Exercise 2-2 - Getting Familiar with Regularization in Matlab

Table of Contents

Felix Wittich, 01.06.2021

```
clear all
close all
clc
```

a)

Use the randn() function to generate a predictor Xtrain of length N = 1000, as well as a noise vector eps of length N = 1000 with a standard deviation of 0.8. Make sure to use rng(1000) prior to staring part a) to ensure consistent results.

```
rng(1000)

N = 1000;
X = randn(N,1);
eps = 0.8*randn(N,1);

% Generate a response vector Y of length n = 100 according to
% the model
% Y = beta_0 + beta_1X + beta_2X^2 + beta_3X^3 + epsilon,
% where beta_0=2, beta_1=3, beta_2=-1, and beta_3=0.5. Set seed to 1998.

beta_0 = 2;
beta_1 = 3;
beta_2 = -1;
beta_3 = 2;
Y = beta_0 + beta_1*X + beta_2*X.^2 + beta_3*X.^3 + eps;

Ytest = Y(101:end);
Ytrain = Y(1:100);
Xtest = X(101:end,:);
Xtrain = X(1:100,:);

% Now fit a lasso model to the simulated data, again using X,X2,
```

```
% . . . , X9 as predictors. Use cross-validation to select the optimal
% value of ?. Create plots of the cross-validation error as a function
% of ?. Pick the model with minium MSE CV. Report the resulting coefficient
 estimates, and discuss the
% results obtained.
```

b)

Use the logspace() function to generate a sequence of 100 logarithmically spaced values for λ in between 102 and 10–5 specifying the degree of regularization.

```
lambda = logspace(2,-5,100);
```

c)

Generate regression matrices for training and test data sets

```
for li = 1:9
    PhiTrain(:,li) = Xtrain.^li;
    PhiTest(:,li) = Xtest.^li;
end

% estimate using lasso
[B, Stats] = lasso(PhiTrain,Ytrain,'CV',10,'Lambda',lambda,'PredictorNames',
{'x1','x2','x3','x4','x5','x6','x7','x8','x9'});
```

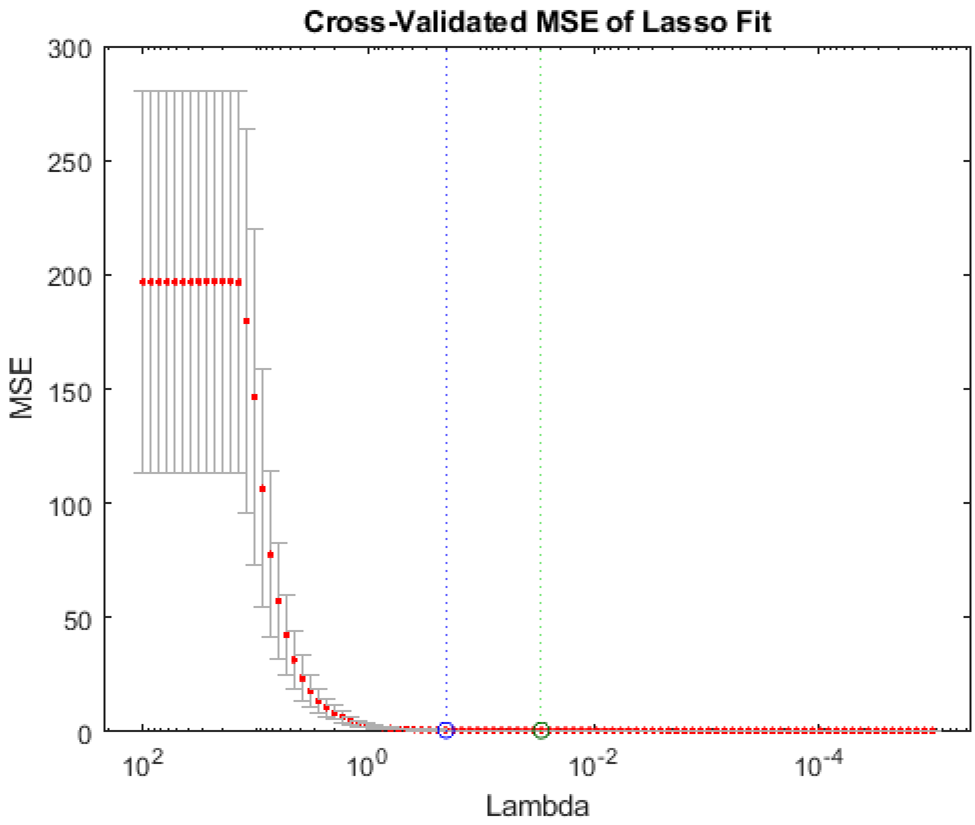
d)

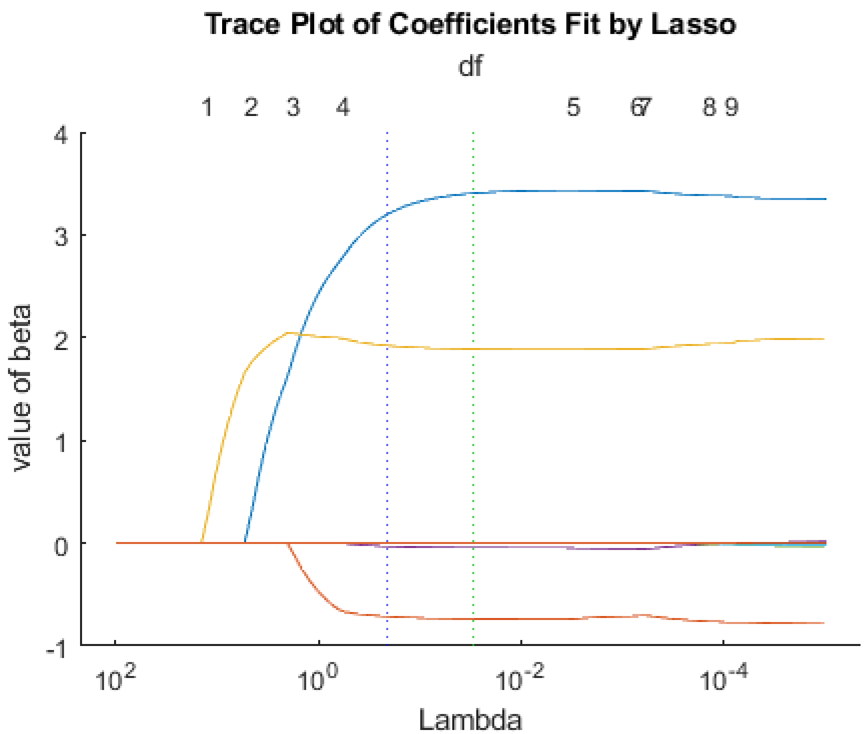
plot lasso results

```
lassoPlot(B, Stats, 'PlotType', 'CV','PredictorNames',
{'x1','x2','x3','x4','x5','x6','x7','x8','x9'})
lassoPlot(B, Stats, 'PlotType', 'Lambda','XScale','log','PredictorNames',
{'x1','x2','x3','x4','x5','x6','x7','x8','x9'})
ylabel('value of beta')


beta_0_Lasso = Stats.Intercept(Stats.IndexMinMSE);
BetaLasso = [beta_0_Lasso B(:,Stats.IndexMinMSE)']'
```

```
BetaLasso =

    1.7669
    3.4040
   -0.7361
    1.8886
   -0.0443
         0
         0
         0
```

```
0
0
```

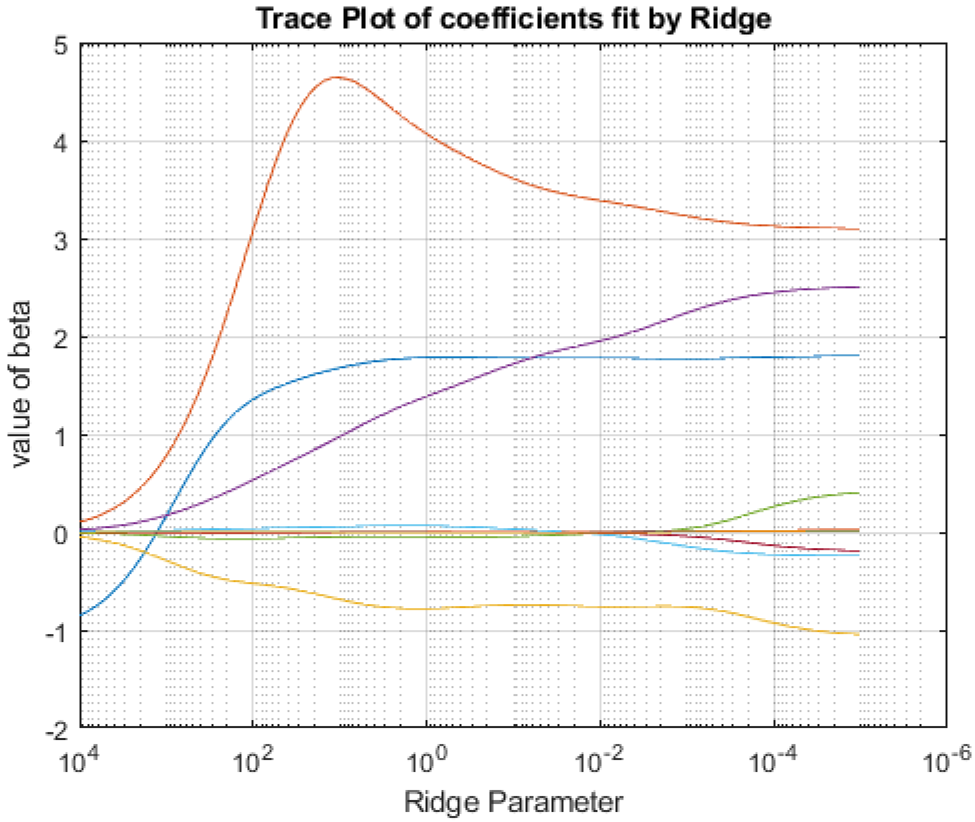

Cross-Validated MSE of Lasso Fit
300
250
200
150
100
50
0
MSE
10^2
10^0
10^-2
10^-4
Lambda



e)

Perform Ridge Regression

```
lambda = logspace(4,-5,100);
B = ridge(Ytrain,PhiTrain,lambda,0);
figure
semilogx(lambda,B)
grid on
xlabel('Ridge Parameter')
ylabel('value of beta')
title('Trace Plot of coefficients fit by Ridge')
set(gca, 'XDir','reverse')
```

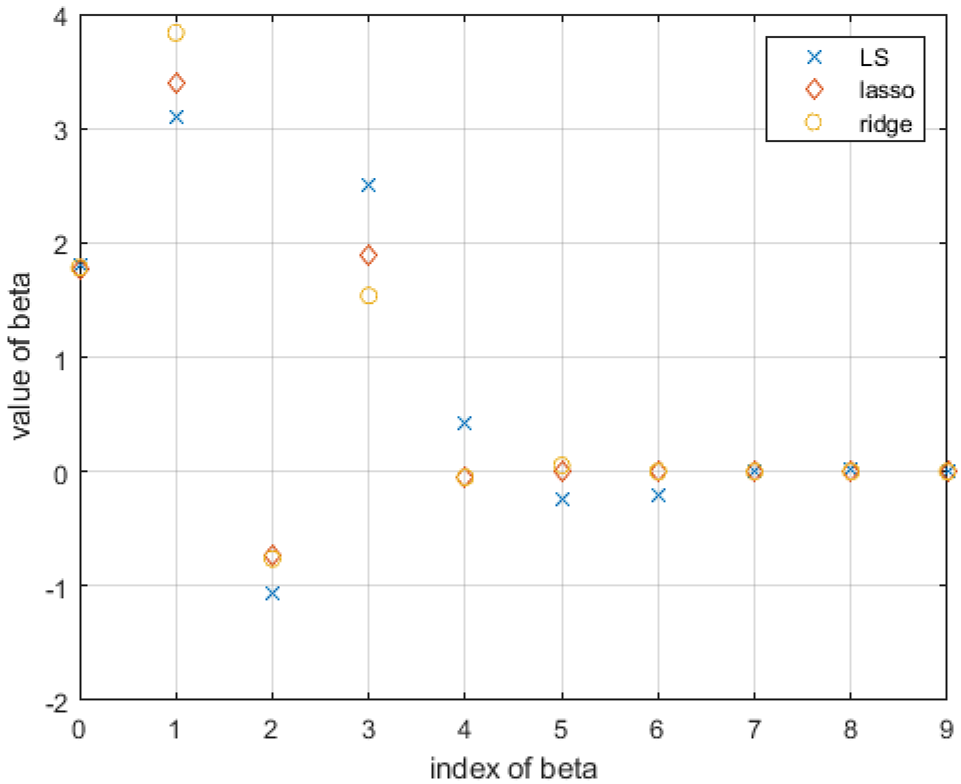


f)

Least Squares Estimation

```
BetaLS = [ones(size(Ytrain)) PhiTrain]\Ytrain;
BetaRidge = B(:,Stats.IndexMinMSE)

figure
plot([0:9],BetaLS,'x')
hold all
plot([0:9],BetaLasso,'d')
plot([0:9],BetaRidge,'o')
legend('LS','lasso','ridge')
grid on
ylabel('value of beta')
xlabel('index of beta')

% evaluate on test data
yLS = [ones(size(Ytest)) PhiTest]*BetaLS;
yLasso = [ones(size(Ytest)) PhiTest]*BetaLasso;
yRidge = [ones(size(Ytest)) PhiTest]*BetaRidge;

RMSEls = sqrt(mean((Ytest-yLS).^2))
RMSElasso = sqrt(mean((Ytest-yLasso).^2))
RMSEridge = sqrt(mean((Ytest-yRidge).^2))
```

```
BetaRidge =

    1.7833
    3.8347
   -0.7635
    1.5381
   -0.0494
    0.0547
    0.0010
    0.0004
    0.0005
   -0.0001


RMSEls =

    3.7188


RMSElasso =

    0.8462


RMSEridge =

    0.8671
```



Published with MATLAB® R2021b