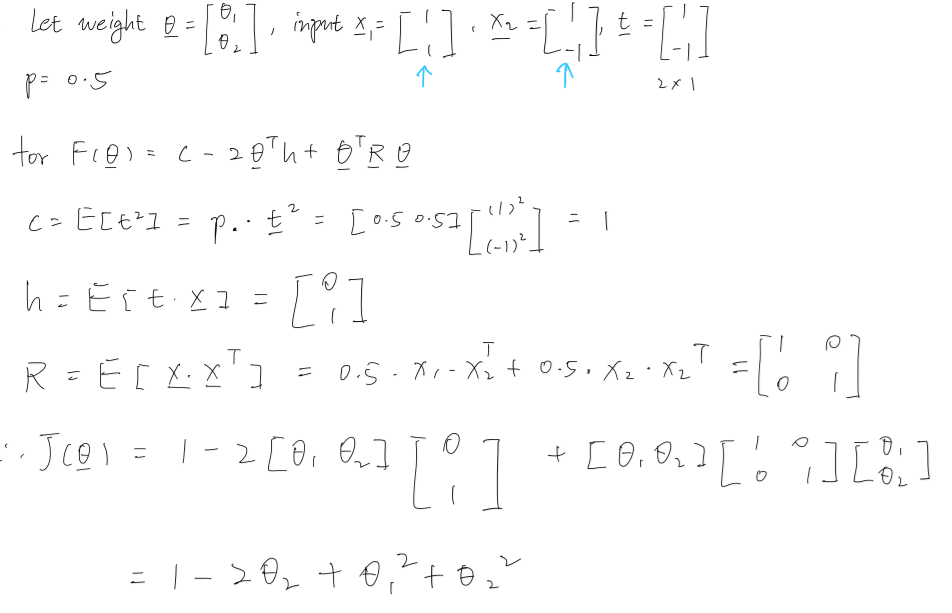
Name: Xinrun Zhang

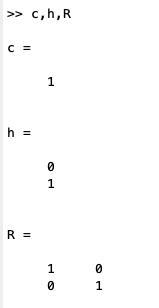
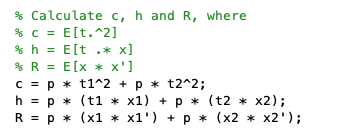
ML Homework 6 Report

Instructor: Prof. Kaliappan Gopalan

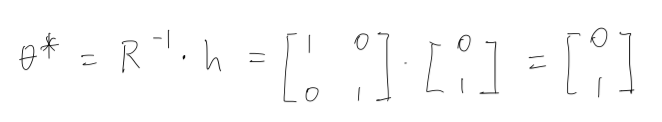
Time: 04/03/2019

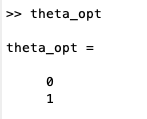
1. Part 1
2. Calculate the quadratic cost function.



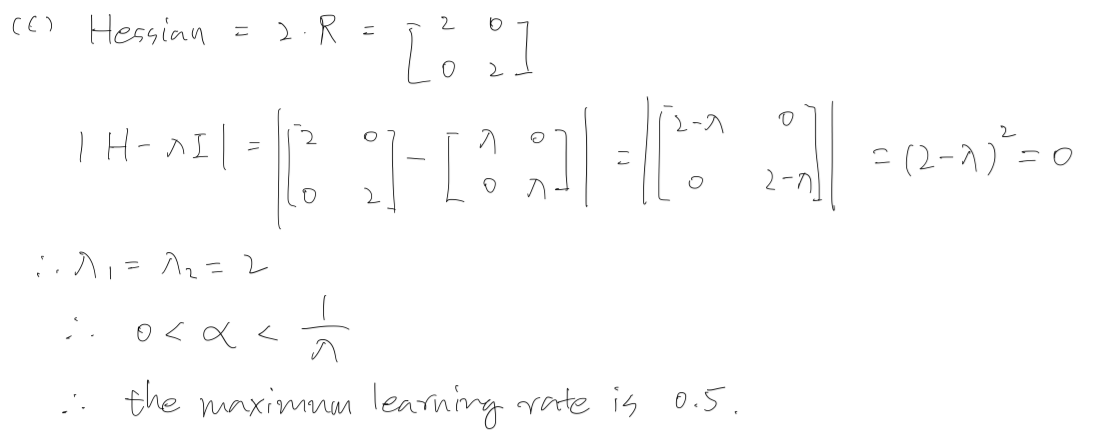


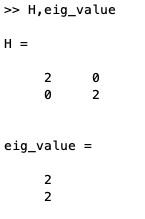
1. Obtain the optimal weight vector.





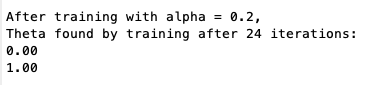
1. From the hessian, what is the maximum learning rate for iterative implementation of the LMS algorithm?





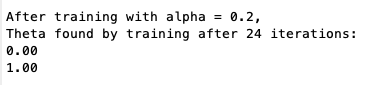
1. Part 2
2. How many iterations did your code take to converge?

24.

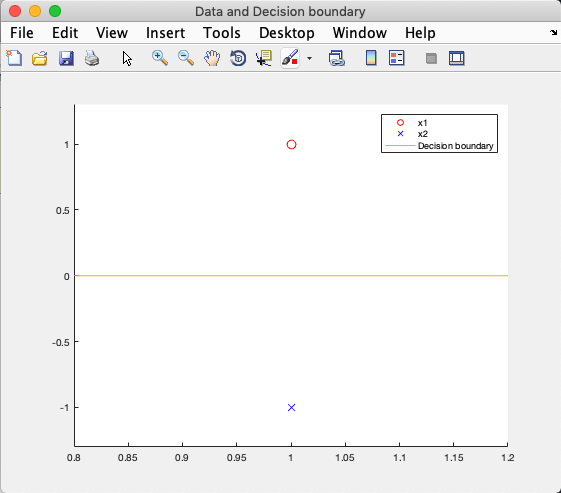


1. What are the theta values?

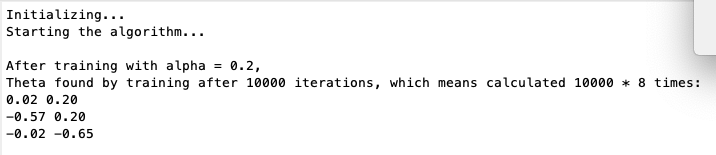
[0; 1]



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



1. Part 3 (after correction)
2. Theta.

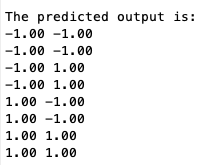


1. Norm.

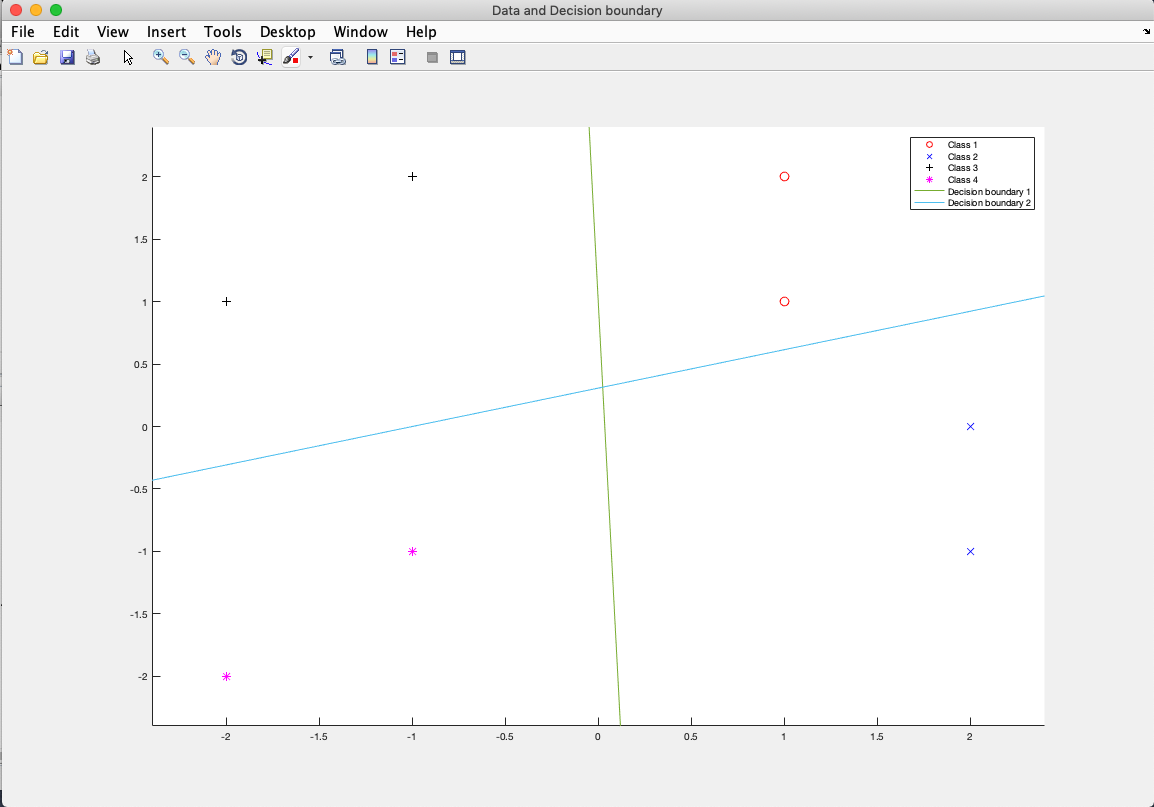


1. Final predicted output.

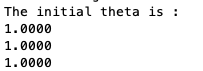
Same as t.



1. Plot.



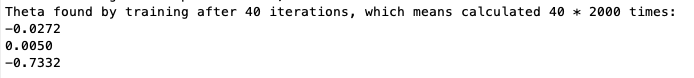
1. Part 4 (after correction)
2. Initial theta.



1. Alpha.



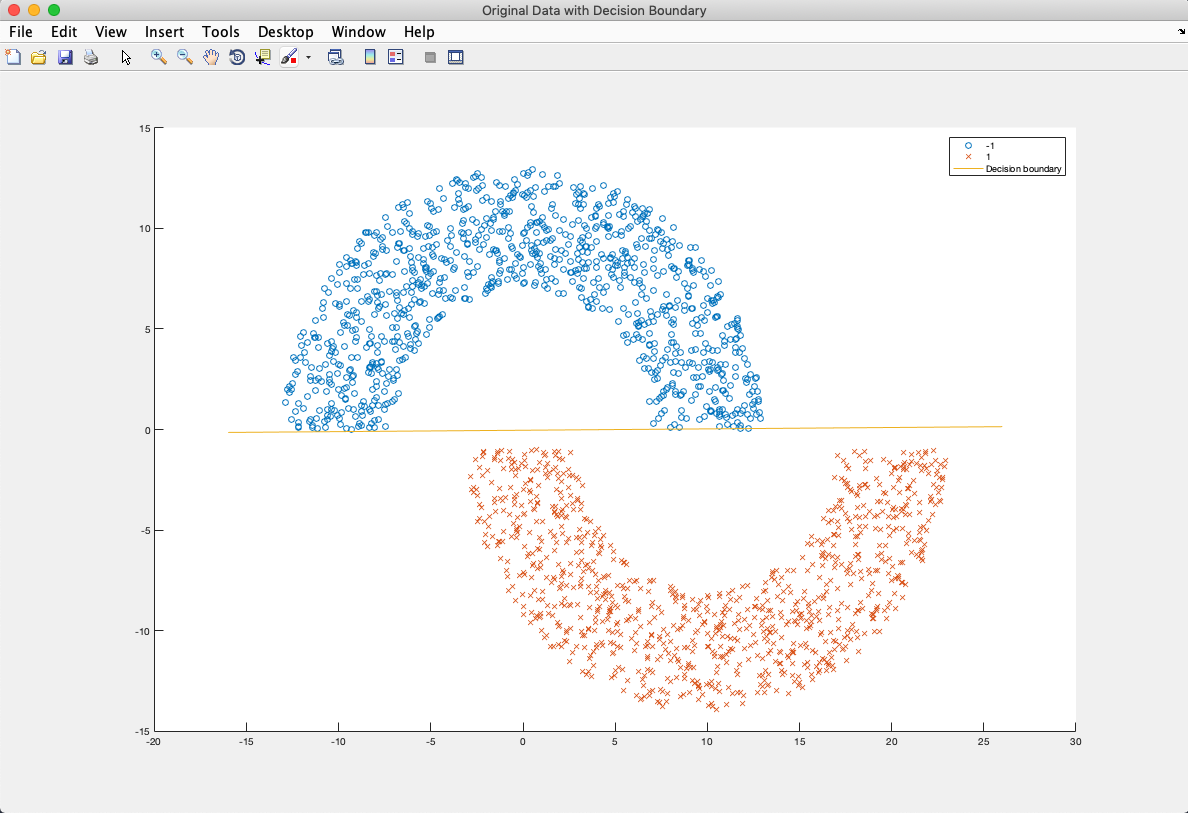
1. Final theta



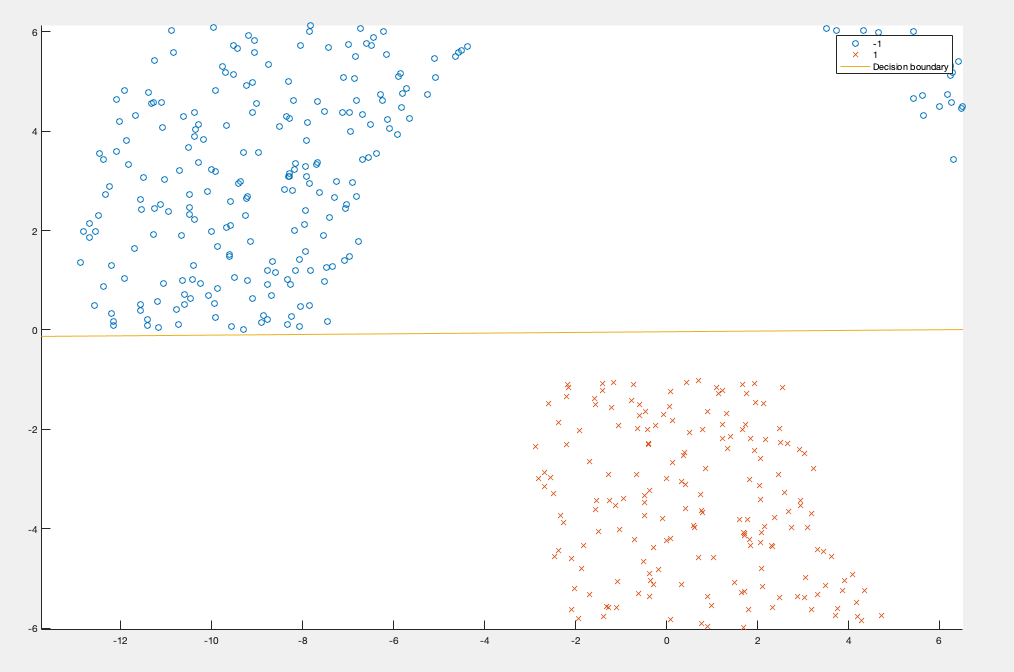
1. Norm of final theta.

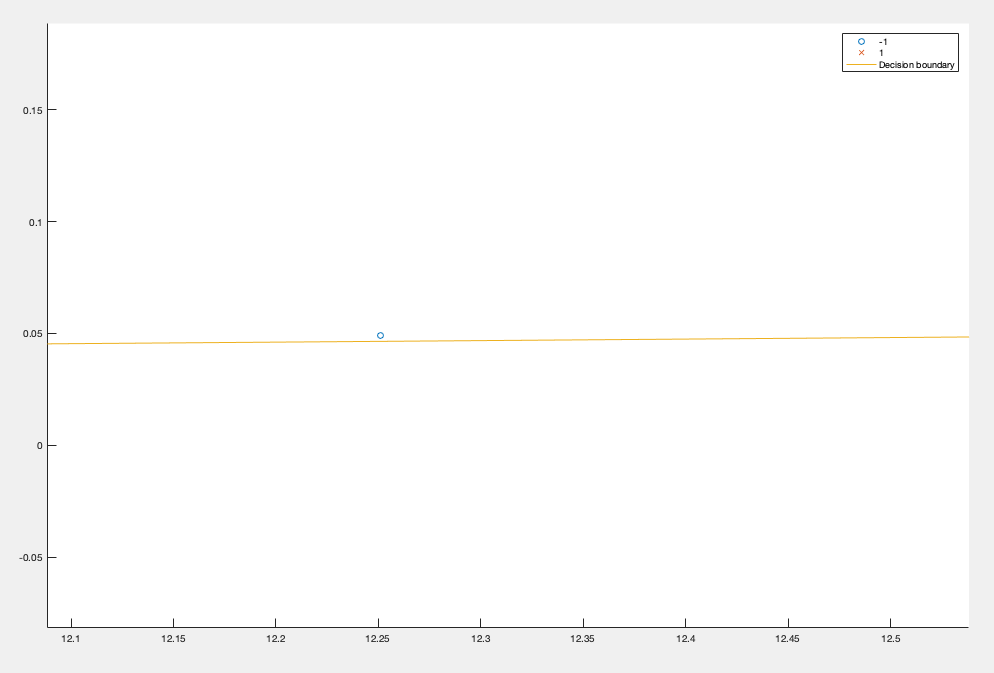


1. Plot of data with the decision boundary.



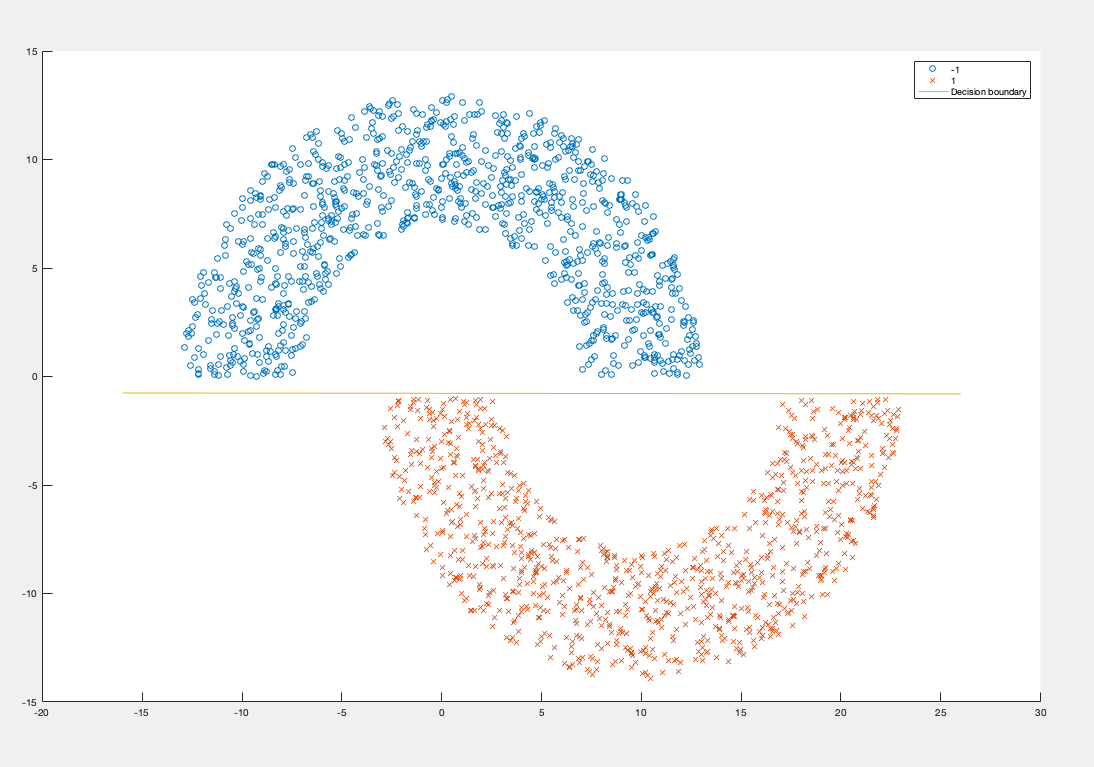
If zoom in and take a detail look:



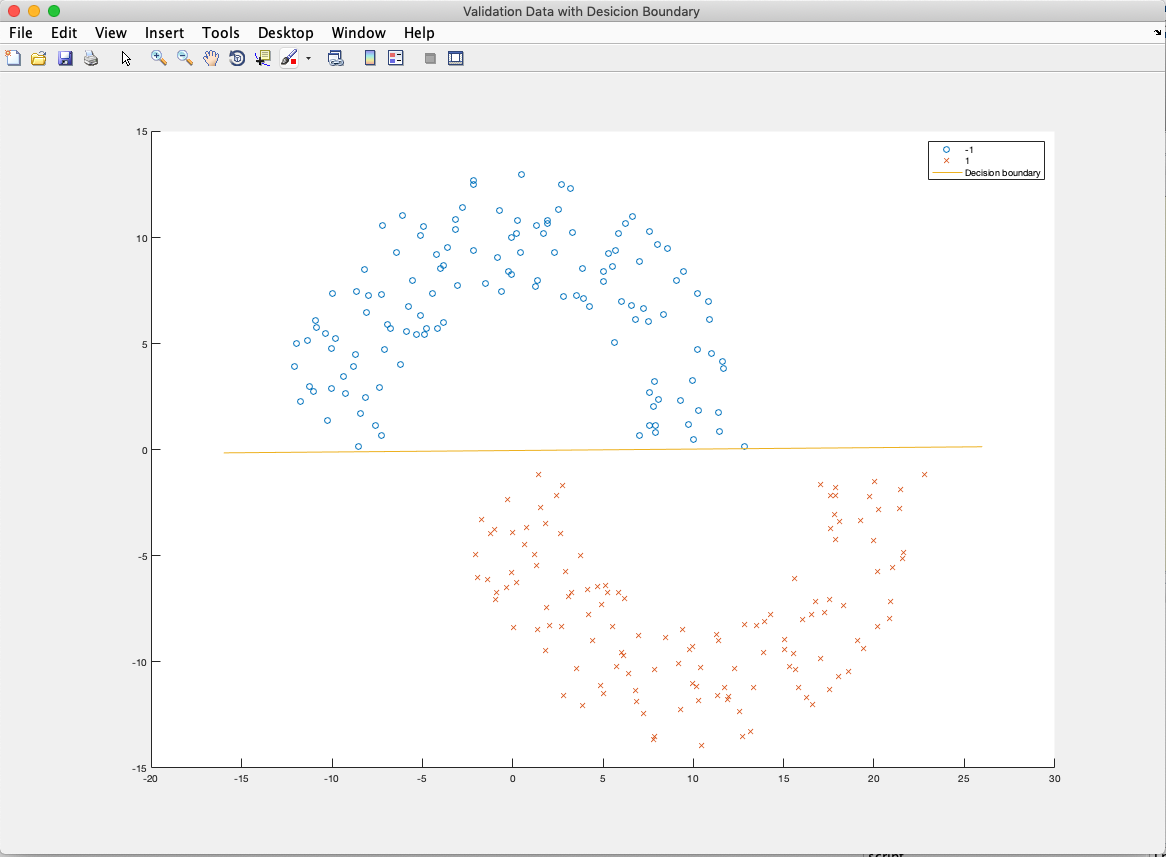


Points of class 0 are all above the decision boundary, which means the error is 0. However, the decision boundary is close to class 0. It’s not good for classification for noise point.

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.



1. Plot of *hmtest.mat* with the decision boundary line.



1. Accuracy of classification for *hmtest.mat.*



1. Code
2. 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);

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

%% 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');

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

1. 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];

%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');

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

1. 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');

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