17.4.27（多项式回归，学习曲线，误差度量）

Polynomial regression

对于训练集，当 d 较小时，模型拟合程度更低，误差较大；随着 d 的增长，拟合程度提高，误差减小。

对于交叉验证集，当 d 较小时，模型拟合程度低，误差较大；但是随着 d 的增长，误差呈现先减小后增大的趋势，转折点是我们的模型开始过拟合训练数据集的时候。

d 即为多项式的最高项次数。

$$
\begin{aligned}
h_\theta(x) &= \theta_0 + \theta_1 * (\text{waterLevel}) + \theta_2 * (\text{waterLevel})^2 + \cdots + \theta_p * (\text{waterLevel})^p \\
&= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_p x_p.
\end{aligned}
$$

Notice that by defining $x_1 = (\text{waterLevel}), x_2 = (\text{waterLevel})^2, \ldots, x_p = (\text{waterLevel})^p$, we obtain a linear regression model where the features are the various powers of the original value (waterLevel).

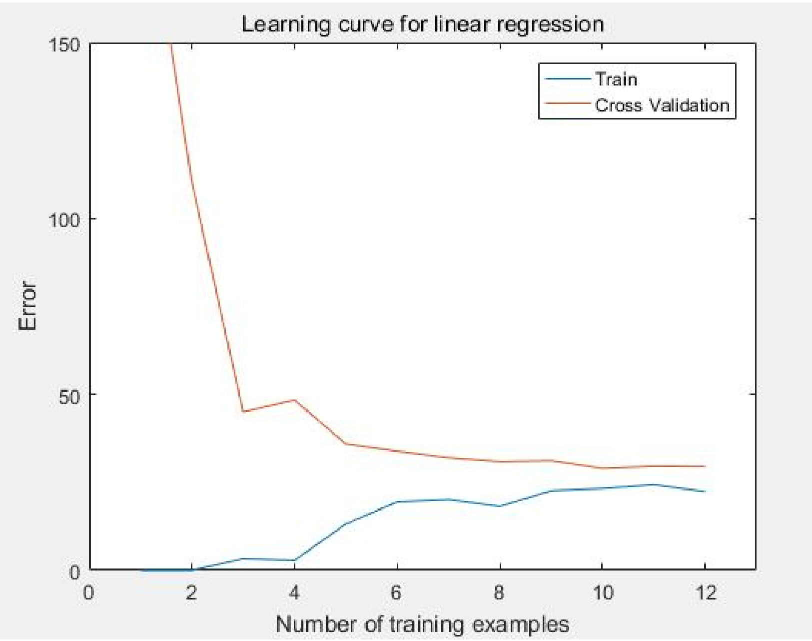
增加多项式回归方法：

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

代码运行完结果：X_poly(i, :) = [X(i) X(i).^2 X(i).^3 ... X(i).^p];

进来时X是Mx1的，执行完就变成MxP的。也就完成了多项式回归。

Learning Curve for Linear Regression线性回归的学习曲线



横轴是训练集的数量， | 轴是cost function。

代码实现：

```
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_val(i) = linearRegCostFunction(Xval, yval, theta, 0);
end
```

循环M就是训练集的样本数，第一次循环中先算一个样本时的theta，然后计算这时的训练集（一个样本）的cost function，然后计算测试集的cost function，后面的每一次循环就增加一个训练集样本知道增加到M个，也就是求M次theta，训练集和测试集的cost function

矩阵为：

```
% You need to return these values correctly
error_train = zeros(m, 1);
error_val   = zeros(m, 1);
```

- 当 λ 较小时，训练集误差较小（过拟合）而交叉验证集误差较大
- 随着 λ 的增加，训练集误差不断增加（欠拟合），而交叉验证集误差则是先减小后

增加

正确选择归一化的值 lambda

一般我们会选择0-10 之间的呈现 2 倍关系的值（如0,0.01,0.02,0.04,0.08,0.15,0.32,0.64,1.28,2.56,5.12,10 共 12 个）。

通过这12个lambda训练出十二个模型，然后计算交叉验证集，比较交叉验证机的误差，取误差小的模型。代码实现如下：

```
% Selected values of lambda (you should not change this)
lambda_vec = [0 0.001 0.003 0.01 0.03 0.1 0.3 1 3 10]';

% You need to return these variables correctly.
error_train = zeros(length(lambda_vec), 1);
error_val = zeros(length(lambda_vec), 1);
```

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

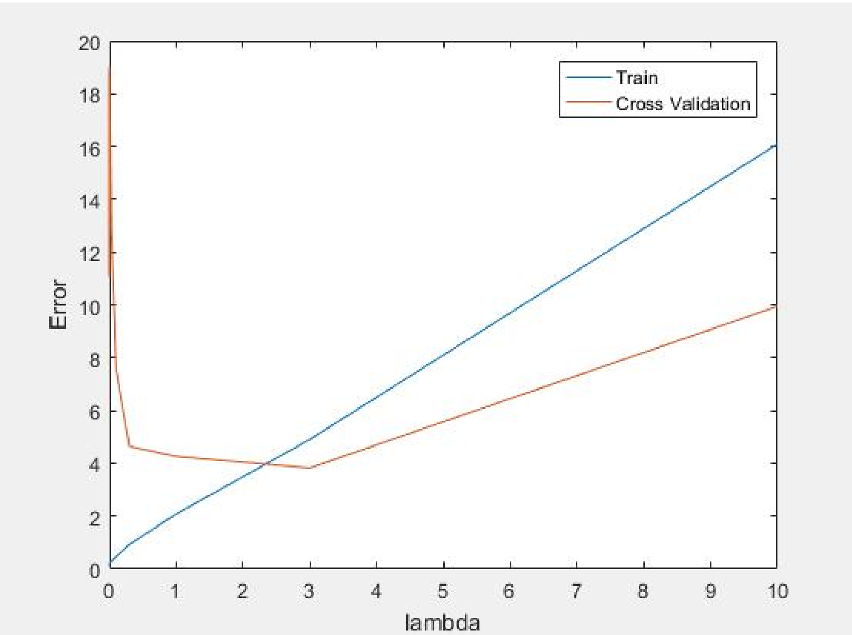
注意：计算error（cost）时，不需要正则化。

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

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\boxed{J_{cv}(\theta)} = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

可以画出来lambda与error的图像，便于观察。由下图可见两条线的交点就是交叉验证集误差最小的时候。



画图代码如下：

```
plot(lambda_vec, error_train, lambda_vec, error_val);
legend('Train', 'Cross Validation');
xlabel('lambda');
ylabel('Error');
```

类偏斜的误差度量（Error Metrics for Skewed Classes）

查准率（Precision）和**查全率**（Recall） 我们将算法预测的结果分成四种情况：

1. **正确肯定**（True Positive,TP）：预测为真，实际为真
2. **正确否定**（True Negative,TN）：预测为假，实际为假
3. **错误肯定**（False Positive,FP）：预测为真，实际为假
4. **错误否定**（False Negative,FN）：预测为假，实际为真

则：

查准率=TP/（TP+FP）例，在所有我们预测有恶性肿瘤的病人中，实际上有恶性肿瘤的病人的百分比，越高越好。

查全率=TP/（TP+FN）例，在所有实际上有恶性肿瘤的病人中，成功预测有恶性肿瘤的病人的百分比，越高越好。