1 Problem 1

Consider the probability density function for a uniform distribution over an interval \([a, b]\):

\[
p(x | a, b) = \begin{cases} \frac{1}{b-a} & x \in [a, b] \\ 0 & otherwise \end{cases}
\]

Now consider the likelihood of our observed training set \((T)\) given an interval:

\[
p(T | a, b) = \prod_{x_i \in T} p(x_i | a, b)
\]

Finally consider the following interval \((I)\):

\[
I = [\min_{x_i \in T} x_i, \max_{x_i \in T} x_i]
\]

In other words, \(I\) is the interval from the minimum observation in \(T\) to the maximum observation in \(T\). We claim that \(I\) is the maximum likelihood estimator of our unknown interval.

We can (informally) show that this interval is the maximum likelihood estimator by arguing that if an interval \(I' = [a', b']\) is the maximum likelihood estimator, it must be the case that \(I' = I\).

First note that if any observation \(x_i \in T\) is outside our interval estimate \((x_i > b' \text{ or } x_i < a')\), then \(p(x_i | a', b') = 0\) and therefore the likelihood of the training set will be 0. It follows that if \(I'\) is the maximum likelihood estimator, then it must be the case that:

\[
a' \leq \min_{x_i \in T} x_i
\]

\[
b' \geq \max_{x_i \in T} x_i
\]

Now consider the case where \(b' > \max_{x_i \in T} x_i\). In this case the likelihood for every observation in the interval will be:

\[
p(x_i | a', b') = \frac{1}{b' - a'} < \frac{1}{\max_{x_i \in T} x_i - a'}
\]
This implies that we could increase the likelihood of each observation by setting $b' = \max_{x_i \in T} x_i$. It follows that in the maximum likelihood estimator $b = \max_{x_i \in T} x_i$. Using the corresponding logic for $a$, we see that the maximum likelihood estimator must also set $a = \min_{x_i \in T} x_i$. Therefore the interval $I$, as defined above, must be the maximum likelihood estimator.

2 Problem 2

2.1 Part a

Recall that our maximum likelihood estimate of $w$ defined as:

$$w_{ML} = \max_w p(T_A, T_B | w)$$

$$= \min_w - \log p(T_A, T_B | w)$$

We’ll start by writing out our negative log-likelihood explicitly and then we’ll simplify our expression to get the result we’re looking for.

$$- \log p(T_A, T_B | w) = \left( \sum_{x_a, y_a \in T_A} - \log p(y_a | x_a, w) \right) + \left( \sum_{x_b, y_b \in T_B} - \log p(y_b | x_b, w) \right)$$

Our negative log-likelihood for a given observation is:

$$- \log p(y | x, w) = \frac{(y - w^T \phi(x))^2}{2\sigma^2} - \log \left( \frac{1}{\sqrt{2\pi \sigma^2}} \right)$$

Noting that the variance of errors is different for observations in $T_A$ and observations in $T_B$, we can formulate our maximum likelihood estimator as follows:

$$\min_w - \log p(T_A, T_B | w) =$$

$$= \min_w \left( \sum_{x_a, y_a \in T_A} \frac{(y_a - w^T \phi(x_a))^2}{2\sigma_A^2} - \log \left( \frac{1}{\sqrt{2\pi \sigma_A^2}} \right) \right) + \left( \sum_{x_b, y_b \in T_B} \frac{(y_b - w^T \phi(x_b))^2}{2\sigma_B^2} - \log \left( \frac{1}{\sqrt{2\pi \sigma_B^2}} \right) \right)$$

As in our original linear regression derivation, we can drop terms that do not affect the optimal value of $w$:

$$= \min_w \frac{1}{2\sigma_A^2} \left( \sum_{x_a, y_a \in T_A} (y_a - w^T \phi(x_a))^2 \right) + \frac{1}{2\sigma_B^2} \left( \sum_{x_b, y_b \in T_B} (y_b - w^T \phi(x_b))^2 \right)$$

We see that this is the minimum of a weighted sum of squares.
2.2 Part b

To solve for the maximum likelihood estimator of $w$ we once again take the derivative of our likelihood objective with respect to each entry of $w$ and solve for the values that make this set of derivatives equal 0.

\[
\frac{\partial}{\partial w_i} = \frac{1}{\sigma_A^2} \left( \sum_{x_a,y_a \in T_A} (y_a - w^T \phi(x_a)) \phi_i(x_a) \right) + \frac{1}{\sigma_B^2} \left( \sum_{x_b,y_b \in T_B} (y_b - w^T \phi(x_b)) \phi_i(x_b) \right) = 0
\]

Rearranging terms:

\[
\sum_{x_a,y_a \in T_A} \frac{1}{\sigma_A^2} w^T \phi(x_a) \phi_i(x_a) + \sum_{x_b,y_b \in T_B} \frac{1}{\sigma_B^2} w^T \phi(x_b) \phi_i(x_b) = 0
\]

Expanding $w^T \phi(x)$:

\[
= \sum_{x_a,y_a \in T_A} \left( \frac{1}{\sigma_A} \phi_i(x_a) \sum_{j=1}^m w_j \phi_j(x_a) \right) + \sum_{x_b,y_b \in T_B} \left( \frac{1}{\sigma_B} \phi_i(x_b) \sum_{j=1}^m w_j \phi_j(x_b) \right)
\]

Finally, swapping the order of summation we get:

\[
= \sum_{j=1}^m w_j \left( \sum_{x_a,y_a \in T_A} \left( \frac{1}{\sigma_A} \phi_i(x_a) \phi_j(x_a) \right) + \sum_{x_b,y_b \in T_B} \left( \frac{1}{\sigma_B} \phi_i(x_b) \phi_j(x_b) \right) \right)
\]

Denoting:

\[
\mathcal{M}_{i,j} = \sum_{x_a,y_a \in T_A} \left( \frac{1}{\sigma_A} \phi_i(x_a) \phi_j(x_a) \right) + \sum_{x_b,y_b \in T_B} \left( \frac{1}{\sigma_B} \phi_i(x_b) \phi_j(x_b) \right)
\]

\[
z_i = \sum_{x_a,y_a \in T_A} \frac{1}{\sigma_A} y_a \phi_i(x_a) + \sum_{x_b,y_b \in T_B} \frac{1}{\sigma_B} y_b \phi_i(x_b)
\]

We can solve for $w$ by solving the linear system:

\[
\mathcal{M} w = z
\]

2.3 Part c

Q1: This derivation of the maximum likelihood estimator suggests that Bob’s measurements should not be ignored because of their larger error variance. Rather our derivation shows that both Alice and Bob’s measurements need to be accounted for in order to calculate the maximum likelihood estimator over both datasets.
Our result shows that the optimal (from a ML perspective) way to incorporate both Alice and Bob’s measurements is to weight each group’s sum of squared errors by the inverse of the group’s error variance. Intuitively this makes sense because we would like to assign more significance to observations that we are more confident about.

2.4 Part d

From our derivation above we see that both $\sigma_A^2$ and $\sigma_B^2$ are needed in order to find the maximum likelihood estimate for $w$. A natural question in the case where $\sigma_A^2$ and $\sigma_B^2$ unknown is whether we can estimate these two values. A reasonable approach is to extend our maximum likelihood estimator to optimize over $\sigma_A^2$ and $\sigma_B^2$, along with $w$, giving us the following setup:

$$\max_{w, \sigma_A^2, \sigma_B^2} p(T_A, T_B | w)$$

Using the negative log-likelihood we derived above, we can write this as:

$$\min_{w, \sigma_A^2, \sigma_B^2} \left( \sum_{x_a, y_a \in T_A} \frac{(y_a - w^T \phi(x_a))^2}{2\sigma_A^2} - \log \left( \frac{1}{\sqrt{2\pi \sigma_A^2}} \right) \right) + \left( \sum_{x_b, y_b \in T_B} \frac{(y_b - w^T \phi(x_b))^2}{2\sigma_B^2} - \log \left( \frac{1}{\sqrt{2\pi \sigma_B^2}} \right) \right)$$

There are multiple ways we could approach this optimization problem. One simple approach is to alternate between minimizing with respect to $w$ and minimizing with respect to $\sigma_A$ and $\sigma_B$ (this is known as coordinate descent). We already know how to minimize with respect to $w$, given fixed $\sigma_A^2$ and $\sigma_B^2$. We can find the minimum with respect to $\sigma_A^2$ (or $\sigma_B^2$) by taking the derivative with respect to $\sigma_A^2/B$ and setting it equal to 0. We see that the optimal value of $\sigma_A^2/B$ is simply the sample variance of the errors of $T_A/B$ given a fixed $w$:

$$\sigma_A^2 = \frac{1}{|T_A|} \sum_{x_a, y_a \in T_A} (y_a - w^T \phi(x_a))^2$$

We can repeat this process of alternating between updating $w$, $\sigma_A$ and $\sigma_B$ until the estimates converge.

Another interesting question is how to initialize $w$ (or $\sigma_A$ and $\sigma_B$) for this iterative process. One reasonable approach might be to initially estimate $w$ assuming $\sigma_A = \sigma_B$, which gives us the standard least squares estimate as a starting point.
3 Question 3

3.1 Part a
3.2 Part b

Based on the results shown above we see that our two regression methods produce drastically different results on data with large outliers. We see that the least absolute deviations method closely fits the majority of the data while “ignoring” outliers. The least squares method on the other hand is heavily influenced by the presence of outliers.

3.3 Part c

Based on the results shown above we see that our two regression methods produce drastically different results on data with large outliers. We see that the least absolute deviations method closely fits the majority of the data while “ignoring” outliers. The least squares method on the other hand is heavily influenced by the presence of outliers.

3.4 Code

```matlab
function [ phi ] = poly_basis( X , d )
%Applies polynomial basis functions up to
% degree d for the input data X.
%   X: Input   (N x 1)
%   d: degree
%   phi: Transformed data (N x d+1)

N = size(X,1);
phi = ones(N, d+1);
for i = 1:(d)
    phi(:, i+1) = X .^ i;
```
function [ w ] = lad_poly_regression( X, Y, d )
%Least absolute deviation regression using polynomial basis functions with degree up to d.
% %
% X: Training data
% Y: Training labels
% d: Degree of polynomial basis functions
% w: Output weight vector

phi = poly_basis(X, d);
m = d + 1;
n = size(X, 1);

C = [zeros(m, 1); ones(n, 1)];
b = [Y; -Y];

A = [phi, -eye(n); -phi, -eye(n)];

xi = linprog(C, A, b);
w = xi(1:m);
end

function [ w ] = poly_regression( X, Y, d )
%Linear regression using polynomial basis functions with degree up to d.
% %
% X: Training data
% Y: Training labels
% d: Degree of polynomial basis functions
% w: Output weight vector

phi = poly_basis(X, d);
M = phi' * phi;
z = phi' * Y;
w = M \ z;
end

function [ Y ] = poly_predict( X, w )
%Predict labels for data X using the weight vector w.
% %
% X: Input data
% w: Weight vector
% Y: Predicted output labels

d = size(w,1) - 1;
phi = poly_basis(X, d);
Y = phi * w;

end

function [ ] = lad_poly_plot( wlad, wls, Xtrain, Ytrain )
% Plot polynomial functions defined by the weight vectors wlad and wls.
% wlad: Trained weight vector using least absolute deviations
% wls: Trained weight vector using least squares regression
% Xtrain: Training data (optional)
% Ytrain: Training labels (optional)

d = size(wlad, 1) - 1;

figure;
hold on
title(sprintf('Polynomial basis of degree: %d', d));

X = linspace(0, 1, 200)';
Ylad = poly_predict(X, wlad);
Ylr = poly_predict(X, wls);

plot(X, Ylad);
plot(X, Ylr, '--');
if nargin > 1
    scatter(Xtrain, Ytrain);
end
hold off

legend('LAD Regression','Least Squares', 'Training Data')

end

% Script to run all trials
load('Xtrain.mat')
load('Ytrain.mat')

for d = [2, 4]
    wlad = lad_poly_regression(Xtrain, Ytrain, d);
    wls = poly_regression(Xtrain, Ytrain, d);

    lad_poly_plot(wlad, wls, Xtrain, Ytrain);
end