**EX-5**

1. **learningCurve**

%DIMENSIONS:

% error\_train = m x 1

% error\_val = m x 1

for i = 1:m

Xtrain = X(1:i,:);

ytrain = y(1:i);

theta = trainLinearReg(Xtrain, ytrain, lambda);

error\_train(i) = linearRegCostFunction(Xtrain, ytrain, theta, 0); %for lambda = 0;

error\_val(i) = linearRegCostFunction(Xval, yval, theta, 0); %for lambda = 0;

end

1. **linearRegCostFunction**

%DIMENSIONS:

% X = 12x2 = m x 1

% y = 12x1 = m x 1

% theta = 2x1 = (n+1) x 1

% grad = 2x1 = (n+1) x 1

h\_x = X \* theta; % 12x1

J = (1/(2\*m))\*sum((h\_x - y).^2) + (lambda/(2\*m))\*sum(theta(2:end).^2); % scalar

% grad(1) = (1/m)\*sum((h\_x-y).\*X(:,1)); % scalar == 1x1

grad(1) = (1/m)\*(X(:,1)'\*(h\_x-y)); % scalar == 1x1

grad(2:end) = (1/m)\*(X(:,2:end)'\*(h\_x-y)) + (lambda/m)\*theta(2:end); % n x 1

1. **polyFeature**

for i = 1:p

X\_poly(:,i) = X.^i;

End

1. **validationCurve**

for i = 1:length(lambda\_vec)

lambda = lambda\_vec(i);

theta\_train = trainLinearReg(X, y, lambda);

[J\_train, grad] = linearRegCostFunction(X, y, theta\_train, 0);

error\_train(i) = J\_train;

%theta\_val = trainLinearReg(Xval, yval, lambda);

[J\_val, grad] = linearRegCostFunction(Xval, yval, theta\_train, 0);

error\_val(i) = J\_val;

end