[stanford coursera 机器学习编程作业 exercise 5（正则化线性回归及偏差和方差）](https://www.cnblogs.com/hapjin/p/6114466.html)

本文根据水库中蓄水标线(water level) 使用正则化的线性回归模型预 水流量(water flowing out of dam)，然后 debug 学习算法 以及 讨论偏差和方差对 该线性回归模型的影响。

**①可视化数据集**

本作业的数据集分成三部分：

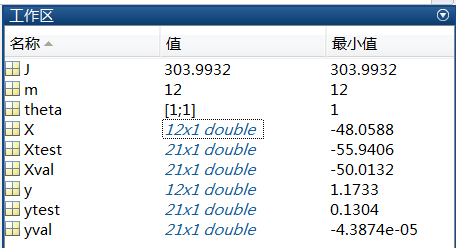
ⓐ训练集(training set)，样本矩阵(训练集)：X，结果标签(label of result)向量 y

ⓑ交叉验证集(cross validation set)，确定正则化参数 Xval 和 yval

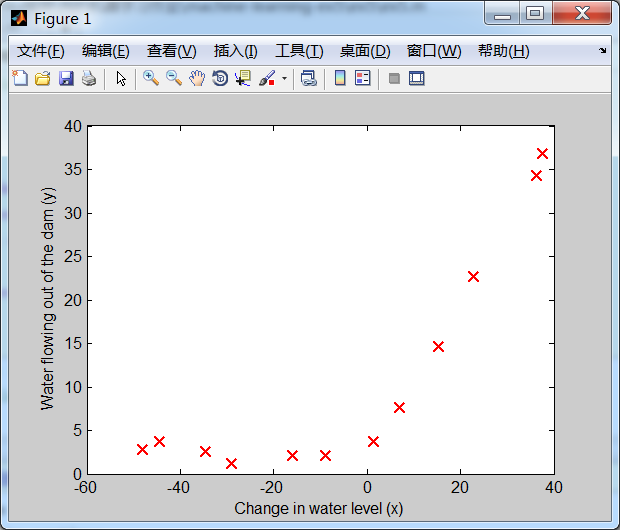
ⓒ测试集(test set) for evaluating performance，测试集中的数据 是从未出现在 训练集中的

将数据加载到Matlab中如下：训练集中一共有12个训练实例，每个训练实例只有一个特征。故假设函数hθ(x) = θ0·x0 + θ1·x1 ，用向量表示成：hθ(x) = θT·x

一般地，x0 为 bais unit，默认 x0==1

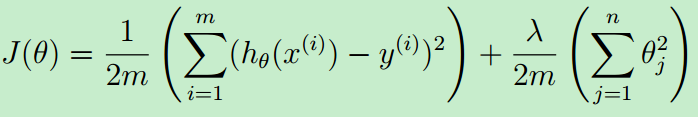


训练数据集中的数据图形表示如下：



**②正则化线性回归模型的代价函数**

代价函数公式如下：



Matlab代码实现如下：这里的代价函数是用**向量(矩阵)乘法**来实现的。

具体证明可参考：[Linear Regression---实现一个线性回归](http://www.cnblogs.com/hapjin/p/6079012.html)

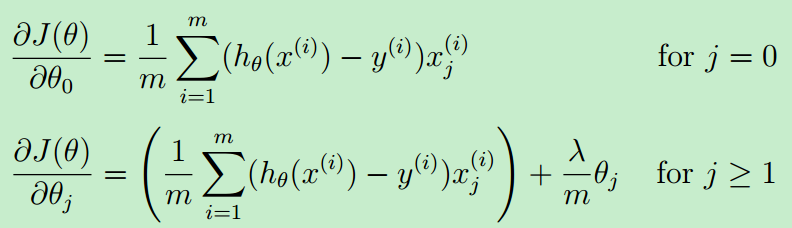
reg = (lambda / (2\*m)) \* ( ( theta( 2:length(theta) ) )' \* theta(2:length(theta)) );

J = sum((X\*theta-y).^2)/(2\*m) + reg;

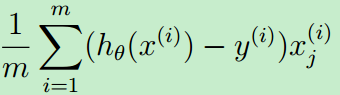
 注意：由于θ0不参与正则化项的，故上面Matlab数组下标是从2开始的(Matlab数组下标是从1开始的，θ0是Matlab数组中的第一个元素)。

**③正则化的线性回归梯度**

梯度的计算公式如下：



其中，下面公式的向量表示就是：[XT · (X·θ - y)]/m，用Matlab表示就是：X'\*(X\*theta-y) / m



梯度的Matlab代码实现如下：

grad\_tmp = X'\*(X\*theta-y) / m;

grad = [ grad\_tmp(1:1); grad\_tmp(2:end) + (lambda/m)\*theta(2:end) ];

**④使用Matlab的函数 fmincg 函数训练线性回归模型，得到模型的参数，trainLinearReg.m如下：**

[复制代码](javascript:void(0);)

function [theta] = trainLinearReg(X, y, lambda)

%TRAINLINEARREG Trains linear regression given a dataset (X, y) and a

%regularization parameter lambda

% [theta] = TRAINLINEARREG (X, y, lambda) trains linear regression using

% the dataset (X, y) and regularization parameter lambda. Returns the

% trained parameters theta.

%

% Initialize Theta

initial\_theta = zeros(size(X, 2), 1);

% Create "short hand" for the cost function to be minimized

costFunction = @(t) linearRegCostFunction(X, y, t, lambda);

% Now, costFunction is a function that takes in only one argument

options = optimset('MaxIter', 200, 'GradObj', 'on');

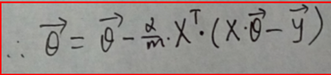
% Minimize using fmincg

theta = fmincg(costFunction, initial\_theta, options);

end

[复制代码](javascript:void(0);)

但是，在[作业一](http://www.cnblogs.com/hapjin/p/6079012.html)中，我们并不是通过Matlab的 fmincg 函数来求得模型参数的，而是通过下面的公式 在for循环中 求得 模型参数θ



其Matlab实现如下：

for iter = 1:num\_iters

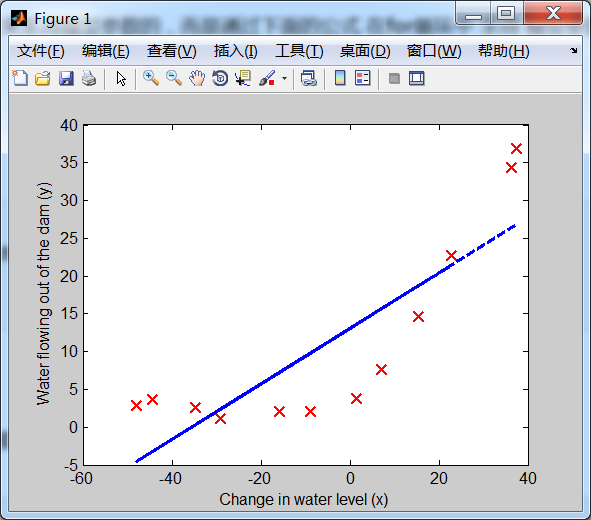
theta = theta - (alpha/m)\*X'\*(X\*theta-y); % theta 就是用上面的向量表示法的 matlab 语言实现

....

end

**⑤线性回归模型的图形化表示**

上面已经通过 fmincg 求得了模型参数了，那么我们求得的模型 与 数据的拟合程度 怎样呢？看下图：



从上图中可以看出，由于我们的数据是二维的，但是却用一个线性模型去拟合，故很明显出现了 underfiting problem

在这里，我们很容易将模型以图形化方式表现出来，因为，我们的训练数据的特征很少(一维)。当训练数据的特征很多(feature variables)时，就很难画图了(三维以上很难直接用图形表示了...)。这时，就需要用 “学习曲线”来检查 训练出来的模型与数据是否很好地拟合了。

The best fit line tells us that the model is not a good fit to the data because the data has a non-linear pattern.

While visualizing the best fit as shown is one possible way to debug your learning algorithm,  
it is not always easy to visualize the data and model(比如，当特征超过3维时...)

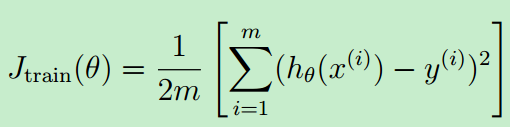
**⑥偏差与方差之间的权衡**

高偏差---欠拟合，underfit

高方差---过拟合，overfit

可以用学习曲线(learning curve)来诊断偏差--方差 问题。学习曲线的 x 轴是训练集大小(training set size)，y 轴则是交叉验证误差和训练误差。

训练误差的定义如下：



注意：训练误差Jtrain(θ)是没有正则化项的，因此在调用linearRegCostFunction时，lambda==0。Matlab实现如下(learningCurve.m)

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

function [error\_train, error\_val] = ...

learningCurve(X, y, Xval, yval, lambda)

%LEARNINGCURVE Generates the train and cross validation set errors needed

%to plot a learning curve

% [error\_train, error\_val] = ...

% LEARNINGCURVE(X, y, Xval, yval, lambda) returns the train and

% cross validation set errors for a learning curve. In particular,

% it returns two vectors of the same length - error\_train and

% error\_val. Then, error\_train(i) contains the training error for

% i examples (and similarly for error\_val(i)).

%因为 m 是一样的，所以 error\_val 和 error\_train 向量有着相同的元素个数

% In this function, you will compute the train and test errors for

% dataset sizes from 1 up to m. In practice, when working with larger

% datasets, you might want to do this in larger intervals.

%

% Number of training examples

m = size(X, 1);

% You need to return these values correctly

error\_train = zeros(m, 1);

error\_val = zeros(m, 1);

% ====================== YOUR CODE HERE ======================

% Instructions: Fill in this function to return training errors in

% error\_train and the cross validation errors in error\_val.

% i.e., error\_train(i) and

% error\_val(i) should give you the errors

% obtained after training on i examples.

%

% Note: You should evaluate the training error on the first i training

% examples (i.e., X(1:i, :) and y(1:i)).

%

% For the cross-validation error, you should instead evaluate on

% the \_entire\_ cross validation set (Xval and yval).

%

% Note: If you are using your cost function (linearRegCostFunction)

% to compute the training and cross validation error, you should

% call the function with the lambda argument set to 0.

% Do note that you will still need to use lambda when running

% the training to obtain the theta parameters.

%

% Hint: You can loop over the examples with the following:

%

% for i = 1:m

% % Compute train/cross validation errors using training examples

% % X(1:i, :) and y(1:i), storing the result in

% % error\_train(i) and error\_val(i)

% ....

%

% end

%

% ---------------------- Sample Solution ----------------------

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);

% -------------------------------------------------------------

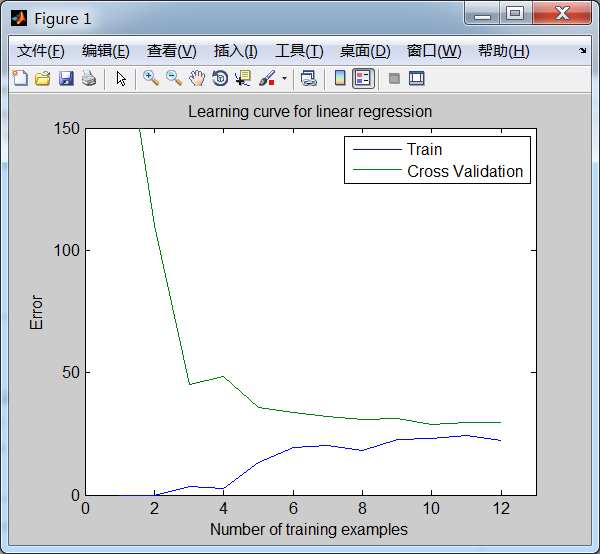
% =========================================================================

end

[复制代码](javascript:void(0);)

学习曲线的图形如下：可以看出欠拟合时，在 training examples 数目很少时，训练出来的模型还能拟合"一点点数据"，故训练误差相对较小；但对于交叉验证误差而言，它是使用未知的数据得算出来到的，而现在模型欠拟合，故几乎不能 拟合未知的数据，因此交叉验证误差非常大。

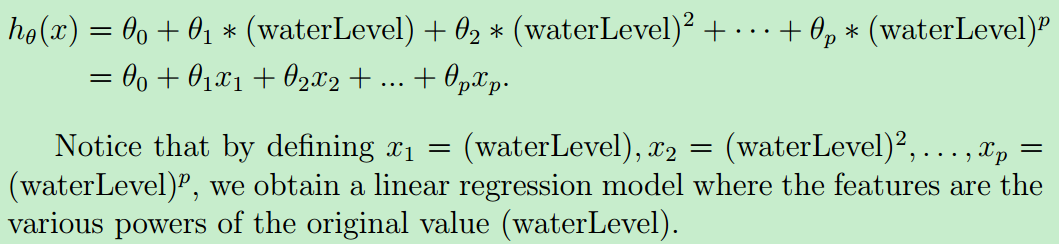
随着 training examples 数目的增多，由于欠拟合，训练出来的模型越来越来能拟合一些数据了，故训练误差增大了。而对于交叉验证误差而言，最终慢慢地与训练误差一致并变得越来越平坦，此时，再增加训练样本(training examples)已经对模型的训练效果没有太大影响了---在欠拟合情况下，再增加训练集的个数也不能再降低训练误差了。



**⑦多项式回归**

从上面的学习曲线图形可以看出：出现了underfit problem，通过添加更多的特征(features)，使用更高幂次的多项式来作为假设函数拟合数据，以解决欠拟合问题。

多项式回归模型的假设函数如下：



通过对特征“扩充”，以添加更多的features，代码实现如下：polyFeatures.m

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

function [X\_poly] = polyFeatures(X, p)

%POLYFEATURES Maps X (1D vector) into the p-th power

% [X\_poly] = POLYFEATURES(X, p) takes a data matrix X (size m x 1) and

% maps each example into its polynomial features where

% X\_poly(i, :) = [X(i) X(i).^2 X(i).^3 ... X(i).^p];

%

% You need to return the following variables correctly.

X\_poly = zeros(numel(X), p);

% ====================== YOUR CODE HERE ======================

% Instructions: Given a vector X, return a matrix X\_poly where the p-th

% column of X contains the values of X to the p-th power.

%

%

%X\_ploy(:, 1) = X;

for i = 1:p

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

end

% =========================================================================

end

[复制代码](javascript:void(0);)

“扩充”了特征之后，就变成了多项式回归了，但由于多项式回归的特征取值范围差距太大（比如有些特征的取值很小，而有些特征的取值非常大），故需要用到Normalization(归一化)，归一化的代码如下：

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

function [X\_norm, mu, sigma] = featureNormalize(X)

%FEATURENORMALIZE Normalizes the features in X

% FEATURENORMALIZE(X) returns a normalized version of X where

% the mean value of each feature is 0 and the standard deviation

% is 1. This is often a good preprocessing step to do when

% working with learning algorithms.

mu = mean(X);

X\_norm = bsxfun(@minus, X, mu);

sigma = std(X\_norm);

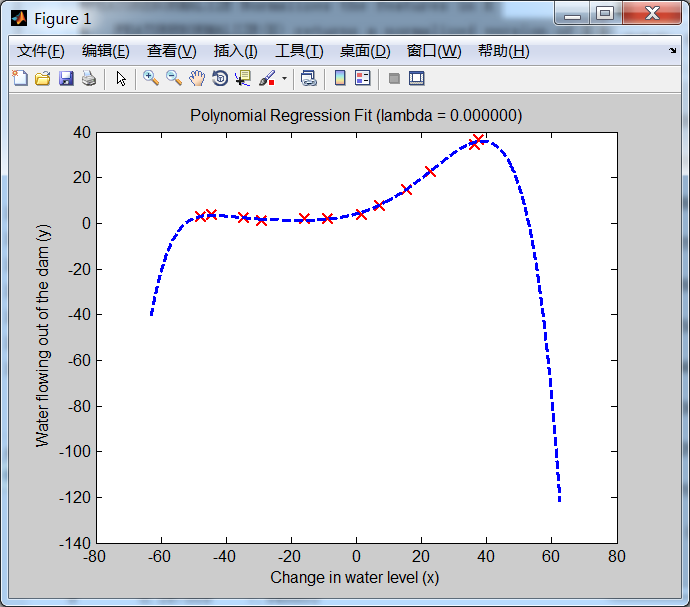
X\_norm = bsxfun(@rdivide, X\_norm, sigma);

% ============================================================

end

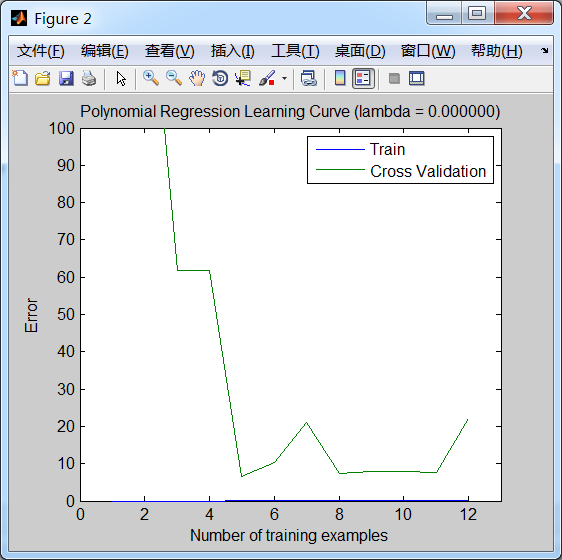
[复制代码](javascript:void(0);)

继续再用原来的linearRegCostFunction.m计算多项式回归的代价函数和梯度，得到的多项式回归模型的假设函数的图形如下：（注意：lambda==0，没有使用正则化）：



从多项式回归模型的 图形看出：它几乎很好地拟合了所有的训练样本数据。因此，可认为出现了：过拟合问题(overfit problem)---高方差

多项式回归模型的学习曲线 图形如下：

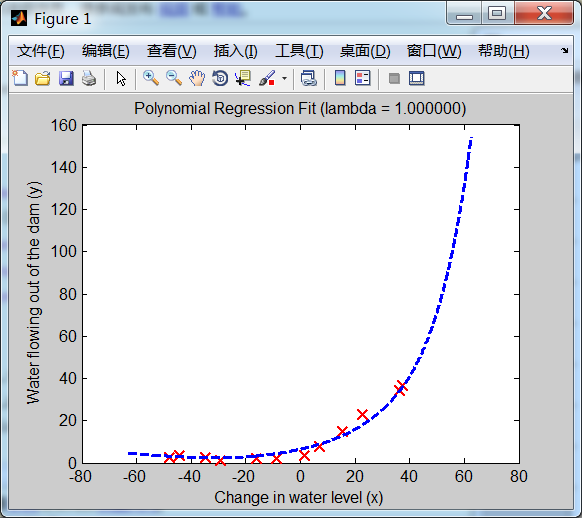


从多项式回归的学习曲线图形看出：训练误差几乎为0(非常贴近 x 轴了)，这正是因为过拟合---模型几乎完美地穿过了训练数据集中的每个数据点，从而训练误差非常小。

交叉验证误差先是很大(训练样本数目为2时)，然后随着训练样本数目的增多，cross validation error 变得越来越小了(训练样本数目2 增加到 5 过程中)；然后，当训练样本数目再增多时（11个以上的训练样本时...），交叉验证误差又变得大了(过拟合导致泛化能力下降)。

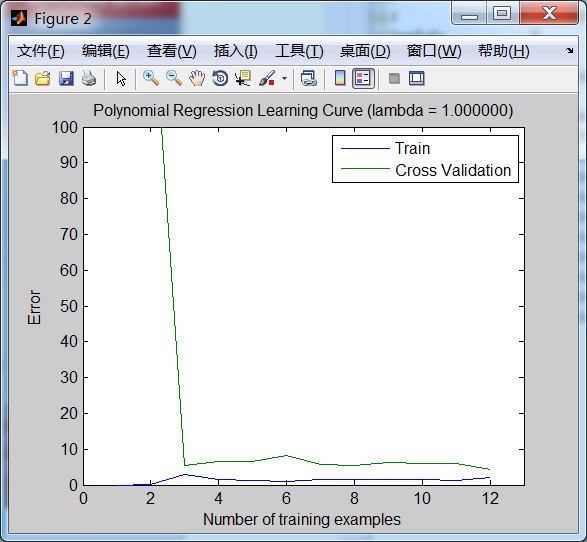
**⑧使用正则化来解决多项化回归模型的过拟合问题**

设置正则化项 lambda == 1(λ==1)时，得到的模型假设函数图形如下：



可以看出：这里的拟合曲线不再是 lambda == 0 时 那样弯弯曲曲的了，也不是非常精准地穿过每一个点，而是变得相对比较平滑。这正是 正则化 的效果。

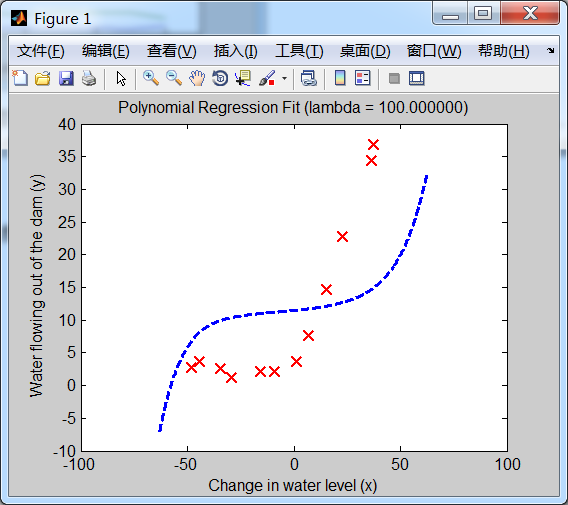
lambda==1 (λ==1) 时的学习曲线如下：



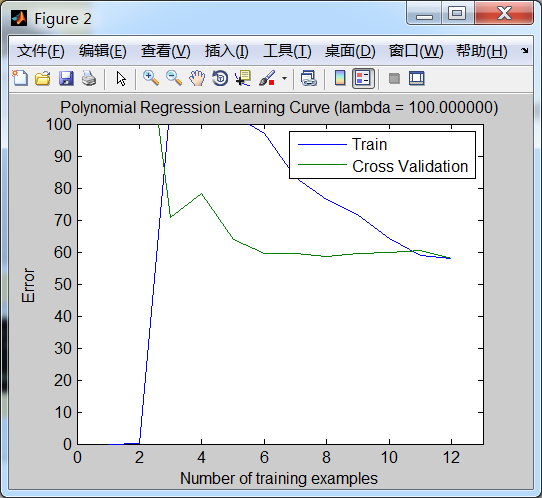
lambda==1时的学习曲线表明：该模型有较好的泛化能力，能够对未知的数据进行较好的预测。因为，它的交叉验证误差和训练误差非常接近，且非常小。（训练误差小，表明模型能很好地拟合数据，但有可能出现过拟合的问题，过拟合时，是不能很好地对未知数据进行预测的；而此处交叉验证误差也小，表明模型也能够很好地对未知数据进行预测）

最后来看下，多项式回归模型的正则化参数 lambda == 100(λ==100)时的情况：（出现了underfit problem--欠拟合--高偏差）

模型“假设函数”曲线如下：



学习曲线图形如下：



**⑨如何自动选择合适的 正则化参数 lambda(λ) ？**

从第⑧点中看出：正则化参数 lambda(λ) 等于0时，出现了过拟合， lambda(λ) 等于100时，又出现了欠拟合， lambda(λ) 等于1时，模型刚刚好。

那在训练过程中如何自动选择合适的lambda参数呢？

可以使用交叉验证集(根据交叉验证误差来选择合适的 lambda 参数)

Concretely, you will use a cross validation set to evaluate how good each lambda value is.  
 After selecting the best lambda value using the cross validation set,   
we can then evaluate the model on the test set to estimate how well the model will perform on actual unseen data.

 具体的选择方法如下：

首先有一系列的待选择的 lambda(λ) 值，在本λ作业中用一个lambda\_vec向量保存这些 lambda 值（一共有10个）：

lambda\_vec = [0 0.001 0.003 0.01 0.03 0.1 0.3 1 3 10]'

然后，使用训练数据集 针对这10个 lambda 分别训练 10个正则化的模型。然后对每个训练出来的模型，计算它的交叉验证误差，选择交叉验证误差最小的那个模型所对应的lambda(λ)值，作为最适合的 λ 。（注意：在计算训练误差和交叉验证误差时，是没有正则化项的，相当于 lambda==0）

for i = 1:length(lambda\_vec)

theta = trainLinearReg(X,y,lambda\_vec(i));%对于每个lambda,训练出模型参数theta

%compute jcv and jval without regularization,causse last arguments(lambda) is zero

error\_train(i) = linearRegCostFunction(X, y, theta, 0);%计算训练误差

error\_val(i) = linearRegCostFunction(Xval, yval, theta, 0);%计算交叉验证误差

end

对于这10个不同的 lambda，计算出来的训练误差和交叉验证误差如下：

[复制代码](javascript:void(0);)

lambda Train Error Validation Error

0.000000 0.173616 22.066602

0.001000 0.156653 18.597638

0.003000 0.190298 19.981503

0.010000 0.221975 16.969087

0.030000 0.281852 12.829003

0.100000 0.459318 7.587013

0.300000 0.921760 4.636833

1.000000 2.076188 4.260625

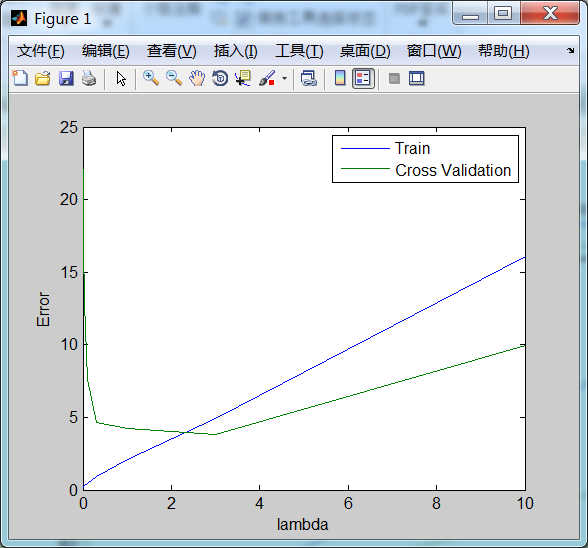
3.000000 4.901351 3.822907

10.000000 16.092213 9.945508

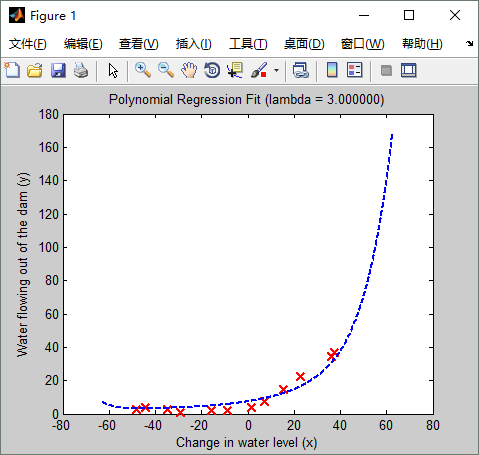
[复制代码](javascript:void(0);)

训练误差、交叉验证误差以及 lambda 之间的关系 图形表示如下：

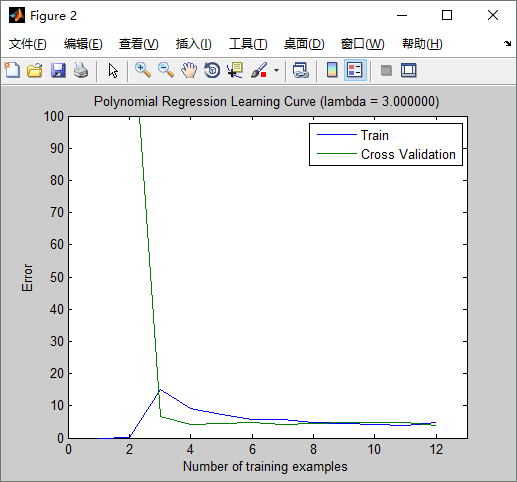
当 lambda >= 3 的时候，交叉验证误差开始上升，如果再增大 lambda 就可能出现欠拟合了...



从上面看出：lambda == 3 时，交叉验证误差最小。lambda==3时的拟合曲线如下：（可与 lambda==1时的拟合曲线及学习曲线对比一下，看有啥不同）



学习曲线如下：



完整地理一遍这个东西，真的好累啊。还是有很多不懂。。。。

原文：http://www.cnblogs.com/hapjin/p/6114466.html