Homework Set 1, CPSC 8420, Spring 2022

Huang, Gangtong

Due 03/03/2022, Thursday, 11:59PM EST

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. Proof of $(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}_p)^{-1}\mathbf{A}^T = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I}_n)^{-1}$

$(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}_p)^{-1}\mathbf{A}^T = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I}_n)^{-1}$
$= \mathbf{A}^T\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I}_p\mathbf{A}^T$
$= \mathbf{A}^T\mathbf{A}\mathbf{A}^T + \lambda\mathbf{A}^T$
$= \mathbf{A}^T\mathbf{A}\mathbf{A}^T + \mathbf{A}^T\lambda$
$= \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda)$
$= \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I}_n)$

2. Time consumption with varied p

Implemented with Python.

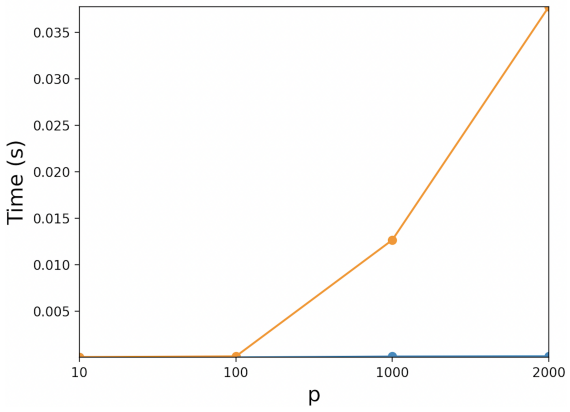


Figure 1: Ridge regression time consumption with varied p.

Bias–variance trade-off for k-NN

Assume $y = f(x) + \epsilon$ where $E(\epsilon) = 0, Var(\epsilon) = \sigma^2$. Please show that:

$$Err(x_0) = \sigma^2 + \frac{\sigma^2}{k} + [f(x_0) - \frac{1}{k}\sum_{l=1}^{k} f(x_l)]^2, \tag{1}$$

where x_l denotes the nearest neighbour data. Please justify Bias and Variance change when k increases and explain if necessary.

$$\begin{aligned} Err(x_0) &= Err[(y - \hat{f}_k(x))^2 | x = x_0] \\ &= Err[(y - \hat{f}_k(x_0))^2] \\ &= Var(\hat{f}(x_0)) + Bias(\hat{f}(x_0))^2 + Var(\epsilon) \end{aligned} \tag{2}$$

Denote the l-th neighbor of x_0's nearest neighbor as x_l, then:

$$\hat{f}(x_0) = \frac{1}{k}\sum_{l=1}^{k} y(x_l) = \frac{1}{k}\sum_{l=1}^{k}[f(x_l) + \epsilon] \tag{3}$$

$$\begin{aligned} Var(\hat{f}(x_0)) &= Var\{\frac{1}{k}\sum_{l=1}^{k}[f(x_l) + \epsilon]\} \\ &= \frac{1}{k^2} Var\{\sum_{l=1}^{k}[f(x_l) + \epsilon]\} \\ &= \frac{1}{k^2}\sum_{l=1}^{k} Var(f(x_l) + \epsilon) \\ &= \frac{1}{k^2}\sum_{l=1}^{k}\{Var[f(x_l)] + Var(\epsilon_i)\} \end{aligned} \tag{4}$$

And because $Var[f(x_l)] = 0$, $Var(\epsilon_i) = \sigma^2$,

$$Var(\hat{f}(x_0)) = \frac{1}{k^2} k\sigma^2 = \frac{\sigma}{k} \tag{5}$$

$$
\begin{aligned}
Bias(\hat{f}_k(x_0)) &= (y(x_0) - E(\hat{f}_k(x_0)))^2 \\
&= [fx_0 + \epsilon - \frac{1}{k}\sum_{l=1}^{k}(y(x_l))]^2 \\
&= [fx_0 + \epsilon - \frac{1}{k}\sum_{l=1}^{k}(f(x_l) + \epsilon)]^2 \\
&= [fx_0 + \epsilon - \frac{1}{k}\sum_{l=1}^{k}(f(x_l)) - \frac{1}{k}k\epsilon]^2 \\
&= [fx_0 - \frac{1}{k}\sum_{l=1}^{k} f(x_l)]^2
\end{aligned} \tag{6}
$$

Combining Equations (4)-(6), we have:

$$
\begin{aligned}
Err(x_0) &= Err[(y - \hat{f}_k(x))^2 | x = x_0] = Var(\hat{f}(x_0)) + Bias(\hat{f}(x_0))^2 + Var(\epsilon) \\
&= \sigma^2 + \frac{\sigma^2}{k} + [f(x_0) - \frac{1}{k}\sum_{l=1}^{k} f(x_l)]^2
\end{aligned} \tag{7}
$$

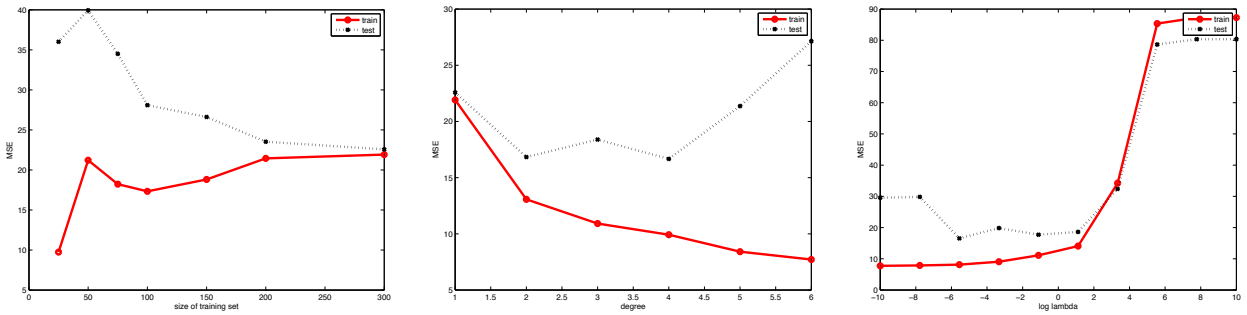
Shrinkage Methods

For vanilla linear regression model: $\min \|\mathbf{y} - \mathbf{A}\boldsymbol{\beta}\|_2^2$, we denote the solution as $\hat{\boldsymbol{\beta}}_{LS}$; for ridge regression model: $\min \|\mathbf{y} - \mathbf{A}\boldsymbol{\beta}\|_2^2 + \lambda * \|\boldsymbol{\beta}\|_2^2$, we denote the solution as $\hat{\boldsymbol{\beta}}_{\lambda}^{Ridge}$; for Lasso model: $\min \frac{1}{2}\|\mathbf{y} - \mathbf{A}\boldsymbol{\beta}\|_2^2 + \lambda * \|\boldsymbol{\beta}\|_1$, we denote the solution as $\hat{\boldsymbol{\beta}}_{\lambda}^{Lasso}$; for Subset Selection model: $\min \frac{1}{2}\|\mathbf{y} - \mathbf{A}\boldsymbol{\beta}\|_2^2 + \lambda * \|\boldsymbol{\beta}\|_0$, 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.)

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



(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.

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): $[26.11, -0.58, 3.02, \ldots, -0.21, -0.27, -1.16]$.

   The optimum parameter in linear regression is: $\hat{\beta} = (X^T X)^{-}1 X^T Y$. The linear regression is implemented with MatLAB (code in Appendix). The testing and training errors with varied training set size are shown in Fig. 3. See codes in Appendix.

   The increase in training error is caused by the increased number of data points that is to fit with the linear model, while the decrease in the testing error is the result of the linear fit being more accurate due to larger training set size.
   Since all the data points that constitute both the training and testing sets share the same source, it can be expected that the random error per data point, ϵ, is similar across all data

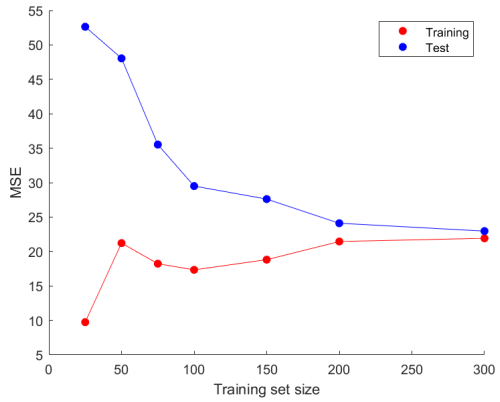


Figure 3: Ridge regression time consumption with varied p.

points, i.e. the variability of the data points is stable. The convergence between the testing and training data set occurs because the training and testing set sizes (300 and 206) are similar.

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:

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{pmatrix} \rightarrow \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1d} & x_{11}^2 & x_{12}^2 & \dots & x_{1d}^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} & x_{n1}^2 & x_{n2}^2 & \dots & x_{nd}^2 \end{pmatrix} \tag{8}$$

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.

The optimum parameter in linear regression is: $\hat{\beta} = (X^T X)^{-}1 X^T Y$, where the training set X is now expanded with quadratic terms. The testing and training errors with varied degrees of polynomials deg are shown in Fig. 4. See codes in Appendix.

Degree expansion can increase the degrees of freedom of the fitting model and let the model approximate the data points with more flexibility. The training error monotonically decrease with increasing degrees of polynomial expansion, because with more parameters the fitting to the training set will improve. The testing error reaches a minimum at degree of expansion $deg = 2$, indicating an improvement in the predictive performance of the model over the one without degree expansion. However, due to overfitting at higher deg values, the testing error then increases with increasing deg and reaches a peak at $deg = 5$.
Note: the result presented in Fig. 4 saw a significant decrease as testing error as deg increases

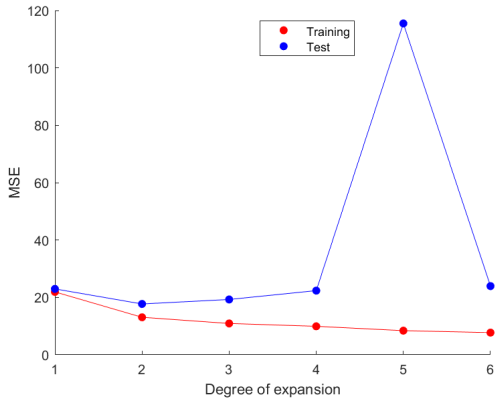


Figure 4: Ridge regression time consumption with varied p.

from 5 to 6. As a reference, we can compare this result with the result in Fig. 5 at $\lambda = 0$, since when $\lambda = 0$, we have $\hat{\beta}_{linear} = (X^T X)^{-}1X^T Y = (X^T X + 0\mathbf{I}_n)^{-1}X^T Y = \hat{\beta}_{ridge}(\lambda = 0)$, which are both $MSE_{test} \approx 22$ and $MSE_{train} \approx 10$. The identical results rule out the possibility of calculation errors. The rapid decrease in testing error may be due to other reasons. For example, testing error may actually reach a local minimum at $deg = 6$, but then increase with larger deg.

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.

The optimum parameter in ridge regression is: $\hat{\beta} = (X^T X + \lambda\mathbf{I}_n)^{-1}X^T Y$. The testing and training errors with varied λ are shown in Fig. 5. See codes in Appendix.

The penalty term in ridge regression, λ, imposes constraints on the magnitudes of elements in $\hat{\beta}$ to avoid overfitting. Since $\hat{\beta}_{linear} = (X^T X)^{-1}X^T Y$ already gives the optimal linear fit to the training data set, the deviation of $\hat{\beta}_{ridge}$ from $\hat{\beta}_{linear}$ caused by λ will result in the increase of MSE_{train}. The constraints on magnetudes of parameters avoids overfitting and can improve the predictive performance of the fitting model, as reflected in Fig. 5 when $log(\lambda)$ increases from -10 to 1. However, larger penalty term will supress the contribution of certain parameters to the fitting model, which may cause the loss of information represented by these parameters and eventually affect the accuracy of the fitting model.

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.

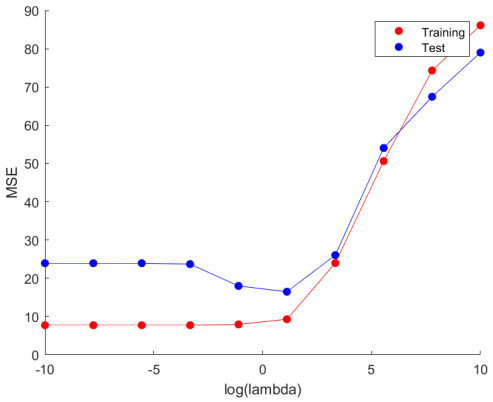


Figure 5: Ridge regression time consumption with varied p.

The optimum parameter in Lasso method does not have an analytical solution and can only be solved numerically. The testing and training errors with varied λ are shown in Fig. 5. See codes in Appendix.

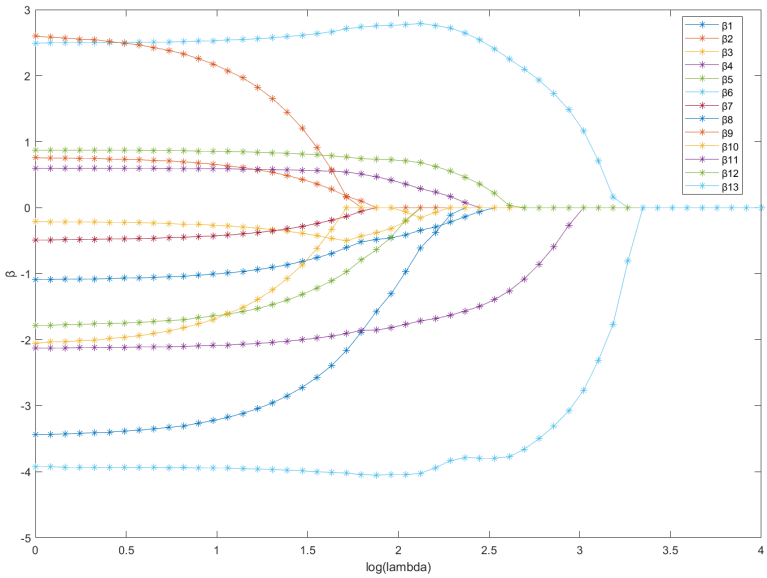


Figure 6: Ridge regression time consumption with varied p.

As expected, the Lasso method supresses the magnitudes of parameters. With larger λ the parameters effectively approximate 0.

Appendix

Problem 4.2: Linear regression implemented with MatLab

```
clear all;

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);
N = length(data);

l_mseTrain = [];
l_mseTest = [];

for Ntrain = [25 50 75 100 150 200 300]
    Ntest = N - Ntrain;

    Xtrain = x(1:Ntrain,:); ytrain = y(1:Ntrain);
    %Standardization
    Xtraincent = zscore(Xtrain);
    Xtest = x(Ntrain+1:end,:); ytest = y(Ntrain+1:end);
    Xtestcent = zscore(Xtest);

    XtrainApp = [ones(Ntrain,1) Xtraincent];
    XtestApp = [ones(Ntest,1) Xtestcent];

    beta_hat = pinv(XtrainApp'*XtrainApp)*XtrainApp'*ytrain;

    ytrainPred = beta_hat' * XtrainApp';
    ytestPred = beta_hat' * XtestApp';

    seTrain = 0;
    for i = 1:Ntrain
        seTrain = (ytrainPred(i) - ytrain(i))^2 + seTrain;
    end
    mseTrain = seTrain/Ntrain;
    l_mseTrain = [l_mseTrain, mseTrain];

    seTest = 0;
    for i = 1:Ntest
        seTest = (ytestPred(i) - ytest(i))^2 + seTest;
```

```
    end
    mseTest = seTest/Ntest;
    l_mseTest = [l_mseTest, mseTest];
end

% Plotting
hold on
scatter([25 50 75 100 150 200 300],l_mseTrain,'red','filled')
line([25 50 75 100 150 200 300],l_mseTrain,'Color','red')
xlabel("Training set size")
%xticks([1:7])
%xticklabels([25 50 75 100 150 200 300])
ylabel("MSE")
scatter([25 50 75 100 150 200 300],l_mseTest,'blue','filled')
line([25 50 75 100 150 200 300],l_mseTest,'Color','blue')
legend('Training','','Test','')
```

Problem 4.3: Degree expansion and linear regression implemented with MatLab

```
clear all;

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);
N = length(data)

Ntrain = 300;
Ntest = N - Ntrain;
Xtrain = x(1:Ntrain,:); ytrain = y(1:Ntrain);
Xtest = x(Ntrain+1:end,:); ytest = y(Ntrain+1:end);
Xtraincent = zscore(Xtrain);
Xtestcent = zscore(Xtest);


l_mseTrain = [];
l_mseTest = [];

%Standardize twice

for deg = [1 2 3 4 5 6]
    %Standardize and add the constant column
```

```
    XtrainApp = degexpand(Xtraincent,deg,1);
    XtestApp = degexpand(Xtestcent,deg,1);

    beta_hat = pinv(XtrainApp'*XtrainApp)*XtrainApp'*ytrain;

    ytrainPred = beta_hat' * XtrainApp';
    ytestPred = beta_hat' * XtestApp';

    seTrain = 0;
    for i = 1:Ntrain
        seTrain = (ytrainPred(i) - ytrain(i))^2 + seTrain;
    end
    mseTrain = seTrain/Ntrain;
    l_mseTrain = [l_mseTrain, mseTrain];

    seTest = 0;
    for i = 1:Ntest
        seTest = (ytestPred(i) - ytest(i))^2 + seTest;
    end
    mseTest = seTest/Ntest;
    l_mseTest = [l_mseTest, mseTest];

end
l_mseTest
l_mseTrain

%% Plotting %%
hold on
N_pts = 6
scatter([1:N_pts],l_mseTrain,'red','filled')
line([1:N_pts],l_mseTrain,'Color','red')
xlabel("Degree of expansion")

ylabel("MSE")
scatter([1:N_pts],l_mseTest,'blue','filled')
line([1:N_pts],l_mseTest,'Color','blue')
legend('Training','','Test','')
%% Function definitions %%
function xx = degexpand(x, deg, addOnes)
    % Expand input vectors to contain powers of the input features
    % This file is from pmtk3.googlecode.com

    [n,m] = size(x);
    if nargin < 3
        addOnes = 0;
```

```
    end

    xx = repmat(x, [1 1 deg]);
    degs = repmat(reshape(1:deg, [1 1 deg]), [n m]);
    xx = xx .^ degs;
    xx = reshape(xx, [n, m*deg]);

    if addOnes
      xx = [ones(n,1) xx];
    end
end
```

Problem 4.4: Degree expansion and ridge regression implemented with MatLab

```
clear all;

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);
N = length(data)

Ntrain = 300;
Ntest = N - Ntrain;
Xtrain = x(1:Ntrain,:); ytrain = y(1:Ntrain);
Xtest = x(Ntrain+1:end,:); ytest = y(Ntrain+1:end);
Xtraincent = zscore(Xtrain);
Xtestcent = zscore(Xtest);

l_mseTrain = [];
l_mseTest = [];

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

for lambda = lambdas
    Ntest = N - Ntrain;

    Xtrain = x(1:Ntrain,:); ytrain = y(1:Ntrain);
    %Standardization
    Xtraincent = zscore(Xtrain);
    Xtest = x(Ntrain+1:end,:); ytest = y(Ntrain+1:end);
```

```
    XtrainApp = degexpand(Xtraincent,6,1);
    XtestApp = degexpand(Xtestcent,6,1);

    beta_hat = pinv(XtrainApp'*XtrainApp + lambda*eye(79))*XtrainApp'*ytrain;

    ytrainPred = beta_hat' * XtrainApp'
    ytestPred = beta_hat' * XtestApp'

    seTrain = 0;
    for i = 1:Ntrain
        seTrain = (ytrainPred(i) - ytrain(i))^2 + seTrain;
    end
    mseTrain = seTrain/Ntrain;
    l_mseTrain = [l_mseTrain, mseTrain];

    seTest = 0;
    for i = 1:Ntest
        seTest = (ytestPred(i) - ytest(i))^2 + seTest;
    end
    mseTest = seTest/Ntest;
    l_mseTest = [l_mseTest, mseTest];
end

l_mseTest
l_mseTrain

%% Plotting
hold off
hold on
lambdas_plt = arrayfun(@(x) log10(x), lambdas)
scatter(lambdas_plt,l_mseTrain,'red','filled')
line(lambdas_plt,l_mseTrain,'Color','red')
xlabel("log(lambda)")
ylabel("MSE")
scatter(lambdas_plt,l_mseTest,'blue','filled')
line(lambdas_plt,l_mseTest,'Color','blue')
legend('Training','','Test','')

%% Reference using built-in ridge() function
%
% l_mseTrain = [];
% l_mseTest = [];
%
% for lambda = lambdas
%     B = ridge(ytrain, XtrainApp, lambda)
```

```
%     ytrainPred = B' * XtrainApp'
%     ytestPred = B' * XtestApp'
%
%     seTrain = 0;
%     for i = 1:Ntrain
%         seTrain = (ytrainPred(i) - ytrain(i))^2 + seTrain;
%     end
%     mseTrain = seTrain/Ntrain;
%     l_mseTrain = [l_mseTrain, mseTrain];
%
%     seTest = 0;
%     for i = 1:Ntest
%         seTest = (ytestPred(i) - ytest(i))^2 + seTest;
%     end
%     mseTest = seTest/Ntest;
%     l_mseTest = [l_mseTest, mseTest];
% end
%
% hold off
% hold on
% lambdas_plt = arrayfun(@ (x) log(x), lambdas)
% scatter(lambdas_plt,l_mseTrain,'red','filled')
% line(lambdas_plt,l_mseTrain,'Color','red')
% xlabel("Training set size")
% %xticks([1:7])
% %xticklabels([25 50 75 100 150 200 300])
% ylabel("MSE")
% scatter(lambdas_plt,l_mseTest,'blue','filled')
% line(lambdas_plt,l_mseTest,'Color','blue')
% legend('Training','','Test','')

%%
function xx = degexpand(x, deg, addOnes)
    % Expand input vectors to contain powers of the input features
    % This file is from pmtk3.googlecode.com

    [n,m] = size(x);
    if nargin < 3
        addOnes = 0;
    end

    xx = repmat(x, [1 1 deg]);
    degs = repmat(reshape(1:deg, [1 1 deg]), [n m]);
    xx = xx .^ degs;
    xx = reshape(xx, [n, m*deg]);
```

```
    if addOnes
      xx = [ones(n,1) xx];
    end
end
```

Problem 4.5: Lasso method implemented with MatLab

```
clear all;

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);
N = length(data)

Ntrain = 300;
Ntest = N - Ntrain;
Xtrain = x(1:Ntrain,:); ytrain = y(1:Ntrain);
Xtest = x(Ntrain+1:end,:); ytest = y(Ntrain+1:end);
Xtraincent = zscore(Xtrain);
Xtestcent = zscore(Xtest);

l_beta = [];

lambdas = [logspace(0,4)];

for lambda = lambdas
    beta_test = lassoAlg(Xtraincent, ytrain, lambda);
    l_beta = [l_beta beta_test];
end

%Plotting
for i = length(l_beta)
    lambdas_plt = arrayfun(@(x) log10(x), lambdas);
    for i = [1 : length(l_beta(:,1))]
        plot(lambdas_plt,l_beta(i,:),'-*')
        hold on
    end
end
colormap(jet(13))
xlabel("log(lambda)")
```

```
ylabel("beta")
legends = arrayfun(@(x) strcat('beta',string(x)), [1:13])
legend('Parameters', legends)
%
% l_beta(1,:)

%% Lasso Optimization Algorithm %%
% inputs: A (nxd matrix), y (nx1 vector), lam (scalar)
% return: xh (dx1 vector)

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
```