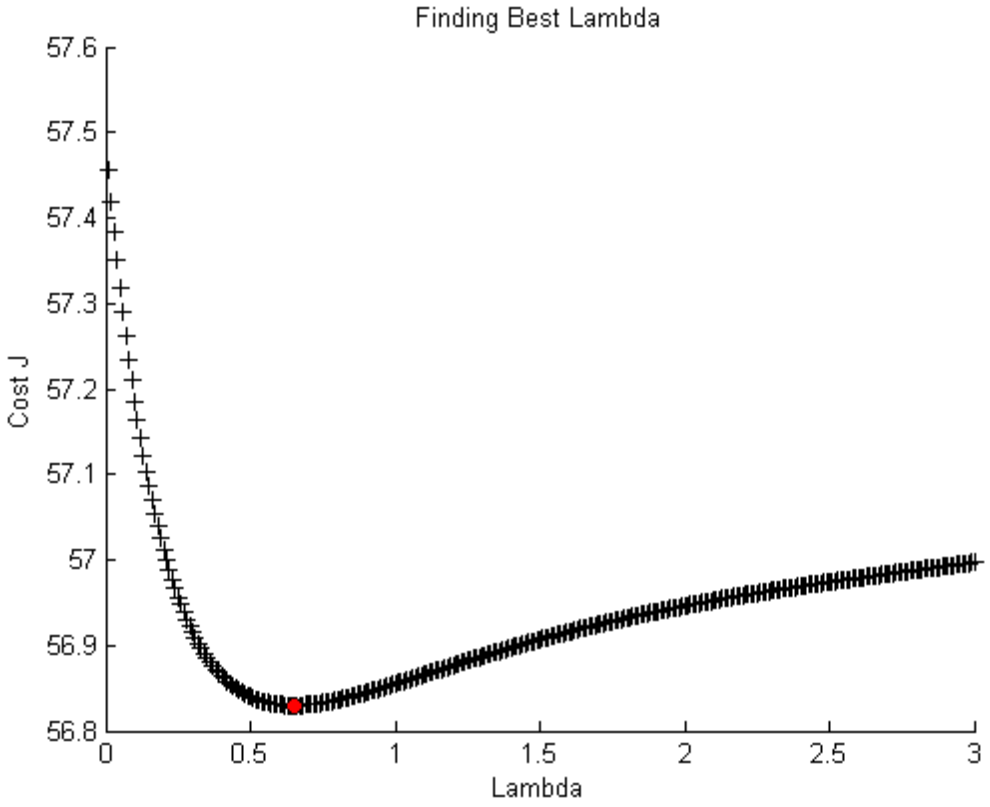
```
%%MADAN MOHAN KURUBA
%%NISHAL PATTAN
%%SRI CHAKRA
%%RAJ KIRAN UPPARI

load('hm2data2.mat')
X = data(:,1:3);
fprintf('\nSize of X');
size(X)
Y = data(:,4);
fprintf('\nSize of Y');
size(Y)

%mean and standard deviation of data X
muX = mean(X);
stdX = std(X);
repstd = repmat(stdX, size(X,1), 1);
repmu = repmat(muX, size(X,1), 1);

%Normalizing X
X_norm = (X-repmu)./(repstd);
X_norm = [ones(97,1) X_norm];
size(X_norm)
%mean(X_norm)
%std(X_norm)


%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%L2-regularization%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%size(rand_indices);
X_train = X_norm(1:48,:);
Y_train = Y(1:48,:);
size(X_train)
X_test = X_norm(49:97,:);
Y_test = Y(49:97,:);
size(X_test);



%cost_Reg2(X_train, Y_train, theta, alpha, num_iters,lambda);

%%%%%Best Lambda
fprintf('Finding the best lambda ...\n');
%%Using Normal Equation form
%%Find the best lambda
lambda = best_lambda2(X_train, Y_train, X_test, Y_test)




%%%3-Fold validation
% Randomly select half data points to as training_data other as test_data
X_norm = X_norm(:,1:3);
rand_indices = randperm(size(X_norm))';
%size(rand_indices);
X1 = X_norm(rand_indices(1:32),:);
Y1 = Y(rand_indices(1:32),:);
size(X1);
X2 = X_norm(rand_indices(33:64),:);
Y2 = Y(rand_indices(33:64),:);
size(X2);
```

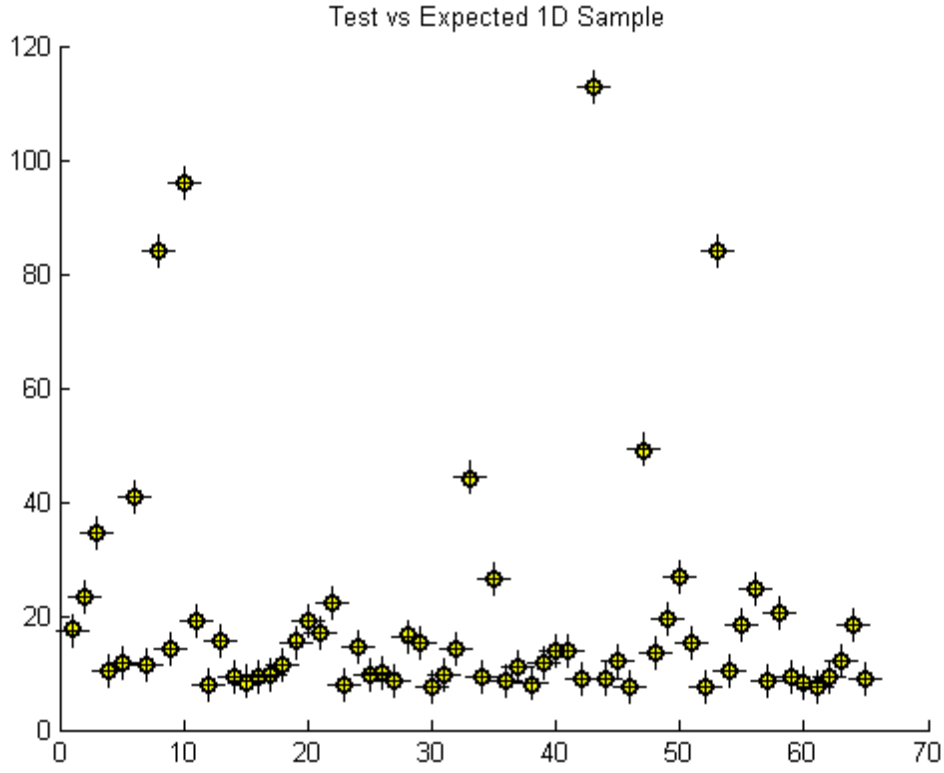
```
X3 = X_norm(rand_indices(65:97),:);
Y3 = Y(rand_indices(65:97),:);
size(X3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% Degree d = 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
error_1d=0;
%Training on X1
theta = zeros(3, 1);
[J, theta] = gdMulti(X1, Y1, theta, 0.5, 600);
%Testing on X2
figure;hold on;
plot([Y2;Y3],'ko','MarkerFaceColor','y','LineWidth',2,'MarkerSize',7);
plot([X2*theta;X3*theta],'k+','MarkerFaceColor','r','MarkerSize',12);
title(' Test vs Expected 1D Sample')
error_1d= error_1d + computeError(theta,X2,Y2);
%Testing on X3
error_1d= error_1d + computeError(theta,X3,Y3);



%Training on X2
theta = zeros(3, 1);
[J, theta] = gdMulti(X2, Y2, theta, 0.5, 600);
figure; hold on;
plot(1:100,J(1:100));
xlabel('Iterations');ylabel('Cost Value');
title('Gradient Descent');

error_1d =error_1d + computeError(theta,X1,Y1);
error_1d =error_1d + computeError(theta,X3,Y3);

%Training on X3
theta = zeros(3, 1);
[J, theta] = gdMulti(X2, Y2, theta, 0.5, 600);
error_1d =error_1d + computeError(theta,X1,Y1);
error_1d =error_1d + computeError(theta,X3,Y3);
fprintf('Avg Error in 1D : ');
error_1d = error_1d/194;
fprintf('Average Error in 1-D Model : %f\n',error_1d);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% degree d = 2 ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%2nd Dimensions
X1 = [X1(:,1) X1(:,2) X1(:,3) X1(:,2).*X1(:,3) X1(:,2).^2 X1(:,3).^2];
X2 = [X2(:,1) X2(:,2) X2(:,3) X2(:,2).*X2(:,3) X2(:,2).^2 X2(:,3).^2];
X3 = [X3(:,1) X3(:,2) X3(:,3) X3(:,2).*X3(:,3) X3(:,2).^2 X3(:,3).^2];

error_2d = 0;
%Training on X1
theta = zeros(6, 1);
[J, theta] = gdMulti(X1, Y1, theta, 0.05, 600);
theta;
%Testing on X2
error1 = Y2-X2*theta;
figure;hold on;
plot([Y2;Y3],'kO','MarkerFaceColor','g','LineWidth',2,'MarkerSize',7);
plot([X2*theta; X3*theta],'k+','MarkerFaceColor','b','MarkerSize',12);
title('Test vs Expected 2D Sample')

error_2d= error_2d + computeError(theta,X2,Y2);
%Testing on X3
error_2d = error_2d + computeError(theta,X3,Y3);
```

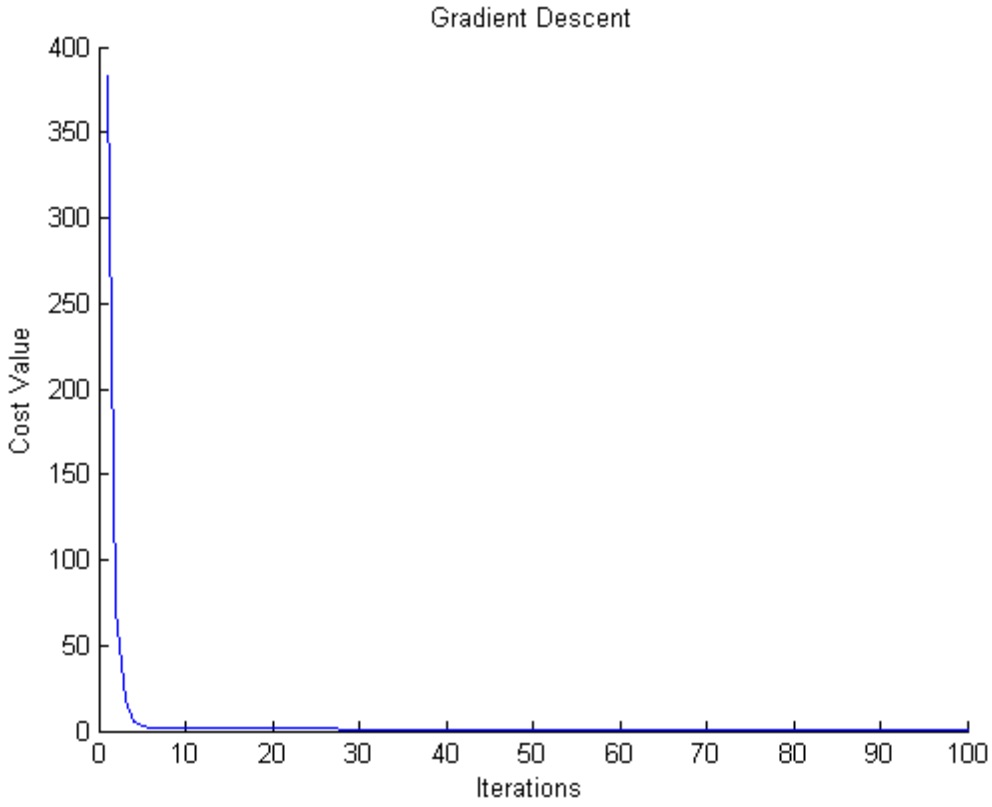
```
%Training on X2
theta = zeros(6, 1);
[J, theta] = gdMulti(X2, Y2, theta, 0.05, 600);
error_2d =error_2d + computeError(theta,X1,Y1);
error_2d =error_2d + computeError(theta,X3,Y3);

%Training on X3
theta = zeros(6, 1);
[J, theta] = gdMulti(X2, Y2, theta, 0.05, 600);
error_2d =error_2d + computeError(theta,X1,Y1);
error_2d =error_2d + computeError(theta,X3,Y3);
fprintf('Avg Error in 2D : ');
error_2d = error_2d/194;

fprintf('Average Error in 2-D Model : %f\n',error_2d);


if error_1d*100 < error_2d*100
   fprintf('\nAverage Error in 1 Dimentional model is less than 2D\n So we will pr
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Modeling and Generalization Errors %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
modelingError=zeros(100,1);
generalizationError=zeros(100,1);
for i=1:100
    rand_indices = randperm(97)';
    %size(rand_indices);
    X1 = X_norm(rand_indices(1:48),:);
    Y1 = Y(rand_indices(1:48),:);
    size(X1);
    X2 = X_norm(rand_indices(49:97),:);
    Y2 = Y(rand_indices(49:97),:);
    size(X2);
    %%%Train the Model
    theta = zeros(3,1);
    [J, theta] = gdMulti(X1, Y1, theta, 0.5, 600);
    modelingError(i) = computeError(theta, X1, Y1)/(length(X1));
    generalizationError(i) = computeError(theta, X2, Y2)/(length(X2));
end

figure;hold on;
plot(1:100, modelingError,1:100 ,generalizationError,'LineWidth',2,'MarkerSize',8)
xlabel('Iterations');ylabel('Error Value');


%plot(1:100, generalizationError,'--','LineWidth',2,'MarkerSize',8)
legend('Modeling Error','Generalization Error');

fprintf('\n\n*************************\n\t Modeling Errors\n*****************
fprintf('\n Minimum Modeling Error = %f\n',min(modelingError));
fprintf('\n Maximum Modeling Error = %f\n',max(modelingError));
fprintf('\n Average Modeling Error = %f\n',mean(modelingError));

fprintf('\n\n*************************\n\t Generalization Errors\n************
fprintf('\n Minimum Generalization Error = %f\n',min(generalizationError));
fprintf('\n Maximum Generalization Error = %f\n',max(generalizationError));
fprintf('\n Average Generalization Error = %f\n',mean(generalizationError));

hold off;

%%%Regression Line Plots
figure;
```

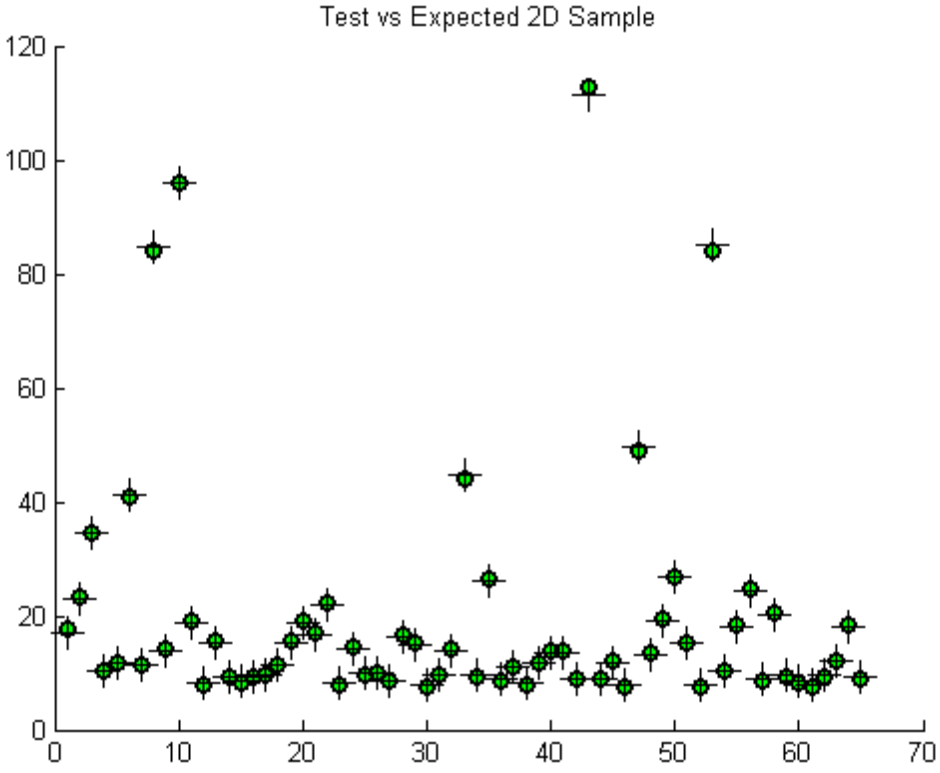
```
subplot(1,2,1);
plot(X2(:,2),Y2,'rx','MarkerSize',10);
hold on;
plot(X2(:,2), X2*theta, 'b-')
xlabel('Attribute W1');
ylabel('Actual and Predicted Values');

subplot(1,2,2);
plot(X2(:,3),Y2,'gx','MarkerSize',10);
hold on;
plot(X2(:,3), X2*theta, 'b--')
xlabel('Attribute W2');
ylabel('Actual and Predicted Values');
legend('Actual Output','Linear regression');


% Fill out J_vals
% Grid over which we will calculate J
W1_vals = linspace(-20, 80, 100);
W2_vals = linspace(-20, 50, 100);
[J, theta] = gdMulti(X1, Y1, theta, 0.5, 600);
% initialize J_vals to a matrix of 0's
J_vals = zeros(length(W1_vals), length(W2_vals));

for i = 1:length(W1_vals)
    for j = 1:length(W2_vals)
          t = [theta(1,1);W1_vals(i); W2_vals(j)];
          J_vals(i,j) = computeCost(X1(:,1:3), Y1, t,0);
    end
end

J_vals = J_vals';

% Surface plot
figure;
subplot(1,2,1)
surf(W1_vals, W2_vals, J_vals)
xlabel('w_1'); ylabel('w_2');
title('Surface')

% Contour plot
subplot(1,2,2)
contour(W1_vals, W2_vals, J_vals, logspace(-2, 10, 20))
xlabel('W_1'); ylabel('W_2');
hold on;
plot(theta(2), theta(3), 'rx', 'MarkerSize', 10, 'LineWidth', 2);
title('Contour');



Size of X
ans =

    97     3


Size of Y
ans =

    97     1


ans =

    97     4
```

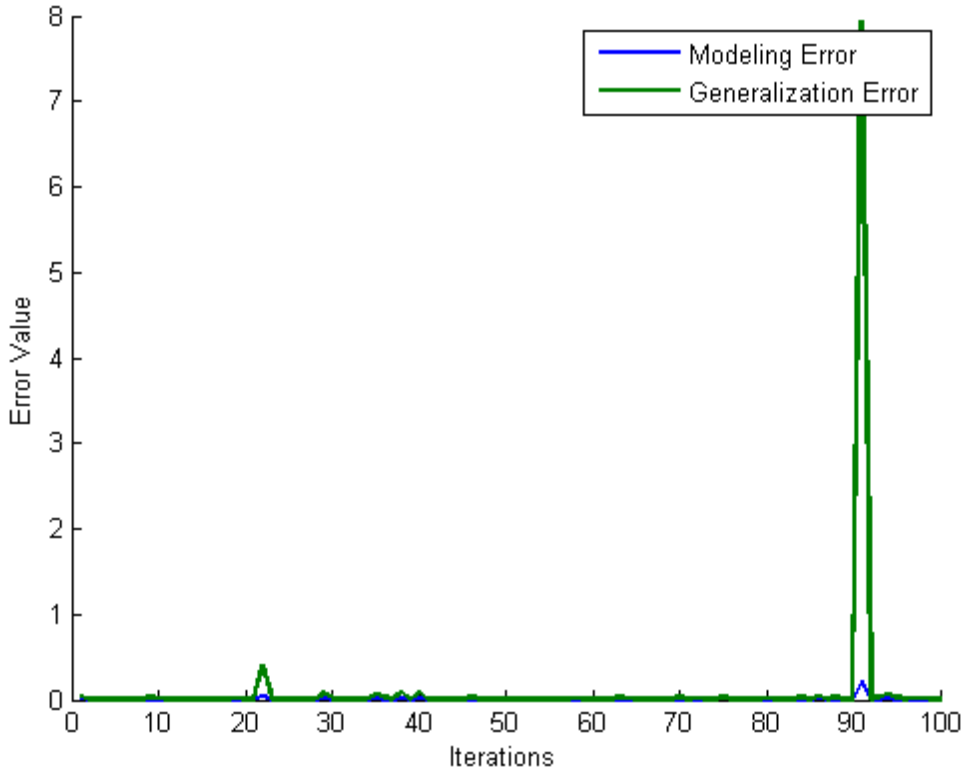
```
ans =

    48     4

Finding the best lambda ...
Minimum J , theta, lambda

J_min =

   56.8299


theta_at_minJ =

   10.3696
    3.1394
   18.9239
    1.3924


lambda =

    0.6500


lambda =

    0.6500

Warning: Input arguments must be scalar.
Avg Error in 1D : Average Error in 1-D Model : 0.016594
Avg Error in 2D : Average Error in 2-D Model : 0.062250

Average Error in 1 Dimentional model is less than 2D
 So we will proceed with is 1D model


****************************
         Modeling Errors
****************************
 Minimum Modeling Error = 0.001957

 Maximum Modeling Error = 0.216535

 Average Modeling Error = 0.008058


****************************
         Generalization Errors
****************************
 Minimum Generalization Error = 0.001029

 Maximum Generalization Error = 7.943282

 Average Generalization Error = 0.095516
```

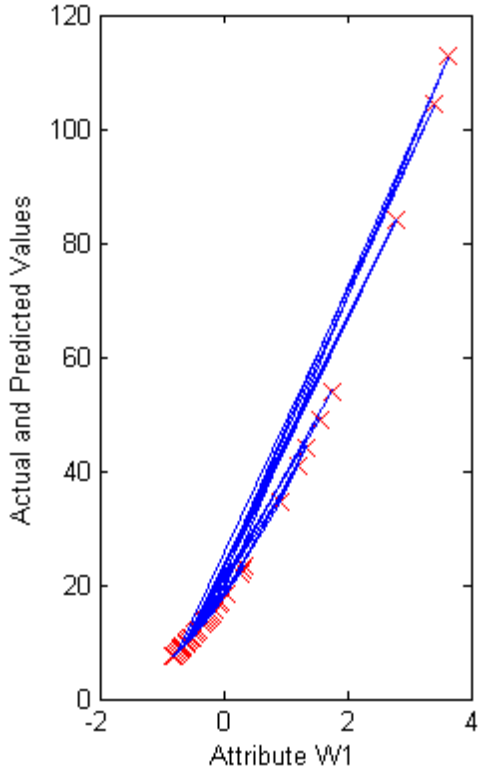

Finding Best Lambda
Cost J
Lambda
57.6
57.5
57.4
57.3
57.2
57.1
57
56.9
56.8
0
0.5
1
1.5
2
2.5
3

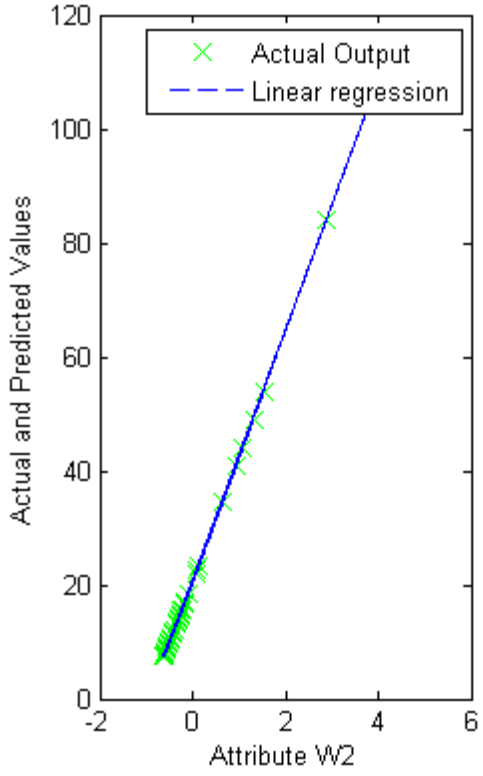

Test vs Expected 1D Sample
120
100
80
60
40
20
0
0
10
20
30
40
50
60
70

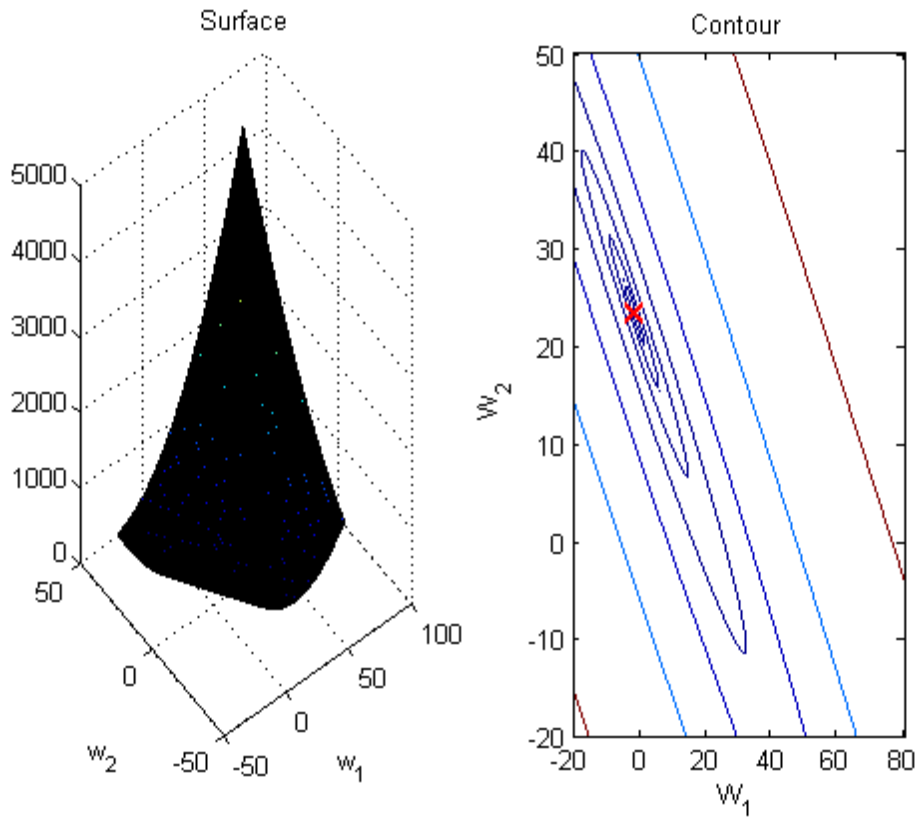

Gradient Descent
Cost Value
Iterations
400
350
300
250
200
150
100
50
0
0
10
20
30
40
50
60
70
80
90
100


Test vs Expected 2D Sample
120
100
80
60
40
20
0
0
10
20
30
40
50
60
70


Modeling Error
Generalization Error
Error Value
Iterations
0
1
2
3
4
5
6
7
8
0
10
20
30
40
50
60
70
80
90
100


Actual and Predicted Values
Attribute W1
0
20
40
60
80
100
120
-2
0
2
4


Actual Output
Linear regression
Actual and Predicted Values
Attribute W2
0
20
40
60
80
100
120
-2
0
2
4
6




*Published with MATLAB® 7.10*