Name: Xinrun Zhang

ML Homework 6 Report

Instructor: Prof. Kaliappan Gopalan

Time: 04/03/2019

1. Part 1

a) Calculate the quadratic cost function.

Let weight $\underline{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$, input $\underline{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\underline{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $\underline{t} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}_{2\times 1}$

$p = 0.5$

for $F(\underline{\theta}) = c - 2\underline{\theta}^T h + \underline{\theta}^T \underline{R}\, \underline{\theta}$

$c = E[t^2] = p \cdot \underline{t}^2 = [0.5\ 0.5] \begin{bmatrix} (1)^2 \\ (-1)^2 \end{bmatrix} = 1$

$h = E[t \cdot \underline{x}] = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$R = E[\underline{x} \cdot \underline{x}^T] = 0.5 \cdot x_1 \cdot x_1^T + 0.5 \cdot x_2 \cdot x_2^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\therefore J(\underline{\theta}) = 1 - 2[\theta_1\ \theta_2] \begin{bmatrix} 0 \\ 1 \end{bmatrix} + [\theta_1\ \theta_2] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

$= 1 - 2\theta_2 + \theta_1^2 + \theta_2^2$

```
% Calculate c, h and R, where
% c = E[t.^2]
% h = E[t .* x]
% R = E[x * x']
c = p * t1^2 + p * t2^2;
h = p * (t1 * x1) + p * (t2 * x2);
R = p * (x1 * x1') + p * (x2 * x2');
```

```
>> c,h,R

c =

     1


h =

     0
     1


R =

     1     0
     0     1
```

b) Obtain the optimal weight vector.

$\theta^* = R^{-1} \cdot h = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

```
% Calculate the optimal theta
theta_opt = (R^-1)* h;
```

```
>> theta_opt

theta_opt =

     0
     1
```

c) From the hessian, what is the maximum learning rate for iterative implementation of the LMS algorithm?

(c) Hessian $= 2 \cdot R = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

$$|H - \lambda I| = \left| \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \right| = \left| \begin{bmatrix} 2-\lambda & 0 \\ 0 & 2-\lambda \end{bmatrix} \right| = (2-\lambda)^2 = 0$$

$\therefore \lambda_1 = \lambda_2 = 2$

$\therefore \ 0 < \alpha < \frac{1}{\lambda}$

$\therefore$ the maximum learning rate is 0.5.

```
% Calculate the Hessian
H = 2*R;
eig_value = eig(H);
```

```
>> H,eig_value

H =

     2     0
     0     2


eig_value =

     2
     2
```

2. Part 2

a) How many iterations did your code take to converge?

24.

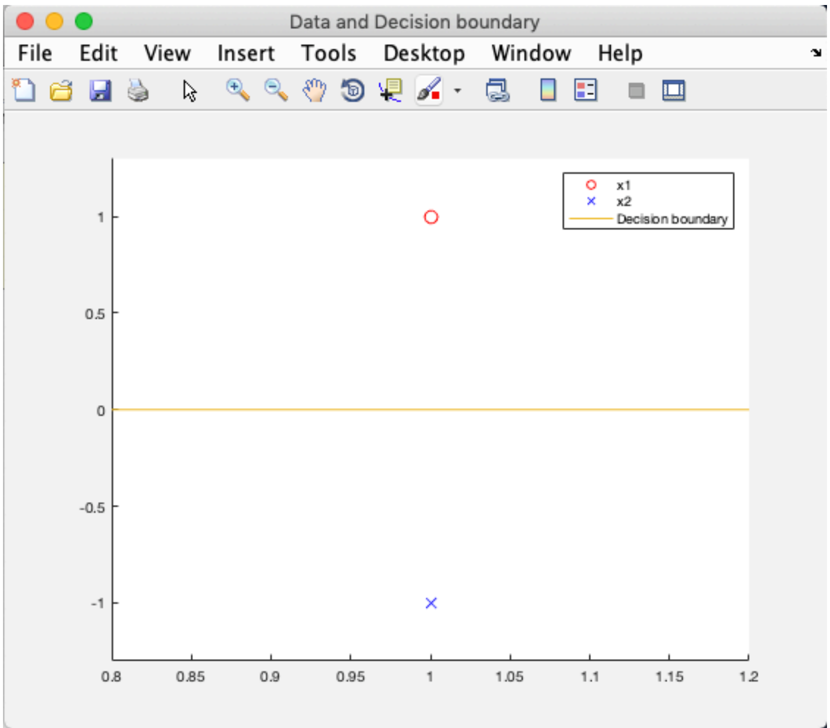
```
After training with alpha = 0.2,
Theta found by training after 24 iterations:
0.00
1.00
```

b) What are the theta values?

[0; 1]

```
After training with alpha = 0.2,
Theta found by training after 24 iterations:
0.00
1.00
```

c) Show a plot of the decision boundary and the example (training) pattern vectors.



3. Part 3 (after correction)

a) Theta.

```
Initializing...
Starting the algorithm...

After training with alpha = 0.2,
Theta found by training after 10000 iterations, which means calculated 10000 * 8 times:
0.02 0.20
-0.57 0.20
-0.02 -0.65
```

b) Norm.

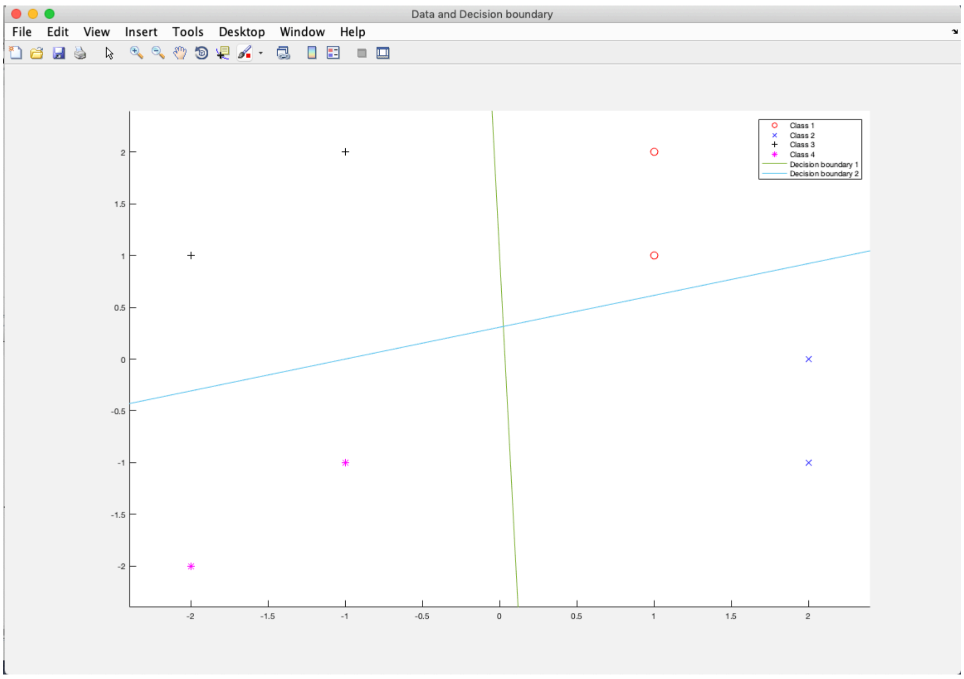
```
The 2-norm of theta is 0.74
```

c) Final predicted output.

Same as t.

```
The predicted output is:
-1.00 -1.00
-1.00 -1.00
-1.00 1.00
-1.00 1.00
1.00 -1.00
1.00 -1.00
1.00 1.00
1.00 1.00
```

d) Plot.



4. Part 4 (after correction)

a) Initial theta.

```
The initial theta is :
1.0000
1.0000
1.0000
```

b) Alpha.

```
The initial learning rate is: alpha = 0.0003
Starting the algorithm...
```

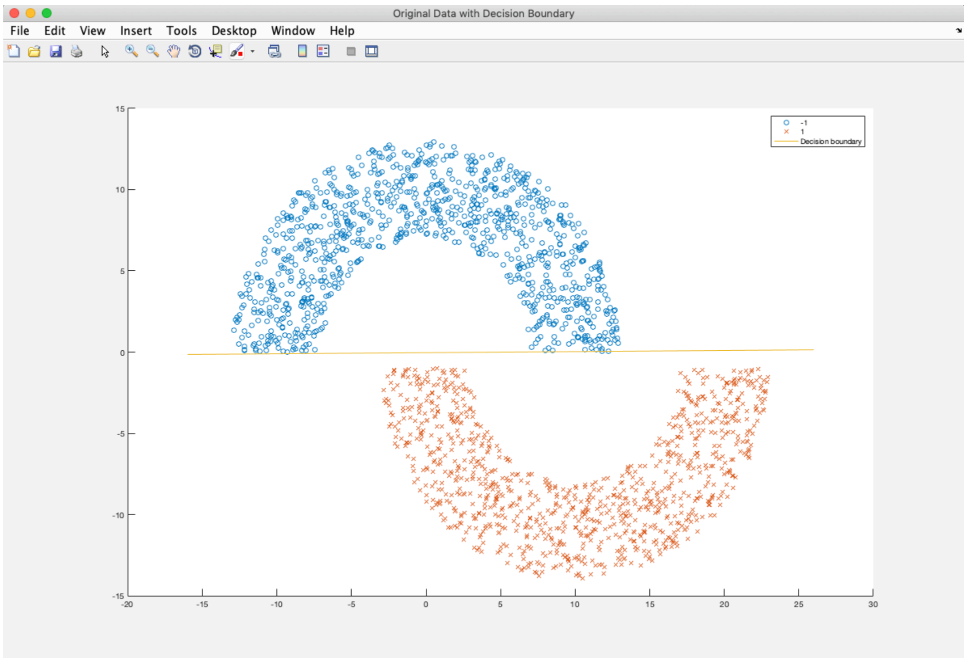
c) Final theta

```
Theta found by training after 40 iterations, which means calculated 40 * 2000 times:
-0.0272
0.0050
-0.7332
```

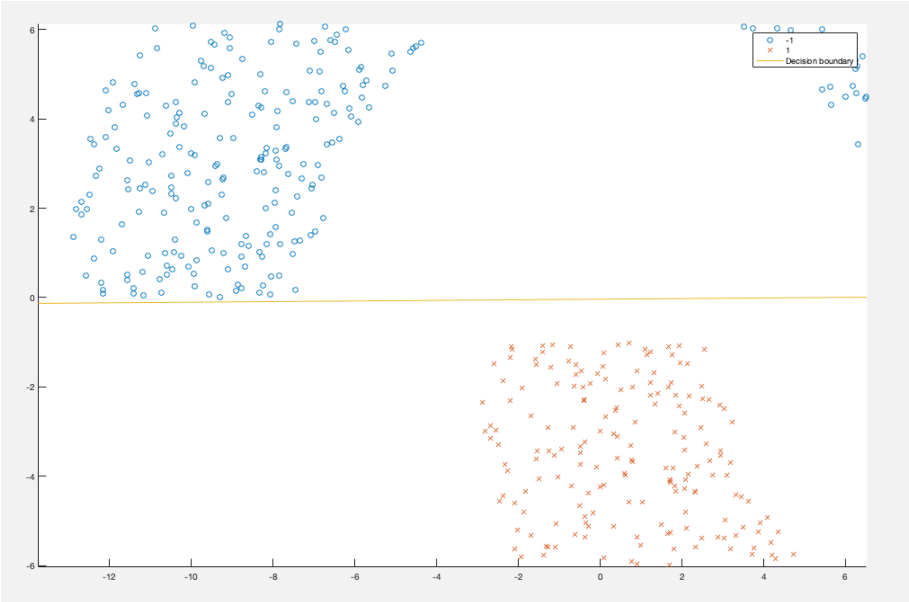
d) Norm of final theta.

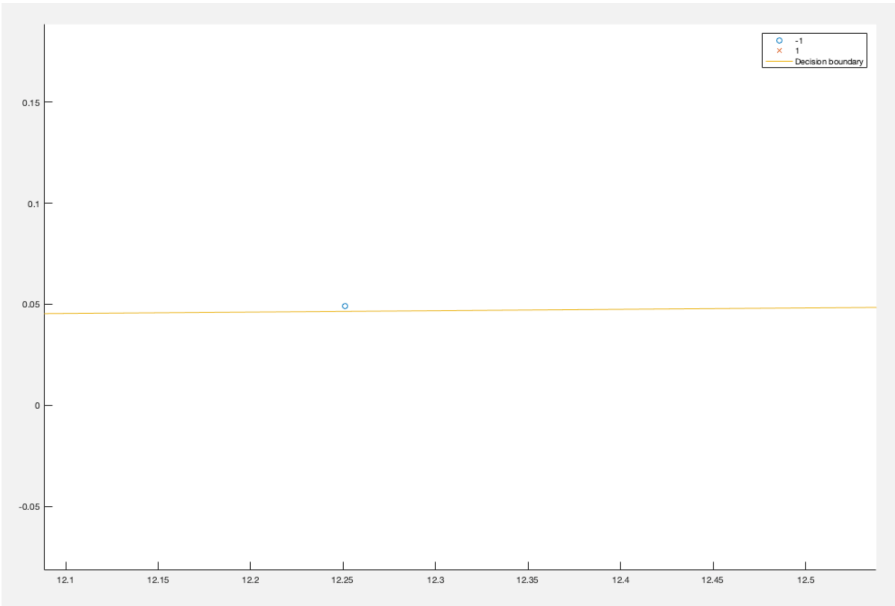
```
The 2-norm of theta is 0.73
```

e) Plot of data with the decision boundary.



If zoom in and take a detail look:

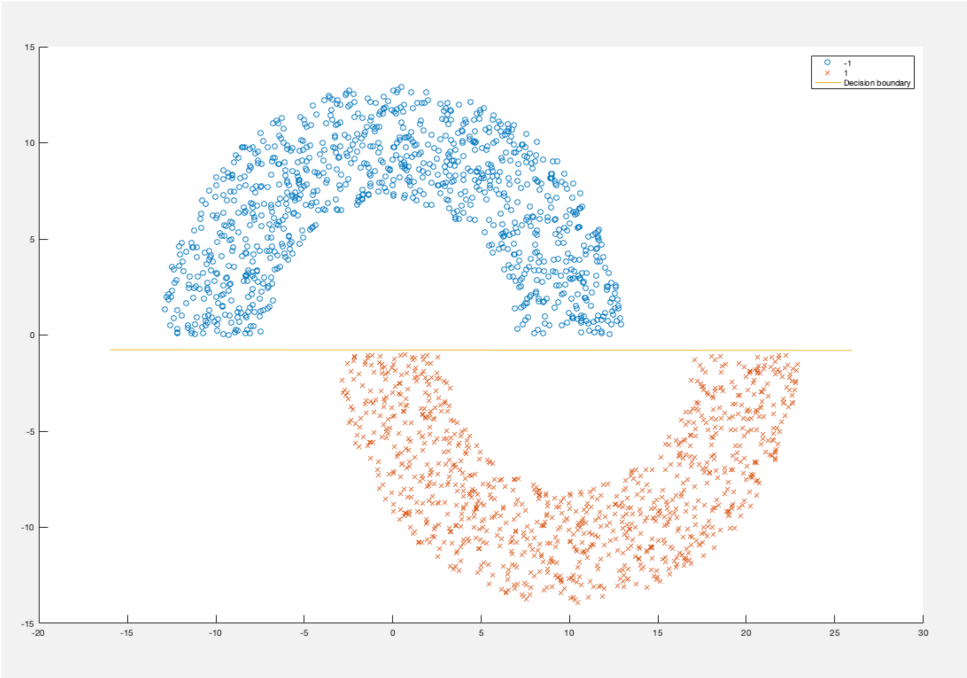




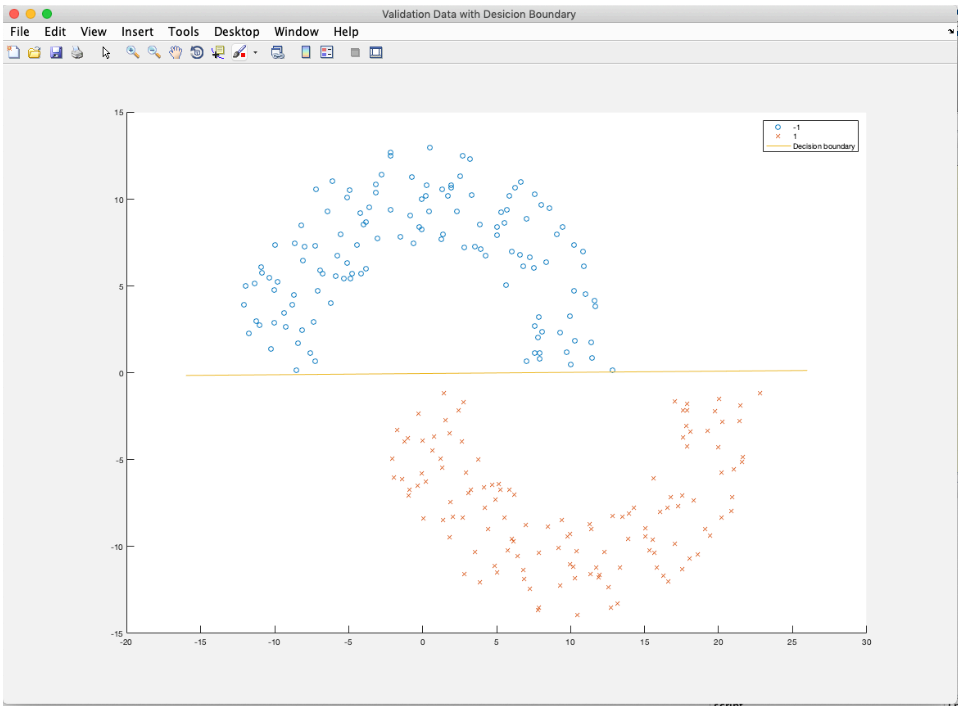
Points of class 0 are all above the decision boundary, which means the error is 0. However, the decision boundary is close to class 0. It’s not good for classification for noise point.

If I put the theta all -1, then the decision boundary is close to class 1. This is because when theta is all 1, the theta converges from positive side, which means the decision boundary moves from class 0 to class 1. Once the error is 0, the training (moving of boundary) stopped and the boundary stayed at class

0 side. Same situation for all negative theta.



f) Plot of *hmtest.mat* with the decision boundary line.



g) Accuracy of classification for *hmtest.mat.*

```
At this time, check if there is non-zero error in e: 0
The accuracy is 100.00
```

5. Code

a) main_part_1_2.m

```
%% Machine Learning Homework 6 part 1 and 2
% Author: Xinrun Zhang
% Time: 04/02/2019 11:04
% =====================================================================

%% Initializing
clear ; close all; clc

fprintf('Initializing...\n')
% Initial the data
x1 = [1; 1];
x2 = [1; -1];
t1 = 1;
t2 = -1;
p = 0.5;

% Calculate c, h and R, where
% c = E[t.^2]
% h = E[t .* x]
% R = E[x * x']
c = p * t1^2 + p * t2^2;
h = p * (t1 * x1) + p * (t2 * x2);
R = p * (x1 * x1') + p * (x2 * x2');


% Calculate the optimal theta
theta_opt = (R^-1)* h;

% Calculate the Hessian
H = 2*R;
eig_value = eig(H);
```

```matlab
% ======================================================================

%% LMS algorithm

% Initial the x and t

x = [1, 1; 1, -1];

t = [1; -1];

m = 2;


% Initial the theta and the learning rate

theta = [0; 0];

alpha = 0.2;


% Initial the iteration and the error

itr = 0;

e = [0;0];



% LMS algorithm

fprintf('Starting the algorithm...\n');

while ( 1 )

    for i = 1:m

        v = x(i,:) * theta;

        e(i) = t(i) - v;

        theta = theta + 2 * alpha * e(i) * x(i,:)';

    end

    itr = itr + 1;

    if ~(any(any(e)))

        break;

    end

end


fprintf('\nAfter training with alpha = 0.2, ')

fprintf('\nTheta found by training after %d iterations:\n', itr);
```

```
fprintf('%.2f\n', theta);
fprintf('\n')
    % =====================================================================

    %% plot the data
a = 0.8:0.1:1.2;
figure('Name','Data and Decision boundary','NumberTitle','off');
scatter(x1(1), x1(2), 80, 'o', 'r');
hold on;
scatter(x2(1), x2(2), 80, 'x', 'b');
hold on;
plot(a, 0*a, '-');
ylim([-1.3, 1.3]);
legend('x1', 'x2','Decision boundary');
    % =====================================================================
```

b) main_part_3.m

```
    %% Machine Learning Homework 6 part 3
    % Author: Xinrun Zhang
    % Time: 04/02/2019 16:52
    % =====================================================================

    %% Initializing
clear ; close all; clc

fprintf('Initializing...\n')
    % Initial the data
x = [1,1; 1,2; 2,-1; 2,0; -1,2; -2,1; -1,-1; -2,-2];
t = [-1,-1; -1,-1; -1,1; -1,1; 1,-1; 1,-1; 1,1; 1,1];
[m, n] = size(x);

    % Initial the theta with bias
theta = [1,1; 1,0; 0,1];
```

```matlab
    %Initial the learning rate
alpha = 0.02;

    % Initial the iteration and
    % initial errors. Set errors to 1 to start the iteration
itr = 10000;
e = zeros(m,2);


    % Data processing
X = [ones(m,1), x(:,1:2)];
    % ======================================================================

    %% LMS algorithm
fprintf('Starting the algorithm...\n');
for iteration = 1:itr
    for i = 1:m
        v = X(i,:) * theta;
        e(i, :) = t(i,:) - v;
        theta = theta + 2 * alpha * X(i,:)' * e(i,:);
    end
end

p = zeros(8,2);
for i = 1:m
    p(i,:) = X(i,:) * theta;
    if p(i,1) >= 0
        p(i,1) = 1;
    else
        p(i,1) = -1;
    end
```

```
    if p(i,2) >= 0

        p(i,2) = 1;

    else

        p(i,2) = -1;

    end

end

fprintf('\nAfter training with alpha = 0.2, ')

fprintf('\nTheta found by training after %d iterations, which means calculated %d * %d
times:\n', itr, itr, m);

fprintf('%.2f %.2f\n', theta');

fprintf('\n')


fprintf('The 2-norm of theta is %.2f\n', norm(theta))

fprintf('\nThe predicted output is:\n')

fprintf('%.2f %.2f\n', p')


      % =====================================================================


      %% plot the data

a = -2.4:0.1:2.4;

figure('Name','Data and Decision boundary','NumberTitle','off');

scatter(x(1:2,1), x(1:2,2), 80, 'o', 'r');

hold on;

scatter(x(3:4,1), x(3:4,2), 80, 'x', 'b');

hold on;

scatter(x(5:6,1), x(5:6,2), 80, '+', 'k');

hold on;

scatter(x(7:8,1), x(7:8,2), 80, '*', 'm');

hold on;

plot(a, (-0.57/0.02)*a + 1, '-');

hold on;

plot(a, (0.2/0.65)*a + 0.2/0.65, '-');

xlim([-2.4, 2.4]);
```

```
ylim([-2.4, 2.4]);
legend('Class 1', 'Class 2', 'Class 3', 'Class 4', 'Decision boundary 1', 'Decision boundary
2');
    % ======================================================================
```

c) main_part_4.m

```
    %% Machine Learning Homework 6 part 4
    % Author: Xinrun Zhang
    % Time: 04/03/2019 14:43
    % ======================================================================

    %% Initializing
clear ; close all; clc

fprintf('Initializing...\n');
    % Import the data
data = importdata('hm.mat');
x = data(:,1:2);
t = data(:,3);
[m,~] = size(x);

    % Initial the theta with bias
    % theta = ones(3,1);
theta = [-1; -1; -1];
fprintf('The initial theta is :\n');
fprintf('%.4f\n', theta);
    % If set the theta as all -1, the decision boundary will
    % converage from the opposite side

    % Initial the learning rate, iteration and error
alpha = 0.0003;
fprintf('The initial learning rate is: alpha = %.4f\n',alpha);
itr = 40;
```

```
e = zeros(m,1);

     % Data processing
X = [ones(m,1) x(:,1:2)];
     t(t==0) = -1; % replace the 0 with -1
     % ====================================================================

     %% LMS algorithm
fprintf('Starting the algorithm...\n');
for iteration = 1:itr
    for i = 1:m
        v = X(i,:) * theta;
        if v >= 0
            v = 1;
        else
            v = -1;
        end
        e(i) = t(i) - v;
        theta = theta + 2 * alpha * e(i) * X(i,:)';
    end
end

fprintf('\nAfter training with alpha = 0.0003, ');
fprintf('\nTheta found by training after %d iterations, which means calculated %d * %d
times:\n', itr, itr, m);
fprintf('%.4f\n', theta);
fprintf('\n');
fprintf('The 2-norm of theta is %.2f\n', norm(theta));
fprintf('At this time, check if there is non-zero error in e: %d\n', any(e));
     % ====================================================================

     %% plot the data
data = sortrows(data,3);
```

```
m = -16:0.1:26;

n = (theta(1)/-theta(3)) + (theta(2)/-theta(3)) * m;

figure('Name','Original Data with Decision Boundary','NumberTitle','off');

scatter(data(1:1000,1), data(1:1000,2), 'o');

hold on;

scatter(data(1001:2000,1), data(1001:2000,2),'x');

hold on;

plot(m,n,'-')

legend('-1', '1', 'Decision boundary');

     % ======================================================================


     %% Validation

     % Import test data

data_val = importdata('hmtest.mat');

x_val = data_val(:,1:2);

t_val = data_val(:,3);

[m_val,~] = size(x_val);


     % Data processing

X_val = [ones(m_val,1) x_val(:,1:2)];

t_val(t_val == 0) = -1;


     % Validation

predict = X_val * theta;

predict(predict >= 0) = 1;

predict(predict < 0) = -1;


accuracy = mean( double(predict == t_val) * 100);

fprintf('The accuracy is %.2f\n', accuracy);

     % ======================================================================


     %% Plot

data_val = sortrows(data_val,3);
```

```
figure('Name','Validation Data with Desicion Boundary','NumberTitle','off');

scatter(data_val(1:130,1), data_val(1:130,2), 'o');

hold on;

scatter(data_val(131:260,1), data_val(131:260,2),'x');

hold on;

plot(m,n,'-')

legend('-1', '1', 'Decision boundary');

     % ======================================================================
```