Problem 1

(a)

$$\frac{\partial J}{\partial \mathrm{w}_0} = \left[\frac{\partial}{\partial \mathrm{w}_0}(\boldsymbol{y} - \mathbf{Xw} - \mathrm{w}_0 \mathbf{1})^\top\right](\mathbf{y} - \mathbf{Xw} - \mathrm{w}_0 \mathbf{1}) + (\boldsymbol{y} - \mathbf{Xw} - \mathrm{w}_0 \mathbf{1})^\top\left[\frac{\partial}{\partial \mathrm{w}_0}(\mathbf{y} - \mathbf{Xw} - \mathrm{w}_0 \mathbf{1})\right]$$

$$\frac{\partial J}{\partial \mathrm{w}_0} = -\mathbf{1}^\top(\mathbf{y} - \mathbf{Xw} - \mathrm{w}_0 \mathbf{1}) + (\boldsymbol{y} - \mathbf{Xw} - \mathrm{w}_0 \mathbf{1})^\top(-\mathbf{1})$$

$$\frac{\partial J}{\partial \mathrm{w}_0} = -\mathbf{1}^\top\mathbf{y} + \mathbf{1}^\top\mathbf{Xw} + \mathrm{w}_0\mathbf{1}^\top\mathbf{1} - \mathbf{y}^\top\mathbf{1} + \mathbf{w}^\top\mathbf{X}^\top\mathbf{1} + \mathrm{w}_0\mathbf{1}^\top\mathbf{1}$$

$$0 = -2\sum_{\mathrm{i}}^{\mathrm{n}} \mathrm{y_i} + 2(\mathbf{x_1 w} + \cdots + \mathbf{x_n w}) + 2\mathrm{n}\mathrm{w}_0$$

$$w_0 = \frac{1}{n}\sum_{i}^{n} y_i - \frac{1}{n}(\mathbf{x_1 w} + \cdots + \mathbf{x_n w})$$

$$w_0 = \frac{1}{n}\sum_{i}^{n} y_i - \frac{1}{n}(\mathbf{x_1} + \cdots + \mathbf{x_n})\mathbf{w}$$

$$w_0 = \frac{1}{n}\sum_{i}^{n} y_i - \frac{1}{n}\mathbf{0w}$$

$$w_0 = \frac{1}{n}\sum_{i}^{n} y_i$$

$$\frac{\partial J}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}[(\boldsymbol{y} - \mathbf{Xw} - \mathrm{w}_0 \mathbf{1})^\top(\mathbf{y} - \mathbf{Xw} - \mathrm{w}_0 \mathbf{1}) + \lambda\mathbf{w}^\top\mathbf{w}]$$

$$\frac{\partial J}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}[(\boldsymbol{y}^\top - \mathbf{w}^\top\mathbf{X}^\top - \mathrm{w}_0 \mathbf{1}^\top)(\mathbf{y} - \mathbf{Xw} - \mathrm{w}_0 \mathbf{1}) + \lambda\mathbf{w}^\top\mathbf{w}]$$

$$\frac{\partial J}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}\left[[(\mathrm{y}_1 - \mathbf{w}^\top\mathbf{x}_1^\top - \mathrm{w}_0)\dots(\mathrm{y}_\mathrm{n} - \mathbf{w}^\top\mathbf{x}_\mathbf{n}^\top - \mathrm{w}_0)]\begin{bmatrix}(\mathrm{y}_1 - \mathbf{x_1 w} - \mathrm{w}_0)\\ \vdots \\ (\mathrm{y}_\mathrm{n} - \mathbf{x_n w} - \mathrm{w}_0)\end{bmatrix} + \lambda\mathbf{w}^\top\mathbf{w}\right]$$

Note: $\mathbf{w}^\top\mathbf{x}_\mathbf{i}^\top = \mathbf{x_i w}$

$$\frac{\partial J}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}\left[\left[(\mathrm{y}_1 - \mathbf{x_1 w} - \mathrm{w}_0)^2 + \dots + (\mathrm{y}_\mathrm{n} - \mathbf{w}^\top\mathbf{x}_\mathbf{n}^\top - \mathrm{w}_0)^2\right] + \lambda\mathbf{w}^\top\mathbf{w}\right]$$

$$0 = \mathbf{x_1}^\top y_1 - \mathbf{x_1}^\top \mathbf{x}_1 \mathbf{w} - \mathbf{x_1}^\top w_0 + \ \dots\ + \mathbf{x_n}^\top y_n - \mathbf{x_n}^\top \mathbf{x}_n \mathbf{w} - \mathbf{x_n}^\top w_0 - \lambda \mathbf{w}$$

$$0 = (\mathbf{x_1}^\top y_1 + \cdots + \mathbf{x_n}^\top y_n) - (\mathbf{x_1}^\top \mathbf{x}_1 + \ldots + \mathbf{x_n}^\top \mathbf{x}_n)\mathbf{w} - w_0(\mathbf{x_1} + \ \dots + \mathbf{x_n}) - \lambda \mathbf{w}$$

$$0 = \mathbf{X}^\top \mathbf{y} - \mathbf{X}^\top \mathbf{X}\mathbf{w} - w_0 \mathbf{0} - \lambda \mathbf{w}$$

$$(\mathbf{X}^\top \mathbf{X} + \boldsymbol{\lambda} \mathbf{I})\mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

$$\hat{\mathbf{w}} = \left(\mathbf{X}^\mathbf{T} \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^\mathbf{T} \mathbf{y}$$

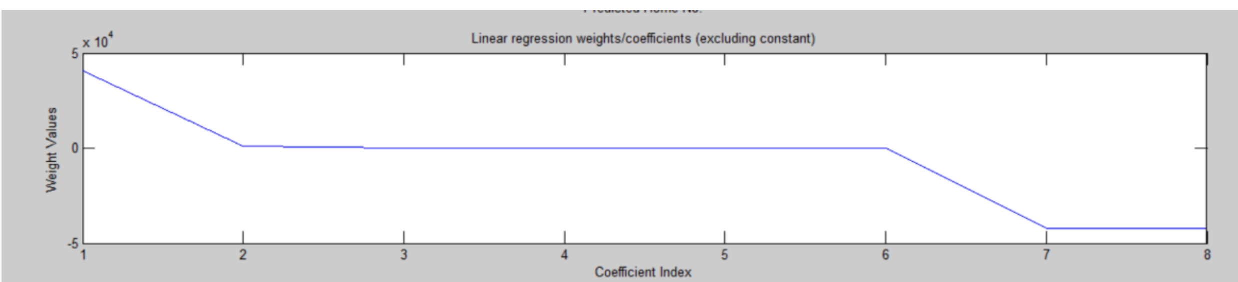
(b)

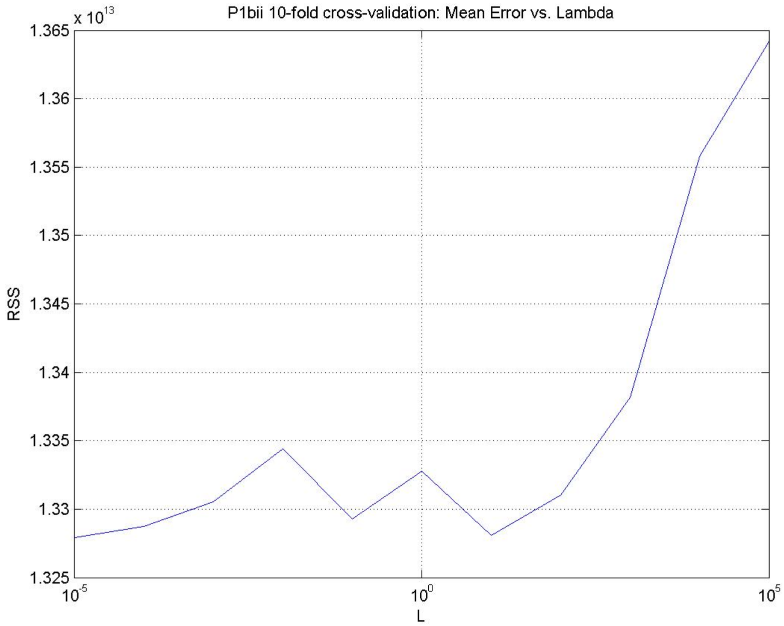
2.b.iii

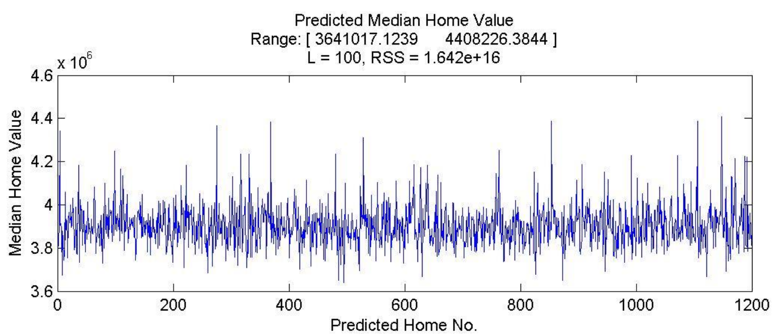
Using 10-fold cross validation, the ridge regression model was modeled. The cross-validation plot is shown below. In addition, the plotted validation home values and weights are also plotted.

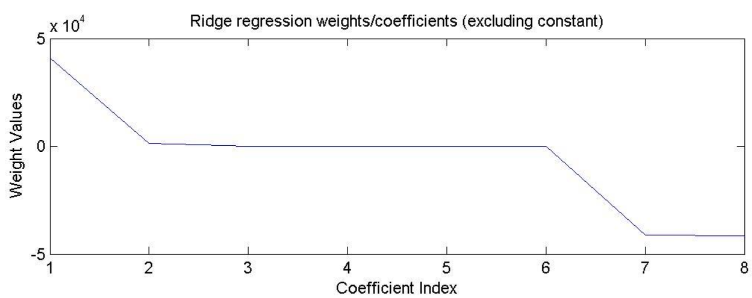
Previously, the RSS for hw3 was ~6x10^12. The RSS for ridge regression is ~2x10~16. This means that ridge regression doesn't fit to the data as much as linear regression.

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).




P1bii 10-fold cross-validation: Mean Error vs. Lambda
x 10^{13}
RSS
L


Predicted Median Home Value
Range: [3641017.1239 4408226.3844]
L = 100, RSS = 1.642e+16
x 10^6
Median Home Value
Predicted Home No.


Ridge regression weights/coefficients (excluding constant)
x 10^4
Weight Values
Coefficient Index

Problem 2

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

$P(A) \equiv$ probability of rolling two 6's on the first roll

$= P(D_6)P(D_6)$, where $P(D_6) \equiv$ probabilityn of rolling a 6

$$= \left(\frac{1}{6}\right)\left(\frac{1}{6}\right)$$

$$P(A) = \frac{1}{36} = 0.278$$

b) $P(A \cup B \cup C) = P(A) + P(B) + P(C) - P(A \cap B) - P(A \cap C) - P(B \cap C) - P(A \cap B \cap C)$

$$= \frac{1}{36} + \frac{1}{36} + \frac{1}{36} - \frac{1}{36^2} - \frac{1}{36^2} - \frac{1}{36^2} - \frac{1}{36^3}$$

$$P(A \cup B \cup C) = 0.081$$

c) can be interpreted many ways.

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

$p_0 = 0.05 \equiv$ risk of being wrong

$p'_0 = \mathrm{m}(0.05) \equiv$ adjusted risk for m models tested

$$p'_0 = 6(0.05),\ \mathrm{m} = 6 = 5 + 1$$

$p'_0 = 0.30 \equiv$ 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.

$p_f = 0.05 \equiv$ family significance level

$p_i = \frac{0.05}{\mathrm{m}} \equiv$ individual significance level for each comparison

$$p_i = \frac{0.05}{50{,}000} = 1.0\text{x}10^{-6}$$

This means that the p-value for each gene $p_g \leq \ p_i$ in order for the gene to be considered significant. The gene in question has $p_g = 1.0\text{x}10^{-4} > p_i$. 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
```