1.2 Regularized linear regression cost function

$$J(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2\right) + \frac{\lambda}{2m}\left(\sum_{j=1}^{n}\theta_j^2\right),$$

```
J = sum((X * theta - y) .^2)/2/m;
temp = theta;
temp(1) = 0;
J += lambda * sum(temp.^2)/2/m;
```

1.3 Regularized linear regression gradient

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \qquad \text{for } j = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j \qquad \text{for } j \geq 1$$

```
grad(0) = sum((X*theta-y).*X(:,1))/m;
for i=2:m,
grad(i) = sum((X *theta -y) .* X(:,i))/m +lambda*theta(i)/m;
end
```

2 Bias-variance

2.1 Learning curves

You can use the `trainLinearReg` function to find the θ parameters. Note that the `lambda` is passed as a parameter to the `learningCurve` function. After learning the θ parameters, you should compute the **error** on the training and cross validation sets. Recall that the training error for a dataset is defined as

$$J_{\text{train}}(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2\right].$$

```
for i=1:m,
theta = trainLinearReg(X(1:i,:),y(1:i),lambda);
error_train(i) = linearRegCostFunction(X(1:i,:),y(1:i),theta,0);
error_test(i) = linearRegCostFunction(Xval,yval,theta,0);
end
```

3 Polynomial regression

polyFeatures:

Now, you will add more features using the higher powers of the existing feature x in the dataset. Your task in this part is to complete the code in `polyFeatures.m` so that the function maps the original training set `X` of size $m \times 1$ into its higher powers. Specifically, when a training set `X` of size $m \times 1$ is passed into the function, the function should return a $m \times p$ matrix `X_poly`,

where column 1 holds the original values of `X`, column 2 holds the values of `X.^2`, column 3 holds the values of `X.^3`, and so on. Note that you don't have to account for the zero-eth power in this function.

```
for i=1:p,
X_poly(:,i) = X .^ i;
end
```

3.3 Selecting lambda using a cross validation set

```
lambda_vec = [0 0.001 0.003 0.01 0.03 0.1 0.3 1 3 10];
for i = 1:length(lambda_vec),
lambda = lambda_vec(i);
theta = trainLinearReg(X,y,lambda);
error_train(i) = linearRegCostFunction(X,y,theta,0);
error_test(i) = linearRegCostFunction(Xval,yval,theta,0);
end
```