경사 하강법(Gradient Descent)

```
import warnings
warnings.filterwarnings('ignore')
```

I. Machine( ) 정의

- numpy Package

```
import numpy as np
```

- def Machine( )

```
def Machine(x, w, b):
    y_hat = (w * x) + b
    return y_hat
```

- x, w, b 객체 지정

```
x = np.array([1, 3, 5, 7, 9])
w = 2
b = 1
```

- Machine( ) 테스트

```
Machine(x, w, b)
```

```
array([ 3,  7, 11, 15, 19])
```

II. Gradient( ) 정의

- def Gradient( )

```
def Gradient(x, y, w, b):
    y_hat = Machine(x, w, b)

    dw = np.mean((y - y_hat) * (-2 * x))
    db = np.mean((y - y_hat) * (-2))

    return dw, db
```

- Gradient( ) 테스트

```
y = np.array([2, 4, 6, 8, 10])

dw, db = Gradient(x, y, w, b)
```

```
print('dw is ', dw)
print('db is ', db)
```

```
dw is  66.0
db is  10.0
```

III. Learning( ) 정의

- def Learning( )

```
def Learning(x, y, w, b, step):
    dw, db = Gradient(x, y, w, b)

    uw = w - step * dw
    ub = b - step * db

    return uw, ub
```

- Learning( ) 테스트

```
step = 0.05

uw, ub = Learning(x, y, w, b, step)
```

```
print('Updated_w is ', '%.3f' % uw)
print('Updated_b is ', '%.3f' % ub)
```

```
Updated_w is  -1.300
Updated_b is  0.500
```

IV. testData.csv에 적용

- pandas & matplotlib Packages

```
import pandas as pd
import matplotlib.pyplot as plt
```

- Colab File Upload
  - testData.csv

```
!ls -l
```

```
total 132
drwxr-xr-x 1 root root   4096 Feb 16 16:35 sample_data
-rw-r--r-- 1 root root 128698 Feb 19 05:27 testData.csv
```

- testData.csv Information

```
DATA = pd.read_csv('testData.csv')
```

```
DATA.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   inputs   5000 non-null   float64
 1   outputs  5000 non-null   float64
dtypes: float64(2)
memory usage: 78.2 KB
```

```
DATA.head()
```

	inputs	outputs
0	0.2362	0.162367
1	0.9415	0.479356
2	0.3495	0.095733
3	0.3200	-0.111783
4	0.8335	0.386012

- testData.csv Visualization

```
plt.scatter(DATA.inputs, DATA.outputs, s = 0.5)
plt.show()
```

- 1500번 학습 실행

```
w = 2
b = 3
step = 0.05
```


```
for i in range(0, 1500):
    uw, ub = Learning(DATA.inputs, DATA.outputs, w, b, step)
    w = uw
    b = ub
```

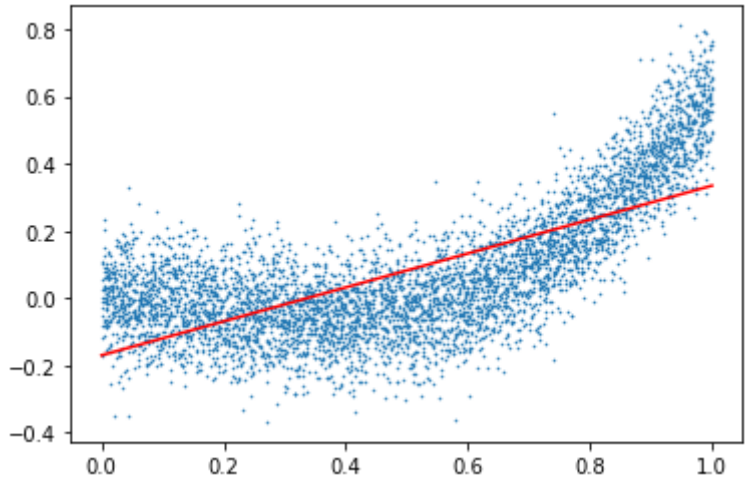
```
print('Learned_w is ', '%.3f' % w)
print('Learned_b is ', '%.3f' % b)
```

```
Learned_w is  0.505
Learned_b is  -0.170
```

- 학습결과 회귀선 그리기

```
X = np.linspace(0, 1, 100)
Y = (w * X) + b

plt.scatter(DATA.inputs, DATA.outputs, s = 0.3)
plt.plot(X, Y, '-r', linewidth = 1.5)
plt.show()
```



V. Loss Visualization

- Gradient( )에 Loss 추가

```
def Gradient(x, y, w, b):
    y_hat = Machine(x, w, b)

    dw = np.mean((y - y_hat) * (-2 * x))
    db = np.mean((y - y_hat) * (-2))
    Loss = np.mean((y - y_hat)**2)

    return dw, db, Loss
```

- Learning( )에 Loss 추가

```
def Learning(x, y, w, b, step):
    dw, db, Loss = Gradient(x, y, w, b)

    uw = w - step * dw
    ub = b - step * db

    Loss = Loss

    return uw, ub, Loss
```

- 1500번 학습 실행

```
w = 2
b = 3
step = 0.001
Error = []
```

```
for i in range(0, 1500):
    uw, ub, Loss = Learning(DATA.inputs, DATA.outputs, w, b, step)

    w = uw
    b = ub
    Error.append(Loss)
```

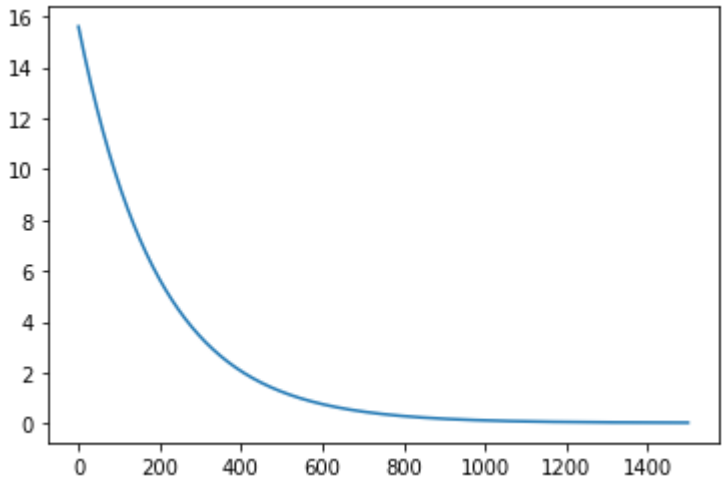
- Loss 감소 확인

```
Error[0:10]
```

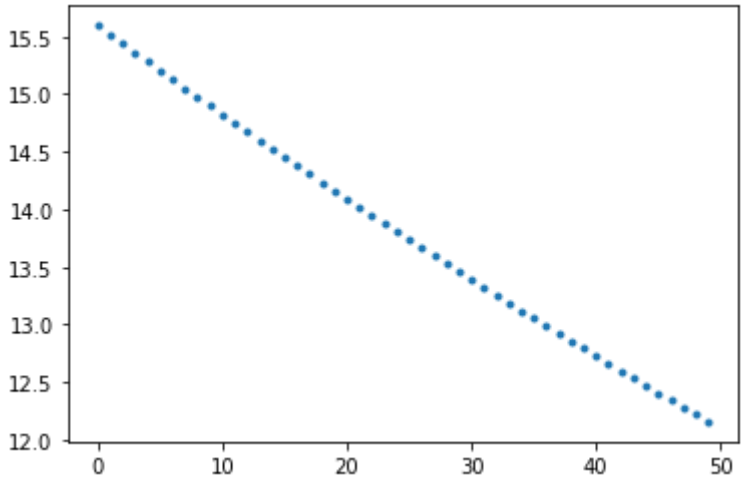
```
[15.595575679087696,
 15.516493615452518,
 15.437813155278901,
 15.359532259084617,
 15.28164889774523,
 15.204161052440144,
 15.127066714601533,
 15.050363885861731,
```

```
 14.97405057800144,
 14.898124812898125]
```

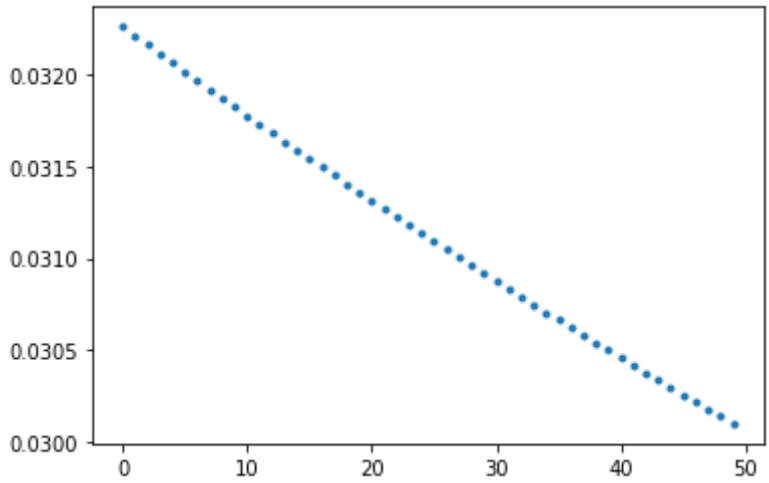
```
plt.plot(Error)
plt.show()
```




```
plt.plot(Error[0:50], '.')
plt.show()
```




```
plt.plot(Error[1450:1500], '.')
plt.show()
```



#

#

#

The End

#

#

#