实验 正则化与“偏差-方差”理论

实验目的

1. 掌握正则化线性回归的代价函数组成及其优化
2. 通过学习曲线（learning curve）了解机器学习模型对数据的拟合情况（判定过拟合、欠拟合）
3. 通过多项式特征扩展，将线性回归模型扩展至非线性回归模型，并通过正则化控制其拟合程度

实验数据

ex8data1.mat - 存储了本实验使用的6组数据，其应用背景是学习水库水位变化(x)和水坝排水间(y)的关系，其中，

- X，y： 用于训练的水库水位变化数据和对应的水坝排水量数据；
- Xval，yval： 用于确定正则项权重\lambda（也是超参数hyper-parameter）的验证数据集。
- Xtest, ytest： 未出现在训练或超参数调参过程中的测试数据。

1. 正则化的线性回归

1.1 数据读取及可视化

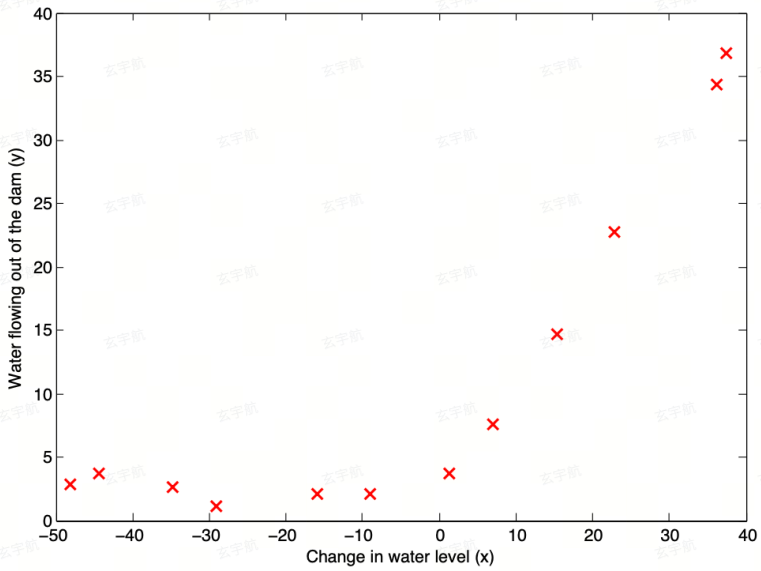
loadData(path)函数使用pandas.loadmat()读取6个数据集。读取数据集后输出返回值的维度应为(12, 1) (12, 1) (21, 1) (21, 1) (21, 1) (21, 1)。

```
def loadData(path):
    data = loadmat(path)
    X, y, Xval, yval, Xtest, ytest = data['X'], data['y'], data['Xval'], data['yval'], data['Xtest'], data['ytest']
    return X, y, Xval, yval, Xtest, ytest
```

plotData(X,y)函数可以绘制数据集的散点图。

```
# 可视化
def plotData(X,y):
    plt.figure(figsize=(12,8))
    plt.scatter(X, y,c='r', marker='x')
    plt.xlabel('Change in water level (x)')
```

```
    plt.ylabel('Water flowing out of the dam (y)')
    plt.grid(True)
plotdata(X,y)
```



然后为X、Xval、Xtest各增加一个全1列，增加后维度应为(12, 2) (21, 2) (21, 2)。

```
X, Xval, Xtest = [np.insert(x, 0, np.ones(x.shape[0]), axis=1) for x in (X, Xval, Xtest)]
```

1.2 正则化线性回归的代价函数

根据下式给出的代价函数，完成linearRegCostFunc(theta, X, y, lmd=1)函数。

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

代价函数中超参数lambda用来控制正则项的作用程度，抑制模型过于复杂而进入过拟合（overfitting）情况。一般来说，对于使用多项式特征扩展的模型，特征的次数（degree）越高，使用正则项增长得越厉害，进而实现对高阶特征的抑制。

此外，对于由\theta平方和构成的正则项，不包括theta_0。这个在代码实现时需要注意。

如果theta初始化为[1，1]，lambda=1（参数lmd），输出结果应该是303.993。

1.3 代价函数的梯度

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

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

根据上式给出的梯度公式，完成linearRegCostFuncGradient(theta, X, y, lmd=1)。

如果\theta初始化为[1;1]，\lambda=1（参数lmd），梯度的计算结果应为[-15.30;598.250]

1.4 训练正则化线性回归（Fitting linear regression）与欠拟合（underfitting）

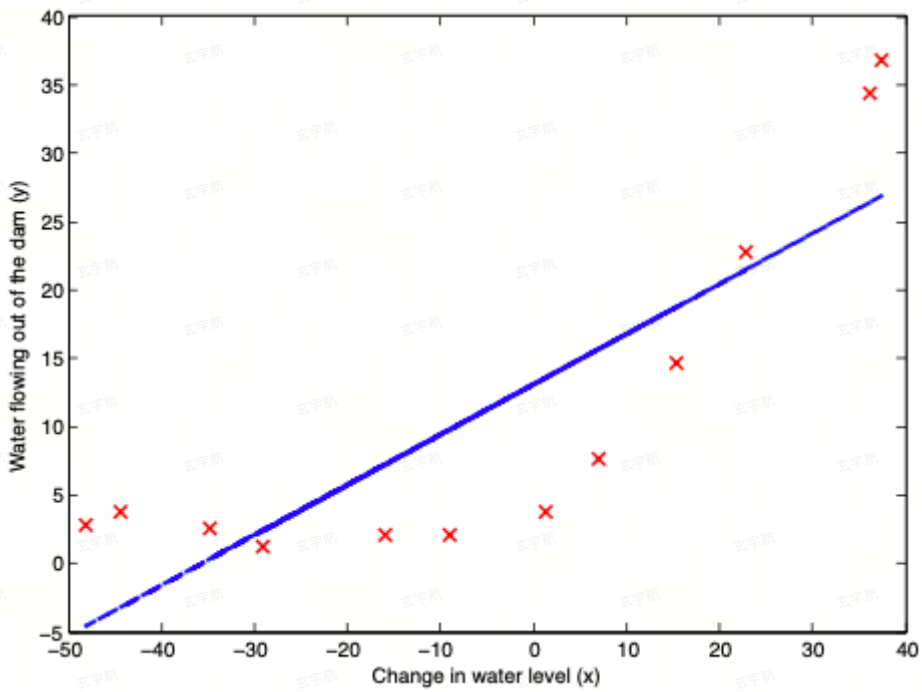
完成函数trainLinearRegression(X, y, lmd)，其中，正则化的超参数lambda设为0，因为当不采用多项式法扩增特征，正则项不太起作用。此处的优化采用scipy.optimize.minimize()函数对代价函数进行最小化。

之后，再绘制训练得到的线性回归模型。

```
from scipy.optimize import minimize

def trainLinearRegression(X, y, lmd):
    theta = np.zeros(X.shape[1])
    rslt = minimize(fun = linearRegCostFunc, x0 = tehta, args = (X, y, lmd), methed = 'TNC', jac = linearRegCostFuncGradient)

    return rslt.x
```



从上图可以看出，对于非线性数据，线性回归模型由于其只使用一阶特征的线性组合，因此，线性模型很难拟合非线性数据，这种模型过于简单导致无法很好地拟合数据的情况被称之为欠拟合（underfitting），相应地，应该提升模型的复杂程度（描述能力）以更好地适应任务数据。

2. 偏差-方差理论

偏差和方差（bias-variance）是机器学习中用于度量模型拟合任务数据的重要指标，其中偏差（bias）主要用来描述模型在训练数据集上的预测性能，方差（variance）则用来描述在同分布下的不同训练数据集上的预测性能的方差，在很多场合下也被认作是在测试数据集上的预测性能。

在实际的机器学习模型应用中，偏差和方差的两种组合经常出现：（1）高偏差、低方差：这种情况一般是由于模拟过于简单以至于难以拟合任务数据，被称为欠拟合（underfitting）；（2）低偏差、高方差：一般是由于模型过于复杂使得模型过于拟合任务中的训练数据，以至于来自于同分布的多个不同训练数据集训练得到的模型性能变化过大，或对于来自于同分布的测试数据集上的预测性能表现不好（而在训练数据集上该模型的训练误差已经很小）。这种情况被称之为过拟合（overfitting）。

上一组实验结果便是用于展现欠拟合的情况。

为了更好地展现模型的拟合情况，多使用学习曲线（Learning curve）来展现偏差方差、欠拟合和过拟合情况。

2.1 学习曲线

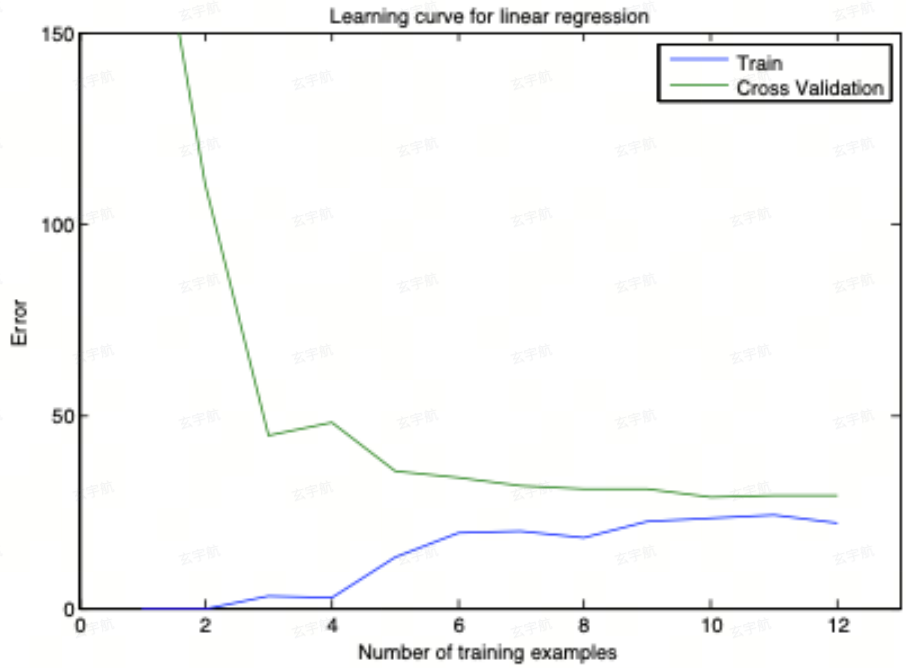
为了绘制学习曲线，需要不同大小的训练数据集和验证集来计算训练误差和验证误差。为了获得不同的训练集大小，通过调整变量i来获取，也就是使用训练数据集的前i个训练样本。

模型的训练可以调用之前的trainLinearRegression(X, y, lmd)来进行，需要注意的是对于训练误差请将lambda设为0。

```
def plotLearningCurve(X,y,Xval,yval,l):
    size = range(1, len(X)+1)
    error_train, error_val = [], []
    for i in size:
        i_theta = trainLinearRegression(X[:i, :], y[:i], l)
        i_train_cost = CostReg(i_theta, X[:i, :], y[:i], 0)
        i_val_cost = CostReg(i_theta, Xval, yval)
        error_train.append(i_train_cost)
        error_val.append(i_val_cost)

    fix, ax = plt.subplots(figsize=(12, 8))
    ax.plot(size,error_train,label="Train",color="blue")
    ax.plot(size,error_val,label="Cross Validation",color="green")
    ax.legend()
    ax.set_xlabel("Number of training examples")
    ax.set_ylabel("Error")
    ax.set_title("Learning curve for linear regression")
    ax.grid(False)
    plt.show()

plotLearningCurve(X, y, Xval, yval, 0)
```



该图可以进一步反映上一节中训练得到的线性回归模型无法很好地拟合训练数据，具体体现为训练误差（error_train）和验证误差（error_val）都比较高。尤其是作为基本条件的训练误差很高（high bias），而此种情况反映的是模型过于简单导致线性模型无法拟合非线性数据集。需要增强模型的描述能力，即将线性回归扩展为可以拟合非线性数据的多项式回归。

3. 可以拟合非线性数据的多项式回归（polynomial regression）

由于之前的水坝放水任务数据集中，作为输入的水位变化(x)和作为输出的排水量(y)之间不呈线性关系，因此，线性回归模型y=theta_1 * x+ theta_0不能很好地拟合非线性的任务数据。

为解决此问题，在回归模型中引入水位变化的高阶表示以提升模型拟合非线性数据的能力，即

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

请仿照之前特征的多项式扩展，完成polyFeatures(X, degree)

```
def polyFeatures(X, degree):
    # 计算X的各次幂
    '''code here'''
    return Xpoly
```

3.1 学习多项式回归

类似于之前的特征的多项式扩展，因为p次方的原因，各维数据取值范围差距大，因此，需要对特征进行标准化（normalization），使其取值范围大致一致。此处使用均值标准差进行标准化。注意在

featureNormalization(X, means, stds)函数中，第一列是配合theta_0的，因此，保留全1从第二列开始。可以调用已经给出的get_means_std(X)函数。对验证集、测试集进行标准化处理时也要使用训练集的均值和标准差。公式如下：

$$x_{new} = \frac{x - \mu}{\sigma}$$

其中μ为均值，σ为标准差。

```
def featureNormalization(X, means, stds):
    '''code here'''
    return Xnorm
```

```
power = 8
train_means, train_stds = get_means_std(polyFeatures(X, power))
X_norm = featureNormalize(polyFeatures(X, power), train_means, train_stds)
Xval_norm = featureNormalize(polyFeatures(Xval, power), train_means,
train_stds)
Xtest_norm = featureNormalize(polyFeatures(Xtest, power), train_means,
train_stds)
```

在做完数据准备后，调用trainLinearRegression()函数进行模型学习，然后观察绘制的模型和学习曲线，尤其是和上一节绘制的学习曲线进行对比。

此外，因为引入水位变化的一阶至八阶特征，使得模型变得更为复杂，描述能力更强，以至于已经可以使得训练误差为0。但通过对学习曲线的观察，在验证集上的误差很难再下降，这就是前面所提的第二种情况——过拟合（overfitting）。

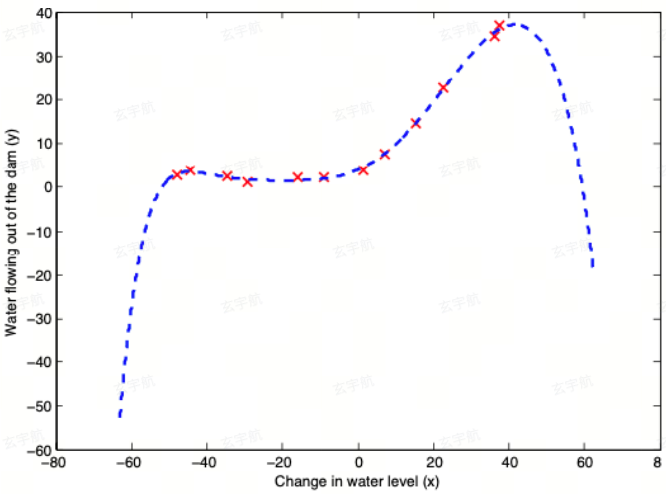


Figure 4: Polynomial fit, $\lambda = 0$

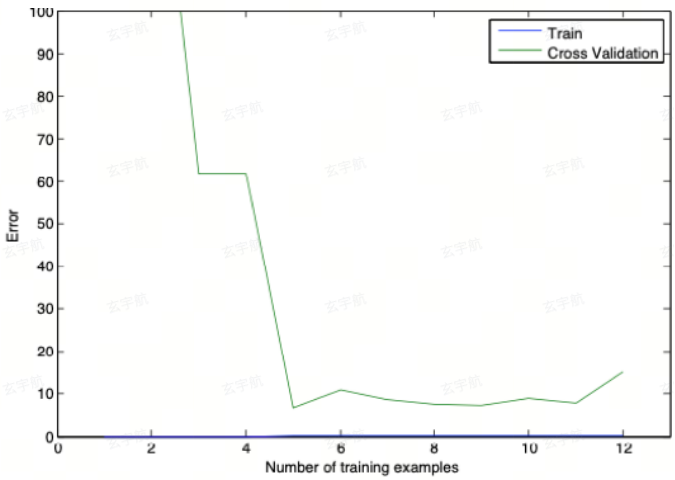


Figure 5: Polynomial learning curve, $\lambda = 0$

3.2 调整正则化参数

为解决过拟合问题，可以通过设置代价函数中的正则化参数lambda，抑制高阶特征在代价函数中的占比，进而抑制模型的复杂程度（阶数）不会过高。

此处，尝试lambda=1，然后，观测对应的模型和学习曲线。从中，应该可以看出，虽然训练误差已经不为0，但验证误差下降明显，这在一定程度解决了过拟合中的高方差问题。

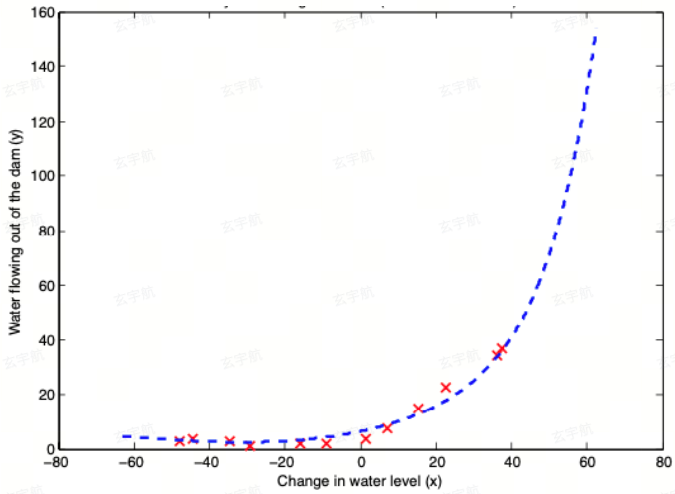


Figure 6: Polynomial fit, $\lambda = 1$

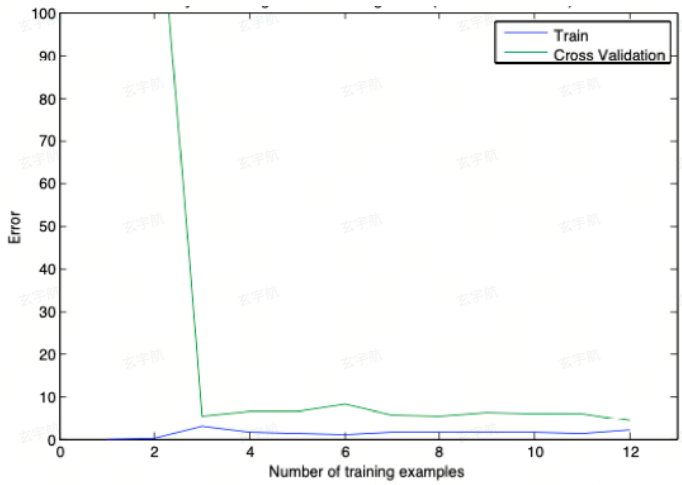


Figure 7: Polynomial learning curve, $\lambda = 1$

当设置lambda=100后，通过打印出来的模型可以看出，由于过分抑制模型的复杂程度，导致模型拟合任务数据的能力大幅下降。

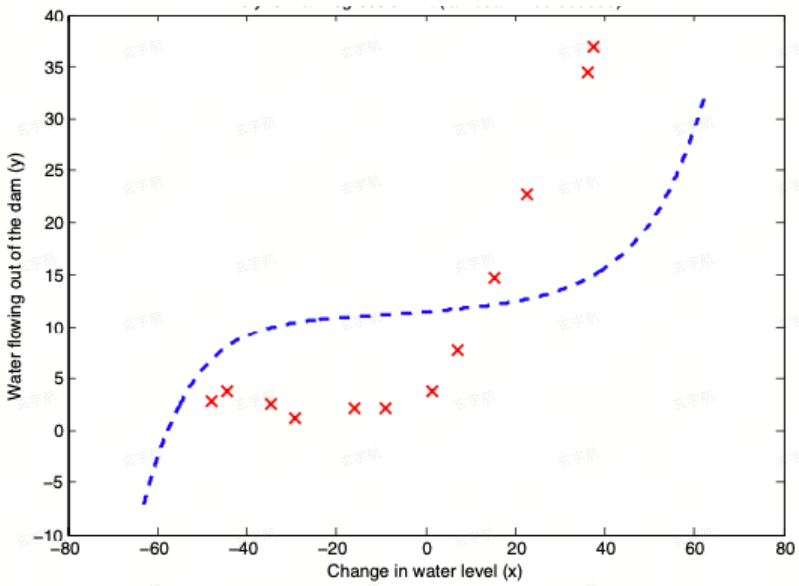


Figure 8: Polynomial fit, $\lambda = 100$

3.3 使用交叉验证数据集确定正则化参数lambda

由于正则化参数lambda是人为设定的超参数（hyper parameter），无法由机器自动学习，所以，通过交叉验证选择性能最高时对应的lambda参数作为最后的模型参数。

在实验中，因为不了解训练误差和模型惩罚的量纲，因此，使用指数方式设定lambda的初步选值范围，如{0, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10}。

```
lambdas = [0., 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1., 3., 10.]
errors_train, errors_val = [], []
for l in lambdas:
    theta = trainLinearRegression(X_norm, y, l)
    errors_train.append(linearRegCostFunc(theta,X_norm,y,0))  # 记得把lambda = 0
    errors_val.append(linearRegCostFunc(theta,Xval_norm,yval,0))
```

在积累下各lambda下的训练误差和验证误差后，可以通过绘制学习曲线观察性能变化情况。从中可看出lambda取3附近，验证误差最小。如果需要更细致的调参，可以在3周围进行等间隔（如3±0.2 * i，i=1, ... , 5）调参，作为最后的模型参数。

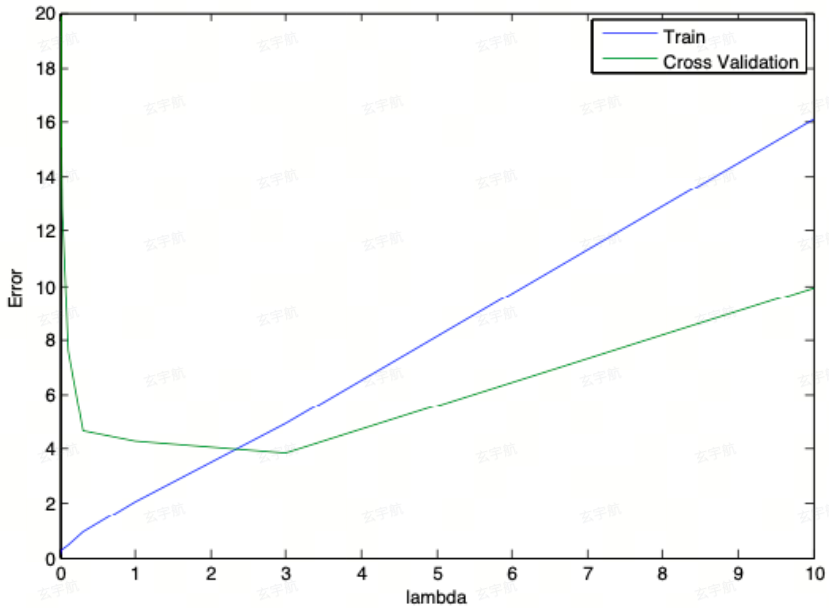


Figure 9: Selecting λ using a cross validation set

3.4 计算测试集误差

在评估各类机器学习模型上，其真实性能只有在真实环境下才能测量。在具体实践中，往往从标注数据中选出部分数据中，使其不出现在模型学习（或训练）和超参数的调整（或验证）上，以尽量模拟真实任务。

此处，使用数据集中的Xtest和ytest作为测试数据，使用lambda=3和对应的模型参数\theta作为“最后”的模型，计算其误差。其值应约为3.6。

```
theta = trainLinearRegression(X_norm, y, 3)
print('Test error(lambda={}) = {}'.format(3, linearRegressionCostFunc(theta,
Xtest_norm, ytest, 0)))
```