**Problem 1**

**(a)**

Note:

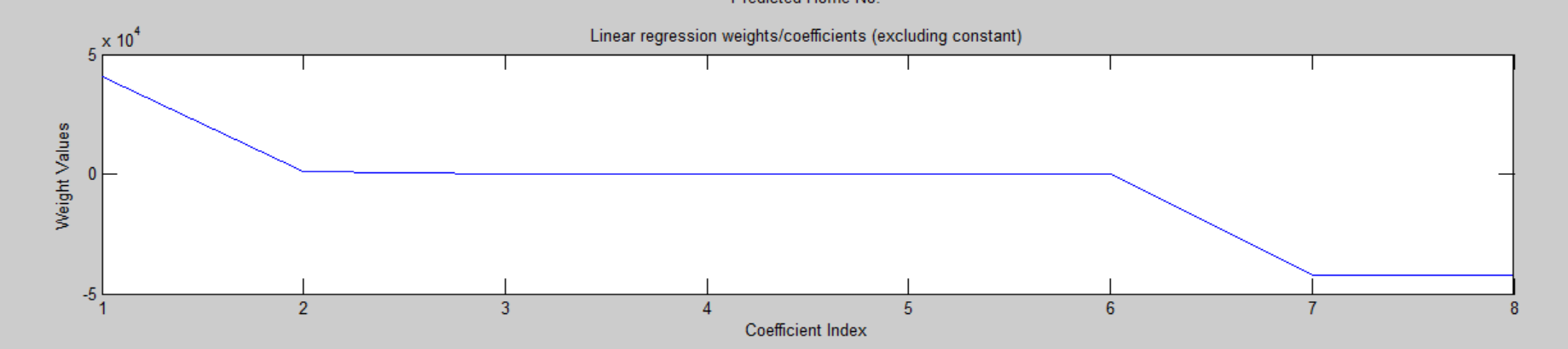
**(b)**

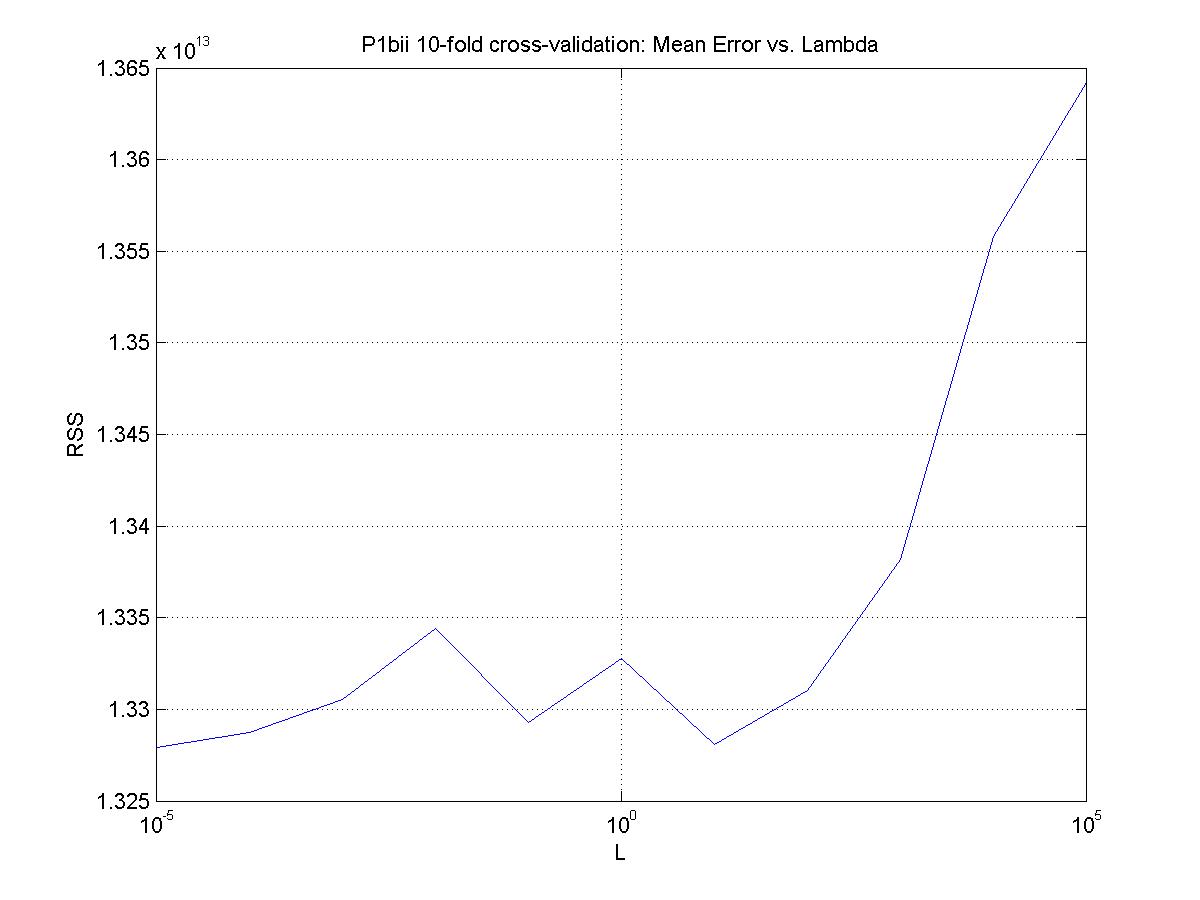
2.b.iii

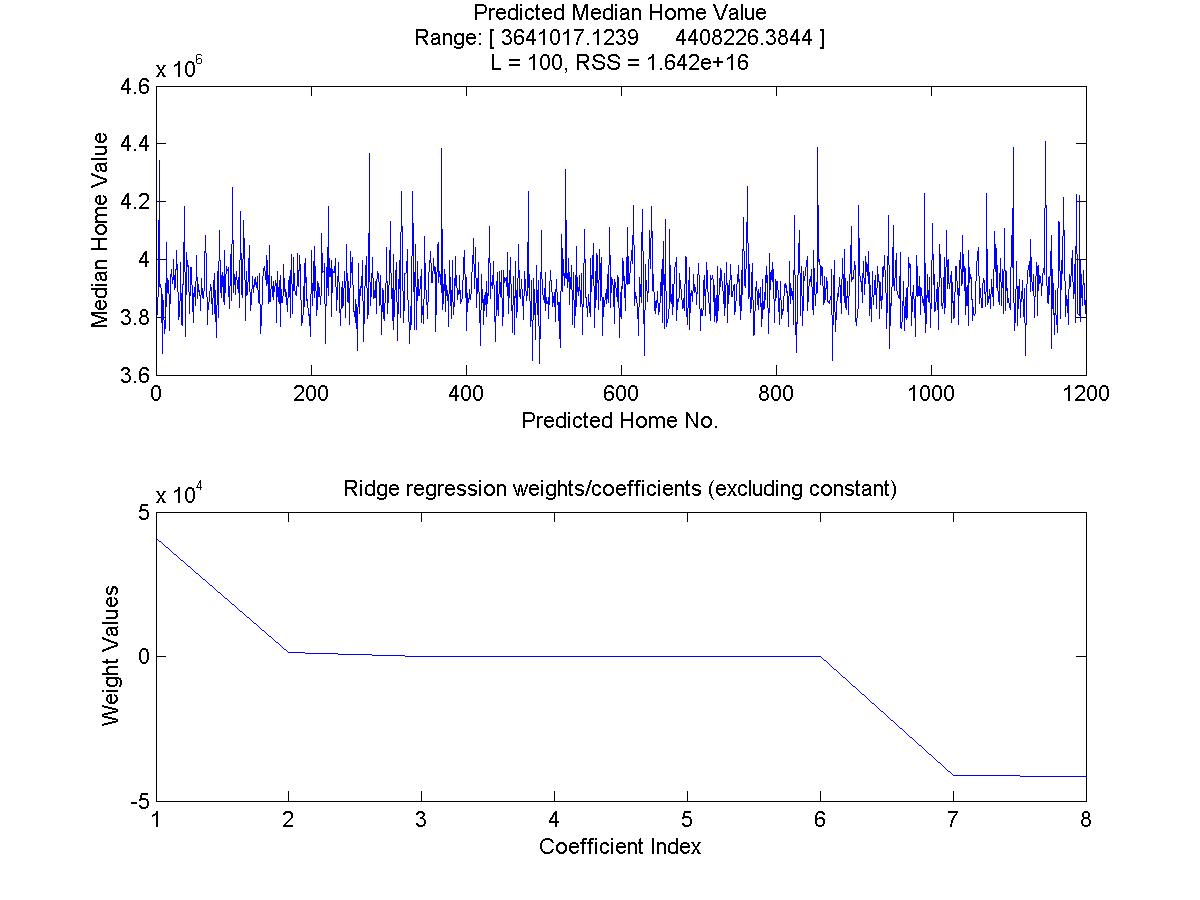
Using 10-fold cross validation, the ridge regression model was modeled. The cross-validation plot is shown below. In addition, the plotted validation home values and weights are also plotted.

Previously, the RSS for hw3 was ~6x10^12. The RSS for ridge regression is ~2x10~16. This means that ridge regression doesn’t fit to the data as much as linear regression.

The plot for the weights are shown below. The weights are very similar to ridge regression. One note, though, is that we are no longer getting unrealistic values, since our ranges are all positive (unline linear regression).



****



**Problem 2**

a) Each die has to prob. of 1/6 of rolling a six. These are independent, random events. Let

C:\Users\Jesus\AppData\Local\Temp\msohtmlclip1\02\clip_image020.png

probability of rolling two 6's on the first roll

, where probabilityn of rolling a 6

b)

c) can be interpreted many ways.

d) No, since we tested a hypothesis multiple times, we need to adjust our risk of being wrong

risk of being wrong

adjusted risk for m models tested

new risk of being wrong

e) From part a), our objective was to roll a six on both dice. The probability of getting this event was 1/36. The probability of getting this event at least once increases for every roll we make approximately by m\*P(A). To adjust a certain hypothesis test occurring by chance, the Bonferroni correction adjusts the p-value by the number of comparisons/tests made.

f) In order for a single comparison to be considered significant, the family of comparisons p-value must be adjusted with the Bonferroni correction.

family significance level

individual significance level for each comparison

This means that the p-value for each gene in order for the gene to be considered significant. The gene in question has . This means that this gene is not significant.

The Bonferroni correction gives a value greater than 1 since our initial significance level is 0.05, which means we are at risk of 1/20 experiments to be significant by chance. It only takes 20 tests with the Bonferroni correction to be 100% certain that at least one of our tests reached 0.05 significance by chance. In this case, since m = 50,000, we are certain that at least 50,000/20 = 2,500 experiments will be significant at p = 0.05 level.

Code for P1

%Problem 1: Centering and Ridge Regression

%get the current directory

currDir = cd;

I = strfind(currDir, '\');

I = I(end);

parentDir = currDir(1:I-1);

plotsDir = [parentDir '\plots'];

dataDir = [parentDir '\data'];

%assign the plots directory

if ~isdir(plotsDir)

mkdir(plotsDir);

end

%Load the data (Xtrain, Xvalidate, Ytrain, Yvalidate)

load([dataDir '\housing\_data']);

%% 1.1 Ridge Regression

%assign the data to our equation variables and Center our data

n = size(Xtrain,1);

%Xtrain = [ones(n,1) Xtrain]; %add a constant term

X = center\_data(Xtrain);

y = Ytrain;

L = 1;

%compute the weights

w0 = 1/n\*sum(y);

w = inv(X'\*X + L)\*X'\*y;

%% 1.2 Cross-validation and Residual sum-of-squares (RSS)

%for each value of L,

k = 10;

L = [1e-5 1e-4 1e-3 1e-2 1e-1 1e0 1e1 1e2 1e3 1e4 1e5];

mean\_errs = zeros(1, numel(L));

for ii = 1:1:numel(L)

%

disp('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

disp(['Training for L = ' num2str(L(ii))]);

%Randomly split the data into k parts

[split\_data\_cell, split\_labels\_cell, rand\_inds\_cell] = rand\_split\_data(Xtrain, Ytrain, 1, k);

err\_vec = zeros(1, k);

for jj = 1:1:k

%grab the jth dataset and labels to separate for validation

XVal = center\_data(split\_data\_cell{jj});

yV = split\_labels\_cell{jj};

%grab the rest for use as the training data

ind\_vec = 1:1:k;

ind\_vec(jj) = [];

X = split\_data\_cell(ind\_vec); %get rest of data (class cell)

X = center\_data(cat(1,X{:})); %center and concatenate along 1st dimension

y = split\_labels\_cell(ind\_vec);

y = cat(1,y{:}); %concatenate along first dimension

%compute the weights

w0 = 1/n\*sum(y);

w = inv(X'\*X + L(ii))\*X'\*y;

%compute the predicted value

yp = XVal\*w + w0;

%compute the RSS

RSS = (yV - yp)'\*(yV-yp);

%save the error of the jth iteration

err\_vec(jj) = RSS;

end

%store the mean of the errors

mean\_errs(ii) = mean(err\_vec);

end

%plot the mean error as a function of L

%plot the accuracy vs. number of samples selected

h0 = figure('visible', 'on', 'units', 'normalized','outerposition',[0 0 1 1]);

semilogx(L, mean\_errs), title('P1bii 10-fold cross-validation: Mean Error vs. Lambda'),

xlabel('L'), ylabel('RSS'), grid('on');

saveas(h0, strcat(plotsDir, '\P1bii - 10-fold Err vs Lambda.jpg'));

%}

%%%%% Get the RSS for L = 1.0x10^2

%get our

X = center\_data(Xtrain);

y = Ytrain;

yV = Yvalidate;

XVal = Xvalidate;

L = 10^2;

%compute the weights

w0 = 1/n\*sum(y);

w = inv(X'\*X + L)\*X'\*y;

%compute the predicted value

yp = XVal\*w + w0;

%compute the RSS

RSS = (yV - yp)'\*(yV-yp);

%plot the predicted values and get the range

ypMax = max(yp);

ypMin = min(yp);

ypRange = [ypMin ypMax];

disp(ypRange);

h = figure('visible', 'on','units', 'normalized','outerposition',[0 0 1 1]);

subplot(2,1,1), plot(yp);

title({'Predicted Median Home Value'; ['Range: [ ' num2str(ypRange) ' ]']; ...

['L = ' num2str(L) ', RSS = ' num2str(RSS, 4)]});

xlabel('Predicted Home No.');

ylabel('Median Home Value');

%% 1.3 Plot w as function of its index

subplot(2,1,2), plot(w);

title('Ridge regression weights/coefficients (excluding constant)')

xlabel('Coefficient Index');

ylabel('Weight Values');

saveas(h, [plotsDir '\1biii Ridge Regression Coefficients.jpg']);

%}

%%%%%%%%%%%%% FUNCTIONS %%%%%%%%%%%%%%%%

function Xcen = center\_data(X)

%this function centers the data matrix X, assuming each row in the matrix

%is a sample and each column represents a feature.

%Get the expected value of each column of X and repeat it for every row

EX = repmat(mean(X), size(X,1),1);

Xcen = X -EX;

end

function [split\_data\_cell, split\_labels\_cell, rand\_inds\_cell] = rand\_split\_data(data, labels, dim, k)

%This function randomly splits data into k equal parts.

%create an index vector

I = (1:1:numel(labels))';

%Separate the index vector by labels

%Get the number of unique labels

uniq\_labels = unique(labels);

rand\_inds\_cell = cell(k, numel(uniq\_labels));

for ii = 1:1:numel(uniq\_labels)

%Find all instances of the current lable, returns logical

I\_temp = labels == uniq\_labels(ii);

%get the corresponding indeces from I

I\_temp = I(I\_temp);

%randomly permute the indeces

I\_temp = I\_temp(randperm(numel(I\_temp)));

%Split the vector into k-part cell column

a = floor(numel(I\_temp)/k); %compute the number of elements for each split, minus the kth split

b = mod(numel(I\_temp),k); %compute the remaining number of elements for the kth split

if b == 0

c = repmat(a, 1, k);

else

c = repmat(a, 1, k);%create cell for splitting rand\_ind

c(1:b) = c(1:b) + 1;

end

I\_temp = mat2cell(I\_temp, c, 1);

%store in our cell along the first row

rand\_inds\_cell(:, ii) = I\_temp;

end

%Transpose rand\_inds\_cell to stack each k group along the columns of the

%cell

rand\_inds\_cell = rand\_inds\_cell';

%Concatenate the columns to create the final indeces

for ii = 1:1:k

I\_temp = cat(1, rand\_inds\_cell{:, ii});

%store it in the first row

rand\_inds\_cell{1,ii} = I\_temp;

%empty rows 2 through end

rand\_inds\_cell(2:end,ii) = cell(size(rand\_inds\_cell, 1) - 1,1);

end

%remove the empty cells of rand\_inds\_cell

rand\_inds\_cell = rand\_inds\_cell(1,:);

split\_data\_cell = cell(size(rand\_inds\_cell));

split\_labels\_cell = cell(size(rand\_inds\_cell));

for ii = 1:1:numel(rand\_inds\_cell)

%get the current random indeces

curr\_rands = rand\_inds\_cell{ii};

%split the labels

split\_labels\_cell{ii} = labels(curr\_rands);

%split the data

if dim == 1

split\_data\_cell{ii} = data(curr\_rands, :);

elseif dim == 2

split\_data\_cell{ii} = data(:, curr\_rands);

elseif dim ==3

split\_data\_cell{ii} = data(:,:,curr\_rands);

else

end

end

end