COMP540 Statistical Machine Learning

Spring 2020

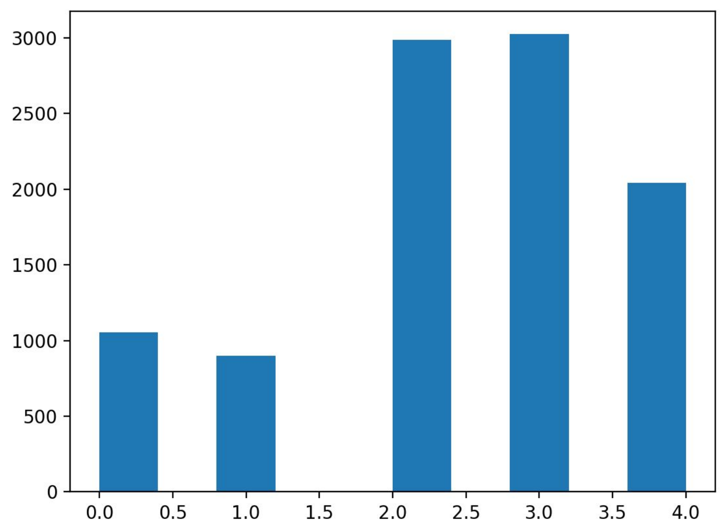
HW1

Yunhao Zheng (yz157)

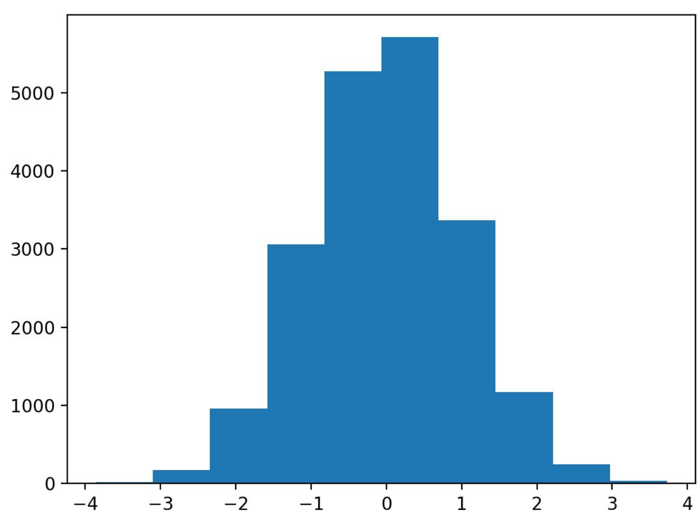
Ziqing Dai (zd15)

0 Background refresher

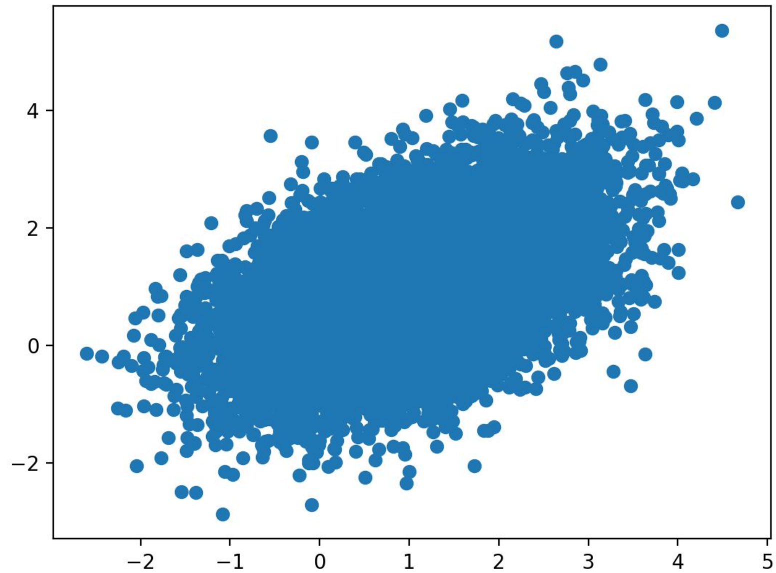
- Plot the histogram of samples generated by a categorical distribution with probabilities [0.1,0.1,0.3,0.3,0.2] (the weights of the different components of work required for this class).



- Plot the univariate normal distribution with mean of 0 and standard deviation of 1.



- Produce a scatter plot of the samples for a 2-D Gaussian with mean at [1,1] and 2a covariance matrix [[1,0.5],[0.5,1]].



- Test your mixture sampling code by writing a function that implements an equal-weighted mixture of four Gaussians in 2 dimensions, centered at (1, 1) and having covariance I. Estimate the probability that a sample from this distribution lies within the unit circle centered at (0.1, 0.2) and include that number in your writeup.

  Result = 0.187

Written problems

0.1

Problem 0.

0.1 Let X, Y be two independent random variables. $Z = X + Y$.

$$p(Z=z) = \sum_{k=0}^{z} p(X=k \,\&\, Y=z-k) = \sum_{k=0}^{z} p(X=k)\cdot p(Y=z-k)$$ (since X, Y are independent).

$$= \sum_{k=0}^{z} \frac{e^{-\lambda_x}\lambda_x^k}{k!} \cdot \frac{e^{-\lambda_y}\lambda_y^{z-k}}{(z-k)!} = \sum_{k=0}^{z} \frac{e^{-(\lambda_x+\lambda_y)}\lambda_x^k \lambda_y^{z-k}}{k!\,(z-k)!}$$

$$= \sum_{k=0}^{z} \binom{z}{k} \cdot \lambda_x^k \lambda_y^{z-k} \cdot \frac{e^{-(\lambda_x+\lambda_y)}}{z!} = (\lambda_x+\lambda_y)^z \frac{e^{-(\lambda_x+\lambda_y)}}{z!}$$ (using binomial distribution coefficient)

$$= \frac{\lambda_z^{z} \cdot e^{-\lambda_z}}{z!}$$ (Let $\lambda_x+\lambda_y = \lambda_z$).

$\Rightarrow$ Z is also Poission random variable.

0.2

0.2. $$p(X_1 = x_1) = \int p(X_1 = x_1 \mid X_0 = x_0) \cdot p(X_0 = x_0)\, dx_0 = a_0 a \int e^{-\frac{(x_0^2 - 2x_0\mu_0 + \mu_0^2)\sigma^2 + (x_1^2 - 2x_1x_0 + x_0^2)\sigma_0^2}{2\sigma_0^2\sigma^2}} dx_0.$$

$$= a_0 a \int e^{\frac{-x_0^2(\sigma^2 + \sigma_0^2) + 2x_0(\mu_0\sigma^2 + x_1\sigma_0^2) - (\mu_0^2\sigma^2 + x_1^2\sigma_0^2)}{2\sigma_0^2\sigma^2}} dx_0.$$

$$= a_0 a \sqrt{\frac{2\pi\sigma_0^2\sigma^2}{\sigma^2+\sigma_0^2}} \cdot e^{\frac{1}{2\sigma_0^2\sigma^2}\left[\frac{(\mu_0\sigma^2 + x_1\sigma_0^2)^2}{\sigma^2+\sigma_0^2} - (\mu_0^2\sigma^2 + x_1^2\sigma_0^2)\right]} \cancel{dx_0}$$ (Gaussian Integrials)

$$= a_0 a \sqrt{\frac{2\pi\sigma_0^2\sigma^2}{\sigma^2+\sigma_0^2}} \cdot e^{-\frac{(x_1-\mu_0)^2}{2(\sigma^2+\sigma_0^2)}} dx_0 = a_1 e^{-\frac{(x_1-\mu_1)^2}{2\sigma_1^2}}$$

$$\Rightarrow a_1 = a_0 a \sqrt{\frac{2\pi\sigma_0^2\sigma^2}{\sigma^2+\sigma_0^2}}, \quad \mu_1 = \mu_0, \quad \sigma_1 = \sqrt{\sigma^2+\sigma_0^2}$$

0.3

0.3. $$p(A \mid B, C) > p(A \mid B)$$

$$\frac{p(A,B,C)}{p(B,C)} > \frac{p(A,B)}{p(B)}$$

$$\frac{p(C \mid A,B) \cdot p(A,B)}{p(B) \cdot p(C \mid B)} > \frac{p(A,B)}{p(B)}$$

$$\Rightarrow p(C \mid A,B) > p(C \mid B)$$

$$p(C^c \mid A,B) < p(C^c \mid B) \quad ①$$

$$① \Rightarrow \frac{p(C^c \mid A,B) \cdot p(A,B)}{p(C^c \mid B) \cdot p(B)} < \frac{p(A,B)}{p(B)}$$

$$\frac{p(A,B,C^c)}{p(B,C^c)} < p(A \mid B)$$

$$p(A \mid B, C^c) < p(A \mid B)$$

Hence proved.

0.4-0.7

0.4. $M = uv^T = \begin{bmatrix} 2 & 3 \\ 4 & 6 \end{bmatrix}$

$|M - \lambda E| = \begin{bmatrix} 2-\lambda & 3 \\ 4 & 6-\lambda \end{bmatrix} = \lambda(\lambda - 8) = 0.$

Eigenvalues $\lambda_1 = 0$. $\lambda_2 = 8$.

$\lambda_1 = 0$. eigenvector $v_1 = \begin{bmatrix} -3 \\ 2 \end{bmatrix}$

$\lambda_2 = 8$. eigenvector $v_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

0.5. Let λ be an eigenvalue of A. v be an eigenvector of A.

$v^T A v \cancel{\geq 0} = \lambda v^T v \geq 0.$

$\cancel{v^T A v}$

Since $v^T v = \sum_i v_i^2 \geq 0$. $\Rightarrow \lambda \geq 0$. therefore, eigenvectors of A are non-negative

0.6. a. Let $A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$. $B = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$.

$(A+B)^2 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$. $A^2 + 2AB + B^2 = \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix}$ $\Rightarrow (A+B)^2 \neq A^2 + 2AB + B^2$

b. Let $A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \neq 0$, $B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \neq 0$.

$AB = 0$

0.7. $A^T = I - (2uu^T)^T = I - 2uu^T$.

$A^T A = (I - 2uu^T)^2 = I - 4Iuu^T + 4uu^T uu^T$, ~~since $u^Tu = 1$~~

$= I - 4uu^T + 4uu^T = I$, since $u^T u = 1$

0.8

0.8. a. $f''(x) = 6x \geq 0$. $x \geq 0$.

$\Rightarrow f(x) = x^3$ is convex for $x \geq 0$.

b. $f(\lambda x + (1-\lambda)y) = \max(\lambda x_1 + (1-\lambda)y_1, \lambda x_2 + (1-\lambda)y_1, \lambda x_1 + (1-\lambda)y_2, \lambda x_2 + (1-\lambda)y_2)$ ①

$\lambda f(x) + (1-\lambda)f(y) = \lambda \cdot \max(x_1, x_2) + (1-\lambda)\max(y_1, y_2)$ ②

$\lambda > 0$ & $1-\lambda > 0 \Rightarrow$ ② $\geq$ ①.

~~Therefore, $f(x) \geq x$~~ Hence proved.

c. $f \cdot g$ are convex $\Rightarrow$ $\cancel{f''(x) + g''(x)}$ $f''(x) \geq 0$. $g''(x) \geq 0$. $x \in S$.

$\Rightarrow f''(x) + g''(x) \geq 0$. $x \in S$.

$f+g$ is convex on S.

d. $(fg)'' = f''g + 2f'g' + fg''$.

$f \cdot g$ ≥ 0 ~~are non-n~~ $f'' \cdot g'' \geq 0 \Rightarrow f''g + fg'' \geq 0$.

Let x_0 be the minimum point for $f \cdot g$ on S.

$$\begin{cases} \text{when } x \leq x_0.\ f' \leq 0.\ g' < 0. \Rightarrow f'g' \geq 0. \\ \text{when } x > x_0.\ f' \geq 0.\ g' > 0 \Rightarrow f'g' > 0. \\ \text{when } x = x_0.\ f'g' = 0. \end{cases} \Rightarrow f'g' \geq 0 \text{ on } S.$$

$\Rightarrow (fg)'' \geq 0$ on S. fg is convex on S.

0.9

0.9. $\overset{H}{\cancel{f}}(p) = -\sum_{i=1}^{K} p_i \log p_i$, subject to $g(p) = \sum_{i=1}^{K} p_i - 1 = 0$.

$\mathcal{L}(p, \lambda) = f(p) + \lambda g(p) = -\sum_{i=1}^{K} p_i \log p_i + \lambda(p_i - 1)$

$$\frac{\partial \mathcal{L}(p_i, \lambda)}{\partial p_i} = -\log p_i - 1 + \lambda = 0 . \Rightarrow p_i = e^{\lambda - 1}$$

$\Rightarrow$ All p_i are equal and depend on λ only.

Since $\sum_{i=1}^{K} p_i = 1$ $\Rightarrow$ when $p_i = \frac{1}{K}$, the distribution has the highest entropy.

i.e. ~~th~~uniform distribution

1 Locally weighted linear regression

1. Locally weighted linear regression

1.1 $J(\theta) = \frac{1}{2} \sum_{i=1}^{m} w^{(i)} (\theta^T x^{(i)} - y^{(i)})^2$

according to the question,

X is a $m \times d$ input matrix

y is a $m \times 1$ vector denoting the associated outputs

$$J(\theta) = (X\theta - y)^T W (X\theta - y)$$

Let W equal to a diagnal matrix

which is $diag(w_1, w_2, \dots, w_m)$

let $X\theta - y = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_m \end{bmatrix}$

$$\therefore J(\theta) = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_m \end{bmatrix} \cdot diag(w_1, w_2 \dots w_m) \cdot [n_1, n_2 \dots n_m]$$

$$= n_1^2 w_1 + n_2^2 w_2 + \dots + n_m^2 w_m$$

$$= \sum_{h=1}^{m} n_i^2 w_i$$

$$n_i = \sum_{j=1}^{m} x_{im} \theta_m - y_i$$

$$= \theta^T x^{(i)} - y_i$$

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

$2w_i = w^{(i)}$ $\therefore$ $W = diag(\frac{w^{(1)}}{2}, \dots, \frac{w^{(m)}}{2})$

1.2 according to the question if all $\omega^{(i)}$ are equal to 1, $X^T X \theta = X^T y$

the value of θ that mininizes $J(\theta)$ is $(X^T X)^{-1} X^T y$.

the generalized form is got from:

$J(\theta) = (X\theta - y)^T W (X\theta - y)$

$$= (X\theta)^T W X\theta - (X\theta)^T W y - y^T W X\theta + y^T W y.$$

$$\frac{\partial J(\theta)}{\partial \theta} = X^T \theta^T W X - y^T W X = 0$$

$\therefore X^T \theta^T W X = y^T W X$

$\therefore X^T \theta^T W X = y^T W X$

$\therefore X \theta W^T X^T = y W^T X^T$

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

1.3 $\omega^{(i)} = \exp\left(-\frac{(x - x^{(i)})^T (x - x^{(i)})}{2\tau^2}\right)$

loss function of BGD is $\frac{\partial J(\theta)}{\partial \theta_j} = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j$

$\Rightarrow \theta_j' = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} = \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$

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

$$\cancel{\frac{\partial J(\theta)}{\partial \theta_j}} = \quad \theta_j \leftarrow \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} w^{(i)} (\theta^T x^{(i)} - y^{(i)}) x_j^{(i)}$$

It's a non-parametric method

2 Properties of the linear regression estimator

Problem 2.

2.1 $\begin{cases} \theta = (X^T X)^{-1} X^T y \\ y = X\theta^* + \varepsilon \end{cases}$ $\Rightarrow E(\theta) = E[(X^T X)^{-1} X^T (X\theta^* + \varepsilon)]$

$= E(\theta^*) + (X^T X)^{-1} X^T E(\varepsilon) = E(\theta^*)$ ∵ Since $E(\varepsilon) = 0$.

2.2 $Var(\theta) = Var[(X^T X)^{-1} X^T y]$.

$= (X^T X)^{-1} X^T \, var(y) [(X^T X)^{-1} X^T]^T$

$= (X^T X)^{-1} X^T X (X^T X)^{-1} \sigma^2$

$= (X^T X^{-1}) \sigma^2$

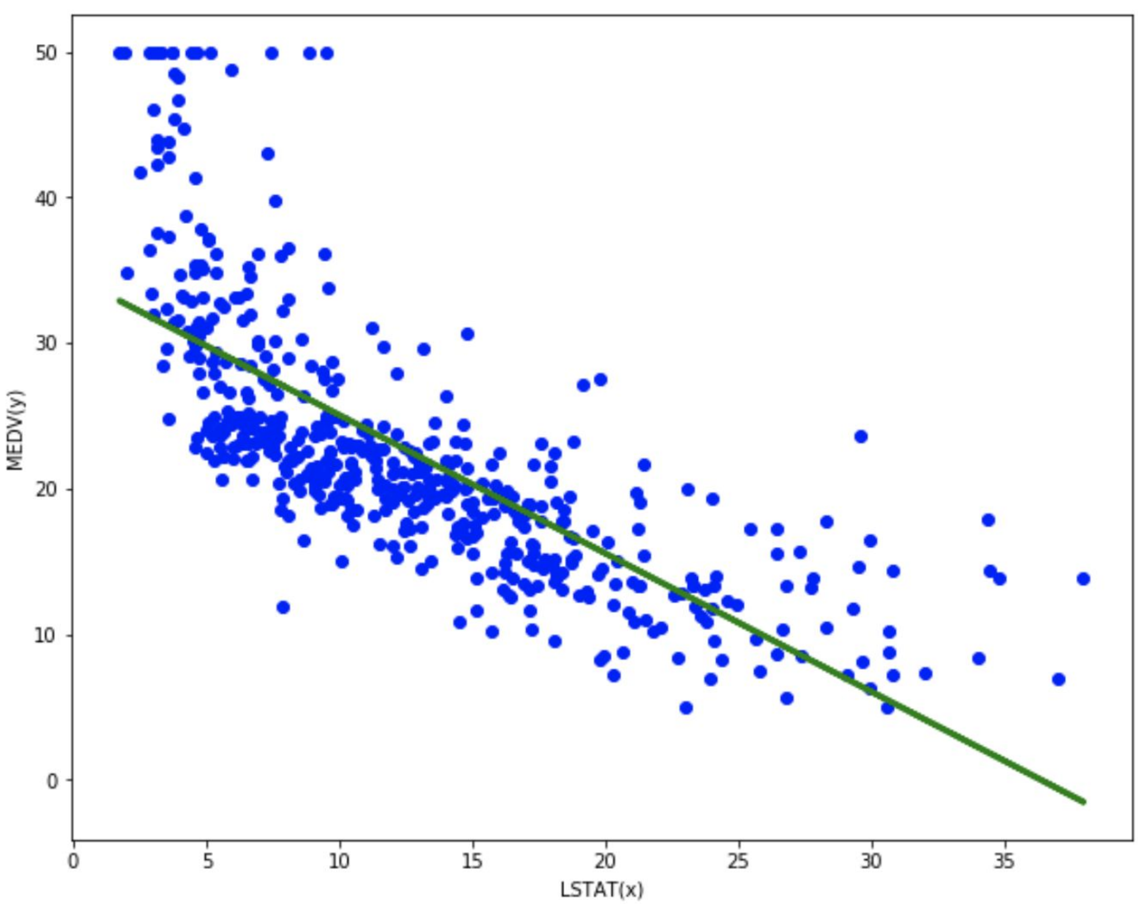
Problem 3: Part 1: Implementing regularized linear regression

Problem 3.1.A1:Computing the cost function J

difference = np.dot(X, self.theta) - y
J = np.dot(difference.T, difference) / (2 * X.shape[0])

Problem 3.2 A2:Implementing gradient descent

What can you say about the quality of the linear fit for this data? In your assignment writeup.pdf, explain how you expect the model to perform at the low and high ends of values for LSTAT? How could we improve the quality of the fit?
The figures of the linear fit this data and the cost function vs number of iterations are shown as below.



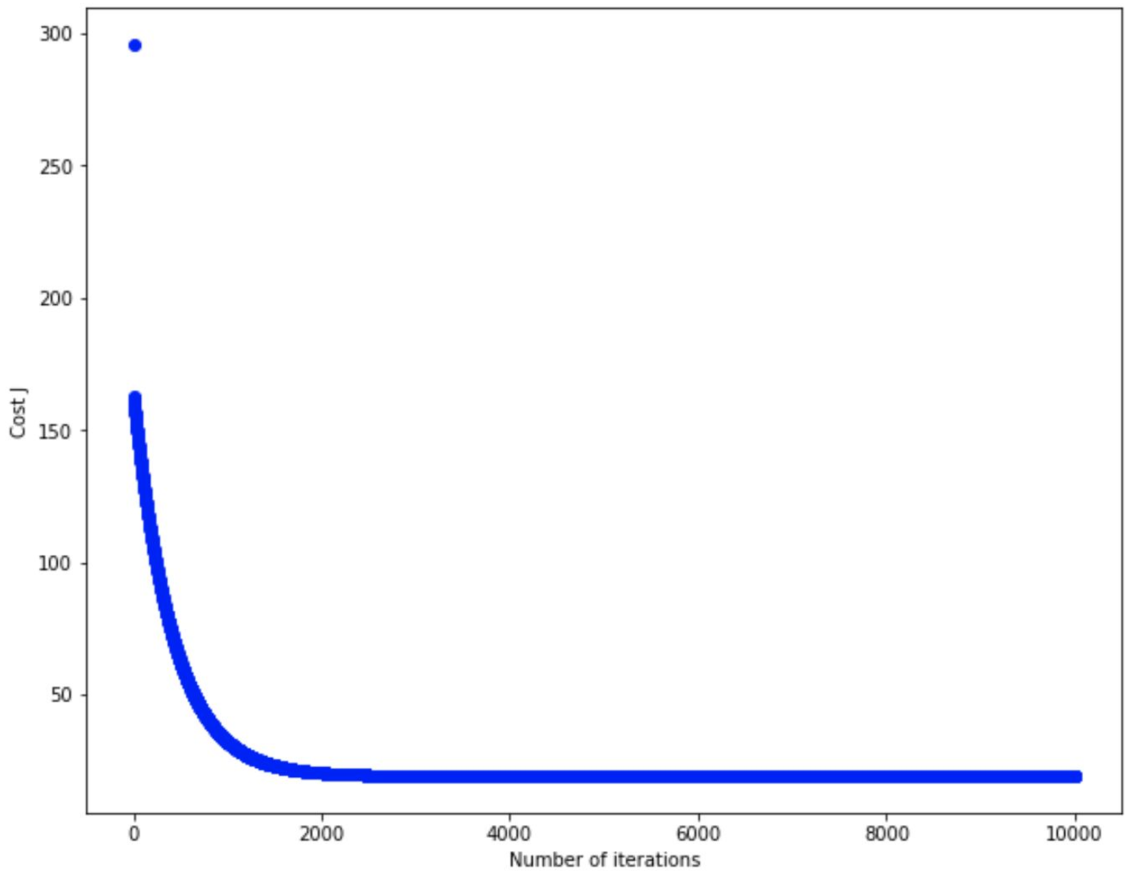


Figure 1: fit line and cost J

From the figure we can see that the low ends of the data don't fit so good while the high ends fit better. So the quality still can be improved. For improving the quality of the model, we need to increase the dimension of x.

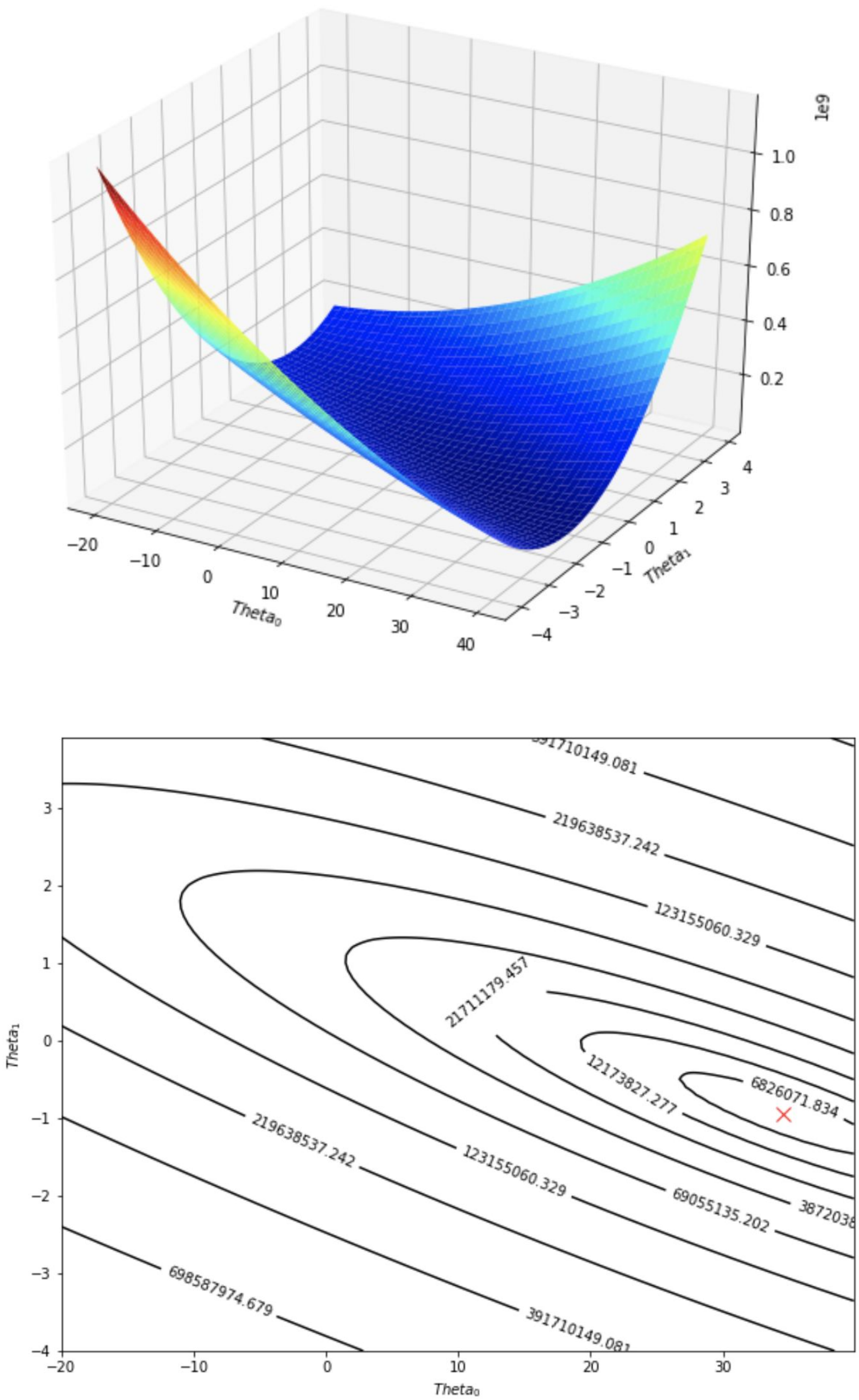


Figure 2: surface of J and contour

Problem 3.2 A3:

When the percentage of population of lower economic status is 5%:
pred_cost = linear_reg.predict(np.array([1,5]))*10000
The result is For lower status percentage = 5, we predict a median home value of 298034.49
When the percentage of population of lower economic status is 50%:
pred_cost = linear_reg.predict(np.array([1,50]))*10000

The result is: For lower status percentage = 50, we predict a median home value of -129482.13

Problem 3.1 B1:Feature normalization

The answer is in the function: feature normalize() in utils.py

Problem 3.1 B2:Loss function and gradient descent

The answer is in the file:linear_regressor_multi.ipynb

Problem 3.1 B3:Making predictions on unseen data

The answer is in the file:ex1 multi.ipynb
For average home in Boston suburbs, we predict a median home value of 225328.06

Problem 3.1 B4:Normal equations

For average home in Boston suburbs, we predict a median home value of 225328.06

Problem 3.1 B5:

In this problem, I choose learning rate as 0.01, 0.03, 0.1, 0.3 respectively.
When learning rate=0.01, the figure of costJ vs Numbers of iterations is shown as below.

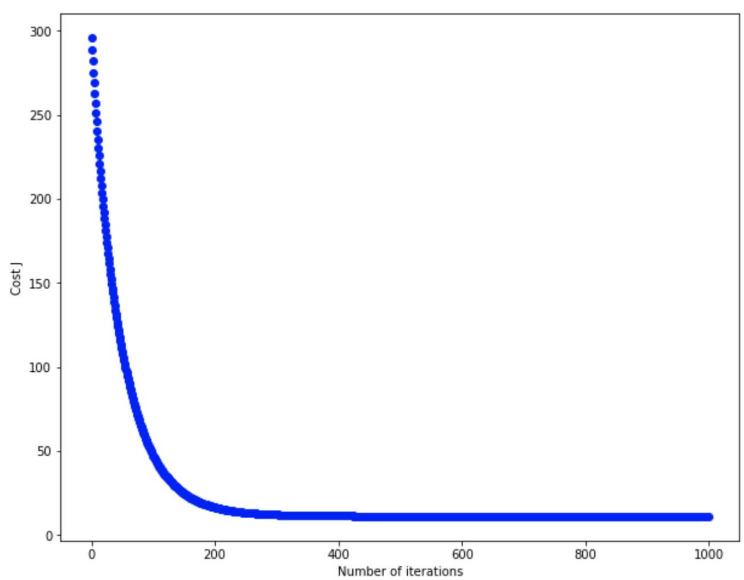


Figure 3:cost J (When learning rate=0.01)

The loss function converges at 11.109868.
When learning rate=0.03, the figure of costJ vs Numbers of iterations is shown as below.

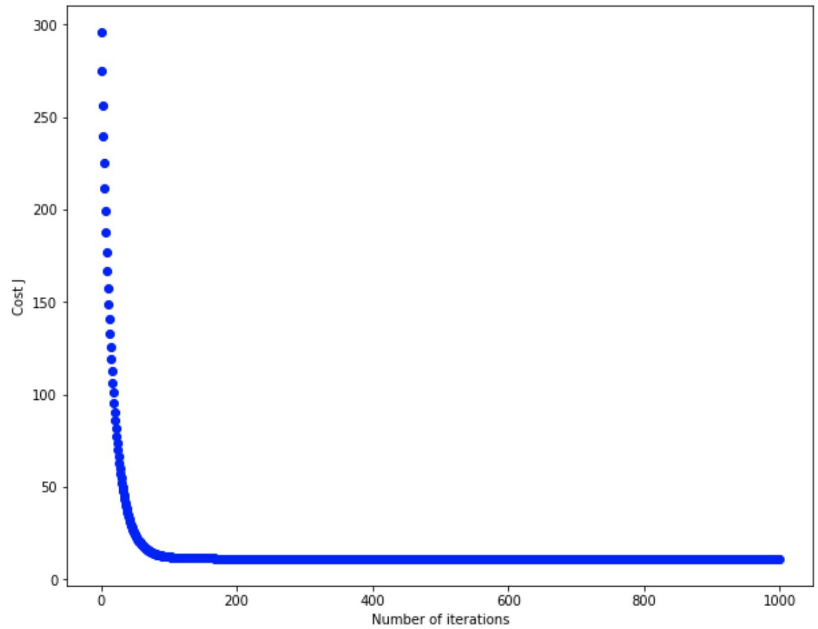


Figure 4:cost J (When learning rate=0.03)

The loss function converges at 10.959215
When learning rate=0.1, the figure of costJ vs Numbers of iterations is shown as below.

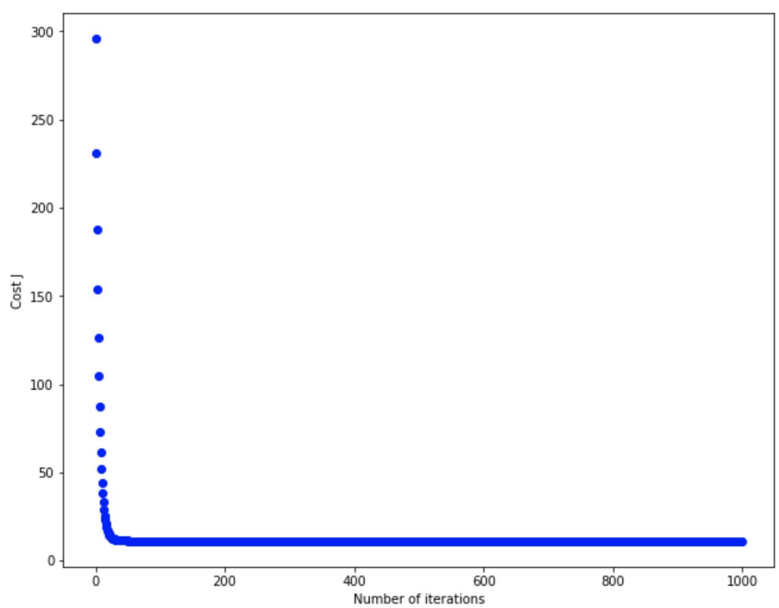


Figure 5:cost J (When learning rate=0.1)

The loss function converges at 10.947419

When learning rate=0.3, the figure of costJ vs Numbers of iterations is shown as below.

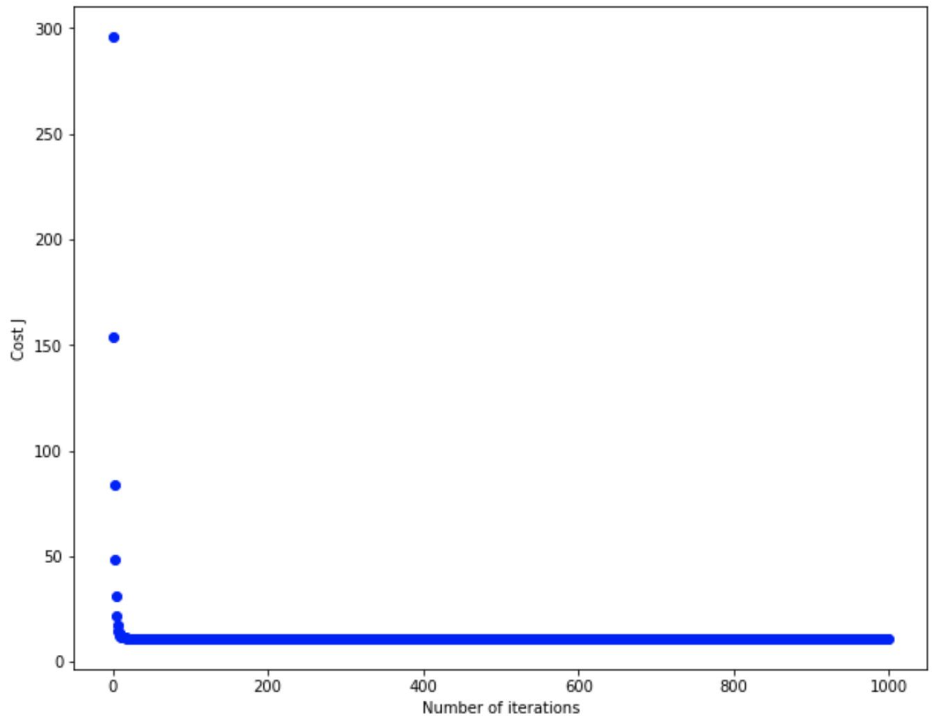


Figure 6:cost J (When learning rate=0.3)

The loss function converges at 10.947416.

From the results, it is easy to get that among the four learning rates we have chosen, When learning rate become larger, the cost function converges quicker and converges at a relatively lower value. When learning rate =0.3, the loss function converges at the smallest value which is 10.947416. It needs around 100 iterations to converge. When learning rate=0.1, it doesn't have too much difference between when learning rate=0.3. It need around 400 iterations to converge.
So leaning rates from 0.1 to 0.3, iterations from 100 to 400 are good.

Problem 3: Part 2: Implementing regularized linear regression

Problem 3.2.A1: Regularized linear regression cost function

Problem 3.2.A2: Gradient of the Regularized linear regression cost function

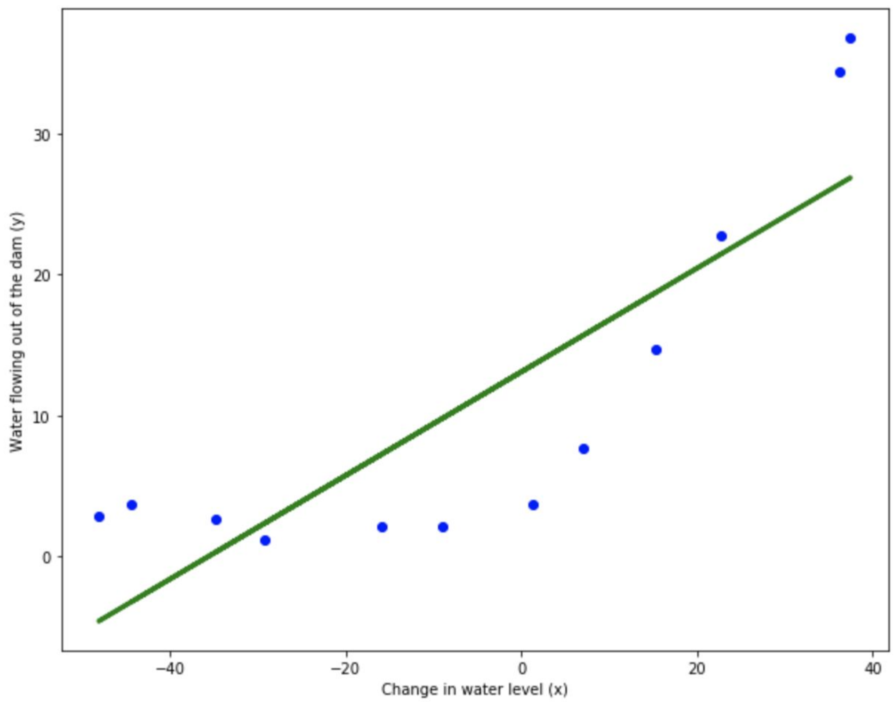


Figure 7. Best fit for linear model

Problem 3.2.A3: Learning curves

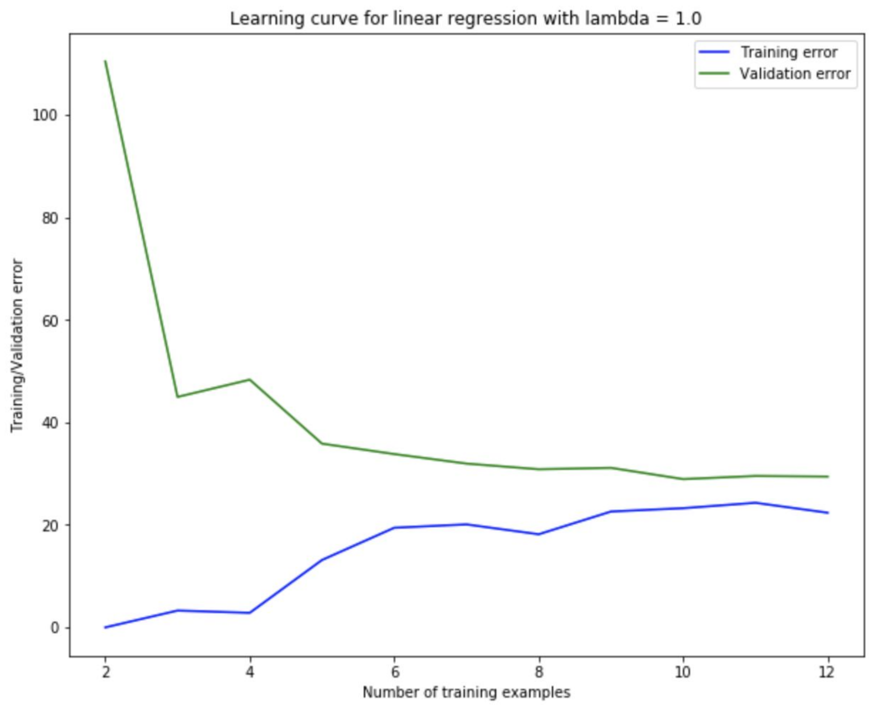


Figure 8. Learning Curves for linear model

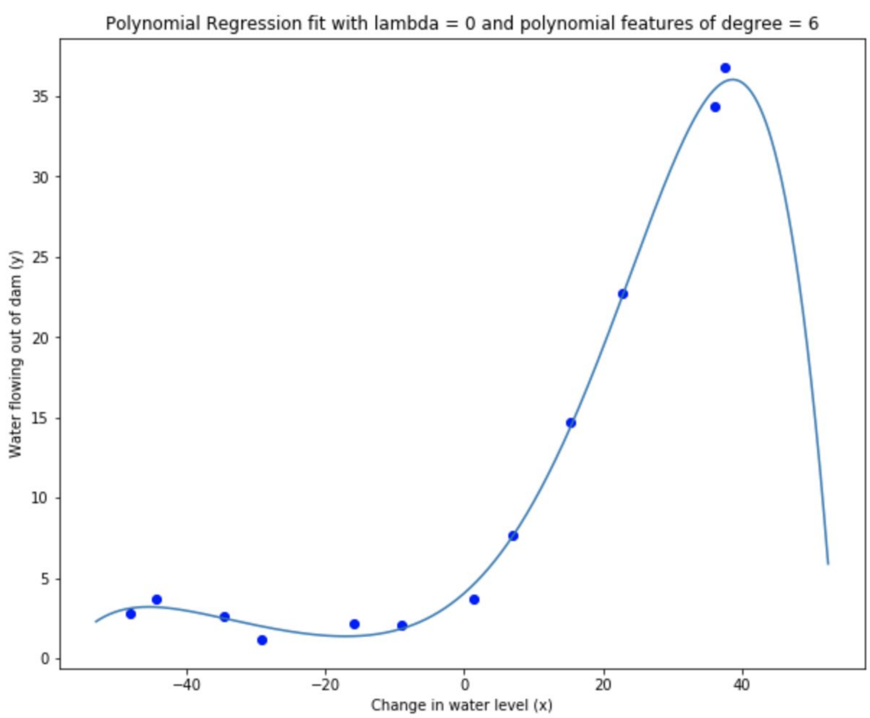


Figure 9: Polynomial fit for lambda = 0 with a p=6 order model.

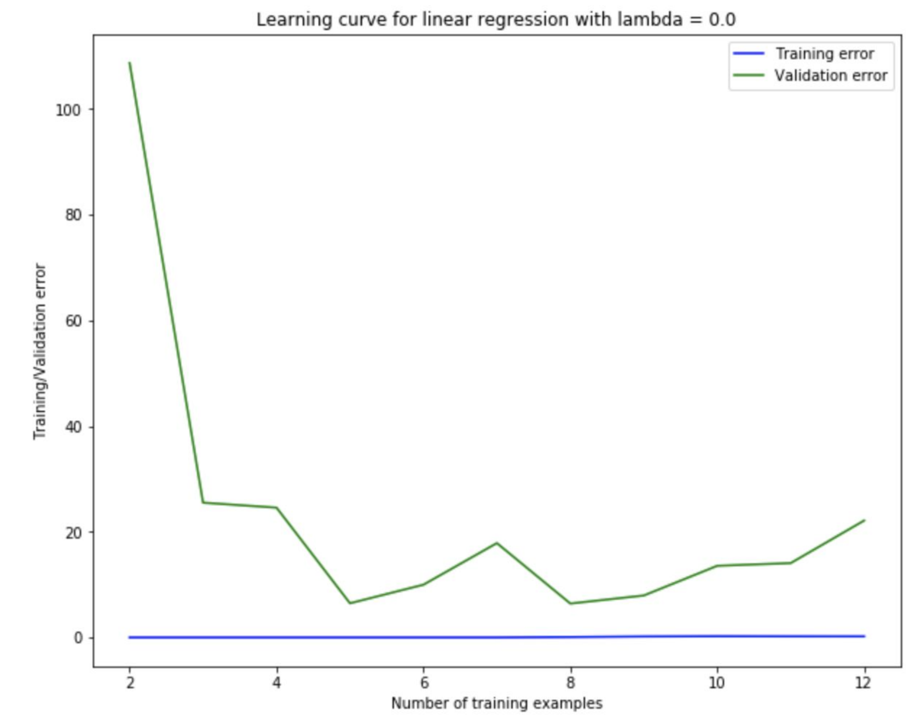
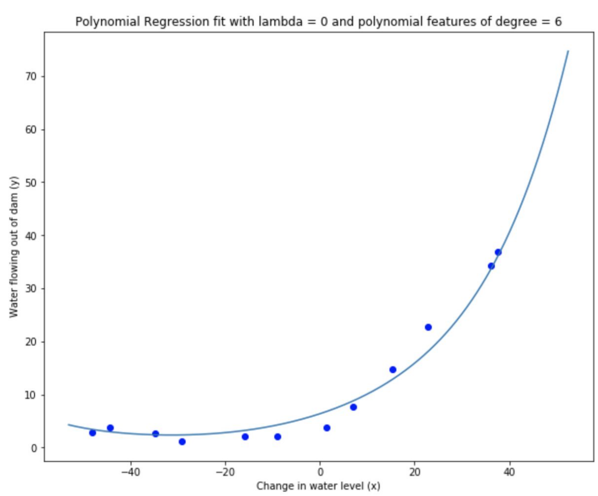
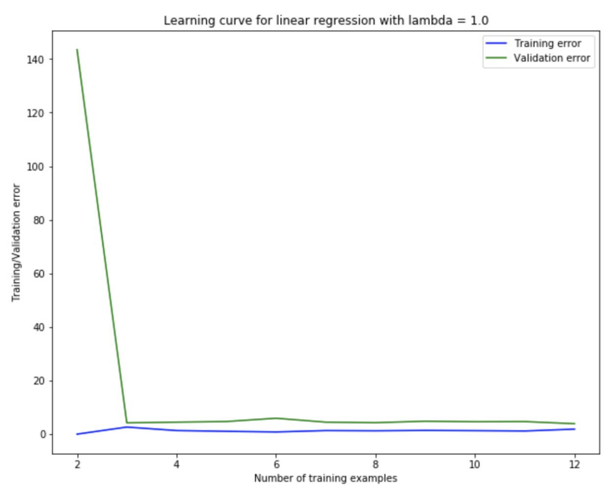


Figure 10: Learning curve for lambda = 0 with a p=6 order model.

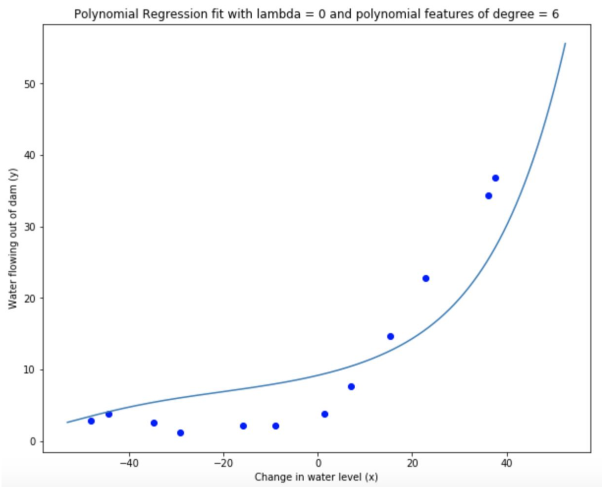
Problem 3.2.A4: Adjusting the regularization parameter

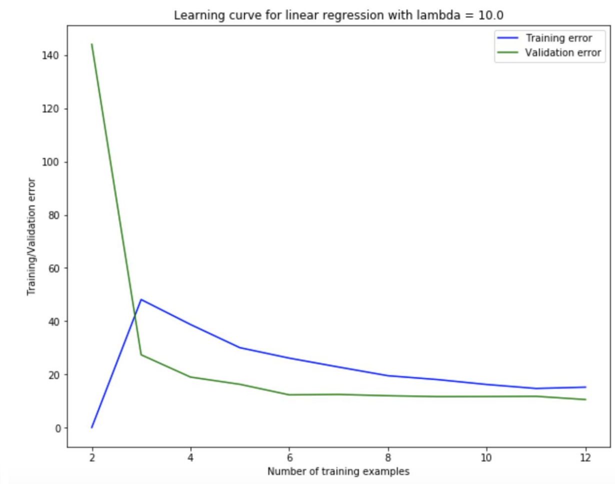
a. λ = 1.0



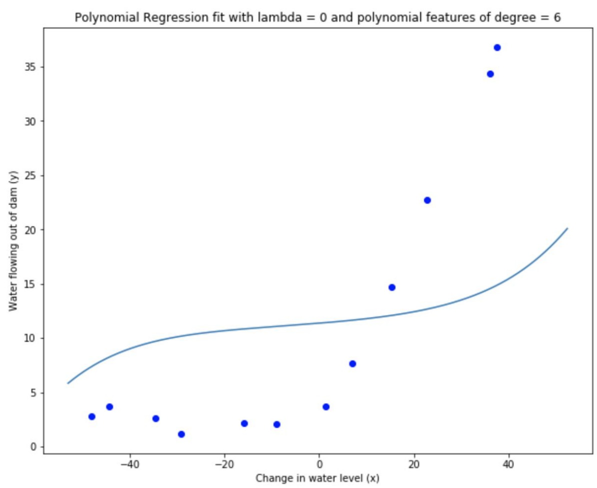


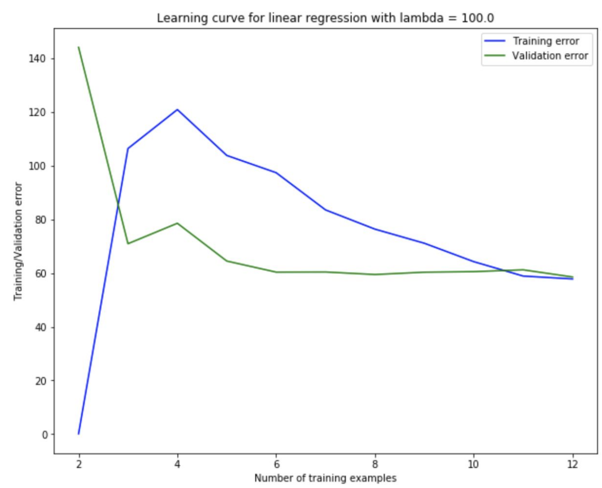
b. λ = 10.0





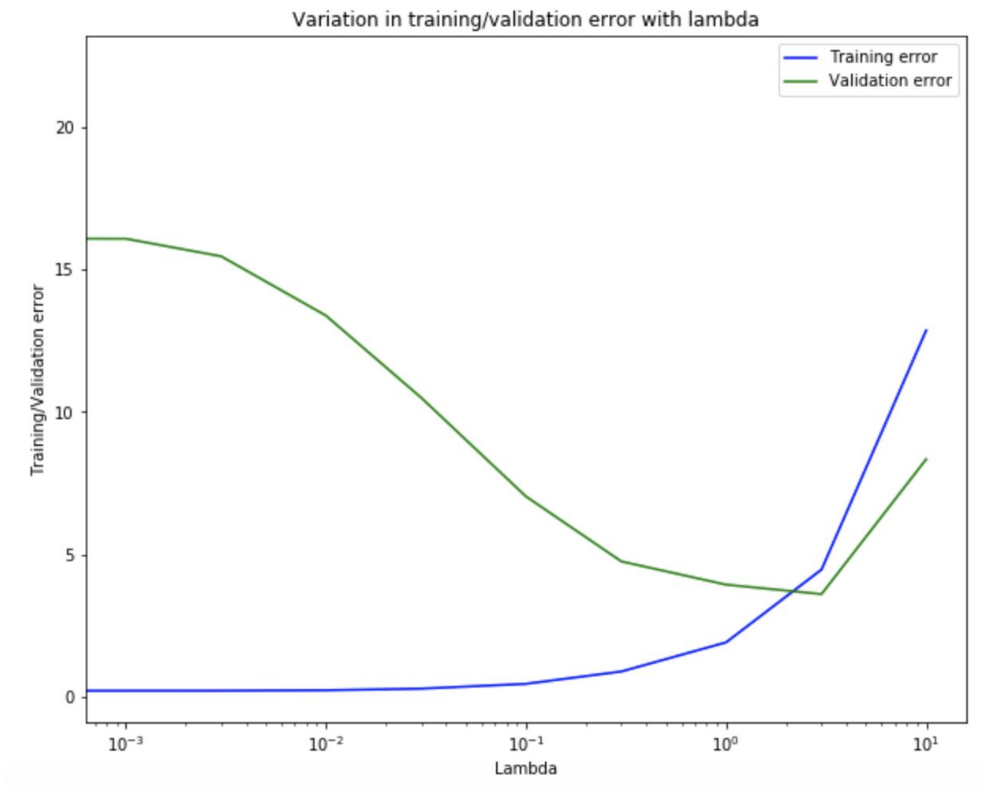
c. λ = 100.0





Increasing λ reduces the variance of learned model and makes it simpler. Without regularization (λ=0), the model has the problem of high variance. However, if the λ is too high(λ=100.0), the model becomes too simple and suffers from high bias. Considering both training error and validation error, λ=1.0 is the best choice.

Problem 3.2.A5: Selecting using a validation set



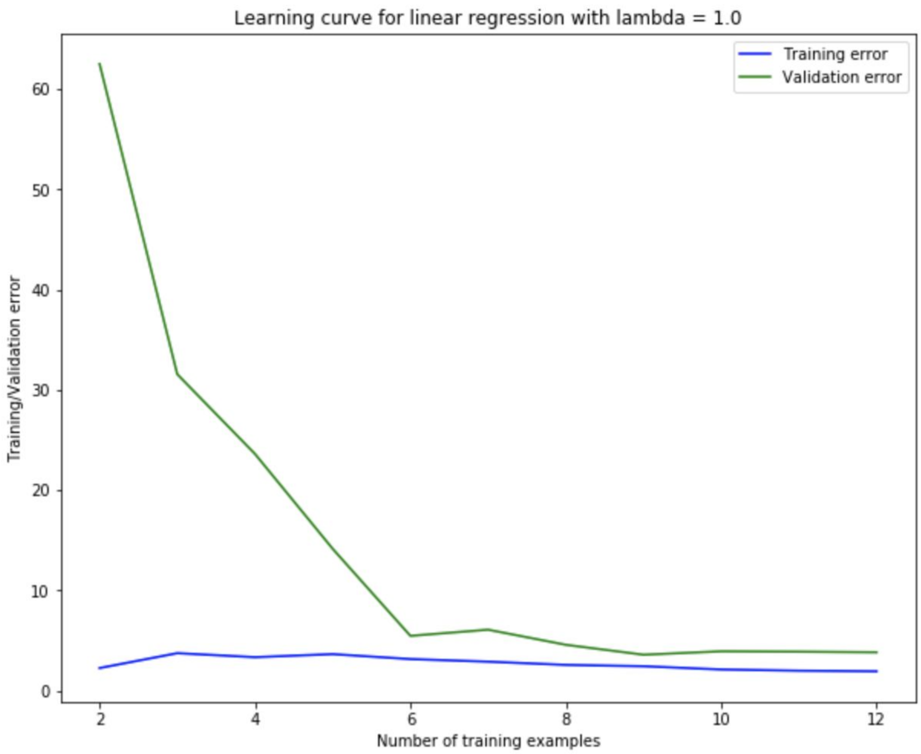
From the plot, validation error reaches the lowest when λ=3.0. When λ<3.0, there is a gap between training error and validation error, showing high variance of the model; when λ>3.0, both training error and validation error, showing high bias of the model. Hence, the best choice is λ=3.0.

Problem 3.2.A6: Computing test set error

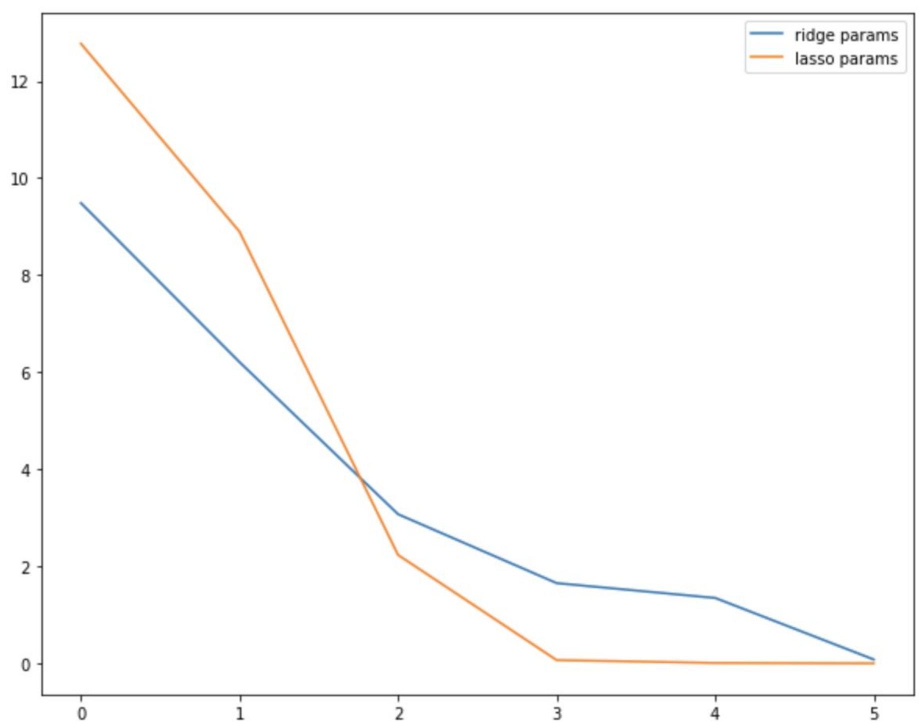
```
Optimization terminated successfully.
         Current function value: 15.237513
         Iterations: 15
         Function evaluations: 16
         Gradient evaluations: 16
reg=3.0, error_test=4.397623
```

When λ=3.0, test set error = 4.3976, which indicates that the model generalizes well.

Problem 3.2.A7: Plotting learning curves with randomly selected examples



Problem 3.2.A8: Comparing ridge regression and lasso regression models



Comparing to ridge regression, the lasso regression model has more centralized parameters. From the picture, the smallest three lasso parameters are close to zero and can be eliminated from the model. It shows that lasso regression results in a simpler model with fewer coefficients.