ2, θ3, θ4] = [93, 2.45, 0.53, 1.53, 0.77]。

（2） = = = 4328.5。

= = = 1.93。

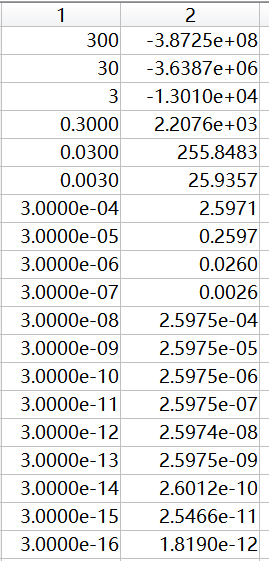
所以 < ，即（1）中所算出的θ1使线性回归中的代价函数𝐽(𝜽)下降。

（3）考虑到学习率每增加或减少一个数量级会对代价函数𝐽(𝜽)下降造成比较大的影响，于是在这里选定学习率α=1000作为初始值，循环计算代价函数𝐽(𝜽)下降的程度 - ，每进行一次循环后α = α / 10，记录下降程度和对应的α值。程序运行结果显示



当学习率α> 1时， - < 0，即此时梯度不下降；学习率α≤ 1时， - > 0，即此时梯度下降，而且随着α的减小， - 越来越小，代价函数𝐽(𝜽)下降越来越慢，说明所以为了使代价函数𝐽(𝜽)下降得更快，学习率α应该选取为 1该数量级，此时 - 的值大约为4326.6。

另外，选定学习率α=3000作为初始值，循环计算代价函数𝐽(𝜽)下降的程度 - ，每进行一次循环后α = α / 10，记录下降程度和对应的α值。程序运行结果显示



当学习率α> 0. 3时， - < 0，即此时梯度不下降；学习率α≤ 0.3时， - > 0，即此时梯度下降，而且随着α的减小， - 越来越小，代价函数𝐽(𝜽)下降越来越慢，说明所以为了使代价函数𝐽(𝜽)下降得更快。通过比较学习率α= 0.3和α= 1时 - 的值，应该选取α= 1，此时 - 的值大约为4326.6。

（4）θ = [θ0, θ1, θ2, θ3, θ4] = [-19.50, 1.69, 0.38, -0.31, -0.44]，hθ(x) = -19.50 + 1.69x1 + 0.38x2 – 0.31x3 – 0.44x4。hθ = 88.94，即该班物理分数88、语文分数73、英语分数87、化学分数92同学的数学分数的预测值为88.94。此时𝐽(𝜽) = 6329。

（5）θ = [θ0, θ1, θ2, θ3, θ4] = [-19.99, 1.47, 0.07, -0.23, -0.06]，hθ(x) = -19.99 + 1.47x1 + 0.07x2 - 0.23x3 - 0.06x4。hθ= 89.87，即该班物理分数88、语文分数73、英语分数87、化学分数92同学的数学分数的预测值为89.87。此时𝐽(𝜽) = 6385，该值大于只利用标准方程得到的结果，所以没有正则化的标准方程求解出的结果更好。

该题详细代码参见附录1。

1. 逻辑回归

（1）调用Matlab中的函数flmfit可以求出相应的参数向量θ = [θ0, θ1, θ2, θ3, θ4] = [-2.66, 2.22, 1.06, -1.77, 2.24]，此时的逻辑回归函数为 ，其中x1, x2, x3, x4分别雌激素使用情况、有无胆囊病史、有无高血压和非雌激素使用情况，取值为0和1。

（2）根据权值绝对值的大小，|θ4| = 2.24最大，可以判断出对影响子宫内膜癌发病的最直接的因素是使用过非雌激素。

（3）

function theta = logistic\_regression\_gradient\_descent(X, y, alpha, step, lamda)

[sample\_num, theta\_num] = size(X);

theta = zeros(theta\_num, 1);

temp = zeros(theta\_num, 1);

for i = 1:step

for j = 1:theta\_num

sum = 0;

for k = 1:sample\_num

if j == 1

sum = sum + (logistic\_regression\_function(theta, X(k, :))

- y(k)) \* X(k, j);

else

sum = sum + (logistic\_regression\_function(theta, X(k, :))

- y(k)) \* X(k, j) + lamda \* theta(j);

end

end

temp(j) = theta(j) - alpha \* sum / sample\_num;

end

for j = 1:theta\_num

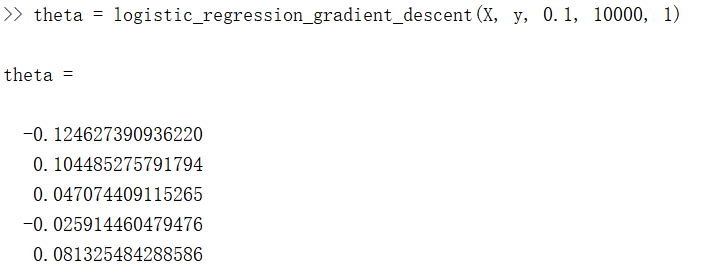
theta(j) = temp(j);

end

end

end

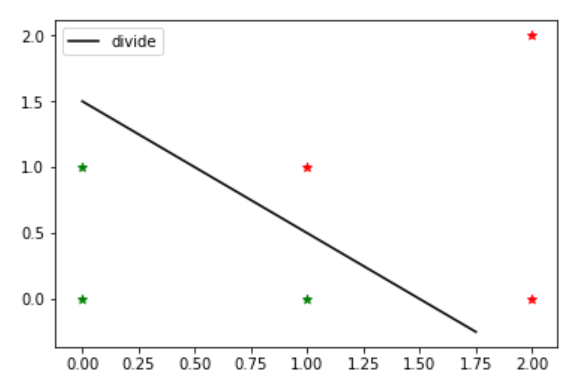
在这里，使用的正则项系数λ= 1，学习率α= 0. 1，步数为10000，输出结果为



编写代码求解出的参数向量θ = [θ0, θ1, θ2, θ3, θ4] = [-0.12, 0.10, 0.05, -0.03, 0.08]，参数代入逻辑回归函数为。

1. 支持向量机

（1）超平面方程为2x1 + 2x2 – 3 = 0。



（2） 在决定最优超平面时只有支持向量起作用，而其他训练样本点并不起作用。如果移动支持向量将改变所求的解，但是如果在决策边界以外移动其他实例点，甚至去掉这些点，则解是不会改变的，支持向量的个数一般很少，所以支持向量机由很少的训练样本点决定。而线性回归方程的参数θ值与训练样本点的值密切相关，无论采用标准方程还是梯度下降法求θ，训练样本点的增加都有可能导致θ值的变化，所以，最优超平面不受新增训练样本点影响，而线性回归会受影响。

（3）支持向量有 (0,1), (1,0), (1,1), (2,0)，“+”向量有(1, 0), (0, 1)，“-”向量有(1,1), (2,0)，这里取(1,0), (1,1)，求得距离之和为。

（4）根据对偶问题求解的KKT条件和互补松弛条件可以得到，，，，其中时为支持向量，时为非支持向量，结合上题中四个支持向量的值，可以生成方程组：

另外，

由此可以得到

附录1：

1. Solution.m

clear;

%% data mean nornalization

X = [87 89 89 92 93; 72 76 74 71 76; 83 88 82 91 89; 90 93 91 89 94];

x\_mean = mean(X');

x\_s = std(X');

for i = 1:4

norm\_X(i,:) = (X(i,:) - x\_mean(i)) / x\_s(i);

end

norm\_X = norm\_X';

temp = ones(5,1);

norm\_X = [temp norm\_X];

X = X';

temp = ones(5,1);

X = [temp X];

y = [89 91 93 95 97];

y = y';

%% Problem 1

fprintf('\nProblem 1\n');

theta = gradient\_descent(norm\_X, y, 1, 1);

fprintf('The theta after the first iteration is\n');

display(theta);

%% Problem 2

fprintf('\nProblem 2\n');

initial\_theta = zeros(5,1);

J0 = cost\_function(initial\_theta, norm\_X, y, 0);

J1 = cost\_function(theta, norm\_X, y, 0);

fprintf('The cost before the first iteration is\n');

display(J0);

fprintf('The cost after the first iteration is\n');

display(J1);

%% Problem 3

fprintf('\nProblem 3\n');

% test begin with 100 and 300

[Js1, descent1] = finding\_learning\_rate(norm\_X, y, 100, 1);

[Js2, descent2] = finding\_learning\_rate(norm\_X, y, 300, 1);

%% Problem 4

fprintf('\nProblem 4\n');

theta\_norm = finding\_solution\_using\_normal\_equation(X, y, 0);

fprintf('The theta obtained by normal equation is\n');

display(theta\_norm);

testX = [88 73 87 92];

testX = [1 testX];

predict\_y = testX \* theta\_norm;

fprintf('The math grade is\n');

display(predict\_y);

predict\_J = cost\_function(theta\_norm, norm\_X, y, 0);

fprintf('The cost is\n');

display(predict\_J);

%% Problem 5

fprintf('\nProblem 5\n');

theta\_norm\_la = finding\_solution\_using\_normal\_equation(X, y, 1);

fprintf('The theta obtained by normal equation is\n');

display(theta\_norm\_la);

predict\_y\_la = testX \* theta\_norm\_la;

fprintf('The math grade is\n');

display(predict\_y\_la);

predict\_J\_la = cost\_function(theta\_norm\_la, norm\_X, y, 0);

fprintf('The cost is\n');

display(predict\_J\_la);

1. cost\_function.m

%% function calculating the cost function for each theta

function J = cost\_function(theta, X, y, lamda)

[sample\_num, ~] = size(X);

temp = X \* theta - y;

J = 1 / (2 \* sample\_num) \* temp' \* temp + lamda \* 0.5 \* theta' \* theta;

end

1. gradient\_descent.m

%% function doing gradient descent

function thetas = gradient\_descent(X, y, alpha, step)

% initialization

[sample\_num, theta\_num] = size(X);

theta = zeros(theta\_num, 1);

temp = zeros(theta\_num, 1);

thetas = [];

% gradient descent

for i = 1:step

for j = 1:theta\_num

sum = 0;

for k = 1:sample\_num

sum = sum + (X(k, :) \* theta - y(k)) \* X(k, j);

end

temp(j) = theta(j) - alpha \* sum / sample\_num;

end

for j = 1:theta\_num

theta(j) = temp(j);

end

thetas = [thetas, theta];

end

end

1. finding\_learning\_rate.m

%% function finding the appropriate learning rate

function [Js, descent] = finding\_learning\_rate(X, y, alpha, step)

Js = [];

alphas = [];

descent = [];

[~,theta\_num] = size(X);

theta = zeros(theta\_num, 1);

initial = cost\_function(theta, X, y, 0);

for j = 1:100

thetas = gradient\_descent(X, y, alpha, step);

steps = ones(1,step);

for i = 1:step

steps(i) = i;

J(i) = cost\_function(thetas(:,i), X, y, 0);

end

% difference

descent = [descent, initial - J];

Js = [Js, J];

alphas = [alphas, alpha];

alpha = alpha / 10;

end

descent = descent';

Js = Js';

alphas = alphas';

descent = [alphas, descent];

end

1. finding\_solution\_using\_normal\_equation.m

%% function using normal equation to find theta for X and y

function theta = finding\_solution\_using\_normal\_equation(X, y, lamda)

[~,theta\_num] = size(X);

regulation\_matrix = eye(theta\_num);

regulation\_matrix(1, 1) = 0;

theta = inv(X' \* X + lamda \* regulation\_matrix) \* X' \* y;

end