**ENGN2520 Homework 2**

**Ming Xu (Banner ID: B01532164)**

**Problem1**

Because x is a random variable with uniform distribution on [*a,b*], then the pmf of x is:

.

Therefore, the likelihood function on the training set can be written as:

Denote the log likelihood as

According to the derivatives of the log-likelihood function, is monotonically increasing. Therefore, the maximum likelihood estimator for should be the largest possible of .

Similarly, the derivatives of the log-likelihood function, is monotonically decreasing. Therefore, the maximum likelihood estimator for should be the largest possible of .

**Problem2**

***(a)***

We know that:

Let be the sample from training set ***A***, and be the sample from training set ***B***

The likelihood function can be written as:

Denote the log likelihood as ,

Where is the sum of squared differences over training set ***A*** and is the sum of squared differences over training set ***B.***

Therefore,

***(b)***

Because ,

So we search for the w such that

This is equivalent to the set of equations

Since, for , the matrix ***M*** and vector ***Z*** are:

Similarly, for , the matrix ***M*** and vector ***Z*** are:

Combining the result above, the final matrix ***M*** and vector ***Z*** are:

Therefore, can be computed by solving the linear system.

***(c)***

From the mathematical model above, they shouldn't ignore Bob's measurements in estimating their function. They can incorporate by combining their test results using a weighted sum method.

***(d)***

If we don't know the and , we can suppose . Then use the mean of and to calculate , and use the mean of and to calculate .

**Problem3**

***(a)***

The degree 2 polynomials that estimated by least square regression and least absolute deviations regression are shown in the figure below.

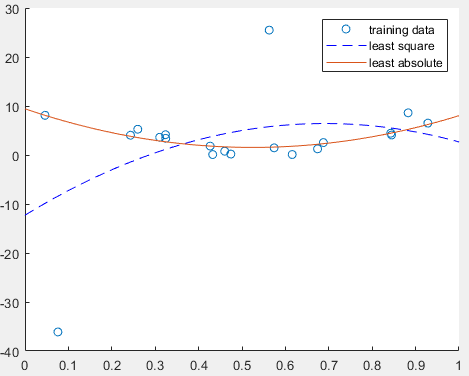


Fig.1. The degree 2 polynomials that estimated from training data

In the figure, the orange solid line indicates the degree 2 polynomials estimated by least absolute deviations regression, while the blue dash line indicates the degree 2 polynomials estimated by least square regression.

***(b)***

The degree 4 polynomials that estimated by least square regression and least absolute deviations regression are shown in the figure below.

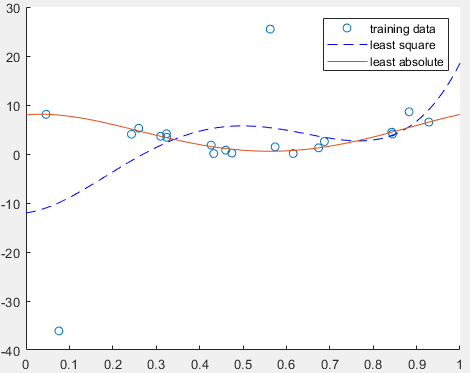


Fig.2. The degree 4 polynomials that estimated from training data

In the figure, the orange solid line indicates the degree 4 polynomials estimated by least absolute deviations regression, while the blue dash line indicates the degree 4 polynomials estimated by least square regression.

***(c)***

Based on the given experiments, least absolute deviations regression is more robust than the least square regression when outliners exist.

***(d)*** Source code

1. Function to calculate w from training set by least square regression: “solveW.m”

function [w] = solveW(x,y,deg)

%initialization

M = zeros(deg+1,deg+1);

Z = zeros(deg+1,1);

%contruct matrix M

for row = 1:deg+1

for col = 1:deg+1

M(row,col) = sum((x.^(row-1)).\*(x.^(col-1)));

end

end

%construct matrix Z

for row = 1:deg+1

Z(row,1) = sum(y.\*(x.^(row-1)));

end

%calculate w

w = M\Z;

end

2. Function to calculate fx by giving x and w: “calculateFxByGivingW.m”

function [fx] = calculateFxByGivingW(x,w)

%get degree

[deg, col] = size(w);

%get size

[rowNum, col] = size(x);

%initialize fx

fx = zeros(rowNum, col);

%calculate fx by usuing polynomial function defined by w

for row = 1:rowNum

for i = 1:deg

fx(row,1) = fx(row,1) + w(i)\*x(row,1)^(i-1);

end

end

end

3. Function to calculate w from training set by least absolute derivations regression: “solveWByLinearProgramming.m”

function [w] = solveWByLinearProgramming(x,y,deg)

%get size

[rowNum, col] = size(x);

deg = deg + 1;

%initialization

A = zeros(2\*rowNum, rowNum+deg);

b = zeros(2\*rowNum,1);

C = zeros(1,rowNum+deg);

%generate C

index = deg+1:rowNum+deg;

C(index) = 1;

%generate b

index = 1:2:2\*rowNum-1;

b(index) = y;

index = 2:2:2\*rowNum;

b(index) = -y;

%generate A

for row = 1:2:2\*rowNum-1

for col = 1:deg

A(row,col) = x(ceil(row/2),1).^(col-1);

end

A(row,deg+ceil(row/2))=-1;

end

for row = 2:2:2\*rowNum

A(row,:) = -A(row-1,:);

A(row,deg+ceil(row/2))=-1;

end

%calculate w by linear program

result = linprog(C,A,b);

w = result(1:deg);

end

4. Main function: “main.m”

%loal data

load Xtrain

load Ytrain

%plot showing the training data and degree 2 polynomial estimated from the

%data

w1 = solveW(Xtrain,Ytrain,4);

w2 = solveWByLinearProgramming(Xtrain,Ytrain,4);

scatter(Xtrain,Ytrain,'DisplayName','training data');

hold on;

x = 0:0.01:1;

x = x';

fx1 = calculateFxByGivingW(x,w1);

fx2 = calculateFxByGivingW(x,w2);

plot(x,fx1,'b--','DisplayName','least square');

plot(x,fx2,'DisplayName','least absolute');

legend;