IE 406 - ML

Lab 1

Chirayu Chaplot (201801038)
Dhanvi Shah (201801167)
Rutwa Rami (201801205)
Shabbir Murtaza (201801428)
Dishita Thaker (201801442)

September 4, 2021

Contents

1 Question 1

1.1 Problem Statement

The aim of the question is to plot θ^2 vs θ and subsequently find the minimum value of the function with the corresponding value of θ.

1.2 Code

```
import matplotlib.pyplot as plt
import numpy as np

x_coord = np.arange(-10,10,0.1) - step size of 0.1
y_coord = x_coord*x_coord

min_index = np.argmin(y_coord) - getting min L(θ)

figure, ax = plt.subplots()
ax.plot(x_coord,y_coord)
plt.annotate('Min value', xy = (x_coord[min_index], y_coord[min_index]),arrowprops = dict(facecolor
='red',shrink = 0.05))
plt.title('Plot of θ VS L(θ)')
plt.xlabel('θ')
plt.ylabel('L(θ)')
plt.show()
print("Minimum Value of L(θ) = " ,y_coord[min_index])
```

Following is the figure obtained for the same:

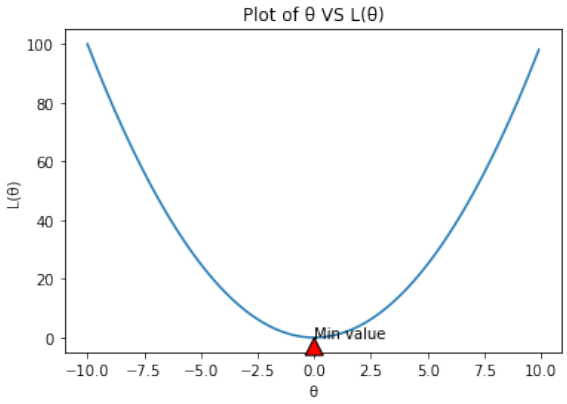

Figure 1: The graph of θ^2 vs θ with the minimum value for θ^2 marked along with the corresponding value of θ

1.3 Observation

As we can see that the function is an open parabola, which is open upwards owing to a positive coefficient of θ^2. The minimum value of the same is when θ is almost 0. This result will match to a standard derivative, maxima-minima analysis that is done to the function with respect to θ.

2 Question 2

2.1 Problem Statement

The aim of the question is to plot $f(\theta) = \theta_1^2 + \theta_2^2$ vs θ_1 and θ_2 and subsequently find the minimum value of the function and corresponding values of the variables.

2.2 Code

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d

theta1=np.arange(-10,10,0.1)
theta2=np.arange(-10,10,0.1)
x,y=np.meshgrid(theta1,theta2,sparse=False,indexing='xy')

z=np.zeros((len(theta1),len(theta1))) - initializing L(θ) with zeroes

min_value = 250
for i in range(len(theta1)):
for j in range(len(theta2)):
z[i,j]=theta1[i]**2+theta2[j]**2 -generating L(θ)
if min_value > z[i,j]: getting minimum L(θ)
min_value = z[i,j]
min_theta1 = theta1[i] minimum θ1
min_theta2 = theta2[j] minimum θ2

print("Min value of L(θ) = ", min_value)
print("Min value of θ1 = ",min_theta1)
print("Min value of θ2 = ",min_theta2)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x,y,z)

ax.set_zlabel(r"L(θ)", fontsize=15)
ax.set_xlabel(r"θ1", fontsize=15)
ax.set_ylabel(r"θ2", fontsize=15)
plt.title('Plot of θ vs L(θ1,θ2)')
plt.show()
```

Following is the plot obtained:

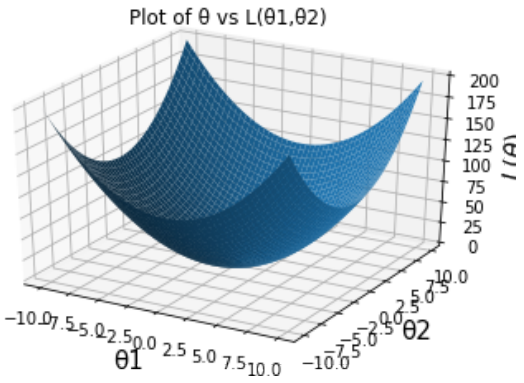

Figure 2: The graph of $\theta_1^2 + \theta_2^2$ vs θ with the minimum value for $\theta_1^2 + \theta_2^2$

2.3 Observation

We can observe that the minimum value of the function is almost equal to 0 and it occurs when $\theta_1 = \theta_2$ which is almost 0. This can be obtained analytically by solving the equations:

1. $df/d\theta_1 = 0$
2. $df/d\theta_2 = 0$

3 Question 3

3.1 Problem Statement

The aim of the question is to plot $L(\theta) = \sum_{i=1}^{m}(y(i) - (\theta_0 + \theta_1 * x(i)))^2$ where; x(i) and y(i) are to be read from a data file and m is the number of input values, and find the minimum value of $L(\theta)$ along with the corresponding values of θ_0 and θ_1.

3.2 Code

3.2.1 3a

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

df=pd.read_csv('https://raw.githubusercontent.com/shabbirmur/shabbirmur/main/DataLAB1.csv')- **reading file**

x=np.array(df['x']) - **getting col x**
y=np.array(df['y']) - **getting col y**

x = (x - np.mean(x)) / np.std(x) - **a z-score gives you an idea of how far from the mean a data point is**
t1 = np.arange(-30, 30, 0.1)
t2 = np.arange(-30, 30, 0.1)
z = np.zeros((len(t1), len(t2))) - **initializing** min_Ltheta = 1000000000000000000
mintheta1 = 0.0
mintheta2 = 0.0

for i in range(len(t1)):
for j in range(len(t2)):
temp = 0.0
for k in range(x.shape[0]):
temp += (y[k] - t1[i] - t2[j] * x[k]) ** 2
z[i, j] = temp
if temp < min_Ltheta:
min_Ltheta = temp
mintheta1 = t1[i]
mintheta2 = t2[j]

print('Min value of θ0 for which L(θ) is minimum = ' ,mintheta1)
print('Min value of θ1 for which L(θ) is minimum = ' ,mintheta2)
print('Min value of L(θ) = ' ,min_Ltheta)
fig = plt.figure()

```
ax = fig.add_subplot(111, projection = '3d')
ax.plot_surface(t1, t2, z, cmap = 'viridis')
plt.title('Plot of θ vs L(θ0,θ1)')
ax.set_xlabel(r'L(θ0)', fontsize = 15)
ax.set_ylabel(r'L(θ1)', fontsize = 15)
ax.set_zlabel(r'L(θ0,θ1)', fontsize = 15)
plt.show()
```

Following is the plot obtained:

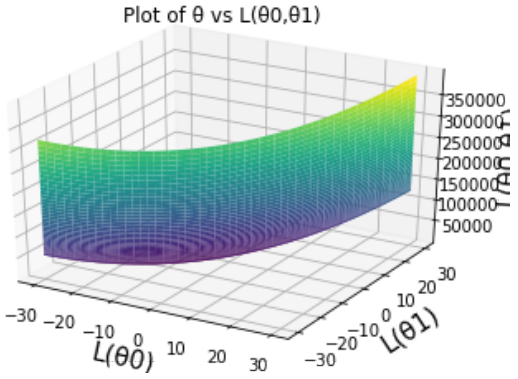

Figure 3: The graph of $L(\theta) = \sum_{i=1}^{m}(y(i) - (\theta_0 + \theta_1 * x(i)))^2$ vs θ_0 and θ_1

3.2.2 3b

```
col = np.ones([len(x),1])
X = np.c_[col,x]
X_T_X = np.matmul(X.T,X) - Multiplying matrix and its transpose
X_INV = np.linalg.inv(X_T_X) - Calculating inverse of the above
X_INV _X = np.matmul(X_INV,X.T)
pseuedo_inverse = np.matmul(X_INV_X,y)
print(pseuedo_inverse)
```

3.3 Observation

We can see that the minimum value of $L(\theta)$ is 1572.7835058203152
The value is obtained at θ_0 being 23.700000000000763 and θ_1 being -6.899999999999672.
The technique of feature scaling(standardisation precisely) has been applied to ensure x falls in between -1 and 1 so that fallacy in the results can be avoided.

3.3.1 3(b)

On applying the pseudo inverse method, to the objective function and the value obtained for θ_0 is 23.71808511, for θ_1 is -6.86699545. They are almost same as compared with the ones obtained in question 3(a).

4 Question 4

4.1 Problem Statement

The aim of the question is to find the value of $L(\theta)$ using the θ vector obtained in 3(b), and subsequently finding the value of $L(\theta)$ for an arbitrary vector.

4.2 Code

x_theta = np.matmul(X,pseuedo_inverse)
L_theta = np.sum((y - x_theta)**2) - **computing L(θ) using θ from pseudo inverse**
print('L(theta) = ' , L_theta)
theta_1 = [9,11] - **initializing 1X2 arbitrary vector**
x_theta2 = np.matmul(X,theta_1)
L_theta2 = np.sum((y - x_theta2)**2)
print('new L(theta) = ', L_theta2)

4.3 Observation

L(θ) = 1572.6503668922917 is the value obtained by putting the vector obtained in 3(b). When [9,11] was put, the value obtained was: L(θ) = 51942.696584924124.
The Pseudo Inverse method is also called the least square method because it particularly aims at reducing the squared error for obtaining coefficients in linear regression. It can be done by equating the gradient of L(θ) to 0. However, the matrix representation of the operation is the same as that of the pseudo-inverse. Hence, we can understand why in the context of Linear Regression, pseudo inverse is called the LS "least squares" method.

5 Question 5

5.1 Problem Statement

The aim is to implement sci-kit learn on X and Y to understand building of a liner model. Later, solve the matrices using equations, realise the difficulty and comprehend why ski-cit learn implementation would still solve the regression problem.

5.2 Code

5.2.1 Q5(a)

import numpy as np
from sklearn.linear_model import LinearRegression
X = np.array([[1,2],[2,4],[3,6],[4,8]])
Y = np.array([2,3,4,5])
reg = LinearRegression()
reg.fit(X,Y)
print('Rank of matrix X is :',reg.rank_) - **gives the rank of matrix X**
print('Estimated coefficients of Linear Regression is :',reg.coef_) **-gives the estimated coefficients for the linear regression problem**

5.2.2 Q5(b)

We have shown that $X * X^T$ is not invertible while solving for normal equation.

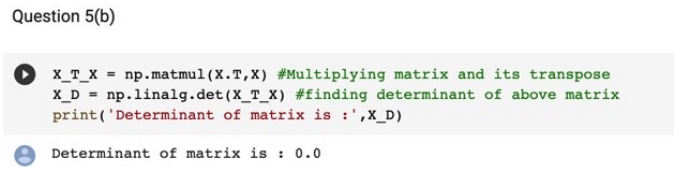

Figure 4: The determinant is 0 for the matrix and hence non-invertible

5.3 Observation

The rank of the matrix obtained is 1, the coefficients for linear regression obtained are 0.2,0.4.
On trying to implement via normal equation, we understand that matrix $A.A^T$ is not invertible. The normal equation $\theta = ((X^T X)^{-1} X^T * Y)$ is obtained with an aim of minimising the objective function, and when equating the gradient of the objective function to 0. When we try to find the XX^T, we get a matrix that has two rows that are linearly dependent on each other as a result of which the matrix becomes non-invertible. It particularly means that θ would not have a unique solution.

However, one can use scikit-learn library which accurately solves the problem. This can be proved by the answer obtained in part 5(a). The sklearn library does not solve for normal equations, hence

non-invertiblity criterion is not a hurdle here. It tries to tackle such rank-deficit matrices by solving for θ to minimise the $|X * \theta - Y|$ by using various other techniques.
We know that the fact that our matrix is rank deficit won't allow for θ to have a unique solution. Sklearn will return one of the possible solutions that minimise the 2 norm of $|X * \theta - Y|$. It uses various techniques for the same, is a wrapper around scipy.linalg.lstsq which is particularly used to find generalised inverse by performing specific minimisation.

6 Question 6

6.1 Problem Statement

6.2 Code

6.2.1 6a

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn import preprocessing
from sklearn.model_selection import KFold
from itertools import combinations

from google.colab import drive
drive.mount('/content/drive')

df = pd.read_excel('/content/drive/My Drive/Realestatevaluationdataset.xlsx')
df.head()

linear_reg = LinearRegression()
col = list(df.columns) - putting the col names in col
# print(col)
# print(df.columns)
col.remove('Y house price of unit area') - removing column to get the training set

y = df['Y house price of unit area'] used for testing
x = df[col] used for training

x_train, x_test, y_train, y_test = train_test_split(x,y,random_state = 42)
- train_test_split basically divides the data set into two subsets in which model trains
the data on one subset and test on another subset.
-random_state = 42 is used to produce the same result across all runs of the code
-x_train will have 1st subset of data from x
-x_test will have 2nd subset of data from x
-y_train will have 1st subset of data from y
-y_test will have 2nd subset of data from y
# print(x_train) linear_reg.fit(x_train,y_train) fit linear model

prediction = linear_reg.predict(x_test) - predict linear model
# print(prediciton)
print('The RMS Error is',np.sqrt(mean_squared_error(prediction,y_test)))
```

6.2.2 6c

linear_new = LinearRegression() - **using different regression model**
x_train_new = preprocessing.scale(x_train) - **scaling the data brings all the values onto one scale eliminating the sparsity**
# print(x_train_new)
x_test_new = preprocessing.scale(x_test)
# print(x_test_new)
linear_new.fit(x_train_new,y_train) - **fitting lienar model with new values**
new_predicted = linear_new.predict(x_test_new)
# print(new_predicted)
print('The RMSE is',np.sqrt(mean_squared_error(new_predicted,y_test)))

6.2.3 6d

Following is the screenshot of the code for Question 6(d):

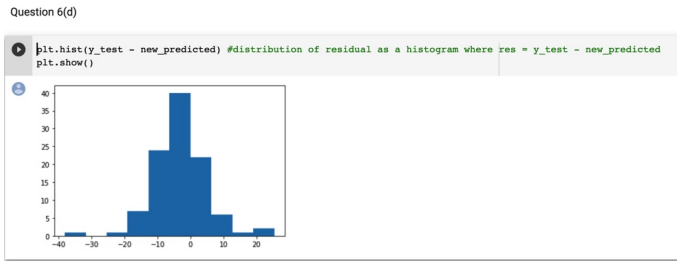

Figure 5: Code for Question 6(d)

6.2.4 6e

import seaborn as sns
plt.figure(figsize=(12,10))
cor = x_train.corr()
sns.heatmap(cor,annot=True,cmap=plt.cm.CMRmap_r)
plt.show()

correlation function
def correlation(RealEstatedataset, threshold) :
col_correlation = set() - **calculates correlation matrix for the dataset provided as input (2D correlation matrix)**
correlation_matrix = RealEstatedataset.corr()
for i in range(len(correlation_matrix.col)) : - **checks for every row in correlation matrix**
for j in range(i) : - **checks for every column in a particular row**
if abs(correlation_matrix.iloc[i,j]) > threshold : - **checks if the value in t**
columnName = correlation_matrix.col[i]
col_correlation.add(columnName)
return col_correlation

```
    remove_cols = correlation(x_train,0.7)

x_train.drop(remove_cols,axis=1)
x_test.drop(remove_cols,axis=1)

linear_regression = LinearRegression()
x_train_new = preprocessing.scale(x_train)
x_test_new = preprocessing.scale(x_test)
linear_regression.fit(x_train_new,y_train)
score = linear_regression.predict(x_test_new)
print('The R2 score value is',np.sqrt(mean_squared_error(pr,y_test)))
```

6.3 Observation

For this problem, we first read the data and applied the train-test technique(also suggested as per the code and the comments as stated above). The RMS error on the test set obtained is 8.250654190112902.
With respect to the regression coefficients, they basically indicate the extent to which the independent variable will affect the dependent variables holding the other independent variables constant. While the sign(positive or negative) shows the nature of co-relation. However, it is not right to assume that larger coefficients imply more important features because all independent variables may have varying different scales and hence comparing them in an absolute sense won't be correct.
Next, we scaled the features to eliminate the sparsity and hence obtained RMSE is 8.688712721416513.Yet, Regression coefficients do not help us infer correctly the importance of features because even after normalising they don't indicate/cover anything about statistical measures like variance, one feature with a larger coefficient might still not speak much about the dependent variable and is just a relative measure for correlation.
The distribution of residuals is as obtained in the Histogram below. As we can observe, the histogram suggests it is a Gaussian distribution.

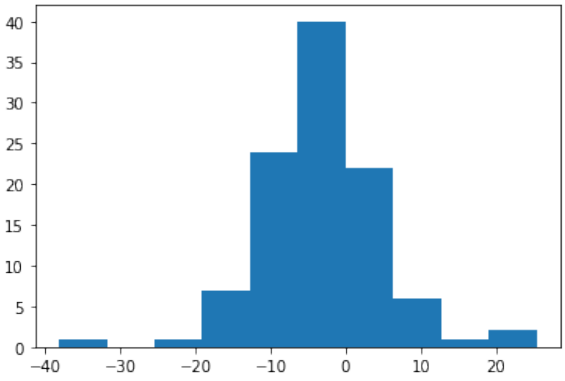

Figure 6: Distribution of residuals