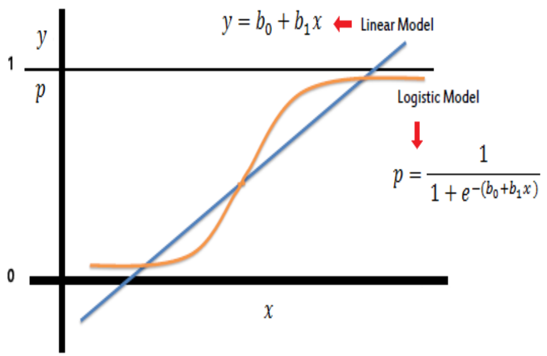
Linear Regression & Logistic Regression



Let's Go to the Deep Learning World!!

쉬운 것부터 시작해봅시다

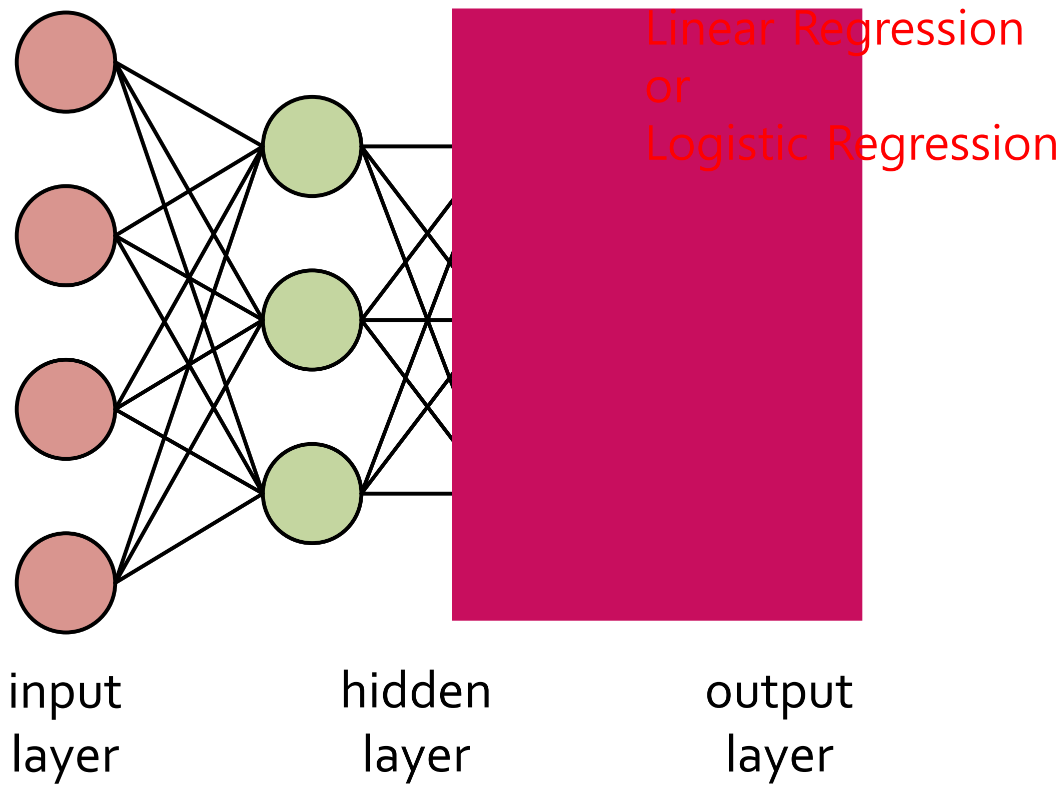
초등학교 6학년 수학 – 정비례

2배 3배 4배

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

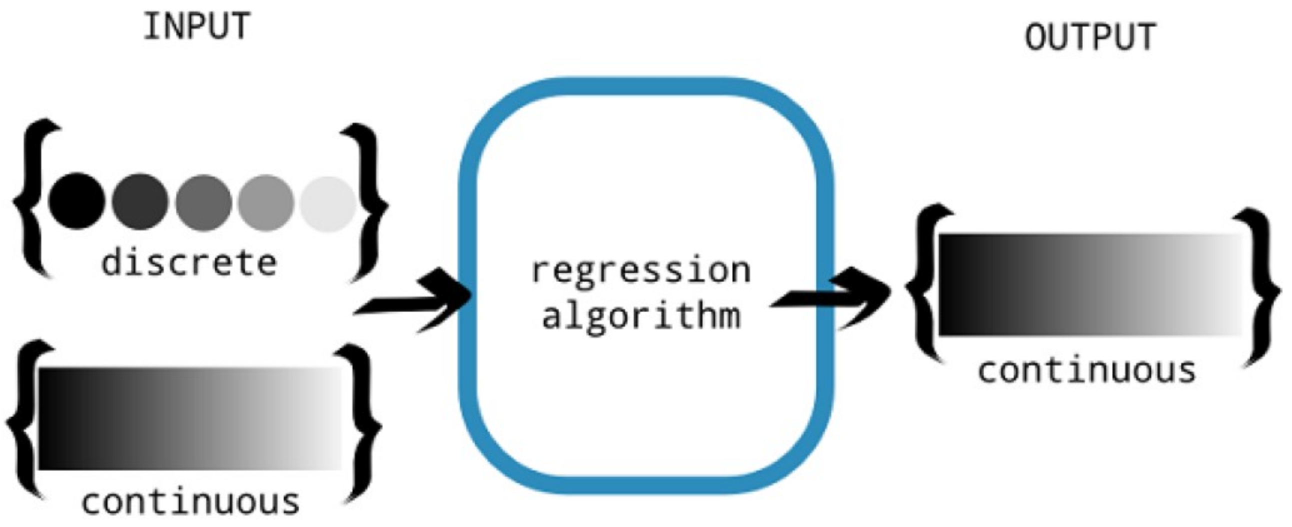
2배 3배 4배

Deep Learning Uses
Linear Regression/Logistic Regression



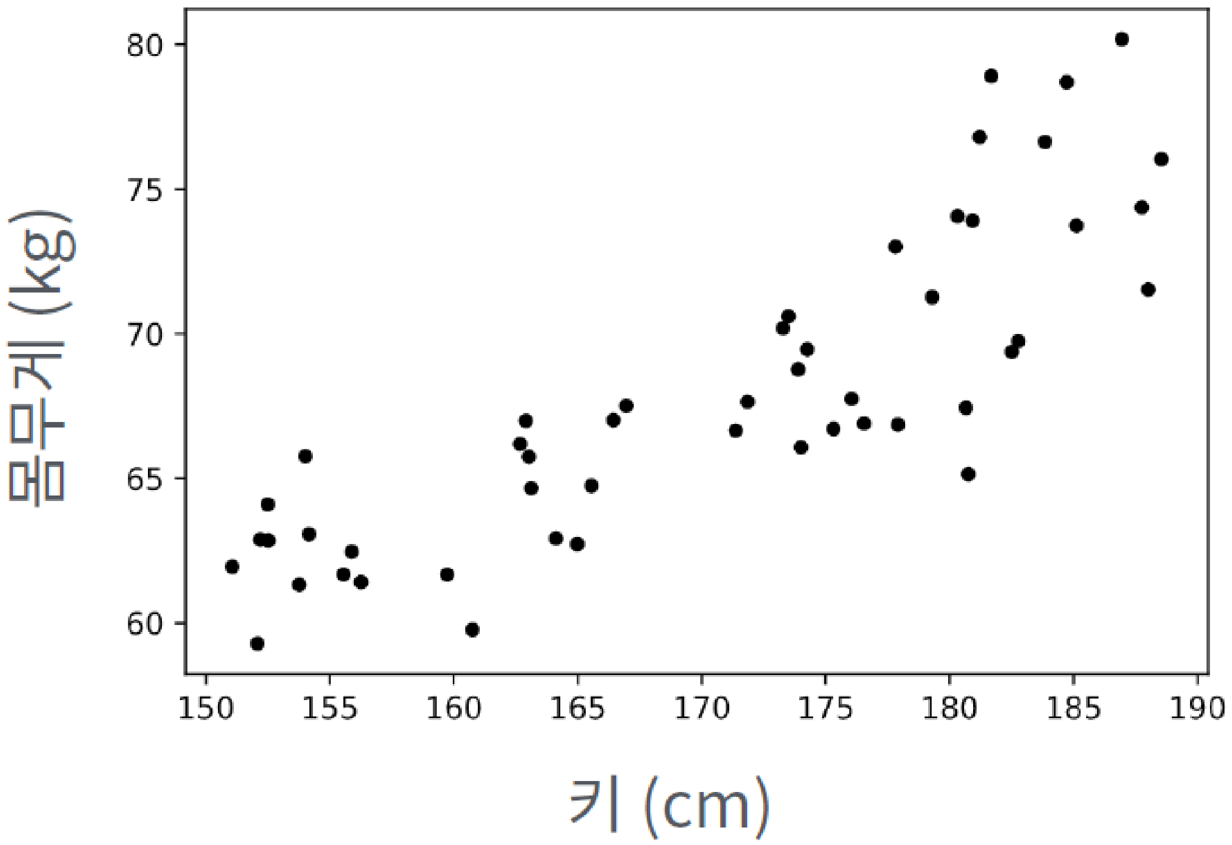
Regression

- Classification
  - Is it a dog or not? (binary)
  - What is this animal? (multi-class)
- Regression
  - What will be the stock value? $y \in \mathbb{R}$
  - What will be the housing price? $y \in [0, \infty)$



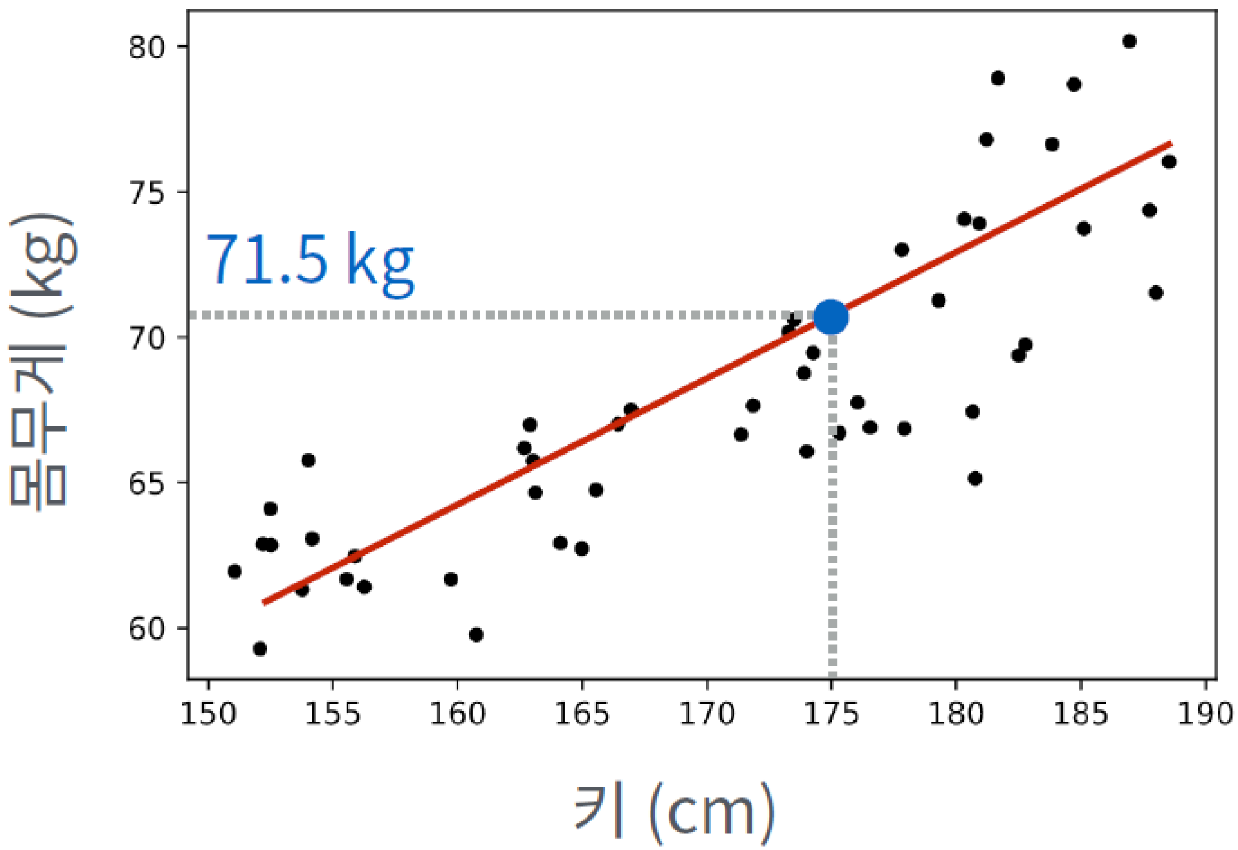
Linear Regression

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



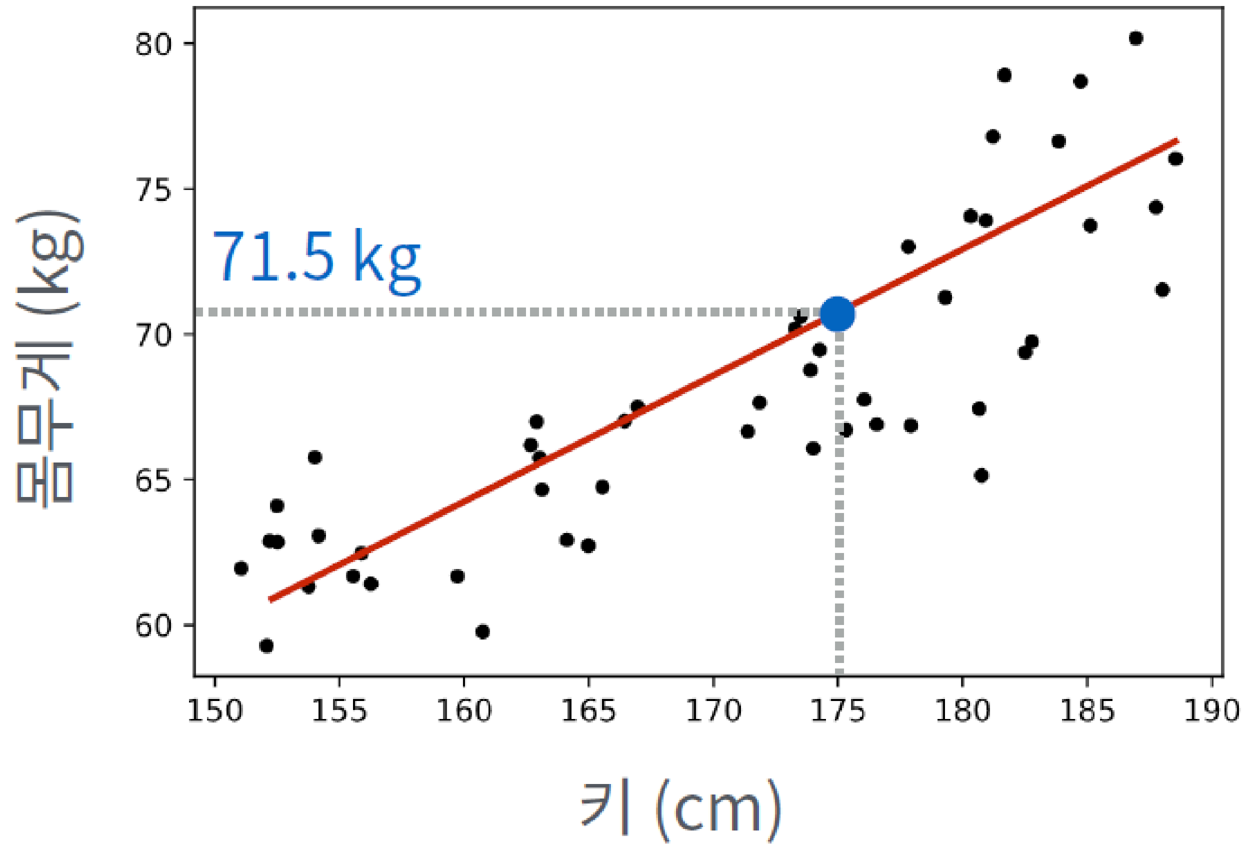
Linear Regression

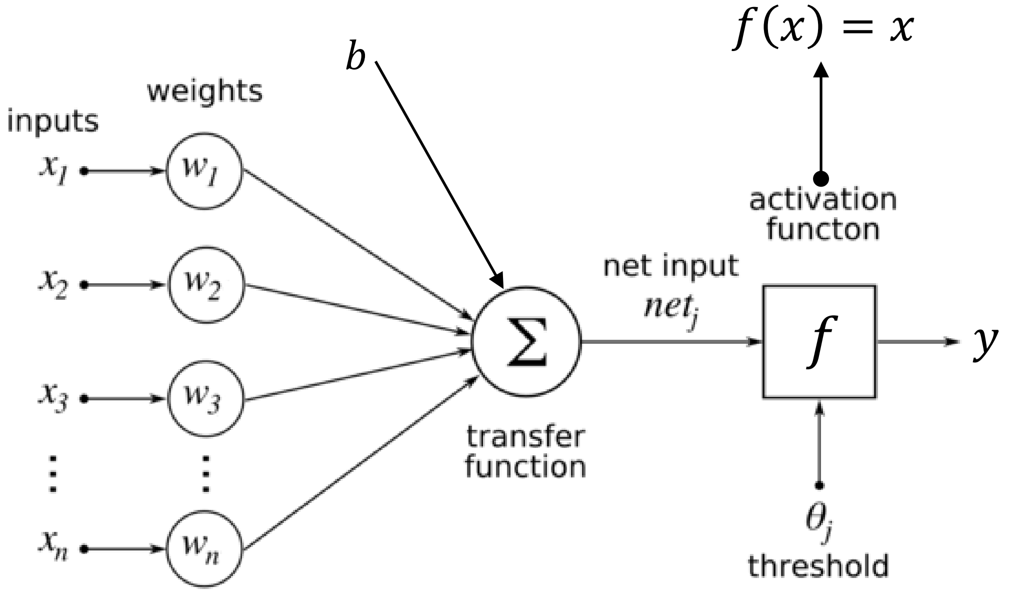
- 어느 학교 학생들의 신체검사 자료
- 새로 전학온 학생 A의 키가 175cm일 때 예상 몸무게는?



Linear Regression

- 선형함수(예 : 1차함수)로 주어진 data를 근사한다
- $y = wx + b$





<Perceptron>

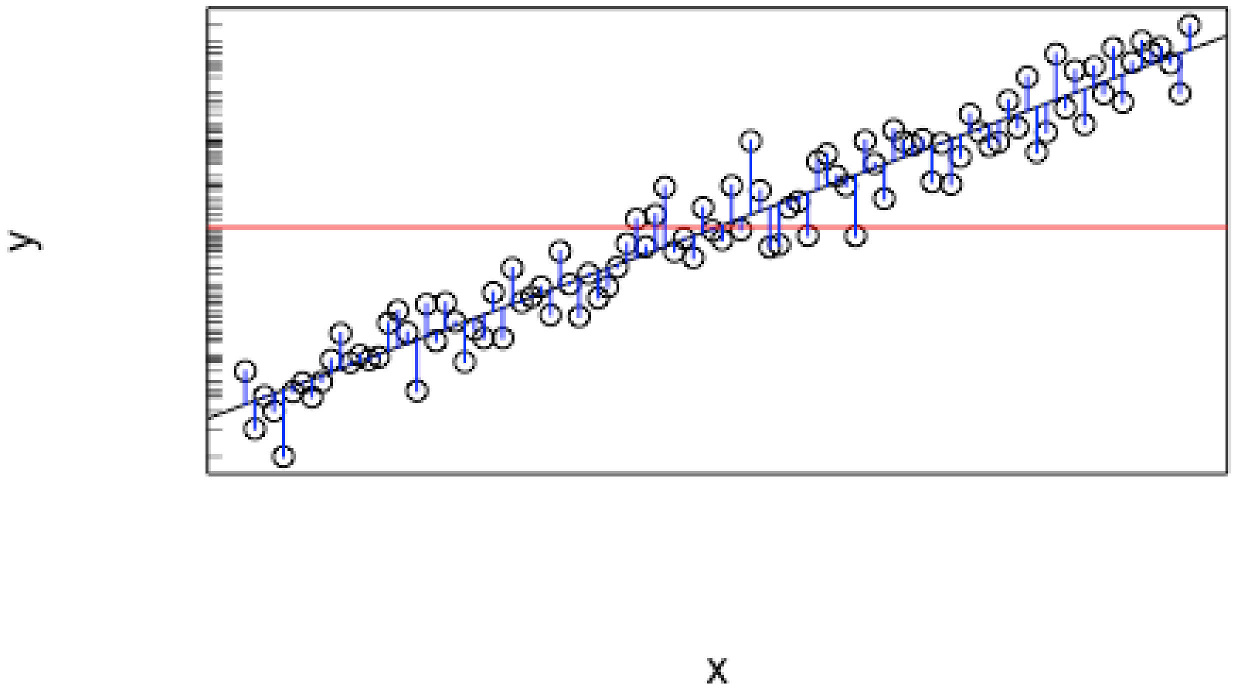
$$y = f(\mathbf{wx} + \mathrm{b})$$

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

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

Linear Regression

- 잘 예측했는지 측정할 척도(metric)가 필요함



$$y^* = wx + b \text{ (예측값)}$$

$$Cost(Loss) = \sum_i (y_i - y_i^*)^2$$

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

Linear Regression

- Cost(Loss) 값을 minimize하는 w와 b를 구하면 될텐데.... 어떻게?
  - Random Search – 가능????
  - Cost function을 미분해서 최솟값(미분=0이되는 점)을 찾자!

약간의 수학을(미분을...) 조금 해야겠습니다

b 구하기

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

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

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

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

w 구하기

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

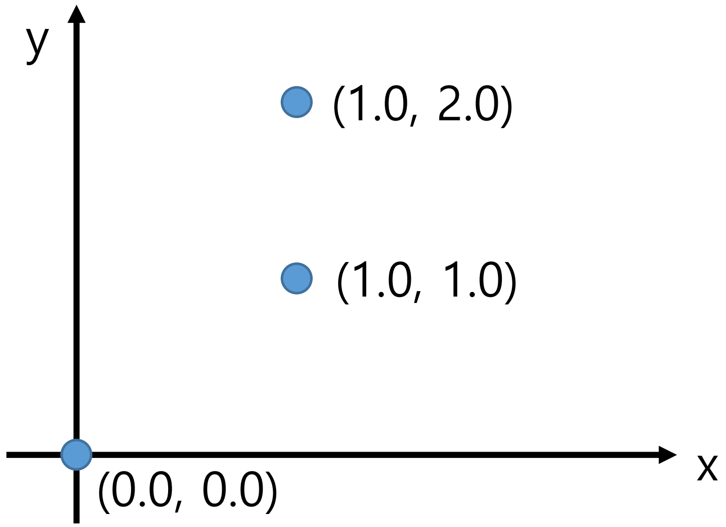
$$\frac{\delta L}{\delta w} = \frac{\delta \sum_i (y_i \ - wx_i \ - b)^2}{\delta w}$$

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

$$= 0$$

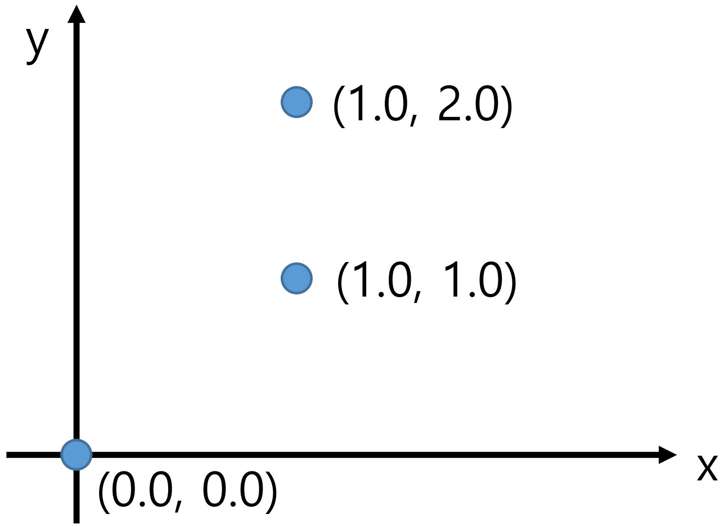
Example

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



Example

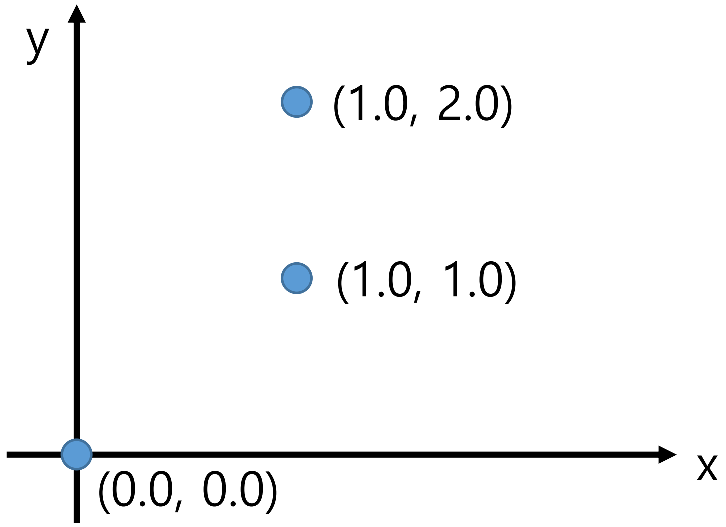
- $H(0, w_0, w_1) \approx 0.0$
- $H(1, w_0, w_1) \approx 1.0$
- $H(1, w_0, w_1) \approx 2.0$



Example

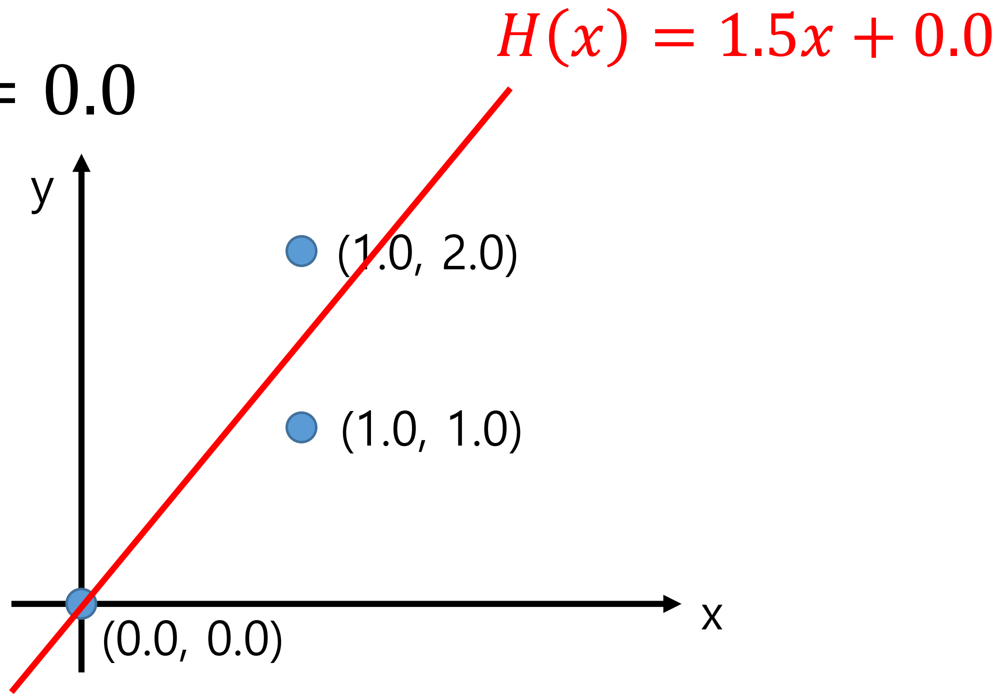
- $L = \sum_i (y_i - w_1 x_i - w_0)^2$

$$= (0.0 - w_1 \cdot 0.0 - w_0)^2 + (1.0 - w_1 \cdot 1.0 - w_0)^2 + (2.0 - w_1 \cdot 1.0 - w_0)^2$$
$$= 2{w_1}^2 + 3{w_0}^2 - 6w_1 - 6w_0 + 4w_1 w_0 + 5$$



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

- x가 scalar값(1개)가 아니라 vector가 된다면??

- Input
  - X1 : Facebook 광고료
  - X2 : TV 광고료
  - X3 : 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

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

Multi Variable Linear Regression

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

$$\nabla \mathrm{E}_{\mathrm{in}}(\mathbf{w}) = \mathbf{0}$$

- From previous equation

$$\nabla \mathrm{E}_{\mathrm{in}}(\mathbf{w}) = \frac{2}{N}(X^{\top}X\mathbf{w} - X^{\top}\mathbf{y})$$

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

Multi Variable Linear Regression

- Two scenarios
  - If $X^T X$ is invertible

    Moore-Penrose Pseudoinverse

    $$\boldsymbol{w} = \boxed{\qquad\qquad} y$$

  - If $X^T X$ is not invertible

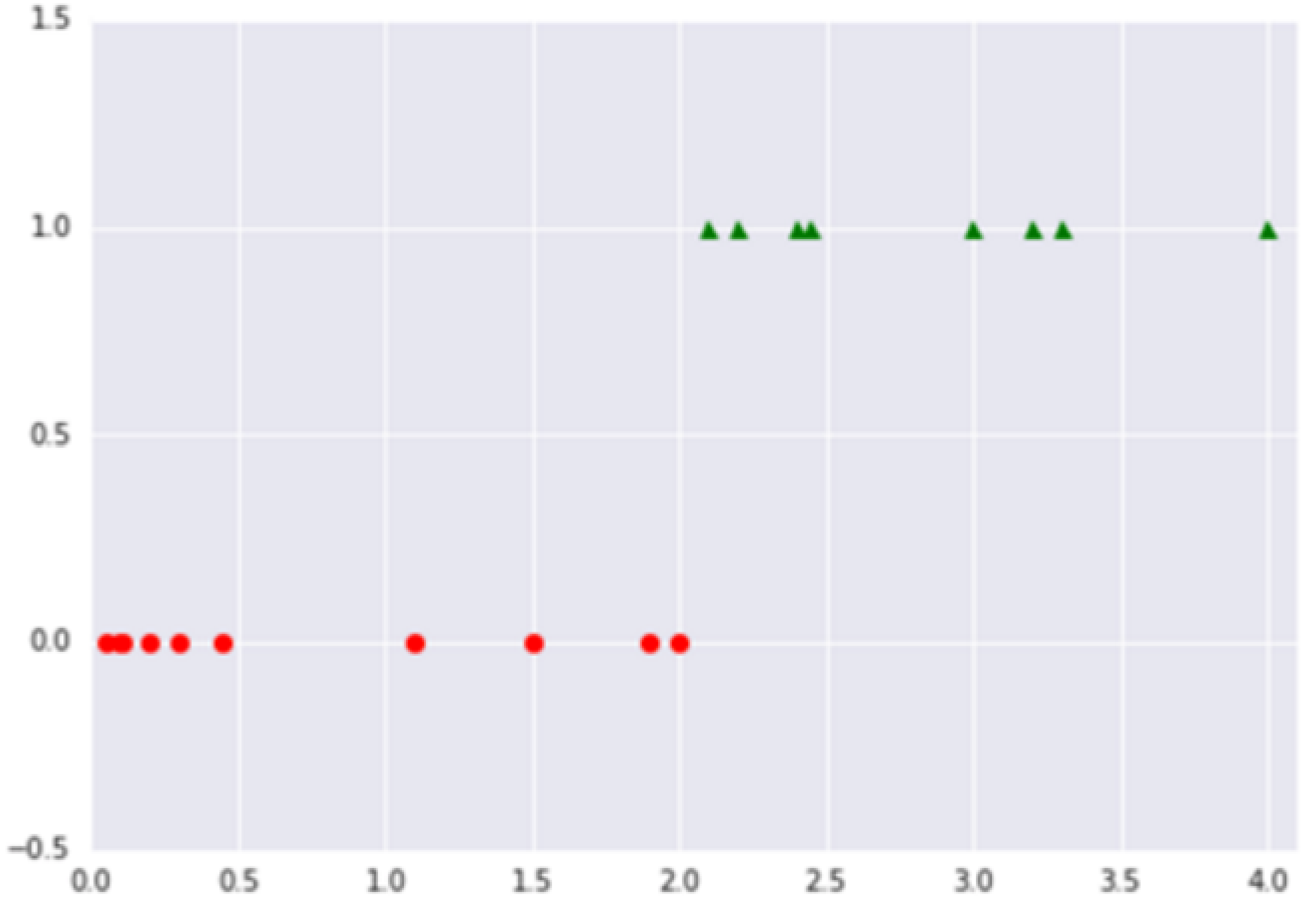
    Pseudo-inverse defined, but no unique solution

<Note>
$X \in \mathbb{R}^{n \times p}$ 일 때(n: sample 수, p: input vector size),
- p가 커질수록 matrix inversion 연산량이 많아짐
- p>n 이면 $X^T X$ is not invertible

Classification도 할 수 있지 않을까요?

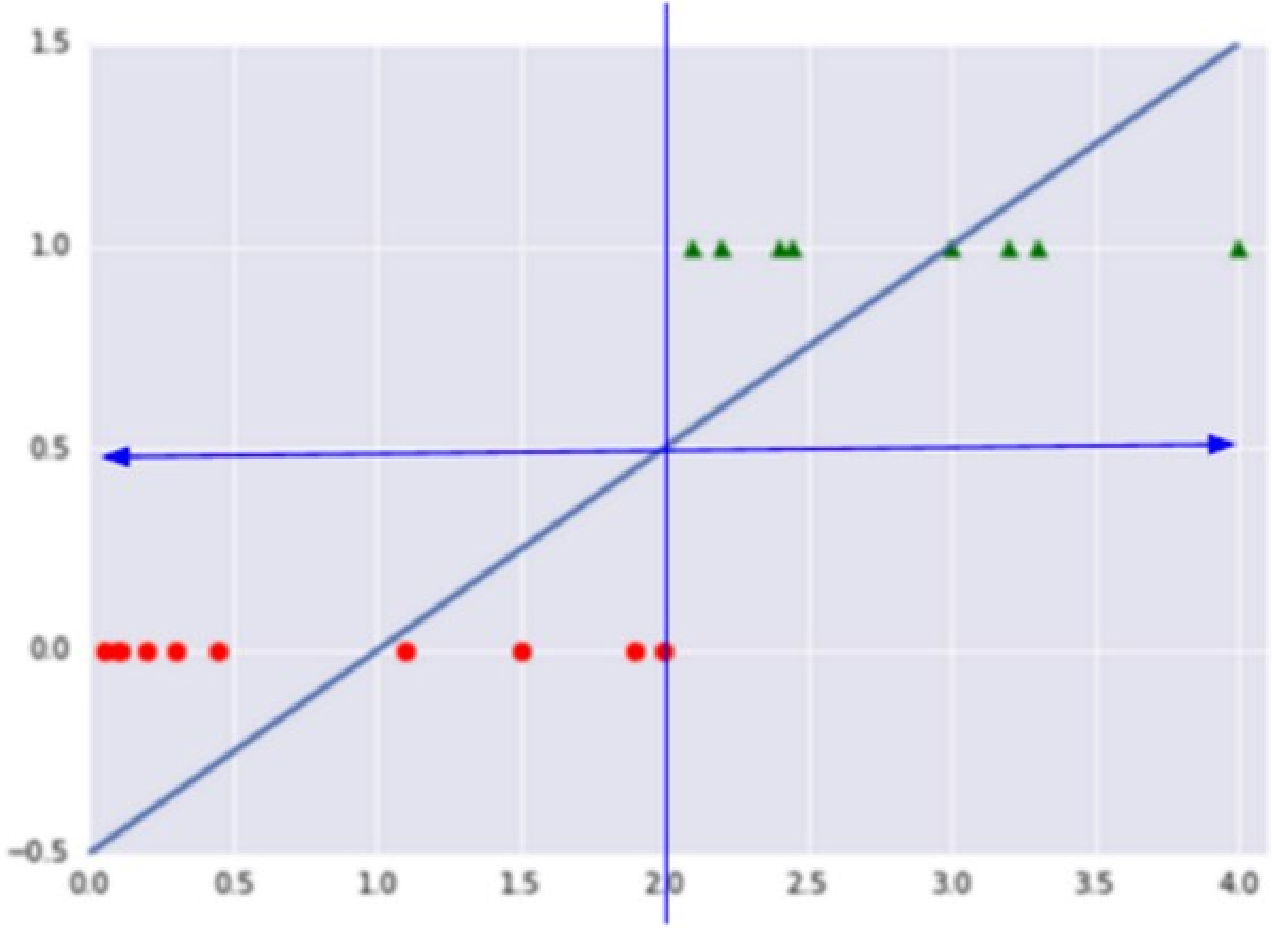
Binary Classification

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



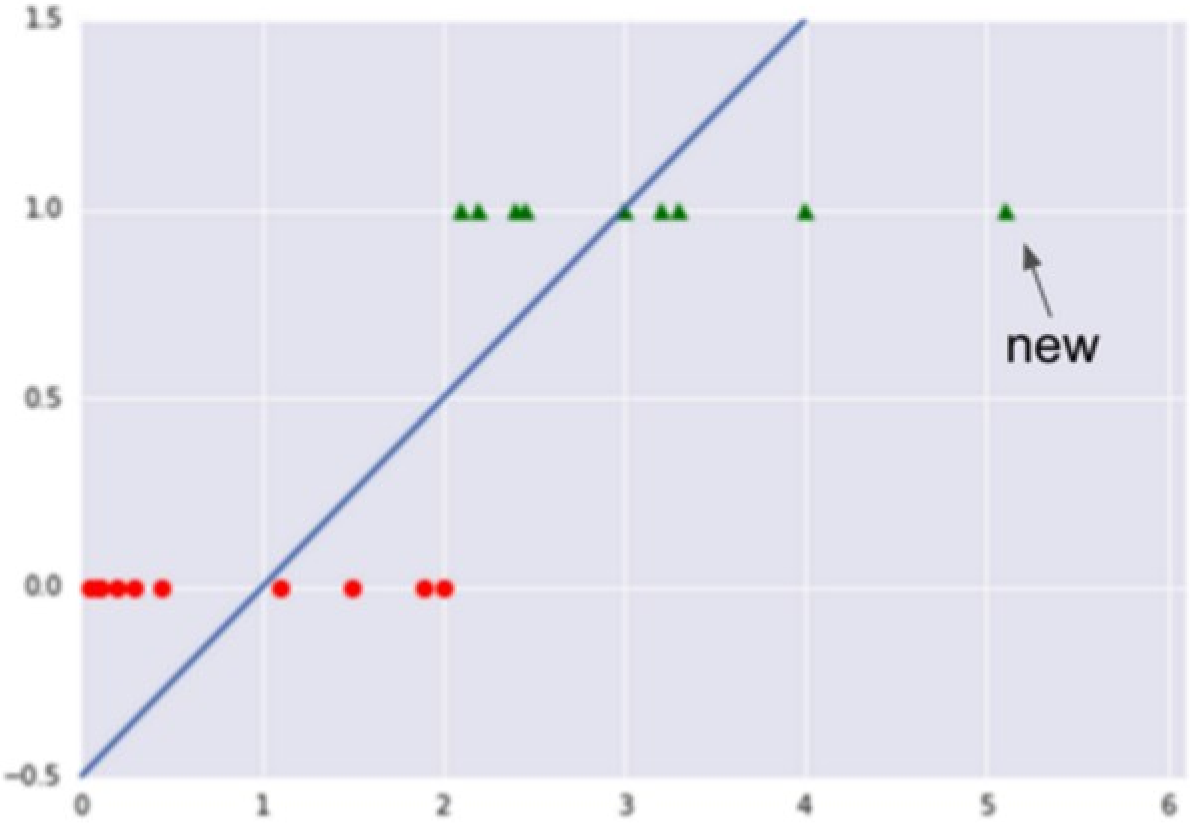
Binary Classification

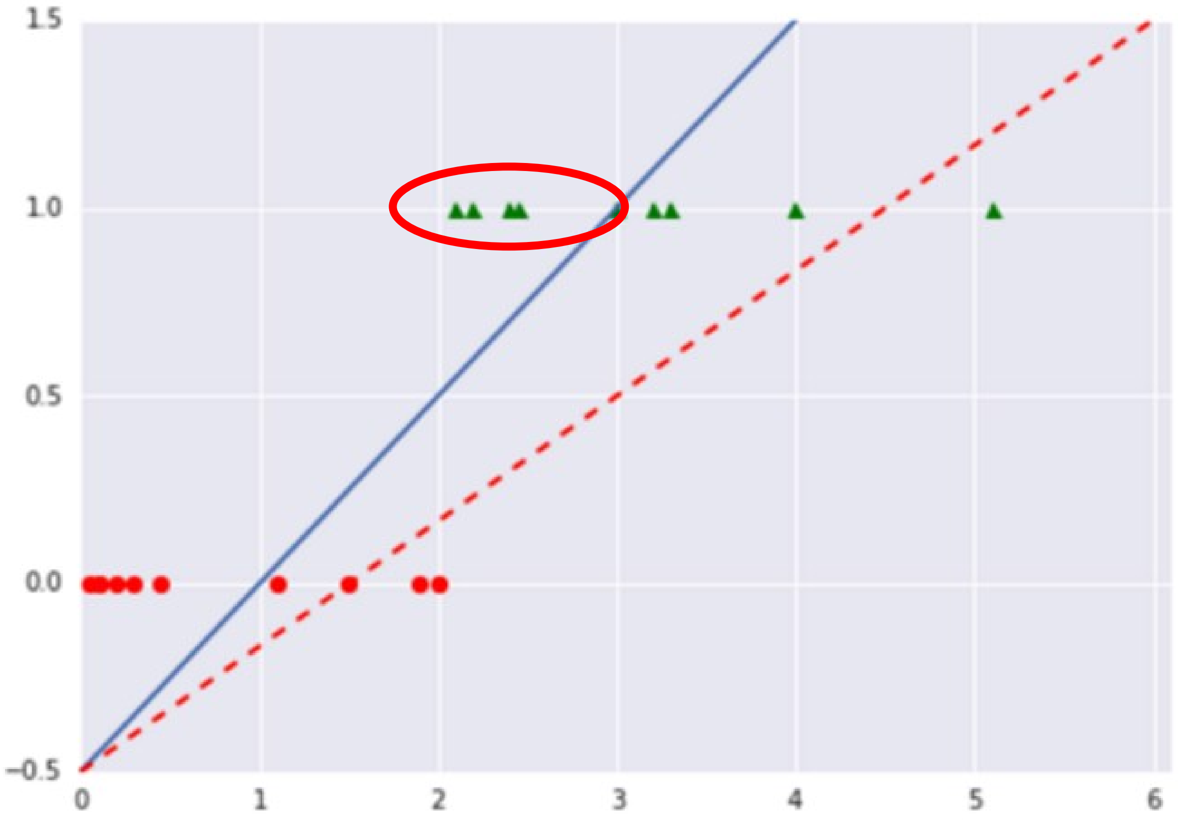
- Linear Regression으로 해봅시다
  - Regression 예측값이 0.5 이상이면 양성, 0.5 이하면 음성으로 판별



Binary Classification

- 종양의 크기가 매우 큰 data(outlier)가 추가된 경우

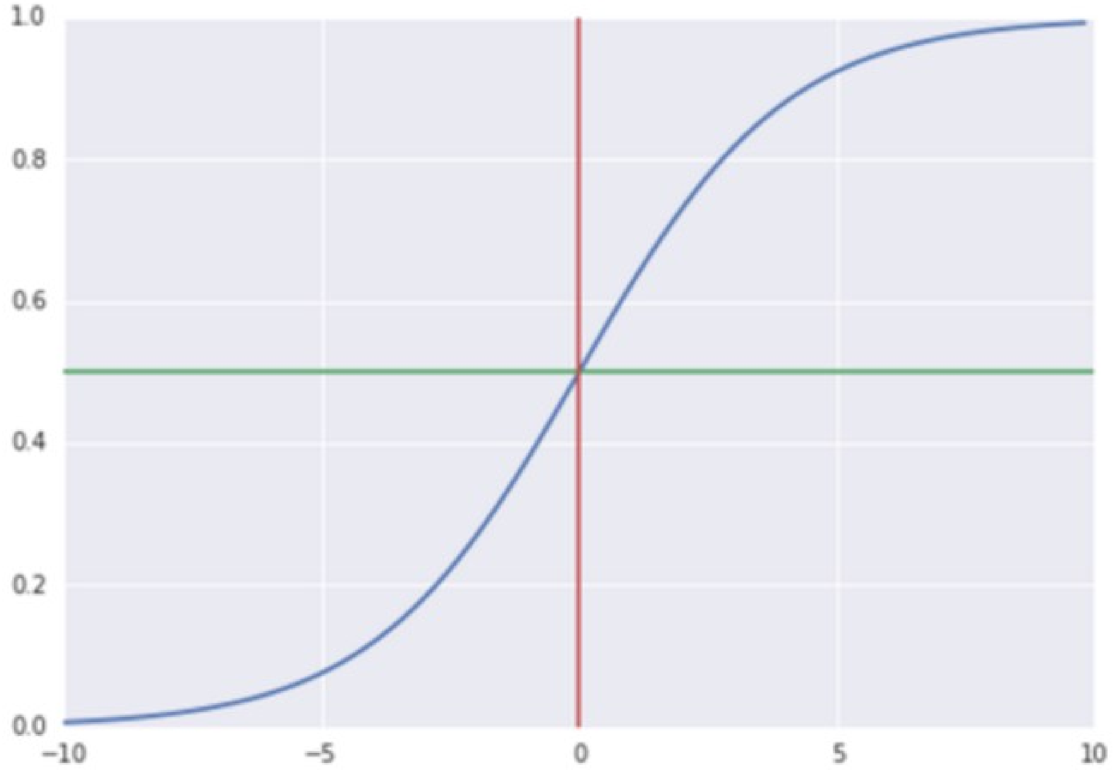




Binary Classification

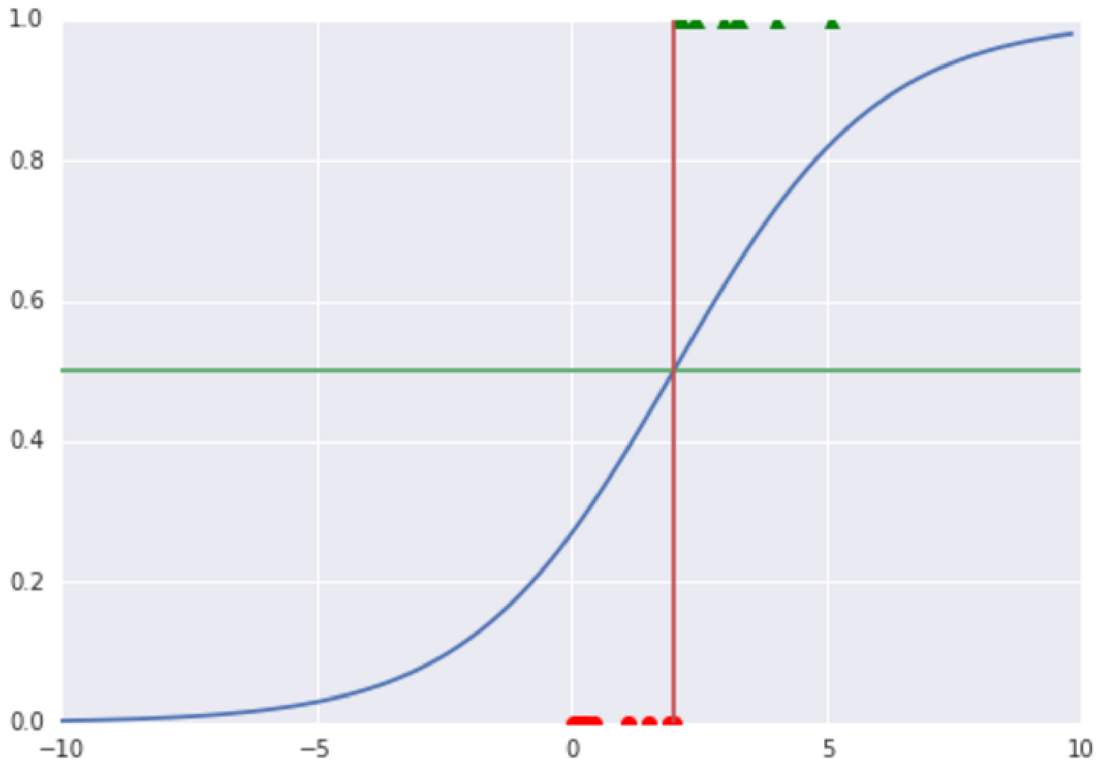
- 아주 크거나 아주 작은 data에 영향을 많이 받지 않았으면 좋겠다
- Binary classification에 맞게 0에서 1사이 값으로 나오면 좋겠다

→ Sigmoid 를 써보자



Logistic Regression

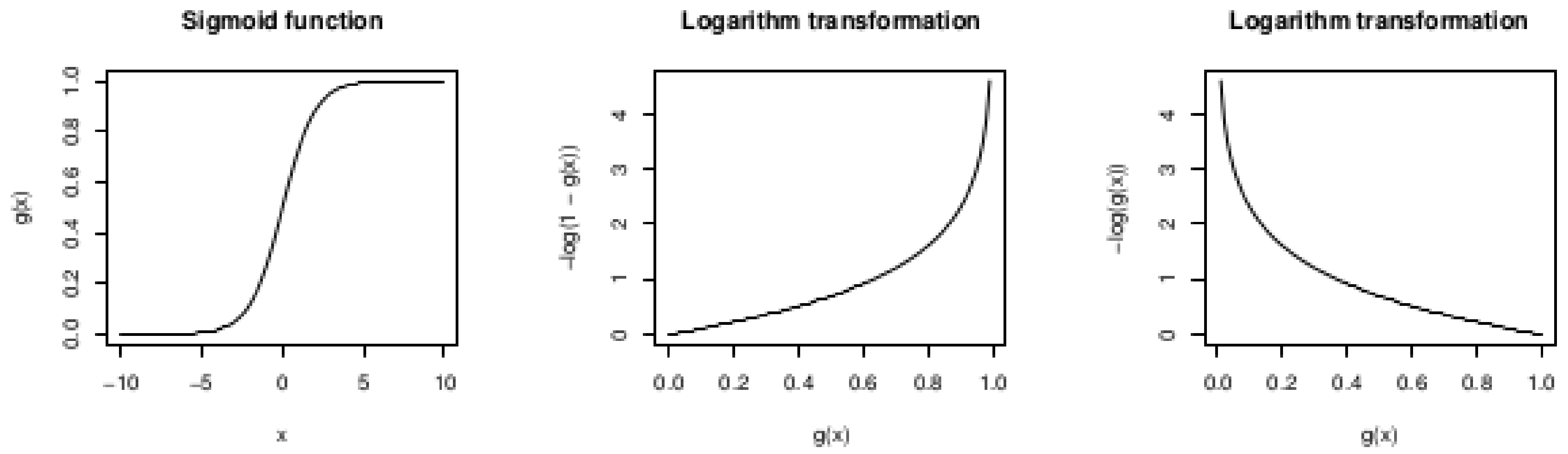
- Linear Regression 식에 Sigmoid 함수를 통과시킨 것
  - $H(x) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b) = \dfrac{1}{1+e^{-(\boldsymbol{w}^T\boldsymbol{x}+b)}} = P(y|\boldsymbol{x})$



Logistic Regression

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

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



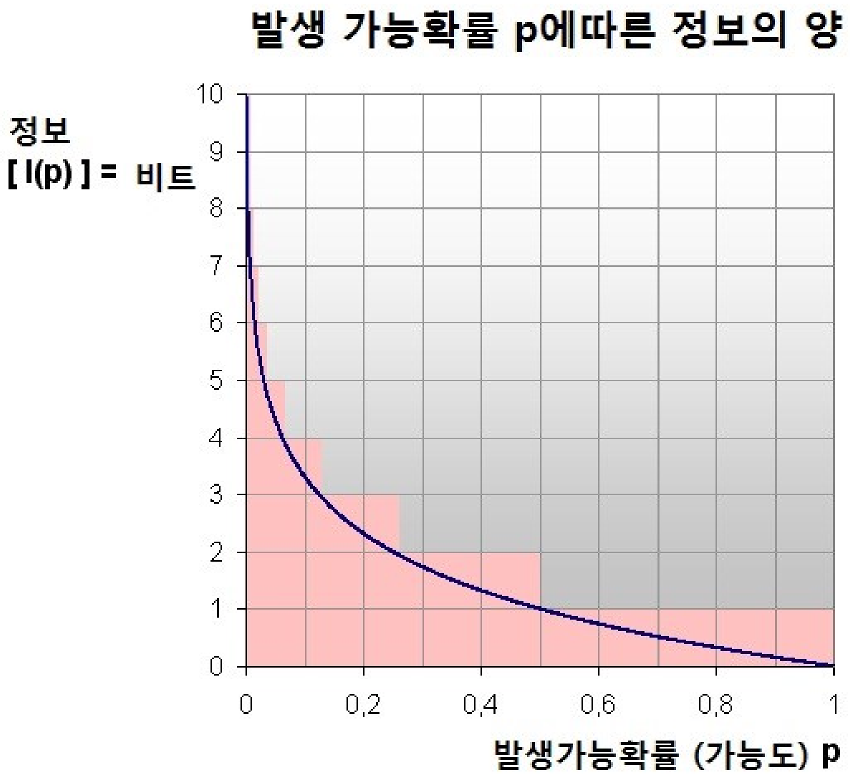
(a) *Sigmoid function.* (b) *Cost for $y = 0$.* (c) *Cost for $y = 1$.*

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

entropy

- Information Theory에서 Entropy란 정보량의 기댓값(평균)이다

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

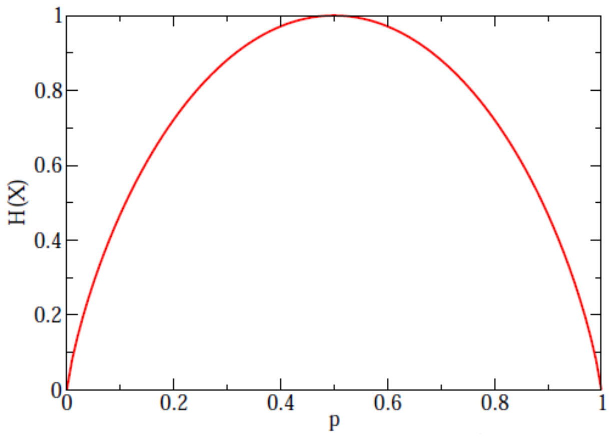


- 항상 일어나는 사건이라면 p=1 이고 이것이 가지고 있는 정보량은 0이다.
- 일어날 확률이 적을 수록 많은 정보를 담고 있다.

Entropy

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

$$= \sum_{i=1}^{n} \mathrm{P}(x_i)\,\mathrm{I}(x_i) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i),$$



Ex) 동전던지기

P(앞) = ½
P(뒤) = ½

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

이 확률 분포를 표현하기 위하여 평균 1 bit가 필요하다는 의미!

Cross Entropy

True distribution p,
Model이 예측한 distribution을 q라고 하면

Cross entropy를 minimize 하는 것은 p와 q의 분포가 가까워지도록 만드는 것과 같다

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

P 자체의 entropy(불변)

p를 기준으로 q가 얼마나 다른가
(p의 distribution과 q의 distribution의 거리)

Kullback–Leibler(KL) Divergence

p의 entropy　　p, q의 cross entropy

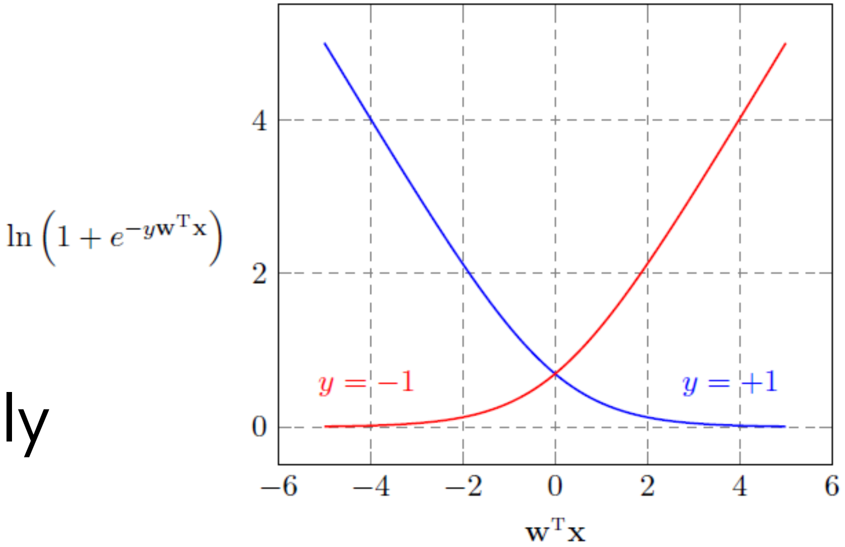
- 두 확률 분포간의 거리(?)를 나타냄
  - $D_{KL}(p \parallel q) = p \log \frac{p}{q} = p \log p + (-p \log q) = -H(p) + H(p, q) = H(p, q)$
  - p는 label, q는 network output 이라고 할 경우, network의 목표는 label이 나타내는 확률 분포 p와 최대한 가까운 q를 학습하고자 하는 것임
  - 일반적으로 label은 one-hot encoding 하기 때문에, 각 label의 값이 확률이라고 생각하면 p의 entropy,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

Minimizing NLL

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

- We can define loss(error) function as below

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



- Unfortunately, not easy to manipulate analytically

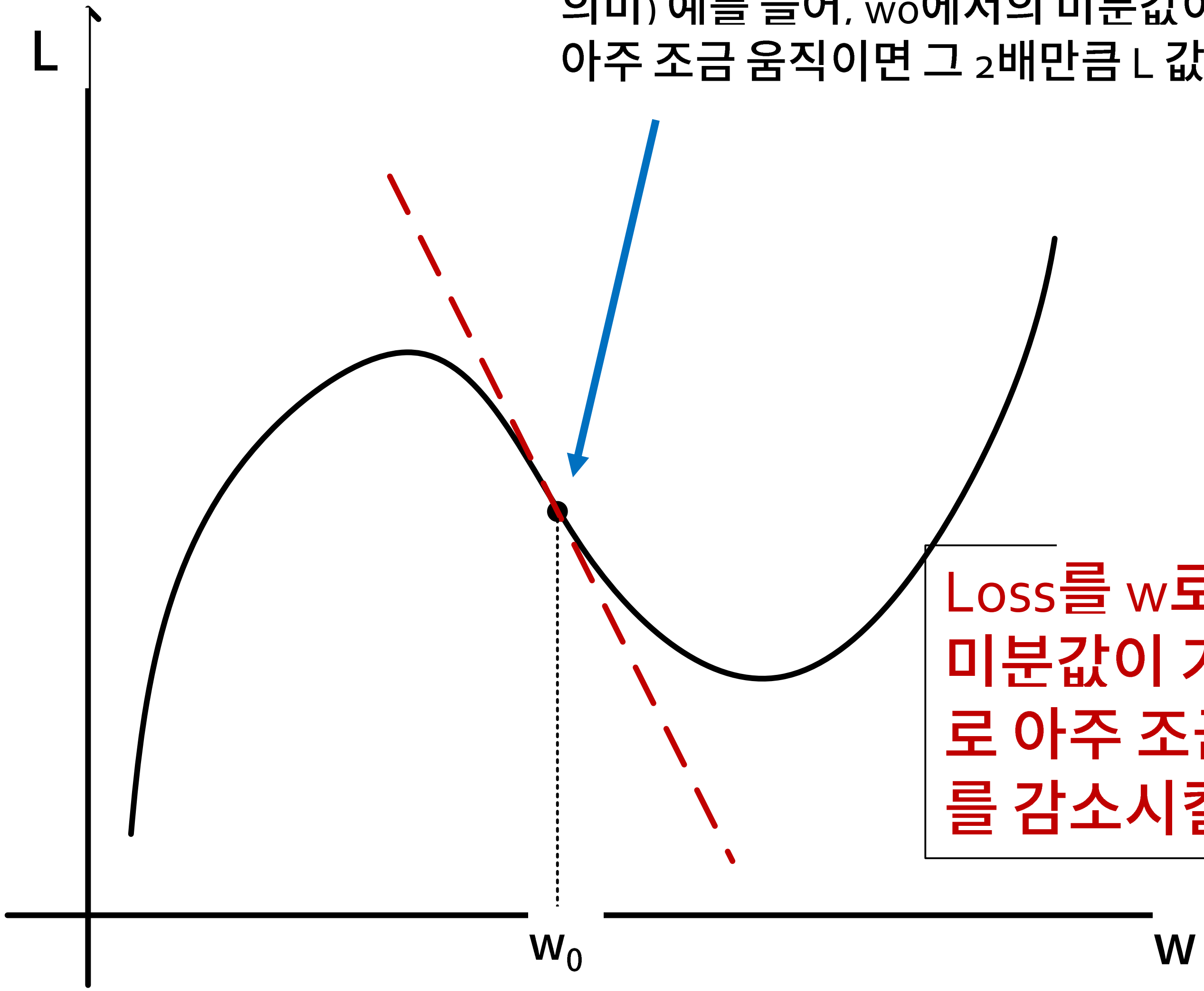
$$\nabla \mathrm{E}_{\mathrm{in}}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N} \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^\mathrm{T} \mathbf{x}_n}}$$

$$= \frac{1}{N}\sum_{n=1}^{N} -y_n \mathbf{x}_n \theta(-y_n \mathbf{w}^\mathrm{T} \mathbf{x}_n)$$

- We need iterative optimization
- Use 미분!

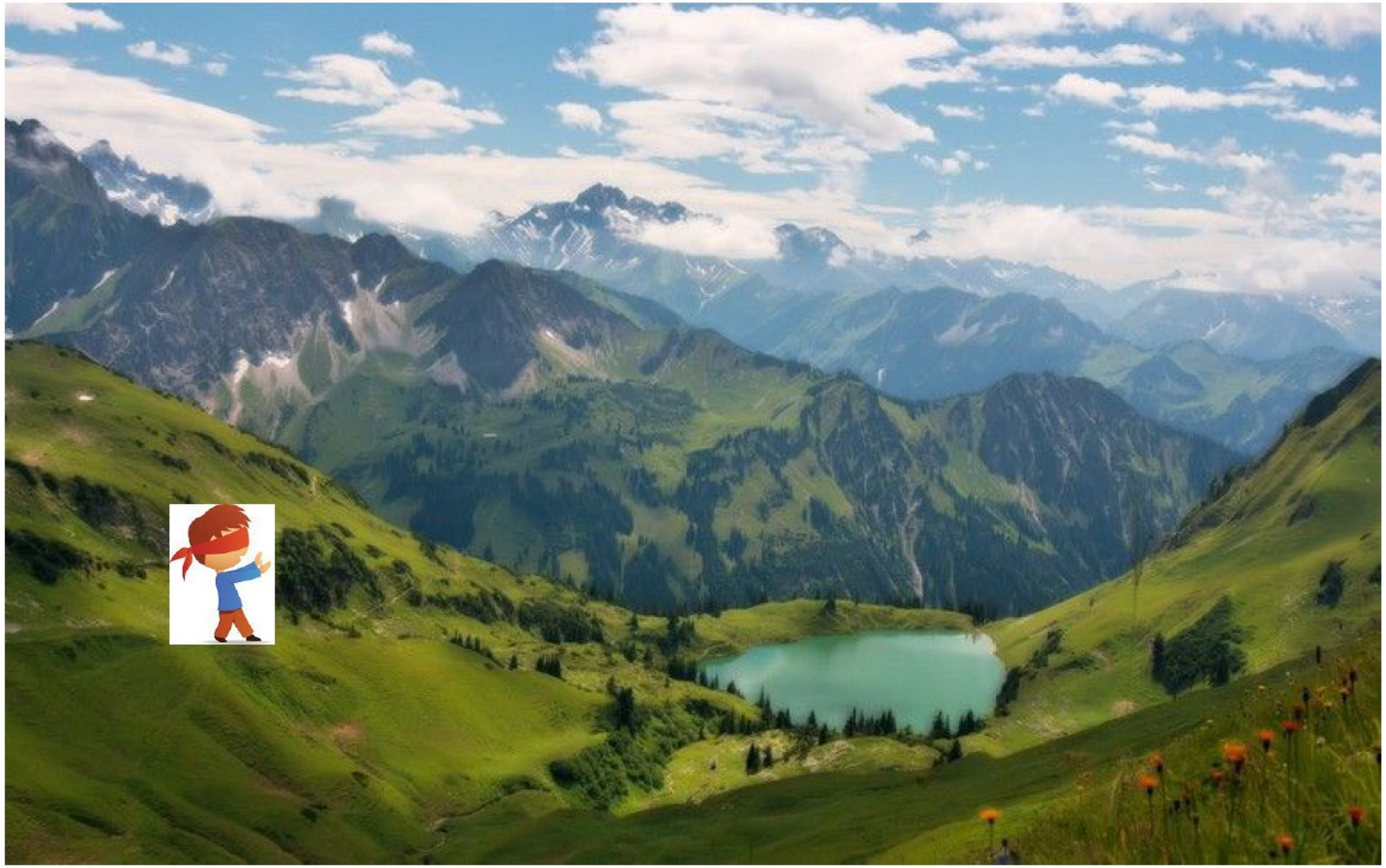
미분??

$w=w_0$에서의 미분값
= 이 점에서의 접선의 기울기
의미) 예를 들어, w_0에서의 미분값이 -2라면, w를 w_0에서 왼쪽(-방향)으로 아주 조금 움직이면 그 2배만큼 L 값이 증가한다는 뜻!



Loss를 w로 미분하고,
미분값이 가리키는 방향의 반대방향으로 아주 조금씩 w를 바꿔나가면 Loss를 감소시킬 수 있다!!

Gradient Descent



Gradient Descent

Loss Function의 미분(Gradient)를 이용하여 weight를 update하는 방법

$$w_{new} = w_{old} - \eta \nabla_w L$$

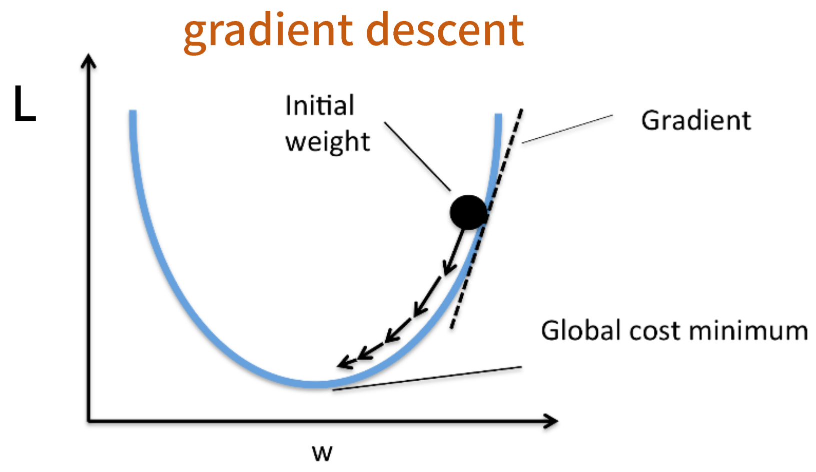
Weight update $= w_{new} - w_{old}$

$$= -\eta \nabla_w L$$

Loss를 감소 시키는 방향 (Descent)

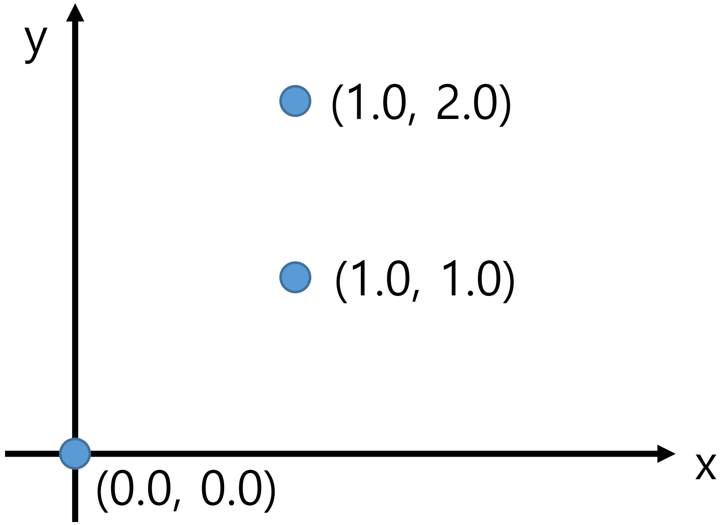
아주 조금씩 이동 (Learning Rate)

미분값 (Gradient)



Linear Regression Again

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



$$L = \sum_i (y_i - w_1 x_i - w_0)^2$$

$$= (0.0 - w_1 \cdot 0.0 - w_0)^2 + (1.0 - w_1 \cdot 1.0 - w_0)^2 + (2.0 - w_1 \cdot 1.0 - w_0)^2$$

$$= 2{w_1}^2 + 3{w_0}^2 - 6w_1 - 6w_0 + 4w_1 w_0 + 5$$

Solving Linear Regression Using GD

- Choose a small value for η such as $\eta = 0.1$
- Randomly select $w_0^0 = 1, w_1^0 = 1$, initially
- Repeat

$$w_0^{t+1} = w_0^t - \eta(4w_1^t + 6w_0^t - 6)$$
$$w_1^{t+1} = w_1^t - \eta(4w_1^t + 4w_0^t - 6)$$

$$\frac{\partial L}{\partial w_1} = 4w_1 + 4w_0 - 6 = 0$$
$$\frac{\partial L}{\partial w_0} = 4w_1 + 6w_0 - 6 = 0$$

Solving Linear Regression Using GD

$$w_0^0 = 1$$
$$w_1^0 = 1$$

$$w_0^1 = 1 - 0.1(4 \times 1 + 6 \times 1 - 6) = 0.6$$
$$w_1^1 = 1 - 0.1(4 \times 1 + 4 \times 1 - 6) = 0.8$$

$$w_0^2 = 0.6 - 0.1(4 \times 0.8 + 6 \times 0.6 - 6) = 0.54$$
$$w_1^2 = 0.8 - 0.1(4 \times 0.8 + 4 \times 0.6 - 6) = 0.84$$

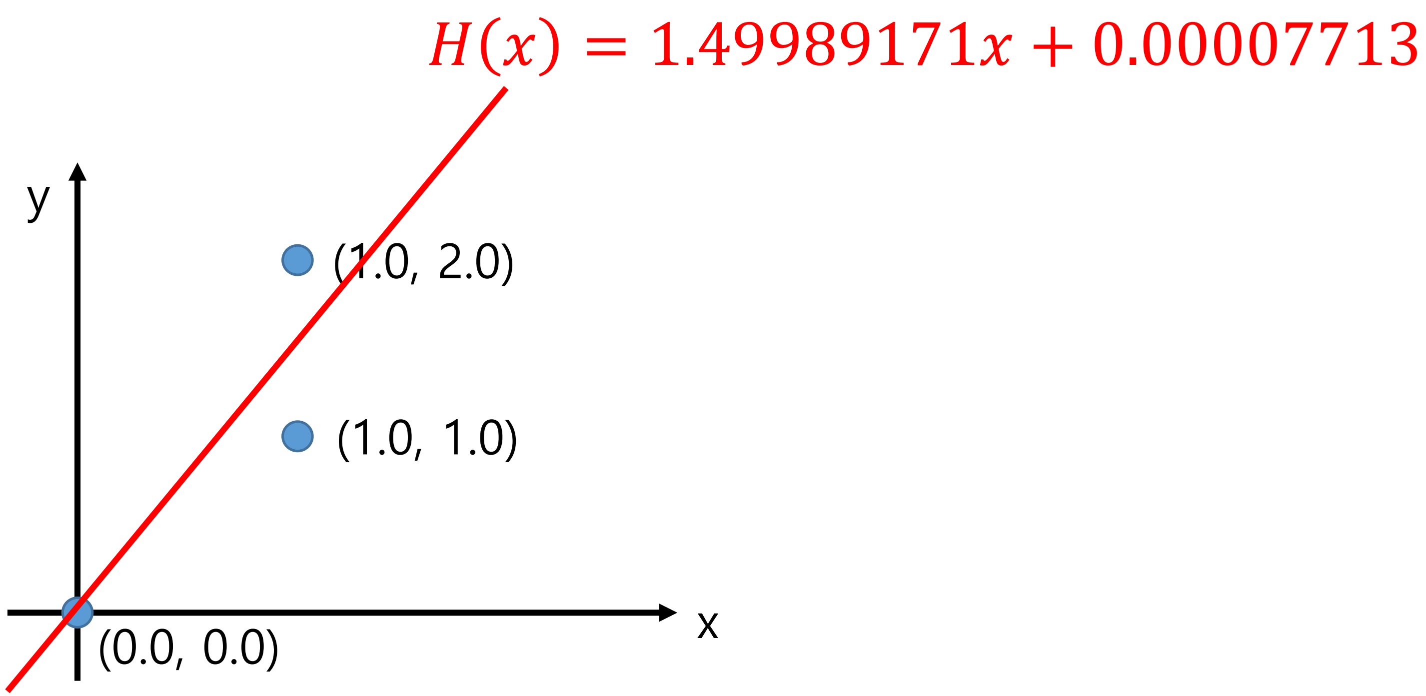
$$w_0^3 = 0.54 - 0.1(4 \times 0.84 + 6 \times 0.54 - 6) = 0.480$$
$$w_1^3 = 0.84 - 0.1(4 \times 0.84 + 4 \times 0.54 - 6) = 0.888$$

Solving Linear Regression Using GD

$$w_0^4 = 0.480 - 0.1(4 \times 0.888 + 6 \times 0.480 - 6) = 0.4368$$
$$w_1^4 = 0.888 - 0.1(4 \times 0.888 + 4 \times 0.480 - 6) = 0.9408$$

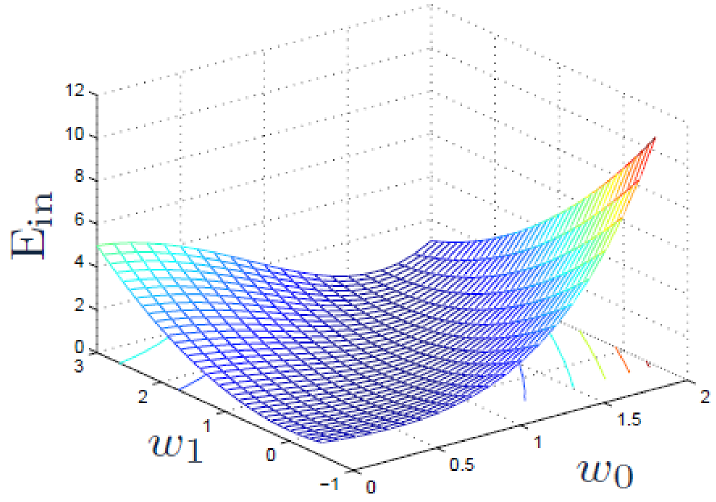
…

$$w_0^{100} = 0.00007713$$
$$w_1^{100} = 1.49989171$$

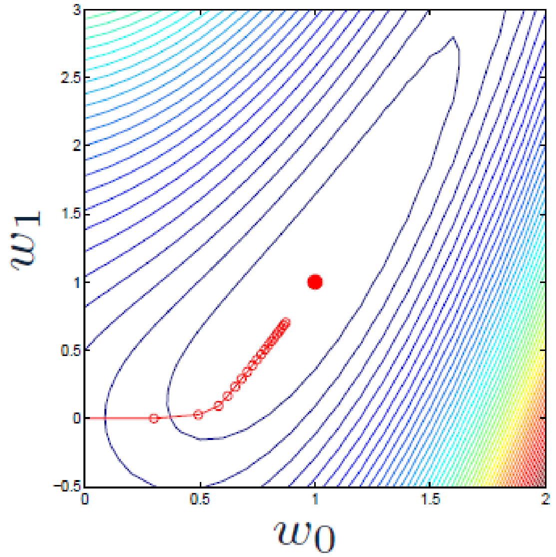


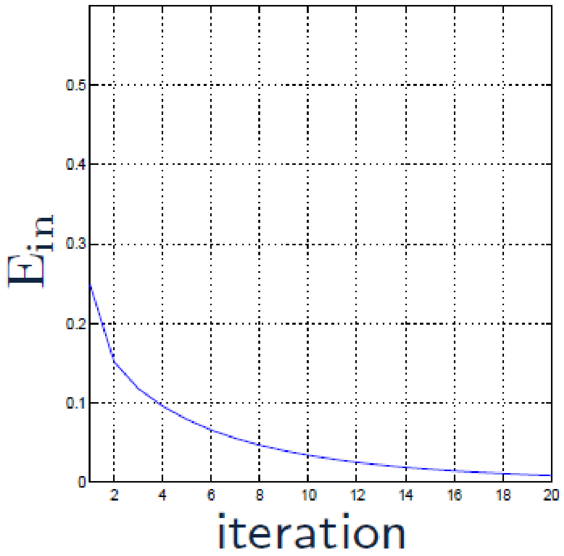
Gradient Decent Example

- **Global minimum 0 at (1,1)**

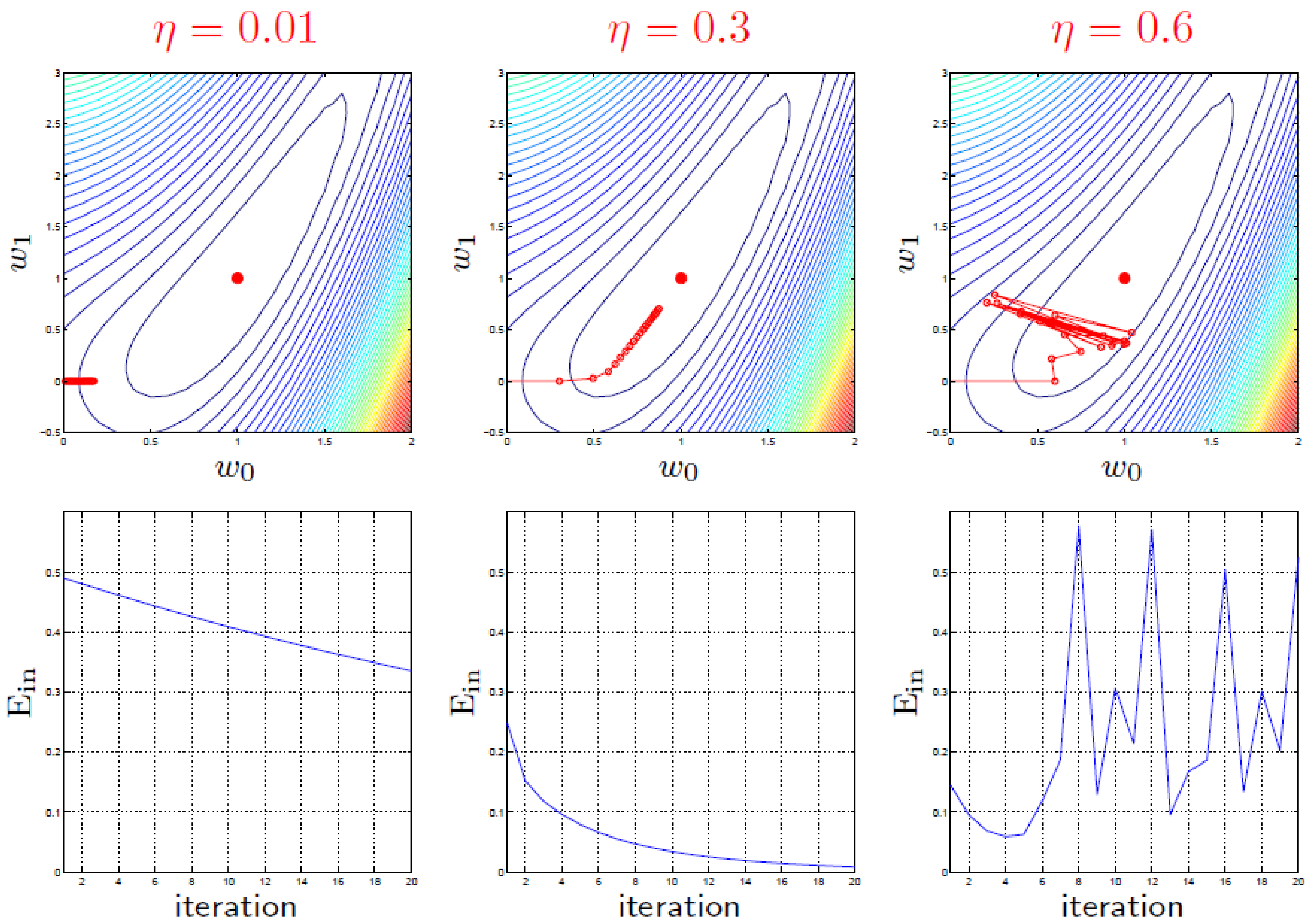


- **Start at (0,0)**
  - # iterations (steps) $= 20$
  - step size $\eta = 0.3$



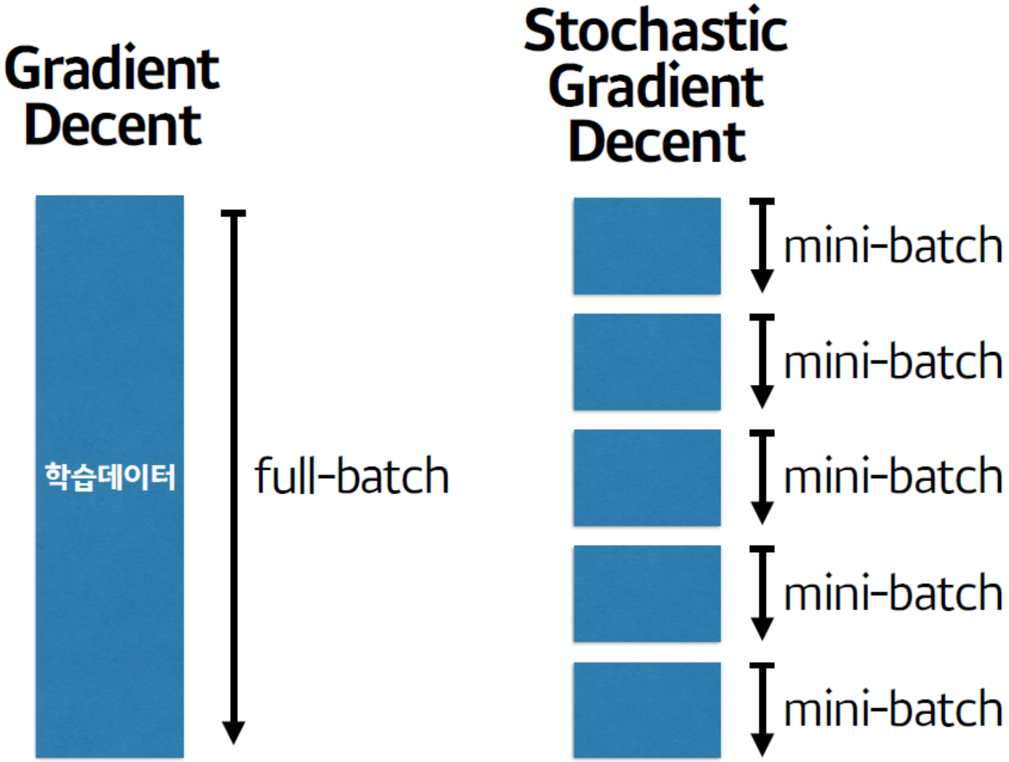


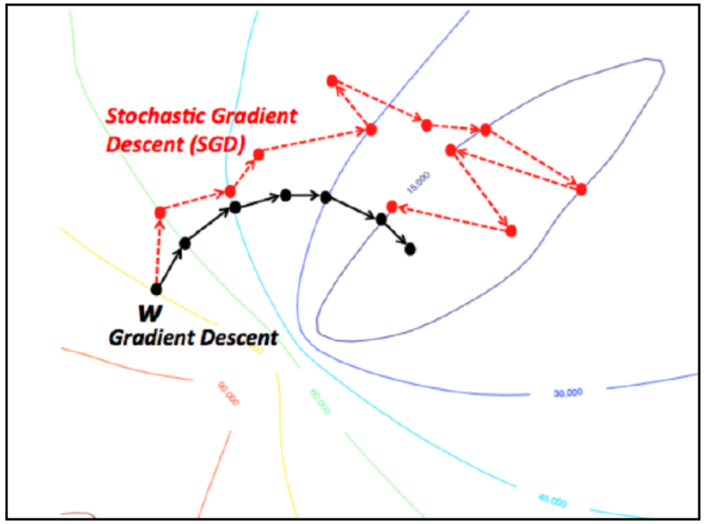
Learning Rate



Stochastic Gradient Descent, Mini-batch Training

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

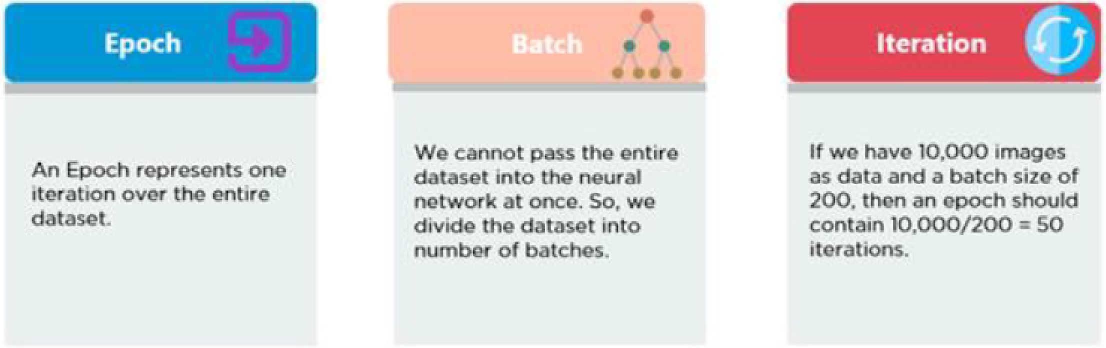




전부다 읽고나서
최적의 1스텝 간다.

작은 토막마다
일단 1스텝간다.

Learning Rate & Mini-Batch



Mini-Batch size: Number of training instances the network evaluates per weight update step.

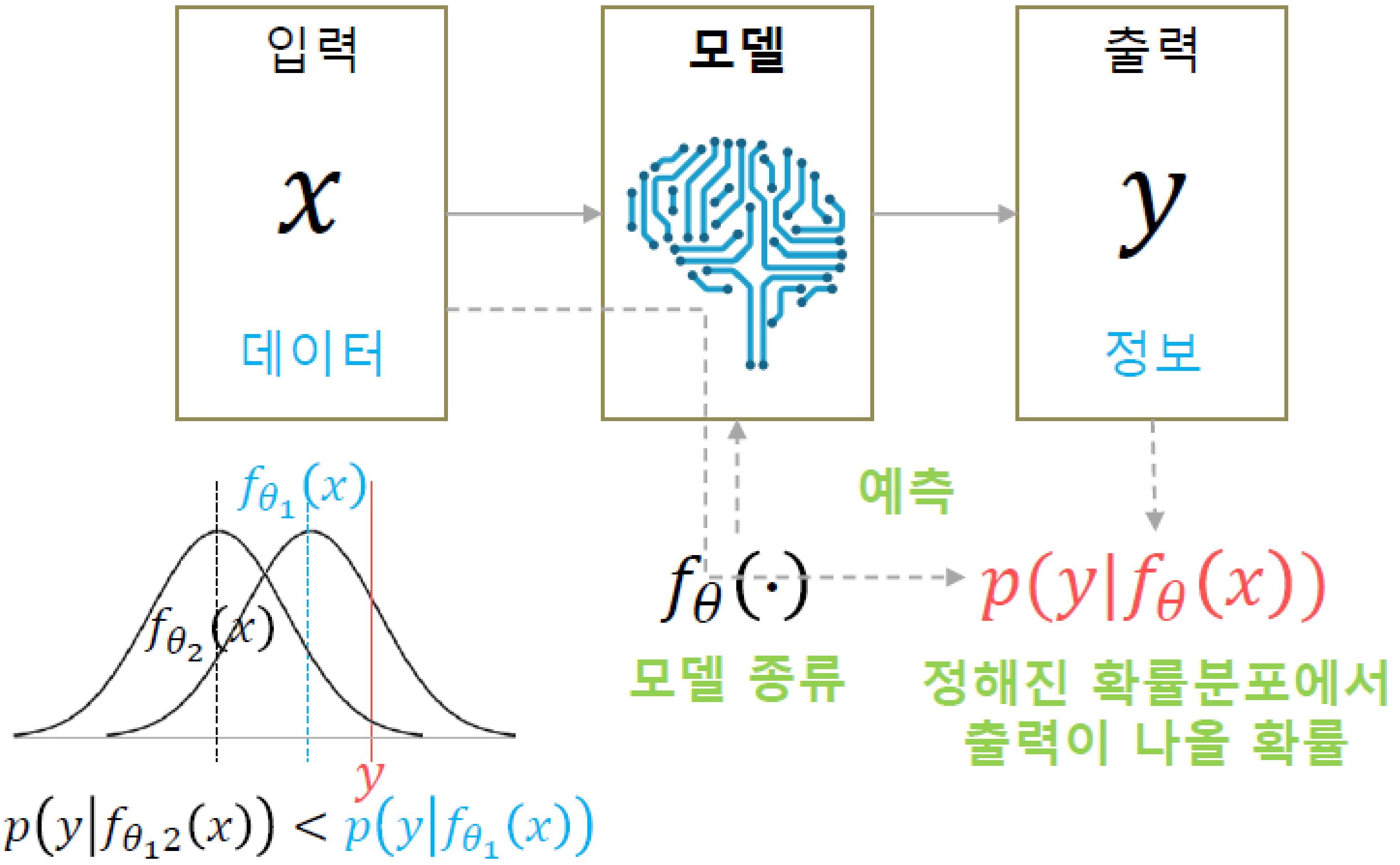
- Larger batch size = more computational speed
- Smaller batch size = (empirically) better generalization

“Training with large minibatches is bad for your health. More importantly, it's bad for your test error. Friends don’t let friends use minibatches larger than 32.”
- Yann LeCun

[Revisiting Small Batch Training for Deep Neural Networks](https://arxiv.org/abs/1804.07612) (2018)

“when increasing the batch size, a linear increase of the learning rate η with the batch size m is required to keep the mean SGD weight update per training example constant”

Probabilistic View



$$\theta^* = \underset{\theta}{\operatorname{argmin}}[-\log(p(y|f_{\theta}(x)))]$$

주어진 데이터를 제일 잘
설명하는 모델 찾기

Probabilistic View
– Maximum Likelihood Estimation

i.i.d Condition on $p(y|f_\theta(x))$

Assumption 1 : Independence

All of our data is independent of each other

$$p(y|f_\theta(x)) = \prod_i p_{D_i}(y|f_\theta(x_i))$$

Assumption 2: Identical Distribution

Our data is identically distributed

$$p(y|f_\theta(x)) = \prod_i p(y|f_\theta(x_i))$$

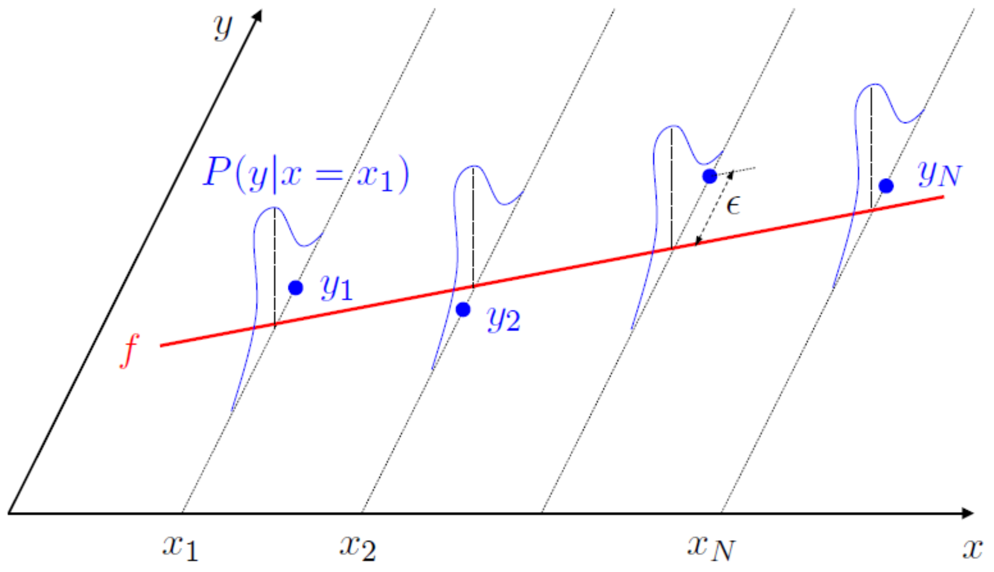
- Negative Log-Likelihood(NLL)

$$-\log(p(y|f_\theta(x))) = -\sum_i \log(p(y_i|f_\theta(x_i)))$$

Probabilistic View

- Linear Regression
  - Training data is considered as the set of iid samples from underlying probabilistic model:

    $y_i = \theta^T x_i + \epsilon_i$, where ϵ_i follows Gaussian distribution $\epsilon_i \sim \mathrm{N}(0, \sigma^2)$



  - Models predict the mean of Gaussian distribution(μ_i)

Probabilistic View

- Logistic Regression
  - Training data is considered as the set of iid samples which follows Bernoulli distribution:

    $$p(x) = p^x(1-p)^{(1-x)}$$

  - Models predict the probability $p(y = 1|x, \theta)$ directly

Probabilistic View
– Minimize Negative Log-Likelihod

Univariate cases

$$-\log\big(p(y_i|f_\theta(x_i))\big)$$

Gaussian distribution

$$f_\theta(x_i) = \mu_i, \sigma_i = 1$$

$$p(y_i|\mu_i,\sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{(y_i-\mu_i)^2}{2\sigma_i^2}\right)$$

$$\log(p(y_i|\mu_i,\sigma_i)) = \log\frac{1}{\sqrt{2\pi}\sigma_i} - \frac{(y_i-\mu_i)^2}{2\sigma_i^2}$$

$$-\log(p(y_i|\mu_i)) = -\log\frac{1}{\sqrt{2\pi}} + \frac{(y_i-\mu_i)^2}{2}$$

$$-\log(p(y_i|\mu_i)) \propto \frac{(y_i-\mu_i)^2}{2} = \frac{\big(y_i - f_\theta(x_i)\big)^2}{2}$$

Mean Squared Error

Bernoulli distribution

$$f_\theta(x_i) = p_i$$

$$p(y_i|p_i) = p_i^{y_i}(1-p_i)^{1-y_i}$$

$$\log(p(y_i|p_i)) = y_i\log p_i + (1-y_i)\log(1-p_i)$$

$$-\log(p(y_i|p_i)) = -[y_i\log p_i + (1-y_i)\log(1-p_i)]$$

Cross-entropy