San Francisco Bay University

CS483 - Fundamentals of Artificial Intelligence
Homework Assignment #1
Due day: 5/28/2022

1. Given a training set of patient records of Systolic Blood Pressure (SBP) regarding **two** features, such as *age* and *weight*, please build up linear regression hypothesis function/loss function /cost function, and then write python program to implement this algorithm by gradient descent method. After hypothesis function training through training set, predict new patient's SBP

Patient's ID	Age(Years)	Weight(Kg)	SBP(mm Hg)
1	60	58	117
2	61	90	120
3	74	96	145
4	57	72	129
5	63	62	132
6	68	79	130
7	66	69	110
8	77	96	163
9	63	96	136
10	54	54	115
11	63	67	118
12	76	99	132
13	60	74	111
14	61	73	112
15	65	85	?
16	79	80	?

*Note: Predict ? value based on training result

Solution:

Feature scaling x1:

Range = max – min = 79-54=25

Mean = $\frac{\sum_{i=1}^{N} x1^{(i)}}{N} = \frac{903}{14} = 64.5$

Scale $(x1^{(i)}) = \frac{|x1^{(i)} - mean|}{range}$ = [0.196, 0.152, 0.413, 0.326, 0.065, 0.152, 0.065, 0.543, 0.065, 0.457, 0.065, 0.5, 0.196, 0.152]

Feature scaling x2:

Range = max – min = 99-54=45

Mean = $\frac{\sum_{i=1}^{N} x2^{(i)}}{N} = \frac{1085}{14} = 77.5$

Scale $(x2^{(i)}) = \frac{|x2^{(i)} - mean|}{range}$ = [0.433,0.278,0.411,0.122,0.344,0.033,0.189,0.411,0.411, 0.522,0.233, 0.478, 0.078, 0.100]

ID(i)	Normalized Age	Normalized Weight
1	0.196	0.433
2	0.152	0.278
3	0.413	0.411
4	0.326	0.122
5	0.065	0.344
6	0.152	0.033
7	0.065	0.189
8	0.543	0.411
9	0.065	0.411
10	0.457	0.522
11	0.065	0.233
12	0.5	0.478
13	0.196	0.078
14	0.152	0.1

Normalized features value

Hypothesis function: $h(\theta) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

; where x_1 and x_2 are the features age, and weight respectively

Loss function: $l(\theta) = [\, h(x^{(i)}) - y^{(i)}]^2$

Cost function: $j(\theta) = \frac{1}{2*N}\sum_{i=1}^{N}[\, h(x^{(i)}) - y^{(i)}]^2 = \frac{1}{2*N}\sum_{i=1}^{N}[\, (\theta_0 + \theta_1 x_1 + \theta_2 x_2) - y^{(i)}]^2$

; where N = 14 (training set)

Minimum values of cost function:

$$\frac{\partial j(\theta)}{\partial \theta_0} = \frac{2}{2*N}\sum_{i=1}^{N}[\, (\theta_0 + \theta_1 x_1 + \theta_2 x_2) - y^{(i)}]*1 = \frac{1}{N}\sum_{i=1}^{N}[\, (\theta_0 + \theta_1 x_1 + \theta_2 x_2) - y^{(i)}]$$

$$\frac{\partial j(\theta)}{\partial \theta_1} = \frac{2}{2*N}\sum_{i=1}^{N}[\, (\theta_0 + \theta_1 x_1 + \theta_2 x_2) - y^{(i)}]*x_1 = \frac{1}{N}\sum_{i=1}^{N}[\, (\theta_0 + \theta_1 x_1 + \theta_2 x_2) - y^{(i)}]*x_1$$

$$\frac{\partial j(\theta)}{\partial \theta_2} = \frac{2}{2*N}\sum_{i=1}^{N}[\, (\theta_0 + \theta_1 x_1 + \theta_2 x_2) - y^{(i)}]*x_2 = \frac{1}{N}\sum_{i=1}^{N}[\, (\theta_0 + \theta_1 x_1 + \theta_2 x_2) - y^{(i)}]*x_2$$

Obtain the **hypothesis function coefficients** using the **gradient descent method**

$$\theta_0 := \theta_0 - \propto \left(\frac{\partial j(\theta)}{\partial \theta_0}\right)$$

$$\theta_1 := \theta_1 - \propto \left(\frac{\partial j(\theta)}{\partial \theta_1}\right)$$

$$\theta_2 := \theta_2 - \propto \left(\frac{\partial j(\theta)}{\partial \theta_2}\right)$$

Code:

```
# Assignment Q1: Linear Regression
# Patients data
# function that returns Partial derivativ J(theta_0,theta_1,theta_2)
def partial_der_J(theta_0, theta_1, theta_2, Age,Weight,SBP):
   theta_0_der = theta_0
   theta_1_der = theta_1
   theta_2_der = theta_2
   for i in range(len(ID)):
      theta_0_der += ((theta_0+theta_1*Age[i]+theta_2*Weight[i])-
SBP[i])
      theta_1_der += (((theta_0+theta_1*Age[i]+theta_2*Weight[i])-
SBP[i])*Age[i])
      theta_2_der += (((theta_0+theta_1*Age[i]+theta_2*Weight[i])-
```

```
SBP[i])*Weight[i])

   return (theta_0_der, theta_1_der, theta_2_der)

# Function that calculates thetha values

def theta_val(theta_0_init,theta_1_init,theta_2_init,alpha,E,Age,Weigh
t,SBP):

  der_theta_0, der_theta_1, der_theta_2 = partial_der_J(theta_0_init,
theta_1_init,theta_2_init,Age,Weight,SBP)

  theta_0,theta_1, theta_2= theta_0_init,theta_0_init,theta_0_init

  m=len(ID)   # Total number of patients (rows)

  #while(abs(der_t0)>= E and abs(der_t1)>= E):Iter =10000
  iter=0
  #while(abs(der_theta_0)>= E or abs(der_theta_1)>= E or abs(der_theta
_2)>= E):
  while(iter<780):
    theta_0-
=(alpha/m)*der_theta_0  # Renew theta0 & theta1 simultaneously
    theta_1-=(alpha/m)*der_theta_1
    theta_2-=(alpha/m)*der_theta_2
    # print('theta_0=',theta0, 'theta1=',theta1,'der_t0=',der_t0, 'der
_t1=',der_t1)  # Testing
    der_theta0,der_theta_1,der_theta_2 = partial_der_J(theta_0,theta_1
,theta_1,Age,Weight,SBP) # Get new derivitive values
    iter+=1
    #print(theta_0,theta_1,theta_2)
  return (theta_0,theta_1,theta_2)

import numpy as np

#Patients data
ID = np.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14])
#unscaled data
# Age =np.array( [60, 61, 74, 57, 63, 68, 66, 77, 63, 54, 63, 76, 60,
61, 65, 79])
# Weight =np.array( [58, 90, 96, 72, 62, 79, 69, 96, 96, 54, 67, 99, 7
4, 73, 85, 80])
# SBP = np.array([117, 120, 145, 129, 132, 130, 110, 163, 136, 115, 11
8, 132, 111, 112])

# Scaled Data
Age =np.array( [0.196, 0.152, 0.413, 0.326, 0.065, 0.152, 0.065, 0.543
```

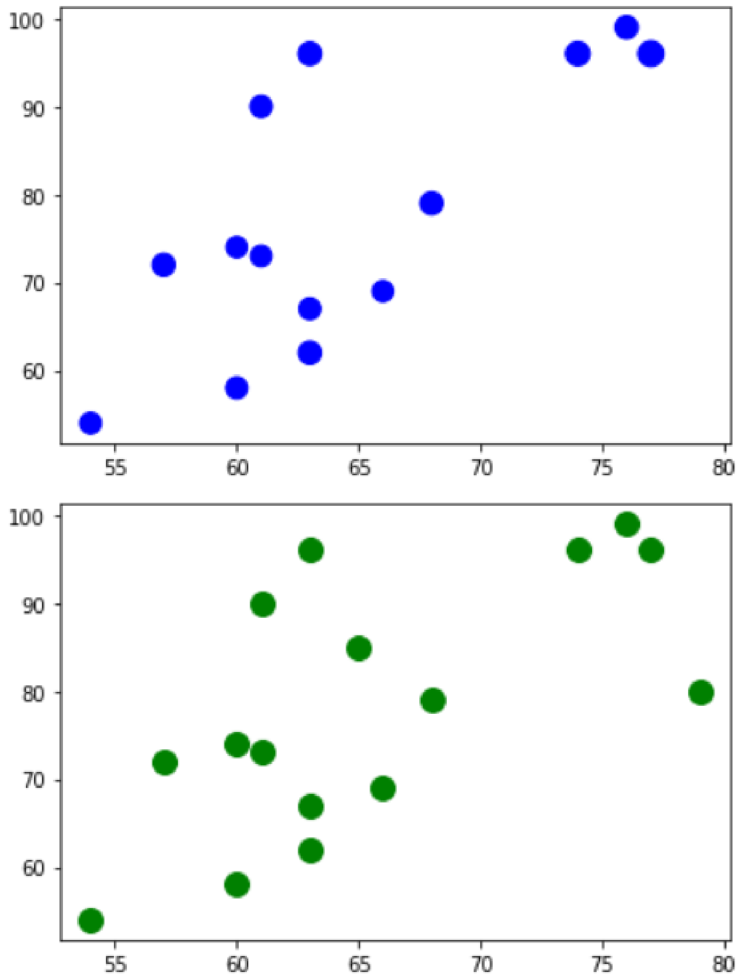
```
, 0.065, 0.457, 0.065, 0.5, 0.196, 0.152 ])
Weight =np.array([0.433,0.278,0.411,0.122,0.344,0.033,0.189,0.411,0.41
1,0.522,0.233,0.478,0.078,0.1])
SBP = np.array([0.178,0.121,0.350,0.049,0.105,0.067,0.310,0.690,0.181,
0.216,0.159,0.105,0.291,0.272])

theta_0_init = 0
theta_1_init = 0
theta_2_init = 0
alpha = 0.01
E = 0.001

theta_0,theta_1,theta_2 = theta_val(theta_0_init,theta_1_init,theta_2_
init,alpha,E,Age,Weight,SBP)

print(" Theta_0 =" ,theta_0,"Theta_1 = ",theta_1,"Theta_2 = ",theta_2)

# Values to be predicted
y15 = theta_0 + theta_1*65 +theta_2*85
y16 = theta_0 + theta_1*79 +theta_2*80
print("Y15 = ",y15, " Y16 =", y16)
```

output
Theta_0 = 1.7238, Theta_1 = -0.716, Theta_2 = -0.974 Y15 = -127.66394278419634 , Y16 = -132.81411844372838

```
#importing pandas module
import pandas as pd

#importing LinearRegression
from sklearn.linear_model import LinearRegression

#import numpy
import numpy as np

# import matplotlib
import matplotlib.pyplot as plt

#Patients Information
Age = [60, 61, 74, 57, 63, 68, 66, 77, 63, 54, 63, 76, 60, 61, 65, 79]
Weights = [58, 90, 96, 72, 62, 79, 69, 96, 96, 54, 67, 99, 74, 73, 85,
 80]
SBP = [117, 120, 145, 129, 132, 130, 110, 163, 136, 115, 118, 132, 111
, 112, 0, 0]

# Display patiencts  data
patientData = {"Age(Years)":Age, "Weight(Kg)":Weights, "SBP(mm Hg)":SB
```

```
P}

#creating data frame
data = pd.DataFrame(data = patientData)

#Obtaining the age and weight features of the patients  and the corres
ponding SBP
feature = data[["Age(Years)", "Weight(Kg)"]]
Cost = data["SBP(mm Hg)"]

#Training and Testing
feature_train = feature[:14]
feature_test = feature[14:]

cost_train = Cost[:14]
cost_test = Cost[14:]

#Obtaining model using  LinearRegression

model = LinearRegression()
model.fit(feature_train, cost_train)

# Obtaining model coefficients
theta_0 = model.intercept_
theta_1 =  model.coef_[0]
theta_2 =  model.coef_[1]

print("Thetha_0 =","{:.3f}".format(theta_0),"Thetha_1 =","{:.3f}".form
at(theta_1),"Thetha_2 = ","{:.3f}".format(theta_2),)
#predicting remaining values
pred = model.predict(feature_test)

for i in range(len(patientData)):
     hyp = theta_0 + theta_1*Age[i] + theta_2*Weights[i]

#printing predicted values
print(pred)
# Graph Results
# graphing values
plt.scatter(Age,Weights,SBP,color='blue')
plt.show()
plt.scatter(Age, Weights,hyp, color='green')
plt.show()
```

```
Thetha_0 = 35.613 Thetha_1 = 1.079 Thetha_2 =  0.274
[129.01956696 142.76293569]
```



2. Assuming that *Y* is the function of *X* in the following training set, please try to take second-order hypothesis function to fit *(X,Y)* coordinate points by curve ***Y = f(X).*** Before writing **python** program to implement regression by gradient descent algorithm, hypothesis function/loss function/cost function are needed for getting all the parameters θs in hypothesis. After training regression module, plot *(X,Y)* points and the fitting curve by **matplotlib** python functions

X	Y
0	4
1	5
2	16
3	21
4	36
5	45
6	64
7	77
8	100
9	117
10	144

Solution

Hypothesis function: $h(\theta) = \theta_0 + \theta_1 x + \theta_2 x^2$

; where x *is the feature , and* $h(\theta) =$ *a second order hypothesis*

Loss function: $l(\theta) = [\, h(x^{(i)}) - y^{(i)}]^2$

Cost function: $j(\theta) = \frac{1}{2*N}\sum_{i=1}^{N}[\, h(x^{(i)}) - y^{(i)}]^2 = \frac{1}{2*N}\sum_{i=1}^{N}[\, (\theta_0 + \theta_1(x)^i + \theta_2(x^2)^i \quad) - y^{(i)}]^2$

; where N = 10 (training set)

Minimum values of cost function:

$$\frac{\partial j(\theta)}{\partial \theta_0} = \frac{2}{2*N}\sum_{i=1}^{N}[\, ((\theta_0 + \theta_1[x]^{(i)} + \theta_2[x^2]^{(i)}) - y^{(i)}]*1 = \frac{1}{N}\sum_{i=1}^{N}[\, ((\theta_0 + \theta_1[x]^{(i)} + \theta_2[x^2]^{(i)}) - y^{(i)}]$$

$$\frac{\partial j(\theta)}{\partial \theta_1} = \frac{2}{2*N}\sum_{i=1}^{N}[\, ((\theta_0 + \theta_1[x]^{(i)} + \theta_2[x^2]^{(i)}) - y^{(i)}]*[x]^{(i)} = \frac{1}{N}\sum_{i=1}^{N}[\, ((\theta_0 + \theta_1[x]^{(i)} + \theta_2[x^2]^{(i)}) - y^{(i)}]*[x]^{(i)}$$

$$\frac{\partial j(\theta)}{\partial \theta_2} = \frac{2}{2*N}\sum_{i=1}^{N}[\, ((\theta_0 + \theta_1[x]^{(i)} + \theta_2[x^2]^{(i)}) - y^{(i)}]*[x^2]^{(i)} = \frac{1}{N}\sum_{i=1}^{N}[\, ((\theta_0 + \theta_1[x]^{(i)} + \theta_2[x^2]^{(i)}) - y^{(i)}]*[x^2]^{(i)}$$

Obtain the **hypothesis function coefficients** using the **gradient descent method**

$$\theta_0 := \theta_0 - \propto (\frac{\partial j(\theta)}{\partial \theta_0})$$

$$\theta_1 := \theta_1 - \propto (\frac{\partial j(\theta)}{\partial \theta_1})$$

$$\theta_2 := \theta_2 - \propto (\frac{\partial j(\theta)}{\partial \theta_2})$$

Code:

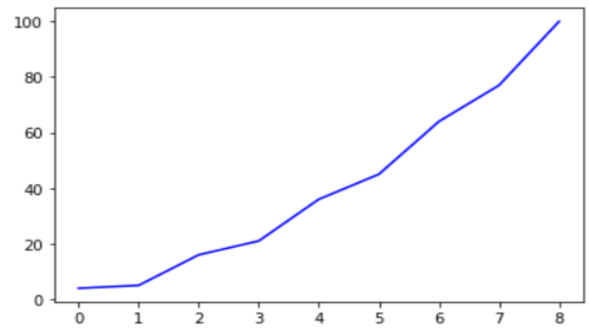
```
# Assignment Q2: Linear Regression second order hypothesis
# Patients data
# function that returns Partial derivativ J(theta_0,theta_1,theta_2)
def partial_der_J(theta_0, theta_1, theta_2, x,y):
   theta_0_der = theta_0
   theta_1_der = theta_1
   theta_2_der = theta_2
   for i in range(len(x)):
      theta_0_der += (((theta_0+theta_1*x[i]+theta_2*(x[i])**2))-y[i])
      theta_1_der +=  (((((theta_0+theta_1*x[i]+theta_2*(x[i])**2))-
y[i]))*x[i]
      theta_2_der +=  ((((theta_0+theta_1*x[i]+theta_2*(x[i])**2))-
y[i])*x[i]**2

   return (theta_0_der, theta_1_der, theta_2_der)

# Function that calculates thetha values

def theta_val(theta_0_init,theta_1_init,theta_2_init,alpha,E,x,y):

  der_theta_0, der_theta_1, der_theta_2 = partial_der_J(theta_0_init,
```

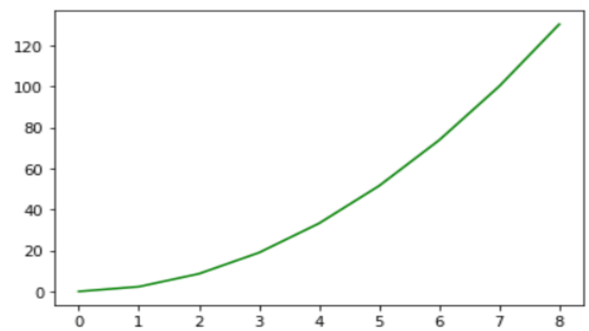
```
theta_1_init,theta_2_init,x,y)

  theta_0,theta_1, theta_2= theta_0_init,theta_0_init,theta_0_init

  m=len(x)     # Total number of patients (rows)

  #while(abs(der_t0)>= E and abs(der_t1)>= E):Iter =10000
  iter=0
  #while(abs(der_theta_0)>= E or abs(der_theta_1)>= E or abs(der_theta
_2)>= E):
  while(iter<700):
    theta_0-
=(alpha/m)*der_theta_0  # Renew theta0 & theta1 simultaneously
    theta_1-=(alpha/m)*der_theta_1
    theta_2-=(alpha/m)*der_theta_2
    # print('theta_0=',theta0, 'theta1=',theta1,'der_t0=',der_t0, 'der
_t1=',der_t1)  # Testing
    der_theta0,der_theta_1,der_theta_2 = partial_der_J(theta_0,theta_1
,theta_1,x,y) # Get new derivitive values
    iter+=1
    #print(theta_0,theta_1,theta_2)
  return (theta_0,theta_1,theta_2)

import numpy as np

# Scaled Data
x = np.array( [0, 1,2,3,4,5,6,7,8 ])
y = np.array([4,5,16,21,36,45,64,77,100])
theta_0_init = 0
theta_1_init = 0
theta_2_init = 0
alpha = 0.000002
E = 0.001

theta_0,theta_1,theta_2 = theta_val(theta_0_init,theta_1_init,theta_2_
init,alpha,E,x,y)

print(" Theta_0 =" ,theta_0,"Theta_1 = ",theta_1,"Theta_2 = ",theta_2)

# Values to be predicted
hy_f = theta_0 + theta_1*x +theta_2*x**2

# Graph Results
# graphing values
# import matplotlib
import matplotlib.pyplot as plt
```

```python
# Actual Model
plt.plot(x,y,color='blue')
plt.show()

## Actual Model
plt.plot(x,hy_f,color='green')
plt.show()
```

Result:

```
Theta_0 = 0.05724444444444448 Theta_1 = 0.30399269146794244 Theta_2 = 1.9976635482961285
```





3. Write Python program to find the parameters θs in the hypothesis function for the above dataset (dataset in q2) by Cramer's rule. And compare the results with those coming from gradient descent algorithm

Solution:

$y^{(i)} = \theta_0 + \theta_1 (x)^i + \theta_2 (x^2)^i$

$$\breve{X} = \begin{vmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \\ 1 & 6 & 36 \\ 1 & 7 & 49 \\ 1 & 8 & 64 \\ 1 & 9 & 81 \\ 1 & 10 & 100 \end{vmatrix}, \ \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}, \ \breve{Y} = \begin{bmatrix} 4 \\ 5 \\ 16 \\ 21 \\ 36 \\ 45 \\ 64 \\ 77 \\ 100 \\ 117 \\ 144 \end{bmatrix} \quad : \theta \breve{X} = \breve{Y} \ = \begin{vmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \\ 1 & 6 & 36 \\ 1 & 7 & 49 \\ 1 & 8 & 64 \\ 1 & 9 & 81 \\ 1 & 10 & 100 \end{vmatrix} * \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \\ 16 \\ 21 \\ 36 \\ 45 \\ 64 \\ 77 \\ 100 \\ 117 \\ 144 \end{bmatrix}$$

From Crammer's rule

$$\theta = \left[\tilde{X}^T \tilde{X}\right]^{-1} \left[\tilde{X}\right]^T \tilde{Y}$$

$$\theta = \left[\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0 & 1 & 4 & 9 & 16 & 25 & 36 & 49 & 64 & 91 & 100 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \\ 1 & 6 & 36 \\ 1 & 7 & 49 \\ 1 & 8 & 64 \\ 1 & 9 & 81 \\ 1 & 10 & 100 \end{bmatrix}\right]^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0 & 1 & 4 & 9 & 16 & 25 & 36 & 49 & 64 & 91 & 100 \end{bmatrix} \begin{bmatrix} 4 \\ 5 \\ 16 \\ 21 \\ 36 \\ 45 \\ 64 \\ 77 \\ 100 \\ 117 \\ 144 \end{bmatrix}$$

$$\theta = \begin{bmatrix} 11 & 55 & 385 \\ 55 & 385 & 3025 \\ 385 & 3025 & 25333 \end{bmatrix}^{-1} \begin{bmatrix} 629 \\ 4685 \\ 38313 \end{bmatrix}$$

$$\theta = \begin{bmatrix} 0.58041958 & -0.22027972 & 0.01748252 \\ -0.22027972 & 0.12564103 & -0.01165501 \\ 0.01748252 & -0.01165501 & 0.0011655 \end{bmatrix} \begin{bmatrix} 629 \\ 4685 \\ 38313 \end{bmatrix}$$

Theta_0 = 2.78
Theta_1 = 3.54
Theta_2= 1.06846

Below are the theta values from question 2, and the three graphs are obtained based on the actual, predicted and calculated hypothesis functions respectively

```
Theta_0 = 3.4347; Theta_1 = 1.384;   Theta_2 = 7.378
```