Table of Contents

..... 1
load data 1
(a) 1
Observations 4
(b) 5
(c) 8


```
% Austin Welch
% EC503 HW7.3
% Overfitting and Ridge Regression
```

load data

```
clear; clc;
rng default; % For reproducibility
load('quad_data.mat');
```

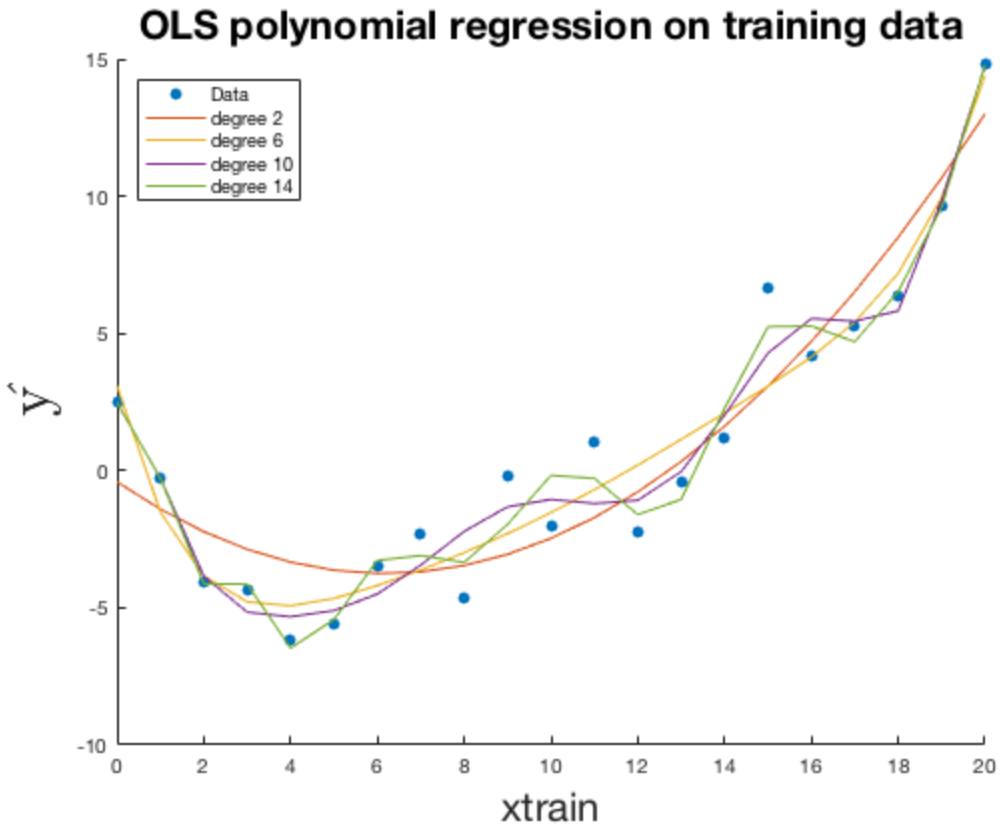
(a)

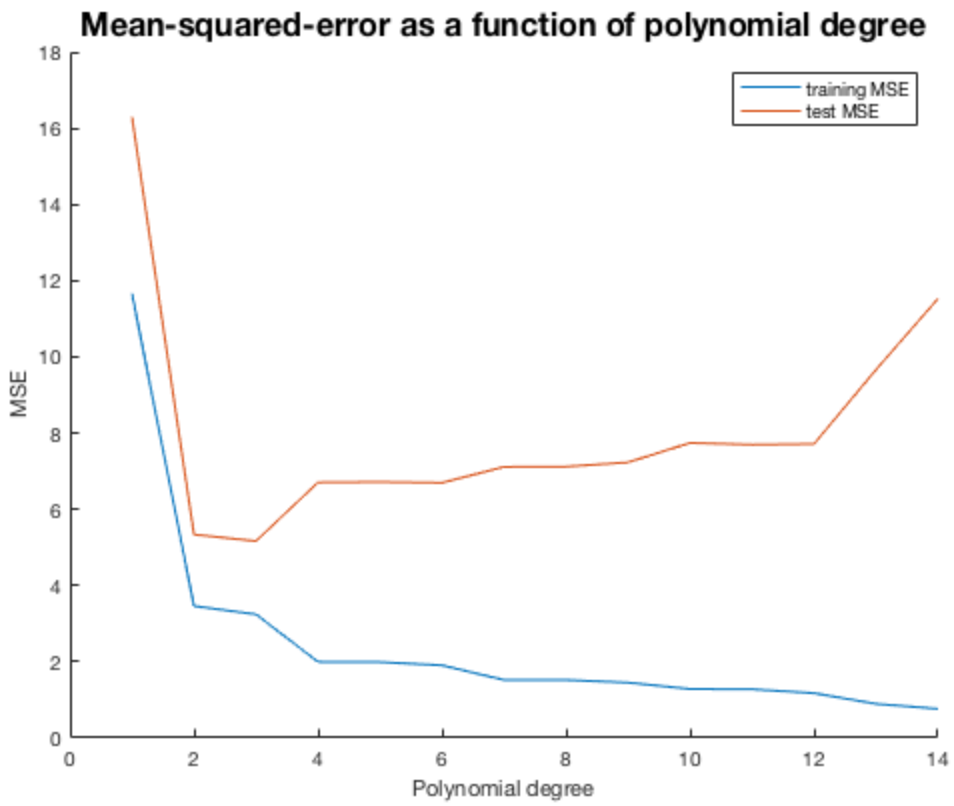
Use OLS to fit polynomials of degree d=1,...,14 to the training data. For this, use MATLAB's ridge function with the vector of ridge parameters (regularization parameters) k set to zero in order to reduce ridge regression to OLS. Also set the parameter 'scaled' to zero in order to ensure that the coefficients estimated are on the same scale as the original data. (i) On one figure, plot the training points, with polynomial curves of degree 2, 6, 10 and 14 overlaid on top. Label accordingly. (ii) On another figure, plot the MSE of the training and testing data as a function of polynomial degree (1 to 14). What do you observe? Note: Polynomials of degree d in the variable x_j are of the form: w_0 + sum_{i=1}^{d} w_i(x_j)^{i}, where x_j is the given sample

```
% OLS to fit polynomials to degree d=1,...,14
% reduce ridge to OLS by setting reg params and scale to zero
d = 14; % polynomial degree range
D = xtrain; % design matrix
W = cell(1,d); % polynomial regression coefficients
H = cell(1,d); % model/hypothesis (y_hat)
for i=1:d
    if i~=1
        D = [D D(:,1).^i];    %#ok<AGROW>
    end
    W{i} = ridge(ytrain,D,0,0);
    H{i} = D*W{i}(2:end,:)+W{i}(1,:);
end


% plot polynomial curves of degree 2,6,10,14 overlaid
figure(1);
hold on;
```

```
scatter(xtrain,ytrain,'filled');
for i = [2 6 10 14]
    plot(xtrain,H{i});
end
legend('Data','degree 2','degree 6','degree 10','degree 14', ...
    'Location','northwest');
title('OLS polynomial regression on training data','FontSize',20);
xlabel('xtrain','FontSize',20);
ylabel('y$\hat{}$','Interpreter','latex','FontSize',30);

% use weights from xtrain for xtest
D2 = xtest;
H2 = cell(1,d);
for i=1:d
    if i~=1
        D2 = [D2 D2(:,1).^i];    %#ok<AGROW>
    end
    H2{i} = D2*W{i}(2:end,:)+W{i}(1,:);
end


%plotting curves for xtest
%{
figure(2)
hold on;
scatter(xtest,ytest);
for i = [2 6 10 14]
    plot(xtest,H2{i});
end
legend('Data','degree 2','degree 6','degree 10','degree 14', ...
    'Location','northwest');
title('OLS polynomial regression on test data (with train
 weights)',...
    'FontSize',14);
xlabel('xtest','FontSize',20);
ylabel('y$\hat{}$','Interpreter','latex','FontSize',30);
%}


% Plot MSE of training and testing data
trainMSE=zeros(1,d);
testMSE=zeros(1,d);
for i=1:d
    trainMSE(i) = mse(ytrain,H{i});
    testMSE(i) = mse(ytest,H2{i});
end

figure(3);
hold on;
plot(trainMSE);
plot(testMSE);
title('Mean-squared-error as a function of polynomial degree', ...
    'FontSize',16);
xlabel('Polynomial degree');
```

```
ylabel('MSE');
legend('training MSE','test MSE');
```





Observations

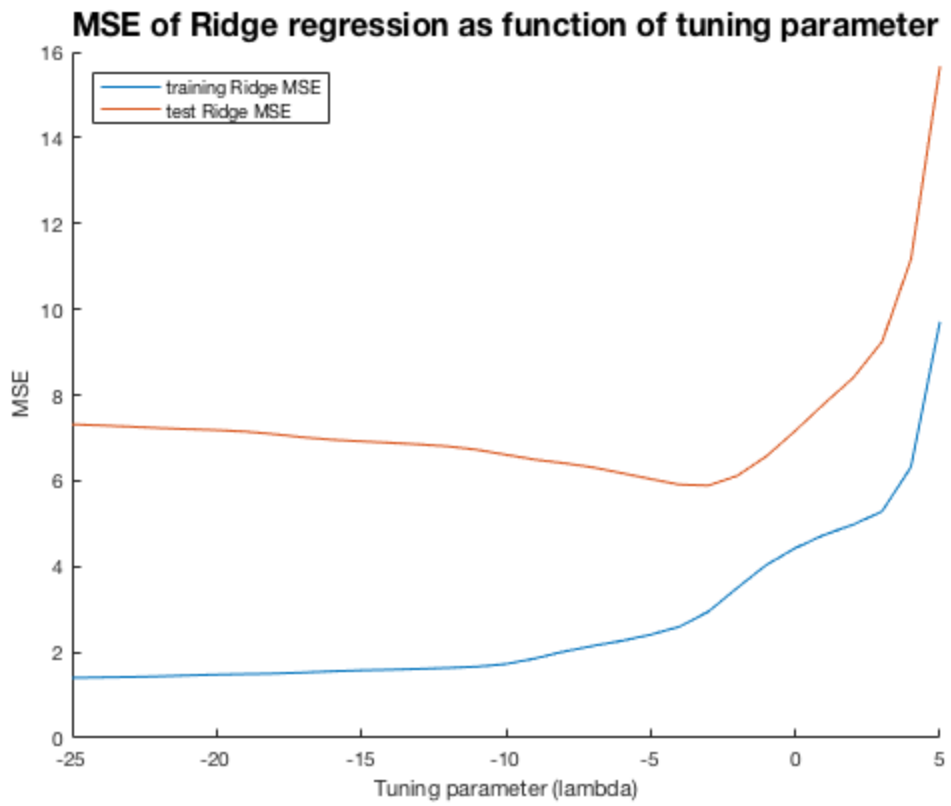
```
% Because the weights in OLS polynomial regression were chosen in
 order
% to minimize the training cost function, the MSE for the training
 data
% will be monotonically decreasing as flexibility increases and allows
 for
% a closer fit.

% On the other hand, fitting the training data more closely will cause
% overfitting on the test data at some threshold. This is obvious from
% the U-shaped graph of the test MSE, where the minima is the best fit
% polynomial degree based on the weights obtained from training, and
 to
% the right of this minima the MSE increases due to overfitting. This
 curve
% represents the bias-variance trade-off of the model.

% The best fit polynomial for the testing data with these weights is
 of
% degree 3.
```

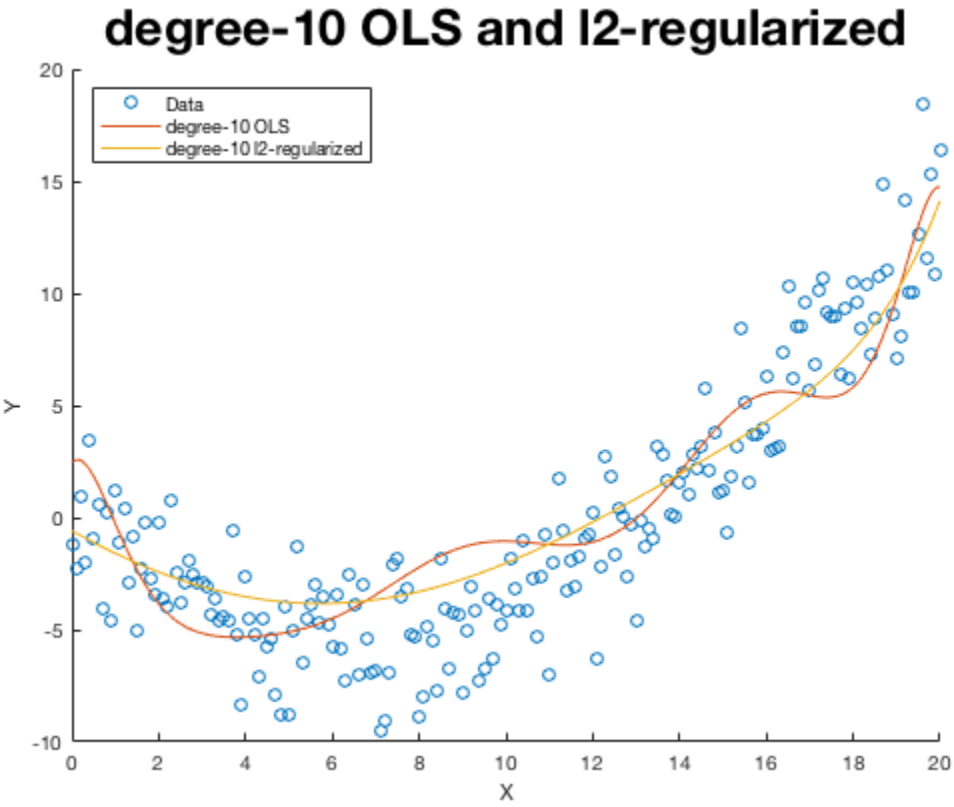
(b)

By adding l2-regularization to linear regression (making it ridge regression) the effects of over-fitting can be alleviated. Fix the polynomial degree to 10. (i) Again, on one figure, plot the mean-squared- error (MSE) of the training and testing data as a function of ln(lambda) for ln(lambda) ranging from -25 to 5 in increments of 1. (ii) On another figure, plot the testing points along with the original (no-regularization) OLS degree 10 polynomial fit and the l2-regularized degree 10 fit which has the smallest testing MSE. What do you observe?

```
% tuning parameter, coefficient of the shrinkage penalty
lambda = exp(-25:5);
degree = 10;

% training data design matrix for 10-degree polynomial
xtr = xtrain;
D3 = [xtr xtr.^2 xtr.^3 xtr.^4 xtr.^5 xtr.^6 xtr.^7 xtr.^8 xtr.^9
 xtr.^10];
W3 = cell(1,length(lambda));
H3 = cell(1,length(lambda));
trainRidgeMSE = zeros(1,length(lambda));

% test data design matrix
xte = xtest;
D4 = [xte xte.^2 xte.^3 xte.^4 xte.^5 xte.^6 xte.^7 xte.^8 xte.^9
 xte.^10];
H4 = cell(1,length(lambda));
testRidgeMSE = zeros(1,length(lambda));

% repeat over all values of lambda
for i=1:length(lambda)
    W3{i} = ridge(ytrain,D3,lambda(i),0); % polynomial coefficients
    H3{i} = D3*W3{i}(2:end,:)+W3{i}(1,:); % train estimate
    trainRidgeMSE(i) = mse(ytrain,H3{i}); % train MSE
    H4{i} = D4*W3{i}(2:end,:)+W3{i}(1,:); % test estimate
    testRidgeMSE(i) = mse(ytest,H4{i}); % test MSE
end

% Plot
figure(4);
hold on;
plot(-25:5,trainRidgeMSE);
plot(-25:5,testRidgeMSE);
title('MSE of Ridge regression as function of tuning parameter', ...
    'FontSize',16);
xlabel('Tuning parameter (lambda)');
ylabel('MSE');
legend('training Ridge MSE','test Ridge MSE', 'Location','northwest');

[~,m] = min(testRidgeMSE); % m = 23
bestLambda = lambda(m); % = e^-3

% The value of the tuning parameter that minimizes the MSE turns out
 to be
```

```
% e^-3. Penalizing complexity with l2-regularization has an obvious
 effect on
% both the training and test MSEs. The shrinkage term has the affect
 of a
% trade-off being made between the normal residual error term and this
% shrinkage term, where as the residual error term gets smaller with
% increased flexibility, the shrinkage term gets larger.


% Plot degree-10 OLS vs l2-regularized degree-10 fit with smallest MSE
figure(5);
hold on;
scatter(xtest,ytest);
plot(xtest,H2{10});
plot(xtest,H4{23});
title('degree-10 OLS and l2-regularized','FontSize',25);
xlabel('X');
ylabel('Y');
legend('Data','degree-10 OLS','degree-10 l2-regularized', ...
    'Location','northwest');

% Observation
% Comparing the degree-10 OLS model versus the degree-10 l2-
regularized
% model, the regularized version has lower MSE. It also visually looks
 like
% it fits the data better, as the non-regularized version looks like
 it was
% overfit to the training daa.
```

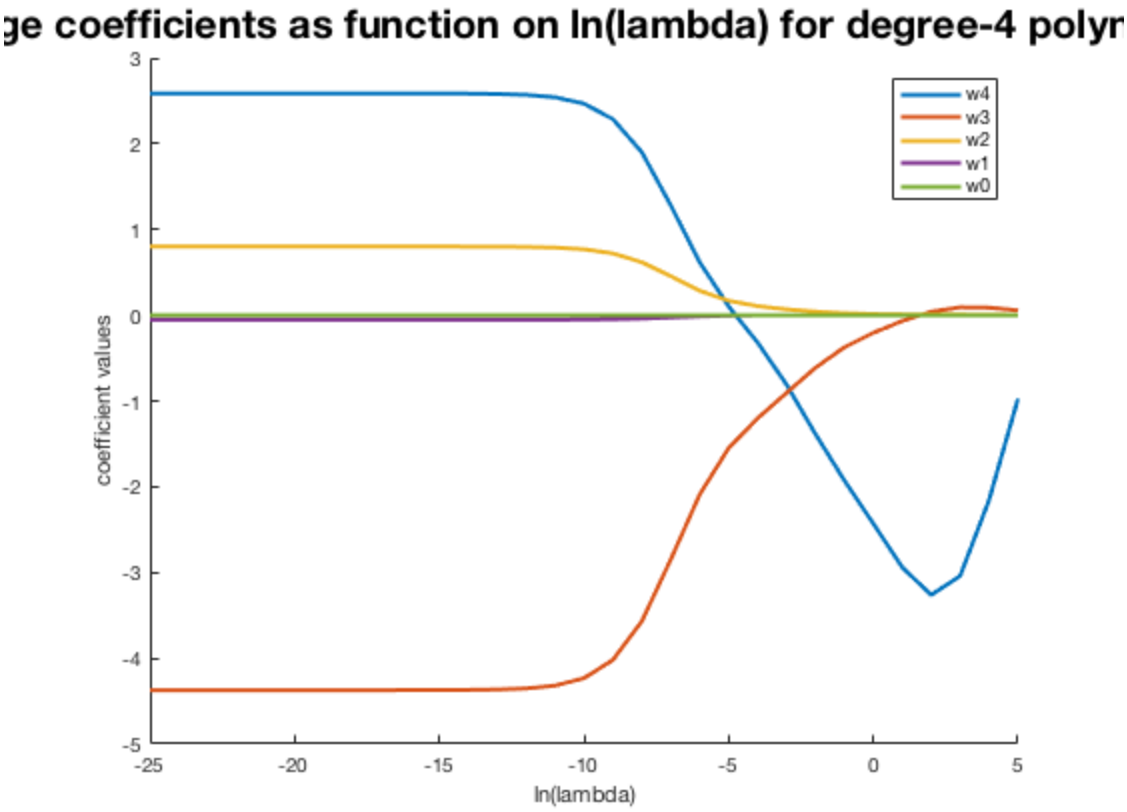

MSE of Ridge regression as function of tuning parameter
training Ridge MSE
test Ridge MSE
MSE
Tuning parameter (lambda)
16
14
12
10
8
6
4
2
0
-25
-20
-15
-10
-5
0
5


degree-10 OLS and l2-regularized
Data
degree-10 OLS
degree-10 l2-regularized
Y
X
20
15
10
5
0
-5
-10
0
2
4
6
8
10
12
14
16
18
20

(c)

On another figure, plot the ridge coefficients as a function of ln(lambda) for polynomial degree equal to 4. Plot the coefficient values as a function of ln(lambda) with ln(lambda) ranging from -25 to 5. There will be 5 curves w_0,...,w_4 (4+1, including the bias term). What do you observe?

```
D4 = [xtr xtr.^2 xtr.^3 xtr.^4];
W4 = cell(1,length(lambda));
vecs = zeros(length(lambda),5);

% repeat over all values of lambda
for i=1:length(lambda)
    W4{i} = ridge(ytrain,D4,lambda(i),0); % polynomial coefficients
    vecs(i,:) = W4{i};
end

% Plots
figure(6);
hold on;
for i=1:5
    plot(-25:5,vecs(:,i),'LineWidth',2);
end
title(['Ridge coefficients as function on ln(lambda) for
 degree-4', ...
    ' polynomial'],'FontSize',20);
legend('w4','w3','w2','w1','w0');
xlabel('ln(lambda)');
ylabel('coefficient values');

% Observations
% This plot shows the complexity penalization as a function of the
 ridge
% tuning parameter.
```



Published with MATLAB® R2017a