



인공지능

# Regression

김선녕([ksycafe@gmail.com](mailto:ksycafe@gmail.com))



**Linear Regression**  
Logistic Regression  
Softmax Regression

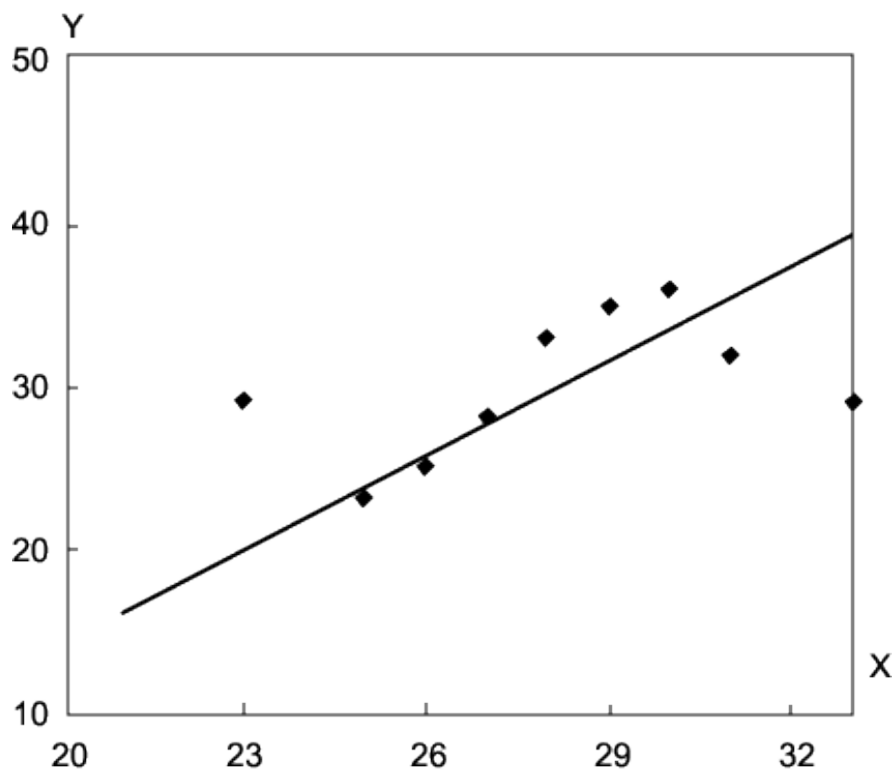
- 변수와 변수 사이의 상관관계를 알아보기 위한 통계적 분석방법
- 독립변수의 값에 의하여 종속변수의 값을 예측 및 측정한다
  - 독립변수(*independent variable, Predictor variable*): 종속변수에 영향을 미치는 변수.
  - 종속변수(*dependent variable, Target variable*): 분석의 대상이 되는 변수. 측정 혹은 예측결과
- 단순회귀분석(단변량, *simple regression analysis*)
  - 하나의 종속변수와 하나의 독립변수 관계를 분석
- 다중회귀분석(다변량, *multiple regression analysis*)
  - 하나의 종속변수와 둘 이상의 독립변수 관계를 분석

## Example

- 기온이 섭씨 1도 올라감에 따라 아이스크림 판매량에 미치는 영향(예측)은?

(단위 : 도, 십만 개)

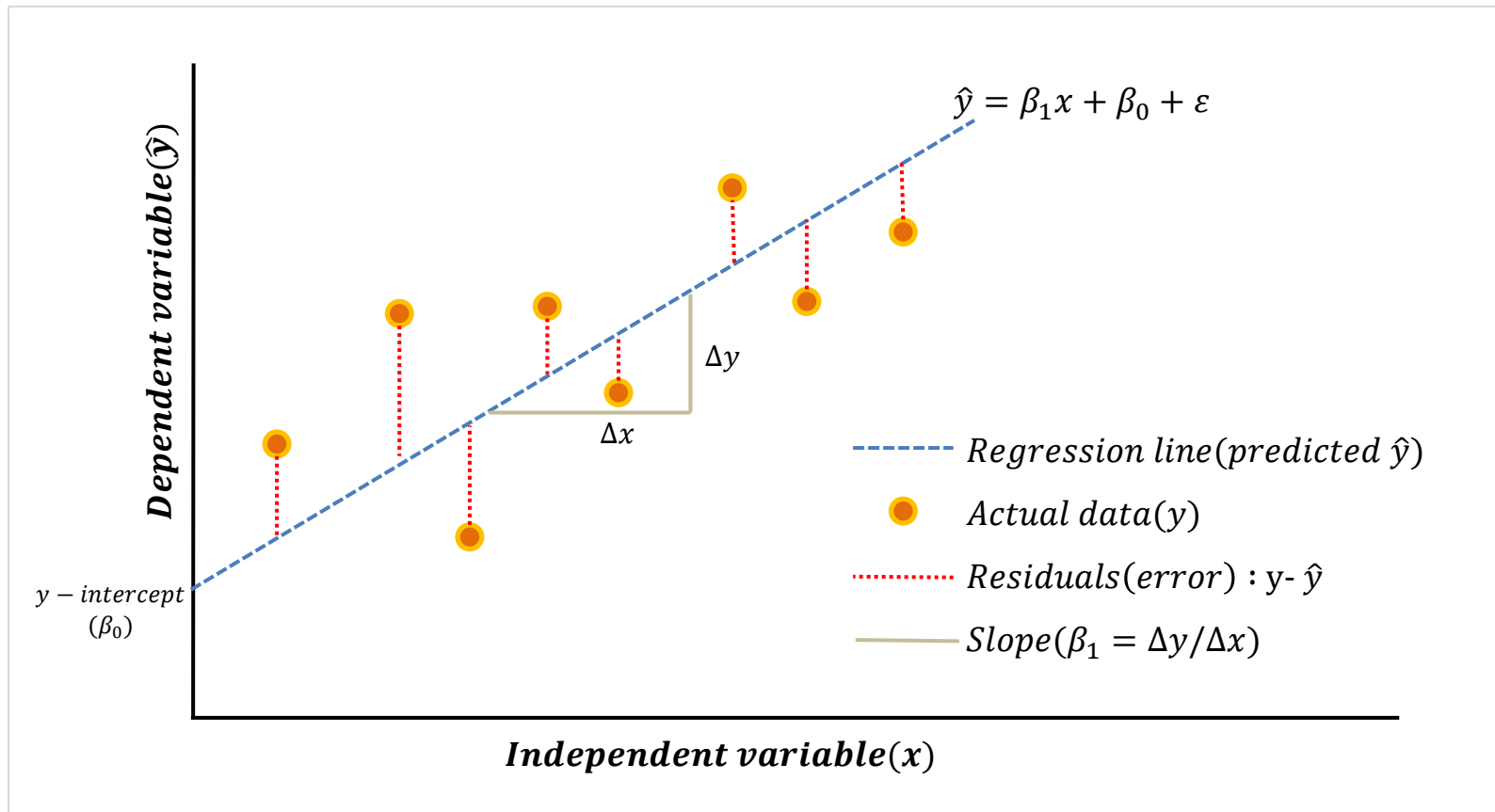
온 도 (X)	23	25	26	27	28	29	30	31	33
판매량 (Y)	29	23	25	28	33	35	36	32	29



$$\hat{y}_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

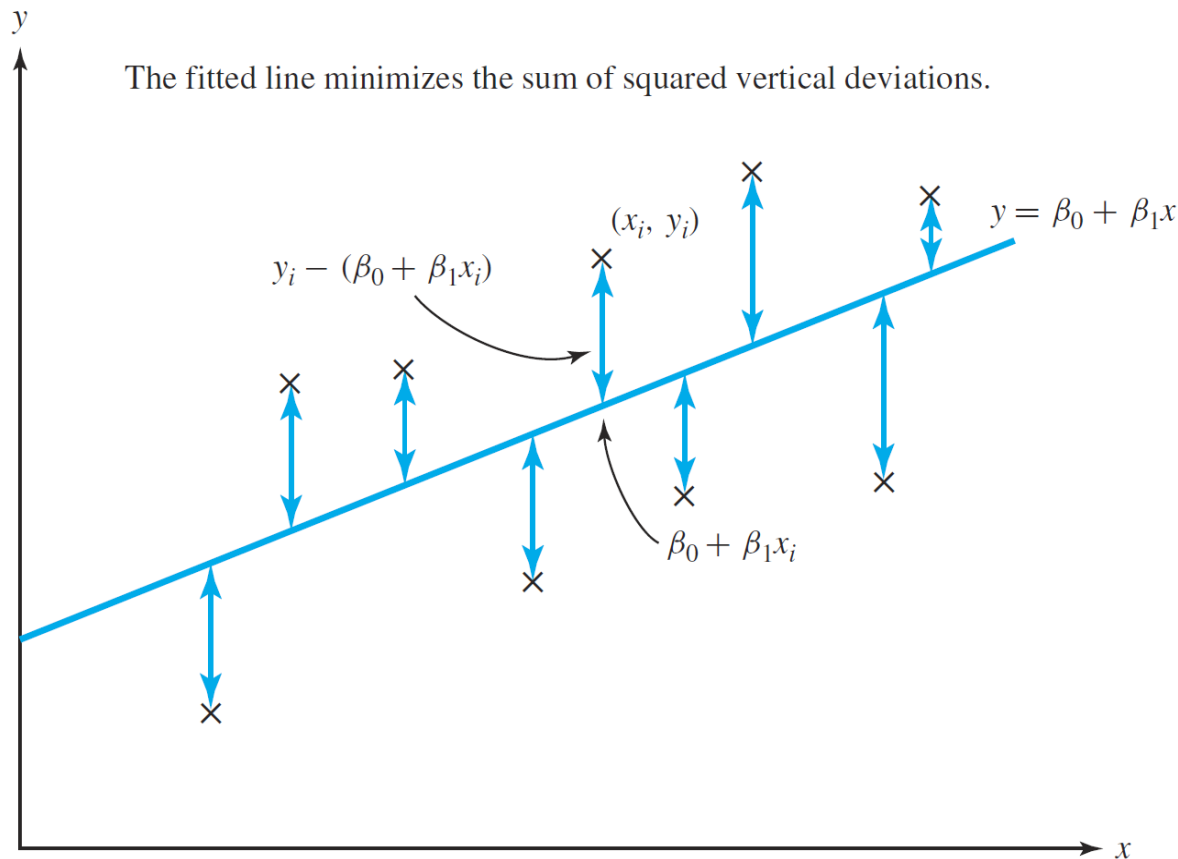
$\beta_0$  : 절편( $\hat{y}$  - intercept),  $\beta_1$  : 기울기(slope),  $\epsilon_i$  : 오차(error)

회귀분석의 목적은 최적의  $\beta_0, \beta_1$  를 찾는 것(최적의 회귀선)

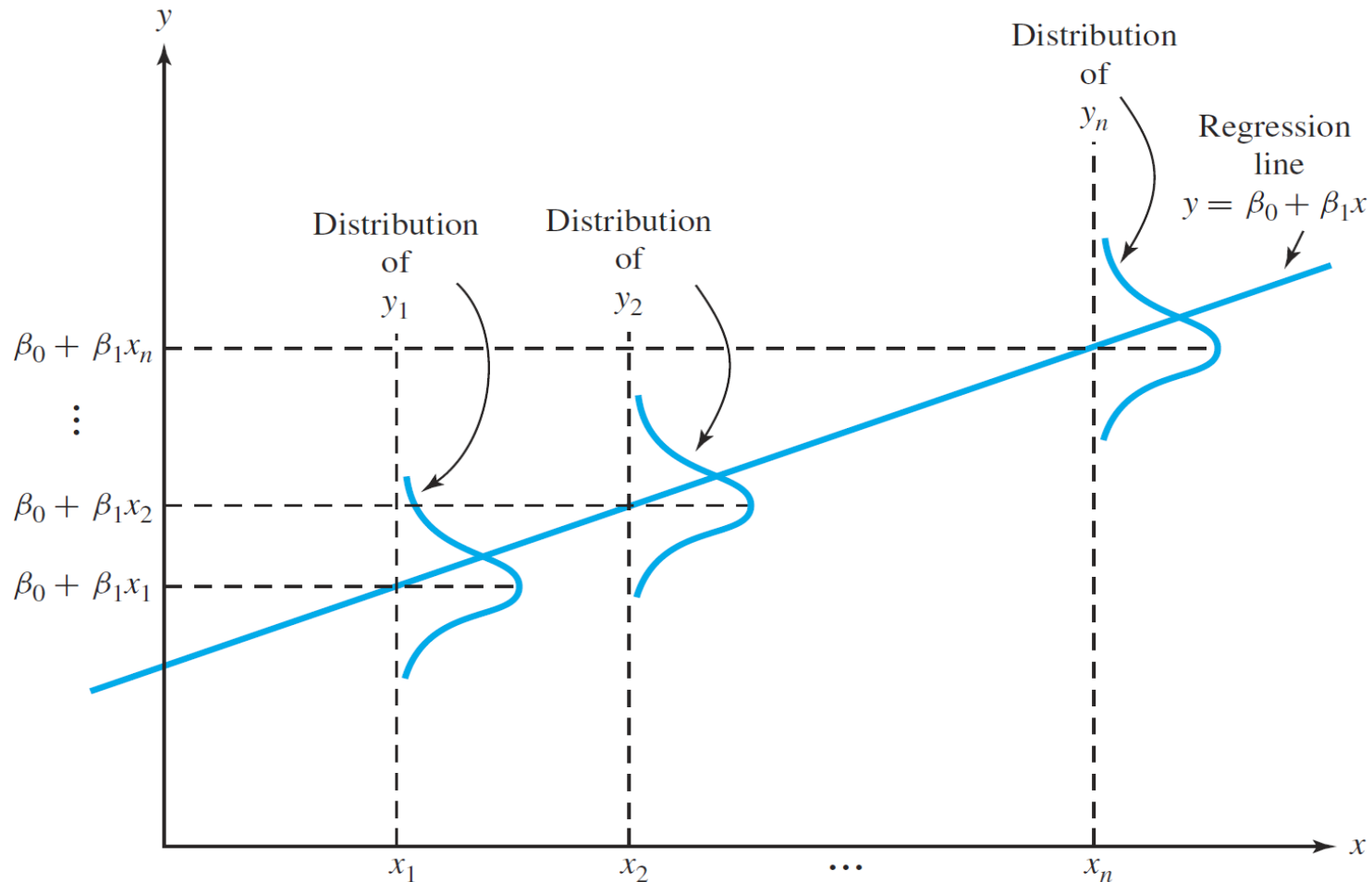


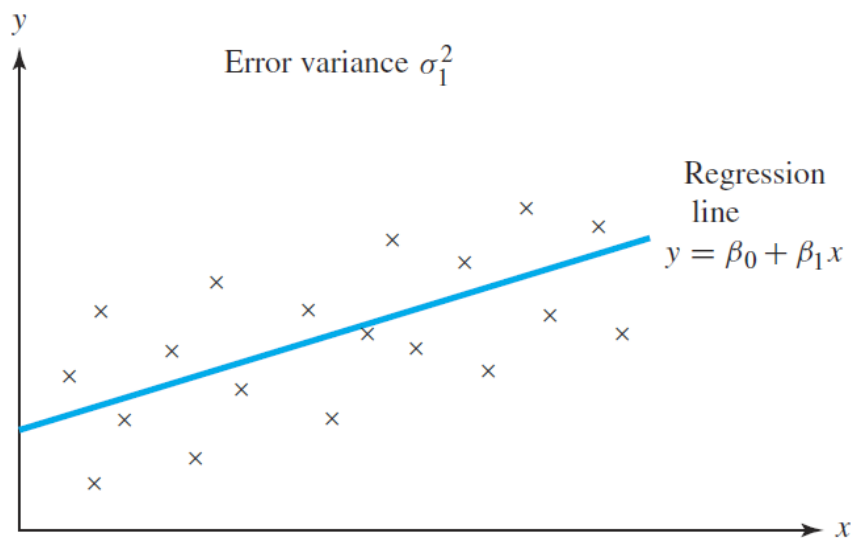
- 회귀직선  $y = \beta_0 + \beta_1 x$  를 데이터  $(x_1, y_1), \dots, (x_n, y_n)$ 에 적합(*fit*)시키는 과정
  - 데이터셋에 가장 근접한 직선을 찾는 과정. 즉, 오차(*Error*)를 최소화하는 직선을 선택하는 것

$$E = \sum_{i=1}^n (y_i - (\beta_0 + \beta_1 x_i))^2$$

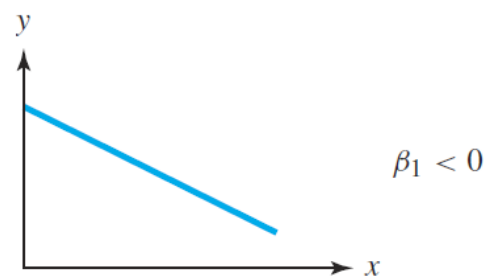
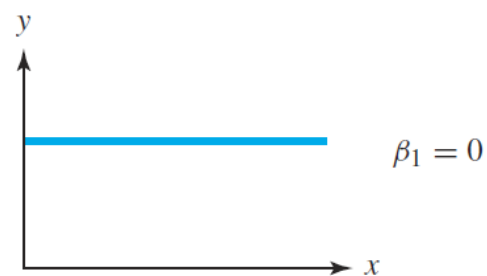
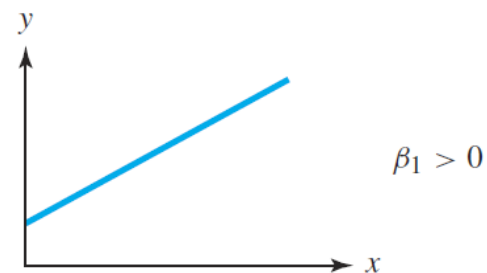
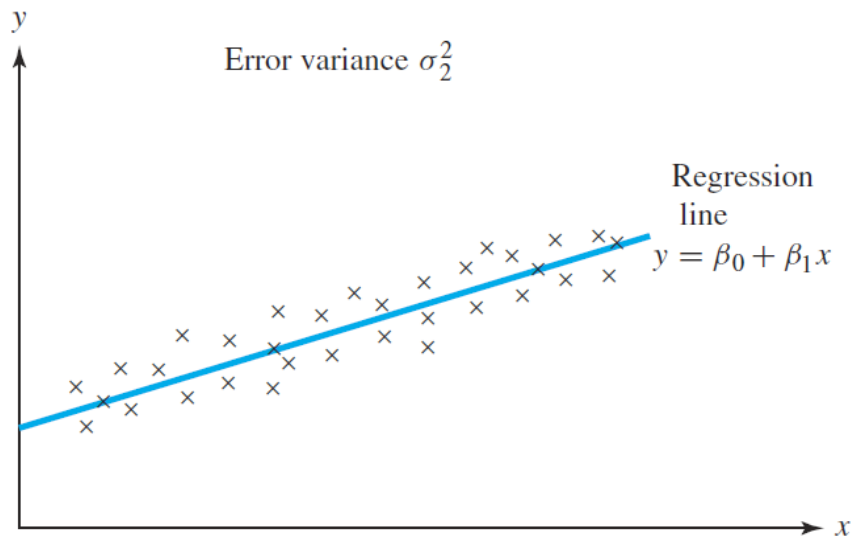


- 오차항  $\epsilon_1, \dots, \epsilon_n$  : 오차분산  $\sigma^2$ 에 대하여 정규분포 한다고 가정  $N(0, \sigma^2)$ 
  - 평균은 0이다. 실제 관찰 값이 회귀선상에 있는 점을 중심으로 분포되어 있다는 뜻
  - 오차분산  $\sigma^2$ 은 모든  $x$  값에 동일하다.
- ❖ 정규분포(normal distribution) :  $N(\mu, \sigma^2)$  (평균  $\mu$ , 분산  $\sigma^2$ )





$$\sigma_1^2 > \sigma_2^2$$

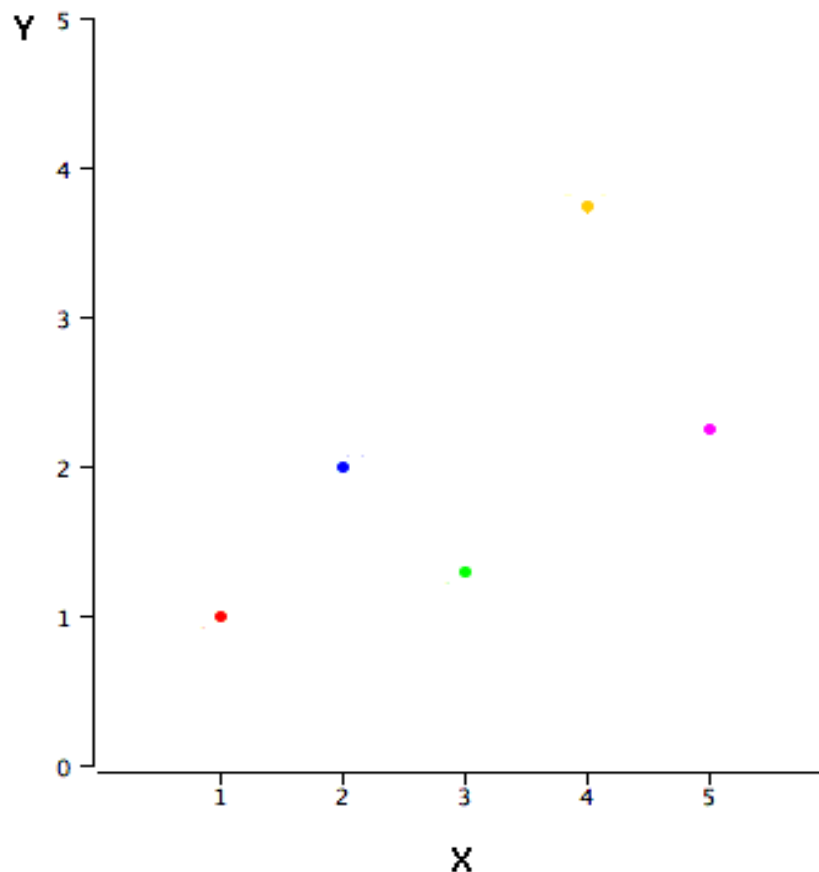




## Example2

- 어느 선이 예측 데이터의 최적의 선인가?
- $X$ 에서  $Y$ 를 예측한다면,  $X$ 의 값이 높을 수록  $Y$ 의 예측 값도 높다.
- $Y = aX + b$ , 기울기(slope) :  $a$ ,  $Y$ 절편(intercept) :  $b$

X	Y
1.00	1.00
2.00	2.00
3.00	1.30
4.00	3.75
5.00	2.25



- 상관계수 : 두 변수들 간의 연관 정도

- $m_x$  : mean of  $X$

- $m_y$  : mean of  $Y$

- $s_x$  : standard deviation of  $X$  :  $\sqrt{\frac{\sum_{i=1}^n (X_i - m_x)^2}{n-1}}$

- $s_y$  : standard deviation of  $Y$  :  $\sqrt{\frac{\sum_{i=1}^n (Y_i - m_y)^2}{n-1}}$

- $r$  : the correlation between  $X$  and  $Y$  (**Pearson's  $r$** ) :

$$r = \frac{\sum xy}{\sqrt{\sum x^2 \sum y^2}}, \quad (x = X - m_x, y = Y - m_y)$$

$m_x$	$m_y$	$s_x$	$s_y$	$r$
3	2.06	1.581	1.072	0.627

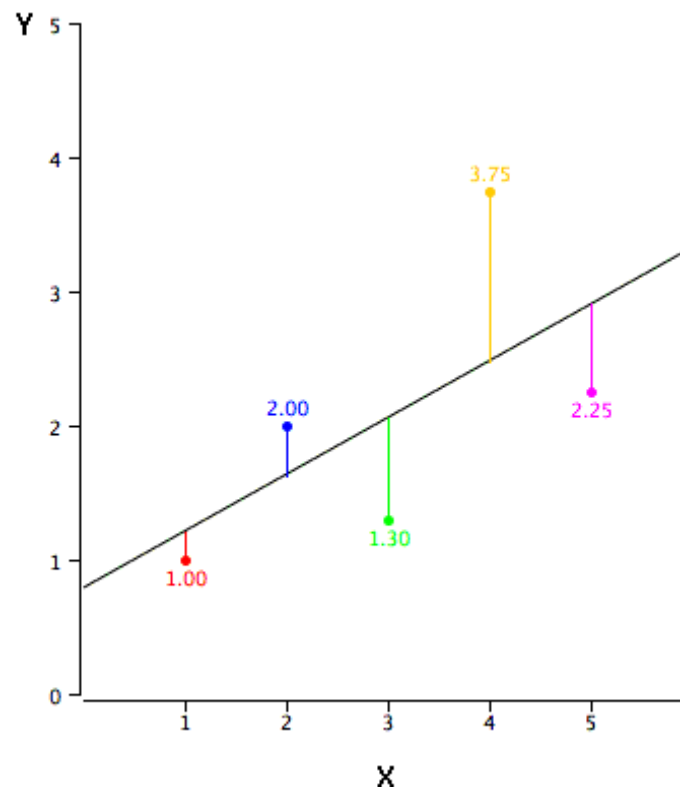
- $slope(a) = r \cdot s_x / s_y = (0.627) \cdot (1.072) / 1.581 = 0.425$
- $intercept(b) = m_y - a \cdot m_x = 2.06 - (0.425)(3) = 0.785$
- $Y = 0.425X + 0.786$**

- $\hat{Y}$  : predicted values
- $Y - \hat{Y}$  : errors of prediction
- $\hat{Y} = 0.425X + 0.786$
- *Best – fitting line*  
*minimizes the sum of the squared errors of prediction.*

$X$	$Y$	$\hat{Y}$	$Y - \hat{Y}$	$(Y - \hat{Y})^2$
1.00	1.00	1.210	-0.210	0.044
2.00	2.00	1.635	0.365	0.133
3.00	1.30	2.060	-0.760	0.578
4.00	3.75	2.485	1.265	1.600
5.00	2.25	2.910	-0.660	0.436

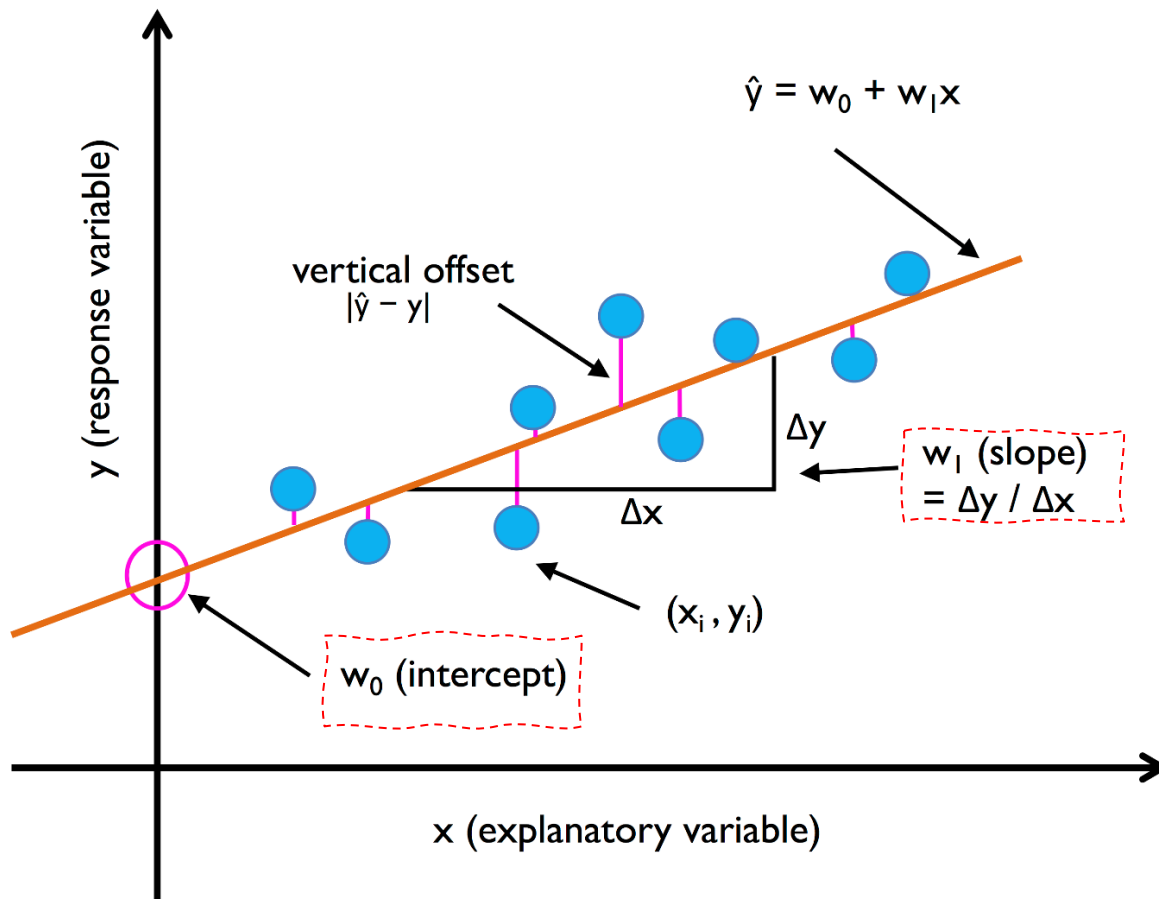
$$\frac{(\hat{y}^{(1)} - y^{(1)})^2 + (\hat{y}^{(2)} - y^{(2)})^2 + \dots + (\hat{y}^{(5)} - y^{(5)})^2}{5}$$

$$= \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2$$



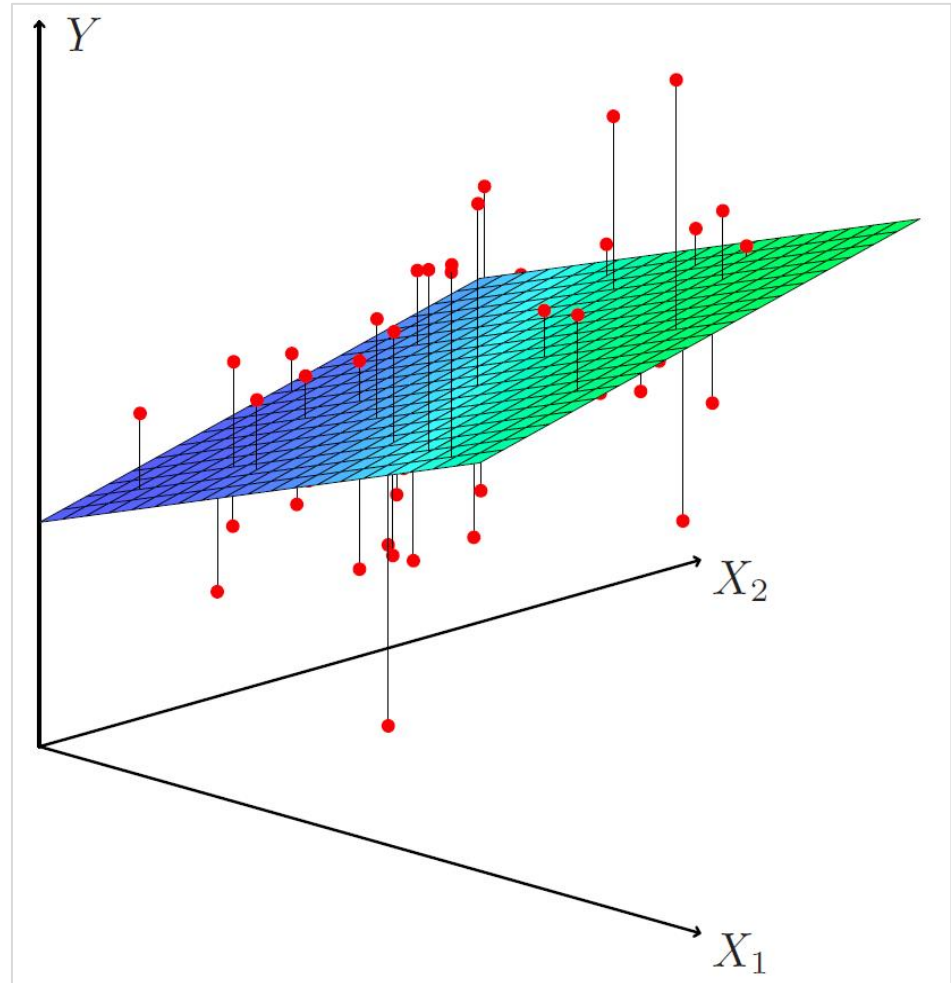
- 하나의 특성(*explanatory variable, x*)과 연속적인 타겟(*response variable, y*) 사이의 관계를 모델링

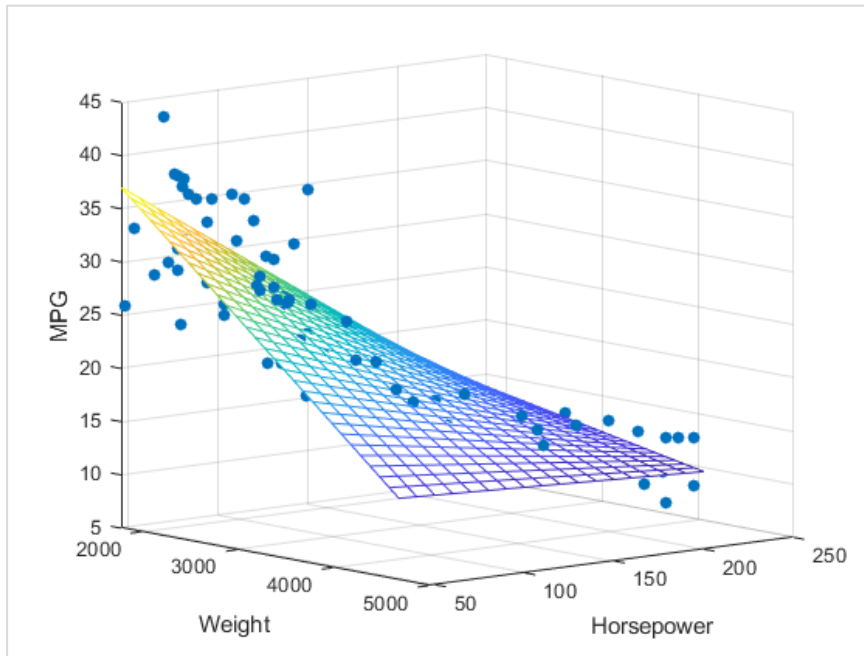
$$\hat{y} = w_0 + w_1 x$$



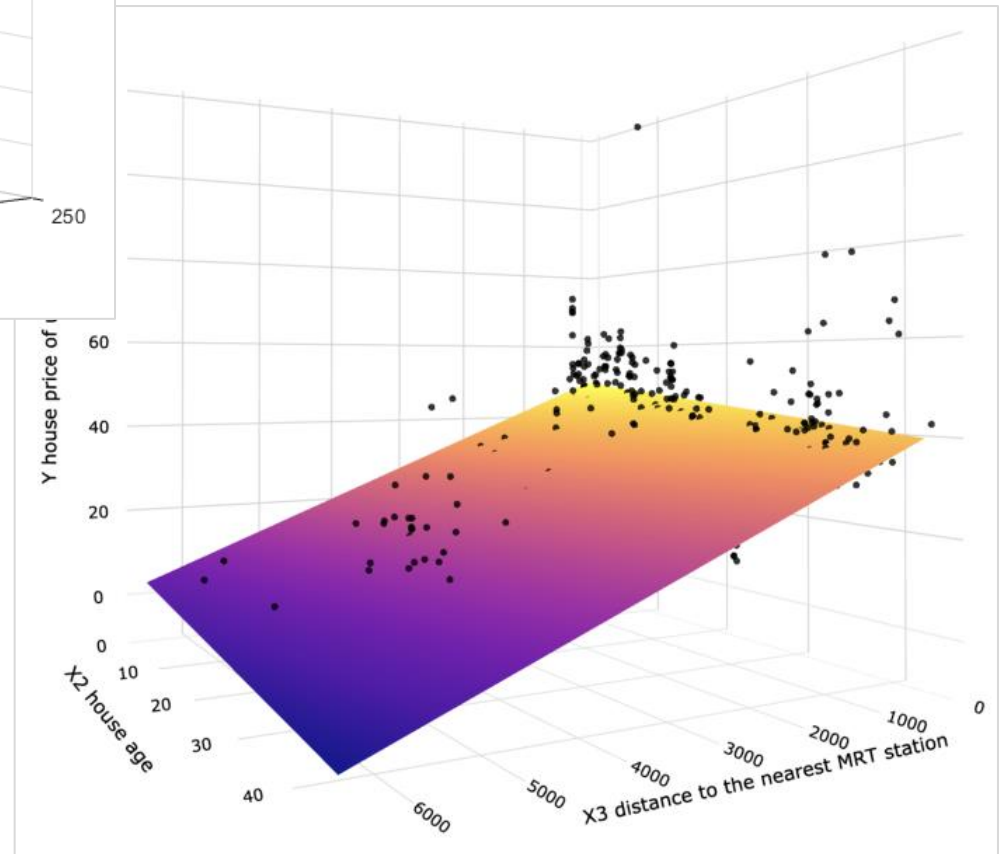
$$\hat{y} = [w_0 \quad w_1 \quad \cdots \quad w_m] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_m \end{bmatrix} = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m$$

$$= \sum_{i=0}^m w_i x_i = w^T x$$





<http://piramvill2.org/?p=3137>



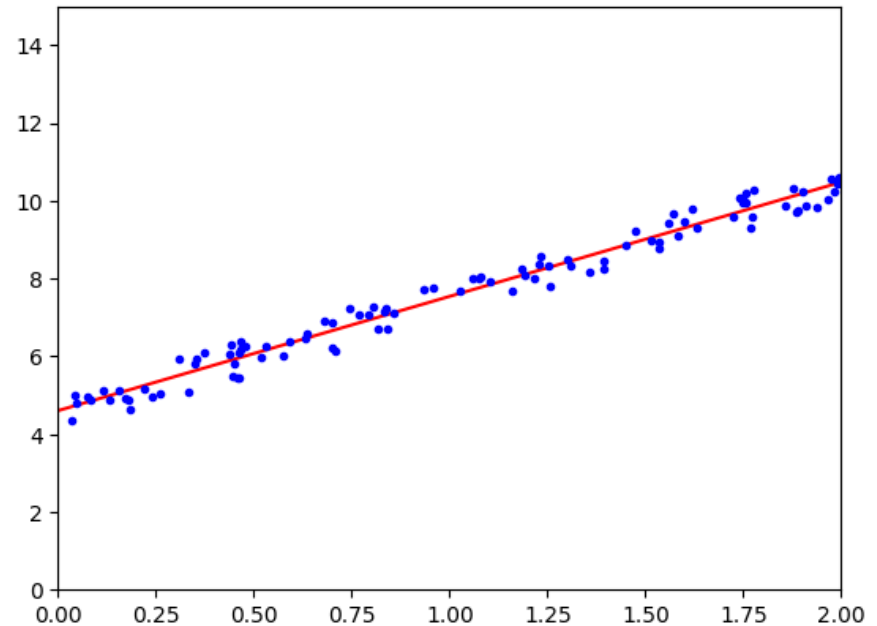
<https://towardsdatascience.com/linear-regression-made-easy-how-does-it-work-and-how-to-use-it-in-python-be0799d2f159>

```
import matplotlib.pyplot as plt
import numpy as np
X = 2 * np.random.rand(100,1)
y = 4 + 3 * X + np.random.rand(100,1)

X_b = np.c_[np.ones((100,1)), X] # 모든 샘플에 x0=1을 추가
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
theta_best # array([[4.42619872], [3.10195277]])

X_new = np.array([[0], [2]])
X_new_b = np.c_[np.ones((2,1)), X_new]
y_predict = X_new_b.dot(theta_best)
y_predict # array([[ 4.42619872], [10.63010426]])

plt.plot(X_new, y_predict, "r-")
plt.plot(X, y, "b.")
plt.axis([0, 2, 0, 15])
plt.show()
```



# 선형회귀 - LinearRegression

16

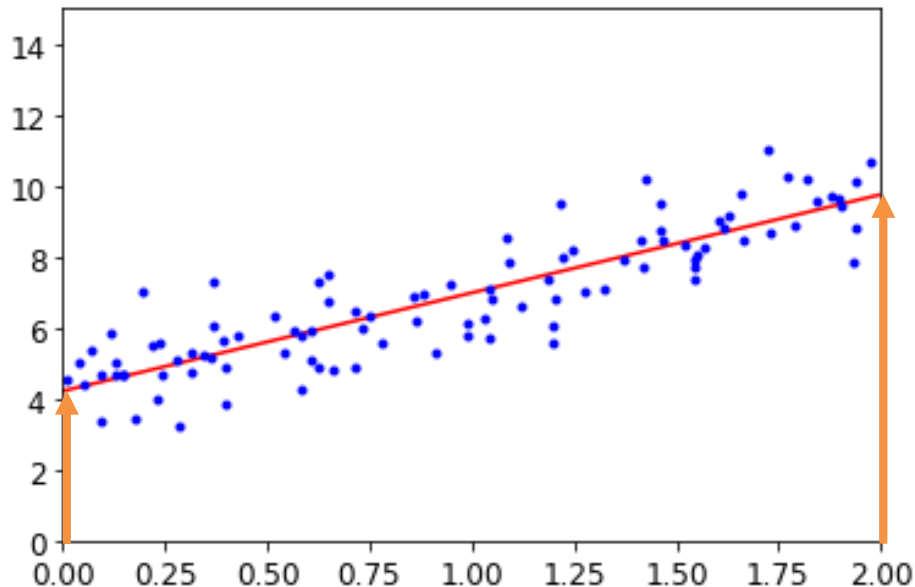
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X, y)
lin_reg.intercept_, lin_reg.coef_
```

$\beta_0, \beta_1$  (array([4.04967799]), array([[3.01093259]]))

```
X_new = np.array([[0], [2]])
lin_reg.predict(X_new)
```

[0], [2] 의 예측 결과 : array([[ 4.04967799], [10.07154317]])

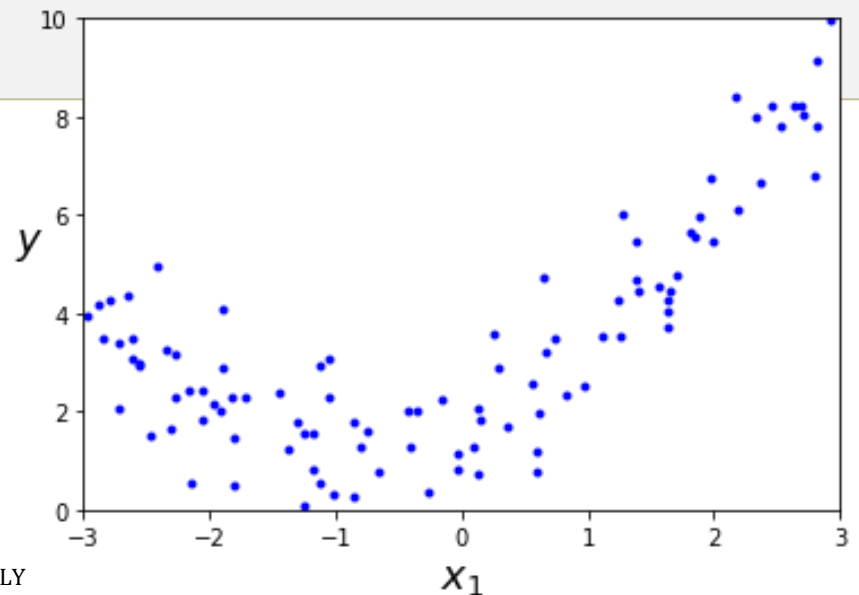




```
import numpy.random as rnd
np.random.seed(42)

# 2차 방정식으로 비선형 데이터 생성
m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.5 * X**2 + X + 2 + np.random.randn(m, 1)

plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([-3, 3, 0, 10])
plt.show()
```



```
# 훈련세트에 있는 각 특성을 제공 (2차 다항)하여 새로운 특성 추가
from sklearn.preprocessing import PolynomialFeatures
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
X[0]
```

```
array([-0.75275929])
```

```
X_poly[0]
```

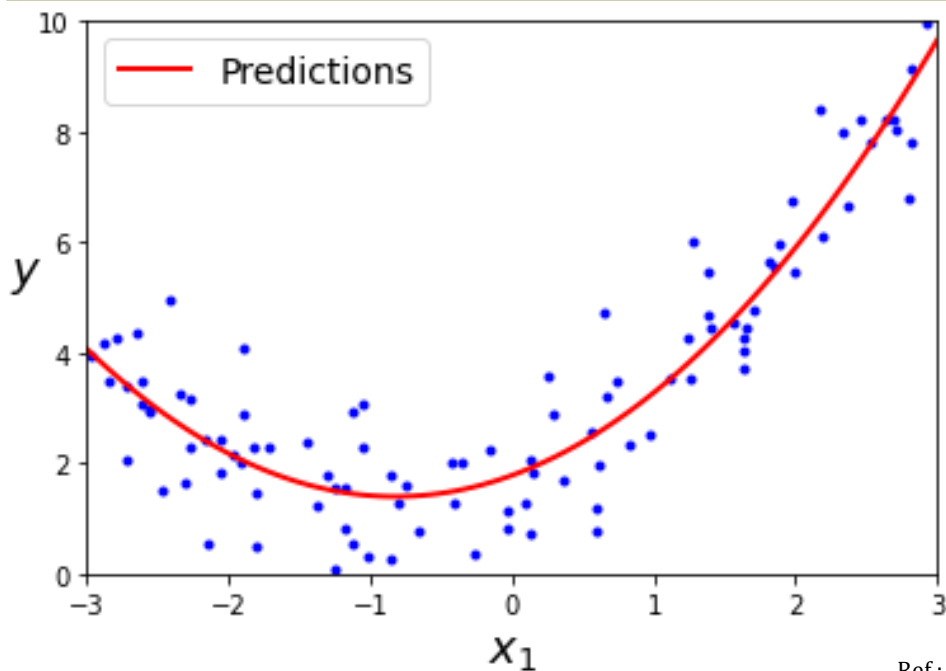
```
array([-0.75275929, 0.56664654])
```

```
lin_reg = LinearRegression()
lin_reg.fit(X_poly, y) # X_poly : 원래 특성 x와 새로운 특성 포함
lin_reg.intercept_, lin_reg.coef_
```

```
(array([1.78134581]), array([[0.93366893, 0.56456263]]))
```

- 실제함수 :  $y = 0.5x_1^2 + 1.0x_1 + 2.0$
- 예측모델 :  $y = 0.56x_1^2 + 0.93x_1 + 1.78$

```
X_new=np.linspace(-3, 3, 100).reshape(100, 1)
X_new_poly = poly_features.transform(X_new)
y_new = lin_reg.predict(X_new_poly)
plt.plot(X, y, "b.")
plt.plot(X_new, y_new, "r-", linewidth=2, label="Predictions")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.legend(loc="upper left", fontsize=14)
plt.axis([-3, 3, 0, 10])
plt.show()
```





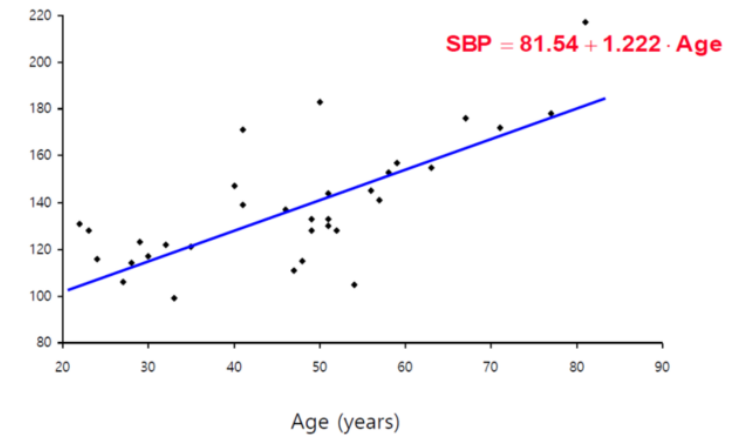
Linear Regression  
**Logistic Regression**  
Softmax Regression

Age	SBP
22	131
23	128
24	116
27	106
28	114
29	123
30	117
32	122
33	99
35	121
40	147

Age	SBP
41	139
41	171
46	137
47	111
48	115
49	133
49	128
50	183
51	130
51	133
51	144

Age	SBP
52	128
54	105
56	145
57	141
58	153
59	157
63	155
67	176
71	172
77	178
81	217

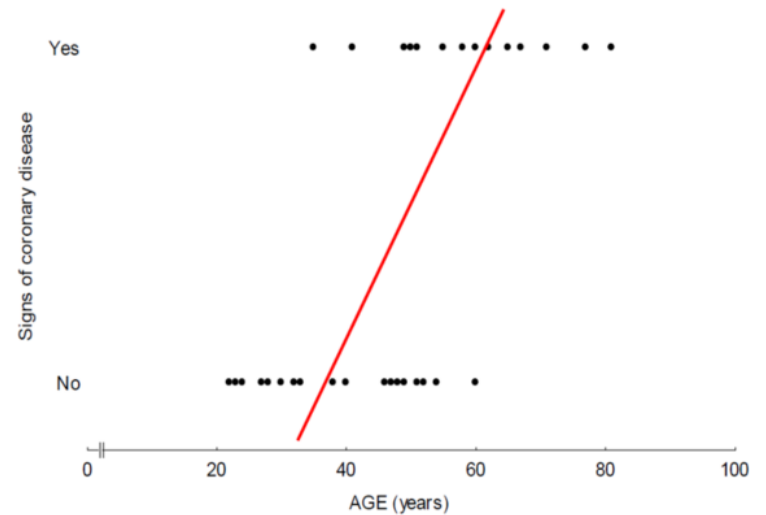
SBP (mm Hg)



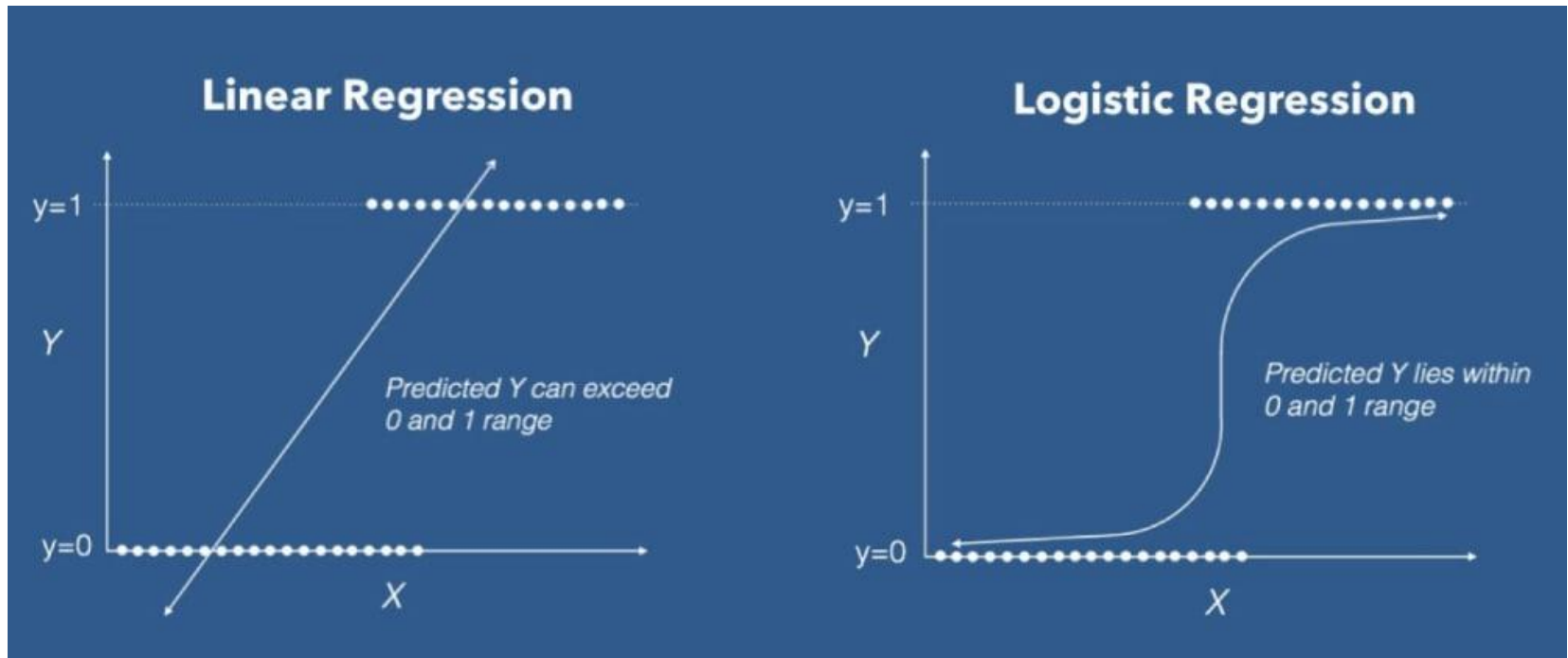
Age	CD
22	0
23	0
24	0
27	0
28	0
30	0
30	0
32	0
33	0
35	1
38	0

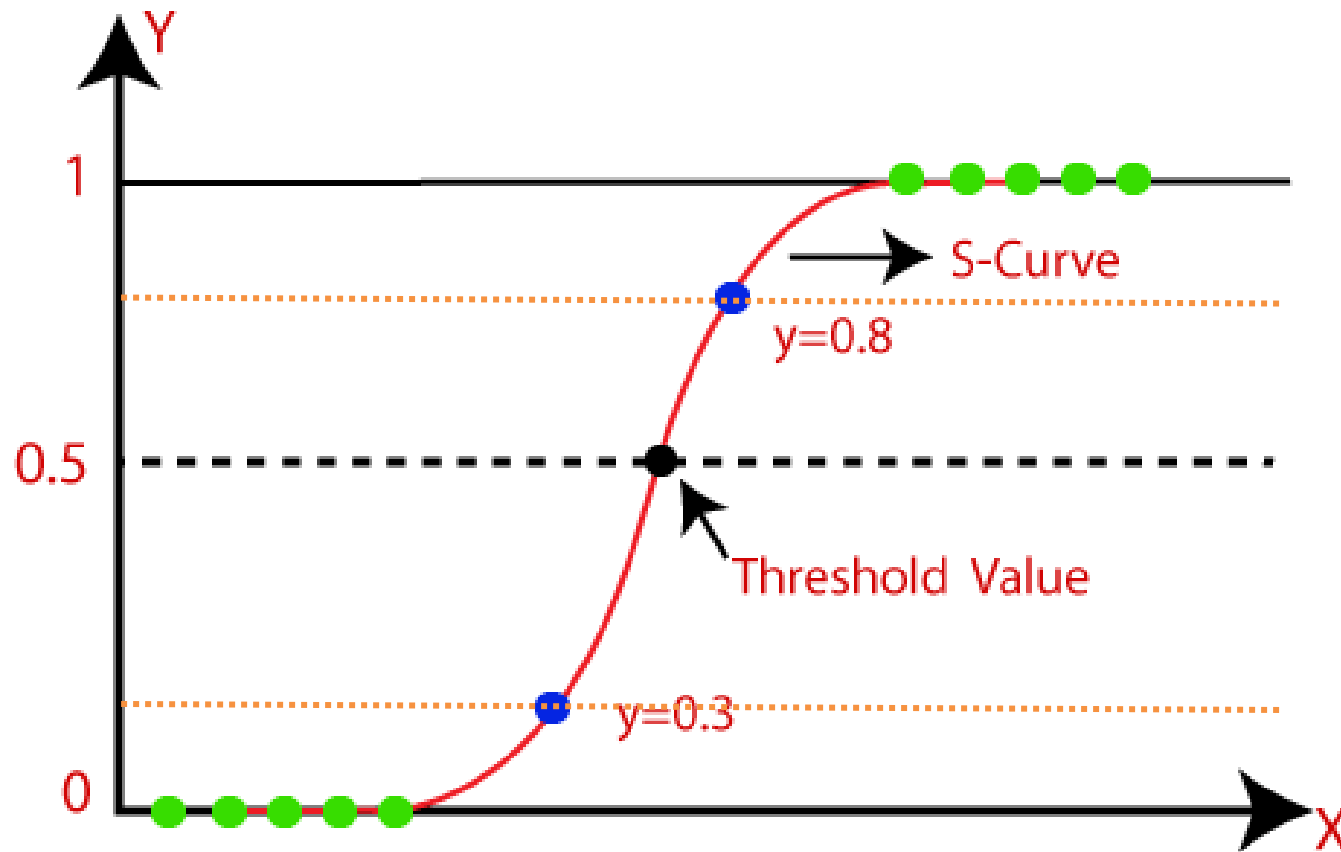
Age	CD
40	0
41	1
46	0
47	0
48	0
49	1
49	0
50	1
51	0
51	1
52	0

Age	CD
54	0
55	1
58	1
60	1
60	0
62	1
65	1
67	1
71	1
77	1
81	1



- 데이터가 어떤 **범주**에 속할 확률을 0에서 1 사이의 값으로 예측
- 확률에 따라 가능성이 더 높은 범주에 속하는 것으로 분류해주는 알고리즘
- 로지스틱 회귀분석은 **이진 분류**를 수행하는 데 사용





- Odds
  - 임의의 사건 A가 발생하지 않을 확률 대비 일어날 확률
  - 실패에 비해 성공할 확률의 비율

$$odds = \frac{P(A)}{1 - P(A)}$$

- (예1) 게임에서 이길 확률 1/5

$$odds = \frac{p}{1 - p} = \frac{1/5}{1 - 1/5} = \frac{1}{4}$$

(5번 게임에서 4번 질 동안 1번 이긴다)

- (예2) 월드컵에서 우승할 Odds = 9:2

$$odds = \frac{p}{1 - p} = \frac{2}{9}, \quad 9p = 2 - 2p, \quad 11p = 2, p = \frac{2}{11} (18\%)$$



## Example

	의약품 A	의약품 B	합계
생존율(0)	32	24	56
생존율(1)	20	42	62
합계	52	66	118

- $Odds(A)$ 
  - $P(A) = 20/52 = 0.38$
  - $Odds(A) = 0.38/1 - 0.38 = 0.61$
  - A약을 먹으면 100명 사망할 동안 61명 생존
- $Odds(B)$ 
  - $P(B) = 42/66 = 0.63$
  - $Odds(B) = 0.63/1 - 0.63 = 1.7$
  - B약을 먹으면 100명 사망할 동안 170명 생존
- $B$ 에 대한  $A$ 의  $Odds\ ratio = 0.61/1.7 = 0.36$ 
  - $B$ 에 비해  $A$ 의 생존(성공)이 0.36배
  - $A$ 가  $B$ 보다 생존율(성공률)이 64% 떨어진다

확률값이 0부터 1로 변화함에 따라 Odds는 0부터 무한대 값

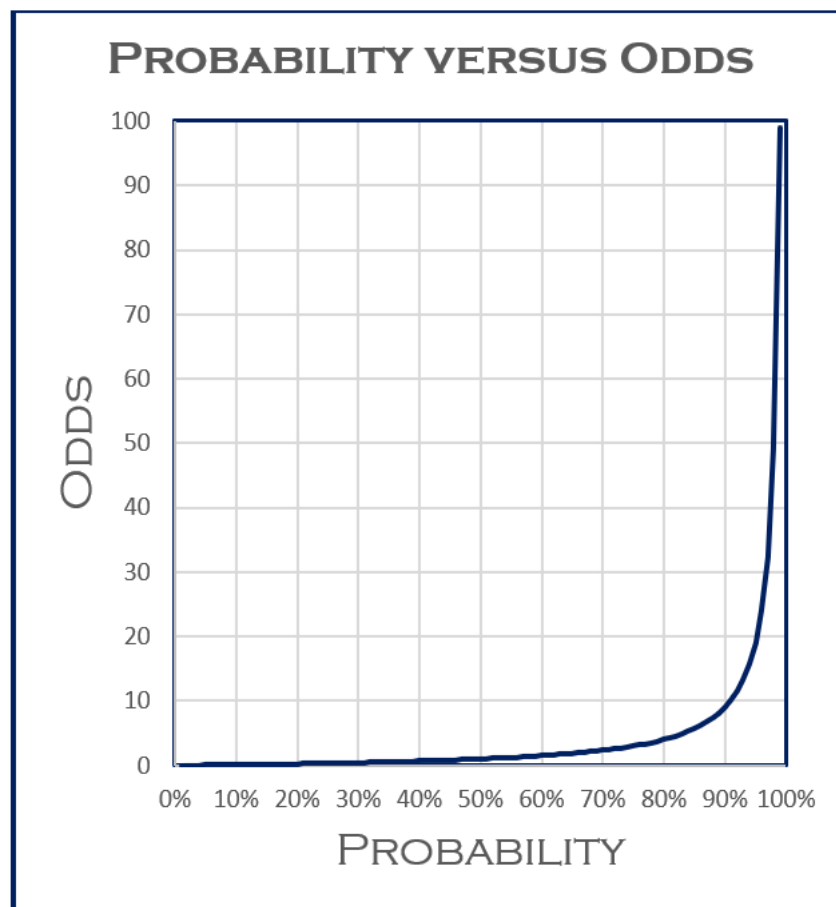
$$0 \leq \frac{p}{1-p} \leq \infty, (0 < p < 1, \quad 0 < 1-p < 1)$$

·  $p$ 가 0에 가까우면  $\frac{0}{1-0} = 0$

$$\lim_{p \rightarrow 0} \frac{p}{1-p} = 0$$

·  $p$ 가 1에 가까우면  $\frac{1}{1-1} = \infty$

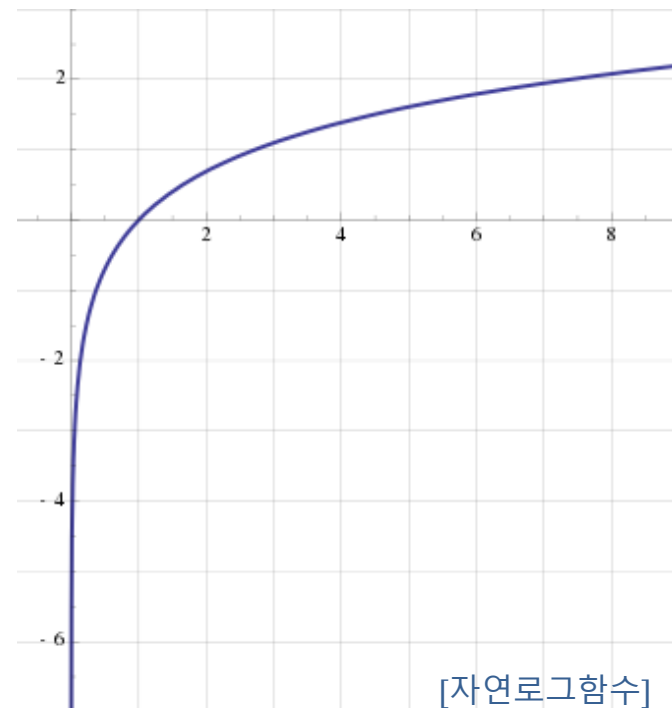
$$\lim_{p \rightarrow 1} \frac{p}{1-p} = \infty$$



$$0 \leq \frac{p}{1-p} \leq \infty, (0 < p < 1, \quad 0 < 1-p < 1)$$

음의 무한대를 포함시키기 위하여  $\log_e$ 를 취한다

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



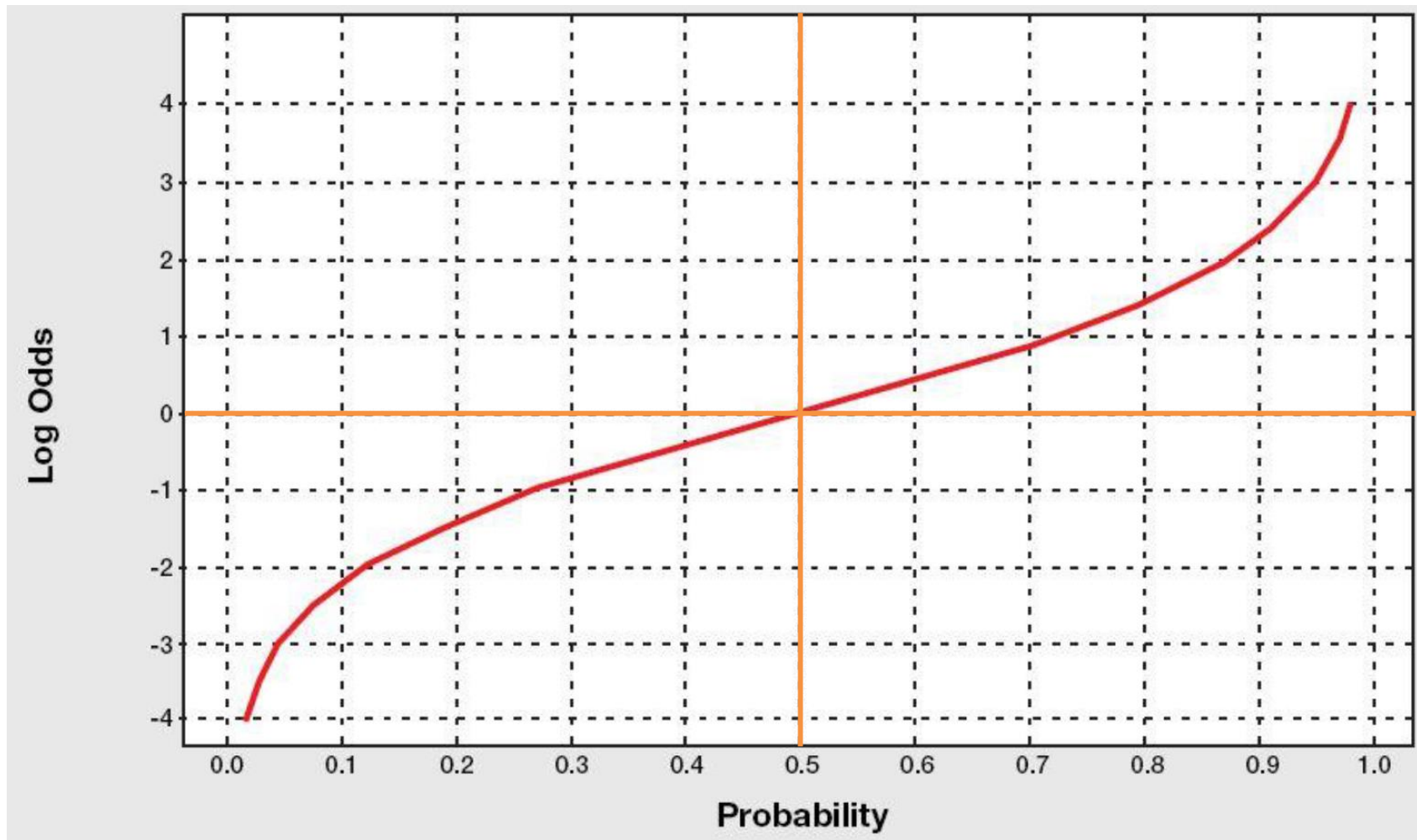
[참고]

- $\log_e 0 = ?$  : 정의되지 않는다.  $e^x = 0$ 를 만족시키는  $x$ 는 없다
- $x$ 가 양의 변 ( $0+$ )에서 0에 가까워질 때  $x$ 의 자연 로그 한계는 음의 무한대

$$\lim_{x \rightarrow 0} \log_e x = -\infty$$

$$\bullet \log 0 = -\infty, \log 1 = +\infty$$

- 확률값이 0부터 1까지 변화함에 따라 Log Odds는  $-\infty, \infty$ 의 값을 가지며 대칭



# Logistic Function

linear regression :  $y_i = \beta_0 + \beta_1 x_i$

logistic regression :  $\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_i$

( $\therefore$  multi case :  $w_0 x_0 + w_1 x_1 + \dots + w_m x_m$ )

$$y = \beta_0 + \beta_1 x = \mathbf{w} \cdot \mathbf{x}$$

$$\log \frac{p}{1-p} = \mathbf{w} \cdot \mathbf{x}$$

$$\frac{p}{1-p} = e^{wx}$$

$$p = e^{wx}(1-p) = e^{wx} - e^{wx}p$$

$$p + e^{wx}p = e^{wx} \rightarrow p(1 + e^{wx}) = e^{wx}$$

$$p = \frac{e^{wx}}{1 + e^{wx}} = \frac{\frac{e^{wx}}{e^{wx}}}{\frac{1 + e^{wx}}{e^{wx}}} = \frac{1}{\frac{1}{e^{wx}} + \frac{e^{wx}}{e^{wx}}} = \frac{1}{1 + e^{-wx}}$$

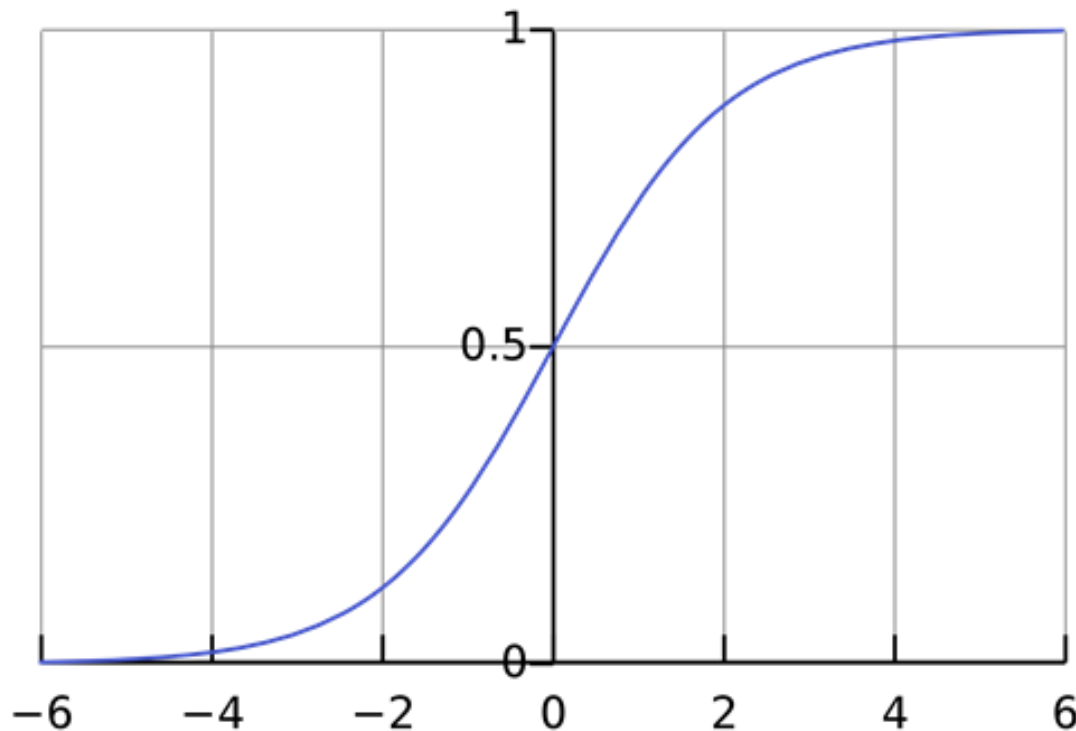
$$cf : y = a^x \Rightarrow x = \log_a y, \quad y = e^x \Rightarrow x = \log_e y$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{1}{1 + \exp(-x)}$$

- $0 < \sigma(x) < 1$
- $e : 2.7182 \dots$  (자연상수)
- $\sigma(1) = 0.731 \dots$
- $\sigma(2) = 0.880 \dots$

- 회귀모델예측

$$\hat{y} = \begin{cases} 0 & \text{if } \sigma(x) < 0.5 \\ 1 & \text{if } \sigma(x) \geq 0.5 \end{cases}$$

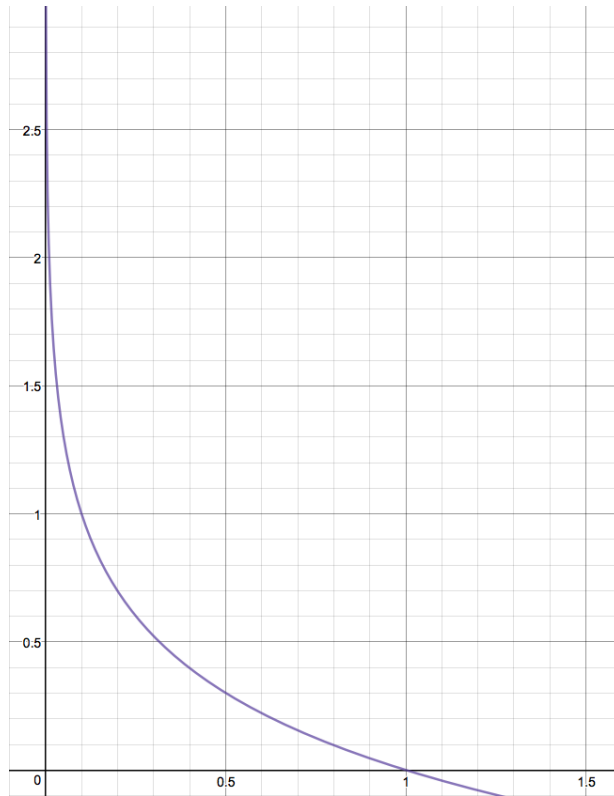


## 비용함수(Cost function)

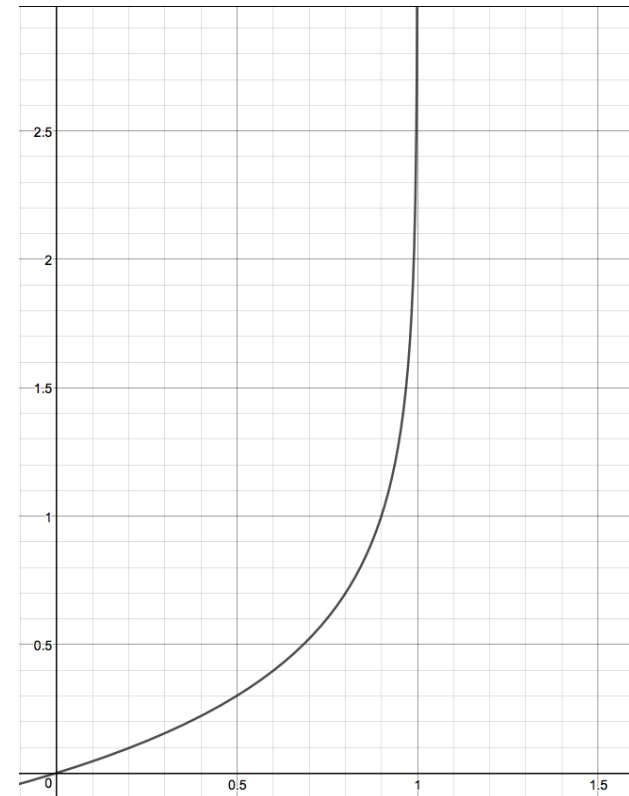
31

양성샘플( $y = 1$ )에 대해서는 높은 확률을, 음성샘플( $y = 0$ )에 대해서는 낮은 확률을 추정하는 모델의 파라미터  $\theta$ 를 찾는 것

$$\text{cost}(w) = \begin{cases} -\log(\hat{y}) & \text{if } y = 1 \\ -\log(1 - \hat{y}) & \text{if } y = 0 \end{cases}$$



$$y = 1 \quad \begin{array}{l} \hat{y} = 1, \quad \text{cost}(w) = 0 \\ \hat{y} = 0, \quad \text{cost}(w) = \infty \end{array}$$



$$y = 0 \quad \begin{array}{l} \hat{y} = 0, \quad \text{cost}(w) = 0 \\ \hat{y} = 1, \quad \text{cost}(w) = \infty \end{array}$$

- 하나의 훈련 샘플에 대한 비용함수

$$\text{cost}(w) = \begin{cases} -\log(\hat{y}) & \text{if } y = 1 \\ -\log(1 - \hat{y}) & \text{if } y = 0 \end{cases}$$

- 전체 훈련세트에 대한 비용함수(Log Loss, Cross Entropy) : 모든 훈련 샘플의 비용 평균

$$L(w) = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

- 로지스틱 회귀 모델 훈련

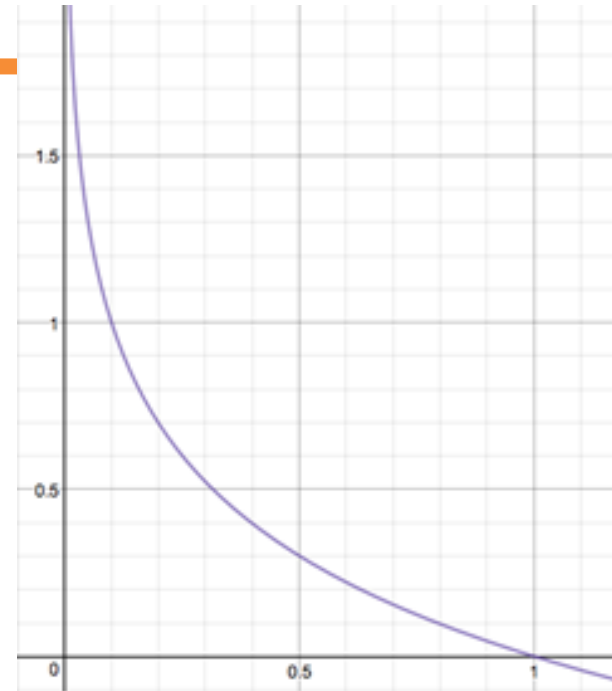
$$\frac{\delta}{\delta w} L(w) = \frac{1}{n} \sum_{i=1}^n (\text{sigmoid}(wx_i) - y_i) x_i$$
$$\left( cf : y = \beta_0 + \beta_1 x = wx, \quad \text{sigmoid}(wx) = \frac{1}{1 + e^{-wx}} \right)$$

$$\therefore y = 1, \quad f = -\log(\hat{y}) \quad \times$$

$$\therefore y = 0, \quad f = \quad \times \quad -1 * \log(1 - \hat{y})$$



$$D(\hat{y}, y) = - \sum_i^N y_i \log(\hat{y}_i)$$



- 실제값( $y$ ) =  $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$  이다. 예측해 보자.

$$\hat{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \text{ true인 경우, } \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot -\log \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} \infty \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow 0 + 0 = \mathbf{0} \text{ (Σ이므로)}$$

$$\hat{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \text{ false인 경우, } \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot -\log \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \begin{bmatrix} 0 \\ \infty \end{bmatrix} \rightarrow 0 + \infty = \infty$$

- 실제값( $y$ ) =  $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$  일 때

$$\hat{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \text{ true인 경우, } \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot -\log \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow 0 + 0 = \mathbf{0}$$

$$\hat{y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \text{ false인 경우, } \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot -\log \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} \infty \\ 0 \end{bmatrix} = \begin{bmatrix} \infty \\ 0 \end{bmatrix} \rightarrow \infty + 0 = \infty$$

$$h = \sum_{i=0}^n w_i x_i$$

$$s = \frac{1}{1 + \exp(-h)}$$

$$L(\text{Loss}) = (s - y)^2$$

$$\frac{\delta L}{\delta w} = \frac{\delta L}{\delta s} \cdot \frac{\delta s}{\delta h} \cdot \frac{\delta h}{\delta w} = (s - y) \cdot s(1 - s) \cdot x$$

$$\frac{\delta L}{\delta s} = s - y$$

$$\frac{\delta s}{\delta h} = \frac{\exp(-h)}{(1 + \exp(-h))^2} = -\frac{1}{1 + \exp(-h)} \cdot \frac{\exp(-h)}{1 + \exp(-h)} = s(1 - s)$$

$$\frac{\delta h}{\delta w} = x$$

# 붓꽃 데이터 로드

```
from sklearn import datasets
```

```
iris = datasets.load_iris()
```

```
list(iris.keys())
```

['data', 'target', 'frame', 'target\_names', 'DESCR', 'feature\_names', 'filename']

```
X = iris["data"][:,3:]      # 꽃잎의 너비만 사용
```

```
y = (iris["target"]==2).astype("int") #iris-Versinica이면 1, 아니면 0
```

# 로지스틱 회귀모델 훈련

```
from sklearn.linear_model import LogisticRegression
```

# 향후 버전이 바뀌더라도 동일한 결과를 만들기 위해

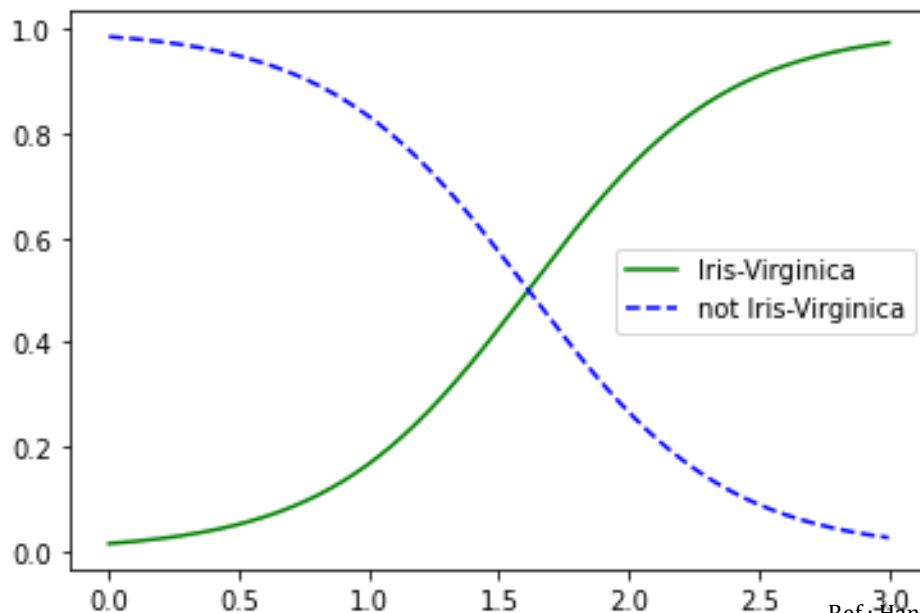
# 사이킷런 0.22 버전의 기본값인 solver="lbfgs"로 지정

```
log_reg = LogisticRegression(solver="lbfgs", random_state=42)
```

```
log_reg.fit(X,y)
```

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

# 꽃잎의 너비가 0~3cm인 꽃 1,000개에 대해 모델의 추정확률을 계산
X_new = np.linspace(0, 3, 1000).reshape(-1, 1)
y_proba = log_reg.predict_proba(X_new)
plt.plot(X_new, y_proba[:, 1], "g-", label = "Iris-Virginica")
plt.plot(X_new, y_proba[:, 0], "b--", label = "not Iris-Virginica")
plt.legend()
plt.show()
```



```

X_new = np.linspace(0, 3, 1000).reshape(-1, 1)
y_proba = log_reg.predict_proba(X_new)
decision_boundary = X_new[y_proba[:, 1] >= 0.5][0]

plt.figure(figsize=(8, 3))
plt.plot(X[y==0], y[y==0], "bs") # 음성범주 pointing
plt.plot(X[y==1], y[y==1], "g^") # 양성범주 pointing

# 결정경계 표시
plt.plot([decision_boundary, decision_boundary], [-1, 2], "k:", linewidth=2)

# 추정확률 plotting
plt.plot(X_new, y_proba[:, 1], "g-", linewidth=2, label="Iris virginica")
plt.plot(X_new, y_proba[:, 0], "b--", linewidth=2, label="Not Iris virginica")

plt.text(decision_boundary+0.02, 0.15, "Decision boundary", fontsize=14, color="k", ha="center")
plt.arrow(decision_boundary, 0.08, -0.3, 0, head_width=0.05, head_length=0.1, fc='b', ec='b')
plt.arrow(decision_boundary, 0.92, 0.3, 0, head_width=0.05, head_length=0.1, fc='g', ec='g')
plt.xlabel("Petal width (cm)", fontsize=14)
plt.ylabel("Probability", fontsize=14)
plt.legend(loc="center left", fontsize=14)
plt.axis([0, 3, -0.02, 1.02])

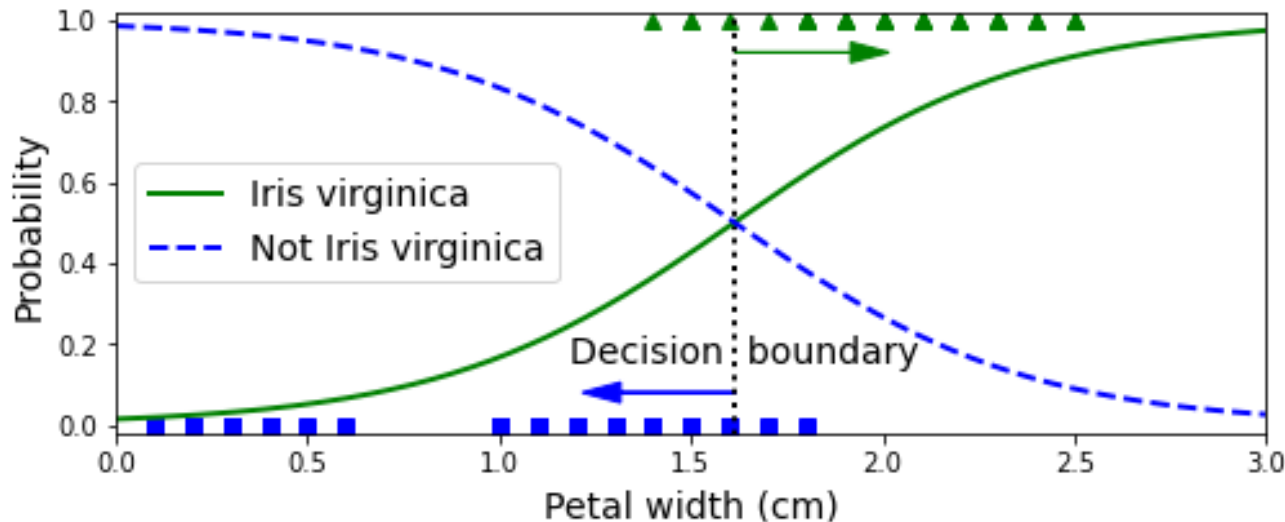
plt.show()

```

## 결정경계(Decision Boundary)

38

- *Iris - Verginica*( $y = 1$ )의 꽃잎 너비 : 1.4~2.5cm 사이에 분포(초록 삼각형)
- *Iris - Verginica*가 아닌 붓꽃의 꽃잎 너비 : 0.1~1.8cm에 분포(파란 사각형)
- 중첩되는 구간이 존재



```
decision_boundary → array([1.61561562])
```

```
# 양쪽의 확률이 50%가 되는 1.6cm 근방에서 결정경계가 만들어지고,  
# 분류기는 1.6cm보다 크면 Iris-Verginica로 분류하고 작으면 아니라고 예측한다.
```

```
log_reg.predict([[1.7], [1.5]]) → array([1, 0])
```

```
# 두번째 꽃잎 너비와 꽃잎 길이 2개의 특성을 이용해 훈련
from sklearn.linear_model import LogisticRegression

X = iris["data"][:, (2, 3)] # petal length, petal width
y = (iris["target"] == 2).astype(np.int)

# LogisticRegression모델의 규제강도를 조절하는 하이퍼파라미터는 alpha가 아니라 그 역
# 수에 해당하는 c이다. c가 높을수록 모델의 규제가 줄어든다.
log_reg = LogisticRegression(solver="lbfgs", C=10**10, random_state=42)
log_reg.fit(X, y)

x0, x1 = np.meshgrid(
    np.linspace(2.9, 7, 500).reshape(-1, 1),
    np.linspace(0.8, 2.7, 200).reshape(-1, 1),
)
X_new = np.c_[x0.ravel(), x1.ravel()]

y_proba = log_reg.predict_proba(X_new)
```

```
plt.figure(figsize=(10, 4))
plt.plot(X[y==0, 0], X[y==0, 1], "bs")
plt.plot(X[y==1, 0], X[y==1, 1], "g^")

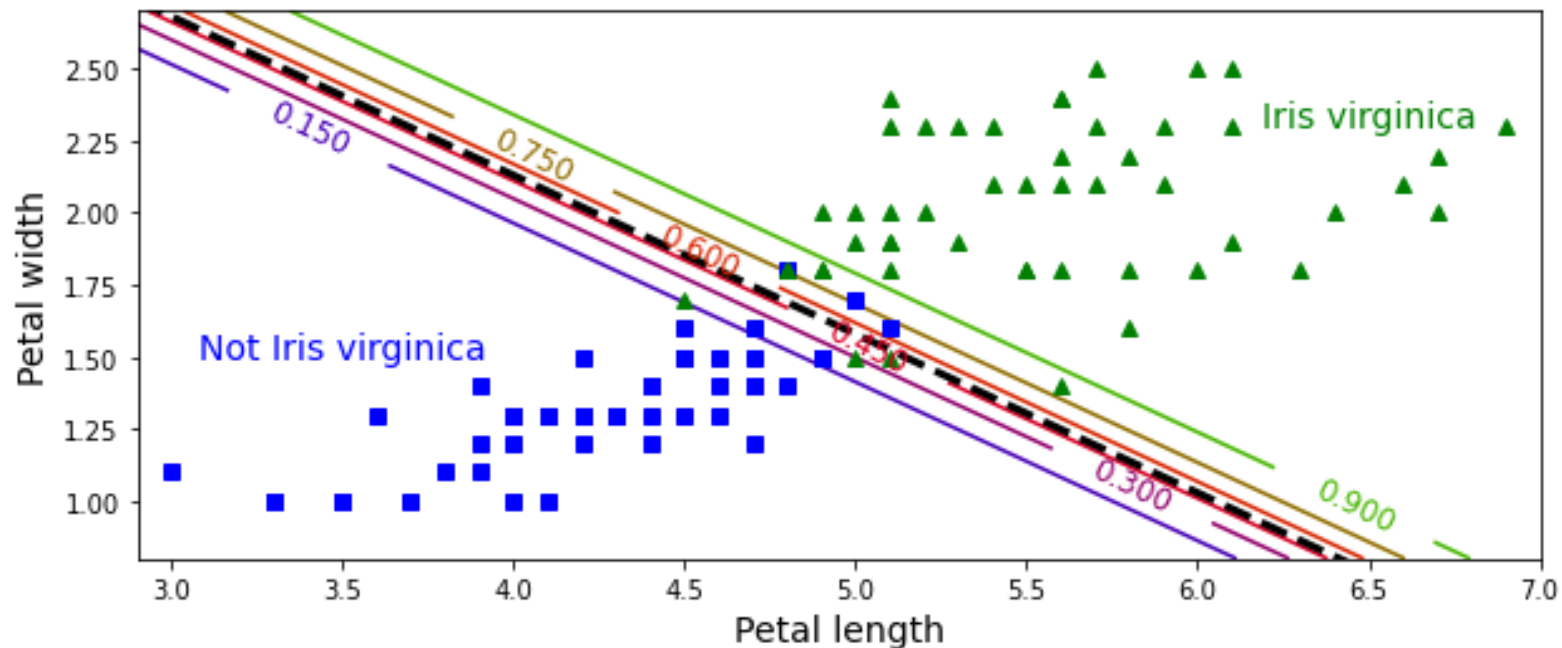
zz = y_proba[:, 1].reshape(x0.shape)
contour = plt.contour(x0, x1, zz, cmap=plt.cm.brg)

left_right = np.array([2.9, 7])
boundary = -
(log_reg.coef_[0][0] * left_right + log_reg.intercept_[0]) / log_reg.coef_[0][1]

plt.clabel(contour, inline=1, fontsize=12)
plt.plot(left_right, boundary, "k--", linewidth=3)
plt.text(3.5, 1.5, "Not Iris virginica", fontsize=14, color="b", ha="center")
plt.text(6.5, 2.3, "Iris virginica", fontsize=14, color="g", ha="center")
plt.xlabel("Petal length", fontsize=14)
plt.ylabel("Petal width", fontsize=14)
plt.axis([2.9, 7, 0.8, 2.7])

plt.show()
```





- 검은색 점선은 50% 확률을 추정하는 지점으로, 이 모델의 **결정경계**이다.
- 이 경계는  $\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$ 을 만족하는  $(x_1, x_2)$ 의 집합인 **선형경계**이다
- *Iris - Verginica*에 속할 확률을 15%부터 90%까지 나타내고 있다.



Linear Regression  
Logistic Regression  
**Softmax Regression**

- 로지스틱 함수를 여러 개의 입력인 경우 로지스틱 함수를 사용할 수 있도록 일반화 한 것 : 다항 로지스틱 회귀(*multinomial logistic regression*)
- 이진 분류기를 훈련시켜 연결하지 않고 직접 다중 클래스를 지원할 수 있도록 일반화한 것
- 샘플에 대해 각 클래스의 점수가 계산되면 소프트맥스 함수를 통과시켜 해당 되는 클래스에 속할 확률 추정
- Logistic(Sigmoid) : 이진분류(Binary classification)**

$$x = \{x_1, x_2, x_3, x_4\} \rightarrow wx + b \rightarrow \text{sigmoid} \rightarrow 0.77 \rightarrow \begin{cases} > 0.5 & \text{true} & \text{Class 1} \\ < 0.5 & \text{false} & \text{Class 2} \end{cases}$$

- Softmax : 다중분류(Multi - classification)**

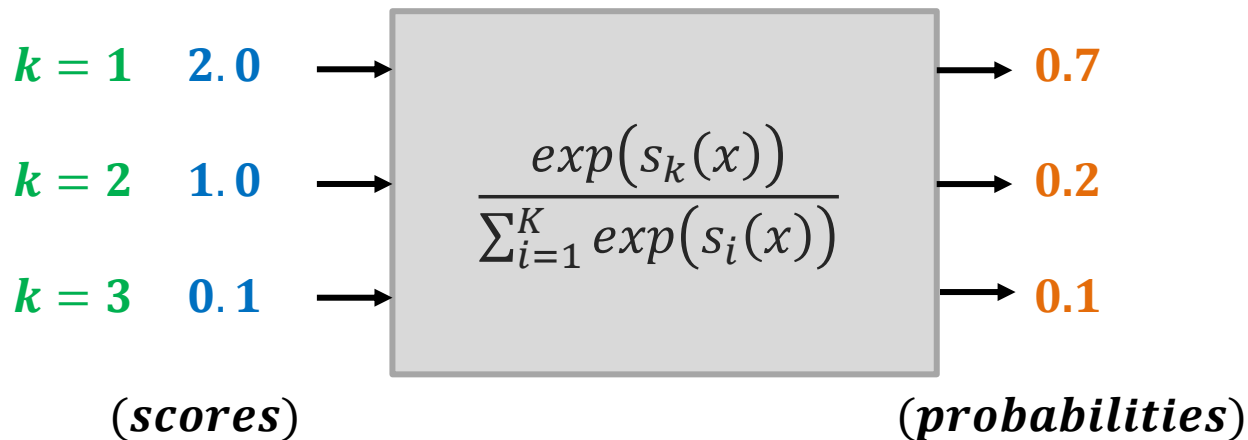
$$x = \{x_1, x_2, x_3, x_4\} \rightarrow wx + b \rightarrow \text{softmax} \rightarrow \begin{bmatrix} 0.10 \\ 0.77 \\ 0.13 \end{bmatrix} \rightarrow \begin{matrix} \text{Class 1} \\ \text{Class 2} \\ \text{Class 3} \end{matrix}$$

## 소프트맥스 함수(softmax function)

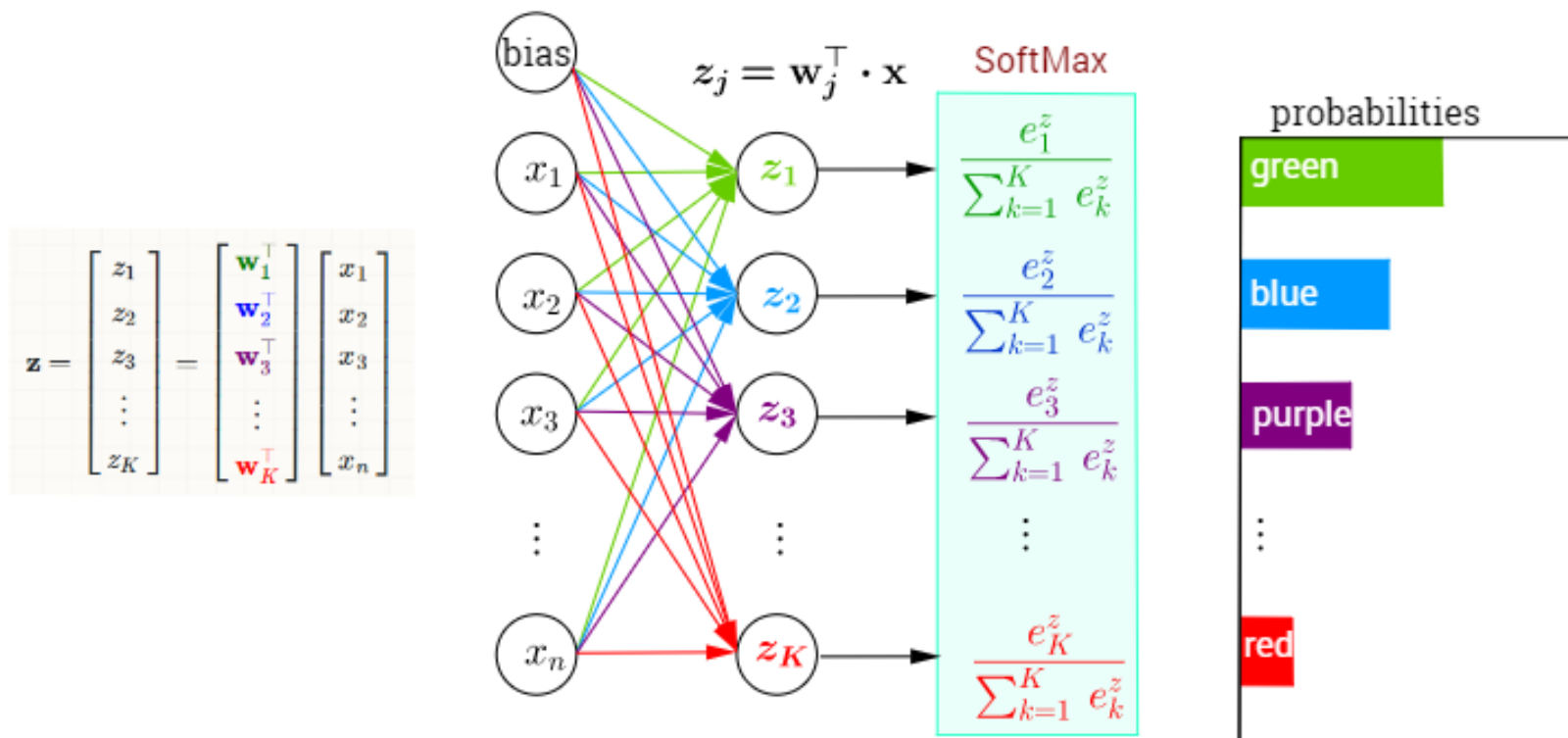
- 소프트맥스 함수의 출력합계는 1
- $n$ 차 실수 벡터를 0과 1 사이의 실수로 변환하여 출력

$$\sigma(s(x))_k = \frac{\exp(s_k(x))}{\sum_{i=1}^K \exp(s_i(x))}$$

- $K$ : 클래스의 수
- $s(x)$ : 샘플  $x$ 에 대한 각 클래스의 점수를 담은 벡터
- $\sigma(s(x))_k$ : 샘플  $x$ 에 대한 각 클래스의 점수가 주어졌을 때 이 샘플이 클래스  $k$ 에 속할 추정 **확률**



## Multi-Class Classification with NN and SoftMax Function



```
'''
소프트맥스 함수
로지스틱 회귀모델은 여러 개의 이진 분류기를 훈련시켜 연결하지 않고,
직접 다중 범주를 지원하도록 일반화될 수 있다.
이를 소프트맥스 회귀 또는 다항 로지스틱 회귀라고 한다.
'''

X = iris["data"][:, (2, 3)] # 꽃잎 길이와 너비 변수
y = iris["target"]         # 3개의 범주 그대로 사용

'''
- 사이킷런의 LogisticRegression은 범주가 2이상이면 일대다(OvA) 전략을 사용
- multi_class='multinomial' 옵션
- 소프트맥스 회귀를 사용하려면 solver='lbfgs' 옵션을 준다
- lbfgs: 소프트맥스 회귀를 지원하는 알고리즘
'''

softmax_reg = LogisticRegression(multi_class="multinomial",
                                solver="lbfgs", C=10, random_state=42)
softmax_reg.fit(X, y)
```

```

# 훈련시킨 소프트맥스 분류기의 결정경계를 시각화. 새로운 샘플 생성
x0, x1 = np.meshgrid(
    np.linspace(0, 8, 500).reshape(-1, 1),
    np.linspace(0, 3.5, 200).reshape(-1, 1),
)
X_new = np.c_[x0.ravel(), x1.ravel()]

y_proba = softmax_reg.predict_proba(X_new)
y_predict = softmax_reg.predict(X_new)

zz1 = y_proba[:, 1].reshape(x0.shape)
zz = y_predict.reshape(x0.shape)

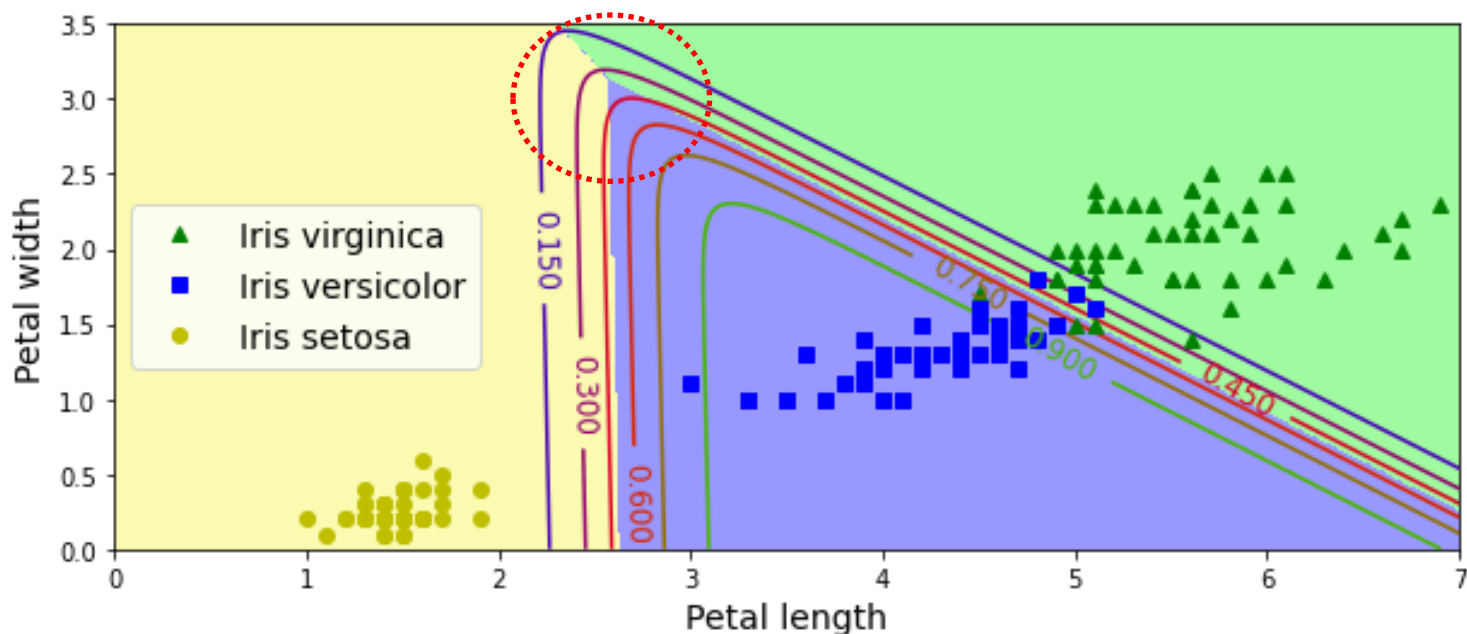
plt.figure(figsize=(10, 4))
plt.plot(X[y==2, 0], X[y==2, 1], "g^", label="Iris virginica")
plt.plot(X[y==1, 0], X[y==1, 1], "bs", label="Iris versicolor")
plt.plot(X[y==0, 0], X[y==0, 1], "yo", label="Iris setosa")

from matplotlib.colors import ListedColormap
custom_cmap = ListedColormap(['#fafab0', '#9898ff', '#a0faa0'])

plt.contourf(x0, x1, zz, cmap=custom_cmap)
contour = plt.contour(x0, x1, zz1, cmap=plt.cm.brg)
plt.clabel(contour, inline=1, fontsize=12)
plt.xlabel("Petal length", fontsize=14)
plt.ylabel("Petal width", fontsize=14)
plt.legend(loc="center left", fontsize=14)
plt.axis([0, 7, 0, 3.5])

plt.show()

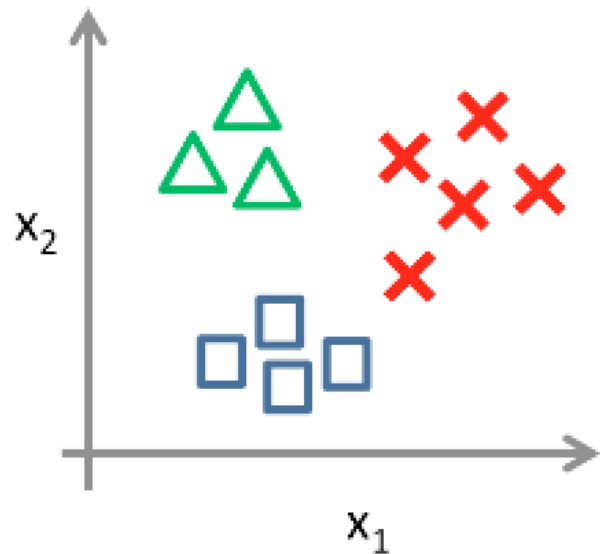
```






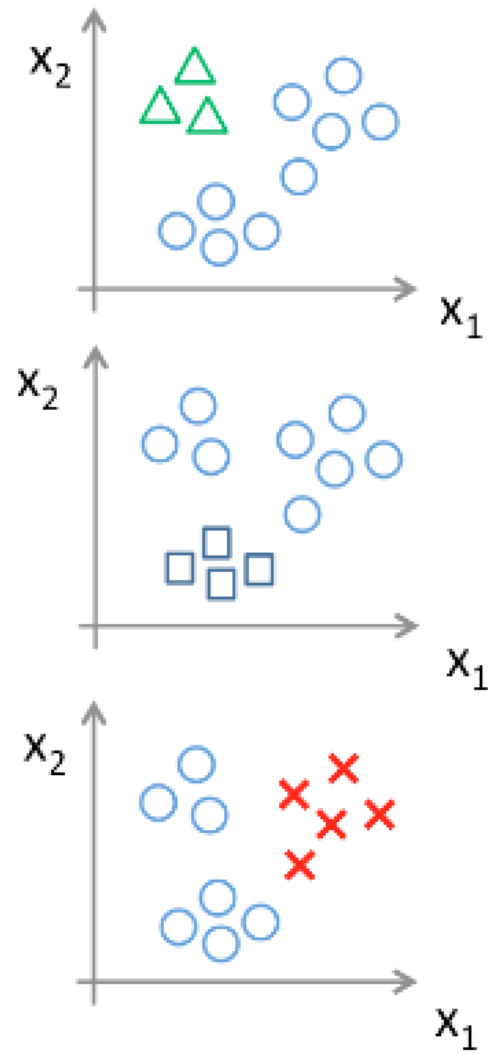
- 범주와 범주 사이의 결정경계(배경색)가 모두 선형 (3개 범주).
- 이진분류와 달리 0.5의 경계가 아니라 범주에 속할 확률이 0.5 이하 라도 분류한다 (범주가 2개보다 많으므로).



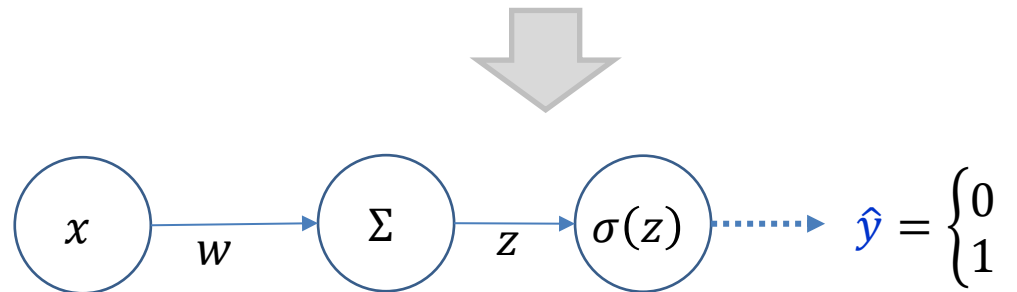
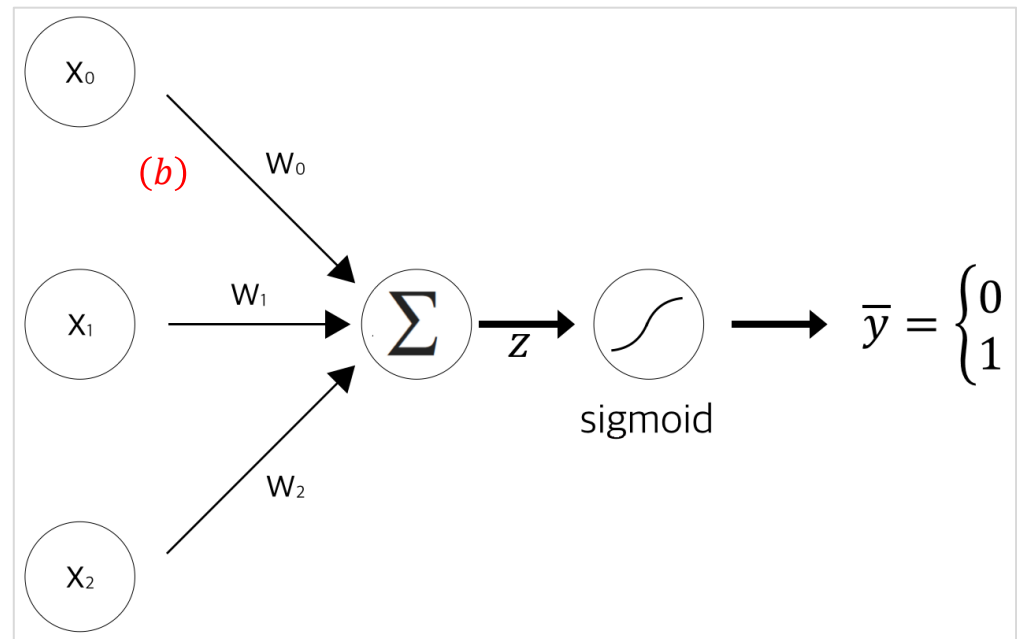
One-vs-all (one-vs-rest):



Class 1:   
 Class 2:   
 Class 3: 

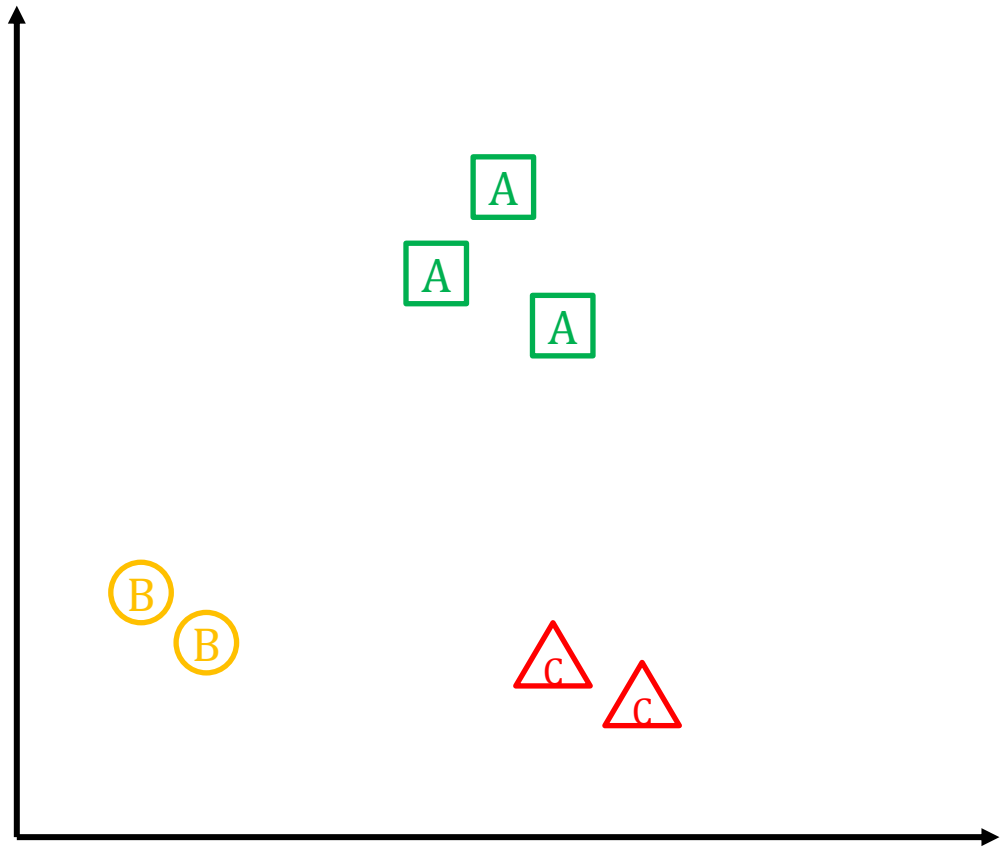


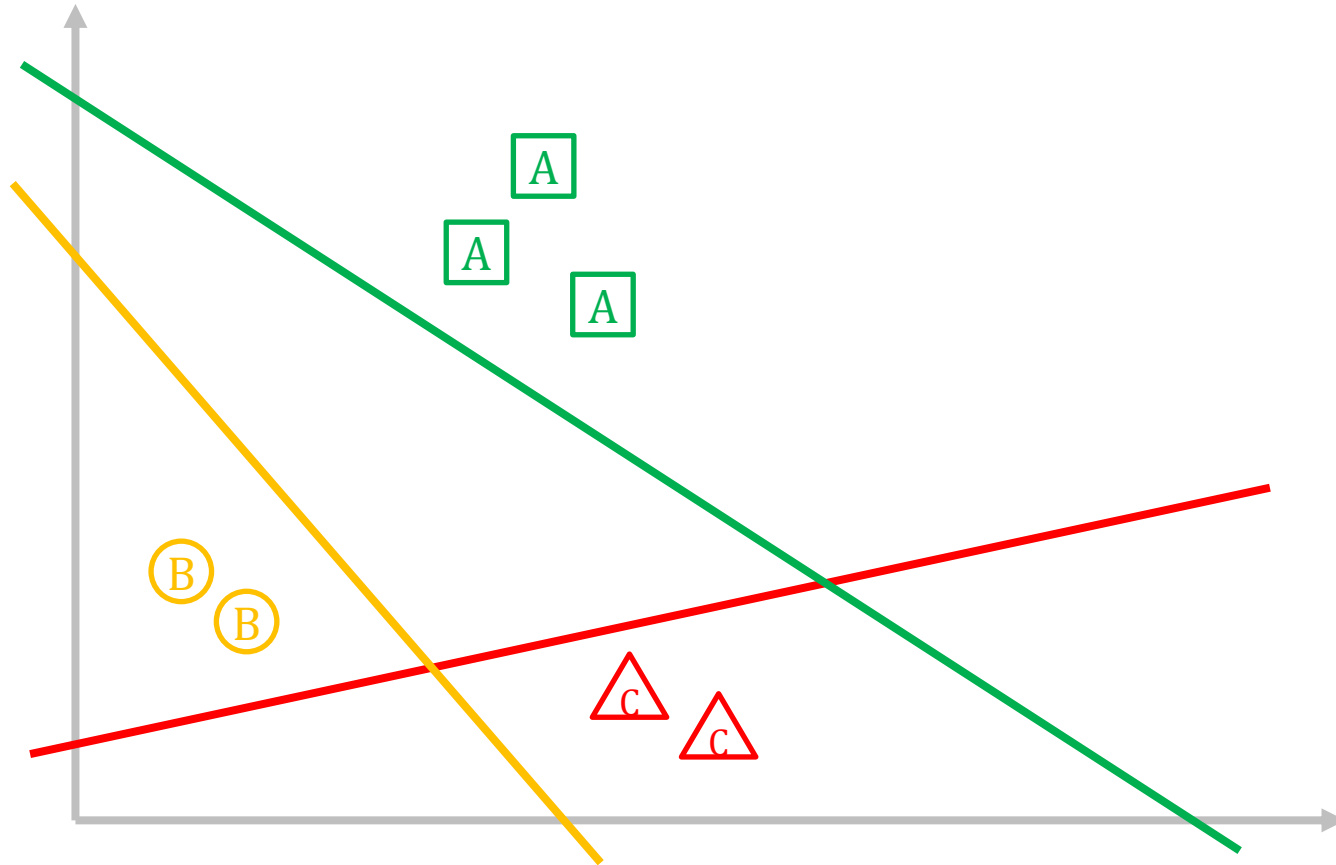
$$\hat{y} = \frac{1}{1 + e^{-wx+b}}$$

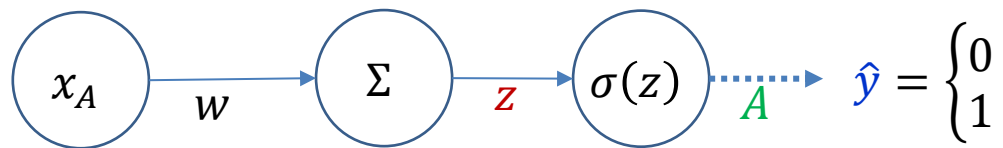
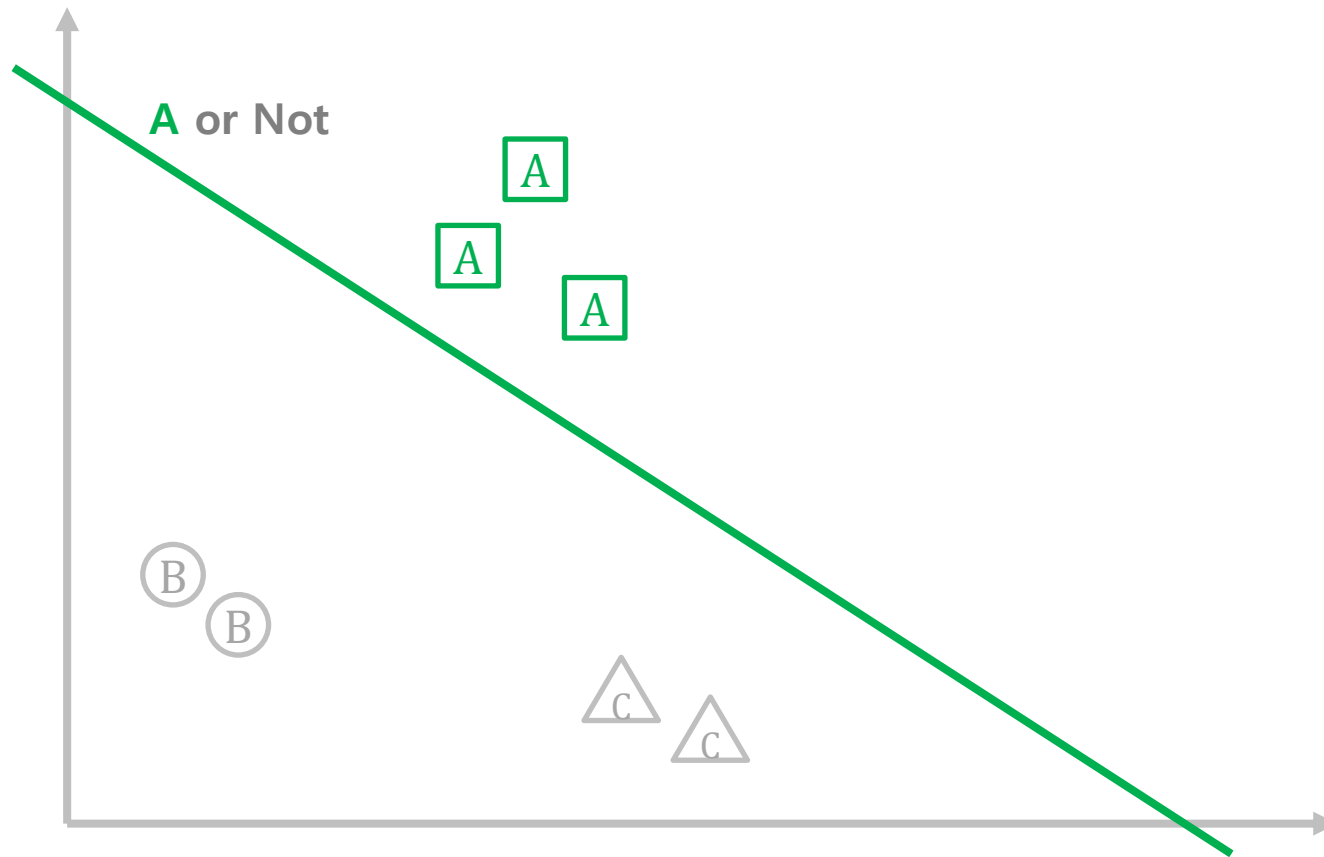


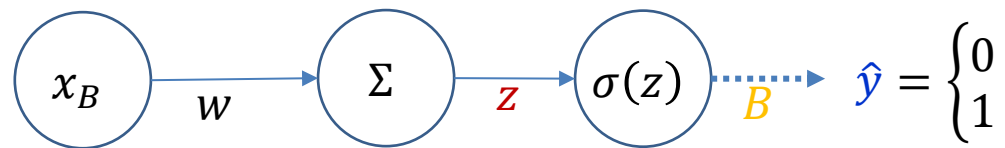
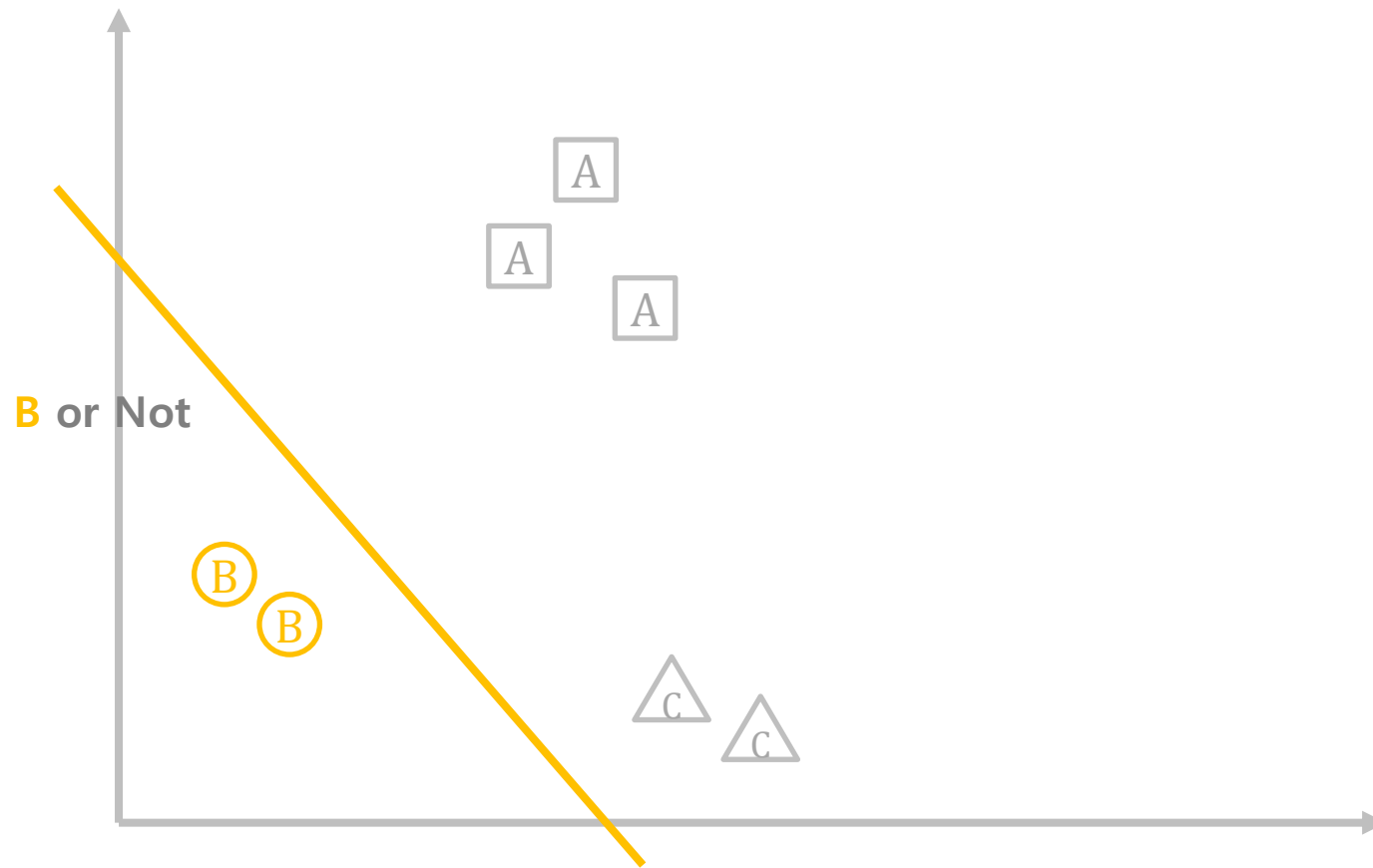
## Example

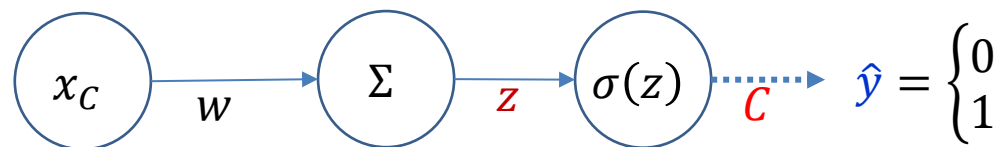
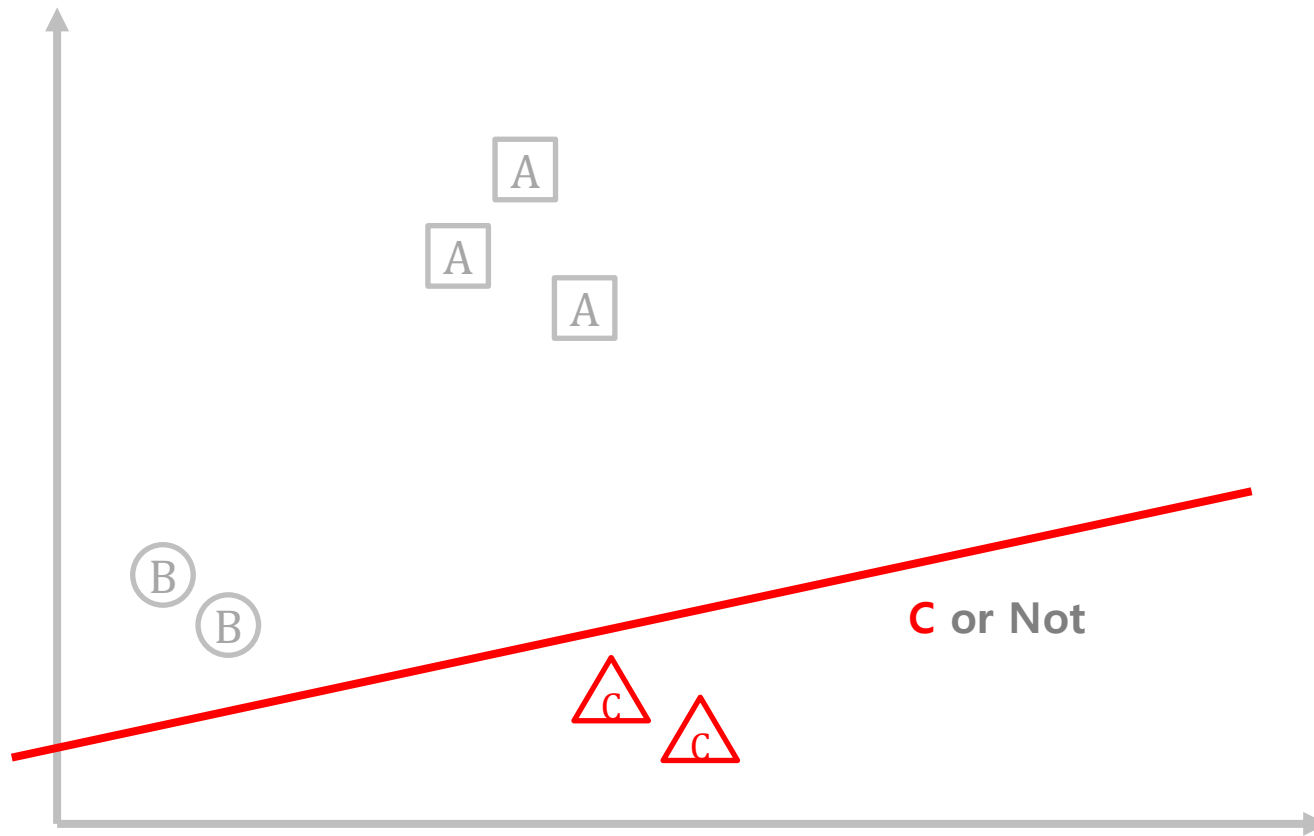
x1 (시험1)	x2 (시험2)	x3 (출석)	Y (학점)
100	100	5	A
90	100	5	A
80	80	4	A
70	90	4	B
90	80	2	B
50	50	3	C
70	50	2	C

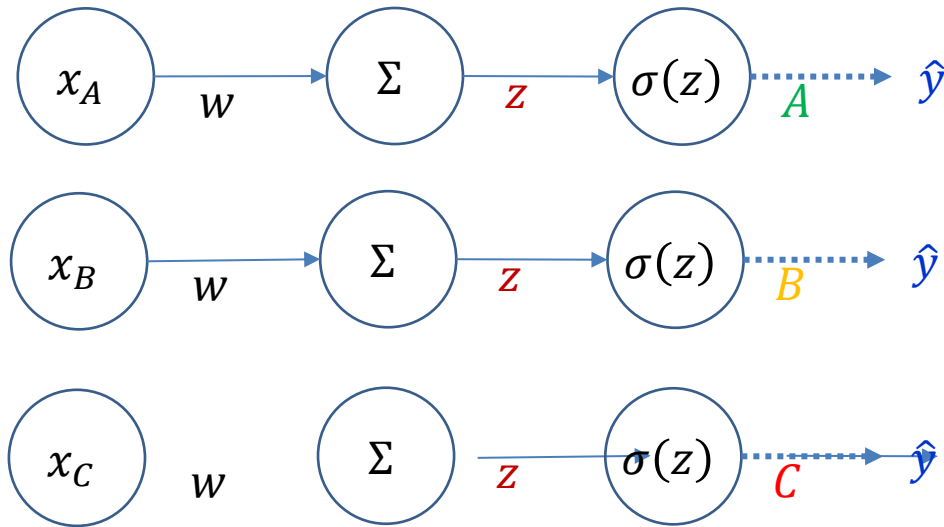












$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_{A1} \\ x_{A2} \\ x_{A3} \end{bmatrix} = [w_1 x_{A1} + w_2 x_{A2} + w_3 x_{A3}]$$

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_{B1} \\ x_{B2} \\ x_{B3} \end{bmatrix} = [w_1 x_{B1} + w_2 x_{B2} + w_3 x_{B3}]$$

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_{C1} \\ x_{C2} \\ x_3 \end{bmatrix} = [w_1 x_{C1} + w_2 x_{C2} + w_3 x_{C3}]$$



$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_{A1} \\ x_{A2} \\ x_{A3} \end{bmatrix} = [w_1 x_{A1} + w_2 x_{A2} + w_3 x_{A3}]$$

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_{B1} \\ x_{B2} \\ x_{B3} \end{bmatrix} = [w_1 x_{B1} + w_2 x_{B2} + w_3 x_{B3}]$$

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_{C1} \\ x_{C2} \\ x_3 \end{bmatrix} = [w_1 x_{C1} + w_2 x_{C2} + w_3 x_{C3}]$$

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_{A1} & x_{B1} & x_{C1} \\ x_{A2} & x_{B2} & x_{C2} \\ x_{A3} & x_{B3} & x_{C3} \end{bmatrix} = \begin{bmatrix} w_1 x_{A1} + w_2 x_{A2} + w_3 x_{A3} \\ w_1 x_{B1} + w_2 x_{B2} + w_3 x_{B3} \\ w_1 x_{C1} + w_2 x_{C2} + w_3 x_{C3} \end{bmatrix} = \begin{bmatrix} \hat{y}_A \\ \hat{y}_B \\ \hat{y}_C \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}$$

$$\begin{aligned} \text{softmax} \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix} &\rightarrow p = 0.7 \text{ } A(\text{O}) \\ &\rightarrow p = 0.2 \text{ } B(\times) \\ &\rightarrow p = 0.1 \text{ } C(\times) \end{aligned}$$

$$\therefore \hat{y} = A$$

airplane

automobile

bird

cat

deer

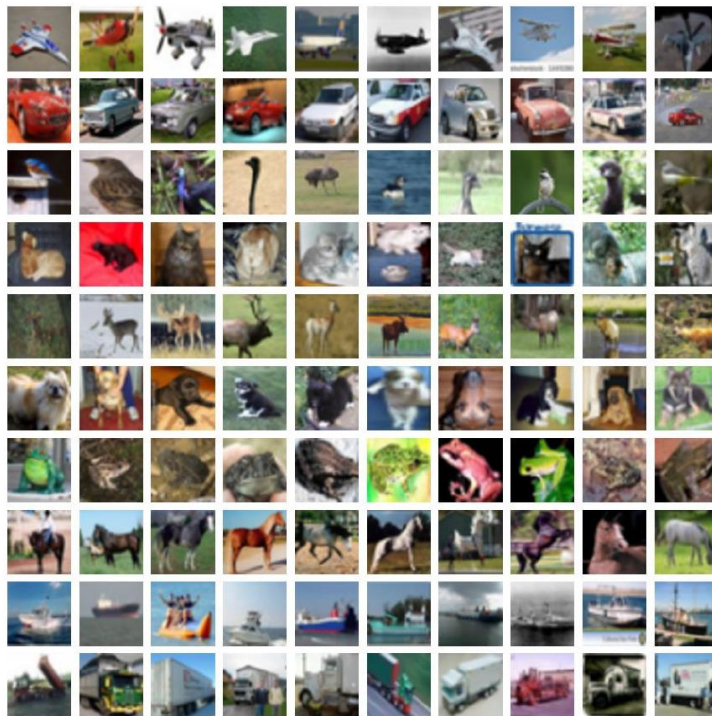
dog

frog

horse

ship

truck

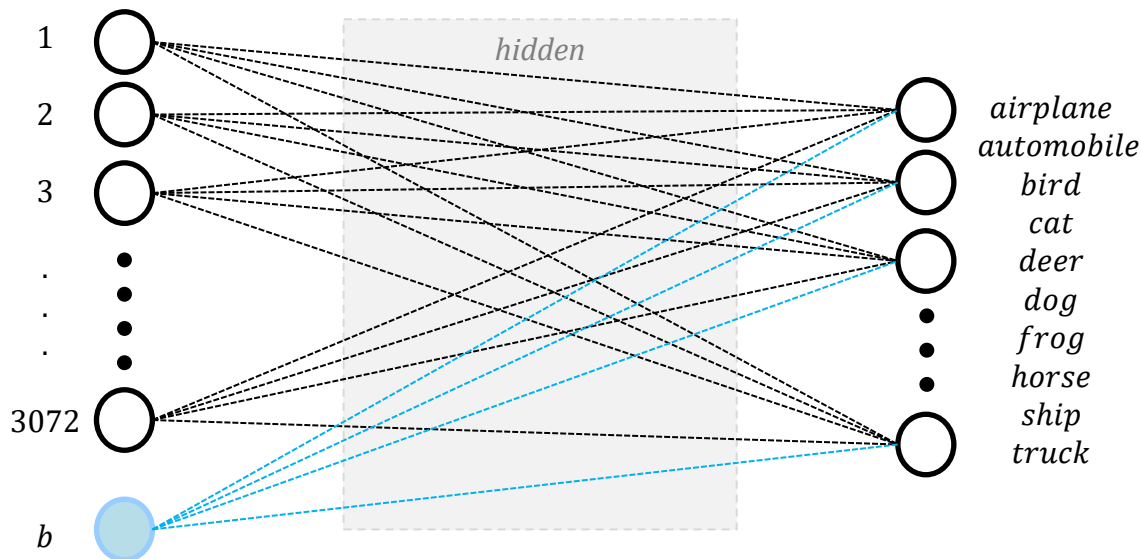


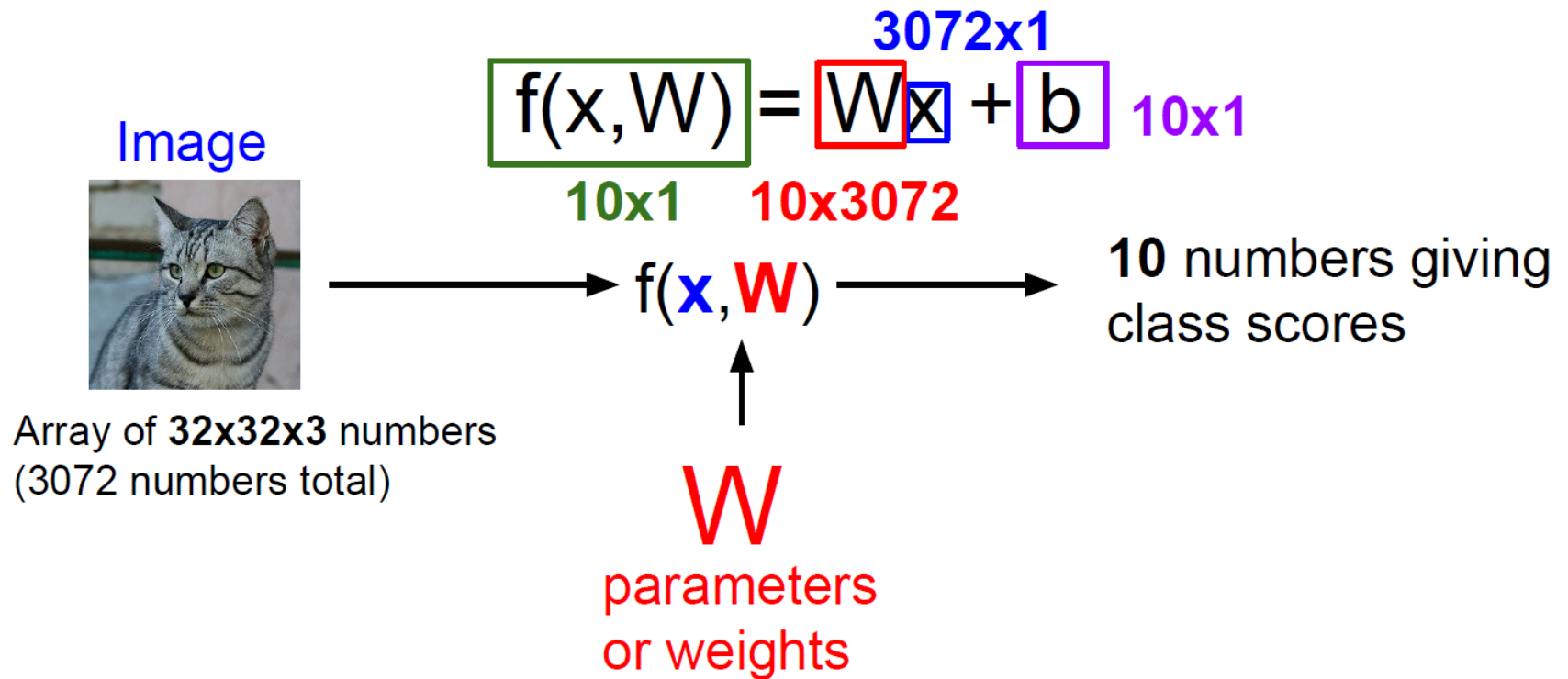
**50,000** training images  
each image is **32x32x3**

**10,000** test images.

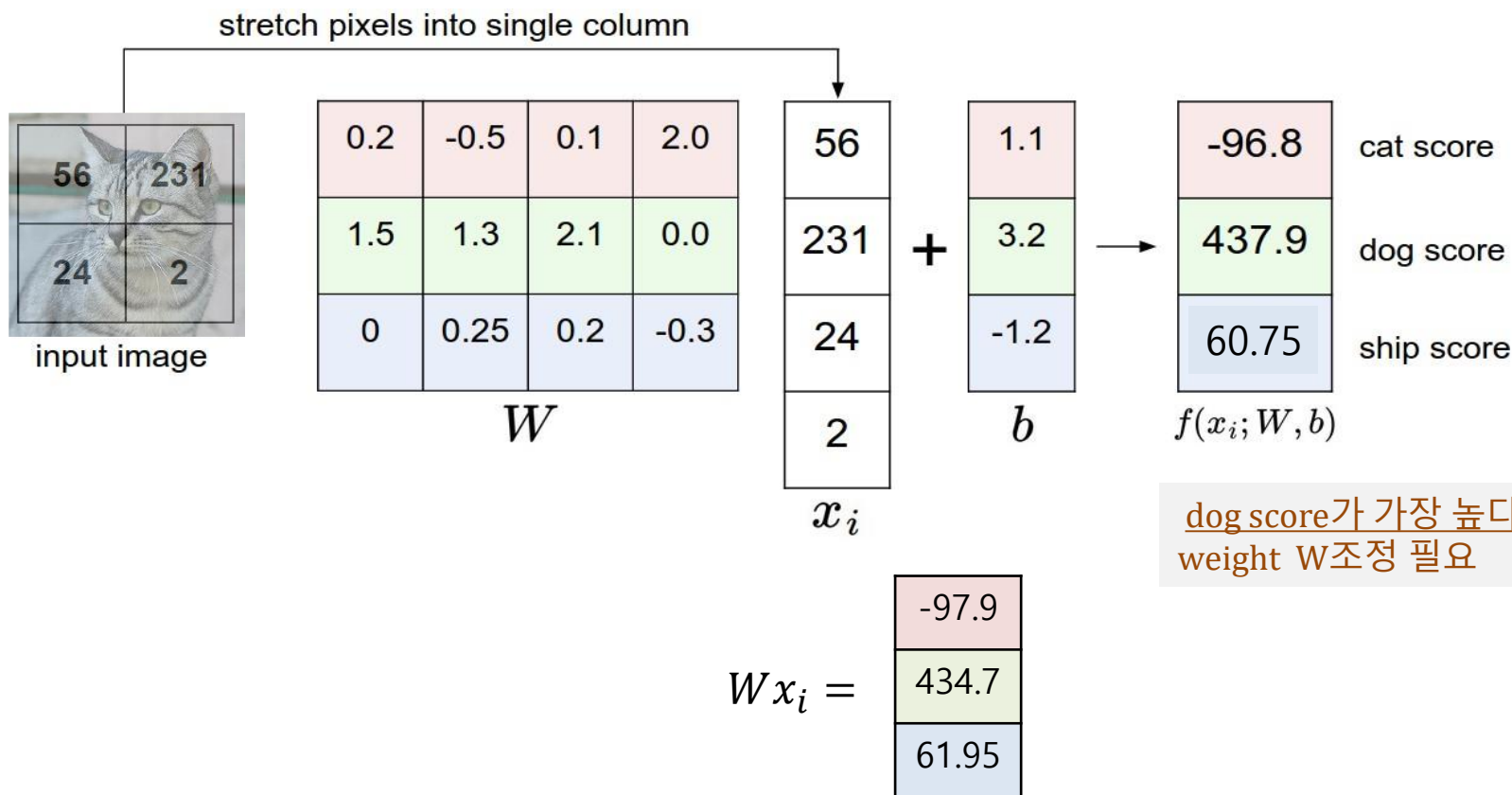
- Training dataset of images :  $x_i \in R^D$ , each associated with a label  $y_i$   
( $i = 1 \dots N$  and  $y_i \in 1 \dots K$ )
- We have  $N$  examples (each with a dimensionality  $D$ ) and  $K$  distinct categories.
- **For example, in CIFAR-10**
  - training set of  $N = 50,000$  images, each with  $D = 32 \times 32 \times 3 = 3072$  pixels, and  $K = 10$ , since there are 10 distinct classes (dog, cat, car, etc).
  - We will now define the score function  $f: R^D \mapsto R^K$  that maps the raw image pixels to class scores.

- $f(x_i, W, b) = Wx_i + b$ 
  - $x_i$  : image. has all of its pixels flattened out to a single column vector of shape  $[D \times 1]$ .
  - $W$  : matrix, of size  $[K \times D]$ , weights
  - $b$  : vector, of size  $[K \times 1]$ , bias vector
- In CIFAR-10
  - $x_i$  : all pixels in the  $i$  – th image flattened into a single  $[3072 \times 1]$  column
  - $W$  :  $[10 \times 3072]$
  - $b$  :  $[10 \times 1]$





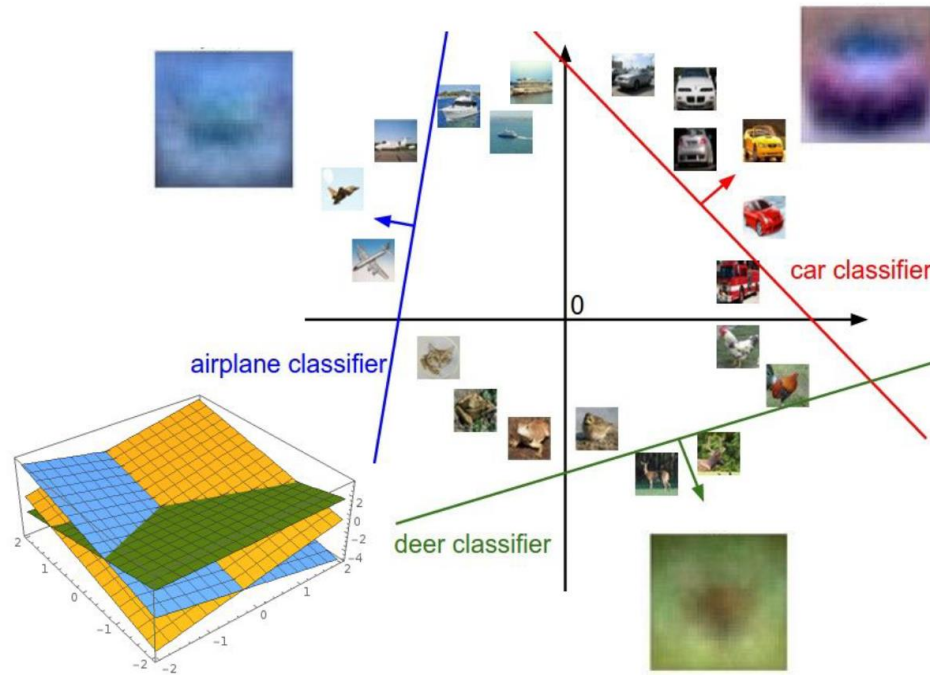
- 간결화를 위하여 4 pixel로 조정
- 3 class : red(cat), green(dog), blue(ship). (rgb채널과 관련없음)
- 이미지 픽셀을 열벡터로 변환 → 행렬 곱셈 → 각 클래스 점수



# Analogy of images as high – dimensional points

63

- CIFAR-10의 각 이미지는  $32 \times 32$  픽셀의 3072 차원 공간에 있는 점.
  - 전체 데이터 셋은 (레이블이 지정된) 점 집합.
- 각 클래스의 점수를 모든 이미지 픽셀의 가중치로 정의  $\rightarrow$  3072차원을 2차원으로 압축



$$f(x, W) = Wx + b$$



Array of **32x32x3** numbers  
(3072 numbers total)

- *Template Class*





## How can we tell whether this $W$ is good or bad?

64

- Defined a (linear) score function
- How can we tell whether this  $W$  is good or bad?



airplane	-3.45	-0.51	3.42
automobile	-8.87	<b>6.04</b>	4.64
bird	0.09	5.31	2.65
cat	<b>2.9</b>	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	<b>-4.34</b>
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14