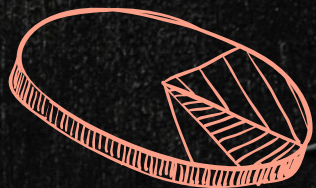
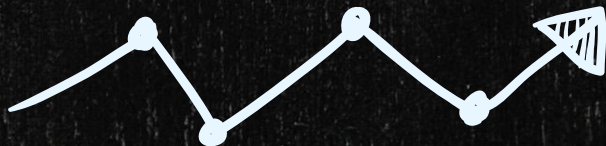
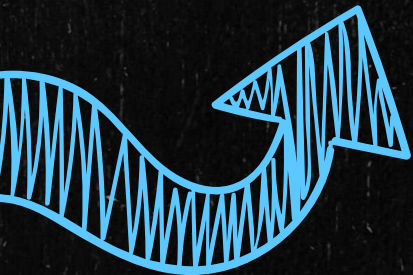


# Everyone's Deep Learning

## CH5. 참 거짓 판단 장치: 로지스틱 회귀



3조: 길태호, 심재성, 양지윤, 이용빈





# CONTENTS



01

## 로지스틱 회귀

참, 거짓 판단 장치

02

## 시그모이드 함수

활성화 함수,  
시그모이드 함수의 특징,  
왜 사용해야 하는가.



03


## 오차 함수

오차 함수의 의미와 종류, 의의,  
활용성

04

## Pytorch

tensorflow의 경쟁자  
딥러닝 프레임 워크 torch

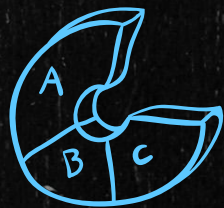
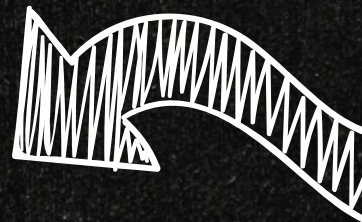






# 로지스틱 회귀

참과 거짓을 판단하는 장치





# Logistic Regression



머신 러닝

컴퓨터가 빅데이터 속에서 특정한 규칙을 찾아내어 추론할 수 있게 하는 기술

비지도 학습

지도 학습

회귀

분류

이진 분류

다중 분류

## \*회귀와 분류의 차이점

- 회귀: 예측값이 실수(숫자)이며 연속적이다.
- 분류: 예측값이 범주형 변수이며 이산적이다.



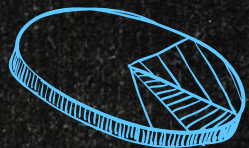
로지스틱 회귀

Logistic Regression





# 회귀 분석 Regression의 의미



## 회귀 분석

\*회귀: 한 바퀴 돌아 제자리로 돌아옴.

\*회귀 분석: 평균보다 크거나 작은 값들이 평균으로 돌아간다는 가정을 이용한 분석 방법

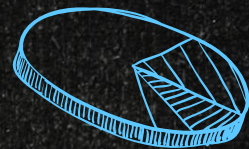
=>평균으로의 회귀

->연속적인 독립변인이 종속변인에 영향을 미치는지 알아보고자 할 때 실시.



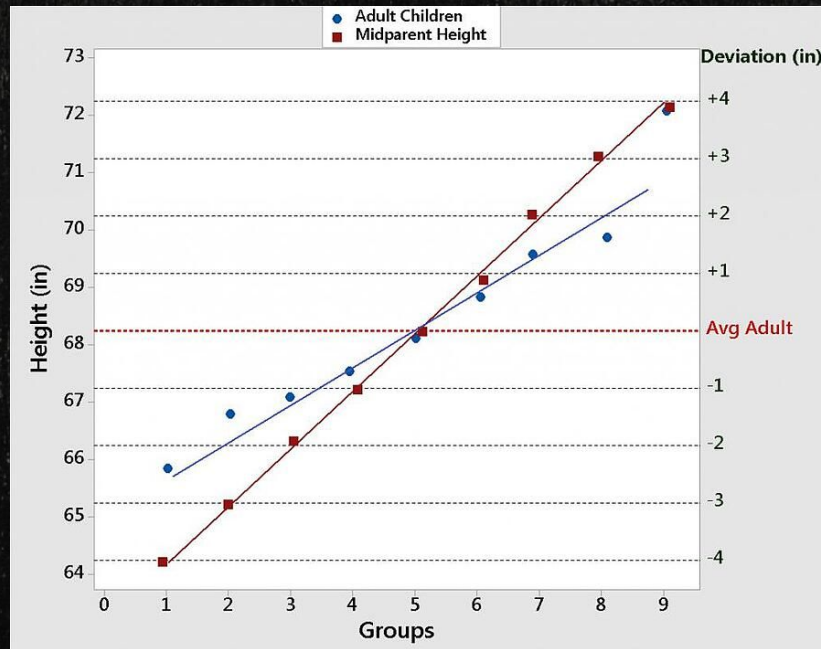


# 회귀 분석 Regression의 시초



키가 큰 아버지는 키가 조금 작은  
자식을, 키가 작은 아버지는 키가  
조금 큰 자식을 갖는다.

-Francis Galton





# 회귀 분석 Regression 종류



- 단순 선형 회귀(독립변수 1개) & 다중 선형 회귀(독립변수 2개 이상)
- 경사 하강법 이용
- 점들의 특성을 담은 직선



- 종속변수가 이항변수  
ex) 참 or 거짓, 1 or 0, 합격 or 불합격
- 경사 하강법 이용
- 점들의 특성을 담은 S자 곡선

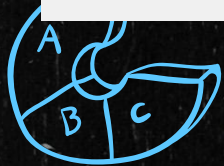
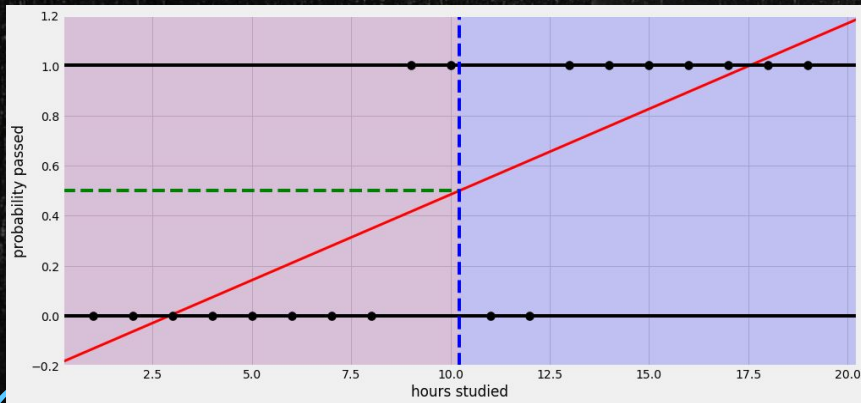
# 딥러닝의 기본 계산 원리

=> 선형 회귀(선형) & 로지스틱 회귀(0 or 1)

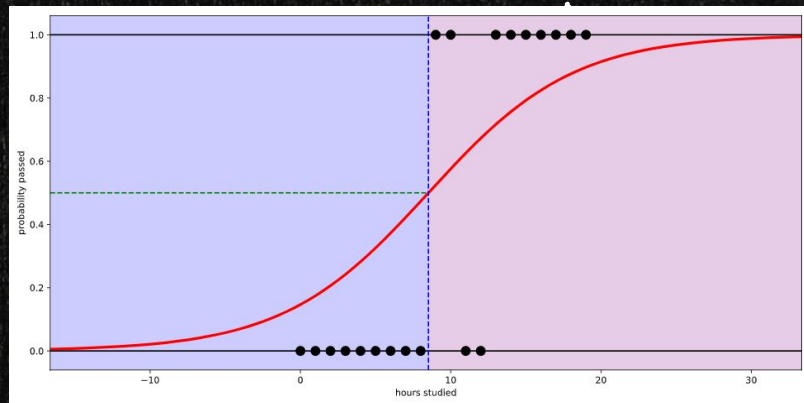


## 로지스틱 회귀

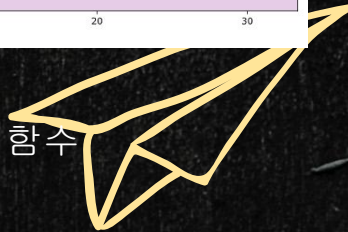
참(1)과 거짓(0) 사이를 구분하는 S자 형태의 적절한 선을 그리는 과정



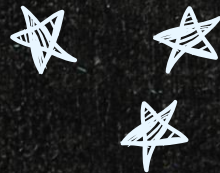
선형 회귀 분석 함수



로지스틱 회귀 분석 함수







# 시그모이드 함수

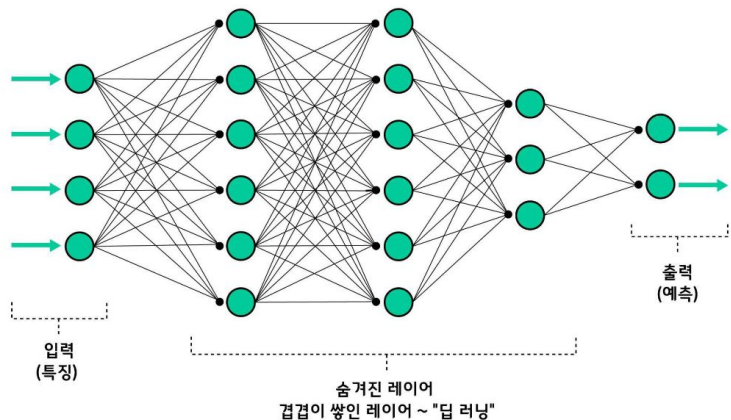
02

활성화 함수, 시그모이드 함수의 특징,  
왜 사용해야 하는가.

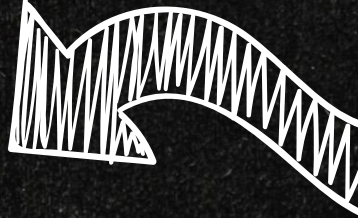
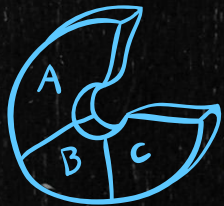


# 활성화 함수(activation function)

딥 러닝 (Deep Learning) - 인공 신경망의 구조

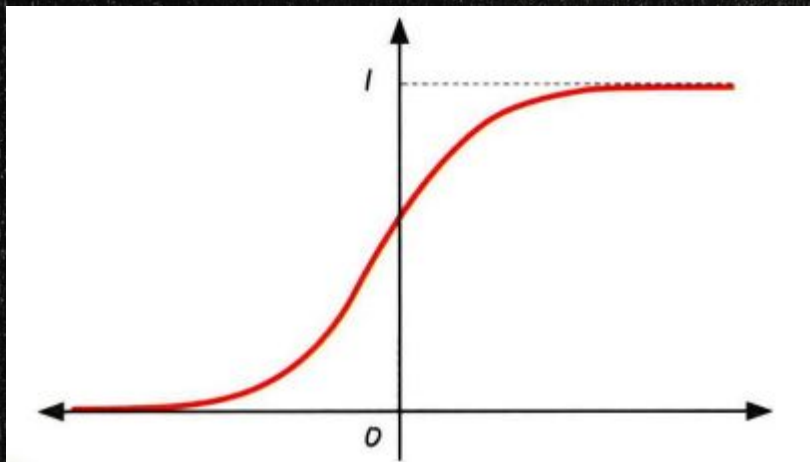


활성화 함수의 개념은 뉴런을 활성화할지 여부를 결정하는 것에서 비롯된 것이므로 활성화 함수는 신경망을 **비선형**으로 만드는 데 중요한 역할을 합니다.



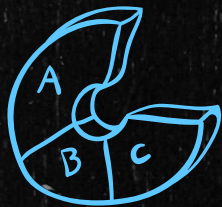
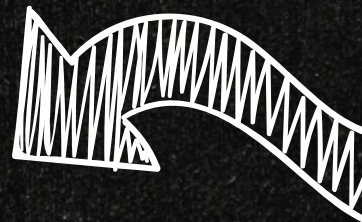


# 시그모이드 함수(sigmoid function)

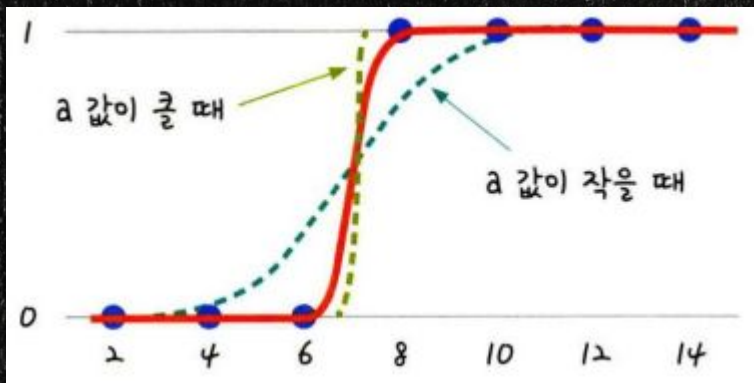


보통 자연 상수  $e$ 가 지수함수에 포함되어  
분모에 들어가면 시그모이드 함수가 된다.

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

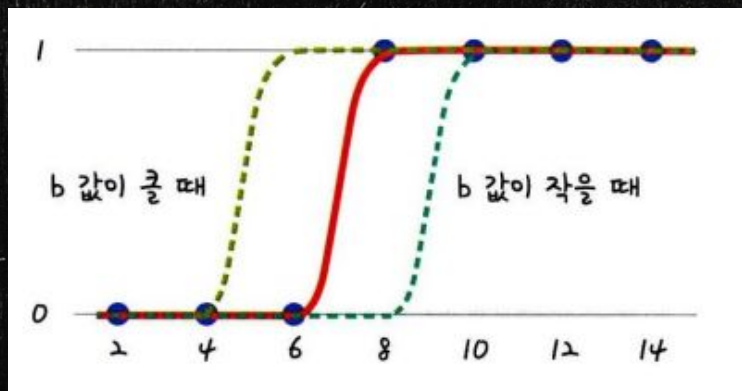
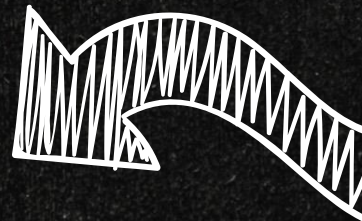


# 시그모이드 함수(sigmoid function)의 특징



$$y = \frac{1}{1 + e^{-(ax+b)}}$$

$a \rightarrow$  그래프의 경사도.



$b \rightarrow$  그래프의 좌우 이동.



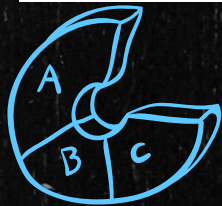


# 시그모이드 함수(sigmoid function)는 왜 사용할까?

$$\text{Odds}(p) := \frac{p}{1-p}$$

$$\log(\text{odds}(p)) = ax + b$$

$$p(x) = \frac{1}{1 + e^{-(ax+b)}}$$



로지스틱 회귀분석에서는 종속변수가 0 or 1  
따라서  $y = ax + b$ 를 이용해서 예측은 의미 X

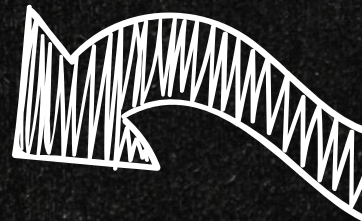
따라서 Odds를 이용

확률  $p$ 가 주어질 경우  $\text{Odds}(p)$ 의 범위  $(0, \infty)$

$\text{Odds}(p)$ 에 로그를 취하면 범위가  $(-\infty, \infty)$

따라서 이 값에 선형회귀를 하는 것은 의미가 있음.

$\log(\text{Odds}(p))$ 를 선형회귀하여  $ax+b$ 값을 얻고  
 $p(x)$ 와 같은 시그모이드 함수를 얻을 수 있다





# 오차함수

오차 함수의 의미와 종류, 의의, 활용성

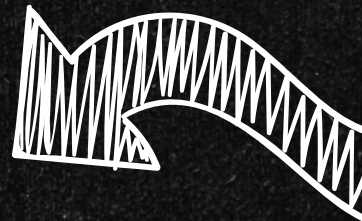
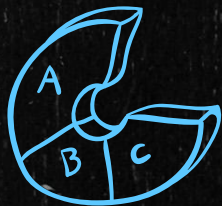




# 1. 오차함수의 의미



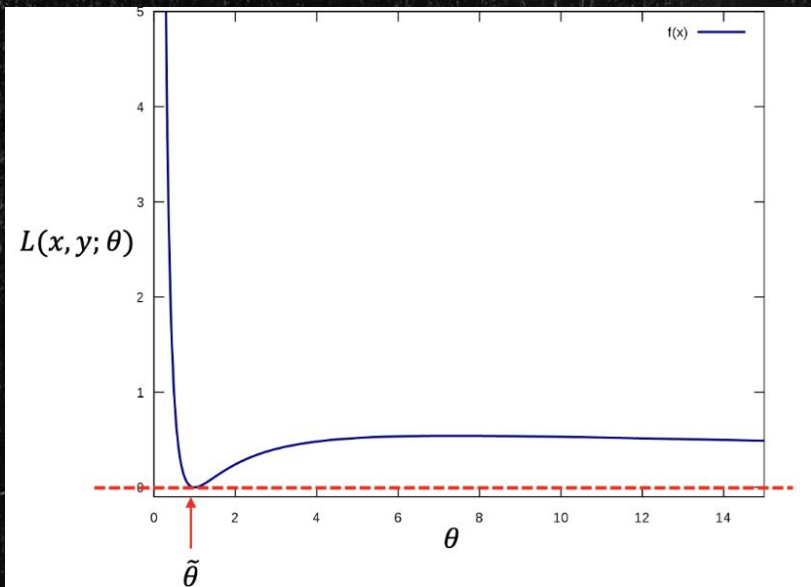
- 지도학습(Supervised Learning) 시 알고리즘이 예측한 값과 실제 정답의 차이를 비교하기 위한 함수
- 최적화(Optimization)를 위해 최소화하는 것이 목적인 함수



# 1. 오차함수의 의미



- 수학적 의미: 알고리즘 학습 시 사용되는 최적의 파라미터를 찾아 오차를 최소화하는 과정



$$\tilde{\theta} = \arg \min_{\theta} L(x, y; \theta)$$

L: 손실함수

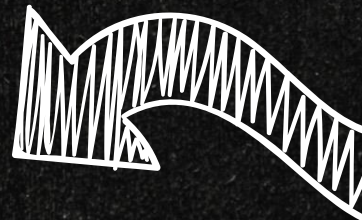
arg min: 목적함수를 최소화하는 입력값을 찾는 역할

x: 학습 데이터의 입력값으로, x로 얻어낸 예측값( $\hat{y}$ )과 정답(y)을 비교

y: 학습 데이터의 정답

$\theta$ : 알고리즘 학습 시 사용되는 모든 파라미터의 벡터

$\tilde{\theta}$ : 추정된 최적의 파라미터





## 2. 오차함수의 종류

손실함수  
(Loss Function)

회귀  
(Regression)

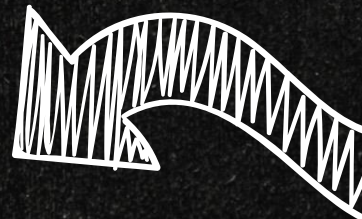
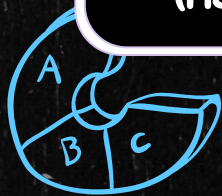
분류  
(Classification)

평균제곱오차  
(MSE)

평균절대오차  
(MAE)

교차 엔트로피  
(Cross-Entropy)

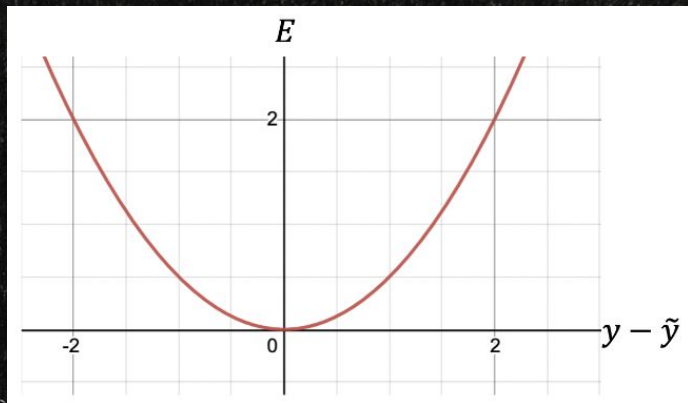
경첩  
(Hinge)



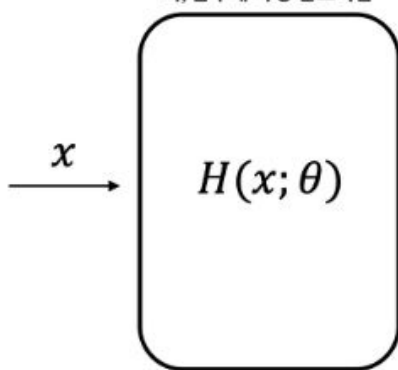
## 2. 오차함수의 종류

- 평균제곱오차(MSE): 오차를 제곱한 값의 평균

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - t_i)^2$$



키, 몸무게 측정 알고리즘



$$y = [40.5, 21.3]$$

$$\widetilde{y}_1 = [39.2, 19.7] \quad E_1 = (40.5 - 39.2)^2 + (21.3 - 19.7)^2 = 4.25$$

↓ Update

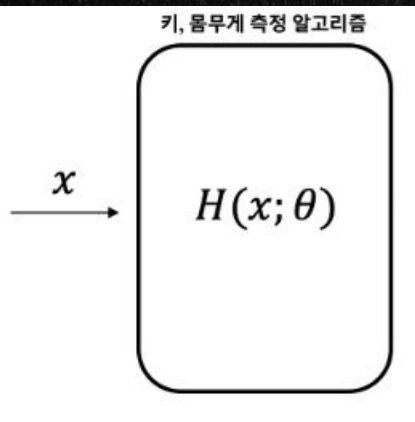
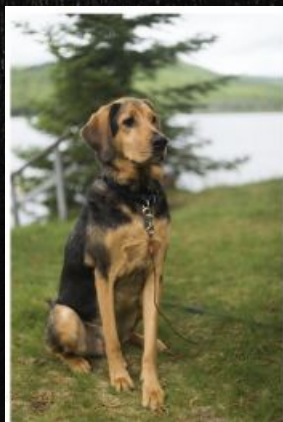
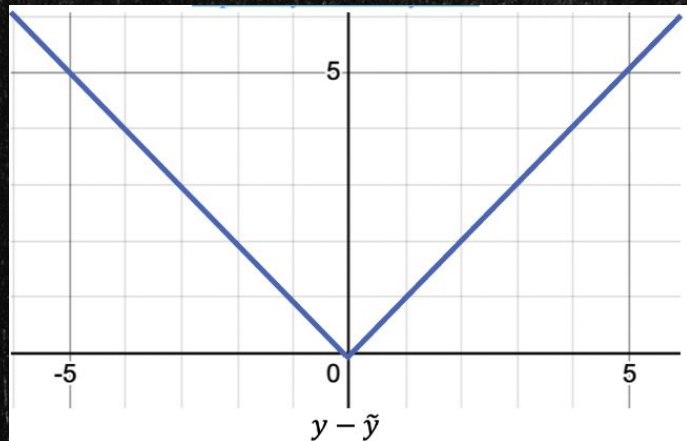
$$\widetilde{y}_2 = [40.1, 20.9] \quad E_2 = (40.5 - 39.5)^2 + (21.3 - 19.9)^2 = 1.16$$



## 2. 오차함수의 종류

- 평균절대오차(MAE): 모든 절대오차의 평균

$$\text{MAE} = \frac{1}{n} \sum |\hat{y} - y|$$



$$y = [40.5, 21.3]$$

$$\tilde{y}_1 = [39.2, 19.7]$$

↓ Update

$$\tilde{y}_2 = [40.1, 20.9]$$

$$E_1 = (40.5 - 39.2)^2 + (21.3 - 19.7)^2 = 4.25$$

$$E_2 = (40.5 - 39.5)^2 + (21.3 - 19.9)^2 = 1.16$$

## 2. 오차함수의 종류

- **교차 엔트로피(Cross-Entropy):** 실제 분포에 대해서 알지 못하는 상태에서 모델링을 통해 구한 분포를 통해 실제분포를 예측

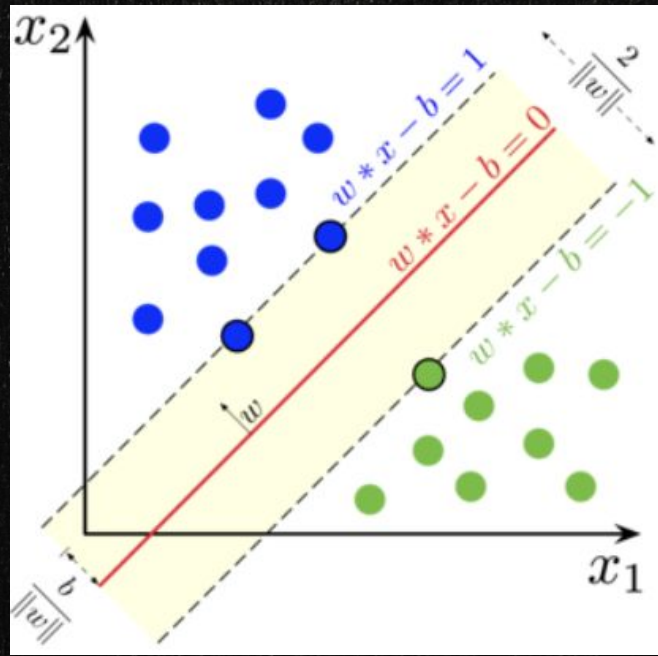
$$H(p, q) = - \sum_{k=1}^K p(X = k) \log q(X = k)$$

- 범주형 교차 엔트로피: 일반적인 경우
- 이항 교차 엔트로피: 두개의 클래스 중 예측하는 경우

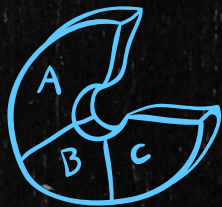
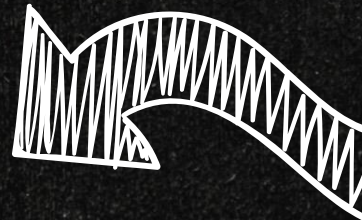
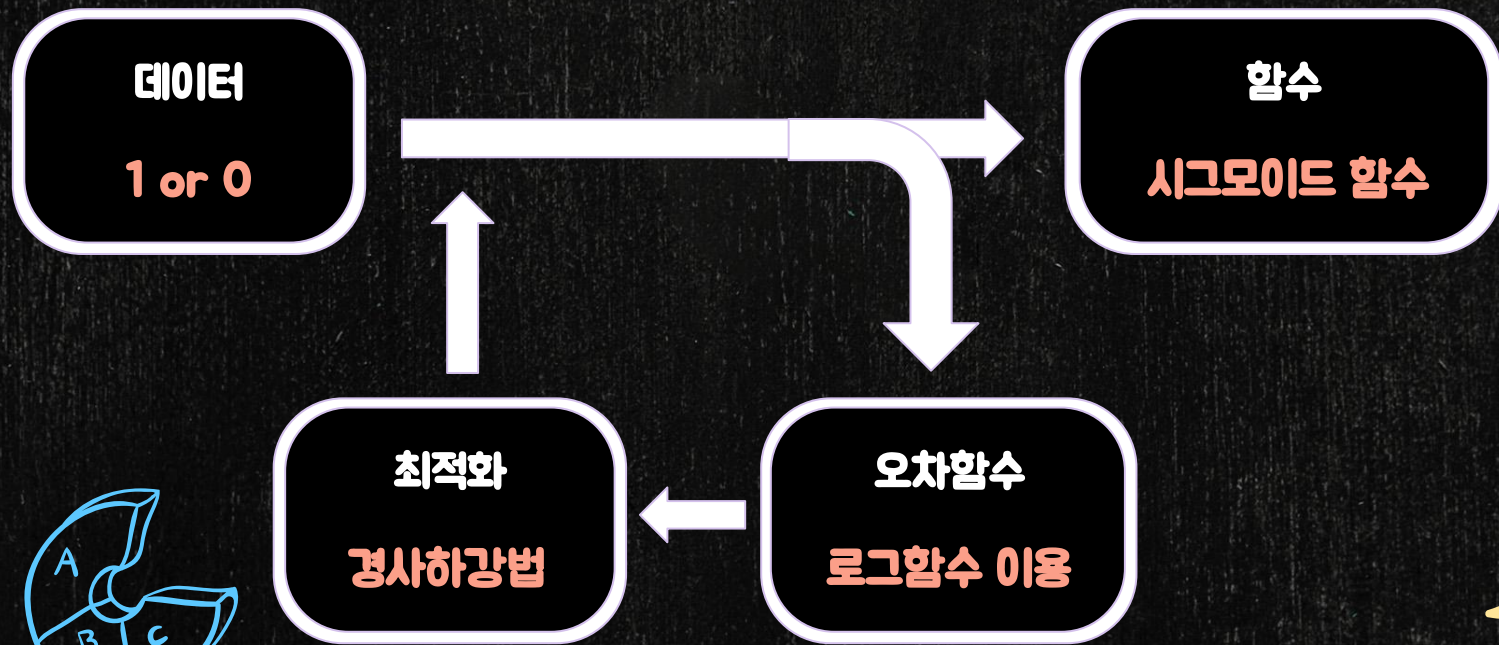


## 2. 오차함수의 종류

- 경첩(Hinge): 통계적 분류 훈련에 사용되는 오차함수로 '최대마진' 분류에 사용됨
- 서포트 벡터 머신(SVM): 패턴 인식, 자료 분석을 위한 지도 학습 모델로 label을 가장 잘 분류하는 초평면을 찾는 것을 목표로 하는 모델



# Meaning of loss function

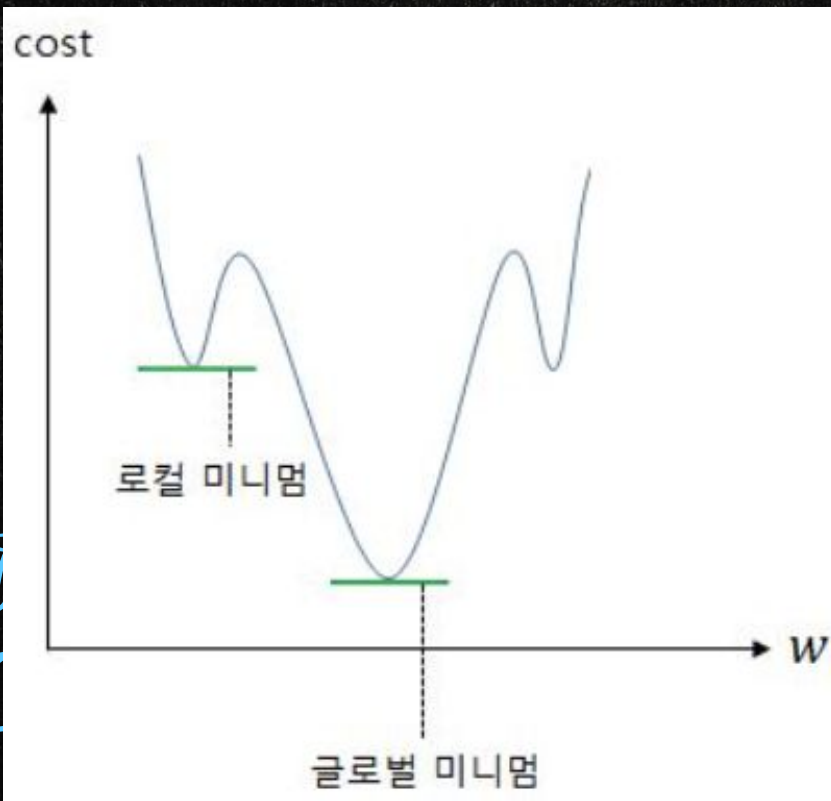




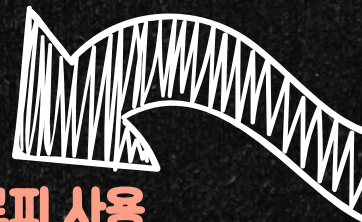
Q. 왜 평균제곱오차를 사용하면 안될까?



A.

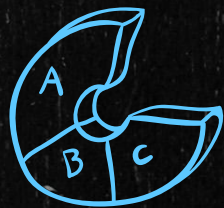
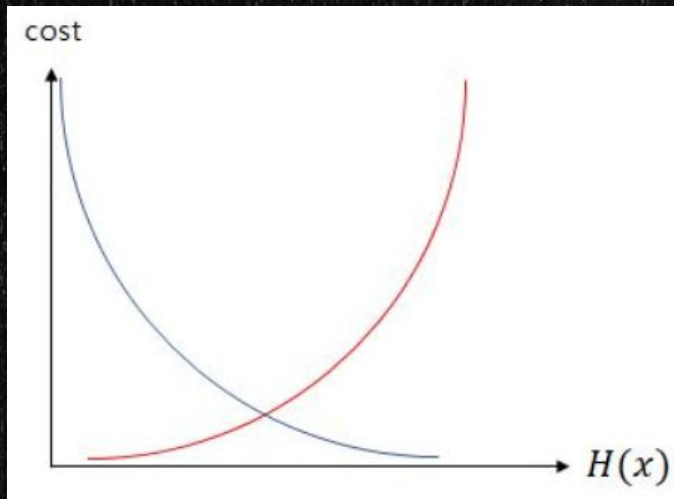


교차 엔트로피 사용



### 3. 오차함수의 의미

- 로그함수를 이용해 오차함수 구하기



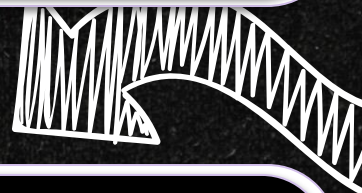
로그함수

$$\text{cost}(H(x), y) = - [y \log H(x) + (1-y) \log (1-H(x))]$$



오차함수

$$-1/n \sum y \log h + (1-y) \log (1-h)$$





# Machine Learning

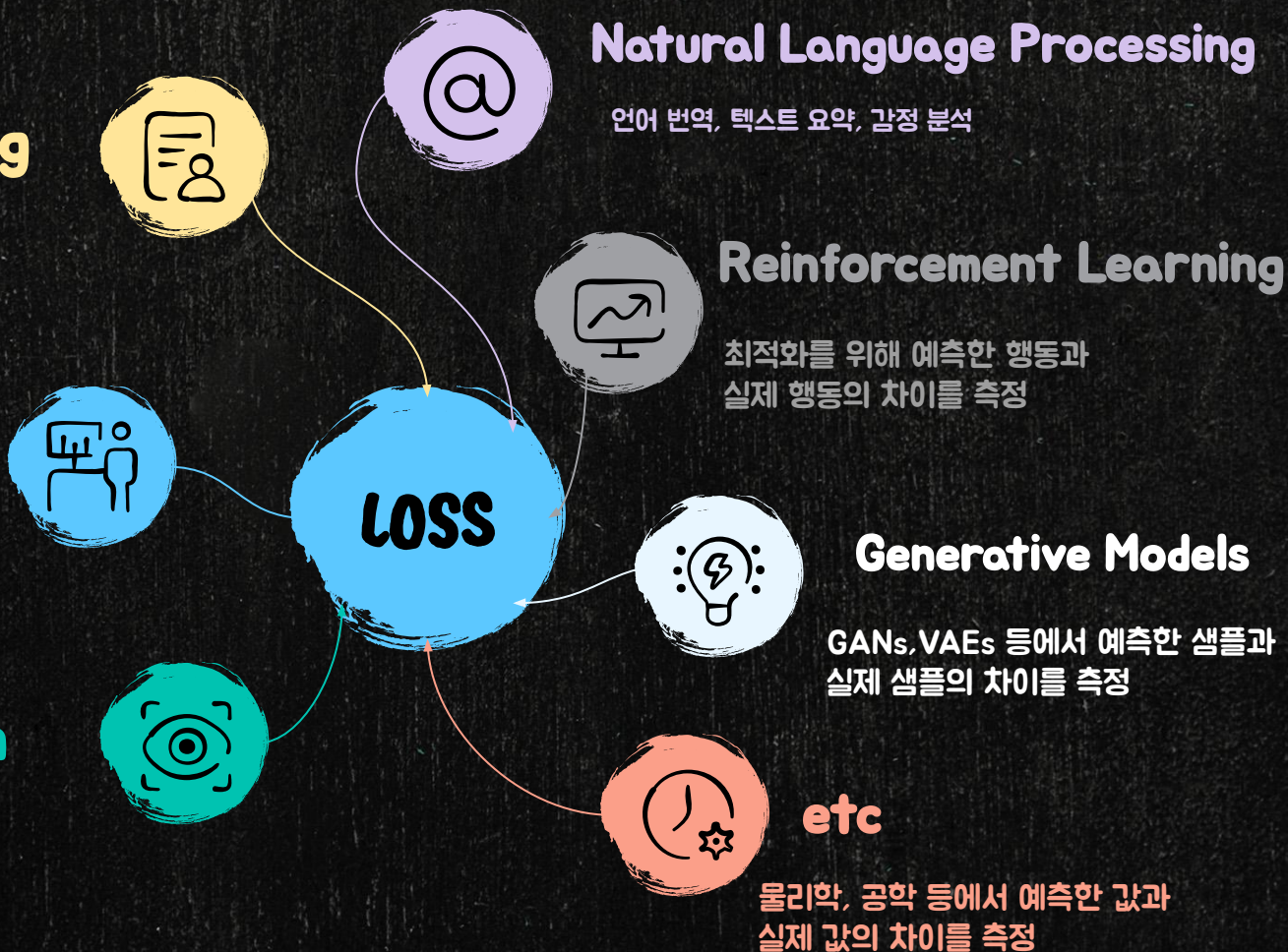
선형회귀, 로지스틱 회귀 등에서  
예측값과 실제값의 차이를 측정

# Optimization

손실함수의 결과값이  
최소가 되도록 최적화

# Computer Vision

이미지 분류, 물체 인식,  
시멘틱 분할 등

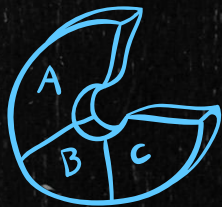

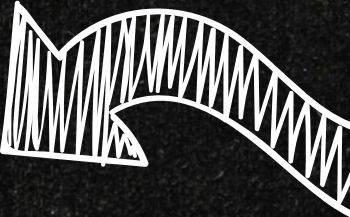





04

# Pytorch

Tensorflow의 경쟁자  
딥러닝 프레임 워크 Pytorch





# DEEP LEARNING FRAMEWORK



**Pytorch**

made by facebook  
more active!!  
use in Journal

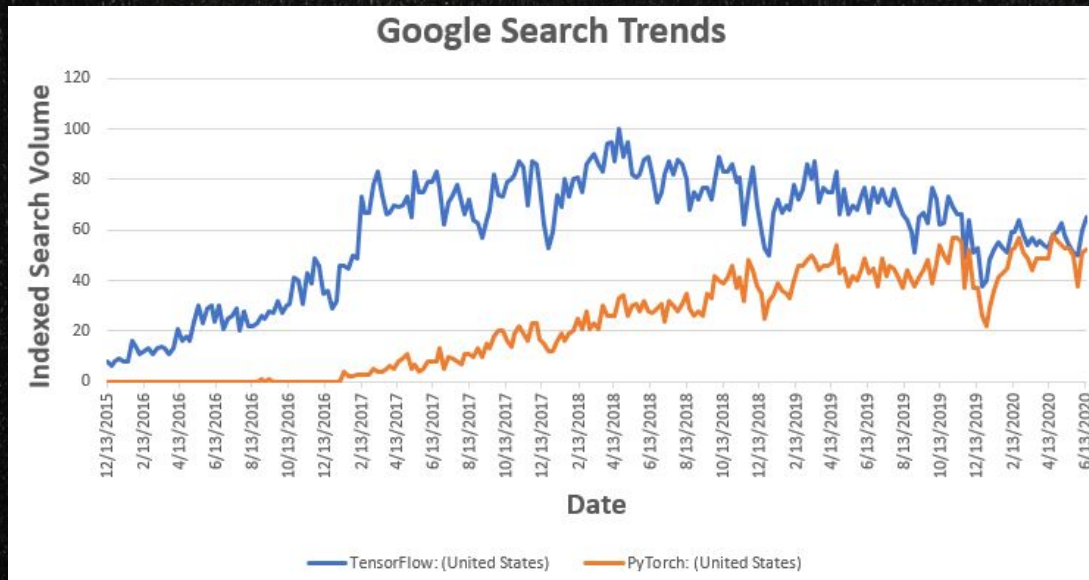


**Tensorflow**

made by Google  
more static..  
use in Industry

**pytorch is more pythonistic!?**

# Trends is...



Tensorflow → Pytorch



# Code

```
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(in_channels=3, out_channels=64, kernel_size=5),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2)
        )

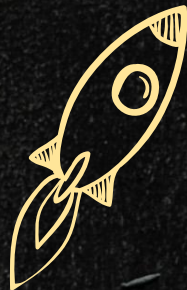
        self.layer2 = nn.Sequential(
            nn.Conv2d(in_channels=64, out_channels=30, kernel_size=5),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2)
        )

        self.layer3 = nn.Sequential(
            nn.Linear(in_features=30*5*5, out_features=10, bias=True),
            nn.ReLU(inplace=True),
        )

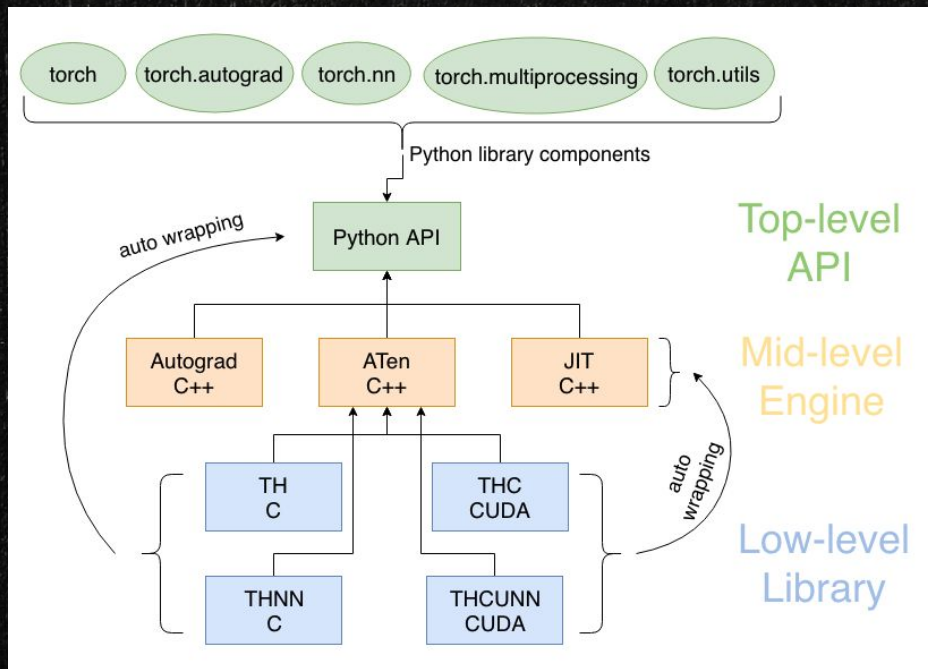
    def forward(self, x):
        x = self.layer1(x)
        x = self.layer2(x)
        x = x.view(x.shape[0], -1) # Flatten
        x = self.layer3(x)
        return x
```

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

## Which code is Pytorch ??



# overview



autograd, nn, multiprocessing, utils



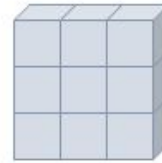
# Tensor



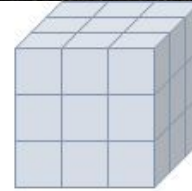
**0D Tensor(Scalar)**  
Rank: 0  
Shape: ()



**1D Tensor(Vector)**  
Rank: 1  
Shape: (3, )



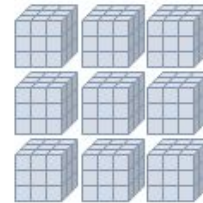
**2D Tensor(Matrix)**  
Rank: 2  
Shape: (3, 3)



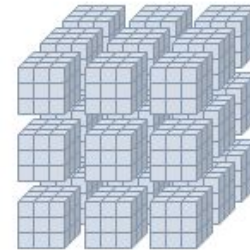
**3D Tensor**  
Rank: 3  
Shape: (3, 3, 3)



**4D Tensor**  
Rank: 4  
Shape: (3, 3, 3, 3)



**5D Tensor**  
Rank: 5  
Shape: (3, 3, 3, 3, 3)

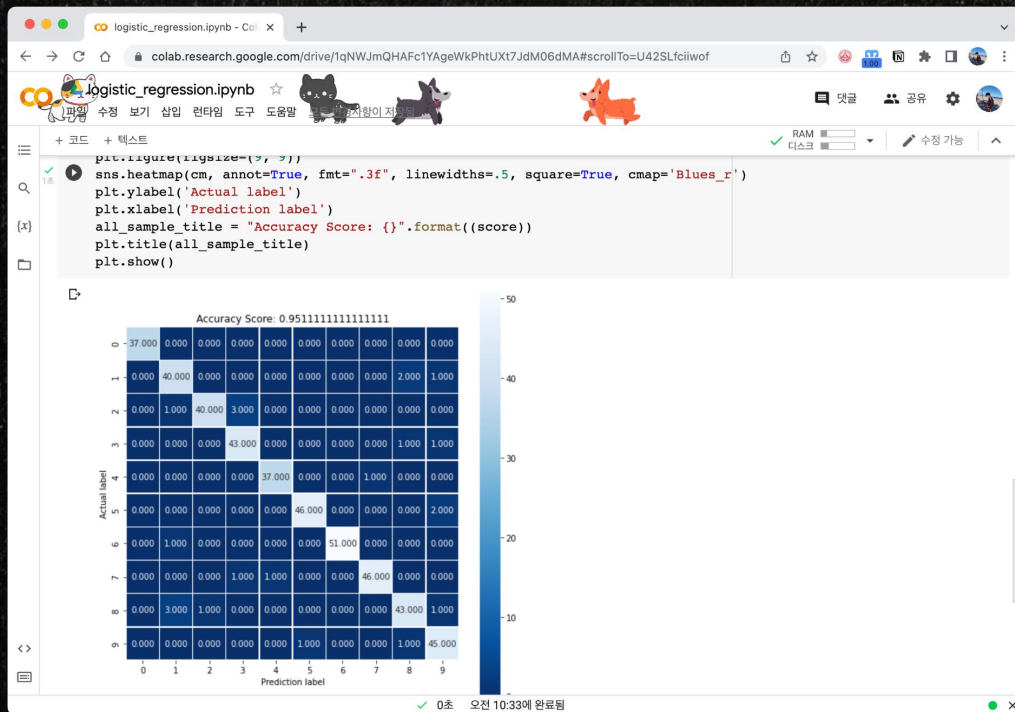


**6D Tensor**  
Rank: 6  
Shape: (3, 3, 3, 3, 3, 3)

**vector -> matrix -> tensor!**



# Logistic Regression using colab



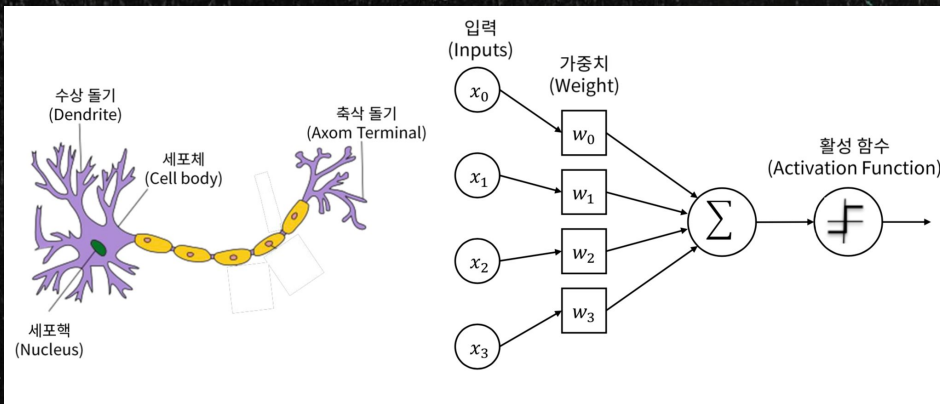
<https://colab.research.google.com/drive/1qNWJmQHAFc1YAgeWkPhtUXt7JdM06dMA?usp=sharing>



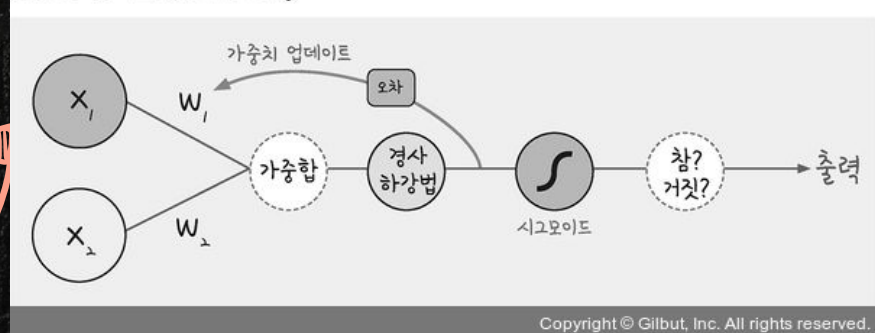
# 예고편

## <퍼셉트론 PERCEPTRON>

다수의 입력으로 하나의 결과를 내보내는 알고리즘



(비교) 로지스틱 회귀 모델





Thank You!  
감사합니다!

