COMP 540 Statistical Machine Learning

Xiang Zhou (xz58) - Guangyuan Yu ()

2017.01.17

1 Locally weighted linear regression

- Part 1

$$J(\theta) = \sum_{i=1}^{m} w^{(i)}(\theta^T x^{(i)} - y^{(i)})^2 = (X\theta - y)^T W (X\theta - y)$$

$$A = X\theta - y = \begin{pmatrix} x_1^{(1)} & \cdots & x_D^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(N)} & \cdots & x_D^{(N)} \end{pmatrix}$$

$$J(\theta) = A^T W A = (x^{(1)^T} - y^{(1)} \cdots x^{(N)^T}\theta - y^{(N)}) \begin{pmatrix} w^{(1)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & w^{(n)} \end{pmatrix} \begin{pmatrix} x^{(1)^T}\theta - y^{(1)} \\ \vdots \\ x^{(N)^T}\theta - y^{(N)} \end{pmatrix}$$

$$= \sum_{i=1}^{m} w^{(i)}(x^{(i)^T}\theta - y^{(i)})^2 = w^{(i)}(\theta^T x^{(i)} - y^{(i)})^2$$

- Part 2

$$\begin{aligned} J(\theta) &= (X\theta - y)^T W (X\theta - y) \\ &= (\theta^T X^T - yT) W (X\theta - y) \\ &= \theta^T X^T W X \theta - \theta^T X^T W X y - y^T W X \theta + y^T W y \end{aligned}$$

Because $(\theta^T X^T W)$ and y are $1 \times N$ and $N \times 1$ respectively, $\theta^T X^T W y = y^T W X \theta$.

$$\frac{dJ(\theta)}{d\theta} = 0 = \frac{d}{d\theta}[\theta^T X^T W X \theta] - 2\frac{d}{d\theta}[\theta^T X^T W y] + \frac{d}{d\theta}[y^T W y]$$

By taking the matrix derivative, we get:

$$0 = 2X^T W X \theta - 2X^T W y$$

$$\hat{\theta} = (X^T W X)^{-1} X^T W y$$

- Part 3

Algorithm 1 Calculating θ by Batch Gradient Descent

Input: Data Matrix $X \in m \times d+1$, vector $y \in m \times 1$, learning rate $\alpha \in \mathbb{R}$, input vector $x \in \mathbb{R}^{d+1}$, bandwidth of sphere of influence around x τ
Output: Vector $\theta \in \mathbb{R}^{d+1}$ that minimizes weighted LSE

$w \leftarrow m \times n$ zeros matrix
$\theta \leftarrow d \times 1$ zeros matrix
$grad \leftarrow d \times 1$ zeros matrix
for $j = 0$ to m **do**
  $w_j^{(j)} \leftarrow \frac{(x-X^{(j)})^T(x-X^{(j)})}{2\tau^2}$
end for
for $j = 0$ to 5000 **do** ▷ arbitrary number of iterations
  $grad \leftarrow \frac{X^T w(X\theta - y)}{m}$
  $\theta \leftarrow \theta - \alpha * grad$
end for
return θ

2 Properties of the linear regression estimator

- Under the assumption that $E[\epsilon] = 0$,

$$\begin{aligned} \theta &= (X^T X)^{-1} X^T y \\ E[\theta] &= E[(X^T X)^{-1} X^T y] \\ &= (X^T X)^{-1} X^T E[X\theta^* + \epsilon] \\ &= (X^T X)^{-1}(X^T X)\theta^* + (X^T X)^{-1} X^T E[\epsilon] \\ &= \theta^* + 0 = \theta^* \end{aligned}$$

- Under the assumption that $Var(\epsilon) = \sigma^2$,

$$\begin{aligned} Var(\theta) &= Var((X^T X)^{-1} X^T y) \\ &= Var((X^T X)^{-1} X^T (X\theta^* + \epsilon)) \\ &= Var(\theta^*) + Var((X^T X)^{-1}\epsilon) \\ &= 0 + (X^T X)^{-1} Var(\epsilon) \\ &= (X^T X)^{-1}\sigma^2 \end{aligned}$$

3 Problem 3.1 Implementing linear regression

3.1 Implementing linear regression with one variable

3.2 Problem 3.1 A1Computing the cost function J(θ)

see our code

Figure 1: Plotting the data

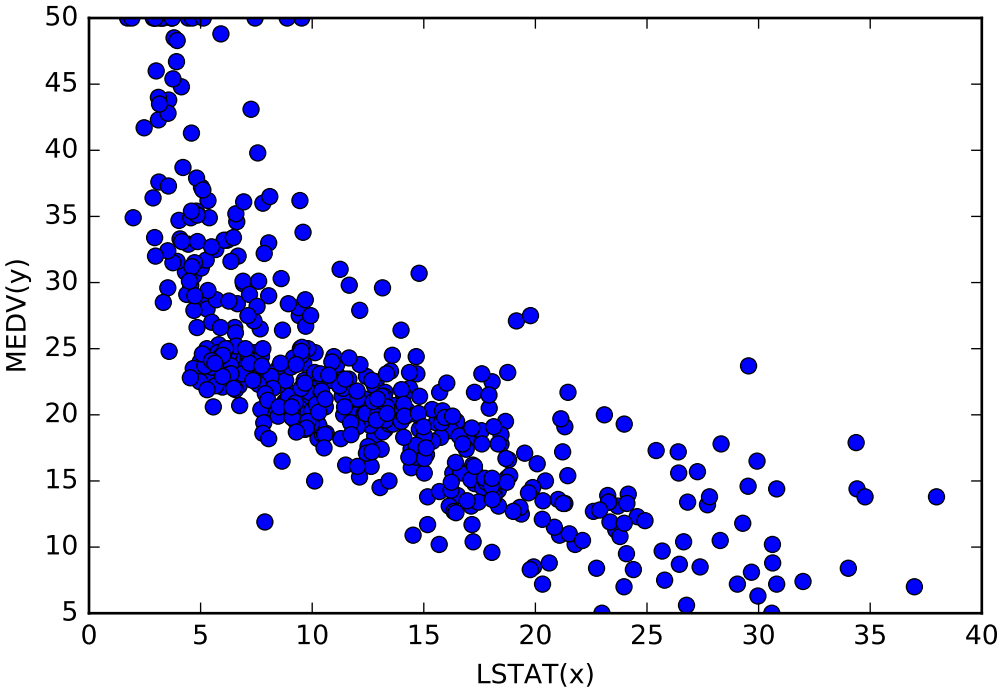


Figure 2: Fitting a linear model to the data in fig 1

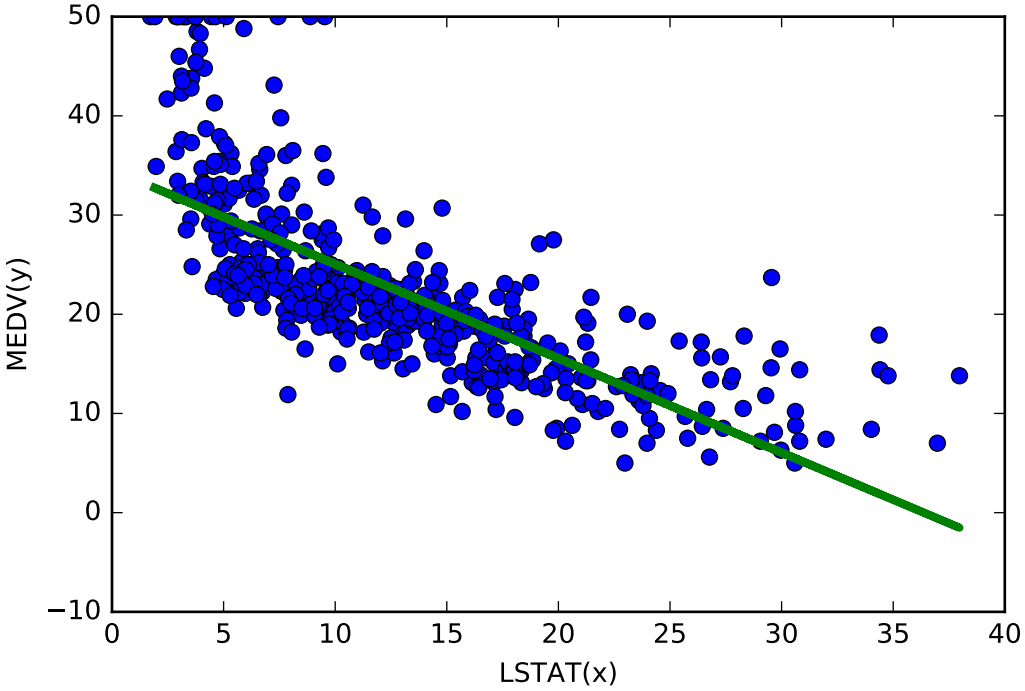
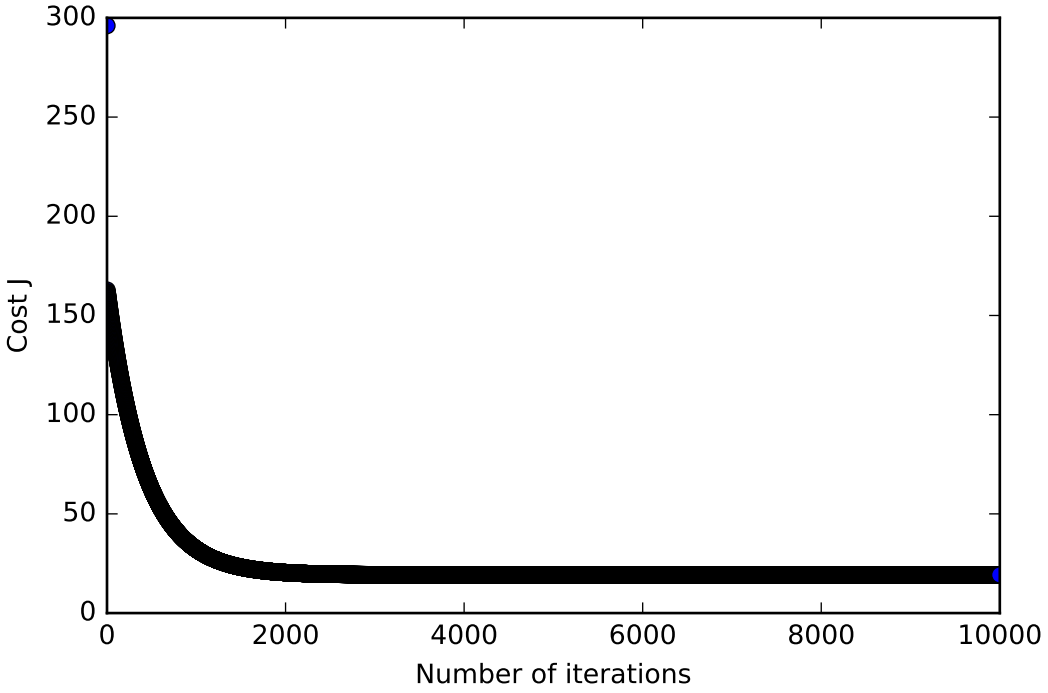


Figure 3: convergence of gradient



3.3 Problem 3.1 A2 Implmenting gradient descent

3.4 Problem 3.1 A3 Predicting on unseen data

Theta found by gradient descent: [34.55363411 -0.95003694] For lower status percentage = 5%, we predict a median home value of 298034.494122 For lower status percentage = 50%, we predict a median home value of -129482.128898

3.5 assessing model quality

The coefficients computed by sklearn: 34.5538408794 and -0.950049353758 2 fold cross_validation MSE = 39.5116296814 2 fold cross_validation r_squared = 0.510083569547

5 fold cross_validation MSE = 42.61890333 5 fold cross_validation r_squared = 0.297106799977

10 fold cross_validation MSE = 41.8292437273 10 fold cross_validation r_squared = -0.18466981353 So k=2 is the best

3.6 Problem 3.1 B Linear regression with multiple variables

3.7 Problem 3.1 B1Feature normalization

see in the code

3.8 Problem 3.1 B2 Loss function and gradient descent

Theta computed by gradient descent: [2.25328063e+01 -9.13925619e-01 1.06949712e+00 1.07531669e-01 6.87258582e-01 -2.05340341e+00 2.67719690e+00 1.55788957e-02 -3.10668099e+00 2.56946272e+00

Figure 4: Surface plot of J

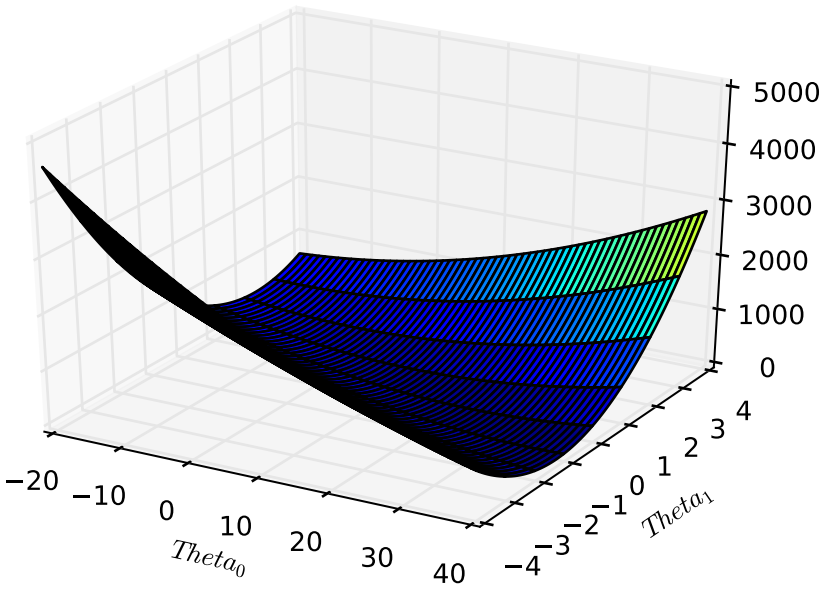


Figure 5: contour plot of J

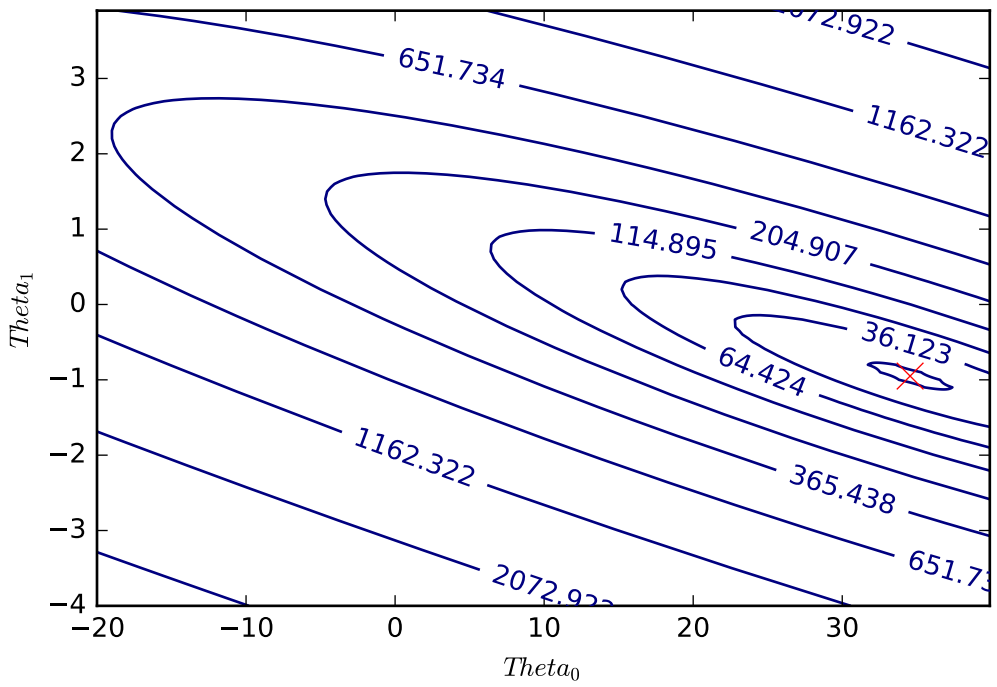
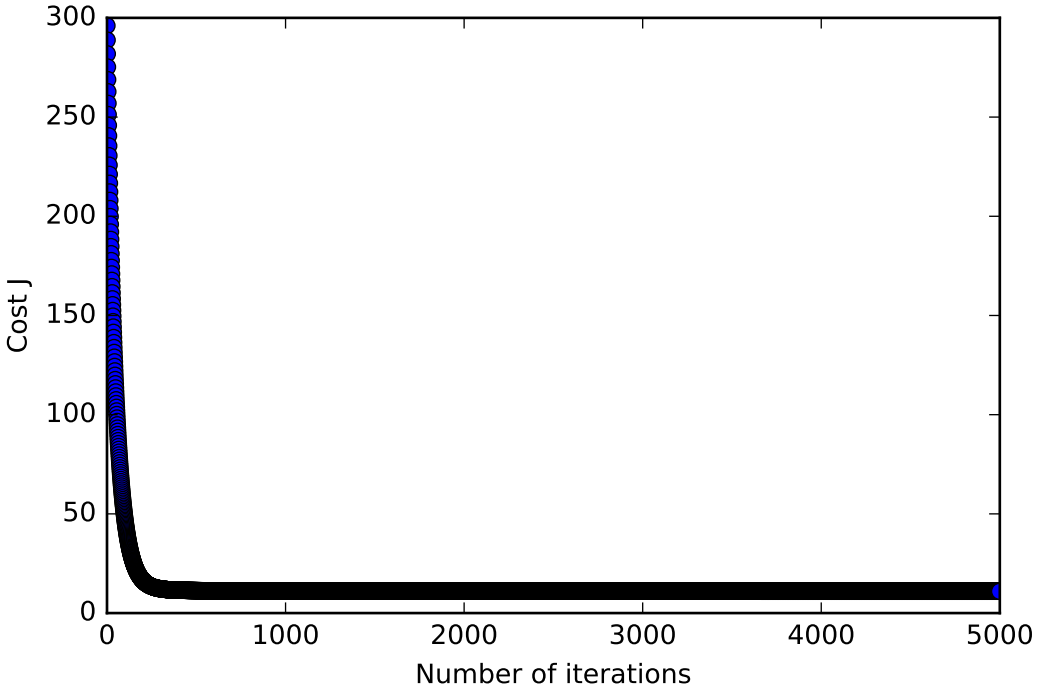


Figure 6: J for multiple variables



-1.97453430e+00 -2.05873147e+00 8.55982884e-01 -3.74517559e+00]

3.9 Problem 3.1 B3 Making predictions on unseen data

For average home in Boston suburbs, we predict a median home value of 225328.063241 The predictions match very well. Even though the theta are different but we can still have a good prediction because we have a lot of parameters. The difference of theta may come from the normalization of data.

3.10 Problem 3.1 B4 Normal equation

Theta computed by direct solution is: [3.64911033e+01 -1.07170557e-01 4.63952195e-02 2.08602395e-02 2.68856140e+00 -1.77957587e+01 3.80475246e+00 7.51061703e-04 -1.47575880e+00 3.05655038e-01 -1.23293463e-02 -9.53463555e-01 9.39251272e-03 -5.25466633e-01] For average home in Boston suburbs, we predict a median home value of 225328.063241

3.11 Problem 3.1 B5 Convergence of gradient descent

After running the four different learning rates provided (.01, .03, .1, and .3), we found that all converged, none with oscillation, on the same value. Thus, the best choice for a learning rate out of these four is 0.3 due to the fact that it converges the fastest and thus requires the least amount of iterations to finish training.

Figure 7: J for different learning rate

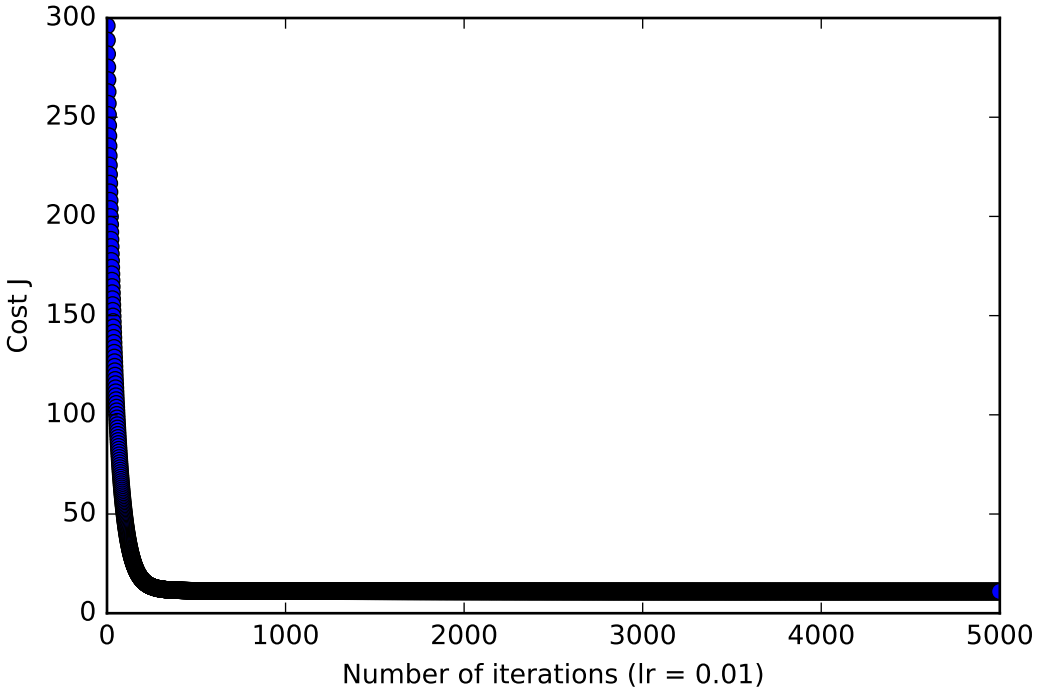


Figure 8: J for different learning rate

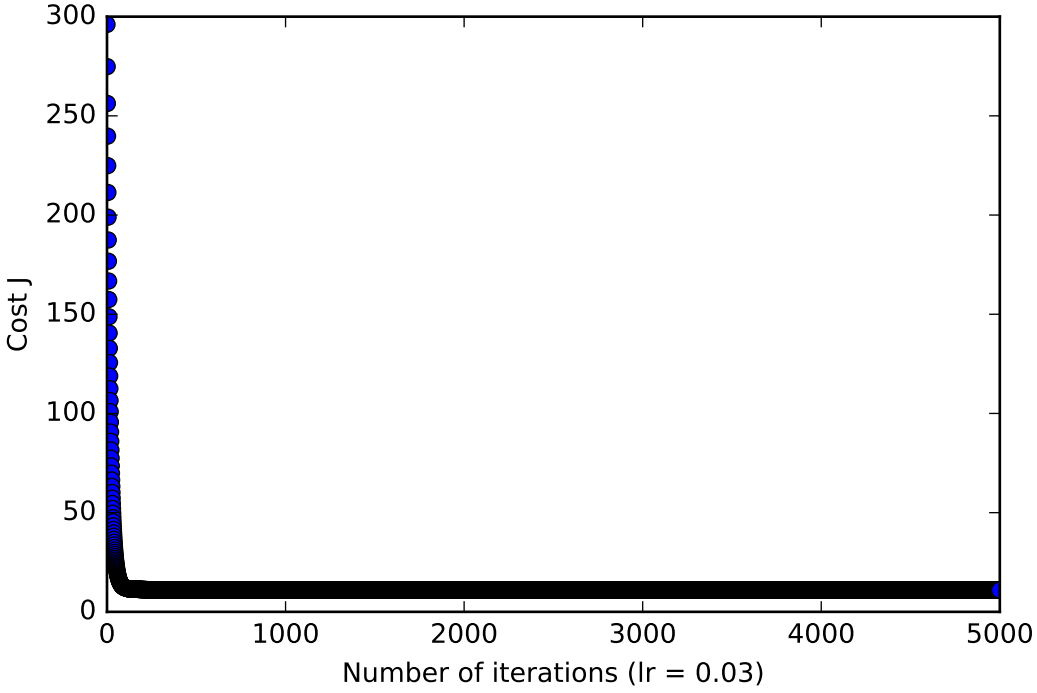


Figure 9: J for different learning rate

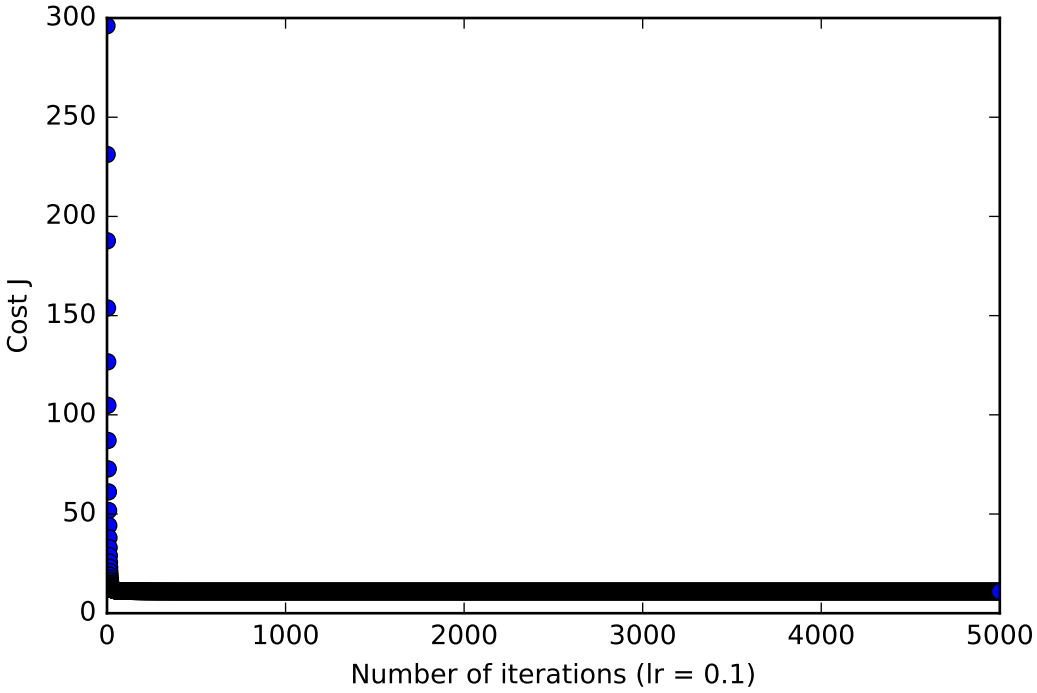


Figure 10: J for different learning rate

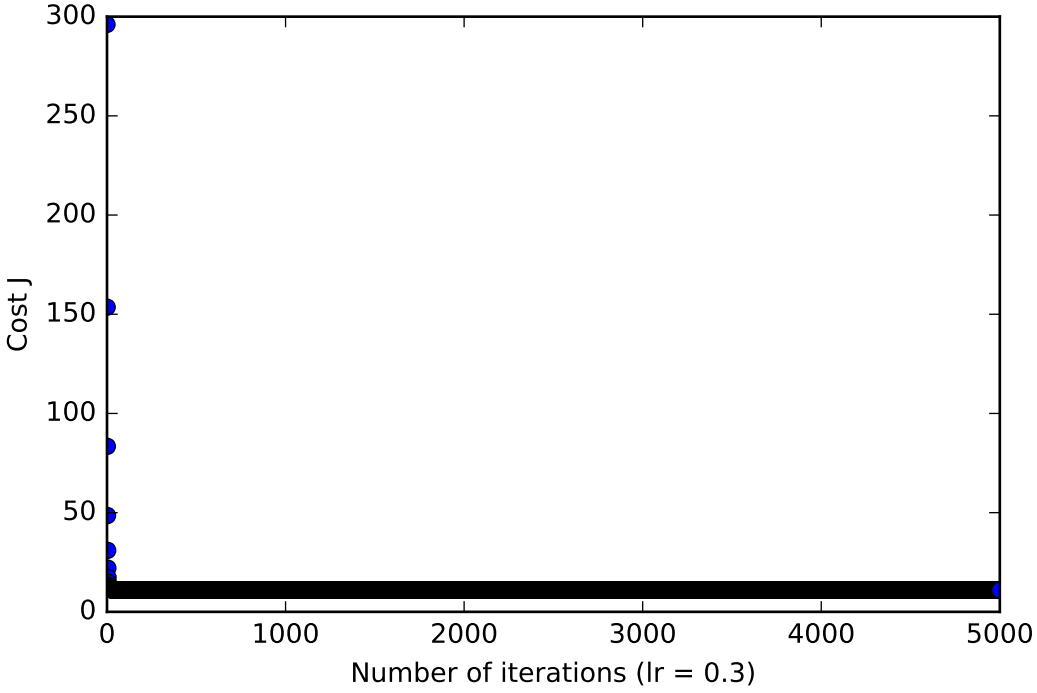
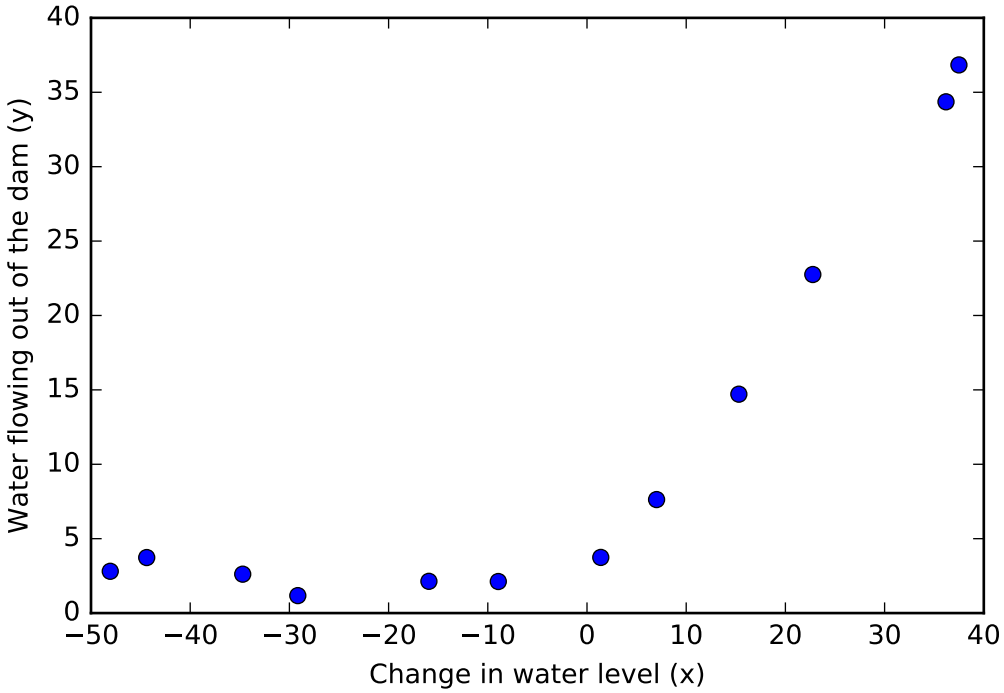


Figure 11: JThe training data



4 Problem 3 part 2 Implementing regularized linear regression

4.1 Problem 3.2 A1 regression cost function

see code

4.2 Problem 3.2 A2 Gradient of the cost function

Learning linear regression mocel Optimization terminated successfully. Current function value: 22.373906 Iterations: 5 Function evaluations: 6 Gradient evaluations: 6 Theta at lambda = 0 is [13.08790353 0.36777923]

4.3 Problem 3.2 A3 Learning curve

4.4 Learning polynomial regression models

4.5 Problem 3.2 A4 Adjusting the regularization parameter

We try 1,10,100 for 1: for 10 for 100

4.6 Problem 3.2 A5 selecting lambda using a validation set

Based on our error curve, We find a clear minimum for validation error between 0.3 and 1. As for the training data the minimum is about at 0.3, so we make a conclusion that 0.3 is the best choice of λ for this problem.

Figure 12: The best fit for the data

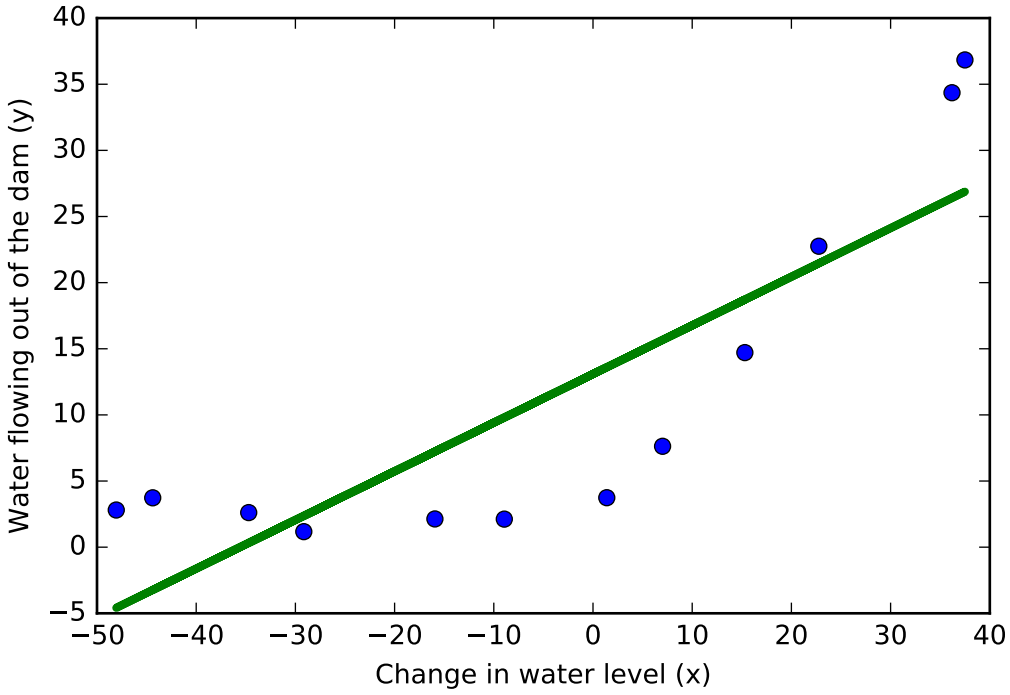


Figure 13: Learning curves

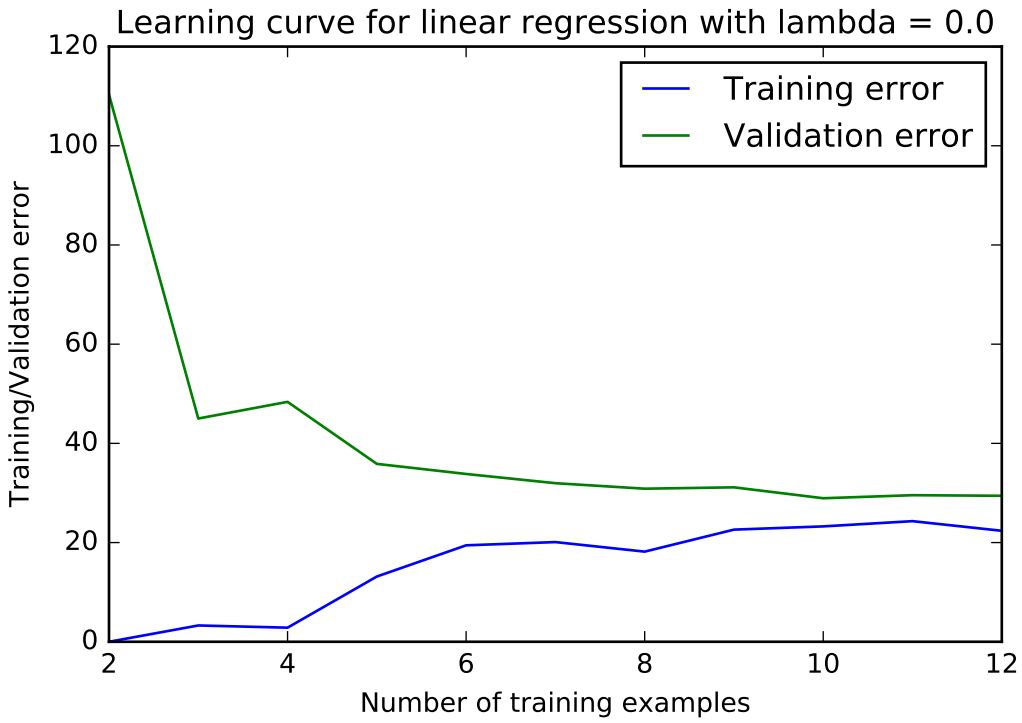


Figure 14: Polynomial fit for reg=0 with p=6

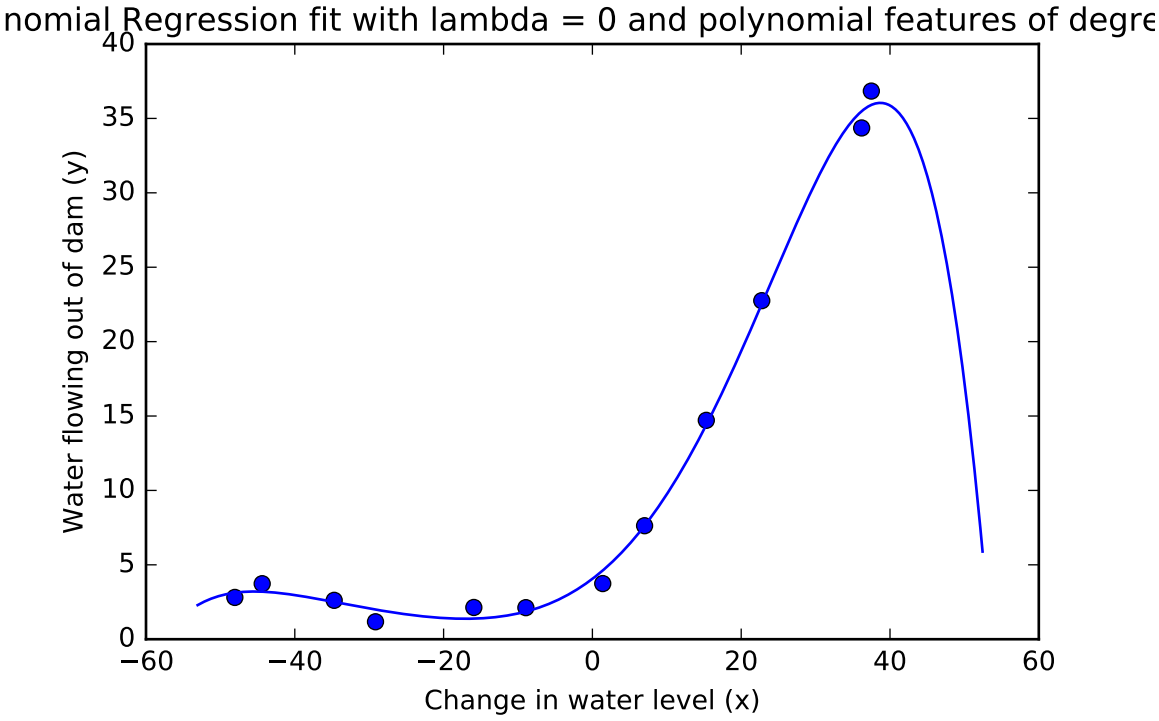


Figure 15: Learning curves for reg=0

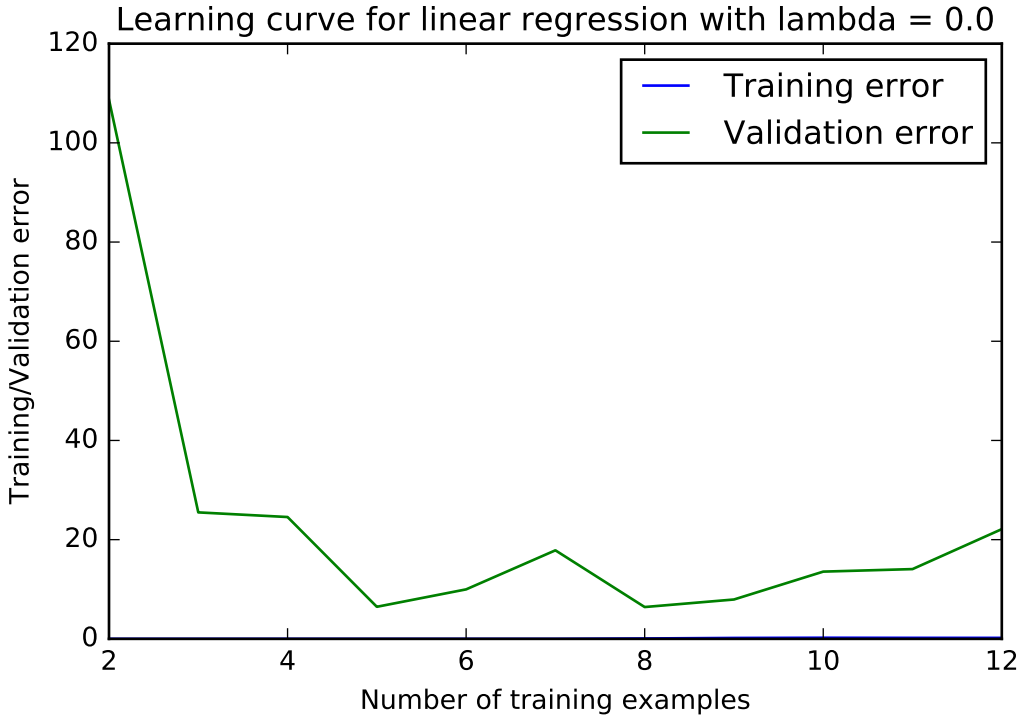


Figure 16: Polynomial fit for reg=1

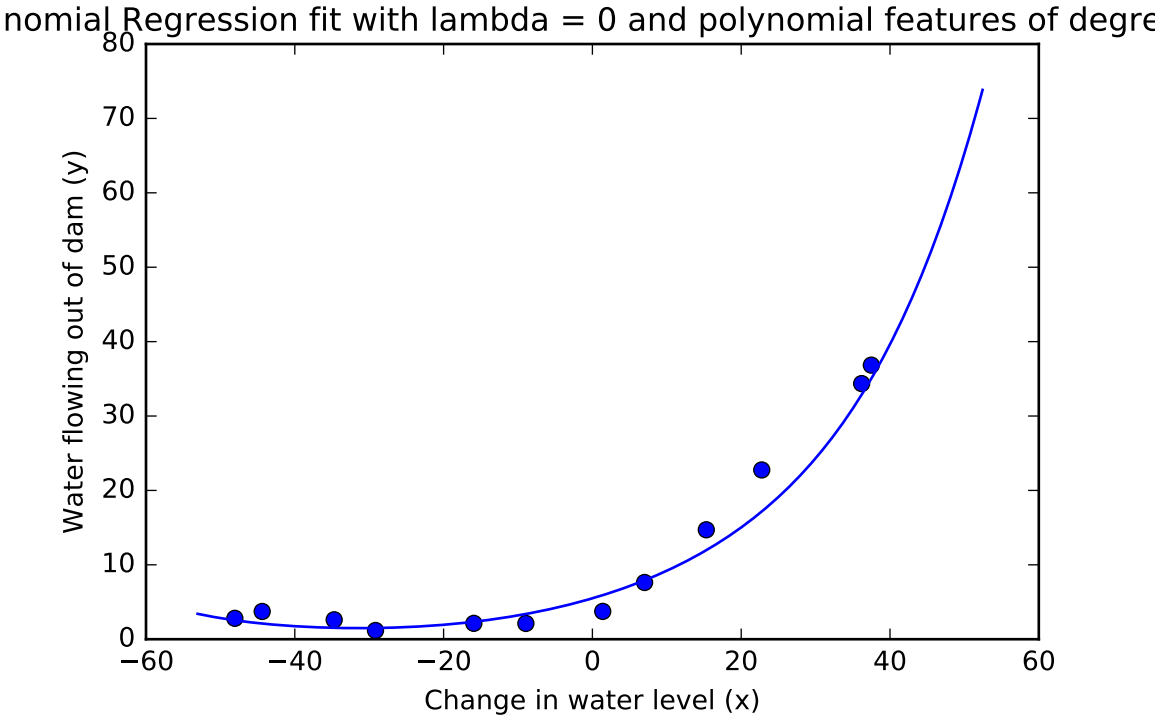


Figure 17: Learning curves for reg=1

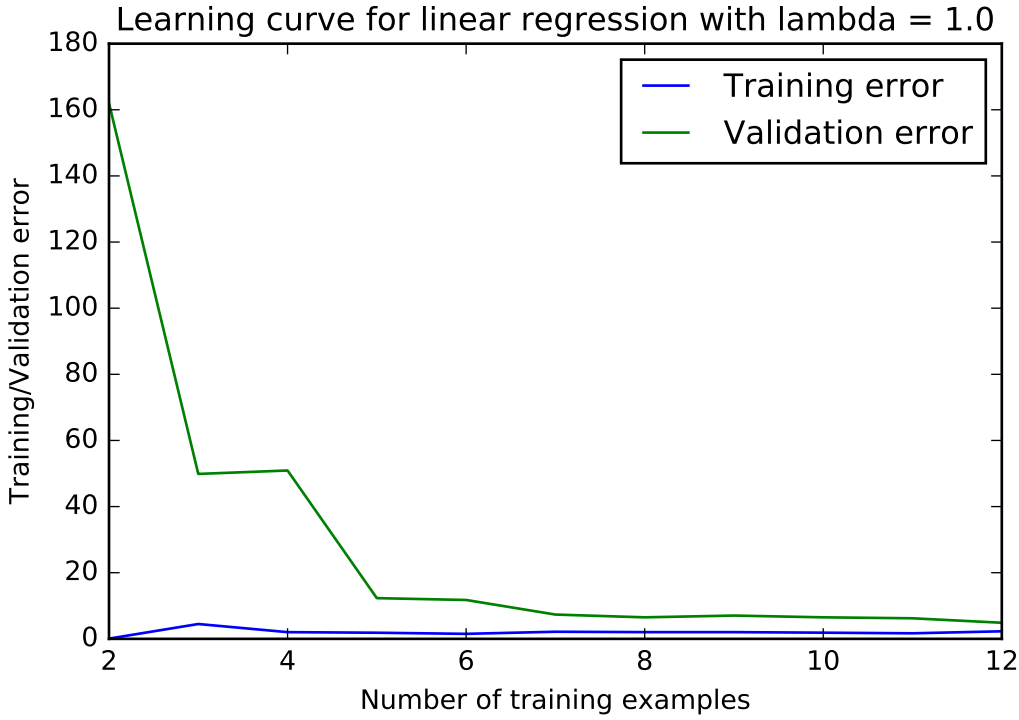


Figure 18: Polynomial fit for reg=10

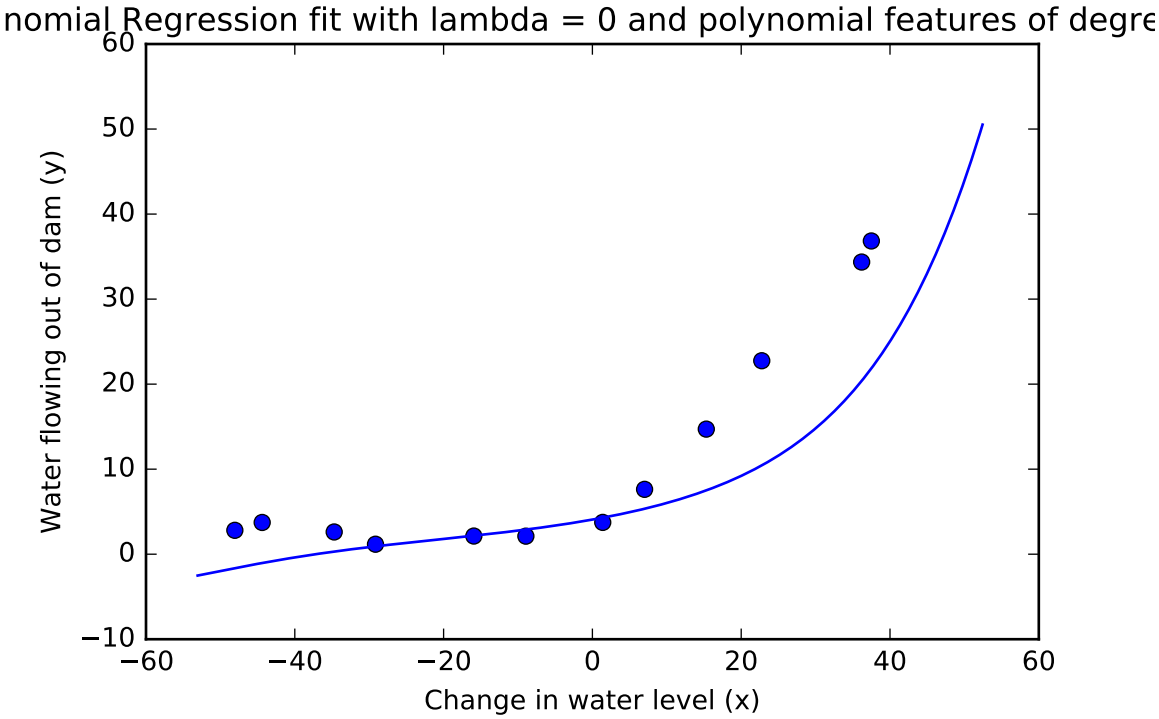


Figure 19: Learning curves for reg=10

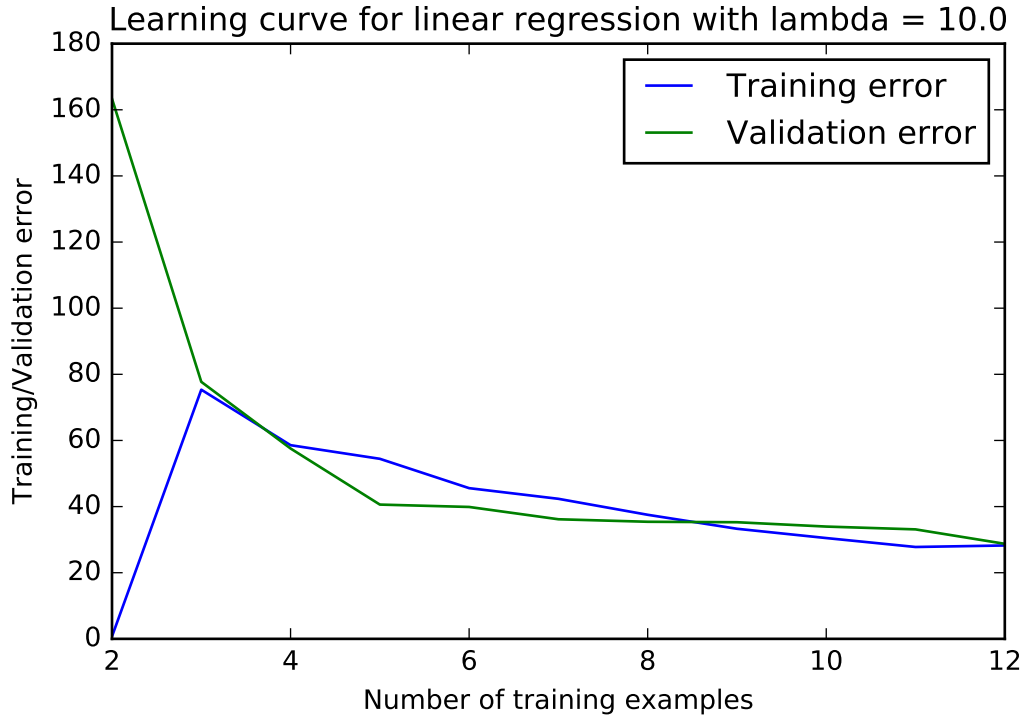


Figure 20: Polynomial fit for reg=100

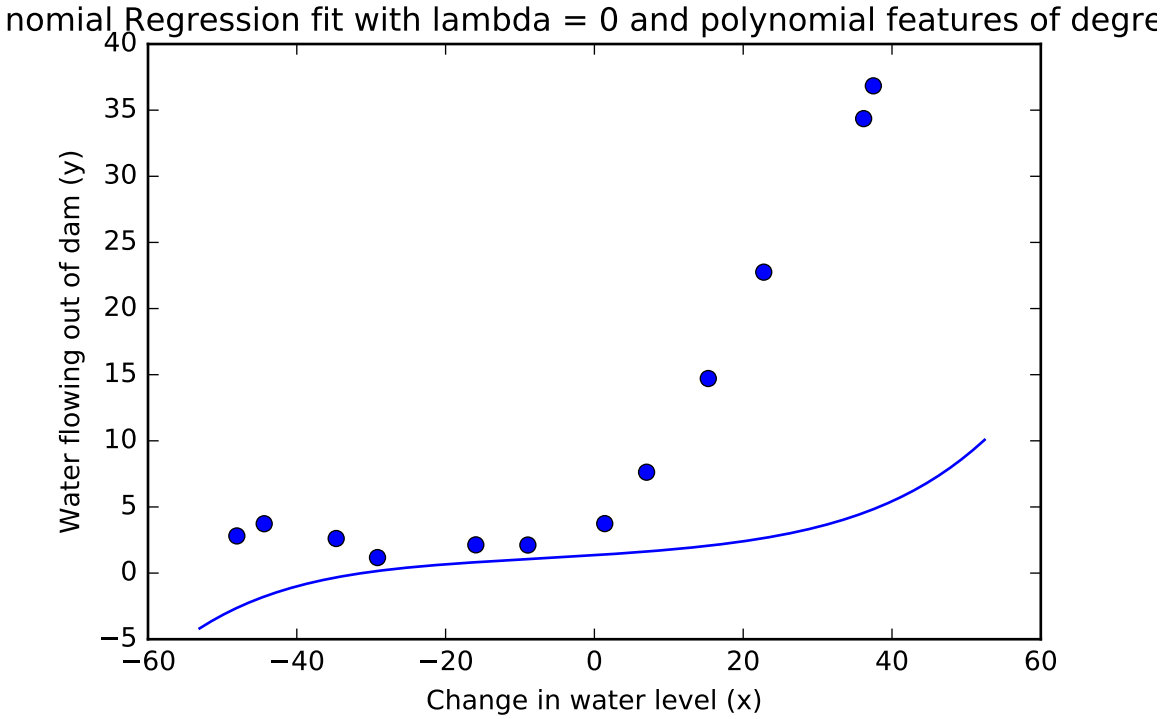


Figure 21: Learning curves for reg=100

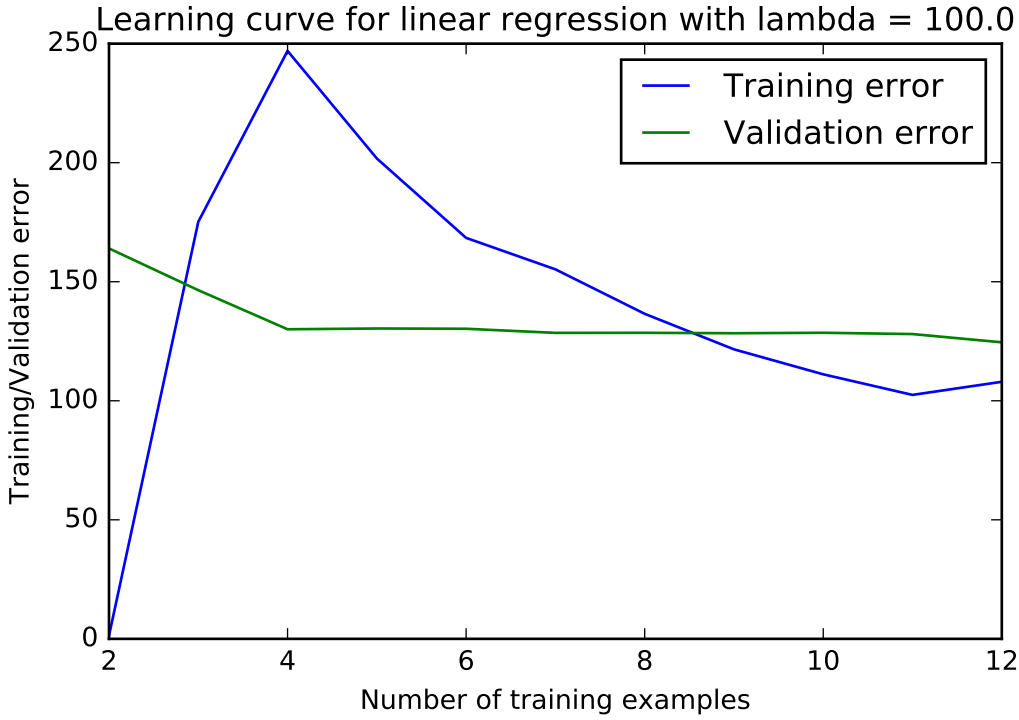
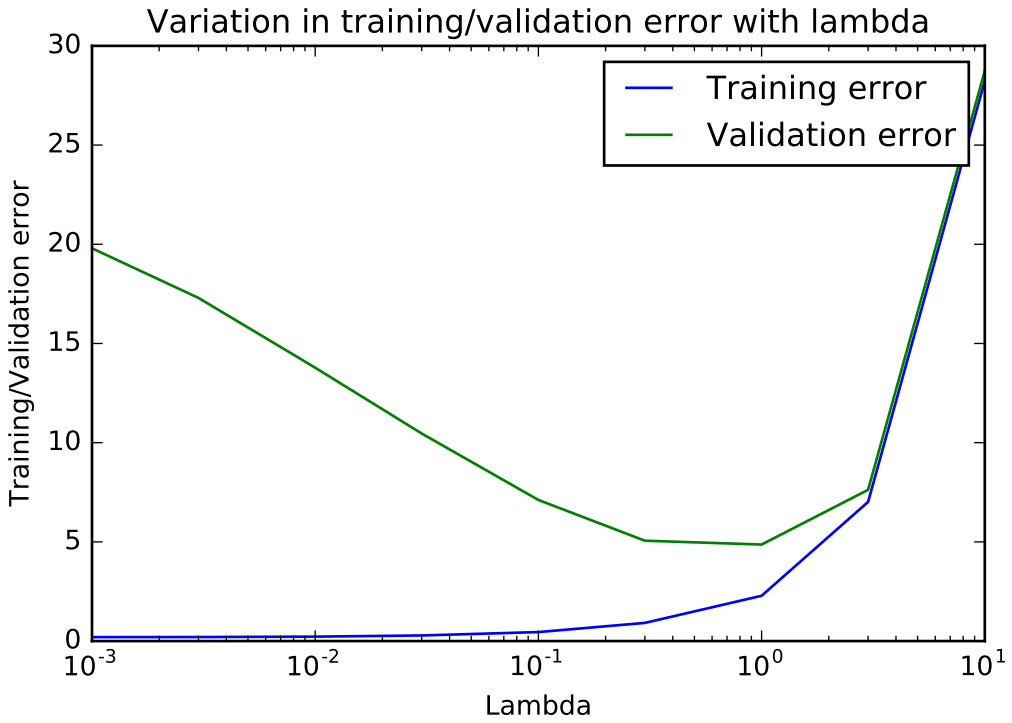


Figure 22: Selecting lambda



4.7 Problem 3.2 A6 Test error with the best model

We find out reg=1 is the best model

4.8 Problem 3.2 A6 Test error with the best model

We plot the test error in the plot. Test error is similar to the training error

4.9 Problem 3.2 A7 Plotting learning curves with randomly examples

Figure 23: Test error with the best model

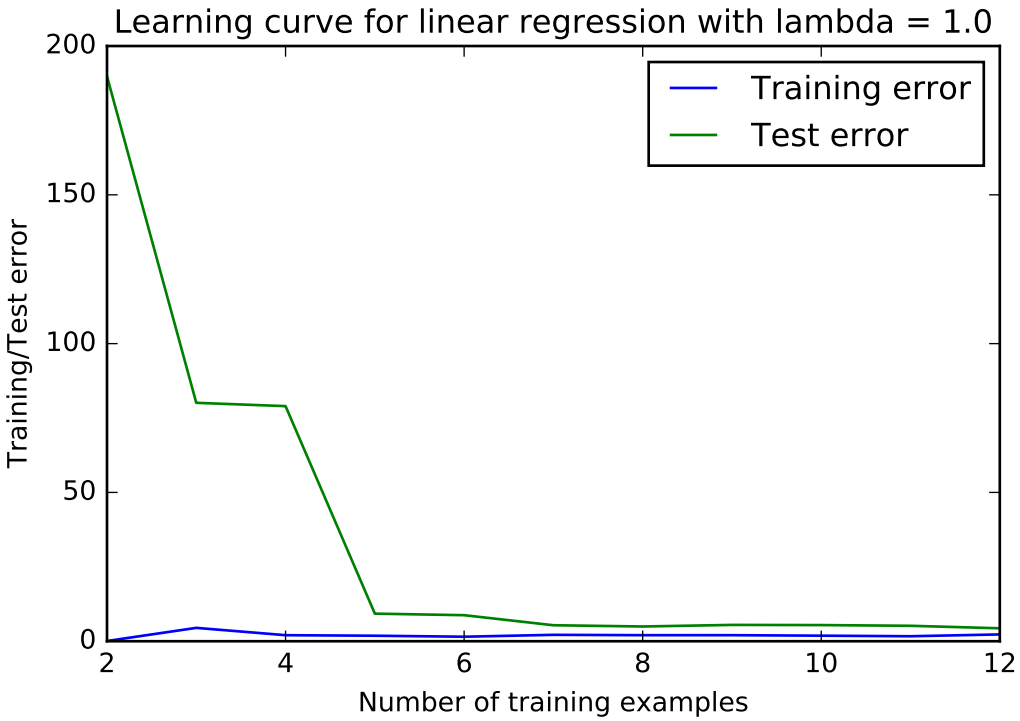


Figure 24: Test error with the best model

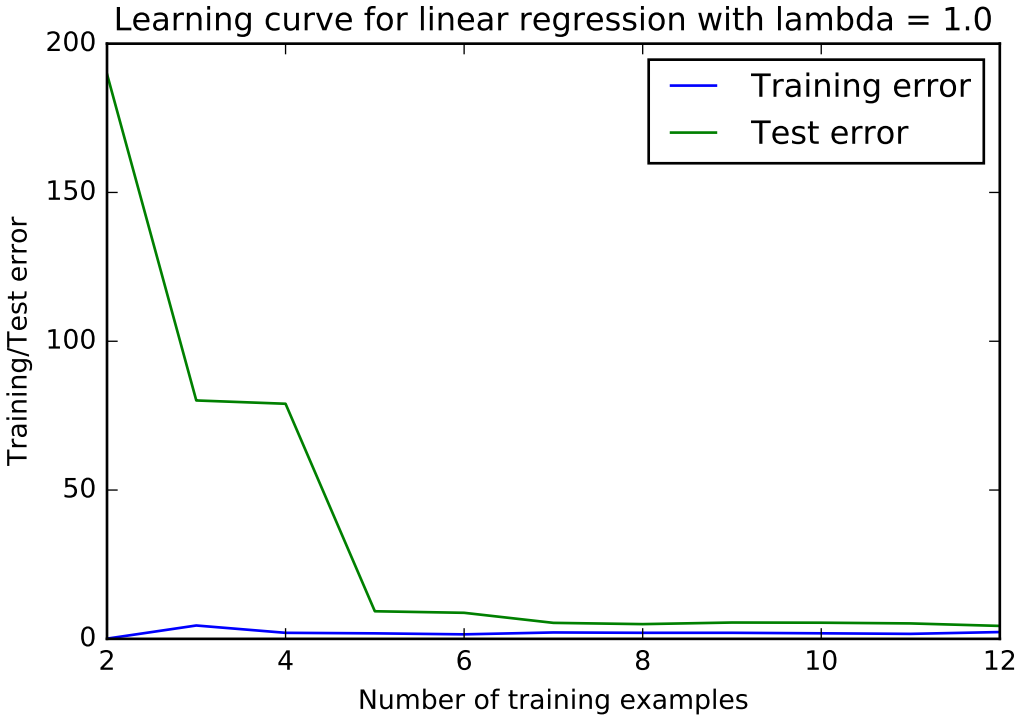


Figure 25: Averaged Learning curve for lambda=1