RICE UNIVERSITY

COMP540 Assignment 1

Yawen Guo and Negar Erfanian

January 24, 2020

1 BACKGROUND REFRESHER

1.1

See sampler.py

- Pictures are in second page.
- The probability that a sample from this distribution lies within the unit circle centered at (0.1, 0.2) is around 0.1837. This number will change each time when we execute it.

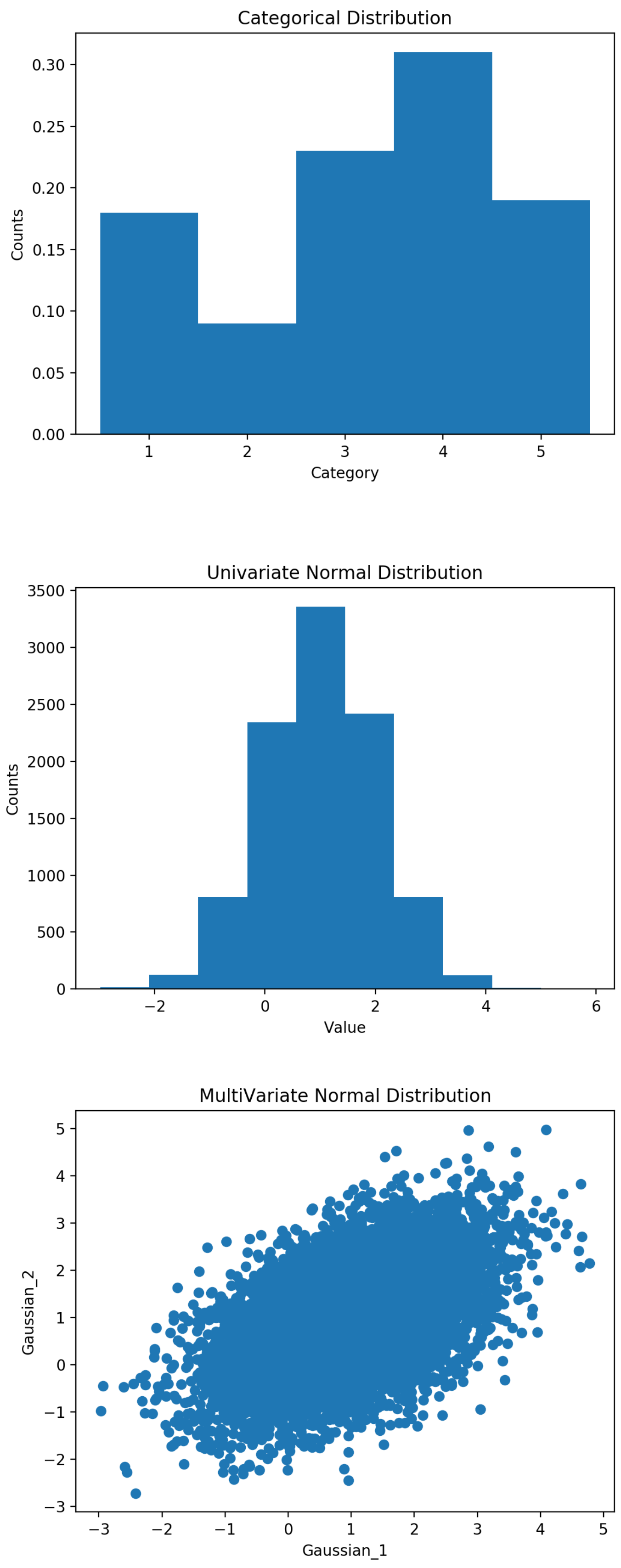

Figure 1.1: Plots generated by sampler.py

1.2

Let X and Y be independent Poisson random variables with parameters λ and μ, respectively, then we have

$$P\{X=k\}=\frac{\lambda^k e^{-\lambda}}{k!} \qquad ,P\{Y=k\}=\frac{\mu^k e^{-\mu}}{k!} \tag{1.1}$$

If we set $Z=X+Y$, we can write

$$\begin{aligned}
P\{Z=k\} &= \sum_{i=0}^{k} P\{X=i\}\times P\{Y=k-i\} \\
&= \sum_{i=0}^{k} \frac{\lambda^i e^{-\lambda}}{i!}\cdot\frac{\mu^{k-i}e^{-\mu}}{(k-i)!} \\
&= e^{-\lambda-\mu}\sum_{i=0}^{k}\frac{\lambda^i\mu^{k-i}}{i!(k-i)!} \\
&= e^{-\lambda-\mu}\sum_{i=0}^{k}\frac{k!}{i!\times(k-i)!}\frac{\lambda^i\mu^{k-i}}{k!} \\
&= e^{-\lambda-\mu}\sum_{i=0}^{k}\frac{C_k^i\cdot\lambda^i\mu^{k-i}}{k!} \\
&= e^{-\lambda-\mu}\cdot\frac{(\lambda+\mu)^k}{k!}
\end{aligned} \tag{1.2}$$

Therefore, we realize that Z is also a Poisson random variable with parameter $\lambda+\mu$.

1.3

$$\begin{aligned}
P(X_1=x_1) &= \int P(X_1=x_1|X_0=x_0)P(X_0=x_0)dx_0 \\
&= \alpha_0\alpha_1\int exp(-\frac{1}{2}\frac{(x_0-\mu_0)^2}{\sigma_0^2}+\frac{(x_1-x_0)^2}{\sigma^2})dx_0 \\
&= \alpha_0\alpha_1\int exp(-\frac{1}{2}\frac{\sigma^2(x_0-\mu_0)^2+\sigma_0^2(x_1-x_0)^2}{\sigma_0^2\sigma^2})dx_0 \\
&= \alpha_0\alpha_1\int exp(-\frac{1}{2}\frac{(\sigma^2+\sigma_0^2)x_0^2-2(\sigma^2\mu_0+\sigma_0^2x_1)x_0+\sigma^2\mu_0^2+\sigma_0^2x_1^2}{\sigma_0^2\sigma^2})dx_0 \\
&= \alpha_0\alpha_1\int exp(-\frac{1}{2}\frac{x_0^2-2(\frac{\sigma^2\mu_0+\sigma_0^2x_1}{\sigma^2+\sigma_0^2})x_0+(\frac{\sigma^2\mu_0+\sigma_0^2x_1}{\sigma^2+\sigma_0^2})^2}{\sigma_0^2\sigma^2/(\sigma^2+\sigma_0^2)}exp(-\frac{1}{2}\frac{\sigma^2\mu_0^2+\sigma_0^2x_1^2-\frac{(\sigma^2\mu_0+\sigma_0^2x_1)^2}{\sigma^2+\sigma_0^2}}{\sigma^2\sigma_0^2})dx_0
\end{aligned} \tag{1.3}$$

Assume the normalization constant is α', then we have

$$\begin{aligned}
P(X_1 = x_1) &= \frac{\alpha_0\alpha_1}{\alpha'} exp(-\frac{1}{2}\frac{(\sigma^2\mu_0^2 + \sigma_0^2 x_1^2)(\sigma^2+\sigma_0^2) - (\sigma^2\mu_0 + \sigma_0^2 x_1)^2}{\sigma^2\sigma_0^2(\sigma^2+\sigma_0^2)}) \\
&= \frac{\alpha_0\alpha_1}{\alpha'} exp(-\frac{1}{2}\frac{[\sigma_0^2(\sigma^2+\sigma_0^2) - \sigma_0^4]x_1^2 - 2\sigma^2\sigma_0^2\mu_0 x_1 + \sigma^2\mu_0^2(\sigma^2+\sigma_0^2) - \sigma^4\mu_0^2}{\sigma^2\sigma_0^2(\sigma^2+\sigma_0^2)}) \\
&= \frac{\alpha_0\alpha_1}{\alpha'} exp(-\frac{1}{2}\frac{x_1^2 - 2\mu_0 x_1 + \mu_0^2}{\sigma^2+\sigma_0^2} \\
&= \frac{\alpha_0\alpha_1}{\alpha'} exp(-\frac{1}{2}\frac{(x_1-\mu_0)^2}{\sigma^2+\sigma_0^2}
\end{aligned} \tag{1.4}$$

1.4

Assuming we have $P(A|B,C) > P(A|B)$, using the chain rule in Bayes theorem we can write

$$P(A|B,C) = \frac{P(A,B|C)P(C)}{P(B,C} = \frac{P(A,B|C)}{P(B|C)} \tag{1.5}$$

If we multiply both the numerator and the denominator by $P(C)$, we have

$$P(A|B,C) = \frac{P(A,B|C)P(C)}{P(B|C)P(C)} = \frac{P(A,B,C)}{P(B,C)} = \frac{P(C|A,B)P(A|B)}{P(C|B)} > P(A|B) \tag{1.6}$$

Therefore, we have

$$\frac{P(C|A,B)}{P(C|B)} = \frac{1-P(C^c|A,B)}{1-P(C^c|B)} > 1 \tag{1.7}$$

and consequently

$$\frac{P(C^c|A,B)}{P(C^c|B)} < 1 \tag{1.8}$$

If we multiply both sides of (1.8) by $P(A|B)$ we have

$$\begin{gathered}
\frac{P(C^c|A,B)P(A|B)}{P(C^c|B)} < P(A|B) \Longrightarrow \\
\frac{P(C^c|A,B)P(A,B)}{P(C^c|B)P(B)} = \frac{P(C^c,A,B)}{P(C^c,B)} = \frac{P(A,B|C^c)}{P(B|C^c)} = P(A|B,C^c) < P(A|B)
\end{gathered} \tag{1.9}$$

1.5

$$M = \begin{pmatrix}1\\2\end{pmatrix}\begin{pmatrix}2 & 3\end{pmatrix} = \begin{pmatrix}2 & 3\\4 & 6\end{pmatrix} \tag{1.10}$$

If we have $M.v = \lambda.v$, where λ represents the eigenvalues and v represents the eigenvectors, in order to find the eigenvalues we write

$$\begin{gathered}
|M - \lambda I| = 0 \\
|\begin{pmatrix}2-\lambda & 3\\4 & 6-\lambda\end{pmatrix}| = 0 \\
(2-\lambda)(6-\lambda) - 12 = 0
\end{gathered} \tag{1.11}$$

Therefore, the eigenvalues are $\lambda_1 = 0$ and $\lambda_2 = 8$.
Having $\lambda_1 = 0$:

$$\begin{aligned} 2x_1 + 3x_2 &= 0 \\ 4x_1 + 6x_2 &= 0 \\ 2x_1 &= -3x_2 \end{aligned} \tag{1.12}$$

Therefore, $v_1 = \begin{pmatrix} 3 \\ -2 \end{pmatrix}$.
Having $\lambda_2 = 8$:

$$\begin{aligned} -6x_1 + 3x_2 &= 0 \\ 4x_1 - 2x_2 &= 0 \end{aligned} \tag{1.13}$$

Therefore, $v_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$.

1.6

If A is a positive semin-definite matrix, then $\forall x \in R^n$ we have $x^\top Ax \geqslant 0$. On the other hand, if v_1 and λ_1 represent the eigenvector and eigenvalue of A, respectively, we have $A.v_1 = \lambda_1.v_1$. Therefore we can write $v_1^\top.A.v_1 = \lambda_1.v_1^\top.v_1$.
According to the definition of a positive semi-definite matrix we have $v_1^\top.A.v_1 \geqslant 0$ and since always $v_1^\top.v_1 \geqslant 0$, therefore, $\lambda_1 \geqslant 0$. Accordingly, we can prove that all the eigenvalues of A must be bigger than or equal to 0.

1.7

1.7.1

Since $(A+B)^2 = A^2 + AB + BA + B^2$, if $AB \neq BA$, then $(A+B)^2 \neq A^2 + 2AB + B^2$
For instance:

$$A = \begin{pmatrix} 2 & 2 \\ 0 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 0 \\ 1 & 0 \end{pmatrix}$$

Where $AB = \begin{pmatrix} 6 & 0 \\ 0 & 0 \end{pmatrix}$ and $BA = \begin{pmatrix} 4 & 4 \\ 2 & 2 \end{pmatrix}$. Therefore, $AB \neq BA$ and accordingly, $(A+B)^2 \neq A^2 + 2AB + B^2$.

1.7.2

Having

$$A = \begin{pmatrix} 2 & 2 \\ 0 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 0 \\ -2 & 0 \end{pmatrix}$$

we realize that $AB = 0$, where $A \neq 0$ and $B \neq 0$.

1.8

$$\begin{aligned}
A^T A &= (I-2uu^T)^T(I-2uu^T) \\
&= (I-2uu^T)(I-2uu^T) \\
&= I-4uu^T+4(uu^T)(uu^T) \\
&= I-4uu^T+4u(u^Tu)u^T \\
&= I-4uu^T+4uu^T \\
&= I
\end{aligned} \tag{1.14}$$

1.9

1.9.1

Since $0 \leqslant \lambda \leqslant 1$, we can easily show that

$$\begin{aligned}
f(\lambda x_1 + (1-\lambda)x_2) &= (\lambda x_1 + (1-\lambda)x_2)^3 \\
&= \lambda^3 x_1^3 + (1-\lambda)^3 x_2^3 + 3\lambda^2 x_1^2(1-\lambda)x_2 + 3\lambda x_1(1-\lambda)^2 x_2^2 \\
&\leqslant \lambda x_1^3 + (1-\lambda)x_2^3 \\
&= \lambda f(x_1) + (1-\lambda)f(x_2)
\end{aligned} \tag{1.15}$$

and

$$f''(x) = 6x \geq 0 \quad for \quad x \geq 0$$
$$So \quad f(x) = x^3 \quad is \quad convex \quad for \quad x \geq 0$$

1.9.2

Since $max\{x_1, x_2\}$ is the intersection of the area above functions $f(x_1) = x_1$ and $f(x_2) = x_2$, if we prove that $f(x) = x$ is a convex function, and since the intersection of two convex functions is a convex function, we can prove that $f(x_1, x_2) = max\{x_1, x_2\}$ is a convex function. Having $f(x) = x$ we can write:

$$\begin{aligned}
f(\lambda x_1 + (1-\lambda)x_2) = \lambda x_1 + (1-\lambda)x_2 = \lambda f(x_1) + (1-\lambda)f(x_2) \\
\text{and} \quad f''(x) = 0
\end{aligned} \tag{1.16}$$

Therefore, we realize that $f(x) = x$ and accordingly $f(x_1, x_2) = max\{x_1, x_2\}$ are convex functions.

1.9.3

If functions f and g are convex functions we can write

$$\begin{aligned}
f(\lambda x + (1-\lambda)y) \leqslant \lambda f(x) + (1-\lambda)f(y) \\
g(\lambda x + (1-\lambda)y) \leqslant \lambda g(x) + (1-\lambda)g(y) \\
\text{and} \;\; f''(x) = g''(0) \geqslant 0
\end{aligned} \tag{1.17}$$

Therefore, we have

$$\begin{aligned}
f(\lambda x + (1-\lambda)y) + g(\lambda x + (1-\lambda)y) \leqslant \lambda(f(x) + g(x)) + (1-\lambda)(f(y) + g(y)) \\
\text{and} \quad (f(x) + g(x))'' \geqslant 0
\end{aligned} \tag{1.18}$$

Which proves that $f + g$ is a convex function.

1.9.4

Let $h(x) = f(x).g(x)$. Therefore, $h(x)' = f(x)'.g(x)+f(x).g(x)'$ and $h(x)'' = f(x)''.g(x)+f(x).g(x)''+2f'(x).g'(x)$.
Since f and g are convex functions then $f''(x) \geqslant 0$ and $g''(x) \geqslant 0$. Therefore, for $h(x)'' \geqslant 0$ we have to check $2f'(x).g'(x) \geqslant 0$. Since f and g have their minimum at the same point, $2f'(x).g'(x)$ is always non-negative. As a result, we proved that $h(x) = f(x).g(x)$ is a convex function.

1.10

For each p_i subject to normalization constraint $\sum_{i=1}^{K} p_i = 1$

$$\begin{aligned}
\sum_{i=1}^{K} p_i - 1 &= 0 \\
L = -\sum_{i=1}^{K} p_i \log(p_i) + \lambda(\sum_{i=1}^{K} p_i - 1) \\
\frac{\partial L}{\partial p_i} &= -\log p_i - 1 + \lambda = 0 \\
\text{Therefore,} \quad \log p_i &= \lambda - 1
\end{aligned}$$

It means all p_i has the equal value and in order to satisfy normalization constraint, $p_i = 1/K$ Therefore when all $p_i = \frac{1}{K}$, $H(p)$ has the maximum value and the highest entropy subjects to even distribution. In this regard, we will have $\lambda = 1 - \log K$.

2 Locally weighted linear regression

2.1

We can write the cost function as

$$\begin{aligned}
J(\theta) &= \frac{1}{2}\sum_{i=1}^{m} w^{(i)}(\theta^\top x^{(i)} - y^{(i)})^2 \\
&= \frac{1}{2}(w^{(1)}(\theta^\top x^{(1)} - y^{(1)})^2 + w^{(2)}(\theta^\top x^{(2)} - y^{(2)})^2 + \ldots + w^{(m)}(\theta^\top x^{(m)} - y^{(m)})^2) \\
&= \frac{1}{2}\begin{pmatrix}(\theta^\top x^{(1)} - y^{(1)}) & (\theta^\top x^{(2)} - y^{(2)}) & \ldots & (\theta^\top x^{(m)} - y^{(m)})\end{pmatrix}\begin{pmatrix} w^{(1)}(\theta^\top x^{(1)} - y^{(1)}) \\ w^{(2)}(\theta^\top x^{(2)} - y^{(2)}) \\ \vdots \\ w^{(m)}(\theta^\top x^{(m)} - y^{(m)})\end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix}(\theta^\top x^{(1)} - y^{(1)}) & (\theta^\top x^{(2)} - y^{(2)}) & \ldots & (\theta^\top x^{(m)} - y^{(m)})\end{pmatrix}\begin{pmatrix} w^{(1)} & 0 & \ldots & 0 \\ 0 & w^{(2)} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & w^{(m)}\end{pmatrix}\begin{pmatrix}(\theta^\top x^{(1)} - y^{(1)}) \\ (\theta^\top x^{(2)} - y^{(2)}) \\ \vdots \\ (\theta^\top x^{(m)} - y^{(m)})\end{pmatrix} \\
&= \frac{1}{2}(X\theta - y)^\top W(X\theta - y)
\end{aligned} \tag{2.1}$$

Where W is an $m \times m$ diagonal matrix and $w^{(i)}$, $i : \{1,2,...,m\}$ are the components on the diagonal.

2.2

Since we have $J(\theta) = \frac{1}{2}(X\theta - y)^\top W(X\theta - y)$, if we take the derivative of the weighted cost function with respect to θ and put it equal to zero, we have

$$\frac{\partial J(\theta)}{\partial \theta} = X^\top W(X\theta - y) = X^\top WX\theta - X^\top Wy = 0 \tag{2.2}$$

Therefore, we realize that

$$\hat{\theta} = (X^\top WX)^{-1} X^\top Wy \tag{2.3}$$

2.3

According to the rule in batch gradient descent, we have

$$\begin{aligned} \theta_{\text{new}} := \theta_{\text{old}} - \alpha \frac{\partial J(\theta)}{\partial \theta} = \theta_{\text{old}} - \alpha X^\top W(X\theta_{\text{old}} - y) \\ \theta_{\text{old}} := \theta_{\text{new}} \end{aligned} \tag{2.4}$$

where α is the learning rate. We continue this until θ_{old} and θ_{new} are very close to each other and θ_{new} does not change significantly.
Batch gradient descent is used when we are dealing with a limited number of examples in a data. Therefore, to do optimization we use the whole data at the same time as shown by matrix X. In (2.4), W is a $m \times m$ diagonal matrix, where m is the number of examples in data X, and w_is are the components on the diagonal of W.
Locally weighted linear regression is an non-parametric method.Since instead of fitting a single regression line, we need to fit many linear regression models.

3 Properties of the linear regression estimator

3.1

3.1.1

If X is an $m \times d$ matrix of $x^{(i)}$ and y is a $m \times 1$ column vector of $y^{(i)}$, where $i : \{1, 2, ..., m\}$
we have $E(y) = X\theta^*$. Therefore, based on Least square estimators, we can write

$$\theta = (X^\top X)^{-1} X^\top y \;\Rightarrow\; E[\theta] = (X^\top X)^{-1} X^\top E(y) = (X^\top X)^{-1} X^\top X\theta^* = \theta^* \tag{3.1}$$

3.1.2

Since we have $\text{Var}(y) = \text{Var}(\epsilon) = \sigma^2 I$, we can write

$$\begin{aligned} \text{Var}(\theta) = \text{Var}[(X^\top X)^{-1} X^\top y] = (X^\top X)^{-1} X^\top \text{Var}(y) X (X^\top X)^{-1} \\ = (X^\top X)^{-1} X^\top \sigma^2 I X (X^\top X)^{-1} = (X^\top X)^{-1} \sigma^2 \end{aligned} \tag{3.2}$$

4 Implementing linear regression and regularized linear regression

4.1 .A: Linear regression with one variable

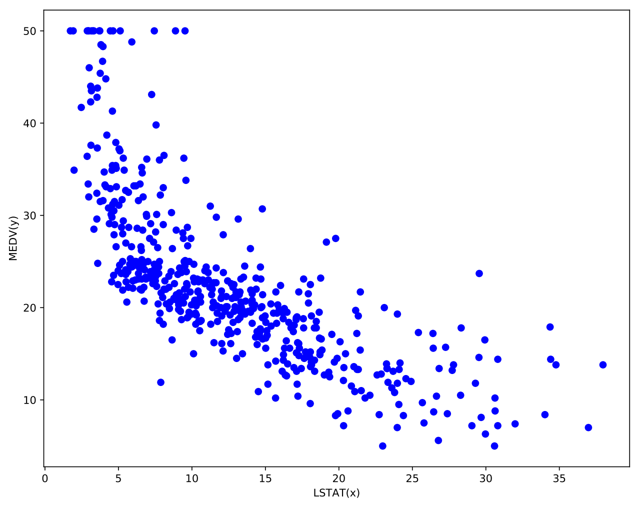

Figure 4.1: Scatter plot of training data

4.2 .A1: Computing the cost function

Code is attached.

4.3 .A2: Implementing gradient descent

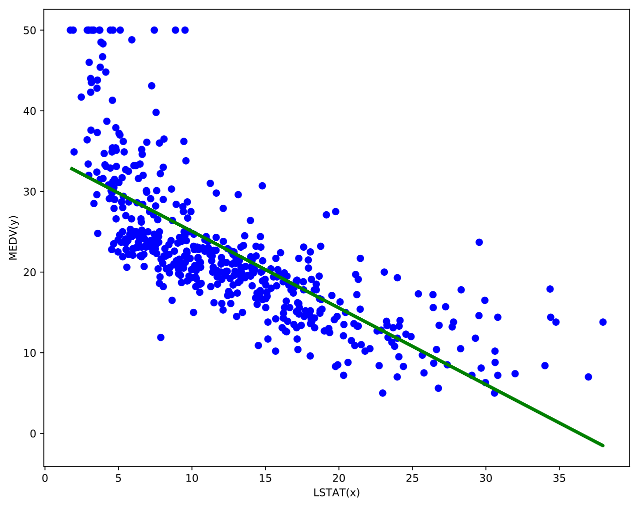

Figure 4.2: Fitting a linear model to the data in Figure1

4.4 .A3: Predicting on unseen data

For lower status percentage = 5, we predict a median home value of 298034.49
For lower status percentage = 50, we predict a median home value of -129482.13

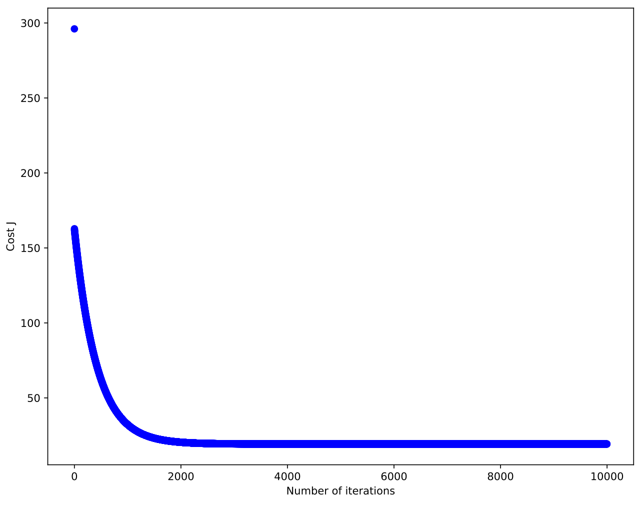

Figure 4.3: Convergence graph

4.5 .B: Linear regression with multiple variables

4.6 .B1: Feature normalization

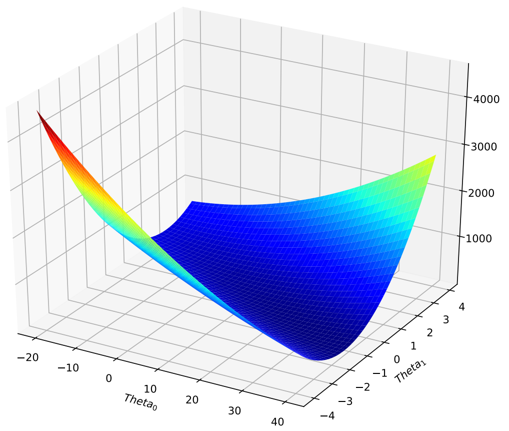

Figure 4.4: Surface plot of cost function J

4.7 .B2: Loss function and gradient descent

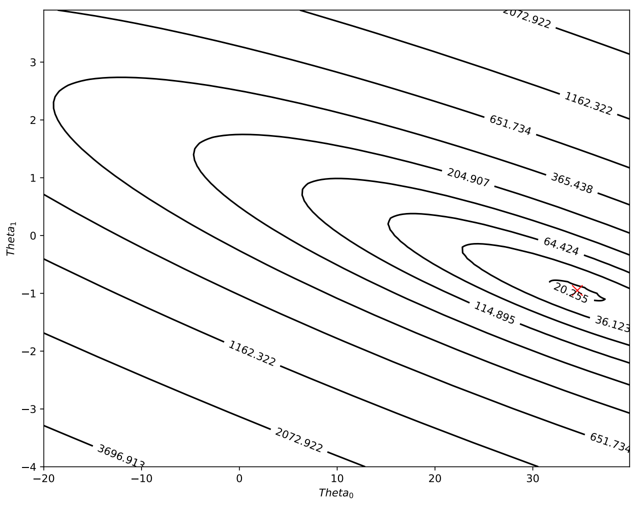

Figure 4.5: Contour plot of cost function J

4.8 .B3: Making predictions on unseen data

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

4.9 .B4: Normal equations

For average home in Boston suburbs, we predict a median home value of 225328.06 using the normal equations.

4.10 .B5: Exploring convergence of gradient descent

In this problem, we have chosen the iteration to be 5000, and we chose different values of $0.3, 0.1, 0.03, 0.01$ for the learning rate. We realized that if we use any of these learning rates, the gradient descent converges after doing less than 400 iterations. Figure 4.7 shows that if we use a learning rate= 0.3, we have the fastest convergence with almost 100 iterations.

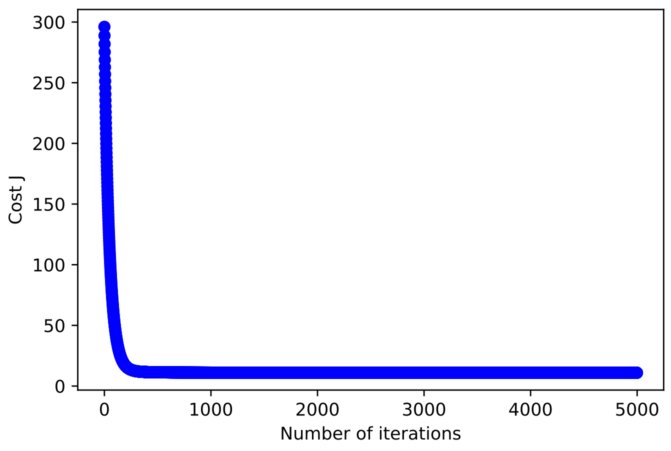

Figure 4.6: Convergence of gradient descent for linear regression with multiple variables (Boston housing data set)

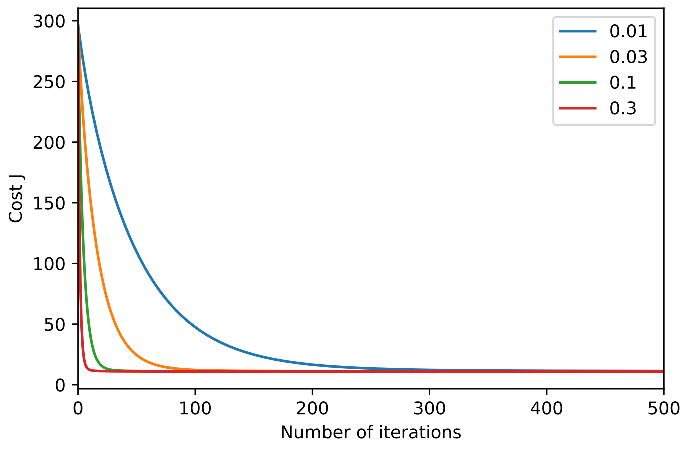

Figure 4.7: Convergence graph with different learning rates

5 Implementing regularized linear regression

5.1 A1: Regularized linear regression cost function

Code is attached.

5.2 A2: Gradient of the Regularized linear regression cost function

Code is attached.

5.3 A3: Learning curves

In Figure 5.3, you can observe that both the train error and cross validation error are high when the number of training examples is increased. This reflects a high bias problem in the model – the linear regression model is too simple and is unable to fit our dataset well.

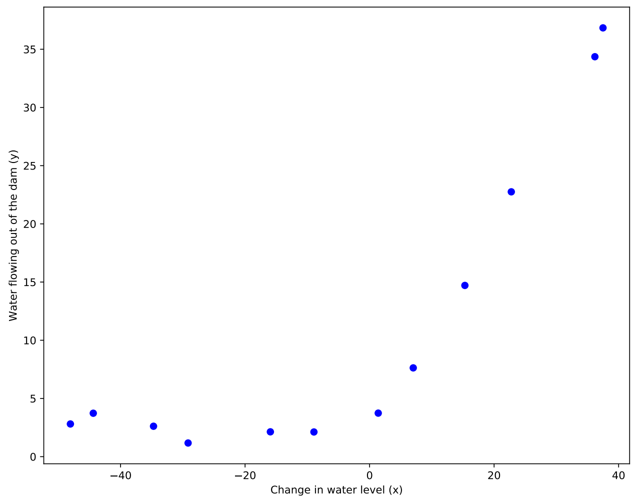

Figure 5.1: The training data for regularized linear regression

5.4 A4: Adjusting the regularization parameter

Figures 5.4 and 5.5 represent the fit and the learning curve when $\lambda = 0$.
Figures 5.6 and 5.7 represent the fit and the learning curve when $\lambda = 1$.
Figures 5.8 and 5.9 represent the fit and the learning curve when $\lambda = 10$.
Figures 5.10 and 5.11 represent the fit and the learning curve when $\lambda = 100$.
We realize that as we increase the reguralization parameter λ, we decrease the overfitting problem, therefore the variance decreases, the training error increases and the training and validation error get closer to each other. But if we increase λ so much,like the example where $\lambda = 100$, we underfit the model which causes very high training, validation and test error. Therefore, finding a good range for λ is critical. In this example, $\lambda = 1$ seems reasonable.

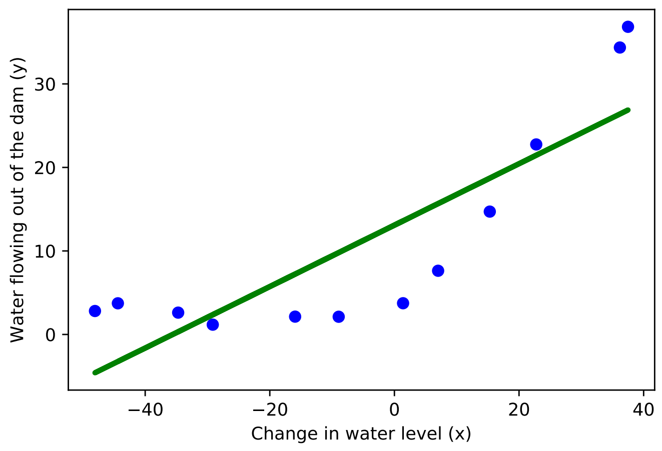

Figure 5.2: The best fit line for the training data

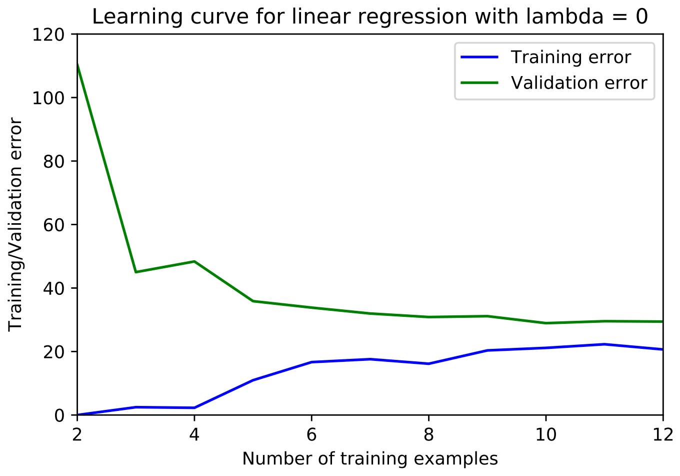

Figure 5.3: The best fit line for the training data

5.5 A5: Selecting λ using a validation set

As shown in figure 5.12, good choices for λ can be $0.3,\ 1$ and 3. The vertical line shows $\lambda = 0.3$.

5.6 A6: Computing test set error

As we realized in the previous section, after we find our parameters, good choices for λ are $0.3,\ 1$ and 3 which cause the testing error to be $6.748635,\ 9.055139$ and 19.59522, respectively.

5.7 A7: Plotting learning curves with randomly selected examples

Figure 5.13 represents the learning curves for both the training and validation errors as we randomly select subsets of the training and validation sets to tune the parameters. We realize that the learning curves in this case become smoother compared to when we did not do random sampling and averaging while $\lambda = 1$.

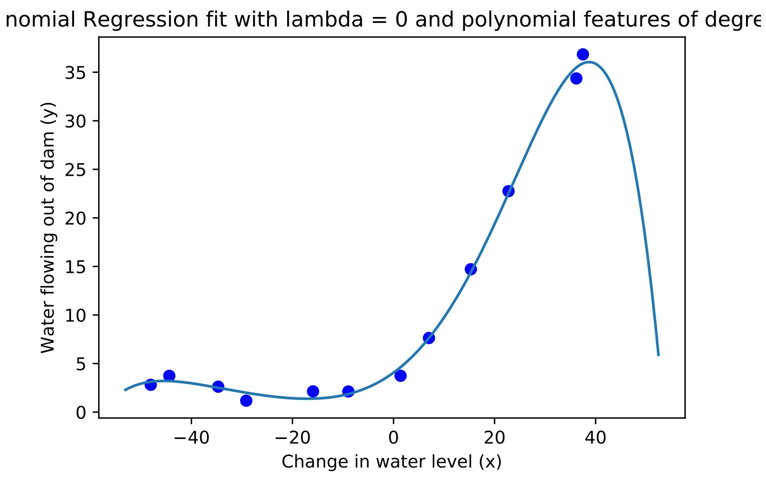

Figure 5.4: Polynomial fit for $\lambda = 0$ with a p=6 order model

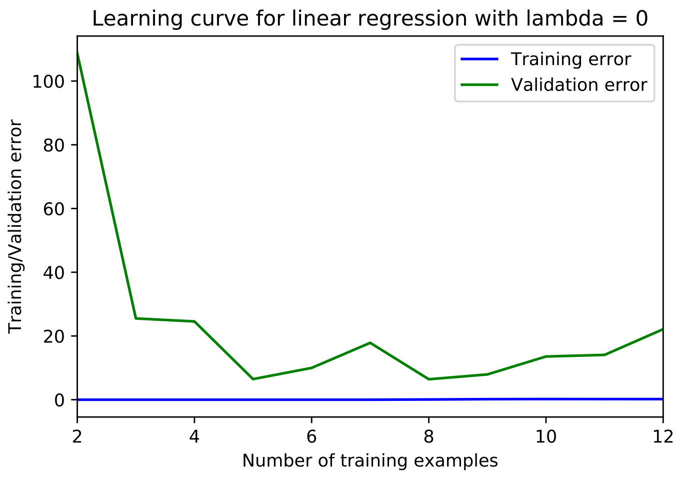

Figure 5.5: Learning curve for $\lambda = 0$

5.8 A8: Comparing ridge regression and lasso regression models

Figure 5.14 represents the magnitude of the entries of θ trained from the lasso and ridge regression in descending order. Lasso and ridge regression both decrease the complexity of a learning model and prevent overfitting. Ridge regression reduces the very high parameters where lasso puts them equal to zero. Therefore, in figure 5.14,we can see that the magnitude of the entries of θ decrease faster when we they are trained from the lasso regression.

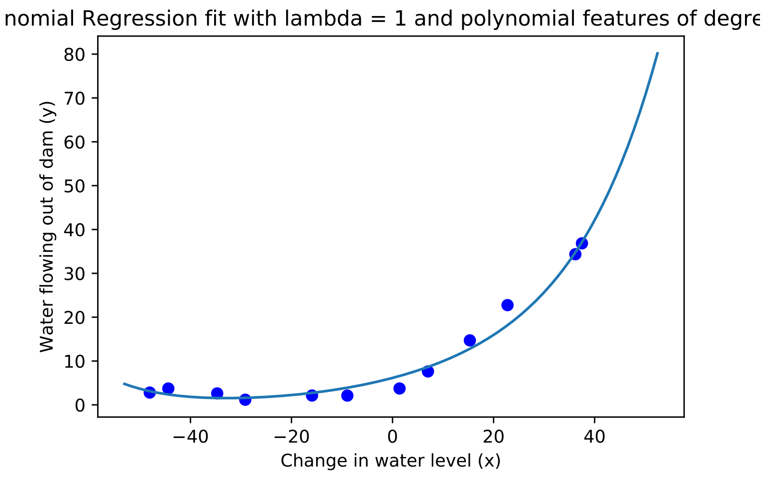

Figure 5.6: Polynomial fit for $\lambda = 1$ with a p=6 order model

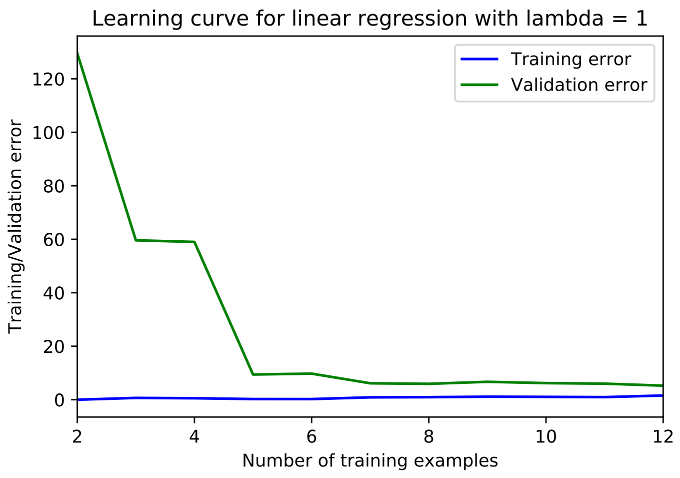

Figure 5.7: Learning curve for $\lambda = 1$

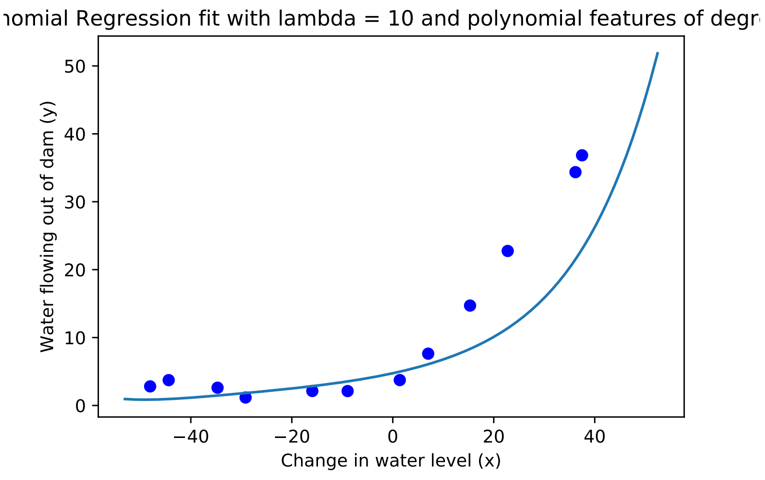

Figure 5.8: Polynomial fit for $\lambda = 10$ with a p=6 order model

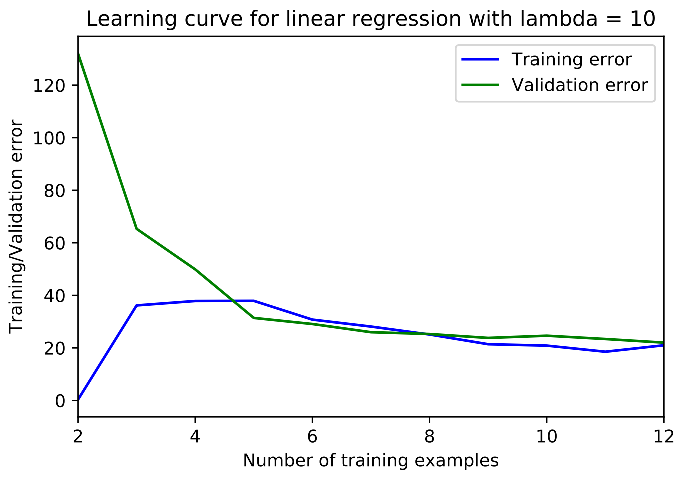

Figure 5.9: Learning curve for $\lambda = 10$

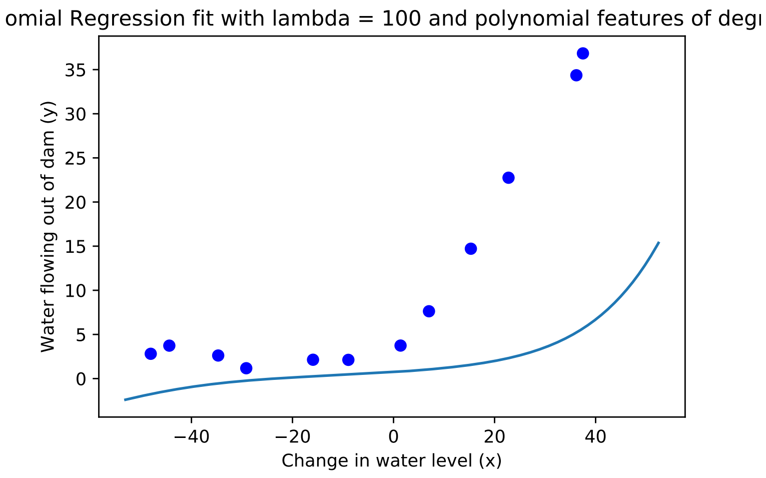

Figure 5.10: Polynomial fit for $\lambda = 100$ with a p=6 order model

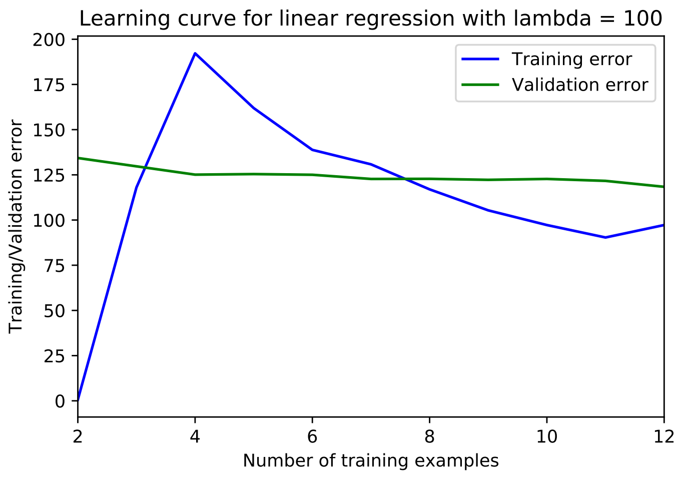

Figure 5.11: Learning curve for $\lambda = 100$

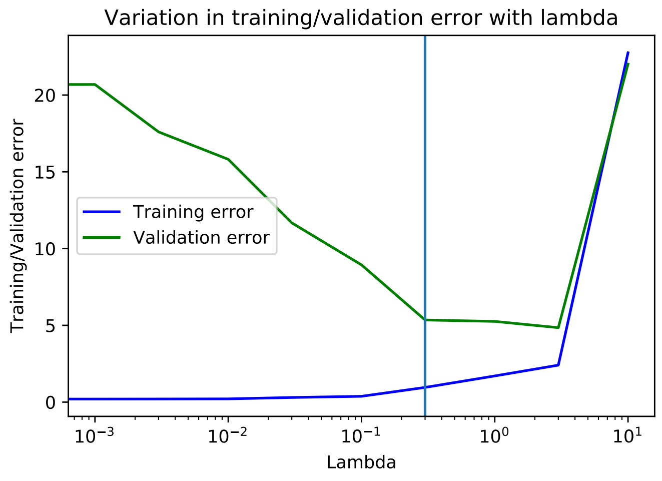

Figure 5.12: Changes of training/validation error as we change λ. The vertical line represents $\lambda = 0.3$.

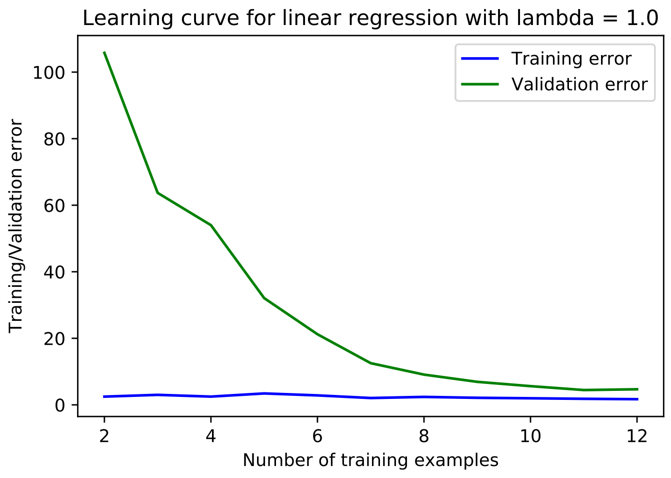

Figure 5.13: learning curves with randomly selected examples.

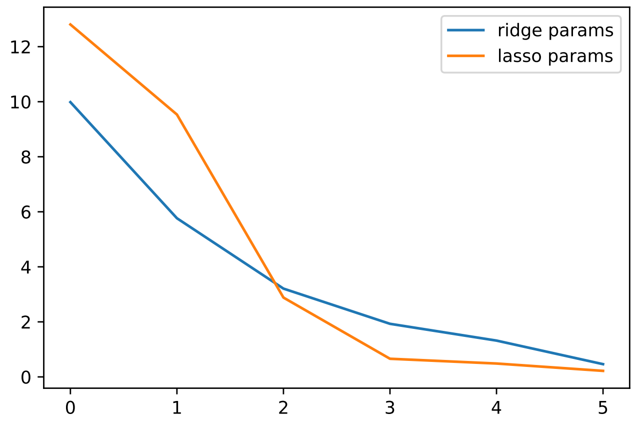

Figure 5.14: Comparing ridge regression and lasso regression models.