정밀의료 데이터 분석을 위한
머신러닝 부트캠프

Linear regression and Logistic regression

정다래

Machine Learning

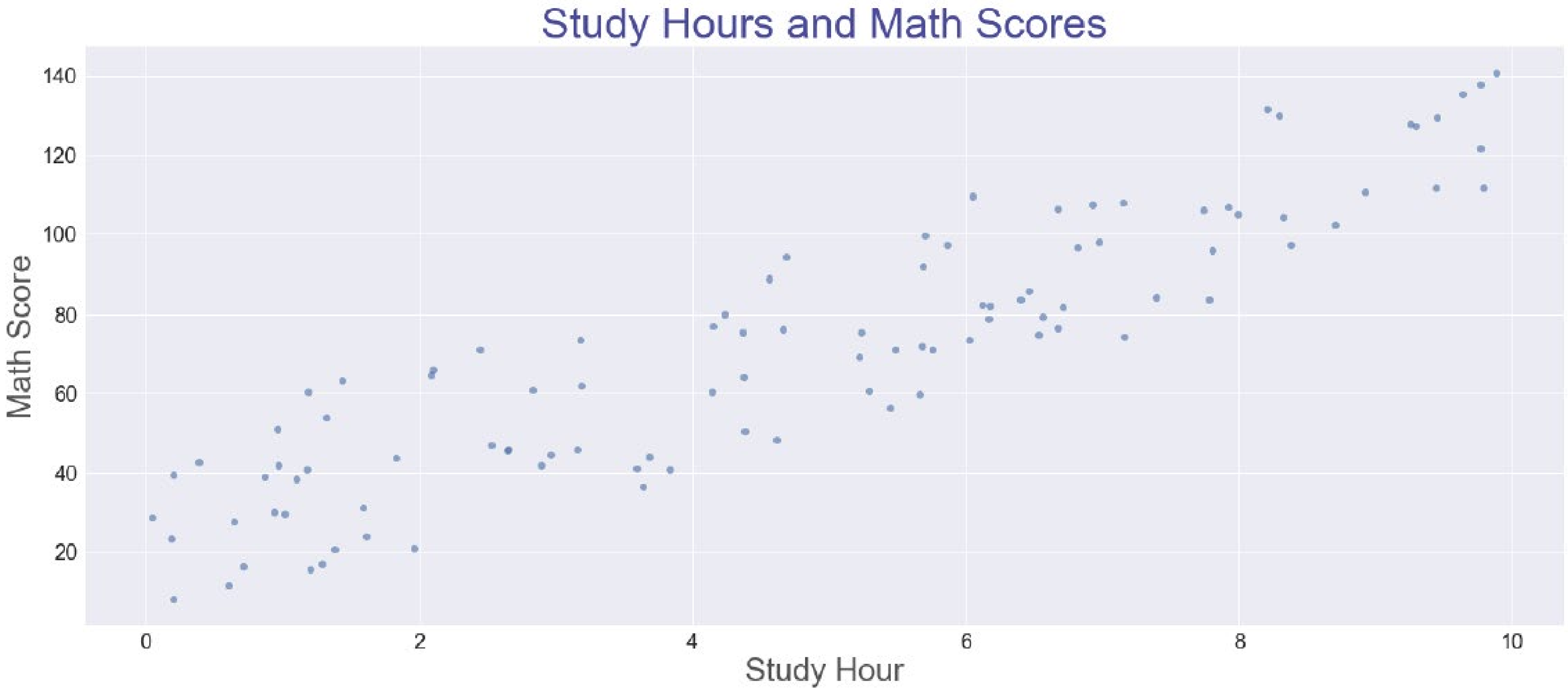
Supervised Learning
(Precategorized Data)

- **Classification** 분류
- **Regression** 회귀

Unsupervised Learning
(Unlabelled Data)

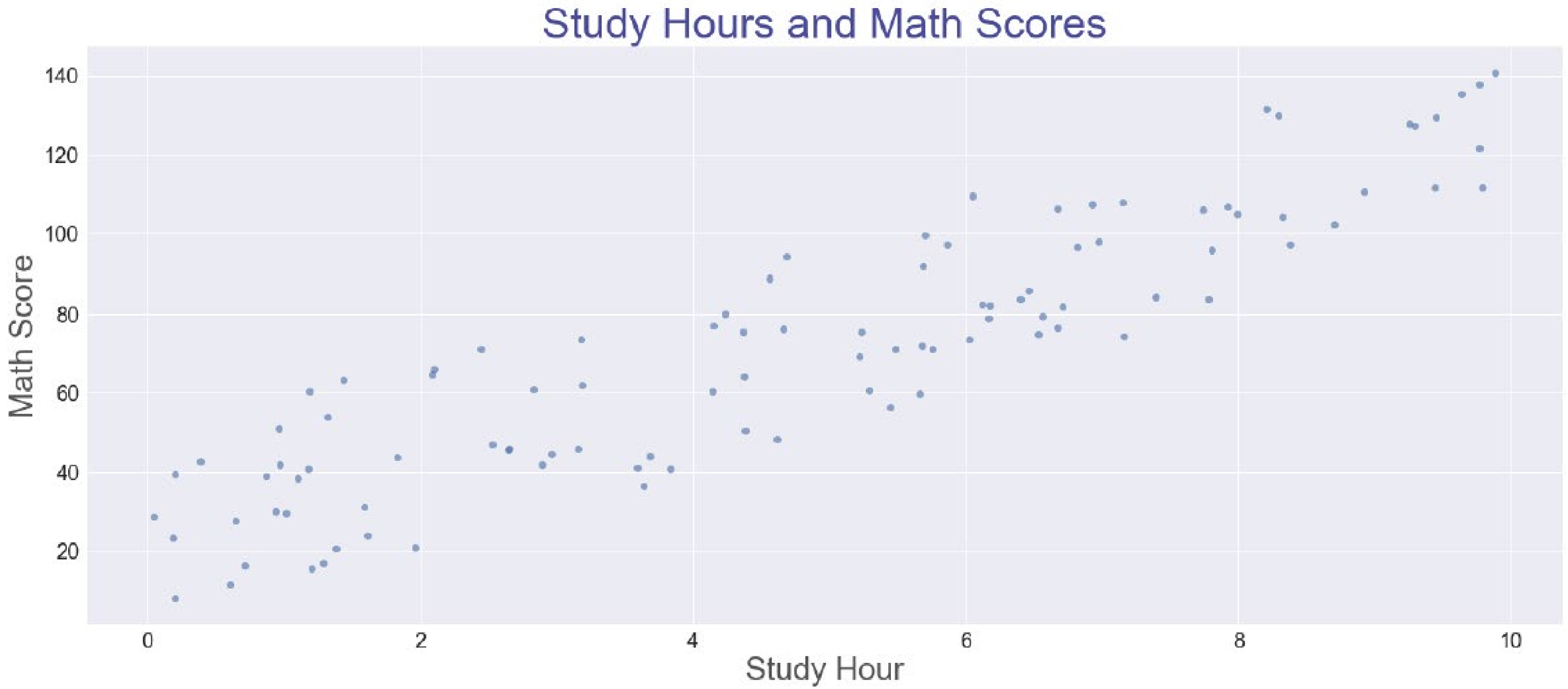
- **Clustering** 군집화
- **Association** 연관
- **Dimension Reduction** 차원축소

What's Regression?



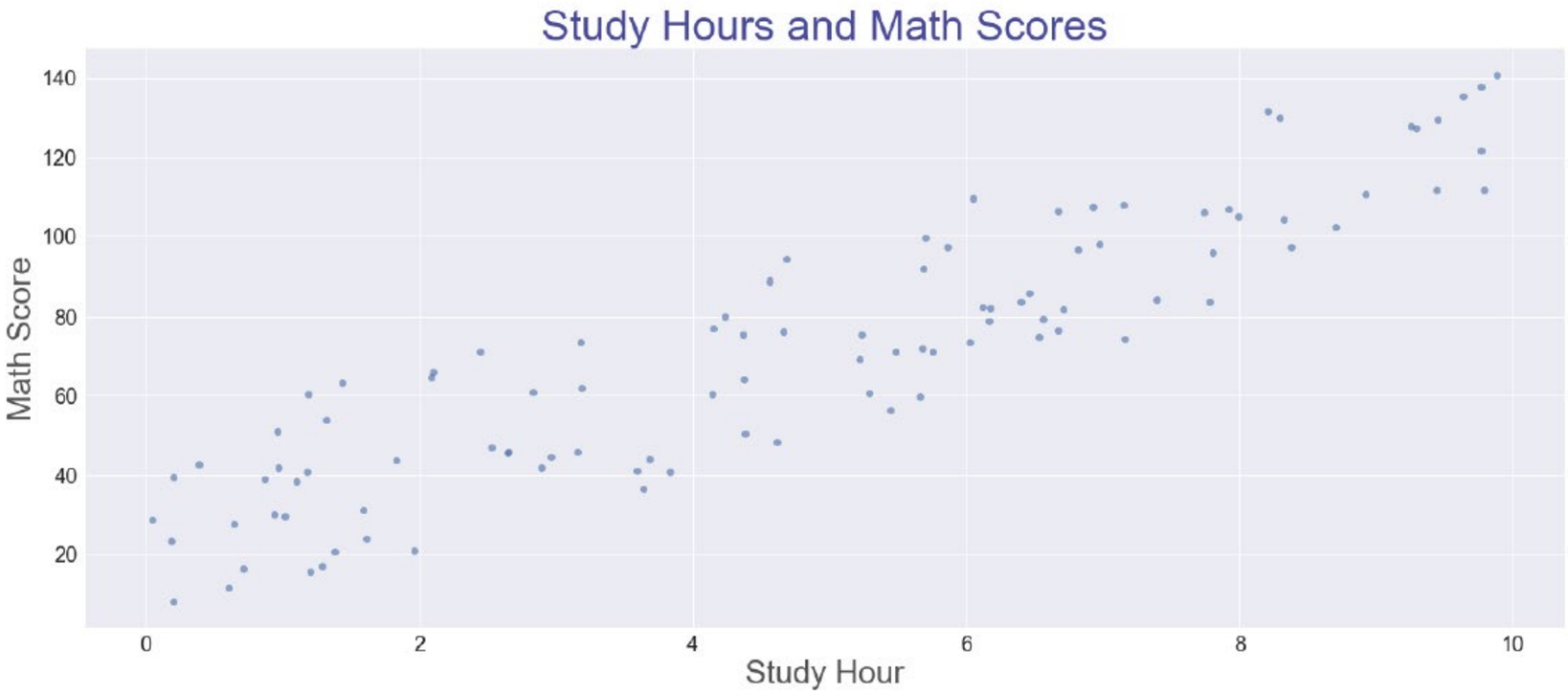
What's Regression?

predict the continuous response for unseen data



What's Linear Regression?

Dataset

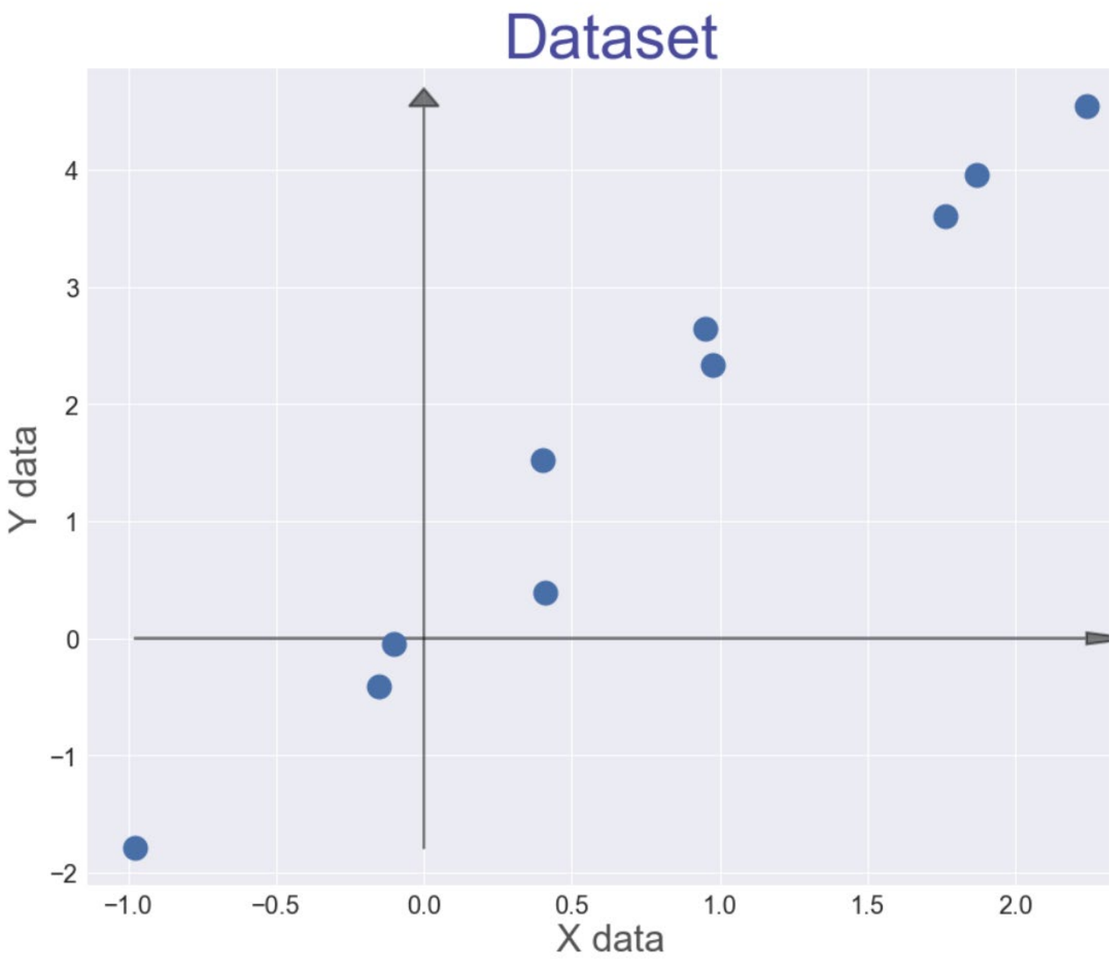


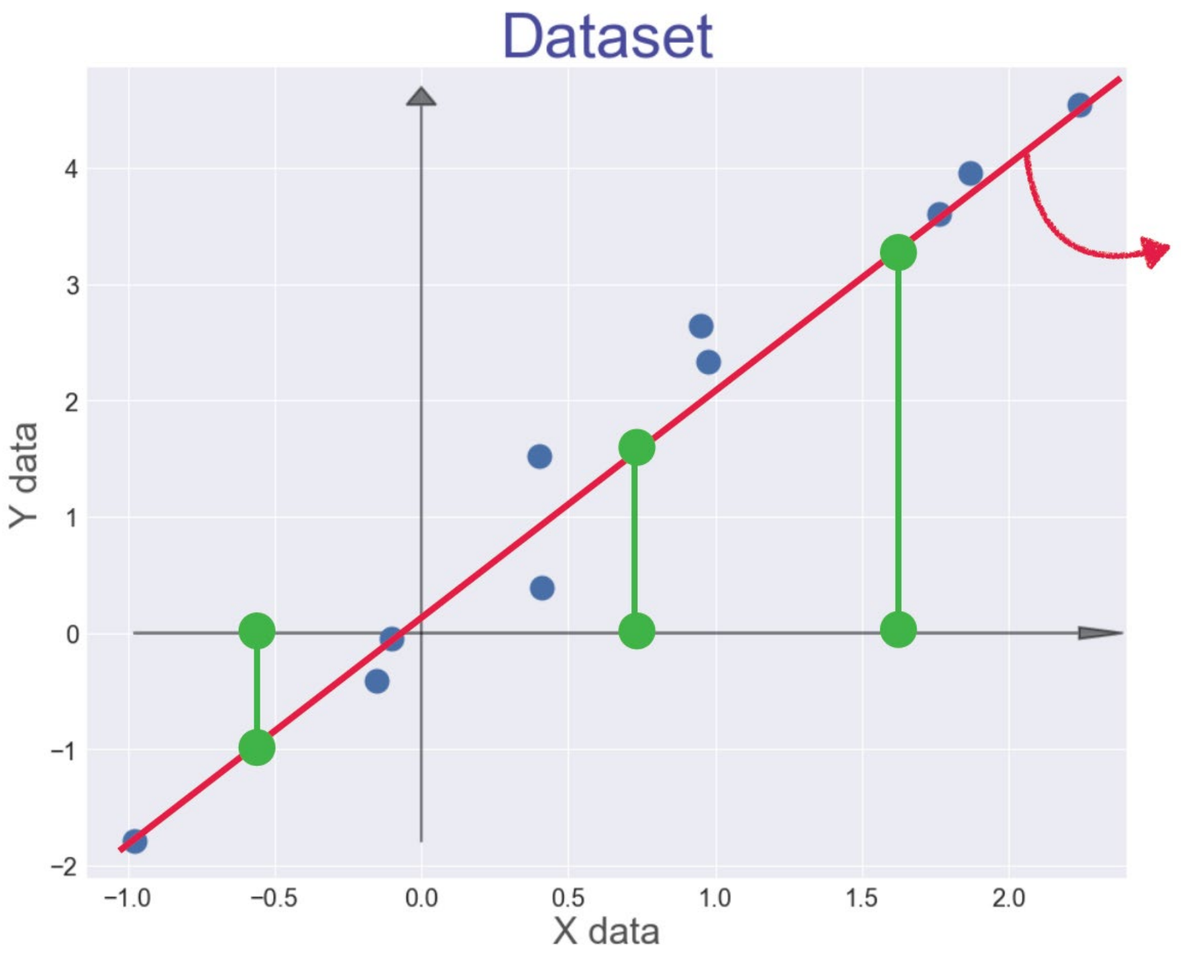
Model Setting

Learning

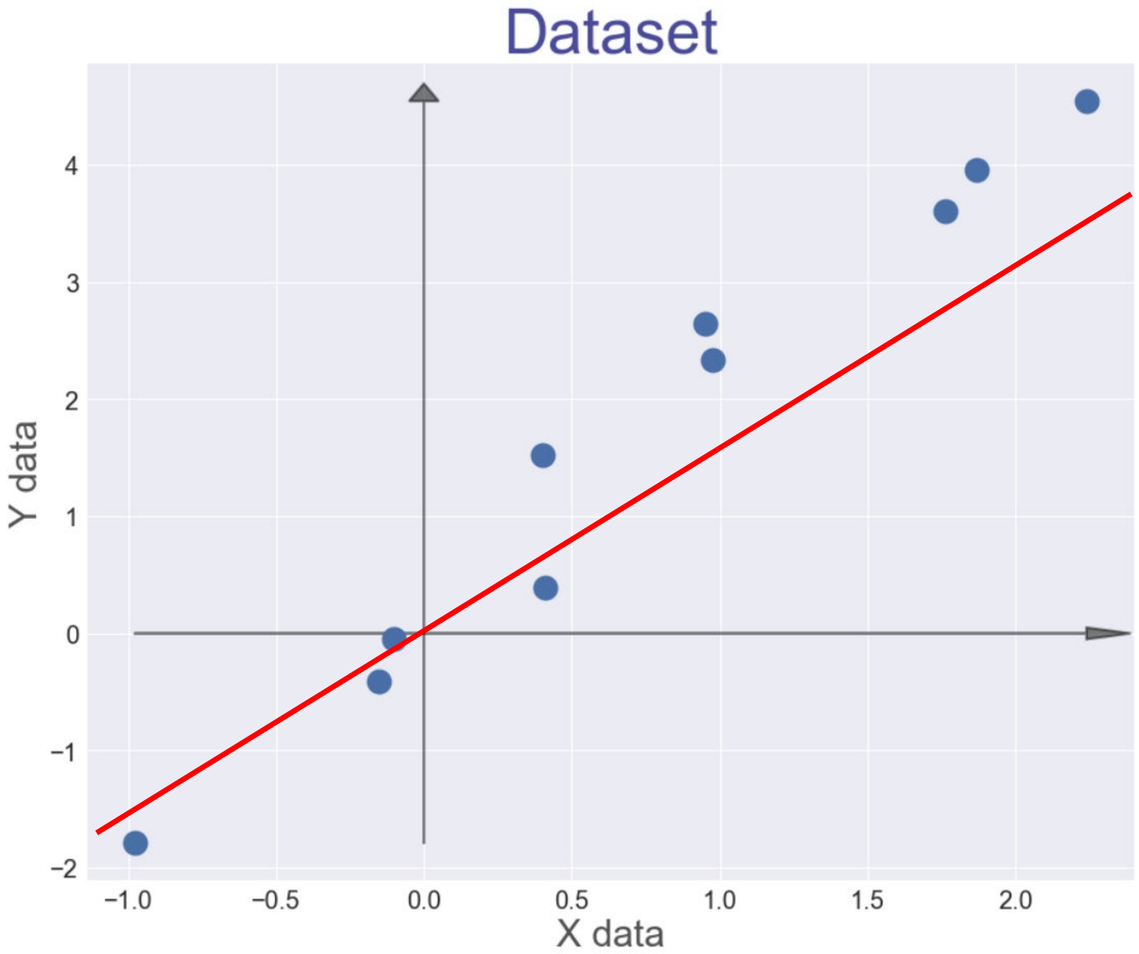
Test

Linear Regression





Loss and Cost



Loss function for one sample

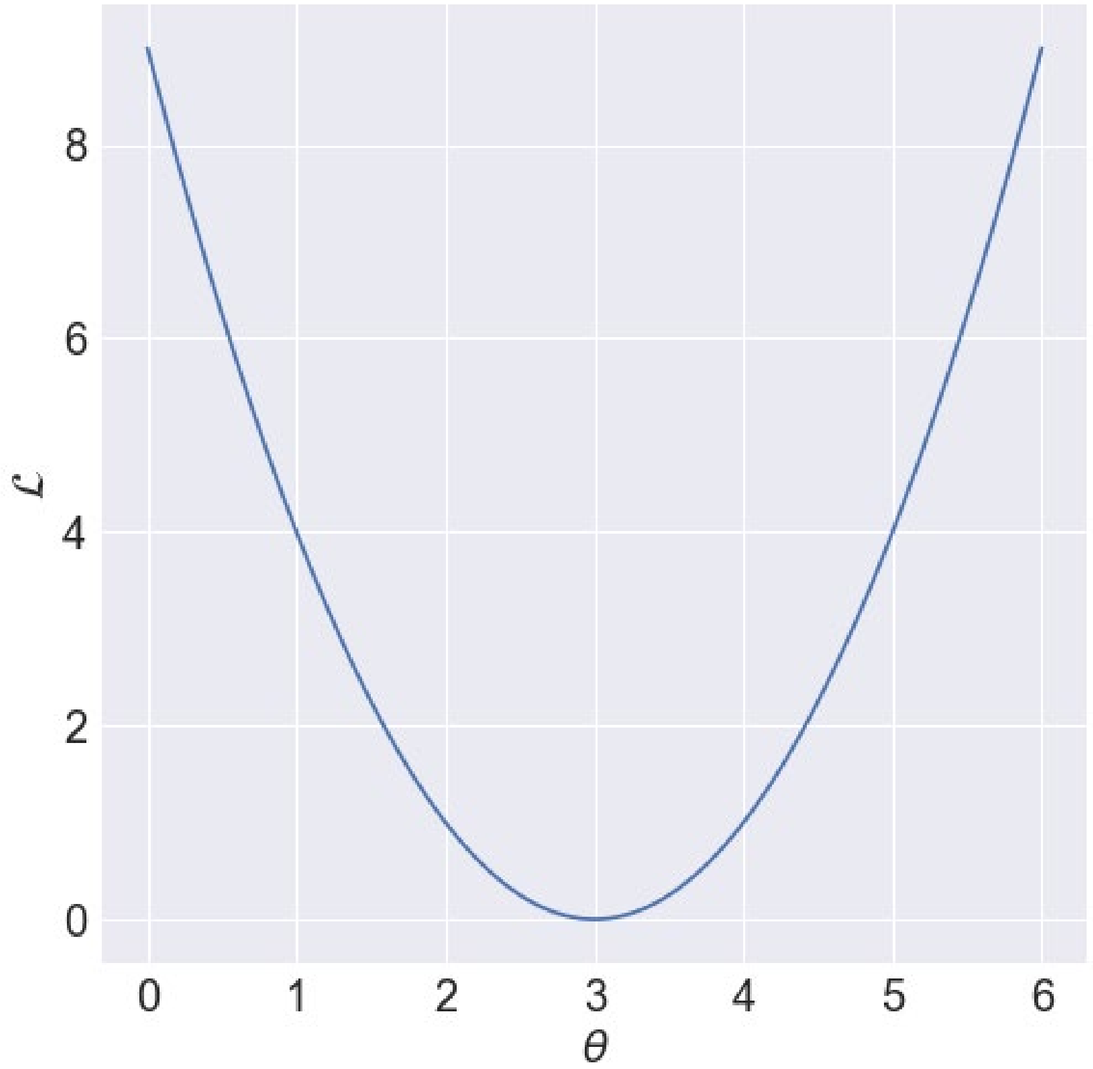
(1) Data sample (x, y) = (1,3) 하나에 대한 loss function을 기술해보자.

(2) Data sample (x, y) = (1,3) 하나에 대한 loss function을 그려보자.

```
import numpy as np
import matplotlib.pyplot as plt

x1 = 1
y1 = 3*x1

th_range = np.linspace(y1 - 3, y1 + 3, 100)
loss_funct = np.power(y1 - th_range*x1, 2)
```

```
fig, ax = plt.subplots(figsize = (5,5))
ax.plot(th_range, loss_funct)
ax.tick_params(axis = 'both', labelsize = 20)
ax.set_xlabel(r'$₩theta$', fontsize = 20)
ax.set_ylabel(r'$₩mathcal{L}$', fontsize = 20)
```

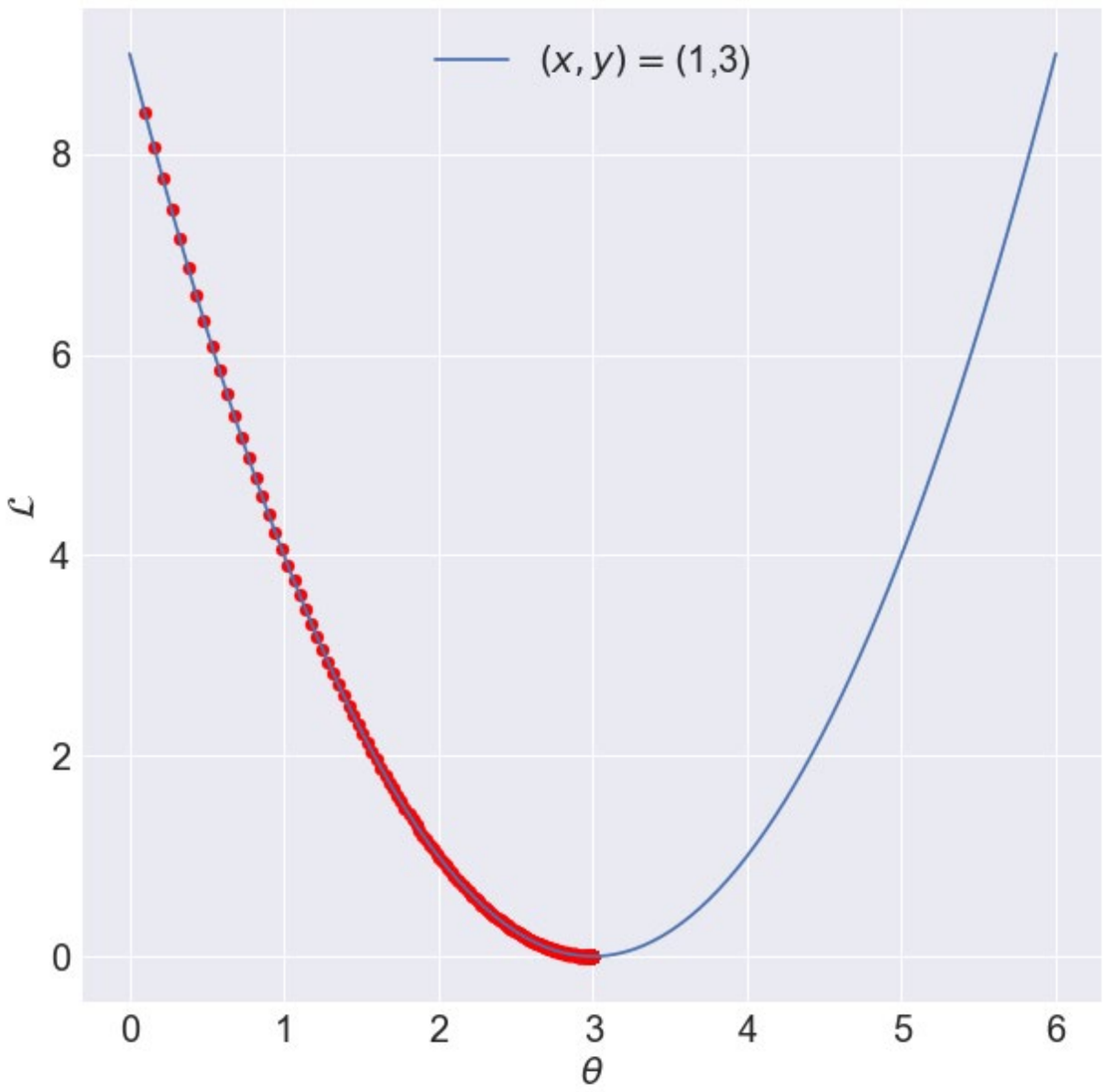
Loss function for one sample



loss function에서 loss가 최소가 되는 θ는?

다시말하면, data sample을 가장 잘 표현하는 θ는 loss function을 최소화시키는 ___이다.

Learning with one sample



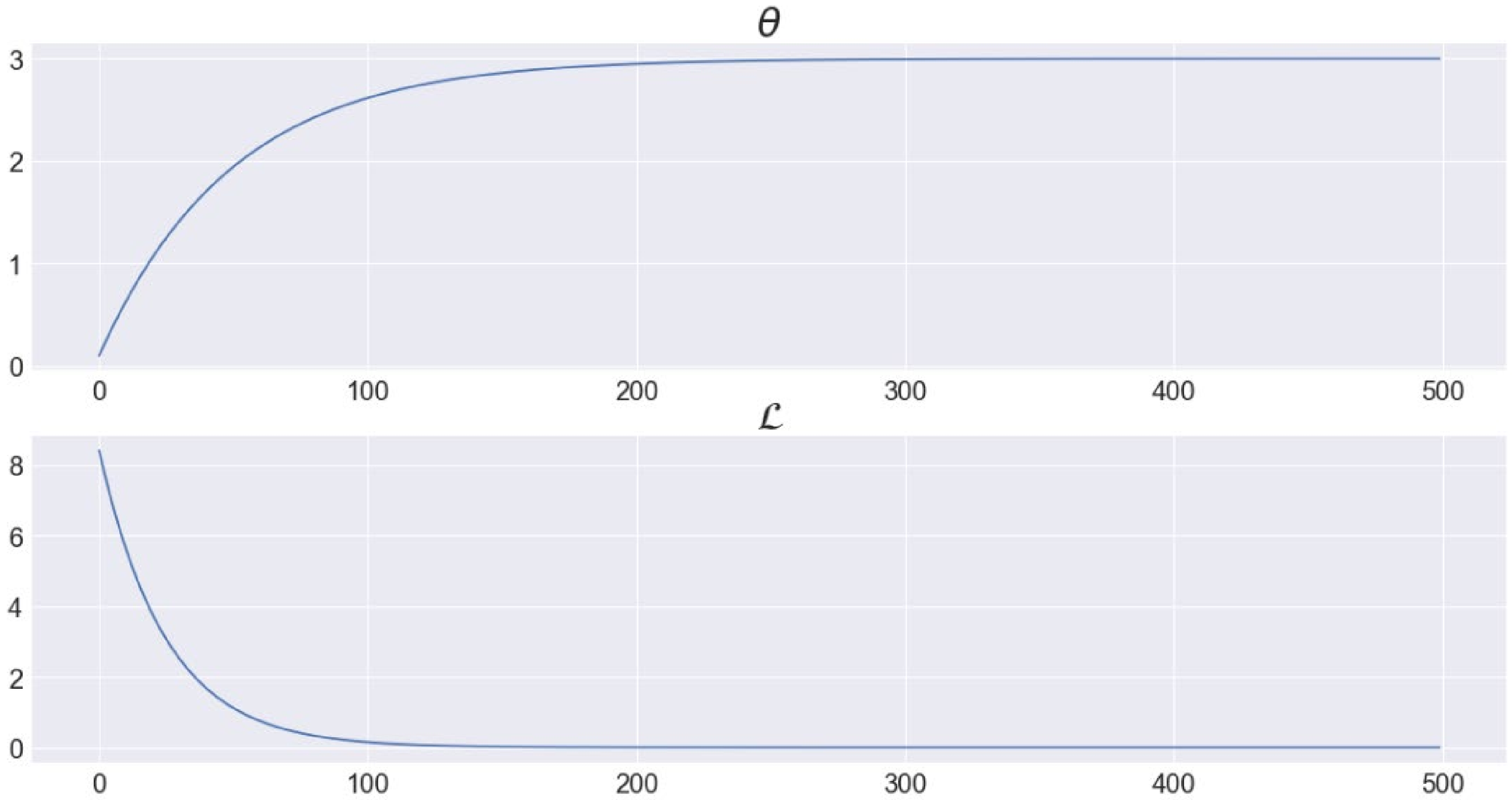
Learning with one sample

위의 x1, y1를 이용하여 predictor를 학습시켜보자.
data sample이 하나이므로 epoch보단 iteration이라는 표현이 적절하다.

```
th = 0.1
lr = 0.01
iterations = 500

th_list = []
loss_list = []

for iteration in range(iterations):
    pred = th*x1
    loss = np.power(y1 - pred, 2)

    th_list.append(th)
    loss_list.append(loss)

    th = th + 2*x1*lr*(y1 - pred)
```

```
fig, ax = plt.subplots(2, 1, figsize = (20,10))
ax[0].plot(th_list)
ax[1].plot(loss_list)
ax[0].set_title(r'$₩theta$', fontsize = 30)
ax[1].set_title(r'$₩mathcal{L}$', fontsize = 30)
for ax_idx in range(2):
    ax[ax_idx].tick_params(axis = 'both', labelsize = 20)
```

Learning with one sample



Learning with different data samples

Data sample (x1, y1) = (0.5,1.5), (x2, y2) = (1,3), (x3, y3) = (2,6)을 이용하여
학습이 진행되는 모습을 비교해보자.

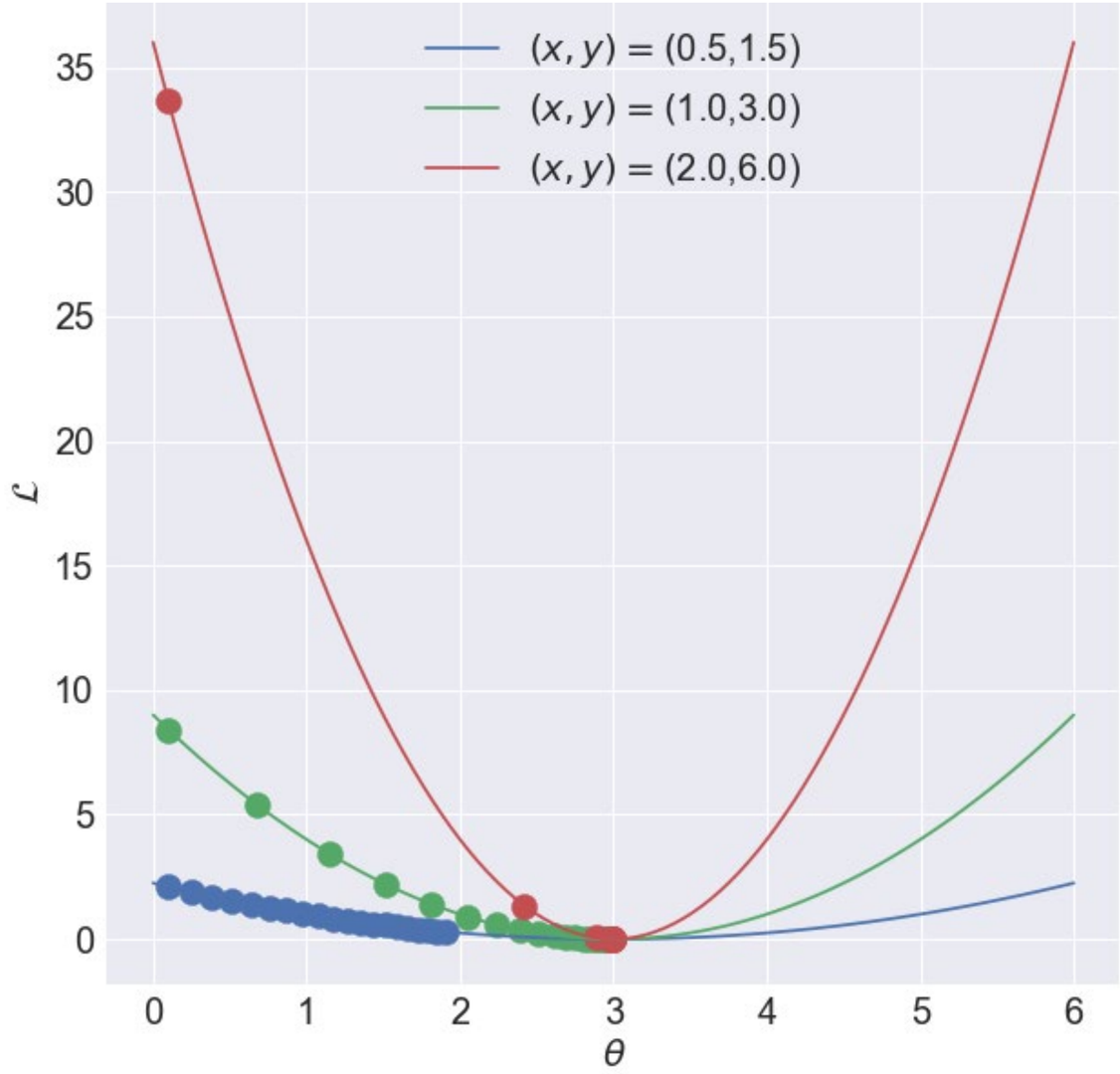
```
def get_loss_funct(x, y, ax):
    th_range = np.linspace(3 - 3, 3 + 3, 100)
    loss_funct = np.power(y - th_range*x, 2)

    ax.plot(th_range,
            loss_funct,
            label = r'$(x,y) = ($' + str(x) + ',' + str(y)+ ')')
    ax.tick_params(axis = 'both', labelsize = 20)
    ax.set_xlabel(r'$₩theta$', fontsize = 20)
    ax.set_ylabel(r'$₩mathcal{L}$', fontsize = 20)
    ax.legend(loc = 'upper center', fontsize = 20)
```

Learning with different data samples

```
def trainer(iterations, lr, x, y):
    th = 0.1
    th_list = []
    loss_list = []

    for iteration in range(iterations):
        pred = th*x
        loss = np.power(y - pred, 2)
        th_list.append(th)
        loss_list.append(loss)

        th = th + 2*x*lr*(y - pred)
    return th_list, loss_list
```

Learning with different data samples

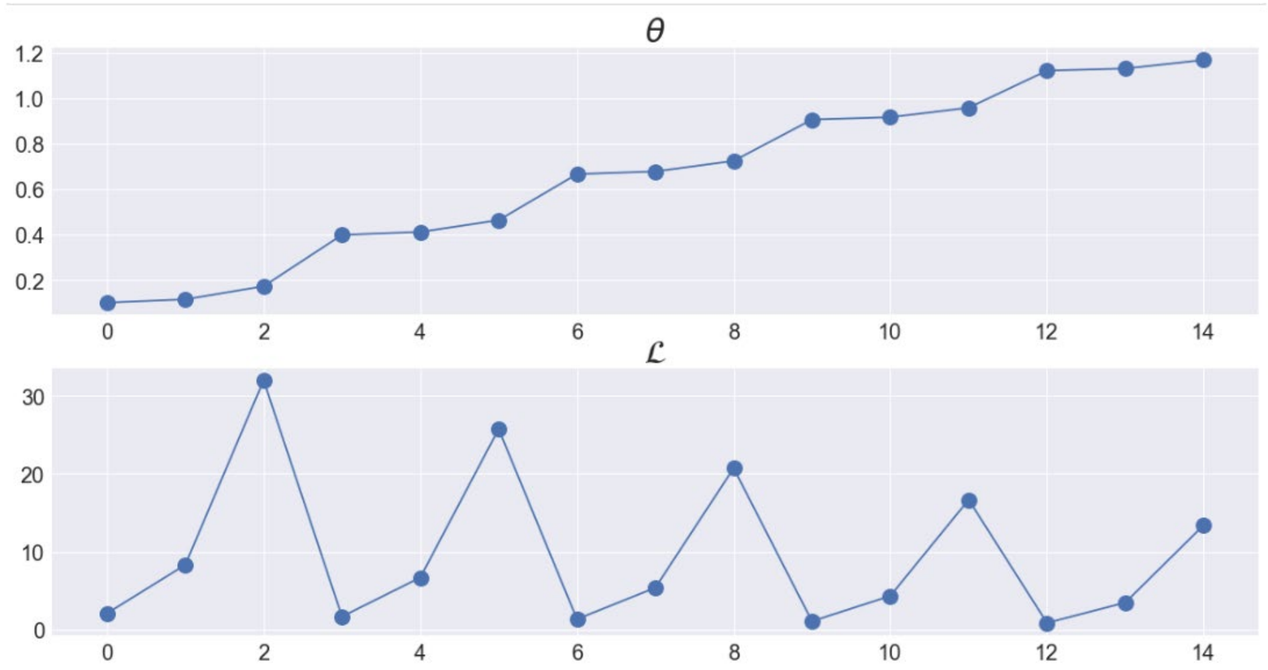
```
x_data = np.array([0.5, 1, 2])
y_data = 3*x_data

fig, ax = plt.subplots(figsize = (10,10))

for x, y in zip(x_data, y_data):
    get_loss_funct(x, y, ax)
    th_list, loss_list = trainer(20, 0.1, x, y)
    ax.scatter(th_list, loss_list, s = 200)
```



Learning with different data samples

```
x_data = np.array([0.5, 1, 2])
y_data = 3*x_data

th = 0.1
lr = 0.01
epochs = 5

th_list = []
loss_list = []

for epoch in range(epochs):
    for x,y in zip(x_data, y_data):
        pred = th*x
        loss = np.power(y - pred, 2)

        th_list.append(th)
        loss_list.append(loss)
        th = th + 2*x*lr*(y - pred)
```

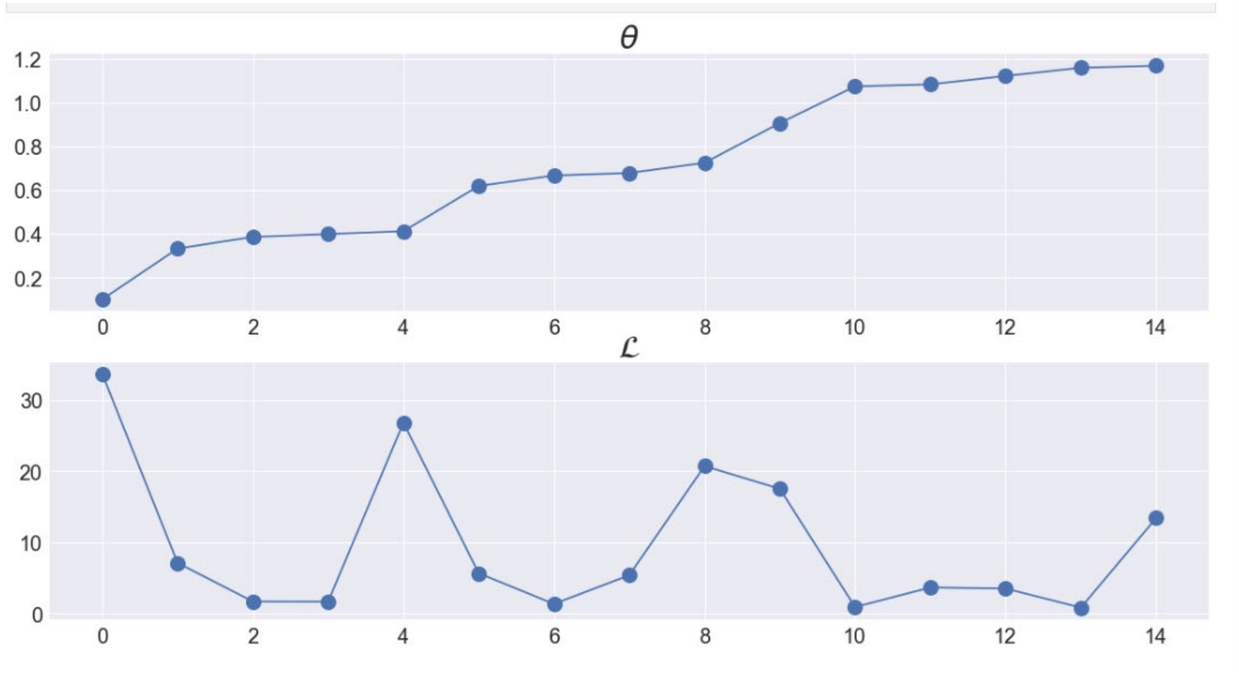
```
fig, ax = plt.subplots(2, 1, figsize = (20,10))
ax[0].plot(th_list, marker = 'o', markersize = 15)
ax[1].plot(loss_list, marker = 'o', markersize = 15)
ax[0].set_title(r'$₩theta$', fontsize = 30)
ax[1].set_title(r'$₩mathcal{L}$', fontsize = 30)
for ax_idx in range(2):
    ax[ax_idx].tick_params(axis = 'both', labelsize = 20)
```



Learning with random shuffling

```
x_data = np.array([0.5, 1, 2]).reshape(-1,1)
y_data = 3*x_data
data = np.hstack((x_data, y_data))

th = 0.1
lr = 0.01
epochs = 5
th_list = []
loss_list = []
for epoch in range(epochs):
    np.random.shuffle(data)
    for x,y in data:
        pred = th*x
        loss = np.power(y - pred, 2)

        th_list.append(th)
        loss_list.append(loss)
        th = th + 2*x*lr*(y - pred)
```

```
fig, ax = plt.subplots(2, 1, figsize = (20,10))
ax[0].plot(th_list, marker = 'o', markersize = 15)
ax[1].plot(loss_list, marker = 'o', markersize = 15)
ax[0].set_title(r'$\theta$', fontsize = 30)
ax[1].set_title(r'$\mathcal{L}$', fontsize = 30)
for ax_idx in range(2):
    ax[ax_idx].tick_params(axis = 'both', labelsize = 2
```



Epoch and batch

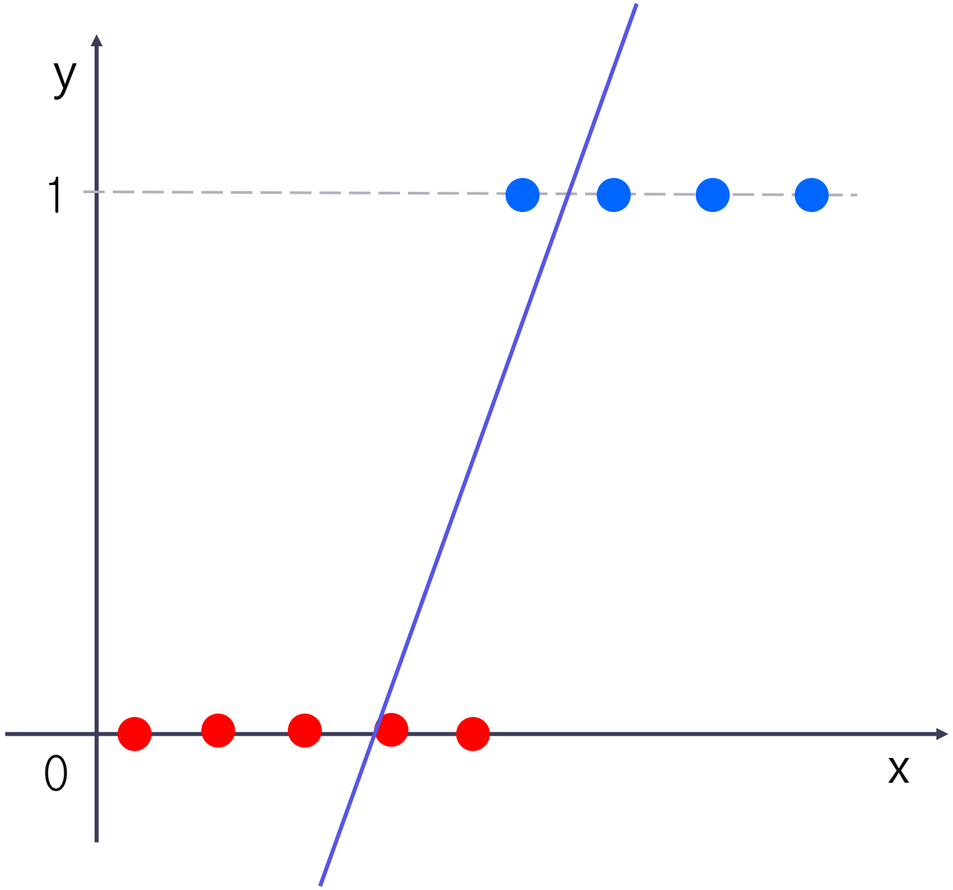
1 Epoch : 모든 데이터 셋을 한 번 학습
1 iteration : 1회 학습

minibatch : 데이터 셋을 batch size 크기로 쪼개서 학습

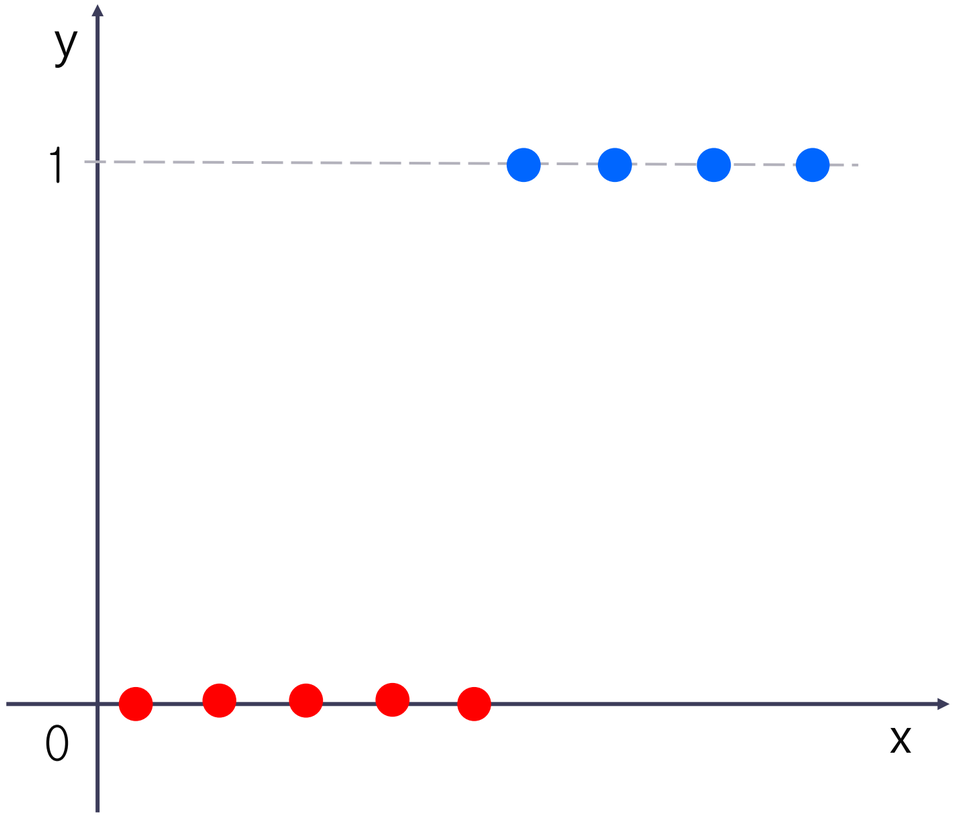
ex) 총 데이터가 100개, batch size가 10이면,
1 iteration = 10개 데이터에 대해서 학습
1 Epoch = 100/batch size = 10 iteration

로지스틱 회귀분석

사망/생존, 실패/성공, 불합격/합격



Linear regression



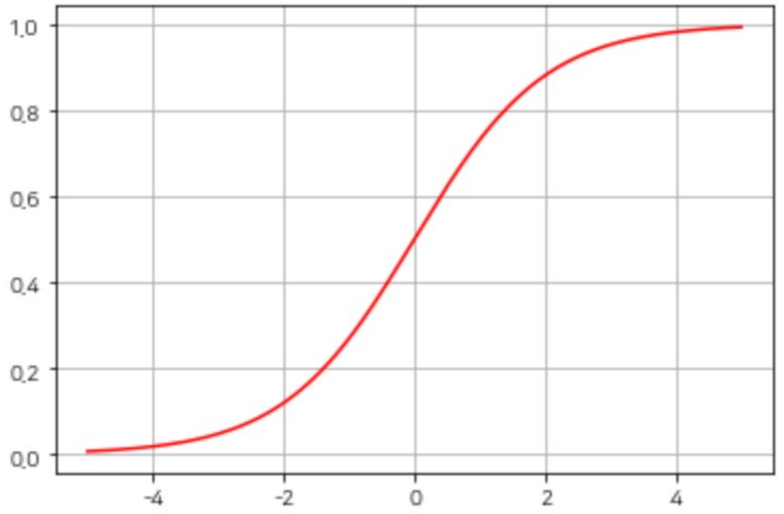
Logistic regression

승산율(오즈비, odd ratio)

odd ratio = 성공확률/실패확률

성공확률을 p라고 하면, 오즈비 = p/(1-p)

로짓함수(logit function)



시그모이드 함수

· 시그모이드 함수

종속변수의 모든 실수 값에 대해

(1) 유한한 구간 (a,b) 사이의 한정된(bounded) 값을 갖는 함수: a<f(x)<b

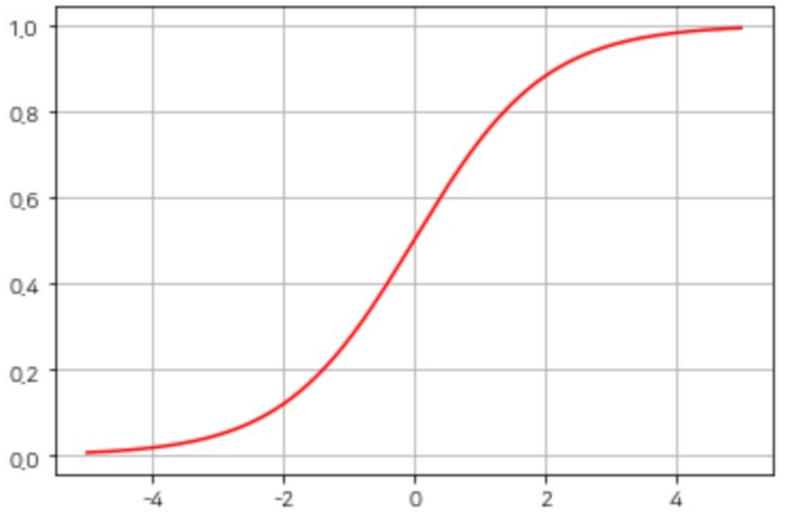
(2) 항상 양의 기울기를 가지는 단조증가하는 함수: a > b→ f(a)>f(b)

을 만족하는 함수 집합을 의미한다. 예) logistic, tanh, error 함수

· 로지스틱 함수

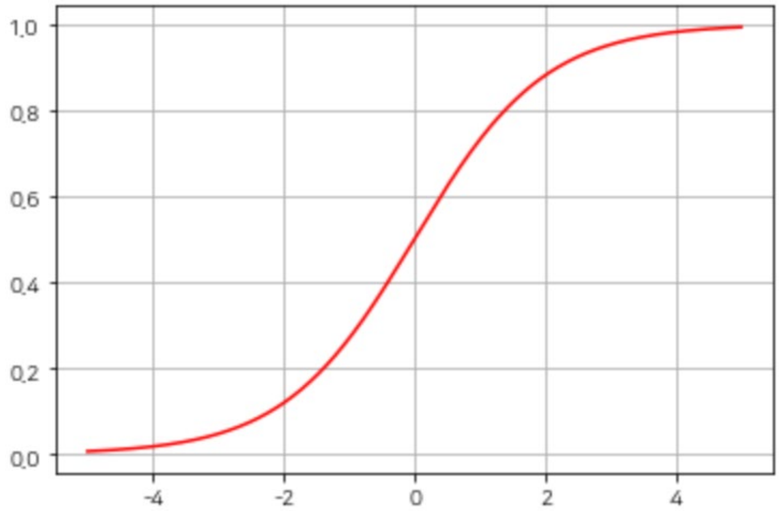
$$\sigma(z) = \frac{1}{1 + exp(-z)}$$

```
:  xx = np.linspace(-5, 5, 1000)
   plt.plot(xx, 1/(1+np.exp(-xx)), 'r-')
   plt.grid()
```



로지스틱 함수

-∞부터 ∞까지의 실수값을 0부터 1사이의 실수값으로
1 대 1 대응시키는 시그모이드함수



로지스틱 회귀분석 모형의 모수 추정

정밀의료 데이터 분석을 위한
머신러닝 부트캠프