Silvia Ionescu
10-28-2016

Assignment 7

Problem 7.1 – Exploring Boston Housing Data with Regression Trees

For this part we use the housing dataset that has 13 features and is used to estimate the output, median value of owner-occupied homes.

a) The dataset contained training and testing pairs, and the names of the features and the output was listed in feature_names and output_name. A visualization of the training data regression tree was performed using the build-in Matlab function. The regression tree has a minimum of 20 observations per leaf and it is shown below:

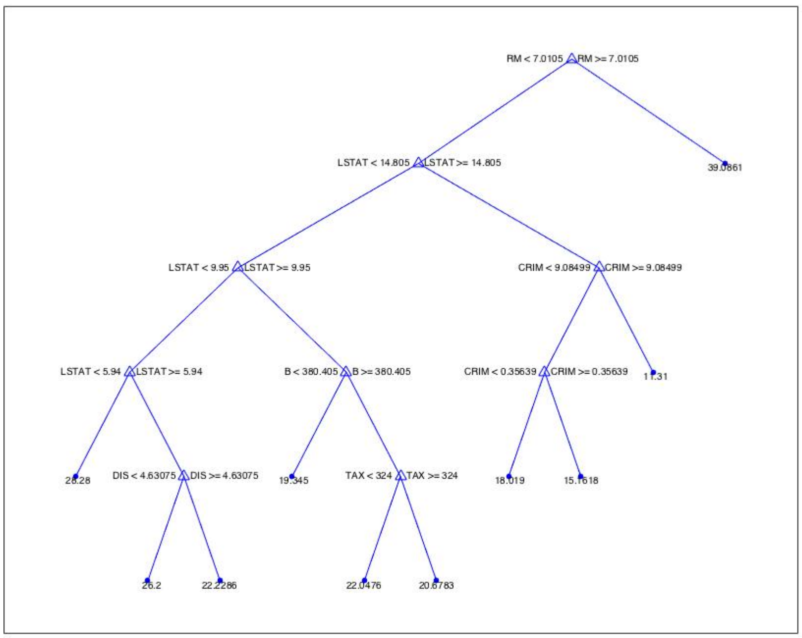

Fig. 1: Training data regression tree with minimum of 20 observations per leaf

b) For the trained regression tree that has a minimum of 20 observations per leaf the estimated median value of owner-occupied homes in 1000's , ***MEDV = 22.0476*** , for the test vector: CRIM = 5, ZN = 18, INDUS = 2.31, CHAS = 1, NOX = 0.5440, RM = 2, AGE = 64, DIS =3.7, RAD = 1, TAX = 300, PTRATIO = 15, B = 390, LSTAT = 10.

c) The plot of the mean absolute error, MAE, of the training and testing data as a function of the minimum observations per leaf ranging from 1 to 25 is shown below.

Silvia Ionescu
10-28-2016

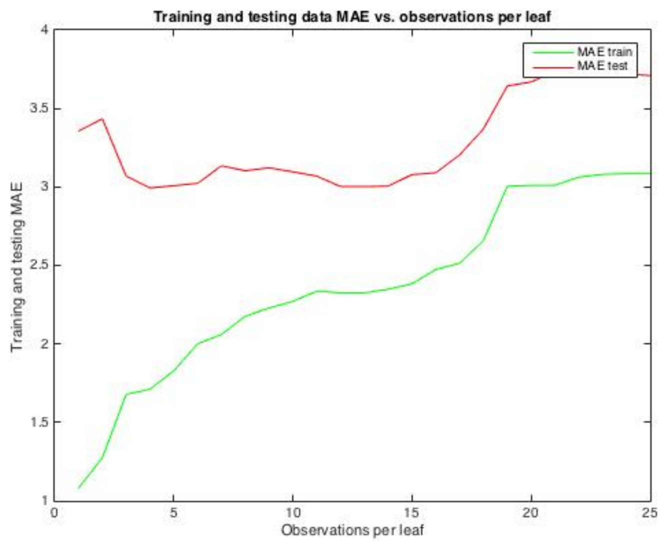

Fig. 2: MAE of training and testing data as a function of observations per leaf ranging from 1 to 25

It can be noticed that the MSE of the train and test dataset plateaus (stabilized in the sense that it does not chance so much as before) at 20 observations per leaf.

Problem 7.2 – Ordinary Least Squares (OLS) vs. Robust Linear Regression

a) Ordinary least regression (OLS) was implemented for input xdata and output ydata.

The input data matrix, xdata, yields a unique solution, because the matrix is full rank.

Given the below equation:

$$h_{OLS}(x) = x^T w_{OLS} + b_{OLS} \tag{1}$$

The values of $\boldsymbol{w_{OLS} = 0.0233}$ ***and*** $\boldsymbol{b_{OLS} = 2.9738}$ were calculated according to Eq. 2

$$\begin{aligned} \widehat{w}_{OLS} &= \left(\hat{\Sigma}_x\right)^{-1}\hat{\Sigma}_{xy} \\ \hat{b}_{OLS} &= \hat{u}_y - (\widehat{w}_{OLS})^T \hat{u}_x \end{aligned} \tag{2}$$

The mean square error ***MSE = 0.2588*** and the mean absolute error ***MAE = 0.3174*** were calculated according to Eq. 3

$$\begin{aligned} MSE &= \frac{1}{n}\sum_{j=1}^{n} |y_j - h_{OLS}(x_j)|^2 \\ MAE &= \frac{1}{n}\sum_{j=1}^{n} |y_j - h_{OLS}(x_j)| \end{aligned} \tag{3}$$

Silvia Ionescu
10-28-2016

b) Implemented a robust linear regression for the input and output in the linear_data.mat by using Matlab's robust regression. To include 1 a new component in every feature vector const was set to 'on'. The robust fit function was run with the loss functions: 'cauchy', 'fair', 'huber', 'talwar', and the default value for the tune parameter.
Table 1 shows the MSE and MAE for OLS and robust linear regression for different loss functions:

	MAE	MSE
OLS	0.3174	0.2588
Cauchy	0.2426	0.2995
Fair	0.2468	0.2860
Huber	0.2450	0.2921
Talwar	0.2434	0.3024

Table1: MSE and MAE for OLS and robust linear regression for different loss functions.

The values of $\boldsymbol{w_{huber}} = \boldsymbol{0.1274}$ ***and*** $\boldsymbol{b_{huber}} = \boldsymbol{3.0943}$ were calculated.
A plot that compares all the methods: OLS and robust linear regression for all loss functions is shown below:

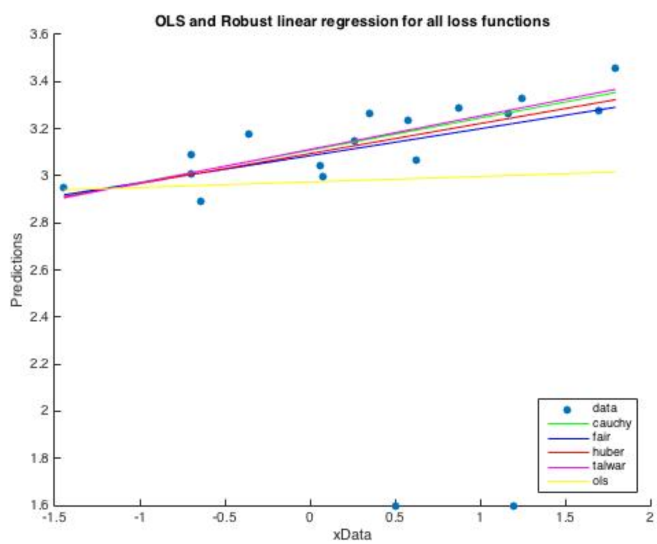

Fig. 3: OLS and robust linear regression for all loss functions.

In the above graph we notice that OLS is not the best fit in the sense that its is influenced by the outliers, while the robust regression for all the loss functions gives a better fit for our data then OLS.

Problem 7.3 - Overfitting and Ridge Regression

The quad_data.mat was used for this part, along with it's corresponding training and test datasets.

Silvia Ionescu
10-28-2016

a) Used OLS to fit polynomials of degree d = 1,…, 14 to the training data. The Matlab ridge regression function was used with the vector of ridge parameters k set to zero in order to reduce the ridge regression to OLS. The 'scaled' parameter was set to zero so that the coefficients estimated are on the same scale as the original data. A plot of the training points with polynomial curves of degree 2, 6, 10, 14 is shown below:

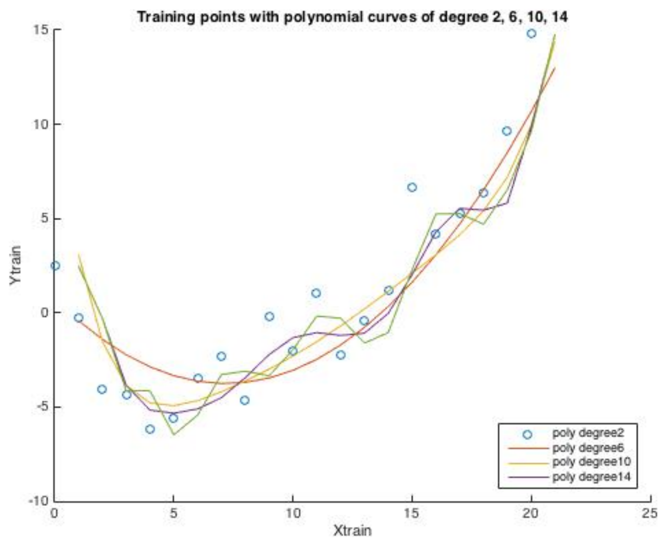

Fig. 4: Training points with polynomial curves of degree 2, 6, 10, and 14.

The plot of the training and testing data MSE as a function of polynomial degree d= 1,…,14 is shown in Fig. 5.

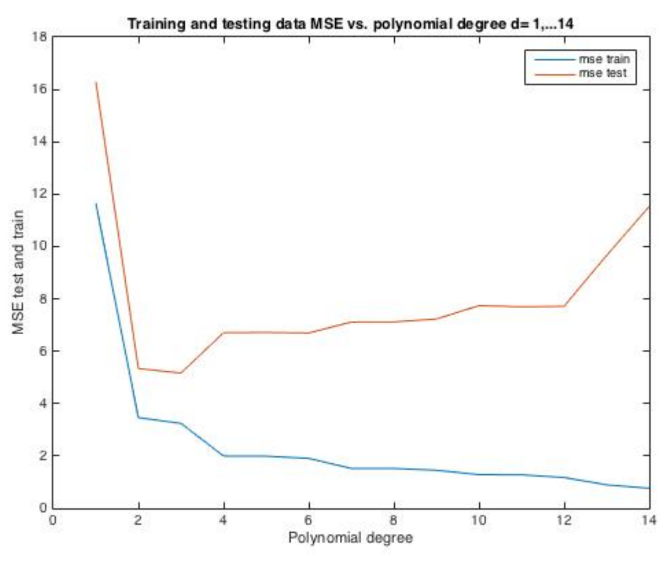

Fig.5: Training and testing data MSE as a function of polynomial degree d= 1,…,14.

Silvia Ionescu
10-28-2016

The MSE test is minimum at polynomial degree 3, while MSE train has the smallest value at degree 14(and it looks like it would keep going down). This means that our training data will not give us the best prediction for out test dataset.

b) Added l_2 regularization to linear regression to alleviate over-fitting. A plot of the mean squared error of the training and testing data as a function of ln λ(range -25 to 5) is shown below:

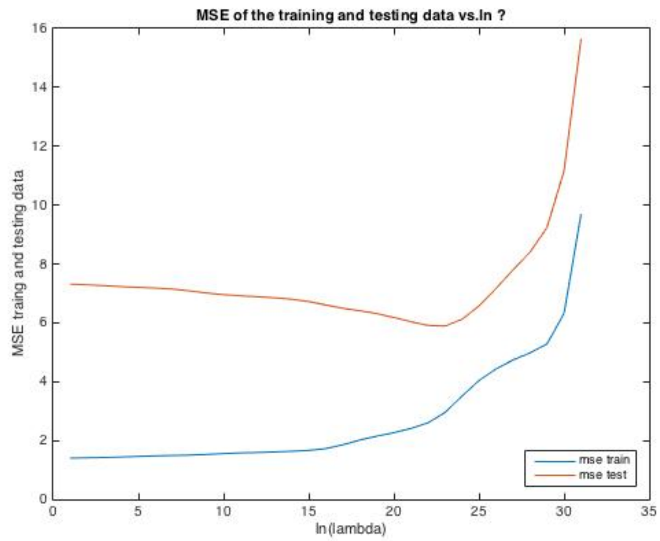

Fig. 6: MSE of the training and testing data vs.ln λ(range -25 to 5)

Testing points plotted along with the non-regularized OLS degree 10 polynomial fit and the l_2-regularized degree 10 fit, with the smallest MSE:

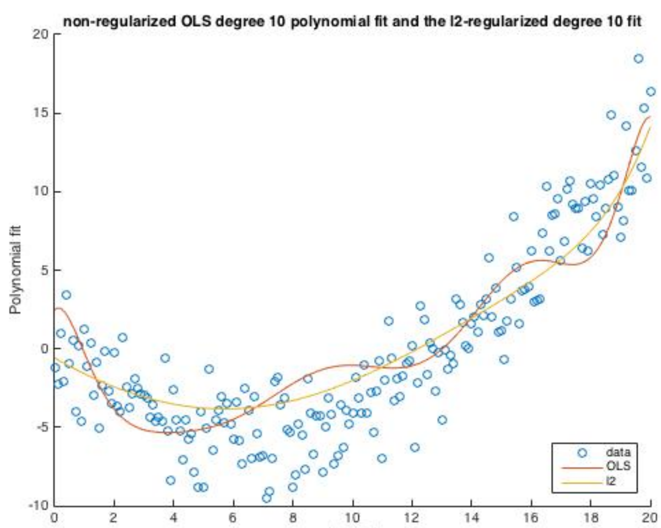

Fig. 7: Non-regularized OLS degree 10 polynomial fit and the l_2- regularized degree 10 fit

Looks like both OLS and l_2-regularized are decent fits at polynomial of degree 10. The l_2-regularized fit looks a bit better, since the OLS seems to fluctuate more between data points.

c) Plot of the ridge coefficients for a polynomial of degree 4 as a function of ln λ, with ln λ ranging form -25 to 5.
The five curves are plotted below:

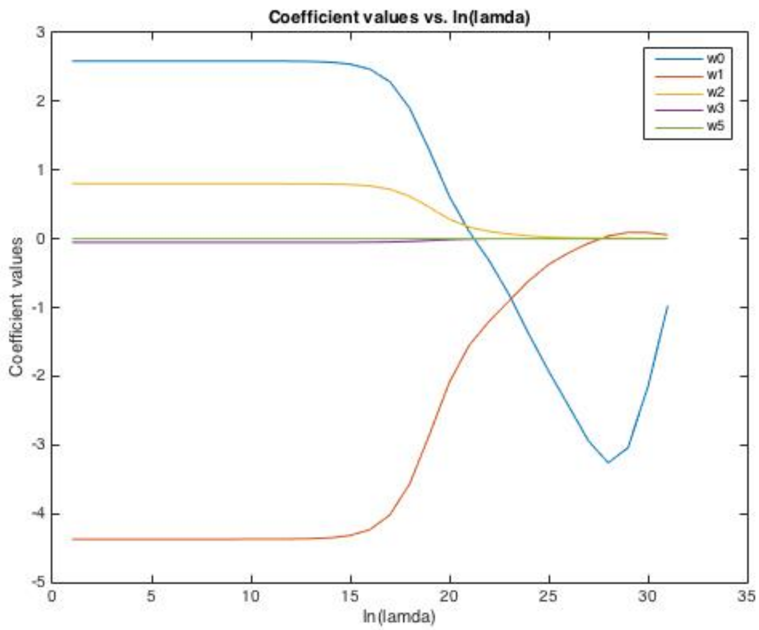

Fig. 8: Ridge coefficients (degree 4) as a function of ln λ.

It looks like coefficients w1 to w5 go to zero as λ increases.

Problem 7.4 – Lasso vs. Ridge

For this part a lasso regression was applied to the multi-dimensional prostate cancer dataset.

a) Implemented a cyclic coordinate descent for lasso. The data was centered and the coefficients were updated at 100 times. A plot of the lasso coefficients of each feature as a function of ln λ, with ln λ ranging from -5 to 10, is shown below:

Silvia Ionescu
10-28-2016

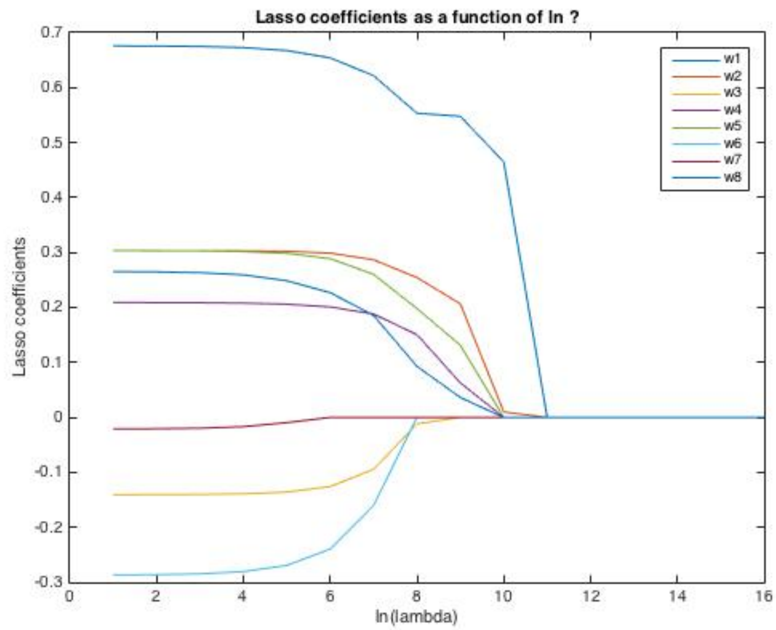

Fig. 9: Lasso coefficients of each feature as a function of ln λ.

Plot of the MSE of both training and testing data as a function of ln λ is shown below:

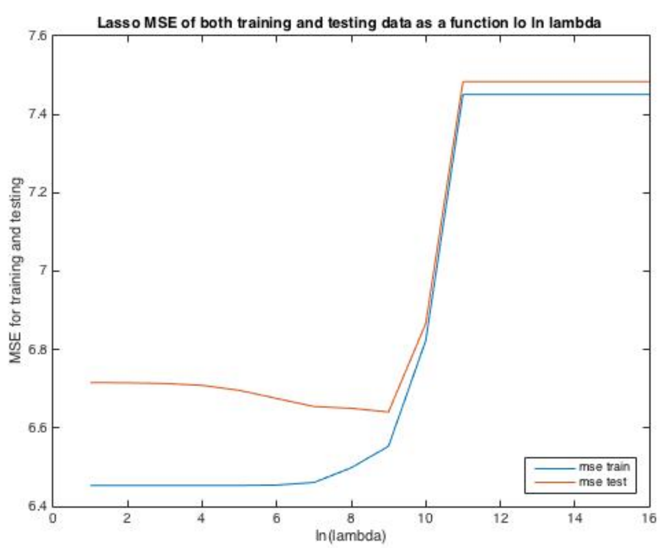

Fig. 10: MSE of both training and testing data as a function of ln λ.

b) As λ increases the lasso coefficients decrease to zero. The 2 most meaningful features with lasso are the first two w's (w1, w2). These could be used in reducing the dimension of our dataset. This gives us an idea of which coefficients are more important then other and therefore we can decide if we could cut down on the number of features.

c) Ridge regression coefficients vs. ln λ, ranging form -5 to 10, are shown below:

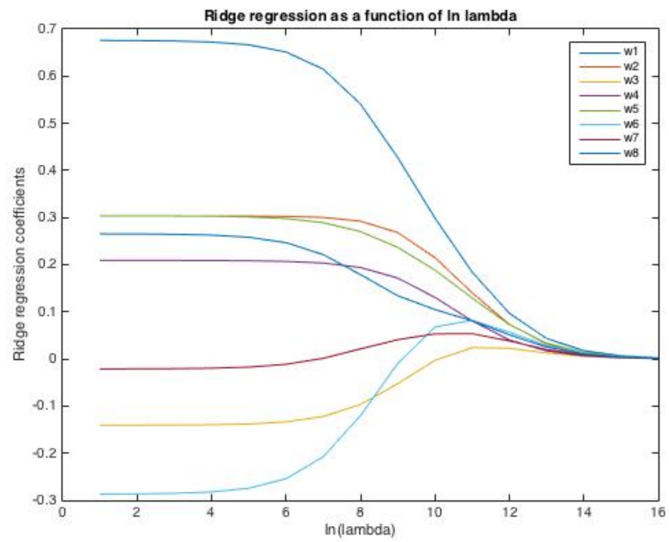

Fig. 11: Ridge regression coefficients vs. ln λ.

The MSE for ridge regression for both the train set and the test set are shown below:

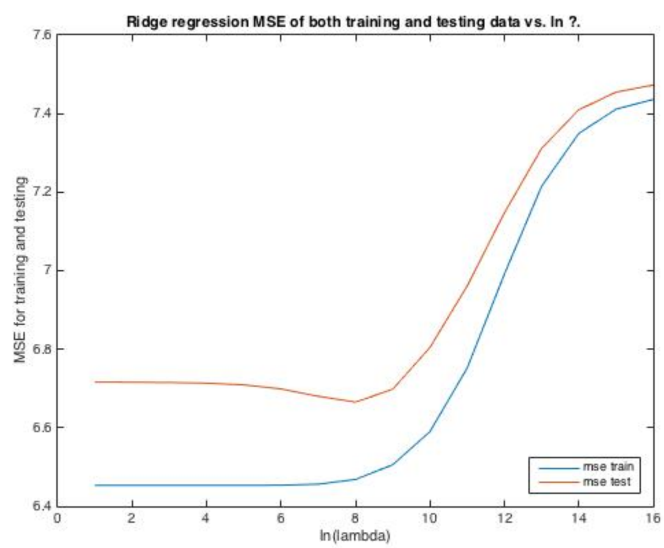

Fig. 12: MSE for ridge regression for the training and testing sets

d) The ridge regression coefficients also decrease towards zero as λ becomes larger, however they decrease slower then the lasso coefficients.

Silvia Ionescu
10-28-2016

```
% Problem 7.1 ? Exploring Boston Housing Data with Regression Trees
% Silvia Ionecu

clear; close all; clc;

% train and test datasets
housing_data = load('housing_data.mat');

train_data = housing_data.Xtrain;
train_label = housing_data.ytrain;

test_data = housing_data.Xtest;
test_label = housing_data.ytest;
feature_names = housing_data.feature_names;

t = classregtree(train_data,train_label, 'method', 'regression', 'minleaf',
20, 'names', feature_names);
view(t);

hw7_1b = [5 18 2.31 1 0.5440 2 64 3.7 1 300 15 390 10];
prediction_b = eval(t,hw7_1b);

% Part c
for i = 1:25
    t2 = classregtree(train_data,train_label, 'method', 'regression',
'minleaf', i, 'names', feature_names);
    predic_train_c = eval(t2,train_data);
    predic_test_c = eval(t2, test_data);
    err_train(i) = immse(predic_train_c,train_label);
    err_test(i) = immse(predic_test_c, test_label);

    mae_train(i) = 1/length(train_label)*sum(abs(train_label-
predic_train_c));
    mae_test(i) = 1/length(test_label)*sum(abs(test_label-predic_test_c));
end

figure(2)
plot(1:25,  mae_train, 'g');
hold on;
plot(1:25, mae_test, 'r');
hold off;
legend('MAE train', 'MAE test');
xlabel('Observations per leaf');
ylabel('Training and testing MAE ');
title('Training and testing data MAE vs. observations per leaf');
```

Silvia Ionescu
10-28-2016

```
% Problem 7.2 ? Ordinary Least Squares (OLS) vs. Robust Linear Regression
% Silvia Ionescu

clear; close all; clc;

% train and test datasets
housing_data = load('linear_data.mat');
xData = housing_data.xData;
yData = housing_data.yData;

ux = sum(xData)/length(xData);
uy = sum(yData)/length(yData);

cov_x = 1/length(xData)*((xData - ux)'*(xData - ux));
cov_xy = 1/length(xData)*sum((yData - uy).*(xData - ux));

w_ols = (cov_x)^-1 * cov_xy;
b_ols = uy - w_ols * ux;

xData_ux = xData - ux;
yData_uy = yData - uy;

% calculate MSE - mean squared error
h_ols = xData*w_ols + b_ols;
mse_ols = 1/length(xData)*sum((yData-h_ols).^2);
mae_ols = 1/length(xData)*sum(abs(yData-h_ols));

% Part b
b_c = robustfit(xData, yData, 'cauchy');
b_f = robustfit(xData, yData, 'fair');
b_h = robustfit(xData, yData, 'huber');
b_t = robustfit(xData, yData, 'talwar');

% predicted y
y_pred_cauchy = b_c(1)+b_c(2)*xData;
y_pred_fair = b_f(1)+b_f(2)*xData;
y_pred_huber = b_h(1)+b_h(2)*xData;
y_pred_talwar = b_t(1)+b_t(2)*xData;

% calculate mse for the above loss functions
mse_cauchy = 1/length(xData)*sum((yData - y_pred_cauchy).^2);
mse_fair = 1/length(xData)*sum((yData - y_pred_fair).^2);
mse_huber = 1/length(xData)*sum((yData - y_pred_huber).^2);
mse_talwar = 1/length(xData)*sum((yData - y_pred_talwar).^2);

% calculate mae for the above loss functions
mae_cauchy = 1/length(xData)*sum(abs(yData - y_pred_cauchy));
mae_fair = 1/length(xData)*sum(abs(yData - y_pred_fair));
mae_huber = 1/length(xData)*sum(abs(yData - y_pred_huber));
mae_talwar = 1/length(xData)*sum(abs(yData - y_pred_talwar));

figure(1)
scatter(xData,yData,'filled');  hold on
plot(xData, y_pred_cauchy,'g')
hold on;
```

```
plot(xData, y_pred_fair,'b')
hold on;
plot(xData, y_pred_huber,'r')
hold on;
plot(xData, y_pred_talwar,'m')
hold on;
plot(xData, h_ols,'y');
lgd = legend('data','cauchy', 'fair', 'huber', 'talwar', 'ols');
lgd.Location = 'southeast';
ylabel('Predictions');
xlabel('xData');
title('OLS and Robust linear regression for all loss functions');
```

```
% Problem 7.3 - Overfitting and ridge regression
% Silvia Ionescu
```

Silvia Ionescu
10-28-2016

```
clear; close all; clc;

% train and test datasets
quad_data = load('quad_data.mat');
xtrain_a = quad_data.xtrain;
ytrain = quad_data.ytrain;

xtest_a = quad_data.xtest;
ytest = quad_data.ytest;

for i = 1:14
    xtrain(:,i) = xtrain_a.^i;
    xtest(:,i) = xtest_a.^i;

end

% Part 7_3a
% calculating coefficients for 1...14 degrees for training data
b1 = ridge(ytrain, xtrain(:,1), 0, 0);
b2 = ridge(ytrain, xtrain(:,1:2), 0, 0);
b3 = ridge(ytrain, xtrain(:,1:3), 0, 0);
b4 = ridge(ytrain, xtrain(:,1:4), 0, 0);
b5 = ridge(ytrain, xtrain(:,1:5), 0, 0);
b6 = ridge(ytrain, xtrain(:,1:6), 0, 0);
b7 = ridge(ytrain, xtrain(:,1:7), 0, 0);
b8 = ridge(ytrain, xtrain(:,1:8), 0, 0);
b9 = ridge(ytrain, xtrain(:,1:9), 0, 0);
b10 = ridge(ytrain, xtrain(:,1:10), 0, 0);
b11 = ridge(ytrain, xtrain(:,1:11), 0, 0);
b12 = ridge(ytrain, xtrain(:,1:12), 0, 0);
b13 = ridge(ytrain, xtrain(:,1:13), 0, 0);
b14 = ridge(ytrain, xtrain(:,1:14), 0, 0);

y1 = b1(2)'*xtrain(:,1)' + b1(1);
y2 = b2(2:end)'*xtrain(:,1:2)' + b2(1);
y3 = b3(2:end)'*xtrain(:,1:3)' + b3(1);
y4 = b4(2:end)'*xtrain(:,1:4)' + b4(1);
y5 = b5(2:end)'*xtrain(:,1:5)' + b5(1);
y6 = b6(2:end)'*xtrain(:,1:6)' + b6(1);
y7 = b7(2:end)'*xtrain(:,1:7)' + b7(1);
y8 = b8(2:end)'*xtrain(:,1:8)' + b8(1);
y9 = b9(2:end)'*xtrain(:,1:9)' + b9(1);
y10 = b10(2:end)'*xtrain(:,1:10)' + b10(1);
y11 = b11(2:end)'*xtrain(:,1:11)' + b11(1);
y12 = b12(2:end)'*xtrain(:,1:12)' + b12(1);
y13 = b13(2:end)'*xtrain(:,1:13)' + b13(1);
y14 = b14(2:end)'*xtrain(:,1:14)' + b14(1);

y_train_predict = [y1;y2;y3;y4;y5;y6;y7;y8;y9;y10;y11;y12;y13;y14]';

y1_t = b1(2)'*xtest(:,1)' + b1(1);
y2_t = b2(2:end)'*xtest(:,1:2)' + b2(1);
y3_t = b3(2:end)'*xtest(:,1:3)' + b3(1);
y4_t = b4(2:end)'*xtest(:,1:4)' + b4(1);
y5_t = b5(2:end)'*xtest(:,1:5)' + b5(1);
```

```
y6_t = b6(2:end)'*xtest(:,1:6)' + b6(1);
y7_t = b7(2:end)'*xtest(:,1:7)' + b7(1);
y8_t = b8(2:end)'*xtest(:,1:8)' + b8(1);
y9_t = b9(2:end)'*xtest(:,1:9)' + b9(1);
y10_t = b10(2:end)'*xtest(:,1:10)' + b10(1);
y11_t = b11(2:end)'*xtest(:,1:11)' + b11(1);
y12_t = b12(2:end)'*xtest(:,1:12)' + b12(1);
y13_t = b13(2:end)'*xtest(:,1:13)' + b13(1);
y14_t = b14(2:end)'*xtest(:,1:14)' + b14(1);

y_test_predict = [y1_t; y2_t; y3_t; y4_t; y5_t; y6_t; y7_t; y8_t; y9_t;
y10_t; y11_t; y12_t; y13_t; y14_t]';

% training point and polynomial curves of degrees 2,6,10,14
figure(1)
scatter(xtrain(:,1),ytrain);
hold on
plot(y2);
hold on
plot(y6);
hold on
plot(y10)
hold on
plot(y14)
lgd = legend ('poly degree2', 'poly degree6', 'poly degree10', 'poly
degree14');
lgd.Location = 'southeast';
title('Training points with polynomial curves of degree 2, 6, 10, 14');
ylabel('Ytrain');
xlabel('Xtrain')
hold off

% calculate mse for train data
for j = 1:14
    mse_train(j) = 1/length(ytrain)*sum((ytrain-y_train_predict(:,j)).^2);
    mse_test(j) = 1/length(ytest)*sum((ytest-y_test_predict(:,j)).^2);
end

% mean-squared error as a function of polynomial degree

figure(2)
plot(1:14, mse_train);
hold on
plot(1:14, mse_test);
legend('mse train', 'mse test');
xlabel('Polynomial degree');
ylabel('MSE test and train');
title ('Training and testing data MSE vs. polynomial degree d= 1,...14');
hold off;

% Part b
reg = exp(-25:5);
for k = 1:length(reg)
    b10_ridge_train = ridge(ytrain, xtrain(:,1:10), reg(k), 0);
    y10_ridge_train = b10_ridge_train(2:end)'*xtrain(:,1:10)' +
b10_ridge_train(1);
    mse_train_ridge(k) = 1/length(ytrain)*sum((ytrain-y10_ridge_train').^2);
```

```
    y10_ridge_test(:,k) = (b10_ridge_train(2:end)'*xtest(:,1:10)' +
b10_ridge_train(1))';
    mse_test_ridge(k) = 1/length(ytest)*sum((ytest-y10_ridge_test(:,k)).^2);

    b4_ridge_train(:,k) = ridge(ytrain, xtrain(:,1:4), reg(k), 0);

end
figure(3)
plot(mse_train_ridge);
hold on
plot(mse_test_ridge);
hold off
lgd = legend('mse train', 'mse test');
lgd.Location = 'southeast';
xlabel('ln(lambda)');
ylabel('MSE traing and testing data');
title('MSE of the training and testing data vs.ln ?');

min_test_mse = find(mse_test_ridge == min(mse_test_ridge));
% Part 3bii
figure(4)
scatter(xtest(:,1),ytest);
hold on
plot(xtest(:,1),y10_t)
hold on
plot(xtest(:,1),y10_ridge_test(:,min_test_mse));
xlabel('xtest data');
ylabel('Polynomial fit');
lgd = legend('data','OLS', 'l2');
lgd.Location = 'southeast';
title('non-regularized OLS degree 10 polynomial fit and the l2-regularized
degree 10 fit');

% Part 3c
figure(5)
plot(b4_ridge_train(1,:))
hold on
plot(b4_ridge_train(2,:))
hold on
plot(b4_ridge_train(3,:))
hold on
plot(b4_ridge_train(4,:))
hold on
plot(b4_ridge_train(5,:))
legend('w0', 'w1', 'w2', 'w3', 'w5');
xlabel('ln(lamda)');
ylabel('Coefficient values');
title('Coefficient values vs. ln(lamda)');




% Problem 7.4 - Lasso vs Ridge
```

Silvia Ionescu
10-28-2016

```
% Silvia Ionescu

clear; close all; clc;
% train and test datasets
quad_data = load('prostateStnd.mat');
xtrain = quad_data.Xtrain;
ytrain = quad_data.ytrain;
xtest = quad_data.Xtest;
ytest = quad_data.ytest;

mean = sum(xtrain,1)/67;
for i = 1:size(xtrain,2)
    xtrain(:,i) = xtrain(:,i) - mean(:,i);
end

% Part 4c - ridge regresion
reg = exp(-5:10);
for j = 1:length(reg)
    b_ridge_train(:,j) = ridge(ytrain, xtrain, reg(j), 0);
    w(:,j) = b_ridge_train(2:end,j);
    for t = 1:100
        for k = 1:size(xtrain,2)
            a = 2*sum(xtrain(:,k).^2 );
            c = 2*xtrain(:,k)'*(ytrain - (w(:,j)'*xtrain')' +
w(k,j).*xtrain(:,k));
            w(k,j) = sign(c/a)* max(0, abs(c/a)-(reg(j)/a));
        end
    end
end

figure(1)
plot(w(1,:));
hold on
plot(w(2,:));
hold on
plot(w(3,:));
hold on
plot(w(4,:));
hold on
plot(w(5,:));
hold on
plot(w(6,:));
hold on
plot(w(7,:));
hold on
plot(w(8,:));
legend('w1', 'w2', 'w3', 'w4', 'w5', 'w6', 'w7','w8');
ylabel('Lasso coefficients');
xlabel('ln(lambda)');
title('Lasso coefficients as a function of ln lambda');

% calculate predictions
for p =1:length(reg)
    y_pred_train = (w(:,p)'*xtrain')';
    y_pred_test = (w(:,p)'*xtest')';
    mse_train(p) = 1/length(ytrain)*sum((ytrain - y_pred_train).^2);
    mse_test(p) = 1/length(ytest)*sum((ytest - y_pred_test).^2);
```

```
    y_pred_train_ridge = (b_ridge_train(2:end,p)'*xtrain')';
    y_pred_test_ridge = (b_ridge_train(2:end,p)'*xtest')';
    mse_train_ridge(p) = 1/length(ytrain)*sum((ytrain -
y_pred_train_ridge).^2);
    mse_test_ridge(p) = 1/length(ytest)*sum((ytest - y_pred_test_ridge).^2);
end

figure(2)
plot(mse_train)
hold on
plot(mse_test);
lgd = legend('mse train', 'mse test');
lgd.Location = 'southeast';
ylabel('MSE for training and testing')
xlabel('ln(lambda)');
title('Lasso MSE of both training and testing data as a function lo ln
lambda')
hold off

% ridge regression
figure(3)
plot(b_ridge_train(2,:));
hold on
plot(b_ridge_train(3,:));
hold on
plot(b_ridge_train(4,:));
hold on
plot(b_ridge_train(5,:));
hold on
plot(b_ridge_train(6,:));
hold on
plot(b_ridge_train(7,:));
hold on
plot(b_ridge_train(8,:));
hold on
plot(b_ridge_train(9,:));
legend('w1', 'w2', 'w3', 'w4', 'w5', 'w6', 'w7','w8');
ylabel('Ridge regression coefficients');
xlabel('ln(lambda)');
title('Ridge regression as a function of ln lambda');

figure(4)
plot(mse_train_ridge)
hold on
plot(mse_test_ridge);
lgd = legend('mse train', 'mse test');
lgd.Location = 'southeast';
ylabel('MSE for training and testing')
xlabel('ln(lambda)');
title('Ridge regression MSE of both training and testing data vs. ln ?. ')
hold off
```