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 $\theta^0 = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4] = [0, 0, 0, 0, 0]$。所以 $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$，其中 m, x_1, x_2, x_3, x_4 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$J(\theta_0, \theta_1, \dots, \theta_4) = \frac{1}{10}\sum_{i=1}^{5}\left(h_\theta\left(x^{(i)}\right) - m^{(i)}\right)^2 = \frac{1}{10}\sum_{i=1}^{5}\left(\theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \theta_3 x_3^{(i)} + \theta_4 x_4^{(i)} - m^{(i)}\right)^2$，$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \dots, \theta_4) = \frac{1}{5}\sum_{i=1}^{5}(\theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \theta_3 x_3^{(i)} + \theta_4 x_4^{(i)} - m^{(i)})\, x_j^{(i)} = \frac{1}{5}\sum_{i=1}^{5}(-m^{(i)})\, x_j^{(i)}$。由于学习率 α = 1，则$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \dots, \theta_4) = \theta_j - \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \dots, \theta_4) = -\frac{1}{5}\sum_{i=1}^{5}(-m^{(i)})\, x_j^{(i)}$。经过一轮迭代以后，可以得到 $\theta^1 = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4] = [93, 2.45, 0.53, 1.53, 0.77]$。

（2）$J(\theta^0) = J(\theta_0, \theta_1, \dots, \theta_4) = \frac{1}{10}\sum_{i=1}^{5}\left(\theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \theta_3 x_3^{(i)} + \theta_4 x_4^{(i)} - m^{(i)}\right)^2$ = 4328.5。

$J(\theta^1) = J(\theta_0, \theta_1, \dots, \theta_5) = \frac{1}{10}\sum_{i=1}^{5}\left(93 + 2.45 x_1^{(i)} + 0.53 x_2^{(i)} + 1.53 x_3^{(i)} + 0.77 x_4^{(i)} - m^{(i)}\right)^2$ = 1.93。

所以$J(\theta^1) < J(\theta^0)$，即（1）中所算出的$\theta^1$使线性回归中的代价函数$J(\boldsymbol{\theta})$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$J(\boldsymbol{\theta})$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$J(\boldsymbol{\theta})$下降的程度$J(\theta^1) - J(\theta^0)$，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

1	2
100	-4.2451e+07
10	-3.4658e+05
1	4.3266e+03
0.1000	822.5059
0.0100	86.1491
1.0000e-03	8.6539
1.0000e-04	0.8658
1.0000e-05	0.0866
1.0000e-06	0.0087
1.0000e-07	8.6582e-04
1.0000e-08	8.6582e-05
1.0000e-09	8.6582e-06
1.0000e-10	8.6582e-07
1.0000e-11	8.6582e-08
1.0000e-12	8.6584e-09
1.0000e-13	8.6584e-10
1.0000e-14	8.6402e-11
1.0000e-15	8.1855e-12
1.0000e-16	9.0949e-13

当学习率 α > 1 时，$J(\theta^1) - J(\theta^0) < 0$，即此时梯度不下降；学习率 α ≤ 1 时，$J(\theta^1) - J(\theta^0) > 0$，即此时梯度下降，而且随着 α 的减小，$J(\theta^1) - J(\theta^0)$越来越小，代价函数$J(\boldsymbol{\theta})$下降越来越慢，说明所以为了使代价函数$J(\boldsymbol{\theta})$下降得更快，学习率 α 应该选取为 1 该数量级，此时$J(\theta^1) - J(\theta^0)$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$J(\boldsymbol{\theta})$下降的程度$J(\theta^1) - J(\theta^0)$，

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

1	2
300	-3.8725e+08
30	-3.6387e+06
3	-1.3010e+04
0.3000	2.2076e+03
0.0300	255.8483
0.0030	25.9357
3.0000e-04	2.5971
3.0000e-05	0.2597
3.0000e-06	0.0260
3.0000e-07	0.0026
3.0000e-08	2.5975e-04
3.0000e-09	2.5975e-05
3.0000e-10	2.5975e-06
3.0000e-11	2.5975e-07
3.0000e-12	2.5974e-08
3.0000e-13	2.5975e-09
3.0000e-14	2.6012e-10
3.0000e-15	2.5466e-11
3.0000e-16	1.8190e-12

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

（4）θ = [θ_0, θ_1, θ_2, θ_3, θ_4] = [-19.50, 1.69, 0.38, -0.31, -0.44]，$h_\theta(x)$ = -19.50 + 1.69x_1 + 0.38x_2 – 0.31x_3 – 0.44x_4。h_θ = 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$J(\boldsymbol{\theta})$ = 6329。

（5）θ = [θ_0, θ_1, θ_2, θ_3, θ_4] = [-19.99, 1.47, 0.07, -0.23, -0.06]，$h_\theta(x)$ = -19.99 + 1.47x_1 + 0.07x_2 - 0.23x_3 - 0.06x_4。h_θ= 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$J(\boldsymbol{\theta})$ = 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。

2. 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 = [θ_0, θ_1, θ_2, θ_3, θ_4] = [-2.66, 2.22, 1.06, -1.77, 2.24]，此时的逻辑回归函数为$h_\theta(x) = \frac{1}{1+e^{-\theta^T X}} = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1+\theta_2 x_2+\theta_3 x_3+\theta_4 x_4)}}$，其中 x_1, x_2, x_3, x_4 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，$|\theta_4|$ = 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
```

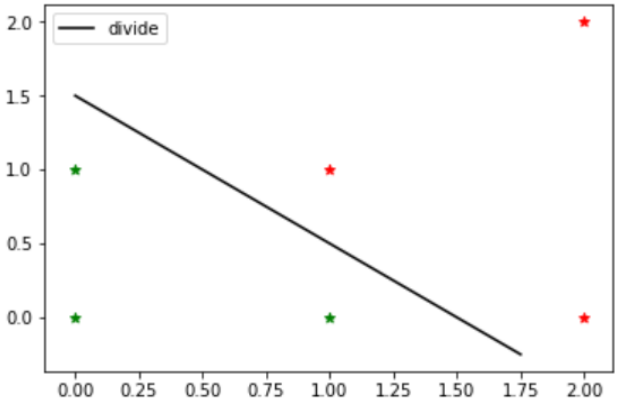
在这里，使用的正则项系数λ= 1，学习率α= 0. 1，步数为10000，输出结果为

```
>> theta = logistic_regression_gradient_descent(X, y, 0.1, 10000, 1)

theta =

  -0.124627390936220
   0.104485275791794
   0.047074409115265
  -0.025914460479476
   0.081325484288586
```

编写代码求解出的参数向量 θ = [θ_0, θ_1, θ_2, θ_3, θ_4] = [-0.12, 0.10, 0.05, -0.03, 0.08]，参数代入逻辑回归函数为$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} = \frac{1}{1+e^{-(-0.12+0.10x_1+0.05x_2-0.03x_3+0.08x_4)}}$。

3. 支持向量机

（1）超平面方程为 $2x_1 + 2x_2 - 3 = 0$。



（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$\frac{\sqrt{2}}{2}$。

（4）根据对偶问题求解的 KKT 条件和互补松弛条件可以得到，$\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}=w$，$\sum_{i=1}^{m}\alpha_i y^{(i)}=0$，$\alpha_i\left(w^T x^{(i)}+b-1\right)y^{(i)}=0$，其中$\alpha_i>0$时为支持向量，$\alpha_i=0$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：

$$2\alpha_1+2\alpha_3-\alpha_5-\alpha_6+b=1$$
$$2\alpha_1+4\alpha_3-2\alpha_5+b=1$$
$$\alpha_1+2\alpha_3-\alpha_5+b=-1$$
$$\alpha_1-\alpha_6+b=-1$$
$$\alpha_1+\alpha_3-\alpha_5-\alpha_6=0$$

另外，

$$w_1=\alpha_1+2\alpha_3-\alpha_5$$
$$w_2=\alpha_1-\alpha_6$$

由此可以得到

$$w_1=2$$
$$w_2=2$$
$$\mathrm{b}=-3$$

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);
```

(2) 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
```

(3) 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
```

(4) 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
```

(5) 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
```