Experiment 10

Dhruv Singhal || 500075346 || R177219074 || AIML || Sem5

Least Squares Method (Direct and Multiple Input)

Importing required Libraries

In [1]:
```
import numpy as np
```

Data Creation

We create a dataset using numpy. we create random values in both variables X and y. X hold random values of shape 100,1 meaning there are 100 rows and 1 column. In y we multiply 3 and add 4 to the random values of X and again generate random values in the shape 1000,1 where the values are in standard normal distribution.

In [2]:
```
x1 = np. random. random( (100, 1))
y= 4 + 3*x1 + np. random. randn(100, 1)
x0 = np. ones ( (100, 1))
X = np. concatenate( (x0, x1), axis = 1)
```

Data Preprocessing & Visualization

in the x0 variable we take a array of shape 100,1 with only 1 values stored. Now in the X variable we concatenate x0 which hold the array of ones and the random values generated previously along the single axis/column present in the dataset. After this we print the shape of both the X data and y data.

Model Building Single Input

Here we use np.linalg.inv to inverse our matrix X. the inverse of X is such that if multiplied by the original X gives us an identity matrix. We store this in temp1. In the variable temp2 we store the dot product of X and temp1. Similarly in w we store the dot product of t2 and the y dataset. Lastly we print w. We do this to directly find the least squares.

In [3]:
```
temp1 = np.linalg.inv(np.dot (X.T, X))
temp2 = np.dot(temp1,X.T)
w = np. dot(temp2, y)
print("------------------------------")
print("Least squares method(Direct) Single Input")
print("------------------------------")
print("W0",w[0])
print("W1",w[1])
```

```
------------------------------
Least squares method(Direct) Single Input
------------------------------
W0 [4.18332282]
W1 [2.58556085]
```

Model building Multiple Input

In [4]:
```
import numpy as np
x1 = np.random. random( (100, 3))
X=np.c_[np.ones((100,1)),x1]
a=[[4,5,8],
   [8,5,7],
   [7,6,3],
   [1,3,8]]
W=np.array(a)
y1=np.dot(X,W)
temp1 = np.linalg.inv(np.dot (X.T, X))
temp2 = np.dot(temp1,X.T)
w = np. dot(temp2, y1)
print("------------------------------")
print("Least squares method(Direct) Multiple Input")
print("------------------------------")
print("W1's are:\n" ,w)
```

```
------------------------------
Least squares method(Direct) Multiple Input
------------------------------
W1's are:
 [[4. 5. 8.]
 [8. 5. 7.]
 [7. 6. 3.]
 [1. 3. 8.]]
```

Multiple Input:

Now we apply the least squares on 4 features. Our 4 features are y1, y2, y3, y4. Each one has a different dataset of same shape i.e 100,1 while the X_ data has 100,3 shape. After that in xwb, X_ is concatenated with np.ones with size 100,1. After that in W_ we multiply the transpose of Xwb with Xwb. in tp1 we store the inverse of W_ and in tp2 we store the dot product of tp1 and transpose of Xwb and after that in W1 store the dot product of tp2 and y1 and similarly in W2 we store dot of tp2 and y2 and so on

In [5]:
```
X_ = np.random. random( (100, 3))
y1= 4 + 3*X_ + np. random. randn(100, 1)
y2= 5 + 2*X_ + np. random. randn(100, 1)
y3= 3 + 6*X_ + np. random. randn(100, 1)
y4= 7 + 9*X_ + np. random. randn(100, 1)
Xwb=np.c_[np.ones((100,1)),X_]
W_=Xwb.T.dot(Xwb)
tp1 = np.linalg.inv(np.dot (Xwb.T, Xwb))
tp2 = np.dot(tp1,Xwb.T)
W1 = np. dot(tp2, y1)
W2 = np. dot(tp2, y2)
W3 = np. dot(tp2, y3)
W4 = np. dot(tp2, y4)


print("------------Modified------------------")
print("Least squares method(Direct) Multiple Input")
print("------------------------------")
print("W1:\n",W1)
print("W2:\n",W2)
print("W3:\n",W3)
print("W4:\n",W4)
```

```
------------Modified------------------
Least squares method(Direct) Multiple Input
------------------------------
W1:
 [[ 3.58946499  3.58946499  3.58946499]
 [ 3.53007295  0.53007295  0.53007295]
 [ 0.7360797   3.7360797   0.7360797 ]
 [-0.22140637 -0.22140637  2.77859363]]
W2:
 [[ 5.00632769  5.00632769  5.00632769]
 [ 1.55928339 -0.44071661 -0.44071661]
 [ 0.08956033  2.08956033  0.08956033]
 [ 0.46577766  0.46577766  2.46577766]]
W3:
 [[ 3.07157824  3.07157824  3.07157824]
 [ 5.97448906 -0.02551094 -0.02551094]
 [ 0.53028384  6.53028384  0.53028384]
 [-0.29725616 -0.29725616  5.70274384]]
W4:
 [[6.54317499 6.54317499 6.54317499]
 [9.34325755 0.34325755 0.34325755]
 [0.43786949 9.43786949 0.43786949]
 [0.03674298 0.03674298 9.03674298]]
```

In [6]:
```
print(np.concatenate((W1,W2,W3,W4)))
```

```
[[ 3.58946499  3.58946499  3.58946499]
 [ 3.53007295  0.53007295  0.53007295]
 [ 0.7360797   3.7360797   0.7360797 ]
 [-0.22140637 -0.22140637  2.77859363]
 [ 5.00632769  5.00632769  5.00632769]
 [ 1.55928339 -0.44071661 -0.44071661]
 [ 0.08956033  2.08956033  0.08956033]
 [ 0.46577766  0.46577766  2.46577766]
 [ 3.07157824  3.07157824  3.07157824]
 [ 5.97448906 -0.02551094 -0.02551094]
 [ 0.53028384  6.53028384  0.53028384]
 [-0.29725616 -0.29725616  5.70274384]
 [ 6.54317499  6.54317499  6.54317499]
 [ 9.34325755  0.34325755  0.34325755]
 [ 0.43786949  9.43786949  0.43786949]
 [ 0.03674298  0.03674298  9.03674298]]
```

In [ ]: