Solution to Assignment 1

Shoeb Mohammed and Zhuo Chen

January 25, 2016

1 Locally weighted linear regression

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

1.1

Matrix X and vectors θ , y are

$$X=\begin{bmatrix} {x^{(i)}}^T \\ \vdots \\ {x^{(m)}}^T \end{bmatrix},\theta=\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_d \end{bmatrix},y=\begin{bmatrix} y^{(i)} \\ \vdots \\ y^{(m)} \end{bmatrix} \tag{2}$$

Let W be the $m\times m$ diagnoal matrix

$$W=\frac{1}{2}diag(w^{(1)},\cdots,w^{(m)}) \tag{3}$$

Using (2) and (3), equation (1) can be written as

$$J(\theta)=(X\theta-y)^T W(X\theta-y) \tag{4}$$

1.2

The normal equations for un-weighted linear regression are

$$X^T X\theta=X^T y \tag{5}$$

Equation 4 can be re-written as

$$\begin{aligned} J(\theta)&=(X\theta-y)^T\sqrt{W}\sqrt{W}(X\theta-y) \\ &=(\sqrt{W}X\theta-\sqrt{W}y)^T(\sqrt{W}X\theta-\sqrt{W}y) \\ &=(X'\theta-y')^T(X'\theta-y') \text{ where } X'=\sqrt{W}X \text{ and } y'=\sqrt{W}y \end{aligned} \tag{6}$$

Now, equation 6 is similar to unweighted $J(\theta)$. Thus, using equation 5, θ in closed form is

$$\theta=\left[(\sqrt{W}X)^T(\sqrt{W}X)\right]^{-1}\left(\sqrt{W}X\right)^T\sqrt{W}y \tag{7}$$

1.3

Locally weighted linear regression is a non-parametric model. To estimate y, given x

- first calculate the weights $w^{(i)}$. This gives the matrix W as defined in equation 3
- Start with random guess for θ
- In the i$^{\text{th}}$ iteration of the algorithm, the θ is updated using the relation

$$\theta(i)\leftarrow\theta(i-1)-\alpha\left(\sqrt{W}X\right)^T\left(\sqrt{W}X\theta(i-1)-\sqrt{W}y\right) \tag{8}$$

- The above step is repeated until θ converges

2 Properties of the linear regression estimator

2.1

Vectors y,θ,ϵ and matrix X are related as

$$y=X\theta+\epsilon \tag{9}$$

ϵ is i.i.d $N(0,\sigma^2)$.

Thus, for the optimal value θ^* and fixed X

$$\begin{aligned} E[y]&=E[X\theta^*]+E[\epsilon] \\ &=E[X\theta^*]+0 \\ &=X\theta^* \end{aligned} \tag{10}$$

Given the normal equations

$$\begin{aligned} \theta&=(X^T X)^{-1}X^T y \\ \implies E[\theta]&=(X^T X)^{-1}X^T E[y] \\ &=(X^T X)^{-1}X^T X\theta^* \\ &=\theta^* \end{aligned} \tag{11}$$

2.2

$$\begin{aligned} Var(\theta) &= E\left[(\theta - E(\theta))(\theta - E(\theta))^T\right] \\ &= E\left[(\theta - \theta^*)(\theta - \theta^*)^T\right] \\ &= E[\theta\theta^T] - \theta^*\theta^{*T} \end{aligned} \tag{12}$$

Use the normal equations (linear regression estimator) to get

$$\begin{aligned} E[\theta\theta^T] &= E\left[(X^TX)^{-1}X^Tyy^T\left((X^TX)^{-1}\right)^T\right] \\ &= (X^TX)^{-1}X^TE[yy^T]\left((X^TX)^{-1}\right)^T \end{aligned} \tag{13}$$

Use equation 9 to get

$$\begin{aligned} E[yy^T] &= E[(X\theta + \epsilon)(X\theta + \epsilon) \\ &= E[X\theta^*\theta^{*T}X^T + X\theta^*\epsilon^T + \epsilon\theta^{*T}X^T + \epsilon\epsilon^T] \\ &= X\theta^*\theta^{*T}X^T + E[\epsilon\epsilon^T] \\ &= X\theta^*\theta^{*T}X^T + diag(\sigma^2) \\ &= X\theta^*\theta^{*T}X^T + \sigma^2 I \end{aligned} \tag{14}$$

Equations 13 and 14 imply

$$\begin{aligned} E[\theta\theta^T] &= (X^TX)^{-1}X^T\left[X\theta^*\theta^{*T}X^T + \sigma^2 I\right]X\left((X^TX)^{-1}\right)^T \\ &= \left[\theta^*\theta^{*T}X^T + (X^TX)^{-1}X^T\sigma^2 I\right]X\left((X^TX)^{-1}\right)^T \\ &= \theta^*\theta^{*T} + (X^TX)^{-1}X^T\sigma^2 IX\left((X^TX)^{-1}\right)^T \\ &= \theta^*\theta^{*T} + \sigma^2(X^TX)^{-1}X^TX\left((X^TX)^{-1}\right)^T \\ &= \theta^*\theta^{*T} + \sigma^2(X^TX)^{-1} \end{aligned} \tag{15}$$

Equations 12 and 15 imply

$$Var(\theta) = \sigma^2(X^TX)^{-1} \tag{16}$$

3 Problem 3: Part 2: Implementing regularized linear regression

Table 1: Files modified and plots generated for Problem3: Part2

Problem	function implemented	Files edited	Output and Plots
3.2.A1	`RegularizedLinearReg_SquaredLoss.loss`	`reg_linear_regressor_multi.py`	Fig 1 and Fig 2
3.2.A2	`RegularizedLinearReg_SquaredLoss.grad_loss`	`reg_linear_regressor_multi.py`	Fig 1 and Fig 2
3.2.A3	`learning_curve` and `feature_normalize`	`utils.py`	Fig 3, Fig 4 and Fig 5
3.2.A4	-	`ex2.py` added code for λ=1,10,100	Fig 6
3.2.A5	`validation_curve`	`utils.py`	Fig 7
3.2.A6	-	`ex2.py` added code for λ=1 from prob3.2.A5	`Error=3.0987791808`
3.2.A7	`averaged_learning_curve`	`utils.py`	Fig 8

3.1 Problem 3.2.A1 and Problem 3.2.A2:

- Implemented the `RegularizedLinearReg_SquaredLoss.loss` and `RegularizedLinearReg_SquaredLoss.grad_loss` methods in file `reg_linear_regressor_multi.py` per requirements of the problem.

The updated methods use regularized regression to calculate the loss and gradient. The output from `ex2.py` is shown in Fig 1 and Fig 2.

3.2 Problem 3.2.A3

- Implemented the `learning_curve` function in file `utils.py` per the requirements of the problem. The output from `ex2.py` is shown in Fig 3
- Implemented the `feature_normalize` function in file `utils.py`. The output from `ex2.py` is shown in Fig 4 and Fig 5

3.3 Problem 3.2.A4

- The results for λ=1.0 are shown in Fig 6a and Fig 6b.
- The results for λ=10.0 are shown in Fig 6c and Fig 6d.
- The results for λ=100.0 are shown in Fig 6e and Fig 6f.

  We observe that low value of the regularization parameter λ is strongly biased while a large value for λ has high variance.

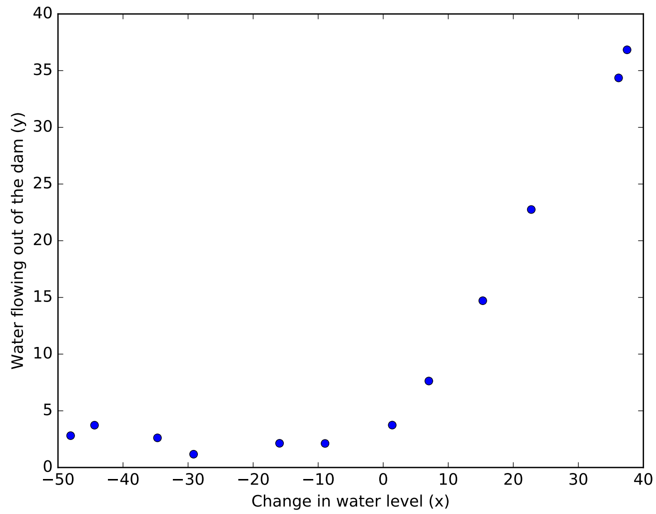


Figure 1: The training data for regularized linear regression.

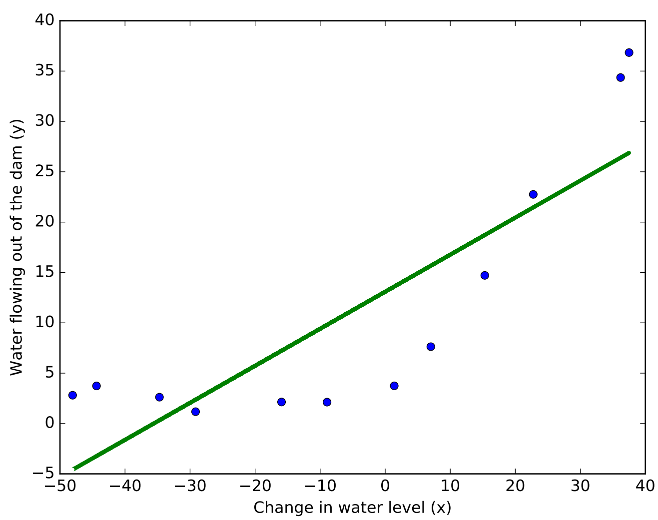


Figure 2: The best fit line for the training data.

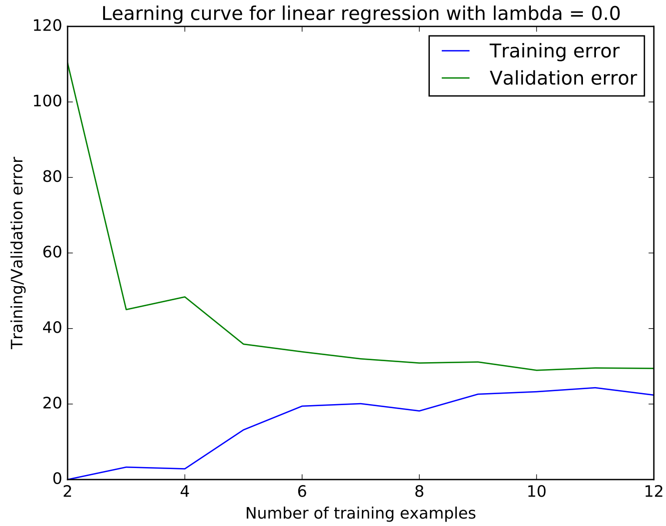


Figure 3: Learning curves.

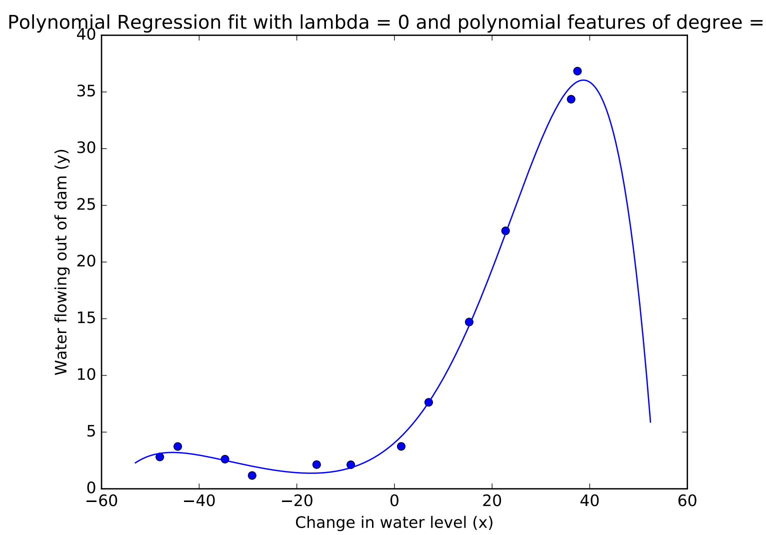


Figure 4: Polynomial fit for lambda = 0 with a p=8 order model.

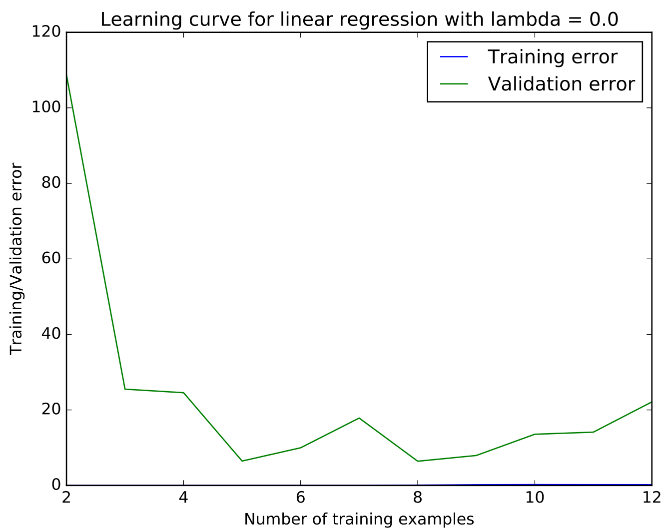
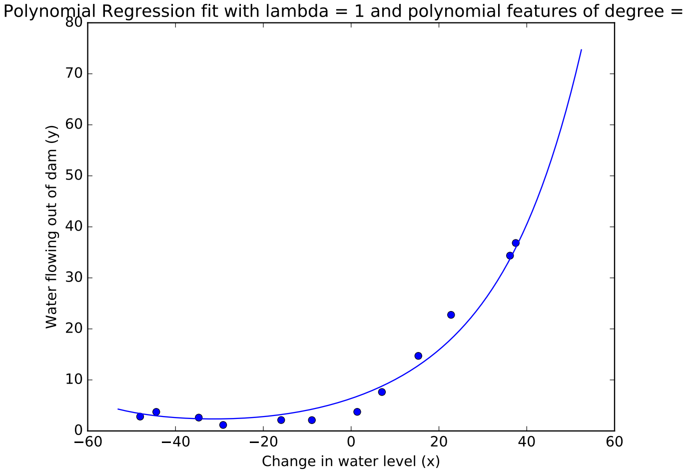
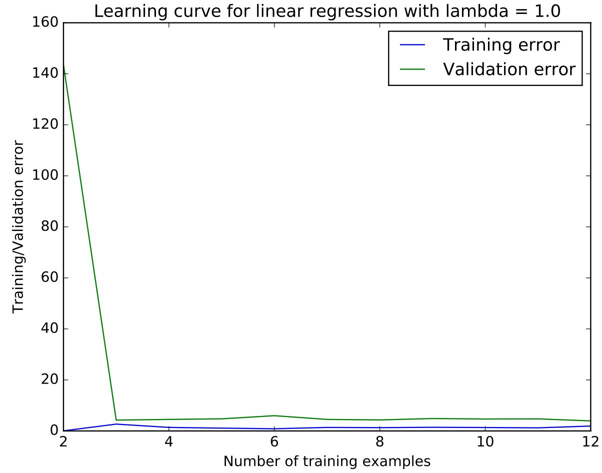


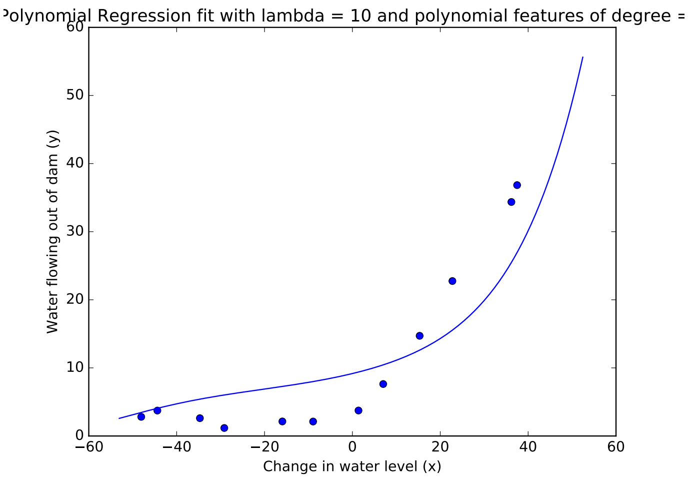
Figure 5: Learning curve for lambda = 0.



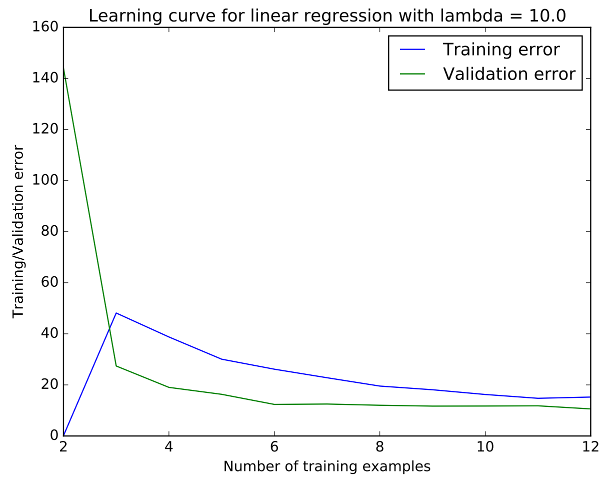
(a) Polynomial fit for lambda = 1 with a p=8 order model.



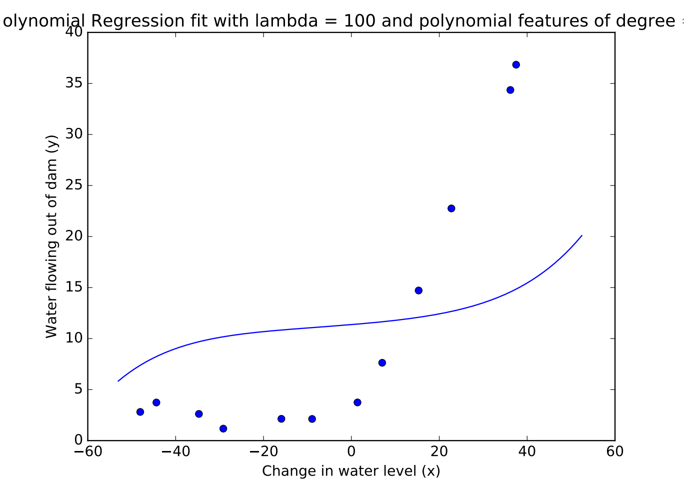
(b) Learning curve for lambda = 1.



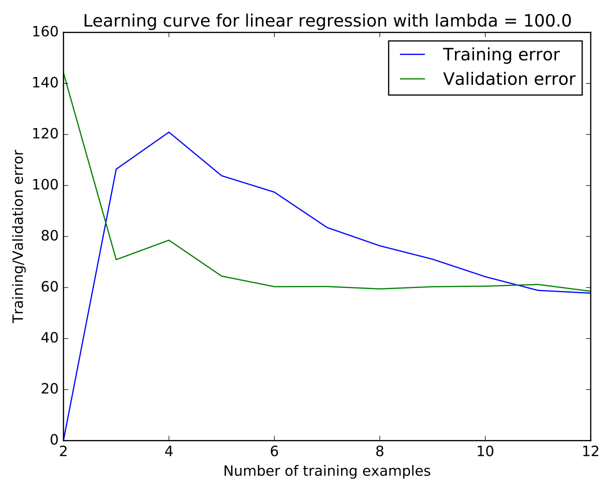
(c) Polynomial fit for lambda = 10 with a p=8 order model.



(d) Learning curve for lambda = 10.



(e) Polynomial fit for lambda = 100 with a p=8 order model.



(f) Learning curve for lambda = 100.

Figure 6: Polynomial fits and learning curves for different lambda

3.4 Problem 3.2.A5

• Completed the `validation_curve` function in the file `utils.py`. The results from `ex2.py` are shown in Fig 7.
Due to randomness in the training and validation splits of the dataset, the cross validation error can sometimes be lower than the training error. We observe that value of $\lambda = 1$ is a good choice for the regularization parameter.

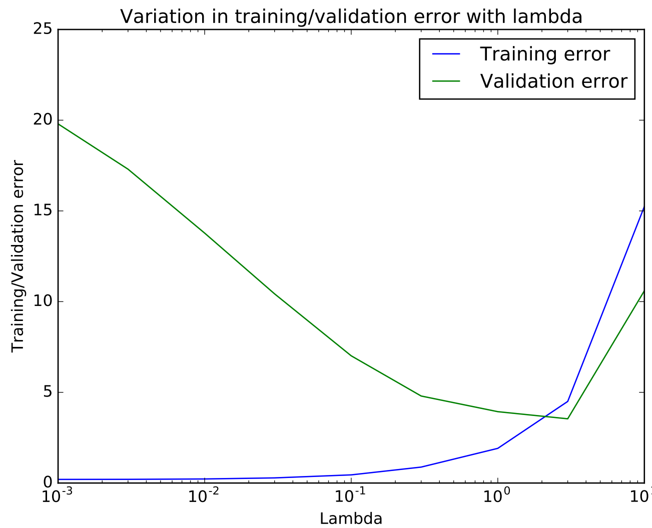


Figure 7: Variation in training/validation error with lambda.

3.5 Problem 3.2.A6

• Per the requirements, added code to `ex2.py`. This uses the best model parameters from section 3.4 and evaluates the error on test data. We picked a value of about $\lambda = 1.0$ for the best model. The output is

```
Theta at lambda = 1.0 (problem3.2.A6) is  [ 11.21759608   8.38067931   5.21899733   3.62613424   2.11030661
   1.95470373   0.78523708]
Error at lambda = 1.0 (problem3.2.A6) is  3.0987791808
```

3.6 Problem 3.2.A7

• Completed the `averaged_learning_curve` function in the file `utils.py` per the requirements of the problem.
The output from `ex2.py` is plotted in Fig 8

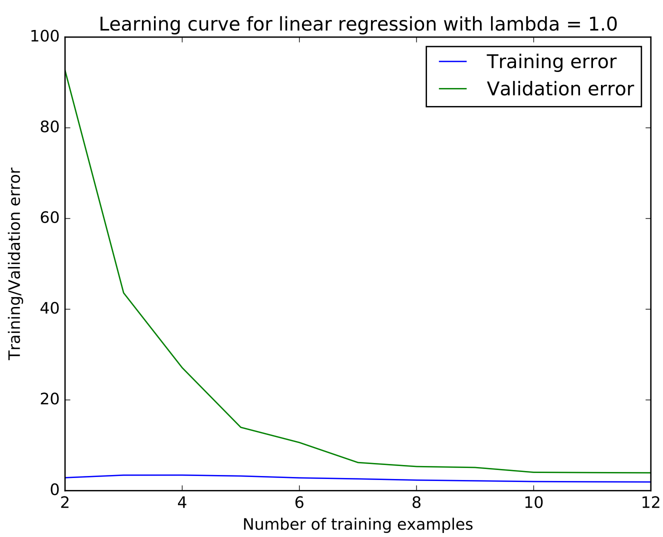


Figure 8: Averaged learning curves for lambda=1.