딥러닝팀

1팀

이정환
김동환
권가민
박준영
박채원

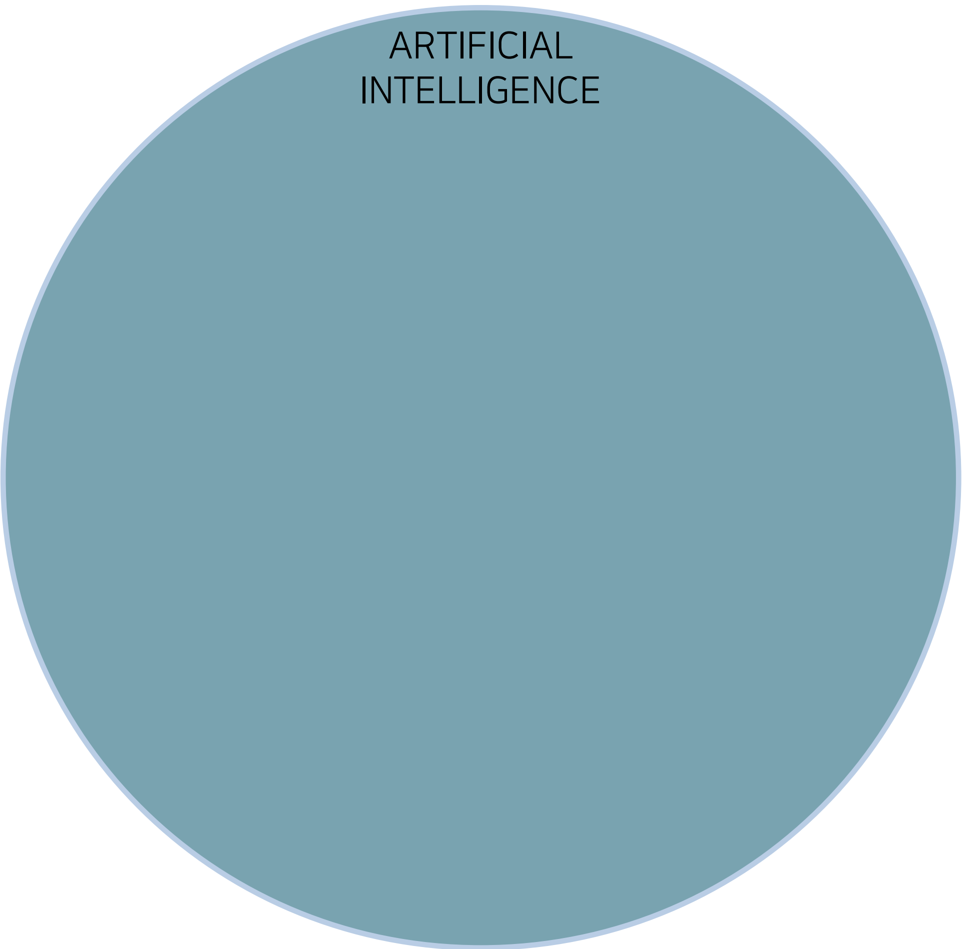
CONTENTS

1

머신러닝과 딥러닝

1 머신러닝과 딥러닝

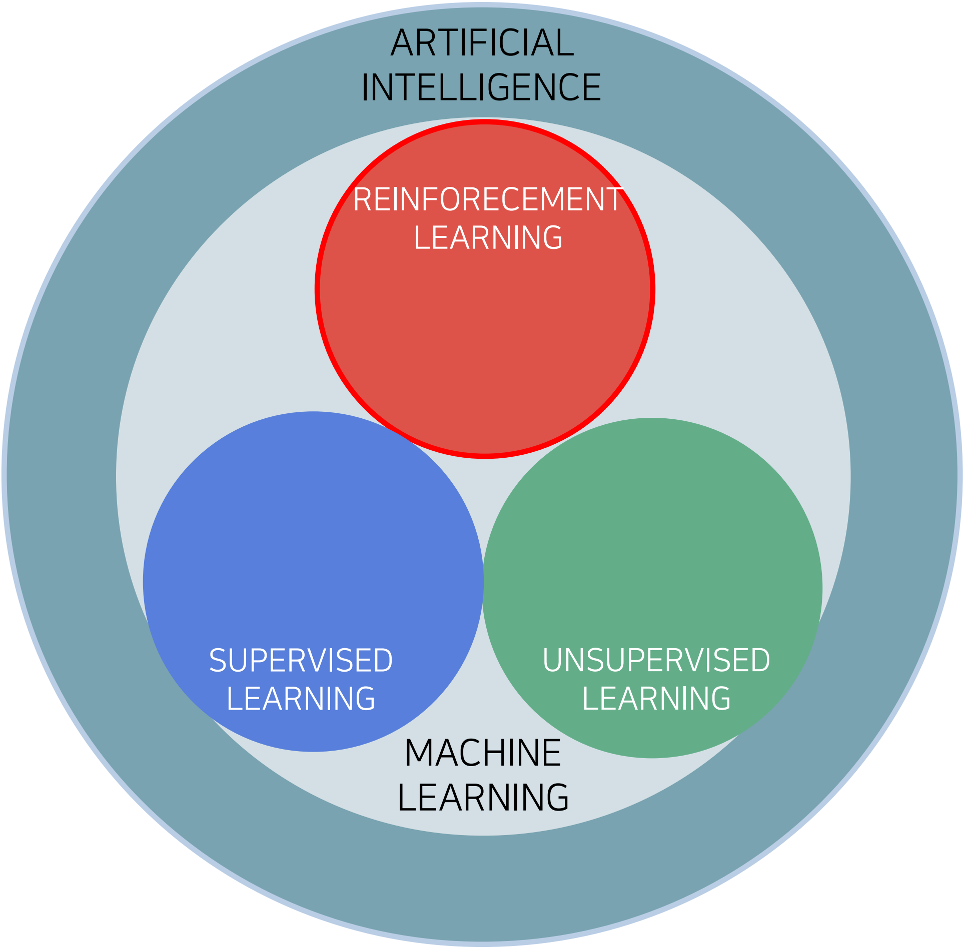
인공지능



문제해결, 패턴인식 등과 같이
인간의 지능과 연결된
문제를 연구하는 컴퓨터 공학 분야

1 머신러닝과 딥러닝

머신러닝

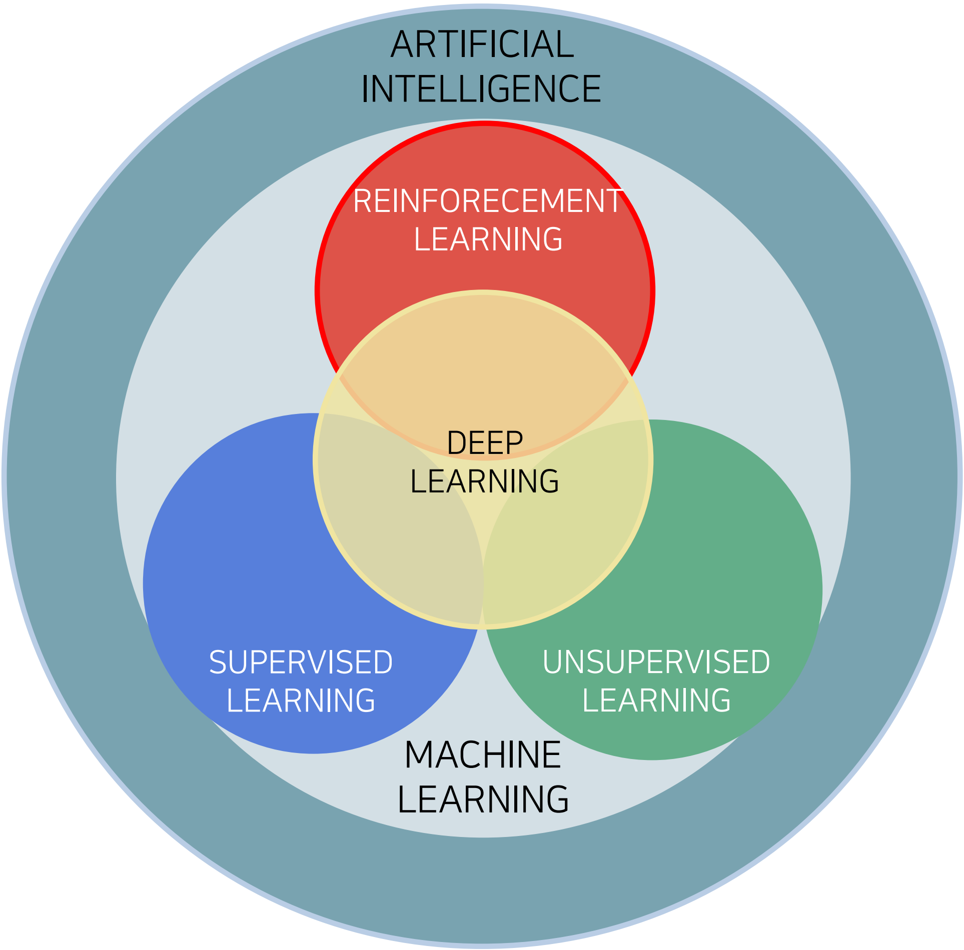


주어진 목표에 대해

컴퓨터가 스스로 학습할 수 있도록

알고리즘과 기술을 개발하는 분야

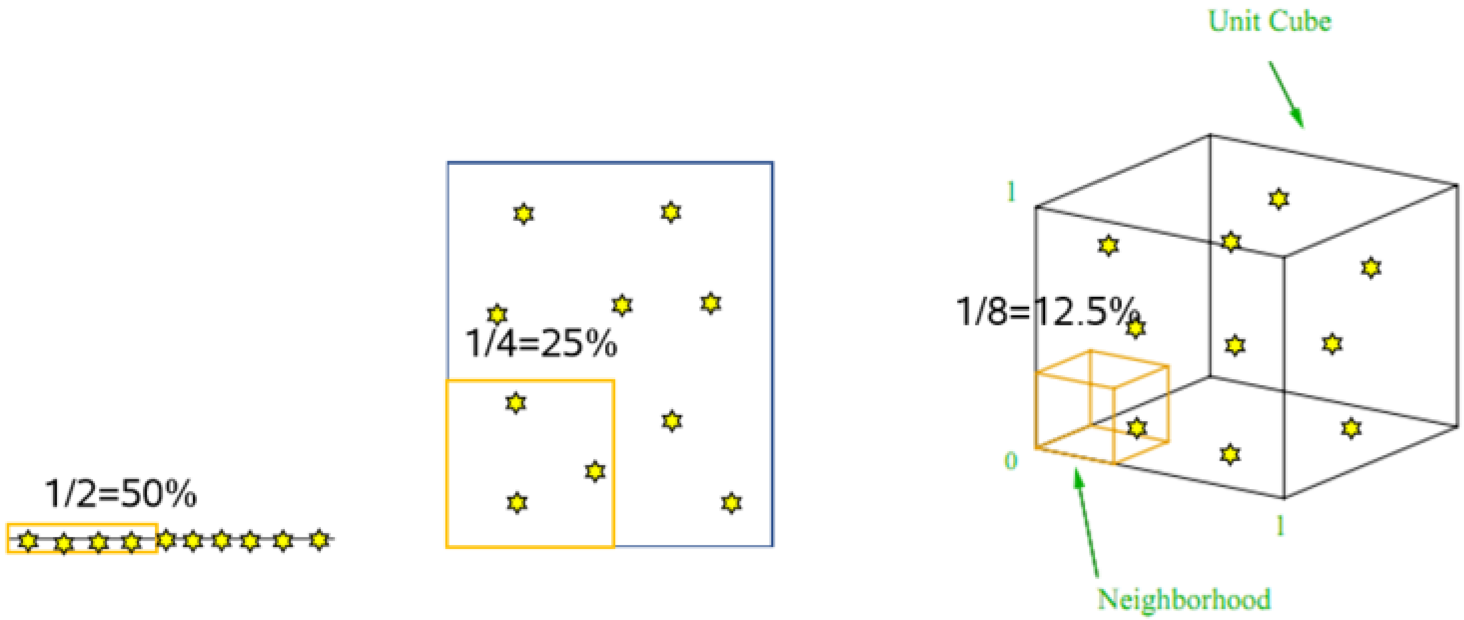
딥러닝



머신 러닝에서 발전된 형태로
사람의 사고방식을 컴퓨터에게
가르치는 머신 러닝의 한 분야

머신러닝의 한계점

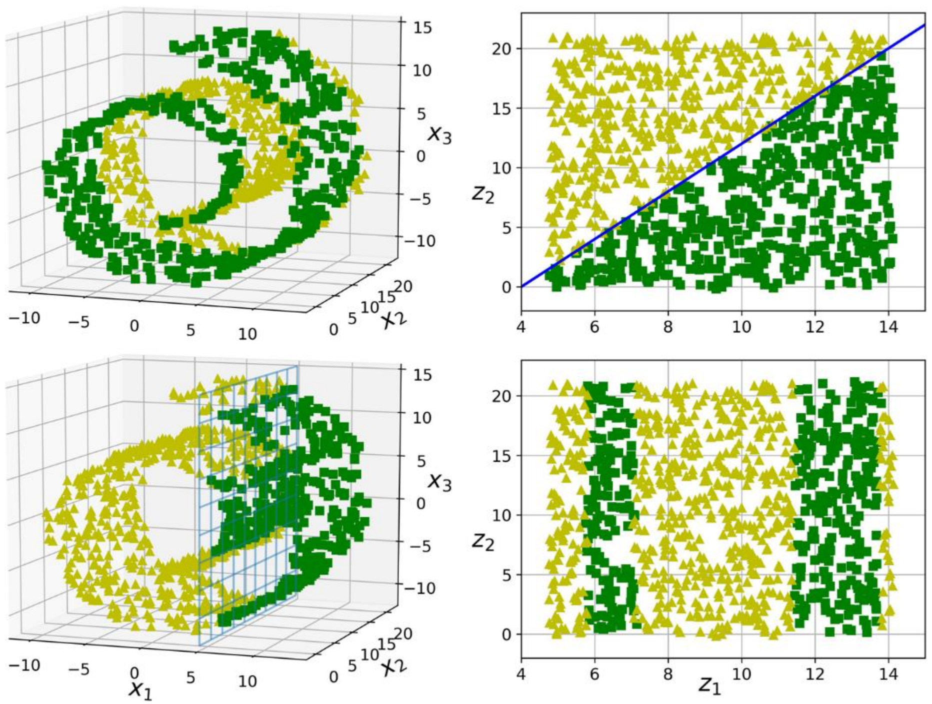
차원의 저주



입력 데이터의 특성 수가 증가함에 따라
알고 있는 정보가 상대적으로 감소하여 모델의 성능 저하되는 현상

머신러닝의 한계점

다양체 가설

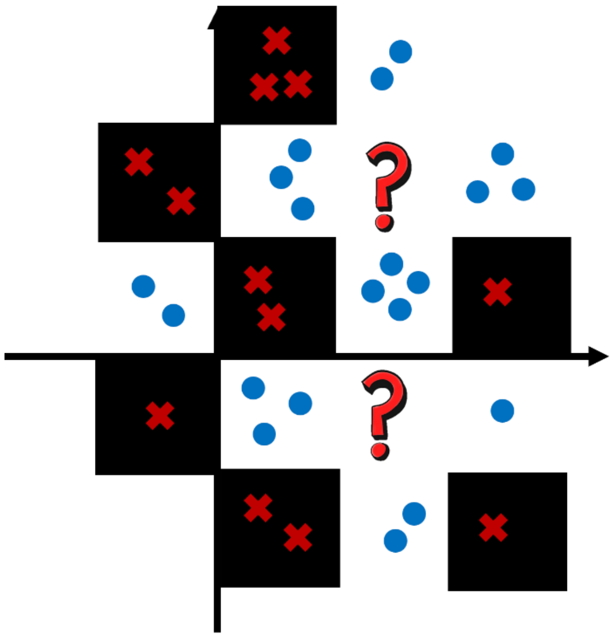


유의미한 데이터를 고차원 공간에서

낮은 차원으로 축소 시 데이터가 더 복잡하게 변환될 가능성 존재

머신러닝의 한계점

국소일치성

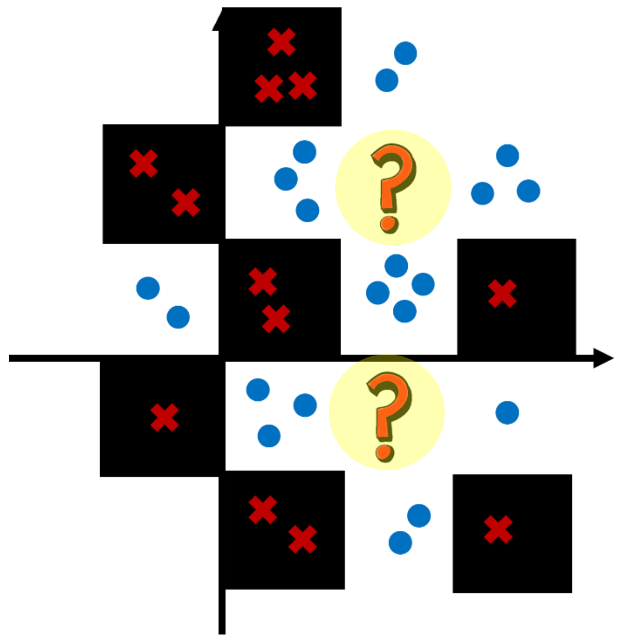


함수가 작은 영역에서 아주 크게 변하면 안 된다는 가정으로

거의 모든 머신러닝, 통계적 기법들이 암묵적으로 가지고 있음

머신러닝의 한계점

국소일치성

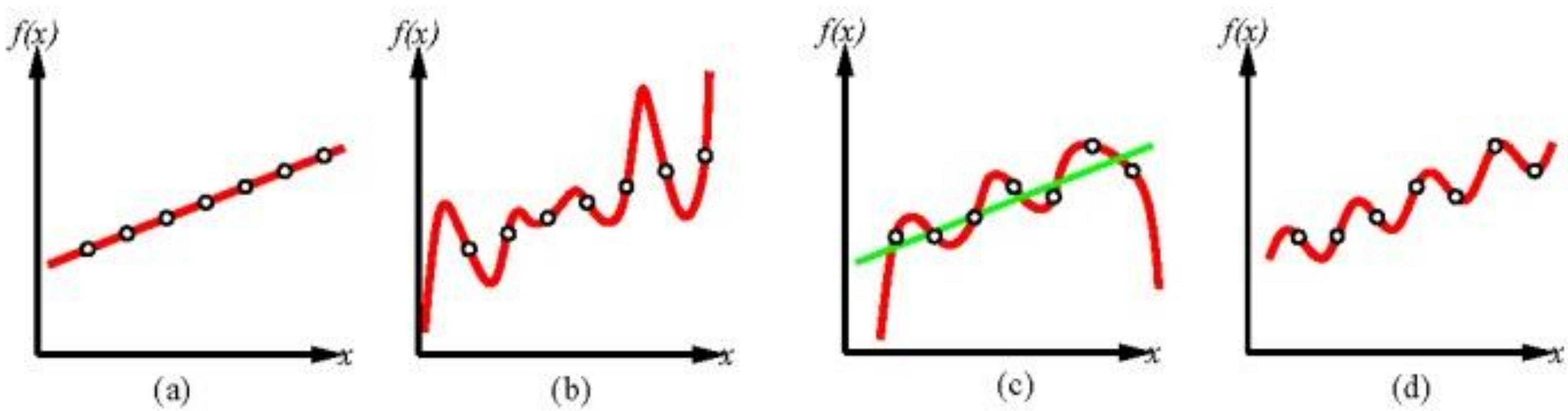


위와 같은 상황에서 이 가정을 이용한 모델들은

예측하려는 공간을 흰색으로 예측하는 오류를 범할 수 있음

딥러닝 모델의 한계점

귀납적 편향



학습 데이터 이외의 데이터에 대해서도 모델의 성능을 일반화하기 위해
모델의 가정이나 구조에 존재하는 휴리스틱한 방법

굳이 체계적이고 합리적인 판단을 할 필요가 없는 상황에서 신속하게 사용하는 어림짐작의 기술

딥러닝 모델의 한계점

귀납적 편향

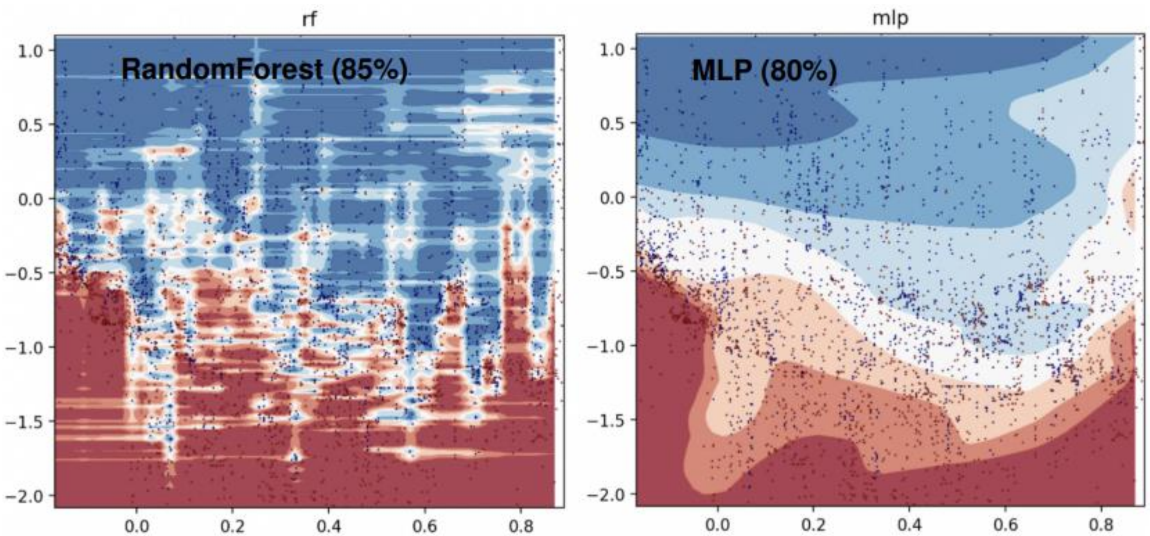


Figure 20: Decision boundaries of a default MLP and RandomForest for the 2 most important features of the *electricity* dataset

MLP 모델이 가진 귀납적 편향(부드러운 결정경계)의 영향으로
정형 데이터에서 성능이 트리기반 모델보다 좋지 못함

머신러닝 vs 딥러닝

특징 공학(Feature Engineering)

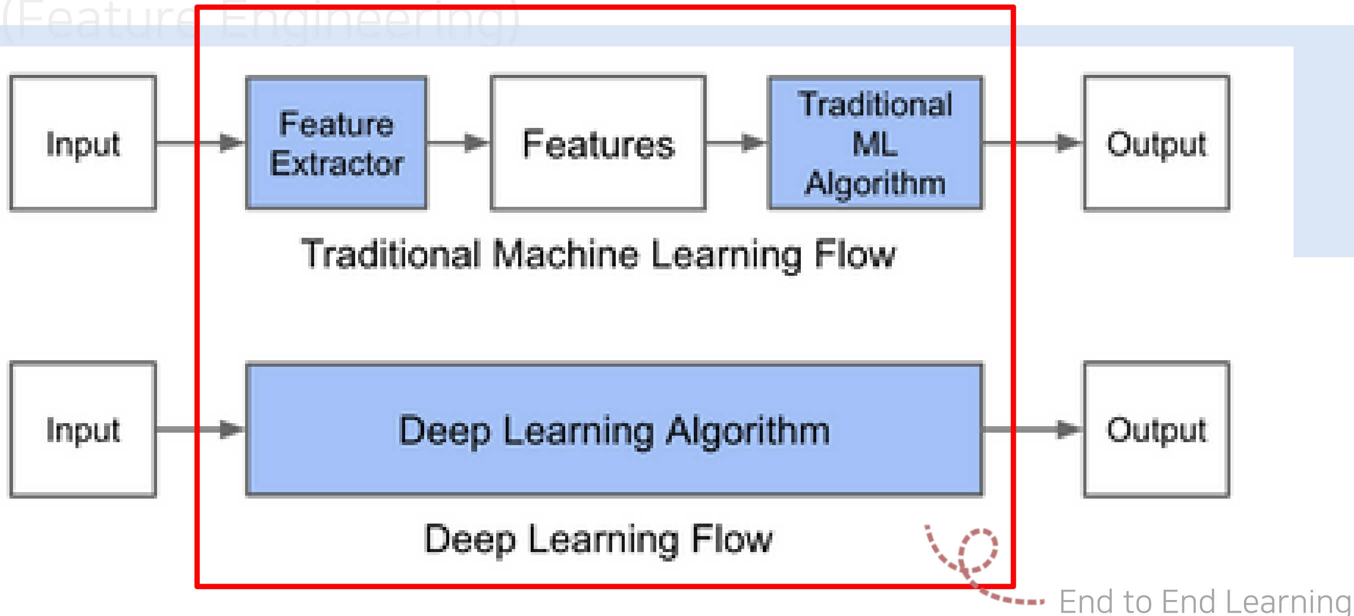
데이터 분석과정에서 입력 데이터로부터

유의미한 변수를 추출하고 가공하는 과정

1 머신러닝과 딥러닝

머신러닝 vs 딥러닝

특징 공학(Feature Engineering)



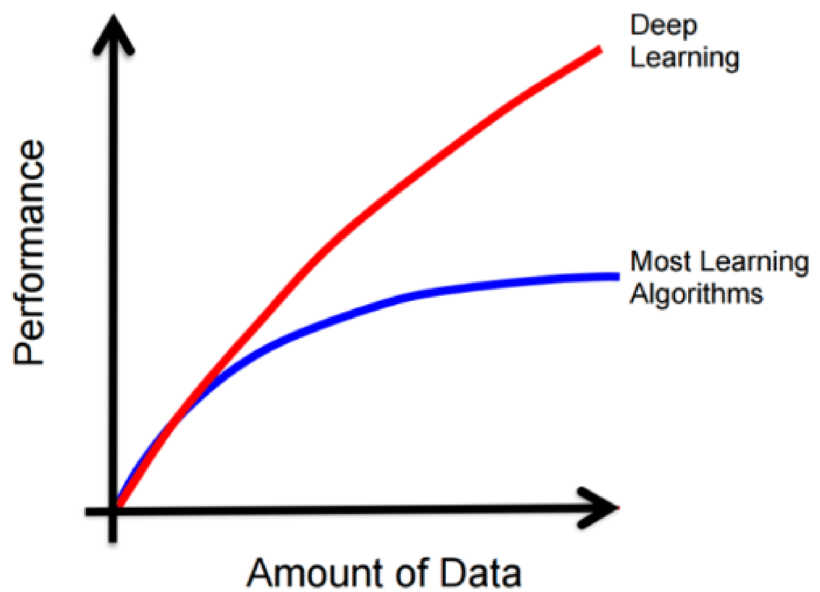
머신러닝과 달리 딥러닝은 모델이 가공되지 않은

입력데이터를 직접 처리하여 Feature engineering의 영향이 낮음

1 머신러닝과 딥러닝

머신러닝 vs 딥러닝

특징 공학(Featu

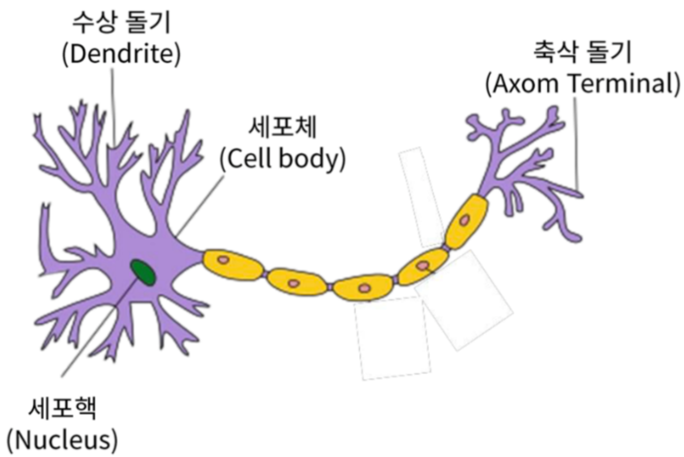


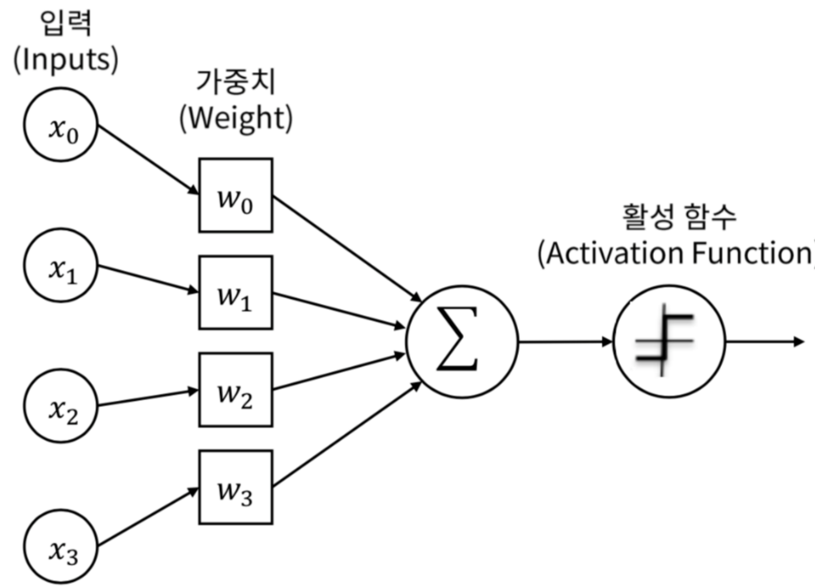
일반적으로 데이터의 양이 많아질 수록 딥러닝 모델은 성능이 향상됨

2

퍼셉트론

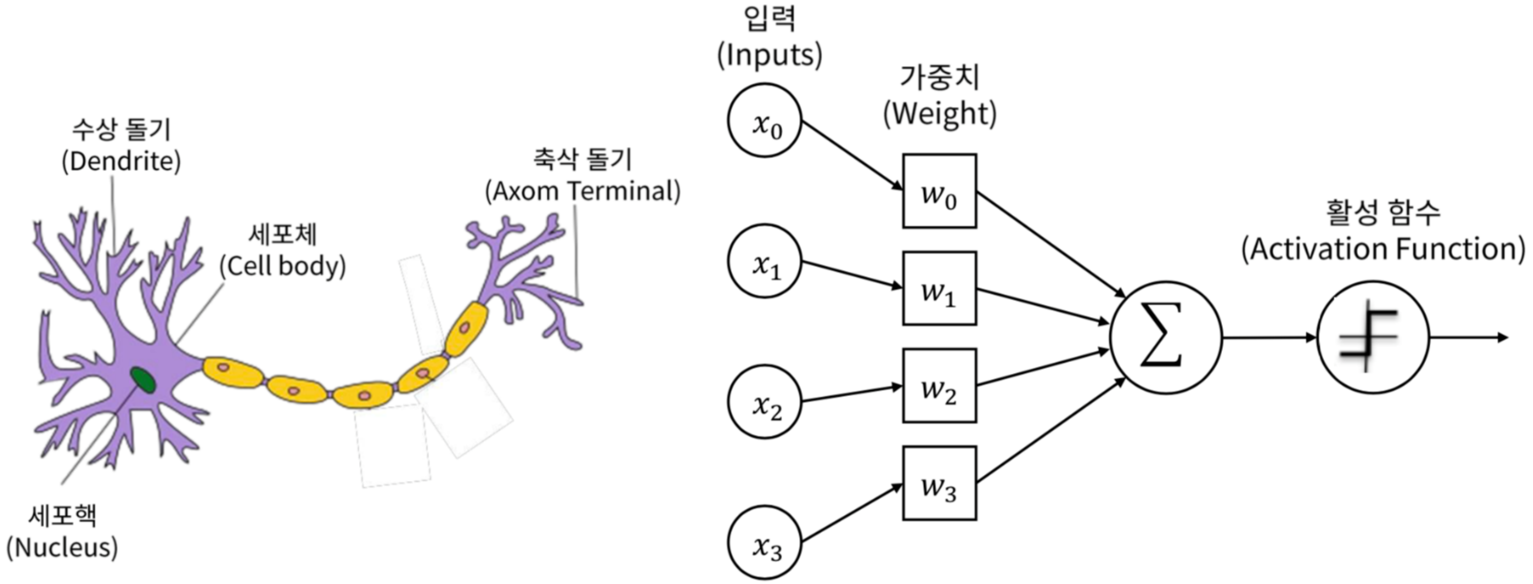
단층 퍼셉트론





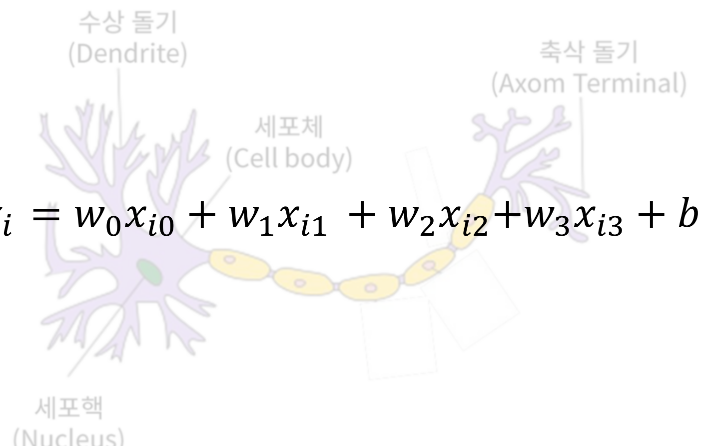
인간 신경의 기본 단위인 뉴런을 모방하여 수학적으로 모델링한 것으로
뉴런과 마찬가지로 입력을 받아서 출력 값을 내보낼 수 있음

퍼셉트론 동작 방식

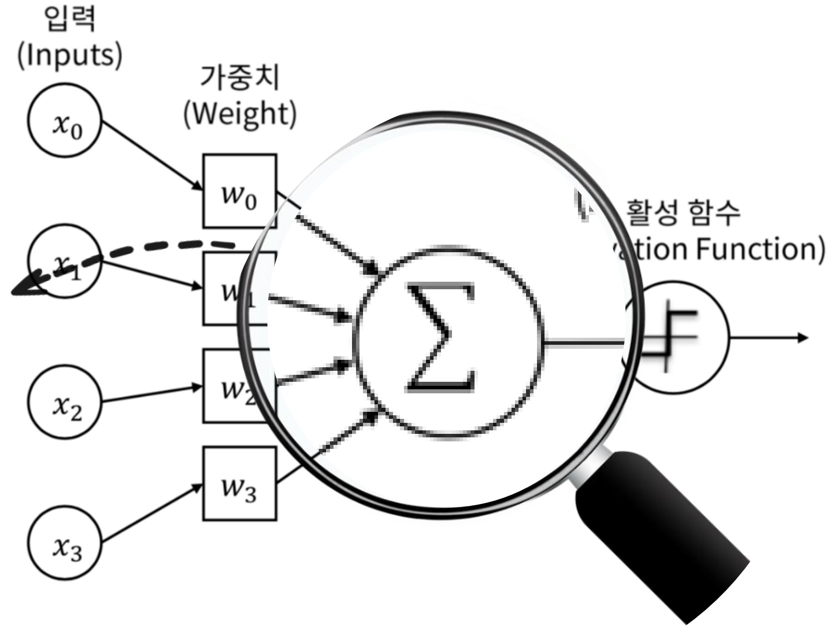


1. i번째 데이터 x_i를 입력으로 가중합 z_i 계산
2. 계단함수를 통해 예측값 $g_w(x_i)$ 계산
3. 예측값과 실제값의 오차에 학습률 η와 입력 데이터의 곱을 w_j에 더해 업데이트

퍼셉트론 동작 방식



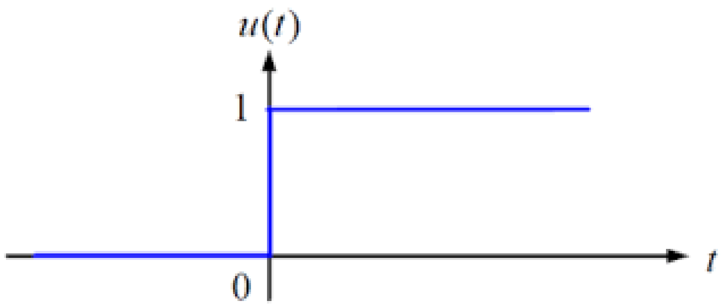
$$z_i = w_0 x_{i0} + w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3} + b$$

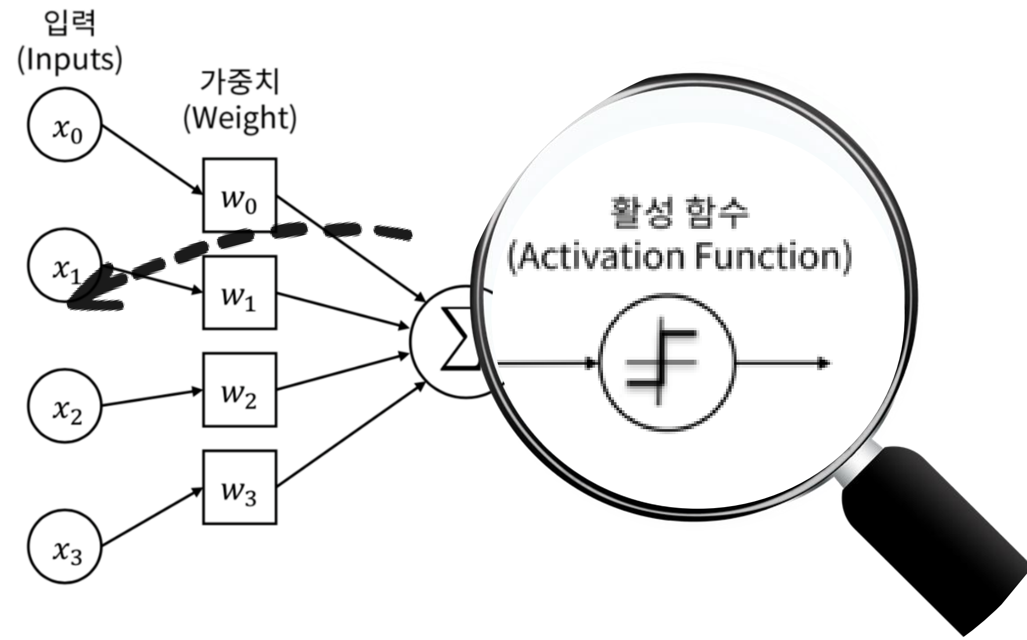


1. i번째 데이터 x_i를 입력으로 가중치 z_i 계산
2. 계단함수를 통해 예측값 $g_w(x_i)$ 계산
3. 예측값과 실제값의 오차에 학습률 η와 입력 데이터의 곱을 w_j에 더해 업데이트

퍼셉트론 동작 방식

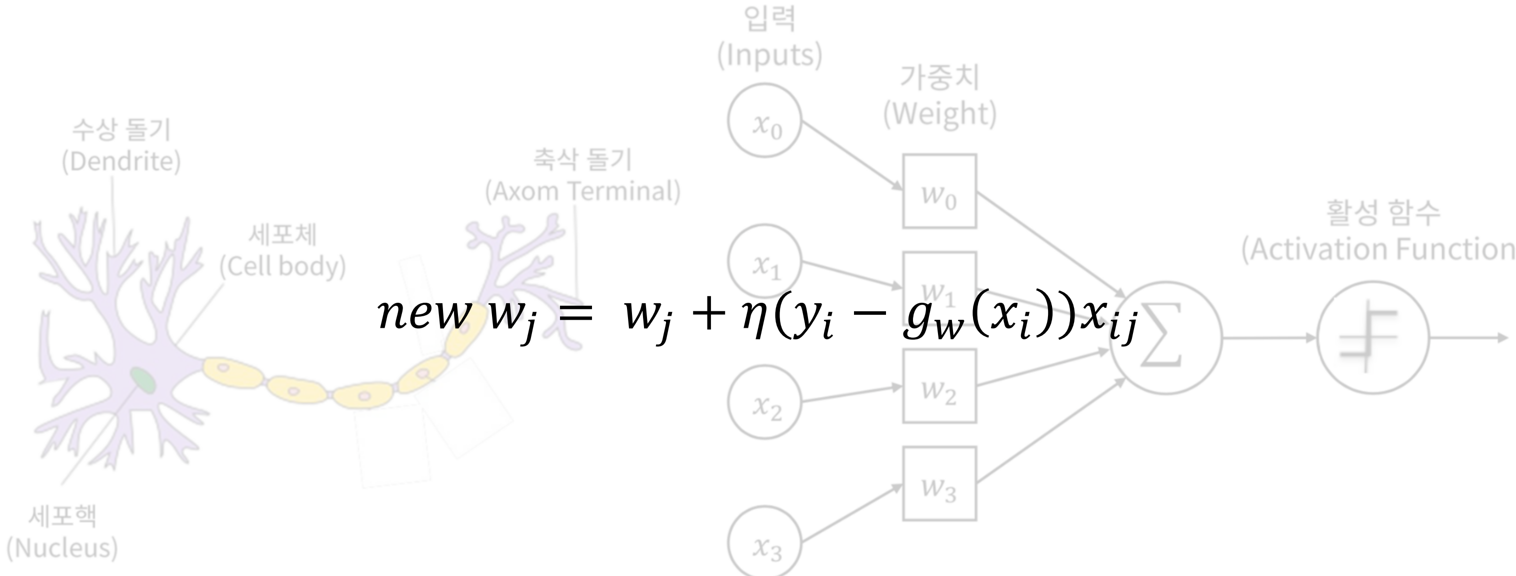
$$g_w(x_i) = \begin{cases} 0, & (z_i \le 0) \\ 1, & (z_i > 0) \end{cases}$$





1. i번째 데이터 x_i를 입력으로 가중합 z_i 계산
2. 계단함수를 통해 예측값 $g_w(x_i)$ 계산
3. 예측값과 실제값의 오차에 학습률 η와 입력 데이터의 곱을 w_j에 더해 업데이트

퍼셉트론 동작 방식

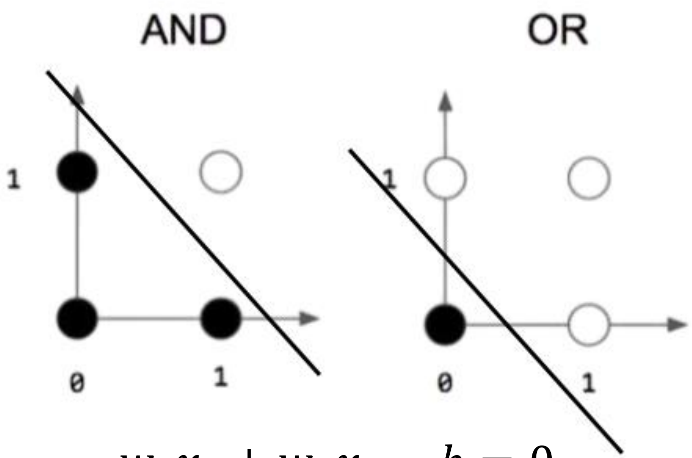


$$new\ w_j = w_j + \eta(y_i - g_w(x_i))x_{ij}$$

1. i번째 데이터 x_i를 입력으로 가중합 z_i 계산
2. 계단함수를 통해 예측값 $g_w(x_i)$ 계산
3. 예측값과 실제값의 오차에 학습률 η와 입력 데이터의 곱을 w_j에 더해 업데이트

논리연산의 한계

논리합/논리곱 문제



$$w_1x_1 + w_2x_2 - b = 0$$

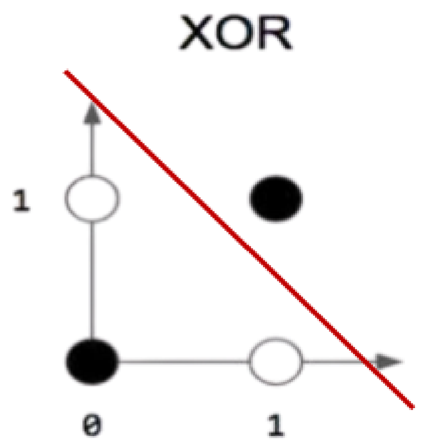
퍼셉트론은 하나의 직선으로 이해 할 수 있음

하나의 직선으로

검은 점과 흰 점 완전 분리 가능

논리연산의 한계

베타적 논리합 문제



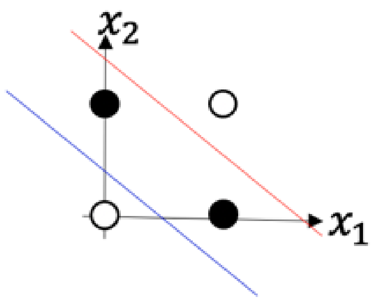
$$w_1x_1 + w_2x_2 - b = 0$$

하나의 직선으로

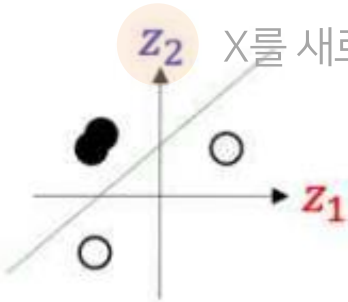
검은 점과 흰 점 완전 분리 불가능

논리연산의 한계

$w_{11}x_1 + w_{12}x_2 - b_1 = 0$



$w_{21}x_1 + w_{22}x_2 - b_2 = 0$

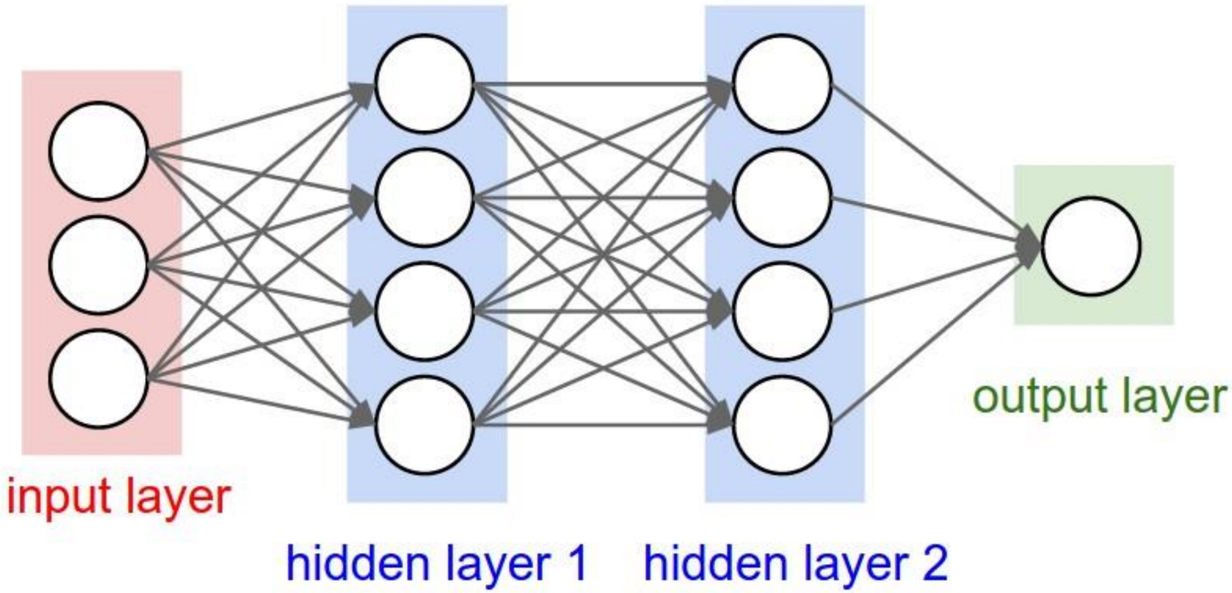


$v_1z_1 + v_2z_2 - b = 0$

2개의 직선을 이용하면

검은 점과 흰 점 완전 분리 가능

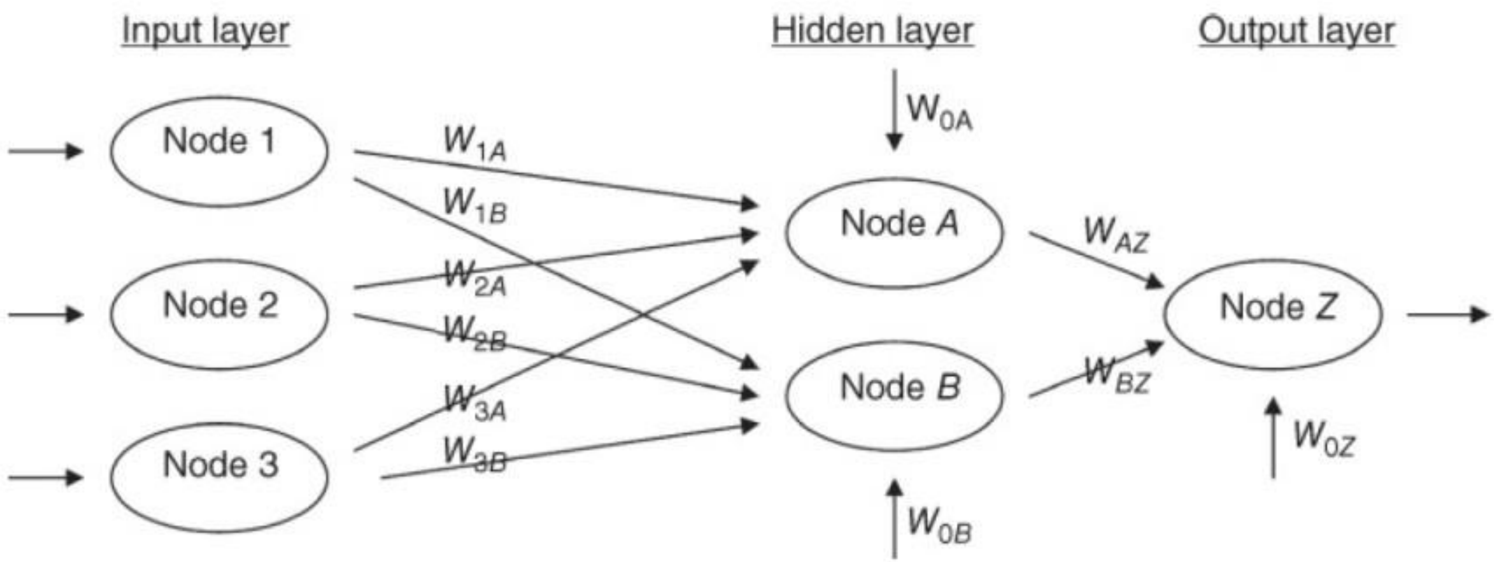
다층 퍼셉트론



여러 개의 퍼셉트론을 쌓아 올린 형태

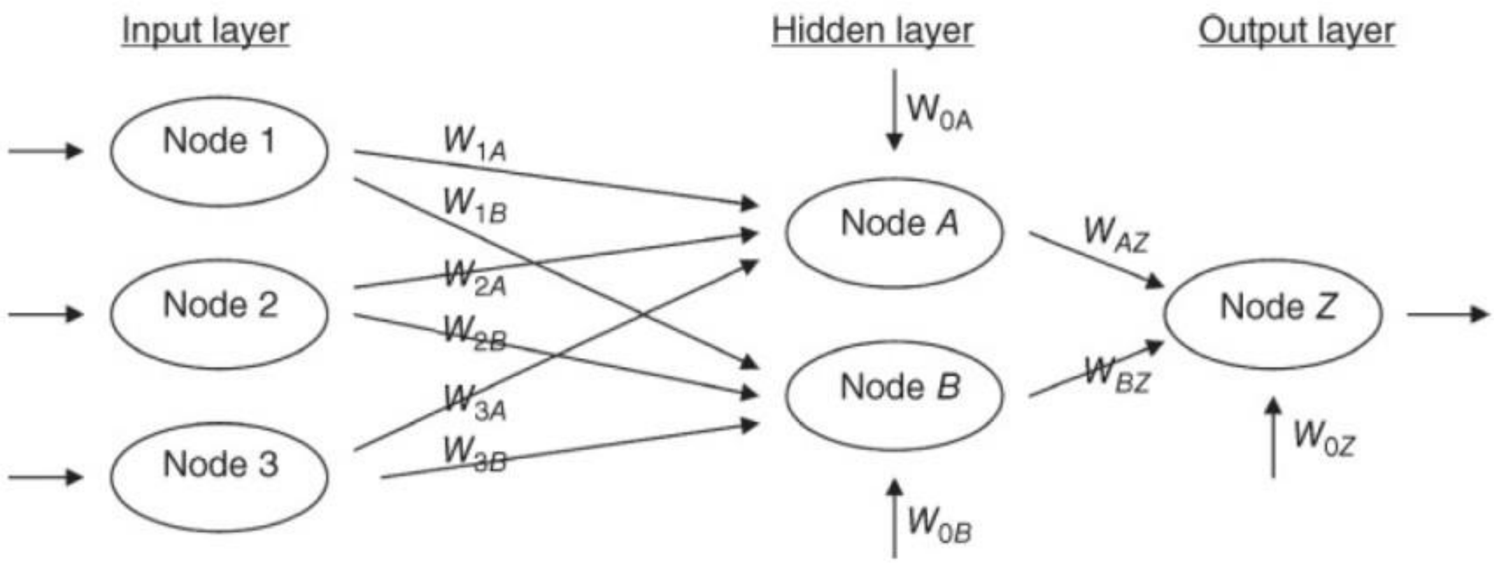
심층 순방향 신경망이라 부름

노드



각 노드는 하나의 퍼셉트론으로 이해 할 수 있으며
이전 노드의 모든 출력값을 입력으로 받아 출력값을 계산함

노드



각 레이어의 노드 출력값을 잠재변수로 생각할 때,

레이어가 깊어질 수록 잠재변수로 다시 잠재변수를

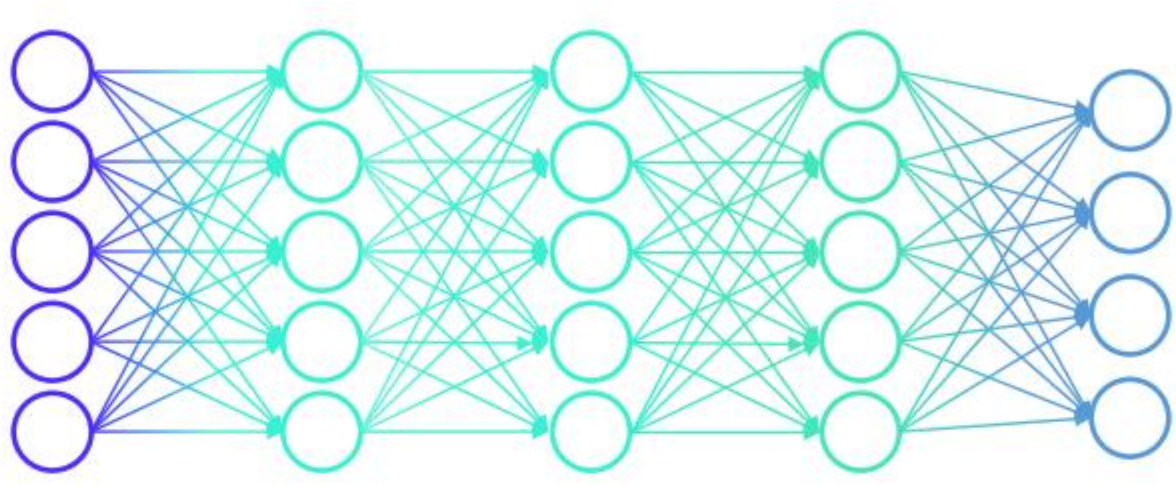
설명해야 하기 때문에 각 노드에 대한 해석이 불가능함

Black Box 모델

3

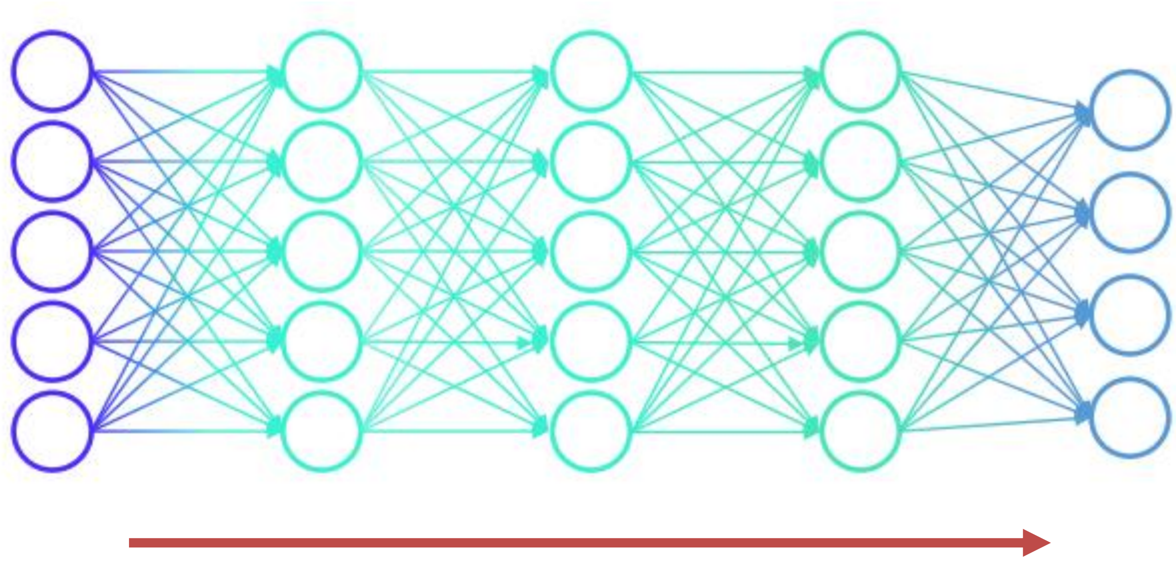
신경망

신경망



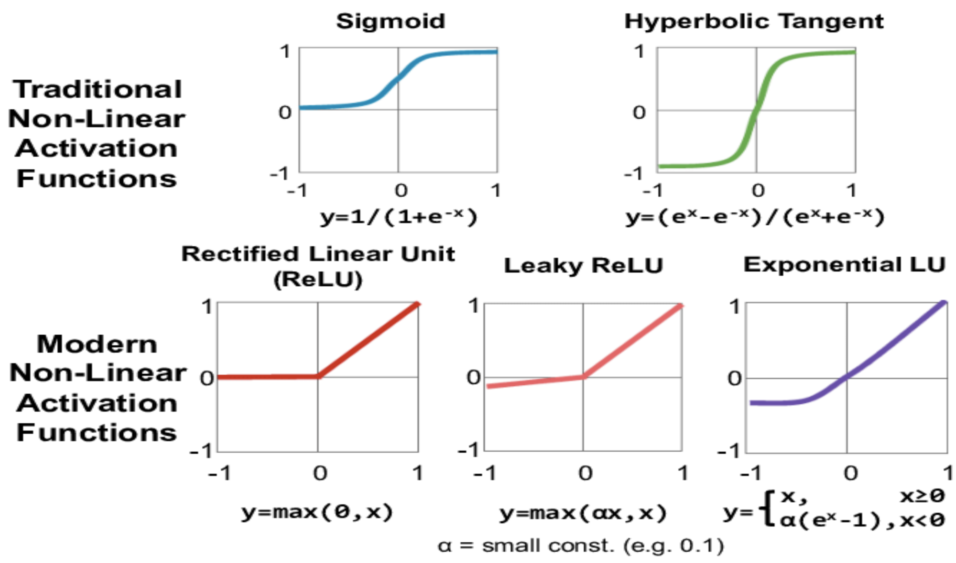
신경의 기본단위인 뉴런이 신호를 보내는 방식에서
영감을 얻어 만들어진 알고리즘

순전파



신경망을 따라 입력층부터 출력층까지 차례대로
변수를 계산하고 추론하는 과정

활성화 함수

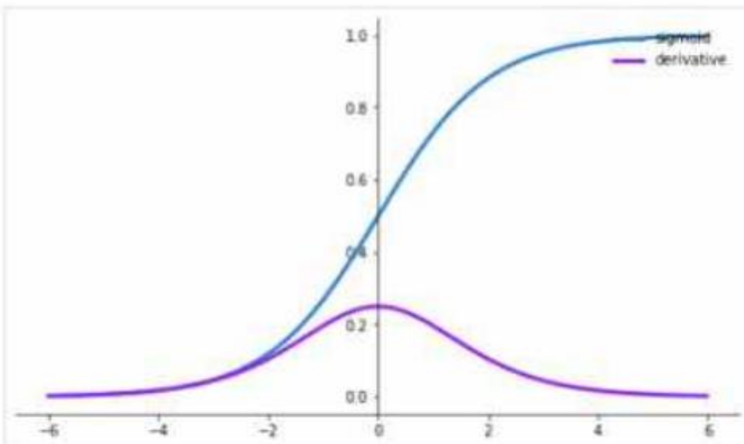


뉴런의 전기신호 수신 과정에서 영감을 얻음

신경망에 비선형성을 부여하는 역할

활성화 함수 종류

시그모이드 함수



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

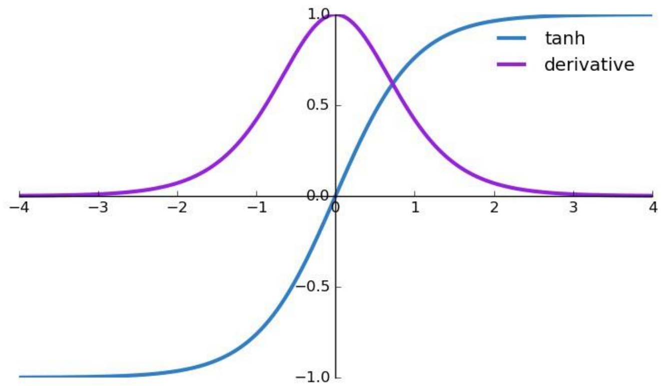
기울기로 0과 0.25사이의 값을 가짐

기본적인 활성화 함수

미분이 가능하여 역전파 가능

활성화 함수 종류

하이퍼볼릭 탄젠트 함수



$$tanh(x) = 2\sigma(2x) - 1$$

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

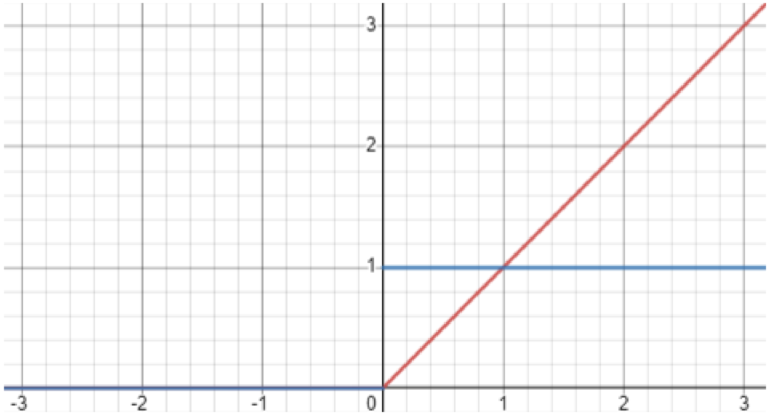
기울기로 0과 1사이의 값을 가짐

입력값이 커지면 기울기가 0에 수렴

지수함수 계산으로 연산이 느리다는 단점

활성화 함수 종류

ReLu 함수



ReLu

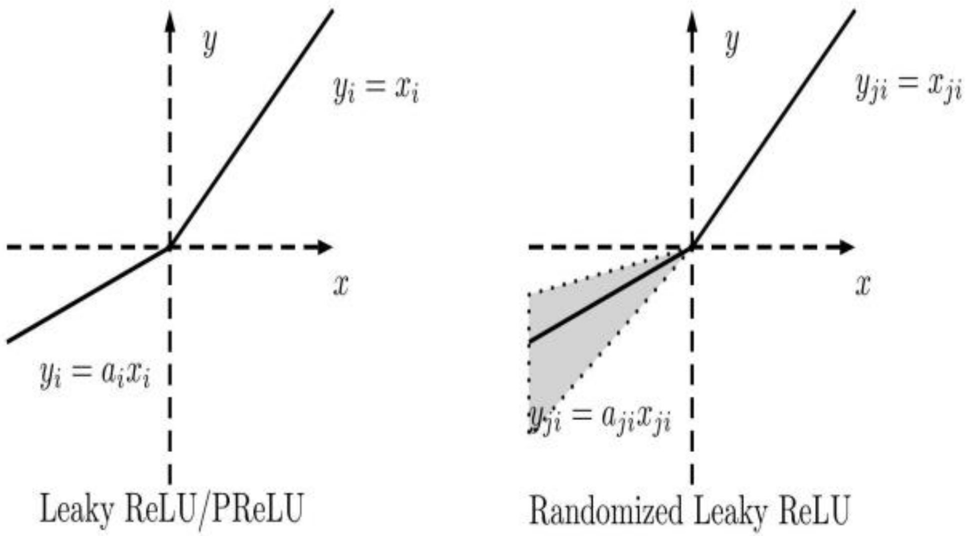
$$f(x) = \max(0, x)$$

은닉층에 주로 사용되며 계산이 쉽고 기울기 소실 문제를 어느정도 해결 하였으나 Knockout 문제가 있음

음수 부분에서 항상 기울기가 0…

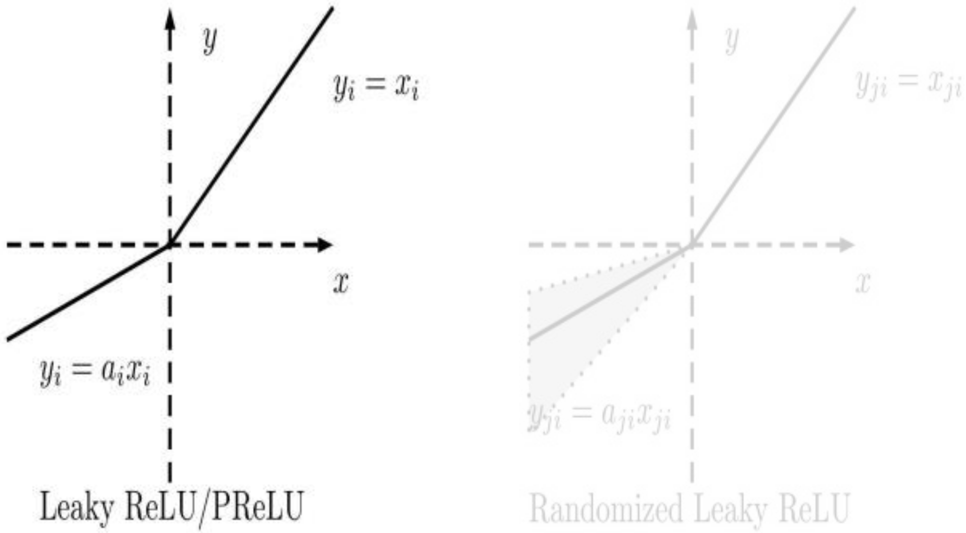
활성화 함수 종류

Leaky / Parametric / Randomized PReLU



활성화 함수 종류

Leaky / Parametric / Randomized PReLU



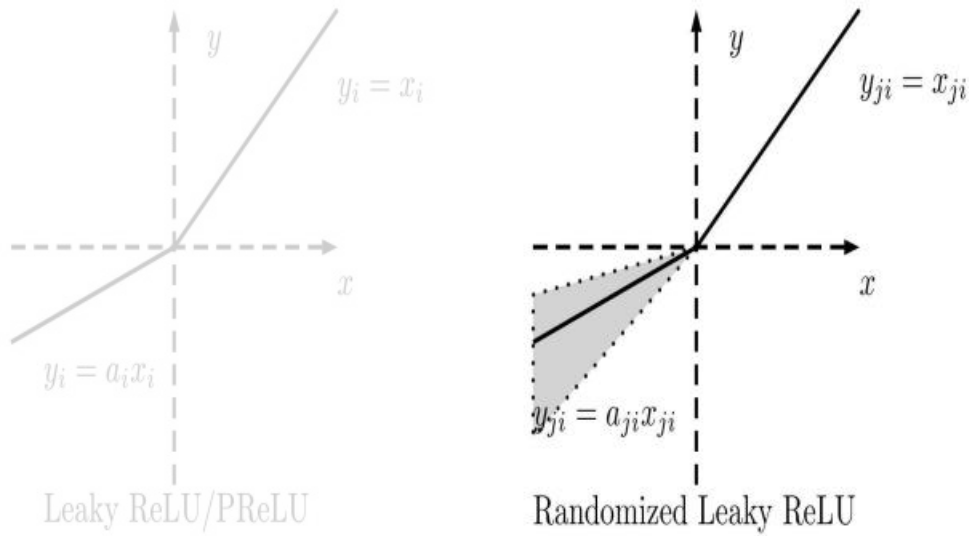
Leaky ReLU/ PReLU

ReLU의 Knockout 문제를 해결한 활성화 함수

PReLU 일 경우 기울기 a_i 를 직접학습함

활성화 함수 종류

Leaky / Parametric / Randomized PReLU



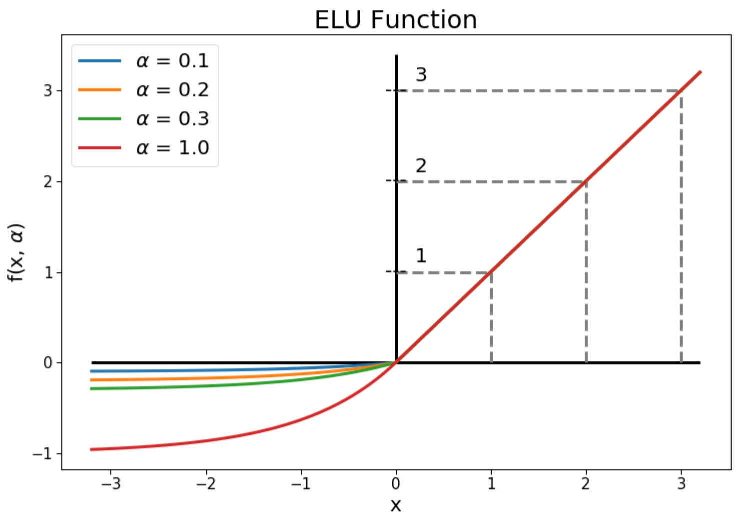
Randomized PReLU

기울기 a_i 를 노드별로

다르게 초기화 하여 과적합을 방지함

활성화 함수 종류

ELU 함수



$$f(x) = \begin{pmatrix} x & if \quad x > 0 \\ \alpha(e^x - 1) & if \quad x \leq 0 \end{pmatrix}$$

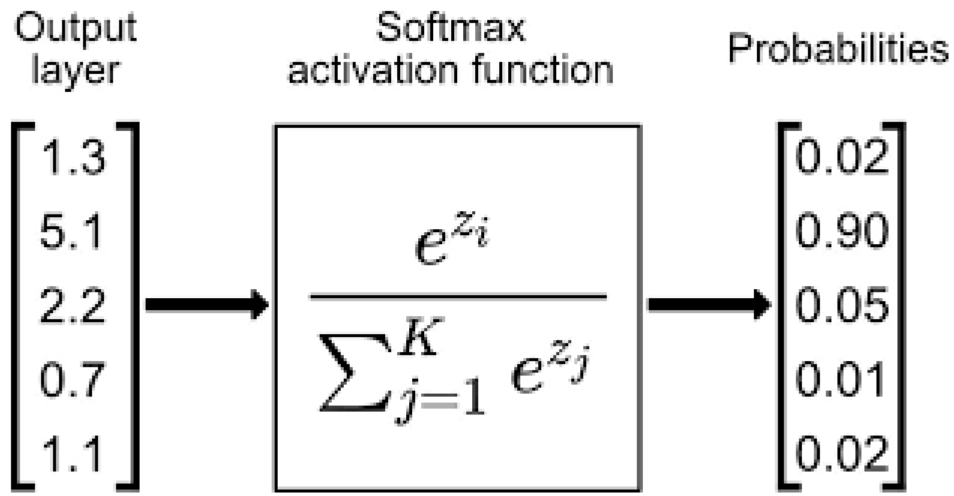
Knockout 문제와 음수부분의 선형성 부분을 해결

하지만 ReLU와 성능이 크게 차이가 나지 않음

CS231N 수업에서는 ReLU->Leaky or ELU 순으로 사용할 것을 권장

활성화 함수 종류

Soft Max 함수



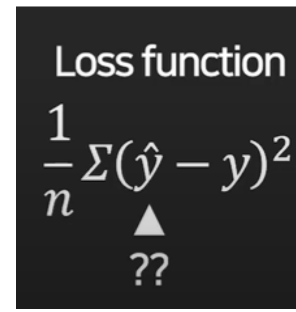
$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

출력층에서 활용되는 활성화 함수

다중 분류 문제에 사용하고 출력값을 확률로 이용가능

손실 함수

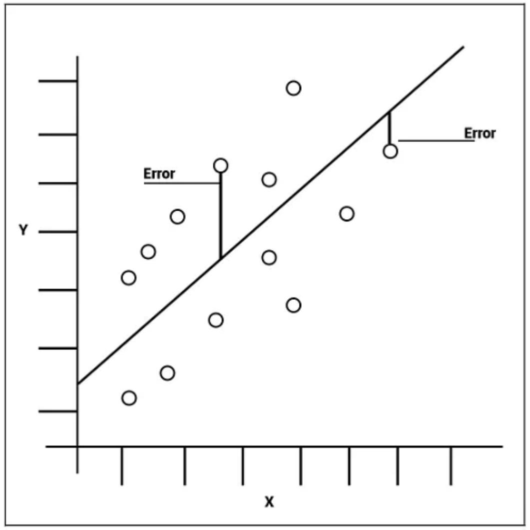
Loss function (Cost function)



역전파 과정에서 예측 결과를 실제 정답과 비교하는 역할

손실 함수 종류

평균 제곱 오차 (Mean Squared Error, MSE)



$$L(\hat{y};\theta) = \frac{1}{2n}\sum_{k=1}^{n}(y_k - \hat{y_k})^2$$

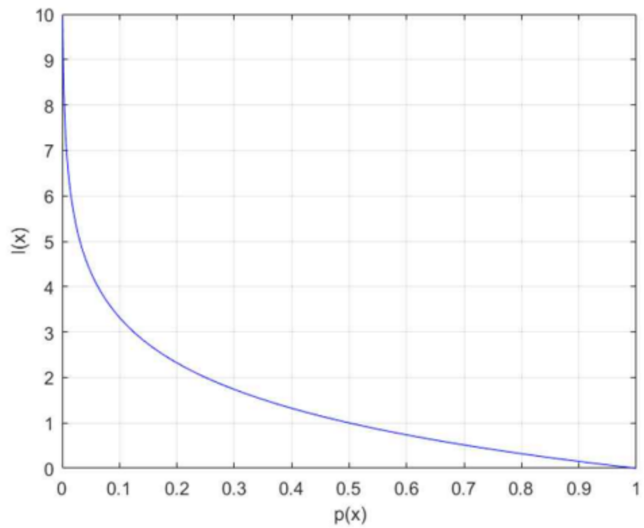
미분했을 때 계산의 편의성을 위한 처리

회귀문제에서 주로 사용되고

실제값과 차이가 클수록 오차의 크기가 커짐

손실 함수 종류

엔트로피 (Entropy)



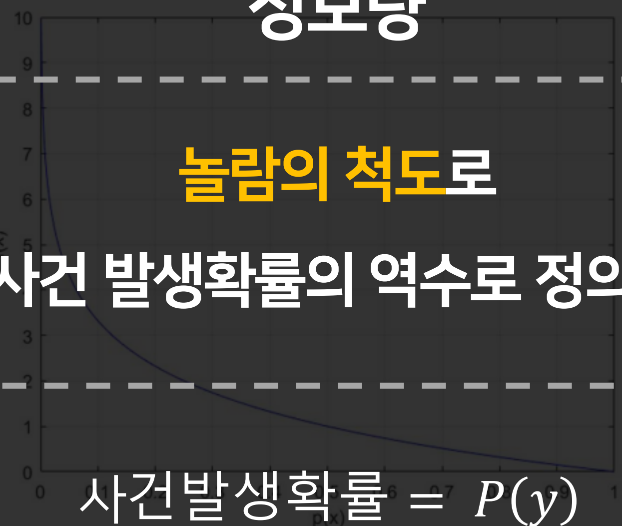
엔트로피란 사건이 가진 정보량을 평균화한 척도로

사건이 발생할 확률이 적을 수록 큰 정보량을 지님

손실 함수 종류

엔트로피 (Entropy)

정보량



놀람의 척도로

사건 발생확률의 역수로 정의됨

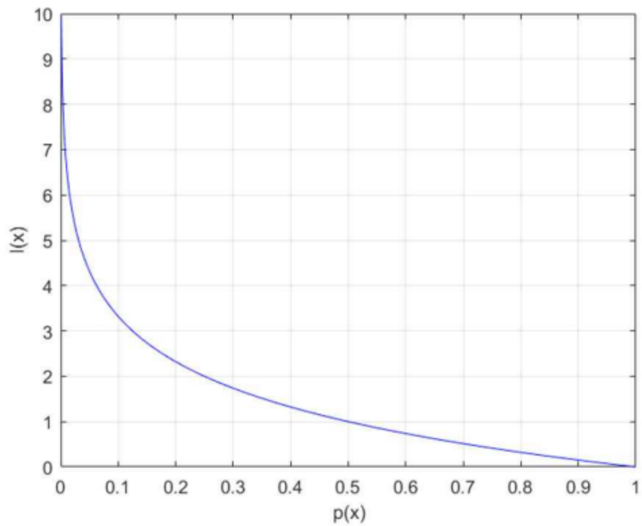
$$\text{사건발생확률} = P(y)$$

$$I(y) = \log\left(\frac{1}{P(y)}\right) = -\log(P(y))$$

엔트로피란 사건이 가진 정보량을 평균화한 척도로
사건이 발생할 확률이 작을 수록 큰 정보량을 지님

손실 함수 종류

교차 엔트로피 오차 (Cross Entropy Loss)



$$L(\hat{y};\theta) = -\sum_{i=1}^{output\ size} y_i \ log\hat{y}_i$$

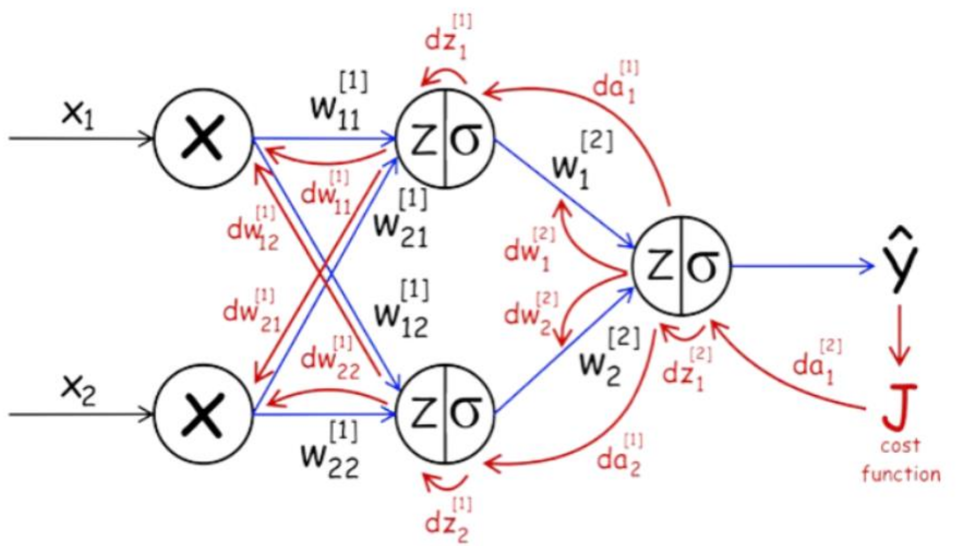
분류문제에서 주로 사용되고

실제값과 차이가 클수록 오차의 크기가 커짐

3 신경망

역전파

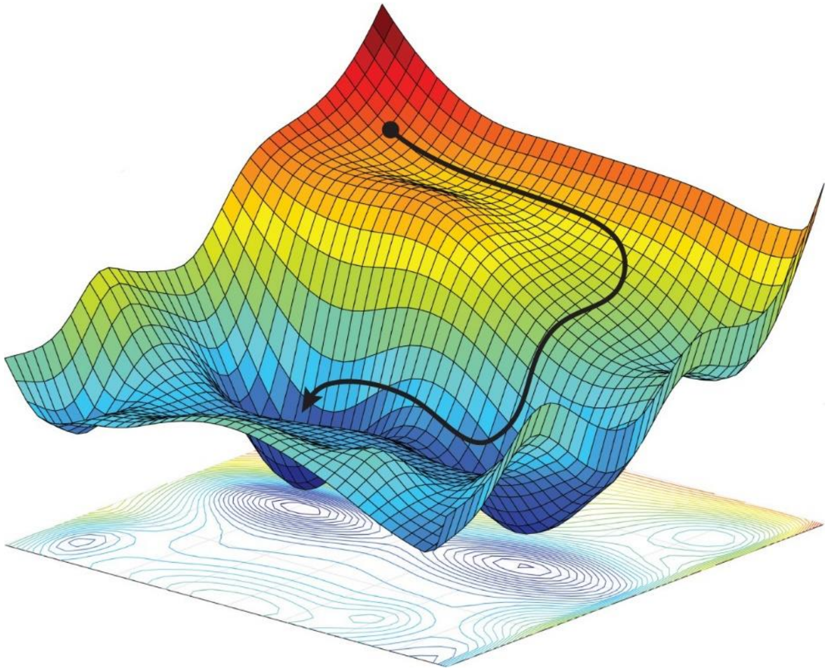
Back Propagation



예측 결과의 오차를 역방향으로

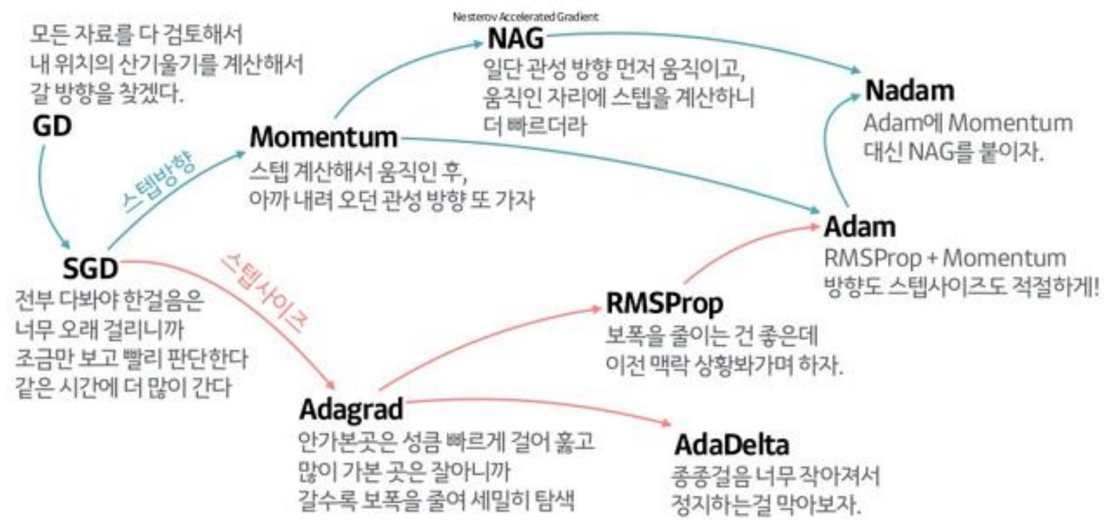
전달하여 가중치와 편향을 업데이트하는 방법

Optimizer



최적화를 수행하는 여러가지 방법들을 통틀어 이르는 말

Optimizer 종류

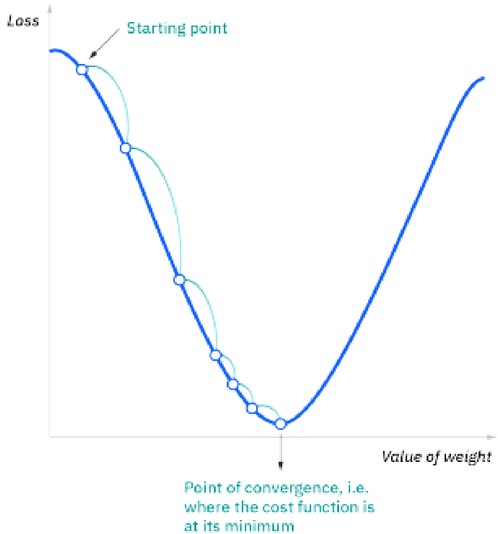


출처 : https://www.slideshare.net/yongho/ss-79607172

여러가지 종류가 있으면 각 방법 마다 장단점이 존재

Optimizer 종류

경사 하강법 (Gradient Descent)



$$W \leftarrow W - \alpha \frac{\partial L}{\partial w}$$

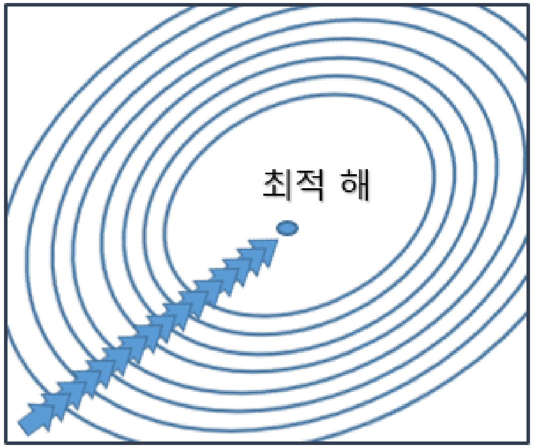
$$b \leftarrow b - \alpha \frac{\partial L}{\partial w}$$

기울기를 이용하여 가중치를 업데이트 하는 방법

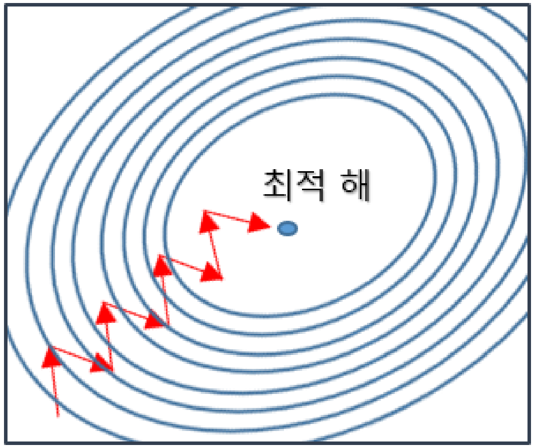
손실함수의 값이 최소가 되면 학습을 종료함

Optimizer 종류

확률적 경사 하강법 (Stochastic Gradient Descent)



경사 하강법



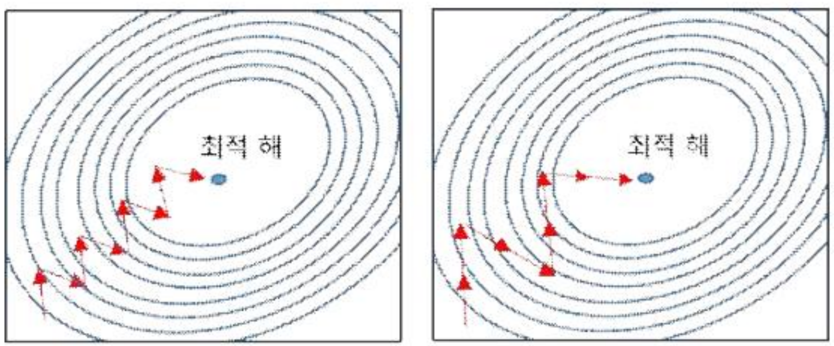
확률적 경사 하강법

데이터의 일부만 이용하여 경사하강법을 진행하는 방식

효율성과 속도 측면에서 뛰어남

Optimizer 종류

모멘텀 (Momentum)



확률적 경사 하강법 모멘텀

관성항

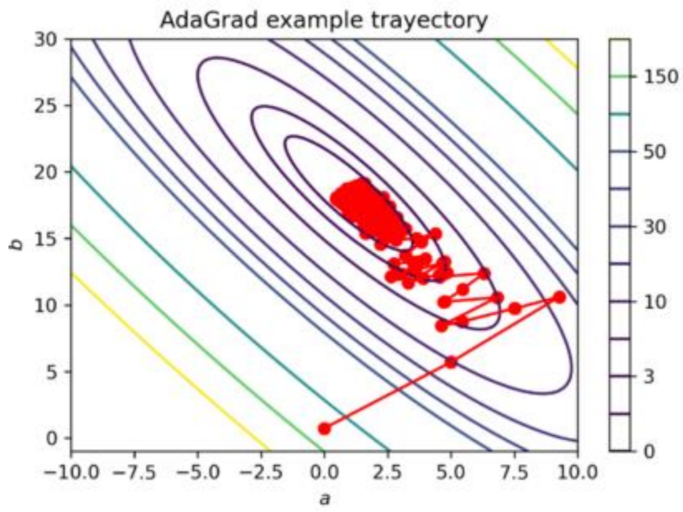
$$V(t) = m \times V(t-1) - \eta \frac{\partial}{\partial w} Loss(w)$$
$$W(t+1) = W(t) + V(t)$$

확률적 경사하강법에 관성의 개념을 도입

Local minimum 문제를 해결 하는데 도움이 될 수 있음

Optimizer 종류

AdaGrad



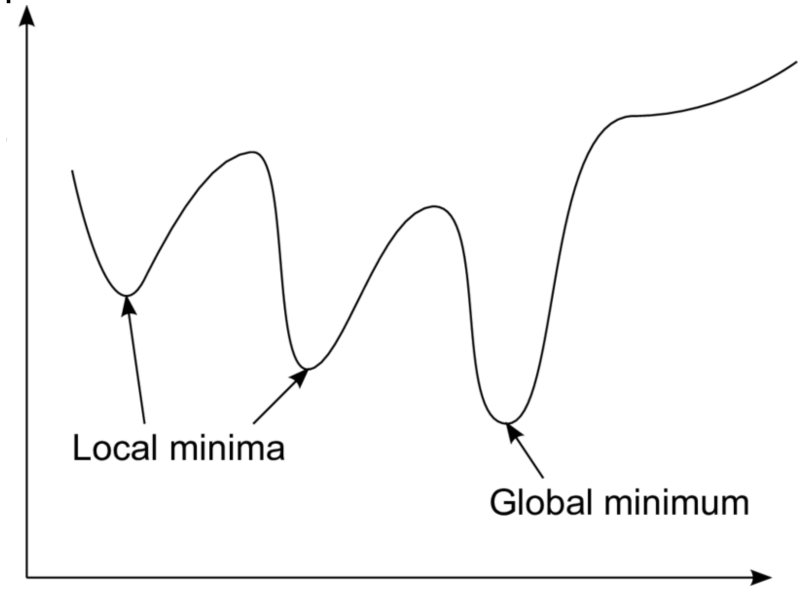
$$G(t) = G(t-1) + \left(\frac{\partial}{\partial w(t)} Loss(w(t))\right)^2$$
$$= \sum_{i=0}^{t} \left(\frac{\partial}{\partial w(i)} Loss(w(i))\right)^2$$

$$W(t+1) = W(t) - \eta \times \frac{1}{\sqrt{G(t) + \epsilon}} \times \frac{\partial}{\partial w(t)} Loss(w(t))$$

가중치 업데이트에 따라 학습률을 조절하여 효율적이지만
학습이 오래 진행될수록 조절값이 0에 수렴하는 문제가 있음

Optimizer에서 발생하는 문제

Local minimum

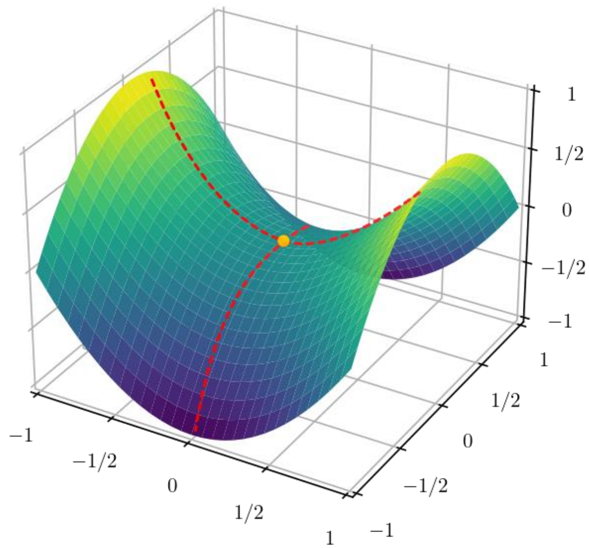


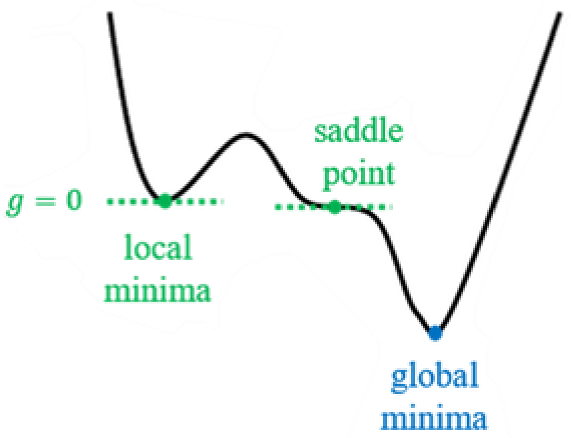
국소 지점이 여러 곳에 존재할 때

Global minimum이 아닌 지점에서도 학습 종료

Optimizer에서 발생하는 문제

Saddle Point





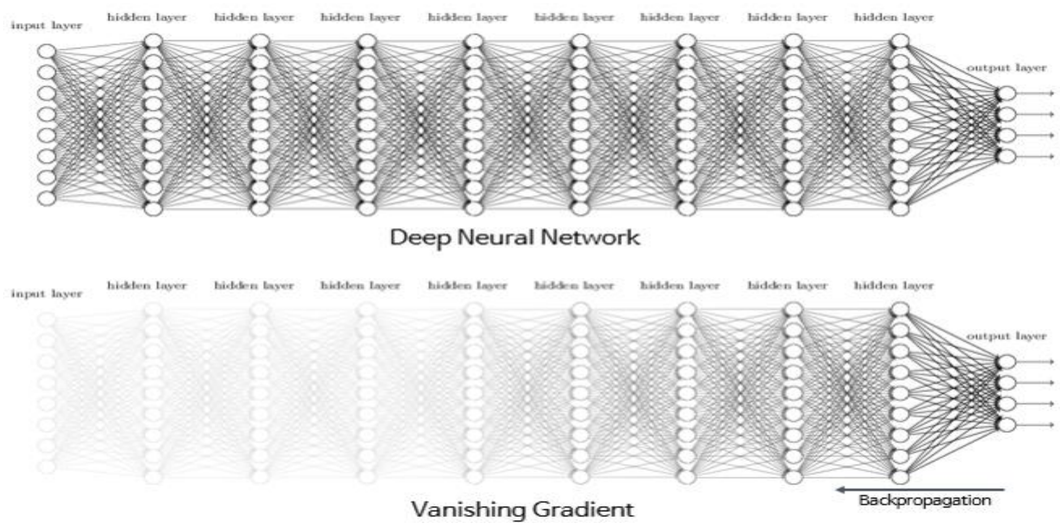
주로 고차원 함수에서,

극소 지점이 아닌 변곡점에서도 학습 종료

3 신경망

Optimizer에서 발생하는 문제

기울기 소실 문제 (Gradient Vanishing Problem)

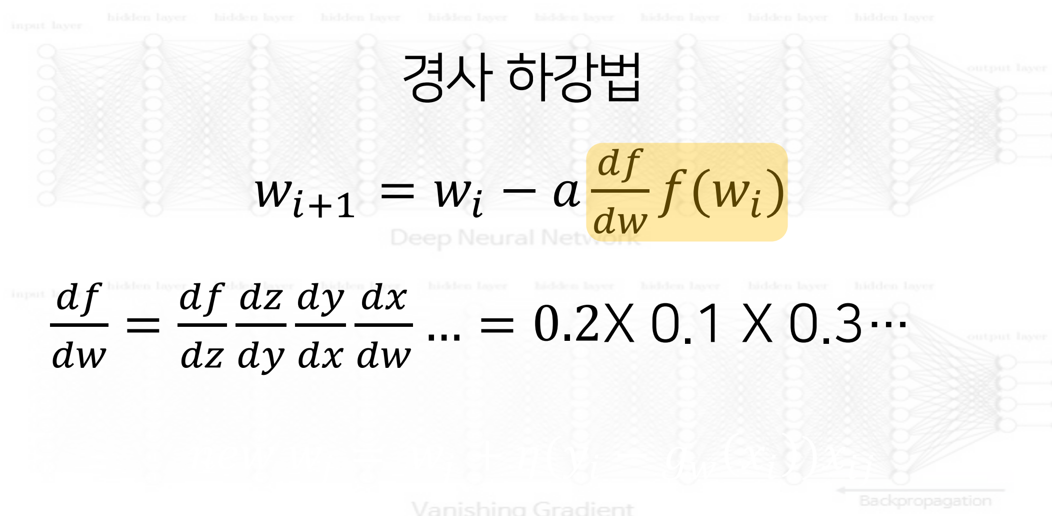


역전파 과정에서 기울기가 점점 0에 수렴하여,

경사 하강법을 더 이상 이용할 수 없게 되는 문제

Optimizer에서 발생하는 문제

기울기 소실 문제 (Gradient Vanishing Problem)



경사 하강법

$$w_{i+1} = w_i - a\frac{df}{dw}f(w_i)$$

$$\frac{df}{dw} = \frac{df}{dz}\frac{dz}{dy}\frac{dy}{dx}\frac{dx}{dw}\ldots = 0.2 \times 0.1 \times 0.3\cdots$$

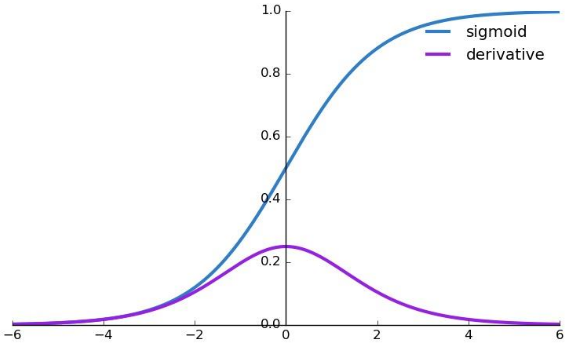
역전파 과정에서 기울기가 계속해서 곱해지다 보면 점점 0에 가까워짐

역전파 과정에서 기울기가 점점 0에 수렴하여, 경사 하강법을 더 이상 이용할 수 없게 되는 문제

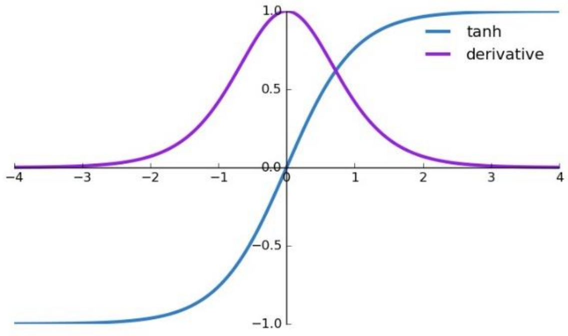
Optimizer에서 발생하는 문제

기울기 소실 문제 (Gradient Vanishing Problem)

sigmoid



tanh



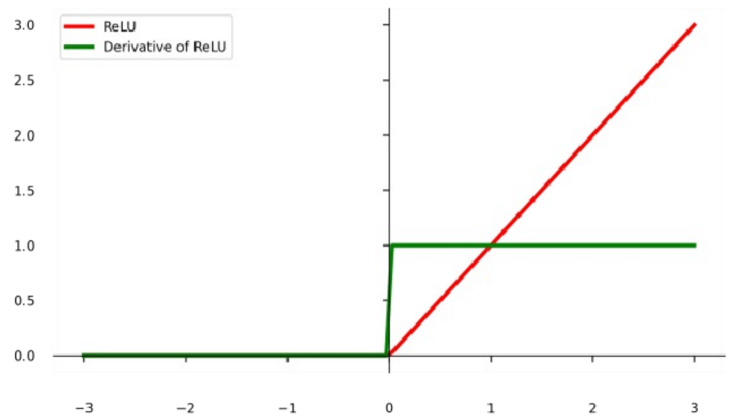
Sigmoid와 tanh 함수는 도함수 최대값이

각각 0.25, 1로 기울기 소실문제가 발생 할 수 있음

Optimizer에서 발생하는 문제

기울기 소실 문제 (Gradient Vanishing Problem)

ReLU



ReLU함수는 미분값이 0 또는 1로 나타나기 때문에

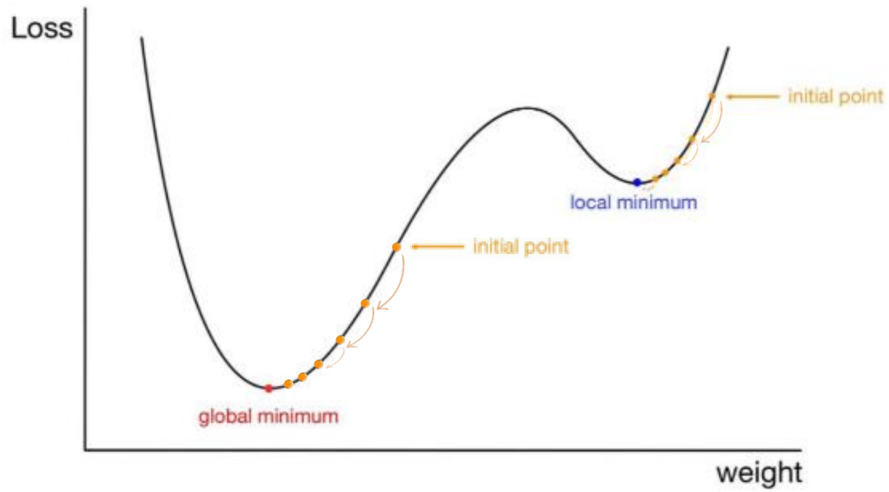
기울기 소실 문제를 상당 부분 극복했다고 할 수 있음

4

성능향상기법

가중치 초기화

weight initialization



초기 값에 따라 수렴속도, 도달하는 지점이 다름

학습 시작 시점의 가중치 설정하는 것이 중요함

가중치 초기화

Zero initialization

파라미터의 가중치를 0으로 초기화

역전파를 통해 갱신할 시 가중치가 같은 값으로 변함

Random initialization

파라미터 값을 정규분포 혹은 균일분포를 이용하여 초기화

Sigmoid 사용 시 값이 치우쳐 기울기 소실 문제

가중치 초기화

Xavier initialization

$\frac{2}{\sqrt{n+m}}$ 을 표준편차로 하는

정규분포로 초기화

He initialization

$\frac{2}{\sqrt{n}}$ 을 표준편차로하는

정규분포로 초기화

n : 이전 층의 노드 수

m : 현재 층의 노드 수

노드 개수를 반영하여 초기화하여

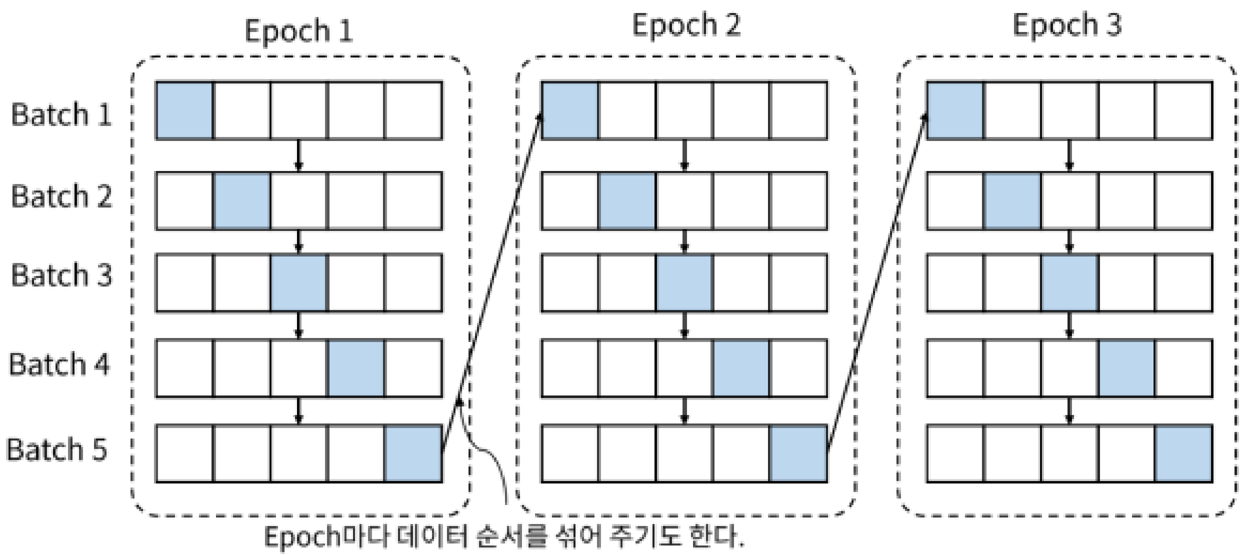
고정된 표준편차를 사용할 때보다 더 강건함

Sigmoid 함수일 때 주로 사용

ReLU 함수일 때 많이 사용하고

기울기 소실 문제 방지

Batch Normalization



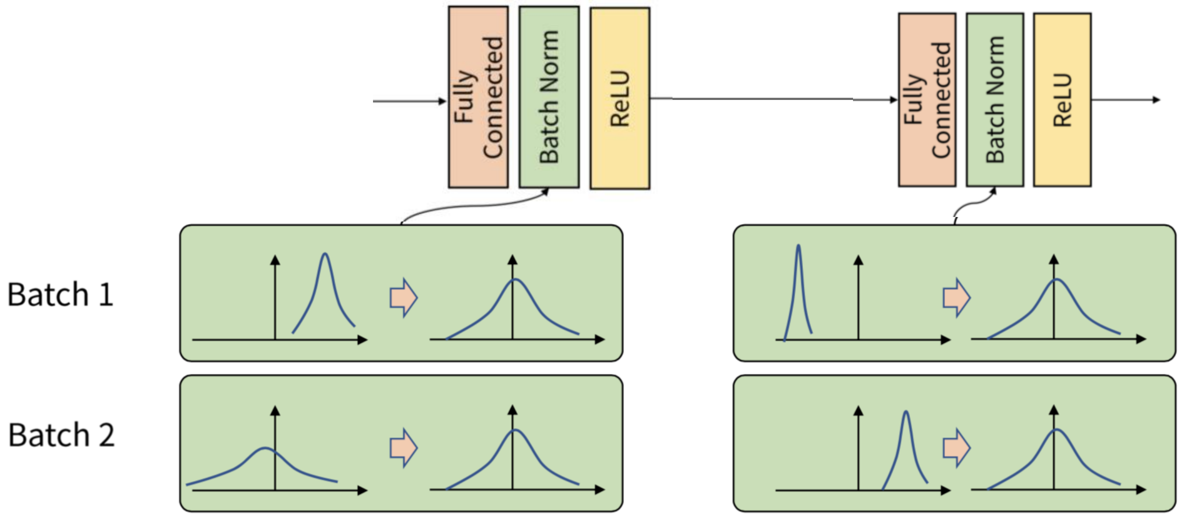
배치 Batch

입력 데이터 셋을 나누는 단위

Batch 크기에 따라 Iteration 결정

1 epoch = batch size × iteration

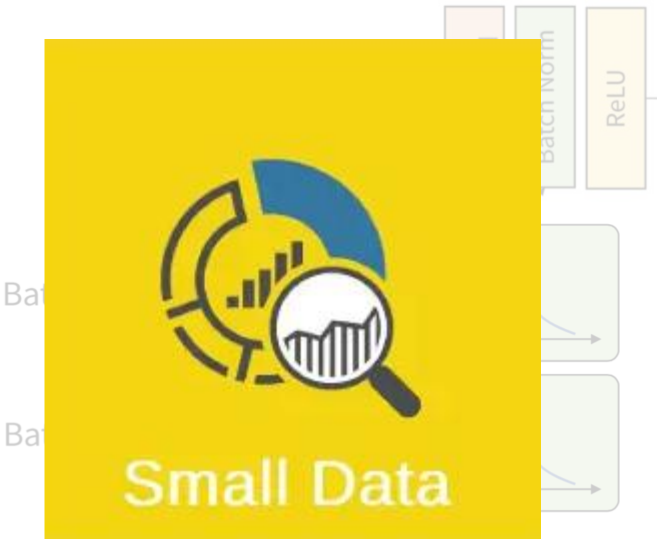
Batch Normalization

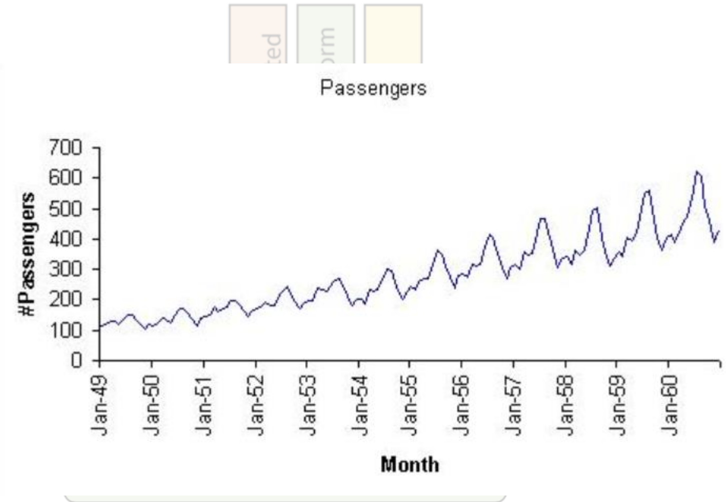


Batch 별 데이터를 평균과 분산을 이용하여 정규화

예측 과정에는 학습 과정에서 얻은 평균과 분산의 평균 이용

Batch Normalization

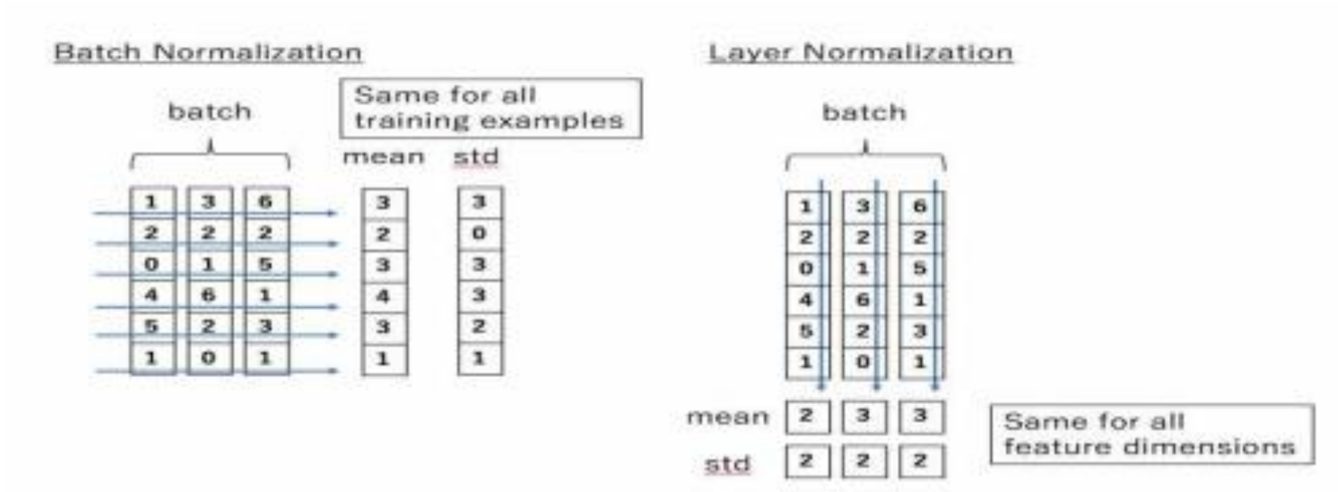




Batch 의 크기가 너무 작거나

Sequential 데이터를 처리하는 경우 적용 시키기 어려움

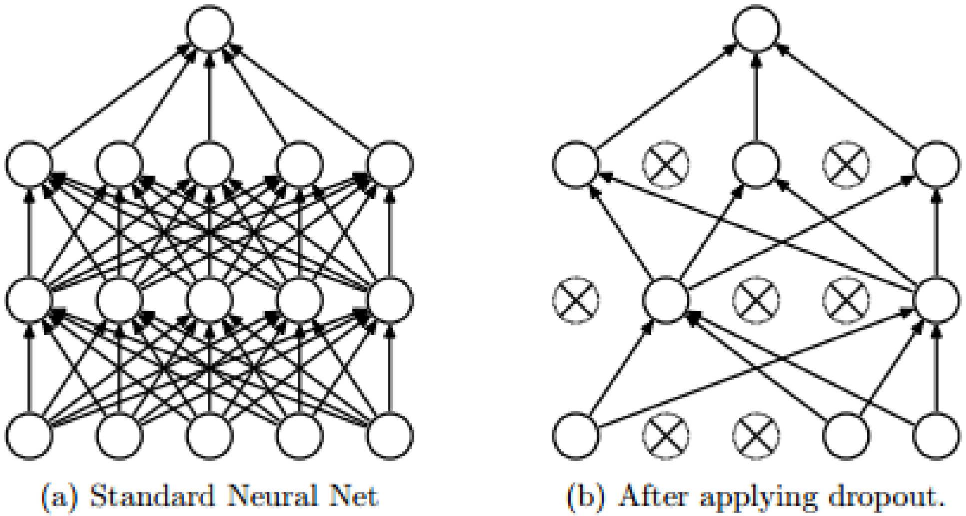
Layer Normalization



Feature 차원에서 정규화를 진행하는 방법

Sequence data에 강건

Dropout



Batch 마다 일정한 비율의 노드를 버리고 학습

앙상블과 유사한 효과를 낼 수 있어 과적합을 방지

THANK YOU