

▼ 1. Input data

- Configuration variable들을 정의하고 Input data를 정규분포를 따르는 랜덤값으로 생성했습니다.

dataset의 크기는 1000, 찾고자하는 함수는 $f(x) = -14x + 20$ 이다. 또한 learning rate은 0.001로 뒀습 random.normal을 통해 정규분포가 $N(0, 3)$ 의 분포를 따르도록 x값과 n을 정의했습니다.

- 학습시킬 linear model을 초기화하고 input data와 linear model을 시각화했습니다.

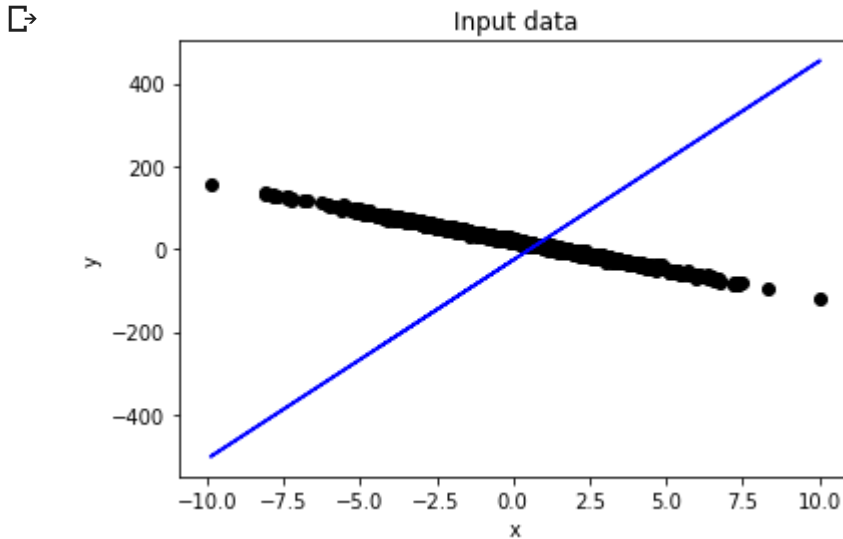
linear model은 $(\theta_0, \theta_1) = (48, -27)$ 을 만족하는 $f(x) = 48x - 27$ 로 초기화하고 linear model과 랜덤값으로 생성한 x,y값을 시각화했습니다.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 '''
4 1. Data
5 - Config
6 - Input Data
7
8 2. Linear Model
9 - Initialize linear model
10 - Visualize data and linear model
11 '''
12
13 # Config
14 m=1000
15 b=20
16 a=-14
17 learning_rate=0.001
18 theta0, theta1=-27, 48
19
20 # Input data
21 x=np.random.normal(0, 3, size=m)
22 temp_y=a*x+b
23 n=np.random.normal(0, 3, size=m)
24 y=temp_y+n
25
26 # Initialize linear model
27 h=theta1*x+theta0
28
29 # Visualize data and linear model
30 plt.scatter(x, y, color='black')
31 plt.plot(x, h, color='blue')
32 plt.xlabel('x')
33 plt.ylabel('y')
34 plt.title('Input data')

```

```
34 plt.title( input_data )
35 plt.show()
```



2. Output results

- 먼저 Objective function, gradient descent, derivation 들을 정의했습니다.
- Gradient descent 방법으로 theta0와 theta1을 update시키면서 학습을 이어갑니다.
이때 학습이 끝나는 조건은 두 값이 수렴하여 이전 theta값들과 upate후의 theta값들이 동일할때입니다.

```
1 '''
2 3. Objective function
3 - Define objective function
4
5 4. Gradient Descent
6 - Define derivations
7 - Update theta0 and theta1
8 '''
9
10 # Define objective function
11 def objective_function() :
12     J = np.sum((h-y)**2) / (2*m)
13     return J
14
15 # Define derivations
16 def dev() :
17     dev0=np.sum(h-y) / m
18     dev1=np.sum((h-y)*x) / m
19     return dev0, dev1
20
21 # Update theta0 and theta1
22 def gradient_descent() :
23     return theta0-(learning_rate*dev0), theta1-(learning_rate*dev1)
24
25 J_list=[]
26 theta0_list=[]
```

```

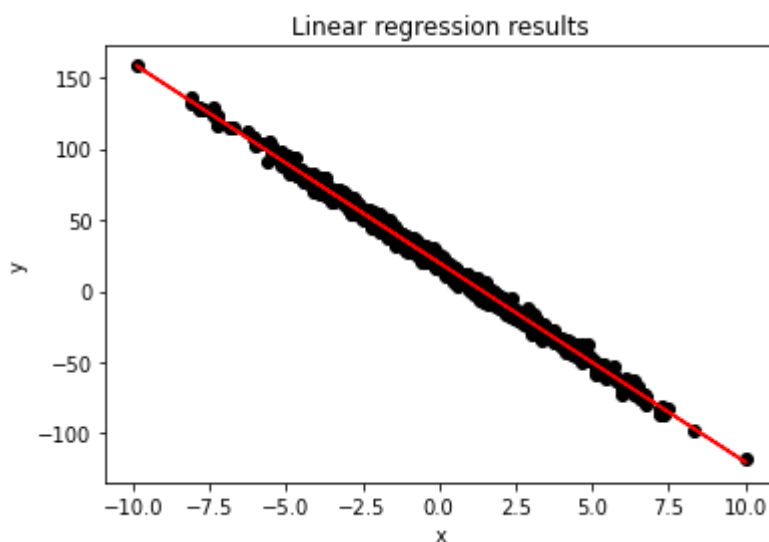
26 theta0_list=[]
27 theta1_list=[]
28 # Train
29 count=0
30 while(1) :
31     count+=1
32     J=objective_function()
33     J_list.append(J)
34     theta0_list.append(theta0)
35     theta1_list.append(theta1)
36     dev0, dev1 = dev()
37     temp0, temp1 = theta0, theta1
38     theta0, theta1 = gradient_descent()
39     h=theta0+theta1*x
40     if(temp0==theta0) and (temp1==theta1) :
41         break
42 print("theta0 : ", theta0)
43 print("theta1 : ", theta1)
44 print("loss : ", J)
45 # Visualize data and linear model
46 plt.scatter(x, y, color='black')
47 plt.plot(x, h, color='red')
48 plt.xlabel('x')
49 plt.ylabel('y')
50 plt.title('Linear regression results')
51 plt.show()

```

```

↳ theta0 : 20.01254840580673
   theta1 : -14.001667149441015
   loss : 4.300044723130642

```



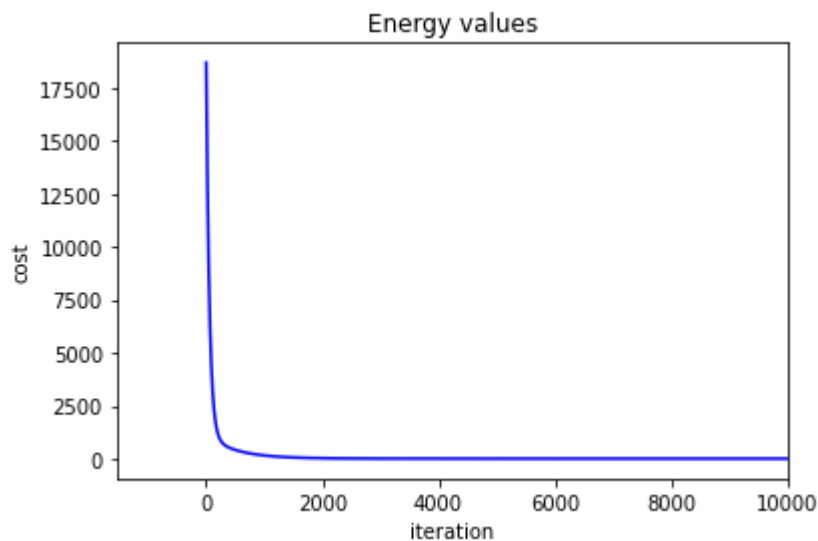
3. Energy Values

- update를 할때마다 J_list에 그동안의 J(objective function의 value) 값들을 넣었습니다.
- 각 iteration마다 J값이 어떻게 변하는지 시각화했습니다.

```

1 plt.plot(J_list, color='blue')
2 plt.xlabel("iteration")
3 plt.xlim(right=10000)
4 plt.ylabel("cost")
5 plt.title("Energy values")
6 plt.show()

```



▼ 4. Model parameters

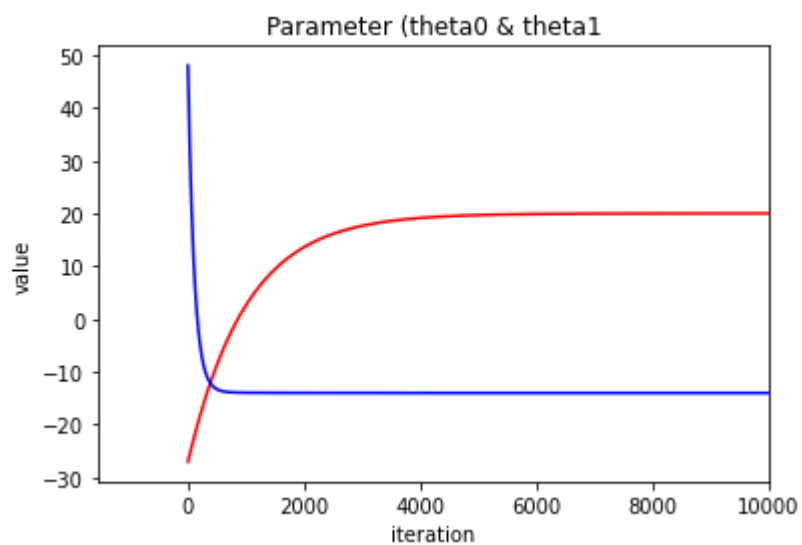
- update를 할때마다 변하는 θ_0 과 θ_1 의 값들을 따로 리스트해 저장해뒀었습니다.
- 각 iteration마다 θ_0 과 θ_1 이 어떻게 변하는지 시각화했습니다.

```

1 plt.plot(theta0_list, color='red')
2 plt.plot(theta1_list, color='blue')
3 plt.xlabel("iteration")
4 plt.xlim(right=10000)
5 plt.ylabel("value")
6 plt.title("Parameter ( $\theta_0$  &  $\theta_1$ )")
7 plt.show()











```






History for [class-MachineLearning](#) / [assignment02](#) / MachineLearning_assignment02.ipynb

🔑 Commits on Apr 1, 2020

Modified text  Ganghee-Lee committed 15 seconds ago
Add text  Ganghee-Lee committed 15 minutes ago
Visualize model parameters  Ganghee-Lee committed 1 hour ago
Visualize energy values  Ganghee-Lee committed 2 hours ago
Visualize linear regression results  Ganghee-Lee committed 2 hours ago
Define objective function and derivations  Ganghee-Lee committed 2 hours ago
Visualize data and linear model  Ganghee-Lee committed 3 hours ago
2. Define Linear Model  Ganghee-Lee committed 3 hours ago
1. Generate data  Ganghee-Lee committed 5 hours ago
1. Data  Ganghee-Lee committed 5 hours ago

🔑 Commits on Mar 31, 2020

first commit  Ganghee-Lee committed 13 hours ago
--