Overview

Linear Regression

Director of TEAMLAB
Sungchul Choi

머신러닝의 학습 방법들

- Gradient descent based learning
- Probability theory based learning
- Information theory based learning
- Distance similarity based learning

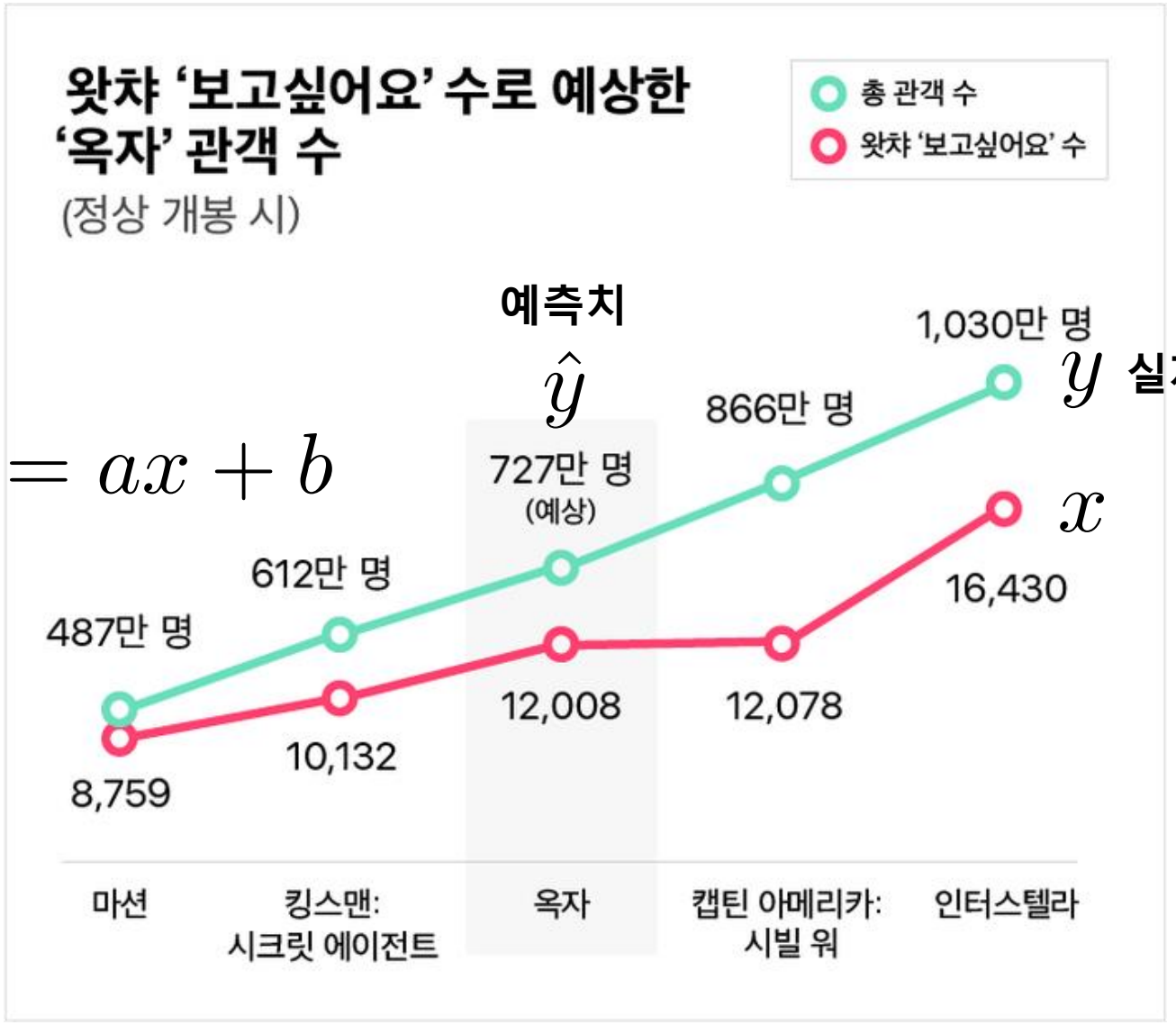
머신러닝의 학습 방법들

- Gradient descent based learning

실제 값과 학습된 모델 예측치의 오차를 최소화

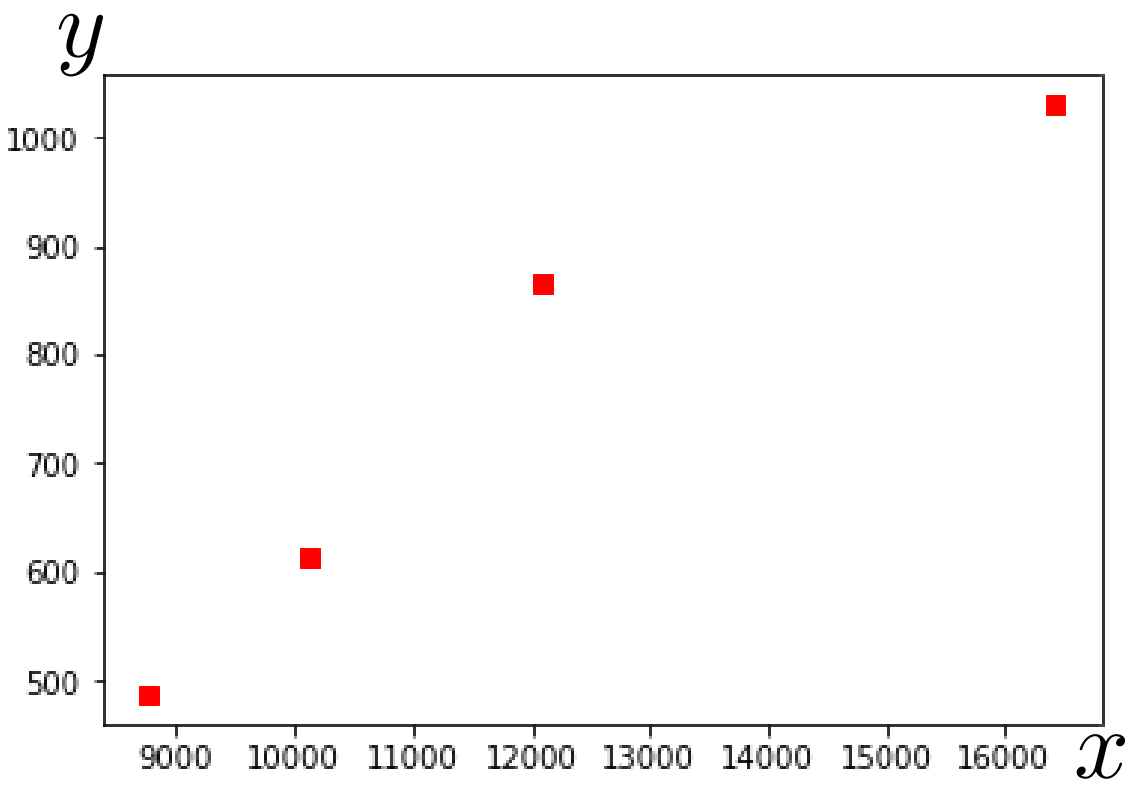
모델의 최적 parameter 찾기가 목적

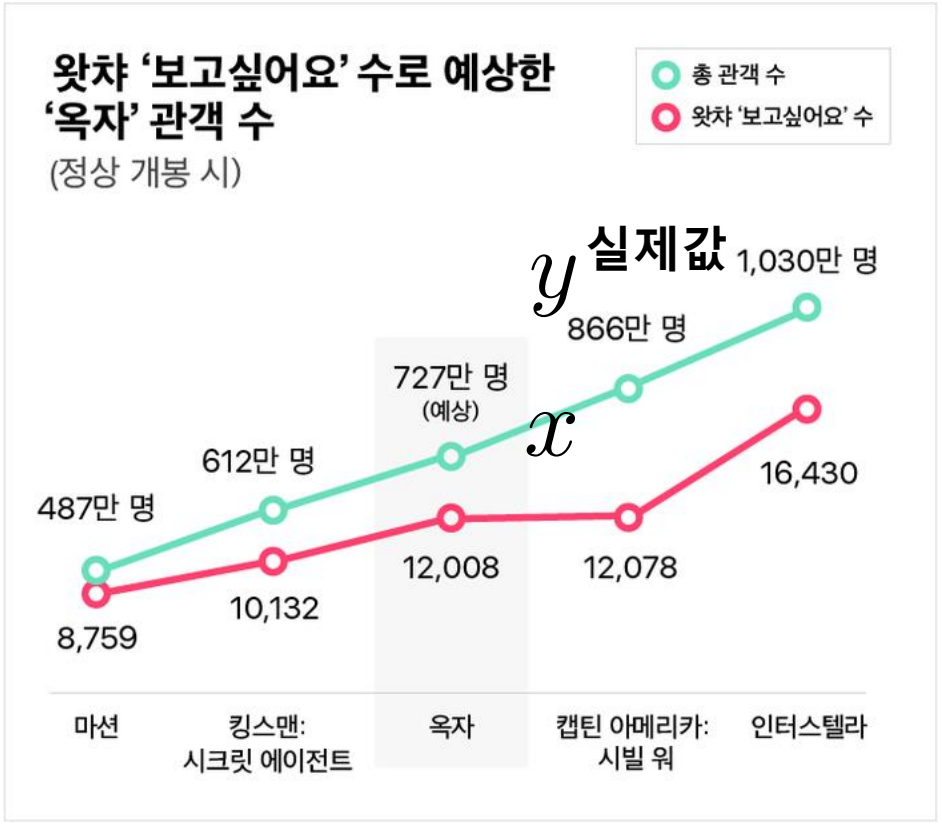
gradient descent 활용



$$\hat{y} = ax + b$$

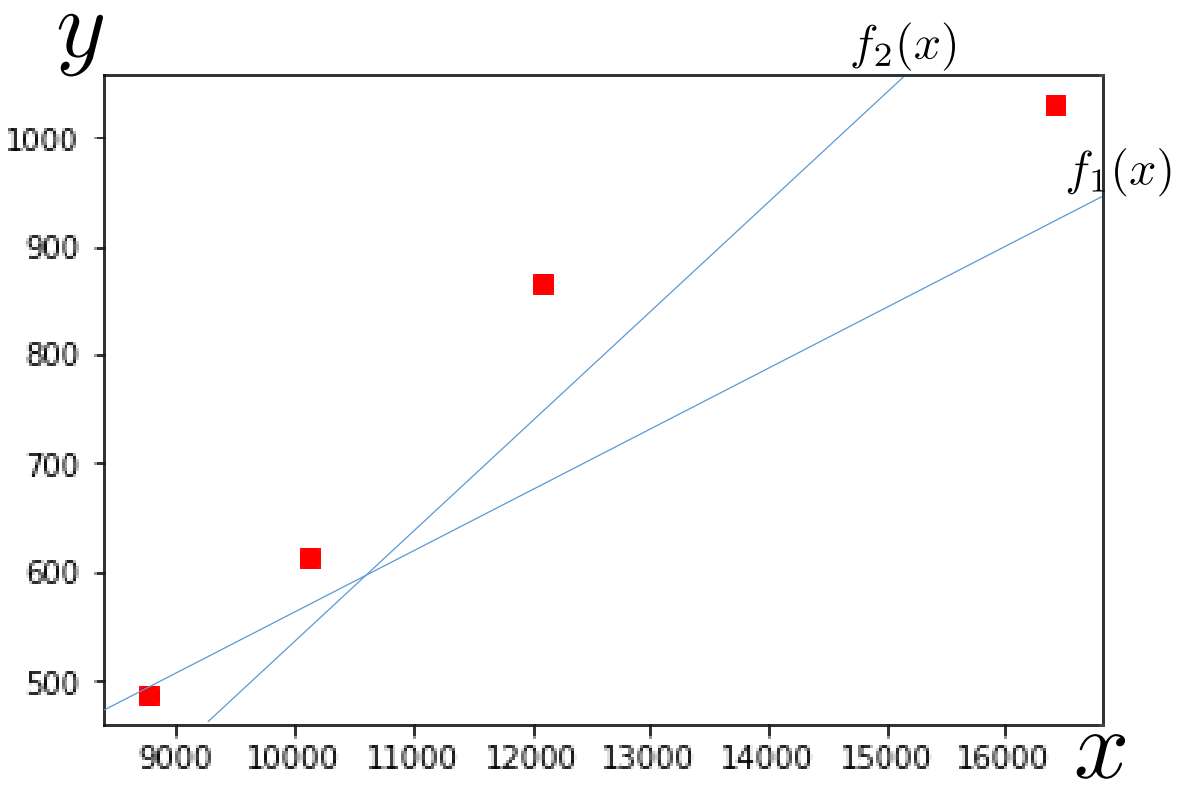
Source: http://platum.kr/archives/83757

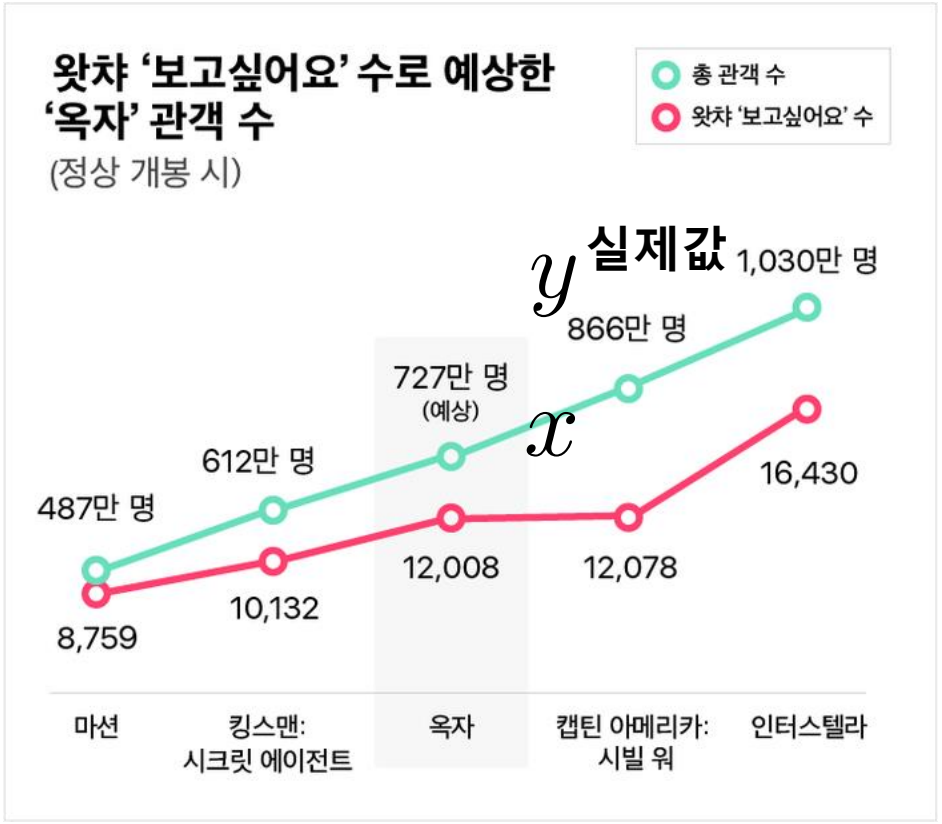




Source: http://platum.kr/archives/83757

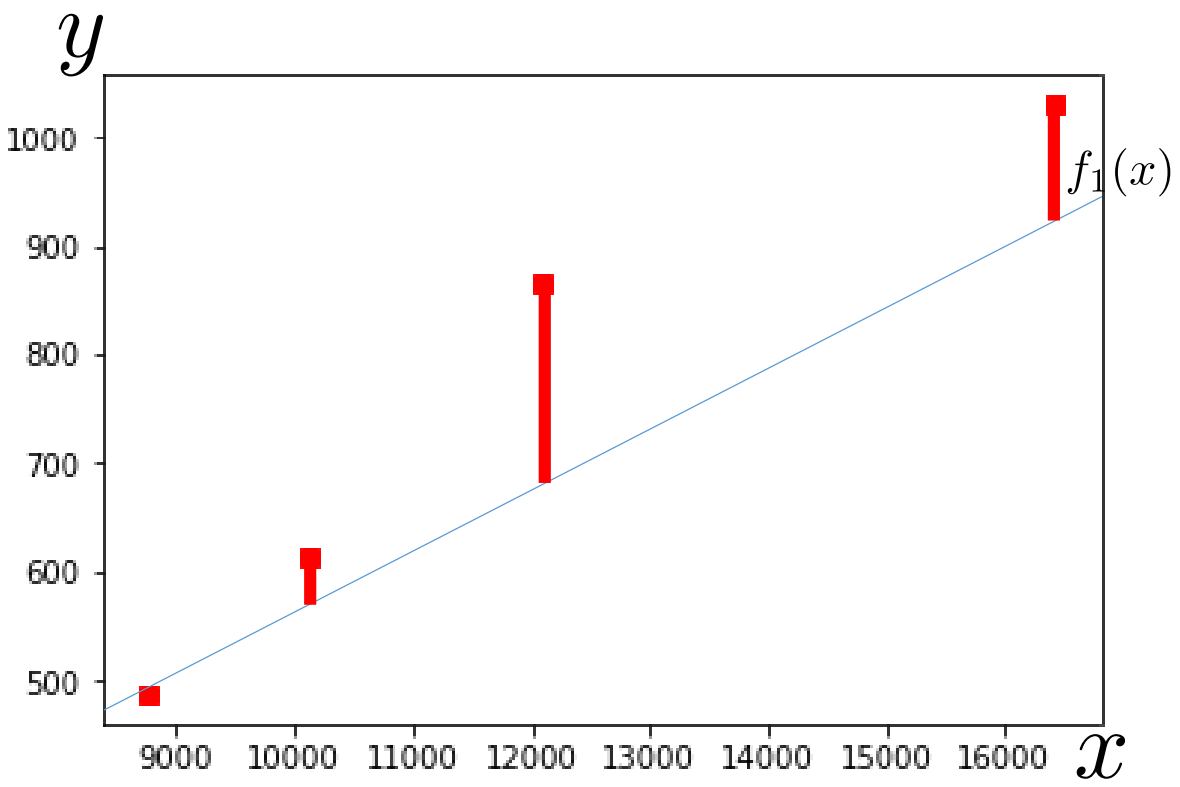
$$f(x) = \hat{y} = ax + b$$

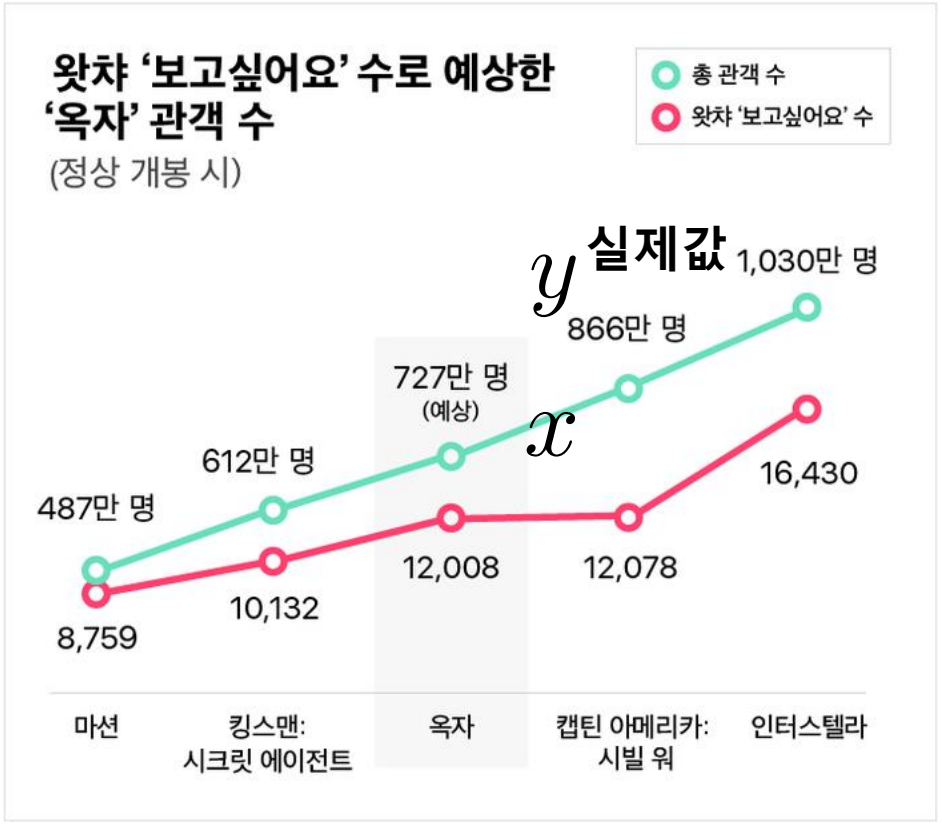




Source: http://platum.kr/archives/83757

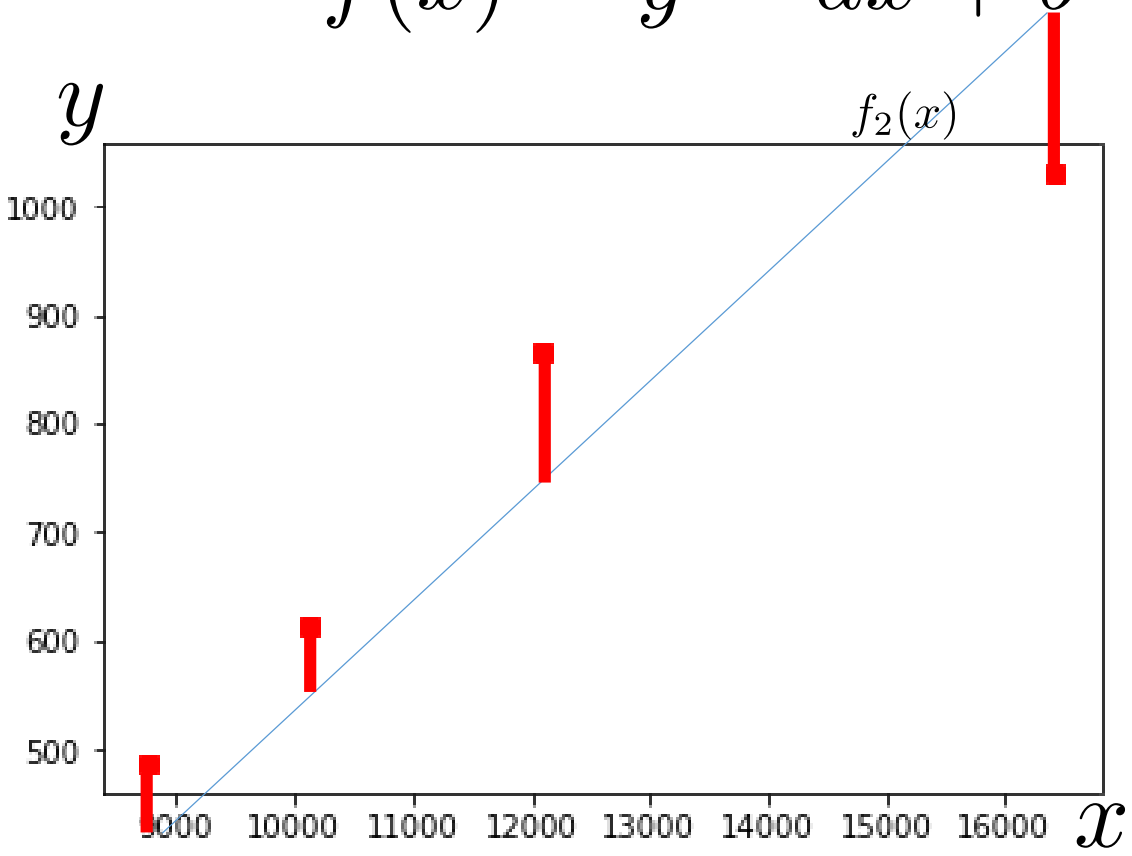
$$f(x) = \hat{y} = ax + b$$

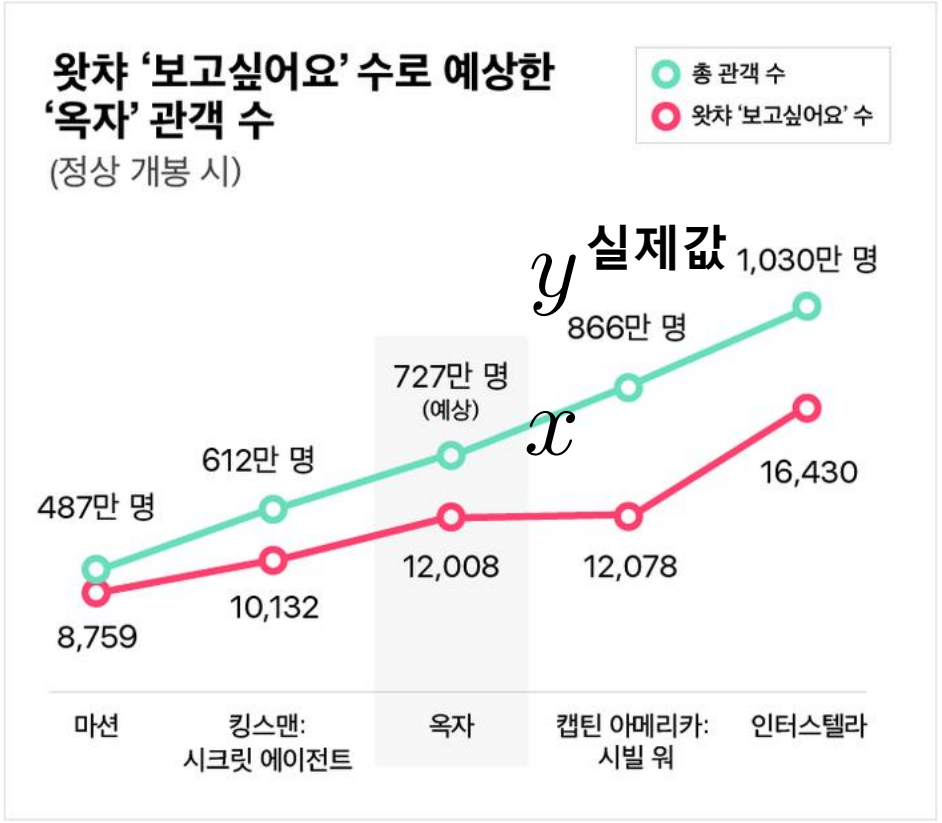




Source: http://platum.kr/archives/83757

$$f(x) = \hat{y} = ax + b$$



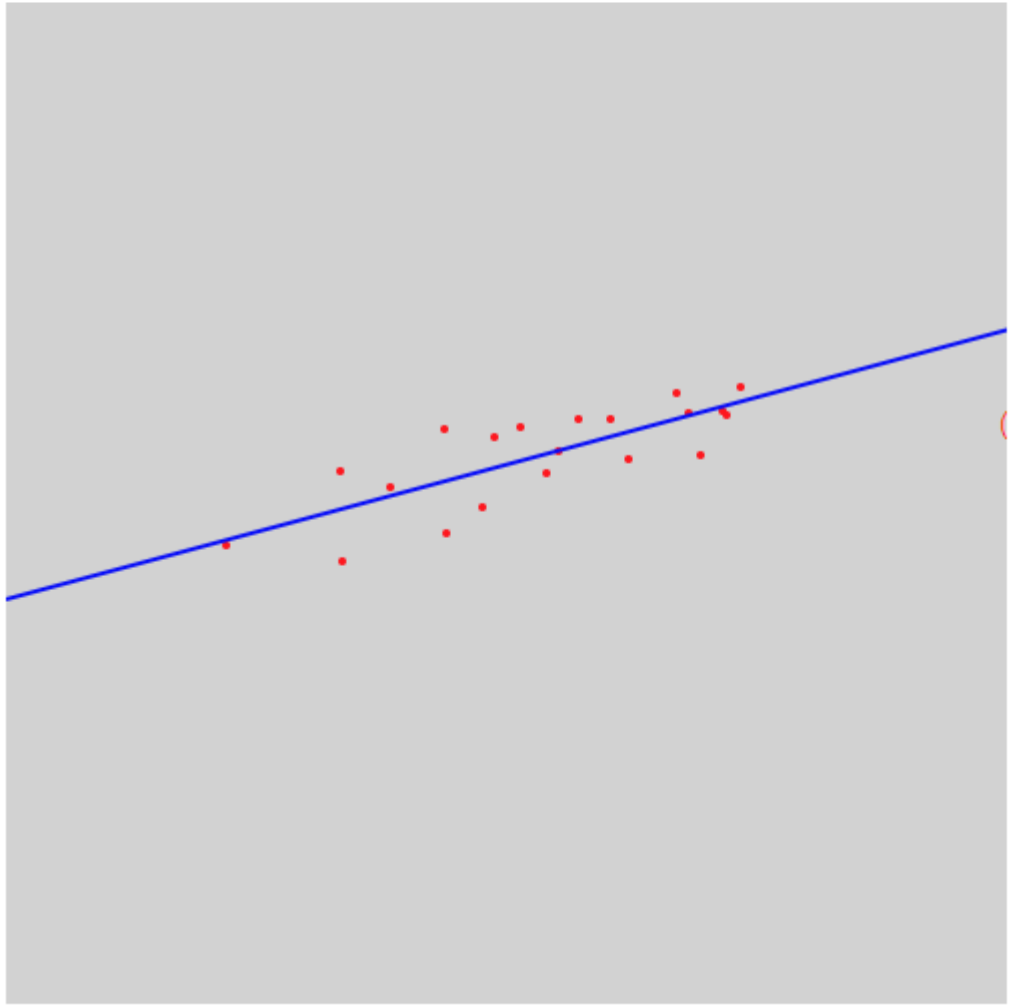


Source: http://platum.kr/archives/83757

- Home
- Linear Regression - Ordinary Least Squares Method

Linear Regression - Ordinary Least Squares Method

Y: -0.2689810286743359 3x+297.9945080961918



https://sujinleeme.github.io/javascript-machine-learning/linear-regression-ols
https://www.facebook.com/sujinlee.me

예측 함수와 실제 값의 오차를 줄여보자!

오차의 합

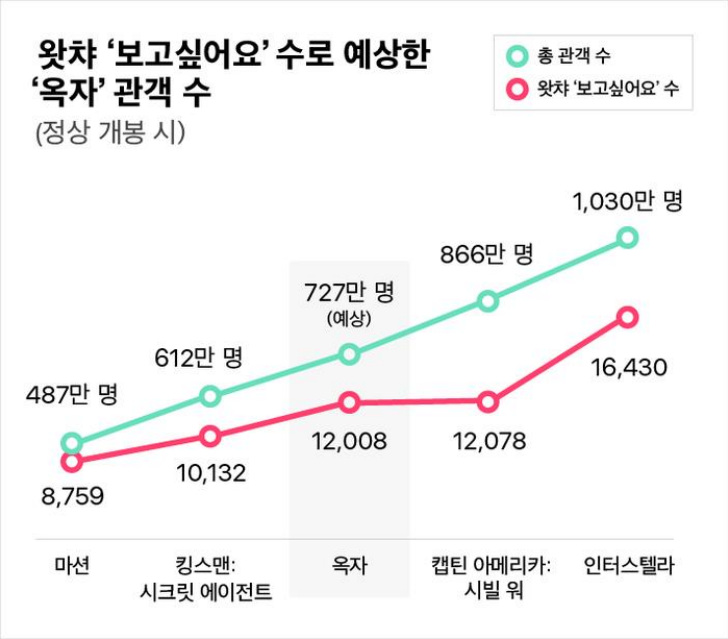
$$(\hat{y}^{(1)} - y^{(1)}) + (\hat{y}^{(2)} - y^{(2)}) + (\hat{y}^{(3)} - y^{(3)}) + (\hat{y}^{(4)} - y^{(4)})$$

오차는 양수 또는 음수 가능 ➔ 상쇄될 수 있음

$$(\hat{y}^{(1)} - y^{(1)})^2 + (\hat{y}^{(2)} - y^{(2)})^2 + (\hat{y}^{(3)} - y^{(3)})^2 + (\hat{y}^{(4)} - y^{(4)})^2$$

제곱의 합으로 변환

$$\sum_{i=1}^{n}(\hat{y}^{(i)} - y^{(i)})^2 \quad \hat{y} = \begin{bmatrix} w_1 \times 8759 + w_0 \\ w_1 \times 10132 + w_0 \\ w_1 \times 12078 + w_0 \\ w_1 \times 16430 + w_0 \end{bmatrix} \quad y = \begin{bmatrix} 487 \\ 612 \\ 866 \\ 1030 \end{bmatrix}$$



$$(\hat{y} - y)^2 = \begin{bmatrix} (w_1 \times 8759 + w_0 - 487)^2 \\ (w_1 \times 10132 + w_0 - 612)^2 \\ (w_1 \times 12078 + w_0 - 866)^2 \\ (w_1 \times 16430 + w_0 - 1030)^2 \end{bmatrix}$$

Squared Error

Squared Error를 최소화 할 수 있는 weight값의 발견

$$\sum_{i=1}^{n}(w_1 x^{(i)} + w_0 \times 1 - y^{(i)})^2$$

최소 또는 최대의 문제 → 미분으로 해결하기

찾고자 하는 값은? w_1, w_0

Human knowledge belongs to the world.

Cost Function

Linear Regression

Director of TEAMLAB
Sungchul Choi

앞으로 우리는

$$f(x) = h_\theta(x)$$

예측 함수를 가설 함수라고 부를 예정

실제값과 가설함수의 차이

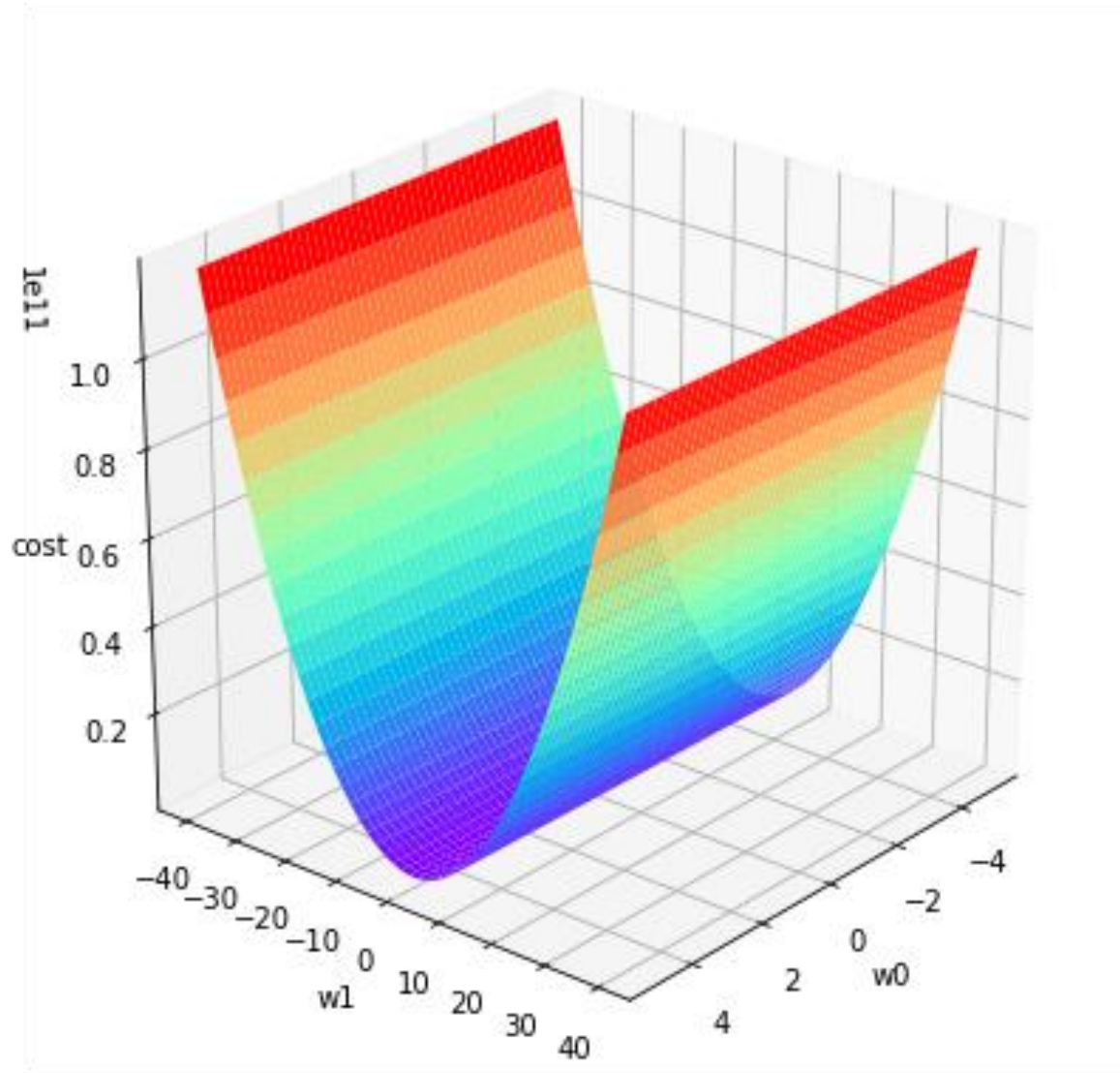
$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Cost function이라고 부를 예정

Cost function에서 구하는 것

$$\arg\min_{\theta} \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

cost function의 최소화를 위한 weight 값


1e11
1.0
0.8
cost
0.6
0.4
0.2
−40
−30
−20
−10
0
10
20
30
40
w1
4
2
0
−2
−4
w0

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)})^2$$

$$\frac{\partial J}{\partial w_0} = \frac{1}{m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)})$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)}) x^{(i)}$$

$$f(x) = \sum_{i=1}^{5} \frac{1}{2} x^2$$

Derivate of $f(g(x))$

$$\frac{df}{dx} = x + x + x + x + x$$

$$\frac{df}{dx} = \frac{df(u)}{du}\frac{du}{dx}, \text{let } u = g(x)$$

$$= \sum_{i=1}^{5} x$$

$$f = (2x-1)^2, \frac{df}{dx} = 2(2x-1) \times 2$$

Review: Scalar derivative rules

Rule	$f(x)$	Scalar derivative notation with respect to x	Example
Constant	c	0	$\frac{d}{dx}99 = 0$
Multiplication by constant	cf	$c\frac{df}{dx}$	$\frac{d}{dx}3x = 3$
Power Rule	x^n	nx^{n-1}	$\frac{d}{dx}x^3 = 3x^2$
Sum Rule	$f + g$	$\frac{df}{dx} + \frac{dg}{dx}$	$\frac{d}{dx}(x^2 + 3x) = 2x + 3$
Difference Rule	$f - g$	$\frac{df}{dx} - \frac{dg}{dx}$	$\frac{d}{dx}(x^2 - 3x) = 2x - 3$
Product Rule	fg	$f\frac{dg}{dx} + \frac{df}{dx}g$	$\frac{d}{dx}x^2x = x^2 + x2x = 3x^2$
Chain Rule	$f(g(x))$	$\frac{df(u)}{du}\frac{du}{dx}$, let $u = g(x)$	$\frac{d}{dx}ln(x^2) = \frac{1}{x^2}2x = \frac{2}{x}$

http://parrt.cs.usfca.edu/doc/matrix-calculus/index.html

weights의 최적값 컴퓨터가 찾는 방법

- 연립방적식 풀기 (normal equation)
- gradient descent

Human knowledge belongs to the world.

Normal equation

Linear Regression

Director of TEAMLAB
Sungchul Choi

Normal equation

Cost Function을 최소화하는 방법

$$\arg\min_{\theta} \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$
\begin{aligned}
y^{(1)} &= w_0 + w_1 x^{(1)} + \epsilon^{(1)} \\
y^{(2)} &= w_0 + w_1 x^{(2)} + \epsilon^{(2)} \\
y^{(3)} &= w_0 + w_1 x^{(3)} + \epsilon^{(3)} \\
y^{(4)} &= w_0 + w_1 x^{(4)} + \epsilon^{(4)} \\
y^{(5)} &= w_0 + w_1 x^{(5)} + \epsilon^{(5)}
\end{aligned}
\qquad
\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \\ y^{(5)} \end{bmatrix}
\qquad
\mathbf{X} = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ 1 & x^{(3)} \\ 1 & x^{(4)} \\ 1 & x^{(5)} \end{bmatrix}
$$

$$
\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}
\qquad
\mathbf{y} = \mathbf{X}\mathbf{w}
$$

$$\mathbf{y} = \mathbf{X}\mathbf{w}$$

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \\ y^{(5)} \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ 1 & x^{(3)} \\ 1 & x^{(4)} \\ 1 & x^{(5)} \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

$$J = \frac{1}{2}\sum_{i=1}^{m}(w_1 x^{(i)} + w_0 - y^{(i)})^2$$

$$\frac{\partial J}{\partial w_0} = \sum(w_1 x^{(i)} + w_0 - y^{(i)}) = 0$$

$$\frac{\partial J}{\partial w_1} = \sum(w_1 x^{(i)} + w_0 - y^{(i)})x^{(i)} = 0$$

위 식을 만족하는 $\hat{w}_j$ 값을 구하기

$$\hat{w}_0 m + \hat{w}_1 \sum x^{(i)} = \sum y^{(i)}$$

$$\hat{w}_0 \sum x^{(i)} + \hat{w}_1 \sum (x^{(i)})^2 = \sum y^{(i)} x^{(i)}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} m & \sum x^{(i)} \\ \sum x^{(i)} & \sum (x^{(i)})^2 \end{bmatrix}$$

$$(\mathbf{X}^T\mathbf{X})\hat{\mathbf{w}} = \mathbf{X}^T\mathbf{y}$$

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\mathbf{X} = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ 1 & x^{(3)} \\ 1 & x^{(4)} \\ 1 & x^{(5)} \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \\ y^{(5)} \end{bmatrix}$$

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} m & \sum x^{(i)} \\ \sum x^{(i)} & \sum (x^{(i)})^2 \end{bmatrix} = \begin{bmatrix} m & m\bar{x} \\ m\bar{x} & \sum (x^{(i)})^2 \end{bmatrix}$$

$$\begin{aligned} |\mathbf{X}^T\mathbf{X}| &= m\sum (x^{(i)})^2 - (m\bar{x})^2 \\ &= m(\sum (x^{(i)})^2 - m\bar{x}^2) \\ &= m\sum (x^{(i)} - \bar{x})^2 \end{aligned}$$

$$\sigma^2 = \frac{\sum (X_i - \mu)^2}{N}$$

$$\begin{aligned} \sum (X_i - \mu)^2 &= \sum (X_i^2 - 2X_i\mu + \mu^2) \\ &= \sum X_i^2 - \sum 2X_i\mu + \sum \mu^2 \\ &= \sum X_i^2 - 2\mu \sum X_i + N\mu^2 \\ &= \sum X_i^2 - 2\mu N\mu_i + N\mu^2 \\ &= \sum X_i^2 - 2N\mu^2 + N\mu^2 \\ &= \sum X_i^2 - N\mu^2 \end{aligned}$$

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} m & \sum x^{(i)} \\ \sum x^{(i)} & \sum (x^{(i)})^2 \end{bmatrix} = \begin{bmatrix} m & m\bar{x} \\ m\bar{x} & \sum (x^{(i)})^2 \end{bmatrix}$$

$$(\mathbf{X}^T\mathbf{X})^{-1} = \frac{1}{m\sum (x^{(i)} - \bar{x})^2} \begin{bmatrix} \sum (x^{(i)})^2 & -m\bar{x} \\ -m\bar{x} & m \end{bmatrix}$$

$$= \frac{1}{\sum (x^{(i)} - \bar{x})^2} \begin{bmatrix} \sum (x^{(i)})^2/m & -\bar{x} \\ -\bar{x} & m \end{bmatrix}$$

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$(\mathbf{X}^T\mathbf{X})^{-1} = \frac{1}{m\sum(x^{(i)} - \bar{x})^2}\begin{bmatrix}\sum(x^{(i)})^2 & -m\bar{x} \\ -m\bar{x} & m\end{bmatrix}$$

$$= \frac{1}{\sum(x^{(i)} - \bar{x})^2}\begin{bmatrix}\sum(x^{(i)})^2/m & -\bar{x} \\ -\bar{x} & m\end{bmatrix}$$

$$\mathbf{X}^T\mathbf{y} = \begin{bmatrix}\sum y^{(i)} \\ \sum x^{(i)}y^{(i)}\end{bmatrix} \quad \mathbf{X} = \begin{bmatrix}1 & x^{(1)} \\ 1 & x^{(2)} \\ 1 & x^{(3)} \\ 1 & x^{(4)} \\ 1 & x^{(5)}\end{bmatrix} \quad \mathbf{y} = \begin{bmatrix}y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \\ y^{(5)}\end{bmatrix}$$

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \begin{bmatrix} \hat{w}_0 \\ \hat{w}_1 \end{bmatrix}$$

$$= \frac{1}{\sum (x^{(i)} - \bar{x})^2} \begin{bmatrix} \sum (x^{(i)})^2/m & -\bar{x} \\ -\bar{x} & m \end{bmatrix} \begin{bmatrix} \sum y^{(i)} \\ \sum x^{(i)} y^{(i)} \end{bmatrix}$$

$$\hat{w}_1 = \frac{\sum x^{(i)} y^{(i)} - m\bar{x}\bar{y}}{\sum (x^{(i)} - \bar{x})^2}$$

$$\hat{w}_0 = \bar{y} - \hat{w}_1 \bar{x}$$

여러개의 변수 일 경우?

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} m & \sum x^{(i)} \\ \sum x^{(i)} & \sum (x^{(i)})^2 \end{bmatrix}$$

가 확대됨

결론

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Normal equation

- $\mathrm{X}^T\mathrm{X}$의 역행렬이 존재할 때 사용
- Iteration 등 사용자 지정 parameter가 없음
- Feature가 많으면 계산 속도가 느려짐

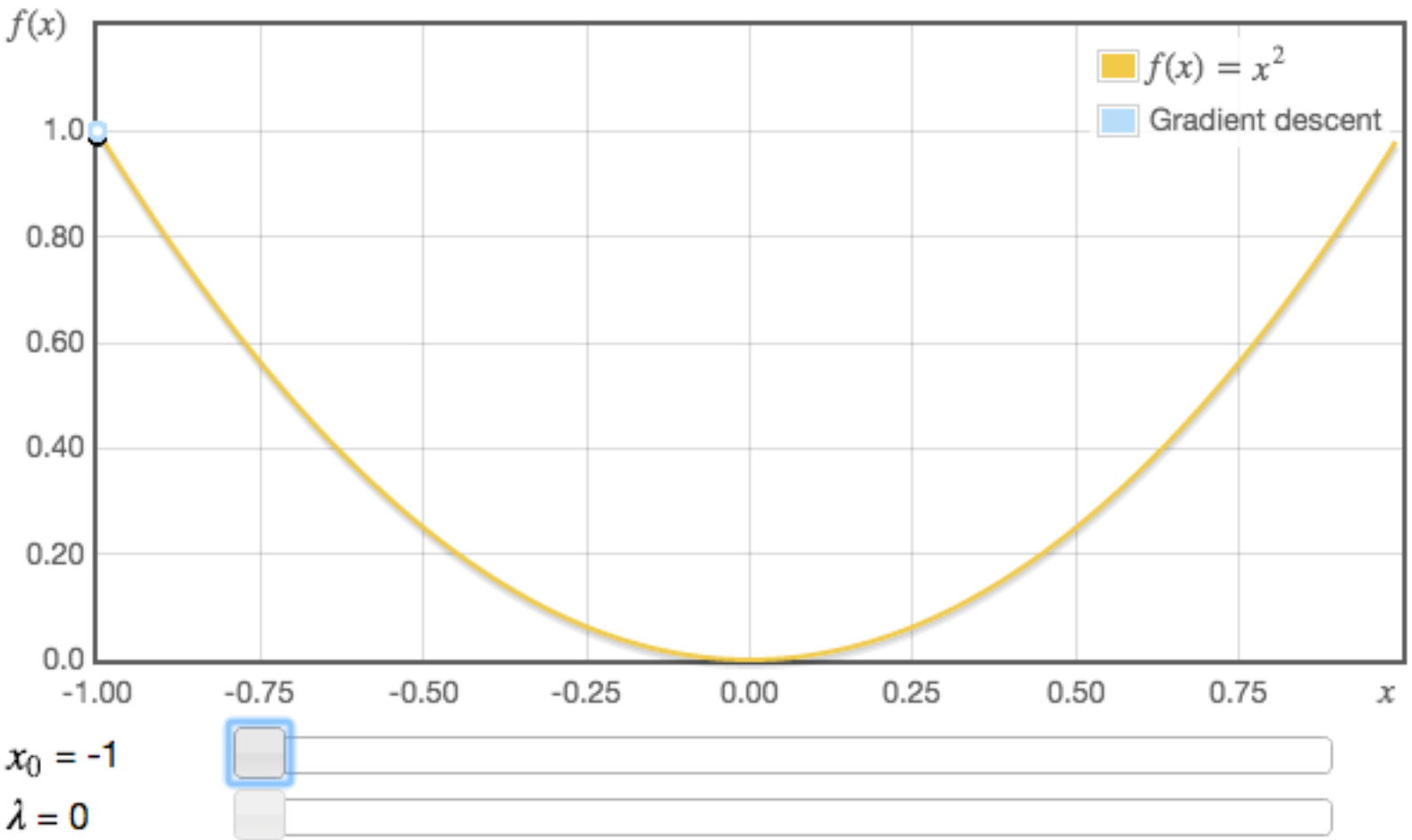
Human knowledge belongs to the world.

Gradient Descent

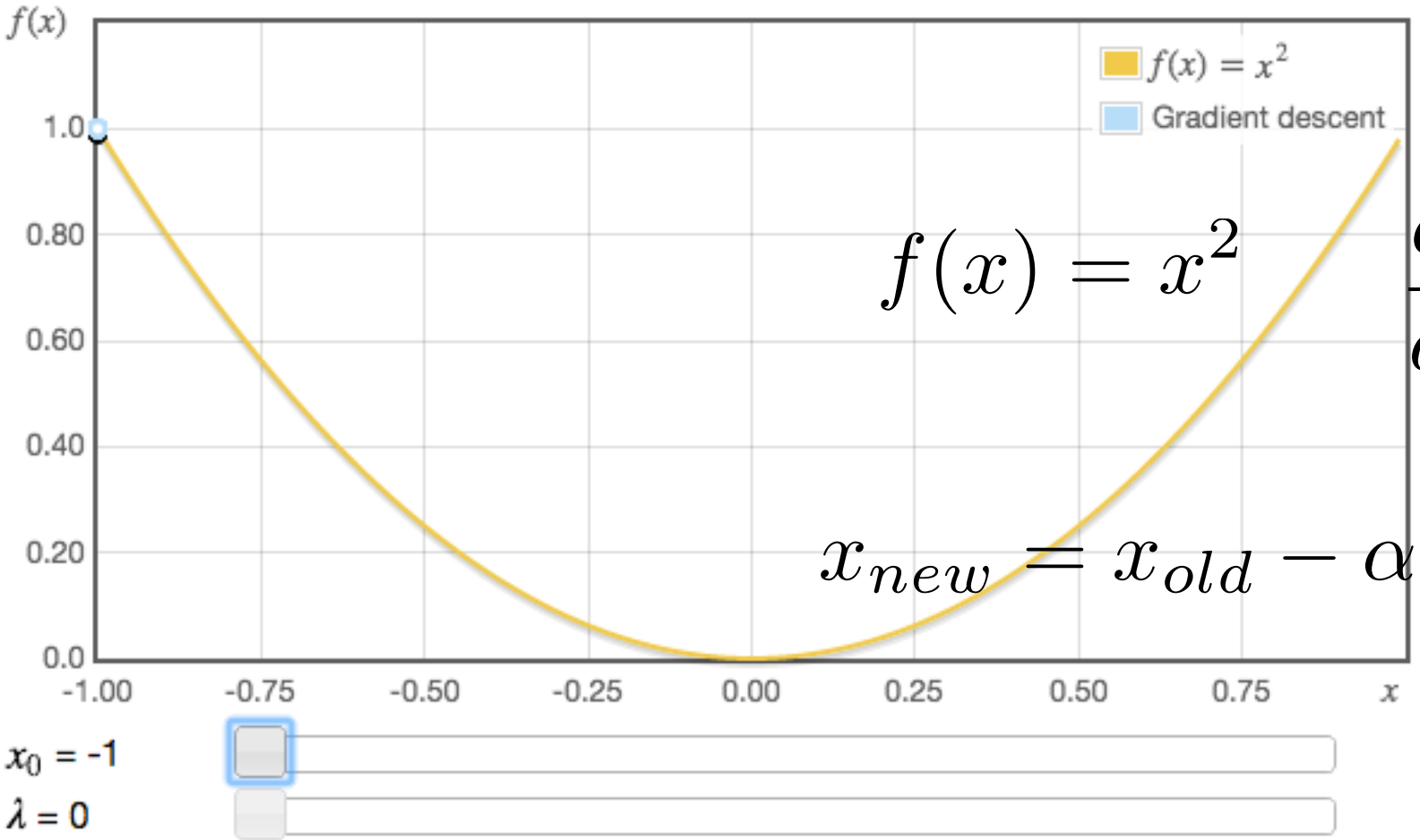
Linear Regression

Director of TEAMLAB
Sungchul Choi

컴퓨터에 x^2 의 최적값 찾기



컴퓨터에 x^2 의 최적값 찾기



$$f(x) = x^2$$

$$\frac{dy}{dx} = 2x$$

$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

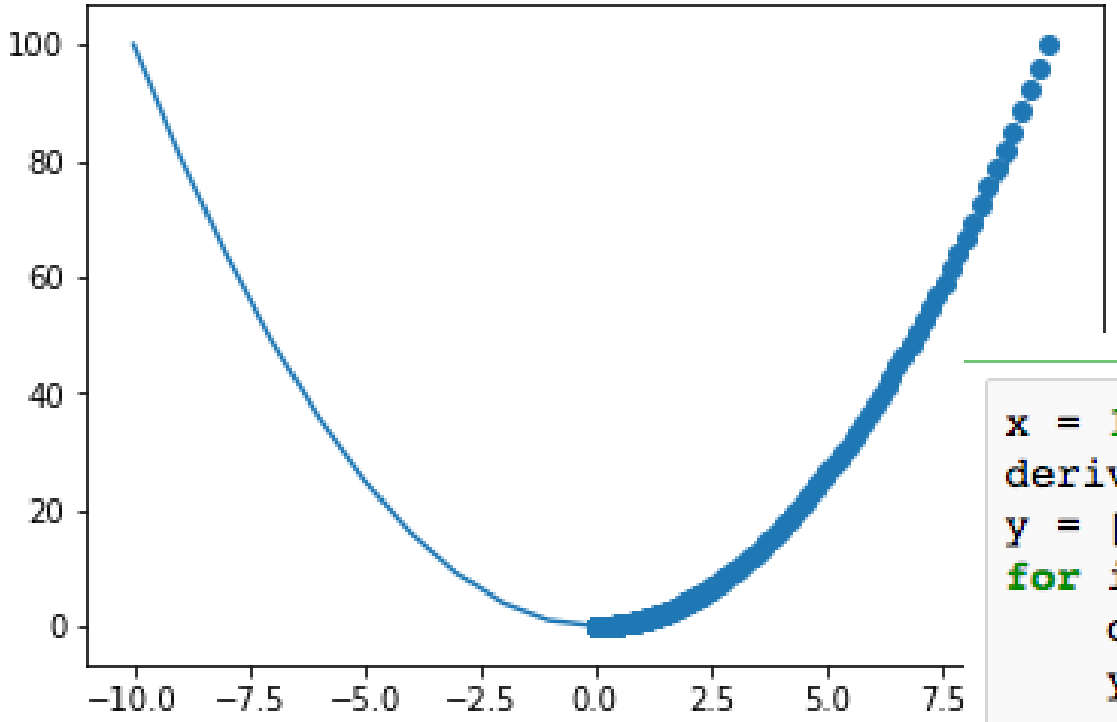
컴퓨터에 x^2 의 최적값 찾기

$$f(x) = x^2 \qquad \frac{dy}{dx} = 2x$$

$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

$$\begin{bmatrix} 1 \\ 1 - 0.1 * 2 * 1 = 0.8 \\ 0.8 - 0.1 * 2 * 0.8 = 0.64 \\ 0.64 - 0.1 * 2 * 0.64 = 0.512 \\ \vdots \end{bmatrix}$$

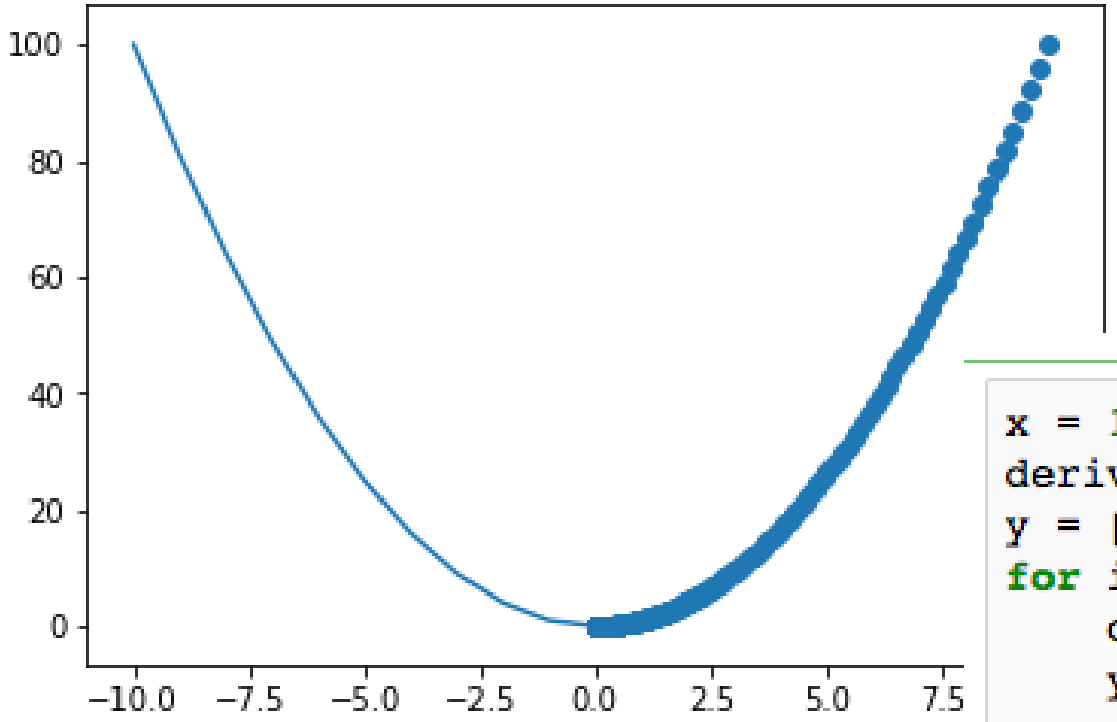
컴퓨터에 x^2 의 최적값 찾기




```python
x = 10
derivative = []
y = []
for i in range(1000):
    old_value = x
    y.append(old_value ** 2)
    derivative.append(old_value - 0.01 * 2 * old_value)
    x = old_value - 0.01 *2* old_value
```

$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

컴퓨터에 x^2 의 최적값 찾기




```
x = 10
derivative = []
y = []
for i in range(1000):
    old_value = x
    y.append(old_value ** 2)
    derivative.append(old_value - 0.0
    x = old_value - 0.01 *2* old_value
```

$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

gradient	v
	10
20	9.8
19.6	9.604
9.604	9.50796
9.50796	9.41288
9.41288	9.318752
9.318752	9.225564
9.225564	9.133308
9.133308	9.041975
9.041975	8.951556
8.951556	8.86204
8.86204	8.77342
8.77342	8.685685
8.685685	8.598829
8.598829	8.51284
8.51284	8.427712
8.427712	8.343435

정해야 하는 것

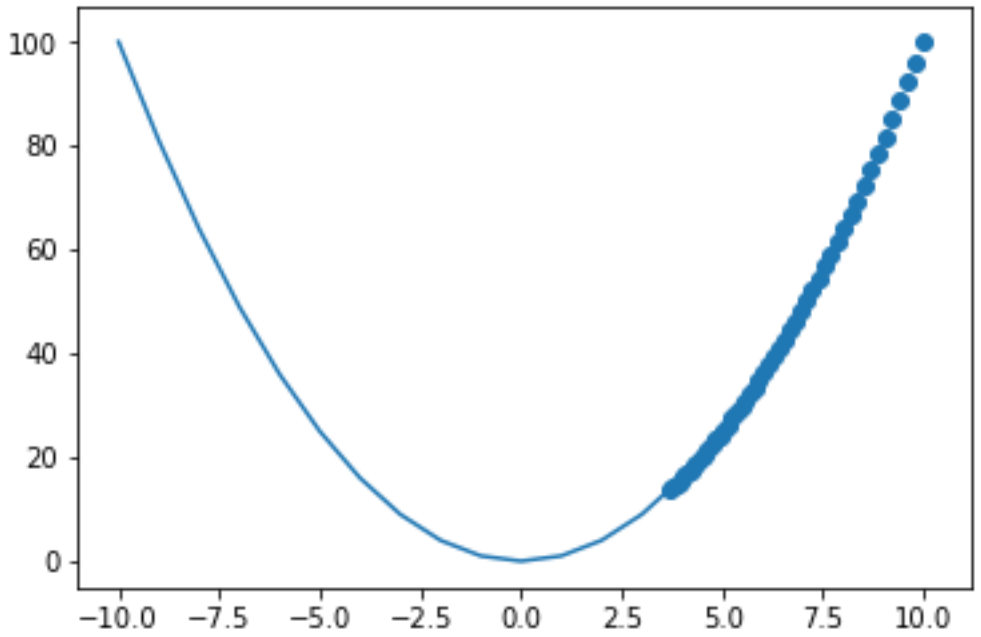
$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

Learning rate에 대한 선정

```
x = 10
derivative = []
y = []
for i in range(1000):
    old_value = x
    y.append(old_value ** 2)
    derivative.append(old_value - 0.01 * 2 * old_value)
    x = old_value - 0.01 *2* old_value
```

얼마나 많이 loop을 돌 것인가?

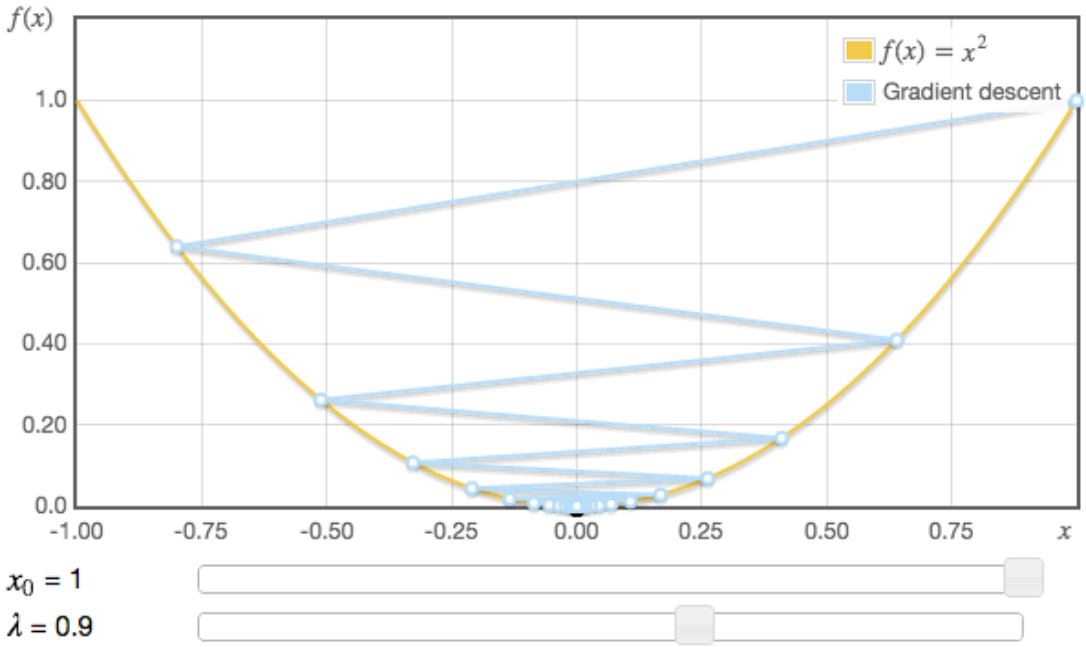
너무 작을 경우



$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

끝까지 못감
시간이 오래 걸림

너무 클 경우

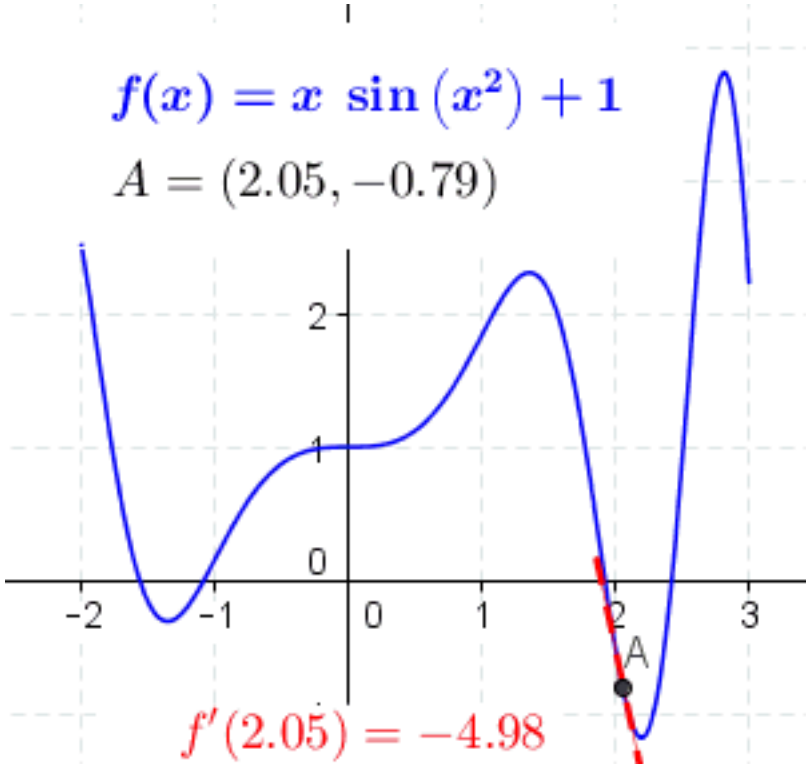


$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

데이터가 튀는 문제가 생김
수렴하지 못하는 경우생김

http://www.onmyphd.com/?p=gradient.descent

굴곡이 많은 함수의 경우는?



$$\frac{df(x)}{dx} = \sin(x^2) + 2x^2\cos(x^2)$$

$$x := x - \alpha \times \sin(x^2) + 2x^2\cos(x^2)$$

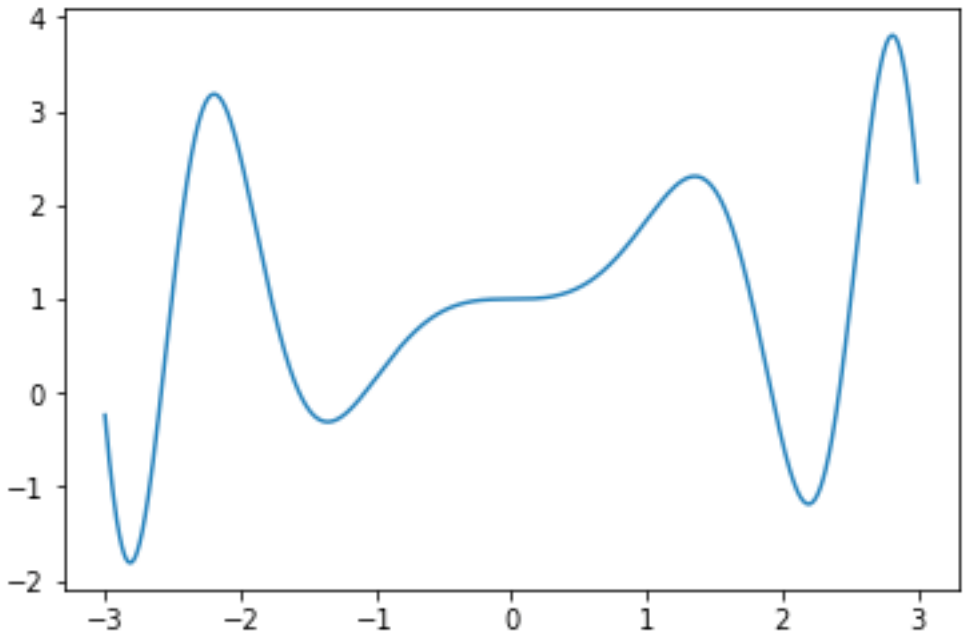
```
def sin_function(x):
    return x * np.sin(x ** 2) + 1

def derivitive_f(x):
    return np.sin(x**2) + 2 * (x **2) * np.cos(x ** 2)
```

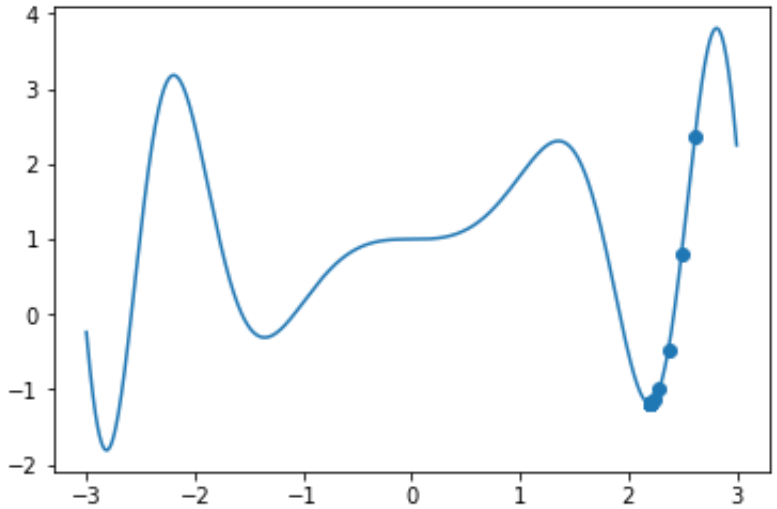
http://www.wolframalpha.com/input/?i=derivative+x+sin(x%5E2)+%2B+1

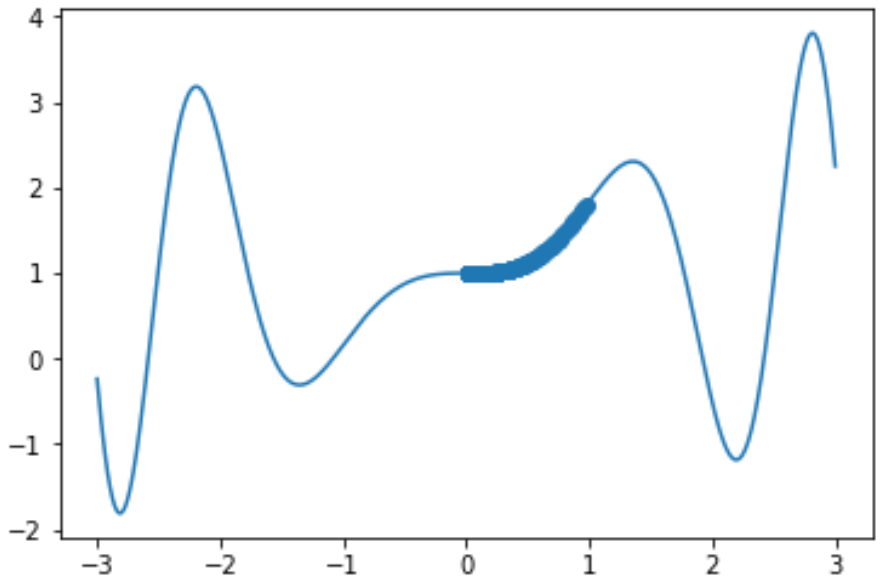
굴곡이 많은 함수의 경우는?

```
x= np.arange(-3,3,0.001)
f_x = sin_function(x)
```

```
plt.plot(x, f_x)
plt.show()
```







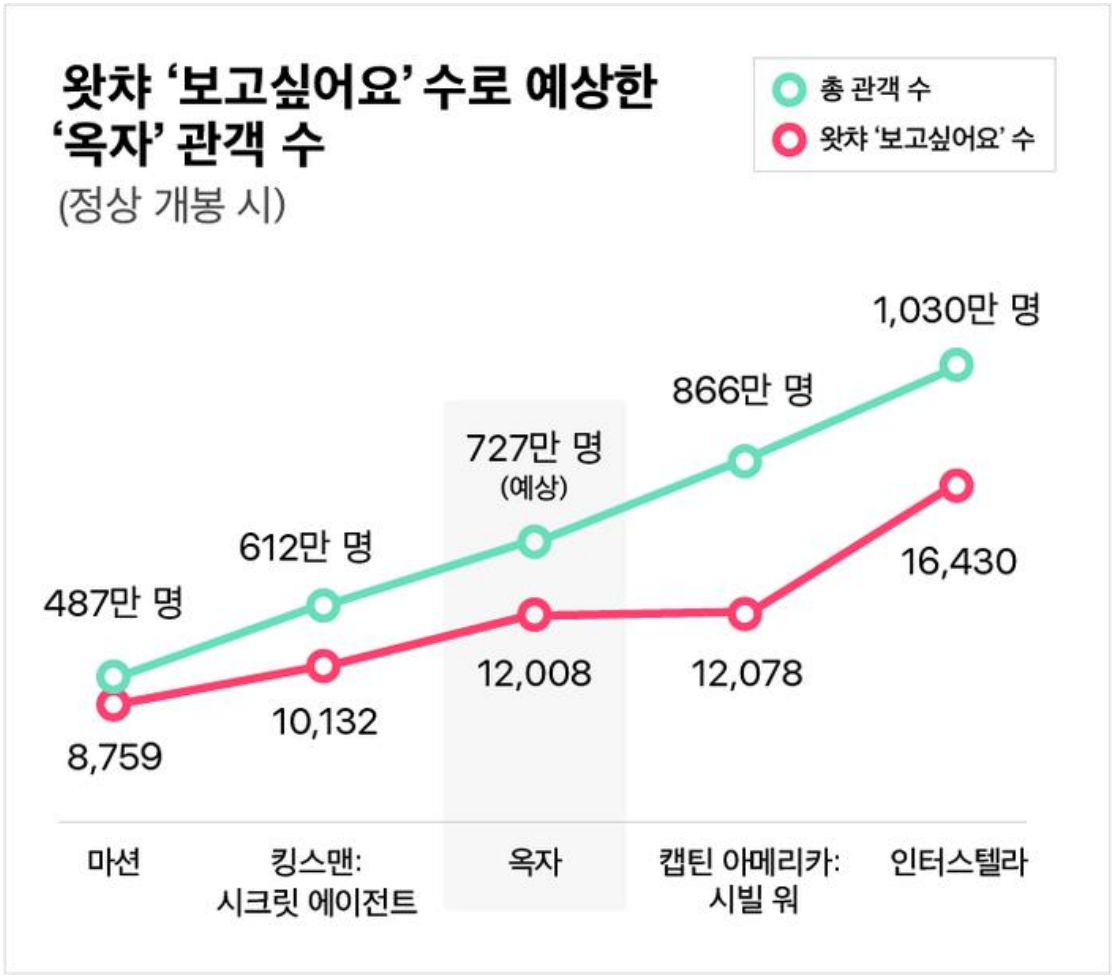
시작점에 따라
다른 최적값을 찾는다

Human knowledge belongs to the world.

Linear Regression with GD

Linear Regression

Director of TEAMLAB
Sungchul Choi



$$\mathbf{y} = \begin{bmatrix} 487 \\ 612 \\ 866 \\ 1030 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & 8,759 \\ 1 & 10,132 \\ 1 & 12,078 \\ 1 & 16,430 \end{bmatrix}$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

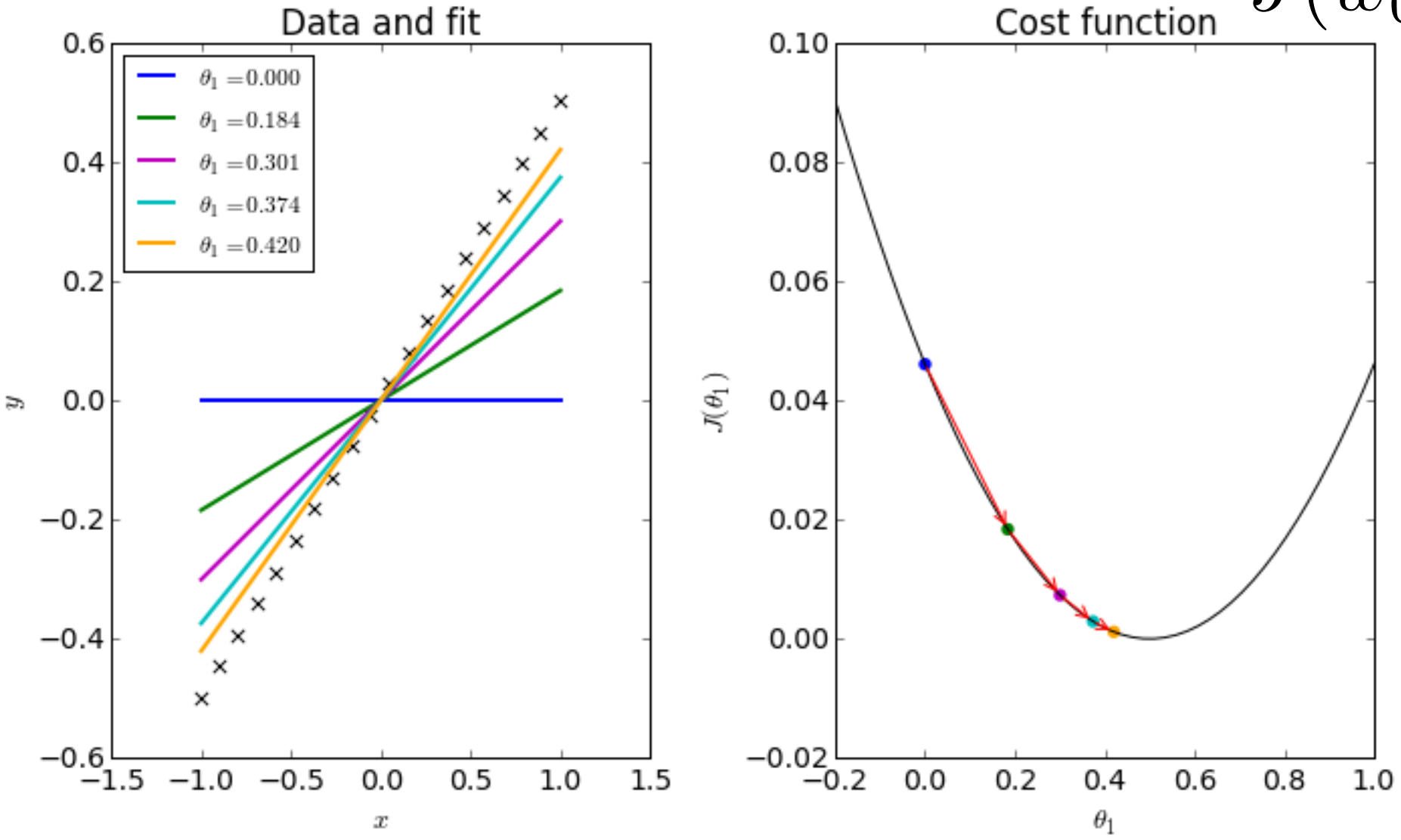
$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)})^2$$

Minimize $J(w_0, w_1)$

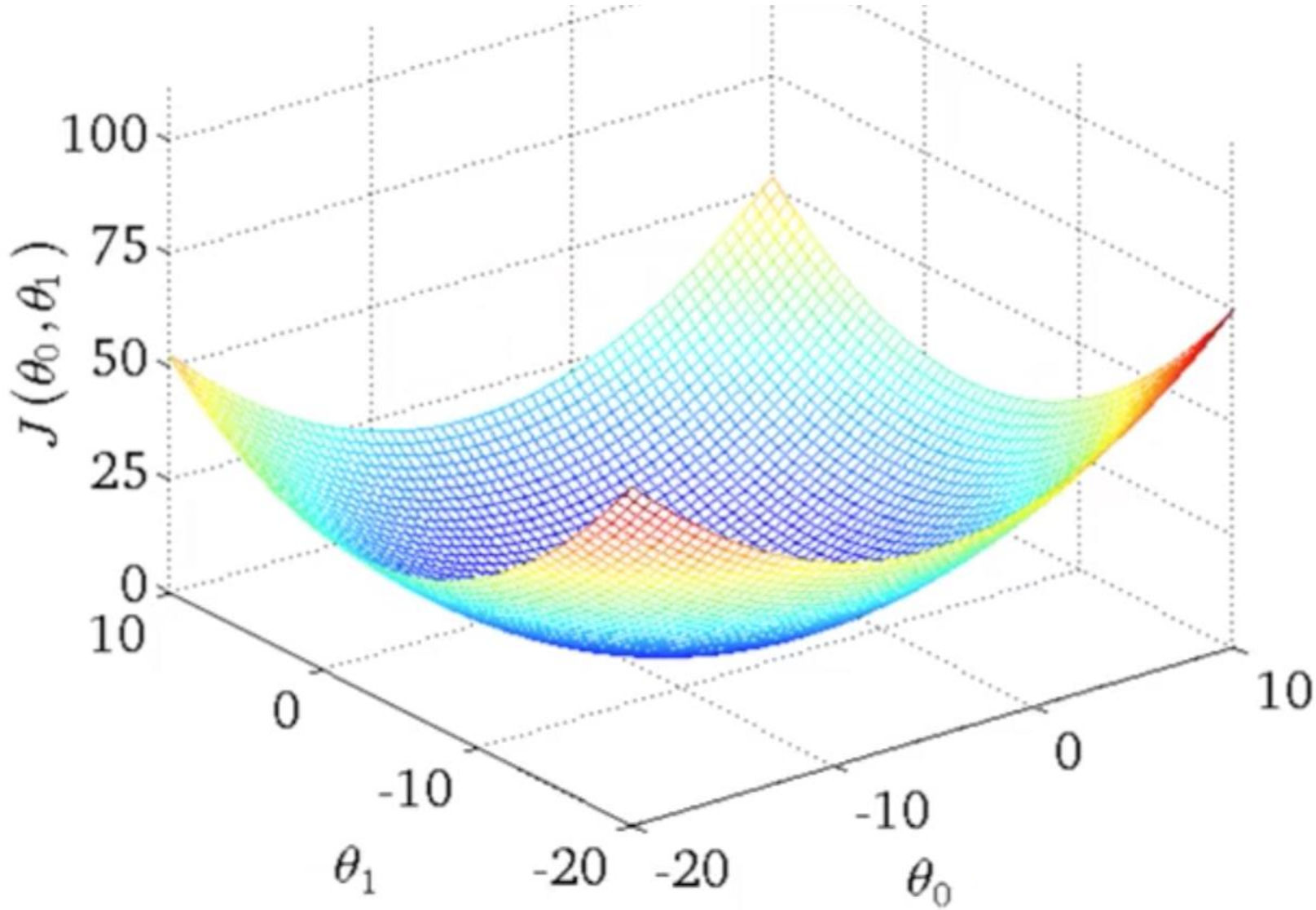
$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)})^2$$

Minimize $J(w_0, w_1)$

$J(w_0, w_1)$



https://scipython.com/blog/visualizing-the-gradient-descent-method/


100
75
50
25
0
$J(\theta_0, \theta_1)$
10
0
-10
-20
θ_1
-20
-10
0
10
θ_0

결국 parameter의 업데이트

Linear regression with GD

- 임의의 θ_0, θ_1값으로 초기화
- Cost function $J(\theta_0, \theta_1)$이 최소화 될 때까지 학습
- 더 이상 cost function이 줄어들지 않거나 학습 횟수를 초과할 때 종료

Linear regression with GD

$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

loop until convergence{

do $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$

}

$$\frac{\partial J}{\partial w_0} = \frac{1}{m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)})$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)}) x^{(i)}$$

simultaneously

Linear regression with GD

- Learning rate, Iteration 횟수 등 Parameter 지정
- Feature가 많으면 Normal equation에 비해 상대적으로 빠름
- 최적값에 수렴하지 않을 수도 있음

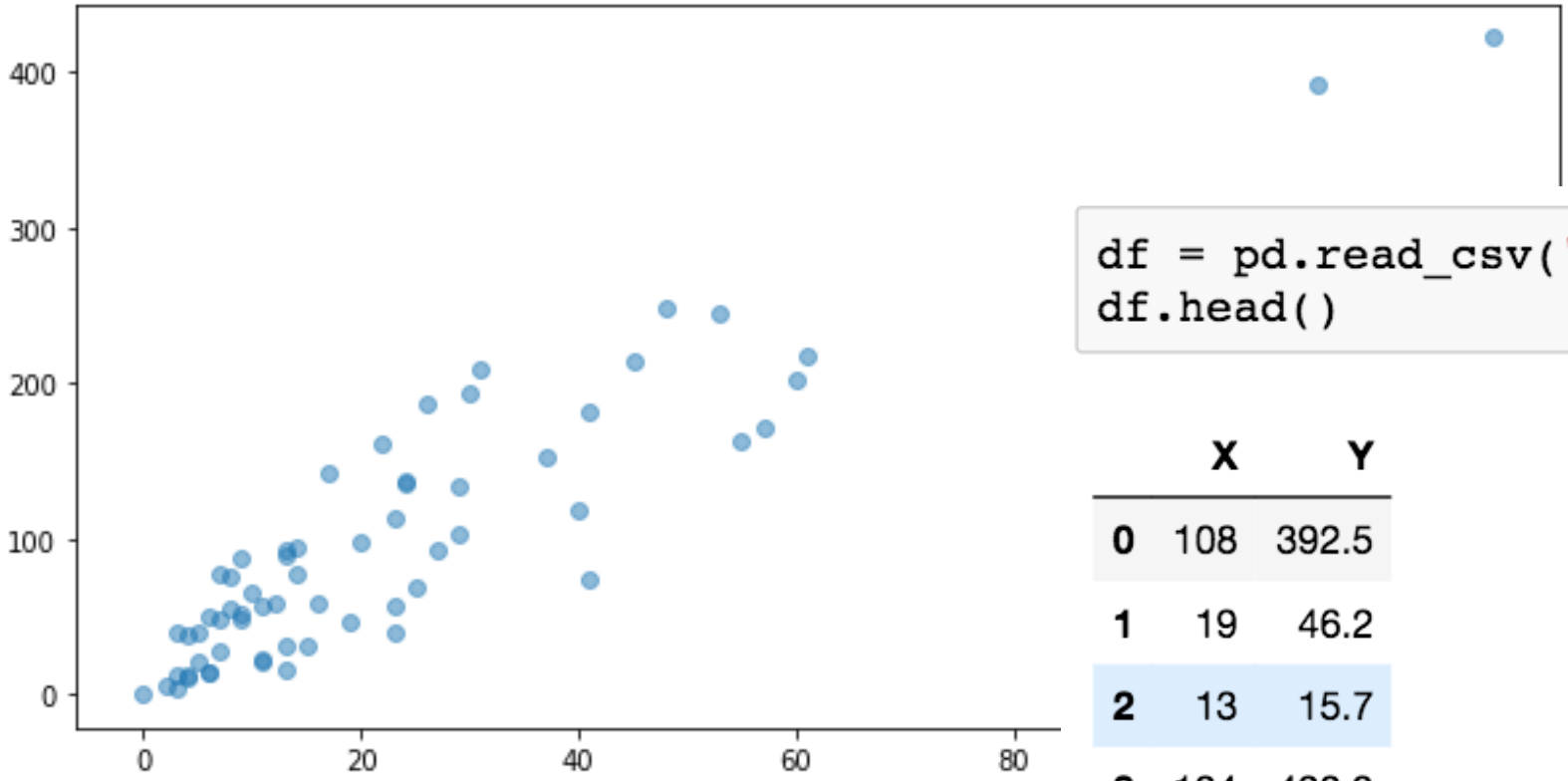
Human knowledge belongs to the world.

Implementation

Linear Regression

Director of TEAMLAB
Sungchul Choi

Gradient Descent로 Linear Regression 구해보기




```
df = pd.read_csv("data/slr06.csv")
df.head()
```

	X	Y
0	108	392.5
1	19	46.2
2	13	15.7
3	124	422.2
4	40	119.4

```
raw_X = df["X"].values.reshape(-1, 1)
y = df["Y"].values
```

```
plt.figure(figsize=(10,5))
plt.plot(raw_X,y, 'o', alpha=0.5)
```

```
raw_X[:5], y[:5]
```

```
(array([[108],
        [ 19],
        [ 13],
        [124],
        [ 40]]), array([ 392.5,   46.2,   15.7,  422.2,  119.4]))
```

```
np.ones((len(raw_X),1))[:5]
```

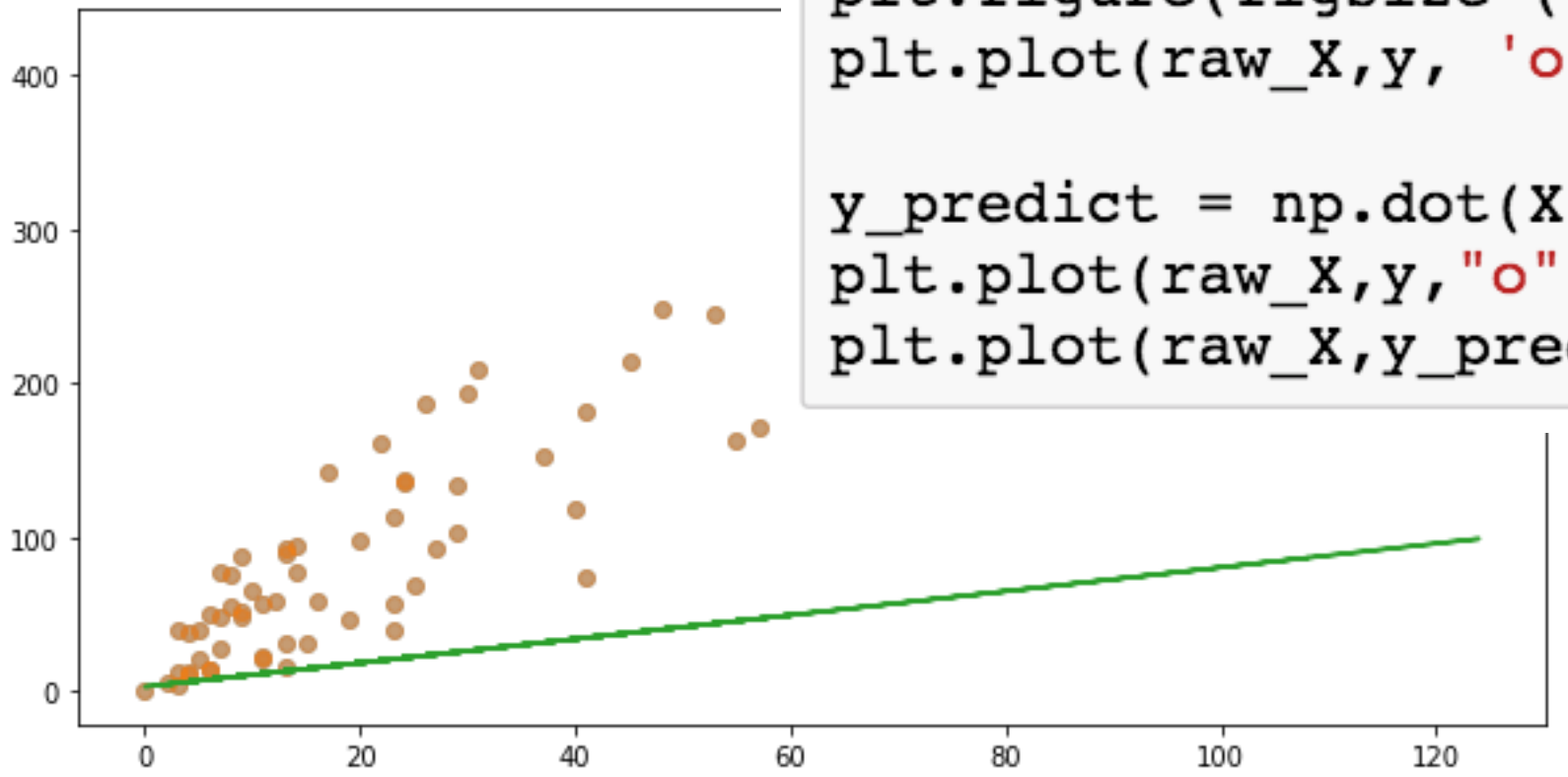
```
array([[ 1.],
       [ 1.],
       [ 1.],
       [ 1.],
       [ 1.]])
```

```
X = np.concatenate( (np.ones((len(raw_X),1)), raw_X ), axis=1)
X[:5]
```

```
array([[   1.,  108.],
       [   1.,   19.],
       [   1.,   13.],
       [   1.,  124.],
       [   1.,   40.]])
```

```
w = np.random.normal((2,1))
# w = np.array([5,3])
w
```

```
array([ 3.19571562,  0.77429904])
```




```
plt.figure(figsize=(10,5))
plt.plot(raw_X,y, 'o', alpha=0.5)

y_predict = np.dot(X, w)
plt.plot(raw_X,y,"o", alpha=0.5)
plt.plot(raw_X,y_predict)
```

$$\hat{y} = \mathbf{X}\mathbf{w}$$

```
def hypothesis_function(X, theta):
    return X.dot(theta)
```

$$f(x) = h_\theta(x)$$

```
hypothesis_function(X,w)[:5]
```

```
array([ 86.82001149,  17.9073973 ,  13.26160309,  99.20879607,  34.16767706])
```

```
def cost_function(h, y):
    return (1/(2*len(y))) * np.sum((h-y)**2)
```

```
h = hypothesis_function(X,w)
cost_function(h, y)
```

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

```
5479.1213778520041
```

```python
def gradient_descent(X, y, w, alpha, iterations):
    theta = w
    m = len(y)

    theta_list = [theta.tolist()]
    cost = cost_function(hypothesis_function(X, theta), y)
    cost_list = [cost]

    for i in range(iterations):
        t0 = theta[0] - (alpha / m) * np.sum(np.dot(X, theta) - y)
        t1 = theta[1] - (alpha / m) * np.sum((np.dot(X, theta) - y) * X[:,1])
        theta = np.array([t0, t1])

        if i % 10== 0:
            theta_list.append(theta.tolist())
            cost = cost_function(hypothesis_function(X, theta), y)
            cost_list.append(cost)


    return theta, theta_list, cost_list
```

$$f(x) = h_\theta(x) \quad J(w_0, w_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\frac{\partial J}{\partial w_0} = \frac{1}{m}\sum_{i=1}^{m}(w_1 x^{(i)} + w_0 - y^{(i)})$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{m}\sum_{i=1}^{m}(w_1 x^{(i)} + w_0 - y^{(i)})x^{(i)}$$

```
def gradient_descent(X, y, w, alpha, iterations):
    theta = w
    m = len(y)

    theta_list = [theta.tolist()]
    cost = cost_function(hypothesis_function(X, theta), y)
    cost_list = [cost]

    for i in range(iterations):
        t0 = theta[0] - (alpha / m) * np.sum(np.dot(X, theta) - y)
        t1 = theta[1] - (alpha / m) * np.sum((np.dot(X, theta) - y) * X[:,1])
        theta = np.array([t0, t1])
```

$$f(x) = h_\theta(x) \quad J(w_0, w_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\frac{\partial J}{\partial w_0} = \frac{1}{m}\sum_{i=1}^{m}(w_1 x^{(i)} + w_0 - y^{(i)})$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{m}\sum_{i=1}^{m}(w_1 x^{(i)} + w_0 - y^{(i)})x^{(i)}$$

loop until convergence{

do $\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1)$

}

```
        .tolist())
        pothesis_function(X, theta), y)

    return theta, theta_list, cost_list
```

```python
def gradient_descent(X, y, w, alpha, iterations):
    theta = w
    m = len(y)

    theta_list = [theta.tolist()]
    cost = cost_function(hypothesis_function(X, theta), y)
    cost_list = [cost]

    for i in range(iterations):
        t0 = theta[0] - (alpha / m) * np.sum(np.dot(X, theta) - y)
        t1 = theta[1] - (alpha / m) * np.sum((np.dot(X, theta) - y) * X[:,1])
        theta = np.array([t0, t1])

        if i % 10== 0:
            theta_list.append(theta.tolist())
            cost = cost_function(hypothesis_function(X, theta), y)
            cost_list.append(cost)


    return theta, theta_list, cost_list
```

$$f(x) = h_\theta(x) \quad J(w_0, w_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

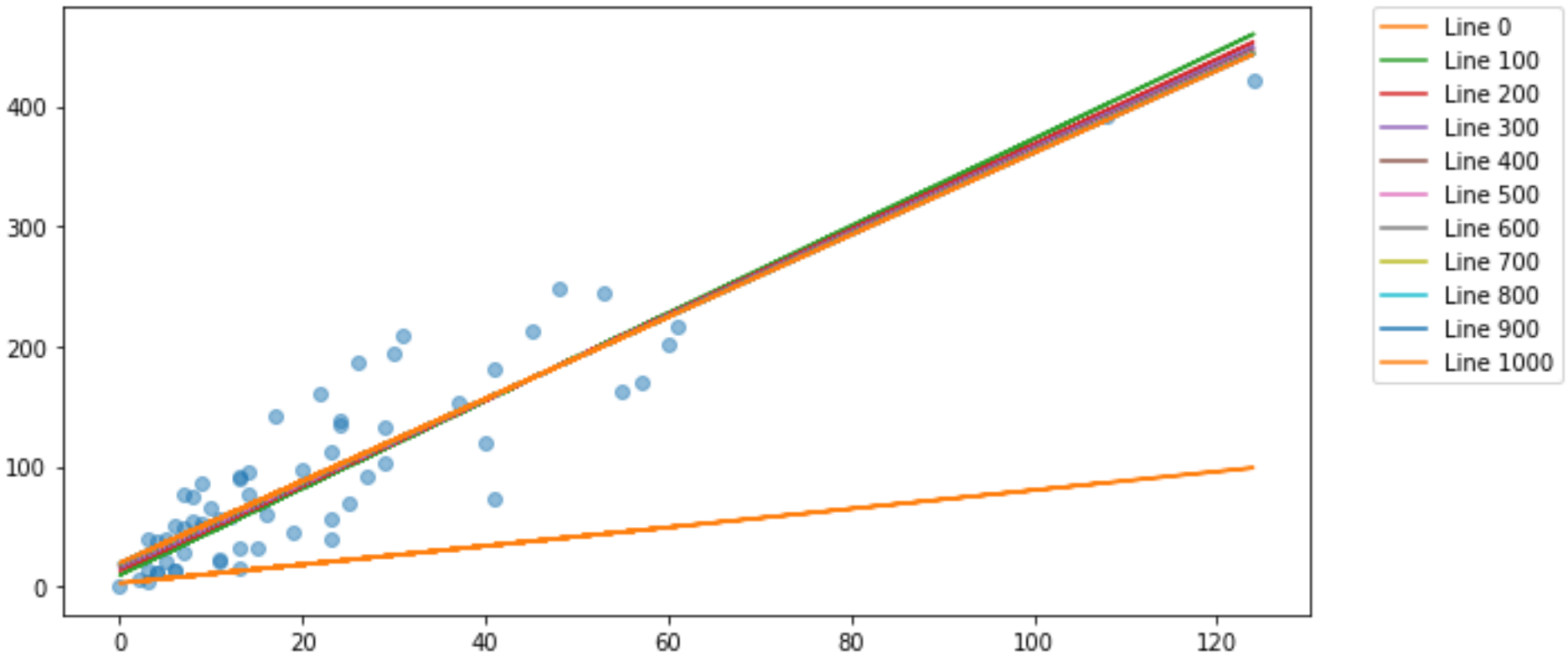
$$\frac{\partial J}{\partial w_0} = \frac{1}{m}\sum_{i=1}^{m}(w_1 x^{(i)} + w_0 - y^{(i)})$$

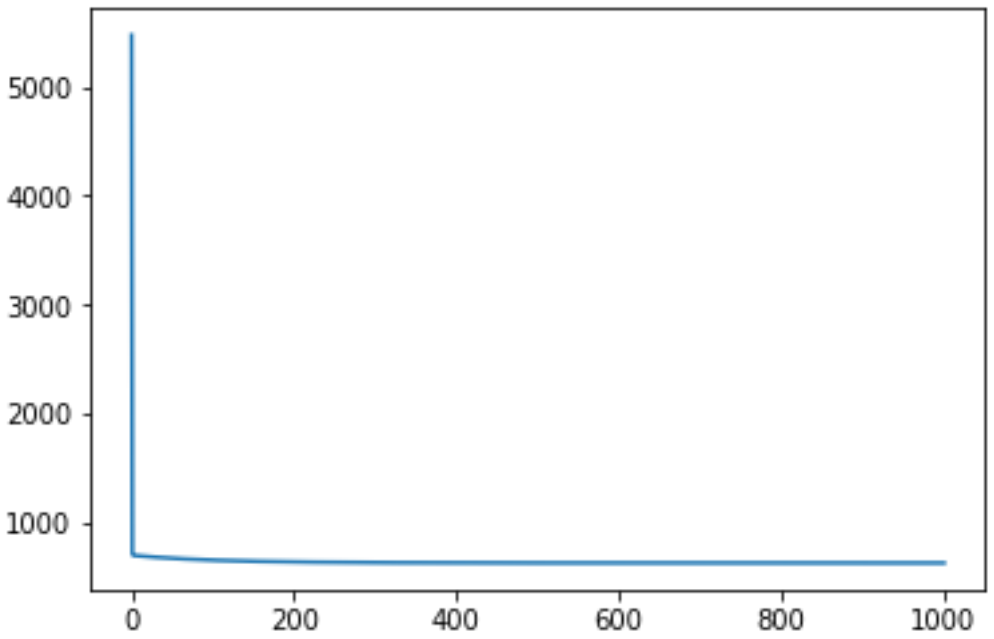
$$\frac{\partial J}{\partial w_1} = \frac{1}{m}\sum_{i=1}^{m}(w_1 x^{(i)} + w_0 - y^{(i)})x^{(i)}$$

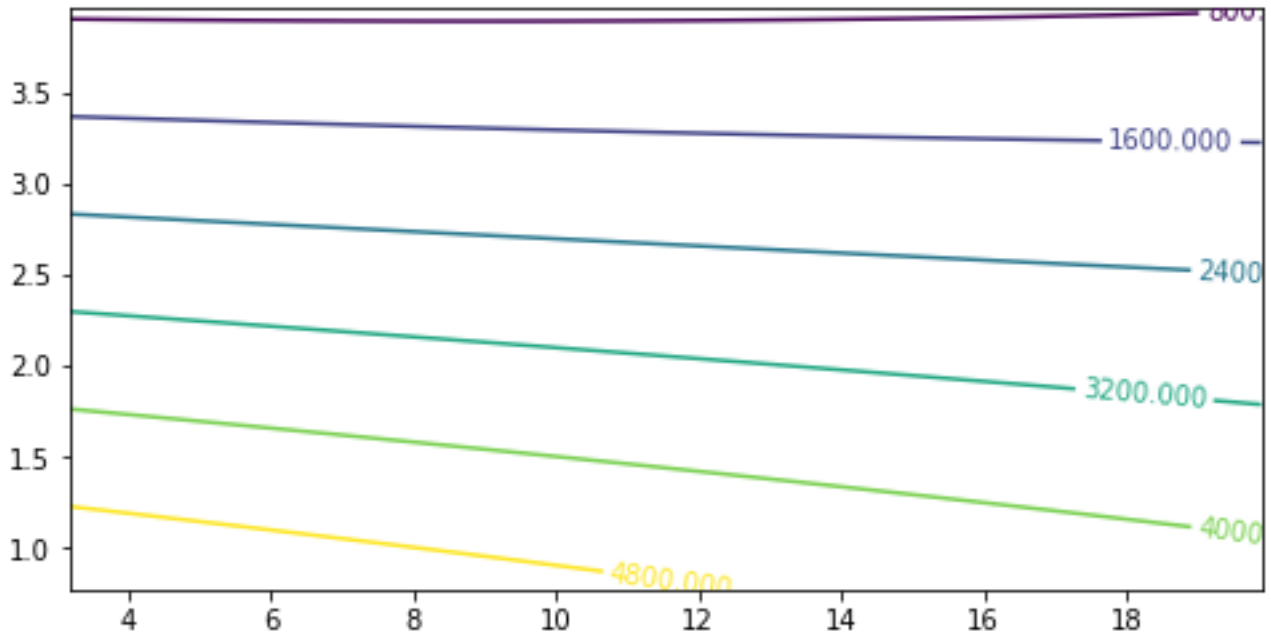
```
iterations = 10000
alpha = 0.001

theta, theta_list, cost_list = gradient_descent(X, y, w, alpha, iterations)
cost = cost_function(hypothesis_function(X, theta), y)

print("theta:", theta)
print('cost:', cost_function(hypothesis_function(X, theta), y))
```

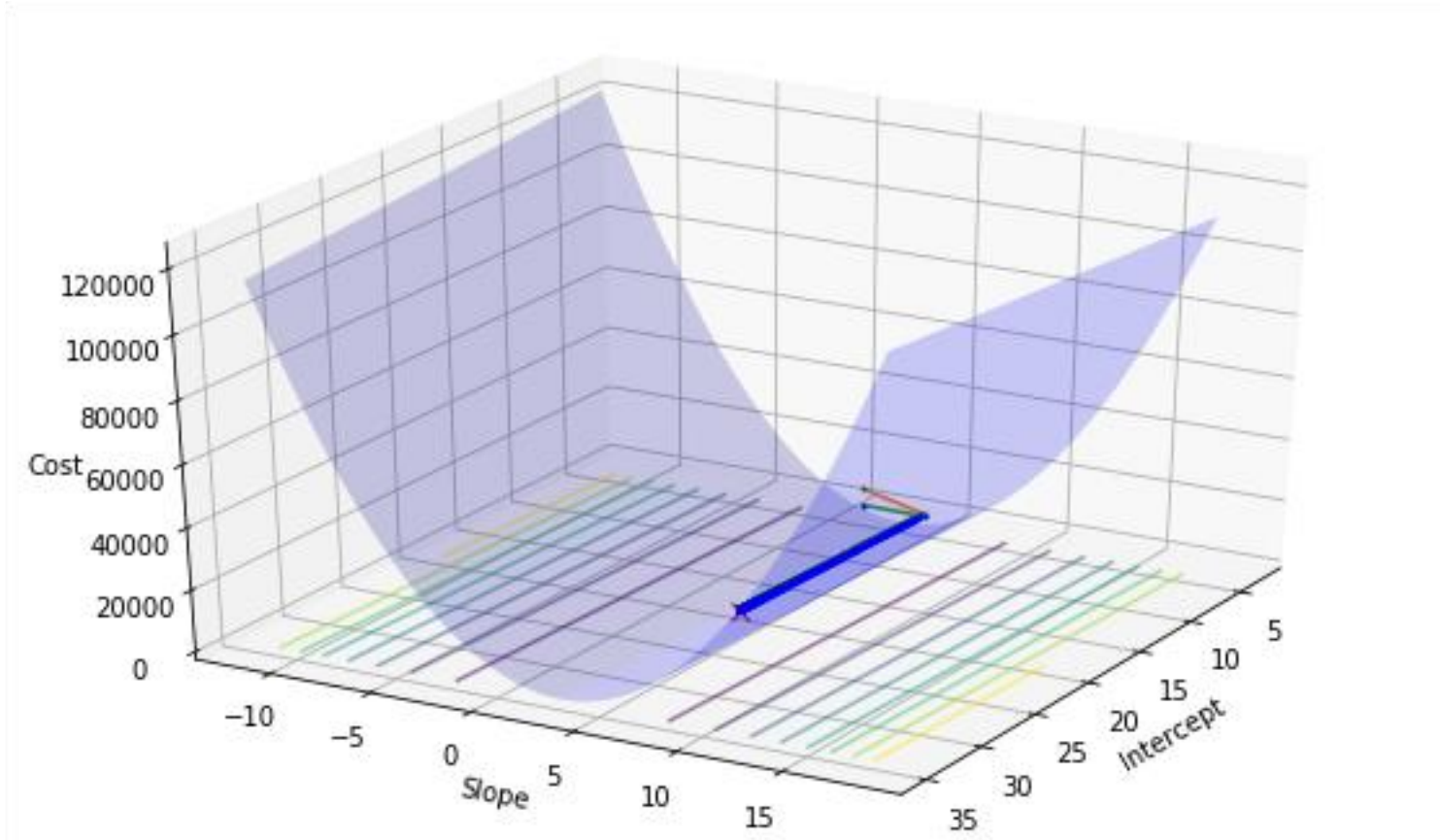
```
theta: [ 19.8878307   3.4161265]
cost: 625.373840751
```

```
plt.figure(figsize=(10,5))

y_predict_step= np.dot(X, theta_list.transpose())
plt.plot(raw_X,y,"o", alpha=0.5)
for i in range (0,len(cost_list),100):
    plt.plot(raw_X,y_predict_step[:,i], label='Line %d'%i)

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.show()
```




5000
4000
3000
2000
1000
0
200
400
600
800
1000


3.5
3.0
2.5
2.0
1.5
1.0
1600.000
2400
3200.000
4000
4800.000
4
6
8
10
12
14
16
18


120000
100000
80000
Cost
60000
40000
20000
0
−10
−5
0
5
10
15
Slope
5
10
15
20
25
30
35
Intercept

Human knowledge belongs to the world.

Multivariate Linear Regression

Linear Regression

Director of TEAMLAB
Sungchul Choi

한 개이상의 Feature로 구성된 데이터를 분석할 때

식은 많아지지만 여전히 Cost 함수의 최적화

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)})^2$$

$$\frac{\partial J}{\partial w_0} = \frac{1}{m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)})$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)}) x^{(i)}$$

$$J(w_0, w_1, \ldots, w_n) = \frac{1}{2m} \sum_{i=1}^{m} (w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_n x_n^{(i)} + w_0 - y^{(i)})^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2$$

$$\frac{\partial J}{\partial w_0} = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)}) \cdot x_1$$

$$\frac{\partial J}{\partial w_n} = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)}) \cdot x_2$$

$$\frac{\partial J}{\partial w_n} = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)}) \cdot x_n$$

Simultaneously update

```
for _ in range(iterations):
    predictions = x.dot(theta)

    for i in range(theta.size):
        partial_marginal = x[:, i]
        errors_xi = (predictions - y) * partial_marginal
        theta[i] = theta[i] - alpha * (1.0 / m) * errors_xi.sum()

    theta_history.append(theta)
    cost_history.append(compute_cost(x, y, theta))
```

$$\frac{\partial J}{\partial w_n} = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)}) \cdot x_n$$

```
for _ in range(iterations):
    predictions = x.dot(theta)

    for i in range(theta.size):
        partial_marginal = x[:, i]
        errors_xi = (predictions - y) * partial_marginal
        theta[i] = theta[i] - alpha * (1.0 / m) * errors_xi.sum()

    theta_history.append(theta)
    cost_history.append(compute_cost(x, y, theta))
```

$$\frac{\partial J}{\partial w_n} = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)}) \cdot x_n$$

$$w_n =: w_n - \alpha \frac{\partial J}{\partial w_0}$$

Human knowledge belongs to the world.

Linear Regression w/sklearn

Linear Regression

Director of TEAMLAB
Sungchul Choi

Boston House Price Dataset

- 머신 러닝 등 데이터 분석을 처음 배울 때,
  가장 대표적으로 사용하는 Example Dataset

- 1978년에 발표된 데이터로, 미국 인구통계 조사 결과
  미국 보스턴 지역의 주택 가격에 영향 요소들을 정리함

http://lib.stat.cmu.edu/datasets/boston

Boston House Price Dataset

X 변수
13개

Y 변수

[01] CRIM	자치시(town) 별 1인당 범죄율
[02] ZN	25,000 평방피트를 초과하는 거주지역의 비율
[03] INDUS	비소매상업지역이 점유하고 있는 토지의 비율
[04] CHAS	찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)
[05] NOX	10ppm 당 농축 일산화질소
[06] RM	주택 1가구당 평균 방의 개수
[07] AGE	1940년 이전에 건축된 소유주택의 비율
[08] DIS	5개의 보스턴 직업센터까지의 접근성 지수
[09] RAD	방사형 도로까지의 접근성 지수
[10] TAX	10,000 달러 당 재산세율
[11] PTRATIO	자치시(town)별 학생/교사 비율
[12] B	1000(Bk-0.63)^2, 여기서 Bk는 자치시별 흑인의 비율을 말함.
[13] LSTAT	모집단의 하위계층의 비율(%)
[14] MEDV	본인 소유의 주택가격(중앙값) (단위: $1,000)

http://www.dator.co.kr/ctg258/textyle/1721307

http://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

데이터 로딩

```
from sklearn.datasets import load_boston
import matplotlib.pyplot as plt
import numpy as np
```

```
boston = load_boston()

x_data = boston.data
y_data = boston.target.reshape(boston.target.size,1)

x_data[:3]
```

데이터 로딩

```
array([[  6.32000000e-03,   1.80000000e+01,   2.31000000e+00,
          0.00000000e+00,   5.38000000e-01,   6.57500000e+00,
          6.52000000e+01,   4.09000000e+00,   1.00000000e+00,
          2.96000000e+02,   1.53000000e+01,   3.96900000e+02,
          4.98000000e+00],
       [  2.73100000e-02,   0.00000000e+00,   7.07000000e+00,
          0.00000000e+00,   4.69000000e-01,   6.42100000e+00,
          7.89000000e+01,   4.96710000e+00,   2.00000000e+00,
          2.42000000e+02,   1.78000000e+01,   3.96900000e+02,
          9.14000000e+00],
       [  2.72900000e-02,   0.00000000e+00,   7.07000000e+00,
          0.00000000e+00,   4.69000000e-01,   7.18500000e+00,
          6.11000000e+01,   4.96710000e+00,   2.00000000e+00,
          2.42000000e+02,   1.78000000e+01,   3.92830000e+02,
          4.03000000e+00]])
```

데이터 스케일링

```
from sklearn import preprocessing

minmax_scale = preprocessing.MinMaxScaler().fit(x_data)
x_scaled_data = minmax_scale.transform(x_data)

x_scaled_data[:3]
```

데이터 스케일링

```
array([[  0.00000000e+00,   1.80000000e-01,   6.78152493e-02,
          0.00000000e+00,   3.14814815e-01,   5.77505269e-01,
          6.41606591e-01,   2.69203139e-01,   0.00000000e+00,
          2.08015267e-01,   2.87234043e-01,   1.00000000e+00,
          8.96799117e-02],
       [  2.35922539e-04,   0.00000000e+00,   2.42302053e-01,
          0.00000000e+00,   1.72839506e-01,   5.47997701e-01,
          7.82698249e-01,   3.48961980e-01,   4.34782609e-02,
          1.04961832e-01,   5.53191489e-01,   1.00000000e+00,
          2.04470199e-01],
       [  2.35697744e-04,   0.00000000e+00,   2.42302053e-01,
          0.00000000e+00,   1.72839506e-01,   6.94385898e-01,
          5.99382080e-01,   3.48961980e-01,   4.34782609e-02,
          1.04961832e-01,   5.53191489e-01,   9.89737254e-01,
          6.34657837e-02]])
```

Train-Test Split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(x_scaled_data, y_data, test_size=0.33)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((339, 13), (167, 13), (339, 1), (167, 1))
```

Linear regression fitting

```
from sklearn import  linear_model

regr = linear_model.LinearRegression(fit_intercept=True, normalize=False, copy_X=True, n_jobs=1)
regr.fit(x_scaled_data, y_data)

# # The coefficients
print('Coefficients: ', regr.coef_)
print('intercept: ', regr.intercept_)
```

```
Coefficients:  [[ -9.53495156   4.63952195   0.56906733   2.6885614   -8.64873871
   19.85700309   0.07292809 -16.22877191   7.03006588  -6.46057746
   -8.96255741   3.7248827  -19.04291078]]
intercept:  [ 26.61291386]
```

$$y = w_1x_1 + w_2x_2 + \beta_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7 + \cdots w_{13}x_{13} + w_0 \cdot 1$$

수식 결과비교

```
regr.predict(x_data[0].reshape(1,-1))
```

```
array([[-483.19114376]])
```

```
x_data[0].dot(regr.coef_.T) + regr.intercept_
```

```
array([-483.19114376])
```

$$\sum_{i=0}^{13} w_i x_i = \boldsymbol{w}^T \cdot \boldsymbol{x}$$

Metric 측정

```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
```

```
y_true = y_test
y_hat = regr.predict(X_test)

r2_score(y_true, y_hat), mean_absolute_error(y_true, y_hat), mean_squared_error(y_true, y_hat)
```

```
(0.683540432559726 3, 3.5517583251413343, 27.19728458900094)
```

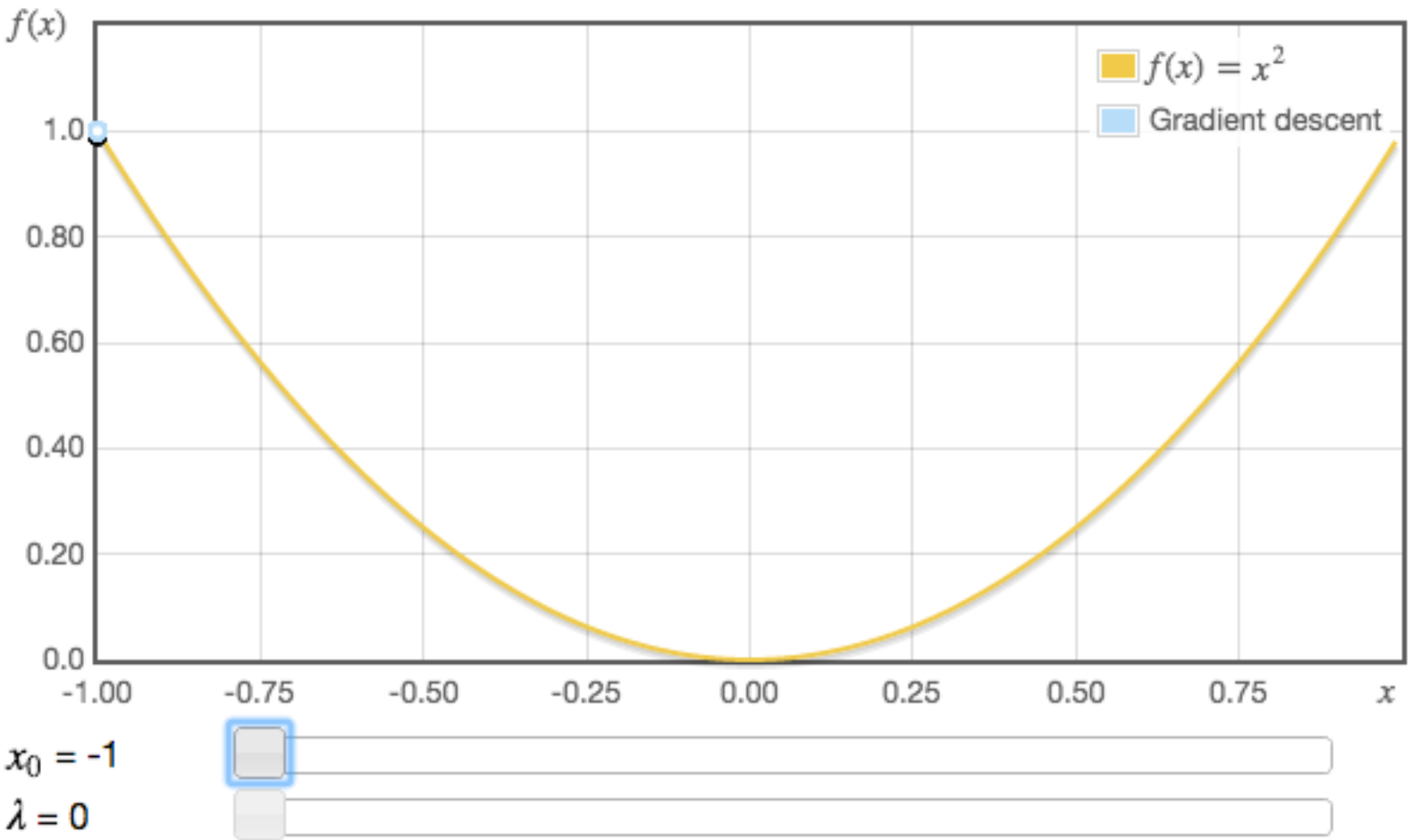
Human knowledge belongs to the world.

Stochastic Gradient Descent

Linear Regression

Director of TEAMLAB
Sungchul Choi

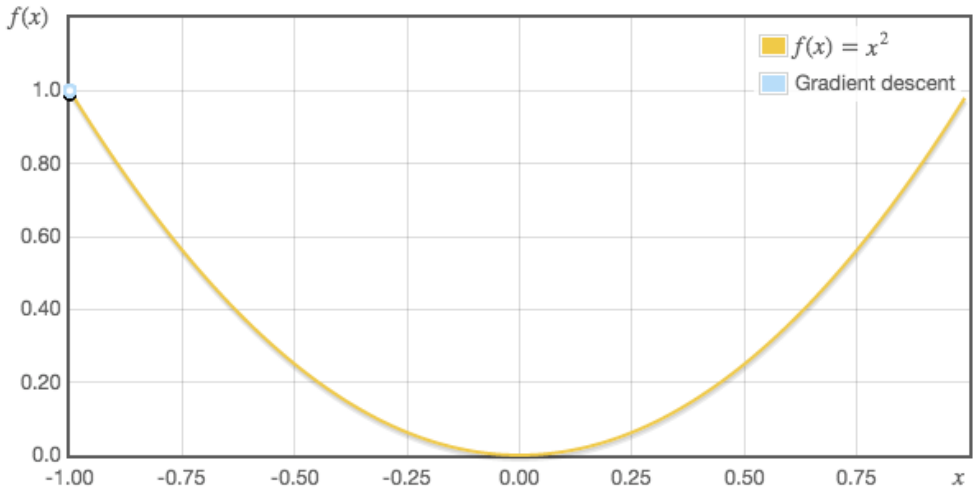
Gradient descent

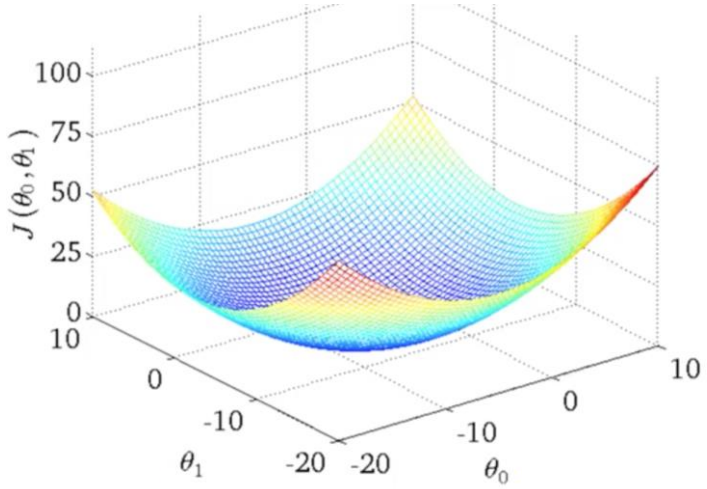


Gradient descent

http://news.jtbc.joins.com/article/article.aspx?news_id=NB10867287

Gradient descent





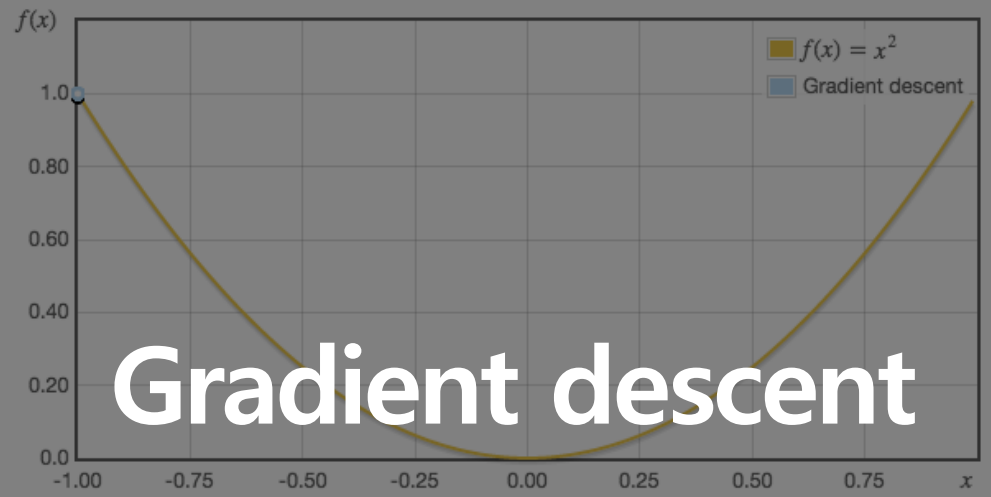
$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$\frac{\partial J}{\partial w_0} = \frac{1}{m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)})$$

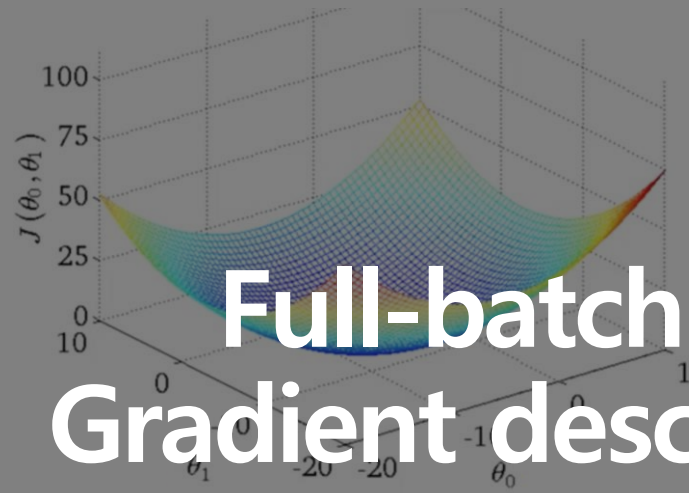
$$\frac{\partial J}{\partial w_1} = \frac{1}{m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)}) x^{(i)}$$

Gradient descent



Gradient descent

$$x_{new} = x_{old} - \alpha \times (2x_{old})$$



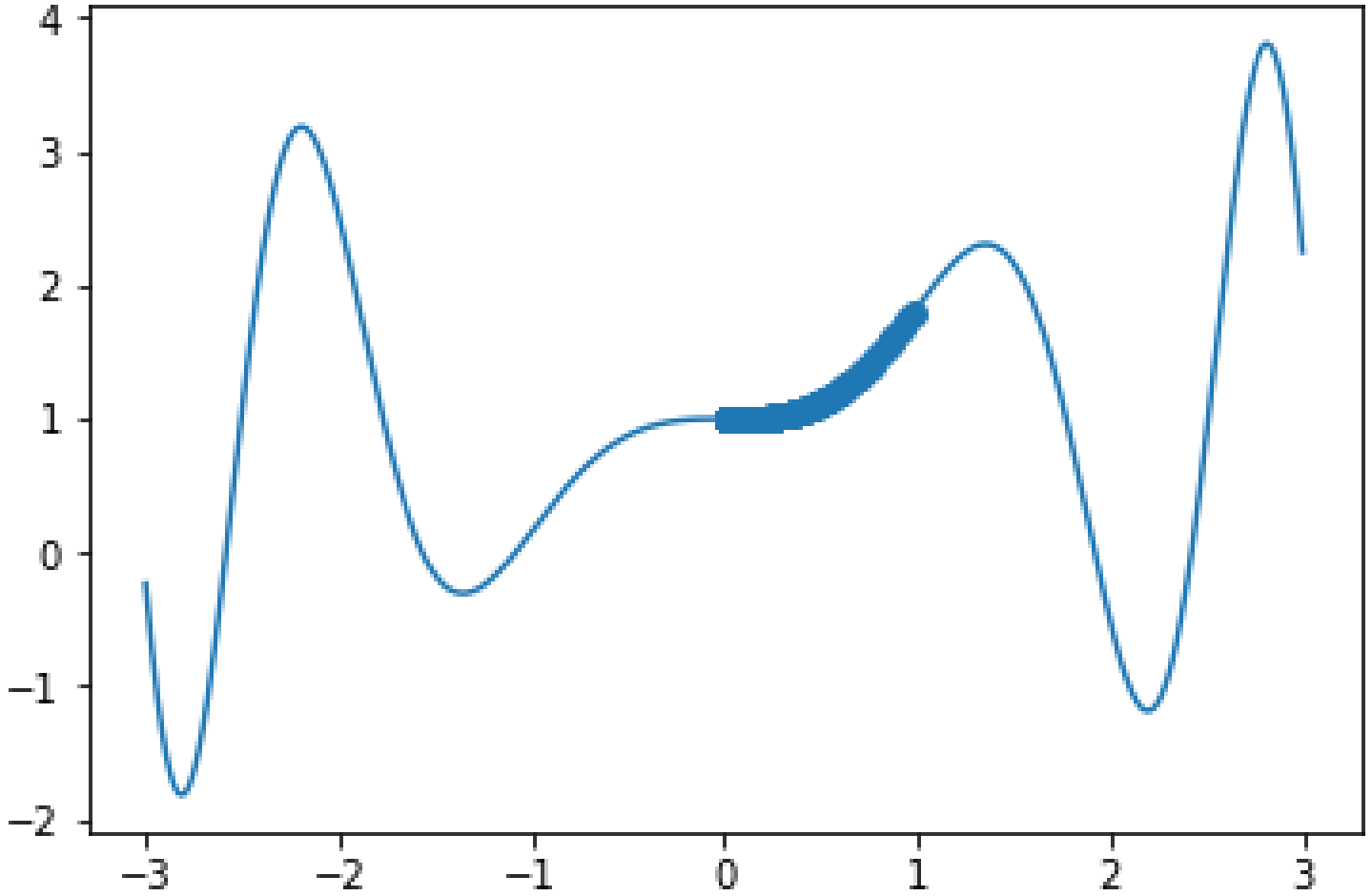
Full-batch Gradient descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$\frac{\partial J}{\partial w_0} = \frac{1}{m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)})$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)}) x^{(i)}$$

Gradient descent



Full-batch gradient descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$\frac{\partial J}{\partial w_n} = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)}) \cdot x_n$$

Full-batch gradient descent

- GD는 1개의 데이터를 기준으로 미분
- 그러나 일반적으로 GD = (full) batch GD라고 가정
- 모든 데이터 셋으로 학습 $\frac{\partial J}{\partial w_n} = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)}) \cdot x_n$

Full-batch gradient descent

- 업데이트 감소 → 계산상 효율적(속도) 가능
- 안정적인 Cost 함수 수렴

- 지역 최적화 가능
- 메모리 문제 (ex – 30억개의 데이터를 한번에?)
- 대규모 dataset → 모델/파라메터 업데이트가 느려짐

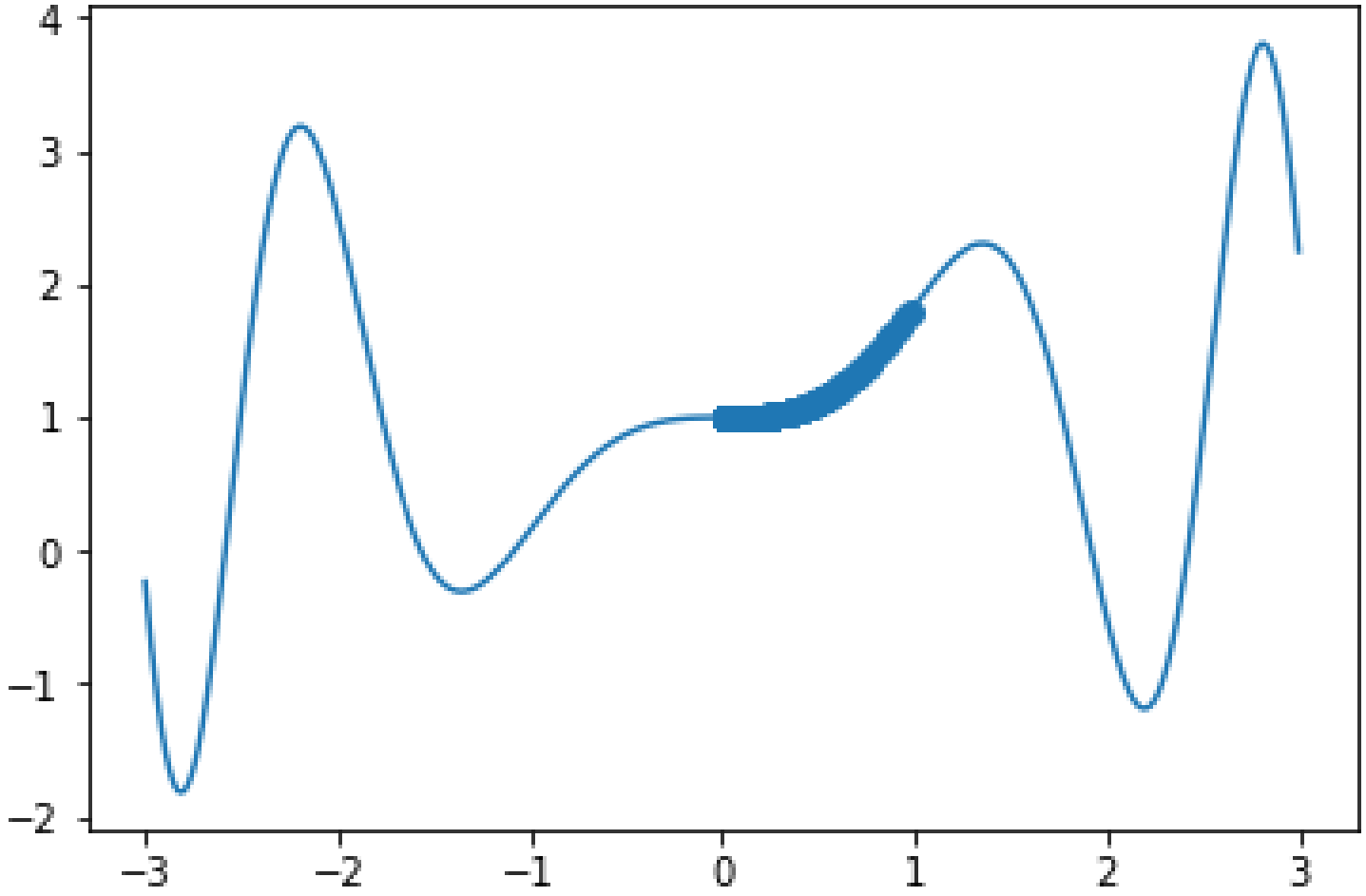
Full-batch gradient descent

- 업데이트 감소 → 계산상 효율적(속도) 가능
- 안정적인 Cost 함수 수렴

- 지역 최적화 가능
- 메모리 문제 (ex – 30억개의 데이터를 한번에?)
- 대규모 dataset → 모델/파라메터 업데이트가 느려짐

Stochastic gradient descent

Stochastic gradient descent



Stochastic gradient descent

- 원래 의미는 dataset에서 random하게 training sample을 뽑은 후 학습할 때 사용함
- Data를 넣기 전에 Shuffle

1: **procedure** SGD
2: `shuffle`(X) ▷ Randomly shuffle data
3: **for** i `in number of` X **do**
4: $\theta_j := \theta_j - \alpha(\hat{y}^{(i)} - y^{(i)})x_j^{(i)}$ ▷ Only one example
5: **end for**
6: **end procedure**

Stochastic gradient descent

- 빈번한 업데이트 모델 성능 및 개선 속도 확인 가능
- 일부 문제에 대해 더 빨리 수렴
- 지역 최적화 회피
- 대용량 데이터시 시간이 오래걸림
- 더 이상 cost가 줄어들지 않는 시점의 발견이 어려움

Mini-batch (stochastic) gradient descent

Mini-batch SGD

- 한번의 일정량의 데이터를 램덤하게 뽑아서 학습
- SGD와 Batch GD를 혼합한 기법
- 가장 일반적으로 많이 쓰이는 기법

Epoch & Batch-size

- 전체 데이터가 Training 데이터에 들어갈 때 카운팅
- Full-batch를 n번 실행하면 n epoch
- Batch-size 한번에 학습되는 데이터의 개수
- 총 5,120개의 Training data에 512 batch-size면 몇 번 학습을 해야 1 epoch이 되는가?

Mini-batch SGD

procedure MINI-BATCH SGD

`shuffle`(X) ▷ Randomly shuffle data

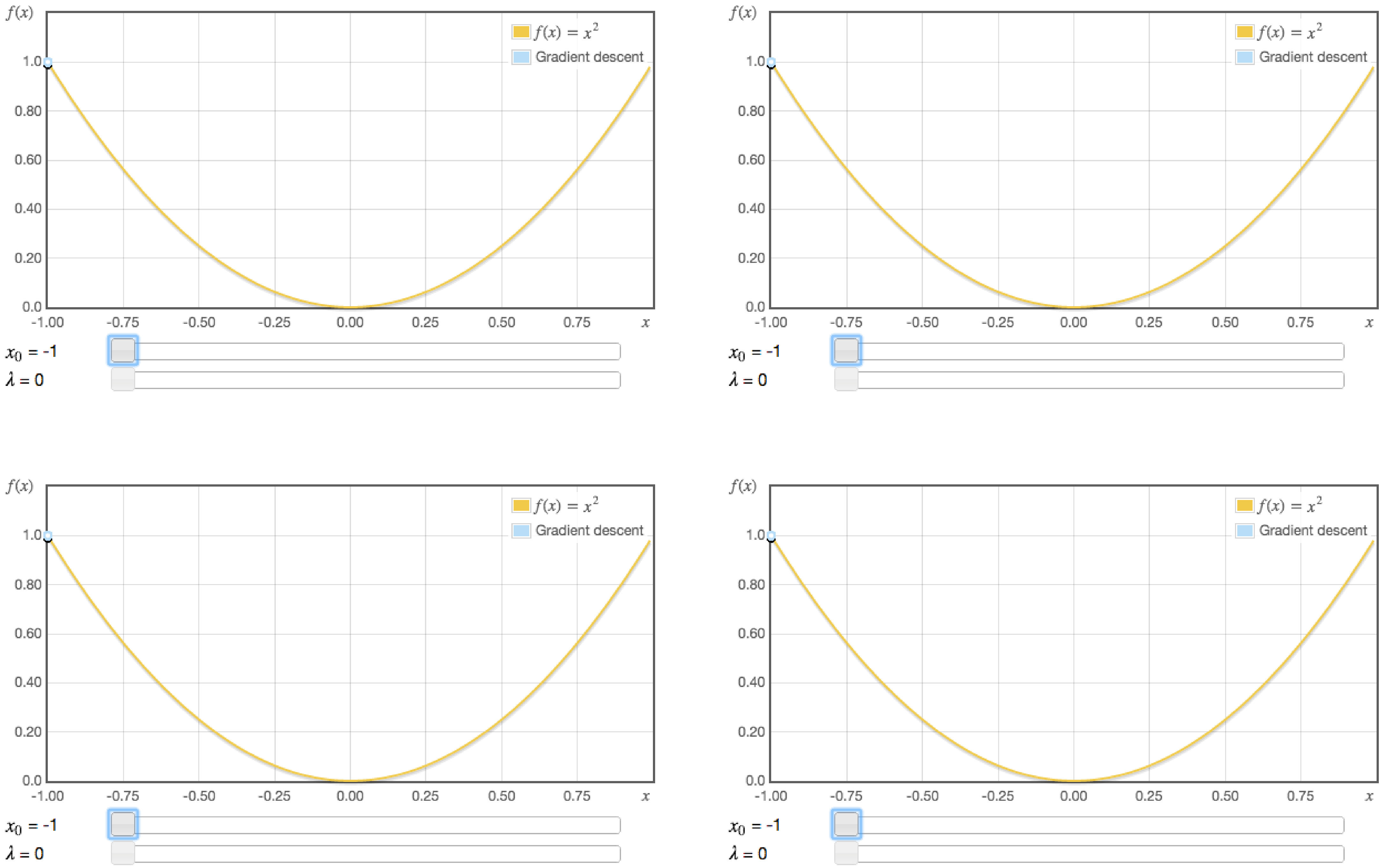
`BS` $\leftarrow$ `BATCH SIZE`

`NB` $\leftarrow$ `Number of Batches`

`NB` $\leftarrow$ `len`(X)`//BS`

for i `in NB` **do**

$\theta_j := \theta_j - \alpha \sum_{k=i\times BS}^{(i+1)*BS} (\hat{y}^{(k)} - y^{(k)}) x_j^{(k)}$ ▷ Batch-sized examples


f(x)
f(x) = x^2
Gradient descent
1.0
0.80
0.60
0.40
0.20
0.0
-1.00
-0.75
-0.50
-0.25
0.00
0.25
0.50
0.75
x
x0 = -1
λ = 0

Human knowledge belongs to the world.

SGD implementation issues

Linear Regression

Director of TEAMLAB
Sungchul Choi

SGD를 구현할 때 생각해봐야 할 일 들

Mini-Batch SGD

```
for epoch in range(epoches):
    X_copy = np.copy(X)
    if is_SGD:
        np.random.shuffle(X_copy)
    batch = len(X_copy) // BATCH_SIZE
    for batch_count in range(batch):
        X_batch = np.copy(
                X_copy[batch_count*BATCH_SIZE : (batch_count+1)*BATCH_SIZE])
        # Do weight Update
    print("Number of epoch : %d" % epoch)
```

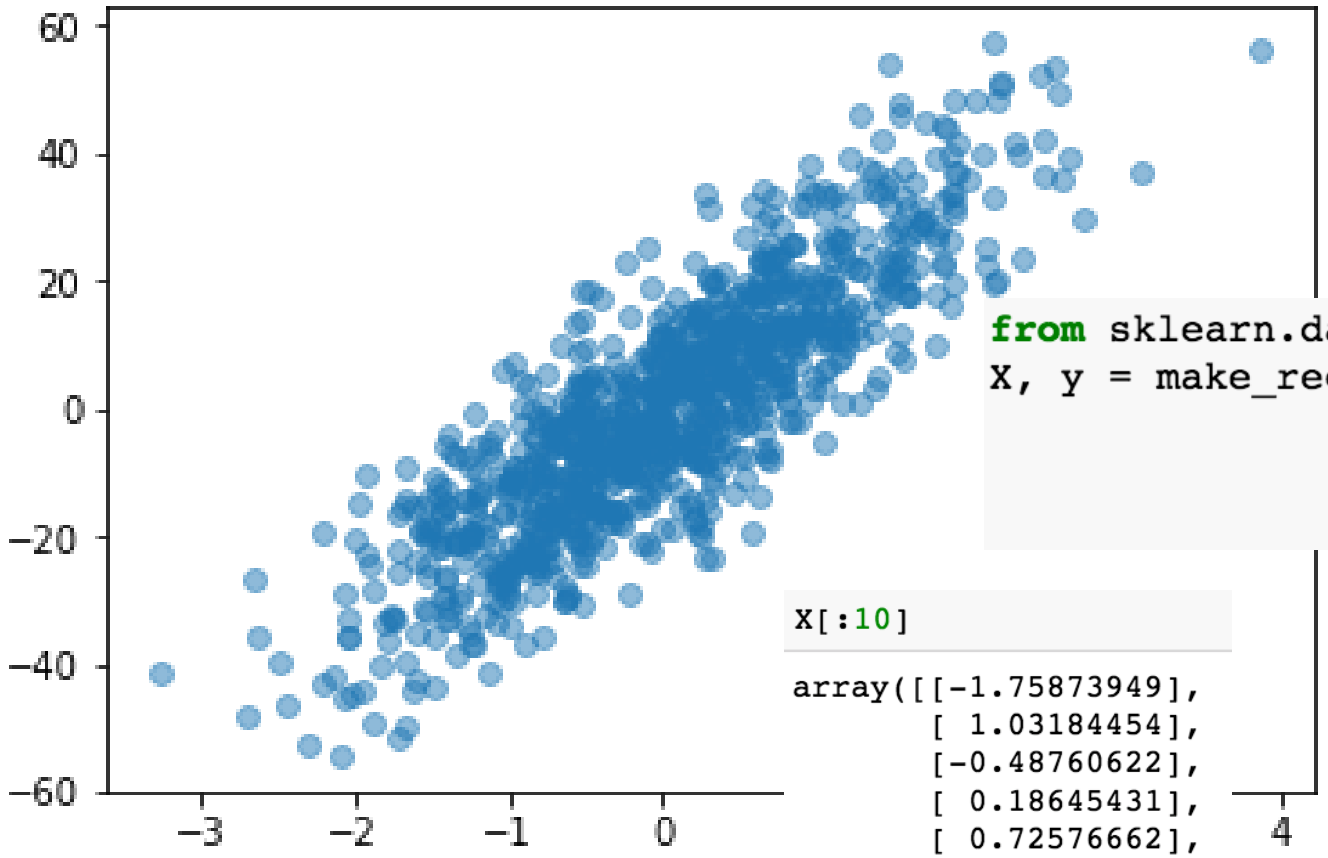
전체 Epoch이 iteration 되는 횟수

SGD 여부 → SGD일 경우 shuffle

한번에 처리하는 BATCH_SIZE

BATCH_SIZE 크기 만큼 X_batch 생성

Convergence process




```
from sklearn.datasets.samples_generator import make_regression
X, y = make_regression(n_samples = 1000,
                       n_features=1,
                       noise=10,
                       random_state=42)
```

```
X[:10]
```

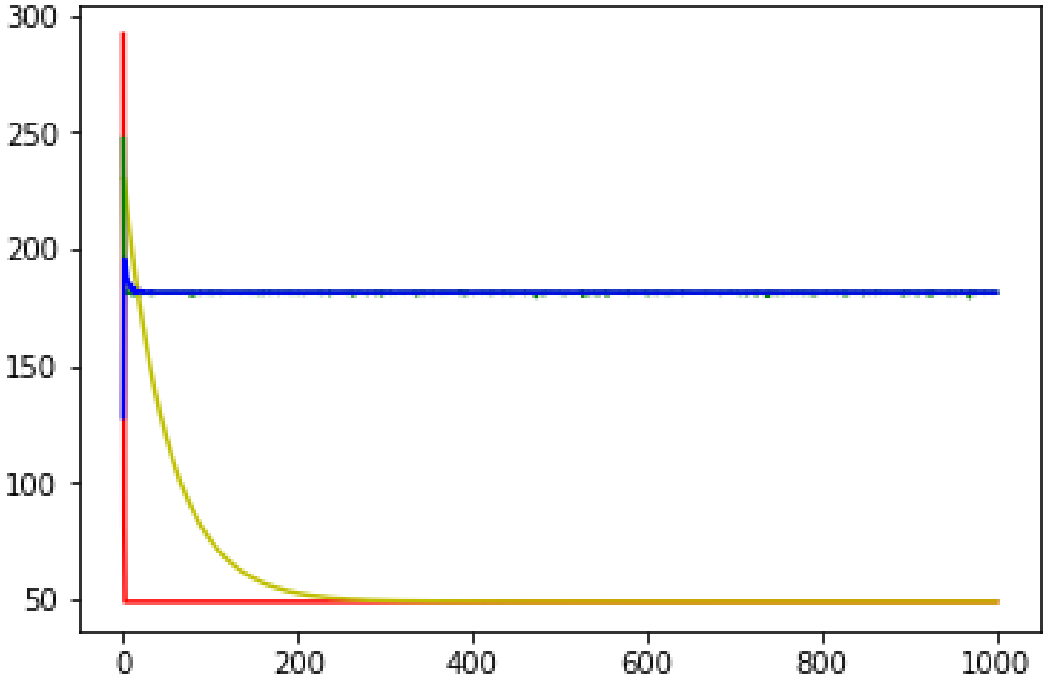
```
array([[-1.75873949],
       [ 1.03184454],
       [-0.48760622],
       [ 0.18645431],
       [ 0.72576662],
       [ 0.97255445],
       [ 0.64537595],
       [ 0.68189149],
       [-1.43014138],
       [ 1.06667469]])
```

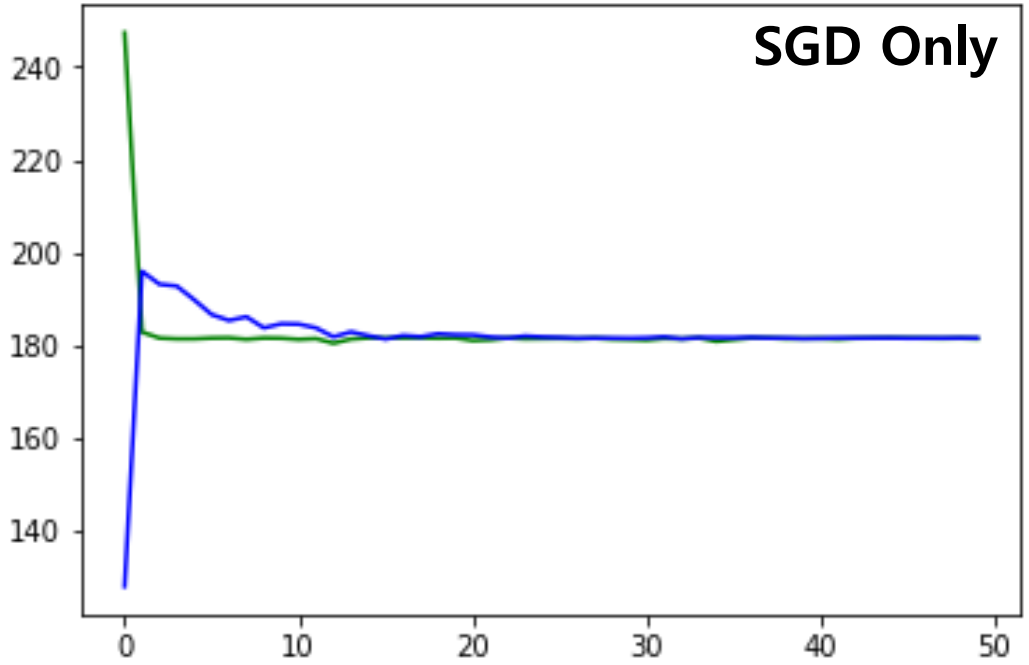
Convergence process

```
gd_lr = linear_model.LinearRegressionGD(eta0=0.001, epochs=10000, batch_size=1, shuffle=False)
bgd_lr = linear_model.LinearRegressionGD(eta0=0.001, epochs=10000, batch_size=len(X), shuffle=False)
sgd_lr = linear_model.LinearRegressionGD(eta0=0.001, epochs=10000, batch_size=1, shuffle=True)
msgd_lr = linear_model.LinearRegressionGD(eta0=0.001, epochs=10000, batch_size=100, shuffle=True)
```

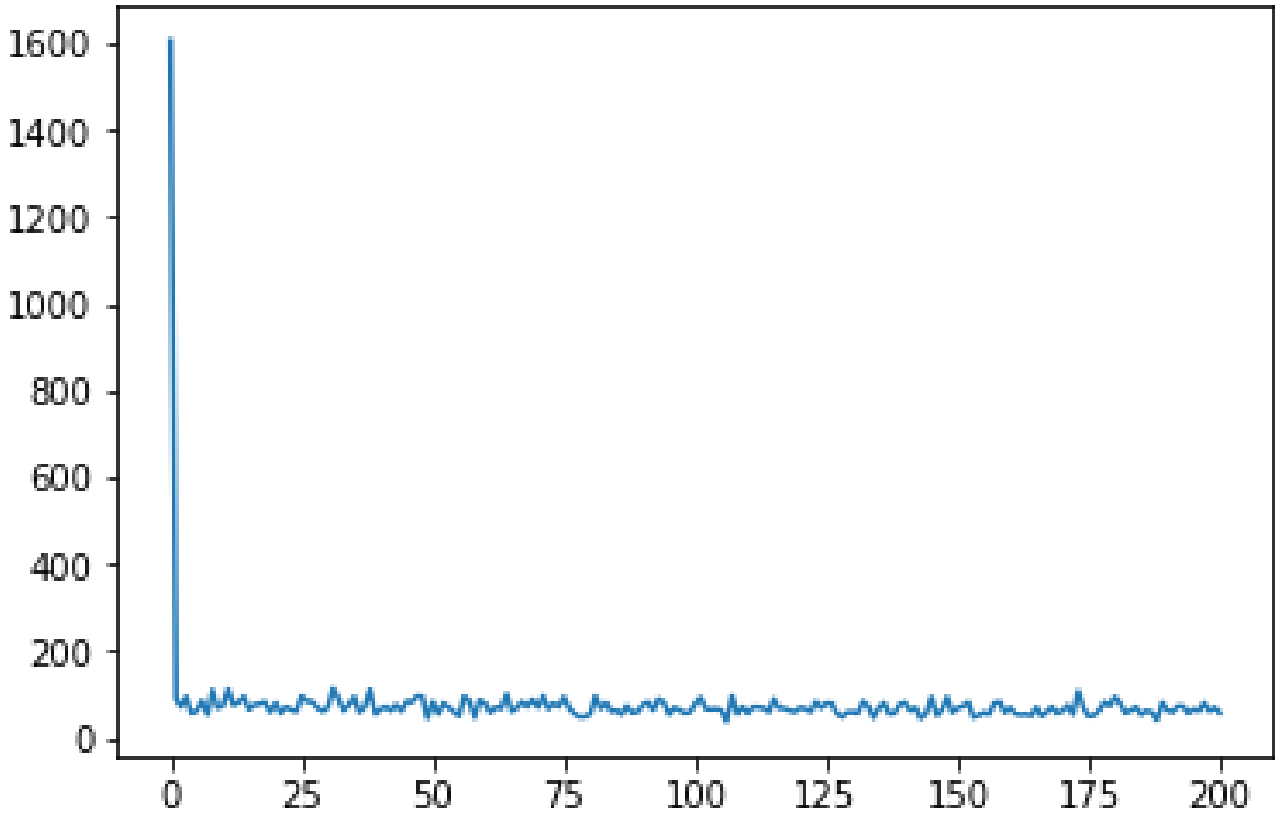
```
for epoch in range(epoches):
    X_copy = np.copy(X)
    if is_SGD:
        np.random.shuffle(X_copy)
    batch = len(X_copy) // BATCH_SIZE
    for batch_count in range(batch):
        X_batch = np.copy(
                X_copy[batch_count*BATCH_SIZE : (batch_count+1)*BATCH_SIZE])
        # Do weight Update
    print("Number of epoch : %d" % epoch)
```

Convergence process





Convergence process



Time-consuming

```
%timeit gd_lr.fit(X,y)
```

Gradient descent

```
1.37 s ± 18.3 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
%timeit bgd_lr.fit(X,y)
```

Full-batch Gradient descent

```
3.74 ms ± 121 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

```
%timeit sgd_lr.fit(X,y)
```

Stochstic Gradient descent

```
1.47 s ± 14.5 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

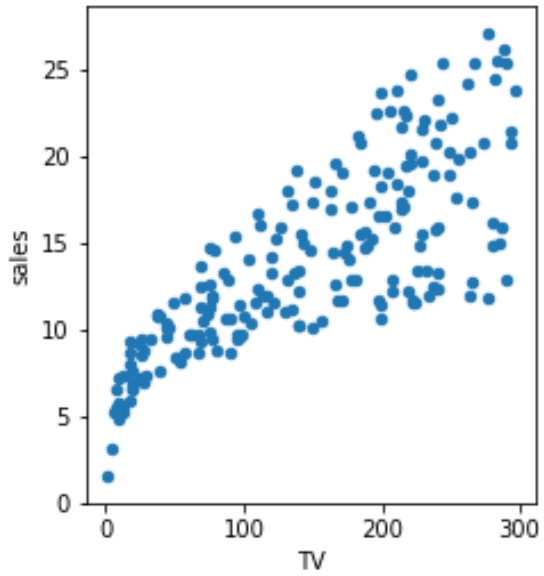
```
%timeit msgd_lr.fit(X,y)
```

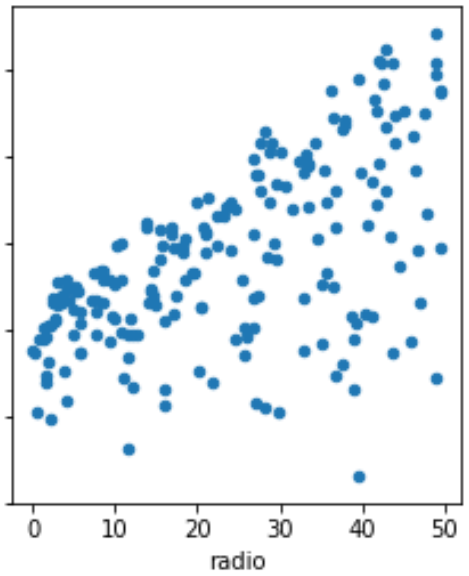
Minibatch-SGD

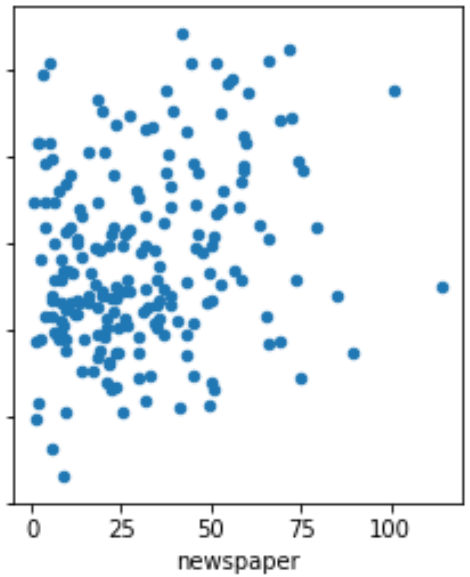
```
122 ms ± 1.07 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Multivariate

	TV	radio	newspaper	sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9



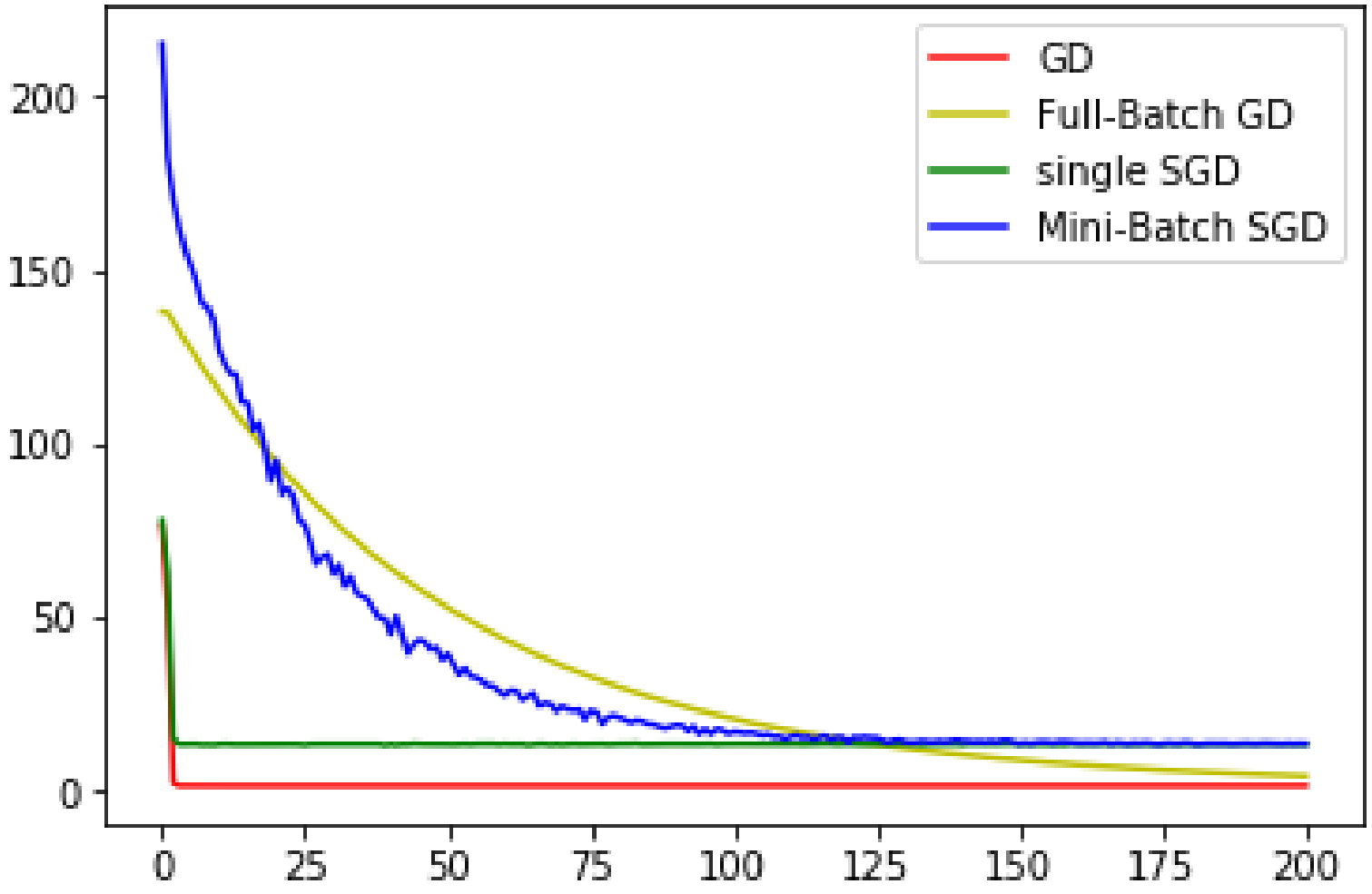




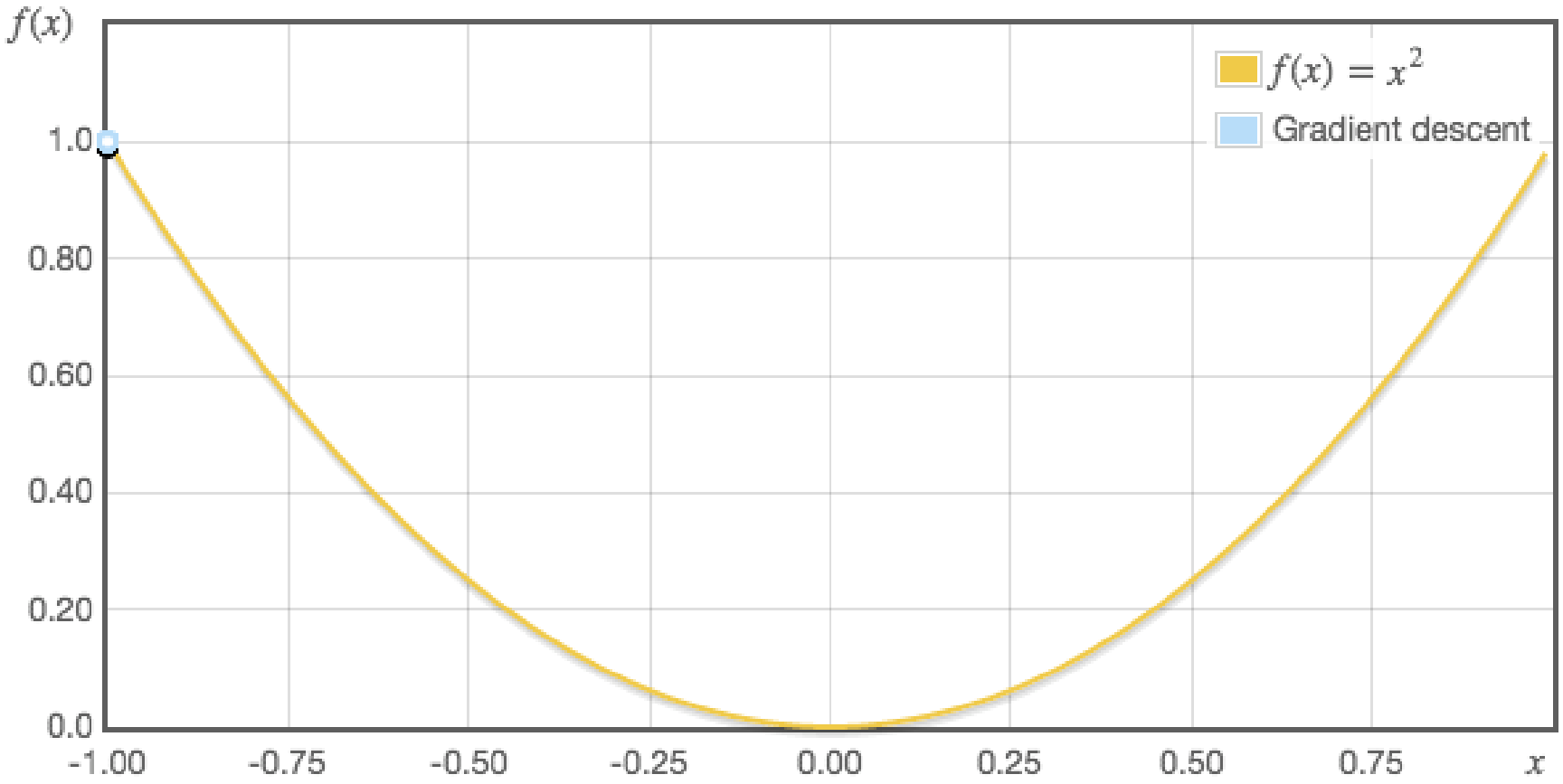

```
from sklearn import preprocessing
X_scaled = preprocessing.scale(ad_cost)
y = sales
X_scaled [:5]
```

```
array([[ 0.96985227,  0.98152247,  1.77894547],
       [-1.19737623,  1.08280781,  0.66957876],
       [-1.51615499,  1.52846331,  1.78354865],
       [ 0.05204968,  1.21785493,  1.28640506],
       [ 0.3941822 , -0.84161366,  1.28180188]])
```

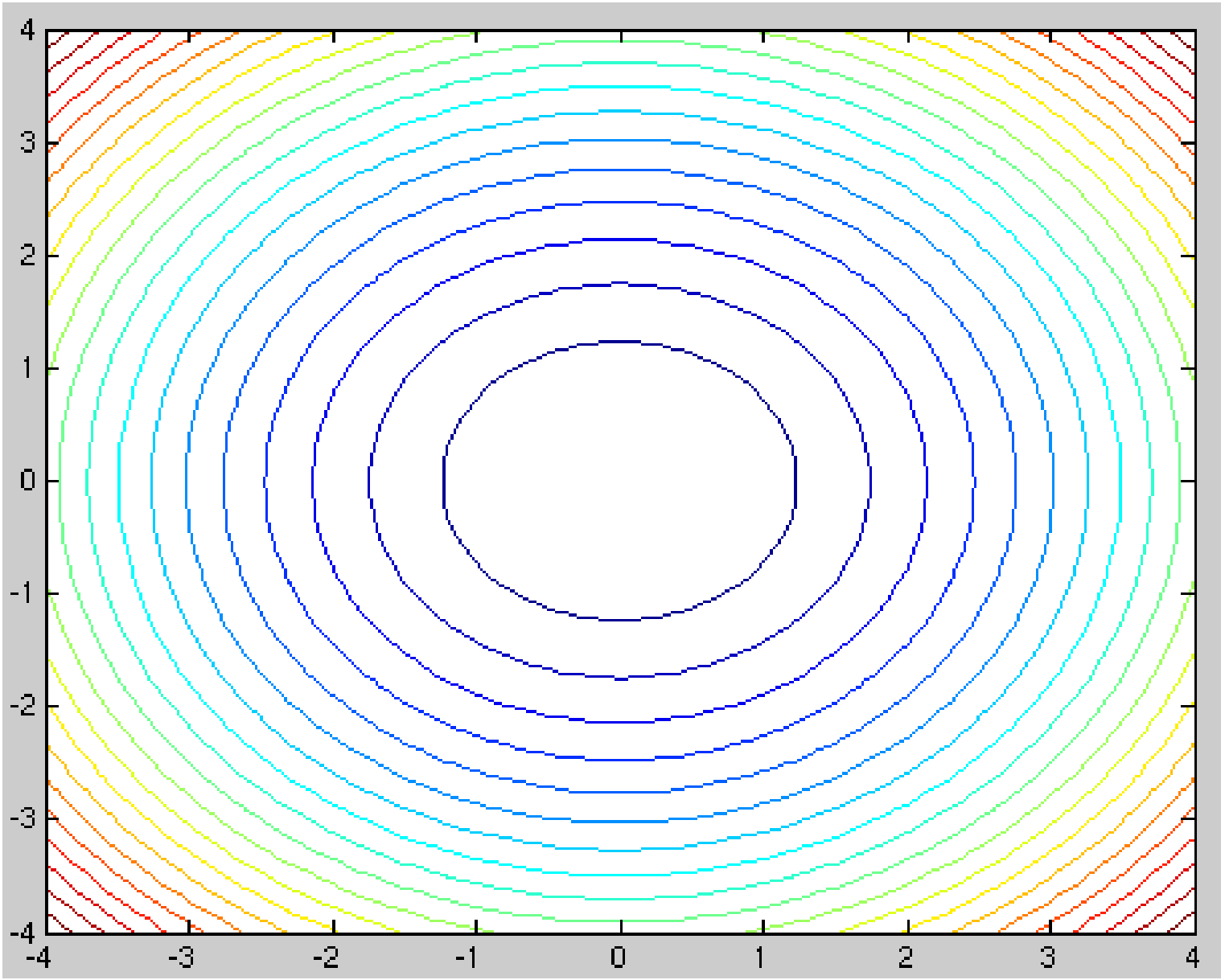
Multivariate



Learning rate는 일정해야 하는가?


$f(x)$
1.0
0.80
0.60
0.40
0.20
0.0
-1.00
-0.75
-0.50
-0.25
0.00
0.25
0.50
0.75
x
$f(x) = x^2$
Gradient descent

contour graph



Learning-rate decay

- 일정한 주기로 Learning rate을 감소시키는 방법
- 특정 epoch 마다 Learning rate를 감소

```
self._eta0 = self._eta0 * self._learning_rate_decay
```

- Hyper-parameter 설정의 어려움
- 지수감소 $\alpha = \alpha_0 e^{-kt}$, 1/t 감소 $\alpha = \frac{\alpha_0}{(1+kt)}$

종료조건 설정

- SGD과정에서 특정 값이하로 cost function이 줄어들지 않을 경우 GD를 멈추는 방법
- 성능이 좋아지지 않는/필요없는 연산을 방지함
- 종료조건을 설정 - tol > loss - previous_loss
- tol은 hyperparameter로 사람 설정함

Human knowledge belongs to the world.

Overfitting and Regularization

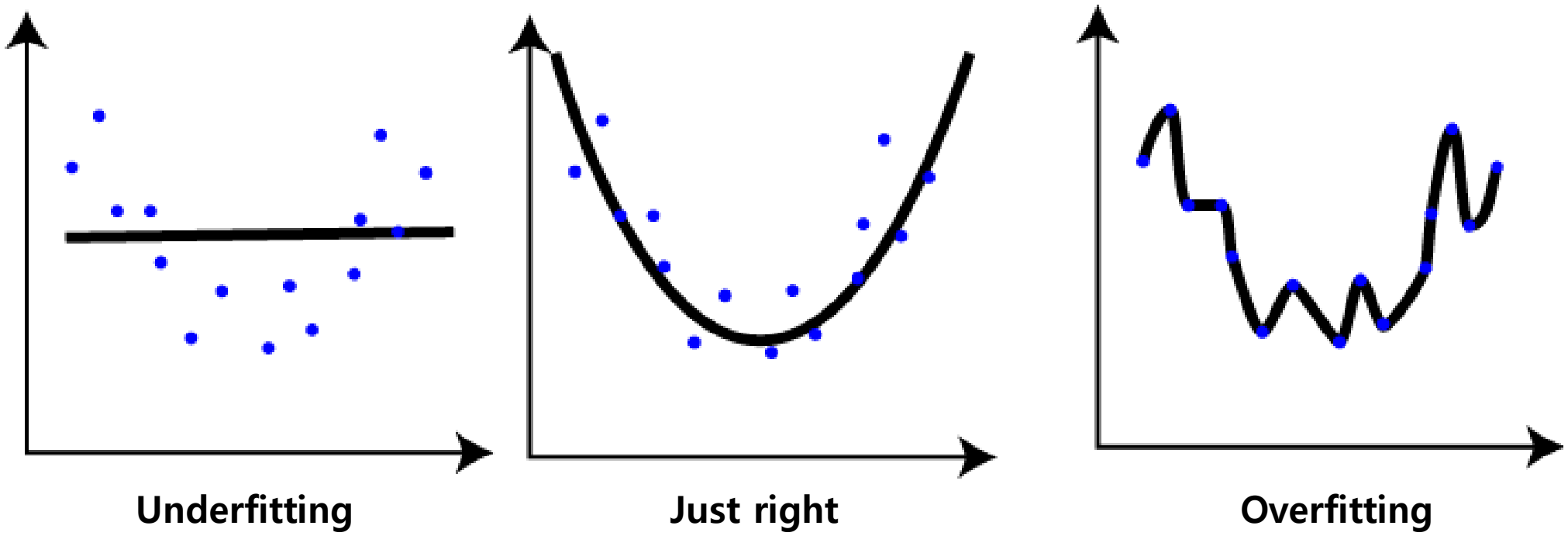
Linear Regression

Director of TEAMLAB
Sungchul Choi

Overfitting

Overfitting

학습데이터 과다 최적화 → 새로운 데이터의 예측 ↓



https://goo.gl/aP8iFa

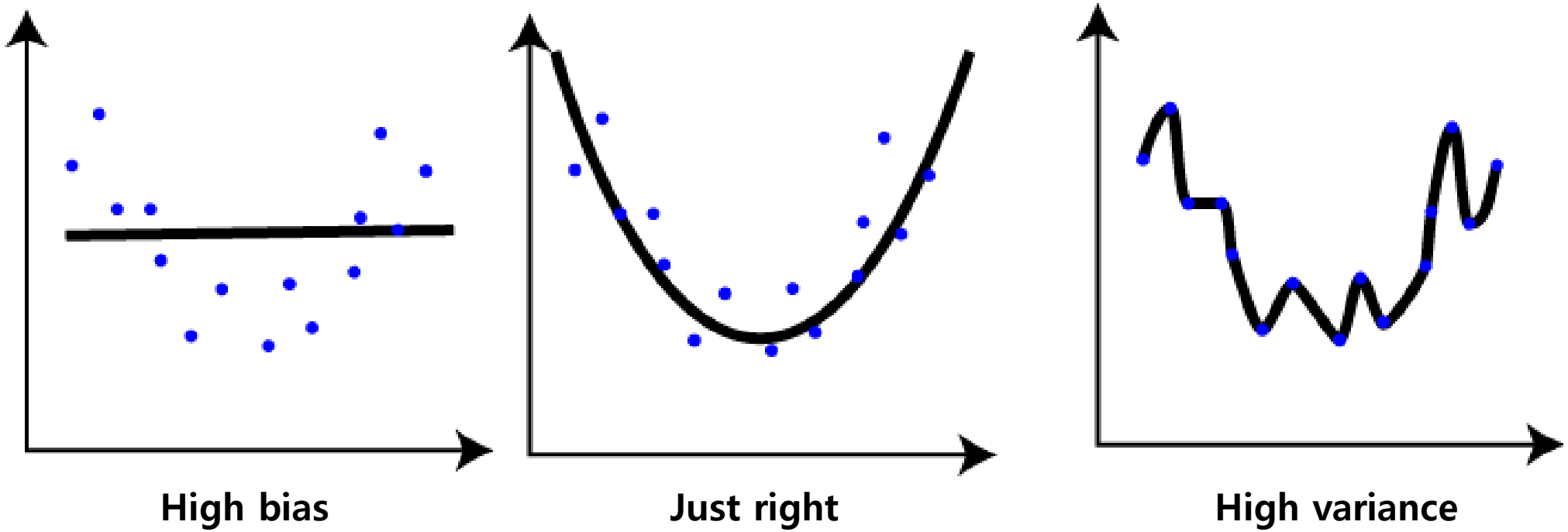
보다 적은 수의 논리로 설명이 가능한 경우, ₩
많은 수의 논리를 세우지 말라

Occam's razor

https://ko.wikipedia.org/wiki/%EC%98%A4%EC%BB%B4%EC%9D%98_%EB%A9%B4%EB%8F%84%EB%82%A0

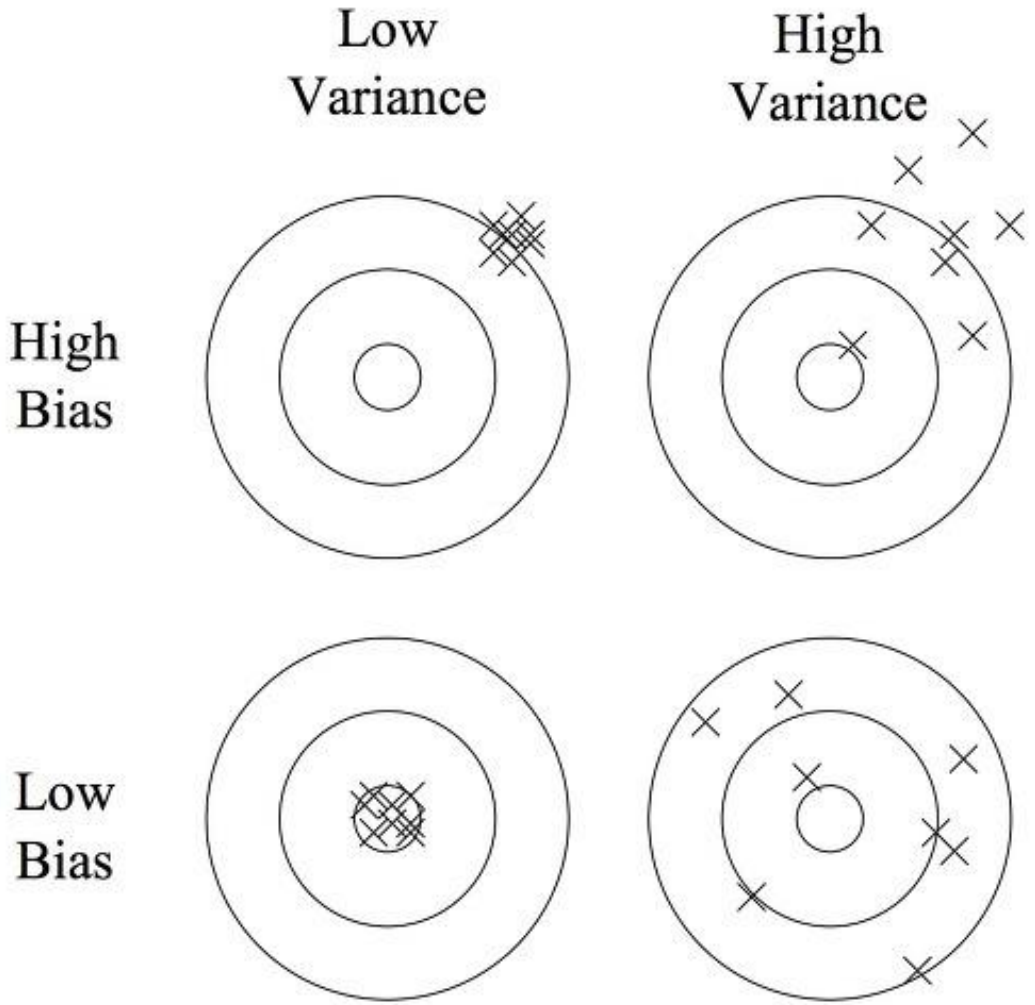
Bias – Variance tradeoff

학습데이터 과다 최적화 → 새로운 데이터의 예측 ↓



https://goo.gl/aP8iFa

Bias – Variance tradeoff

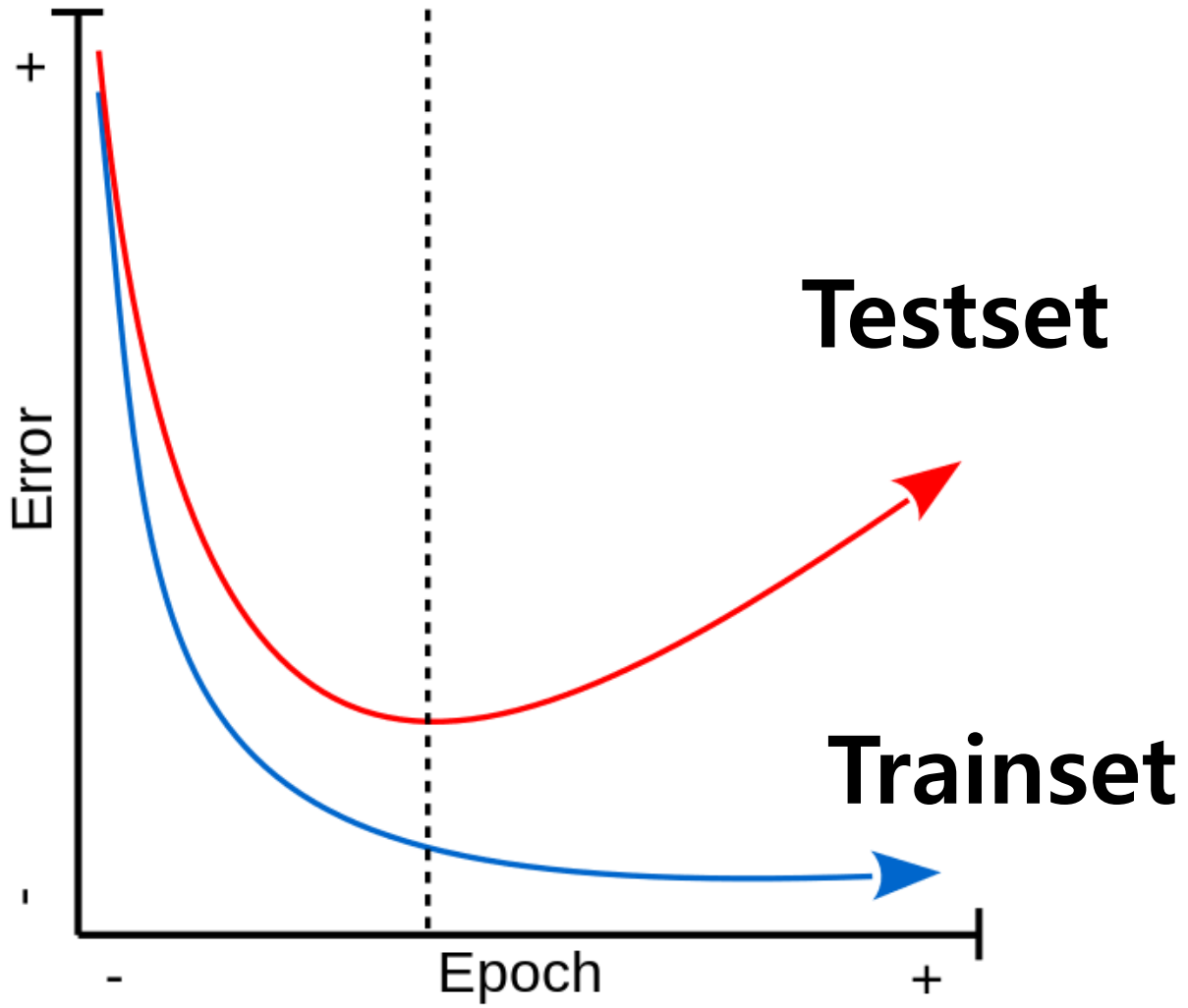


High bias
원래 모델에 많이 떨어짐
잘못된 데이터만 계속 학습함
→ 잘못된 Weight만 Update

High variance
모든 데이터에 민감하게 학습
Error를 고려하지 않음
→ 모든 Weight가 Update

http://scott.fortmann-roe.com/docs/BiasVariance.html

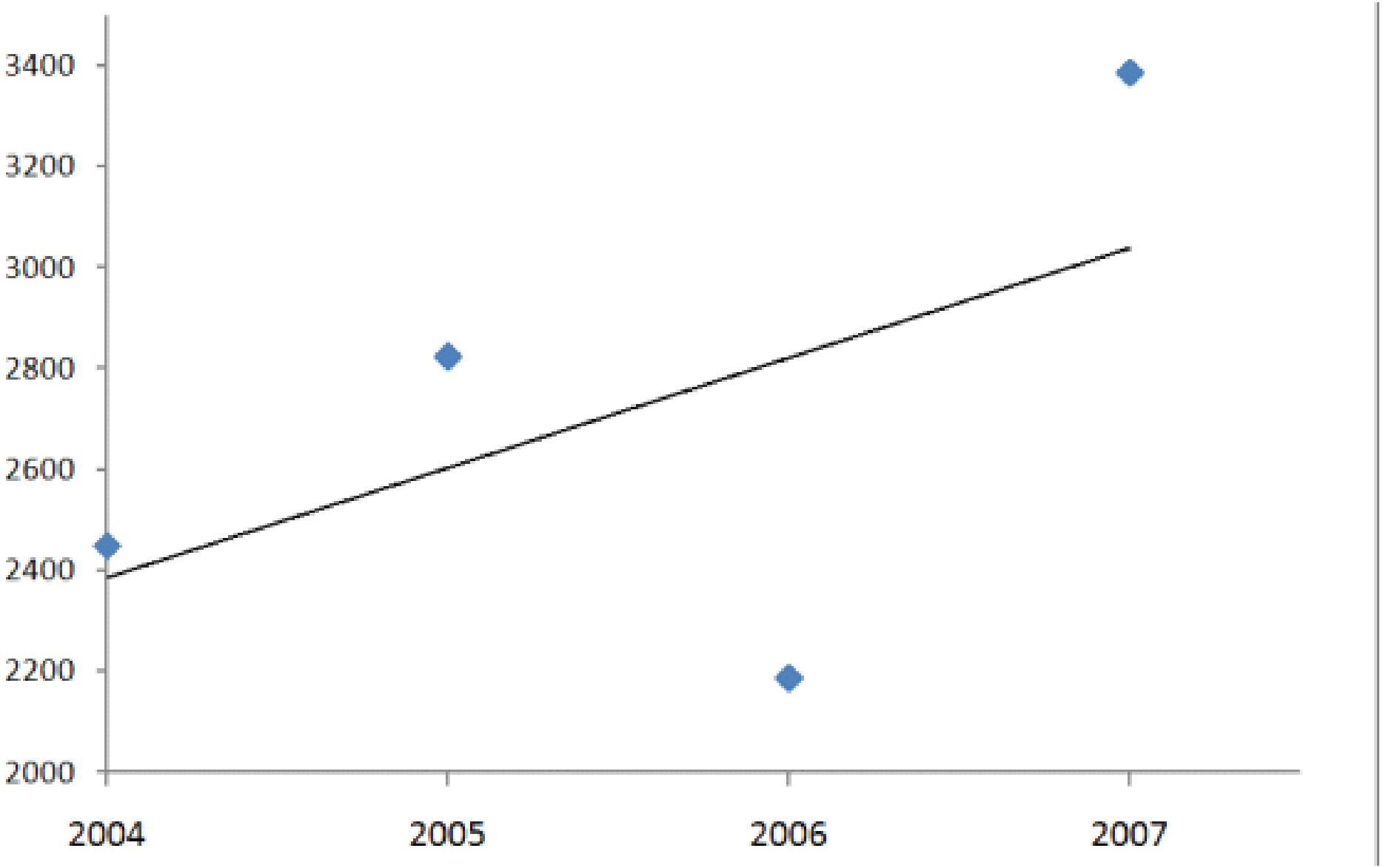
Train-Test Error



Overcoming Overfitting

- 더 많은 데이터를 활용한다.
- Feature의 개수를 줄인다
- 적절히 Parameter를 선정한다
- Regularization

Regularization



Regularization

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)})^2$$

$$\frac{\partial J}{\partial w_0} = \frac{1}{m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)})$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{m} \sum_{i=1}^{m} (w_1 x^{(i)} + w_0 - y^{(i)}) x^{(i)}$$

Human knowledge belongs to the world.

L1 – Regularization / Lasso

Linear Regression

Director of TEAMLAB
Sungchul Choi

L1 regularization

- 기존 Cost function L1(norm) penalty term을 추가

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n}|\theta_j|$$

- norm - 벡터의 길이 혹은 크기를 측정하는 방법

$\|x\|_1 := \sum_{i=1}^{n} |x_i|$

L1는 manhattan distance
원점에서 벡터 좌표까지의 거리

http://taewan.kim/post/norm/

L2 regularization

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n} \theta_j^2$$

$$\theta_0 := \theta_0 - \alpha \, \frac{1}{m} \, \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \, \left[\left(\frac{1}{m} \, \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right] j \in \{1, 2 ... n\} \qquad \}$$

L2 regularization

$$\theta_j := \theta_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right]$$

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Normal equation approach

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n} \theta_j^2$$

$$\begin{aligned} J(\theta) &= (y - X\theta)^T (y - X\theta) + \lambda \theta^T \theta \\ &= y^T y - \theta^T X^T y - y^T X \theta + \theta^T X^T X \theta + \lambda \theta^T \theta \\ &= y^T y - \theta^T X^T y - \theta^T X^T y + \theta^T X^T X \theta + \theta^T \lambda I \theta \\ &= y^T y - 2\theta^T X^T y + \theta^T (X^T X + \lambda I) \theta \end{aligned}$$

Normal equation approach

$$J(\theta) = y^T y - 2\theta^T X^T y + \theta^T (X^T X + \lambda I)\theta$$

$$\frac{\partial J(\theta)}{\partial \theta} = -2X^T y + 2(X^T X + \lambda I)\theta$$

$$(X^T X + \lambda I)\theta = X^T y \rightarrow \hat{\theta} = (X^T X + \lambda I)^{-1} X^T y$$

Human knowledge belongs to the world.

L2 – Regularization / Ridge

Linear Regression

Director of TEAMLAB
Sungchul Choi

L2 regularization

- 기존 Cost function L2(norm) penalty term을 추가

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n}\theta_j^2$$

- norm - 벡터의 길이 혹은 크기를 측정하는 방법

$\|(\theta)\|_2^2$ L2는 Euclidean distance
원점에서 벡터 좌표까지의 거리

L2 regularization

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n} \theta_j^2$$

$$\theta_0 := \theta_0 - \alpha \, \frac{1}{m} \, \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \, \left[\left(\frac{1}{m} \, \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right] j \in \{1, 2 ... n\} \qquad \}$$

L2 regularization

$$\theta_j := \theta_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right]$$

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Normal equation approach

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n} \theta_j^2$$

$$\begin{aligned} J(\theta) &= (y - X\theta)^T (y - X\theta) + \lambda \theta^T \theta \\ &= y^T y - \theta^T X^T y - y^T X \theta + \theta^T X^T X \theta + \lambda \theta^T \theta \\ &= y^T y - \theta^T X^T y - \theta^T X^T y + \theta^T X^T X \theta + \theta^T \lambda I \theta \\ &= y^T y - 2\theta^T X^T y + \theta^T (X^T X + \lambda I) \theta \end{aligned}$$

Normal equation approach

$$J(\theta) = y^T y - 2\theta^T X^T y + \theta^T (X^T X + \lambda I)\theta$$

$$\frac{\partial J(\theta)}{\partial \theta} = -2X^T y + 2(X^T X + \lambda I)\theta$$

$$(X^T X + \lambda I)\theta = X^T y \rightarrow \hat{\theta} = (X^T X + \lambda I)^{-1} X^T y$$

L1 regularization

- 기존 Cost function L1(norm) penalty term을 추가

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n}|\theta_j|$$

- norm - 벡터의 길이 혹은 크기를 측정하는 방법

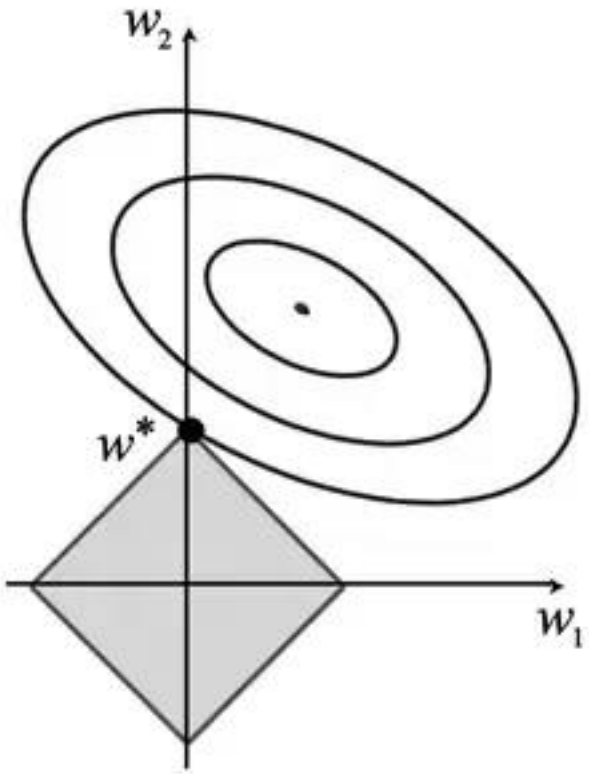
$\|x\|_1 := \sum_{i=1}^{n} |x_i|$

L1는 manhattan distance
원점에서 벡터 좌표까지의 거리

http://taewan.kim/post/norm/

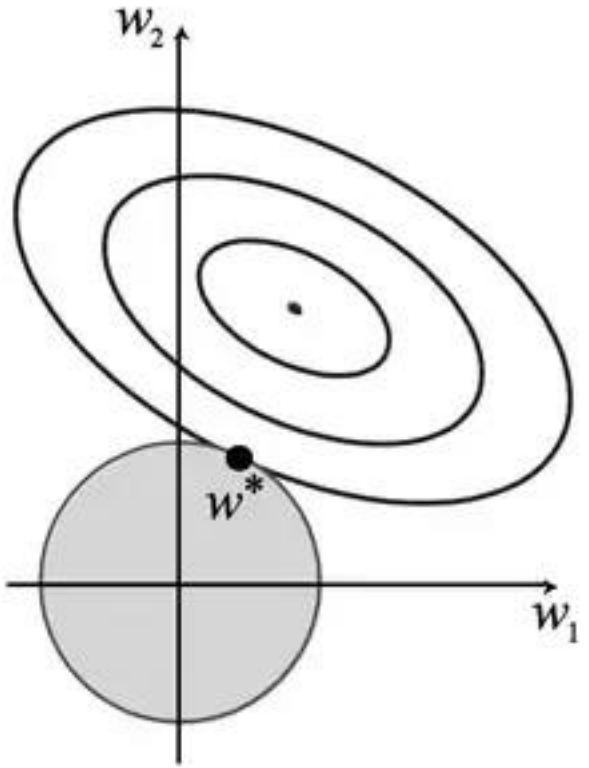
L1 vs L2

$$\sum_{j=1}^{2} |w_i| \leq s$$

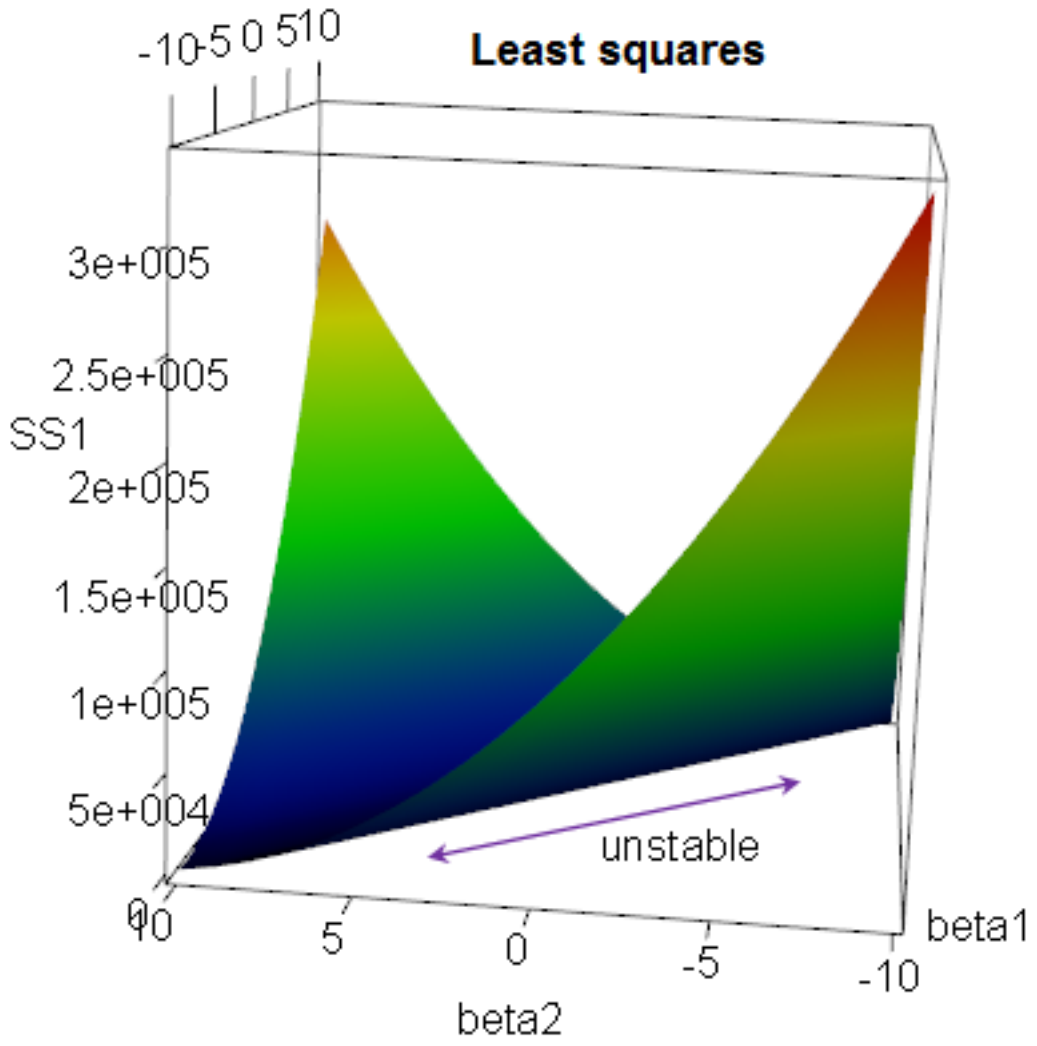


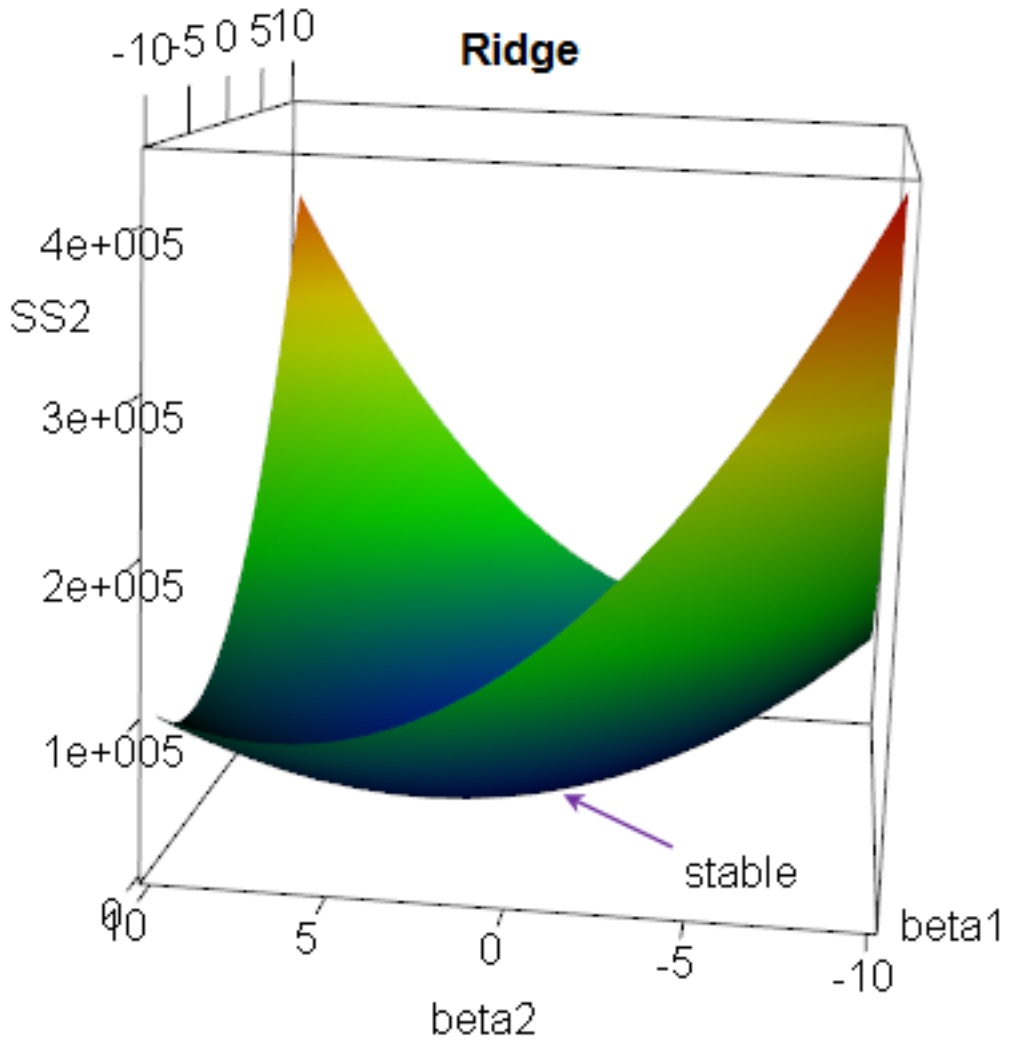
L1

$$\sum_{j=1}^{2} (w_i)^2 \leq s$$



L2





https://stats.stackexchange.com/questions/151304/why-is-ridge-regression-called-ridge-why-is-it-needed-and-what-happens-when

L1	L2
Unstable solution	Stable solution
Always on solution	Only one solution
Sparse solution	Non-sparse solution
Feature selection	

Human knowledge belongs to the world.

Overview

Logistic Regression Classifier

Director of TEAMLAB
Sungchul Choi

머신러닝의 학습 방법들

- Gradient descent based learning
- Probability theory based learning
- Information theory based learning
- Distance similarity based learning

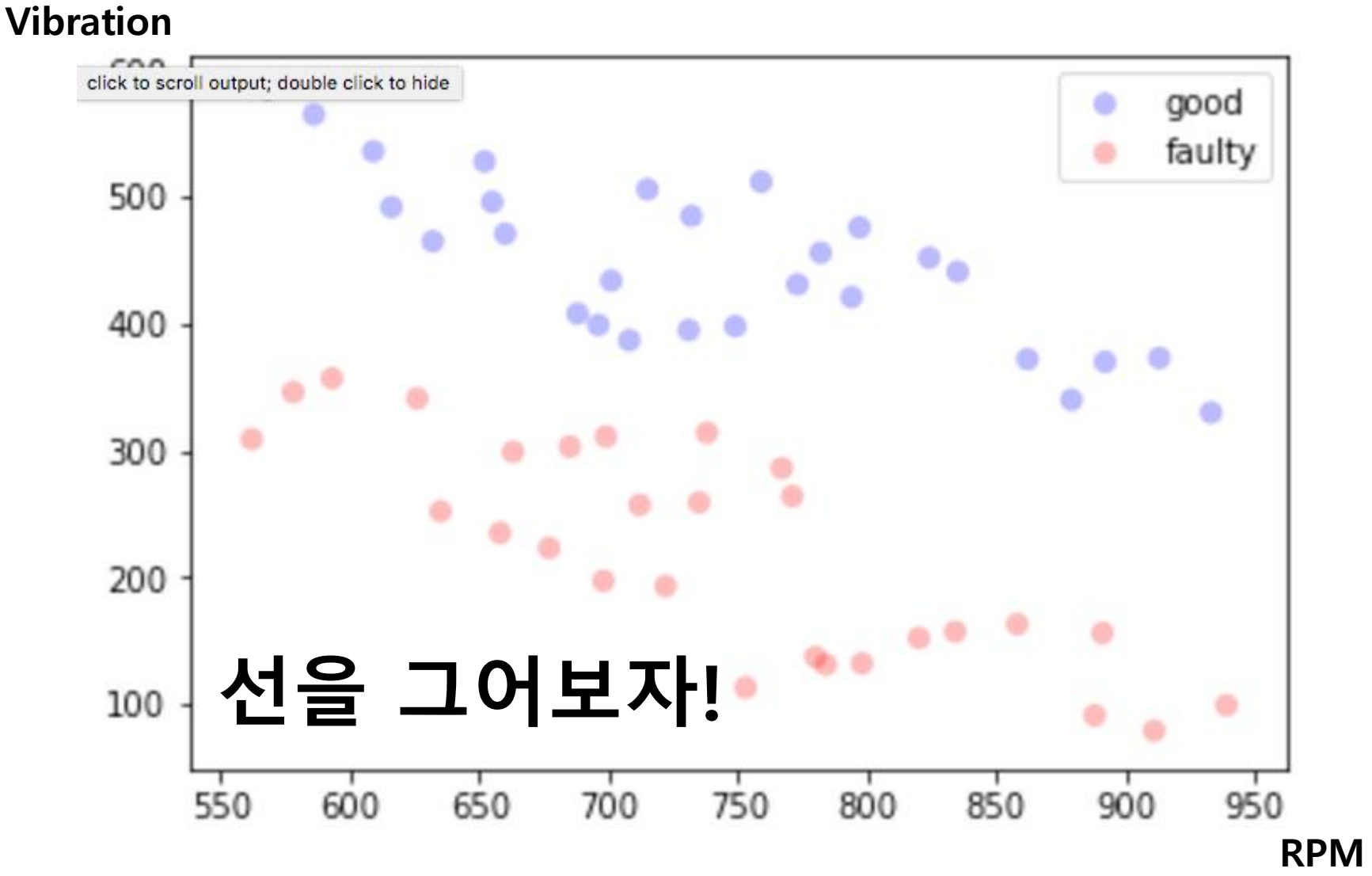
Classification Problem

- 발전소 발전기가 제대로 돌아가고 있는가?

ID	RPM	VIBRATION	STATUS
1	568	585	good
2	586	565	good
3	609	536	good
4	616	492	good
5	632	465	good
6	652	528	good
7	655	496	good
8	660	471	good
9	688	408	good
10	696	399	good
11	708	387	good
12	701	434	good
13	715	506	good
14	732	485	good
15	731	395	good
16	749	398	good
17	759	512	good
18	773	431	good
19	782	456	good
20	797	476	good
21	794	421	good
22	824	452	good
23	835	441	good
24	862	372	good
25	879	340	good
26	892	370	good
27	913	373	good
28	933	330	good

ID	RPM	VIBRATION	STATUS
29	562	309	faulty
30	578	346	faulty
31	593	357	faulty
32	626	341	faulty
33	635	252	faulty
34	658	235	faulty
35	663	299	faulty
36	677	223	faulty
37	685	303	faulty
38	698	197	faulty
39	699	311	faulty
40	712	257	faulty
41	722	193	faulty
42	735	259	faulty
43	738	314	faulty
44	753	113	faulty
45	767	286	faulty
46	771	264	faulty
47	780	137	faulty
48	784	131	faulty
49	798	132	faulty
50	820	152	faulty
51	834	157	faulty
52	858	163	faulty
53	888	91	faulty
54	891	156	faulty
55	911	79	faulty
56	939	99	faulty

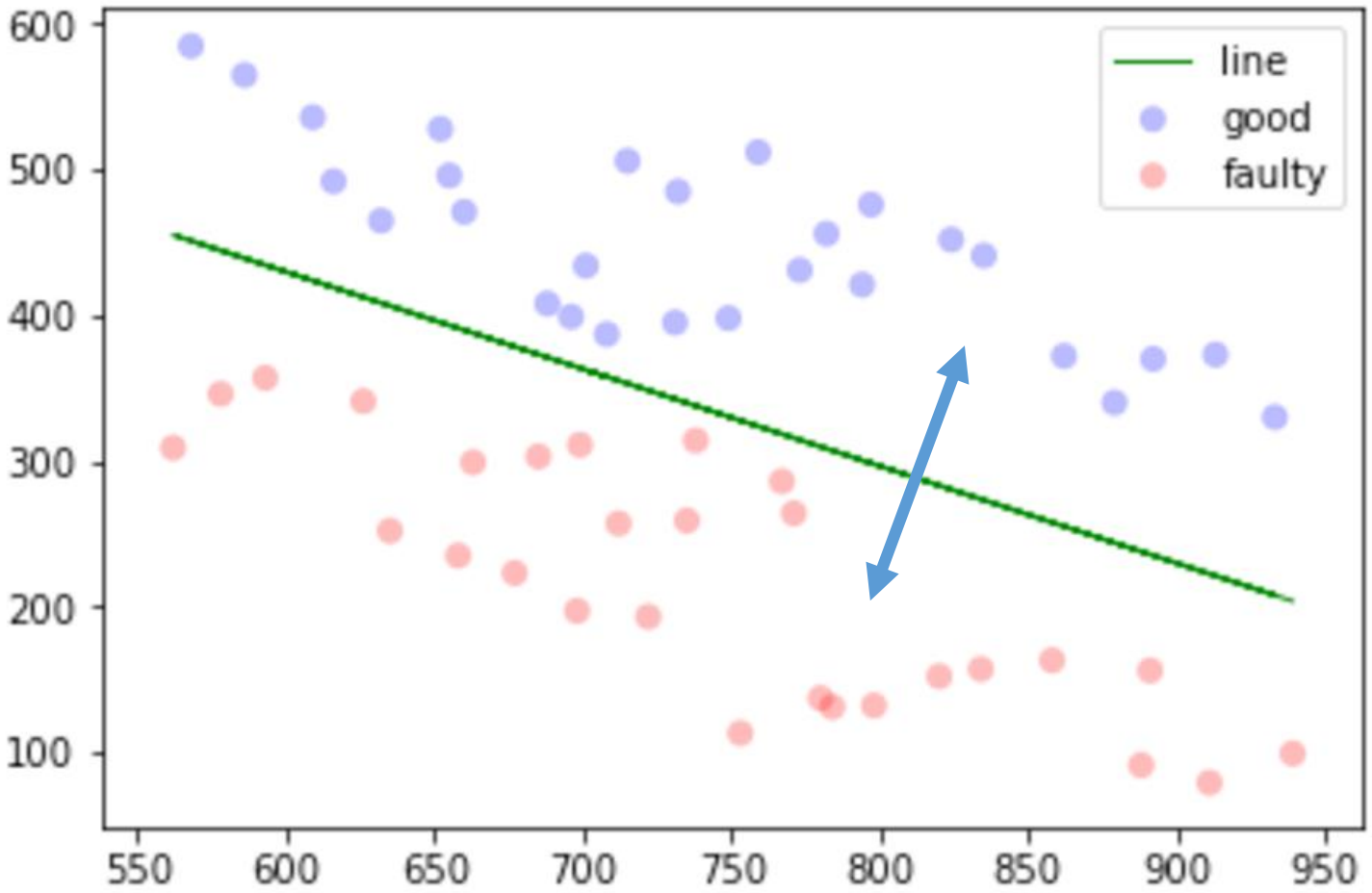
Classification Problem



Classification Problem

Vibration

$$\text{Vibration} = 830 - 0.667 * \text{RPM}$$



RPM

Classification Problem

$$f(x) = 803 - 0.667 * RPM - 1 * Vibration = 0$$

$$Status = \begin{cases} 1 & \text{if } f(x) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Classification Problem

어떻게 학습을 시킬까?

$$f(x) = \underline{803} - \underline{0.667} * RPM - \underline{1} * Vibration = 0$$

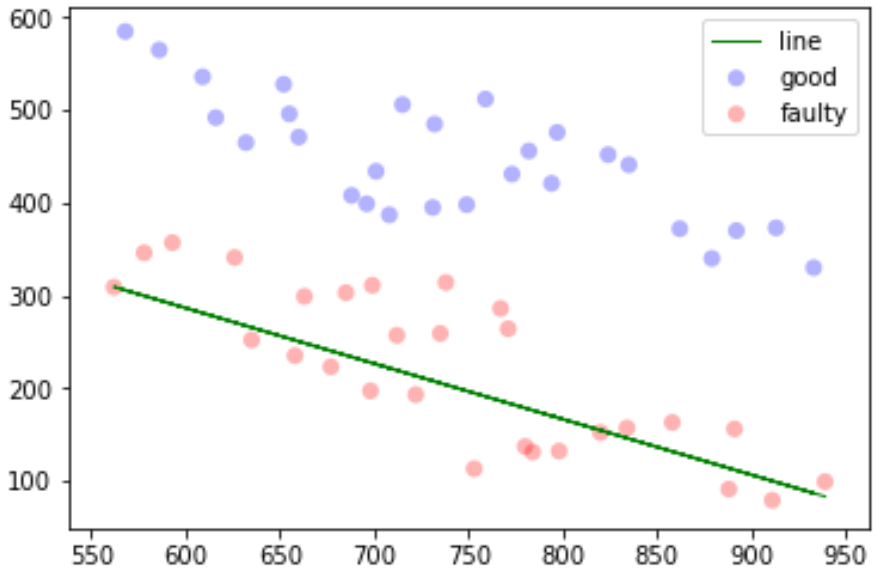
$$Status = \begin{cases} 1 & \text{if } f(x) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Classification Problem

Linear Regression으로 학습해보자

$$f(x) = -2.450750883259760 6 + 0.00227488 * RPM + 0.00379006 * Vibration$$

```
array([  1.05856314e+00,   1.02370973e+00,   9.66120163e-01,
click to scroll output; double click to hide
         9.19162092e-01,   8.35784996e-01,   6.60707802e-01,
         6.44796277e-01,   6.26614080e-01,   7.88822723e-01,
         1.09355523e+00,   1.05263689e+00,   7.09256697e-01,
         7.61574640e-01,   1.21639013e+00,   9.41243609e-01,
         1.05646897e+00,   1.16639329e+00,   9.51115412e-01,
         1.13685352e+00,   1.12018650e+00,   9.20094079e-01,
         8.37485080e-01,   9.80760238e-01,   1.03990281e+00,
         9.22427788e-01,  -1.14240215e-03,   1.75487796e-01,
         2.51301583e-01,   2.65731543e-01,  -5.11098276e-02,
        -6.32186858e-02,   1.90719471e-01,  -6.54767525e-02,
         2.55926977e-01,  -1.16245894e-01,   3.18095711e-01,
         1.43005910e-01,  -7.68091088e-02,   2.02908173e-01,
         4.18186047e-01,  -3.09492679e-01,   3.78035795e-01,
         3.03754000e-01,  -1.57109613e-01,  -1.70750464e-01,
        -1.35112143e-01,  -9.26369286e-03,   4.15348645e-02,
         1.18872240e-01,  -8.57657338e-02,   1.67412730e-01,
        -7.89242969e-02,   6.05734081e-02])
```



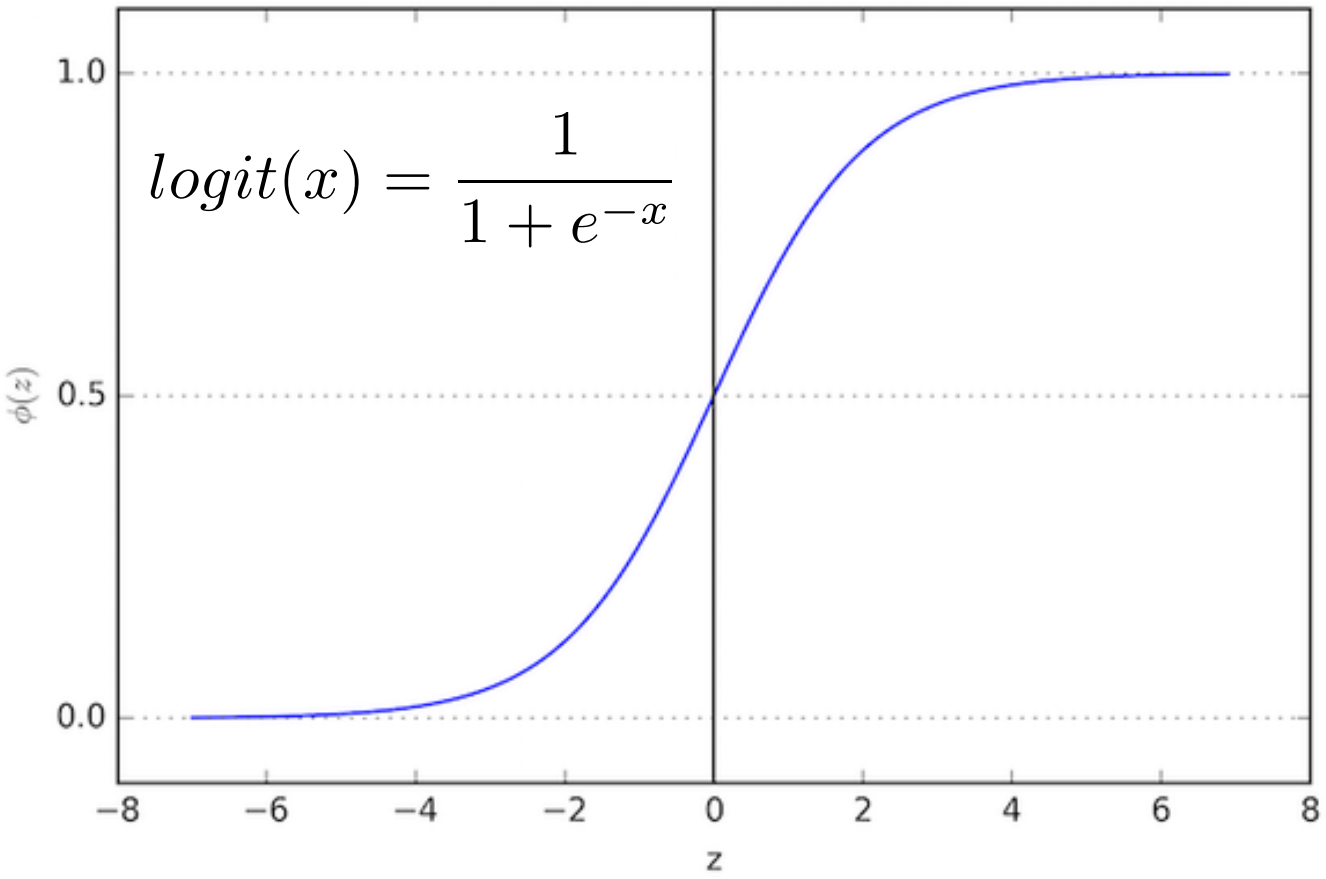
기존 접근의 문제점들

- 1이상 또는 0이하의 수들이 나오는 걸 어떻게 해석?
- 1 또는 0으로 정확히 표현 가능한가? $Status = \begin{cases} 1 & \text{if } f(x) \geq 0 \\ 0 & \text{otherwise} \end{cases}$
- 변수가 Y에 영향을 주는정도가 비례하는 가?
- **확률로** 발생할 사건의 가능성을 표현해야 함

$$f(x) = -2.4507508832597606 + 0.00227488 * RPM + 0.00379006 * Vibration$$

Solution

확률로 나타내자!



Human knowledge belongs to the world.

Sigmoid function

Logistic Regression Classifier

Director of TEAMLAB
Sungchul Choi

분류의 가능성을 확률로 얘기하기

어떤 사건이 일어날 확률

$$P(X)$$

일어날 확률

$$1 - P(X)$$

일어나지 않을 확률

$$0 \leq P(X) \leq 1$$

Odds Ratio

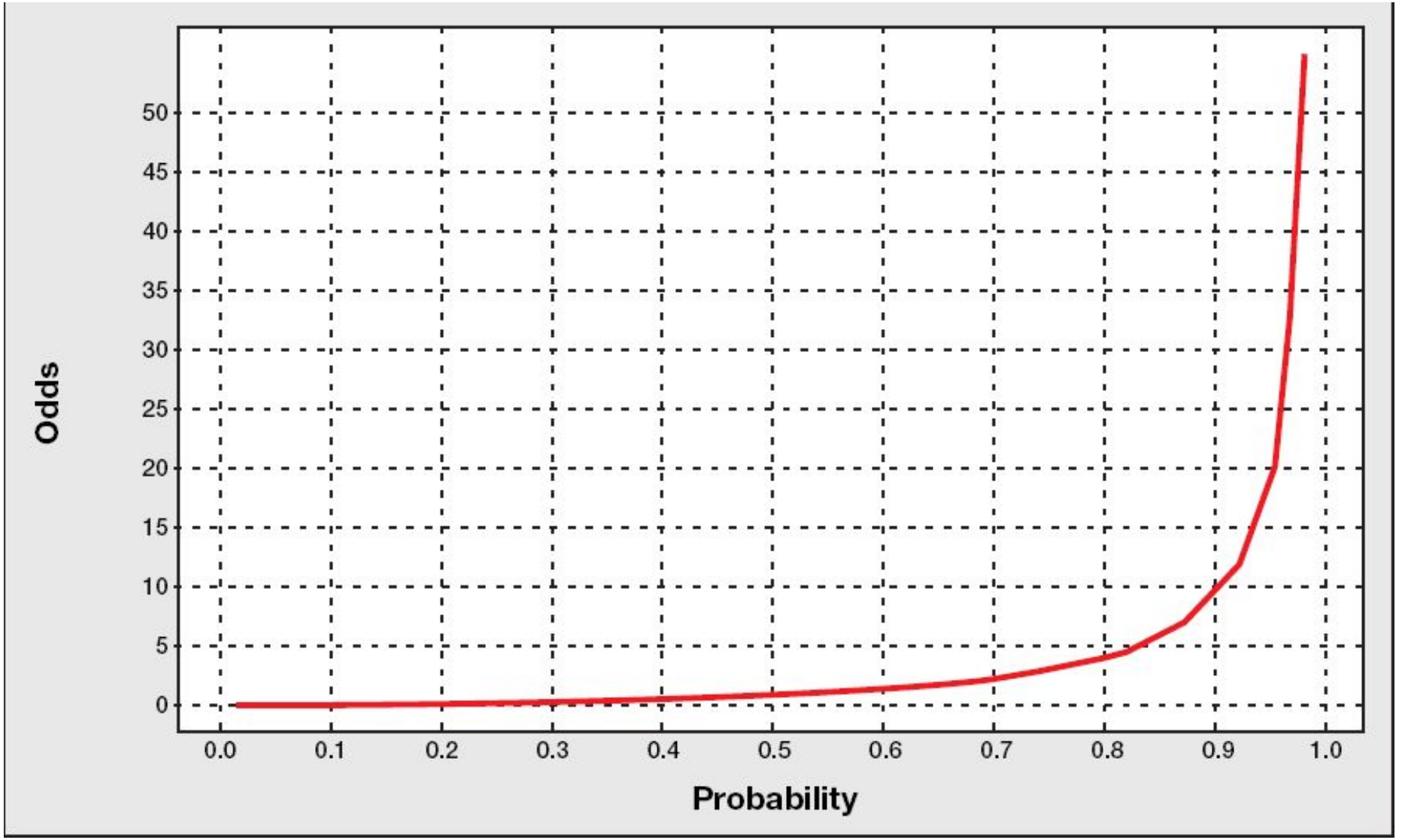
해당 사건이 일어날 확률과 일어나지 않을 확률의 비율

일어날 확률 $P(X)$

일어나지 않을 확률 $1 - P(X)$

$$\frac{P(X)}{1 - P(X)}$$

Odds Ratio

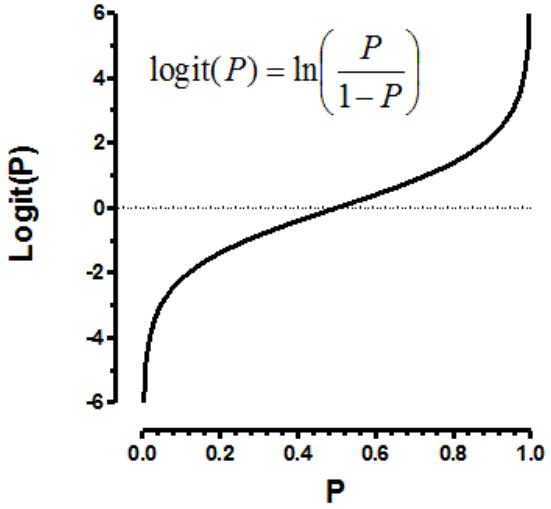


https://goo.gl/9a6zkS

Logit function

X의 값이 주어졌을 때 y의 확률을 이용한 log odds

$$logit(p(y=1|x)) = log_e\left(\frac{p}{1-p}\right)$$



$$= log_e(p) - log_e(1-p)$$

$$= -log_e\left(\frac{1}{p} - 1\right)$$

https://goo.gl/9a6zkS

Sigmoid(=Logistic) Function

Logit 함수의 역함수로 z에 관한 확률을 산출

$$f(z) = y = -log_e\left(\frac{1}{z} - 1\right)$$ 역함수로 바꾸면

$$z = -log_e\left(\frac{1}{y} - 1\right)$$ y에 관한 정리

Sigmoid(=Logistic) Function

$$z = -log_e\left(\frac{1}{y} - 1\right)$$

y에 관한 정리

$$e^{-z} = \frac{1-y}{y}$$

$$y * e^{-z} + y = 1$$

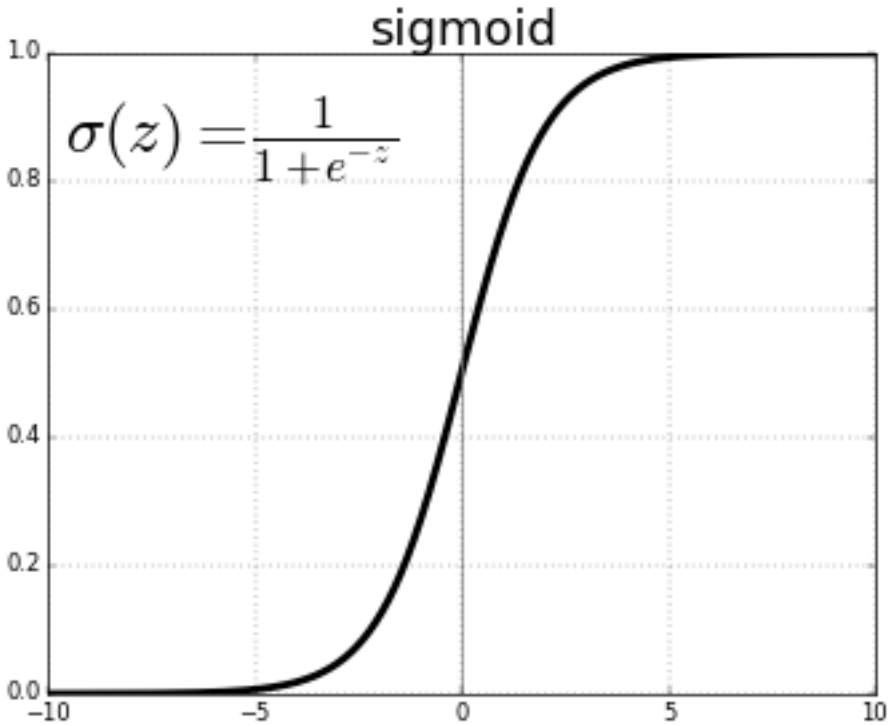
$$y(e^{-z} + 1) = 1$$

$$y = \frac{1}{1+e^{-z}}$$

Logistic Function =
Inverse of logit function

Sigmoid(=Logistic) Function

미분가능한 연속구간으로 변환
S형태로 닮았다고 하여 sigmoid function으로 호칭



https://goo.gl/38SsHw

Sigmoid(=Logistic) Function

선형함수에서 Sigmoid function으로 변환

$$p = \sigma(z) = \frac{1}{1+e^{-z}}, \quad \frac{p}{1-p} = \frac{\frac{1}{1+e^{-z}}}{\frac{e^{-z}}{1+e^{-z}}} = \frac{1}{e^{-z}} = e^{z}$$

$$log_e \frac{p}{1-p} = z$$

$$log_e \frac{p}{1-p} = z = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n$$

Human knowledge belongs to the world.

Cost function

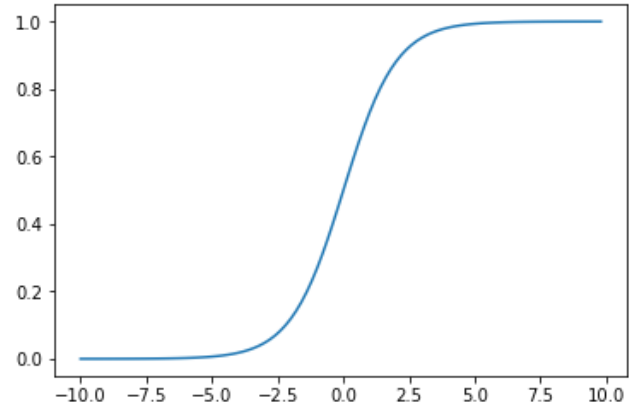
Logistic Regression Classifier

Director of TEAMLAB
Sungchul Choi

Logistic Regression에서 Weight 학습하기

가설 함수

$$h_\theta(x) = g(z) = \frac{1}{1 + e^{-z}}$$



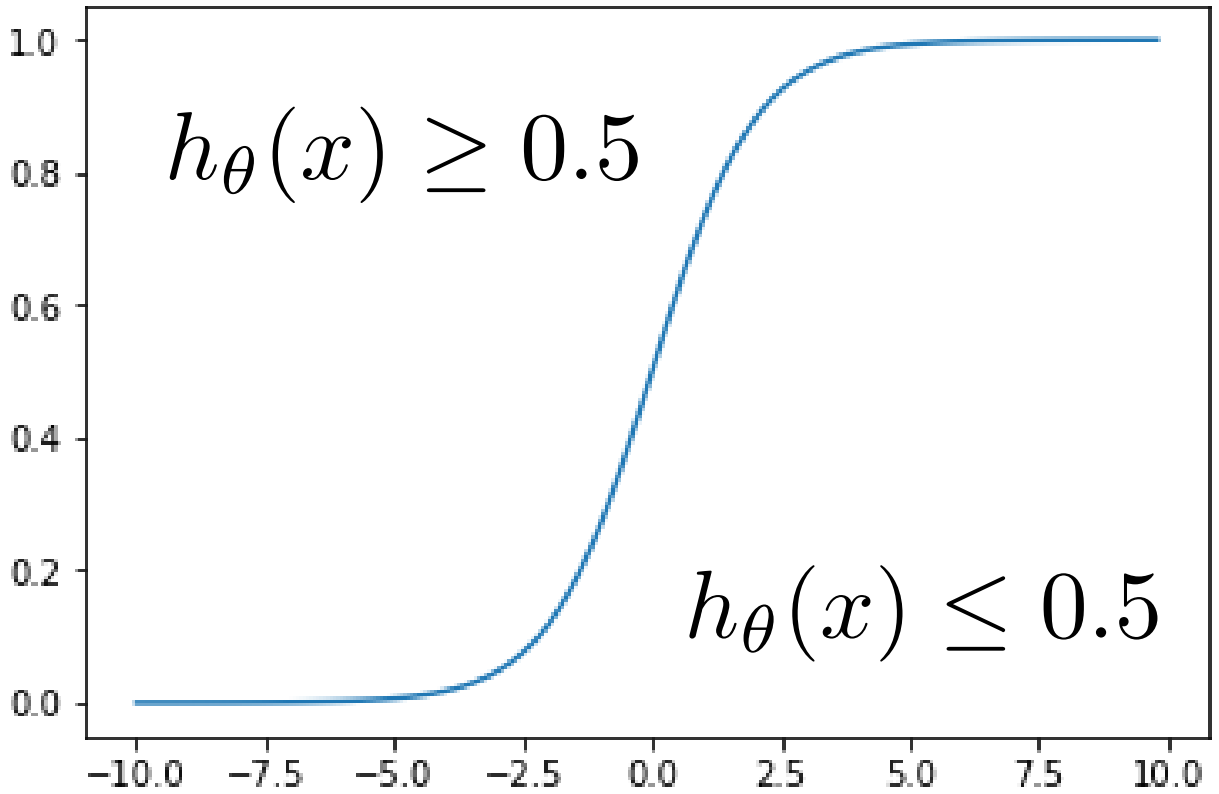
where:

$$\begin{aligned} z &= w_0 x_0 + w_1 x_1 + \cdots + w_n x_n \\ &= \theta^T \mathbf{x} \end{aligned}$$

$$0 \leq h_\theta(x) \leq 1$$

가설 함수

$$h_\theta(x) = g(z) = \frac{1}{1 + e^{-z}}$$



Training θ

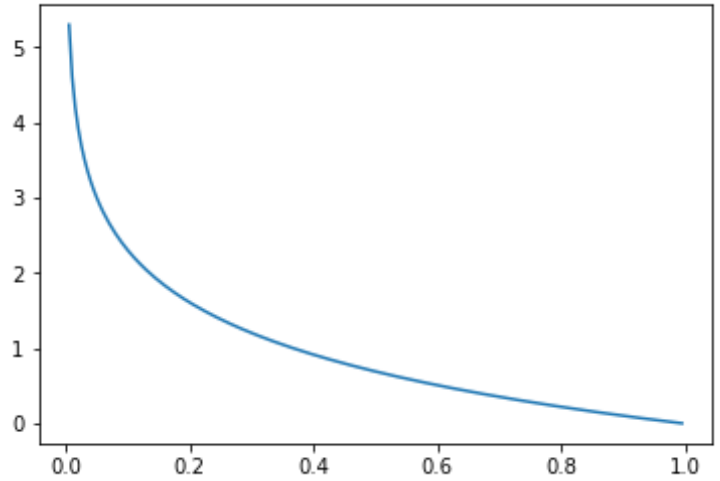
$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$$

ID	RPM	VIBRATION	STATUS
1	568	585	good
2	586	565	good
3	609	536	good
4	616	492	good
5	632	465	good
6	652	528	good
7	655	496	good
8	660	471	good
9	688	408	good
10	696	399	good

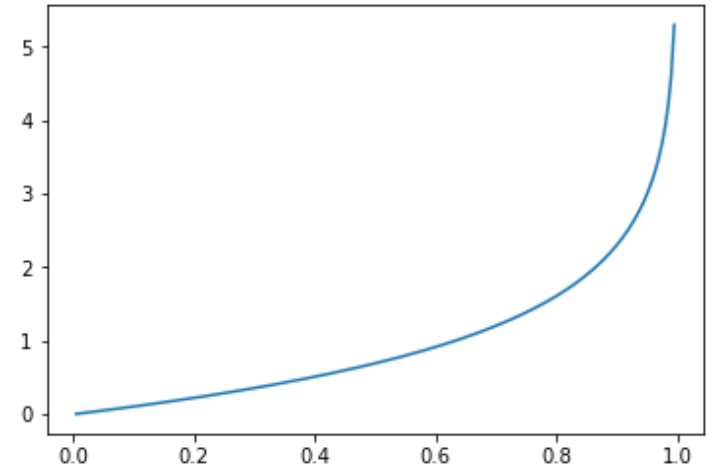
$$\theta^T \mathbf{x} = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n$$

$$y = 0 \text{ or } 1$$

Cost Function



$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



Cost Function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}\left(h_\theta(x^{(i)}), y^{(i)}\right)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

$$\text{find } \theta, \text{ where } \min_\theta J(\theta) \qquad h_\theta(x) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$$

Partial derivation of cost function

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[-y^i(\log(1+e^{-\theta x^i})) + (1-y^i)(-\theta x^i - \log(1+e^{-\theta x^i}))\right]$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i\theta x^i - \theta x^i - \log(1+e^{-\theta x^i})\right]$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left[y_i\theta x^i - \log(1+e^{\theta x^i})\right]$$

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T \mathbf{x}}}$$

$$-\theta x^i - \log(1+e^{-\theta x^i}) = -\left[\log e^{\theta x^i} + \log(1+e^{-\theta x^i})\right]$$

$$= -\log(1+e^{\theta x^i}).$$

Partial derivation of cost function

$$-\frac{1}{m}\sum_{i=1}^{m}\left[y_i\theta x^i - \log(1+e^{\theta x^i})\right]$$

$$z = w_0x_0 + w_1x_1 + \cdots + w_nx_n = \theta^T\mathbf{x}$$

θ 에 관하여 미분하면

$$\frac{\partial}{\partial\theta_j}y_i\theta x^i = y_i x_j^i \quad \frac{d}{dx}ln(2x) = \frac{2}{2x} = \frac{1}{x} \quad \frac{d}{dx}e^{2x} = 2e^x$$

$$\frac{\partial}{\partial\theta_j}\log(1+e^{\theta x^i}) = \frac{x_j^i e^{\theta x^i}}{1+e^{\theta x^i}} = x_j^i h_\theta(x^i),$$

Partial derivation of cost function

$$\frac{\partial}{\partial \theta_j} \log(1 + e^{\theta x^i}) = \frac{x_j^i e^{\theta x^i}}{1 + e^{\theta x^i}} = x_j^i h_\theta(x^i),$$

$$h_\theta(x^i) = \frac{1}{1 + e^{\theta x^i}}$$

$$\begin{aligned}
\frac{x_j^i e^{\theta x^i}}{1 + e^{\theta x^i}} &= \frac{x_j^i}{e^{-\theta x^i} * (1 + e^{\theta x^i})} \\
&= \frac{x_j^i}{e^{-\theta x^i} + e^{-\theta x^i + \theta x^i}} = \frac{x_j^i}{e^{-\theta x^i} + e^0} \\
&= \frac{x_j^i}{e^{-\theta x^i} + 1} = \frac{x_j^i}{1 + e^{-\theta x^i}} \\
&= x_j^i * h_\theta(x^i)
\end{aligned}$$

Partial derivation of cost function

$$-\frac{1}{m}\sum_{i=1}^{m}\left[y_i\theta x^i - \log(1+e^{\theta x^i})\right]$$

$$\frac{\partial}{\partial\theta_j}y_i\theta x^i = y_i x_j^i \qquad \frac{\partial}{\partial\theta_j}\log(1+e^{\theta x^i}) = \frac{x_j^i e^{\theta x^i}}{1+e^{\theta x^i}} = x_j^i h_\theta(x^i),$$

$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i)-y^i)x_j^i$$

Weight update

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_j^i$$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

모든 θ_j 동시에 업데이트

$$:= \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_j^i$$

Human knowledge belongs to the world.

Numpy implementation

Logistic Regression Classifier

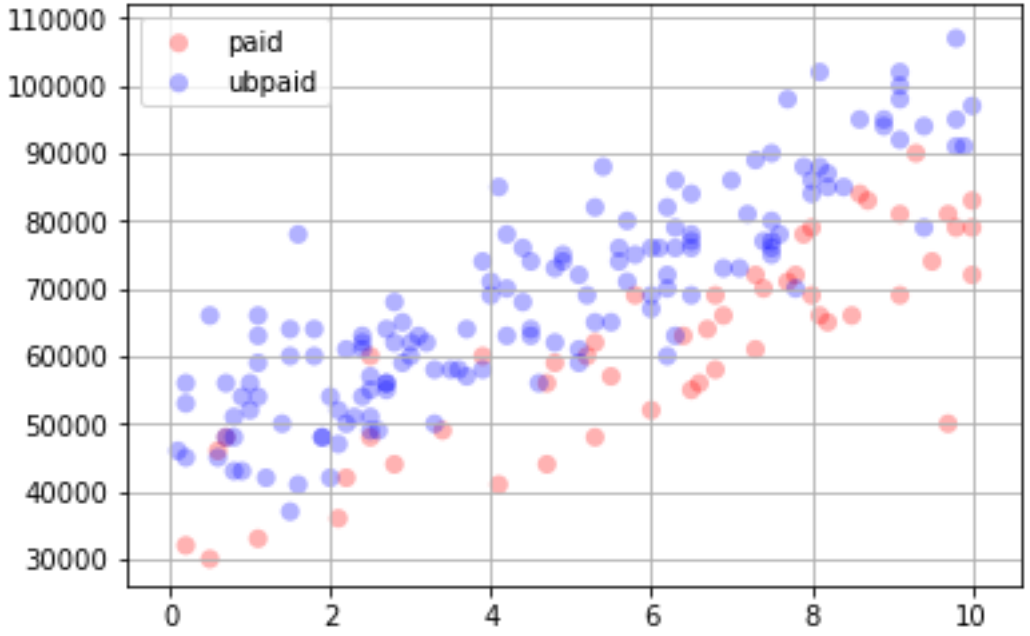
Director of TEAMLAB
Sungchul Choi

Logistic Regression을 Numpy로 구현하기

Data

데이터 과학자로서의 경력과 소득, 유료계정 전환여부

From Data science from scratch



Data

```
x_data = data[:, :2]
x_data = np.insert(x_data, 0 , 1, axis=1)

y_data = data[:, -1]
y_data = y_data.reshape(y_data.shape[0], 1)

x_data.shape, y_data.shape
```

```
((200, 3), (200, 1))
```

Matrix Size

```
x_data[:3]
```

```
array([[ 1.        ,  0.00861027,  0.04956583],
       [ 1.        ,  0.02337075,  0.04956583],
       [ 1.        ,  0.03075098,  0.06195729]])
```

```
y_data[:3]
```

```
array([[ 1.],
       [ 0.],
       [ 1.]])
```

Data normalize

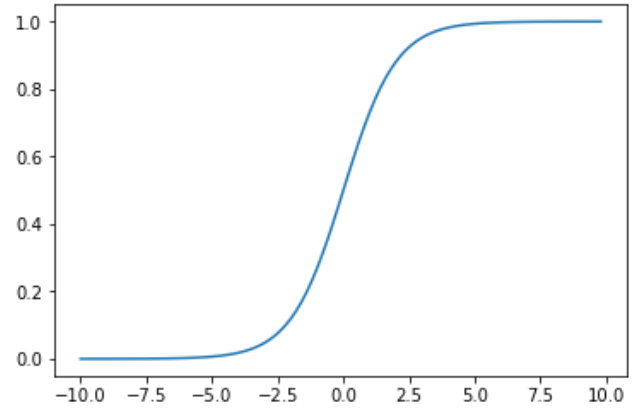
데이터 스케일링

```
from sklearn.preprocessing import normalize

data[:, :2] = normalize(data[:, :2], axis=0)
data[:5]
```

```
array([[ 0.00861027,  0.04956583,  1.        ],
       [ 0.02337075,  0.04956583,  0.        ],
       [ 0.03075098,  0.06195729,  1.        ],
       [ 0.05166165,  0.06505516,  0.        ],
       [ 0.07380236,  0.07847924,  0.        ]])
```

Sigmoid function

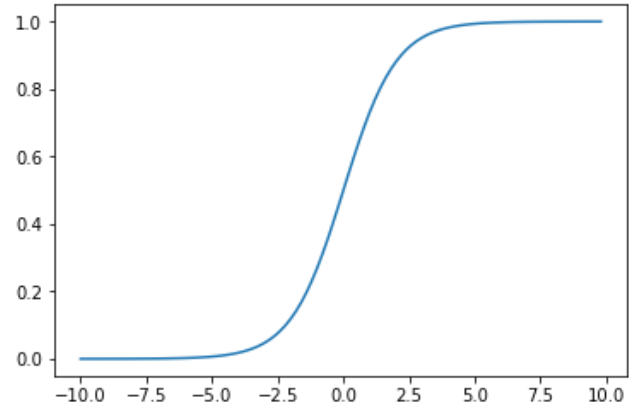
$$h_\theta(x) = g(z) = \frac{1}{1 + e^{-z}}$$




```
def sigmoid(z):
    return 1 / (1 + np.exp(z))
```

Hypothesis function

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$$




```
def hypothesis_function(x, theta):
    z = (np.dot(-x,theta))
    return sigmoid(z)
```

Cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}\left(h_\theta(x^{(i)}), y^{(i)}\right)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

```
def compute_cost(x, y, theta):
    m = y.shape[0]
    J = (-1.0 / m )* (
        y.T.dot(np.log(hypothesis_function(x,theta))) + \
        (1-y).T.dot(np.log(1- hypothesis_function(x,theta))))

    return (-1.0 /m )* (y * np.log(hypothesis_function(x,theta)) + (1-y) \
                        * np.log(1- hypothesis_function(x,theta))).sum()
```

Weight update

```
def minimize_gradient(x, y, theta, iterations=100000, alpha=0.01):
    m = y.size
    cost_history = []
    theta_history = []

    for _ in range(iterations):
        original_theta = theta
        for i in range(theta.size):
            partial_marginal = x[:, i].reshape(x.shape[0], 1)
            delta = hypothesis_function(x, original_theta) - y
            grad_i = delta.T.dot(partial_marginal)

            theta[i] = theta[i] - (alpha  * grad_i)

        if (_ % 10000) == 0:
            theta_history.append(theta)
            cost_history.append(compute_cost(x, y, theta))

    return theta, np.array(cost_history), np.array(theta_history)
```

$$\theta_j := \theta_j - \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_j^i$$

x_j^i

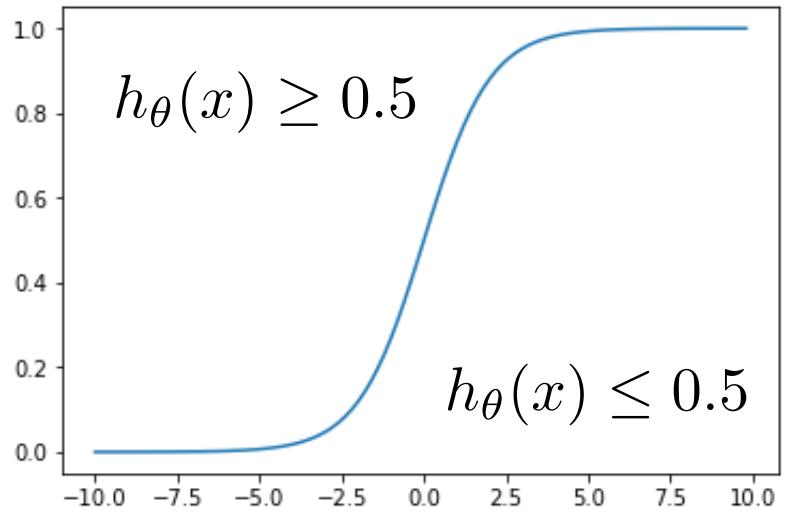
$h_\theta(x^i) - y^i$

Predict

```
theta_initial = np.ones([3,1])
theta, cost_history, theta_history = minimize_gradient(
    x_data, y_data,theta_initial, 1000000, 0.01)
```

```
sum((sigmoid(-x_data.dot(theta)) > 0.5) == y_data) / y_data.shape[0]
```

```
array([ 0.895])
```



Human knowledge belongs to the world.

LR for Scikit-Learn

Logistic Regression Classifier

Director of TEAMLAB
Sungchul Choi

sklearn.linear_model.LogisticRegression

class `sklearn.linear_model.` **`LogisticRegression`** (*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1*) [source]

penalty : str, 'l1' or 'l2', default: 'l2'

dual : bool, default: False

solver : {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'},

multi_class : str, {'ovr', 'multinomial'}, default: 'ovr'

http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

solver : {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'},

default: 'liblinear' Algorithm to use in the optimization problem.

- **For small datasets, 'liblinear' is a good choice, whereas 'sag' and** 'saga' are faster for large ones.
- **For multiclass problems, only 'newton-cg', 'sag', 'saga' and 'lbfgs'** handle multinomial loss; 'liblinear' is limited to one-versus-rest schemes.
- **'newton-cg', 'lbfgs' and 'sag' only handle L2 penalty, whereas** 'liblinear' and 'saga' handle L1 penalty.

https://arxiv.org/pdf/1407.0202.pdf

http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Methods

decision_function (X)	Predict confidence scores for samples.
densify ()	Convert coefficient matrix to dense array format.
fit (X, y[, sample_weight])	Fit the model according to the given training data.
get_params ([deep])	Get parameters for this estimator.
predict (X)	Predict class labels for samples in X.
predict_log_proba (X)	Log of probability estimates.
predict_proba (X)	Probability estimates.
score (X, y[, sample_weight])	Returns the mean accuracy on the given test data and labels.
set_params (**params)	Set the parameters of this estimator.
sparsify ()	Convert coefficient matrix to sparse format.

END

감사합니다.

Confusion Matrix

Logistic Regression Classifier

Director of TEAMLAB
Sungchul Choi

분류 문제의 정확도 성능

실제 Class 대비
얼마나 잘 맞혔는가?

Confusion Matrix (혼합 행렬)

- 실제 라벨과 예측 라벨의 일치 개수를 Matrix 형태로 표현하는 기법

		Prediction	
		1	0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

Confusion Matrix (혼합 행렬)

True Positive (TP)

- 실제 결과 참(1)에 대한 예측이 맞음

True – 예측이 맞음

Positive – 참(1) 인 경우

		Prediction	
		1	0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

Confusion Matrix (혼합 행렬)

True Negative (TN)

- 실제 결과 거짓(0)에 대한 예측이 맞음

True – 예측이 맞음

Negative – 거짓(0) 인 경우

		Prediction	
		1	0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

Confusion Matrix (혼합 행렬)

False Positive (FP)

- 실제 결과 참(1)에 대한 예측이 틀림

False – 예측이 틀림

Positive – 참(1) 인 경우

		Prediction	
		1	0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

Confusion Matrix (혼합 행렬)

False Negative (FN)

- 실제 결과 거짓(0)에 대한 예측이 틀림

False – 예측이 틀림

Negative – 거짓(0) 인 경우

		Prediction	
		1	0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

Confusion Matrix (혼합 행렬)

True Positive (TP)

True Negative (TN)

False Positive (FP)

False Negative (FN)

sklearn.metrics.confusion_matrix

sklearn.metrics. **confusion_matrix** (*y_true, y_pred, labels=None, sample_weight=None*) ¶ [source]

```
from sklearn.metrics import confusion_matrix
y_true = [1, 0, 1, 1, 0, 1]
y_pred = [0, 0, 1, 1, 0, 1]
confusion_matrix(y_true, y_pred)
```

```
array([[2, 0],
       [1, 3]])
```

True Class \ Prediction	0	1
0		
1		

```
tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
(tn, fp, fn, tp)
```

```
(2, 0, 1, 3)
```

Metrics for classification performance

- **Accuracy (정확도)** $$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

- **Error Rate (오차율)** $$Errorrate = \frac{FP+FN}{TP+TN+FP+FN} = (1-Accuracy)$$

- **Precision (정밀도)** $$Precision = \frac{TP}{TP+FP} \quad (PPV: \ Positive \ \ Predict \ \ Value)$$

- **Specificity (특이도)** $$Specificity = \frac{TN}{TN+FP} \quad (TNR: \ True \ \ Negative \ \ Rate)$$

- **Sensitivity (민감도)** $$Sensitivity = \frac{TP}{TP+FP} \quad (TPR: \ True \ \ Positive \ \ Rate)$$

END

감사합니다.

Metrics for classification

Logistic Regression Classifier

Director of TEAMLAB
Sungchul Choi

실제 Class 대비
얼마나 잘 맞혔는가?

정확도 (Accuracy, ACC)

- 전체 데이터 대비 정확하게 예측한 개수의 비율

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

$$ACC = 1 - ERR$$

		Prediction: 1	Prediction: 0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

오차율 (Error Rate, ERR)

- 전체 데이터 대비 부정확하게 예측한 개수의 비율

$$ERR = \frac{FP + FN}{TP + TN + FP + FN}$$

$$ERR = 1 - ACC$$

		Prediction	
		1	0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

```
import numpy as np
from sklearn.metrics import accuracy_score
y_pred = np.array([0, 1, 1, 0])
y_true = np.array([0, 1, 0, 0])
```

```
sum(y_true == y_pred) / len(y_true)
```

```
0.75
```

```
accuracy_score(y_true, y_pred)
```

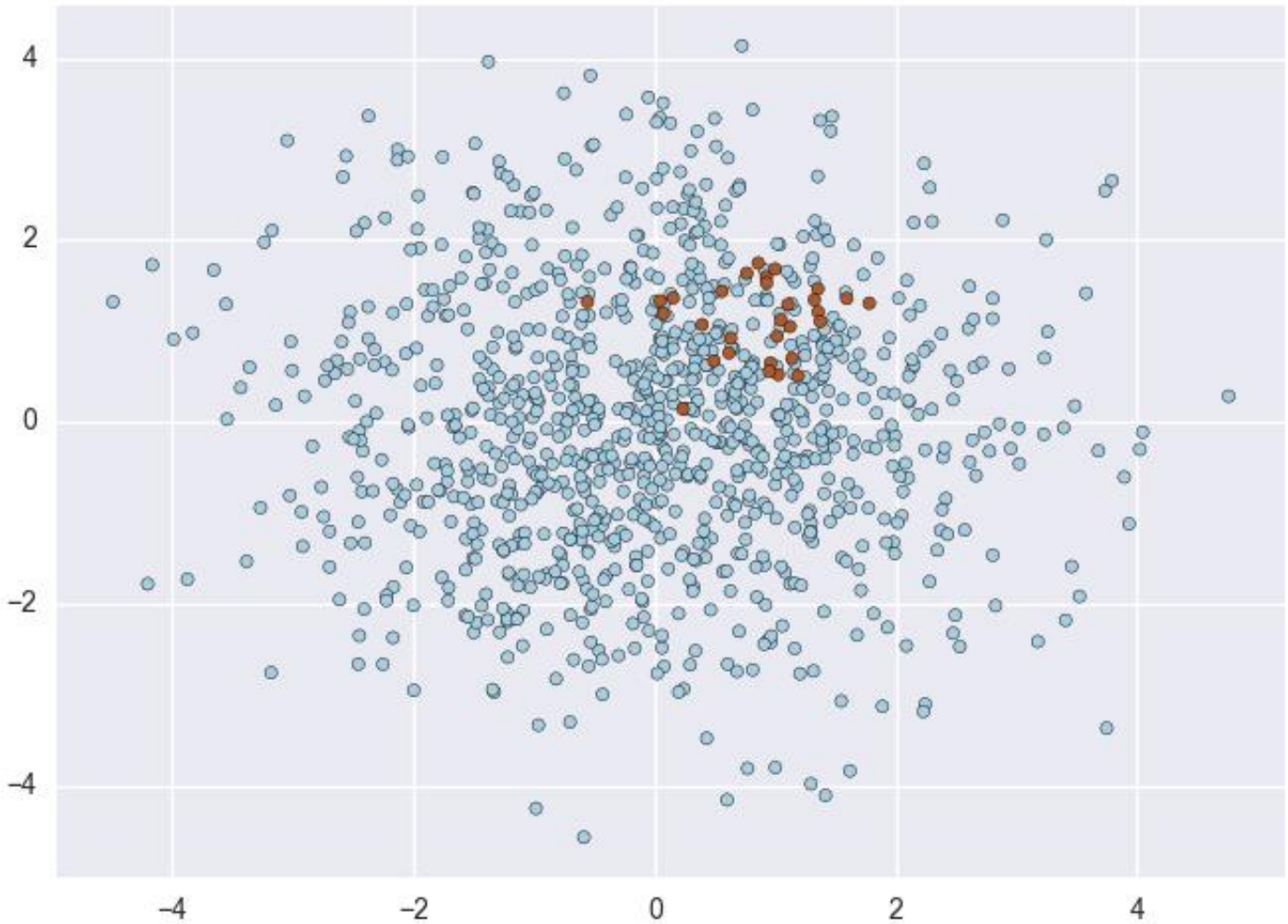
```
0.75
```

불균일한
Dataset의 처리

불균일한 Dataset의 종류

- 14세 이하의 10만명당 암 발병 인원은 14.8, 약 0.015%
- 대학의 학사경고자 평균 비율 3%
- 하버드 입학 지원자의 합격률은 2%
- 이메일 수신자 중 2% 만이 물건을 구매

만약 Accuracy로 구한다면?



https://svds.com/learning-imbalanced-classes/

Metrics for Imbalanced Dataset

Imbalanced dataset

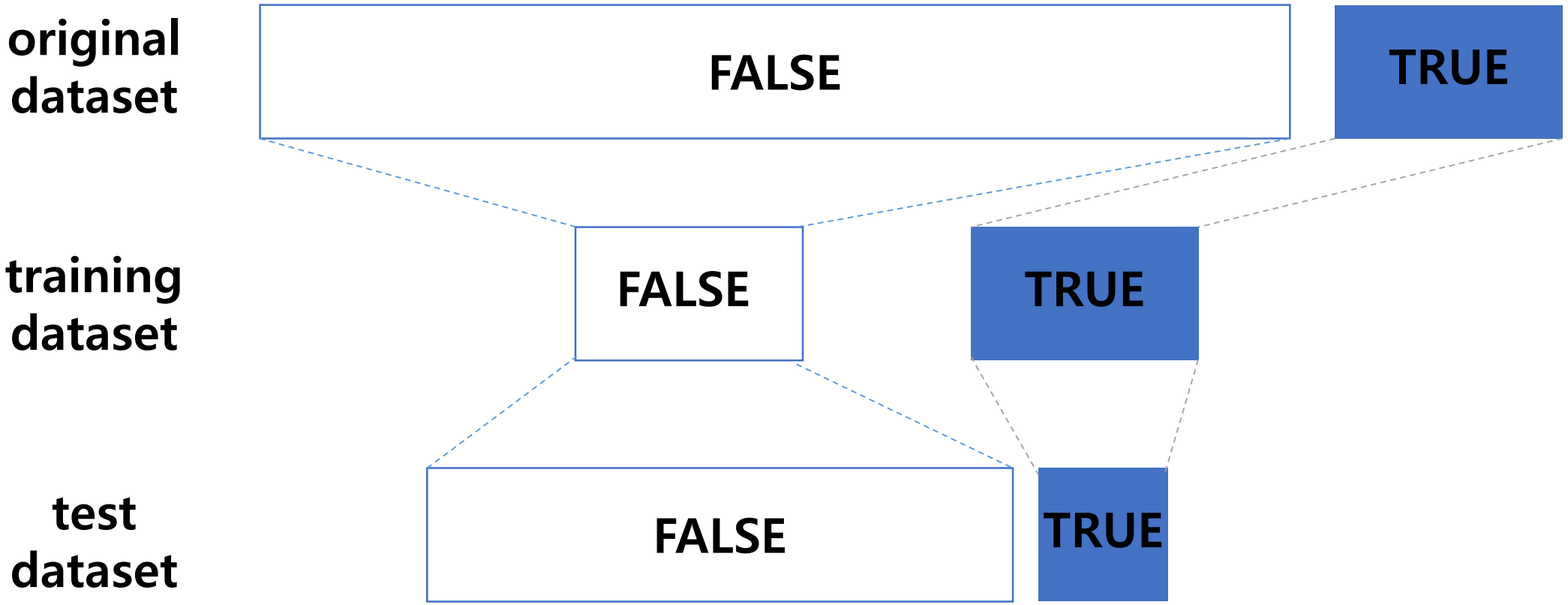
- 유방암 사진 dataset
- 학사 경고자 예측 dataset
- 물건을 구매한 유저의 dataset
- 카드 사기에 관련된 dataset

대부분의 dataset은 imbalanced dataset

How to handle imbalanced dataset

- 적절한 performance metric을 선정 (accuracy X)
  - precision, recall, AUC이 적절
- 적절한 training dataset의 resampling
  - oversampling, under sampling, data augmentation
- Ensemble

Dataset resampling



Dataset resampling

- Imbalanced class가 충분히 많다면

  under sampling → FALSE 데이터를 줄임

- Imbalanced class가 부족하다면

  over sampling → TRUE 데이터를 늘림

imbalanced-learn

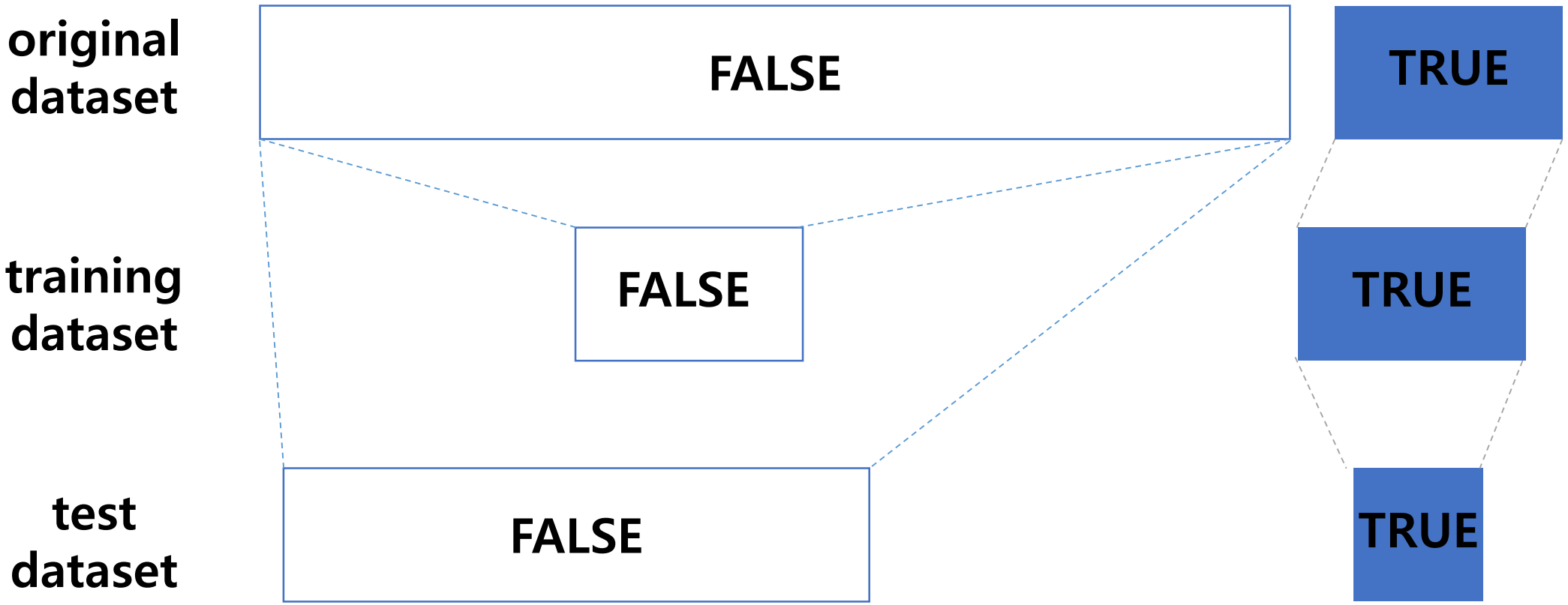
- scikit-learn의 imbalanced dataset 확장 모듈
- under sampling, over sampling, SMOTE 등 제공

https://github.com/scikit-learn-contrib/imbalanced-learn

```
pip install -U imbalanced-learn
```

```
conda install -c conda-forge imbalanced-learn
```

Stratified sampling



Imbalanced dataset handling process

- 전체 dataset에서 test와 dev set을 나눔 (stratified)
- dev set으로 under sampling 또는 oversampling
- 모델의 생성
- Test set으로 모델의 검증

정밀도 (Precision, Positive Predictive Value)

- 긍정이라고 예측한 비율 중 진짜 긍정인 비율
- 긍정이라고 얼마나 잘 예측했는가? 긍정 예측 정밀도?

$$PRECISON(PPV) = \frac{TP}{TP + FP}$$

		Prediction	
		1	0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

```
from sklearn.metrics import precision_score
```

```
y_pred = np.array([0, 1, 1, 0])
y_true = np.array([0, 1, 0, 0])
```

```
sum((y_pred == 1) & (y_pred == y_true)) / sum(y_pred)
```

```
0.5
```

```
precision_score(y_true, y_pred)
```

```
0.5
```

sklearn.metrics.precision_score

```
sklearn.metrics. precision_score (y_true, y_pred, labels=None, pos_label=1, average='binary',
sample_weight=None)                                                                    [source]
```

labels : list, optional
pos_label : str or int, 1 by default
average : string, [None, 'binary' (default), 'micro', 'macro', 'samples', 'weighted']
sample_weight : array-like of shape = [n_samples], optional

average : string, [None, 'binary' (default), 'micro', 'macro', 'samples', 'weighted']

This parameter is required for multiclass/multilabel targets. If `None`, the scores for each class are returned. Otherwise, this determines the type of averaging performed on the data:

`'binary'`:

Only report results for the class specified by `pos_label`. This is applicable only if targets (`y_{true,pred}`) are binary.

`'micro'`:

Calculate metrics globally by counting the total true positives, false negatives and false positives.

`'macro'`:

Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

`'weighted'`:

Calculate metrics for each label, and find their average, weighted by support (the number of true instances for each label). This alters 'macro' to account for label imbalance; it can result in an F-score that is not between precision and recall.

`'samples'`:

Calculate metrics for each instance, and find their average (only meaningful for multilabel classification where this differs from `accuracy_score`).

```
from sklearn.metrics import precision_score
```

```
y_pred = np.array([0, 1, 1, 0])
y_true = np.array([0, 1, 0, 0])
```

```
sum((y_pred == 1) & (y_pred == y_true)) / sum(y_pred)
```

```
0.5
```

```
precision_score(y_true, y_pred)
```

```
0.5
```

```
y_true = [0, 1, 2, 0, 1, 2]
y_pred = [0, 2, 1, 0, 0, 1]
confusion_matrix(y_true, y_pred)
```

```
array([[2, 0, 0],
       [1, 0, 1],
       [0, 2, 0]])
```

전체 평균

'micro':

Calculate metrics globally by counting the total true positives, false negatives and false positives.

'macro': (Label별 값 합)의 평균

Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

```
precision_score(y_true, y_pred, average='macro')
```

```
0.2222222222222221
```

```
precision_score(y_true, y_pred, average=None)
```

```
array([ 0.66666667,  0.        ,  0.        ])
```

```
precision_score(y_true, y_pred, average='micro')
```

```
0.3333333333333331
```

민감도 (Sensitivity, Recall, True Positive Rate)

- 실제 긍정 데이터중 긍정이라고 예측한 비율, 반환율, 재현율
- 얼마나 잘 긍정(예 - 암)이라고 예측하였는가?

$$RECALL(TPR) = \frac{TP}{TP + FN} = \frac{TP}{P}$$

		Prediction: 1	Prediction: 0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

```
from sklearn.metrics import recall_score
```

```
y_pred = np.array([0, 1, 1, 0])
y_true = np.array([0, 1, 0, 0])
```

```
sum((y_true == 1) & (y_pred == y_true)) / sum(y_true)
```

```
1.0
```

$$RECALL(TPR) = \frac{TP}{TP + FN} = \frac{TP}{P}$$

		Prediction: 1	Prediction: 0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

```
recall_score(y_true, y_pred)
```

```
1.0
```

```
y_true = [0, 1, 2, 0, 1, 2]
y_pred = [0, 2, 1, 0, 0, 1]
recall_score(y_true, y_pred, average='macro')
```

```
0.3333333333333331
```

```
recall_score(y_true, y_pred, average='micro')
```

```
0.3333333333333331
```

```
recall_score(y_true, y_pred, average=None)
```

```
array([ 1.,  0.,  0.])
```

특이성 (Specificity, True Negative Rate)

- 부정을 얼마나 잘 부정이라고 인식해는가?
- 전제 부정중 부정을 정확히 찾아낸 비율

$$SPC = \frac{TN}{TN + FP} = \frac{TN}{N}$$

		Prediction	
		1	0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

F1 Score (F-measure, F-score)

- Precision과 Recall의 통합한 측정지표
- Precision과 Recall의 조화평균

$$F_1 = 2\frac{precision * recall}{precision + recall}$$

		Prediction	
		1	0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

```
from sklearn.metrics import f1_score
y_pred = np.array([0, 1, 1, 0])
y_true = np.array([0, 1, 0, 0])
```

```
pre = precision_score(y_true, y_pred)
rec = recall_score(y_true, y_pred)
```

```
2 * (pre * rec) / (pre + rec)
```

```
0.6666666666666663
```

$$F_1 = 2\frac{precision * recall}{precision + recall}$$

```
f1_score(y_true, y_pred)
```

```
0.6666666666666663
```

```
y_true = [0, 1, 2, 0, 1, 2]
y_pred = [0, 2, 1, 0, 0, 1]
f1_score(y_true, y_pred, average='macro')
```

```
0.26666666666666666
```

```
f1_score(y_true, y_pred, average='micro')
```

```
0.3333333333333331
```

```
f1_score(y_true, y_pred, average=None)
```

```
array([ 0.8,  0. ,  0. ])
```

Example

		Prediction		
		1	0	
Actual Class	1	90	210	300
	0	140	9560	9700
		230	9770	10000

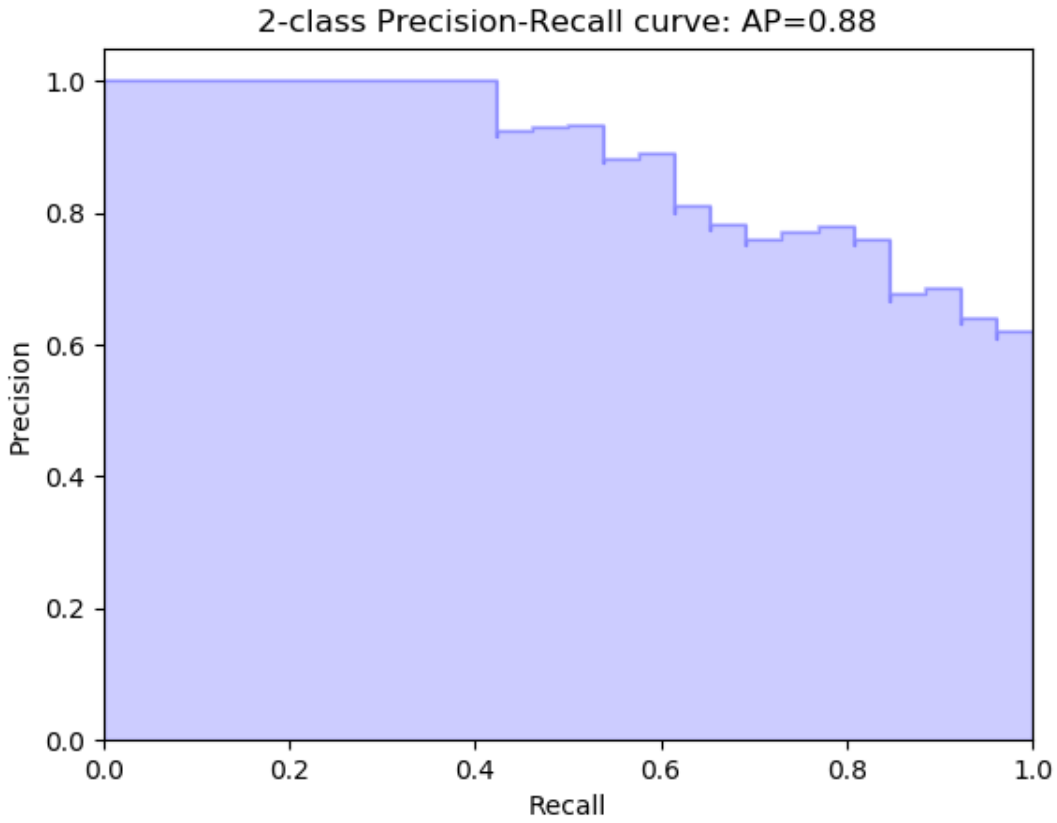
$$PRECISON(PPV) = \frac{TP}{TP + FP}$$

$$RECALL(TPR) = \frac{TP}{TP + FN} = \frac{TP}{P}$$

$$SPC = \frac{TN}{TN + FP} = \frac{TN}{N}$$

Precision - Recall Curve

- 예측 확률 Threshold를 변화시켜 Precision/Recall 측정
- 시각화 할 때 유용하게 사용 가능



```
import numpy as np
from sklearn.metrics import precision_recall_curve
y_true = np.array([0, 0, 1, 1])
y_scores = np.array([0.1, 0.4, 0.35, 0.8])
precision, recall, thresholds = precision_recall_curve(
    y_true, y_scores)
```

```
precision
```

```
array([ 0.66666667,  0.5       ,  1.        ,  1.        ])
```

```
recall
```

```
array([ 1. ,  0.5,  0.5,  0. ])
```

```
thresholds
```

```
array([ 0.35,  0.4 ,  0.8 ])
```

Precision - Classification Report

- Classification 문제에서 한번에 Precision, Recall, F1 결과 출력

	precision	recall	f1-score	support
class 0	0.50	1.00	0.67	1
class 1	0.00	0.00	0.00	1
class 2	1.00	0.67	0.80	3
avg / total	0.70	0.60	0.61	5

```
from sklearn.metrics import classification_report
y_true = [0, 1, 2, 2, 2]
y_pred = [0, 0, 2, 2, 1]
target_names = ['class 0', 'class 1', 'class 2']
```

```
print(classification_report(y_true, y_pred, target_names=target_names))
```

```
             precision    recall  f1-score   support

    class 0       0.50      1.00      0.67         1
    class 1       0.00      0.00      0.00         1
    class 2       1.00      0.67      0.80         3

avg / total       0.70      0.60      0.61         5
```

Confusion matrix:

```
[[87  0  0  0  1  0  0  0  0  0]
 [ 0 88  1  0  0  0  0  0  1  1]
 [ 0  0 85  1  0  0  0  0  0  0]
 [ 0  0  0 79  0  3  0  4  5  0]
 [ 0  0  0  0 88  0  0  0  0  4]
 [ 0  0  0  0  0 88  1  0  0  2]
 [ 0  1  0  0  0  0 90  0  0  0]
 [ 0  0  0  0  0  1  0 88  0  0]
 [ 0  0  0  0  0  0  0  0
 [ 0  0  0  1  0  1  0  0
```

	precision	recall	f1-score	suppo
0	1.00	0.99	0.99	
1	0.99	0.97	0.98	
2	0.99	0.99	0.99	
3	0.98	0.87	0.92	
4	0.99	0.96	0.97	
5	0.95	0.97	0.96	
6	0.99	0.99	0.99	
7	0.96	0.99	0.97	
8	0.94	1.00	0.97	
9	0.93	0.98	0.95	
g / total	0.97	0.97	0.97	8

Human knowledge belongs to the world.

ROC Curve

Logistic Regression Classifier

Director of TEAMLAB
Sungchul Choi

민감도-특이도 Trade-off?

Trade-Off 관계가 지표들 중 무엇을 선택해야 하나

ROC Curve

Receiver Operation Characteristics

ROC curve (수신자 운영 특성?)

- Receiver Operating Characteristics

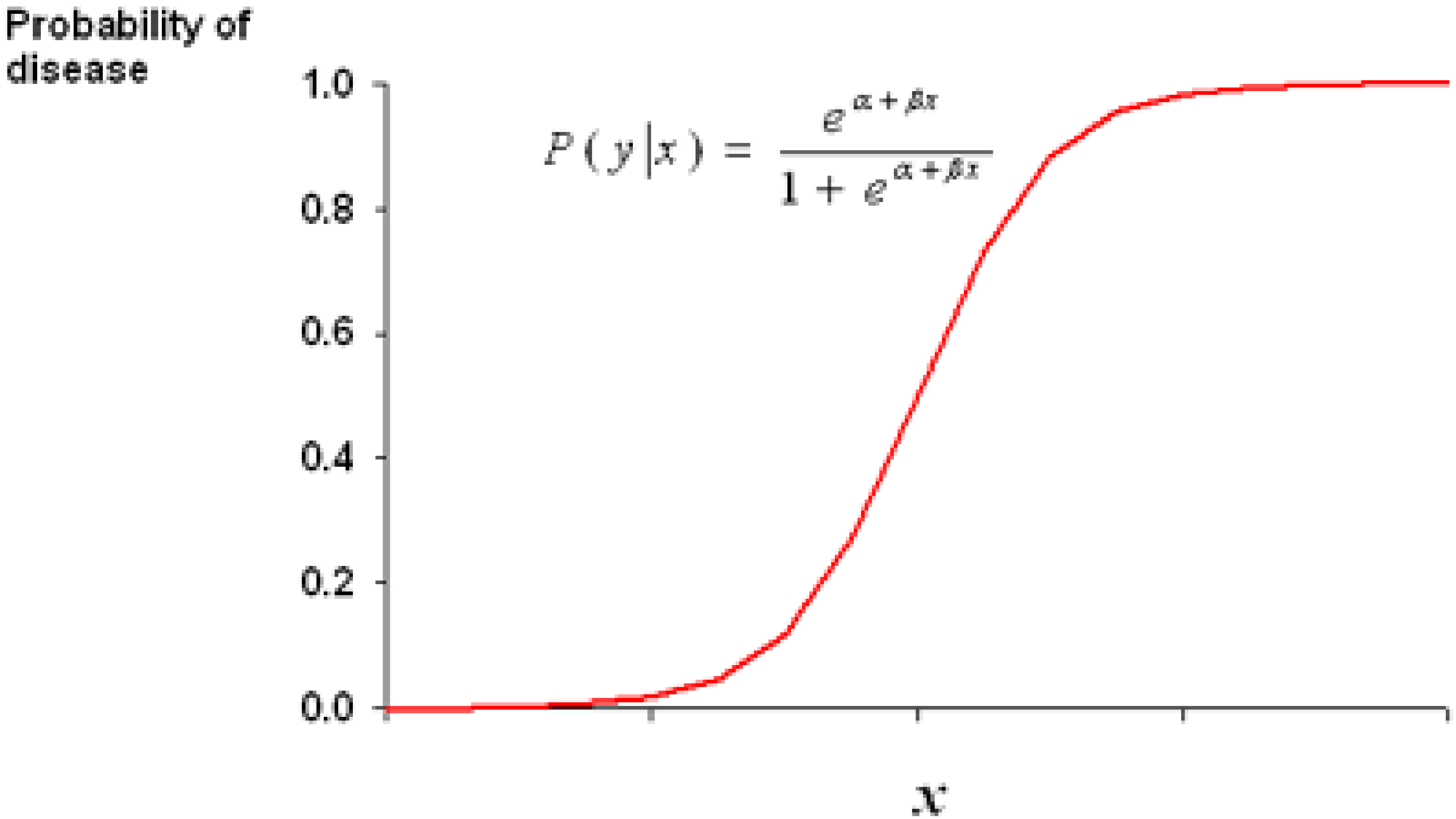
- 2차 세계 대전 중 레이더 이미지를 분석하는 신호감지 이론에서 시작

- Basic Principles of ROC Analysis (Charles Metz, 1978)

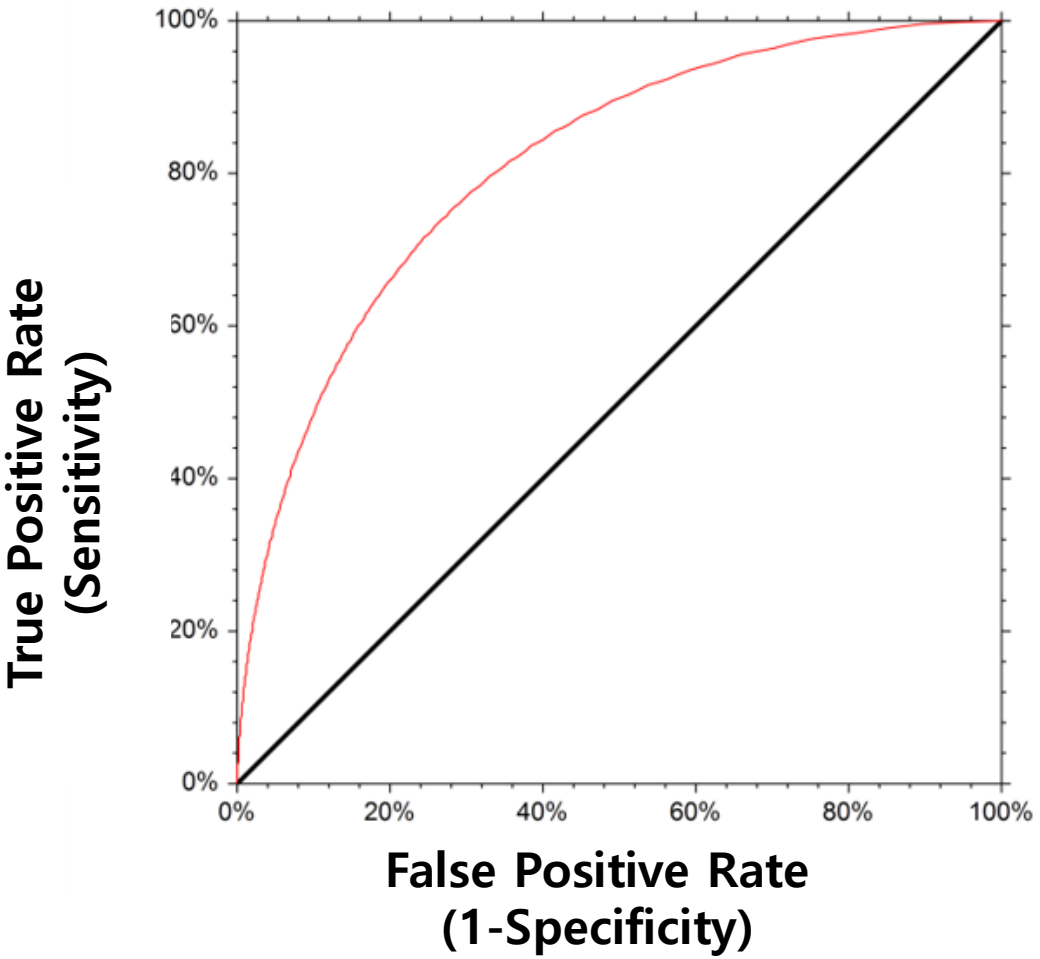
- 분류기의 경계치(Threshold)를 조정, 민감도-특이도간 비율을 도식화

- LogReg, NB과 같은 Class의 예측 확률이 나오는 모델에 사용가능
  → Decision Tree 같은 경우 값 산출을 위한 수정이 필요

Prediction Probability



Prediction Probability



Data	Class	Positive Prediction (Threshold)
1	P	0.9
2	P	0.8
3	N	0.7
4	P	0.6
5	P	0.55
6	N	0.54
7	N	0.53
8	N	0.51
9	P	0.5
10	N	0.4

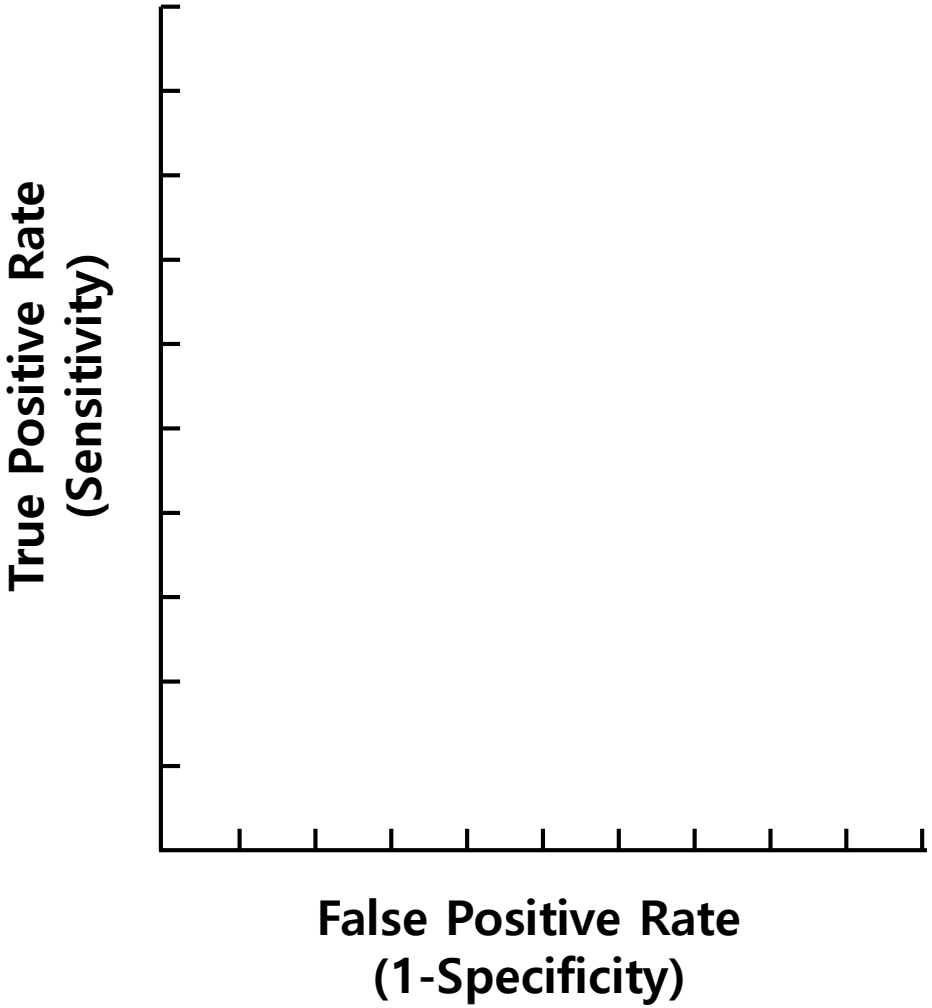
$$Sensitivity(TPR) = \frac{TP}{TP + FN} = \frac{TP}{P}$$

$$FPR = 1 - Specificity(TNR)$$
$$= 1 - \frac{TN}{TN+FP} = 1 - \frac{TN}{N}$$

		Prediction	
		1	0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

From: https://www.ncss.com/software/ncss/roc-curves-ncss/

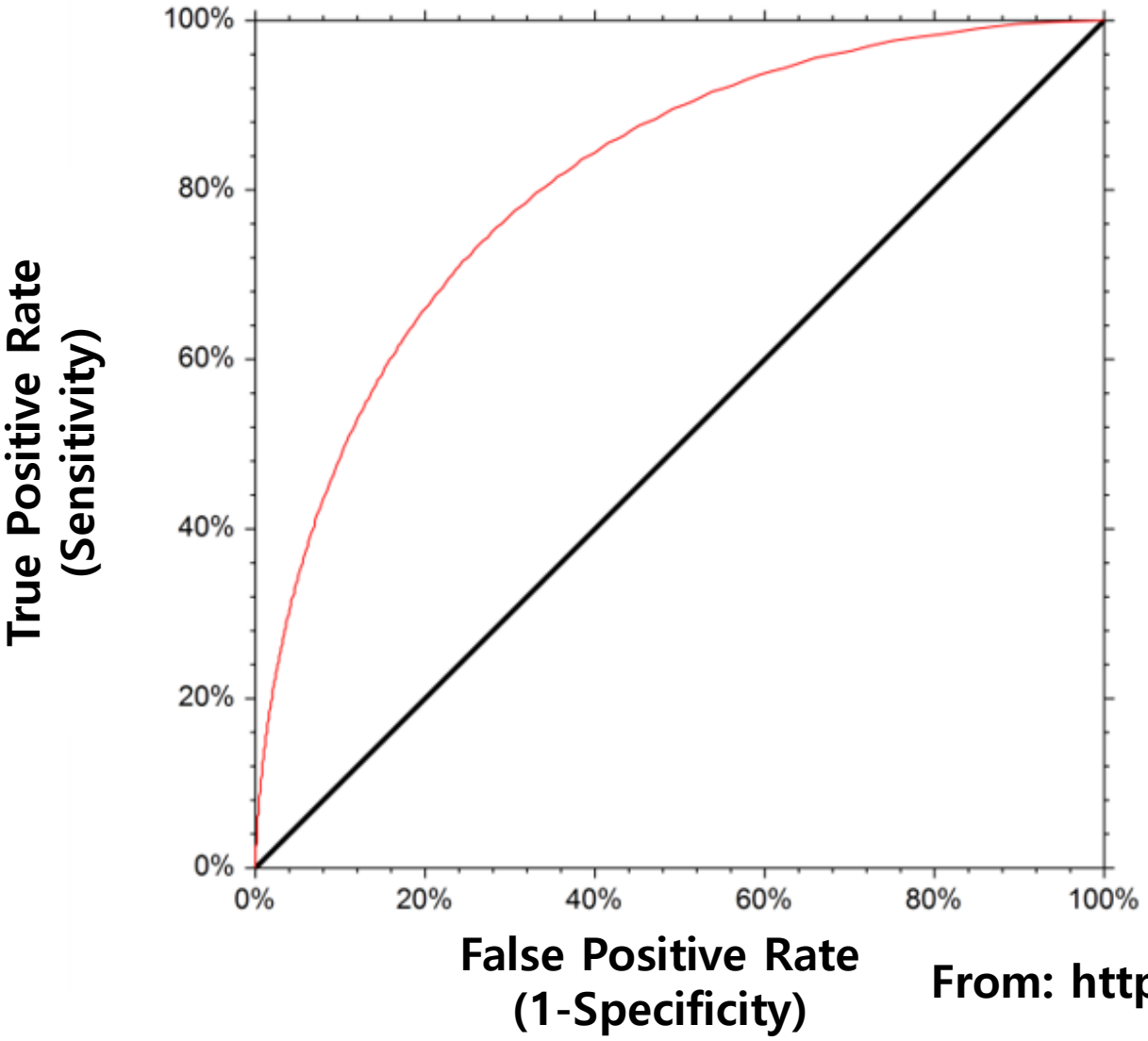
Prediction Probability



		Prediction	
		1	0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

Data	Class	Positive Prediction (Threshold)	TRUE Positive	FALSE Positive	TRUE Negative	FALSE Negative	TPR (Sensitivity)	FPR (1-Specificity)
1	P	0.9	1	0	5	4	0.2	0
2	P	0.8	2	0	5	3	0.4	0
3	N	0.7	2	1	4	3	0.4	0.2
4	P	0.6	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.5	5	4	0	1	1	1
10	N	0.4	5	5	0	0	1	1

Prediction Probability



$$Sensitivity(TPR) = \frac{TP}{TP + FN} = \frac{TP}{P}$$

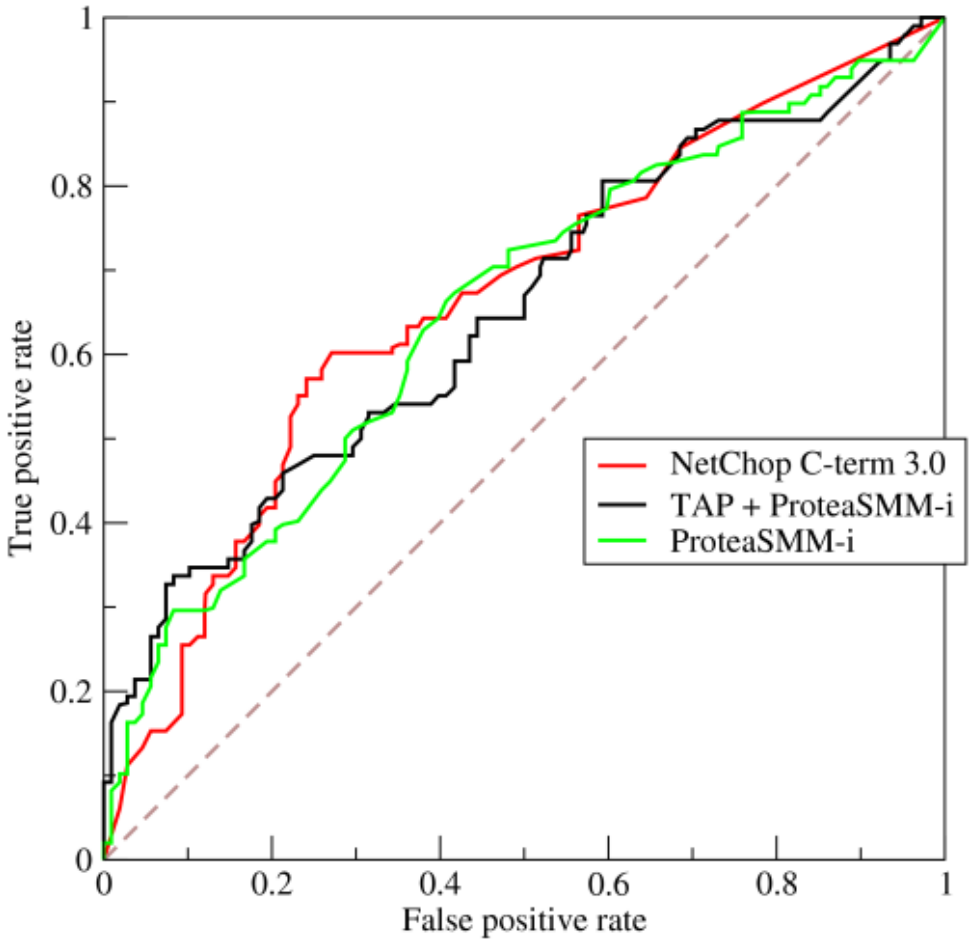
$$\mathrm{FPR} = 1 - Specificity(TNR) = 1 - \frac{TN}{TN + FP} = 1 - \frac{TN}{N}$$

		Prediction	
		1	0
Actual Class	1	True Positive	False Negative
	0	False Positive	True Negative

From: https://www.ncss.com/software/ncss/roc-curves-ncss/

AUC, Area Under Curve

- ROC curve의 하단의 넓이를 의미함
- ROC curve를 단순한 Single Metric (단 하나의 숫자)로 표현할 수 있음
- 대각선을 중심으로 상단에 붙어 있을 수록 높은 성능을 표시함



from Wikipedia(https://goo.gl/itMyAR)

See - http://www.dataschool.io/roc-curves-and-auc-explained/

```
y = np.array([1, 1, 2, 2])
scores = np.array([0.1, 0.4, 0.35, 0.8])
fpr, tpr, thresholds = metrics.roc_curve(y, scores, pos_label=2)
```

```
fpr
```

```
array([ 0. ,  0.5,  0.5,  1. ])
```

```
tpr
```

```
array([ 0.5,  0.5,  1. ,  1. ])
```

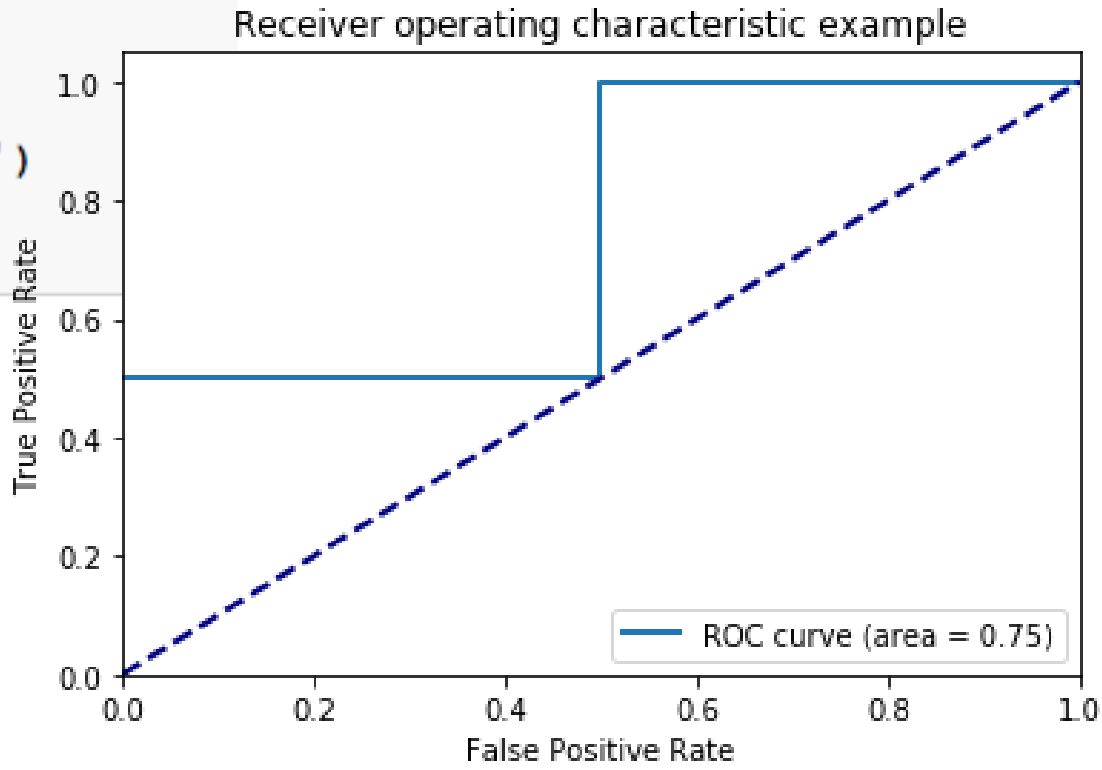
```
thresholds
```

```
array([ 0.8 ,  0.4 ,  0.35,  0.1 ])
```

```
roc_auc = metrics.auc(fpr, tpr)
roc_auc
```

```
0.75
```

```
plt.figure()
lw = 2
plt.plot(fpr, tpr,
         lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



END

감사합니다.

Overview

Multiclass Classification

Director of TEAMLAB
Sungchul Choi

3개 이상의 분류를 하는 방법
Multiclass Classification

개념정리

Multiclass classification

- 두 개 이상의 클래스를 가진 분류 작업
- 오렌지, 사과 또는 배
- 중복 선택 불가 → [1 0 0] 가능, [1 1 0] 불가

Multilabel classification

- 상호 배타적이지 않은 속성을 예측
- 중복 선택 가능한 분류 → [1 1 0] 가능
- 신문기사 분류: 야구선수-연예인 결혼 → 스포츠/연예

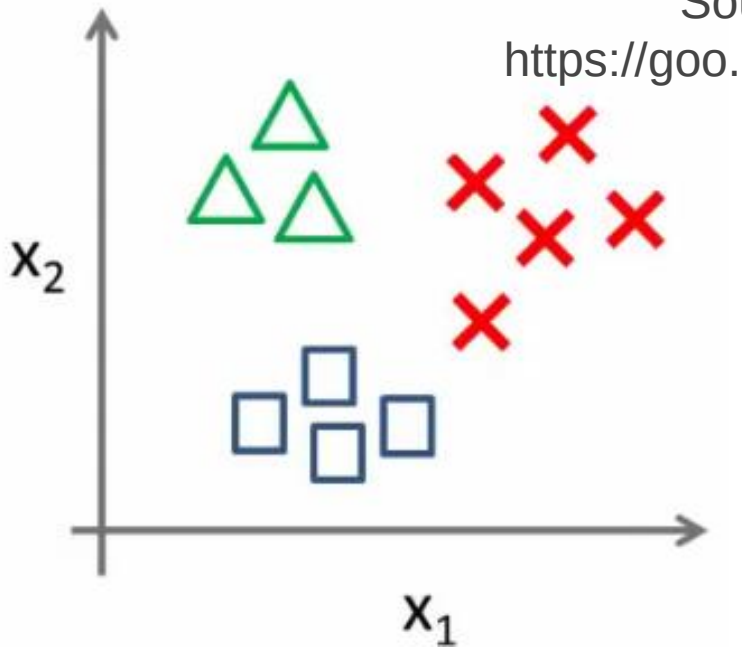
Source from - https://goo.gl/FStnzy

Approach

One-vs-All

- m개의 class가 존재할 때, 클래스마다 classifier 생성

Source from - https://goo.gl/ZQaK65



One-vs-One

- 2 쌍씩의 Class마다 Classifier를 생성, 최종 선택시 Classifier 선택의 투표를 통해 결정
- 총 m(m-1) / 2 개 만큼의 Classifier 생성, 정확도 Up

Human knowledge belongs to the world.

Softmax fucntion

Multiclass Classification

Director of TEAMLAB
Sungchul Choi

Sigmoid function for multiclass

One vs All Approach

- m개의 classifier 함수 $h_m(x;\theta)$ 생성
- $h_m(x;\theta)$의 확률 값중 가장 높은 값을 가진 m 을 선택
- 각 $h_m(x;\theta)$의 확률 합이 1 이상이라는 문제점이 생김

Softmax function for multiclass

모든 class의 확률을 1로 Generalize 함

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad \text{for } j = 1, 2, 3, \ldots, K$$

$$\sum_{j=1}^{K} \sigma(z)_j = \sum_{j=1}^{K} P_j = 1$$

$$\begin{aligned} where: \quad \sigma(z)_j &= \text{probability of class } j \\ K &= \text{last classification class} \end{aligned}$$

Softmax Function

Class가 두개일 때

$$\frac{P_j}{1-P_j} \Rightarrow logit(P_j) = \log_e\left(\frac{P_j}{1-P_j}\right) = z = \theta^T \mathbf{x}$$

Class가 K개 일 때

$$\frac{P_j}{P_K} \Rightarrow logit(P_j) = \log_e\left(\frac{P_j}{P_K}\right) = z_j = \mathbf{x}^T \theta_j$$

Softmax Function

$$\log_e\left(\frac{P_j}{P_K}\right) = z_j \Rightarrow \frac{P_j}{P_K} = e^{z_j} \Rightarrow$$

$$\sum_{j=1}^{K}\frac{P_j}{P_K} = \sum_{j=1}^{K} e^{z_j} \Rightarrow \frac{1}{P_K}\sum_{j=1}^{K} P_j = \sum_{j=1}^{K} e^{z_j} \Rightarrow$$

$$\frac{1}{P_K} * 1 = \sum_{j=1}^{K} e^{z_j} \Rightarrow \qquad (\because \sum_{j=1}^{K} P_j = 1)$$

$$P_K = \frac{1}{\sum_{j=1}^{K} e^{z_j}}$$

http://willwolf.io/2017/05/18/minimizing_the_negative_log_likelihood_in_english/

Softmax Function

$$\frac{P_j}{P_K} = e^{z_j} \Rightarrow \frac{P_j}{\frac{1}{\sum_{j=1}^{K} e^{z_j}}} = e^{z_j} \Rightarrow \qquad \because P_K = \frac{1}{\sum_{j=1}^{K} e^{z_j}}$$

$$P_j = \frac{e^{z_j}}{\sum_{j=1}^{K} e^{z_j}}$$

Softmax Fucntion

$$P_j = \frac{e^{z_j}}{\sum_{j=1}^{K} e^{z_j}} = \frac{e^{z_j}}{\sum_{j=1}^{K} e^{\mathbf{x}^T \theta_j}} \qquad \because z = \mathbf{x}^T \theta_j$$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_j \end{bmatrix} = \begin{bmatrix} w_{10} & w_{11} & \dots & w_{1i} \\ w_{20} & w_{21} & \dots & w_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ w_{j0} & w_{j1} & \dots & w_{ji} \end{bmatrix}$$

Human knowledge belongs to the world.

Softmax regression

Multiclass Classifier

Director of TEAMLAB
Sungchul Choi

$$P_j = \frac{e^{\mathbf{x}^T \theta_j}}{\sum_{j=1}^{K} e^{\mathbf{x}^T \theta_j}}$$

Softmax function 학습

$$P_j = \frac{e^{\mathbf{x}^T \theta_j}}{\sum_{j=1}^{K} e^{\mathbf{x}^T \theta_j}}$$

결국은 θ의 학습

$$P_j = \frac{e^{\mathbf{x}^T \theta_j}}{\sum_{j=1}^{K} e^{\mathbf{x}^T \theta_j}}$$

클래스마다 θ 가 존재함

$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \vdots \\ P(y=K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)} \begin{bmatrix} \exp(\theta^{(1)\top} x) \\ \exp(\theta^{(2)\top} x) \\ \vdots \\ \exp(\theta^{(K)\top} x) \end{bmatrix}$$

확률의 최대화!
확률을 최대화할 θ 를 찾자

Maximum Likelihood Estimation

$$\arg\max_{\theta} \prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta)$$

$$p^y(1-p)^{1-y} \Rightarrow p_1^{v_1} p_2^{v_2} \dots p_j^{v_j} \qquad \text{where } v_j \begin{cases} 1 & \text{if } y = v_j \\ 0 & \text{if } y \neq v_j \end{cases}$$

$$\arg\max_{\theta} \prod_{i=1}^{n} p(x^{(i)};\theta^{(i)})^{y^{(i)}} (1 - p(x^{(i)};\theta^{(i)}))^{1-y^{(i)}}$$

Reference from -

Negative Log-Likelihood

$$L = \prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta) = \prod_{i=1}^{m}\prod_{j=1}^{K} {p_j^{(i)}}^{v_{ij}} \Rightarrow$$

$$-\log L = -\log \prod_{i=1}^{m}\prod_{j=1}^{K} {p_j^{(i)}}^{v_{ij}} = -\sum_{i=1}^{m}\sum_{j=1}^{K} v_{ij} \log p_j^{(i)}$$

$$\text{where } v_{ij} \begin{cases} 1 & \text{if } y^{(i)} \text{ is label } j \\ 0 & \text{if } y^{(i)} \text{ is NOT label } j \end{cases} \qquad p_j^{(i)} = \frac{e^{\mathbf{x}^{(i)T}\theta_j}}{\sum_{j=1}^{K} e^{\mathbf{x}^{(i)T}\theta_j}}$$

Reference from -

Minimize Cost Function

$$l = -\sum_{i=1}^{m}\sum_{j=1}^{K} v_{ij} \log p_j^{(i)} \qquad \text{where } p_j^{(i)} = \frac{e^{z_j^{(i)}}}{\sum_{j=1}^{K} e^{z_j^{(i)}}}$$

$\frac{\partial l}{\partial z}$ ➔ **1) derivate of Negative-Log Likelihood**

2) derivate of Softmax Function

Reference from -

Minimize Cost Function

$$\frac{\partial l}{\partial z_i} = -\sum_{j=1}^{K} v_{ij} \frac{\partial \log p_j}{\partial z_i} = -\sum_{j=1}^{K} v_{ij} \frac{1}{p_j} \frac{\partial p_j}{\partial z_i} \qquad (\because \frac{d \log_e f(x)}{dx} = \frac{1}{f(x)} \frac{df(x)}{dx})$$

$$\frac{\partial p_j}{\partial z_c} = \frac{\partial \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}}{\partial z_c} \Rightarrow f'_j = \frac{g'_j h_j - h'_j g_j}{[h_j]^2} \qquad (\text{where } g_j = e^{z_j}, h_j = \sum_{k=1}^{K} e^{z_k})$$

Minimize Cost Function

if $c = j$

$$\frac{dy}{dx} = \frac{d(e^{x^2})}{x} = e^{x^2}\left(\frac{dx^2}{dx}\right) = 2x(e^{x^2})$$

$$\frac{\partial p_j}{\partial z_c} = \frac{\partial \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}}{\partial z_c} = \frac{e^{z_j} h_j - e^{z_c} e^{z_j}}{[h_j]^2} = \frac{e^{z_j}}{h_j}\frac{h_j - e^{z_c}}{h_j} = p_j(1 - p_j)$$

$$\because p_j = \frac{e^{z_j}}{\sum_{j=1}^{K} e^{z_j}}$$

if $c \neq j$

$$\frac{\partial p_j}{\partial z_c} = \frac{\partial \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}}{\partial z_c} = \frac{0 - e^{z_c} e^{z_j}}{[h_j]^2} = -\frac{e^{z_j}}{h_j}\frac{e^{z_c}}{h_j} = -p_j p_c$$

Minimize Cost Function

$$\frac{\partial l}{\partial z_i} = -\sum_{j=1}^{K} v_{ij} \frac{1}{p_j} \frac{\partial p_j}{\partial z_i} = -\frac{v_i}{p_i} \frac{\partial p_i}{\partial z_i} - \sum_{j \neq i}^{K} \frac{v_j}{p_j} \frac{\partial p_j}{\partial z_i}$$

$$= -\frac{v_i}{p_i} p_i(1 - p_i) - \sum_{j \neq i}^{K} \frac{v_j}{p_j}(-p_j p_i)$$

$$= -v_i + v_i p_i + \sum_{j \neq i}^{K} v_j p_i = -v_i + \sum_{j=1}^{K} v_j p_i$$

$$p_j = \frac{e^{z_j}}{\sum_{j=1}^{K} e^{z_j}}$$

$$= p_i - v_i \qquad (\because \sum_{j=1}^{K} v_j = 1)$$

Update weights

$$
\begin{aligned}
w_{kj} &= w_{kj} - \alpha \frac{\partial l}{\partial w_{kj}} \\
&= w_{kj} - \alpha \frac{\partial l}{\partial z_k} \frac{\partial z_k}{\partial w_{kj}} \\
&= w_{kj} - \alpha (p_k - v_k) x_j \\
&= w_{kj} + \alpha (v_k - p_k) x_j \\
\Rightarrow w_{kj} &= w_{kj} + \alpha \sum_{i=1}^{n} (v_k^{(i)} - p_k^{(i)}) x_j^{(i)}
\end{aligned}
$$

$$
w = \begin{bmatrix}
w_{10} & w_{11} & \dots & w_{1j} \\
w_{20} & w_{21} & \dots & w_{2j} \\
\vdots & \vdots & \ddots & \vdots \\
w_{k0} & w_{k1} & \dots & w_{kj}
\end{bmatrix}
$$

where $\begin{cases} k & \text{index of class} \\ j & \text{index of feature} \end{cases}$

Cross-Entropy Loss Function

Loss Function을 Cross-Entropy Function이라고 부름

- Entropy는 목적 달성을 위한 경우의 수 정량적으로 표현하는 수치 ➔ 작을 수록 경우의 수가 적음

$$H(p) = -\sum_{i=1}^{n} p_i \log p_i$$

- There are n distinct events.
- Each event i has probability p_i.

Human knowledge belongs to the world.

Softmax with Numpy

Multiclass Classifier

Director of TEAMLAB
Sungchul Choi

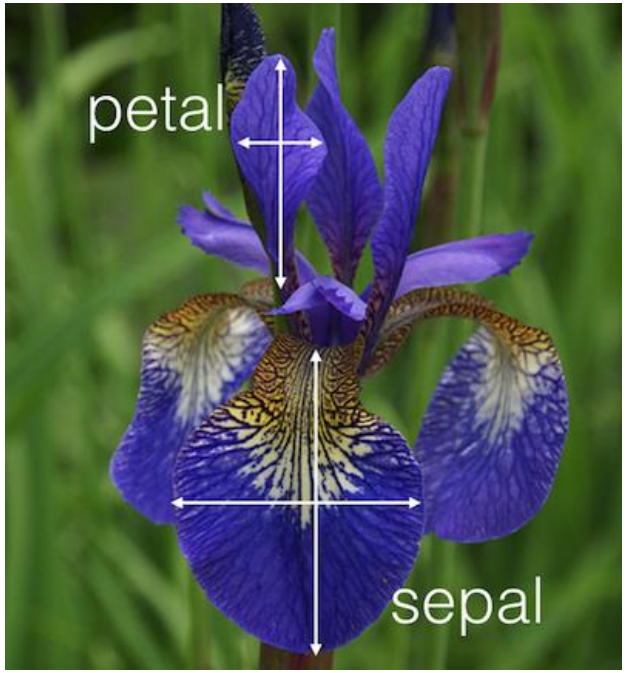
Iris dataset

- 대표적인 Multiclass dataset
- 붓꽃 정보를 모아서 3가지 붓꽃 종류중 어디에 해당하는 지를 찾는 문제
- 일반적으로 통계학자 Fisher가 제안한 데이터 셋으로 Fisher’s Iris dataset라고도 부름

https://thebook.io/006723/ch04/01/

Iris dataset

컬럼명	의미	데이터 타입
Species	붓꽃의 종. setosa, versicolor, virginica 세 가지 값 중 하나	Factor
Sepal.Width	꽃받침의 너비	Number
Sepal.Length	꽃받침의 길이	Number
Petal.Width	꽃잎의 너비	Number
Petal.Length	꽃잎의 길이	Number



https://thebook.io/006723/ch04/01/

Iris dataset

IRIS dataset

Iris Versicolor

Iris Setosa

Iris Virginica

```
from sklearn.datasets import load_iris
datasets = load_iris()
```

```
x_data = datasets["data"]
x_data[:5]
```

```
array([[ 5.1,  3.5,  1.4,  0.2],
       [ 4.9,  3. ,  1.4,  0.2],
       [ 4.7,  3.2,  1.3,  0.2],
       [ 4.6,  3.1,  1.5,  0.2],
       [ 5. ,  3.6,  1.4,  0.2]])
```

```
y_data = datasets["target"]
y_data
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

컬럼명	의미	데이터 타입
Species	붓꽃의 종. setosa, versicolor, virginica 세 가지 값 중 하나	Factor
Sepal.Width	꽃받침의 너비	Number
Sepal.Length	꽃받침의 길이	Number
Petal.Width	꽃잎의 너비	Number
Petal.Length	꽃잎의 길이	Number

Load dataset

Softmax function

$$P_j = \frac{e^{\mathbf{x}^T \theta_j}}{\sum_{j=1}^{K} e^{\mathbf{x}^T \theta_j}}$$

```
def softmax(z):
    e = np.exp(z)
    p = e / np.sum(np.exp(z), axis=1).reshape([-1,1])
    return p
```

Cross Entropy

$$-\log L = -\log \prod_{i=1}^{m} \prod_{j=1}^{K} {p_j^{(i)}}^{v_{ij}} = -\sum_{i=1}^{m} \sum_{j=1}^{K} v_{ij} \log p_j^{(i)}$$

```
def cross_entropy_function(y, x, weights):
    z = x_data_minmax.dot(weights.T)
    result = - np.sum(
                np.sum(
                    (y * np.log(softmax(z))), axis=1).reshape((-1,1))
                )
    return result
```

Weights update

$$w_{kj} = w_{kj} - \alpha \frac{\partial l}{\partial w_{kj}}$$

$$= w_{kj} - \alpha \frac{\partial l}{\partial z_k} \frac{\partial z_k}{\partial w_{kj}}$$

$$= w_{kj} - \alpha (p_k - v_k) x_j$$

$$= w_{kj} + \alpha (v_k - p_k) x_j$$

$$\Rightarrow w_{kj} = w_{kj} + \alpha \sum_{i=1}^{n} (v_k^{(i)} - p_k^{(i)}) x_j^{(i)}$$

```
for _ in range(iterations):
    original_theta = np.copy(theta)
    for k in range(number_of_classes):
        for j in range(number_of_weights):
            partial_x = x[:, j]
            partial_entropy = y - softmax(x.dot(original_theta.T))
            theta[k][j]  = original_theta[k][j] + (
                alpha* partial_entropy[:,k].dot(partial_x.T) ) /150
```

Weights update

$$w_{kj} = w_{kj} - \alpha \frac{\partial l}{\partial w_{kj}}$$

$$= w_{kj} - \alpha \frac{\partial l}{\partial z_k} \frac{\partial z_k}{\partial w_{kj}}$$

$$= w_{kj} - \alpha (p_k - v_k) x_j$$

$$= w_{kj} + \alpha (v_k - p_k) x_j$$

$$\Rightarrow w_{kj} = w_{kj} + \alpha \sum_{i=1}^{n} (v_k^{(i)} - p_k^{(i)}) x_j^{(i)}$$

```
for _ in range(iterations):
    original_theta = np.copy(theta)
    for k in range(number_of_classes):
        for j in range(number_of_weights):
            partial_x = x[:, j]
            partial_entropy = y - softmax(x.dot(original_theta.T))
            theta[k][j]  = original_theta[k][j] + (
                alpha* partial_entropy[:,k].dot(partial_x.T) ) /150
```

One Hot Encoding for Y

```
y_data = y_data.reshape([-1,1])
y_data[:3]
```

```
array([[0],
       [0],
       [0]])
```

where v_{ij} $\begin{cases} 1 & \text{if } y^{(i)} \text{ is label } j \\ 0 & \text{if } y^{(i)} \text{ is NOT label } j \end{cases}$

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
enc.fit(y_data)
y_data = enc.transform(y_data).toarray()
y_data[:3]
```

```
array([[ 1.,  0.,  0.],
       [ 1.,  0.,  0.],
       [ 1.,  0.,  0.]])
```

Human knowledge belongs to the world.

Metrics for Multiclass

Multiclass Classification

Director of TEAMLAB
Sungchul Choi

Confusion matrix for multiclass

- Class별로 True Positive와 Error로 분류
- FN 행기준 , FP 열 기준으로 값 확인

		Prediction				
		A	B	C	D	E
Actual	A	TP_A	E_AB	E_AC	E_AD	E_AE
	B	E_BA	TP_B	E_BC	E_BD	E_BE
	C	E_CA	E_CB	TP_C	E_CD	E_CE
	D	E_DA	E_DB	E_DC	TP_D	E_DE
	E	E_EA	E_EB	E_EC	E_ED	TP_E

Accuracy for multiclass

- 전체 Class중 정확히 일치한 Class의 개수

		Prediction				
		A	B	C	D	E
Actual	A	TP_A	E_AB	E_AC	E_AD	E_AE
	B	E_BA	TP_B	E_BC	E_BD	E_BE
	C	E_CA	E_CB	TP_C	E_CD	E_CE
	D	E_DA	E_DB	E_DC	TP_D	E_DE
	E	E_EA	E_EB	E_EC	E_ED	TP_E

Precision for multiclass

- TP / (TP + FP), 하나의 클래스와 나머지 Column 클래스
- Precision A = TP_A / (TP_A + E_BA + E_CA + E_DA + E_EA)

		Prediction				
		A	B	C	D	E
Actual	A	TP_A	E_AB	E_AC	E_AD	E_AE
	B	E_BA	TP_B	E_BC	E_BD	E_BE
	C	E_CA	E_CB	TP_C	E_CD	E_CE
	D	E_DA	E_DB	E_DC	TP_D	E_DE
	E	E_EA	E_EB	E_EC	E_ED	TP_E

Recall for multiclass

- TP / (TP + FN), 하나의 클래스와 나머지 Row클래스
- Recall A = TP_A / (TP_A + E_AB + E_AC + E_AD + E_AE)

		Prediction				
		A	B	C	D	E
Actual	A	TP_A	E_AB	E_AC	E_AD	E_AE
	B	E_BA	TP_B	E_BC	E_BD	E_BE
	C	E_CA	E_CB	TP_C	E_CD	E_CE
	D	E_DA	E_DB	E_DC	TP_D	E_DE
	E	E_EA	E_EB	E_EC	E_ED	TP_E

Examples

Prediction

Actual

0	1	2	3	4	5	6	7	8	9	10
1	298	2	1	0	1	1	3	1	1	0
2	0	293	7	4	1	0	5	2	0	0
3	1	3	263	0	8	0	0	3	0	2
4	1	5	0	261	4	0	3	2	0	1
5	0	0	10	0	254	3	0	10	2	1
6	0	4	1	1	4	300	0	1	0	0
7	1	3	2	0	0	0	264	0	7	1
8	3	5	3	1	7	1	0	289	1	0
9	0	1	3	13	1	0	11	1	289	0
10	0	6	0	1	6	1	2	1	4	304

Human knowledge belongs to the world.

Multiclass with sklearn

Multiclass Classification

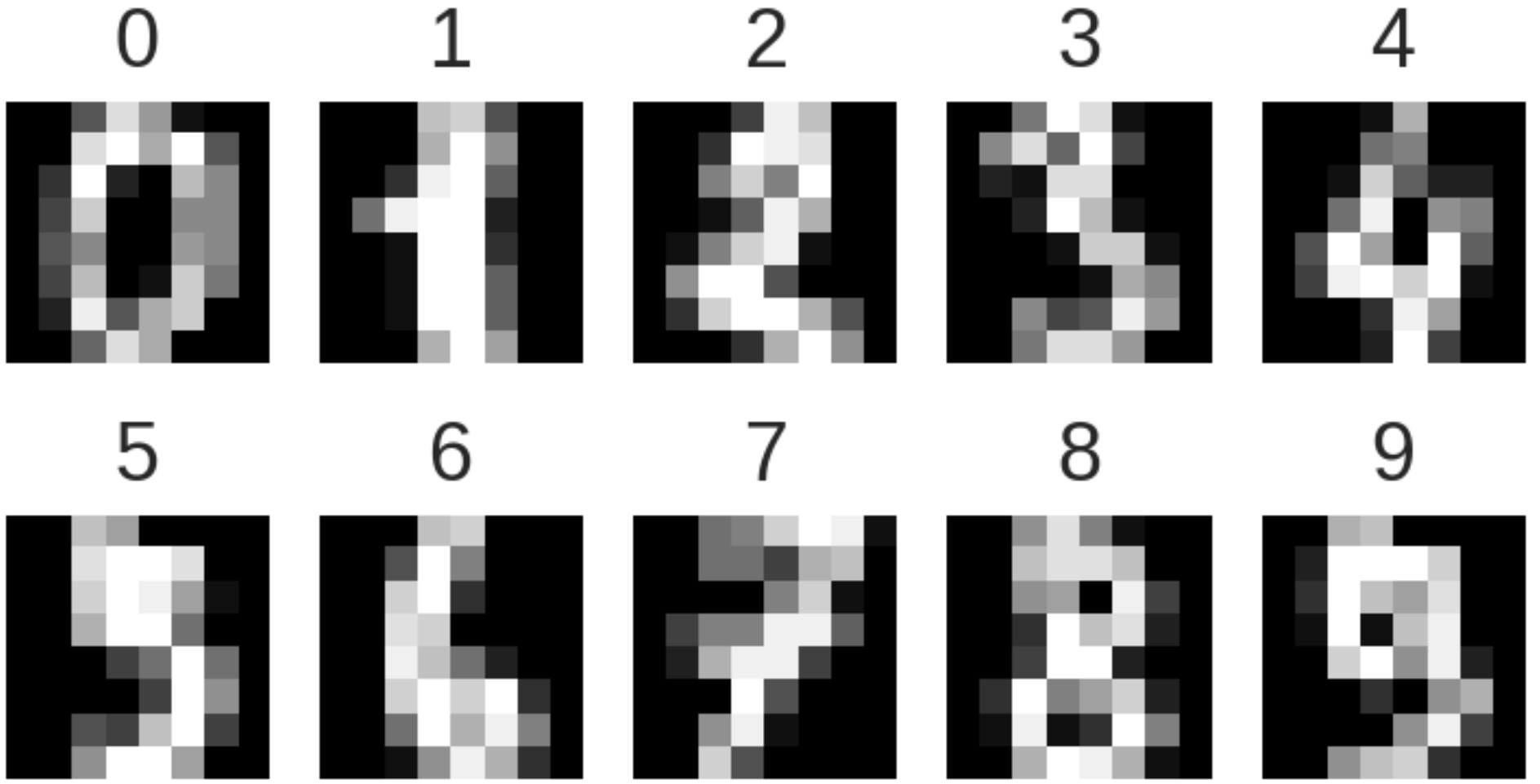
Director of TEAMLAB
Sungchul Choi

digit dataset

- Optical Recognition of Handwritten Digits Data Set
- 손 글씨로 쓴 숫자를 분류하는 데이터 셋
- MNIST가 원조, scikit-learn 에서 8 by 8 image 제공

https://thebook.io/006723/ch04/01/

digit dataset



Data loading

```
from sklearn import datasets
```

```
digit_dataset = datasets.load_digits()
```

```
digit_dataset.keys()
```

```
dict_keys(['data', 'target', 'target_names', 'images', 'DESCR'])
```

Data loading

```
digit_dataset["data"]
```

```
array([[  0.,   0.,   5., ...,   0.,   0.,   0.],
       [  0.,   0.,   0., ...,  10.,   0.,   0.],
       [  0.,   0.,   0., ...,  16.,   9.,   0.],
       ...,
       [  0.,   0.,   1., ...,   6.,   0.,   0.],
       [  0.,   0.,   2., ...,  12.,   0.,   0.],
       [  0.,   0.,  10., ...,  12.,   1.,   0.]])
```

```
digit_dataset["data"].shape
```

```
(1797, 64)
```

```
digit_dataset["target_names"]
```

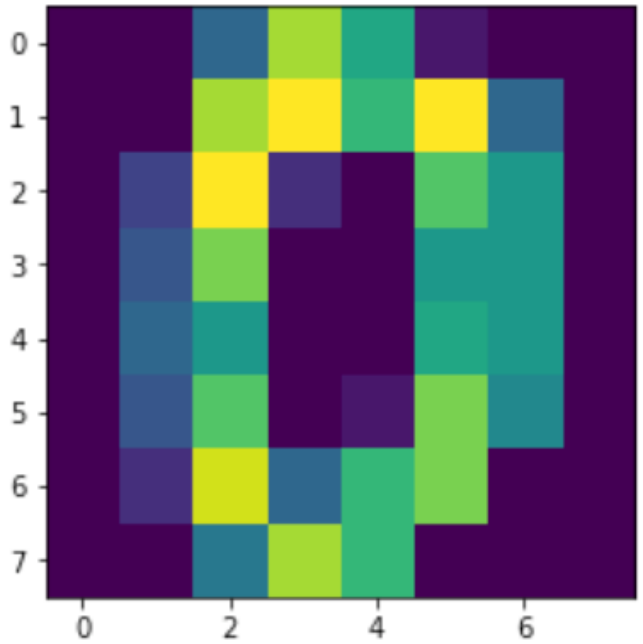
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Data loading

```
digit_dataset["images"][0]
```

```
array([[  0.,   0.,   5.,  13.,   9.,   1.,   0.,   0.],
       [  0.,   0.,  13.,  15.,  10.,  15.,   5.,   0.],
       [  0.,   3.,  15.,   2.,   0.,  11.,   8.,   0.],
       [  0.,   4.,  12.,   0.,   0.,   8.,   8.,   0.],
       [  0.,   5.,   8.,   0.,   0.,   9.,   8.,   0.],
       [  0.,   4.,  11.,   0.,   1.,  12.,   7.,   0.],
       [  0.,   2.,  14.,   5.,  10.,  12.,   0.,   0.],
       [  0.,   0.,   6.,  13.,  10.,   0.,   0.,   0.]])
```

```
from matplotlib import pyplot as plt
plt.imshow(digit_dataset["images"][0])
plt.show()
```



Multiclass for LogisticRegression Class

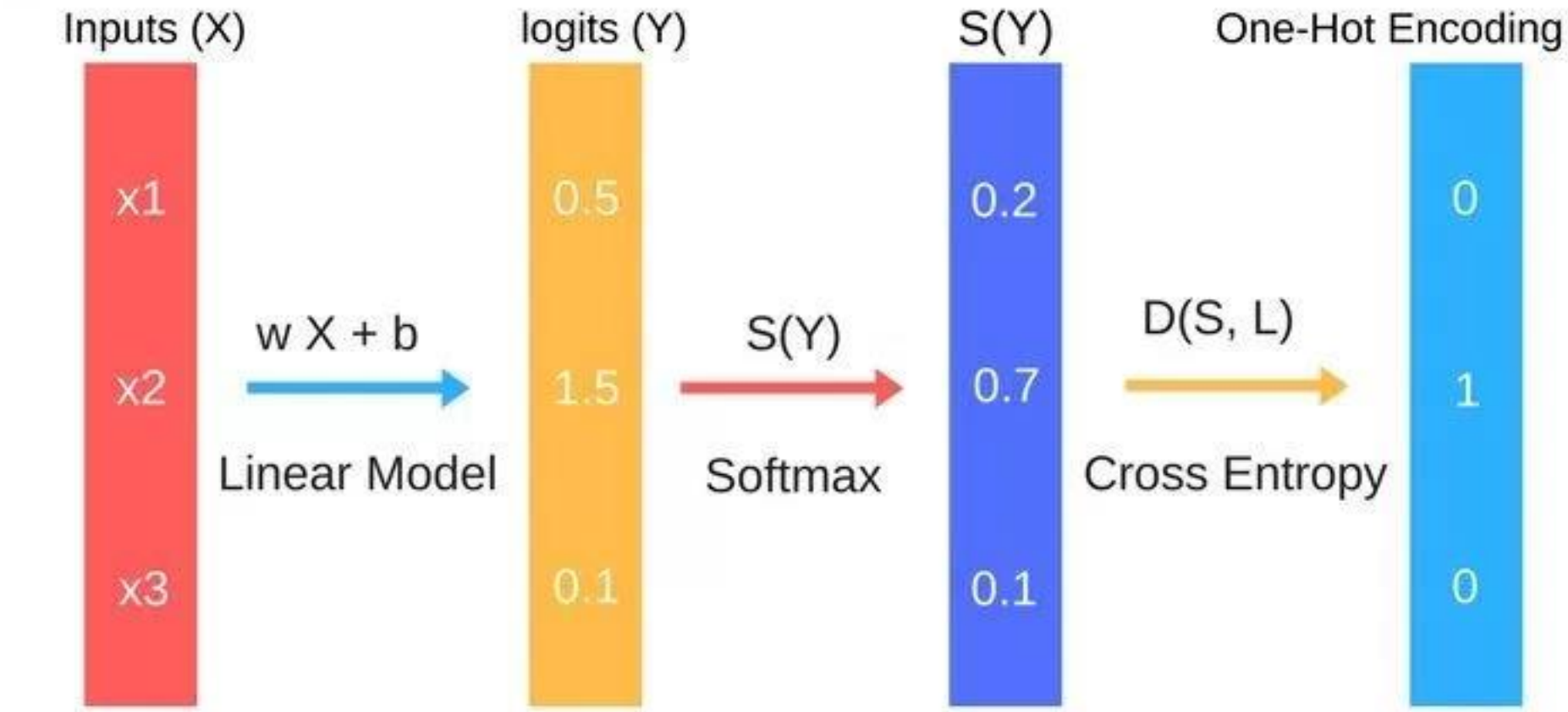
class `sklearn.linear_model.` **`LogisticRegression`** (*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1*) [source]

multi_class : str, {'ovr', 'multinomial'}, default: 'ovr'

Multiclass option can be either 'ovr' or 'multinomial'. If the option chosen is 'ovr', then a binary problem is fit for each label. Else the loss minimised is the multinomial loss fit across the entire probability distribution. Does not work for liblinear solver.

New in version 0.18: Stochastic Average Gradient descent solver for 'multinomial' case.

Multiclass for LogisticRegression Class



Multinomial Logistic Classifier

@ dataaspirant.com

http://dataaspirant.com/2017/03/14/multinomial-logistic-regression-model-works-machine-learning/

Multiclass for LogisticRegression Class

```
from sklearn.linear_model import LogisticRegression

logreg_ovr = LogisticRegression(multi_class="ovr")
logreg_softmax = LogisticRegression(multi_class="multinomial", solver="sag")
```

```
logreg_ovr.fit(X_train, y_train)
logreg_softmax.fit(X_train, y_train)
```

One vs One or One vs Rest Classifier

User guide: See the Multiclass and multilabel algorithms section for further details.

`multiclass.OneVsRestClassifier` (estimator[, …])	One-vs-the-rest (OvR) multiclass/multilabel strategy
`multiclass.OneVsOneClassifier` (estimator[, …])	One-vs-one multiclass strategy
`multiclass.OutputCodeClassifier` (estimator[, …])	(Error-Correcting) Output-Code multiclass strategy

OneVsRestClassifier(logreg_ovr).fit(X_train, y_train)

OneVsOneClassifier(logreg_ovr).fit(X_train, y_train)

Human knowledge belongs to the world.

Probability Overview

Naive Bayes Classifier

Director of TEAMLAB
Sungchul Choi

머신러닝의 학습 방법들

- Gradient descent based learning
- Probability theory based learning
- Information theory based learning
- Distance similarity based learning

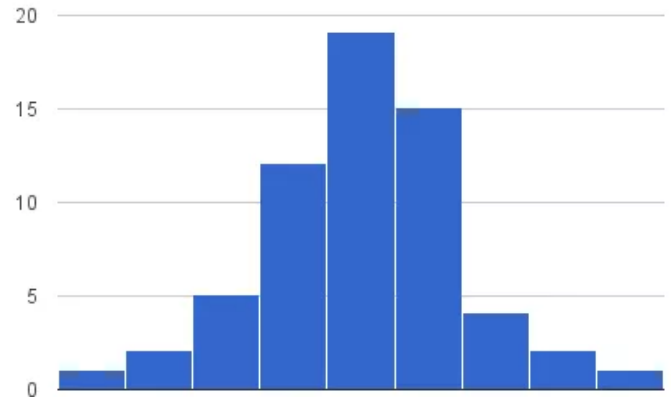
머신러닝의 학습 방법들

- Gradient descent based learning
- **Probability theory based learning**
- Information theory based learning
- Distance similarity based learning

Probability

- **이산형 값**

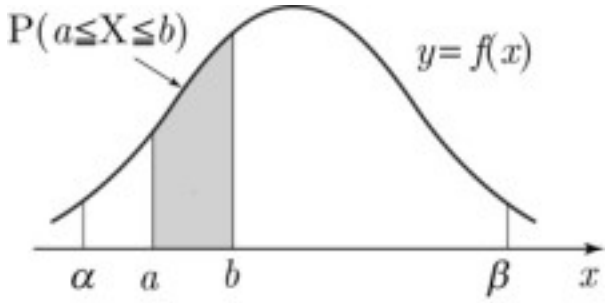
$$P(X) = \frac{count(Event\_X)}{count(ALL\_Event)}$$



Source:
https://goo.gl/D9VCSL

- **연속형 값**

$$P(-\infty < x < \infty) \int_{-\infty}^{\infty} f(x)\,dx = 1$$



Source:
https://goo.gl/DVNSH2

Basic concepts of probability

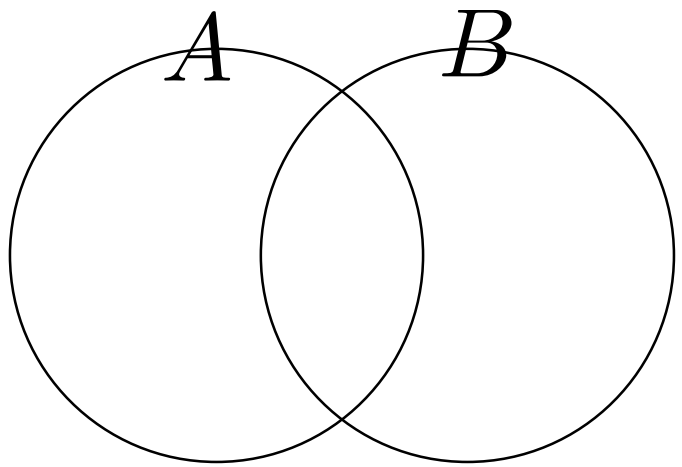
$$0 \leq P(E) \leq 1$$

$$P(S) = \sum_{i=1}^{N} P(E_i) = 1 \qquad \text{if all } E_i \text{ are independent}$$

$$P(A \cap B) \qquad P(A \cup B) \qquad P(A^c) = 1 - P(A)$$

Conditional probability

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$



$A \cap B$

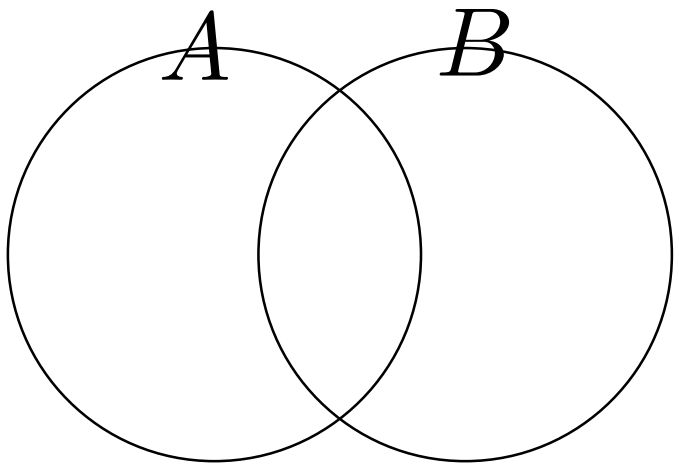
Conditional probability

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$P(A) = \{\text{set of odd number}\}$$

$$P(B) = \{\text{less than 4 in a dice}\}$$

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$



$A \cap B$

Human knowledge belongs to the world.

Bayes's Theorem

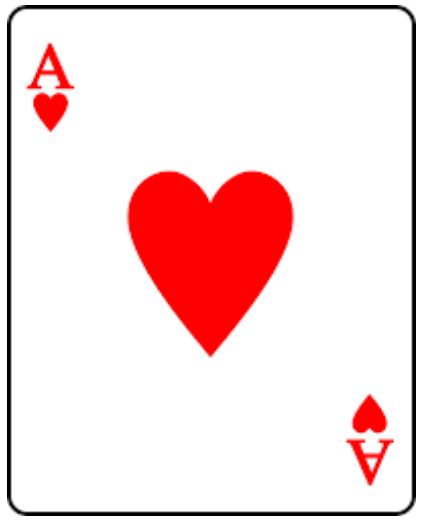
Naive Bayes Classifier

Director of TEAMLAB
Sungchul Choi

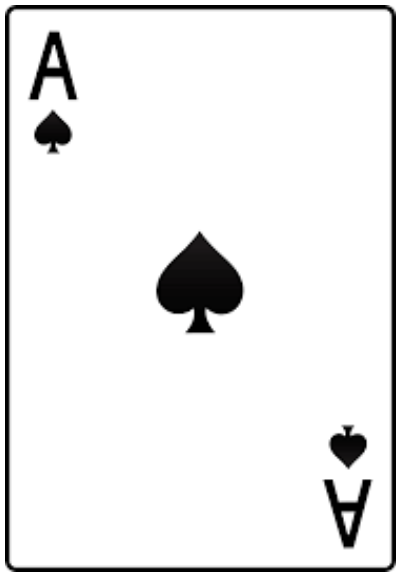
Bayes's theorem

- 경험에 의한 확률의 업데이트
- 사전 확률(given)에서 사건 발생을 통해 사후 확률로
- 빈도주의 vs. 베이즈주의
- 객관적 확률은 존재하지 않는다!

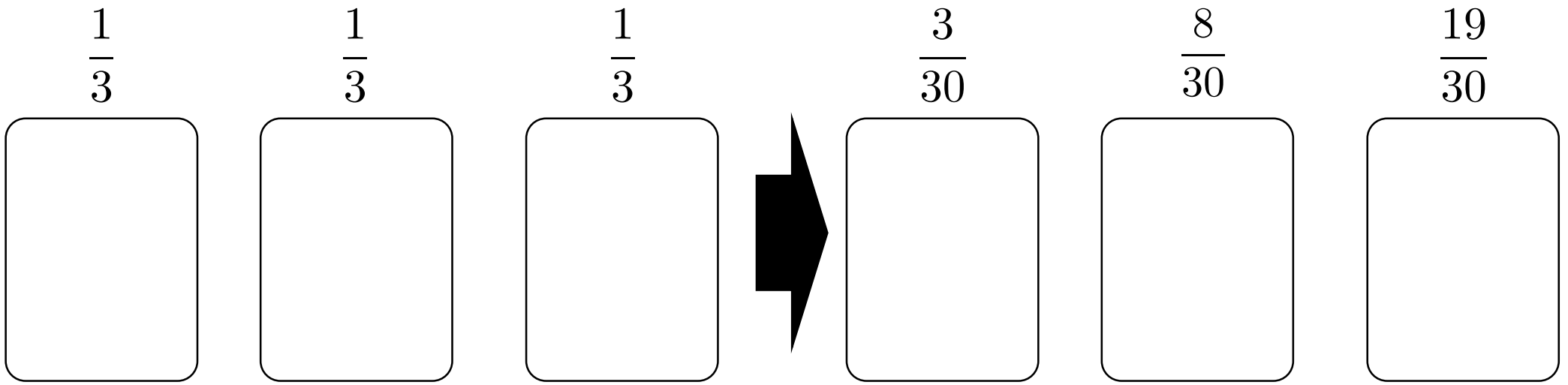
Queen card game


A
A

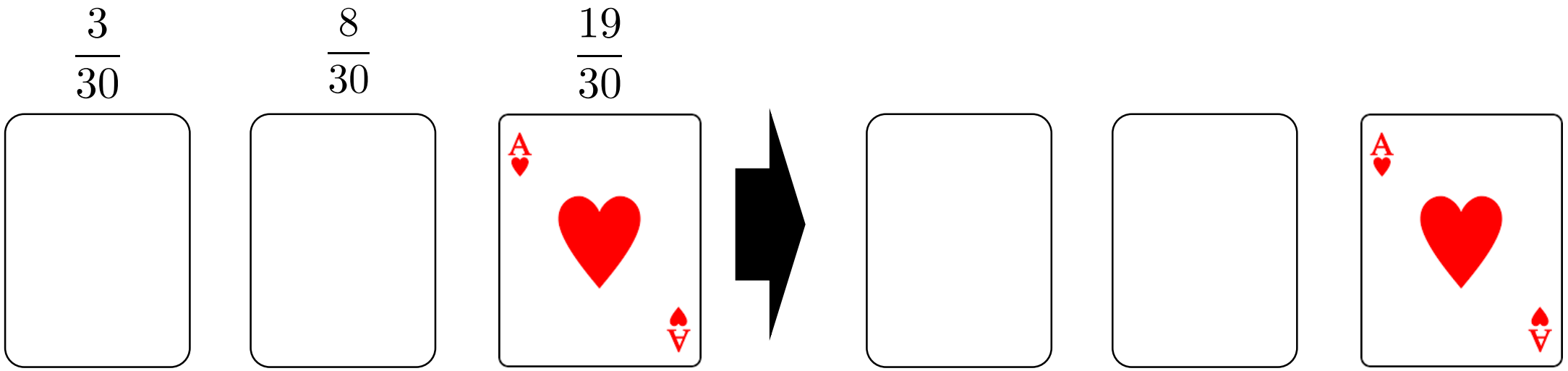

Q
Q


A
A

Probability updated



Probability updated



Bayes's theorem

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \qquad P(B|A) = \frac{P(A \cap B)}{P(A)}$$

$$P(A \cap B) = P(B)P(A|B) = P(A)P(B|A)$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)P(B|A)}{P(B)}$$

Cookie Quiz

쿠키 두 그릇이 있다고 한다. 첫 번째 그릇에는 바닐라 쿠키 30개와 초콜렛 쿠키 10개가 있고, 두 번째 그릇에는 각 쿠키가 20개씩 있다. 임의 쿠키를 집었는데 해당 쿠키가 바닐라 쿠키이다. 이 쿠키가 그릇 1에서 나왔을 확률은?

Source: https://goo.gl/GzopZy

Cookie Quiz

쿠키 두 그릇이 있다고 한다. 첫 번째 그릇에는 바닐라 쿠키 30개와 초콜렛 쿠키 10개가 있고, 두 번째 그릇에는 각 쿠키가 20개씩 있다. 임의 쿠키를 집었는데 해당 쿠키가 바닐라 쿠키이다. 이 쿠키가 그릇 1에서 나왔을 확률은?

$$P(Choco)$$

$$P(Vanilla)$$

$$P(B1)$$

$$P(B2)$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)P(B|A)}{P(B)}$$

Source: https://goo.gl/GzopZy

Bayes's theorem

H is Class

D is Data

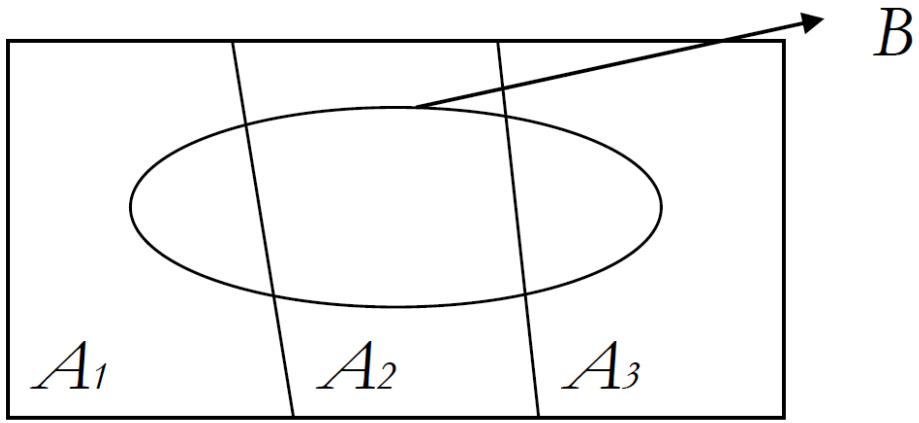
사후확률

사전 확률

우도

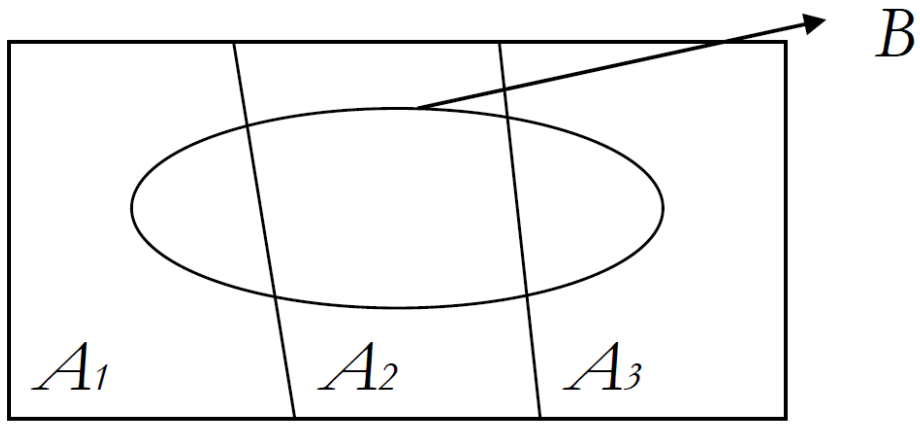
$$P(H|D) = \frac{P(H)P(D|H)}{P(D)}$$

데이터가 발생할 확률
(Evidence)



Bayes's theorem

$$P(H|D) = \frac{P(C)P(D|H)}{P(D)}$$



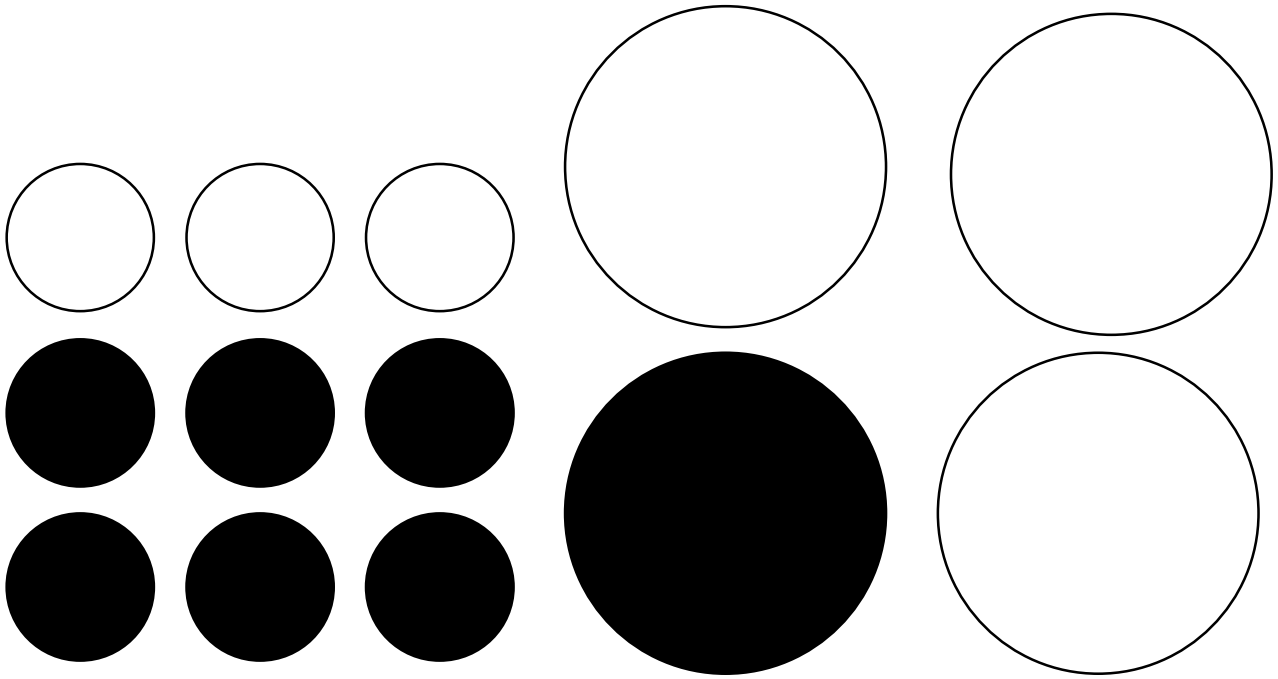
$$P(A_1 \cap B) + P(A_2 \cap B) + P(A_3 \cap B)$$

$$P(B) = P(A_1)P(B \cap A_1) + P(A_2)P(B \cap A_2) + P(A_3)P(B \cap A_3)$$

$$P(A \cap B) = P(B)P(A|B) = P(A)P(B|A)$$

Example

공1 무작위로 1개 나오는 다음과 같은 완구가 있다.

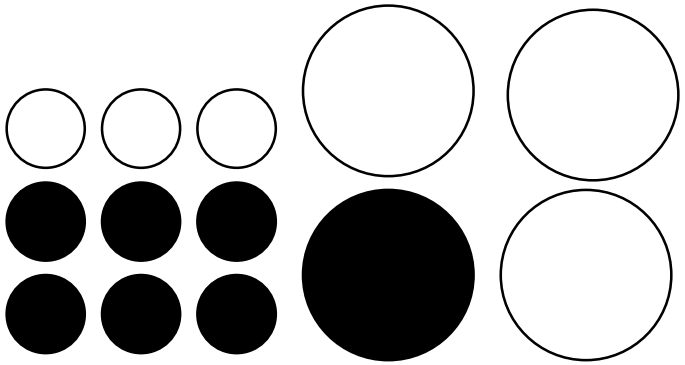


나온 공이 “큰공”일때,

검은색일 확률은?

Source: https://goo.gl/GzopZy

Example



$$P(H|D) = \frac{P(C)P(D|H)}{P(D)}$$

$$P(B) = P(A_1)P(B \cap A_1) + P(A_2)P(B \cap A_2) + P(A_3)P(B \cap A_3)$$

Source: https://goo.gl/GzopZy

Human knowledge belongs to the world.

Simple Bayes Classifier

Naive Bayes Classifier

Director of TEAMLAB
Sungchul Choi

Viagra 스팸 필터기

- Viagra라는 단어의 유무를 통해 스팸 여부 확인
- Viagra 단어가 들어가면 무조건 스팸?
- 어느정도 확률로 스팸이라고 해야할까?

Viagra 스팸 필터기

number	viagra	spam	number	viagra	spam
1	1	1	11	1	0
2	0	0	12	0	0
3	0	0	13	0	0
4	0	0	14	1	0
5	0	0	15	0	0
6	0	0	16	0	0
7	0	1	17	0	0
8	0	0	18	0	1
9	1	1	19	0	1
10	1	0	20	1	1

Viagra 스팸 필터기

$$P(spam|viagra) = \frac{P(spam)P(viagra|spam)}{P(viagra)}$$

number	viagra	spam	number	viagra	spam
1	1	1	11	1	0
2	0	0	12	0	0
3	0	0	13	0	0
4	0	0	14	1	0
5	0	0	15	0	0
6	0	0	16	0	0
7	0	1	17	0	0
8	0	0	18	0	1
9	1	1	19	0	1
10	1	0	20	1	1

$$P(H|D) = \frac{P(C)P(D|H)}{P(D)}$$

$$P(viagra \cap spam)$$

$$P(viagra) = \frac{count(viagra)}{count(ALL_{samples})}$$

$$P(spam) = \frac{count(spam)}{count(ALL_{samples})}$$

Viagra 스팸 필터기

$$P(viagra) = \frac{count(viagra)}{count(ALL_{samples})} \quad P(spam) = \frac{count(spam)}{count(ALL_{samples})}$$

```
# P(Viagra)
p_viagra = sum(np_data[:, 0] == 1) / len(np_data)
p_spam = sum(np_data[:, 1] == 1) / len(np data)
p_v_cap_s = sum((np_data[:, 0]
p_n_v_cap_s = sum((np_data[:,
```

$$P(viagra \cap spam)$$

```
# P(spam | viagra)
p_spam * (p_v_cap_s / p_spa
```

```
1] == 1)) / len(np_data)
:, 1] == 1)) / len(np_data)
```

```
# P(spam | viagra)
p_spam * (p_v_cap_s / p_spam ) / p_viagra
```

```
0.5
```

$$P(spam|viagra) = \frac{P(spam)P(viagra|spam)}{P(viagra)}$$

```
# P(spam | ~viagra)
p_spam * (p_n_v_cap_s / p_spam ) / (1-p_viagra)
```

```
0.2142857142857143
```

스팸 필터기에
여러단어를 검사하고 싶다...

Human knowledge belongs to the world.

NB Classifier Overview

Naive Bayes Classifier

Director of TEAMLAB
Sungchul Choi

제대로 된 스팸 필터기를 만들어보자

- Viagra 단어외에 영향을 주는 단어들은?
- 오히려 스팸을 제외해주는 단어는 어떻게 찾지?
- 한번에 여러 단어들을 고려하는 필터기를 만들자

Feature의 확장

$$P(spam|viagra)$$

$$P(spam|viagra, hello, lucky, marketing...)$$

변수가 많을 때, 조건부 확률의 변화

Multivariate multiplication rule

$$P(Y|X_1, X_2) = \frac{P(Y \cap X_1 \cap X_2)}{P(X_1 \cap X_2)}$$

$$P(Y \cap X_1 \cap X_2) = P(Y|X_1, X_2)P(X_1 \cap X_2)$$

$$P(X_1, X_2, X_3, ...X_n)$$
$$= P(X_1)P(X_2|X_1)P(X_3|X_1, X_2) \dots P(X_n|X_1 \dots X_{x-1})$$

Problems

- 계산이 어려워짐
- Feature의 차원이 증가하면 Sparse Vector가 생성
  → 확률이 0이 되는 값이 늘어남

Naïve Bayes Classifier

- 복잡하게 하지말고 단순(naïve)하게 해결하자
- 각 변수의 관계가 독립임을 가정
- 계산이 용이해지고, 성능이 생각보다 좋음

Joint Probability

$$P(A \cap B) = P(A)P(B) \quad \text{if } A \text{ and } B \text{ are independent}$$

$$P(Y|X_1 \cap X_2) = \frac{P(Y)P(X_1 \cap X_2|Y)}{P(X_1 \cap X_2)}$$

$$\frac{P(Y)P(X_1|Y)P(X_2|Y)}{P(X_1)P(X_2)}$$

Naïve Bayes Classifier

$$P(Y|X_1 \cap X_2) = \frac{P(Y)P(X_1 \cap X_2|Y)}{P(X_1 \cap X_2)} = \frac{P(Y)P(X_1|Y)P(X_2|Y)}{P(X_1)P(X_2)}$$

$$P(Y_c|X_1, \ldots, X_n) = \frac{P(Y_c)\prod_{i=1}^{n} P(X_i|Y_c)}{\prod_{i=1}^{n} P(X_i)} \quad Y_c \text{ is a label}$$

Issues

- 너무 많은 확률값 → 0에 수렴하게 되는 문제
- 곱하지 말고 더하자 → log

$$\log\{P(Y_c)\prod_{i=1}^{n}P(X_i|Y_c)\} = \log P(Y_c) + \sum_{i=1}^{n}\log P(X_i|Y_c)$$

$$P(Y_c|X_1,\ldots,X_n) = \frac{P(Y_c)\prod_{i=1}^{n}P(X_i|Y_c)}{\prod_{i=1}^{n}P(X_i)} \quad Y_c \text{ is a label}$$

Issues

- 확률이 0인 변수들이 존재함 → 전체값 0
- 작게나마 확률이 나올 수 있도록 변경 → 스무딩

$$P(X|Y) = \frac{count(X \cap Y) + k}{count(Y) + (k * |\text{number of class}|)}$$

$$\log\{P(Y_c)\prod_{i=1}^{n} P(X_i|Y_c)\} = \log P(Y_c) + \sum_{i=1}^{n} \log P(X_i|Y_c)$$

Human knowledge belongs to the world.

NB Classifier Implementation

Naive Bayes Classifier

Director of TEAMLAB
Sungchul Choi

Dataset – German Credit

- 대출 사기인가? 아닌가를 예측하는 문제
- 데이터를 NB에 맞도록 간단하게 변환
- Binary 데이터들로 이루어진 대출 사기 데이터

1	ID	History	CoApplicant	Accommodation	Fraud
2	1	current	none	own	true
3	2	paid	none	own	false
4	3	paid	none	own	false
5	4	paid	guarantor	rent	true
6	5	arrears	none	own	false
7	6	arrears	none	own	true
8	7	current	none	own	false
9	8	arrears	none	own	false
10	9	current	none	rent	false
11	10	none	none	own	true
12	11	current	coapplicant	own	false
13	12	current	none	own	true
14	13	current	none	rent	true
15	14	paid	none	own	false
16	15	arrears	none	own	false
17	16	current	none	own	false
18	17	arrears	coapplicant	rent	false
19	18	arrears	none	free	false
20	19	arrears	none	own	false
21	20	paid	none	own	false

Preprocessing

	History	CoApplicant	Accommodation
0	current	none	own
1	paid	none	own
2	paid	none	own
3	paid	guarantor	rent
4	arrears	none	own

One-Hot Encoding

```
x_df = pd.get_dummies(df)
x_df.head()
```

	History_arrears	History_current	History_none	History_paid	CoApplicant_coapplicant	CoApplicant_guarantor	CoApplicant_none	Accommodation_free	Accom
0	0	1	0	0	0	0	1	0	
1	0	0	0	1	0	0	1	0	
2	0	0	0	1	0	0	1	0	
3	0	0	0	1	0	1	0	0	
4	1	0	0	0	0	0	1	0	

```
x_data = x_df.as_matrix()
x_data
```

```
array([[0, 1, 0, 0, 0, 0, 1, 0, 1, 0],
       [0, 0, 0, 1, 0, 0, 1, 0, 1, 0],
       [0, 0, 0, 1, 0, 0, 1, 0, 1, 0],
       [0, 0, 0, 1, 0, 1, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 1, 0, 1, 0],
       [1, 0, 0, 0, 0, 0, 1, 0, 1, 0],
       [0, 1, 0, 0, 0, 0, 1, 0, 1, 0],
       [1, 0, 0, 0, 0, 0, 1, 0, 1, 0],
       [0, 1, 0, 0, 0, 0, 1, 0, 0, 1],
       [0, 0, 1, 0, 0, 0, 1, 0, 1, 0],
       [0, 1, 0, 0, 1, 0, 0, 0, 1, 0],
       [0, 1, 0, 0, 0, 0, 1, 0, 1, 0],
       [0, 1, 0, 0, 0, 0, 1, 0, 0, 1],
       [0, 0, 0, 1, 0, 0, 1, 0, 1, 0],
       [1, 0, 0, 0, 0, 0, 1, 0, 1, 0],
       [0, 1, 0, 0, 0, 0, 1, 0, 1, 0],
       [1, 0, 0, 0, 1, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 1, 1, 0, 0],
       [1, 0, 0, 0, 0, 0, 1, 0, 1, 0],
       [0, 0, 0, 1, 0, 0, 1, 0, 1, 0]], dtype=uint8)
```

Modelling

$$P(Y_c|X_1,\ldots,X_n) = \frac{P(Y_c)\prod_{i=1}^{n} P(X_i|Y_c)}{\prod_{i=1}^{n} P(X_i)} \quad Y_c \text{ is a label}$$

```
P_Y_True = sum(Y_data==True) / len(Y_data)
P_Y_False = 1 - P_Y_True

P_Y_True,P_Y_False
```

```
(0.2999999999999999, 0.6999999999999996)
```

Modelling

```
ix_Y_True = np.where(Y_data)
ix_Y_False = np.where(Y_data==False)

ix_Y_True, ix_Y_False
```

Y의 Index

```
((array([ 0,  3,  5,  9, 11, 12]),),
 (array([ 1,  2,  4,  6,  7,  8, 10, 13, 14, 15, 16, 17, 18, 19]),))
```

```
p_x_y_true = (x_data[ix_Y_True].sum(axis=0)) / sum(Y_data==True)
p_x_y_false = (x_data[ix_Y_False].sum(axis=0)) / sum(Y_data==False)

p_x_y_true, p_x_y_false
```

$P(X_i|Y_c)$

```
(array([ 0.16666667,  0.5       ,  0.16666667,  0.16666667,  0.        ,
         0.16666667,  0.83333333,  0.        ,  0.66666667,  0.33333333]),
 array([ 0.42857143,  0.28571429,  0.        ,  0.28571429,  0.14285714,
         0.        ,  0.85714286,  0.07142857,  0.78571429,  0.14285714]))
```

Classifier

```
# History_arrears  History_current  History_none  History_paid  CoApplicant_coapplicant
x_test = [0,1,0,0,0,1,0, 0,1,0]

p_y_true_test = P_Y_True + p_x_y_true.dot(x_test)
p_y_false_test = P_Y_False + p_x_y_false.dot(x_test)

p_y_true_test , p_y_false_test
```

```
(1.6333333333333333, 1.7714285714285714)
```

```
p_y_true_test < p_y_false_test
```

```
True
```

P(Y=1|X)의 확률과 P(Y=0|X)의 확률 비교

Human knowledge belongs to the world.

Multinomial NB

Naive Bayes Classifier

Director of TEAMLAB
Sungchul Choi

Multinomial Naïve Bayes

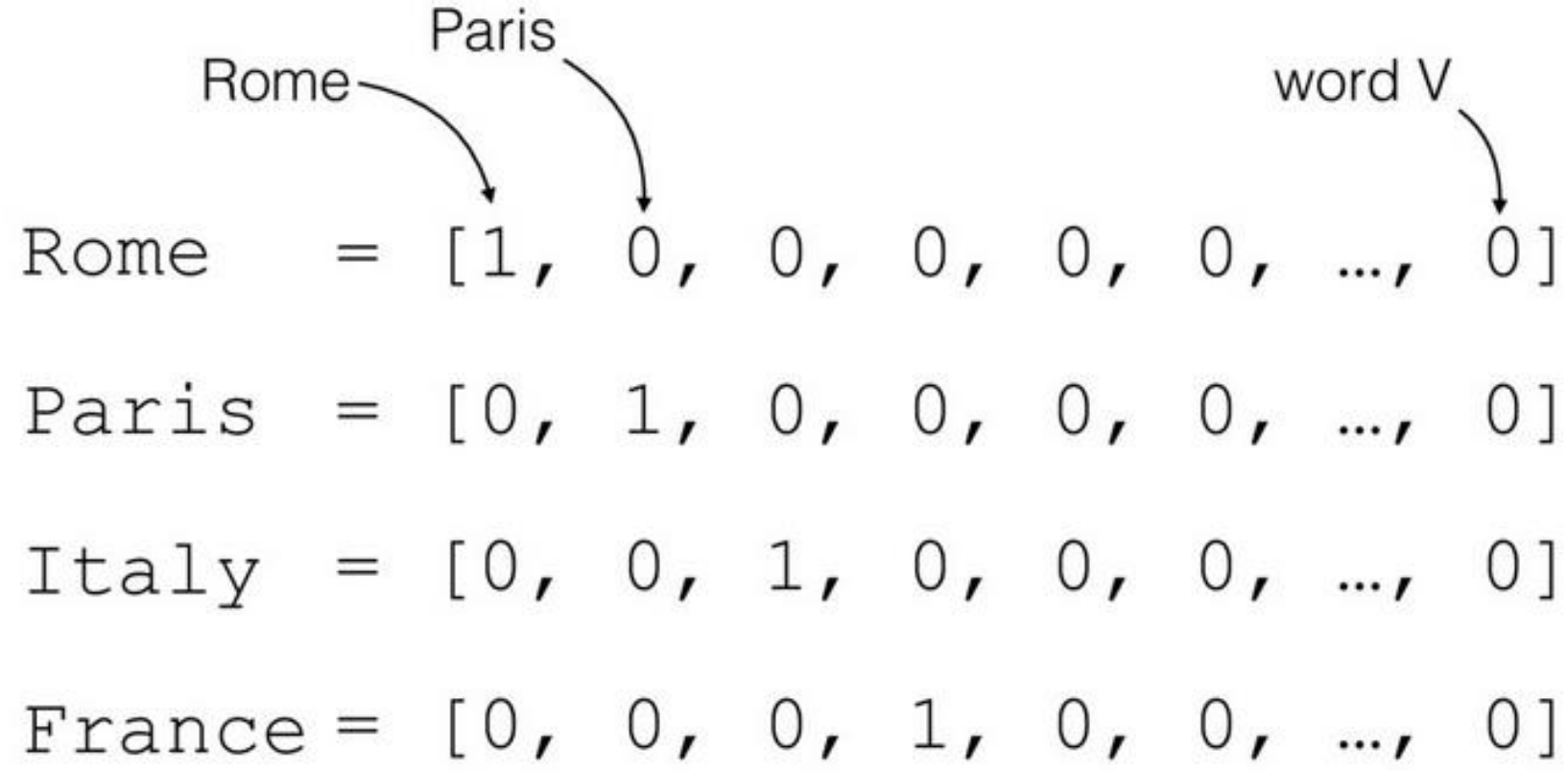
- X값이 Binary가 아니라 1 이상의 값을 가지는 문제
- 일반적으로 Text 문제를 분류할 때 많이 쓰임
- 단어의 존재 유무가 아닌 단어의 출현횟수 Feature로

Text의 Feature 표현?

문자 ➔ Feature

문자를 Vector로 – One-hot Encoding

하나의 단어를 Vector의 Index로 인식, 단어 존재시 1 없으면 0



Bag of words

단어별로 인덱스를 부여해서
한 문장(또는 문서)의 단어의 개수를 Vector로 표현

the dog is on the table

Bag of words

단어별로 인덱스를 부여해서
한 문장(또는 문서)의 단어의 개수를 Vector로 표현

	are	call	from	hello	home	how	me	money	now	tomorrow	win	you
0	1	0	0	1	0	1	0	0	0	0	0	1
1	0	0	1	0	1	0	0	1	0	0	2	0
2	0	1	0	0	0	0	1	0	1	0	0	0
3	0	1	0	1	0	0	0	0	0	1	0	1

다시 돌아가서..

Multinomial Naïve Bayes

$$P(Y_c|X_1,\ldots,X_n) = \frac{P(Y_c)\prod_{i=1}^{n} P(X_i|Y_c)}{\prod_{i=1}^{n} P(X_i)} \quad Y_c \text{ is a label}$$

기본식!!

Multinomial Naïve Bayes

$$P(Y_c|X_1,\ldots,X_n) = \frac{P(Y_c)\prod_{i=1}^{n} P(X_i|Y_c)}{\prod_{i=1}^{n} P(X_i)} \quad Y_c \text{ is a label}$$

Likelihood만 바뀜

Multinomial Naïve Bayes

- 식 계산하는 방식이 다름

$$\prod_{i=1}^{n} P(X_i|Y_c) \quad P(X_i \mid Y_c) = \frac{\sum tf(x_i, d \in Y_c) + \alpha}{\sum N_{d \in Y_c} + \alpha \cdot V}$$

- x_i: A word from the feature vector **x** of a particular sample.
- $\sum tf(x_i, d \in y_c)$: The sum of raw term frequencies of word x_i from all documents in the training sample that belong to class y_c.
- $\sum N_{d \in y_c}$: The sum of all term frequencies in the training dataset for class y_c.
- α: An additive smoothing parameter ($\alpha=1$ for Laplace smoothing).
- V: The size of the vocabulary (number of different words in the training set).

http://sebastianraschka.com/Articles/2014_naive_bayes_1.html

Multinomial Naïve Bayes

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

$$P(X_i \mid Y_c) = \frac{\sum tf(x_i, d \in Y_c) + \alpha}{\sum N_{d \in Y_c} + \alpha \cdot V}$$

$$P(Y_c|X_1, \ldots, X_n) = \frac{P(Y_c) \prod_{i=1}^{n} P(X_i|Y_c)}{\prod_{i=1}^{n} P(X_i)}$$

http://slideplayer.com/slide/10998986/

Multinomial Naïve Bayes

Test	5	Chinese Chinese Chinese Tokyo Japan	?

Conditional Probabilities:

P(Chinese|*c*) = (5+1) / (8+6) = 6/14 = 3/7

P(Tokyo|*c*) = (0+1) / (8+6) = 1/14

P(Japan|*c*) = (0+1) / (8+6) = 1/14

P(Chinese|*j*) = (1+1) / (3+6) = 2/9

P(Tokyo|*j*) = (1+1) / (3+6) = 2/9

P(Japan|*j*) = (1+1) / (3+6) = 2/9

$$P(Y_c|X_1,\ldots,X_n) = \frac{P(Y_c)\prod_{i=1}^{n} P(X_i|Y_c)}{\prod_{i=1}^{n} P(X_i)}$$

http://slideplayer.com/slide/10998986/

Human knowledge belongs to the world.

Gaussian NB

Naive Bayes Classifier

Director of TEAMLAB
Sungchul Choi

Gaussian Naïve Bayes

- Category 데이터가 아닌경우에 NB 의 적용
- Continuous 데이터의 적용을 위해 y의 분포를 정규 분포(gaussian)으로 가정함
- 확률 밀도 함수 상의 해당 값 x 가 나올 확률로 NB를 구현함

Gaussian Naïve Bayes

$$P(x_i \mid Y_c) = \frac{1}{\sqrt{2\pi\sigma_{Y_c}^2}} \exp\left(-\frac{(x_i - \mu_{Y_c})^2}{2\sigma_{Y_c}^2}\right),$$

- 특정 Y_c에 대한 X_i의 평균, 표준편차를 μ, σ 에 대입

Gaussian Naïve Bayes

$$P(x_i \mid Y_c) = \frac{1}{\sqrt{2\pi\sigma_{Y_c}^2}} \exp\left(-\frac{(x_i - \mu_{Y_c})^2}{2\sigma_{Y_c}^2}\right),$$

Person	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

Person	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033*10-02	176.25	1.2292*10+02	11.25	9.1667*10-01
female	5.4175	9.7225*10-02	132.5	5.5833*10+02	7.5	1.6667

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Gaussian Naïve Bayes

$$P(x_i \mid Y_c) = \frac{1}{\sqrt{2\pi\sigma_{Y_c}^2}} \exp\left(-\frac{(x_i - \mu_{Y_c})^2}{2\sigma_{Y_c}^2}\right),$$

Person	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033*10-02	176.25	1.2292*10+02	11.25	9.1667*10-01
female	5.4175	9.7225*10-02	132.5	5.5833*10+02	7.5	1.6667

Person	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

$$\text{posterior (male)} = \frac{P(\text{male})\, p(\text{height} \mid \text{male})\, p(\text{weight} \mid \text{male})\, p(\text{foot size} \mid \text{male})}{evidence}$$

$$\text{posterior (female)} = \frac{P(\text{female})\, p(\text{height} \mid \text{female})\, p(\text{weight} \mid \text{female})\, p(\text{foot size} \mid \text{female})}{evidence}$$

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Gaussian Naïve Bayes

$$P(x_i \mid Y_c) = \frac{1}{\sqrt{2\pi\sigma_{Y_c}^2}} \exp\left(-\frac{(x_i - \mu_{Y_c})^2}{2\sigma_{Y_c}^2}\right),$$

Person	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033*10-02	176.25	1.2292*10+02	11.25	9.1667*10-01
female	5.4175	9.7225*10-02	132.5	5.5833*10+02	7.5	1.6667

Person	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

$$P(\text{male}) = 0.5$$

$$p(\text{height} \mid \text{male}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(6-\mu)^2}{2\sigma^2}\right) \approx 1.5789,$$

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Gaussian Naïve Bayes

$$P(x_i \mid Y_c) = \frac{1}{\sqrt{2\pi\sigma_{Y_c}^2}} \exp\left(-\frac{(x_i - \mu_{Y_c})^2}{2\sigma_{Y_c}^2}\right),$$

Person	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033*10-02	176.25	1.2292*10+02	11.25	9.1667*10-01
female	5.4175	9.7225*10-02	132.5	5.5833*10+02	7.5	1.6667

Person	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

$p(\text{weight} \mid \text{male}) = 5.9881 \cdot 10^{-6}$
$p(\text{foot size} \mid \text{male}) = 1.3112 \cdot 10^{-3}$
$\text{posterior numerator (male)} = \text{their product} = 6.1984 \cdot 10^{-9}$
$P(\text{female}) = 0.5$
$p(\text{height} \mid \text{female}) = 2.2346 \cdot 10^{-1}$
$p(\text{weight} \mid \text{female}) = 1.6789 \cdot 10^{-2}$
$p(\text{foot size} \mid \text{female}) = 2.8669 \cdot 10^{-1}$
$\text{posterior numerator (female)} = \text{their product} = 5.3778 \cdot 10^{-4}$

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Gaussian Naïve Bayes

$$P(x_i \mid Y_c) = \frac{1}{\sqrt{2\pi\sigma_{Y_c}^2}} \exp\left(-\frac{(x_i - \mu_{Y_c})^2}{2\sigma_{Y_c}^2}\right),$$

Person	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033*10-02	176.25	1.2292*10+02	11.25	9.1667*10-01
female	5.4175	9.7225*10-02	132.5	5.5833*10+02	7.5	1.6667

Person	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

$p(\text{weight} \mid \text{male}) = 5.9881 \cdot 10^{-6}$
$p(\text{foot size} \mid \text{male}) = 1.3112 \cdot 10^{-3}$
$\text{posterior numerator (male)} = \text{their product} = 6.1984 \cdot 10^{-9}$
$P(\text{female}) = 0.5$
$p(\text{height} \mid \text{female}) = 2.2346 \cdot 10^{-1}$
$p(\text{weight} \mid \text{female}) = 1.6789 \cdot 10^{-2}$
$p(\text{foot size} \mid \text{female}) = 2.8669 \cdot 10^{-1}$
$\text{posterior numerator (female)} = \text{their product} = 5.3778 \cdot 10^{-4}$

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

$$P(Y_c|X_1,\ldots,X_n) = \frac{P(Y_c)\prod_{i=1}^{n} P(X_i|Y_c)}{\prod_{i=1}^{n} P(X_i)} \quad Y_c \text{ is a label}$$

$$P(x_i \mid Y_c) = \frac{1}{\sqrt{2\pi\sigma_{Y_c}^2}} \exp\left(-\frac{(x_i - \mu_{Y_c})^2}{2\sigma_{Y_c}^2}\right)$$

Human knowledge belongs to the world.

Naïve Bayes with Sklearn

Naive Bayes Classifier

Director of TEAMLAB
Sungchul Choi

CountVectorizer in Scikit-learn

- 문서에서 Bag of Words Vector를 뽑아주는 class

```
class sklearn.feature_extraction.text. CountVectorizer (input='content', encoding='utf-8', decode_error='strict',
strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, stop_words=None, token_pattern='(?
u)\b\w\w+\b', ngram_range=(1, 1), analyzer='word', max_df=1.0, min_df=1, max_features=None, vocabulary=None,
binary=False, dtype=<class 'numpy.int64'>) ¶                                                          [source]
```

http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

CountVectorizer in Scikit-learn

- 다른 전처리 모듈 처럼 생성 → 적용의 과정을 거침

```
Y_example = ["Sports", "Not sports","Sports","Sports","Not sports"]
text_example = ["A great game", "The election was over","Very clean match",
                "A clean but forgettable game","It was a close election", ]

countvect_example =  CountVectorizer()
X_example = countvect_example.fit_transform(text_example)
countvect_example.get_feature_names()[:8]
```

```
['but', 'clean', 'close', 'election', 'forgettable', 'game', 'great', 'it']
```

CountVectorizer in Scikit-learn

- 다른 전처리 모듈 처럼 생성 → 적용의 과정을 거침

```
X_example.toarray()
```

```
array([[0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 2, 0, 1],
       [0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0],
       [1, 1, 0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1]], dtype=int64)
```

```
y_example
```

```
[1, 0, 1, 1, 0]
```

NB classifier family in scikit-learn

- Scikit-learn에서 제공하는 NB classifier
- Bernoulli Naïve Bayes
- Multinomial Naïve Bayes
- Gaussian Naïve Bayes

Read more in the User Guide.

Parameters:

alpha : float, optional (default=1.0)

Additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing).

binarize : float or None, optional (default=0.0)

Threshold for binarizing (mapping to booleans) of sample features. If None, input is presumed to already consist of binary vectors.

fit_prior : boolean, optional (default=True)

Whether to learn class prior probabilities or not. If false, a uniform prior will be used.

class_prior : array-like, size=[n_classes,], optional (default=None)

Prior probabilities of the classes. If specified the priors are not adjusted according to the data.

Attributes:

class_log_prior_ : array, shape = [n_classes]

Log probability of each class (smoothed).

feature_log_prob_ : array, shape = [n_classes, n_features]

Empirical log probability of features given a class, P(x_i|y).

class_count_ : array, shape = [n_classes]

Number of samples encountered for each class during fitting. This value is weighted by the sample weight when provided.

feature_count_ : array, shape = [n_classes, n_features]

Number of samples encountered for each (class, feature) during fitting. This value is weighted by the sample weight when provided.

Bernoulli Naïve Bayes

```
from sklearn.naive_bayes import BernoulliNB

clf = BernoulliNB(binarize=0)
clf.fit(X_example, y_example)
clf.class_log_prior_
```

```
array([-0.91629073, -0.51082562])
```

```
clf.class_count_
```

```
array([ 2.,  3.])
```

```
clf.feature_log_prob_
```

```
array([[-1.38629436, -1.38629436, -0.69314718, -0.28768207, -1.38629436,
        -1.38629436, -1.38629436, -0.69314718, -1.38629436, -0.69314718,
        -0.69314718, -1.38629436, -0.28768207],
       [-0.91629073, -0.51082562, -1.60943791, -1.60943791, -0.91629073,
        -0.22314355, -0.91629073, -1.60943791, -0.91629073, -1.60943791,
        -1.60943791, -0.91629073, -1.60943791]])
```

```
clf.feature_count_
```

```
array([[ 0.,  0.,  1.,  2.,  0.,  0.,  0.,  1.,  0.,  1.,  1.,  0.,  2.],
       [ 1.,  2.,  0.,  0.,  1.,  3.,  1.,  0.,  1.,  0.,  0.,  1.,  0.]])
```

Parameters:

alpha : float, optional (default=1.0)

Additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing).

fit_prior : boolean, optional (default=True)

Whether to learn class prior probabilities or not. If false, a uniform prior will be used.

class_prior : array-like, size (n_classes,), optional (default=None)

Prior probabilities of the classes. If specified the priors are not adjusted according to the data.

Attributes:

class_log_prior_ : array, shape (n_classes,)

Smoothed empirical log probability for each class.

intercept_ : property

Mirrors `class_log_prior_` for interpreting MultinomialNB as a linear model.

feature_log_prob_ : array, shape (n_classes, n_features)

Empirical log probability of features given a class, `P(x_i|y)`.

coef_ : property

Mirrors `feature_log_prob_` for interpreting MultinomialNB as a linear model.

class_count_ : array, shape (n_classes,)

Number of samples encountered for each class during fitting. This value is weighted by the sample weight when provided.

feature_count_ : array, shape (n_classes, n_features)

Number of samples encountered for each (class, feature) during fitting. This value is weighted by the sample weight when provided.

```
from sklearn.naive_bayes import MultinomialNB
```

```
clf = MultinomialNB()
clf.fit(X_example, y_example)
```

```
print(clf.class_log_prior_)
print(clf.feature_log_prob_)
print(clf.class_count_)
print(clf.feature_count_)

print(clf.coef_)
print(clf.intercept_)
```

```
[-0.91629073 -0.51082562]
[[-3.09104245 -3.09104245 -2.39789527 -1.99243016 -3.09104245 -3.09104245
  -3.09104245 -2.39789527 -3.09104245 -2.39789527 -1.99243016 -3.09104245
  -1.99243016]
 [-2.52572864 -2.12026354 -3.21887582 -3.21887582 -2.52572864 -1.42711636
  -2.52572864 -3.21887582 -2.52572864 -3.21887582 -3.21887582 -2.52572864
  -3.21887582]]
[ 2.  3.]
[[ 0.  0.  1.  2.  0.  0.  0.  1.  0.  1.  2.  0.  2.]
 [ 1.  2.  0.  0.  1.  5.  1.  0.  1.  0.  0.  1.  0.]]
[[-2.52572864 -2.12026354 -3.21887582 -3.21887582 -2.52572864 -1.42711636
  -2.52572864 -3.21887582 -2.52572864 -3.21887582 -3.21887582 -2.52572864
  -3.21887582]]
[-0.51082562]
```

Gaussian Naïve Bayes

Read more in the User Guide.

Parameters:	**priors** : array-like, shape (n_classes,) Prior probabilities of the classes. If specified the priors are not adjusted according to the data.
Attributes:	**class_prior_** : array, shape (n_classes,) probability of each class. **class_count_** : array, shape (n_classes,) number of training samples observed in each class. **theta_** : array, shape (n_classes, n_features) mean of each feature per class **sigma_** : array, shape (n_classes, n_features) variance of each feature per class

```
from sklearn.naive_bayes import GaussianNB
```

```
clf = GaussianNB()
clf.fit(X_example.toarray(), y_example)
```

```
GaussianNB(priors=None)
```

```
print(clf.class_count_)
print(clf.class_prior_)
print(clf.theta_)
print(clf.sigma_)
```

```
[ 2.  3.]
[ 0.4  0.6]
[[ 0.          0.          0.5         1.          0.          0.          0.
   0.5         0.          0.5         1.          0.          1.        ]
 [ 0.33333333  0.66666667  0.          0.          0.33333333  1.66666667
   0.33333333  0.          0.33333333  0.          0.          0.33333333
   0.        ]]
[[  8.00000000e-10   8.00000000e-10   2.50000001e-01   8.00000000e-10
    8.00000000e-10   8.00000000e-10   8.00000000e-10   2.50000001e-01
    8.00000000e-10   2.50000001e-01   1.00000000e+00   8.00000000e-10
    8.00000000e-10]
 [  2.22222223e-01   2.22222223e-01   8.00000000e-10   8.00000000e-10
    2.22222223e-01   2.22222223e-01   2.22222223e-01   8.00000000e-10
    2.22222223e-01   8.00000000e-10   8.00000000e-10   2.22222223e-01
    8.00000000e-10]]
```

Human knowledge belongs to the world.

News group classification

Naive Bayes Classifier

Director of TEAMLAB
Sungchul Choi

20newsgroups Dataset

- 대표적인 Text 분류 Toy dataset
- 20개의 뉴스 텍스트 데이터를 분류하라!
- Multiclass classification의 대표적 문제
- 약 20,000개의 news document 존재

http://qwone.com/~jason/20Newsgroups/

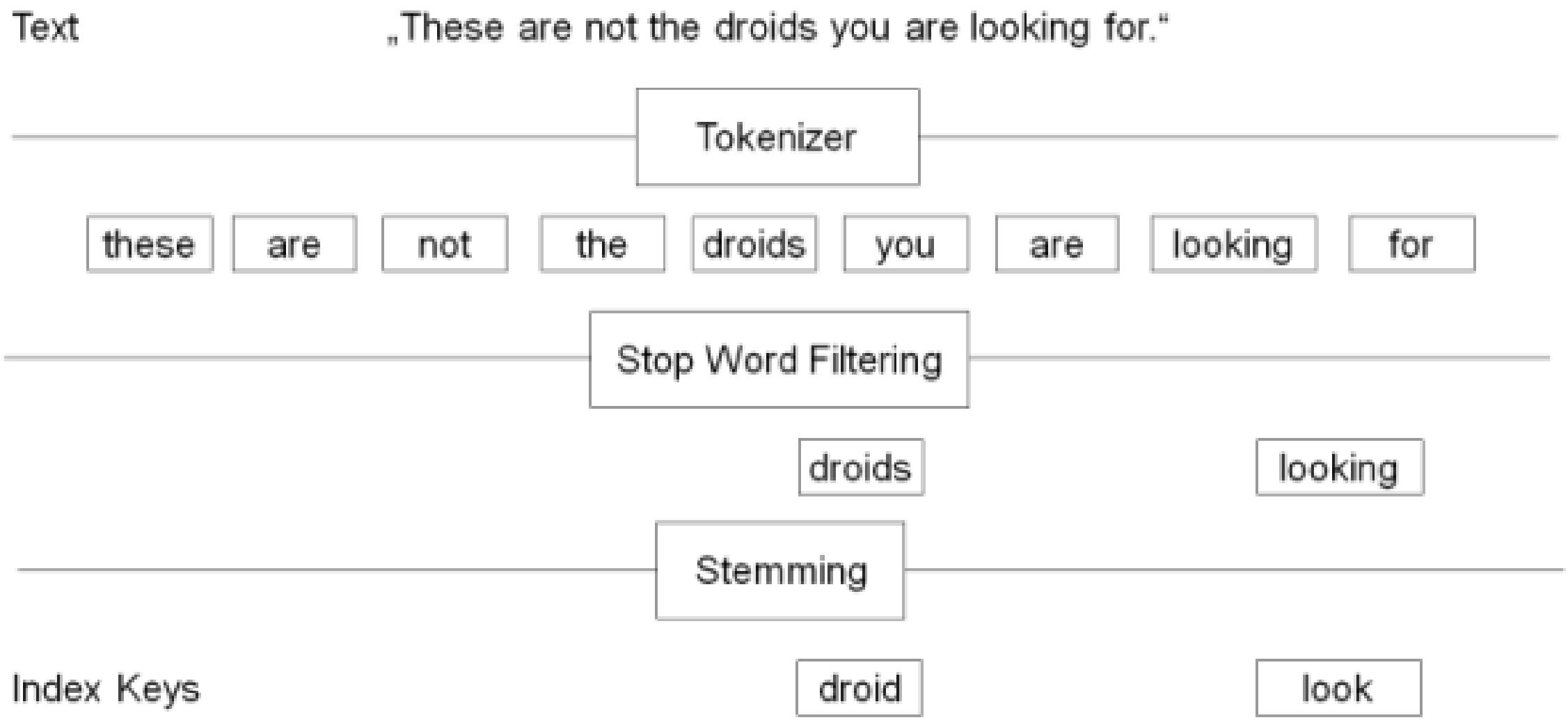
20newsgroups Dataset

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

http://qwone.com/~jason/20Newsgroups/

Process for text classification

데이터 준비	텍스트 전처리	Vectorizer	학습	모델결정
- 데이터 로딩 - 데이터 Tagging	- 데이터 cleansing - Tokenization - Stopword 제거 - Stemming	<u>Hyper parameter</u> - Ngrams - Threadshold <u>Vector</u> - Bag of words - TF-IDF	- Set metric - Train/Test Split - Hyper parameter	- Choose best model



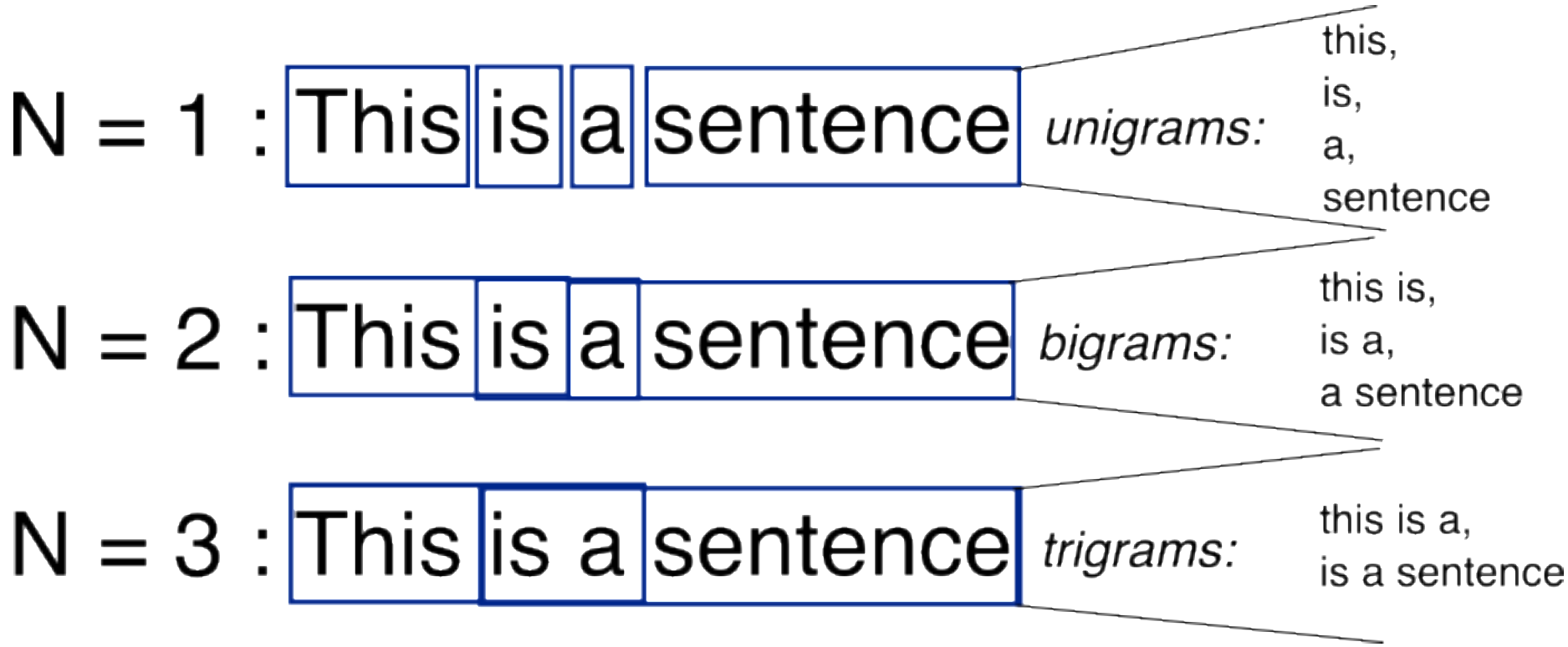
https://blog.codecentric.de/en/2013/01/text-search-mongodb-stemming/

Process for text classification

데이터 준비	텍스트 전처리	Vectorizer	학습	모델결정
- 데이터 로딩 - 데이터 Tagging	- 데이터 cleansing - Tokenization - Stopword 제거 - Stemming	<u>Hyper parameter</u> - Ngrams - Threadshold <u>Vector</u> - Bag of words - TF-IDF	- Set metric - Train/Test Split - Hyper parameter	- Choose best model

Concepts of ngrams

- 한번에 몇 개드이 단어를 묶을 것인가?



https://medium.com/algorithms-data-structures/ngrams-nedir-74c30162c6da

Process for text classification

데이터 준비	텍스트 전처리	Vectorizer	학습	모델결정
- 데이터 로딩 - 데이터 Tagging	- 데이터 cleansing - Tokenization - Stopword 제거 - Stemming	<u>Hyper parameter</u> - Ngrams - Threadshold <u>Vector</u> - Bag of words - TF-IDF	- Set metric - Train/Test Split - Hyper parameter	- Choose best model

TF-IDF

- 전체 문서에서 많이 나오는 단어는 중요도를 낮추고
- 해당 문서에서만 많이 나오면 중요도를 올림

$$w_{i,j} = tf_{i,j} \times \log(\frac{N}{df_i})$$

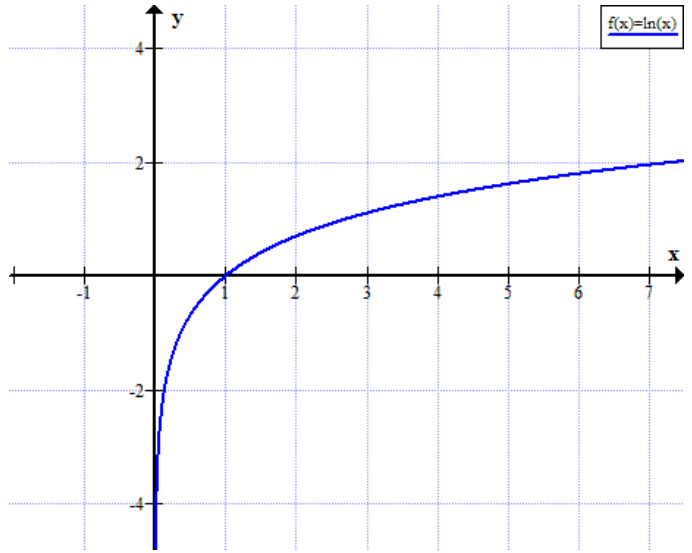
- i 단어의 index, j 문서의 index
- $tf_{i,j}$ Term frequency, j 문서에 i 단어가 존재한 횟수
- N 전체 문서
- df_i i 단어가 있는 문서의 개수

TF-IDF

$$w_{i,j} = tf_{i,j} \times \log(\frac{N}{df_i})$$

- i 단어의 index, j 문서의 index
- $tf_{i,j}$ Term frequency, j 문서에 i 단어가 존재한 횟수
- N 전체 문서
- df_i i 단어가 있는 문서의 개수

	Doc1	Doc2	Doc3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17



Process for text classification

데이터 준비	텍스트 전처리	Vectorizer	학습	모델결정
- 데이터 로딩 - 데이터 Tagging	- 데이터 cleansing - Tokenization - Stopword 제거 - Stemming	<u>Hyper parameter</u> - Ngrams - Threadshold <u>Vector</u> - Bag of words - TF-IDF	- Set metric - Train/Test Split - Hyper parameter	- Choose best model

Data loading

- Scikit-learn 내부 모듈을 활용함

```
from sklearn.datasets import fetch_20newsgroups
news = fetch_20newsgroups(subset='all')
news.keys()
```

```
dict_keys(['data', 'filenames', 'target_names', 'target', 'DESCR', 'description'])
```

Data loading

```
print(news.data[0])
```

```
From: Mamatha Devineni Ratnam <mr47+@andrew.cmu.edu>
Subject: Pens fans reactions
Organization: Post Office, Carnegie Mellon, Pittsburgh, PA
Lines: 12
NNTP-Posting-Host: po4.andrew.cmu.edu



I am sure some bashers of Pens fans are pretty confused about the lack
of any kind of posts about the recent Pens massacre of the Devils. Actually,
I am  bit puzzled too and a bit relieved. However, I am going to put an end
to non-PIttsburghers' relief with a bit of praise for the Pens. Man, they
are killing those Devils worse than I thought. Jagr just showed you why
he is much better than his regular season stats. He is also a lot
fo fun to watch in the playoffs. Bowman should let JAgr have a lot of
fun in the next couple of games since the Pens are going to beat the pulp out of Jersey anyway. I
ed not to see the Islanders lose the final
regular season game.          PENS RULE!!!
```

Data loading

```
news.target[0]
```

```
10
```

I am sure some bashers of Pens fans are pretty confused about the lack of any kind of posts about the recent Pens massacre of the Devils. Actually, I am bit puzzled too and a bit relieved. However, I am going to put an end to non-PIttsburghers' relief with a bit of praise for the Pens. Man, they are killing those Devils worse than I thought. Jagr just showed you why he is much better than his regular season stats. He is also a lot fo fun to watch in the playoffs. Bowman should let JAgr have a lot of fun in the next couple of games since the Pens are going to beat the pulp out of Jersey anyway. I was very disappointed not to see the Islanders lose the final regular season game. PENS RULE!!!

```
news.target_names
```

```
['alt.atheism',
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware',
 'comp.sys.mac.hardware',
 'comp.windows.x',
 'misc.forsale',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.electronics',
 'sci.med',
 'sci.space',
 'soc.religion.christian',
 'talk.politics.guns',
 'talk.politics.mideast',
 'talk.politics.misc',
 'talk.religion.misc']
```

Process for text classification

데이터 준비	텍스트 전처리	Vectorizer	학습	모델결정
- 데이터 로딩 - 데이터 Tagging	- 데이터 cleansing - Tokenization - Stopword 제거 - Stemming	<u>Hyper parameter</u> - Ngrams - Threadshold <u>Vector</u> - Bag of words - TF-IDF	- Set metric - Train/Test Split - Hyper parameter	- Choose best model

Data cleansing

- 어떤 Text를 남길 것 인가?
- 특수 문자는 남겨 둘 것인가? Good vs. Good?
- 숫자는 어떻게 할 것 인가? 4 seasosn vs. seasons
- 특수문자를 제거한다면 공백은? I'm → Im or I m
- 이메일, ip주소등 특수 문자들의 처리는?

Data cleansing

```
def data_cleansing(df):
    delete_email = re.sub(r'\b[\w\+]+@[\w]+.[\w]+.[\w]+.[\w]+\b', ' ', df)
    delete_number = re.sub(r'\b|\d+|\b', ' ',delete_email)
    delete_non_word = re.sub(r'\b[\W]+\b', ' ', delete_number)
    cleaning_result = ' '.join(delete_non_word.split())
    return cleaning_result
```

```
news_df.loc[:, 'News'] = news_df['News'].apply(data_cleansing)
news_df.head()
```

	News	Target
0	From Mamatha Devineni Ratnam Subject Pens fans...	rec.sport.hockey
1	From Matthew B Lawson Subject Which high perfo...	comp.sys.ibm.pc.hardware
2	From hilmi Hilmi Eren Subject Re ARMENIA SAYS ...	talk.politics.mideast
3	From Guy Dawson Subject Re IDE vs SCSI DMA and...	comp.sys.ibm.pc.hardware
4	From Alexander Samuel McDiarmid Subject driver...	comp.sys.mac.hardware

Process for text classification

데이터 준비	텍스트 전처리	Vectorizer	학습	모델결정
- 데이터 로딩 - 데이터 Tagging	- 데이터 cleansing - Tokenization - Stopword 제거 - Stemming	Hyper parameter - Ngrams - Threadshold Vector - Bag of words - TF-IDF	- Set metric - Train/Test Split - Hyper parameter	- Choose best model

sklearn.feature_extraction.text

- Scikit-learn text vector화 모듈
- Text 관련 처리를 Hyper parameter로 한번에 처리
- CountVectorizer, TfidfVectorizer 제공
- Stemmer 등 NLTK 등 외부모듈을 사용

CounterVectorizer

class `sklearn.feature_extraction.text.` **`CountVectorizer`** (*input='content', encoding='utf-8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), analyzer='word', max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.int64'>*) ¶ [source]

encoding : string, 'utf-8' by default.
decode_error : {'strict', 'ignore', 'replace'}
analyzer : string, {'word', 'char', 'char_wb'} or callable
preprocessor : callable or None (default)
tokenizer : callable or None (default)
ngram_range : tuple (min_n, max_n)
stop_words : string {'english'}, list, or None (default)
lowercase : boolean, True by default
max_df : float in range [0.0, 1.0] or int, default=1.0
min_df : float in range [0.0, 1.0] or int, default=1
binary : boolean, default=False

TfidfVectorizer

class `sklearn.feature_extraction.text.` **`TfidfVectorizer`** (*input='content', encoding='utf-8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, analyzer='word', stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.int64'>, norm='l2', use_idf=True, smooth_idf=True, sublinear_tf=False*) ¶ [source]

norm : 'l1', 'l2' or None, optional
use_idf : boolean, default=True
smooth_idf : boolean, default=True
sublinear_tf : boolean, default=False

Stammer 적용하기

- Stammer 또는 Lemma 를 적용하기 위해 nltk사용
- 기존 Vectorizer의 상속받아 새로운 클래스 만들기

Stemmer 적용하기

```
from nltk import stem
stmmer = stem.SnowballStemmer("english")
sentence = 'this is a foo bar sentences and i want to ngramize it'
[stmmer.stem(word) for word in sentence.split()]
```

```
['this',
 'is',
 'a',
 'foo',
 'bar',
 'sentenc',
 'and',
 'i',
 'want',
 'to',
 'ngramiz',
 'it']
```

```
stmmer.stem("images"), stmmer.stem("imaging"), stmmer.stem("imagination")
```

```
('imag', 'imag', 'imagin')
```

StemmedCountVectroizer

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

enlish_stemmer = nltk.stem.SnowballStemmer("english")
class StemmedCountVectorizer(CountVectorizer):
    def build_analyzer(self):
        analyzer = super(StemmedCountVectorizer,self).build_analyzer()
        return lambda doc: (enlish_stemmer.stem(w) for w in analyzer(doc))
```

```
StemmedCountVectorizer(min_df=1, stop_words="english").fit([sentence]).vocabulary_
```

```
{'bar': 0, 'foo': 1, 'ngramiz': 2, 'sentenc': 3, 'want': 4}
```

```
CountVectorizer(min_df=1, stop_words="english").fit([sentence]).vocabulary_
```

```
{'bar': 0, 'foo': 1, 'ngramize': 2, 'sentences': 3, 'want': 4}
```

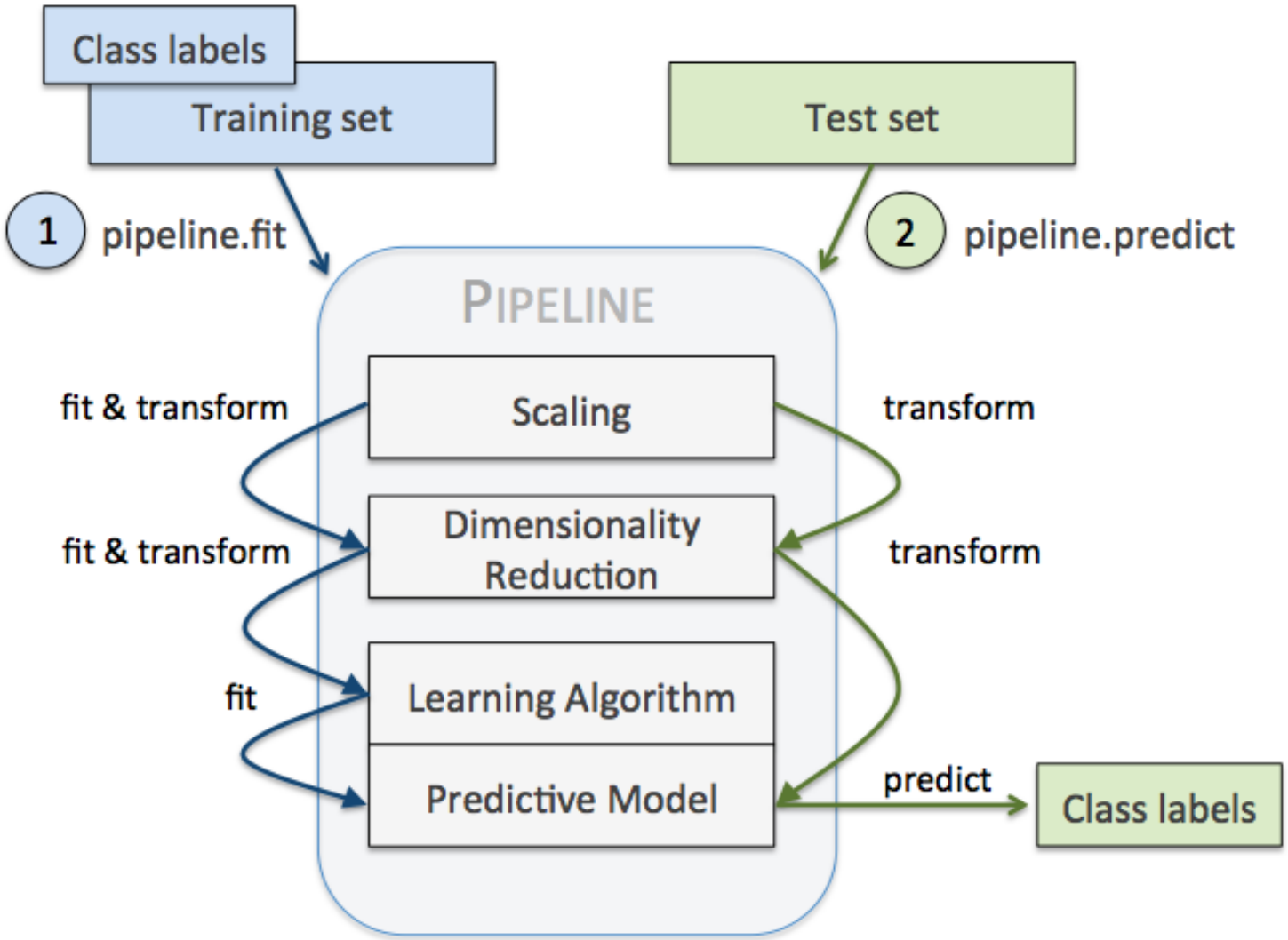
StemmedTfidfVectroizer

```
from sklearn.feature_extraction.text import TfidfVectorizer

enlish_stemmer = nltk.stem.SnowballStemmer("english")
class StemmedTfidfVectorizer(TfidfVectorizer):
    def build_analyzer(self):
        analyzer = super(StemmedTfidfVectorizer,self).build_analyzer()
        return lambda doc: (enlish_stemmer.stem(w) for w in analyzer(doc))
```

Scikit-learn Pipeline 모듈

- Vectorizer를 적용할 때는 Test / train에 한번에!
- 여러분 실험할 때 거쳐야 하는 과정을 정의
- Scikit-learn의 pipeline 모듈 사용

Scikit-learn Pipeline 모듈



In [9]:
```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB
newsgroups_train = fetch_20newsgroups(subset='train')

# For the sake of simplicity of the api we will convert indices of the categories to the actual labels
X = newsgroups_train.data
y = [newsgroups_train.target_names[i] for i in newsgroups_train.target]
```

In [10]:
```
pipeline = Pipeline([('vectorizer', TfidfVectorizer()), ('clf', MultinomialNB(0.01))])
classifier = pipeline.fit(X, y)
```

In [13]:
```
text_example = ['Angela Merkel just walked into her fourth term as chancellor of Germany. Her party, the
predictions = classifier.predict(text_example)
predictions
```

Out[13]:
```
array(['talk.politics.misc'],
      dtype='<U24')
```

In [14]:
```
from sklearn.externals import joblib
joblib.dump(classifier, 'model.pkl')
```

Out[14]:
```
['model.pkl']
```

https://medium.com/@elvin.valiev/5-minutes-from-machine-learning-to-rest-api-e8c6e508a370

Pipeline

```
pipeline = Pipeline([('vectorizer', TfidfVectorizer()), ('clf', MultinomialNB(0.01))])
classifier = pipeline.fit(X, y)
```

```
def fit(self, X, y):
    X_transformed = X
    for name, estimator in self.steps[:-1]:
        # iterate over all but the final step
        # fit and transform the data
        X_transformed = estimator.fit_transform(X_transformed, y)
    # fit the last step
    self.steps[-1][1].fit(X_transformed, y)
    return self
```

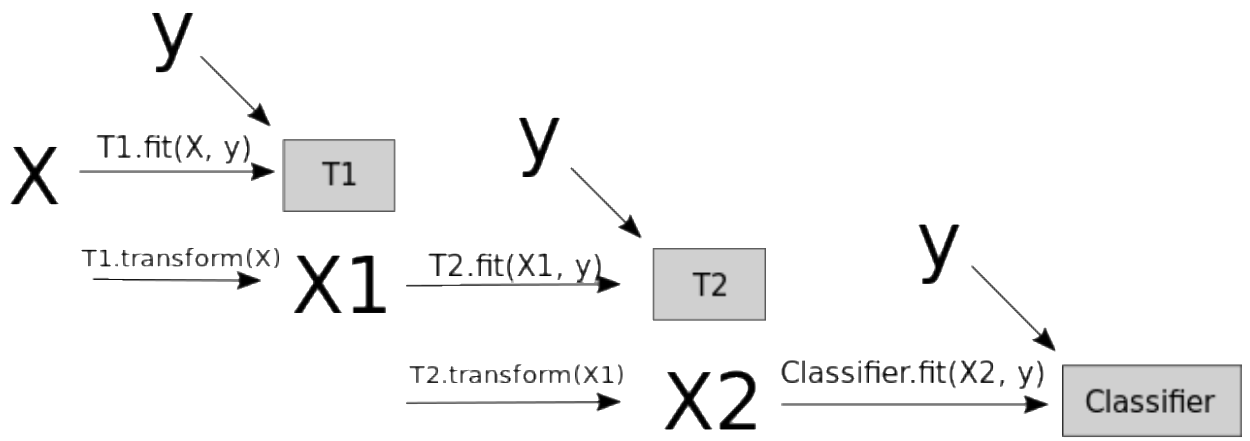
```
def predict(self, X):
    X_transformed = X
    for step in self.steps[:-1]:
        # iterate over all but the final step
        # transform the data
        X_transformed = step[1].transform(X_transformed)
    # predict using the last step
    return self.steps[-1][1].predict(X_transformed)
```

https://github.com/amueller/introduction_to_ml_with_python/blob/master/06-algorithm-chains-and-pipelines.ipynb

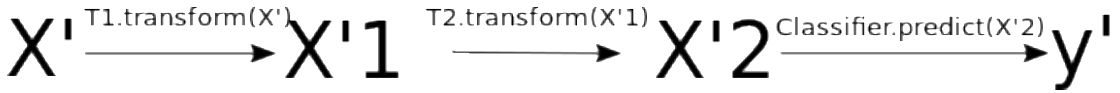
Pipeline

pipe = make_pipeline(T1(), T2(), Classifier())

pipe.fit(X, y)



pipe.predict(X')



https://github.com/amueller/introduction_to_ml_with_python/blob/master/06-algorithm-chains-and-pipelines.ipynb

Gridsearch

- Hyperparameter를 search 할때 사용
- Hyperparameter 변수 조합 자동 생성
- Pipeline 과 합쳐서 estimator 별로 parameter 설정
- GridsearchCV 등 샘플링 class 제공

Gridsearch

```
## GridSearchCV to come with best paramaters
from sklearn.grid_search import GridSearchCV
X = job_list[['is_dataScience', 'is_ml', 'is_python','is_stat', 'is_cs', 'is_visual']]
y = job_list['class']

logreg = LogisticRegression()
C_vals = [0.0001, 0.001, 0.01, 0.1, 0.5, 0.75, 1.0, 2.5, 5.0, 10.0, 100.0, 1000.0]
penalties = ['l1','l2']

gs = GridSearchCV(logreg, {'penalty':penalties, 'C':C_vals}, verbose=True, cv=5, scoring='accuracy')
gs.fit(X, y)
```

https://shettydatascience.wordpress.com/2016/10/31/collecting-data-by-scraping-a-website-and-building-a-binary-predictor-with-logistic-regression/

Pipeline + Gridsearch

```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import GridSearchCV

tfidf = TfidfVectorizer(strip_accents=None, lowercase=False, preprocessor=None)

param_grid = [{'vect__ngram_range': [(1, 1)],
               'vect__stop_words': [stop, None],
               'vect__tokenizer': [tokenizer, tokenizer_porter],
               'clf__penalty': ['l1', 'l2'],
               'clf__C': [1.0, 10.0, 100.0]},
              {'vect__ngram_range': [(1, 1)],
               'vect__stop_words': [stop, None],
               'vect__tokenizer': [tokenizer, tokenizer_porter],
               'vect__use_idf':[False],
               'vect__norm':[None],
               'clf__penalty': ['l1', 'l2'],
               'clf__C': [1.0, 10.0, 100.0]},
              ]

lr_tfidf = Pipeline([('vect', tfidf),
                     ('clf', LogisticRegression(random_state=0))])

gs_lr_tfidf = GridSearchCV(lr_tfidf, param_grid, scoring='accuracy',
                           cv=5, verbose=1, n_jobs=-1)

gs_lr_tfidf.fit(X_train, y_train)
```

Vect__attributename

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
```

Modeling Plan

Vectorizer

- Tfidf
- Count
- StemmedTfidf
- StemmedCount

Algorithm

- LogisticRegression
- Bernoulli NB
- Multinomial
- Gaussian NB

Metrics

- CV 5 times
- Accuracy
- Recall
- Precision
- F1

Human knowledge belongs to the world.