01 인공지능의 정의

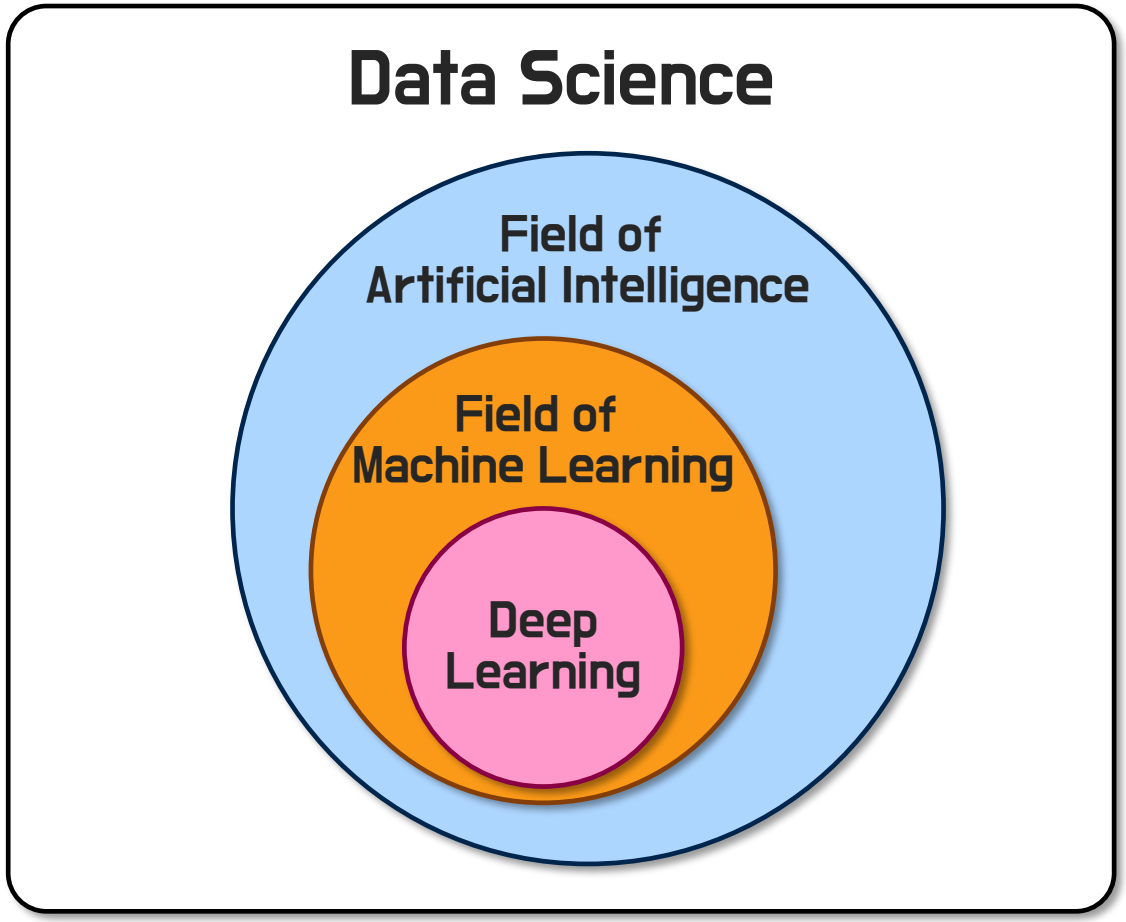
01 인공지능의 정의

반갑다, 인공지능!

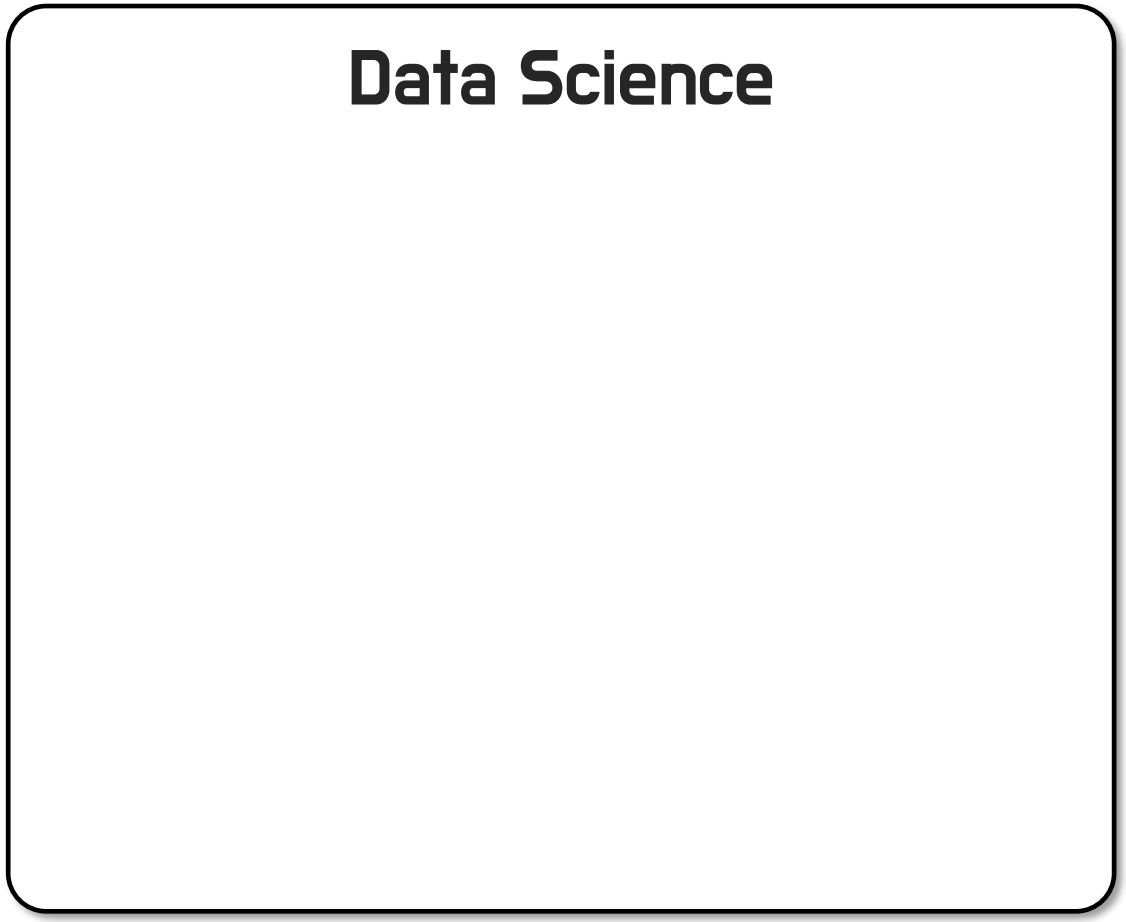
오늘부터 이틀 동안 인공지능에 빠져 봅시다.

그런데, 인공지능이 정확히 뭔데요?

데이터 과학? 인공지능? 머신러닝? 딥러닝?



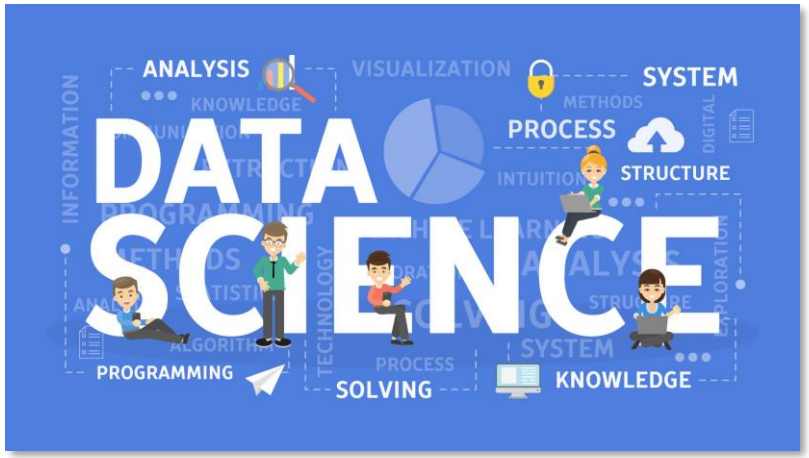
데이터 과학? 인공지능? 머신러닝? 딥러닝?

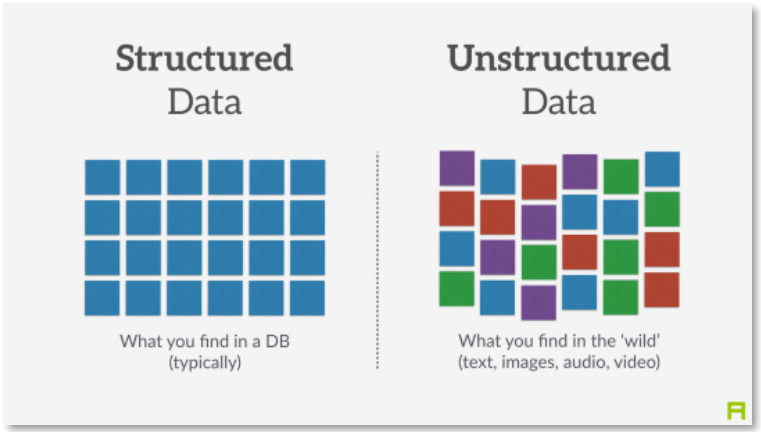


데이터 과학 (Data Science)

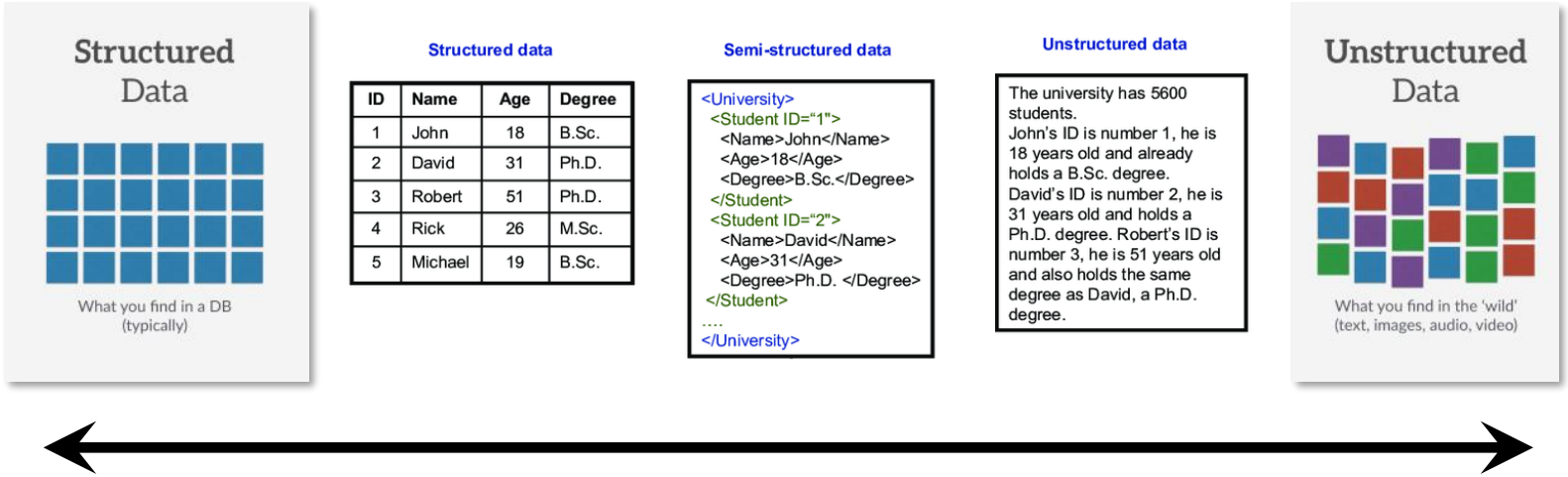
정형 또는 비정형 데이터로부터 지식과 인사이트를 추출하는데

과학적 방법론, 프로세스, 알고리즘, 시스템을 동원하는 융합 분야





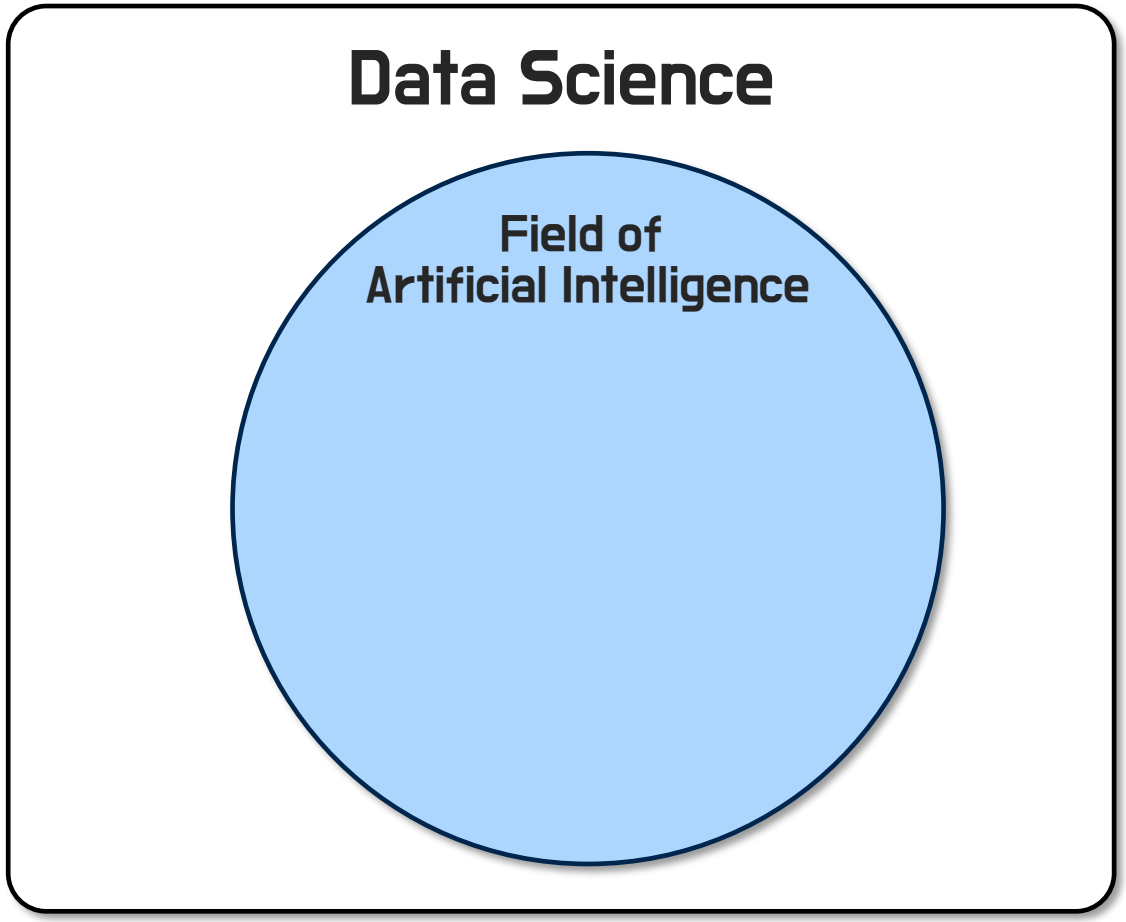
데이터 과학 (Data Science)



[정형 데이터 (Structured Data)]

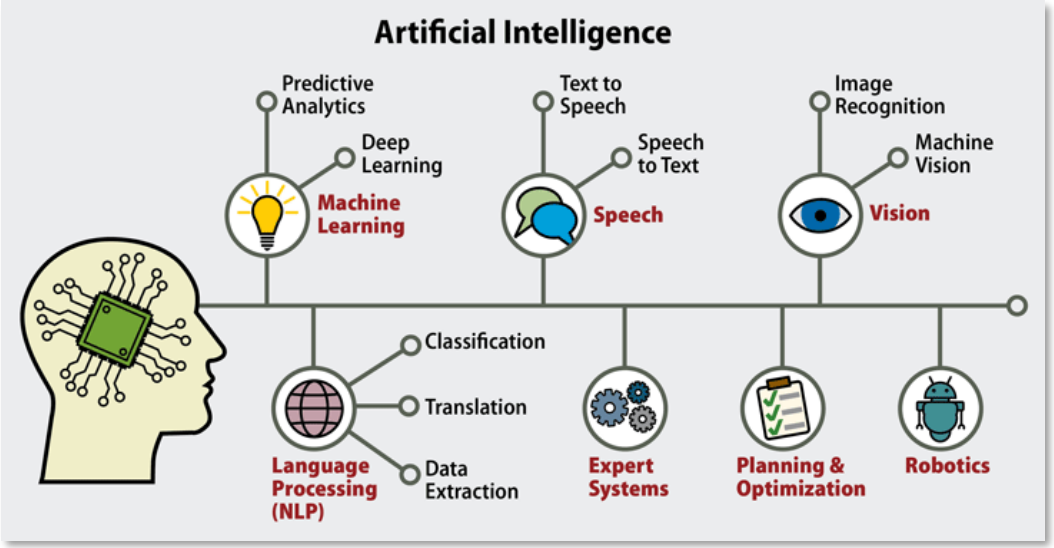
[비정형 데이터] (Unstructured Data)

데이터 과학? 인공지능? 머신러닝? 딥러닝?



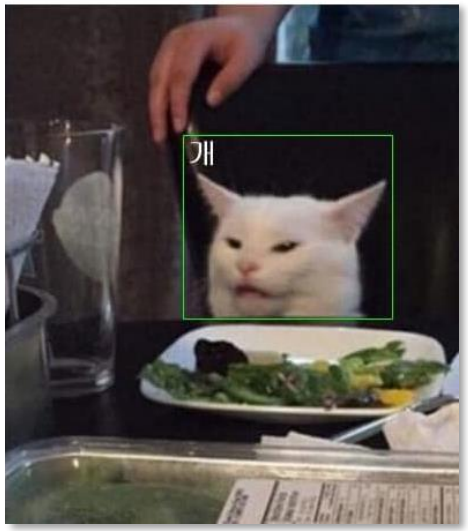
인공지능 (Artificial Intelligence)

- 사람의 지적 행동을 따라 할 수 있는 능력을 인공적으로 구현한 컴퓨터 프로그램 또는 이를 포함한 컴퓨터 시스템
- 학습, 추론, 인식, 계획, 자연어처리, 지각, 개체 조작 능력 문제 등을 다룸

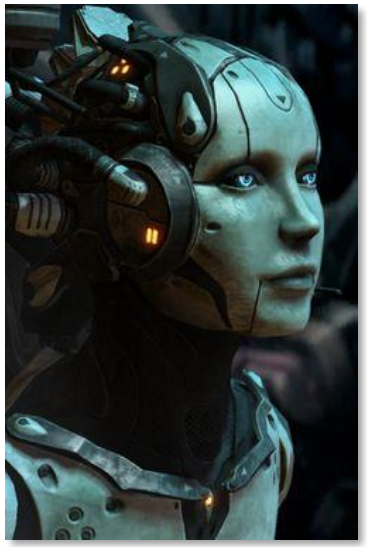


인공지능 (Artificial Intelligence)

약인공지능
(Weak Artificial Intelligence)



강인공지능
(Strong Artificial Intelligence)



초인공지능
(Artificial Super Intelligence)

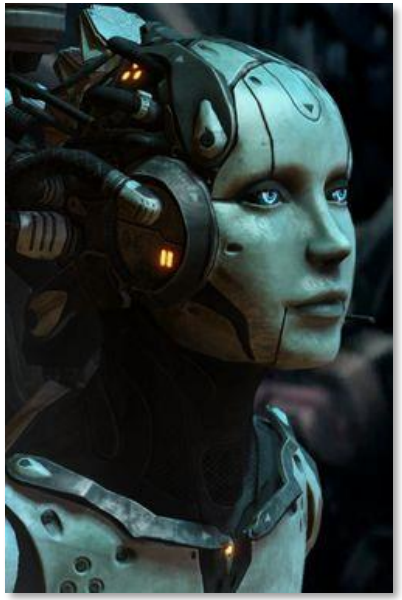
약인공지능 (Weak Artificial Intelligence)

AI는 인간을 멸망시킬 거에요!
심판의 날이 얼마 안 남았다!

실제 AI

- 자의식이 없는 인공지능
- Narrow AI 라고도 불림
- 특정 영역에서만 활용 가능 (음성인식, 자연어처리 등)
- 알고리즘, 기초 데이터, 규칙 입력 필요
- 특정 분야에서는 사람을 넘어서는 인공지능 등장

강인공지능 (Strong Artificial Intelligence)



<Starcraft 2의 부관>

<Iron Man의 Jarvis>

- 사람과 똑같이 스스로 학습하여 똑같이 행동하는 AI
- 자의식이 있는 인공지능
- AI가 스스로 데이터를 찾아서 학습이 가능한 상태
- 강인공지능은 현실적인 어려움으로 인해 현재까지는 큰 성과가 없는 상태

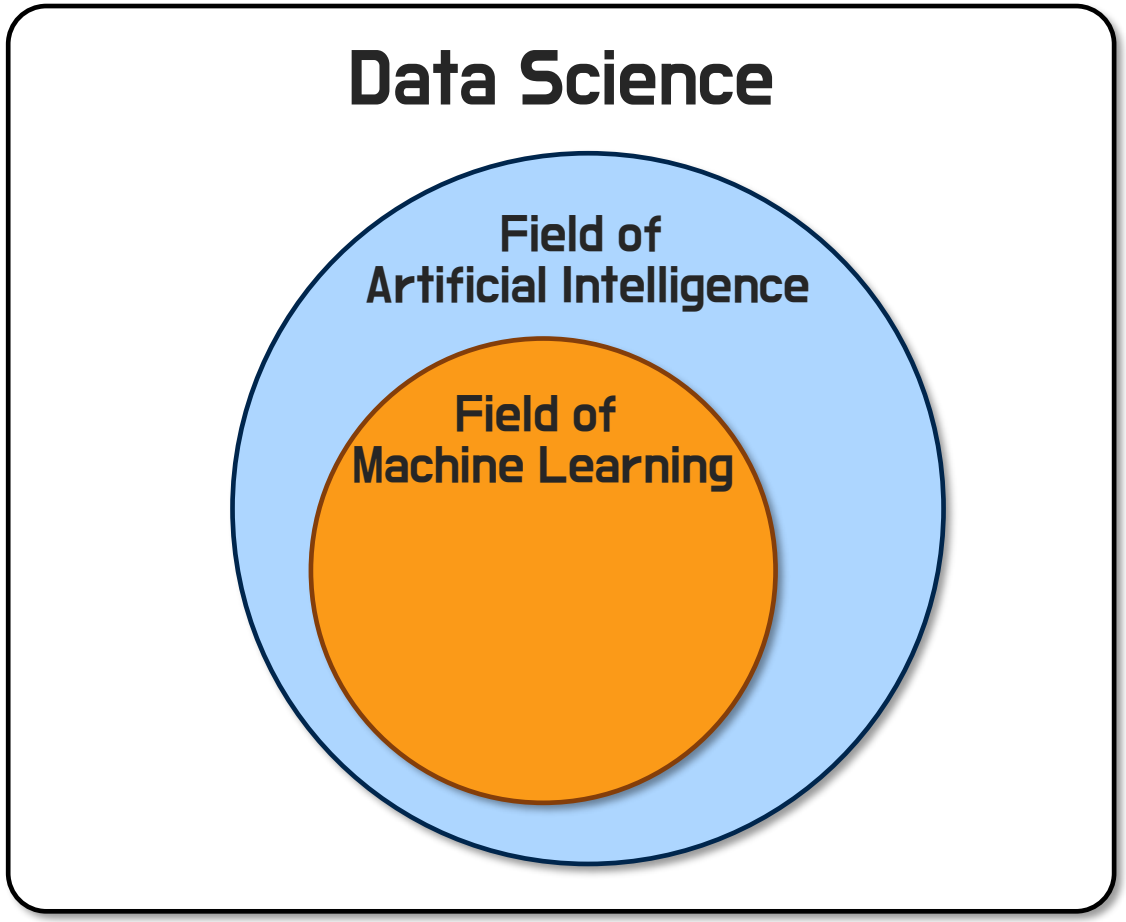
초인공지능 (Artificial Super Intelligence)

<Terminator의 Skynet>

<Matrix의 AI>

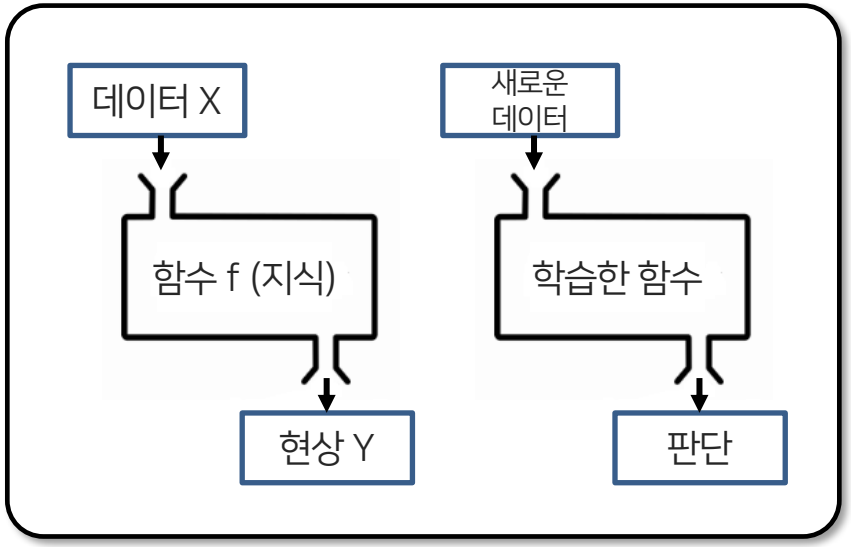
- 인간의 지식을 1000배 이상 초월하고 모든 면에서 월등한 인공지능
- 미래에 등장할 것이라고 예측하는 학자들이 많음
- 스티븐 호킹: “초인공지능의 출현이 인류의 종말로 이어질 것”
- 레이 커즈와일(미래학자): “현재의 인공지능 발전 속도를 감안할 때 2030년에 인공지능은 특이점에 다다를 것이며,이 특이점을 뛰어넘으면 AI 스스로 자신보다 더 똑똑한 AI를 만들어 지능이 무한히 높은 존재가 출현하게 될 것”

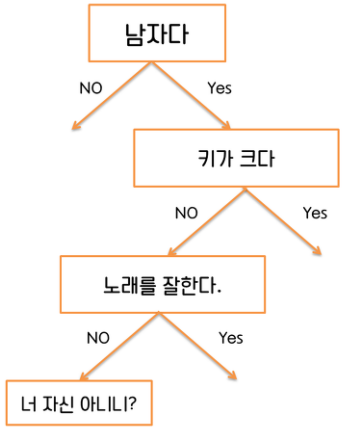
머신러닝 (Machine Learning)



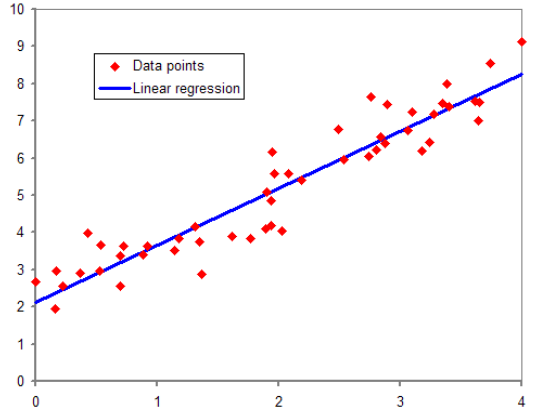
머신러닝 (Machine Learning)

- 학습 과정을 통해 인간의 지적 능력을 모방할 수 있도록 만드는 알고리즘과 기술을 개발하는 분야
- 데이터와 데이터를 설명하는 현상이 필요





의사결정나무
(Decision Tree)



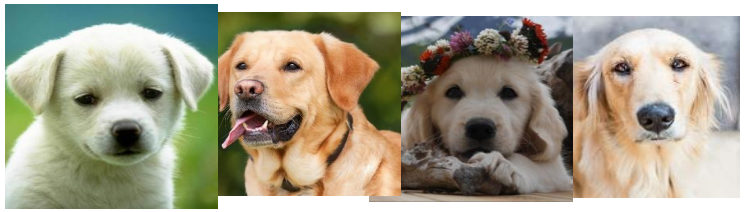
선형회귀
(Linear Regression)

머신러닝 (Machine Learning)

지도학습 (Supervised learning): 정답이 있는 데이터 학습

Cat

Dog

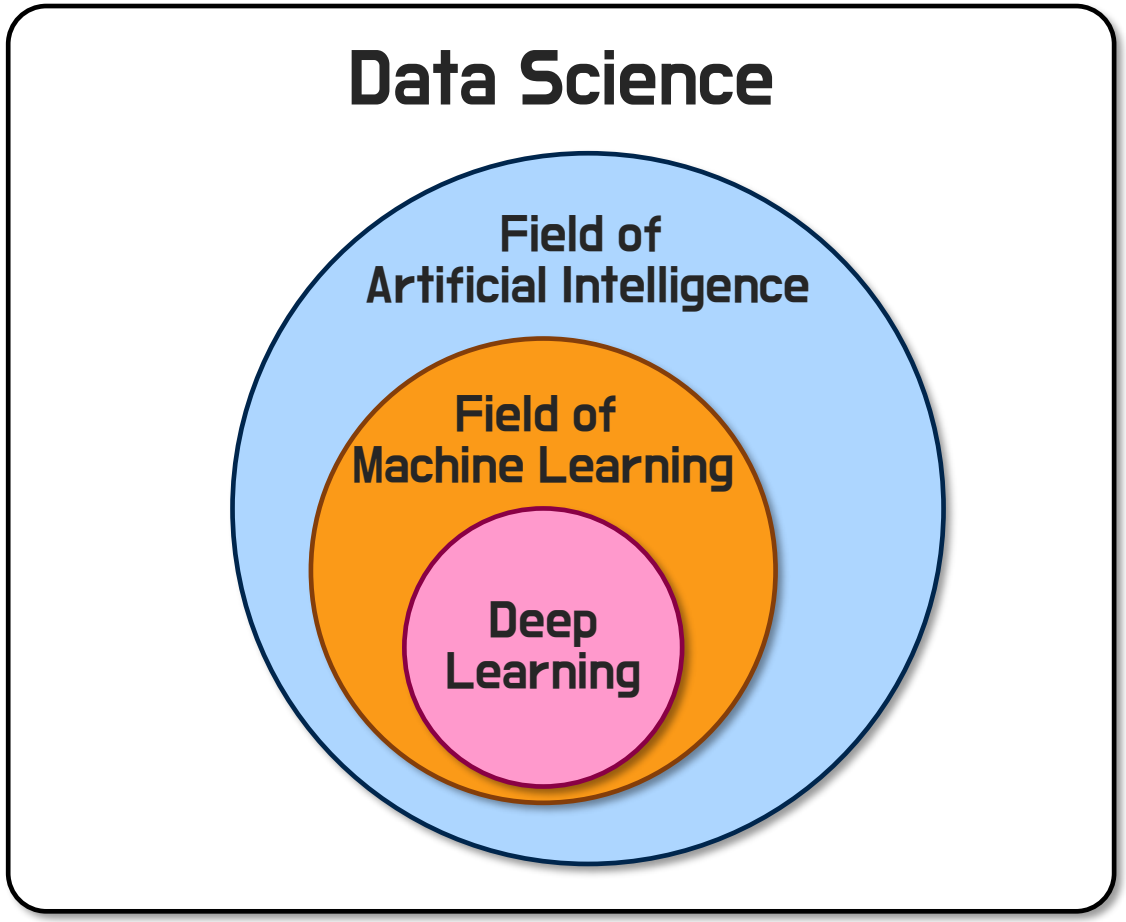


비지도학습 (Unsupervised learning): 정답이 없는 데이터 학습

클러스터링

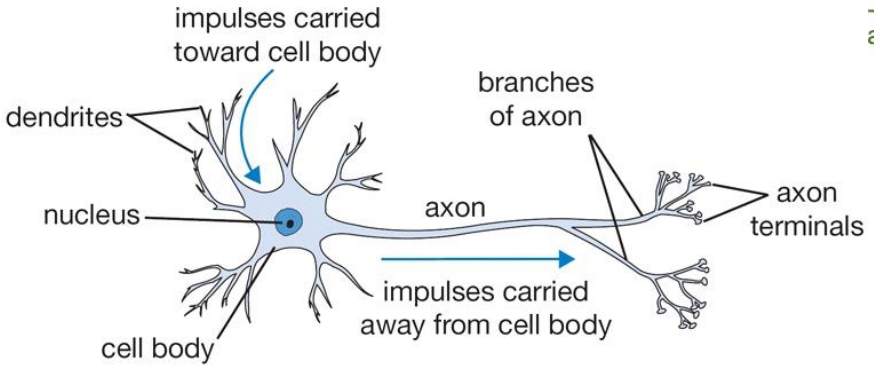
그림에 구멍뚫고 복원하기

딥러닝 (Deep Learning)

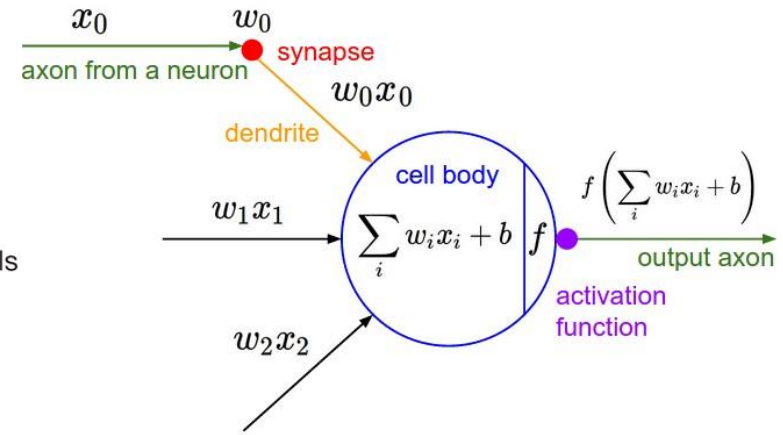


딥러닝 (Deep Learning)

사람의 신경망을 모사한 인공 신경망 (Artificial neural network)에 기반하는 머신러닝 알고리즘의 한 종류



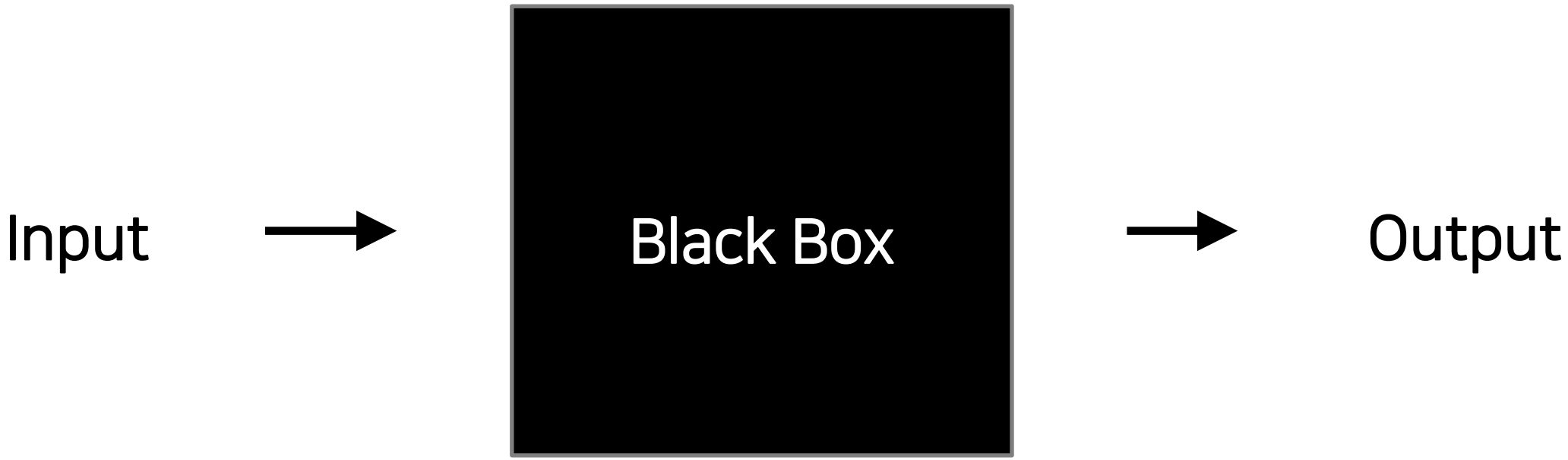
<Neuron>



<Perceptron>

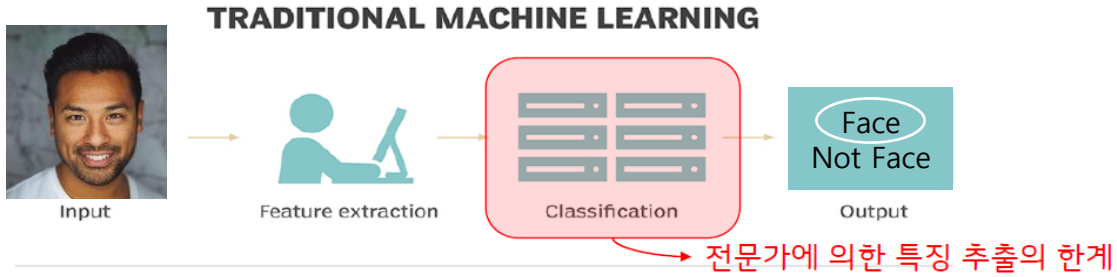
딥러닝 특징

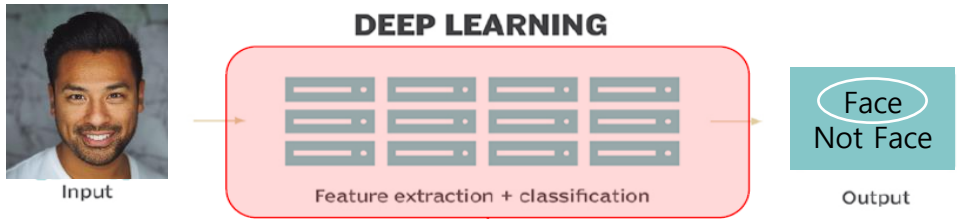
블랙박스 알고리즘?



딥러닝 특징

기존 기계학습: 특징 추출을 위해 사람이 직접 데이터를 가공해야했음
딥러닝: 별도 데이터 가공 없이 오직 데이터에 기반하여 End-to-End로 학습 가능



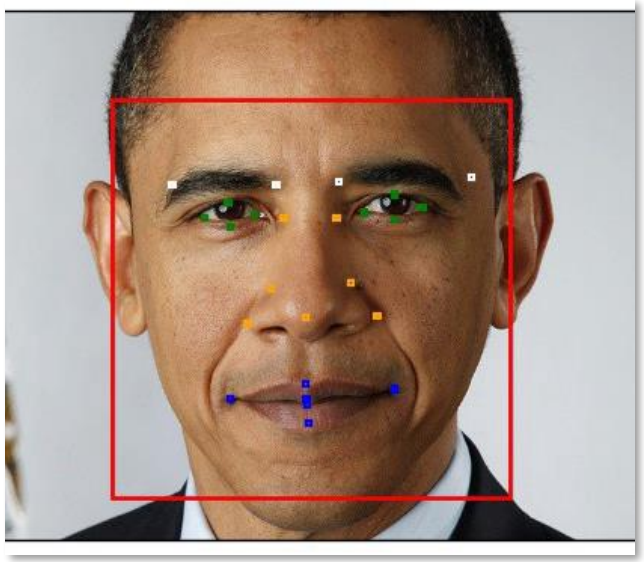


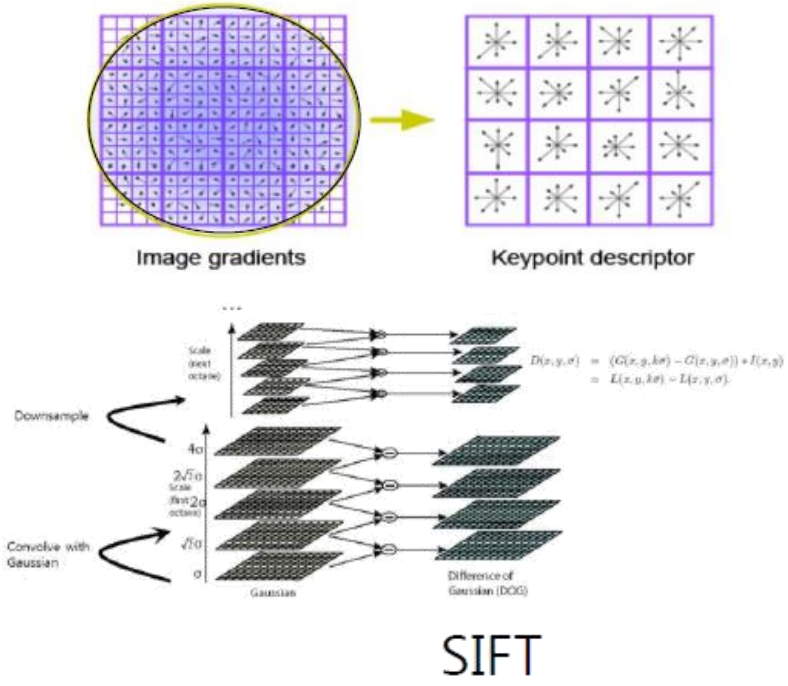
기계가 모든 특징을 추출
정보를 구분할 수 있는 최소단위 입력만으로 학습
(영상 픽셀, 음성 신호 등)

딥러닝 특징

[특징 추출 단계를 대체하는 딥러닝]

- 기존의 머신러닝 기법들은 직접 추출한 (Hand-crafted features) feature에 기반



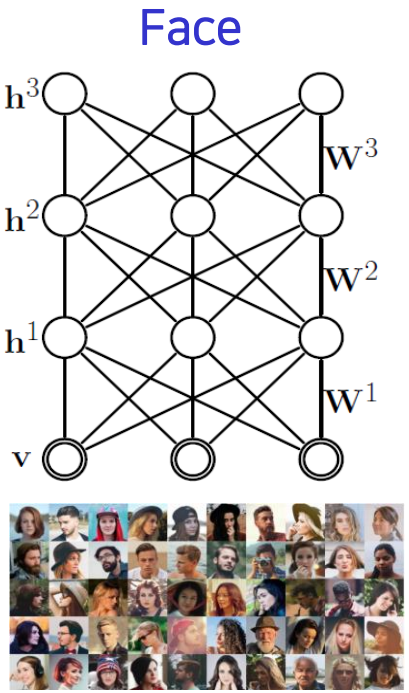


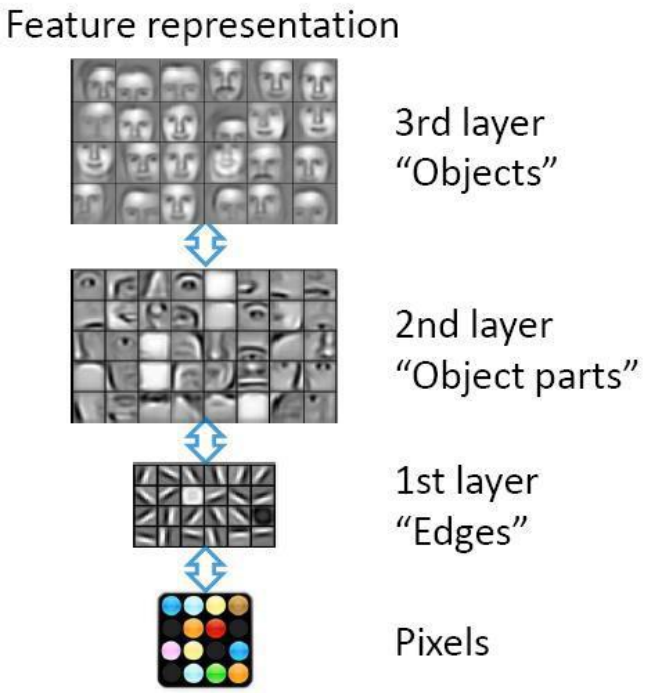
SIFT

딥러닝 특징

[특징 추출 단계를 대체하는 딥러닝]

- 딥러닝은 데이터를 기반으로 자동적으로 특징 표현을 만들어 냄



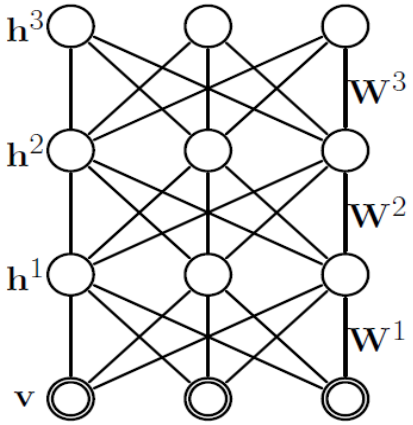


딥러닝 특징

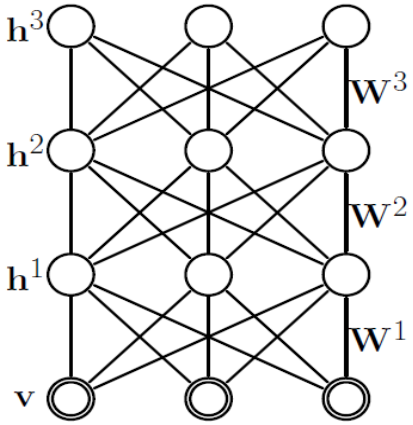
[특징 추출 단계를 대체하는 딥러닝]

- 딥러닝은 데이터를 기반으로 자동적으로 특징 표현을 만들어 냄

Not face!

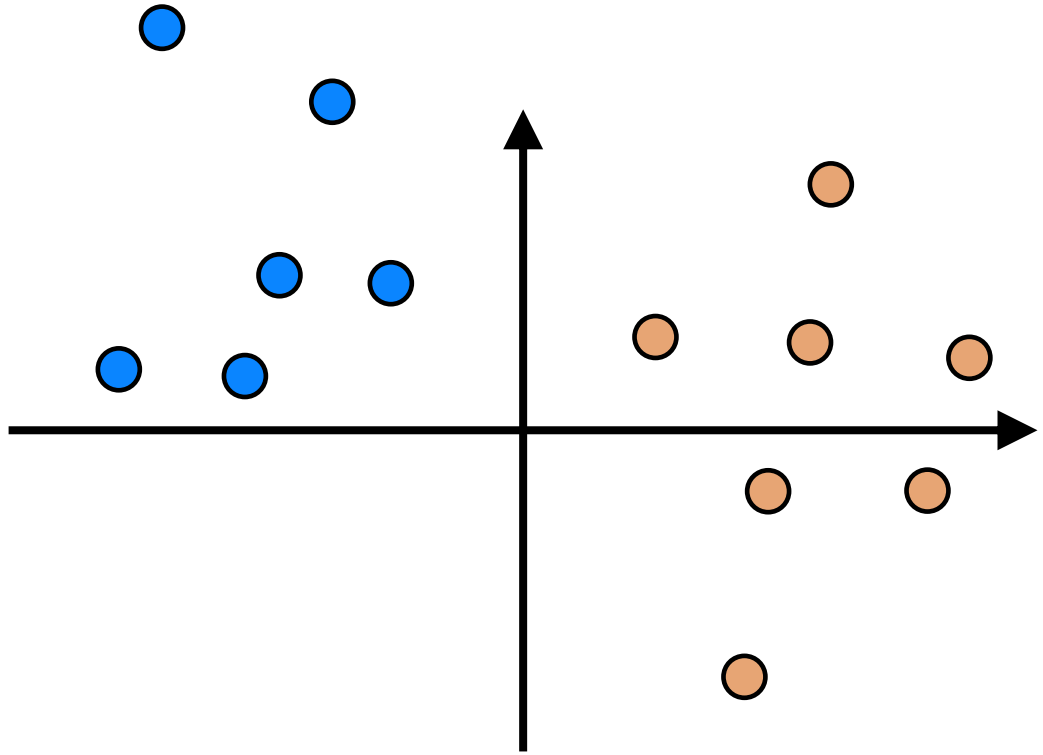


Face!



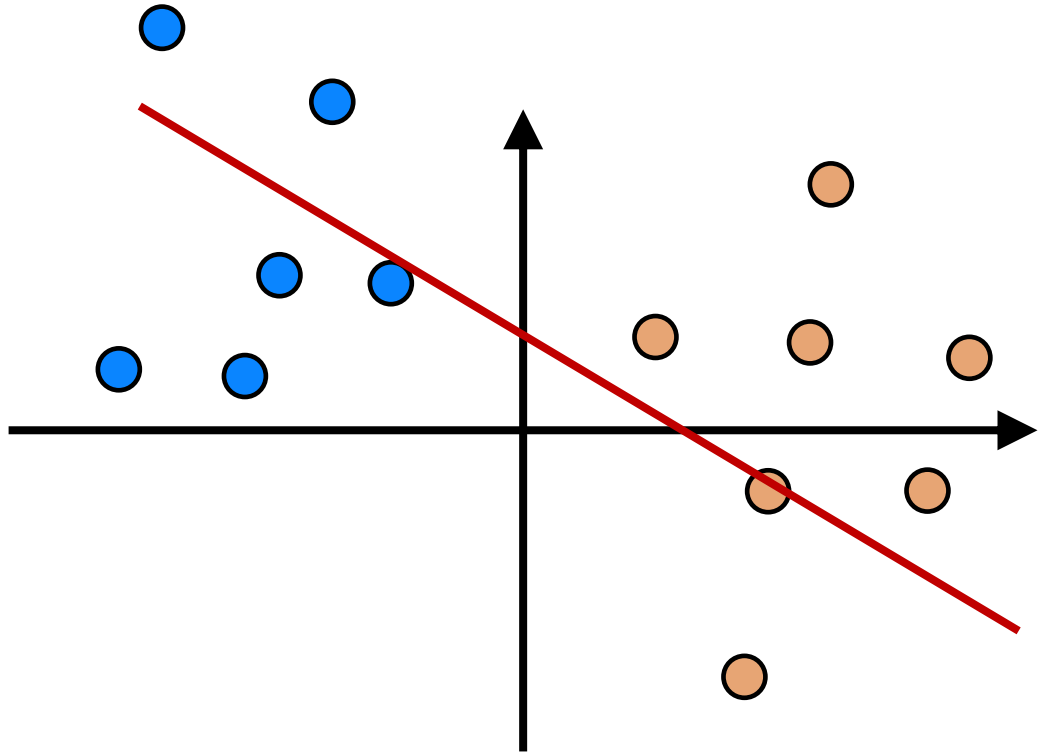
학습이란?

두 색깔이 다른 원들을 가르는 선을 그리기



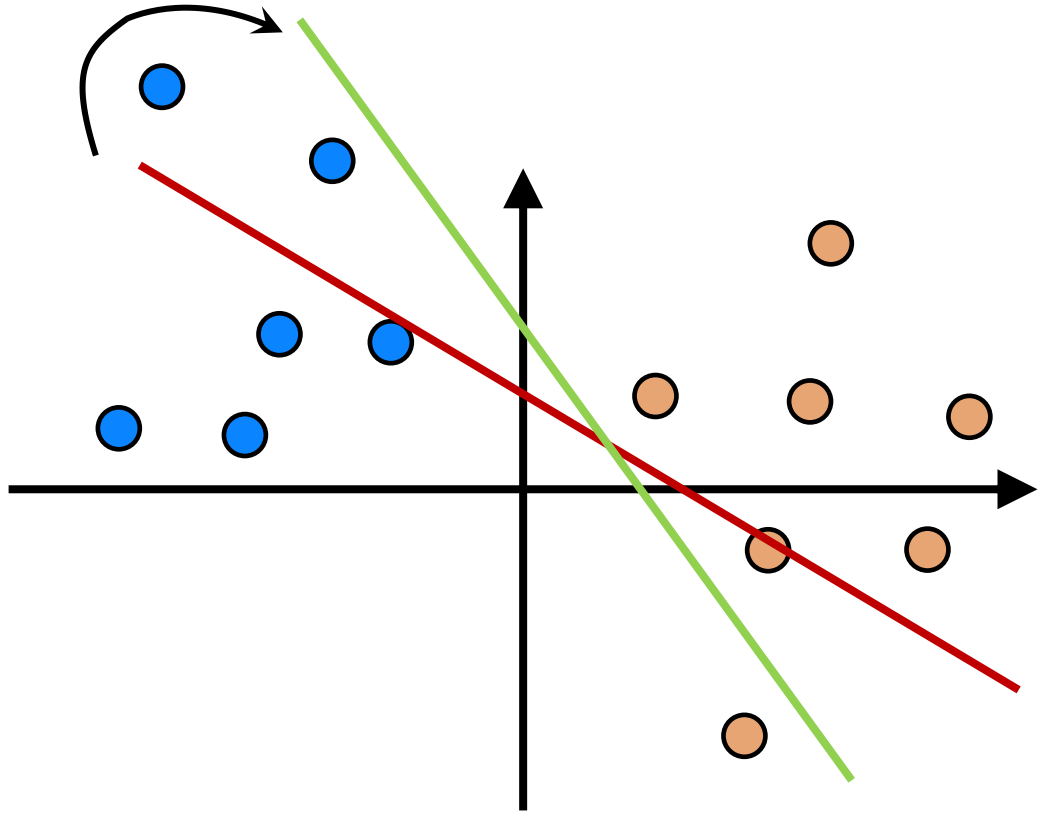
학습이란?

정확히 구분 실패



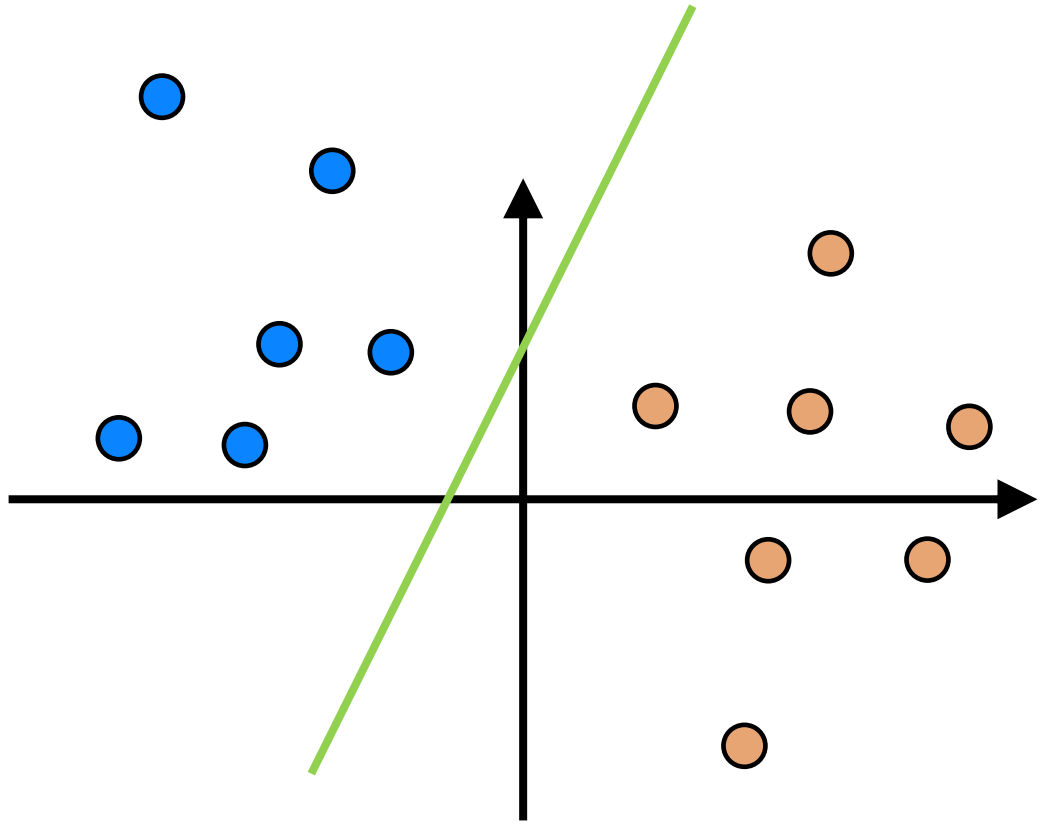
학습이란?

수정: 시계 또는 반시계 방향으로 움직이기



학습이란?

성공!



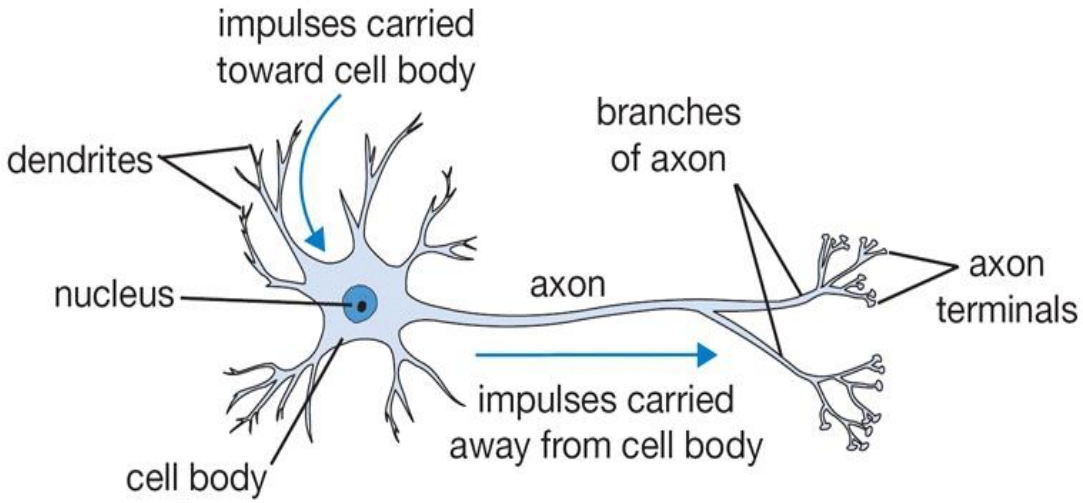
학습이란?

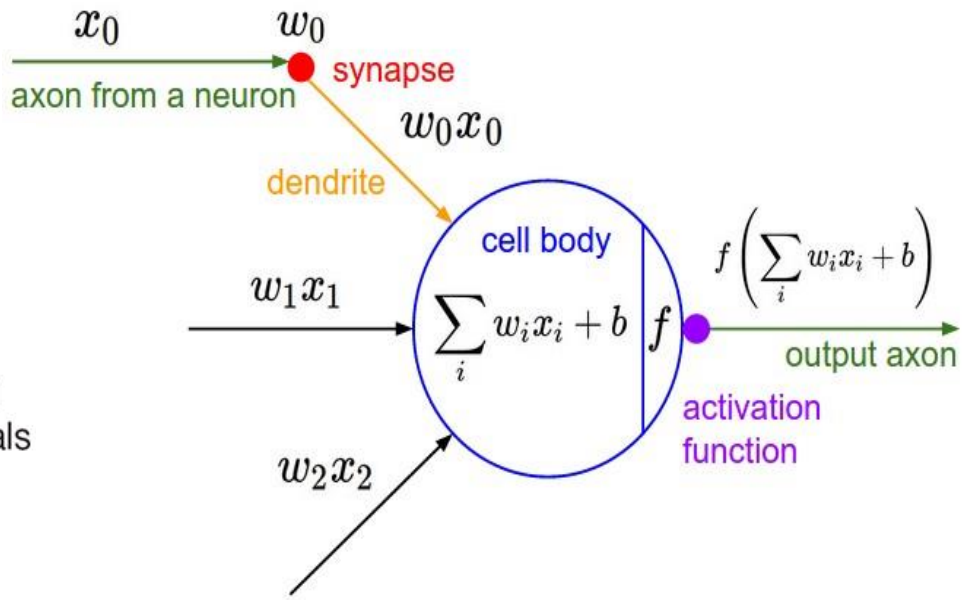
① 선 그리기 → Feedforward
(순전파, Forward propagation)

② 틀린 부분찾기 → Loss and Gradient
(역전파, Backward propagation)

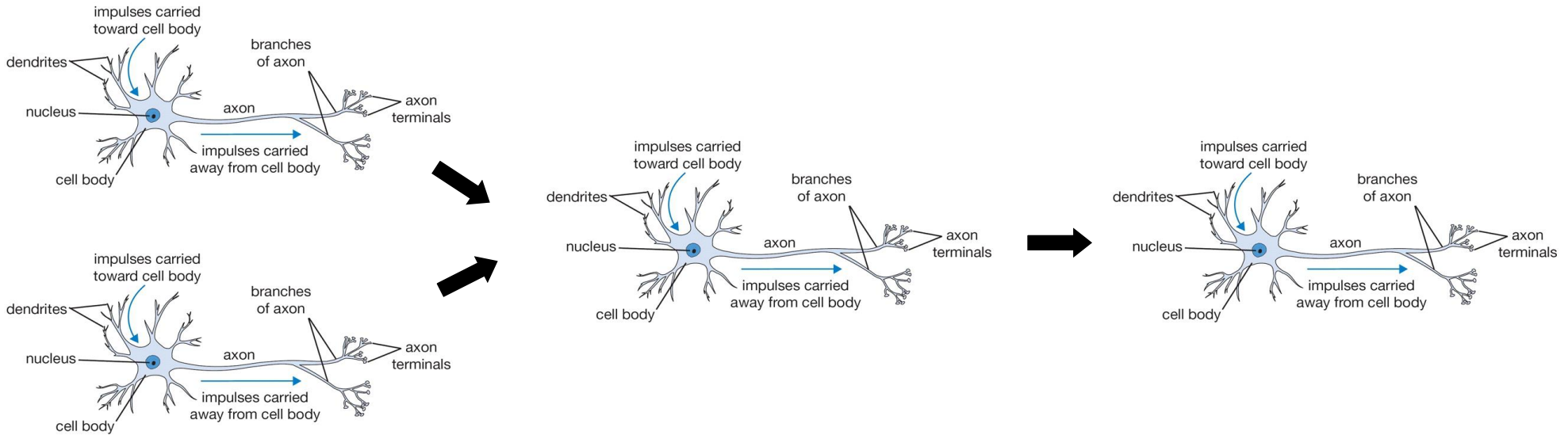
③ 수정하기 → Update
(Optimize)

퍼셉트론 (Perceptron)



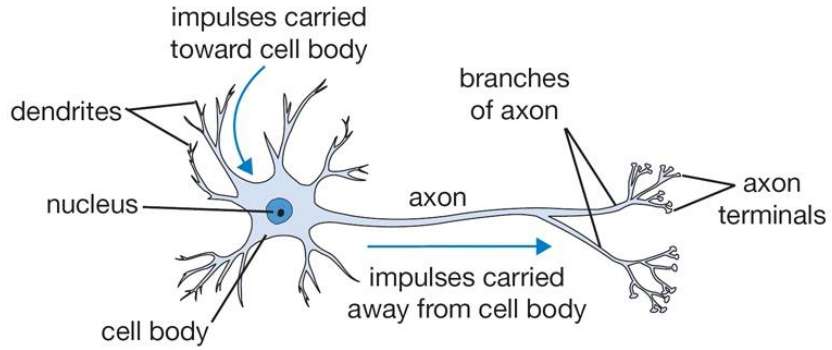


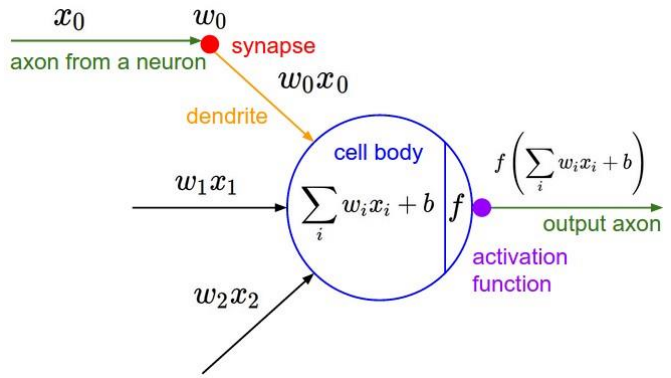
퍼셉트론 (Perceptron)



1. 이전 뉴런들로부터 신호 받기
2. 중요한 정보 신호 받기
3. 받은 신호 처리하기
4. 다음 뉴런에게 전달할지 판단하기
5. 신호 전달하기

퍼셉트론 (Perceptron)

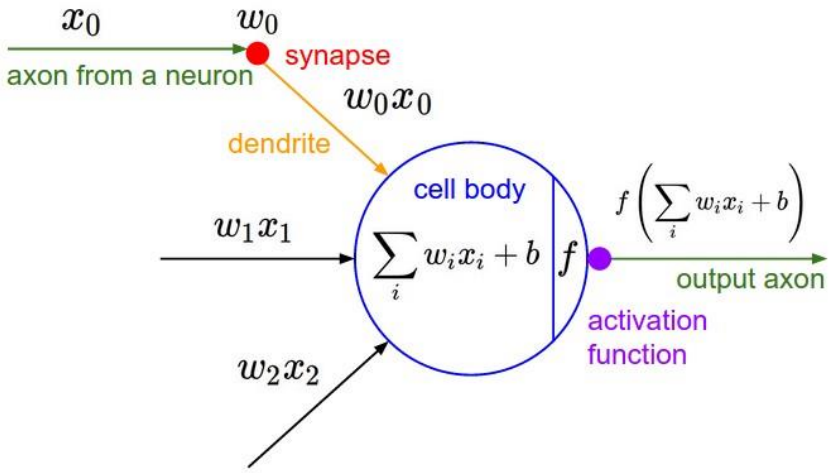




1. 이전 뉴런들로부터 신호 받기
2. 중요한 정보 신호 받기
3. 받은 신호 처리하기
4. 다음 뉴런에게 전달할지 판단하기
5. 신호 전달하기

1. x_1, x_2
2. w_1, w_2
3. $\sum w_i x_i + b$
4. Activation Function
5. Output

퍼셉트론 (Perceptron)



$$f\left(\sum_i w_i x_i + b\right)$$

$$f([w_1 \;\; w_2]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b)$$

퍼셉트론에 들어오는 신호들(x_i)에 가중치를 곱하고($w_i x_i$), 이들과 편향(b)을 더한다.

이후 activation function (f)를 취해 출력값을 만든다.

목표 : 적절한 가중치(w_i)를 찾아내기!

Activation Function (활성화 함수)

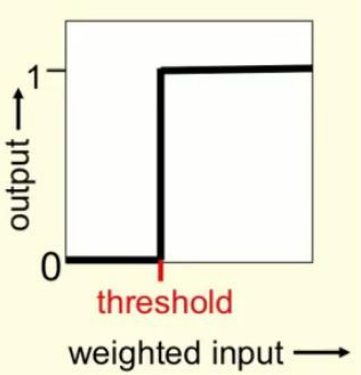
가중합($\sum w_i x_i$)이 임계값 (threshold) 이하이면 0(에 가까운 값)으로,

이상이면 1(에 가까운 값)으로 내보내는 함수

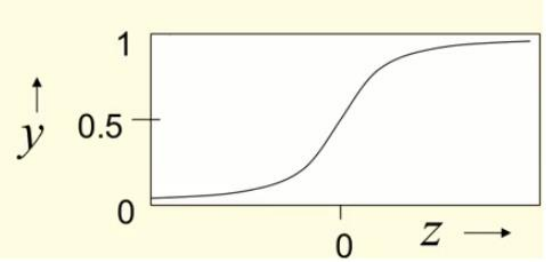
다음 퍼셉트론에 보낼 신호의 강도를 결정

비선형성을 주기 위하여 사용됨

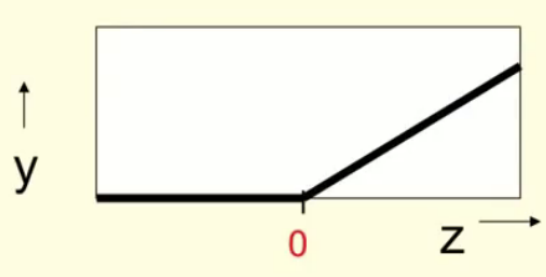
Step function



Sigmoid function



Rectified linear unit (ReLU)

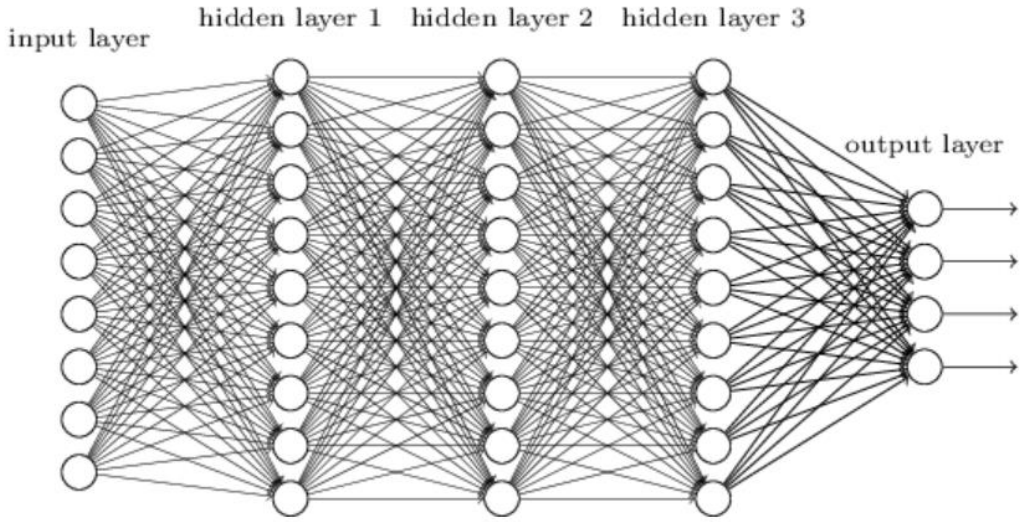


Multi Layer Perceptron (MLP)

= Fully Connected Layer (FC layer)

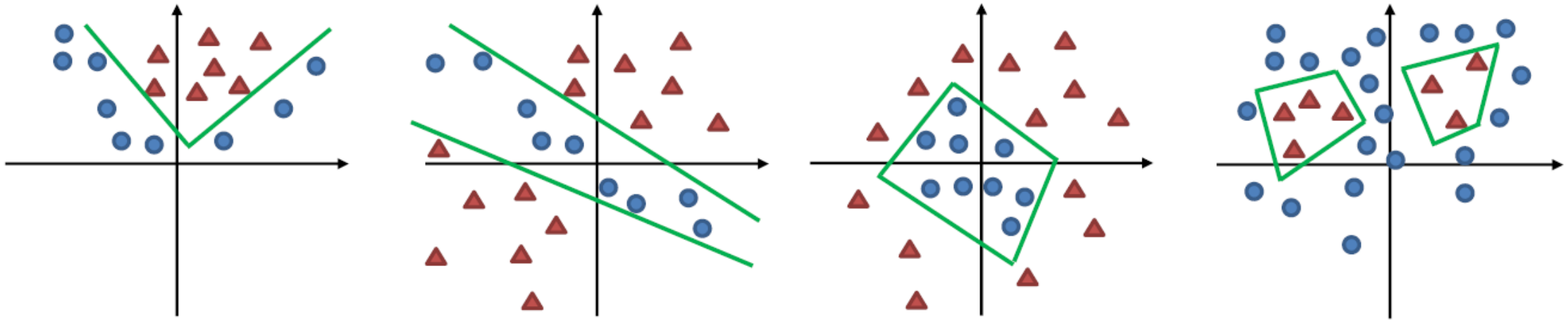
퍼셉트론이 여러 층 (layer)으로 쌓여 있는 구조, 가장 전통적이며 최소한의 성능 척도로 사용됨

현재도 많은 딥러닝 구조에 사용되며, 곱셈과 덧셈 연산으로만 이루어짐



Multi Layer Perceptron (MLP)

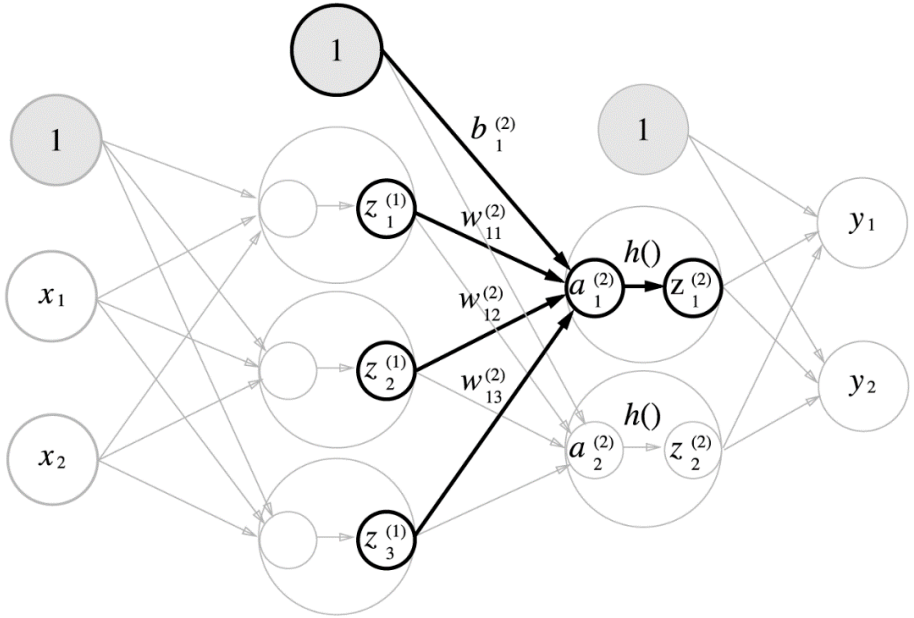
MLP를 활용하면 비선형 문제, 복잡한 문제도 해결 가능

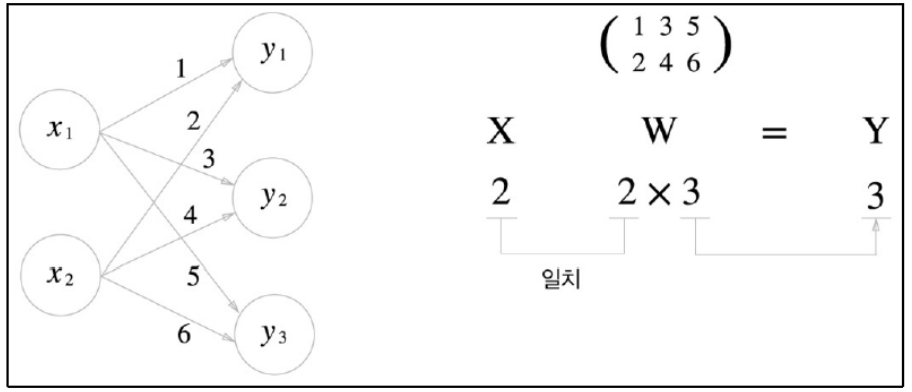


FeedForward

입력과 가중치의 행렬곱을 이용해 출력을 계산하는 것

⟶ Affine 연산

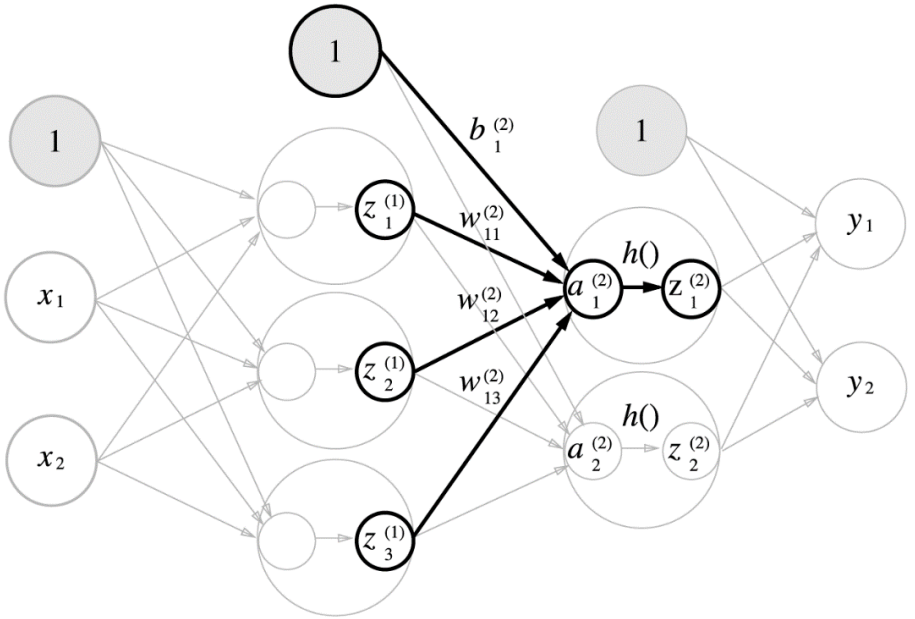




FeedForward

모든 layer를 통과할 때까지 계산

최종적으로 문제에서 정의한 결과물의 점수 (score)가 나옴



Cat: 1.5

Dog: 3.7

Loss Function (손실 함수)

출력으로 나온 데이터가 실제 정답과 얼마나 차이가 나는지를 나타냄

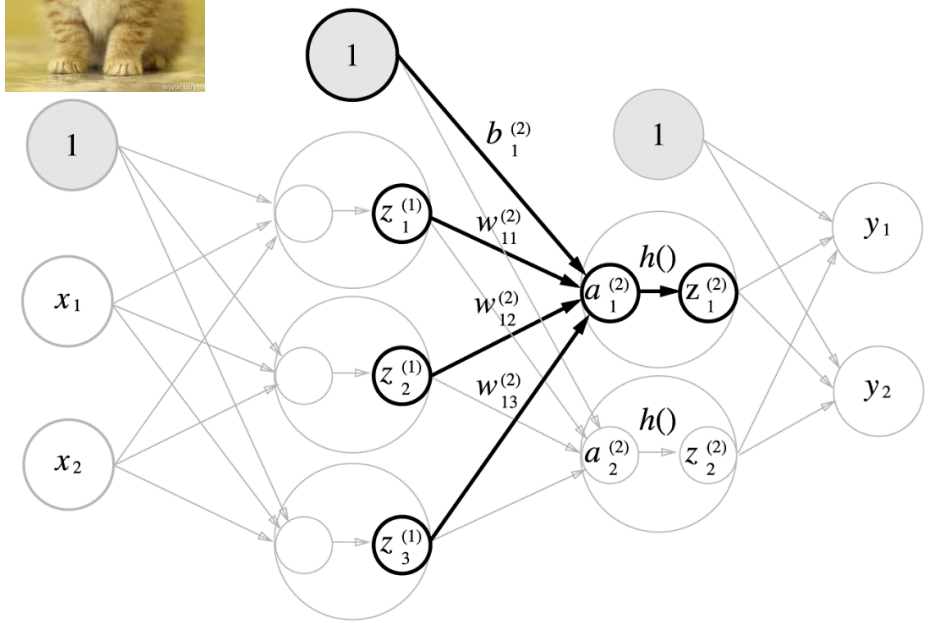
학습의 방향을 설정하는 아주 중요한 부분!

Loss Function의 결과값을 토대로 가중치들이 업데이트 됨
학습이 거듭 반복되면 해당 Loss function은 값이 작아짐

회귀(Regression) 문제 : Mean Squared Error
분류(Classification) 문제 : Cross Entropy

Mean Squared Error (MSE, 평균 제곱 오차)

$$E = \frac{1}{2}\sum_{k}(y_k - t_k)^2$$



Cat: 0.1 ⟷ Label: 1

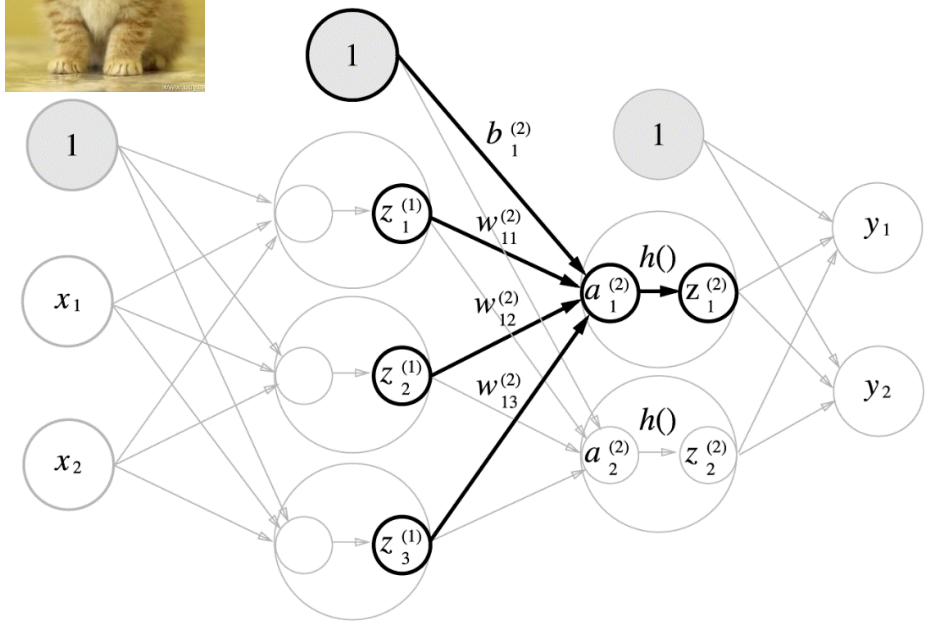
Dog: 0.9 ⟷ Label : 0

Error

$(0.1 - 1)^2$

$(0.9 - 0)^2$

Mean Squared Error (MSE, 평균 제곱 오차)



$$E = -\sum_k t_k \log y_k$$

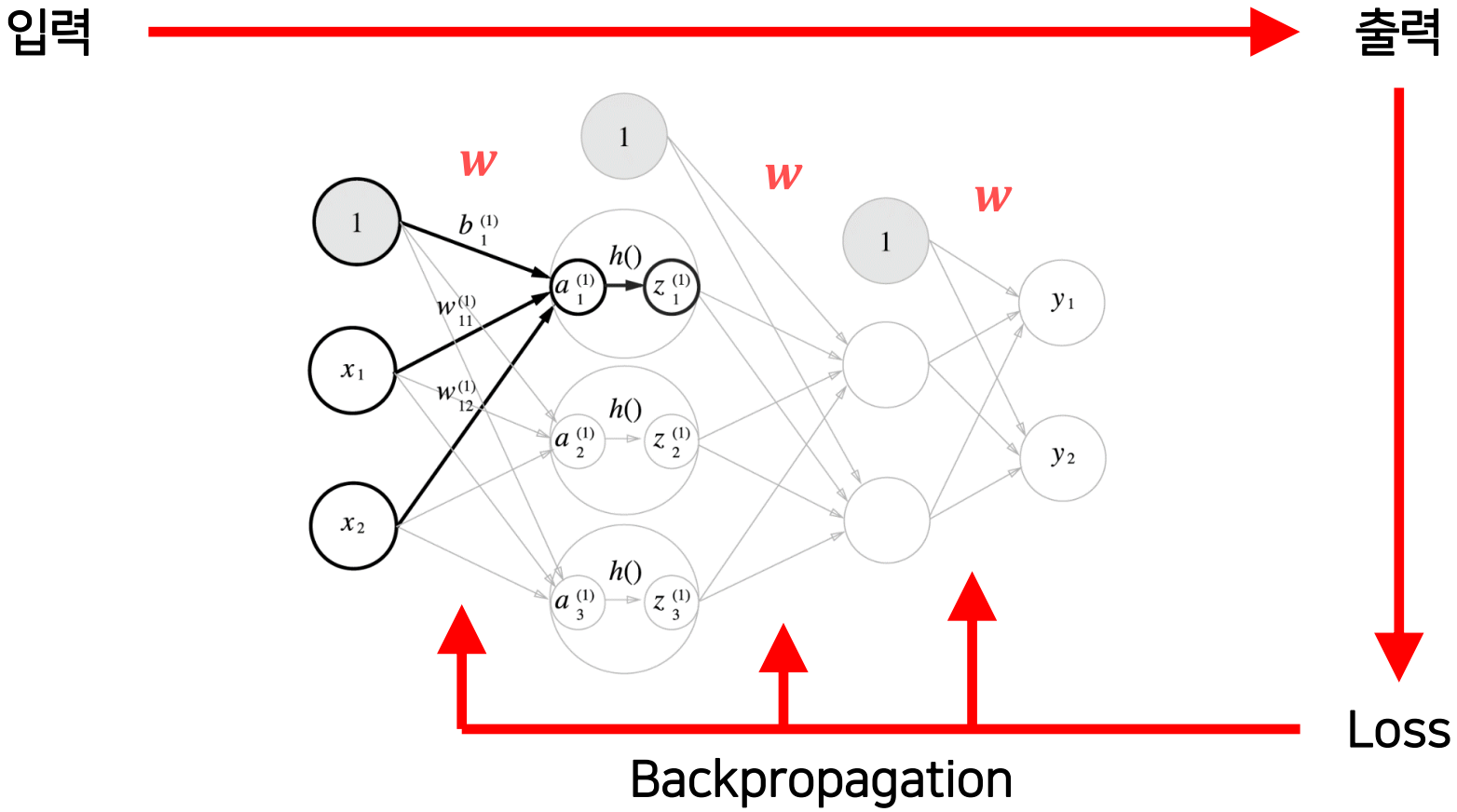
Error

Cat: 0.1 ⟷ Label: 1 $\quad 1 \times log0.1$

Dog: 0.9 ⟷ Label : 0 $\quad 0 \times log0.9$

정답일 때의 출력이 전체 값을 결정!

Backpropagation (역전파)



Backpropagation (역전파)

출력과 Loss는 Feedfoward들의 결과로 나온 값

작은 Loss는 적절한 w들 덕분, 높은 Loss는 적절하지 않은 w들 때문

Loss를 바탕으로 각각의 $\boldsymbol{w}$ 하나 하나가 얼마나 영향을 주었는지 알아내고

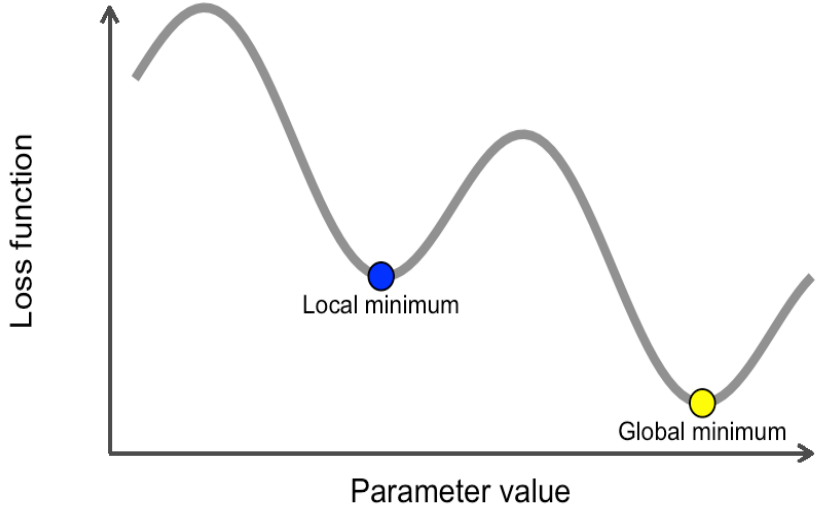
좀 더 적절한 방향으로 $\boldsymbol{w}$들을 변경해 나가는 것을 Backpropagation (역전파)이라 함

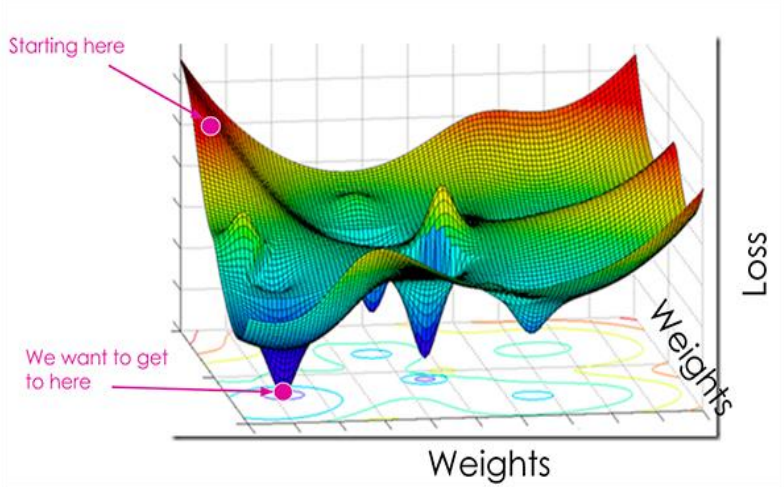
목표 : 적절한 가중치(w_i)를 찾아내기!

Backpropagation (역전파)

w와 loss의 관계를 이용해 좋은 w를 찾아가는 방법이 존재

그래프의 기울기를 이용한 Gradient Decent (경사 하강법)★★★★★



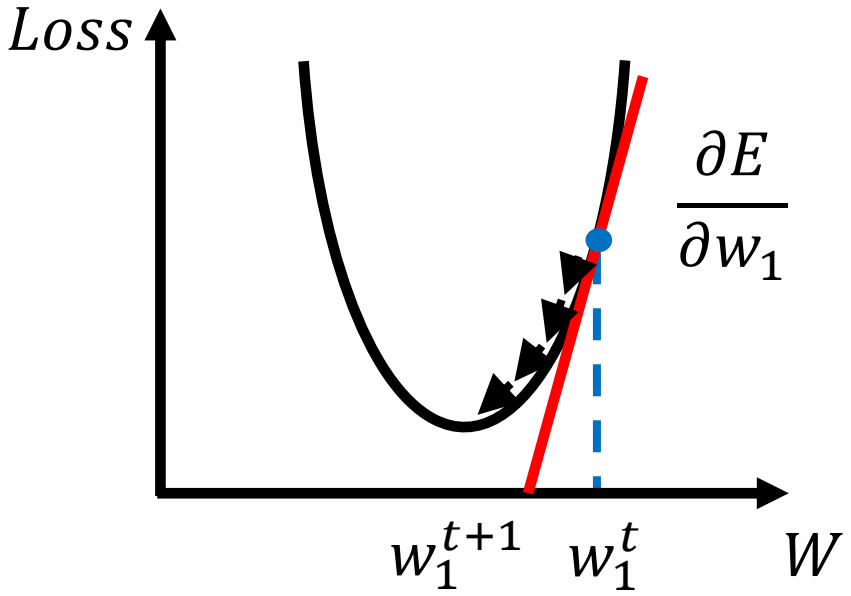


https://bradleyboehmke.github.io/HOML/deep-learning.html
https://www.pyimagesearch.com/2019/10/14/why-is-my-validation-loss-lower-than-my-training-loss/

Gradient Decent (경사 하강법)

임의의 한 지점으로부터 시작해, loss가 줄어드는 방향으로

w (weights=parameters)들을 갱신하는 방법



Gradient Decent (경사 하강법)

목적지 : 산 아래 지점

목적지 방향은 어떻게 알 수 있을까?

→ 발에 집중해 아래쪽으로 보며 방향을 찾음

→ 그 방향으로 얼만큼 움직여야 하는가?

→ 한 발짝

→ [기울어진 방향]으로 [한 발짝] 이동

Trask, Andrew W. Grokking deep learning. Simon and Schuster, 2019.

Gradient Decent (경사 하강법)

목적지 : 산 아래 지점

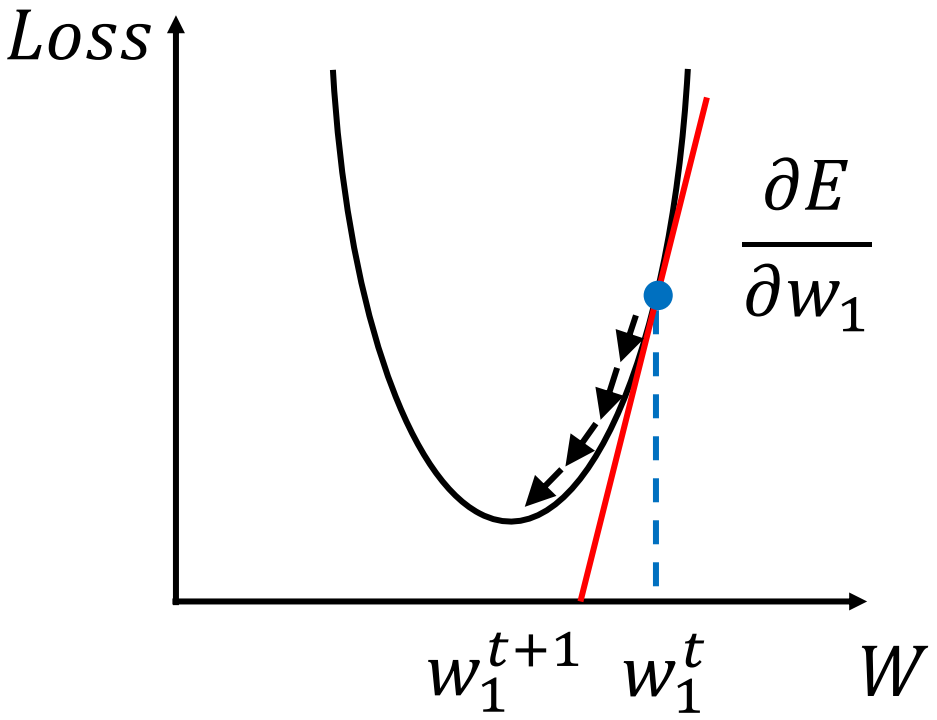
목적지 방향은 어떻게 알 수 있을까?

→ 발에 집중해 아래쪽으로 보며 방향을 찾음

→ 그 방향으로 얼만큼 움직여야 하는가?

→ 한 발짝

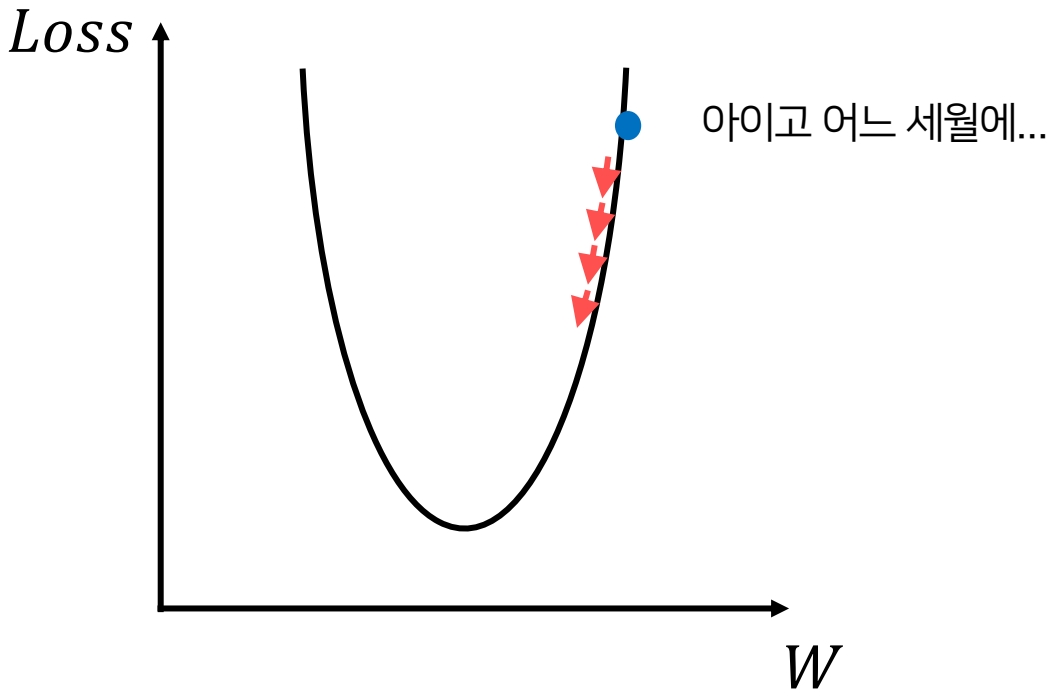
→ [기울어진 방향]으로 [한 발짝] 이동



Gradient Decent (경사 하강법)

step이 너무 작다면?

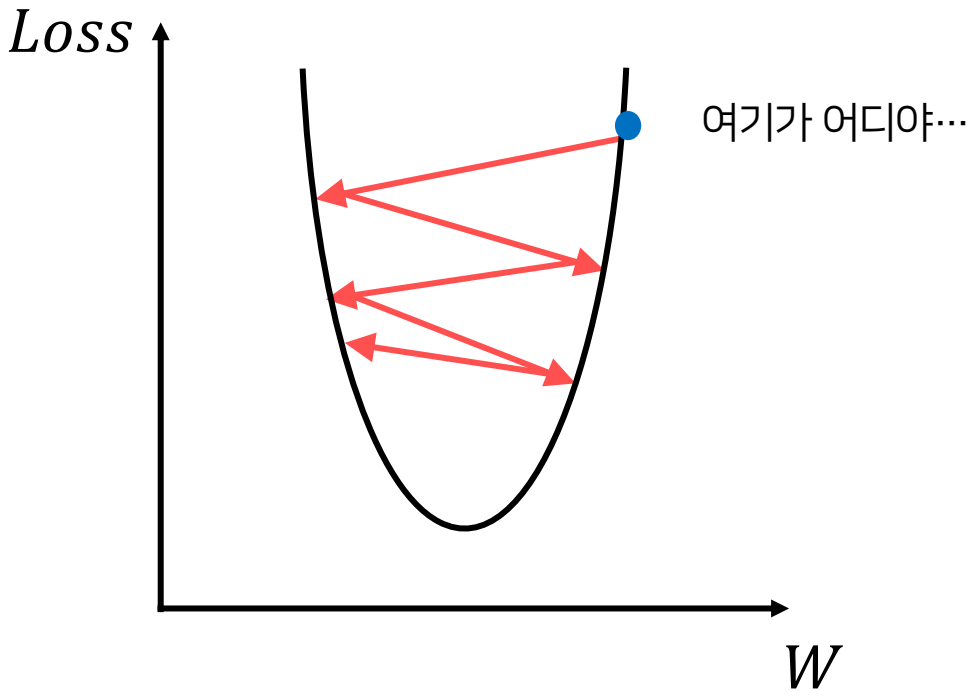
optimum에 도달하기까지 오랜 시간이 걸림



Gradient Decent (경사 하강법)

step이 너무 크다면?

반대편으로 건너뛰어 optimum을 찾기 힘듦



Optimization (최적화)

역전파를 통해 weights(= parameters)을 업데이트 시키는 방법

학습의 안정성 및 효율성을 위해 여러 Optimization 기법들이 제안됨

- Stochastic Gradient Decent (SGD)
- AdaGrad
- RMSProp
- Momentum
- Adam

Batch Learning (Epoch Learning)

전체 훈련 데이터를 사용

대규모 데이터셋에 적용하기 힘듬

데이터 구성이 항상 같기 때문에 local minima에 빠질 가능성이 있음

$$w_1^{t+1} = w_1^t - \varepsilon \frac{\partial E}{\partial w_1}$$

↑

모든 학습 샘플에 대해서 계산되는 오차 함수의 기울기

Batch Learning (Epoch Learning)

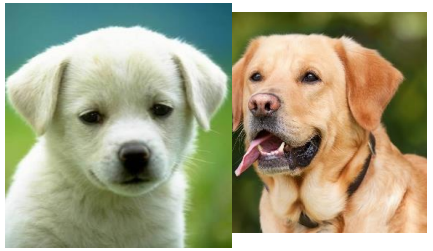
몇 개의 샘플을 하나의 소규모 집합 단위로 하여 가중치를 업데이트

복수의 샘플을 묶은 작은 집합을 미니배치 (Mini-batch)라고 함

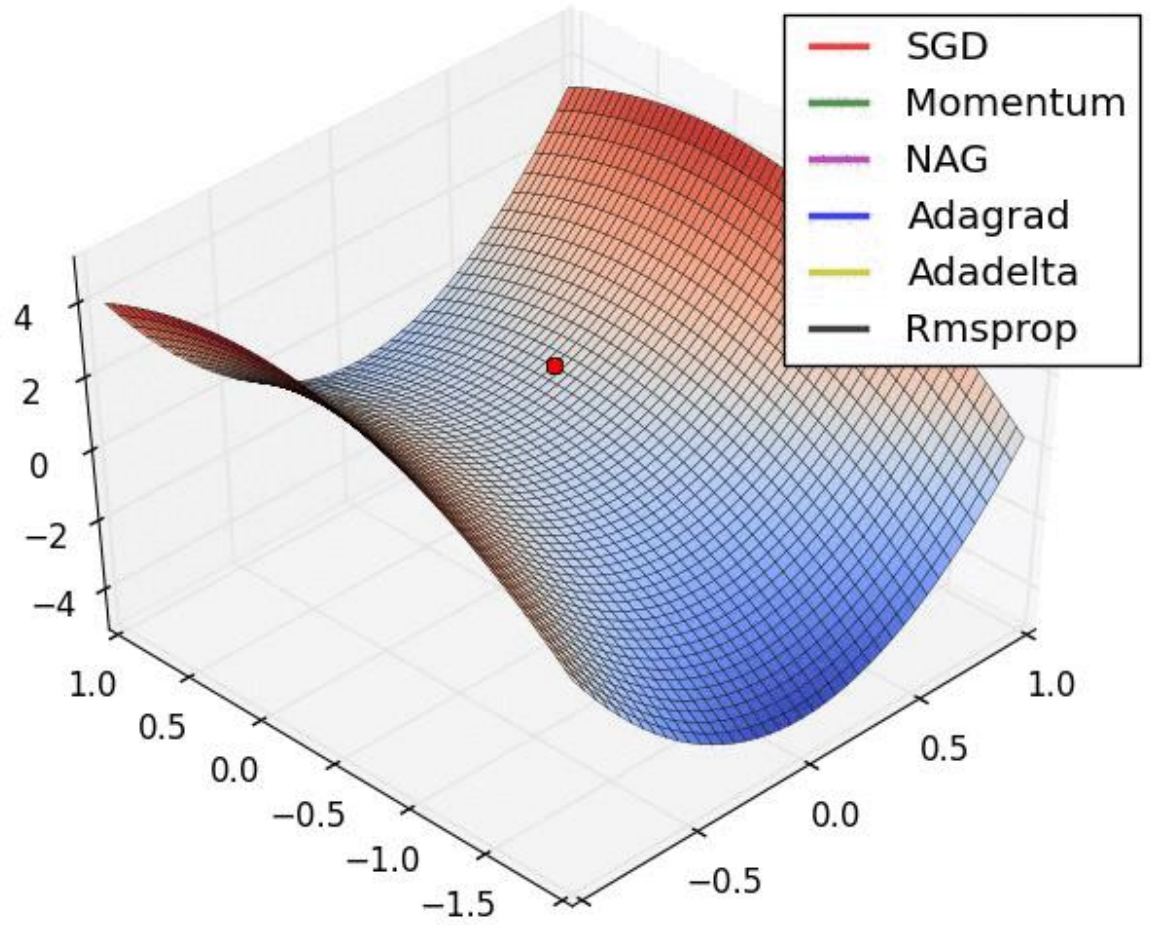
Mini-batch#1

Mini-batch#2

Mini-batch#3



Mini-batch#4


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5