파일
업로드 새로고침
드라이브 마운트 해제
..
drive
My Drive
Abroad_Program
Classroom
Colab Notebooks
Fall|2018
Fall|2019
Spring|2018
Spring|2019
Spring|2020
Compiler
Database_System
Information_Security_Theory
Machine_Learning
CAU-MachineLearning
Assignment01
Assignment02
Assignment03
Assignment04
Assignment04.ipy...
assignment_04_a...
data_test.csv
data_train.csv
README.md
PPT
Testing_repo
Multicore_Computing
Network_Application_and_D...
Template.docx
Summer|2018-Purdue
Summer|2019-USC
디스크 76.87 GB 사용 가능

Information

Writer : *Junhyuck Woo*
Std. ID : 20145337
*Assignment*04 : *Linear regression with multiple variables*
Deadline : *Apr* 16, 2020

Library

```
[ ] import matplotlib.pyplot as plt; import numpy as np; import csv
```

Data - Train

```
[ ] path="/content/drive/My Drive/Spring|2020/Machine_Learning/CAU-MachineLearning/Assignment04/data_train.csv"

    x = []; y = []; z = []; h = [];

    with open(path, newline='') as myfile:
        reader  = csv.reader(myfile, delimiter=',')
        ct = 0
        for i in reader:
            x.append(float(i[0]))
            y.append(float(i[1]))
            z.append(float(i[2]))
            h.append(float(i[3]))
            ct += 1

    x_data = np.array(x)
    y_data = np.array(y)
    z_data = np.array(z)
    h_data = np.array(h)
    m = ct
```

Data - Test

```
[ ] path="/content/drive/My Drive/Spring|2020/Machine_Learning/CAU-MachineLearning/Assignment04/data_test.csv"

    x = []; y = []; z = []; h = [];

    with open(path, newline='') as myfile:
        reader  = csv.reader(myfile, delimiter=',')
        ct = 0
        for i in reader:
            x.append(float(i[0]))
            y.append(float(i[1]))
            z.append(float(i[2]))
            h.append(float(i[3]))
            ct += 1

    x_test = np.array(x)
    y_test = np.array(y)
    z_test = np.array(z)
    h_test = np.array(h)
```

Linear Model

$f_\theta(x, y, z) = \theta_0 + \theta_1 x + \theta_1 y + \theta_1 z,$
where $\theta = (\theta_0, \theta_1, \theta_2, \theta_3)$ *and* $\theta_0, \theta_1, \theta_2, \theta_3 \in R$

InitialState
$\theta_0 = 10$
$\theta_1 = 10$
$\theta_2 = 10$
$\theta_3 = 10$

```
[ ] theta0 = 10; theta0_old = 0
    theta1 = 10; theta1_old = 0
    theta2 = 10; theta2_old = 0
    theta3 = 10; theta3_old = 0
    theta0_history = [theta0]
    theta1_history = [theta1]
    theta2_history = [theta2]
    theta3_history = [theta3]

    f = theta0 + theta1*x_data + theta2*y_data + theta3*z_data
    f_test = theta0 + theta1*x_test + theta2*y_test + theta3*z_test
```

Objective Function

$J(\theta_0, \theta_1, \theta_2, \theta_3) = \frac{1}{2m} \sum_{i=1}^{m} (f_\theta(x^{(i)}, y^{(i)}, z^{(i)}) - h^{(i)})^2$

```
[ ] j = np.sum((f - h_data)**2) / (2*m)
    j_test = np.sum((f_test - h_data)**2) / (2*m)
    j_old = 0
    j_history = [j]
    j_test_history = [j_test]
```

Training Process - Gradient Descent

$\alpha = 0.00002$

$$\theta_0^{(t+1)} := \theta_0^{(t)} - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_\theta(x^{(i)}, y^{(i)}, z^{(i)}) - h^{(i)})$$
$$\theta_1^{(t+1)} := \theta_1^{(t)} - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_\theta(x^{(i)}, y^{(i)}, z^{(i)}) - h^{(i)}) x^{(i)}$$
$$\theta_2^{(t+1)} := \theta_2^{(t)} - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_\theta(x^{(i)}, y^{(i)}, z^{(i)}) - h^{(i)}) y^{(i)}$$
$$\theta_3^{(t+1)} := \theta_3^{(t)} - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_\theta(x^{(i)}, y^{(i)}, z^{(i)}) - h^{(i)}) z^{(i)}$$

```
alpha = 0.00002

# Check the number of iteration
iteration = 1

# Set the two condition because it spent too much time for converge
while  (iteration < 1048576) or (abs(j - j_old) + abs(theta0 - theta0_old) + abs(theta1 - theta1_old) + abs(theta2 - theta2_old) + abs(theta3 - theta
    # Calculate the theta
    theta0_old = theta0
    theta1_old = theta1
    theta2_old = theta2
    theta3_old = theta3
    theta0 = theta0 - alpha*np.sum(f-h_data)/m
    theta1 = theta1 - alpha*np.sum((f-h_data)*x_data)/m
    theta2 = theta2 - alpha*np.sum((f-h_data)*y_data)/m
    theta3 = theta3 - alpha*np.sum((f-h_data)*z_data)/m

    # Update the j, h
    j_old = j
    f = theta0 + theta1*x_data + theta2*y_data + theta3*z_data
    j = np.sum((f - h_data)**2) / (2*m)

    f_test = theta0 + theta1*x_test + theta2*y_test + theta3*z_test
    j_test = np.sum((f_test - h_test)**2) / (2*m)

    # Record the history of parameter
    theta0_history.append(theta0)
    theta1_history.append(theta1)
    theta2_history.append(theta2)
    theta3_history.append(theta3)
    j_history.append(j)
    j_test_history.append(j_test)

    iteration = iteration +1
```

Check the Convergence

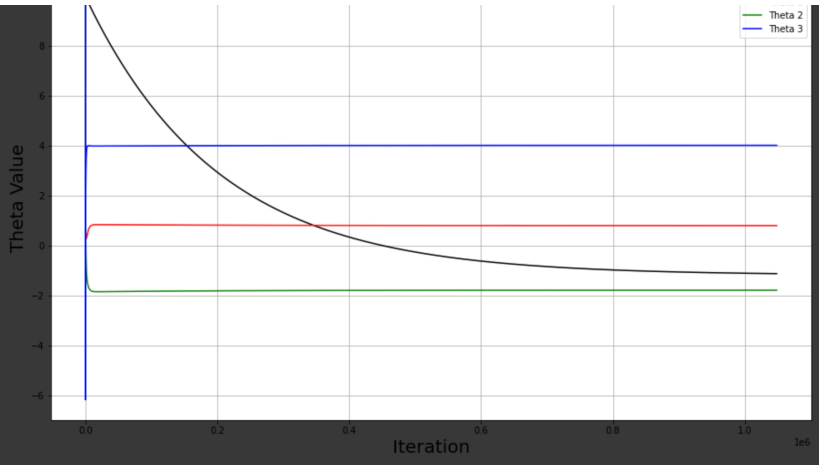
```
# Iteration
print("# Iteration: " + str(iteration) + '\n')

# Theta 0
print("Updated Theta0: "+ str(theta0))
print("Old Theta0: " + str(theta0_old))
print("Diff: " + str(theta0 - theta0_old) + '\n')

# Theta 1
print("Updated Theta1: "+ str(theta1))
print("Old Theta1: " + str(theta1_old))
print("Diff: " + str(theta1 - theta1_old) + '\n')

# Theta 2
print("Updated Theta1: "+ str(theta2))
print("Old Theta1: " + str(theta2_old))
print("Diff: " + str(theta2 - theta2_old) + '\n')

# Theta 3
print("Updated Theta1: "+ str(theta3))
print("Old Theta1: " + str(theta3_old))
print("Diff: " + str(theta3 - theta3_old) + '\n')

# J, Energy Value
print("Updated J: "+ str(j))
print("Old J: " + str(j_old))
print("Diff: " + str(j - j_old) + '\n')
```

```
# Iteration: 1048576

Updated Theta0: -1.1315915325707928
Old Theta0: -1.1315912311904022
Diff: -3.013803906437573e-07

Updated Theta1: 0.7930927395124178
Old Theta1: 0.7930927407011142
Diff: -1.1886964745855266e-09

Updated Theta1: -1.7947307409287647
Old Theta1: -1.7947307427542791
Diff: 1.8255144063061834e-09

Updated Theta1: 4.007792998932928
Old Theta1: 4.007792998058442
Diff: 8.744853730036084e-10

Updated J: 103.45430618097672
Old J: 103.45430618551832
Diff: -4.541604425867263e-09
```

1. Estimated Parameters

```
plt.figure(figsize=(15,9))
plt.plot(theta0_history, color='black', label='Theta 0')
plt.plot(theta1_history, color='red', label='Theta 1')
plt.plot(theta2_history, color='green', label='Theta 2')
plt.plot(theta3_history, color='blue', label='Theta 3')
plt.grid()
plt.legend()
plt.title('Estimated Parameters', fontsize=40)
plt.xlabel('Iteration', fontsize=20)
plt.ylabel('Theta Value', fontsize=20)
```

```
Text(0, 0.5, 'Theta Value')
```



2. Training Error

```
[ ]  plt.figure(figsize=(15,9))
     plt.plot(j_history, color='blue', label='Training Error')
     plt.grid()
     plt.legend()
     plt.title('Training Error', fontsize=40)
     plt.xlabel('Iteration', fontsize=20)
     plt.ylabel('Energy Value', fontsize=20)
```

```
Text(0, 0.5, 'Energy Value')
```

3. Testing Error

```
[ ]  plt.figure(figsize=(15,9))
     plt.plot(j_test_history, color='red', label='Testing Error')
     plt.grid()
     plt.legend()
     plt.title('Training Error', fontsize=40)
     plt.xlabel('Iteration', fontsize=20)
     plt.ylabel('Energy Value', fontsize=20)
```

```
Text(0, 0.5, 'Energy Value')
```