목표

- 딥러닝을 위한 기초 수학을 이해하고, 신경망을 보면서 계산, 학습 과정을 이해 하면 성공

중요 단어

- 미분
- 역전파(Back Propagation)
- 경사하강법(Gradient Descent)
- 활성화 함수(Activation Function)

기초수학

$$y = ax + b$$

Y 에 관한 x의 1차 식

변하는 x에 대해서 y가 어떻게 변하는지 알기를 원한다.

➔ 변수 x에 대한 y의 관계를 알고 싶다 !!

X : 독립변수(Independent variable) → 독립적인 변수 / 다른 변수에 대해 영향을 받지 않는다

Y : 종속변수(Dependent variable) → 독립변수에 영향을 받아 변하는 수 / 의존적인 변수

e

$$e = \lim_{n \to \infty} (1 + \frac{1}{n})^n = 2.718281 \ldots$$

자연상수, 오일러 수, 네이피어 상수

특이한 성질

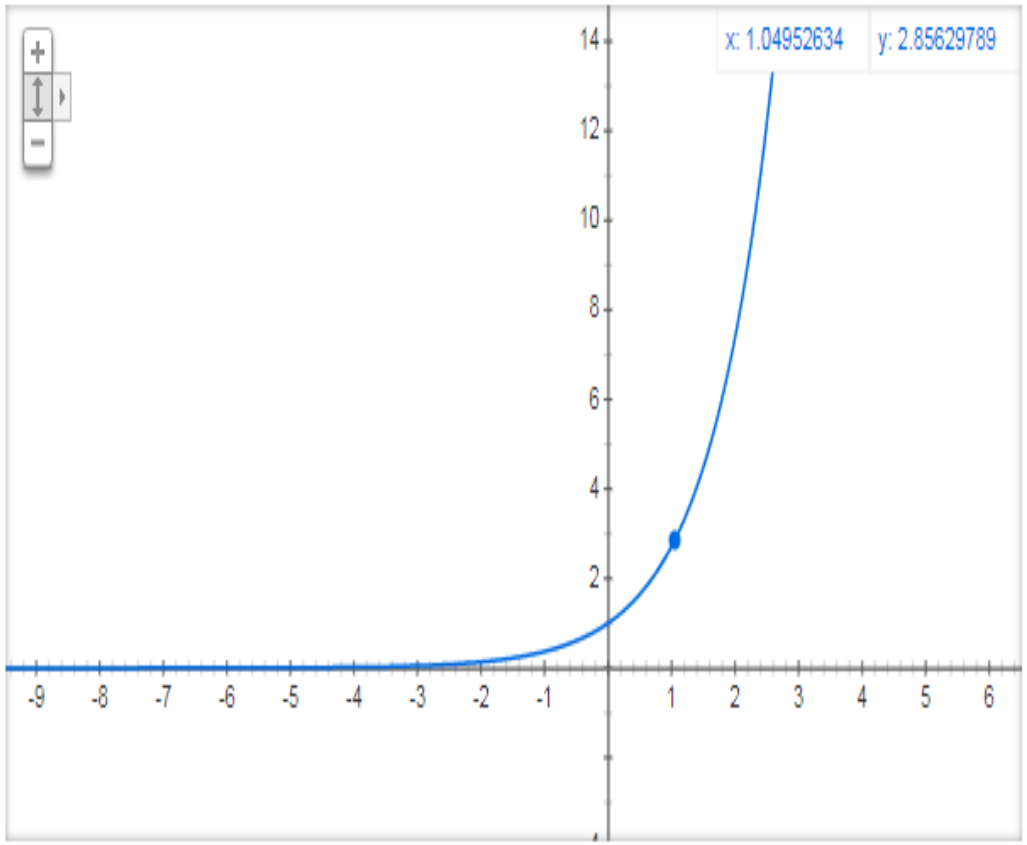
$$\frac{d}{dx} e^x = e^x$$

$$\frac{d}{dx} \ln x = \frac{1}{x}$$

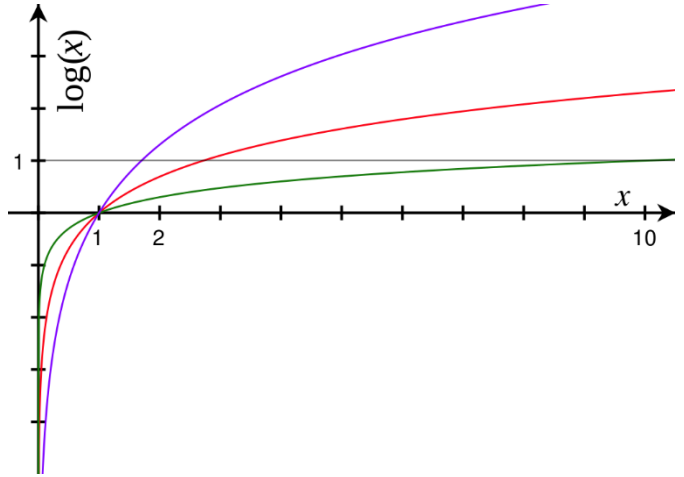
다양한 표현

$$\log_e x = \ln x$$
$$e^x = \exp x = \exp(x)$$

exp(x) 그래프



로그



표기법 : $\log_a X$

수학 함수의 하나로, 특정수를 나타내기 위해서 a(밑수)를 몇번 곱하여야 하는지 표기

지수함수의 역

$$y = a^x \rightarrow x = \log_a y (a > 0, a \neq 1)$$

상수 법칙	$\log_a 1 = 0, \log_a a = 1$
덧셈 법칙	$\log_a xy = \log_a x + \log_a y$
뺄셈 법칙	$\log_a \frac{x}{y} = \log_a x - \log_a y$
지수 법칙	$\log_a x^b = b \log_a x$
밑 변환 법칙	$\log_b x = \frac{\log_k x}{\log_k b}$ (단, $k > 0,\ k \neq 1$)
역수 법칙	$\log_b x = \frac{1}{\log_x b}$ (단, $b \neq 1$)

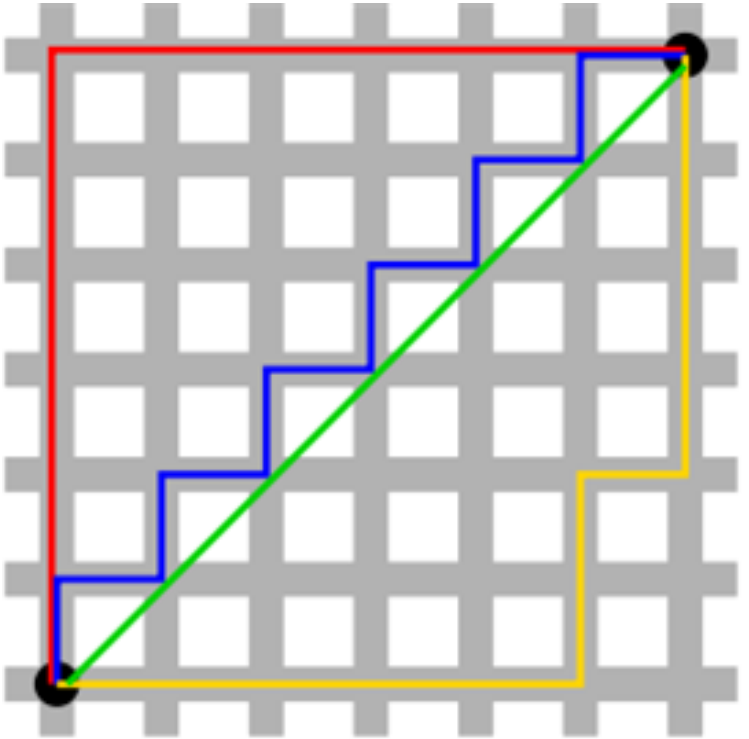
$$\log_a x$$
$$\log_2 x = \mathrm{l}b\, x$$
$$\log_e x = \ln x$$
$$\log_{10} x = \lg x$$

맨하탄 거리

N차원에서 두 점 사이의 이동거리를 구하는 방법

L1 Distance

$$Md = |p_1 - q_1| + |p_2 - q_2| = \sum_{i=1}^{n} |p_i - q_i|$$

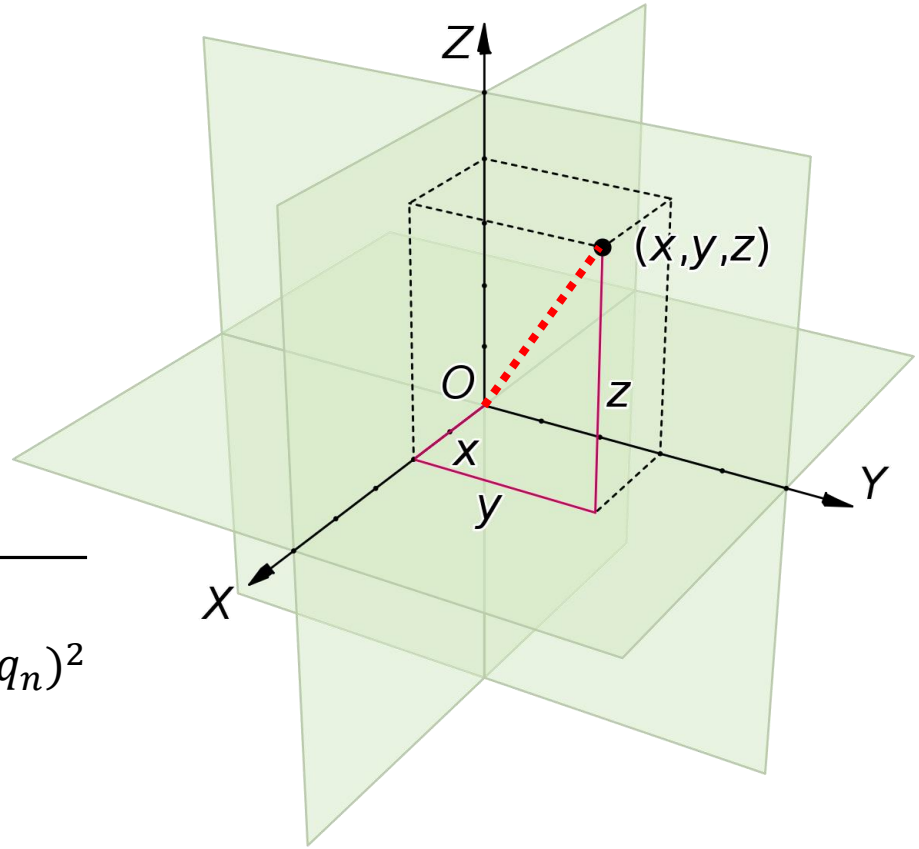


유클리디안 거리

N차원에서 두 점 사이의 최단거리를 구하는 방법

L2 Distance

$$Ed = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_n - q_n)^2}$$



$$\frac{1}{1 + Ed}$$

우리는 이 결과 값을 이용해

두 점이 가깝다면 값이 1에 가까워진다.

두 점이 멀다면 값이 0에 가까워진다.

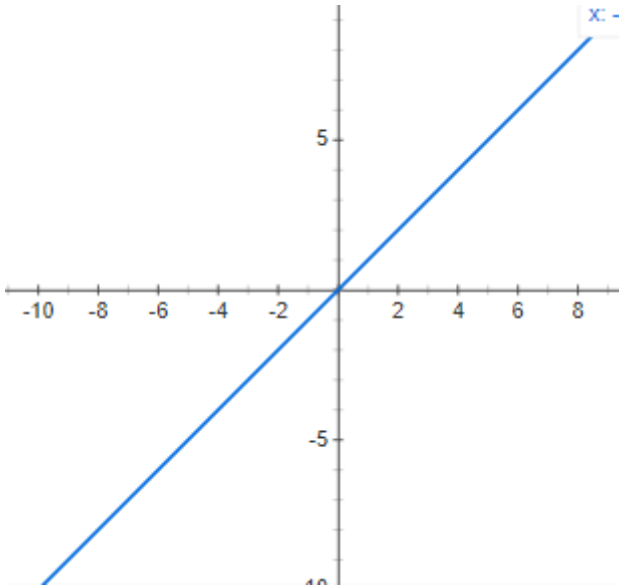
민코우스키 거리

맨하탄 거리와 유클리드 거리를 한번에 표현

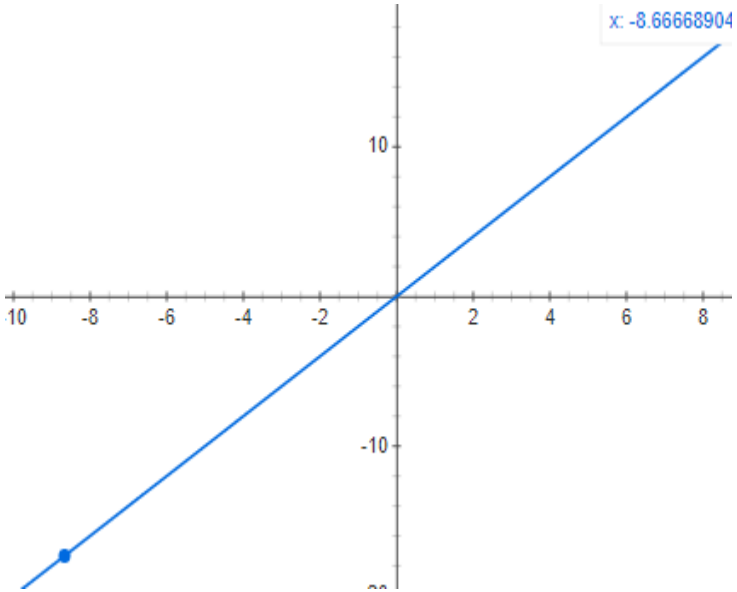
Lm Distance

$$L_m = \sqrt[m]{\sum_{i=1}^{n} (|a_i - b_i|)^m}$$

m = 1 이면 맨하탄 거리
m = 2 이면 유클리디안 거리

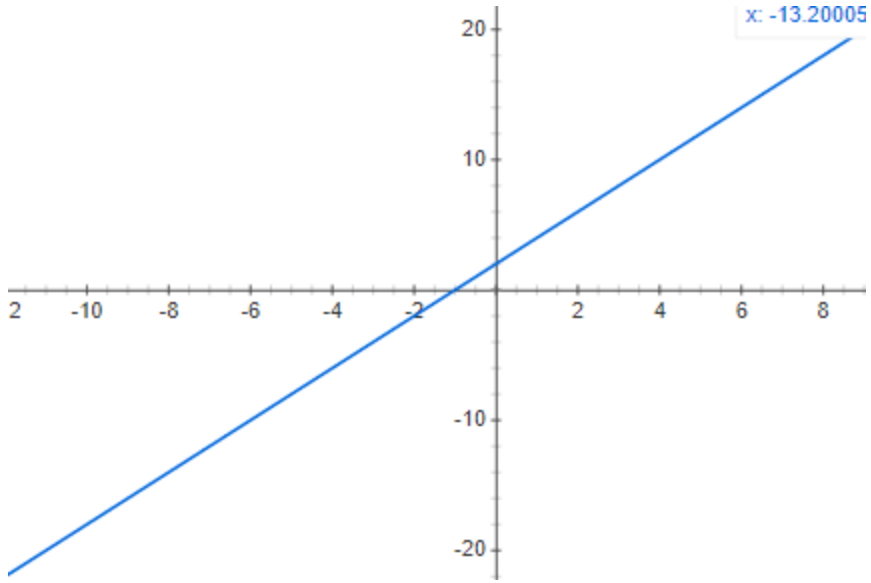


Y = X



Y = AX

X 의 영향력(가중치)이 증가



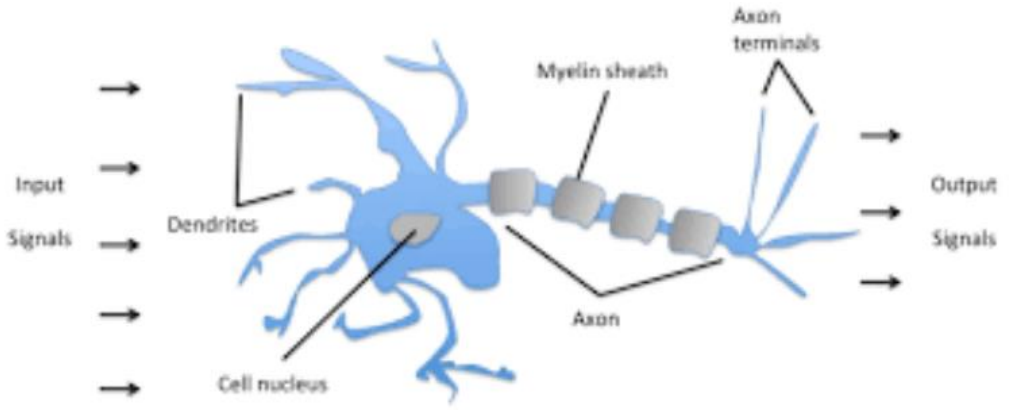
Y = AX + B

B만큼 Y의 값이 증가

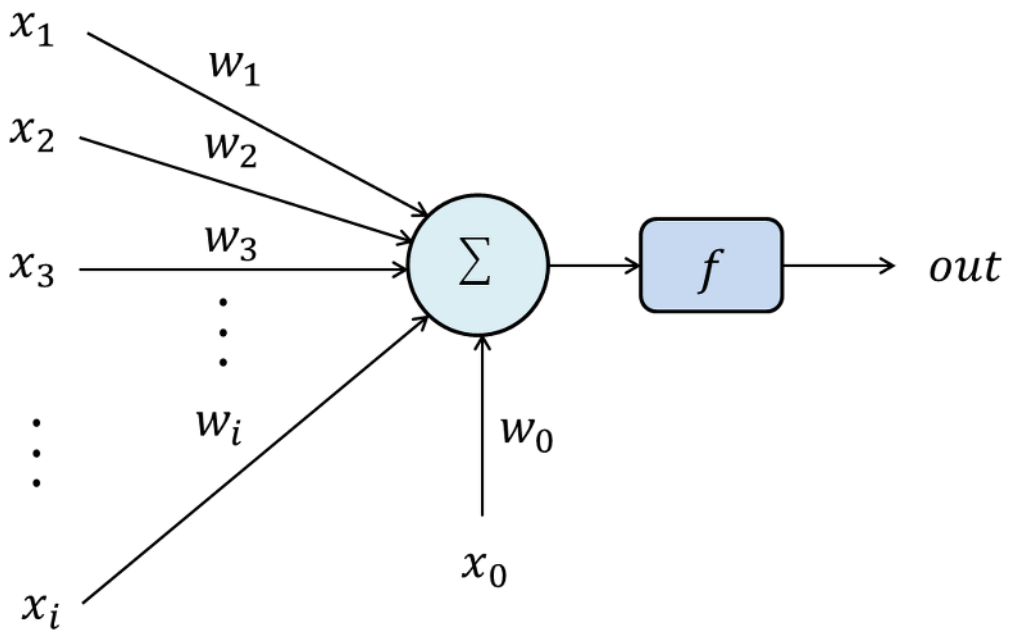
$h(x) = wx + b$

w(weight) 가중치 , b(bias) 편차

Neuron(신경의 최소단위)

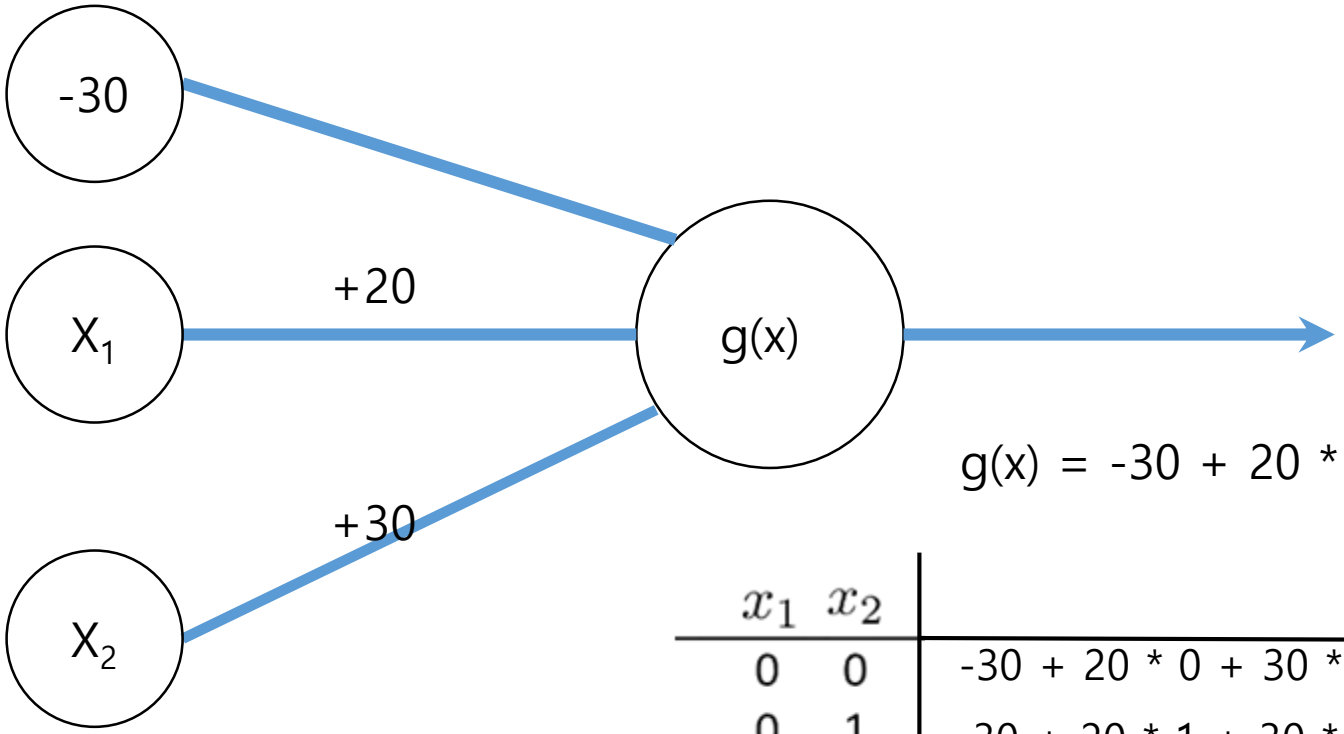


Artificial Neuron(인공 신경)



Perceptron (퍼셉트론) 학습모델

Artificial Neuron(인공 신경)

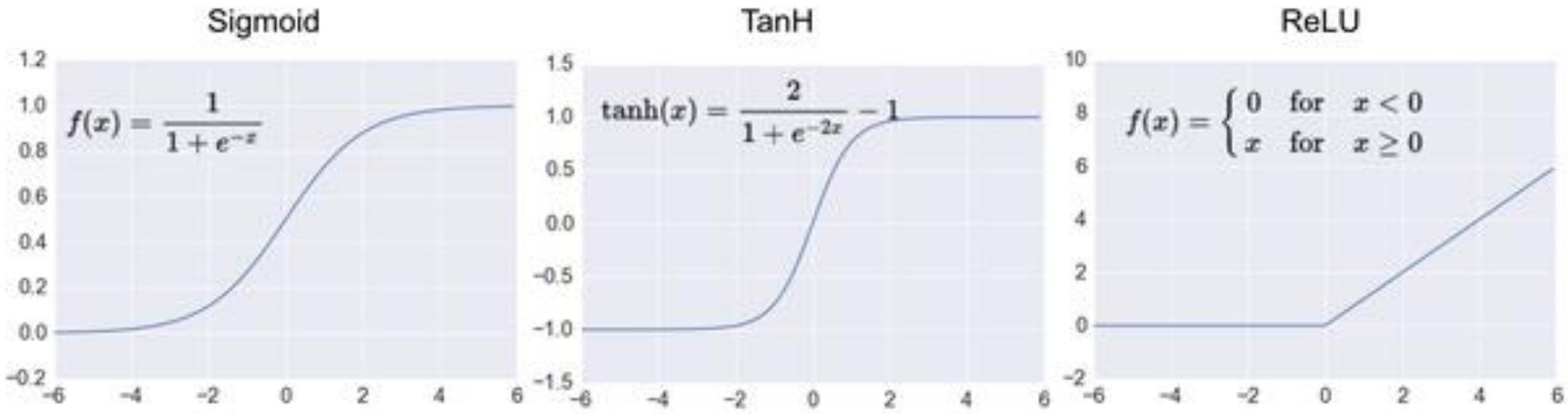


$g(x) = -30 + 20 * X_1 + 30 * X_2$

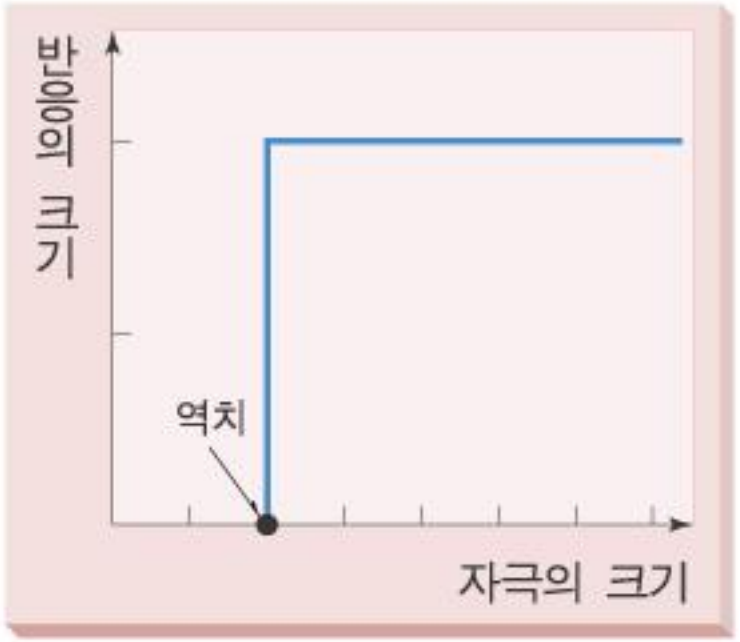
x_1	x_2		g(x)
0	0	-30 + 20 * 0 + 30 * 0 = -30	-30
0	1	-30 + 20 * 1 + 30 * 0 = -10	-10
1	0	-30 + 20 * 0 + 30 * 1 = 0	0
1	1	-30 + 20 * 1 + 30 * 1 = 20	20

x_1	x_2		g(x)	Sigmoid Function
0	0	-30 + 20 * 0 + 30 * 0 = -30	-30	0.00000000000009..
0	1	-30 + 20 * 1 + 30 * 0 = -10	-10	0.00004539786870
1	0	-30 + 20 * 0 + 30 * 1 = 0	0	0.5
1	1	-30 + 20 * 1 + 30 * 1 = 20	20	0.99999999793884

Deep Learning 의 신경망 모델에서 Sigmoid 같은 활성화 함수를 사용한다.



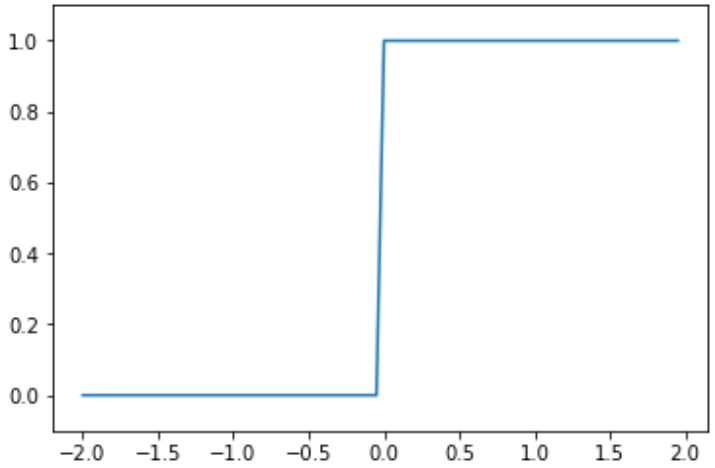
Activation Function(활성 함수)

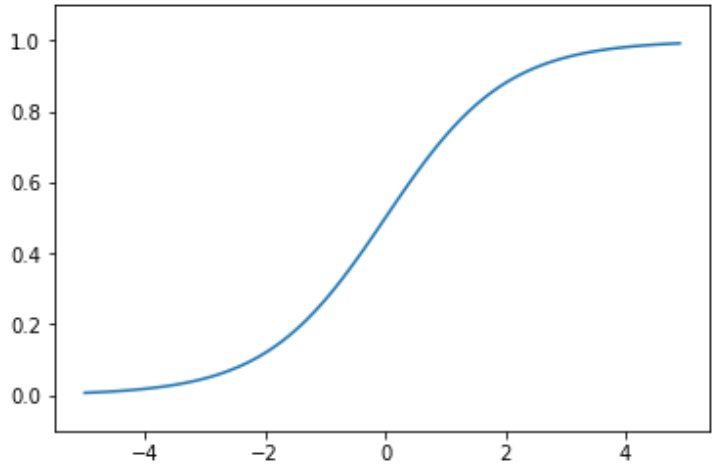


신경은 어떤 역치, 임계값(threshold)을 넘으면 활성화 된다.

마찬가지로 활성 함수도 입력신호의 총합이
활성화를 일으킬지 결정하는 역할

Step Function & Sigmoid Function





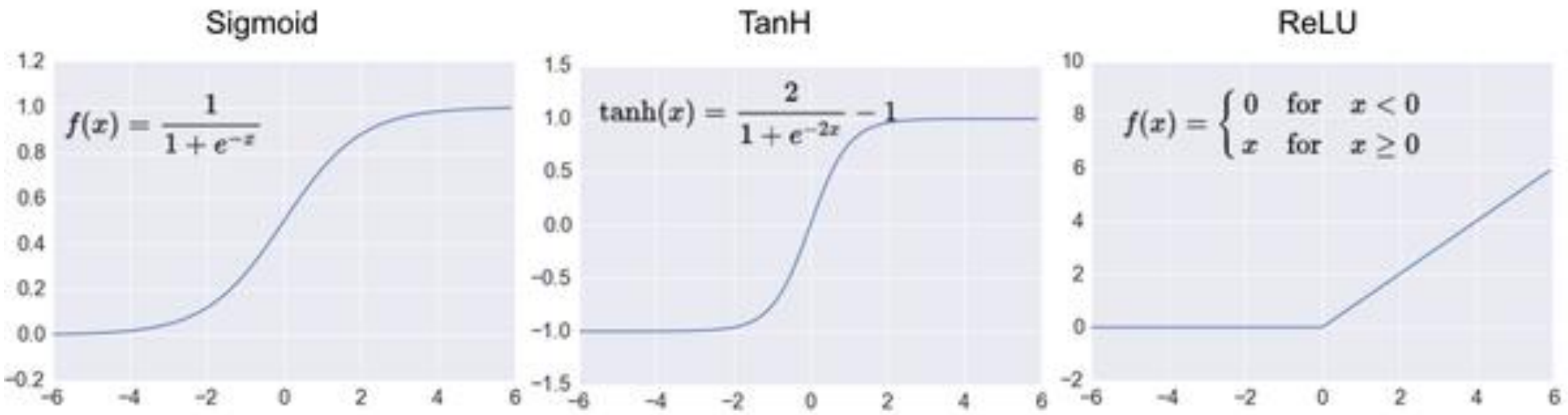
$$f(t) = \frac{1}{1+e^{-t}}$$

비선형

매끄럽게 가중치를 부여

비선형

Activation Function 특징



Why? 비선형 함수

y = ax 라는 선형함수가 있을 때, 활성함수가 여러 개 사용되어도 하나를 사용하는것과 차이가 없다.

y = a(a(a(x))) 라는 선형 활성함수 3개 층 과 $y = a^3(x)$ 는 같다.

미분

- 미분(derivative, 微分) 또는 도함수
- 어떤 함수의 정의역 안에서 각 점에서 함수 값의 변화량과 독립 변수 값의 변화량의 비의 극한

$$
\begin{aligned}
f'(x) &= \frac{df}{dx} \\
&= \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x} \\
&= \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}
\end{aligned}
$$

미분의 아이디어

"어떤 현상, 부분은 전체를 포함한다."

'원인' ➔ '결과'

"모든 사건은 어떤 원인에 의해서 유발된다."

특정 구간 안에서 정의된 임의의 x에 대해서 발생하는 결과를 y라고 했을 때,

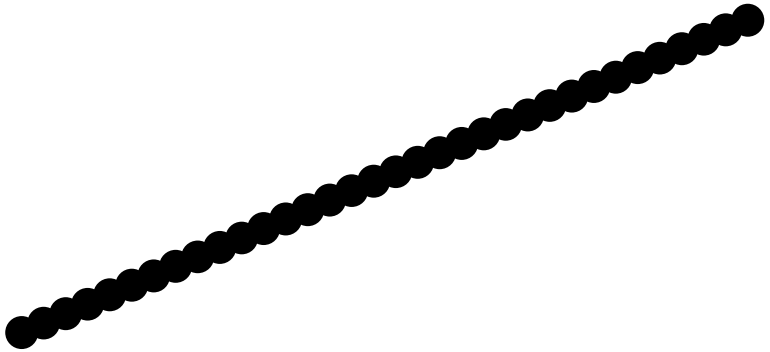
1차식으로 표현하자면 $y = ax + b$

결과가 $y = ax + b$ 라고 표현될 때,

임의의 x는 결과에 대한 원인이 되고 임의의 x는 결과에 대한 정보를 가지고 있을 것이다.

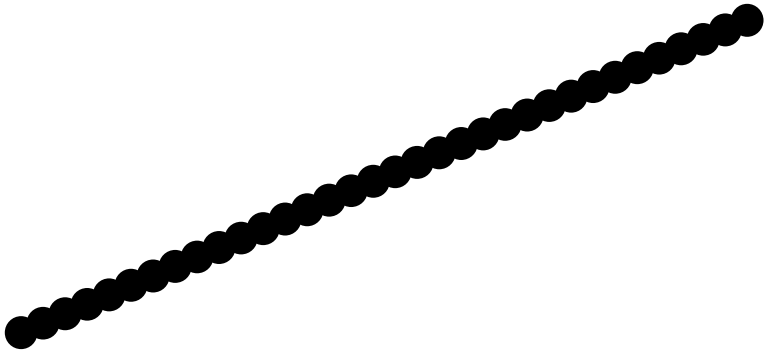
(x, y)

$(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$

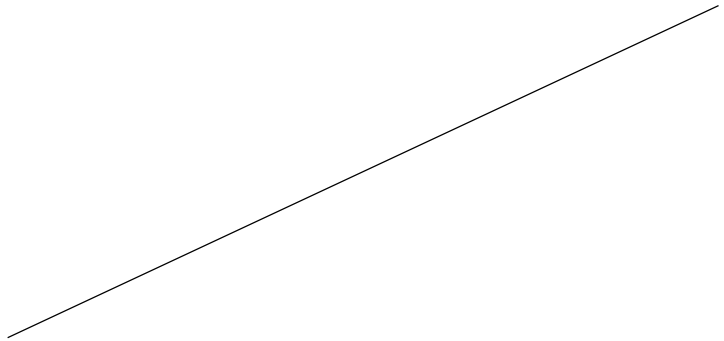


$(1, y_1), (2, y_2), \ldots (n, y_n)$

수열

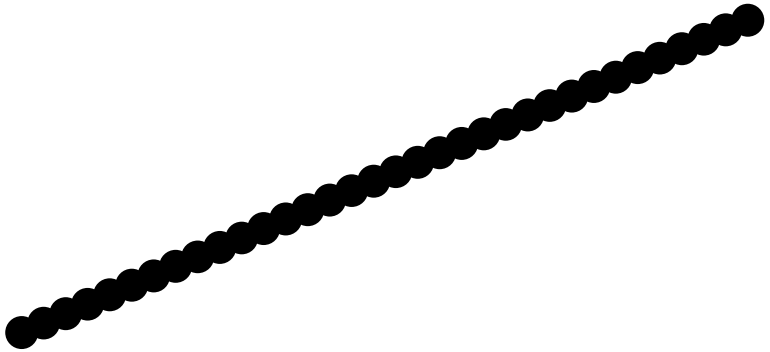


$y = ax + b$



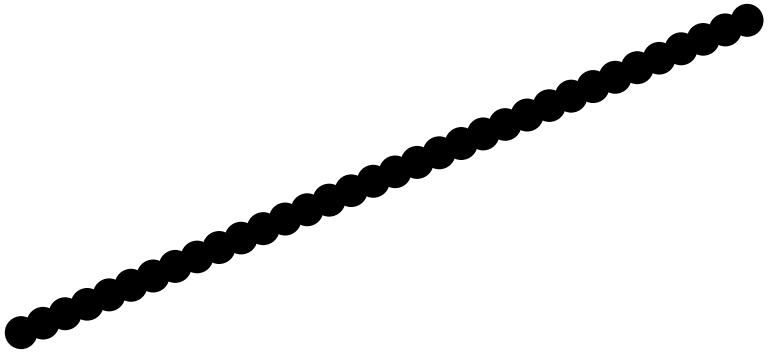
(x, y)

$(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$

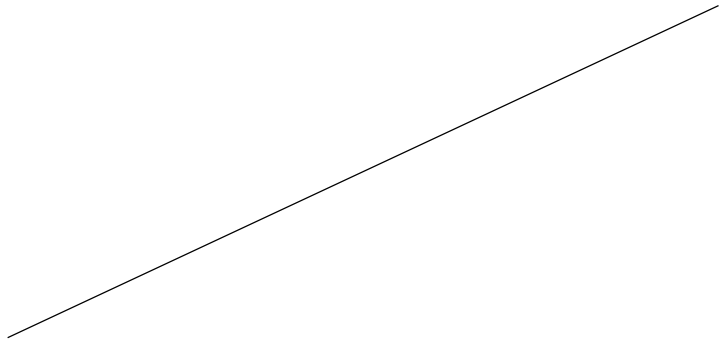


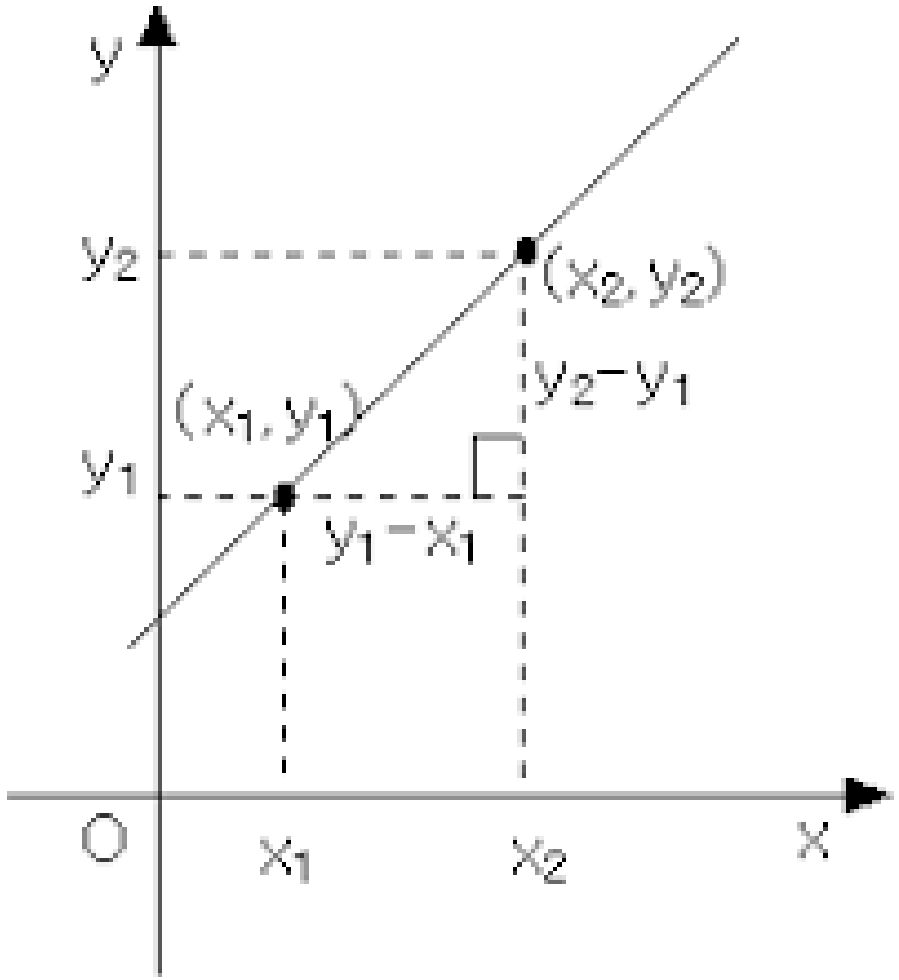
$(1, y_1), (2, y_2), \ldots (n, y_n)$

수열

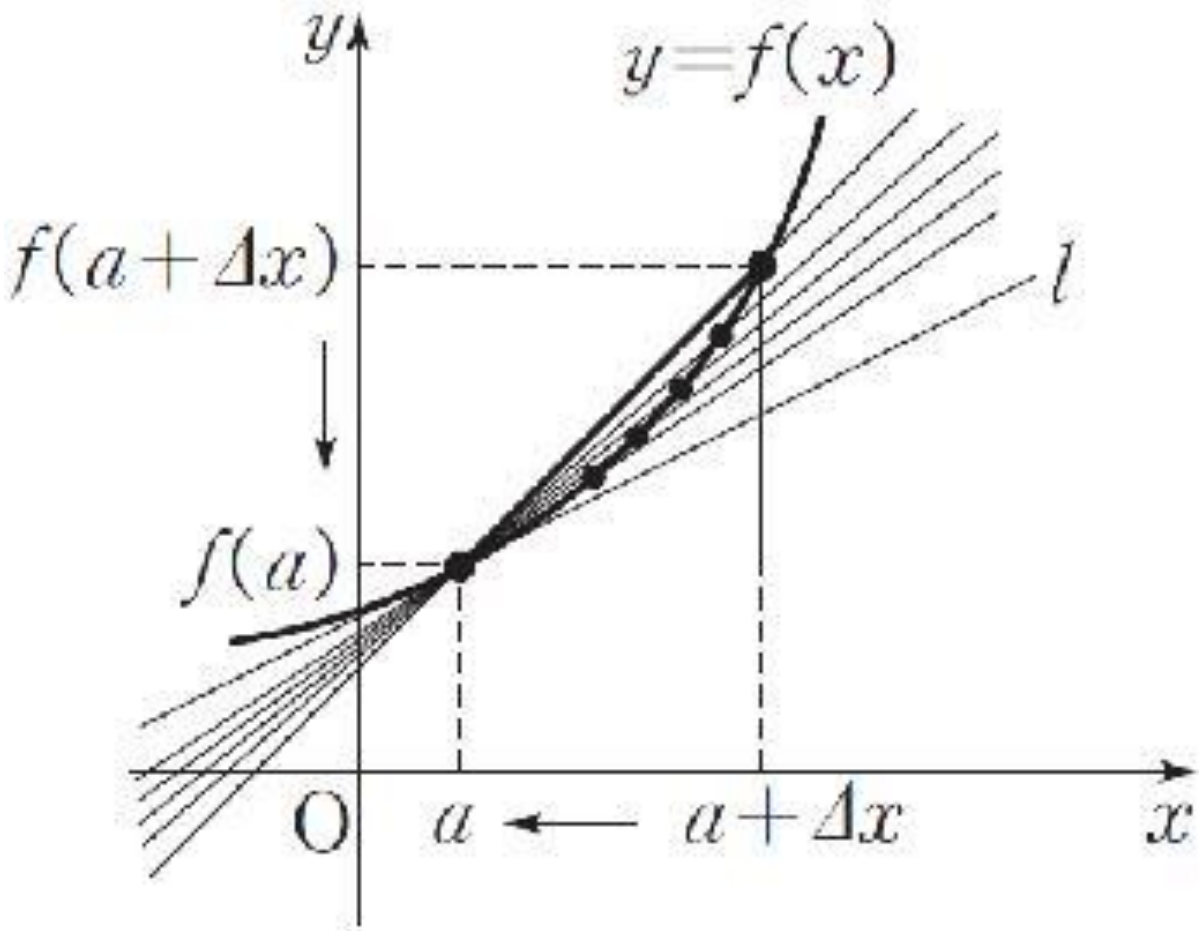


$y = ax + b$

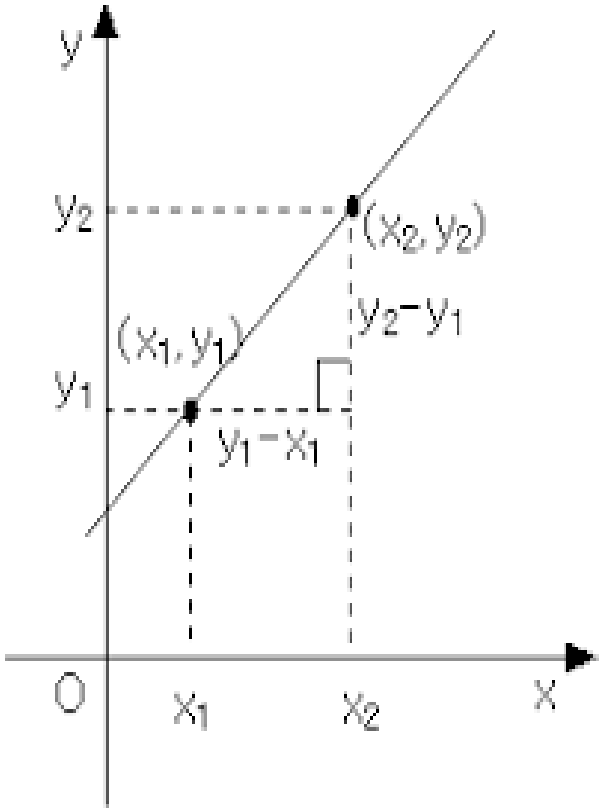




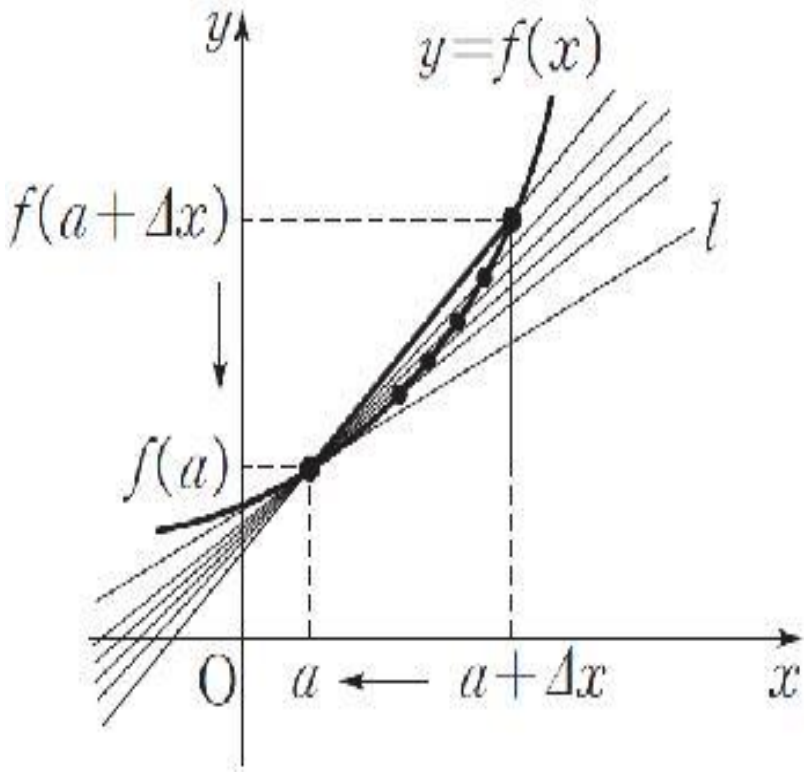
직선의 기울기



접선의 기울기



직선의 기울기



접선의 기울기

$$f'(x) = \frac{df}{dx}$$

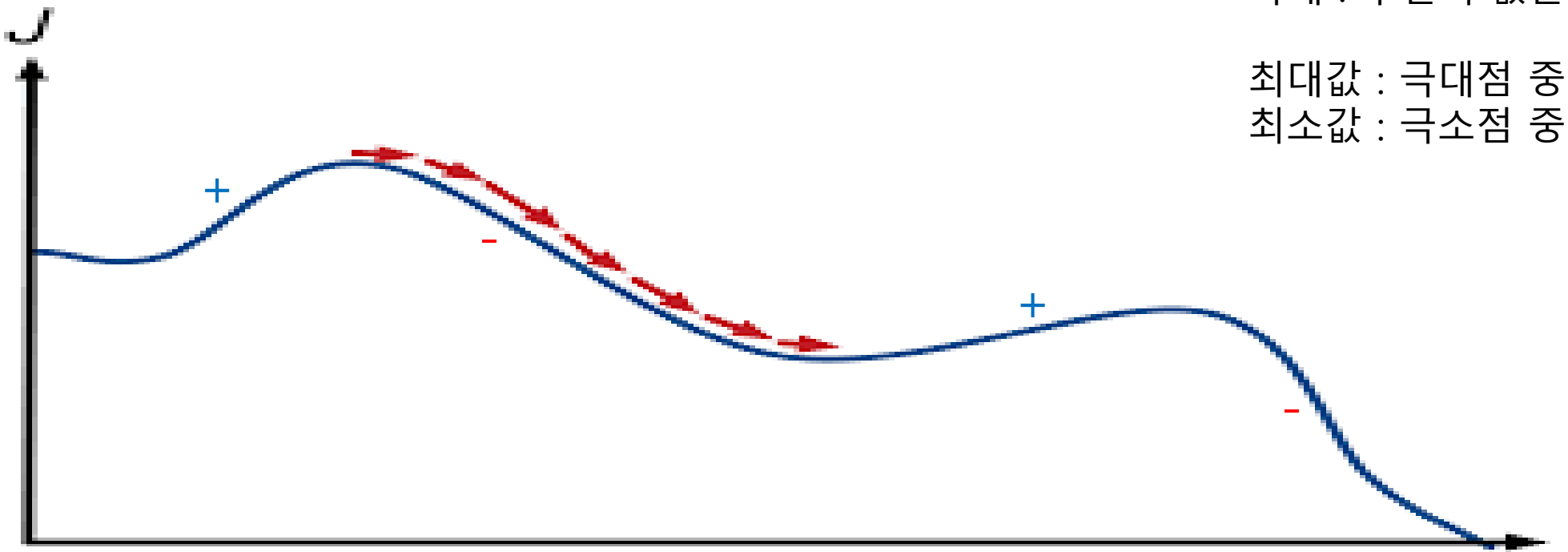
$$= \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

기울기의 변화로 특정 함수의 최대 최소 부분을 탐색

극소 : 주변의 값들보다 작은값
극대 : 주변의 값들보다 큰값

최대값 : 극대점 중 하나
최소값 : 극소점 중 하나



미분을 통한 최대, 최소 값

1차 미분의 결과를 가지고

양에서 음으로 변하는 부분을 극대, 음에서 양으로 변하는 부분을 극소라고 한다.

x	-3	1	10	15	23
$f'(x)$	-	0	+	0	-
$f(x)$	20	4	85	90	73
상태	감소	극소	증가	극대	감소

이걸 이용해서 Loss Function이 최소가 되는 점을 찾아낼겁니다.

다다다음시간에 회귀분석(linear regression)할때 사용해볼겁니다.

Loss Function(손실 함수) or Cost Function

정답과 예측값 사이의 오차를 표현하는 함수
딥러닝에서는 Loss Function을 최소로 만들기 위해 다양한 기법을 사용한다

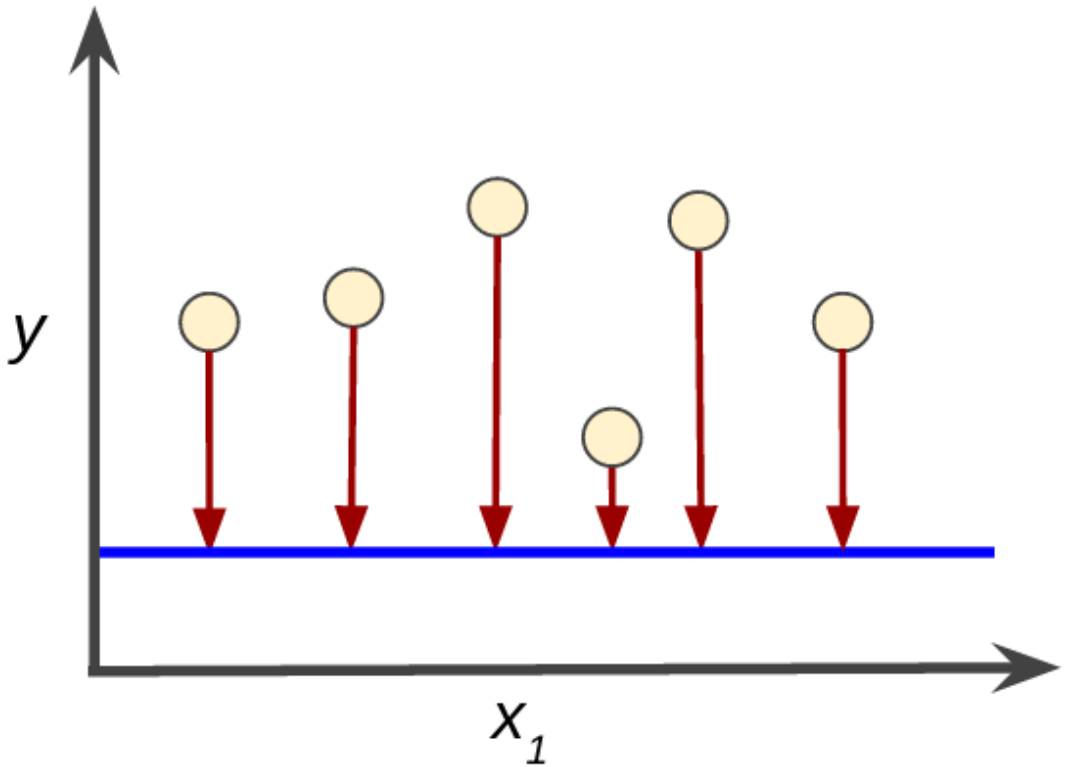
Decision criteria

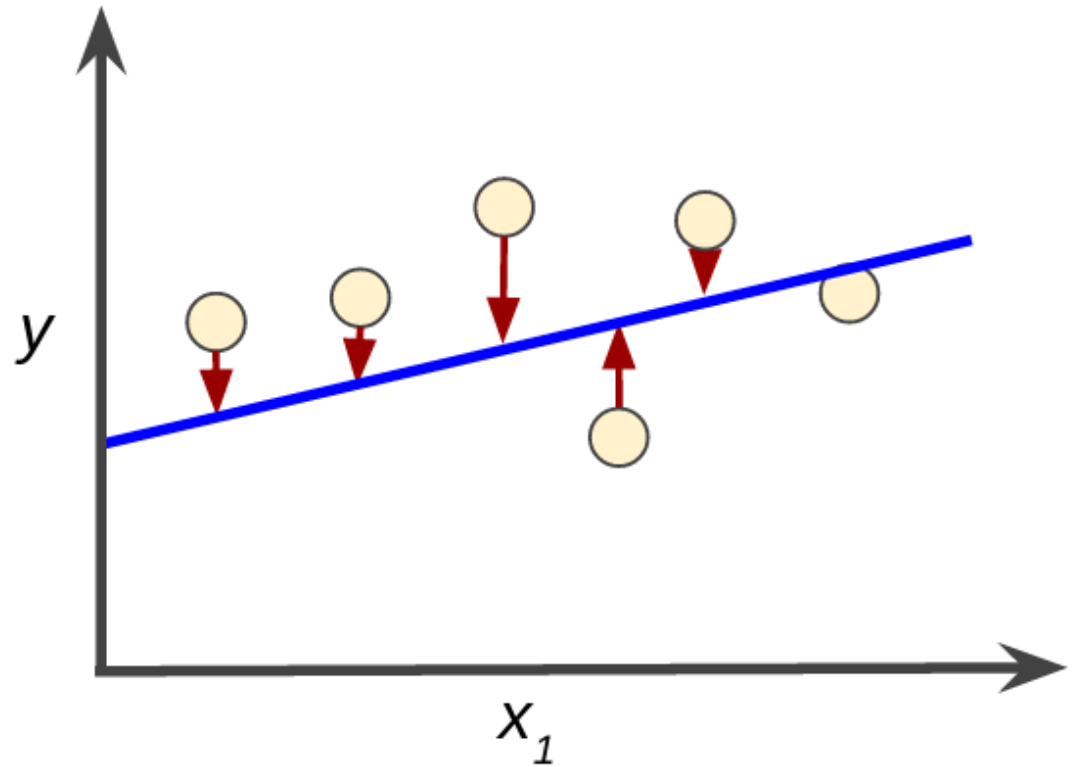
결정을 내리기 위한 조건
- 가장 쉬운 Yes / No Question 처럼 (맞다 / 틀리다) 의 방법
- 가장 Formal하고 많이 쓰이는 방법 Loss Function

즉, 인공지능을 통해 우리가 원하는 결과(target data)와 우리의 모델로 예상된 결과(estimated data)의 차이(Loss Function)이 최소가 되는 모델이 좋은모델

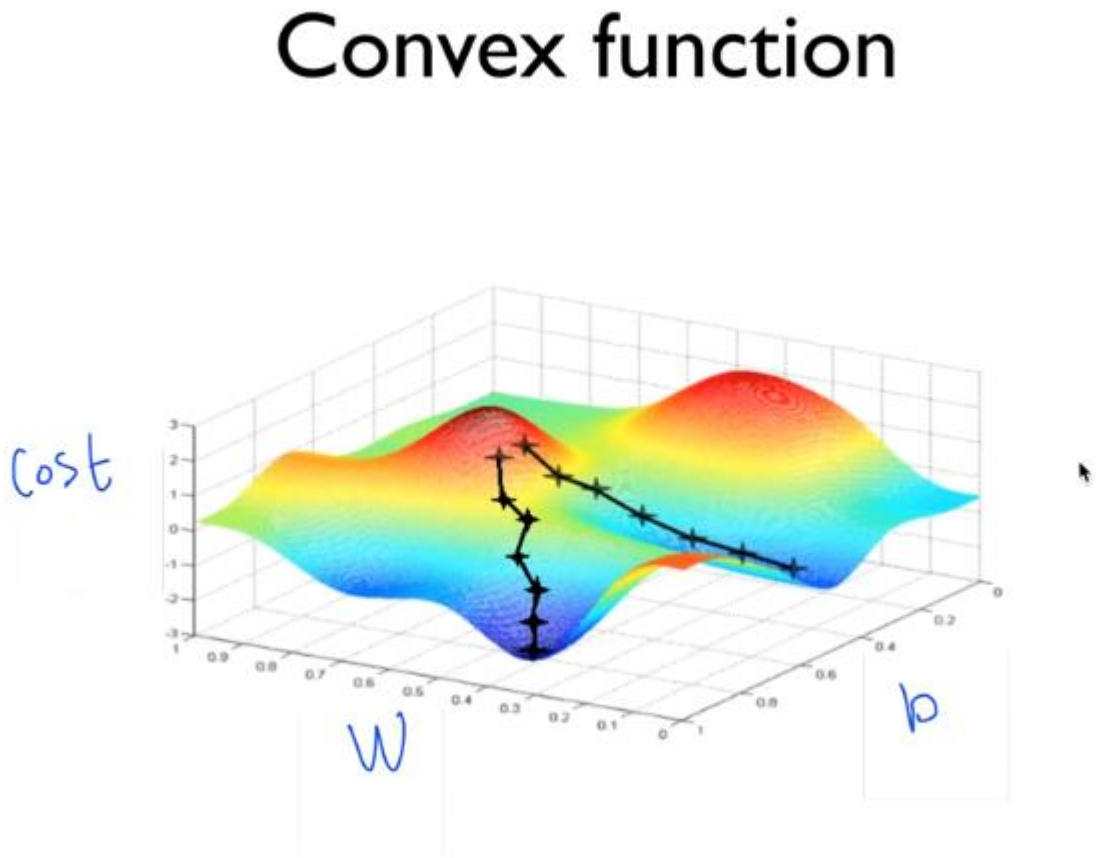
Loss Function은 $L\left(\theta, \hat{\theta}(x)\right)$로 표기되며, θ는 우리가 원하는 또는 정답으로 나와있는 true parameter이고, $\hat{\theta}(x)$은 주어진 x에 대하여 우리의 모델로 estimate한 parameter이다.

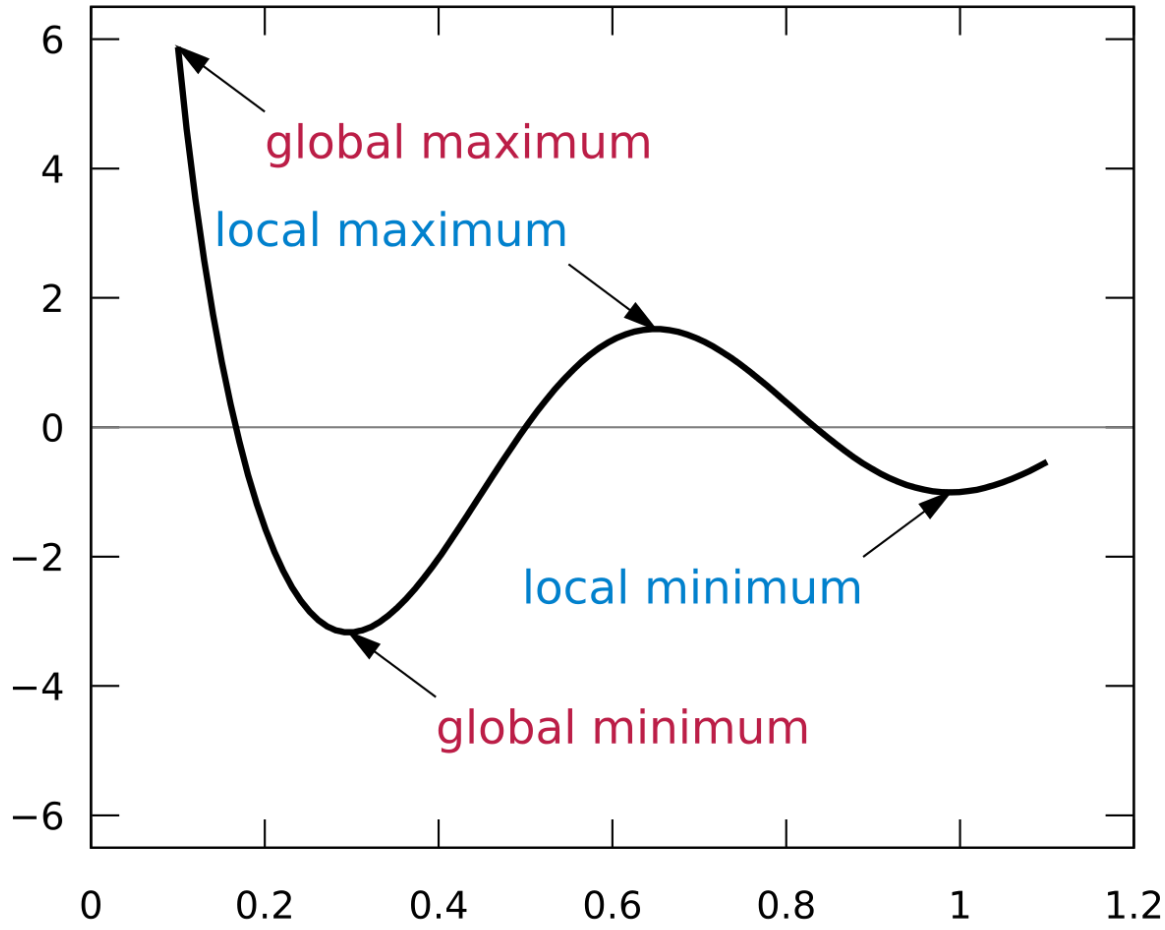
모델의 목표는 두개의 차이가 최소화 되는 모델을 찾는것이다.


y
x_1


y
x_1

Gradient descent(경사 하강법)





www.holehouse.org/mlclass/

수치 미분

$$f'(x) = \frac{df}{dx}$$

$$= \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

Δx 를 한 없이 0에 가깝게 한다는 의미로 lim

분자는 x에 대한 변화량을 의미

하지만 컴퓨터에서 계산할 때는

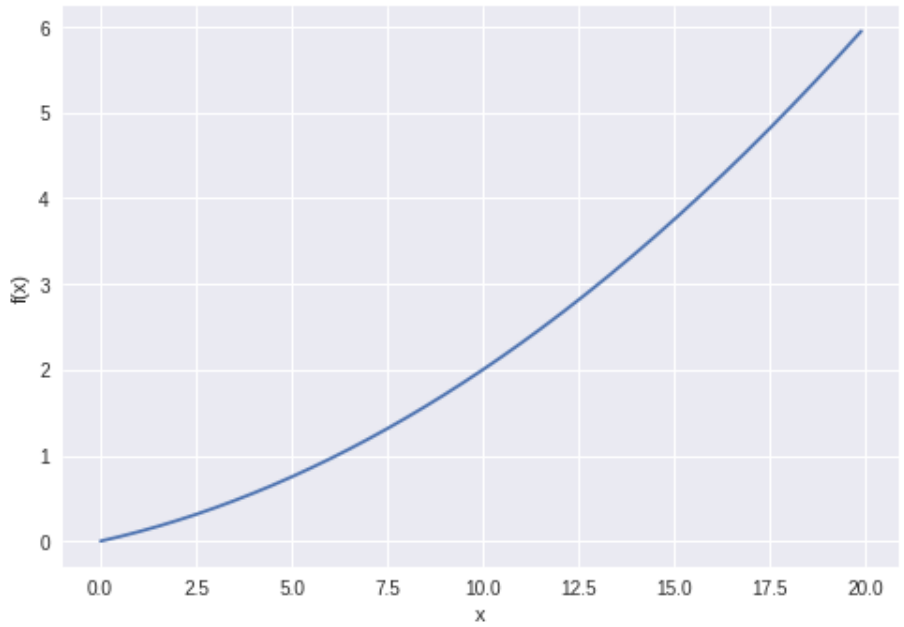
차분(임의의 두점에서 함수 값들의 차이)로 미분을 한다.

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

```
def numercial_diff(f, x):
    h = 1e-4
    return (f(x+h)-f(x-h))/2*h
```

```
def function_1(x):
    return 0.01*x**2 + 0.1*x

x = np.arange(0.0, 20.0, 0.1)
y = function_1(x)
plt.xlabel("x")
plt.ylabel("f(x)")
plt.plot(x,y)
plt.show()
```




```
numercial_diff(function_1, 5)
```

```
1.9999999999908982e-09
```

```
def numercial_diff(f, x):
    h = 1e-4
    return (f(x+h)-f(x-h))/2*h
```

f(x+h)-f(x-h) / 2*h

중심을 기준으로 양쪽 차이를 미분한다

중심 차분 또는 중앙 차분

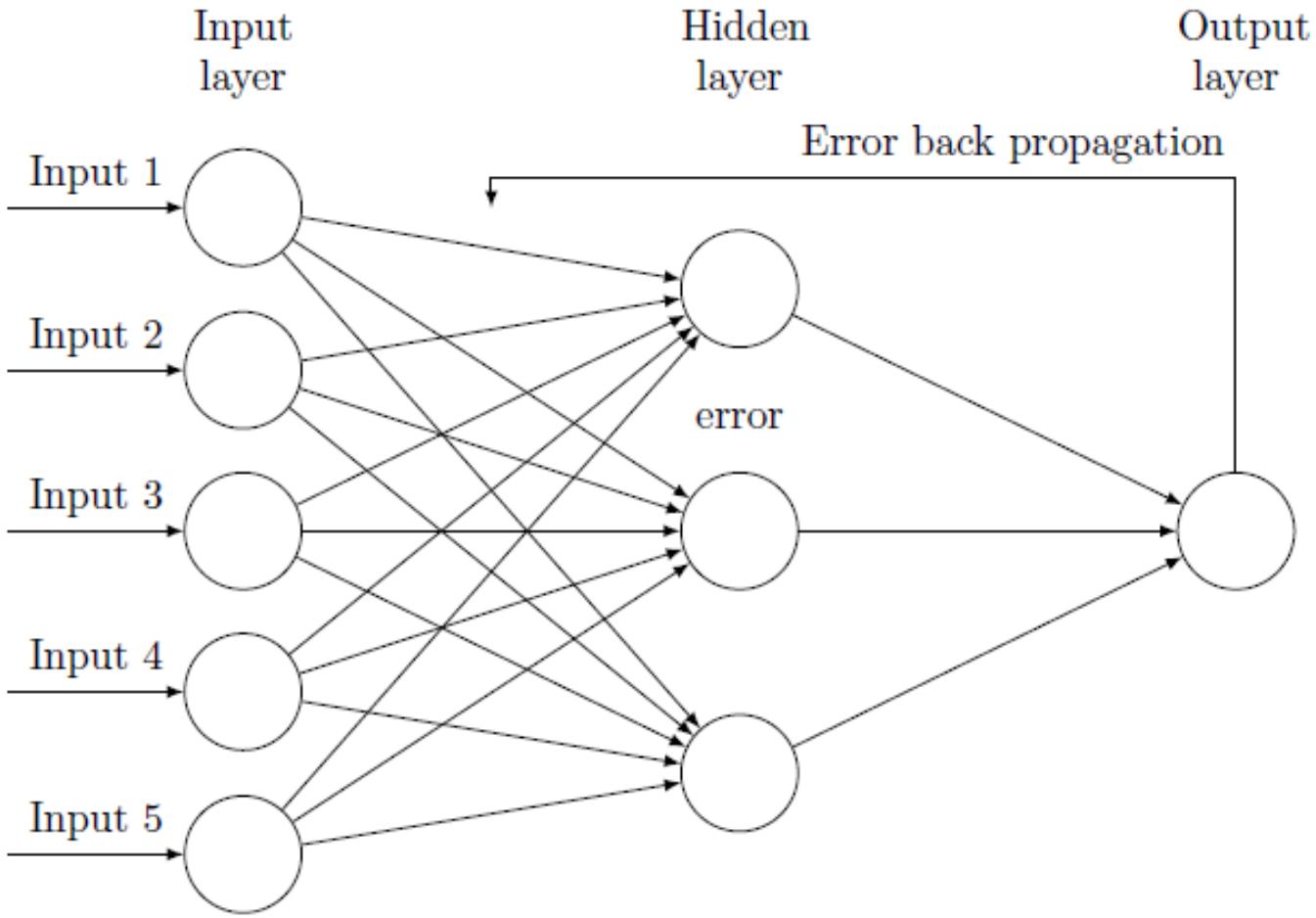
https://drive.google.com/open?id=1rYsRD1R2gUYsNV114ku8q9aD7L3VkWlo

신경망

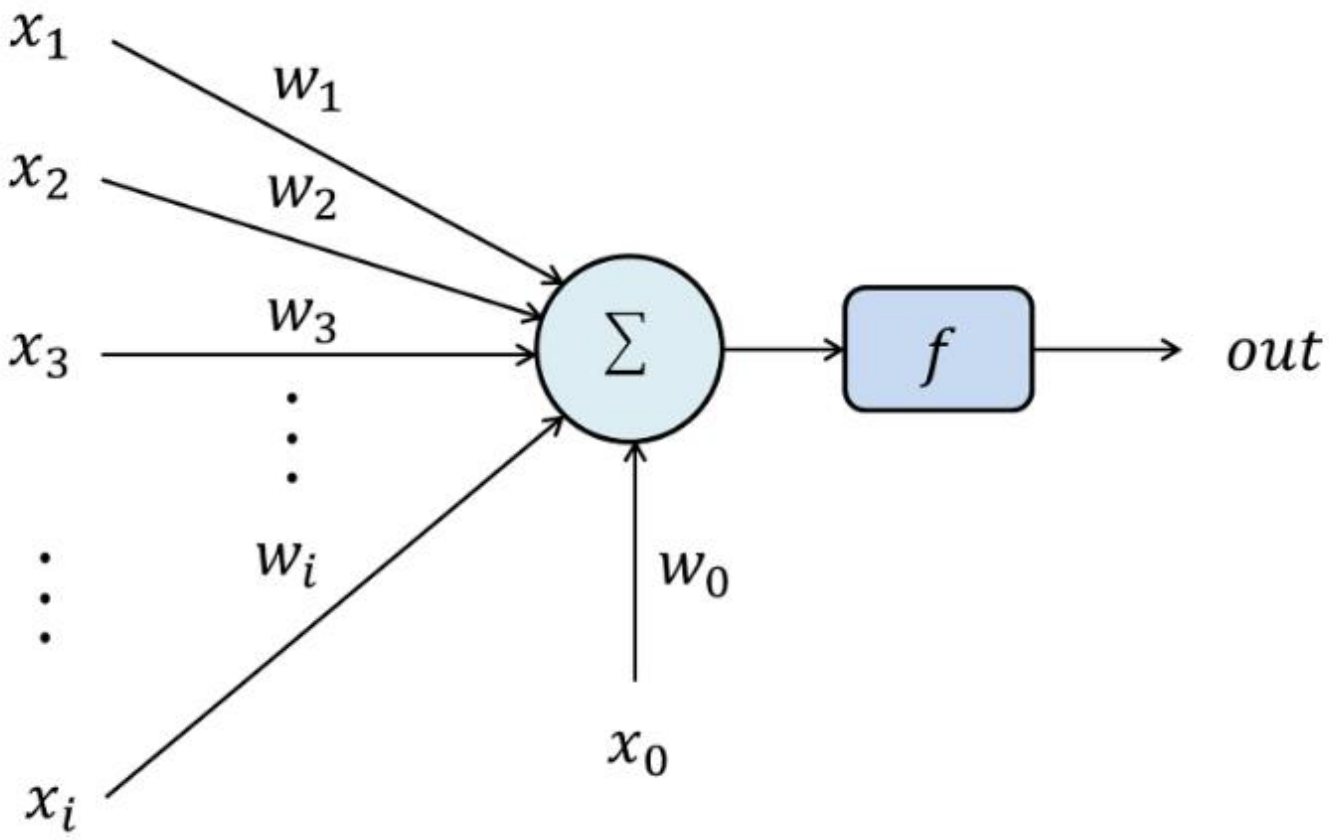
Input : 입력층은 들어온 신호를 그대로 전달
아무런 계산을 하지 않는다.

Hidden : 신경망 외부에서 다른 접근은 불가
가중치를 곱하고 편향을 더한 후
각 노드별 총합 계산.

Output : 출력층에서 노드들의 출력 최종 결과



신경망의 계산



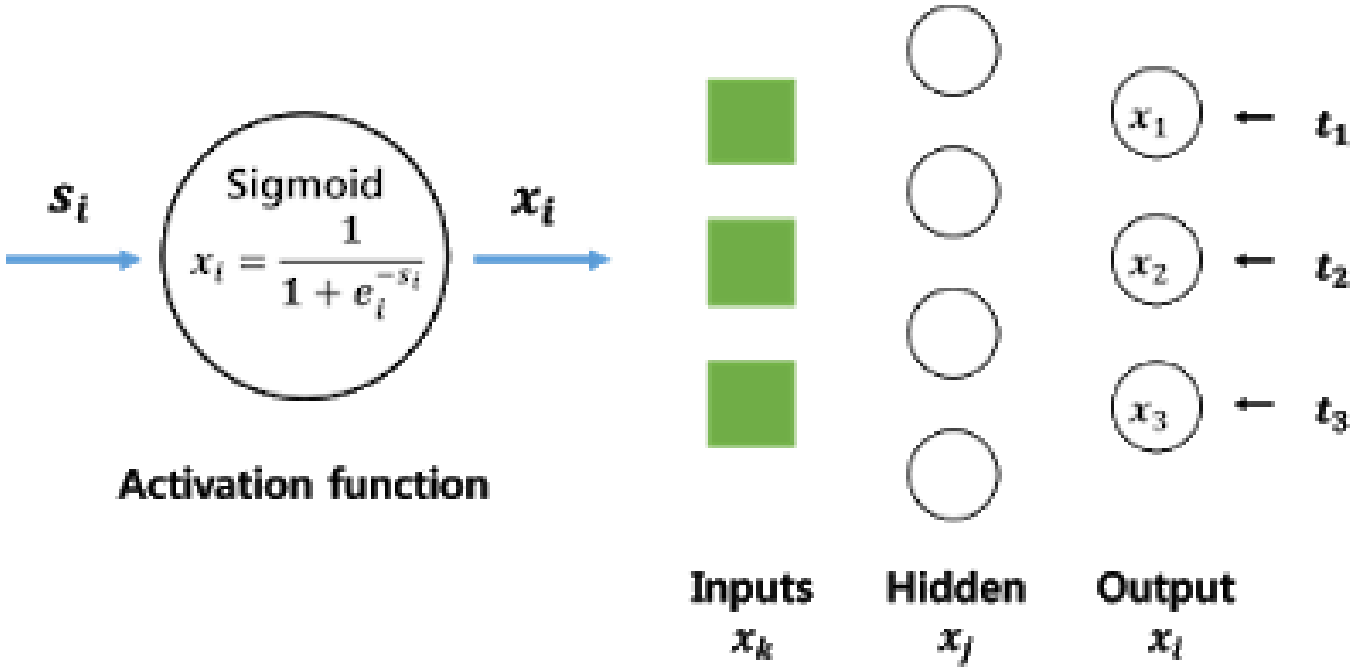
$step1.\ input\ x_0, x_1, x_2, \ldots\ldots, x_n$

$step2.\ x_0\ w_0, \qquad x_1 w_1, \ldots\ldots, x_n w_n$

$$step3.\ \sum_{i=0}^{n} x_i w_i + b_i$$

$$step4.\ S(\sum_{i=0}^{n} x_i w_i + b_i)\,, S(x)\ := Sigmoid\ function$$

미분이 중요한 이유



딥 러닝에서는 다음과 같이 Input → Hidden → Output 레이어로 진행되면서 결과가 나온다.

Output layer에서는 estimated data가 나올것이고 그것과 true parameter와 비교해서 오차를 계산하게 된다. (Loss Function)

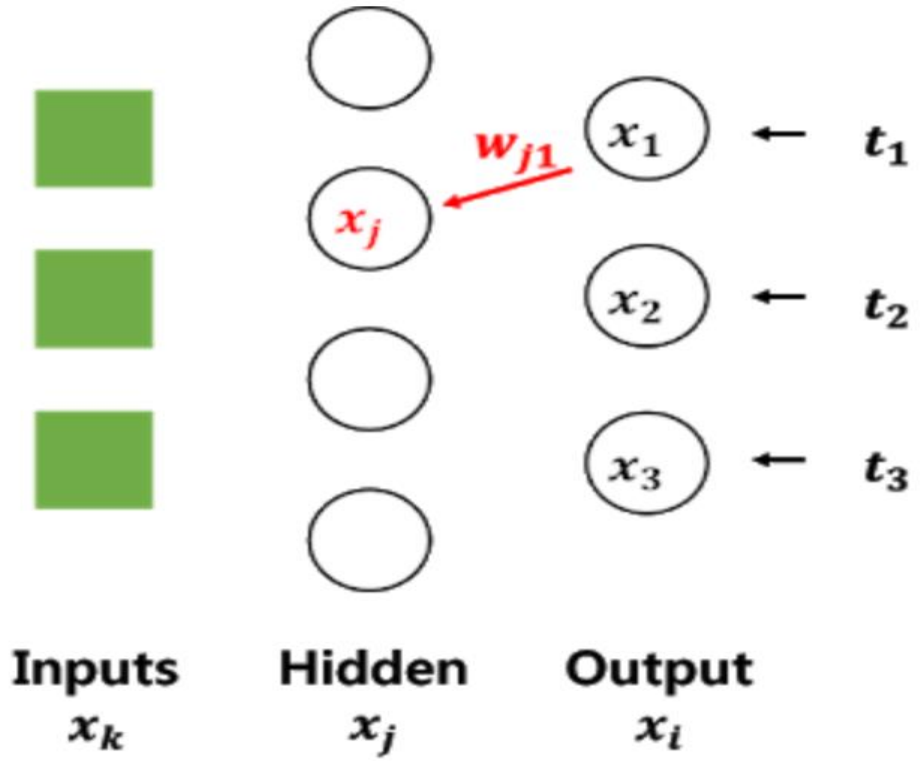
그럼 여기서 estimated data의 Loss 를 어떻게 줄여야하는가?
(Data의 Input을 변경하는것은 우리의 목적이 아니기 때문에 네트워크의 Output값에 영향을 주는 변수중 우리가 수정할 것은 layer간의 weight이다.)

Layer의 Weight에 대한 Loss Function의 기울기(gradient)를 계산하고,

Loss Function이 최소가 되도록 매개변수인 weight를 최적화 함으로 학습한다.

이제 각 weight가 output에 얼마나 영향을 미치는지 확인하기 위해서 역으로 연산을 하기 시작한다

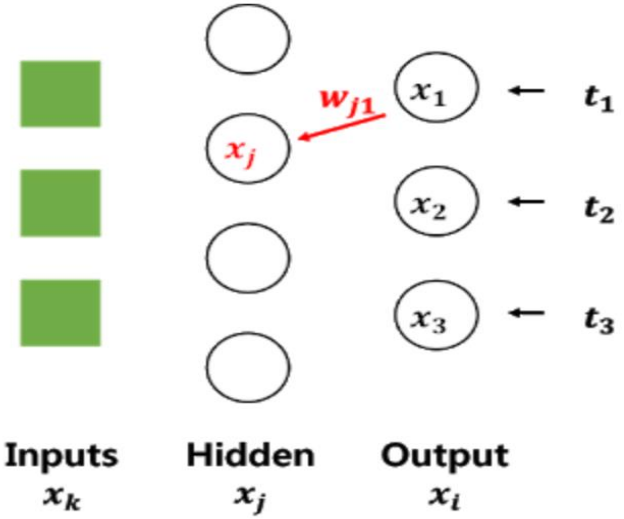
Output layer와 Hidden layer사이의 weight를 변경



수식 :

$$E = -\sum_{i=1}^{nout}(t_i log(x_i) + (1 - t_i) log(1 - x_i))$$

where, $x_i = \frac{1}{1+e^{-s_i}}$, $s_i = \sum_{j=1} x_j w_{ji}$.

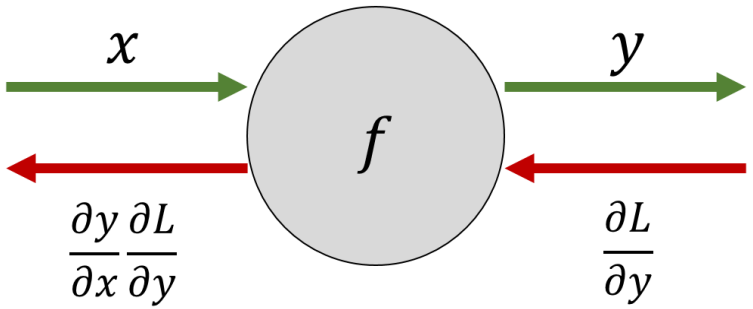


$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial s_i}\frac{\partial s_i}{\partial w_{ji}}$$

1. $\frac{\partial E}{\partial x_i} = \frac{-t_i}{x_i} + \frac{1-t_i}{1-x_i} = \frac{x_i - t_i}{x_i(1-x_i)}$
2. $\frac{\partial x_i}{\partial s_i} = x_i(1 - x_i)$
3. $\frac{\partial s_i}{\partial w_{ji}} = x_j$

$$\therefore \frac{\partial E}{\partial w_{ji}} = (x_i - t_i)x_j$$

합성함수의 미분 (Chain Rule)



순전파(Forward Propagation)

- "신경망이 계산되는 방법"

오차역전파 (Back Propagation)

- "신경망이 학습되는 방법"

- 학습되는 방법을 알아야 뉴럴 네트워크의 레이어도 이해하고 쌓을 수 있다.

$\frac{\partial y}{\partial x}\frac{\partial L}{\partial y} = \frac{\partial L}{\partial x}$ x에 대한 Loss(오차)의 변화량

현재 노드의 x = weight * data 에 대한 변화량
Data는 변화하지 않기 때문에 weight에 대한 오차의 변화량이라고 볼 수 있다