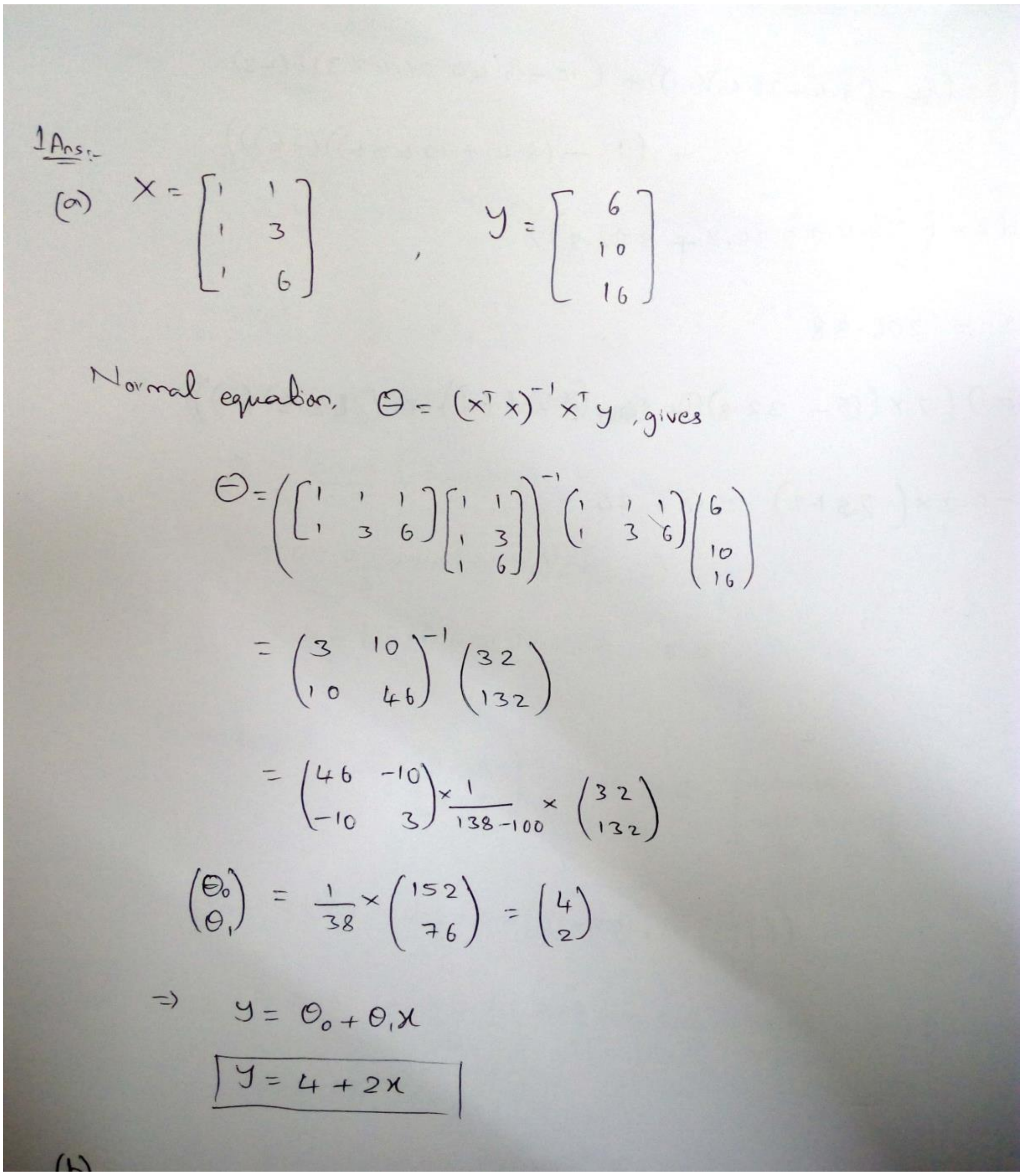
Machine Learning – ES 654

Spring 2019

IITGN

1Ans.

a)



1Ans:-

(a) $X = \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 6 \end{bmatrix}$, $y = \begin{bmatrix} 6 \\ 10 \\ 16 \end{bmatrix}$

Normal equation, $\Theta = (X^T X)^{-1} X^T y$, gives

$$\Theta = \left(\begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 6 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 6 \end{bmatrix} \right)^{-1} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 3 & 6 \end{pmatrix} \begin{pmatrix} 6 \\ 10 \\ 16 \end{pmatrix}$$

$$= \begin{pmatrix} 3 & 10 \\ 10 & 46 \end{pmatrix}^{-1} \begin{pmatrix} 32 \\ 132 \end{pmatrix}$$

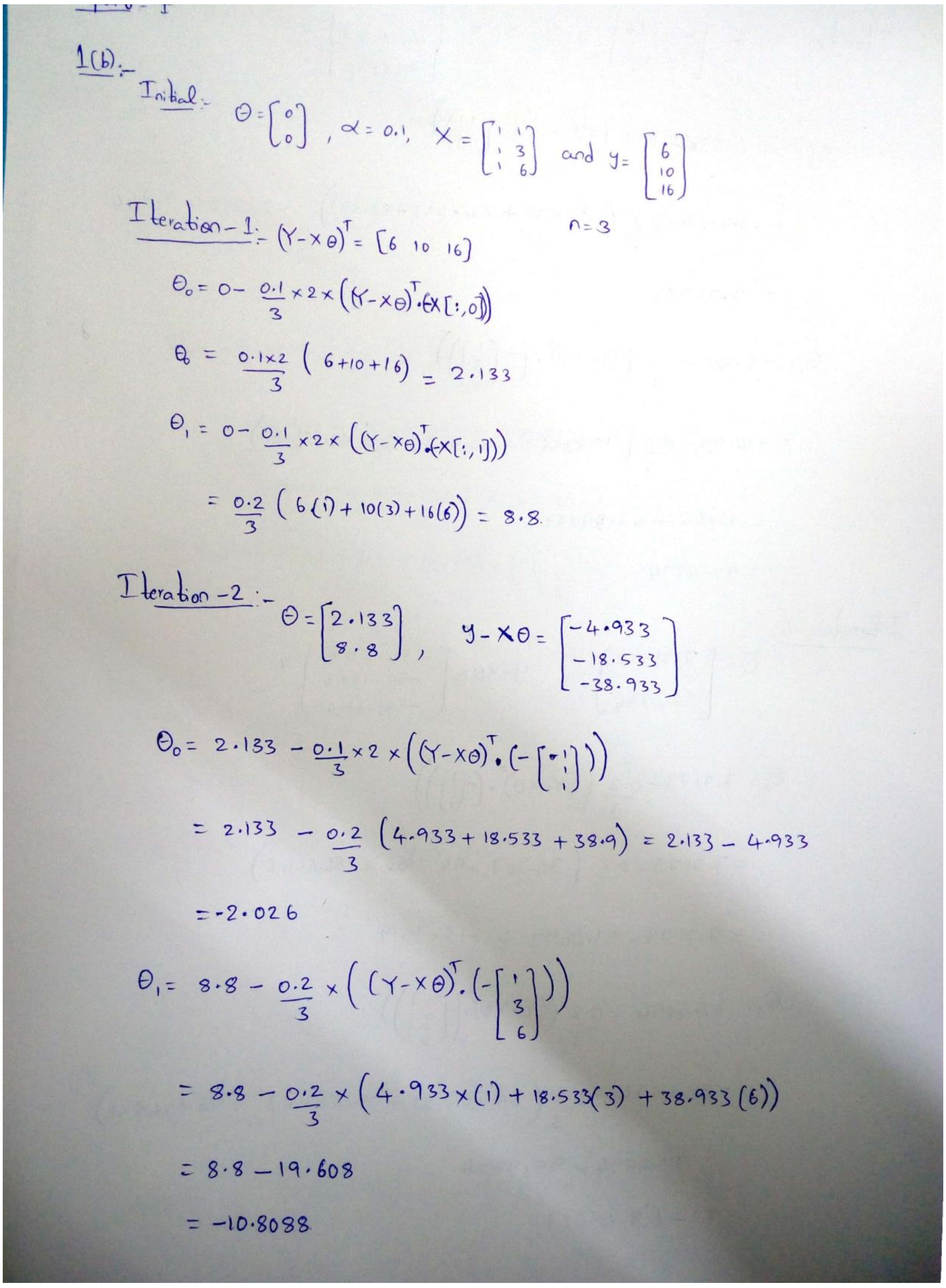
$$= \begin{pmatrix} 46 & -10 \\ -10 & 3 \end{pmatrix} \times \frac{1}{138-100} \times \begin{pmatrix} 32 \\ 132 \end{pmatrix}$$

$$\begin{pmatrix} \Theta_0 \\ \Theta_1 \end{pmatrix} = \frac{1}{38} \times \begin{pmatrix} 152 \\ 76 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$$

$\Rightarrow$ $y = \Theta_0 + \Theta_1 x$

$$\boxed{y = 4 + 2x}$$

b)



1(b):-

Initial:- $\theta = \begin{bmatrix}0\\0\end{bmatrix}$, $\alpha = 0.1$, $X = \begin{bmatrix}1 & 1\\1 & 3\\1 & 6\end{bmatrix}$ and $y = \begin{bmatrix}6\\10\\16\end{bmatrix}$

Iteration-1:- $(Y - X\theta)^T = [6 \;\; 10 \;\; 16]$ $n=3$

$\theta_0 = 0 - \frac{0.1}{3} \times 2 \times \left((Y - X\theta)^T \cdot (-X[:,0])\right)$

$\theta_0 = \frac{0.1 \times 2}{3}(6 + 10 + 16) = 2.133$

$\theta_1 = 0 - \frac{0.1}{3} \times 2 \times \left((Y - X\theta)^T \cdot (-X[:,1])\right)$

$= \frac{0.2}{3}\left(6(1) + 10(3) + 16(6)\right) = 8.8$

Iteration-2:- $\theta = \begin{bmatrix}2.133\\8.8\end{bmatrix}$, $y - X\theta = \begin{bmatrix}-4.933\\-18.533\\-38.933\end{bmatrix}$

$\theta_0 = 2.133 - \frac{0.1}{3} \times 2 \times \left((Y - X\theta)^T \cdot \left(-\begin{bmatrix}1\\1\\1\end{bmatrix}\right)\right)$

$= 2.133 - \frac{0.2}{3}(4.933 + 18.533 + 38.9) = 2.133 - 4.933$

$= -2.026$

$\theta_1 = 8.8 - \frac{0.2}{3} \times \left((Y - X\theta)^T \cdot \left(-\begin{bmatrix}1\\3\\6\end{bmatrix}\right)\right)$

$= 8.8 - \frac{0.2}{3} \times \left(4.933 \times (1) + 18.533(3) + 38.933(6)\right)$

$= 8.8 - 19.608$

$= -10.8088$

Iteration-3:- $\Theta = \begin{bmatrix} -2.0266 \\ -10.808 \end{bmatrix}$, $y - X\Theta = \begin{bmatrix} 18.835 \\ 44.453 \\ 82.88 \end{bmatrix}$

$\Theta_0 = -2.0266 - \frac{0.2}{3}\left((y - X\Theta)^T \cdot \left(-\begin{bmatrix}1\\1\\1\end{bmatrix}\right)\right)$

$= -2.0266 - \frac{0.2}{3}\left(-(18.835 + 44.453 + 82.88)\right) = -2.0266 + 9.744$

$= 7.71792$

$\Theta_1 = -10.808 - \frac{0.2}{3}\left((Y - X\Theta)^T \cdot \left(-\begin{bmatrix}1\\3\\6\end{bmatrix}\right)\right)$

$= -10.808 - \frac{0.2}{3}\left(18.835(-1) + 44.453(-3) + 82.88(-6)\right)$

$= -10.808 + 43.29837$

$= 32.4894$

Iteration-4:

$\Theta = \begin{bmatrix} 7.71792 \\ 32.4894 \end{bmatrix}$, $y - X\Theta = \begin{bmatrix} -34.207 \\ -95.1863 \\ -186.6548 \end{bmatrix}$

$\Theta_0 = 7.7179 - \frac{0.2}{3}\left((Y - X\Theta)^T \cdot \left(-\begin{bmatrix}1\\1\\1\end{bmatrix}\right)\right)$

$= 7.7179 - \frac{0.2}{3}\left(34.207 + 95.1863 + 186.6548\right)$

$= 7.7179 - 21.0699 = -13.3519$

$\Theta_1 = 32.4894 - \frac{0.2}{3}\left((Y - X\Theta)^T \cdot \left(-\begin{bmatrix}1\\3\\6\end{bmatrix}\right)\right)$

$= 32.4894 - \frac{0.2}{3}\left(34.207 \times 1 + 95.1863 \times 3 + 186.6548 \times 6\right)$

$= 32.4894 - 95.9796$

$= -63.49021$

Iteration-5

$$\theta = \begin{bmatrix} -13.3519 \\ -63.4902 \end{bmatrix}, \quad Y - X\theta = \begin{bmatrix} 82.8421 \\ 213.8226 \\ 410.2932 \end{bmatrix}$$

$$\theta_0 = -13.3519 - \frac{0.2}{3}\left((Y-X\theta)^T \cdot \left(-\begin{bmatrix}1\\1\\1\end{bmatrix}\right)\right)$$

$$= -13.3519 + \frac{0.2}{3}\left(82.8421 + 213.8226 + 410.2932\right)$$

$$= -13.3519 + 47.1305 = 33.7785$$

$$\theta_1 = -63.4902 - \frac{0.2}{3}\left((Y-X\theta)^T \cdot \left(-\begin{bmatrix}1\\3\\6\end{bmatrix}\right)\right)$$

$$= -63.4902 + \frac{0.2}{3}\left(82.8421(1) + 213.8226 \times 3 + 410.2932 \times 6\right)$$

$$= -63.4902 + 212.4046$$

$$= 148.9144$$

trained theta, after 5-iterations is

$$\theta = \begin{bmatrix} 33.7785 \\ 148.9144 \end{bmatrix}$$

(c) $\theta_1 = \dfrac{\text{cov}(x,y)}{\text{Var}(x)}, \quad \theta_0 = \bar{y} - \theta_1 \bar{x}$

$$\text{Var}(x) = \frac{1}{3}\sum_{i=1}^{3}\left(x_i - \frac{10}{3}\right)^2 = \frac{1}{3}\times\left[\left(1-\frac{10}{3}\right)^2 + \left(3-\frac{10}{3}\right)^2 + \left(6-\frac{10}{3}\right)^2\right]$$

$$= \frac{1}{3}\times\left[\left(\frac{7}{3}\right)^2 + \left(\frac{1}{3}\right)^2 + \left(\frac{8}{3}\right)^2\right] = \frac{1}{27}\times(114) = \frac{114}{27}.$$

$$\bar{Y} = \frac{6+10+16}{3} = \frac{32}{3}.$$

$$\text{Cov}(x,Y) = \frac{1}{3}\sum_{i=1}^{3}\left(x_i - \frac{10}{3}\right)\left(y_i - \frac{32}{3}\right) = \frac{1}{3}\times\left[\left(1-\frac{10}{3}\right)\left(6-\frac{32}{3}\right) + \left(3-\frac{10}{3}\right)\left(10-\frac{32}{3}\right) + \left(6-\frac{10}{3}\right)\left(16-\frac{32}{3}\right)\right]$$

$$\operatorname{Cov}(x,y) = \frac{1}{3}\left[\frac{14\times\overset{7}{\cancel{8}}}{3^2} + \frac{2}{3^2} + \frac{16\times 8}{3^2}\right] = \frac{1}{27}\left[\overset{98}{\cancel{128}} + 80 + 48\right] = \frac{\overset{28}{\cancel{256}}}{27}.$$

$$\Rightarrow \theta_1 = \frac{\operatorname{cov}(x,y)}{\operatorname{Var}(x)} = \frac{\left(\frac{256}{27}\right)}{\left(\frac{114}{27}\right)} = \frac{\overset{28}{\cancel{256}}}{114} = \cancel{2.24562}$$

$$\theta_0 = \frac{32}{3} - \frac{10}{3}\times 2.\cancel{2456} = \frac{\overset{12}{\cancel{9.544}}}{3} = \cancel{3.1813}\ 4.$$

2Ans. Using scikit-learn model on given matrices X and y

a) Here is the code where I applied linear sklearn model to the given data

b) The matrix (X^T*X) is non-invertible because the matrix doesn't have full rank. This happens because the matrix X has linearly dependent documents which makes it non- invertible. Sklearn implementation computes psuedoinverse using standard matrix factorization called Singular value decomposition. This approach is more efficient than computing the normal equation. This approach handles the cases when

- (X^T*X) is non-invertible matrix
- When no.of features>no.of samples
- When some features are redundant

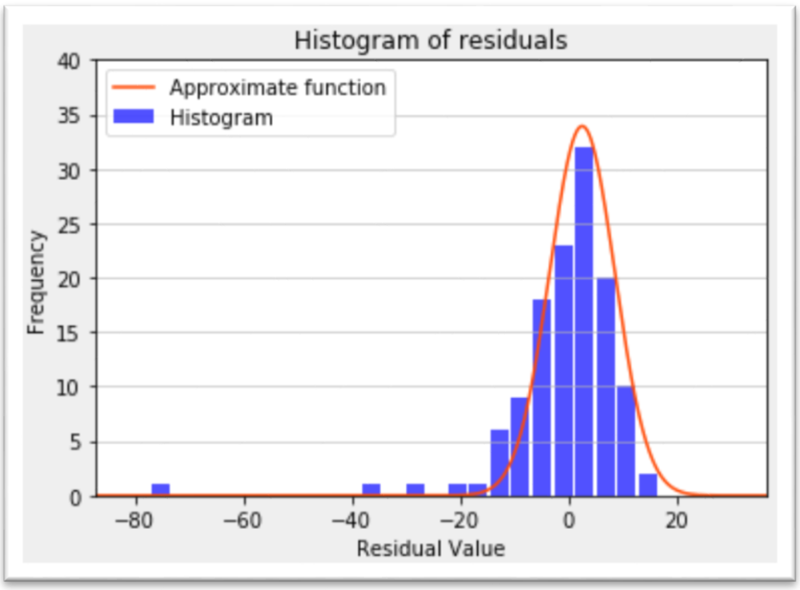
3Ans.

(a) Here is the usage of scikit-learn linear fit on real estate price prediction data. RMS error on test set (where test_set size=0.3) is `10.4315`.

(b) Here are the regression coefficients for the features "X2 house age","X3 distance to the nearest MRT station","X4 number of convenience stores","X5 latitude","X6 longitude" are as follows [-2.29377540e-01 -4.04098382e-03 1.21812654e+00 2.27792859e+02 -5.13529276e+00]. Based on the regression values we can't directly tell the importance of the feature. Also, the values that various features take are not on a same scale, hence interpreting importance of features from trained model coefficients is not a good thing to do.

(c) Once we normalize the feature vectors, since all of them fall into same scale. We can now see that change in one unit per feature will reflect in the output correspondingly on a same scale for all the features. Now considering importance of features based model coefficients is a good parameter

(d) Here is the distribution of the residuals and a fit of approximate function to the residuals.



The approx. function is a normal distribution centred at zero. This indicates that the model has very low bias.

(e) Finding the optimal feature set

(1) Trying all possible feature sets on the validation set, gave ['x2', 'x3', 'x4', 'x5'] as optimal feature set. Here is the code for the same. But in contrast on test set the feature set ['x2', 'x3', 'x4', 'x5', 'x6'] (rms error: 10.4315) performs better when compared to the feature set['x2', 'x3', 'x4', 'x5'] (rms error: 10.4326)

(2) Using greedy approach to find the optimal feature set I got, ['x2', 'x3', 'x4', 'x5'] as optimal feature set. Here is the code for the same. Since it is the same set as found by the exhaustive enumeration, the reported rms error on test set is 10.4326

4Ans.

Part-(a): Here is the code containing the function normalEquationRegressor(X,y) which returns a theta matrix of size d+1. Here *d* is the no.of columns in the input dataframe X.

Part-(b): Here is the code for the gradientDescentRegression to learn co-efficients

Part-(c): Here is the code for the gradientAutogradDescentRegression to learn co-efficients

Part-(d): Here is the code for the gradientDescentRegression using pytorch to learn theta coefficients

Part-(e):

Errors:

Normal equation: 18.255

Gradient descent: 441.98202

Autograd gradient descent: 603.5168

Time taken:

Normal equation : `0.015626907348632812`

Gradient descent: `0.3163888454437256`

Autograd gradient descent: `1.6267085075378418`

5Ans.

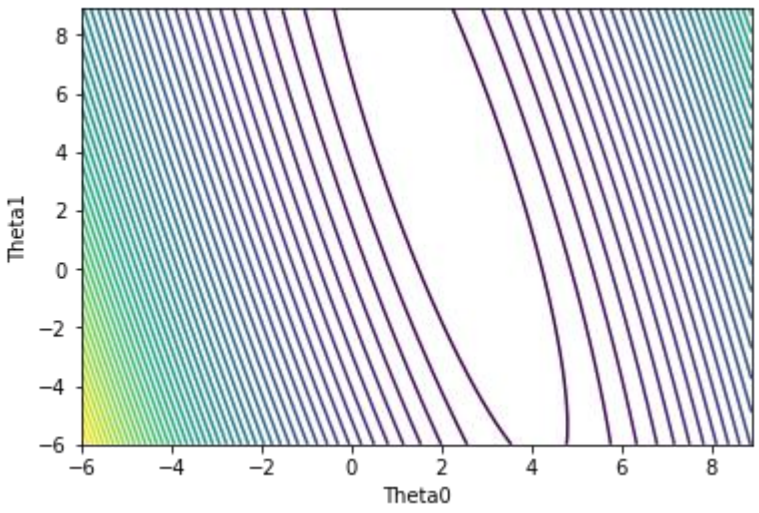
Here is the code for the given polynomial regression using normal equation. But the co-efficients aren't comparable to the given ones because the learning rate is not the efficient one.

6Ans:

Part-(a):

Code for the contour plot

Here is the image of the plot.



References:

- *http://www.adeveloperdiary.com/data-science/how-to-visualize-gradient-descent-using-contour-plot-in-python/*
- *https://www.oreilly.com/library/view/hands-on-machine-learning/9781491962282/ch04.html*