NEVIN LIANG ML HW 2

1. $X^T X \omega = X^T y \quad X \in \mathbb{R}^{N \times M} \quad Y \in \mathbb{R}^N$

Prove: Gram matrix $X^T X$ = nonsingular iff X has lin. indep columns.

Case I: X has lin indep columns. Thus, ~~AX=0~~ ~~for~~ $Xx = 0$ for only $x=0$

$X^T \cdot (X \cdot x) = X^T \cdot 0 = 0$

$X^T (X \cdot x)$ will only be 0 when $x = 0$ b/c X^T is non-zero.

$X^T (X \cdot x) = (X^T \cdot X) \cdot x$ and therefore $X^T \cdot X$ is singular

Case II: If $X^T X$ is nonsingular then X has linearly independent columns.

if $X^T X$ is nonsingular $\Rightarrow (X^T \cdot X) x = 0$ has only the trivial solution $x=0$

$X^T \cdot (X \cdot x) = 0$ has only the trivial solution $x = 0$

~~X^T b/c both sides~~ if $x=0$, $X \cdot x$ must be 0 for the equation to hold true.

thus, X^T is nonsingular. the transpose of a nonsingular matrix

is nonsingular, so X is nonsingular. Thus, X has linearly indep columns.

2. a) $H = X(X^T X)^{-1} X^T$

$H^T = X \left(\left((X^T X)^{-1} \right)^T \right) X^T$

$= X \left(\left((X^T X)^T \right)^{-1} \right) X^T$

$= X (X^T X)^{-1} X^T$ ✓

(b) $H^2 = X(X^T X)^{-1} X^T X (X^T X)^{-1} X^T$

$= X \cdot X^{-1} (X^T)^{-1} X^T X \cdot X^{-1} (X^T)^{-1} \cdot X^T$

$= X \cdot (X^T X)^{-1} \cdot X^T$

$H^2 = H$ b/c $H \cdot H = H$

by induction, $H^k = H$

(c) $(I-H)(I-H) = I^2 - H \cdot I - I \cdot H + H^2$

$= I - H - H + H^2$

$= I - 2 \cdot H + H = I - H$

thus, by induction, $(I-H)^k = I - H$

(d) $H = X(X^T X)^{-1} X^T$

$\text{Trace}(H) = \text{Trace}$

$\text{Trace}(X^T \cdot X \cdot (X^T X)^{-1}) = I$

but $I = M \times M$ so $\text{trace}(H) = M$

3. a) $\frac{\partial J}{\partial w_0} = \sum_{n=1}^{N} a_n \left(2 \cdot (w_0 + w_1 x_n - y_n) \cdot 1 \right)$

$= 2 \sum_{n=1}^{N} (w_0 + w_1 x_n - y_n) \cdot a_n$

$\frac{\partial J}{\partial w_1} = \sum_{n=1}^{N} a_n \left(2 (w_0 + w_1 x_n - y_n) \cdot x_n \right)$

(b) if $a_i = 0$ for some i $[1, N]$, then that point is not part of calculating the cost func. essentially, it's ok if the point is far from the line.

(c) if one of the a_i is much greater, then the line created will have to be very close. the gradient will be minimized when (w_0, w_1) are close the the i^{th} point.

4. $\frac{dJ}{dw} = \frac{d}{dw}\left(\frac{1}{2}\|w\|^2 + C \cdot \frac{1}{n} \max(0, 1 - y_i (w^T x_i)) \right)$

if $\max = 0$

$$\frac{dJ}{dw} = \frac{1}{2} \cdot \frac{d}{dw}(w \cdot w) + \frac{C}{n} \cdot 0$$

$$= \frac{1}{2} \cdot \left(\vec{w} \cdot \frac{dw}{dw} + \frac{dw}{dw} \cdot \vec{w} \right) + 0$$

$$= 2w \cdot \frac{1}{2} = \vec{w}$$

$$w_{t+1} = w_t - \eta \cdot \frac{\vec{w}}{\|\vec{w}\|}$$

if $\max = 1 - y_i (w^T x_i)$

$$\cancel{\frac{dJ}{dw} = \vec{w} + \frac{C}{n} \cdot \frac{d}{dw}\left(y_i (w^T x_i)\right)}$$

$$\cancel{= \vec{w} + \frac{C y_i}{n} \cdot \left(\frac{d x_i^T w^T}{dw} \right)}$$

$$\cancel{= \vec{w} + \frac{C}{n}(y_i \cdot x_i^T)}$$

$$\cancel{w_{t+1} = w_t - \eta \frac{\vec{w} + \frac{C}{n}(y_i \cdot x_i^T)}{\|\vec{w} + \frac{C}{n}(y_i \cdot x_i^T)\|}}$$

Made a mistake
- forgot negative sign.

$$\frac{dJ}{dw} = \vec{w} + \frac{C}{n} \cdot \frac{d}{dw}\left(1 - y_i (w^T x_i)\right)$$

$$= \vec{w} + \frac{-C y_i}{n} \cdot x_i^T$$

$$w_{t+1} = w_t - \eta \frac{\vec{w} - \frac{C}{n} y_i \cdot x_i^T}{\|\vec{w} - \frac{C}{n} y_i \cdot x_i^T\|}$$

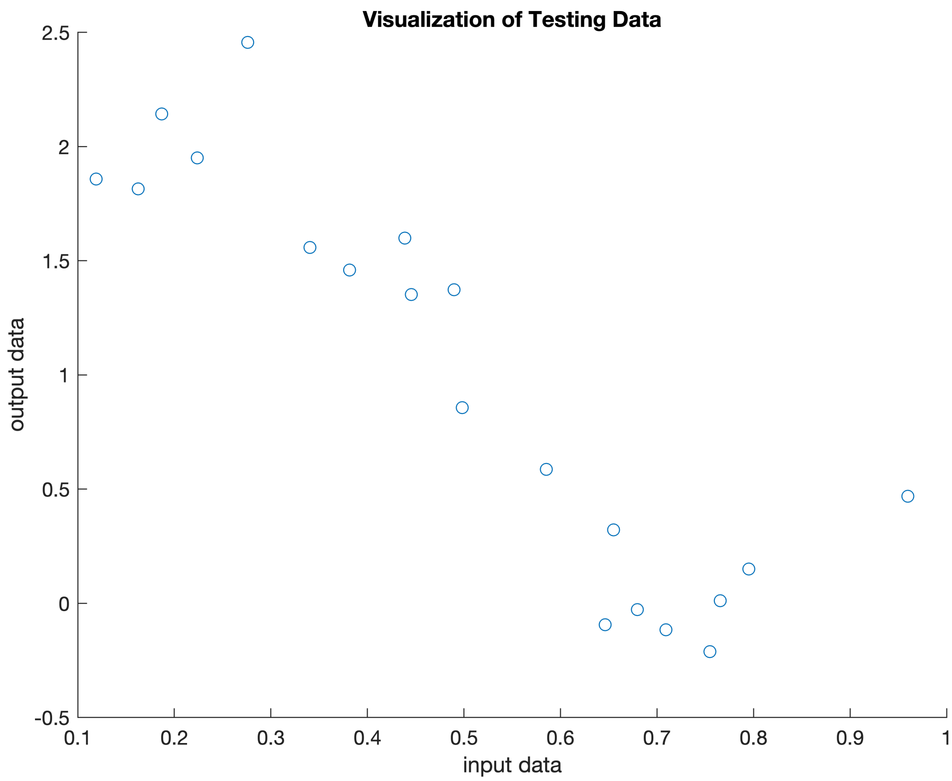
Problem 5 Homework 2

(a)

```
traindata = readtable("regression_train.csv");
testdata = readtable("regression_test.csv");

figure(1);
hold on;
train_in = traindata{:,1};
train_out = traindata{:,2};
scatter(train_in, train_out);
title("Visualization of Training Data");
xlabel("input data");
ylabel("output data");

figure(2);
test_in = testdata{:,1};
test_out = testdata{:,2};
scatter(test_in, test_out);
title("Visualization of Testing Data");
xlabel("input data");
ylabel("output data");
```

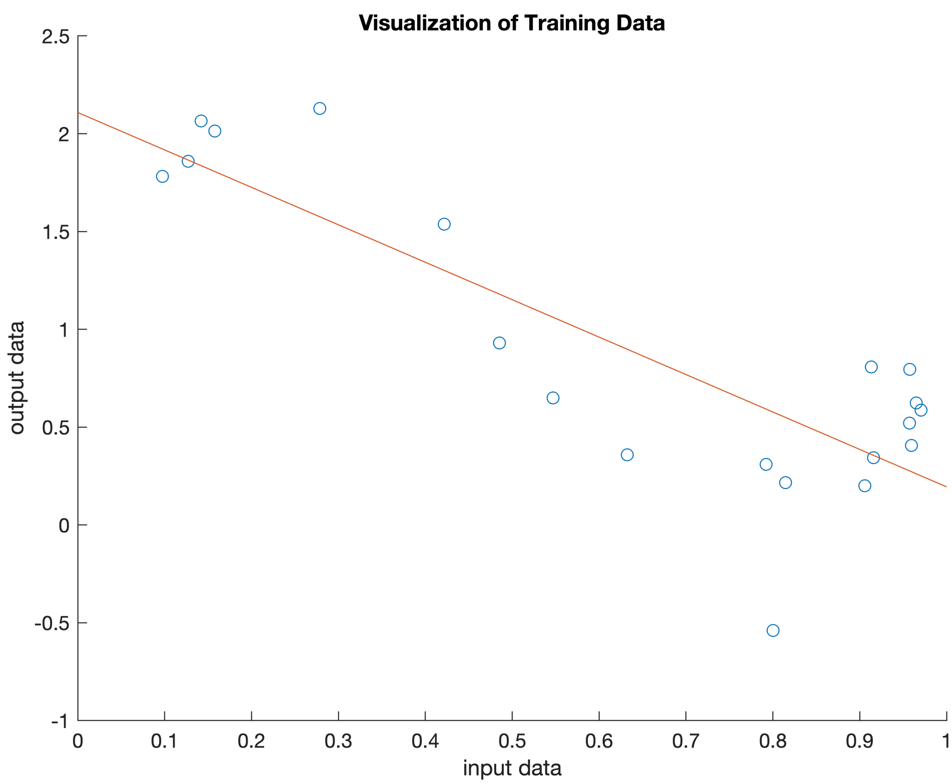


For these two scatter plots, it seems like the training one is less like a linear graph, and more like a cubic or a sinusoid. If we forced linear regression, it would seem like the fit wouldn't be that good and some points would be far away from the line.

However, for the testing data plot, it seems like it does fit the shape of a line a lot better.

(b)

```
n = size(traindata, 1);
y = train_out;
X = [ones(n, 1) train_in];
ws = (X' * X)^(-1) * X' * y;
J = norm(X * ws - y)^2;
xval = (0:0.01:1);
yval = ws(1) + ws(2) * xval;
figure(1);
plot(xval, yval);
hold off;
```



(c)

```
for eta = [0.05, 0.001, 0.0001, 0.00001]
    w = zeros(2, 1);
    Jprev = norm(X * w - y);
    for iters = 1:10000
        w1 = w(1) - eta * (X * w - y)' * X(:, 1);
        w2 = w(2) - eta * (X * w - y)' * X(:, 2);
        w = [w1; w2];
        J = norm(X * w - y)^2;
```

```
        if abs(J - Jprev) < 0.0001
            break
        end
        Jprev = J;
    end
    eta
    iters
    J
end
```

```
eta = 0.0500
iters = 66
J = 3.2606
eta = 1.0000e-03
iters = 2009
J = 3.2947
eta = 1.0000e-04
iters = 10000
J = 3.8878
eta = 1.0000e-05
iters = 10000
J = 11.5685
```

The lower the learning rate, η, the more iterations we need until completion. In addition, it seems like the final value of J is higher for lower η.

(d)

```
eta = 0.05
```

```
eta = 0.0500
```

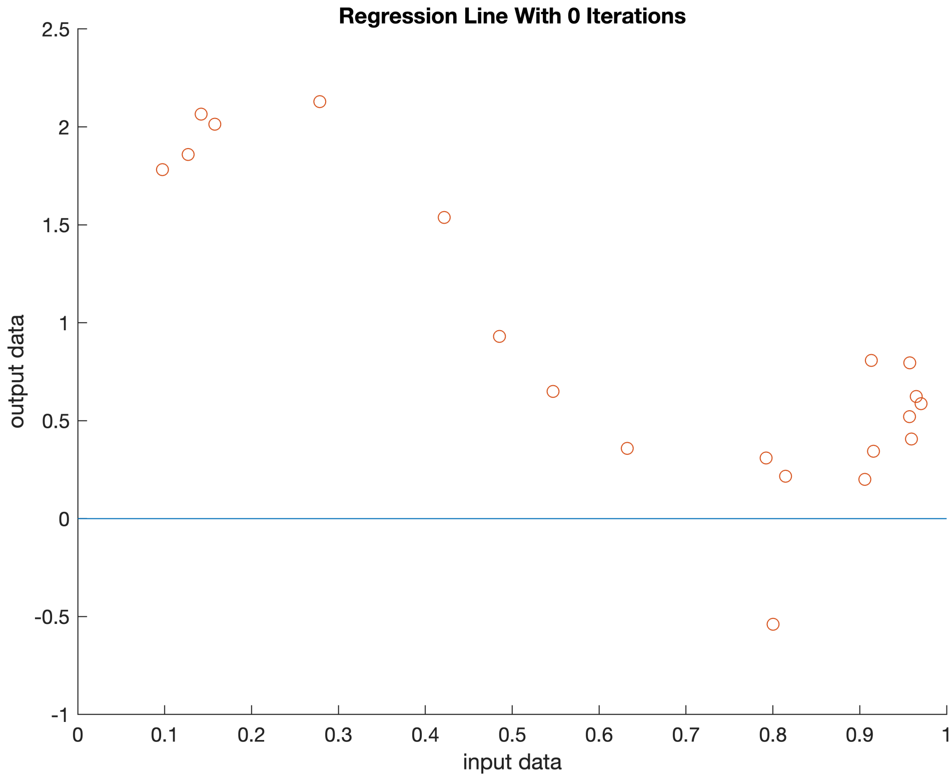
```
w = zeros(2, 1);
J = norm(X * w - y)^2;
f = 3;
w
```

```
w = 2×1
     0
     0
```

```
J
```

```
J = 26.3134
```

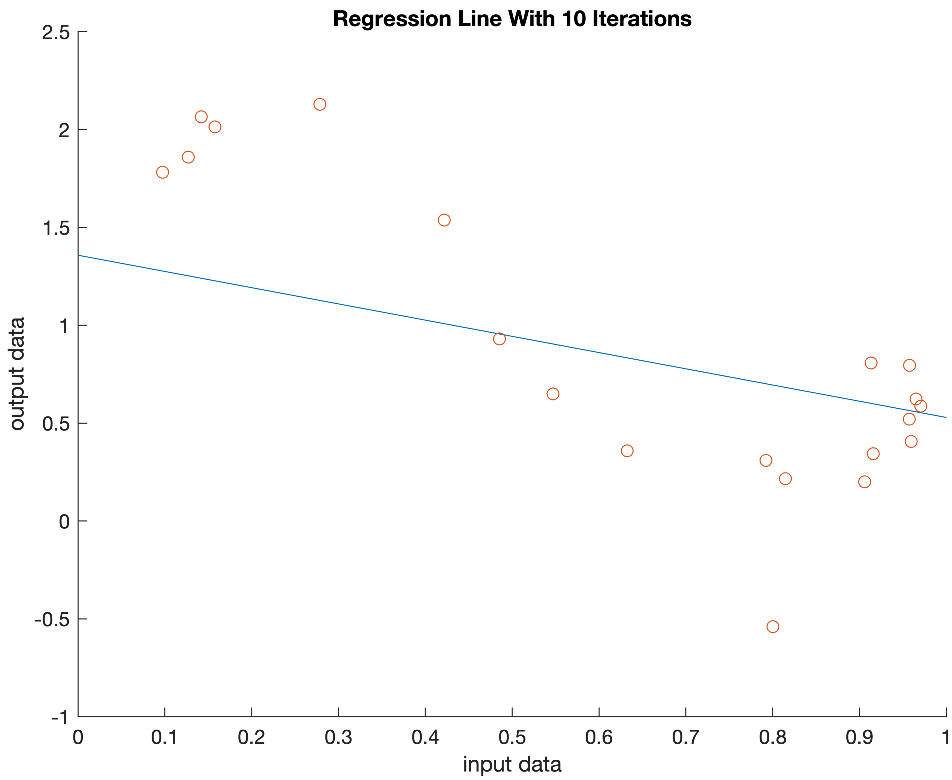
```
figure(f);
hold on;
xval = (0:0.01:1);
yval = w(1) + w(2) * xval;
plot(xval, yval);
xlabel("input data");
ylabel("output data");
title(sprintf("Regression Line With %d Iterations", 0))
scatter(train_in, train_out);
```



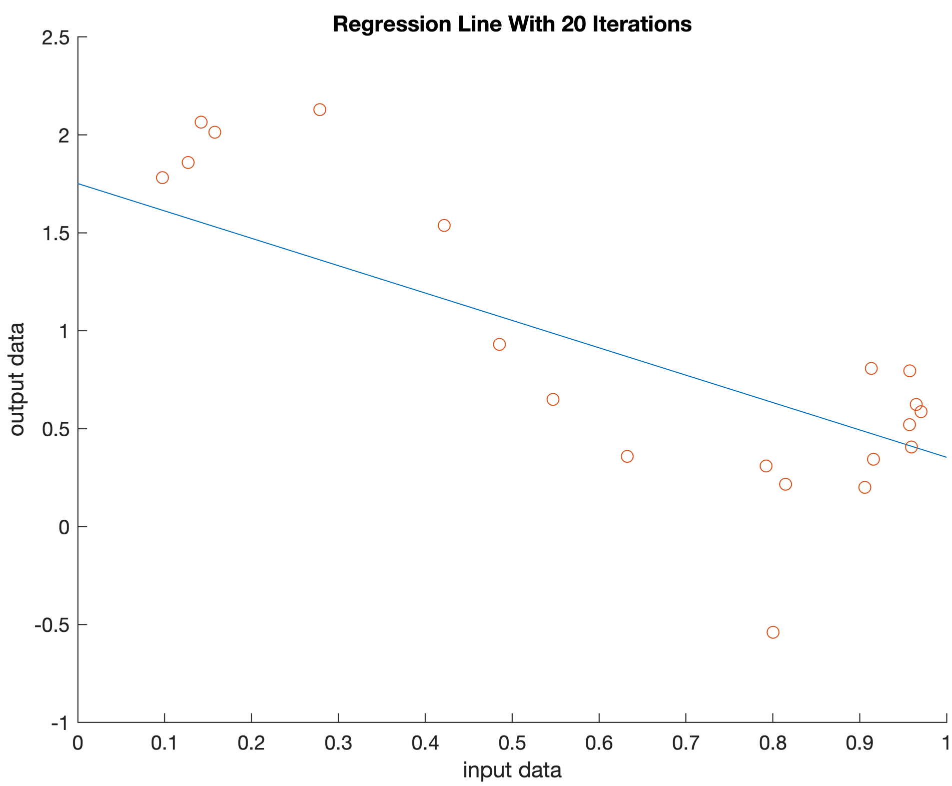

```
f = f + 1;

for iters = 1:40
    w1 = w(1) - eta * (X * w - y)' * X(:, 1);
    w2 = w(2) - eta * (X * w - y)' * X(:, 2);
    w = [w1; w2];
    J = norm(X * w - y)^2;
    if mod(iters, 10) == 0
        w
        J
        figure(f);
        hold on;
        xval = (0:0.01:1);
        yval = w(1) + w(2) * xval;
        plot(xval, yval);
        xlabel("input data");
        ylabel("output data");
        title(sprintf("Regression Line With %d Iterations", iters))
        scatter(train_in, train_out);
        f = f + 1;
    end
end
```

```
w = 2×1
    1.3580
   -0.8289
J = 5.7519
```

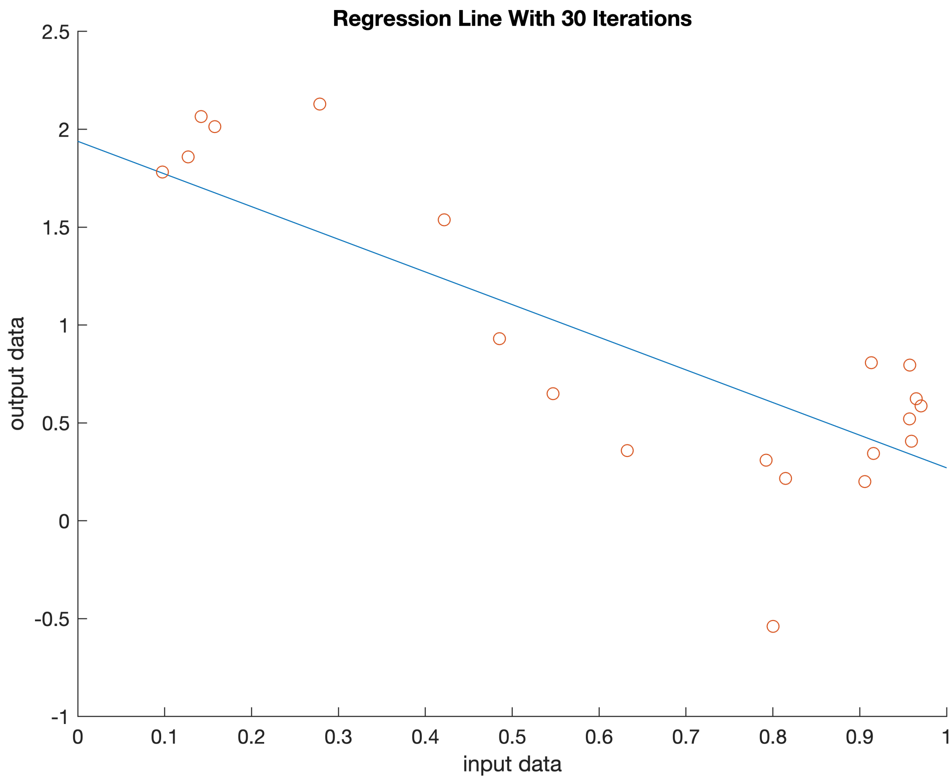



```
w = 2×1
    1.7515
   -1.3976
J = 3.8237
```

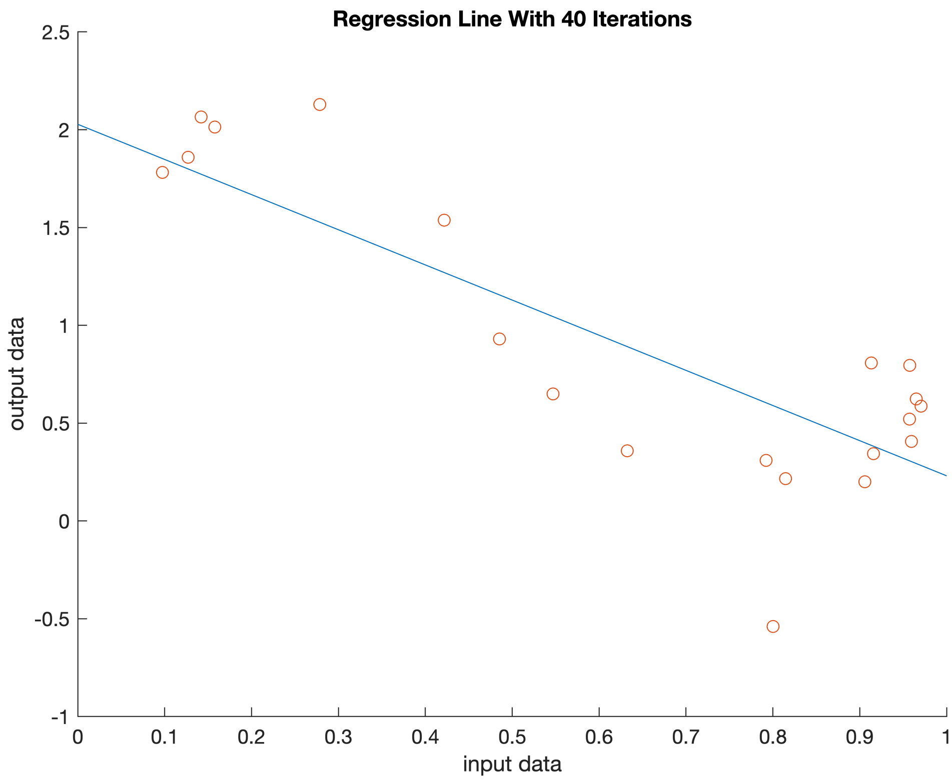



```
w = 2×1
```

```
    1.9386
   -1.6682
J = 3.3875
```




```
w = 2×1
    2.0276
   -1.7969
J = 3.2888
```



Over time, the line gets closer and closer to fitting the data, which is what we expect. The fitted lines originally have 0 slope but get closer and closer to what I had in (b).

(ei)

```
ntrain = size(traindata, 1);
ytrain = train_out;
xtrain = train_in;
Xtrain = ones(ntrain, 1);

ntest = size(testdata, 1);
ytest = test_out;
xtest = test_in;
Xtest = ones(ntest, 1);

Etrain_rms = [];
Etest_rms = [];

for m = 0:10
    wstrain = (Xtrain' * Xtrain)^(-1) * Xtrain' * ytrain;
    m
    Jtest = norm(Xtest * wstrain - ytest)^2
    Jtrain = norm(Xtrain * wstrain - ytrain)^2;

    Etrain_rms(end + 1) = sqrt(Jtrain / n);
    Etest_rms(end + 1) = sqrt(Jtest / n);
```

```
    Xtrain = [Xtrain xtrain];
    xtrain = xtrain .* xtrain;
    Xtest = [Xtest xtest];
    xtest = xtest .* xtest;
end
```

```
m = 0
Jtest = 14.6448
m = 1
Jtest = 5.4816
m = 2
Jtest = 4.0882
m = 3
Jtest = 1.0160
m = 4
Jtest = 0.8068
m = 5
Jtest = 0.7795
m = 6
Jtest = 0.8481
m = 7
Jtest = 1.0131
m = 8
Jtest = 1.0005
m = 9
Jtest = 1.0088
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
 1.147752e-19.
m = 10
Jtest = 0.9871
```

Best model seems like the one where m = 5, as that one fits the test data the best. We'll calculate Erms for this value of m.

```
fprintf("Optimal E_RMS for training data: %f\n", Etrain_rms(6));
```

```
Optimal E_RMS for training data: 0.194756
```

```
fprintf("Optimal E_RMS for test data: %f\n", Etest_rms(6));
```

```
Optimal E_RMS for test data: 0.197419
```

(eii)

```
xval = (0:1:10);
y1val = Etrain_rms;
y2val = Etest_rms;
figure(20);
hold on;
scatter(xval, y1val, "blue");
scatter(xval, y2val, "red");
title("Degree of polynomial vs E_{rms} for Training and Test Data");
xlabel("Degree of Polynomial");
ylabel("RMSE value");
```

Red is the test data, and Blue is the training data RMSE values.

(eiii)

I think that the degree of 4 or 5 best fits the data. 5 is where the values for test and train are closest together, but I think the individual values of 4 are lower. There is definitely overfitting in the higher degree values, because the testing data has much higher values of Erms than the training data. If there was overfitting, then while the fit for training might be very low, the fit for test would not be because of overfitting. We can see the spread between test and train Erms values as degree gets higher. For 4 and 5 though they are very close!