IE406 Machine Learning

Lab Assignment - 1

Group 14

201901466: Miti Purohit
202001430: Aryan Shah

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import pandas as pd
```

Listing 1: Libraries

Question 1

Plot $\theta vs. L(\theta)$, where $L(\theta) = \theta^2$. θ varies from -10 to +10 with a step size of 0.1. Now locate the minimum value of $L(\theta)$ with corresponding θ value from the plot.

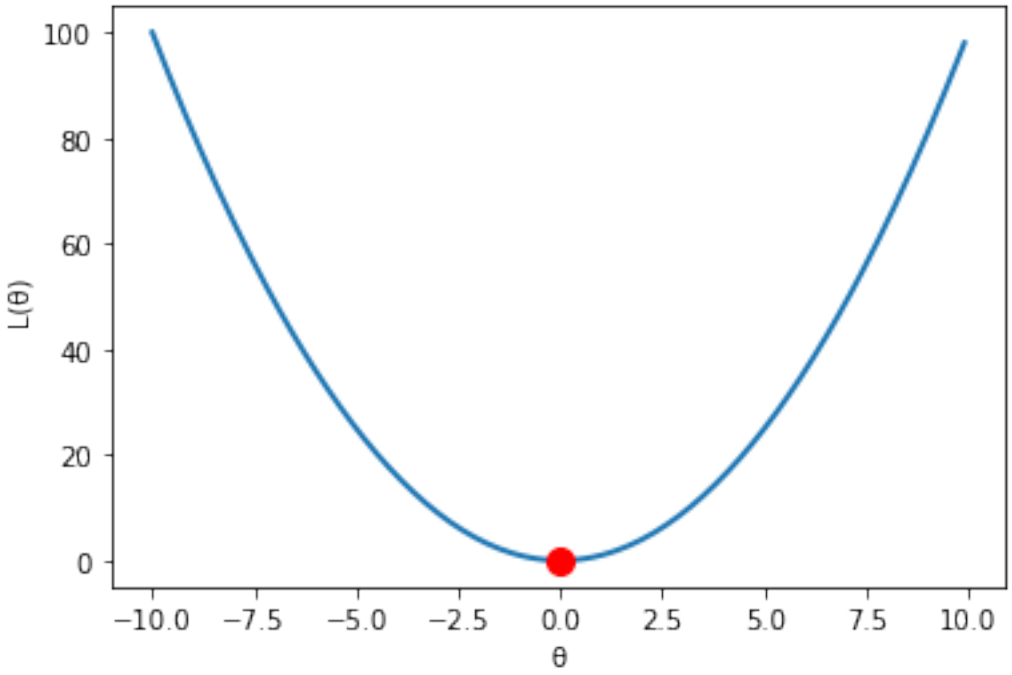
Answer

code

```python
def L(theta):
  return theta**2

theta = np.arange(-10, 10, 0.1)
func_L = L(theta)
plt.figure()
plt.plot(theta, func_L, linewidth=2)

min_func_L = min(func_L)
min_location = np.argwhere(func_L == min_func_L)
#print(min_location)
plt.plot(theta[min_location[0,0]], min_func_L, 'ro', markersize=10)
plt.xlabel('  ')
plt.ylabel('L(  )')
plt.show()
```

Listing 2: Question 1

Result



Observation/ Justification

The minimum value of L(θ) is 0.0 at 0.0

Question 2

Plot $\theta \, vs. \, L(\theta)$, where $L(\theta) = {\theta_1}^2 + {\theta_2}^2$. θ_1 and θ_2 vary from -10 to +10 with a step size of 0.1. Locate the minimum value of $L(\theta_1, \theta_2)$ with corresponding θ_1, θ_2 values from the plot.

Answer

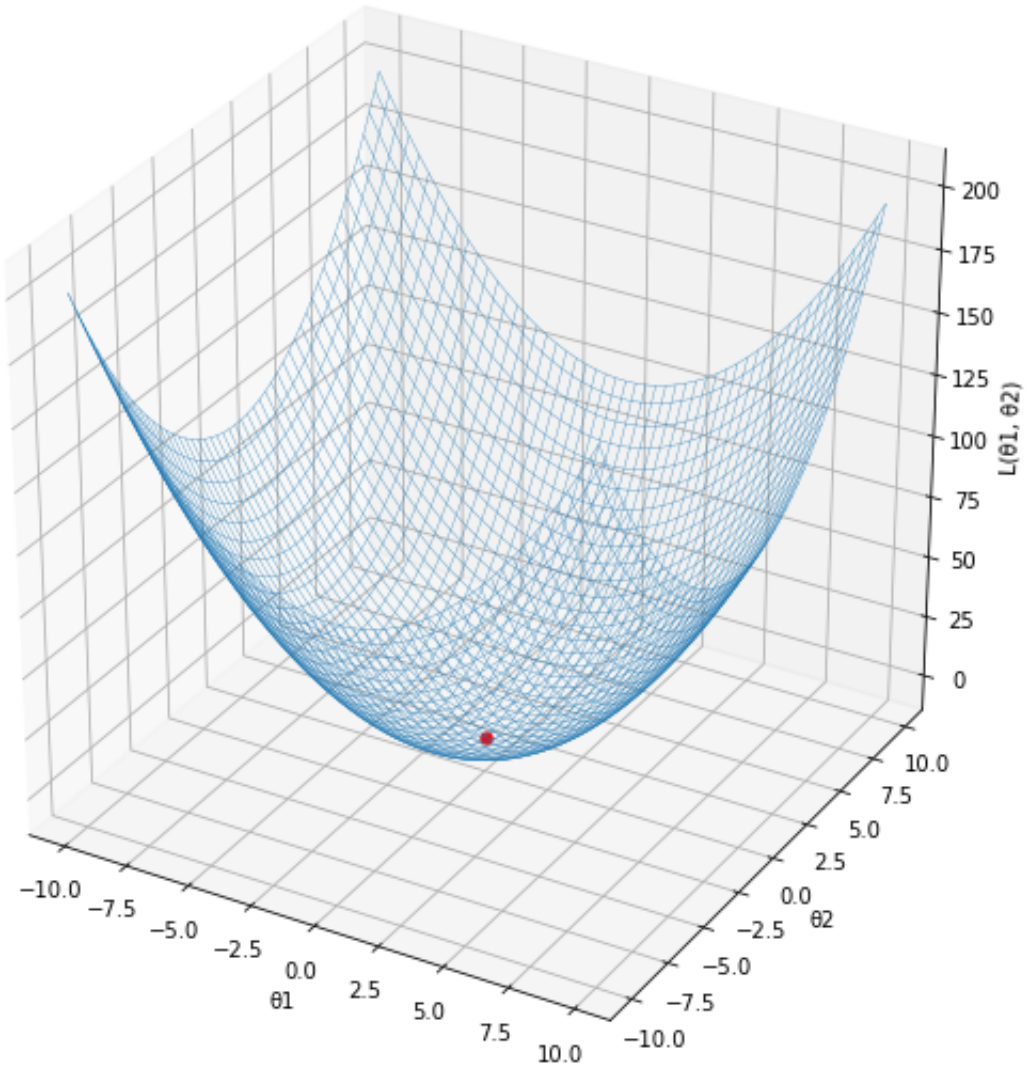
code

```
ddef L(theta1, theta2):
    return theta1**2 + theta2**2

theta1 = np.arange(-10,10,0.1)
theta2 = np.arange(-10,10,0.1)

func_L = np.zeros((len(theta1),len(theta2)))
for i in range(len(theta1)):
    for j in range(len(theta2)):
        func_L[i][j] = L(theta1[i], theta2[j])

X, Y = np.meshgrid(theta1, theta2)
z = L(X, Y)
fig = plt.figure(figsize=(10, 10))
ax = plt.axes(projection='3d')
ax.plot_wireframe(X, Y, z, linewidth=0.3)


min_func_L = np.min(func_L)
#print(min_func_L)
min_loc = np.argwhere(func_L == min_func_L)
#print(z[100,100])
x_coord = min_loc[0,0]
y_coord = min_loc[0,1]
min_value = z[x_coord, y_coord]
#print(min_value)
ax.scatter(theta1[x_coord], theta2[y_coord], min_value, marker="o", c="red", s=28)
```

```
ax.set_xlabel(' 1 ')
ax.set_ylabel(' 2 ')
ax.set_zlabel('L( 1 ,  2 )')
plt.show()
```

Listing 3: Question 2

Result



Observation/ Justification

The minimum value of L(θ) is — at —-

Question 3(a)

Plot for $L(\theta) = \Sigma_{i=1}^{m}\big(y(i)–(\theta_0 + \theta_1.x(i))\big)^2$,where m is the number of input examples and x(i), y(i) are the values taken from the given data file. Obtain the minimum value of $L(\theta)$ with corresponding θ_0, θ_1 values from the plot.

Answer

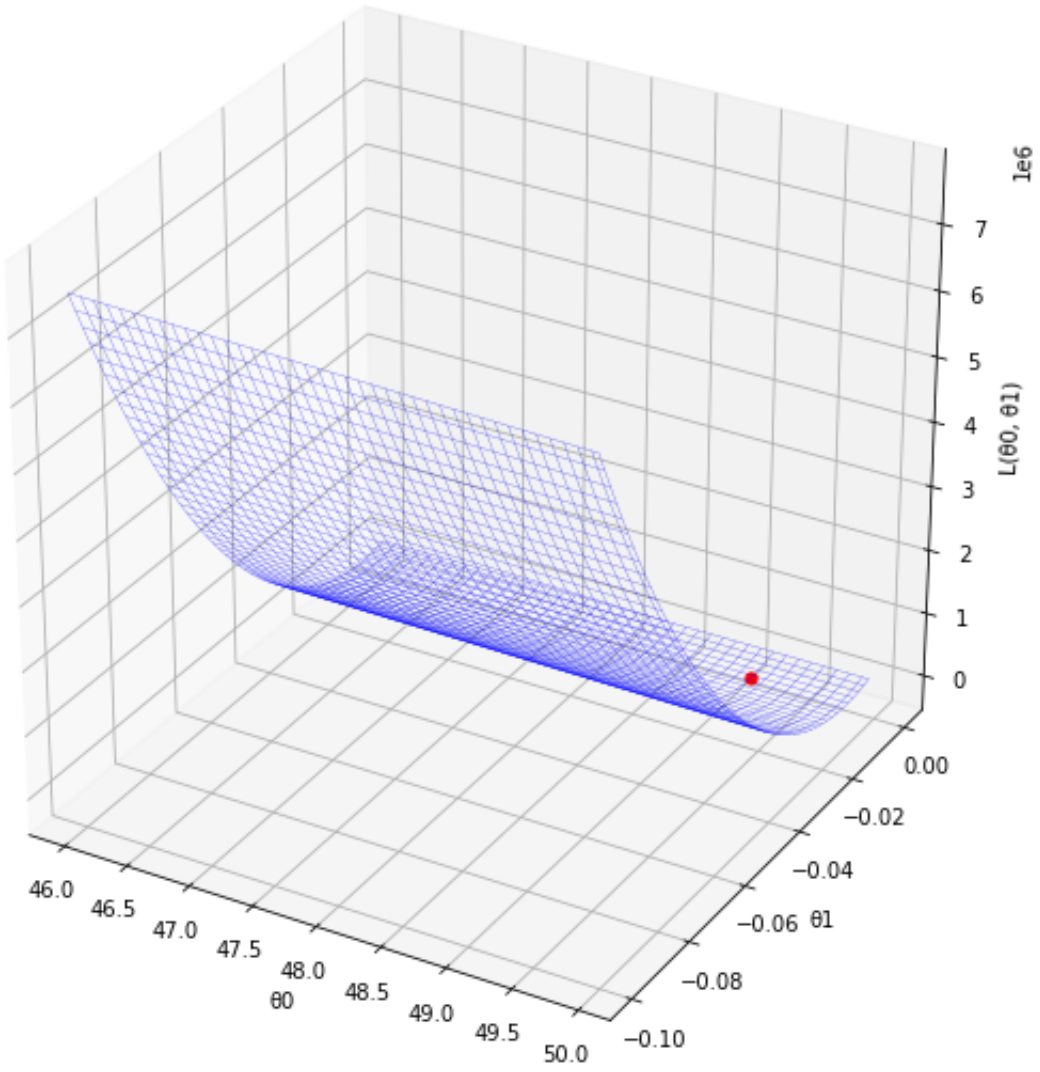
code

```
def L(theta):
  return theta**2

theta = np.arange(-10, 10, 0.1)
func_L = L(theta)
plt.figure()
plt.plot(theta, func_L, linewidth=2)

min_func_L = min(func_L)
min_location = np.argwhere(func_L == min_func_L)
#print(min_location)
plt.plot(theta[min_location[0,0]], min_func_L, 'ro', markersize=10)
plt.xlabel('  ')
plt.ylabel('L(  )')
plt.show()
```

Listing 4: Question 3(a)

Result

OUTPUT:
Minimum value of $L(\theta 0, \theta 1)$ is 1572.6593617600001 $\theta 0 and \theta 1$ values for minimum value of $L(\theta 0, \theta 1)$ are 49.19999999999982 -0.008599999999997388 respectively.



Observation/ Justification

Question 3(b)

Apply Pseudo Inverse (Least Squares (LS)) approach to get θ vector for the cost function (objective function) $L(\theta)$ given in example 3(a). Verify whether θ_0, θ_1 obtained are the same as that found in Q3(a).

Answer

code

```
temp = np.copy(data_x)
x = np.ones((np.size(temp,0),2))
x[:,1] = temp[:,0]

print('determinant of x.T*x = ',np.linalg.det(np.matmul(x.T,x)))

theta = np.matmul(np.linalg.inv(np.matmul(x.T,x)),np.matmul(x.T,data_y))
print('For minima, theta0 = ',theta[0,0],' and theta1 = ',theta[1,0])
```

Listing 5: Question 3(b)

Result

OUTPUT:
determinant of x.T*x = 5618077959.999997 For minima, $\theta 0$ = 49.23762989433493 and $\theta 1$ = -0.008611934783475328

Observation/ Justification

Question 4

Calculate the value of $L(\theta)$ using the θ vector obtained by Pseudo Inverse (as done in Q3(b)). Now Assume any θ vector (other than the one obtained in Q3(b)) and compute the new $L(\theta)$ value. Comment on why the Pseudo Inverse is also called LS method.

Answer

code

```
LS = np.sum(np.square(data_y - (theta[0,0] + (theta[1,0]*temp))))
print('Value of L using theta values from LS method(Pseudo Inverse) = ',LS)
print()
for i in range(10):
    L_other_value = np.sum(np.square(data_y - (np.random.randint(theta0.shape[0]) - (np.random
    .randint(theta1.shape[0])*temp))))
    print('Value of L for some other theta = ',L_other_value)
```

Listing 6: Question 3(b)

Result

OUTPUT:
Value of L using theta values from LS method(Pseudo Inverse) = 1572.6503668922921
Value of L for some other theta = 2484947822720726.0 Value of L for some other theta = 269611008855161.75
Value of L for some other theta = 1140041054488897.8 Value of L for some other theta = 259697580389616.75
Value of L for some other theta = 1422578963889964.8 Value of L for some other theta = 12106098743706.75
Value of L for some other theta = 1385272296102789.8 Value of L for some other theta = 2036913111196393.8
Value of L for some other theta = 2838967870749331.0 Value of L for some other theta = 1151197388894299.8

Observation/ Justification

It can be observed that the value $L(\theta)$ for $\theta_0 and \theta_1$ found in the 4th Question gives a lesser value that other values found. Thus, it can be said that the Pseudo Inverse method gives the least values for the given function. Hence this is the reason why Pseudo Inverse is also called the Least Squares method.

Question 5(a)

For the following X and Y, use scikit-learn to learn a linear model.

$$X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} Y = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

Answer

code

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

x = np.array([
        [1, 2],
        [2, 4],
        [3, 6],
        [4, 8]
    ])
y = np.array([
        [2],
        [3],
        [4],
        [5]
    ])

reg = LinearRegression().fit(x,y)
print('Score: ', reg.score(x, y))
print('Coefficient: ', reg.coef_)
print('Intercept: ', reg.intercept_)
```

Listing 7: Question 5(a)

Result

Score: 1.0 Coefficient: [[0.2 0.4]] Intercept: [1.]

Observation/ Justification

Question 5(b)

Solve the problem using normal equations. You may find that one of the matrices in the normal equation is non-invertible. Why does the matrix turn out to be non-invertible? Why can scikit-learn implementation still correctly solve this regression problem?

Answer

Observation/ Justification

We got Singular Matrix Error because det(X'X) = 0. Even though, scikit learn can solve the model as it uses pseudo inverse instead of normal inverse. So, if we use pinv fucntion instead of inv function, we can still get the correct results.

Question 6(a)

Show the usage of scikit-learn's linear regression module for the real estate price prediction regression problem. What is the RMS error on the test set?

Answer

code

```
real_estate_data = pd.read_excel('/content/sample_data/real_estate_valuation_dataset.xlsx')
#real_estate_data
x1 = np.reshape(np.array(real_estate_data['X1 transaction date']), (-1, 1))
x2 = np.reshape(np.array(real_estate_data['X2 house age']), (-1, 1))
x3 = np.reshape(np.array(real_estate_data['X3 distance to the nearest MRT station']), (-1, 1))
x4 = np.reshape(np.array(real_estate_data['X4 number of convenience stores']), (-1, 1))
x5 = np.reshape(np.array(real_estate_data['X5 latitude']), (-1, 1))
x6 = np.reshape(np.array(real_estate_data['X6 longitude']), (-1, 1))
X = np.hstack((x1, x2, x3, x4, x5, x6))
y = np.reshape(np.array(real_estate_data['Y house price of unit area']), (-1, 1))

reg = LinearRegression().fit(X, y)
print('Score: ', reg.score(X, y))
print('Coefficient: ', reg.coef_)
print('Intercept: ', reg.intercept_)

y_pred1 = reg.predict(X)

print('RMSE ->', np.sqrt(mean_squared_error(y, y_pred1)))
```

Listing 8: Question 6(a)

Result

OUTPUT:
Score: 0.5823850447850936 Coefficient: [[5.14901721e+00 -2.69696735e-01 -4.48750825e-03 1.13332498e+00 2.25470143e+02 -1.24290612e+01]] Intercept: [-14441.98271918] RMSE -¿ 8.78231297536097

Observation/ Justification

Question 6(b)

Based on the regression coefficients, what can you comment about the importance of different features? Is it correct to assume that larger coefficients mean more important features?

Answer

Observation/ Justification

We know that the sign of a regression coefficient indicates whether there is a positive of negative correlation between each dependent variable and independent variable. A positive coefficient suggests that as the value of the independent variable increases, the mean of the dependent variable also tends to increase. A negative coefficient indicates that as the independent variable increases, the dependent variable tends to decrease.

The coefficient value signifies how much the mean of the dependent variable changes when the independent variable is shifted by one-unit while the other variables in the model are kept constant. By keeping the other variables constant it allows us to assess the effect of each variable in isolation from the others.

Although since the X values are not standardized, we cannot come to a conclusion on the importance of a feature by just comparing the coefficients.

Question 6(c)

Now, standardize the dataset to have all features on a scale of 0 to 1. Re-learn the regression coefficients and now comment on the importance of different features.

Answer

code

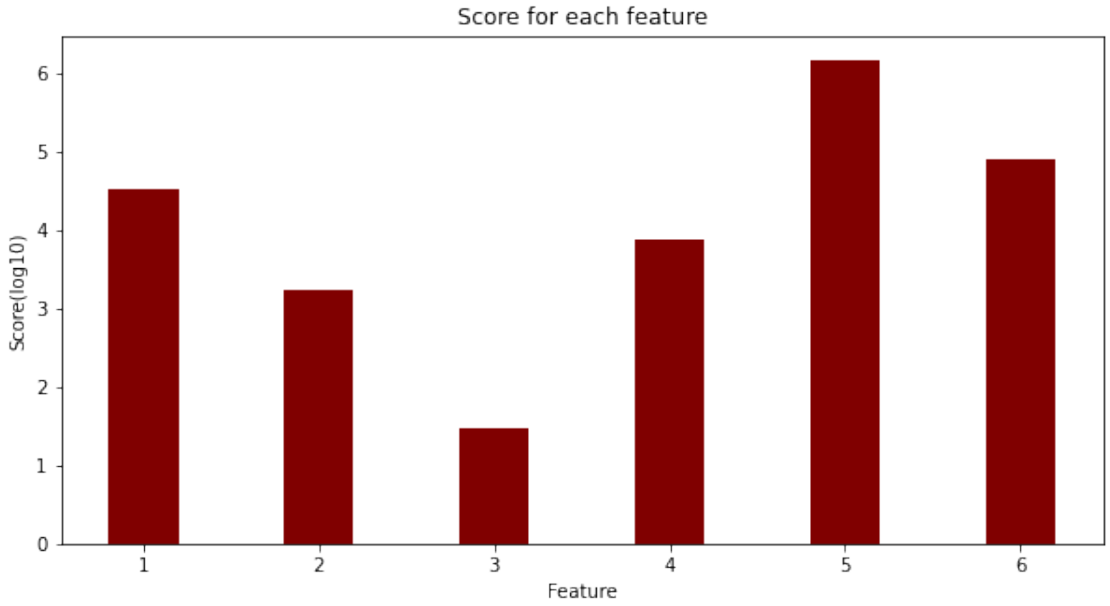
```
X_norm = (X - X.min())/(X.max()-X.min())
reg = LinearRegression().fit(X_norm, y)
print('Score: ', reg.score(X_norm, y))
print('Coefficient: ', reg.coef_)
print('Intercept: ', reg.intercept_)

y_pred2 = reg.predict(X_norm)

print('RMSE ->', np.sqrt(mean_squared_error(y, y_pred2)))
```

```
10
11 # creating bar graph
12 fig = plt.figure(figsize = (10, 5))
13 plt.bar([1,2,3,4,5,6], np.log10(abs(reg.coef_[0])), color ='maroon',
14         width = 0.4)
15
16 plt.xlabel("Feature")
17 plt.ylabel("Score(log10)")
18 plt.title("Score for each feature")
19 plt.show()
```

Listing 9: Question 3(b)

Result

OUTPUT:
Score: 0.5823850447850758 Coefficient: [[3.34069318e+04 -1.74979808e+03 -2.91150478e+01 7.35303628e+03 1.46285502e+06 -8.06400099e+04]] Intercept: [-14441.98271918] RMSE = 8.782312975361156



Observation/ Justification

Adding on to the observations made in 6(a), we can now say that after being normalized, the coefficients can be used as a measure of the importance of each feature w.r.t. the dependent variable. As seen in the graph, the 5th and 6th features are the most important ones, i.e., the latitude and longitude of the house matters the most. Moreover, the 3rd feature is the least important.

Question 6(e)

Answer

code

```
1 X = np.hstack((x1, x2, x3, x4, x5))
2 y = np.reshape(np.array(real_estate_data['Y house price of unit area']), (-1, 1))
3
4 reg = LinearRegression().fit(X, y)
5 print('Score: ', reg.score(X, y))
6 print('Coefficient: ', reg.coef_)
7 print('Intercept: ', reg.intercept_)
8
9 y_pred1 = reg.predict(X)
10
11 print('RMSE ->', np.sqrt(mean_squared_error(y, y_pred1)))
```

Listing 10: Question 6e)

Result

OUTPUT:
Score: 0.5823178829759132 Coefficient: [[5.13755503e+00 -2.69380517e-01 -4.35333829e-03 1.13619277e+00 2.26879404e+02]] Intercept: [-15964.80386271] RMSE -¿ 8.783019142977878

Observation/ Justification

Upon removing the 6th feature, the error reduces. Thus the 6th feature is not that important. Similarly we can try for other features too and we will get different results for different features.