Homework Set 1, CPSC 8420

Zhang, Yue

Due 10/05/2023, 11:59PM EST

1 Ridge Regression

Please show that for arbitrary $\mathbf{A} \in \mathbb{R}^{n \times p}$, $(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}_p)^{-1}\mathbf{A}^T = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I}_n)^{-1}$, where $\lambda > 0$. Now assume $n = 100$, please compare the time consumption when $p = [10, 100, 1000, 2000]$ and plot the results appropriately (*e.g.* in one figure where X-axis denotes p while Y-axis the time consumption).

1.1 Proof

Left-multuply matirx $(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}_p)$ and then right-multiply $(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I}_n)$ on both sides of the equation, the left hand side becomes

$$\mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I}_n) \tag{1}$$

and right hand side becomes

$$(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}_p)\mathbf{A}^T \tag{2}$$

then (1) - (2), we get

$$\begin{aligned}
&\mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I}_n) - (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}_p)\mathbf{A}^T \\
&= \mathbf{A}^T\mathbf{A}\mathbf{A}^T + \lambda\mathbf{A}^T\mathbf{I}_n - \mathbf{A}^T\mathbf{A}\mathbf{A}^T - \lambda\mathbf{I}_p\mathbf{A}^T \\
&= \lambda(\mathbf{A}^T\mathbf{I}_n - \mathbf{I}_p\mathbf{A}^T) \\
&= \lambda(\mathbf{A}^T - \mathbf{A}^T) = \mathbf{0}
\end{aligned}$$

1.2 plot

See figure 1

2 Least Squares Extension

Assume $\mathbf{A}, \mathbf{X}, \mathbf{C}, \mathbf{Y}, \mathbf{U} \in \mathbb{R}^{n \times n}$. Given the fact that $vec(\mathbf{AUC}) = (\mathbf{C}^T \otimes \mathbf{A})vec(\mathbf{U})$, where $\otimes$ denotes *Kronecker product*, please use *least squares* we learned in class to solve:

$$\min_{\mathbf{X}} \|\mathbf{AX} + \mathbf{XC} - \mathbf{Y}\|_F^2. \tag{3}$$

To verify the correctness of your solution: 1) randomly generate $\mathbf{A}, \mathbf{X}^*, \mathbf{C}$; 2) set $\mathbf{Y} = \mathbf{AX}^* + \mathbf{X}^*\mathbf{C}$; 3) by making use of least squares, you can obtain the optimal $vec(\mathbf{X})$ given $\mathbf{A}, \mathbf{C}, \mathbf{Y}$ and 4) compare with $vec(\mathbf{X}^*)$.

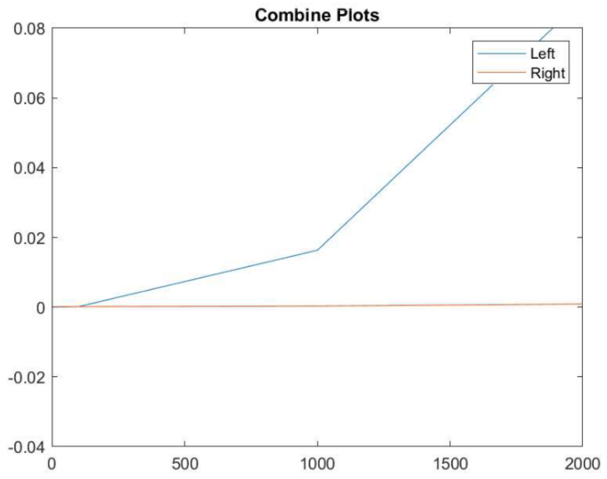


Figure 1: Q1

2.1 Solution

The problem can be converted into:

$$\begin{aligned}
&\min_{\mathbf{X}} \|vec(\mathbf{AX}) + vec(\mathbf{XC}) - vec(\mathbf{Y})\|_2^2 \\
&\Rightarrow \min_{\mathbf{X}} \|vec(\mathbf{AXI}) + vec(\mathbf{IXC}) - vec(\mathbf{Y})\|_2^2 \\
&\Rightarrow \min_{\mathbf{X}} \|(\mathbf{I}^T \otimes \mathbf{A})vec(\mathbf{X}) + (\mathbf{C}^T \otimes \mathbf{I})vec(\mathbf{X}) - vec(\mathbf{Y})\|_2^2 \\
&\Rightarrow \min_{\mathbf{X}} \|[(\mathbf{I}^T \otimes \mathbf{A}) + (\mathbf{C}^T \otimes \mathbf{I})]vec(\mathbf{X}) - vec(\mathbf{Y})\|_2^2
\end{aligned}$$

According to the regression model

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{y}\|_2^2, \mathbf{x}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$$

we can get

$$vec(\mathbf{X}^*) = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T vec(\mathbf{Y}) \tag{4}$$

where

$$\mathbf{M} = (\mathbf{I}^T \otimes \mathbf{A}) + (\mathbf{C}^T \otimes \mathbf{I})$$

2.2 Verification

```
n = 3;
A = rand(n);
X_star = rand(n);
C= rand(n);

Y = A*X_star+X_star*C;
M = kron(eye(n),A)+kron(C',eye(n));
vec_Y = Y(:);
vec_X = inv(M'*M)*M'*vec_Y;
```

```
erro = vec_X-X_star(:)
```

$$erro = 1.0e - 14 * [-0.0111 - 0.2331 - 0.6550 - 0.14430.1998 - 0.0666 - 0.17620.04440.2220]$$

3 Shrinkage Methods

For vanilla linear regression model: $\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{A}\boldsymbol{\beta}\|_2^2$, we denote the solution as $\hat{\boldsymbol{\beta}}_{LS}$; for ridge regression model: $\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{A}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2$, we denote the solution as $\hat{\boldsymbol{\beta}}_{\lambda}^{Ridge}$; for Lasso model: $\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{A}\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{\beta}\|_1$, we denote the solution as $\hat{\boldsymbol{\beta}}_{\lambda}^{Lasso}$; for Subset Selection model: $\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y}-\mathbf{A}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_0$, we denote the solution as $\hat{\boldsymbol{\beta}}_{\lambda}^{Subset}$, now please derive each $\hat{\boldsymbol{\beta}}$ given $\mathbf{y}, \mathbf{A}(s.t.\ \mathbf{A}^T\mathbf{A} = \mathbf{I}), \lambda$. Also, show the relationship of (each element in) $\hat{\boldsymbol{\beta}}_{\lambda}^{Ridge}, \hat{\boldsymbol{\beta}}_{\lambda}^{Lasso}, \hat{\boldsymbol{\beta}}_{\lambda}^{Subset}$ with (that in) $\hat{\boldsymbol{\beta}}_{LS}$ respectively. (you are encouraged to illustrate the relationship with figures appropriately.)

3.1 Ridge vs. vanilla Linear regresion

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{LS} &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y} = \mathbf{A}^T\mathbf{y}\\ \hat{\boldsymbol{\beta}}_{\lambda}^{Ridge} &= (\mathbf{A}^T\mathbf{A} + 2\lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{y} = (2\lambda+1)^{-1}\mathbf{A}^T\mathbf{y}\end{aligned}$$

Since in our case $\mathbf{A}\mathbf{A}^T = \mathbf{I}$, then

$$\hat{\boldsymbol{\beta}}_{LS} = \mathbf{A}^T\mathbf{y} \tag{5}$$

$$\hat{\boldsymbol{\beta}}_{\lambda}^{Ridge} = (2\lambda+1)^{-1}\mathbf{A}^T\mathbf{y} \tag{6}$$

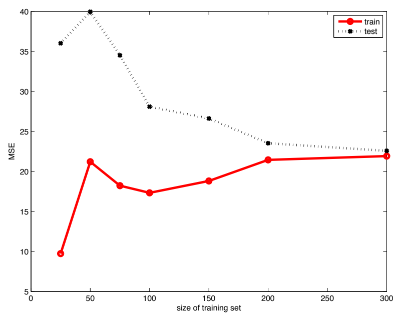
Take (5) into (6), we can get

$$\hat{\boldsymbol{\beta}}_{\lambda}^{Ridge} = (2\lambda+1)^{-1}\hat{\boldsymbol{\beta}}_{LS} \tag{7}$$

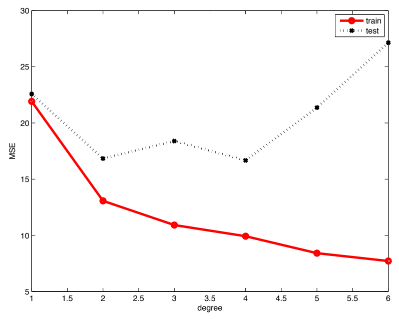
3.2 Lasso vs. vanilla LR

We need to split it into two cases: one for positive $\hat{\boldsymbol{\beta}}_{\lambda}^{Lasso}$ and one for negative $\hat{\boldsymbol{\beta}}_{\lambda}^{Lasso}$

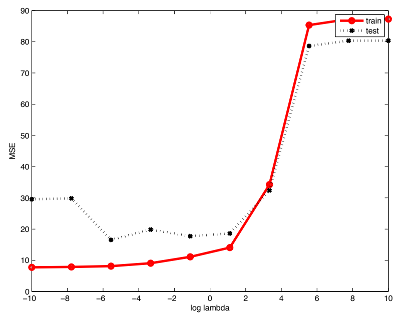
$$\hat{\boldsymbol{\beta}}_{\lambda}^{Lasso} = \begin{cases} (\mathbf{A}^T\mathbf{A})^{-1}(\mathbf{A}^T\mathbf{y} - \lambda) = \mathbf{A}^T\mathbf{y} - \lambda = \hat{\boldsymbol{\beta}}_{LS} - \lambda, \text{if } \hat{\boldsymbol{\beta}}_{\lambda}^{Lasso} > 0 \Rightarrow \hat{\boldsymbol{\beta}}_{LS} > \lambda \\ (\mathbf{A}^T\mathbf{A})^{-1}(\mathbf{A}^T\mathbf{y} + \lambda) = \mathbf{A}^T\mathbf{y} + \lambda = \hat{\boldsymbol{\beta}}_{LS} + \lambda, \text{if } \hat{\boldsymbol{\beta}}_{\lambda}^{Lasso} < 0 \Rightarrow \hat{\boldsymbol{\beta}}_{LS} < \lambda \end{cases} \tag{8}$$



(a)



(b)



(c)

Figure 2: MSE vs (a) training set size, (b) polynomial degree, (c) size of ridge penalty. Solid Red = training, dotted black = test.

3.3 Subset: the constrain of subset method is not a convex set

We need to split it into two cases: one for taking the ith feature, then the corresponding element in $\hat{\boldsymbol{\beta}}_{\lambda}^{Subset}$ equals to 1, and one for not ,which means its corresponding element in $\hat{\boldsymbol{\beta}}_{\lambda}^{Subset}$ equals to 0

$$\hat{\boldsymbol{\beta}}_{\lambda}^{Subset} = \begin{cases} \hat{\boldsymbol{\beta}}_{LS}, \text{if } \hat{\boldsymbol{\beta}}_{\lambda}^{Subset} \neq 0 \Rightarrow \hat{\boldsymbol{\beta}}_{LS} > \lambda \\ \mathbf{0} \end{cases} \tag{9}$$

4 Linear Regression and its Extension

In the Boston housing dataset, there are 506 records. We will use first 13 features as inputs, x, and the 14th feature, median house price, as the output y. All features are continuous, except feature 4, which is binary. However, we will treat this like any other continuous variable.

1. Load the housing.data file. We will use the first 300 cases for training and the remaining 206 cases for testing. However, the records seem to be sorted in some kind of order. To eliminate this, we will shuffle the data before splitting into a training/test set. So we can all compare results, let use the following convention:

```
data = load('housing.data');
x = data(:, 1:13);
y = data(:,14);
[n,d] = size(x);
seed = 2; rand('state',seed); randn('state', seed);
```

```
perm = randperm(n); % remove any possible ordering fx
x = x(perm,:); y = y(perm);
Ntrain = 300;
Xtrain = x(1:Ntrain,:); ytrain = y(1:Ntrain);
Xtest = x(Ntrain+1:end,:); ytest = y(Ntrain+1:end);
```

2. Now extract the first n records of the training data, for $n \in \{25, 50, 75, 100, 150, 200, 300\}$. For each such training subset, standardize it (you may use *zscore* function in Matlab), and fit a linear regression model using least squares. (Remember to include an offset term.) Then standardize the whole test set in the same way. Compute the mean squared error on the training subset and on the whole test set. Plot MSE versus training set size. You should get a plot like Figure 1(a). Turn in your plot and code. Explain why the test error decreases as n increases, and why the train error increases as n increases. Why do the curves eventually meet? As a debugging aid, here are the regression weights I get when I train on the first 25 cases (the first term is the offset, w0): $[26.11, -0.58, 3.02, \ldots, -0.21, -0.27, -1.16]$.

   (a) Code

```
data = load('housing.data');
x = data(:, 1:13);
y = data(:,14);
[n,d] = size(x);
seed = 2; rand('state',seed); randn('state', seed);
perm = randperm(n); % remove any possible ordering fx
x = x(perm,:); y = y(perm);

Ntrains = [25 50 75 100 150 200 300];

test_errors = [];
train_errors = [];

for Ntrain = Ntrains
    Xtrain = x(1:Ntrain,:); ytrain = y(1:Ntrain);
    Xtest = x(Ntrain+1:end,:); ytest = y(Ntrain+1:end);

    Xtrain = zscore(Xtrain); ytrain = zscore(ytrain);
    Xtest = zscore(Xtest); ytest = zscore(ytest);

    Xtrain = [ones(Ntrain,1) Xtrain];
    Xtest = [ones(n-Ntrain,1) Xtest];

    weight =  LR(Xtrain, ytrain);

    train_error = immse(Xtrain*weight,ytrain);
    test_error = immse(Xtest*weight,ytest);

    test_errors = [test_errors test_error];
    train_errors = [train_errors train_error];
end
```

```
% plot
x = Ntrains;
y1 = train_errors;
plot(x,y1,'DisplayName','Train')
hold on
y2 = test_errors;
plot(x,y2,'DisplayName','Test')
hold off
legend
function w = LR(X, Y)
    w = (X'*X)\X'*Y;
end
```

(b) Figure : See Figure 3

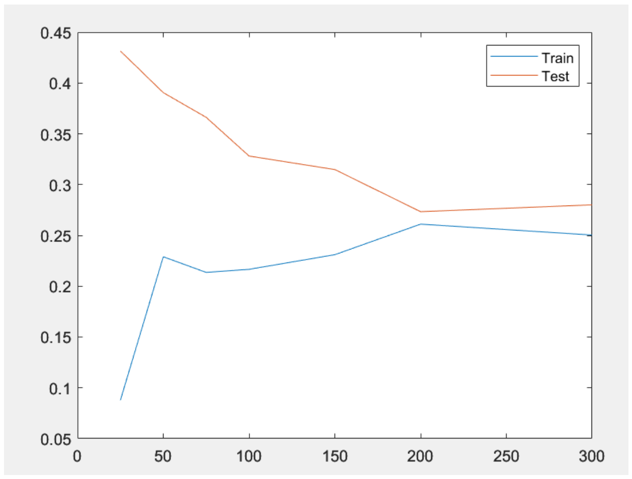


Figure 3: Q4-2

(c) The testing error is mainly caused by bias, and when the number of training samples increases, bias decreases However, it is not possible for the testing error to be smaller than the training error, so the best outcome is for the testing error to gradually approach the training error

3. We will now replace the original features with an expanded set of features based on higher order terms. (We will ignore interaction terms.) For example, a quadratic expansion gives:

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{pmatrix} \to \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1d} & x_{11}^2 & x_{12}^2 & \dots & x_{1d}^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} & x_{n1}^2 & x_{n2}^2 & \dots & x_{nd}^2 \end{pmatrix} \tag{10}$$

The provided function degexpand(X,deg,addOnes) will replace each row of X with all powers up to degree deg. Use this function to train (by least squares) models with degrees 1 to 6. Use all the the training data. Plot the MSE on the training and test sets vs degree. You should get a plot like Figure 1(b). Turn in your plot and code. Explain why the test error decreases and then increases with degree, and why the train error decreases with degree.

(a) Code

```
data = load('housing.data');
x = data(:, 1:13);
y = data(:,14);
[n,d] = size(x);
seed = 2; rand('state',seed); randn('state', seed);
perm = randperm(n); % remove any possible ordering fx
x = x(perm,:); y = y(perm);

degrees = [1 2 3 4 5 6];



test_errors = [];
train_errors = [];
Ntrain = 300;

for degree = degrees
    xx = degexpand(x, degree, true);
    Xtrain = xx(1:Ntrain,:); ytrain = y(1:Ntrain);
    Xtest = xx(Ntrain+1:end,:); ytest = y(Ntrain+1:end);

    Xtrain = zscore(Xtrain); ytrain = zscore(ytrain);
    Xtest = zscore(Xtest); ytest = zscore(ytest);

    %Xtrain = [ones(Ntrain,1) Xtrain];
    %Xtest = [ones(n-Ntrain,1) Xtest];

    weight =  LR(Xtrain, ytrain);

    train_error = immse(Xtrain*weight,ytrain);
    test_error = immse(Xtest*weight,ytest);

    test_errors = [test_errors test_error];
    train_errors = [train_errors train_error];
end

% plot
x = degrees;
y1 = train_errors;
plot(x,y1,'DisplayName','Train')
hold on
y2 = test_errors;
plot(x,y2,'DisplayName','Test')
hold off
legend

function w = LR(X, Y)
    w = pinv(X'*X)*X'*Y;
end
```

(b) Figure : See Figure 4

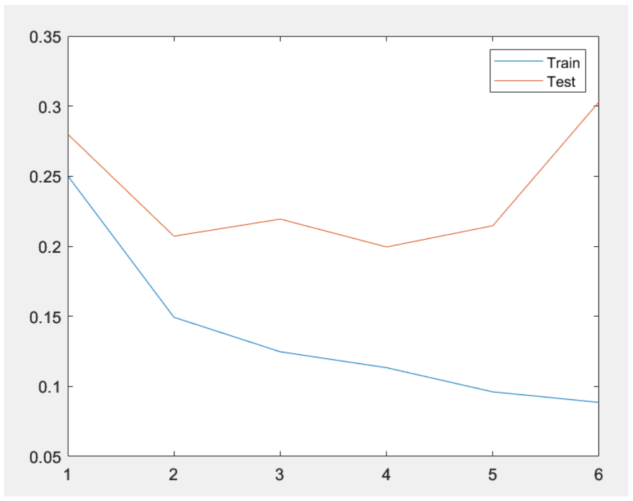


Figure 4: Q4-3

(c) The testing error is mainly caused by variance, and when the number of degrees increases, variance increases.

4. Now we will use ridge regression to regularize the degree 6 polynomial. Fit models using ridge regression with the following values for λ:

$$lambdas = [0\ logspace(-10, 10, 10)]$$

Use all the training data. Plot the MSE on the training and test sets vs $log_{10}(\lambda)$. You should get a plot like Figure 1(c). Turn in your plot and code. Explain why the test error goes down and then up with increasing λ, and why the train error goes up with increasing λ.

(a) Code

```
data = load('housing.data');
x = data(:, 1:13);
y = data(:,14);
[n,d] = size(x);
seed = 2; rand('state',seed); randn('state', seed);
perm = randperm(n); % remove any possible ordering fx
x = x(perm,:); y = y(perm);

degrees = [1 2 3 4 5 6];

lambdas = [0 logspace(-10,10,10)];

test_errors = [];
train_errors = [];
Ntrain = 300;
xx = degexpand(x, 6, true);
for lambda = lambdas
```

```
        Xtrain = xx(1:Ntrain,:); ytrain = y(1:Ntrain);
        Xtest = xx(Ntrain+1:end,:); ytest = y(Ntrain+1:end);

        Xtrain = zscore(Xtrain); ytrain = zscore(ytrain);
        Xtest = zscore(Xtest); ytest = zscore(ytest);

        %Xtrain = [ones(Ntrain,1) Xtrain];
        %Xtest = [ones(n-Ntrain,1) Xtest];

        %weight =  LR(Xtrain, ytrain);
        weight = Ridge(Xtrain, ytrain,lambda);
        train_error = immse(Xtrain*weight,ytrain);
        test_error = immse(Xtest*weight,ytest);

        test_errors = [test_errors test_error];
        train_errors = [train_errors train_error];
    end

    % plot
    x = log10(lambdas);
    y1 = train_errors;
    plot(x,y1,'DisplayName','Train')
    hold on
    y2 = test_errors;
    plot(x,y2,'DisplayName','Test')
    hold off
    legend

    hold off
    function w = Ridge(X, Y, lambda)
        w = pinv(X'*X+2*lambda)*X'*Y;
    end
    function w = LR(X, Y)
        w = pinv(X'*X)*X'*Y;
    end
```

(b) Figure See Figure 5

(c) Training Error:
See Figure (6), As λ increases, $\beta^T\beta$ decreases. It leads to the constraint area becomes smaller and smaller. The optimal solution we can find moves farther away from the center of the contour.

(d) Testing Error:
The initial decrease is because of the reduction in variance by introducting regulariztion term. The reason for the subsequent increase is the same as the training error above.

5. We turn to Lasso method with objective $\frac{1}{2}\|\mathbf{X}\beta - y\|^2 + \lambda\|\beta\|_1$ where λ varies in:

$$lambdas = [logspace(-10, 10, 10)]$$

and we make use of all training samples with no feature expansion. Please plot the changes of β with λ changes.

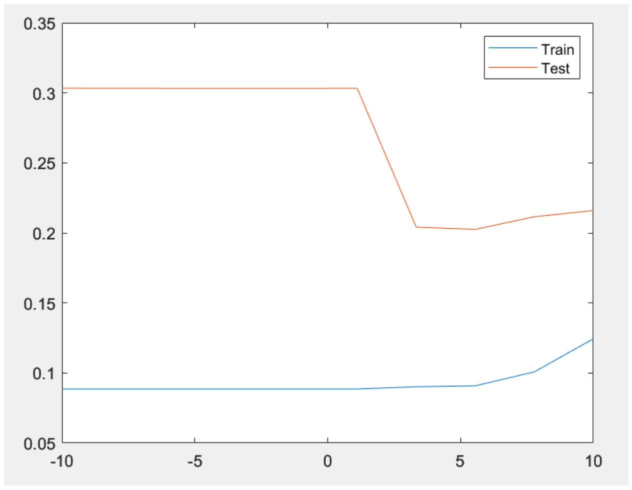


Figure 5: Q4-4

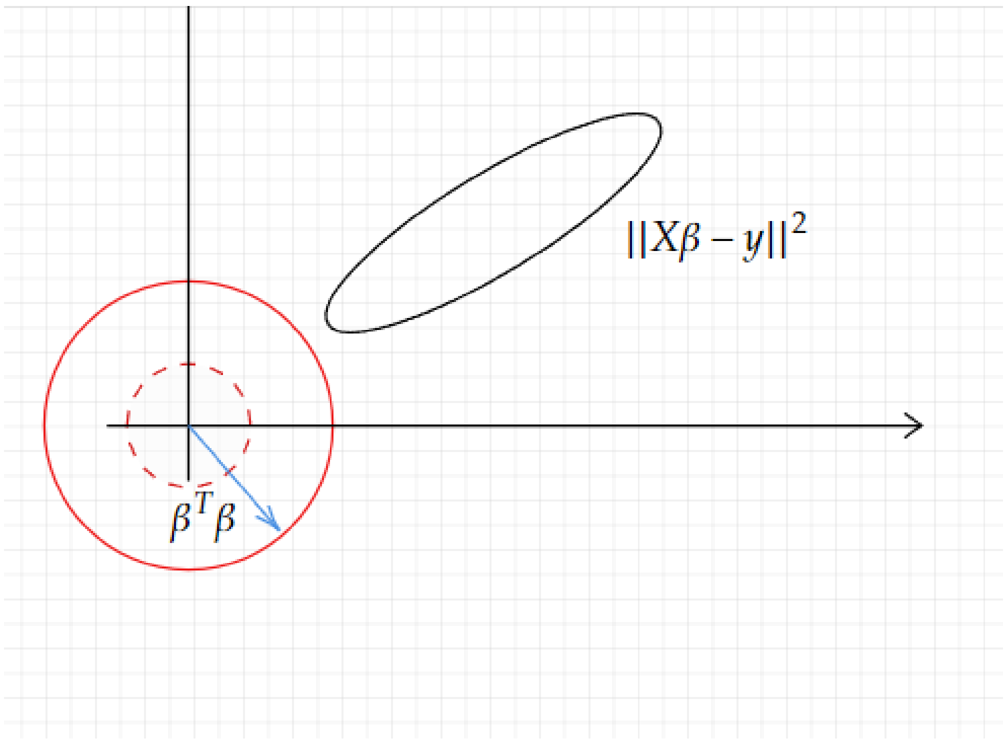


Figure 6:

(a) Code:

```
data = load('housing.data');
x = data(:, 1:13);
y = data(:,14);
[n,d] = size(x);
```

```
seed = 2; rand('state',seed); randn('state', seed);
perm = randperm(n); % remove any possible ordering fx
x = x(perm,:); y = y(perm);

lambdas = [logspace(-10,10,10)];

test_errors = [];
train_errors = [];
Ntrain = 300;
for lambda = lambdas

    Xtrain = x(1:Ntrain,:); ytrain = y(1:Ntrain);
    Xtest = x(Ntrain+1:end,:); ytest = y(Ntrain+1:end);

    Xtrain = zscore(Xtrain); ytrain = zscore(ytrain);
    Xtest = zscore(Xtest); ytest = zscore(ytest);

    Xtrain = [ones(Ntrain,1) Xtrain];
    Xtest = [ones(n-Ntrain,1) Xtest];

    weight = lassoAlg(Xtrain, ytrain,lambda);
    train_error = immse(Xtrain*weight,ytrain);
    test_error = immse(Xtest*weight,ytest);

    test_errors = [test_errors test_error];
    train_errors = [train_errors train_error];
end

% plot
x = log10(lambdas);
y1 = train_errors;
plot(x,y1)
title('Combine Plots')
text(x(end),y1(end),"train error");
hold on

y2 = test_errors;
plot(x,y2)
text(x(end),y2(end),"test error");


hold off
function w = Ridge(X, Y, lambda)
    w = pinv(X'*X+lambda)*X'*Y;
end
function w = LR(X, Y)
    w = pinv(X'*X)*X'*Y;
end

function xh = lassoAlg(A,y,lam)
    xnew = rand(size(A,2),1);   % "initial guess"
    xold = xnew+ones(size(xnew)); % used zeros so the while loop initiates
    loss = xnew - xold;
    thresh = 10e-3;     % threshold value for optimization

    while norm(loss) > thresh
        xold = xnew;    % need to store the previous iteration of xh
        for i = 1:length(xnew)
            a = A(:,i);     % get column of A
            p = (norm(a,2))^2;
            % from notes: -t = sum(aj*xj) - y for all j != i
            % i.e., sum(aj*xj) - ai*xi - y (my interpretation)
            % hence t = (above) * -1
            % want to be sure this the correct definition of t?
            t =  a*xnew(i) + y - A*xnew;
            q = a'*t;
            % update xi
            xnew(i) = (1/p) * sign(q) * max(abs(q)-lam, 0);
        end
        loss = xnew - xold;     % update loss
    end
    xh = xnew;
end
```

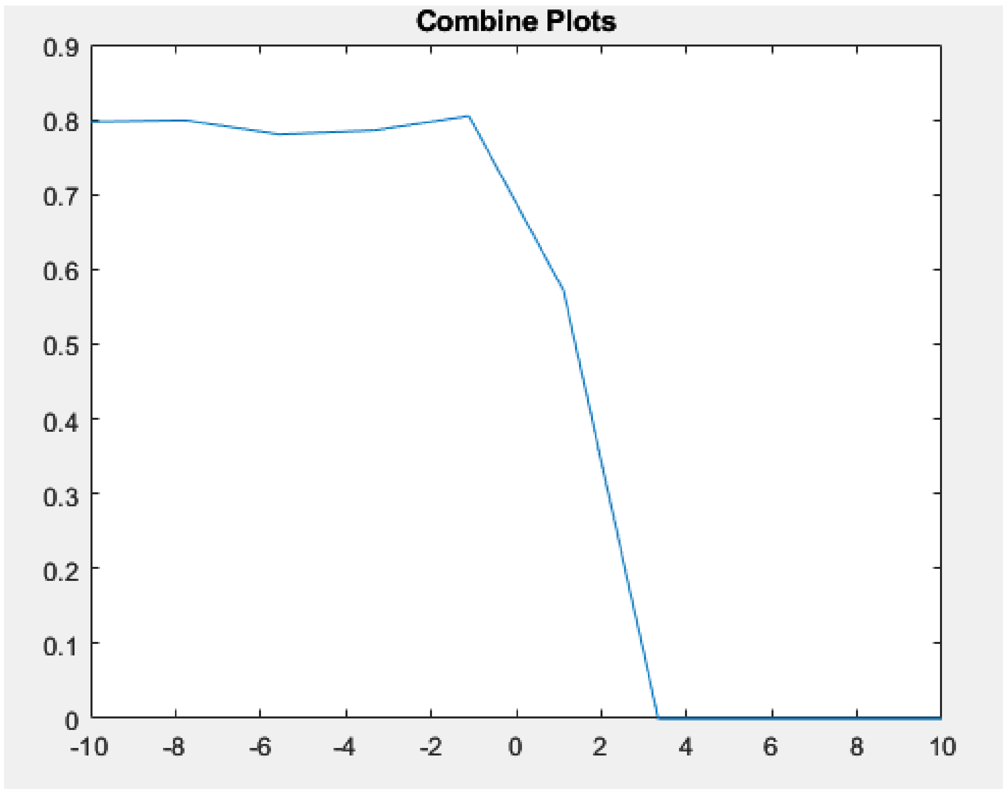


Figure 7:

(b) Figure: See Figure(7)