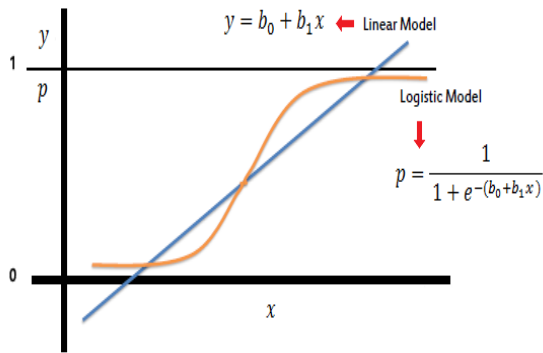


Linear Regression &

Logistic Regression



Fast Campus
Start Deep Learning with TensorFlow



Let's Go to the Deep Learning World!!

쉬운 것부터 시작해봅시다

초등학교 6학년 수학 – 정비례

닭 수 (마리)	1	2	3	4	5
다리 수 (개)	2	4	6	8	10

Diagram illustrating the relationship between the number of chickens (닭 수) and the number of legs (다리 수). The table shows that as the number of chickens increases, the number of legs increases proportionally. Arrows indicate the multiplier for each step:

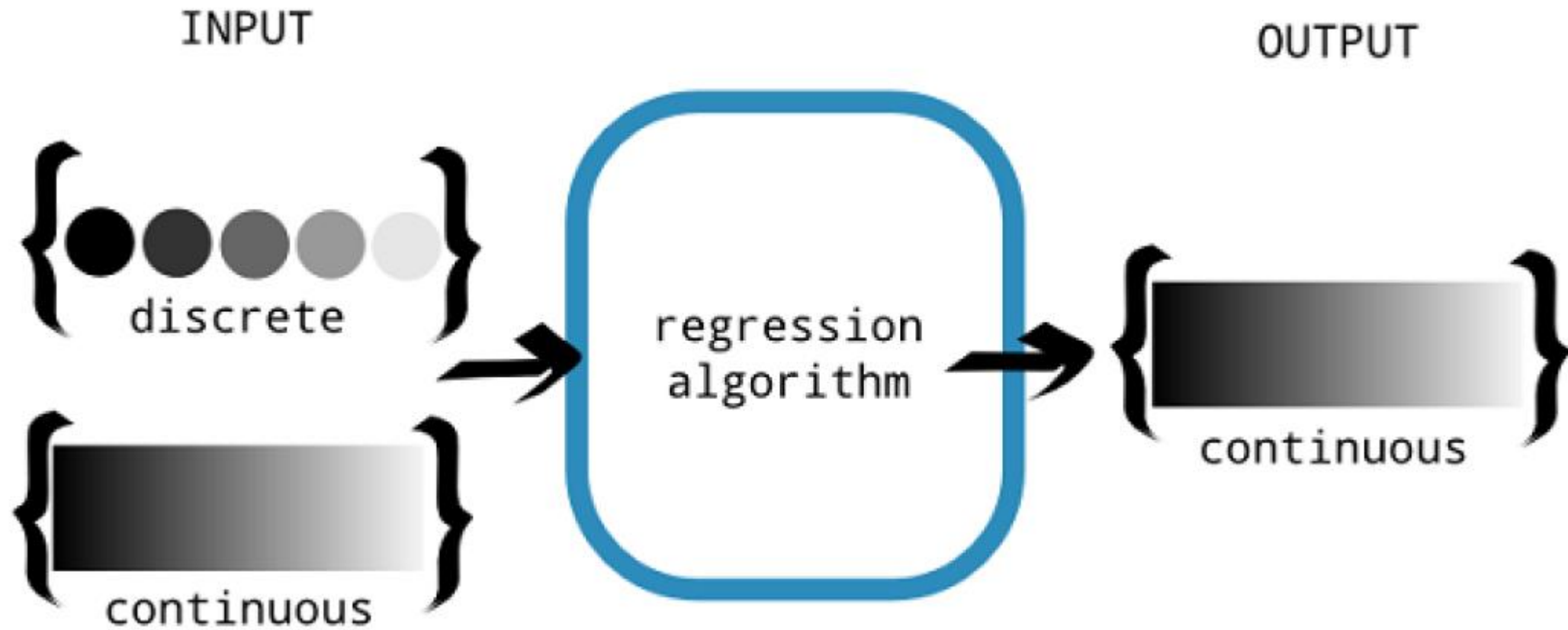
- From 1 to 2 chickens: 2배 (2 times)
- From 2 to 3 chickens: 3배 (3 times)
- From 3 to 4 chickens: 4배 (4 times)
- From 4 to 5 chickens: 5배 (5 times)

Similarly, for the legs:

- From 2 to 4 legs: 2배 (2 times)
- From 4 to 6 legs: 3배 (3 times)
- From 6 to 8 legs: 4배 (4 times)
- From 8 to 10 legs: 5배 (5 times)

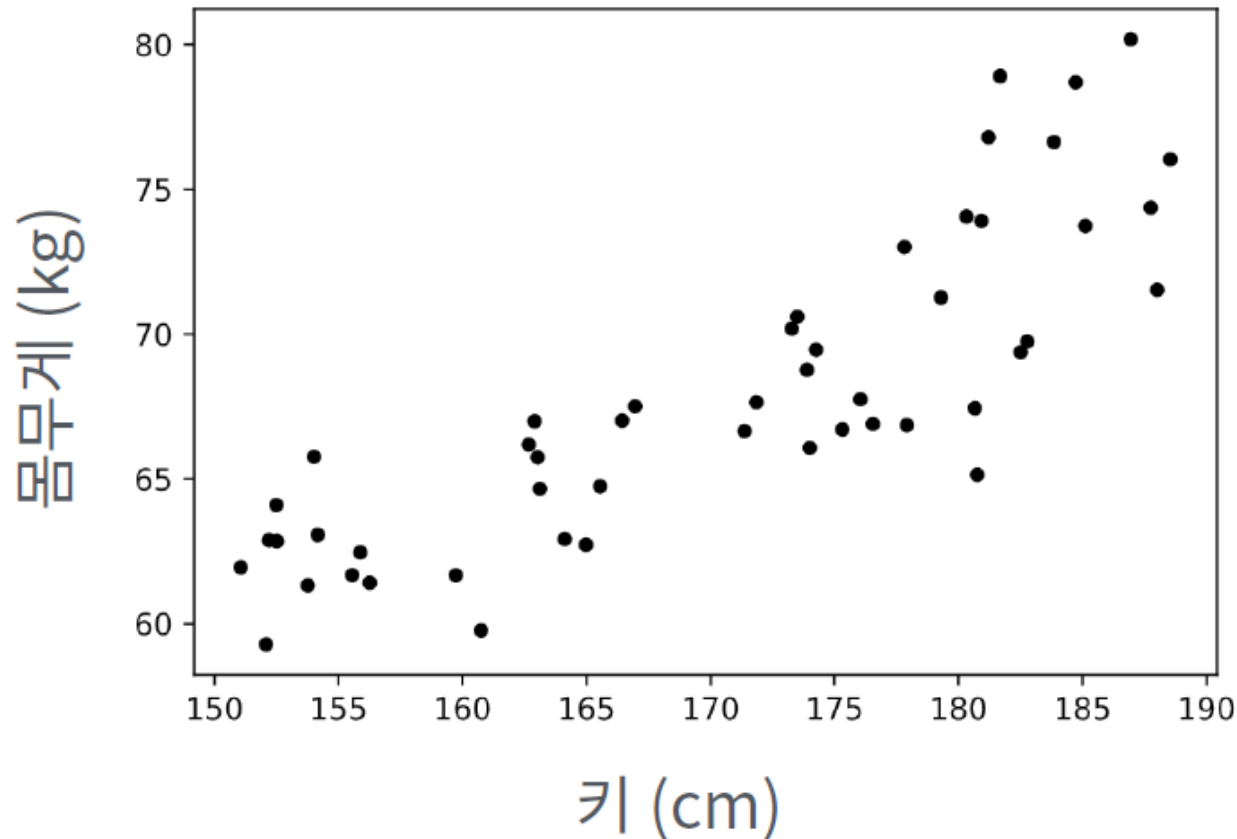


Regression?



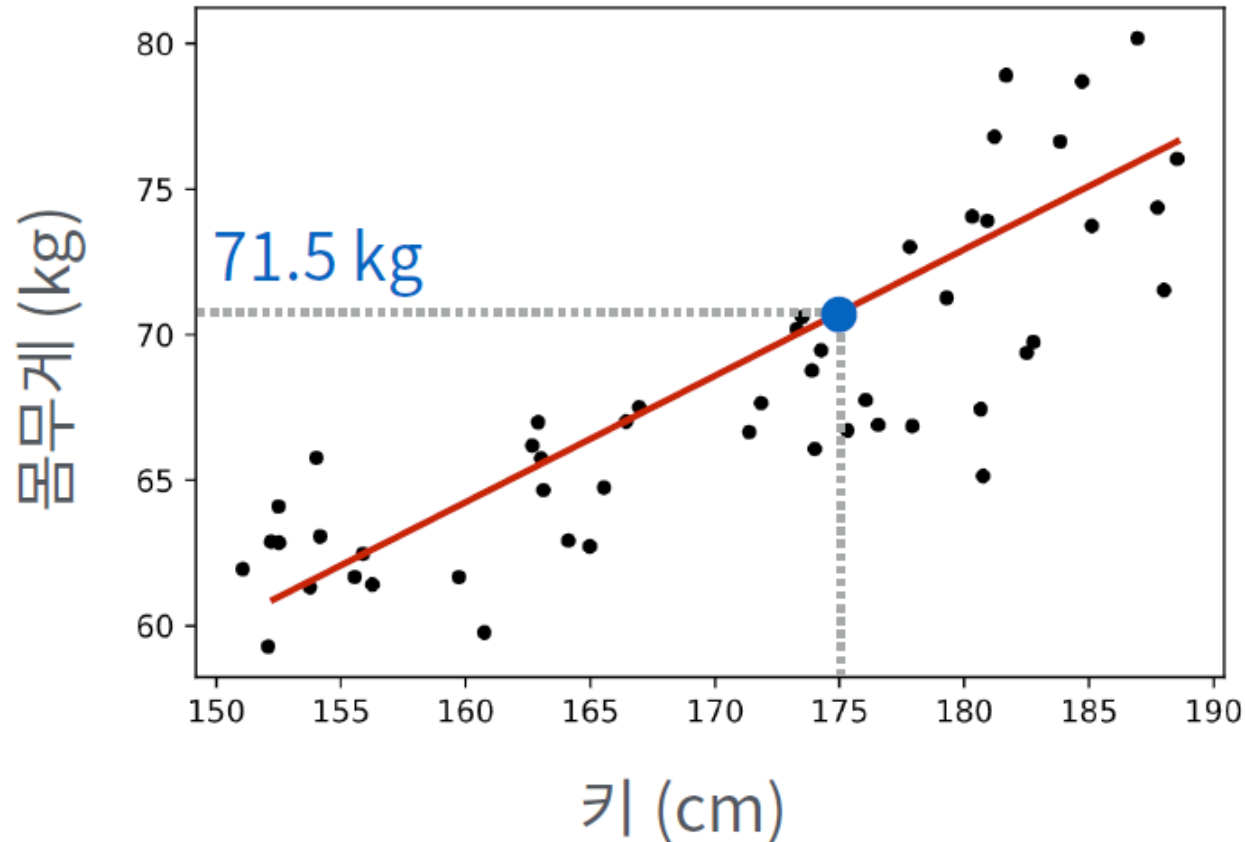
Linear Regression

- 어느 학교 학생들의 신체검사 자료
- 새로 전학온 학생 A의 키가 175cm일 때 예상 몸무게는?



Linear Regression

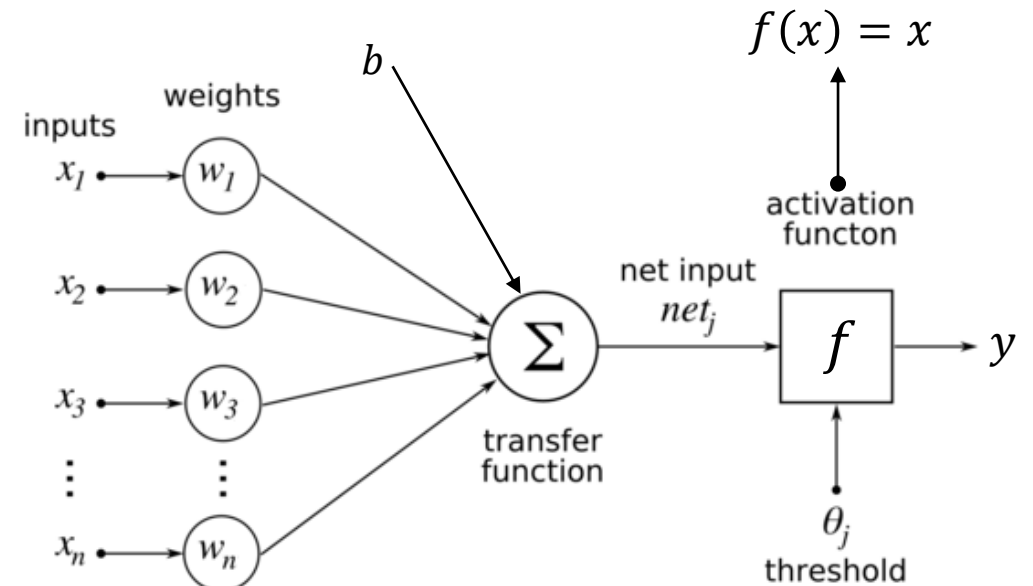
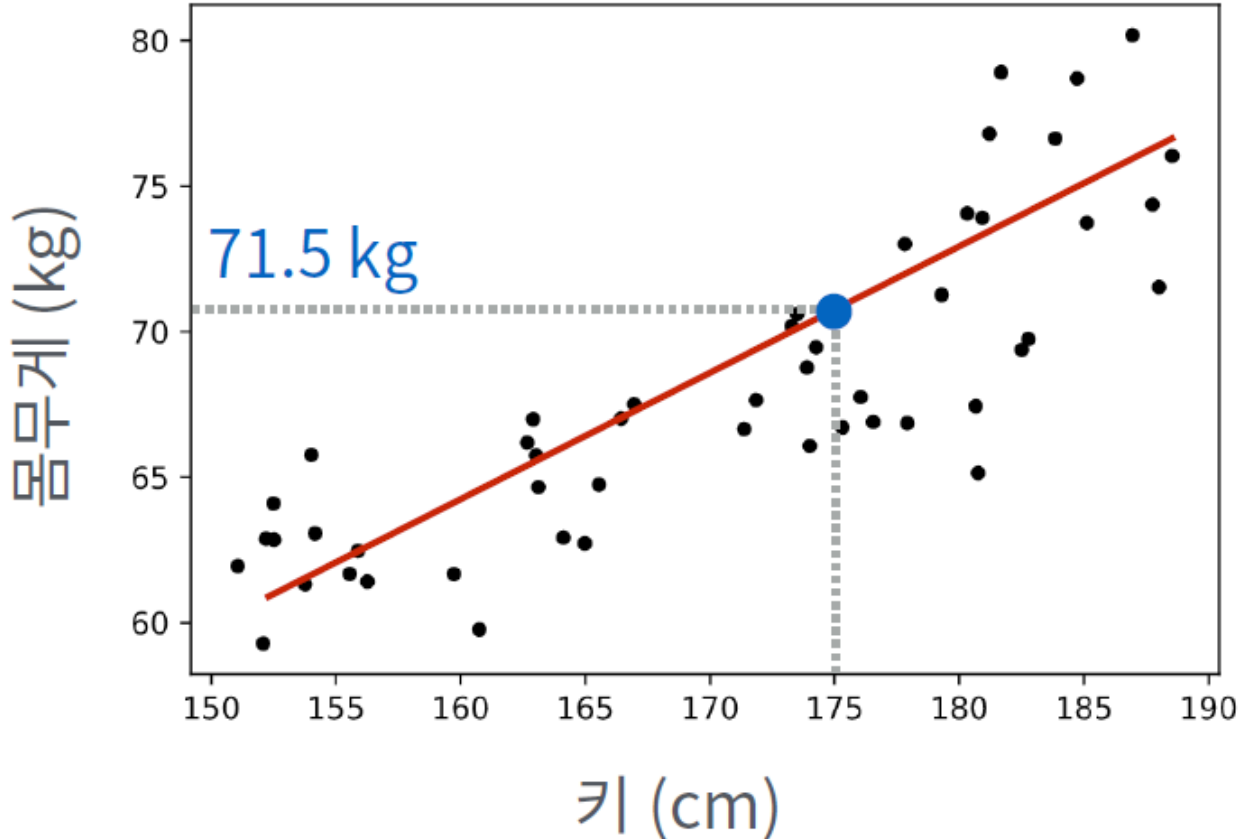
- 어느 학교 학생들의 신체검사 자료
- 새로 전학온 학생 A의 키가 175cm일 때 예상 몸무게는?



Linear Regression

- 선형함수(예 : 1차함수)로 주어진 data를 근사한다

- $y = wx + b$



<Perceptron>

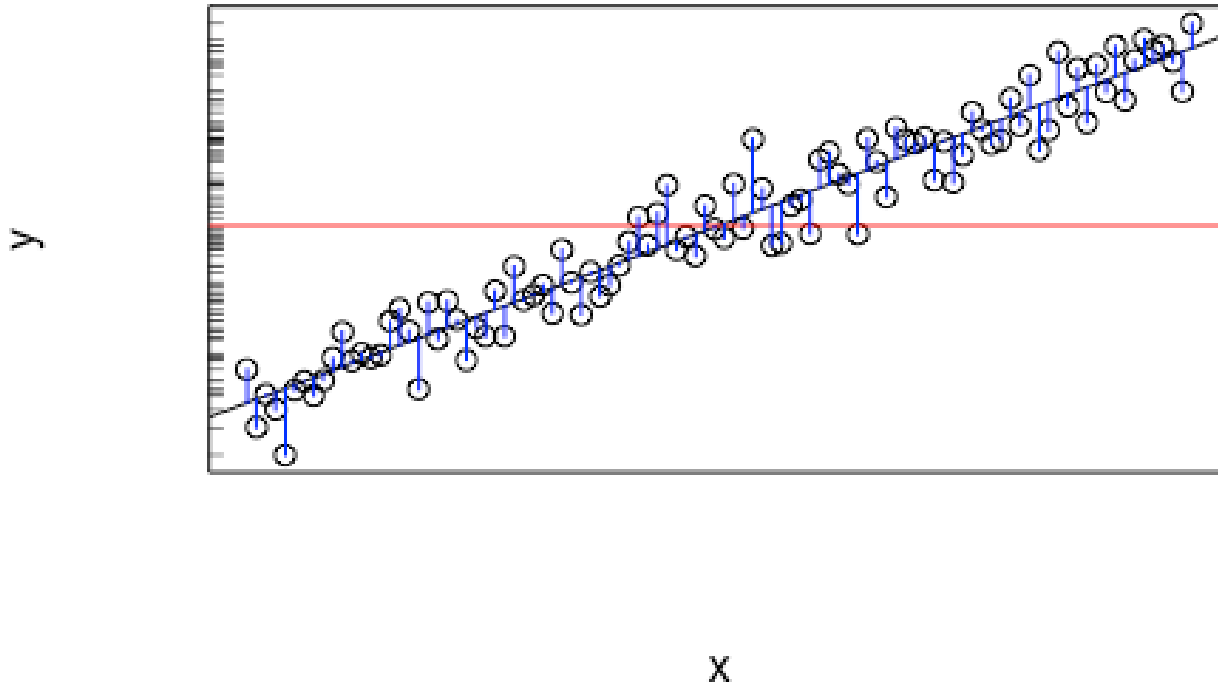
$$y = f(\mathbf{w}\mathbf{x} + b)$$

$$\mathbf{w} = [w_1 \ w_2 \ w_3 \ \dots \ w_n]$$

$$\mathbf{x} = [x_1 \ x_2 \ x_3 \ \dots \ x_n]^T$$

Linear Regression

- 잘 예측했는지 측정할 척도(metric)가 필요함



$$y^* = wx + b \text{ (예측값)}$$

$$\begin{aligned} \text{Cost}(\text{Loss}) &= \sum_i (y_i - y_i^*)^2 \\ &= \sum_i (y_i - wx_i - b)^2 \end{aligned}$$

Linear Regression

- Cost(Loss) 값을 minimize하는 w 와 b 를 구하면 될 텐데.... 어떻게?
 - Random Search – 가능????
 - Cost function을 미분해서 최솟값(미분=0이 되는 점)을 찾자!

약간의 수학을(미분을...) 조금 해야겠습니다

b 구하기

$$L = \sum_i (y_i - wx_i - b)^2$$

$$\frac{\delta L}{\delta b} = \frac{\delta \sum_i (y_i - wx_i - b)^2}{\delta b}$$

$$= -2 \sum_i (y_i - wx_i - b) = ny_{avg} - nwx_{avg} - nb = 0$$

$$\therefore \mathbf{b = y_{avg} - wx_{avg}}$$

w 구하기

$$L = \sum_i (y_i - wx_i - b)^2$$

$$\frac{\delta L}{\delta w} = \frac{\delta \sum_i (y_i - wx_i - b)^2}{\delta w}$$

$$= -2 \sum_i x_i (y_i - wx_i - b) = -2 \sum_i x_i (y_i - wx_i - y_{avg} + wx_{avg})$$

$$= 0$$

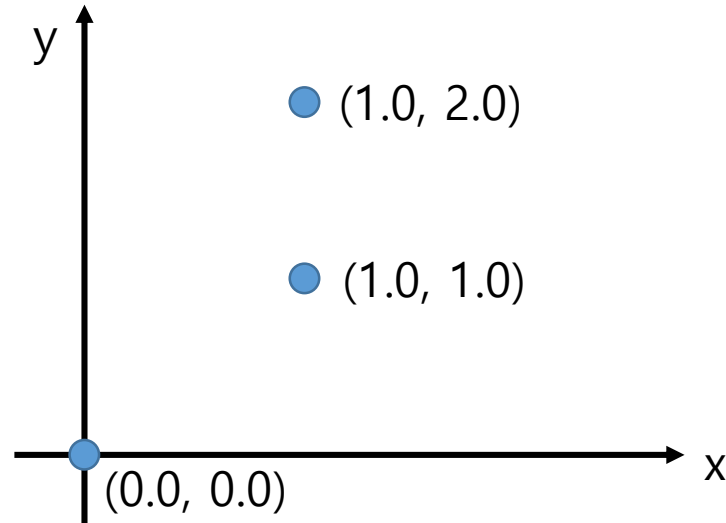
Example

- Find the linear function(f) that best describes the given data
 - $H(x, w_0, w_1) = w_1x + w_0$



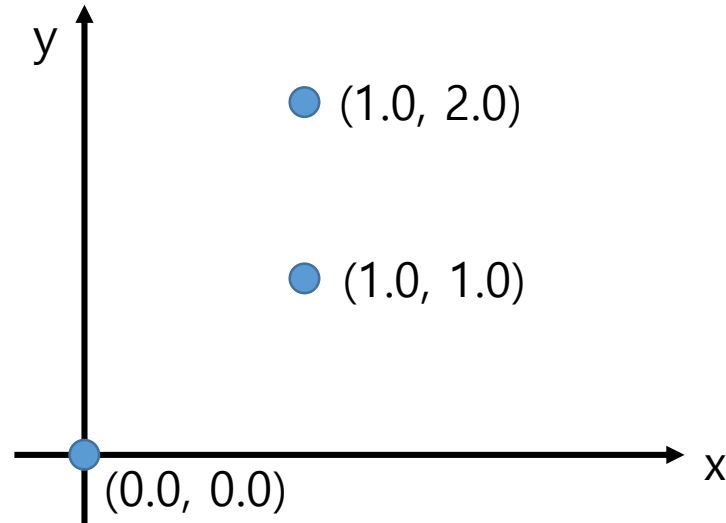
Example

- $H(0, w_0, w_1) \approx 0.0$
- $H(1, w_0, w_1) \approx 1.0$
- $H(1, w_0, w_1) \approx 1.0$



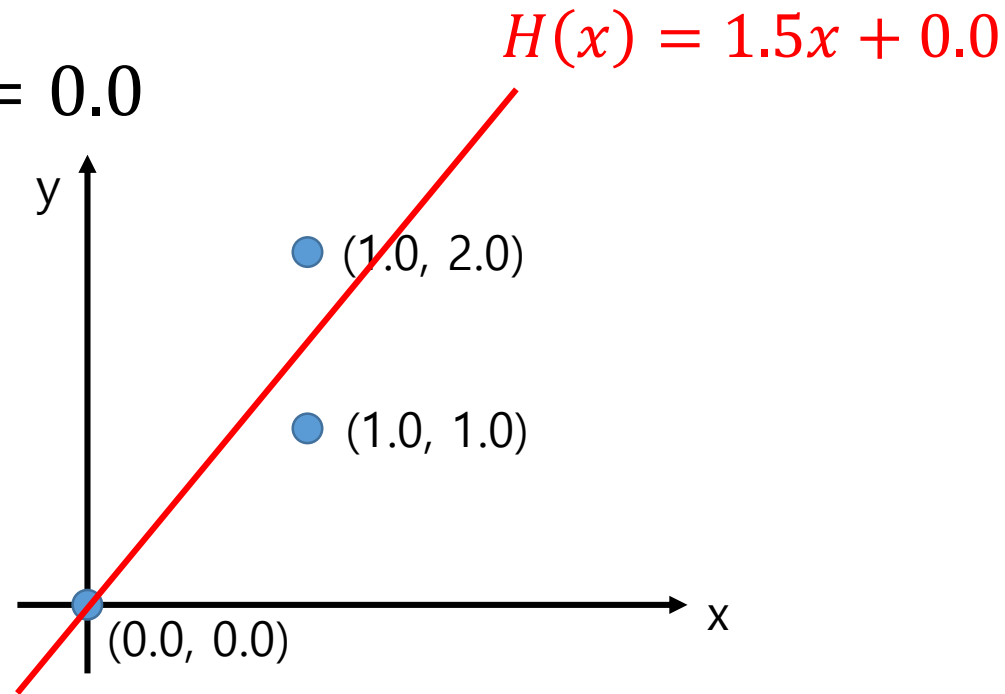
Example

- $L = \sum_i (y_i - w_1 x_i - w_0)^2$
 $= (0.0 - w_1 \cdot 0.0 - w_0)^2 + (1.0 - w_1 \cdot 1.0 - w_0)^2 + (2.0 - w_1 \cdot 1.0 - w_0)^2$
 $= 2w_1^2 + 3w_0^2 - 6w_1 - 6w_0 + 4w_1w_0 + 5$



Example

- $\frac{\partial L}{\partial w_1} = 4w_1 + 4w_0 - 6 = 0$
- $\frac{\partial L}{\partial w_0} = 4w_1 + 6w_0 - 6 = 0$
- $\therefore w_1 = 1.5, w_0 = 0.0$



Multi Variable Linear Regression

- x 가 scalar값(1개)가 아니라 vector가 된다면??

- Input

- X_1 : Facebook 광고료
- X_2 : TV 광고료
- X_3 : 신문 광고료

- Output

- 판매량

FB	TV	신문	판매량
X_1	X_2	X_3	Y
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9
8.7	48.9	75	7.2
57.5	32.8	23.5	11.8
⋮	⋮	⋮	⋮

Multi Variable Linear Regression

$$\begin{aligned}\mathbf{w}_{\text{lin}} &= \underset{\mathbf{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} E_{\text{in}}(\mathbf{w}) \\ &= \underset{\mathbf{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \frac{1}{N} \|X\mathbf{w} - \mathbf{y}\|^2 \\ &= \underset{\mathbf{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \frac{1}{N} (\mathbf{w}^\top X^\top X \mathbf{w} - 2\mathbf{w}^\top X^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y})\end{aligned}$$

Multi Variable Linear Regression

- Find w that minimize $E_{in}(w)$ by requiring

$$\nabla E_{in}(\mathbf{w}) = \mathbf{0}$$

- From previous equation

$$\nabla E_{in}(\mathbf{w}) = \frac{2}{N}(X^T X \mathbf{w} - X^T \mathbf{y})$$

$$\begin{aligned}\mathbf{w}_{lin} &= \operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^{d+1}} E_{in}(\mathbf{w}) \\ &= \operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^{d+1}} \frac{1}{N} \|X \mathbf{w} - \mathbf{y}\|^2 \\ &= \operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^{d+1}} \frac{1}{N} (\mathbf{w}^T X^T X \mathbf{w} - 2 \mathbf{w}^T X^T \mathbf{y} + \mathbf{y}^T \mathbf{y})\end{aligned}$$

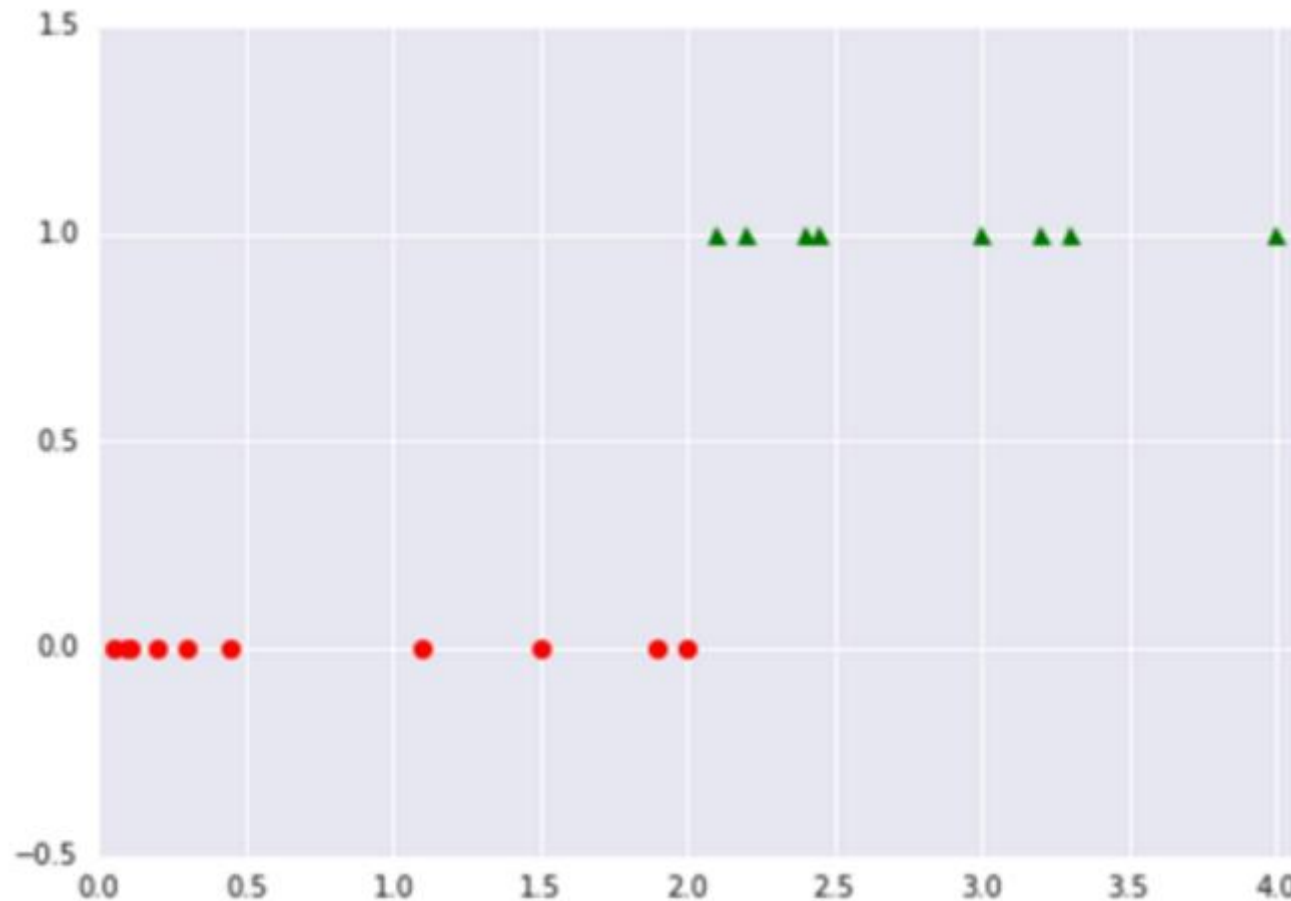
Multi Variable Linear Regression

- Two scenarios
 - If $X^T X$ is invertible
$$\mathbf{w} = (X^T X)^{-1} X^T$$
 - If $X^T X$ is not invertible
Pseudo-inverse defined, but no unique solution

Classification도 할 수 있지 않을까요?

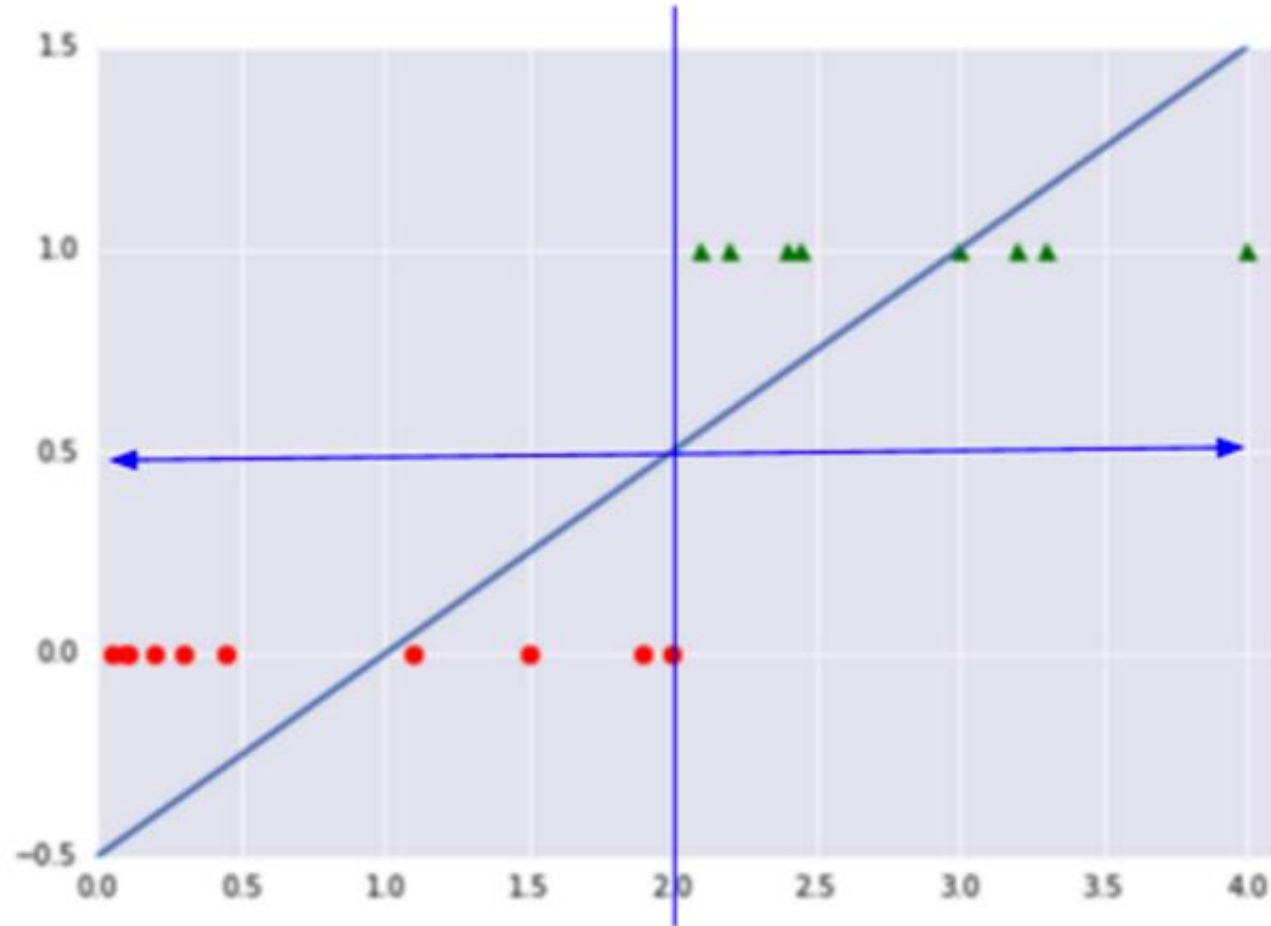
Binary Classification

- 종양의 크기에 따른 양성/음성 판별 문제
 - 1: 양성(암), 0: 음성(정상)



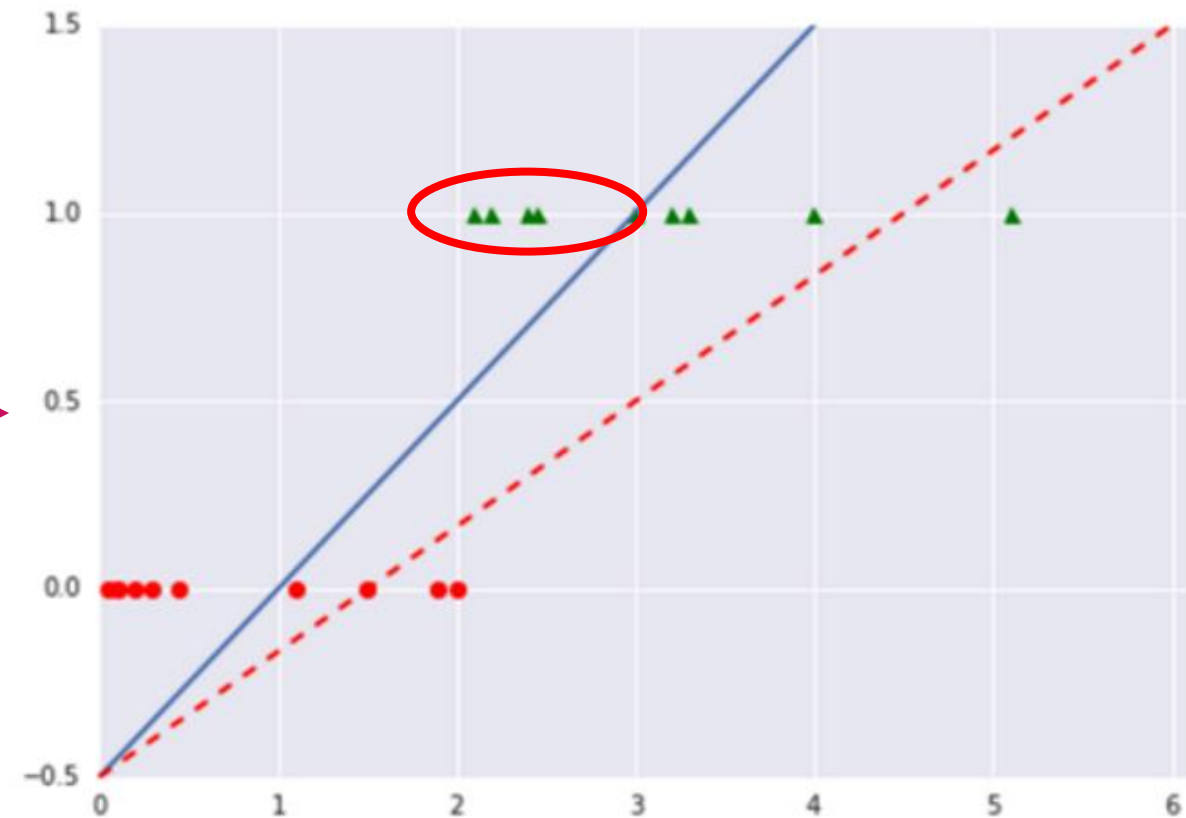
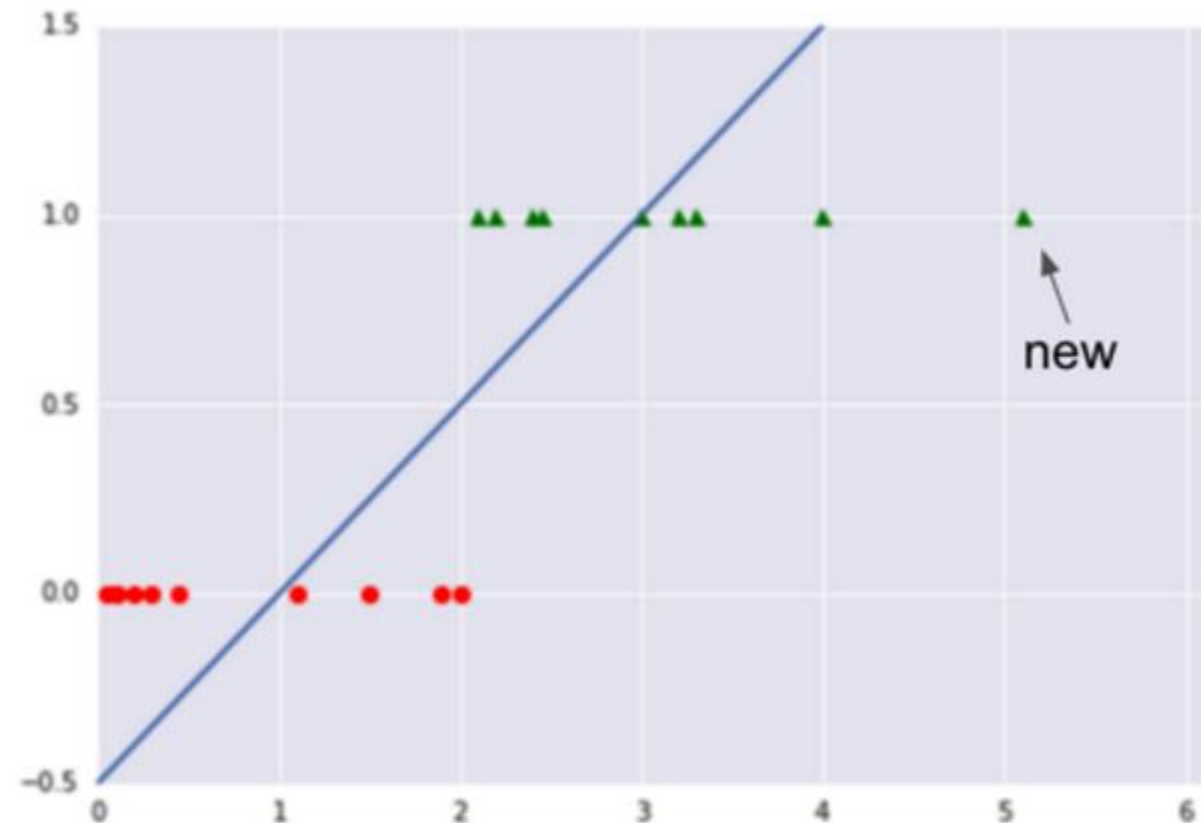
Binary Classification

- Linear Regression으로 해봅시다
 - Regression 예측값이 0.5 이상이면 양성, 0.5 이하면 음성으로 판별



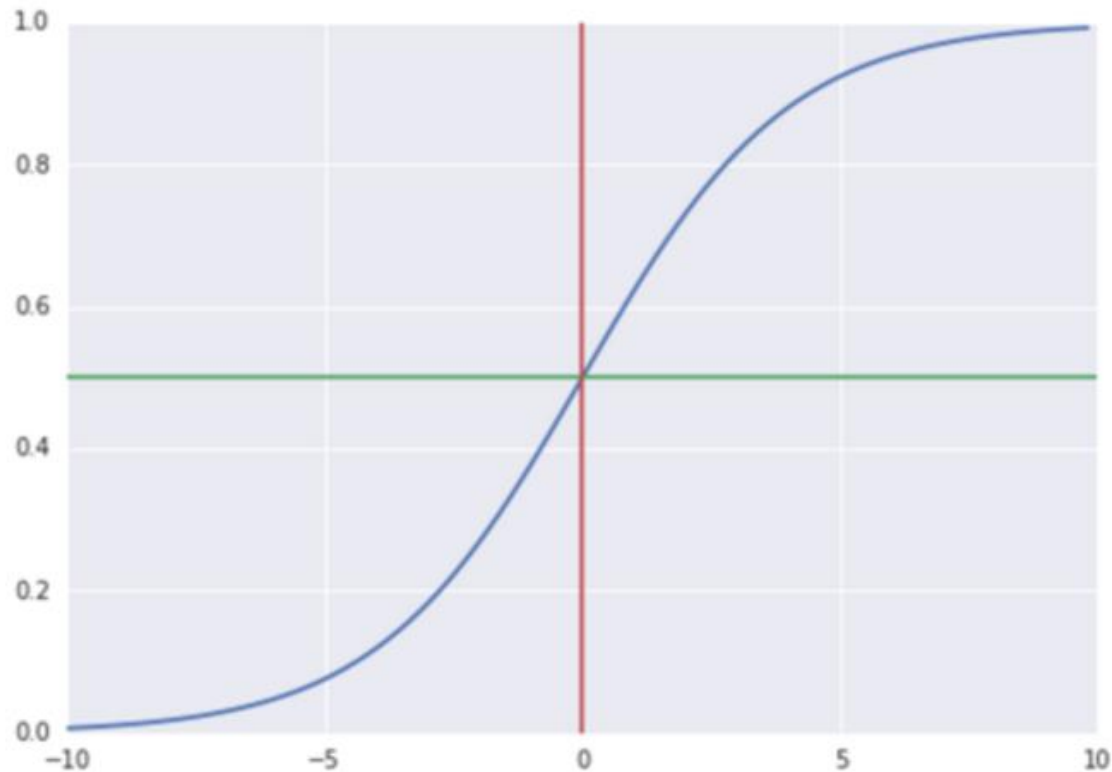
Binary Classification

- 종양의 크기가 매우 큰 data(outlier)가 추가된 경우



Binary Classification

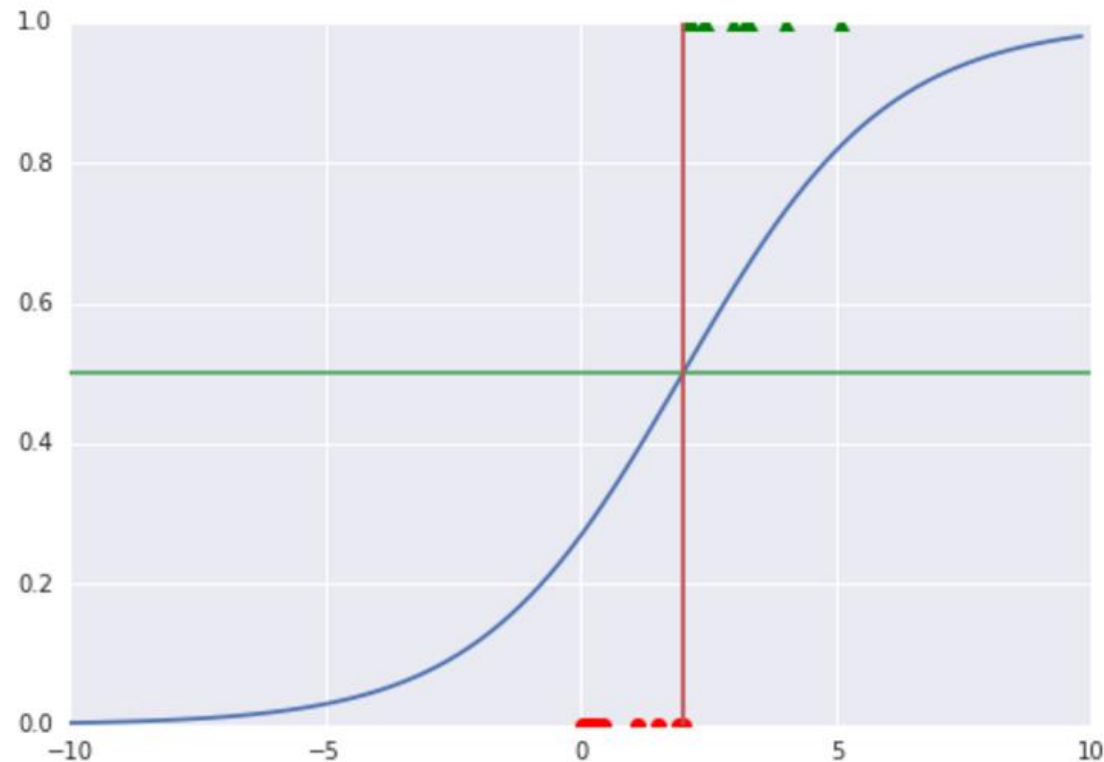
- 아주 크거나 아주 작은 data에 영향을 많이 받지 않았으면 좋겠다
 - Binary classification에 맞게 0에서 1사이 값으로 나오면 좋겠다
- Sigmoid 를 써보자



Logistic Regression

- Linear Regression 식에 Sigmoid 함수를 통과시킨 것

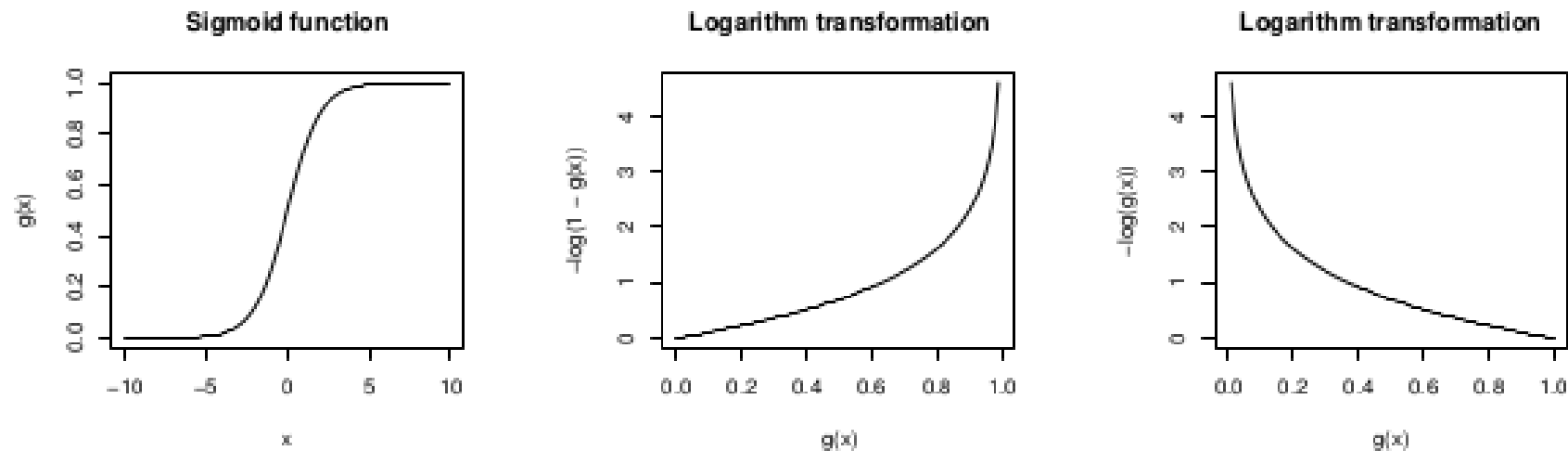
$$\blacksquare H(x) = \sigma(\mathbf{w}^T \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + b)}} = P(y|\mathbf{x})$$



Logistic Regression

- 새로운 Cost(Loss) function을 정의(maximum likelihood estimation)

$$\text{cost}(W) = -\frac{1}{m} \sum y \log(H(x)) + (1 - y) \log(1 - H(x))$$



(a) Sigmoid function.

(b) Cost for $y = 0$.

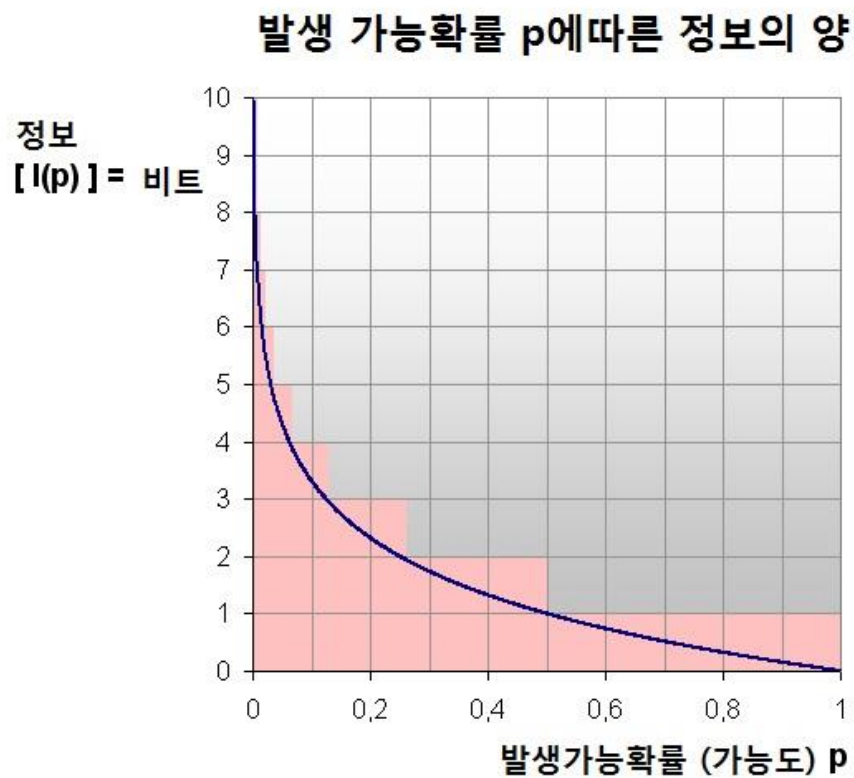
(c) Cost for $y = 1$.

Figure B.1: Logarithmic transformation of the sigmoid function.

entropy

- Information Theory에서 Entropy란 정보량의 기댓값(평균)이다

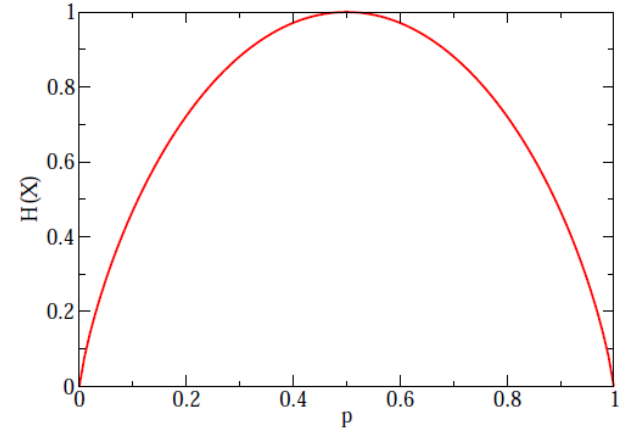
$$H(X) = E[I(X)] = E[-\ln(P(X))].$$



- 항상 일어나는 사건이라면 $p=1$ 이고 이것이 가지고 있는 정보량은 0이다.
- 일어날 확률이 적을 수록 많은 정보를 담고 있다.

Entropy

$$\begin{aligned} H(X) &= E[I(X)] = E[-\ln(P(X))]. \\ &= \sum_{i=1}^n P(x_i) I(x_i) = - \sum_{i=1}^n P(x_i) \log_b P(x_i), \end{aligned}$$



Ex) 동전 던지기

$$P(\text{앞}) = 1/2$$

$$P(\text{뒤}) = 1/2$$

$$H(X) = -(0.5 * \log_2 0.5 + 0.5 * \log_2 0.5) = 0.5 + 0.5 = \mathbf{1}$$

이 확률 분포를 표현하기 위하여 평균 1 bit가 필요하다는 의미!

Cross Entropy

True distribution p ,
Model이 예측한 distribution을 q 라고 하면

Cross entropy를 minimize 하는 것은 p 와 q 의 분포가 가까워지도록 만드는 것과 같다

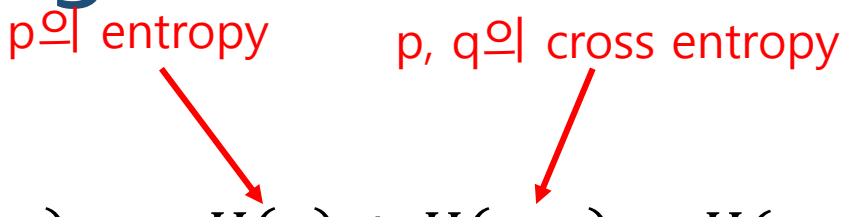
$$\begin{aligned} H(p, q) &= - \sum_i p_i \log q_i \\ &= - \sum_i p_i \log q_i - \sum_i p_i \log p_i + \sum_i p_i \log p_i \\ &= \boxed{H(p)} + \sum_i p_i \log p_i - \sum_i p_i \log q_i \\ &= H(p) + \sum_i p_i \log \frac{p_i}{q_i} \\ &= H(p) + D_{KL}(p \parallel q) \end{aligned}$$

$H(p)$ p 자체의 entropy(불변)

$D_{KL}(p \parallel q)$ p 를 기준으로 q 가 얼마나 다른가
(p 의 distribution과 q 의 distribution의 거리)

Kullback–Leibler(KL) Divergence

- 두 확률 분포간의 거리(?)를 나타냄

- $D_{KL}(p \parallel q) = p \log \frac{p}{q} = p \log p + (-p \log q) = -H(p) + H(p, q) = H(p, q)$
 - 
- p는 label, q는 network output 이라고 할 경우, network의 목표는 label이 나타내는 확률 분포 p와 최대한 가까운 q를 학습하고자 하는 것임
- 일반적으로 label은 one-hot encoding 하기 때문에, 각 label의 값이 확률이 라고 생각하면 p의 entropy, $H(p) = 0$
- $D_{KL}(p \parallel q) > 0$

Likelihood

- Likelihood – 2개의 class($y = +1$, $y = -1$)의 경우,

$$P(y|\mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{for } y = +1 \\ 1 - h(\mathbf{x}) & \text{for } y = -1 \end{cases} \quad h(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + b)$$

$$P(y|\mathbf{x}) = h(\mathbf{x})\mathbb{I}^{y=+1}(1 - h(\mathbf{x}))\mathbb{I}^{y=-1}$$

- Likelihood of data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

$$\prod_{n=1}^N P(y_n|\mathbf{x}_n) = \prod_{n=1}^N h(\mathbf{x}_n)\mathbb{I}^{y_n=+1}(1 - h(\mathbf{x}_n))\mathbb{I}^{y_n=-1}$$

Negative Log-Likelihood(NLL)

- Maximize likelihood = Minimize negative log-likelihood(NLL)

$$\begin{aligned} NLL(\mathbf{w}) &\propto -\frac{1}{N} \log \left\{ \prod_{n=1}^N P(y_n | \mathbf{x}_n) \right\} \\ &= -\frac{1}{N} \log \left\{ \prod_{n=1}^N h(\mathbf{x}_n)^{\mathbb{I}[y_n=+1]} (1 - h(\mathbf{x}_n))^{\mathbb{I}[y_n=-1]} \right\} \\ &= \frac{1}{N} \sum_{n=1}^N \left\{ \mathbb{I}[y_n = +1] \log \frac{1}{h(\mathbf{x}_n)} + \mathbb{I}[y_n = -1] \log \frac{1}{1 - h(\mathbf{x}_n)} \right\} \end{aligned}$$

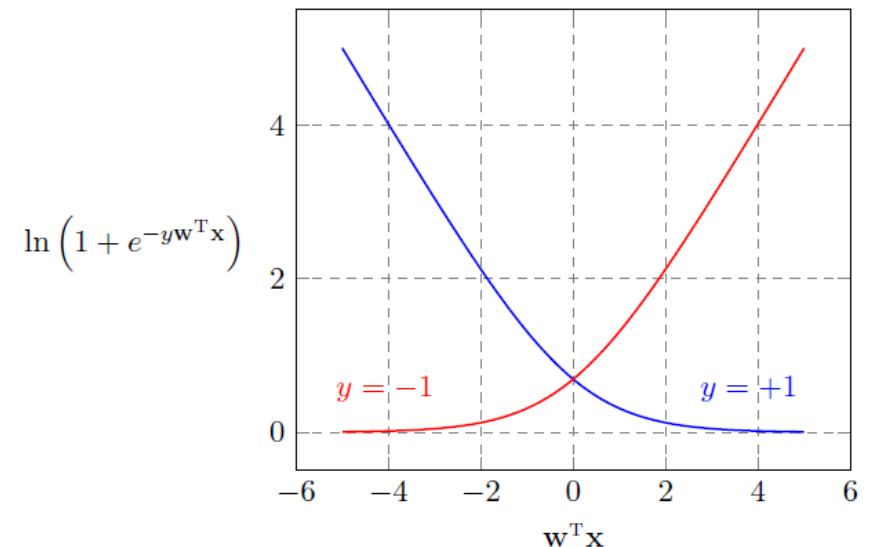
Cross-Entropy!

Minimizing NLL

- Minimize $-\frac{1}{N} \log \left\{ \prod_{n=1}^N P(y_n | \mathbf{x}_n) \right\} = \frac{1}{N} \sum_{n=1}^N \log \frac{1}{P(y_n | \mathbf{x}_n)}$
- We can define loss(error) function as below

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \log \left(1 + e^{-y_n \mathbf{w}^\top \mathbf{x}_n} \right)$$

$$\mathbf{e}(h(\mathbf{x}_n), y_n) = \ln \left(1 + e^{-y_n \mathbf{w}^\top \mathbf{x}_n} \right)$$



Minimizing NLL

- Unfortunately, not easy to manipulate analytically

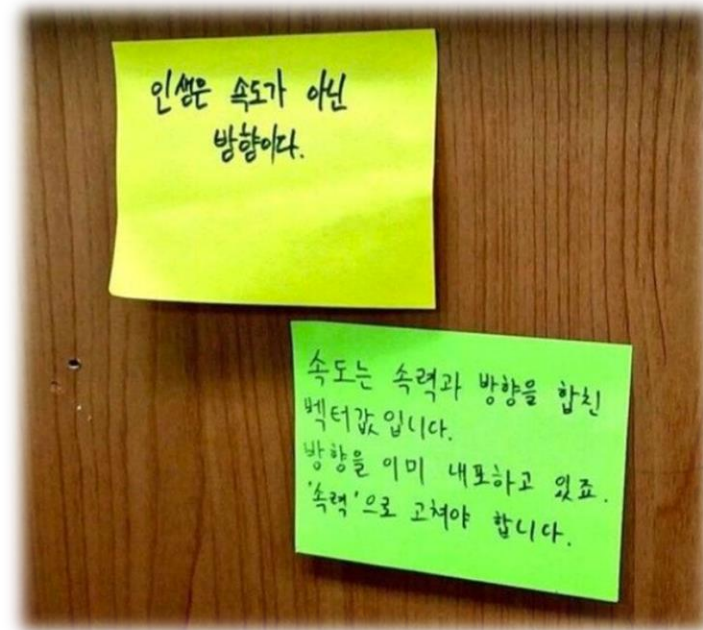
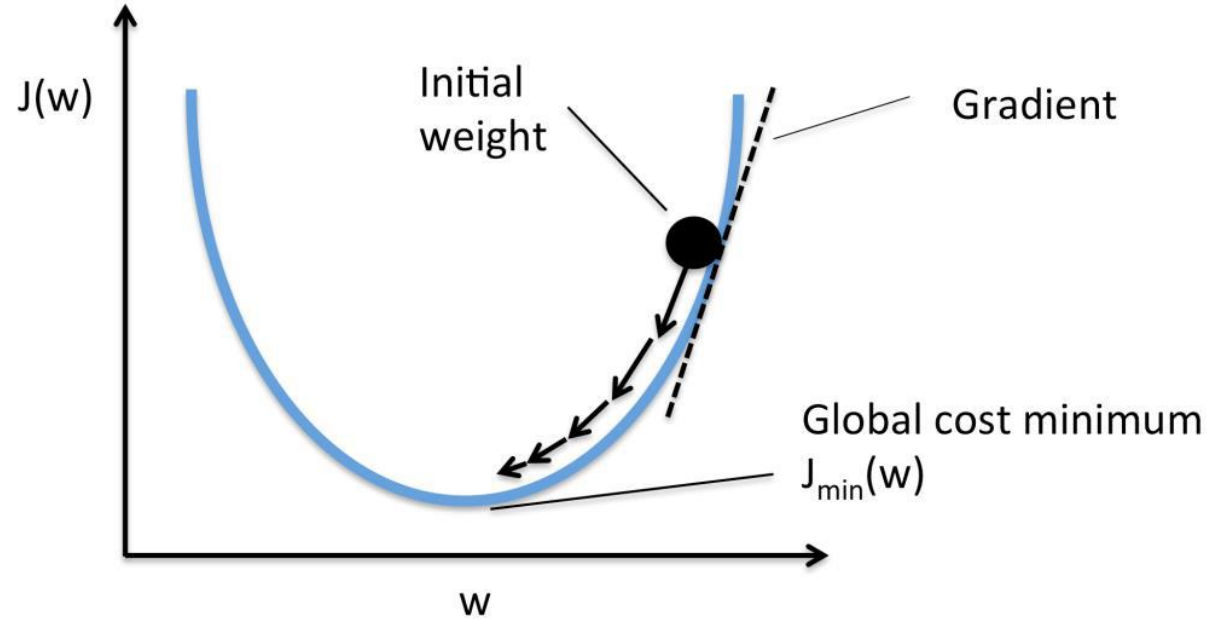
$$\begin{aligned}\nabla E_{\text{in}}(\mathbf{w}) &= -\frac{1}{N} \sum_{n=1}^N \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^T \mathbf{x}_n}} \\ &= \frac{1}{N} \sum_{n=1}^N -y_n \mathbf{x}_n \theta(-y_n \mathbf{w}^T \mathbf{x}_n)\end{aligned}$$

- We need **iterative** optimization

Gradient Descent

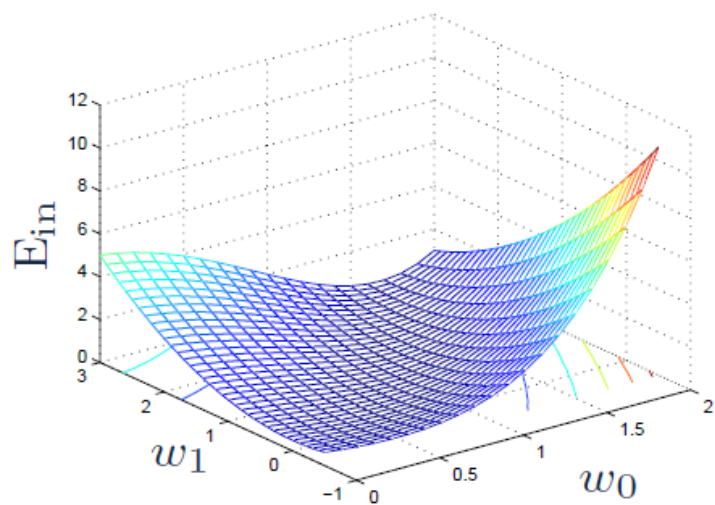
$$w_{new} = w - \eta \frac{\delta L}{\delta w}$$

- 방향 : 그 지점에서의 - gradient
- 속력(보폭) : learning rate(η)



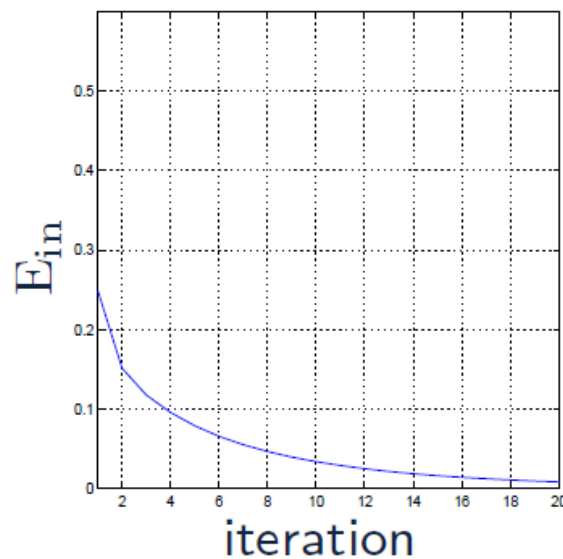
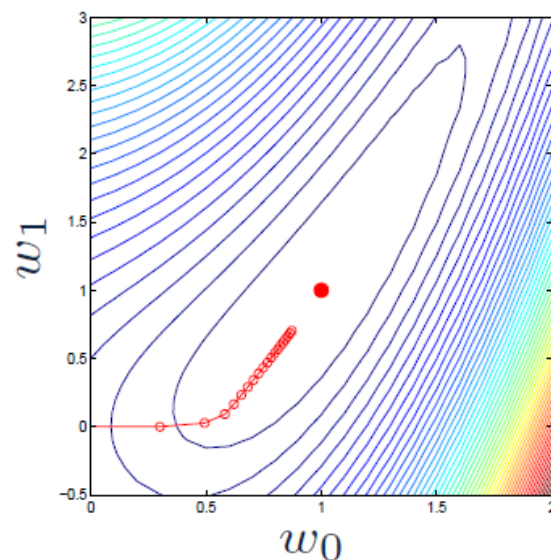
Gradient Decent Example

- Global minimum 0 at (1,1)



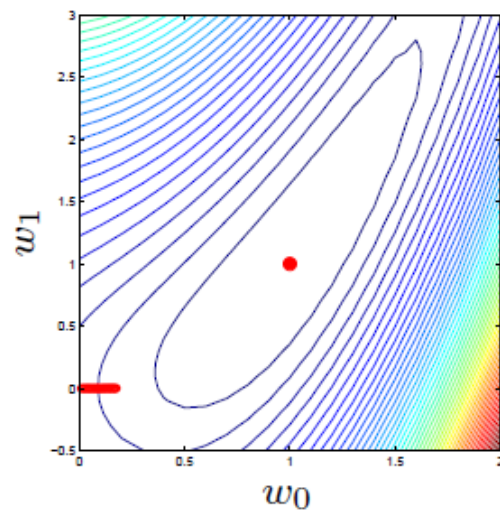
- Start at (0,0)

- # iterations (steps) = 20
- step size $\eta = 0.3$

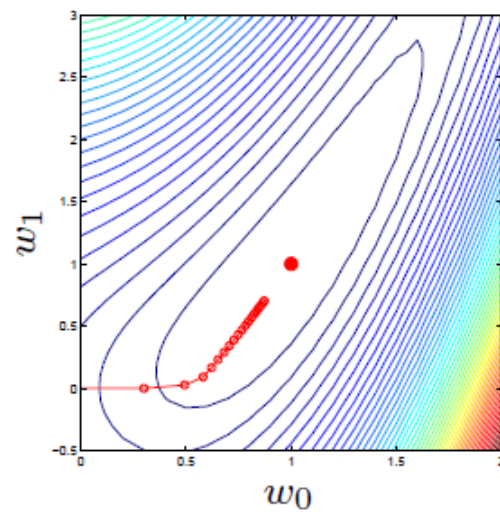


Learning Rate

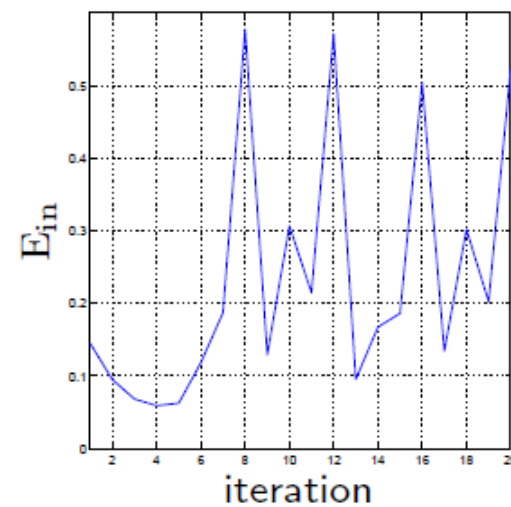
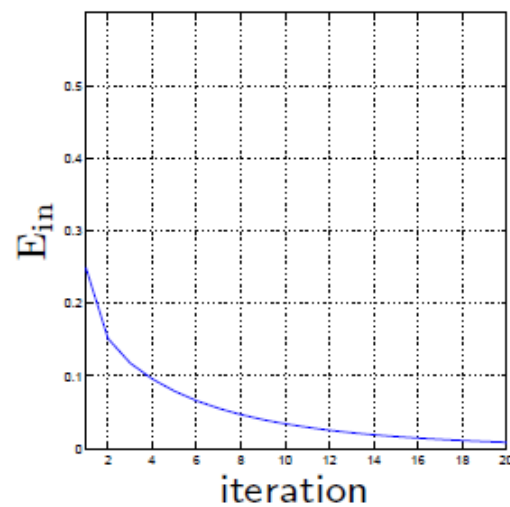
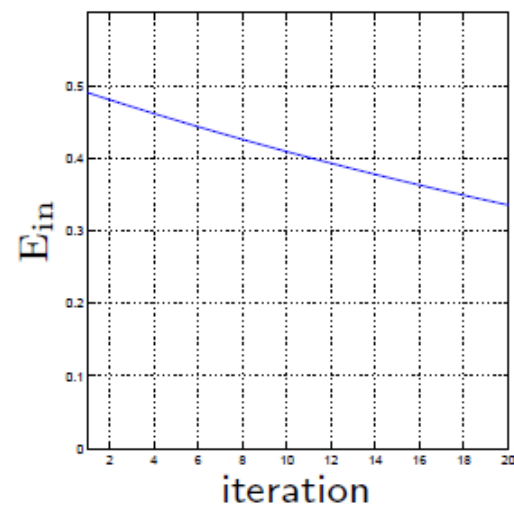
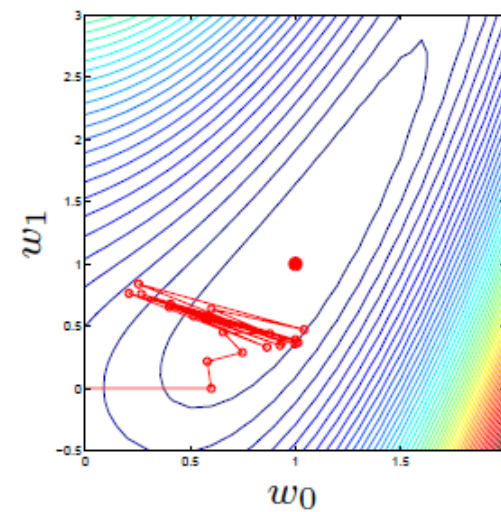
$\eta = 0.01$



$\eta = 0.3$



$\eta = 0.6$



Stochastic Gradient Descent, Mini-batch Training

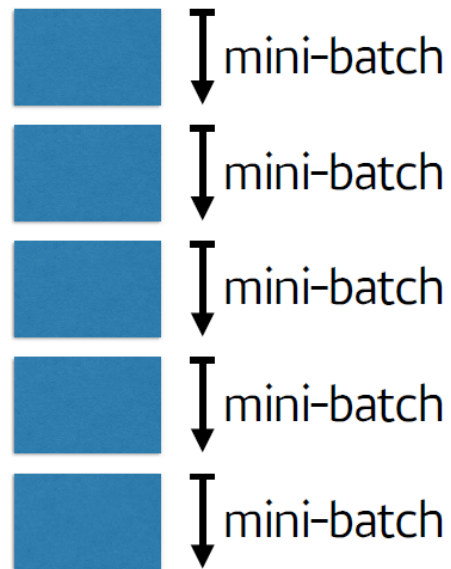
- Data가 너무 많아서 한번에 다 넣고 학습하면 시간도 오래걸리고, memory도 부족하게 됨

Gradient Decent



전부다 읽고나서
최적의 1스텝 간다.

Stochastic Gradient Decent



작은 토막마다
일단 1스텝간다.

