Table of Contents


```
% Austin Welch
% EC503 HW7.4
% LASSO vs. Ridge

% In this problem, we explore the trade-offs between l1-regularization
% (lasso) and l2-regularization (ridge) in a multi-dimensional
 prostate
% cancer dataset prostateStnd.mat. In this dataset, 8 features: lcavol
 -
% log(cancer volume), lweight - log(prostate weight), age, lbph -
% log(benign prostate hyperplasia amount), svi (seminal vesicle
 invasion),
% lcp - log(capsular penetration), gleason (Gleason score), and pgg45
% (percentage Gleason scores 4 or 5) are used to estimate lpsa (log
% (prostate specific antigen)). In the same fashion as the last
 problem,
% training and test data re provided: (xtrain, ytrain), (xtest,
 ytest). The
% first 8 features correspond to the first 8 entries of names. The
 ninth
% entry of names (the last one) is the output in (ytest, ytrain).
```

(a)

Implement cyclic coordinate descent for lasso (shooting algorithm). Use centered data. Update co-efficients at most 100 times or until coefficients have converged (no change). Do not use MATLAB's lasso. (i) Plot the lasso coefficient of each feature (8 total - yielding 8 curves) as a function of ln(lambda) for ln(lambda) ranging from -5 to 10 in steps of 1. Label the plot accordingly. (ii) In another figure, plot the mean-squared-error (MSE) of both the training and the test data as a function of ln(lambda).

```
% clear and load data
clear; clc;
load('prostateStnd.mat');

% center training data
muXtrain = mean(Xtrain);
xTrainTilda = Xtrain - muXtrain;
% center test data
muXtest = mean(Xtest);
xTestTilda = Xtest - muXtest;

% ridge weights
```

```
lambda = exp(-5:10);
W_ridge = cell(1,length(lambda));
coeff_ridge = zeros(length(W_ridge),size(xTrainTilda,2));
HtrainRidge = cell(1,length(lambda));
HtestRidge = cell(1,length(lambda));
ridgeTrainMSE = zeros(1,length(lambda));
ridgeTestMSE = zeros(1,length(lambda));

% ridge estimates and MSEs to compare against lasso
for i=1:length(lambda)
    W_ridge{i} = ridge(ytrain,xTrainTilda,lambda(i));
    coeff_ridge(i,:) = W_ridge{i};
    HtrainRidge{i} = xTrainTilda*W_ridge{i};
    HtestRidge{i} = xTestTilda*W_ridge{i};
    ridgeTrainMSE(i) = mse(ytrain,HtrainRidge{i});
    ridgeTestMSE(i) = mse(ytest,HtestRidge{i});
end
```

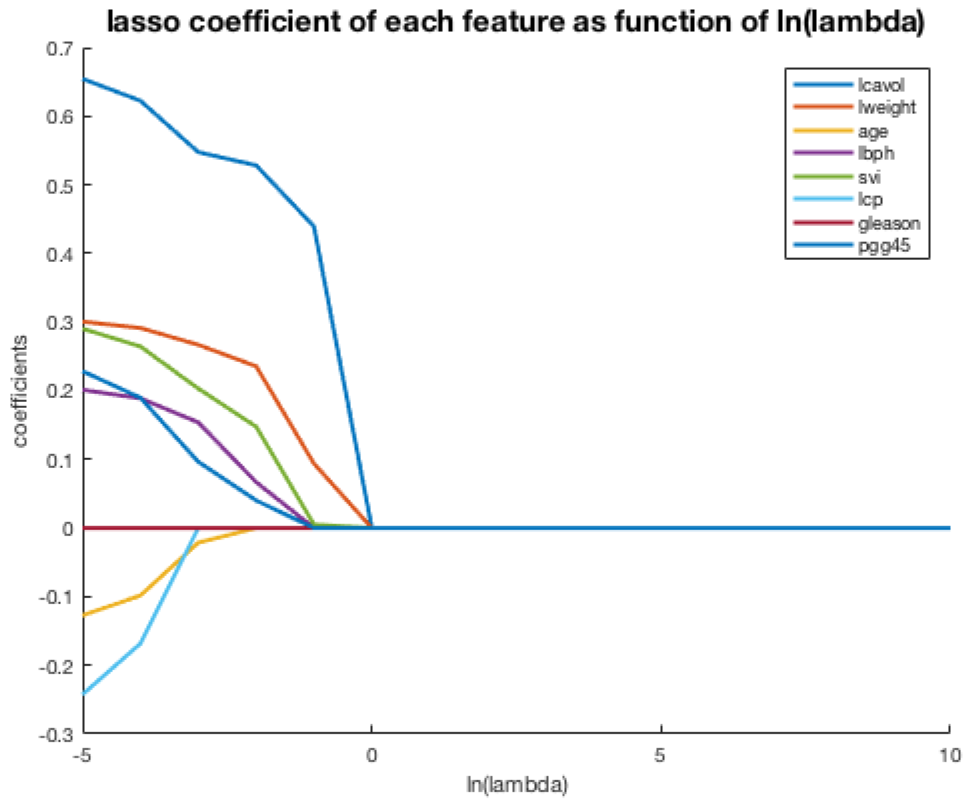
Cylic coordinate descent

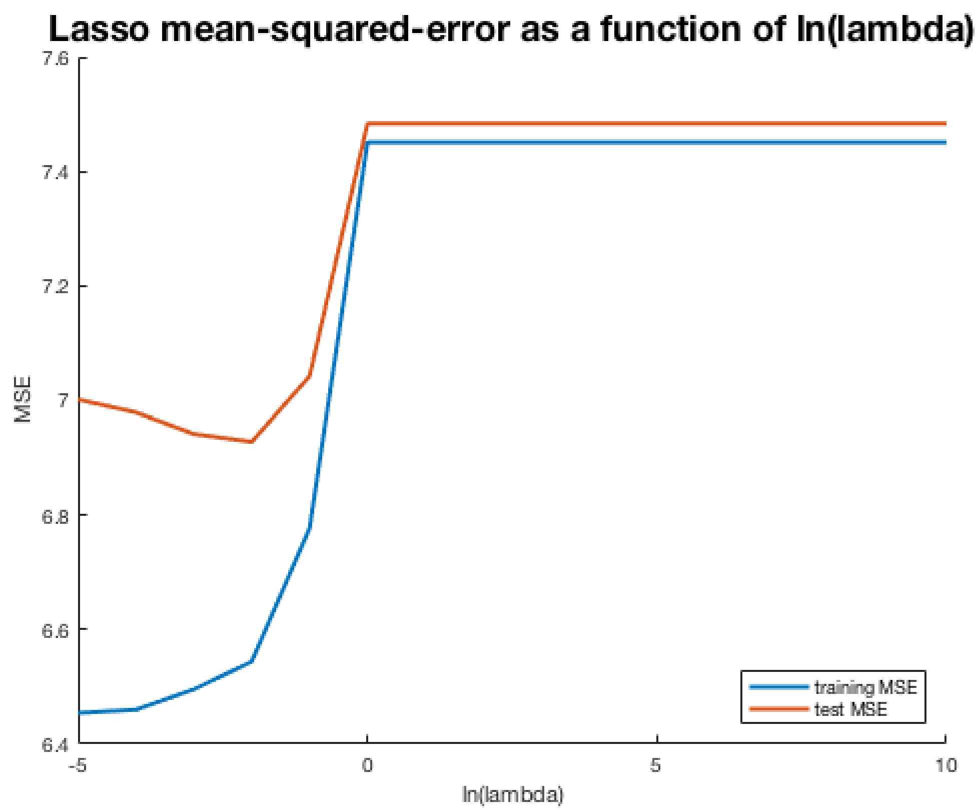
```
% initialize lasso weights to W_ridge
W_lasso = W_ridge;
% max iterations
tmax = 100;
% keep track of previous value for convergence
W_k_past = W_lasso;
numIterations = zeros(1,length(lambda));
coeffs = zeros(16,8); % same as W_lasso, rearranged for convenience
 later
% estimates
H_train = cell(1,length(lambda));
H_test = cell(1,length(lambda));
% MSEs
trainMSE = zeros(1,length(lambda));
testMSE = zeros(1,length(lambda));
% tuning parameter
for i=1:length(lambda)
    d = length(W_lasso{1});
    for t=1:tmax % interations
        for k=1:d % cycle through each coordinate
            % argmin(z) of cost()
            a_k = 2*sum(xTrainTilda(:,k).^2);
            c_k = 0;
            n = length(xTrainTilda);
            for j=1:n
                c_k = c_k + (2*xTrainTilda(j,k)*(ytrain(j)- ...
                    W_lasso{i}'*xTrainTilda(j,:)'+ ...
                    W_lasso{i}(k)*xTrainTilda(j,k)));
            end
            W_lasso{i}(k) = wthresh(c_k/a_k,'s',lambda(i));
        end
        % check for convergence
        if W_lasso{i} == W_k_past{i}
```

```
            break;
        end
        W_k_past{i} = W_lasso{i};
    end
    numIterations(i) = t;
    coeffs(i,:) = W_lasso{i};
    H_train{i} = xTrainTilda*W_lasso{i};
    H_test{i} = xTestTilda*W_lasso{i};
    trainMSE(i) = mse(ytrain,H_train{i});
    testMSE(i) = mse(ytest,H_test{i});
end
%disp(numIterations);

% (i) plot lasso coeff. of each feat. as fn. of ln(lambda)
figure(1);
hold on;
for m=1:size(coeffs,2)
plot(-5:10, coeffs(:,m), 'LineWidth',2);
end
xlabel('ln(lambda)');
ylabel('coefficients');
title('lasso coefficient of each feature as function of
 ln(lambda)', ...
    'FontSize', 15);
legend(names(1:end-1));

% (ii) plot MSE for training and test as function on ln(lambda)
figure(2);
hold on;
plot(-5:10,trainMSE, 'LineWidth',2);
plot(-5:10,testMSE, 'LineWidth',2);
title('Lasso mean-squared-error as a function of ln(lambda)', ...
    'FontSize', 18);
xlabel('ln(lambda)');
ylabel('MSE');
legend('training MSE', 'test MSE', 'Location','southeast');
```


lasso coefficient of each feature as function of ln(lambda)
coefficients
ln(lambda)
lcavol
lweight
age
lbph
svi
lcp
gleason
pgg45
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-5
0
5
10


Lasso mean-squared-error as a function of ln(lambda)
MSE
ln(lambda)
training MSE
test MSE
7.6
7.4
7.2
7
6.8
6.6
6.4
-5
0
5
10

(b)

What happens to the lasso coefficients as lambda gets larger? What are the 2 most meaningful f eatures with lasso (last 2 features to converge)? How can this information be used?

```
% As the tuning parameter increases the cost function this term begins
 to
% outway the first term which decreases due to increased flexibility.
 This
% causes the coefficients to move toward zero.

% From the graph, it looks like 'lcavol' and 'lweight' are the last
% features to converge. You can use subset selection to model on the
 most
% important features that were found. This reduces the dimensionality
 of
% the data which will alleviate exponential growth as a function of
% dimensions for many models such as K-nearest-neighbors.
```

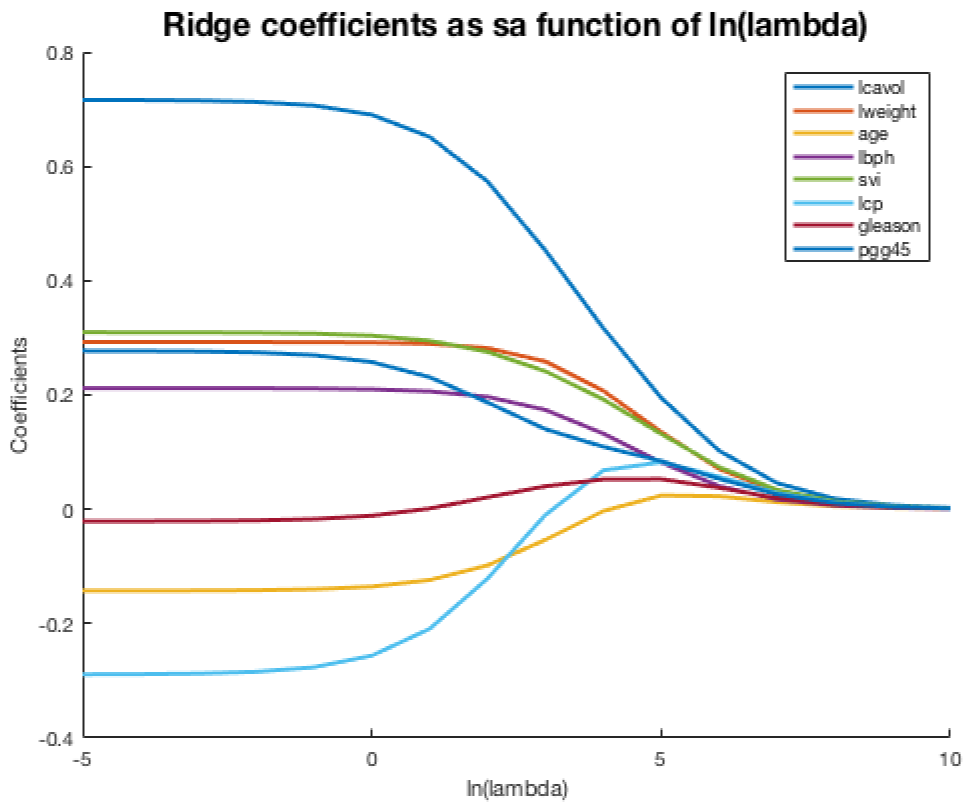
(c)

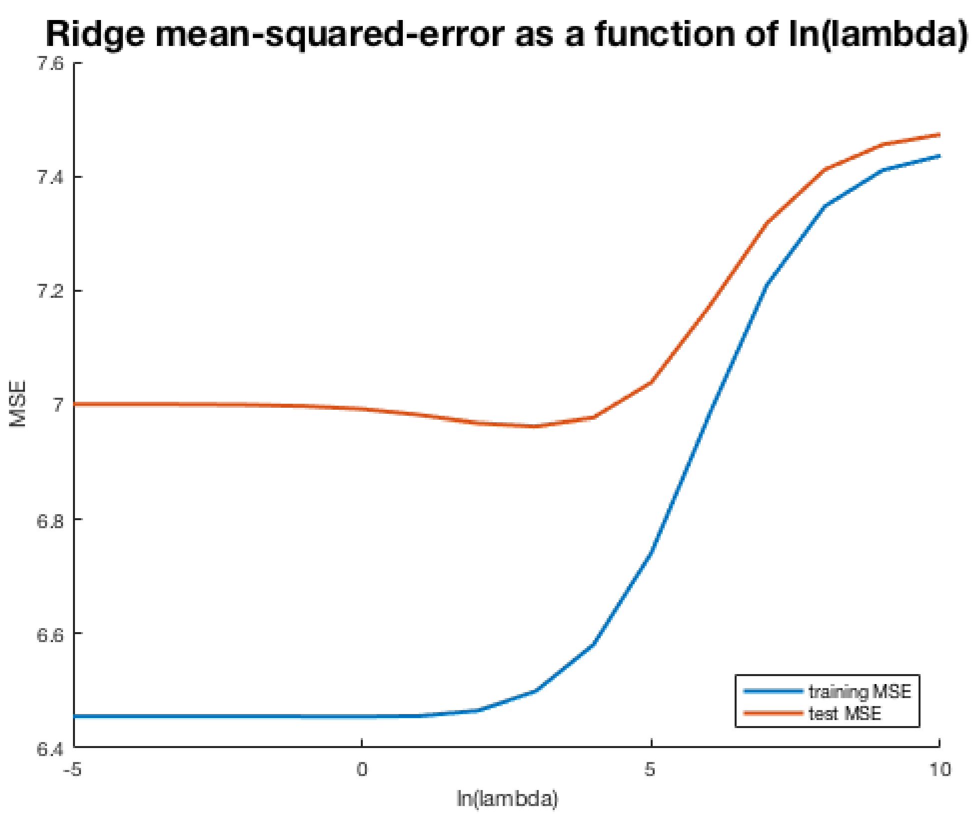
Use MATLAB's ridge regression on this dataset. (i) Again, plot the ridge coefficients of each feature (8 total - yielding 8 curves) against ln(lambda) for values of ln(lambda) ranging from -5 to 10 in steps of 1. Label the plot accordingly. (ii) in another figure, plot the mean-squared error (MSE) of both the training and testing data.

```
% (i) plot ridge coefficients against ln(lambda)
figure(3);
hold on;
for k=1:size(coeff_ridge,2)
    plot(-5:10,coeff_ridge(:,k), 'LineWidth',2);
end
title('Ridge coefficients as sa function of ln(lambda)', ...
    'FontSize', 16);
xlabel('ln(lambda)');
ylabel('Coefficients');
legend(names(1:end-1));

% (i) plot ridge mean-squared-error for train/test

figure(4);
hold on;
plot(-5:10,ridgeTrainMSE, 'LineWidth',2);
plot(-5:10,ridgeTestMSE, 'LineWidth',2);
title('Ridge mean-squared-error as a function of ln(lambda)', ...
    'FontSize', 18);
xlabel('ln(lambda)');
ylabel('MSE');
legend('training MSE', 'test MSE', 'Location','southeast');
```


Ridge coefficients as sa function of ln(lambda)
Coefficients
ln(lambda)
lcavol
lweight
age
lbph
svi
lcp
gleason
pgg45
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-5
0
5
10


Ridge mean-squared-error as a function of ln(lambda)
MSE
ln(lambda)
training MSE
test MSE
7.6
7.4
7.2
7
6.8
6.6
6.4
-5
0
5
10

(d)

What happens to the ridge coefficients as lambda becomes larger? How is this different from lasso?

```
% The ridge coefficients also shrink as a function of lambda. The
% difference compared to lasso though, is that the ridge coefficients
 seem
% to approach zero, but not quite get there. I believe this means that
 it
% does not create sparsity so it cannot be used for feature selection
 like
% in the case of lasso.
```

Published with MATLAB® R2017a