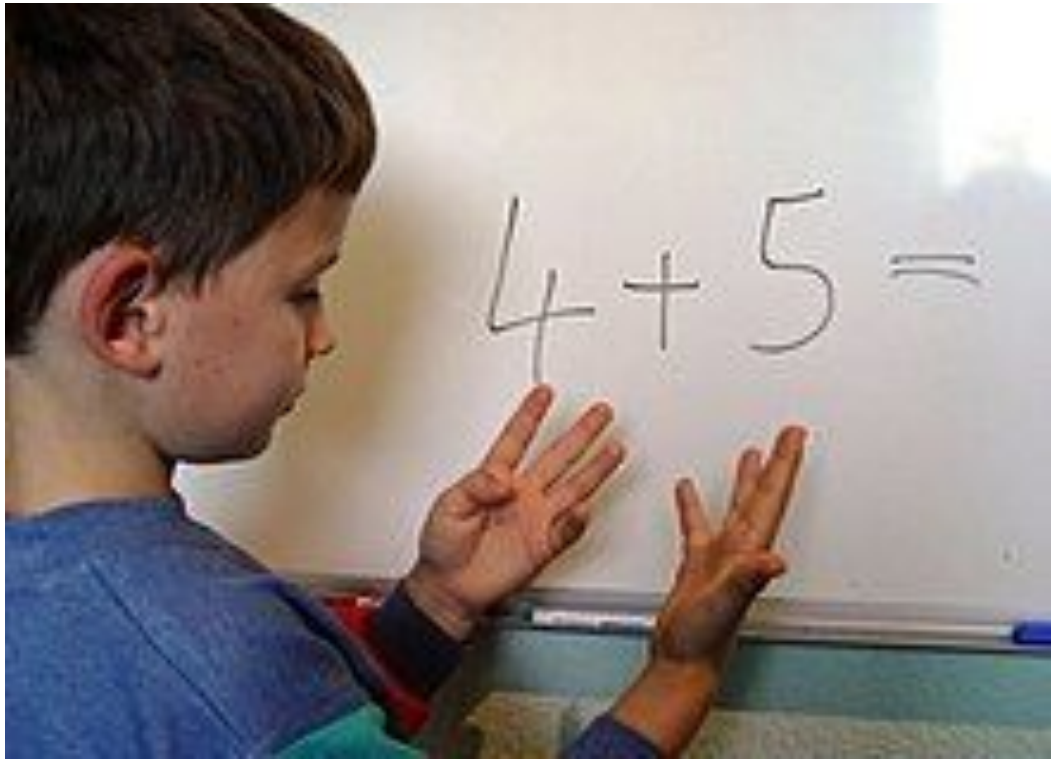
Deep Learning I

김연지
kimyeonji3@gmail.com

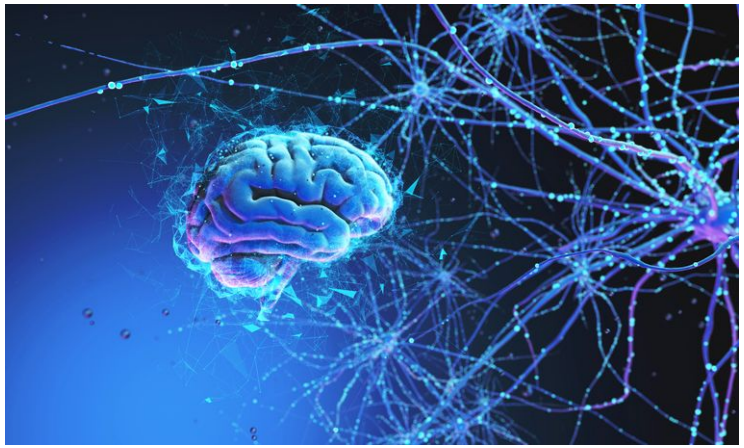
딥러닝(Deep Learning)



인간은 컴퓨터가 아주 짧은 시간에
할 수 있는 계산도 잘 못한다.

컴퓨터는 인간이 쉽게 인지하는
사진이나 음성을 해석하지 못한다.

딥러닝(Deep Learning)



사칙연산도 잘 못하는 인간이
소리, 음성처럼 훨씬 더 복잡한 패턴을
잘 인식하는 이유는?

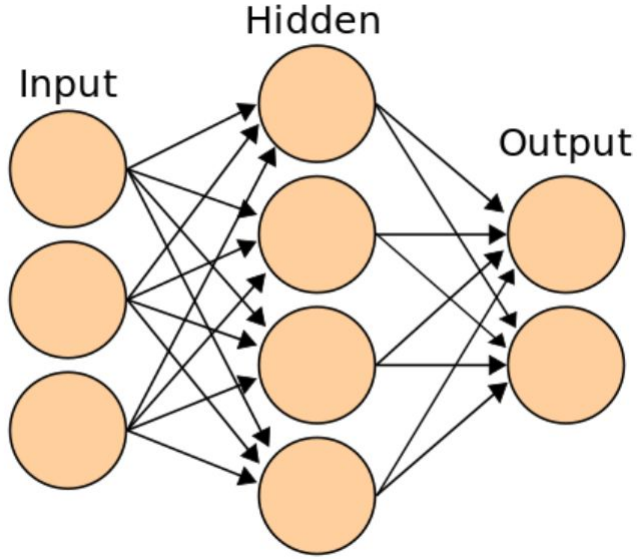
인간의 뇌는 엄청난 수의 뉴런Neuron과 시냅스Synapse로 이루어져 있다.

각각의 뉴런은 큰 연산을 수행하는 능력이 없지만
수많은 뉴런들이 복잡하게 연결되어 병렬연산을 수행하면
컴퓨터가 하지 못하는 음성, 영상인식을 수월하게 할 수 있다.

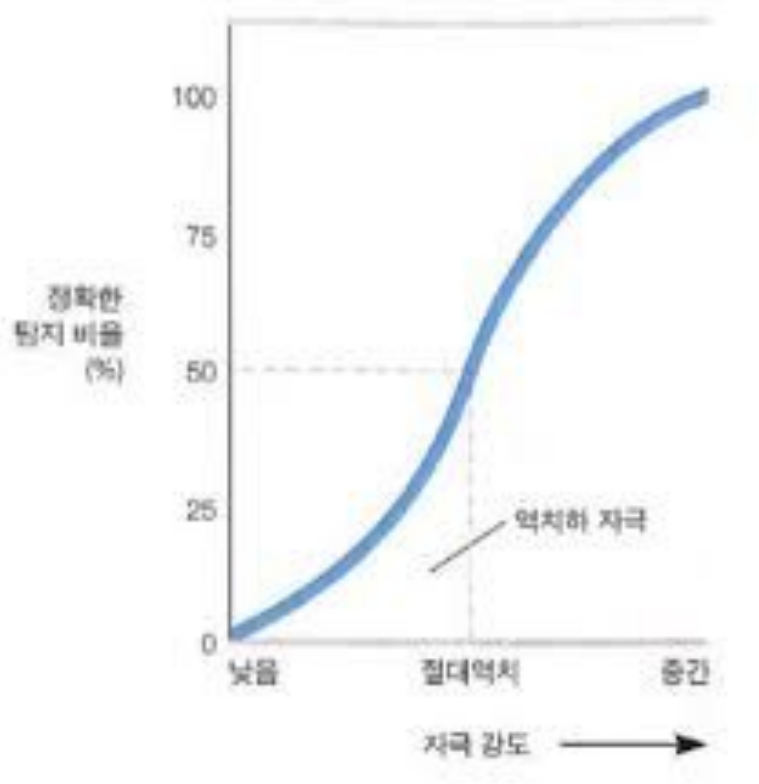
딥러닝은 이 뉴런과 시냅스의 병렬연산을 컴퓨터로 재현하는 방법

딥러닝(Deep Learning)

- 머신러닝의 대표적인 학습법
- 여러 층을 거쳐 점점 추상화 단계로 접어드는 알고리즘 형태
- 패턴을 찾기 위해선, 패턴을 견고하게 만드는 많은 훈련데이터가 필요하다



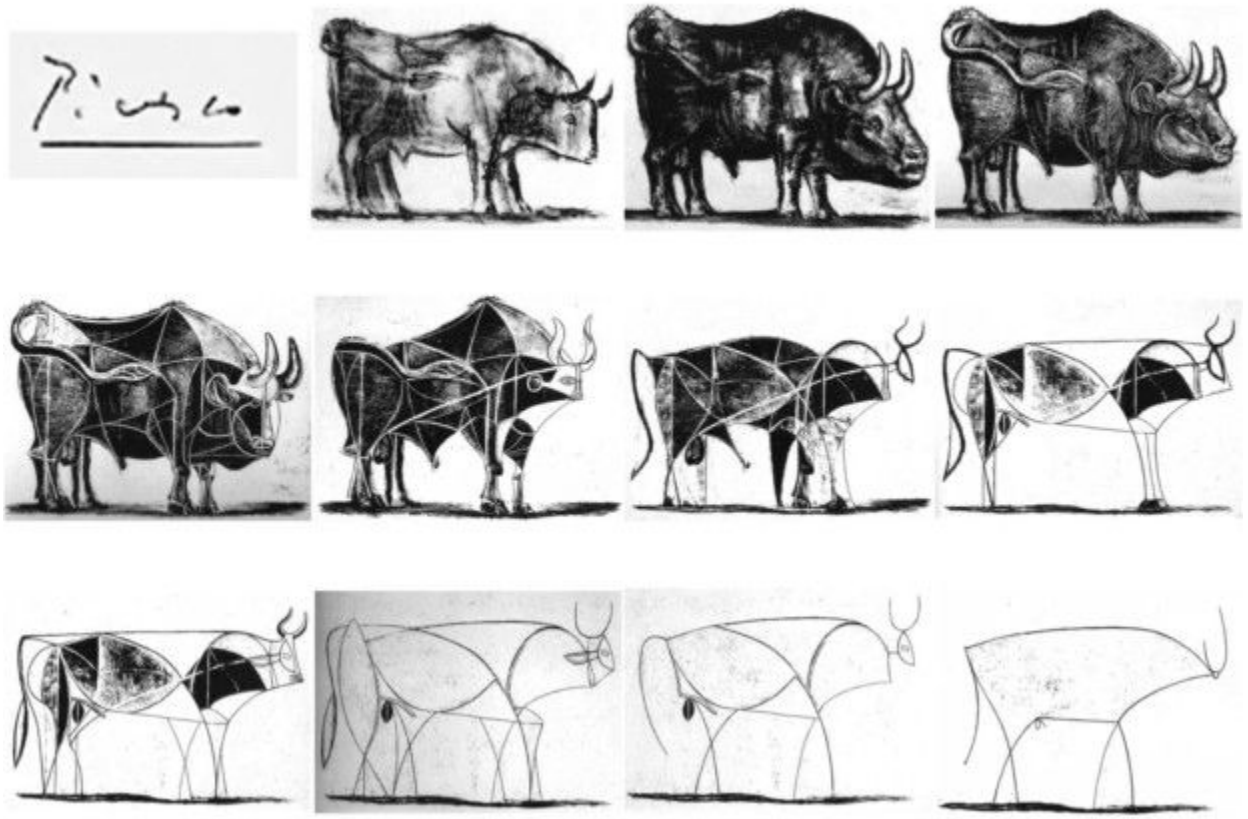
딥러닝(Deep Learning)



역치(threshold) : 생물체가 자극에 대한 반응을 일으키는 데 필요한 최소한도의 자극의 세기를 나타내는 수치. 우리의 몸은 들어오는 모든 자극을 대뇌로 전송하지 않는다. 따라서 일정 강도 이상의 자극이 가해지지 않으면 자극의 변화를 느낄 수 없는데, 그때의 일정 강도치를 역치라고 한다. 물리학에서는 일반적으로 반응이나 기타의 현상을 일으키게 하기 위하여 계(系)에 가하는 물리량의 최소치를 말한다.

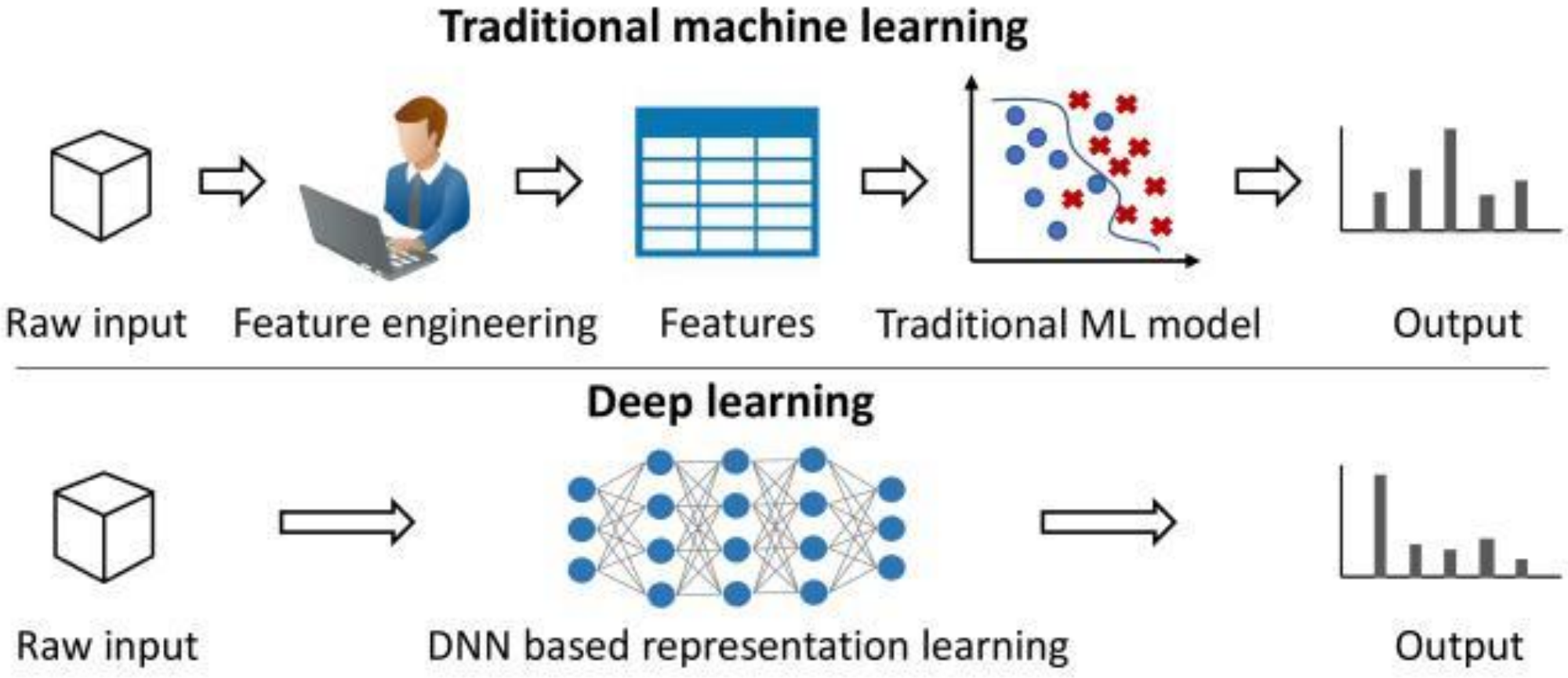
활성화함수와 편향(bias)

딥러닝(Deep Learning)



Bull, Pablo Picasso, 1945
https://medium.com/@youlin.li_31343/painters-machine-learning-and-software-engineering-b124e0d6c22b

딥러닝(Deep Learning)



https://deepai.org/publication/techniques-for-interpretable-machine-learning

딥러닝의 역사

1. **1세대 – 퍼셉트론 (Perceptron)**
   - 인공신경망의 기원
   - 1958년에 제안된 방법으로, n개의 입력을 받아 특정한 연산을 거쳐 하나의 값을 출력하는 방식
   - 1차함수의 형태f(x) = w * x + b를 띄며, 여기에 활성화 함수Activation Function을 적용하여 최종 값을 출력
   - 0 vs 1

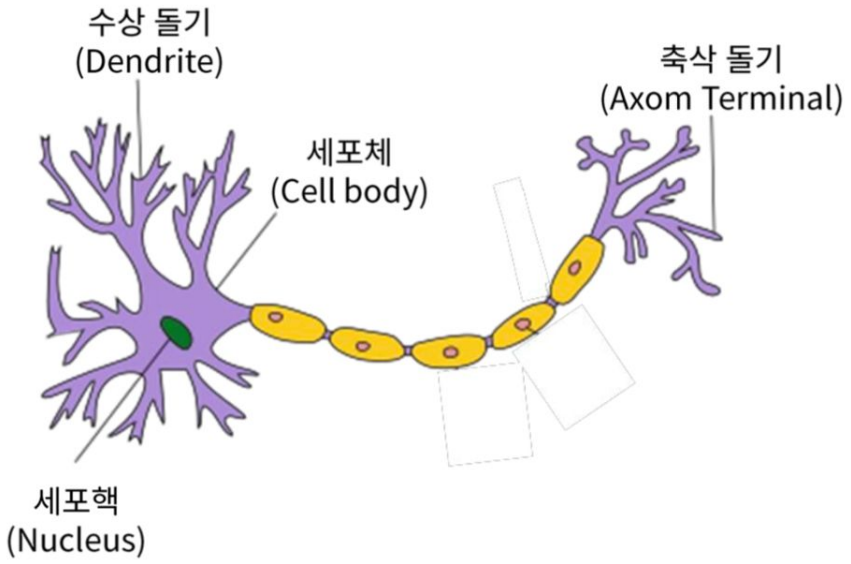
- xor 연산을 학습하지 못하는 문제가 있어 표류

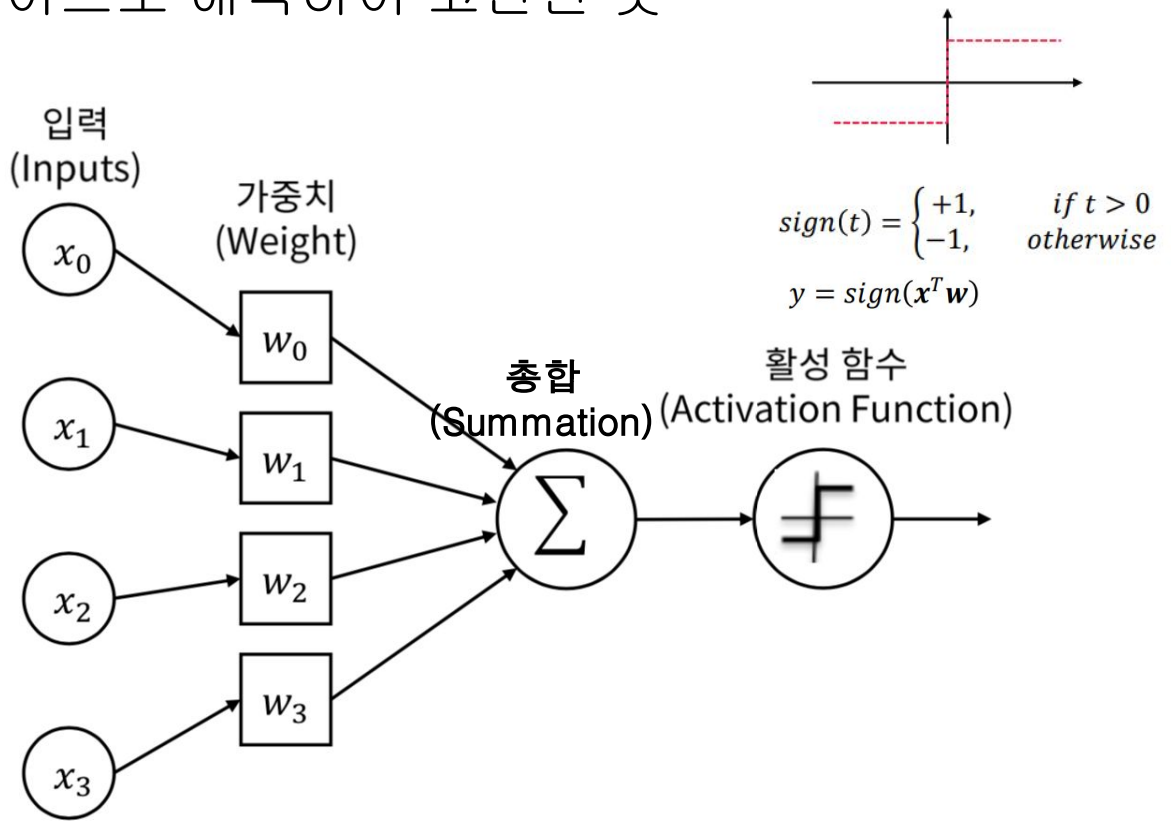
2. **2세대 - 다층 퍼셉트론 (Multylayer Perceptron)**
   - 입력과 출력 사이에 하나 이상의 은닉층hidden layer을 추가해 Xor 문제 해결
   - 그러나 은닉층의 개수가 증가할수록 가중치의 개수도 증가해 학습이 어려워짐
   - 역전파 알고리즘 Backpropagation Algorithm의 등장으로 학습은 가능해졌으나 여러 단점 때문에 침체
   - 단점
     - 많은 레이블Label이 필요
     - 과적합Overfitting으로 인한 성능 저하 우려
     - 기울기 소실Gradient Vanishing
     - 로컬 미니멈Local minimum

3. 3세대 - 비지도학습 – 볼츠만 머신 (Unsupervised Learning - Boltzmann Machine)
   - 다층 퍼셉트론의 단점으로 인해 인공신경망 이론이 잘 활용되지 못하던 와중,
   - 2006년 볼츠만 머신(비지도학습, 즉 레이블 데이터 없이 미리 충분히 학습을 한 이후 기존의 지도학습을 수행)을 이용한 학습방법이 고안됨
   - 이 방법으로 기울기 소실, 과적합 문제를 극복하였고,
     지도학습 이전의 비지도 사전학습Unsupervised pre-train을 통한 올바른 초기값 선정으로 로컬 미니멈 문제도 해결

퍼셉트론

- 로젠블렛이 1958년에 고안한 알고리즘
- 입력 데이터를 2개의 부류중 하나로 분류하는 분류기(Classifier)
- 신경세포를 이진 출력의 단순 논리 게이트로 해석하여 고안한 것





https://compmath.korea.ac.kr/appmath2021/_images/neuron-node1.png

퍼셉트론의 학습방법

- 전체 알고리즘

```
initialize_w(random)
for _ in range(max_iter):
    for x, y in zip(X, Y):
        h = dot_product(x, w)
        y_ = activation_func(h)
        w = w + eta * (y - y_) * x
```

1) 임의로 선을 긋는다

2) 입력을 하나씩 넣어 출력을 낸다

3) 정답과 비교해서 틀린 경우
선을 옮겨 긋는다

- 가중치 업데이트 수식

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta(\boldsymbol{y} - \tilde{\boldsymbol{y}})\boldsymbol{x}$$

$\boldsymbol{w}_{t+1}$: 업데이트 후 가중치

$\boldsymbol{w}_t$: 업데이트 전 가중치

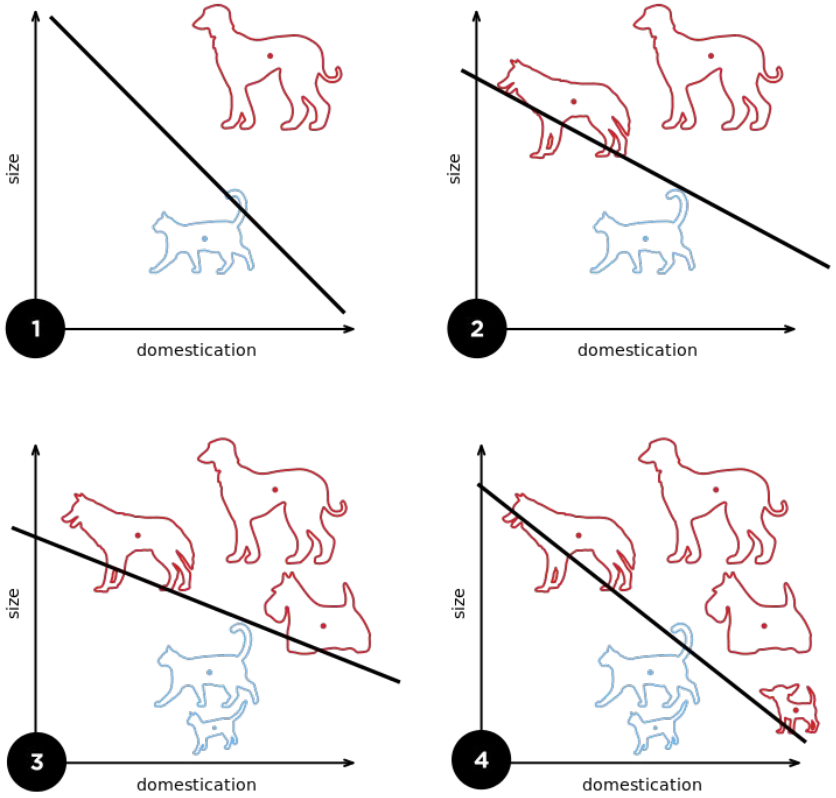
η : 학습률(Learning rate, eta)

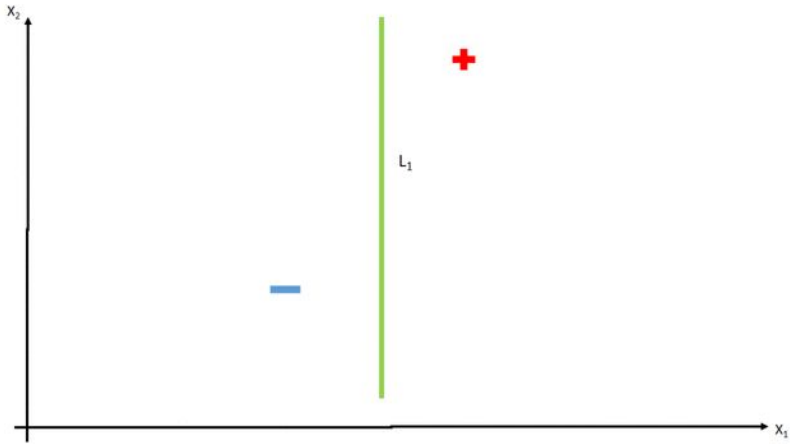
$\boldsymbol{y}$: 학습데이터 정답(Groundtruth)

$\tilde{\boldsymbol{y}}$: 입력에서 추정된 출력(Estimation)

$\boldsymbol{x}$: 입력 데이터

퍼셉트론의 학습방법

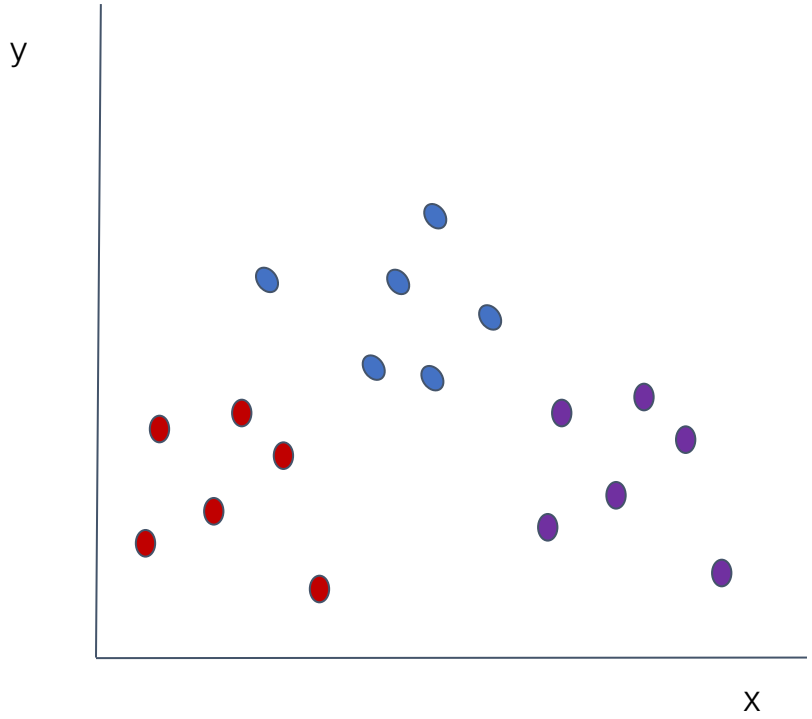




https://upload.wikimedia.org/wikipedia/commons/thumb/c/c4/Perceptron_algorithm.gif/640px-Perceptron_algorithm.gif

퍼셉트론의 이진분류문제

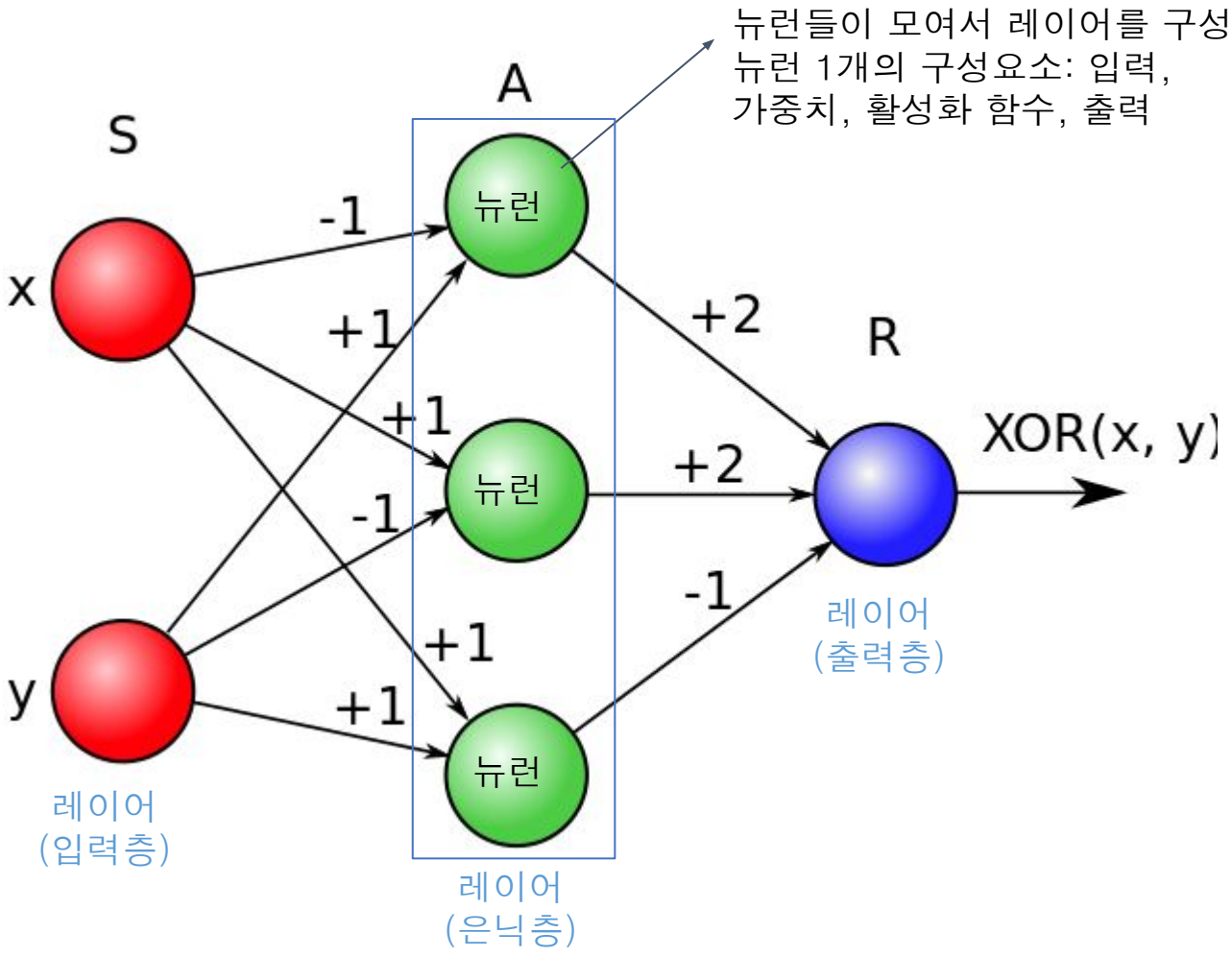
- 이진 출력의 단순 논리 게이트로는 XOR(배타적 논리합) 문제를 풀 수 없다!



입력		출력
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

배타적 논리합(exclusive or)은
수리 논리학에서 주어진 2개의 명제 가운데 1개만 참일 경우를 판단하는 논리 연산이다. 약칭으로 XOR, EOR, EXOR라고도 쓴다.

멀티 퍼셉트론(Multi Perceptron, MLP)



- 입력층 (Input Layer)
  - 신경망에 할당하는 입력 정보를 가져옴
  - 데이터에서 얻은 값을 그대로 출력하는 단순한 유닛

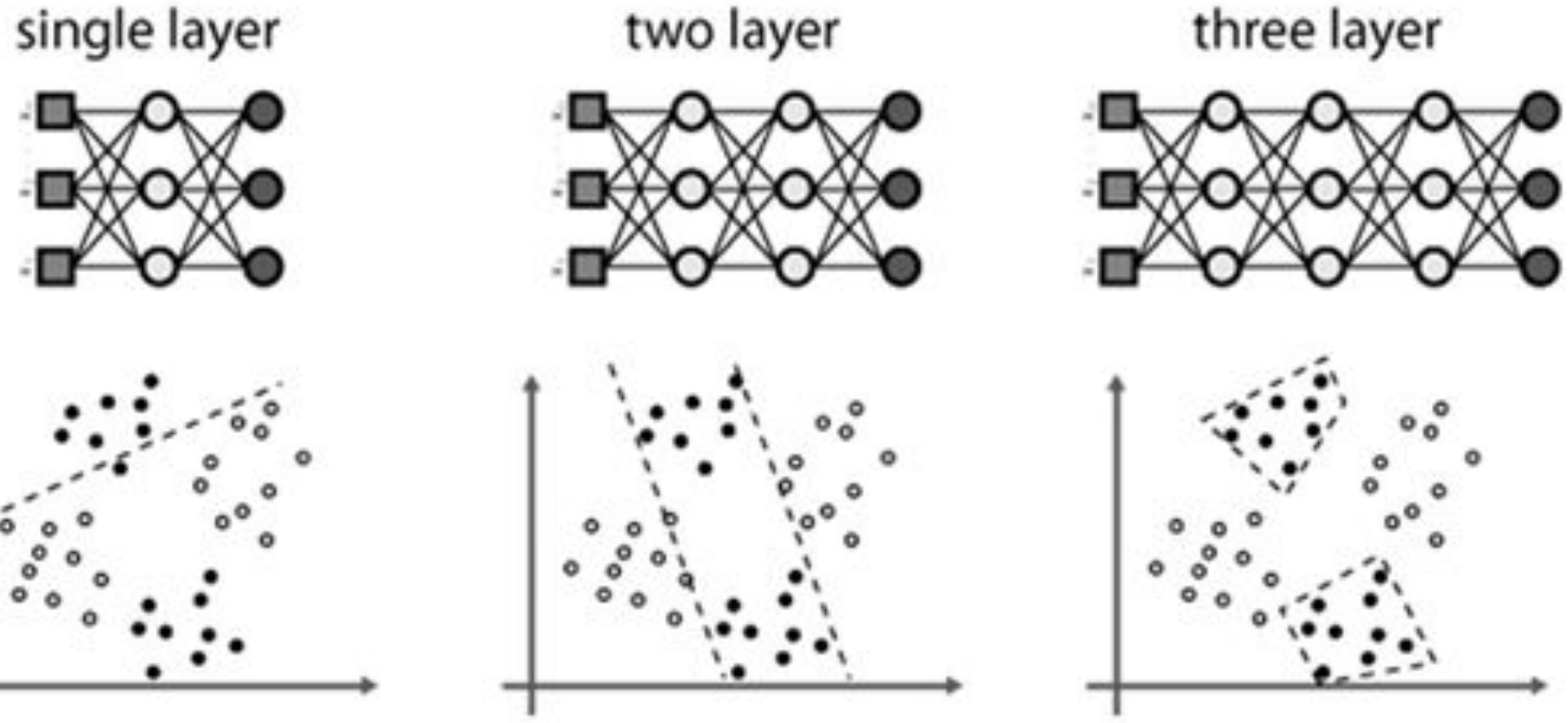
- 은닉층 (Hidden Layer)
  - 신경망에서 실제로 정보를 처리하는 부분

- 출력층 (Output Layer)
  - 중간층과 마찬가지로 처리하면서 신경망에서 계산한 결과를 출력
  - 신경망 전체의 출력이기도 함

https://upload.wikimedia.org/wikipedia/commons/thumb/9/97/Perceptron_XOR_task_v2.svg/640px-Perceptron_XOR_task_v2.svg.png

멀티 퍼셉트론(Multi Perceptron, MLP)

- 기본 신경망 : 입력층, 중간층(은닉층), 출력층의 3층 구조 네트워크로 구성된 기본 구조
- 딥러닝 : 4개층 이상으로 깊은 네트워크 구조로 구성된 구조



멀티 퍼셉트론(Multi Perceptron, MLP)

- 하나의 퍼셉트론

굉장히 단순한 식별기
복잡한 식별 경계를 만들 수 없음
따라서 선형

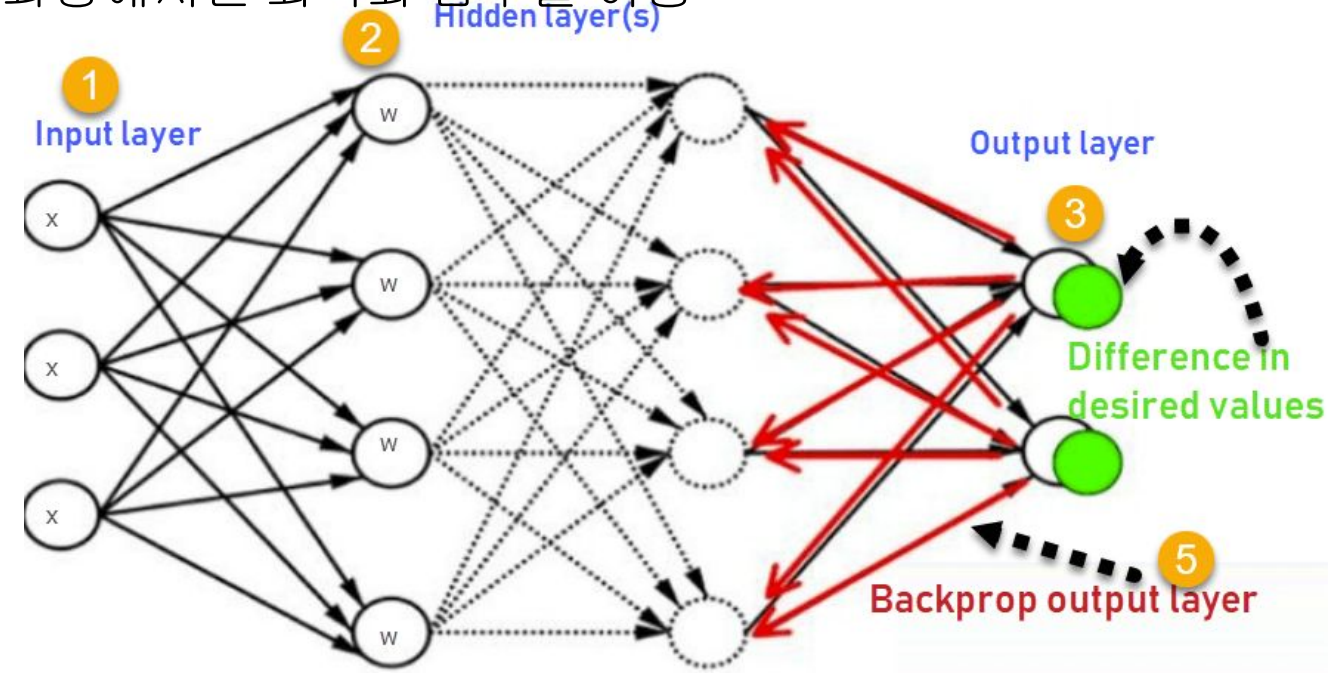
- 다층 퍼셉트론

비선형 퍼셉트론
여러 층으로 쌓아 올린 퍼셉트론을 의미
신경망(Neural Network)이라고도 함

멀티 퍼셉트론(Multi Perceptron, MLP)

역전파(backpropagation)

- 역전파는 오차 역전파법, 오류 역전파 알고리즘 이라고도 하며, 동일 입력층에 대해 원하는 값이 출력되도록 각 계층의 가중치를 조정하는 방법으로 사용
- 즉, 예측값과 실제값의 차이인 오차를 계산하고, 이것을 다시 역으로 전파하여 가중치를 조정
- 이때, 역전파 과정에서는 최적화 함수를 이용

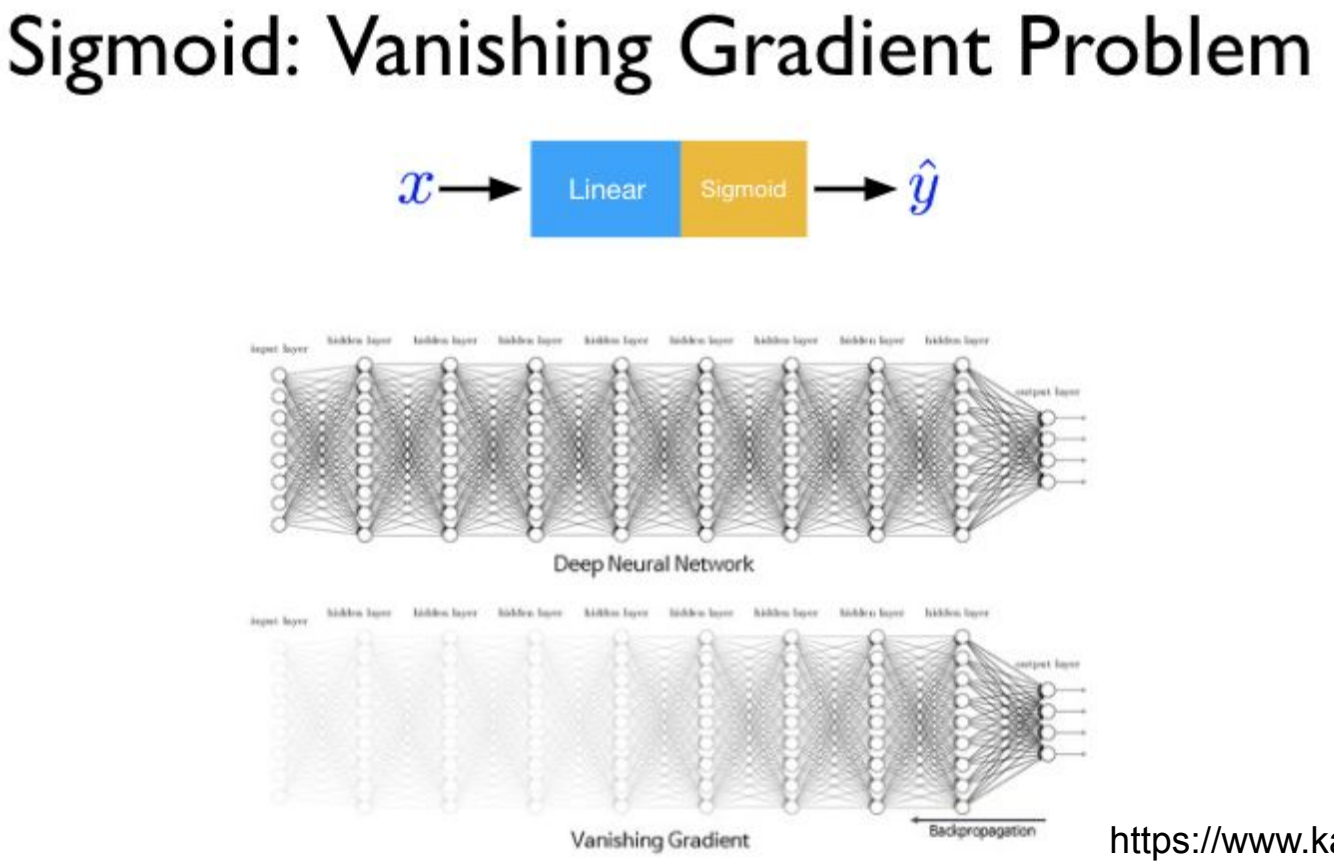


https://www.guru99.com/images/1/030819_0937_BackPropaga1.png

멀티 퍼셉트론(Multi Perceptron, MLP)

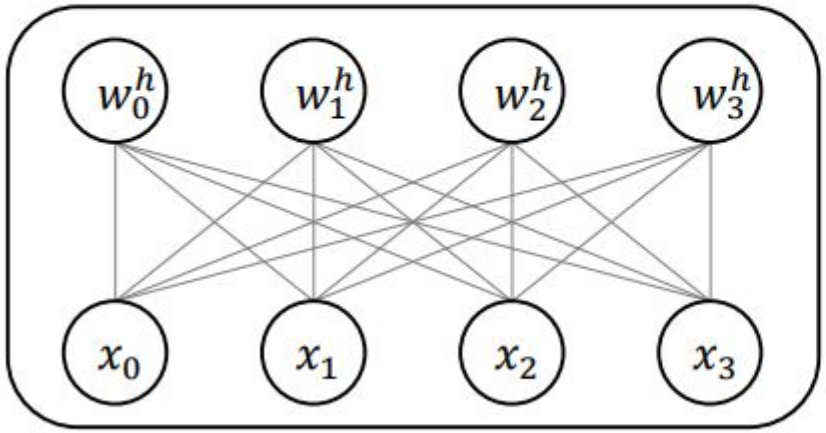
Vanishing Gradient Problem(기울기 소실, 기울기값이 사라지는 문제)

- 계층이 깊어질수록 학습이 어려운 기술기 소실 문제 발생

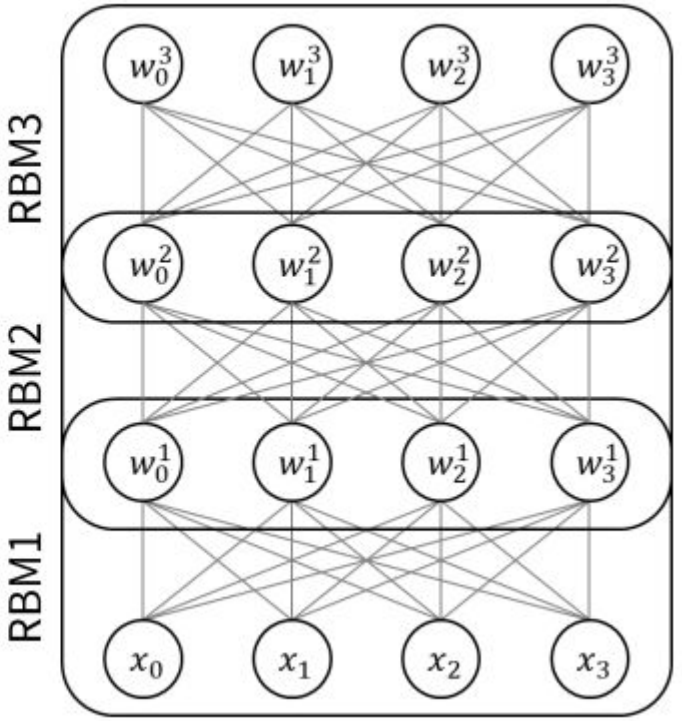


https://www.kaggle.com/getting-started/118228

심층 신뢰 신경망의 등장

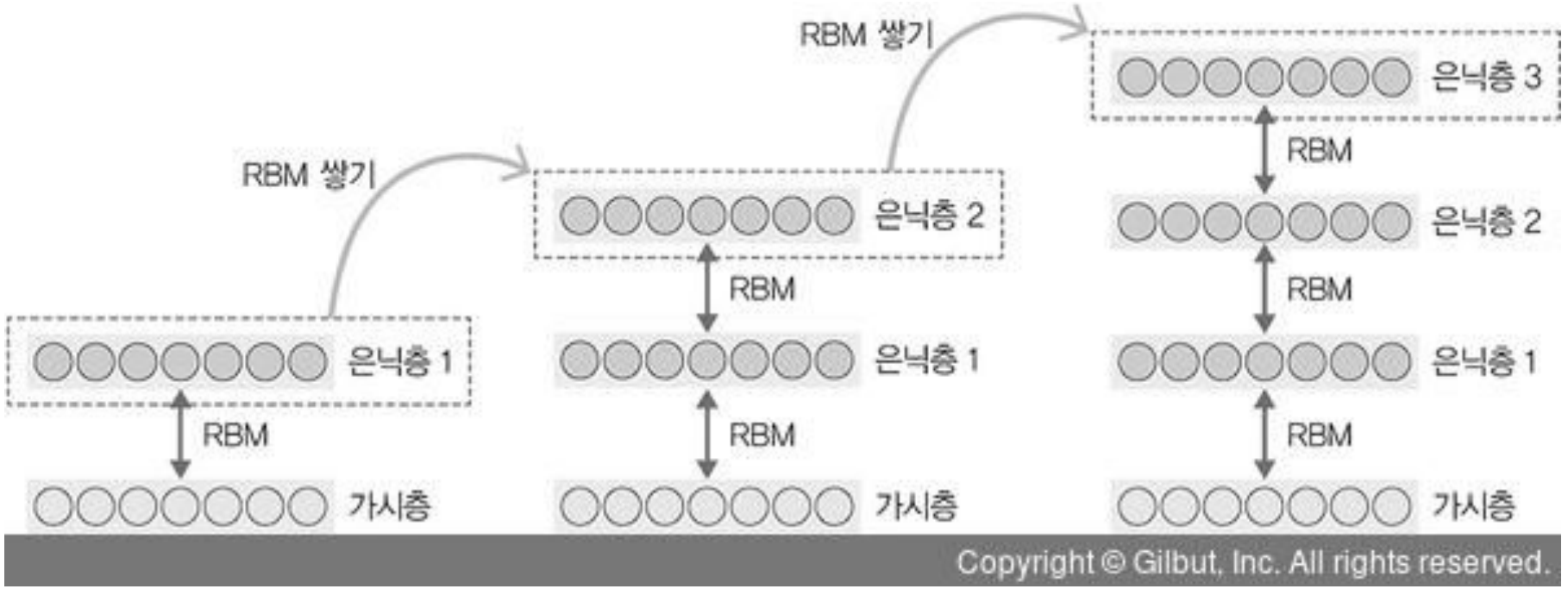


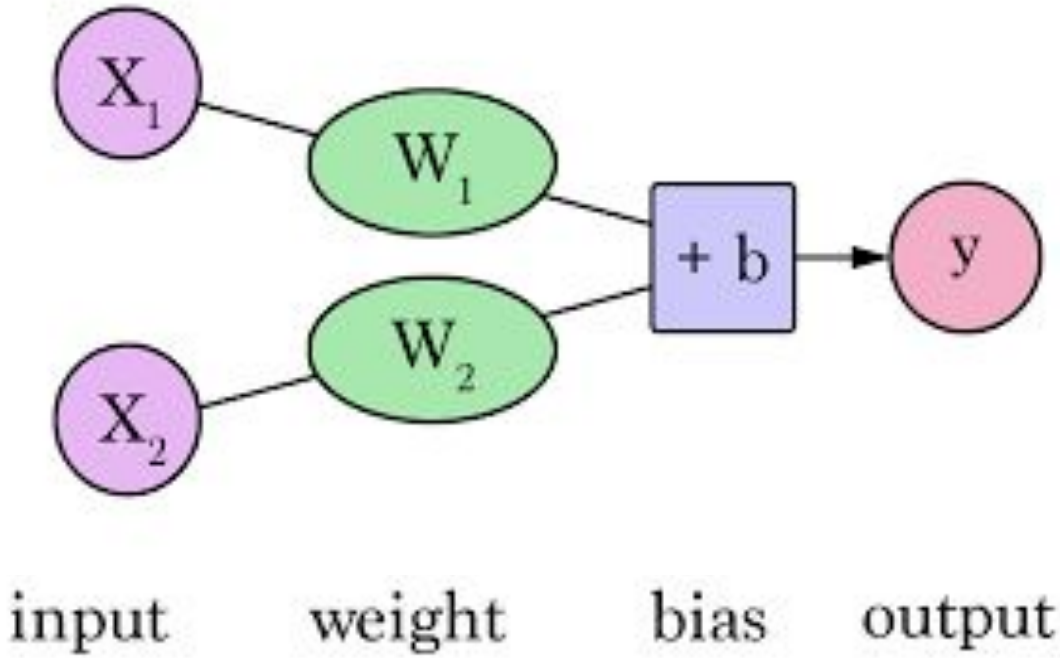
RBM
(Restricted Boltzmann machine,
제한된 볼츠만 머신)



DBN
(DBN, Deep Belief Network, 심층 신뢰 신경망)

심층 신뢰 신경망의 등장

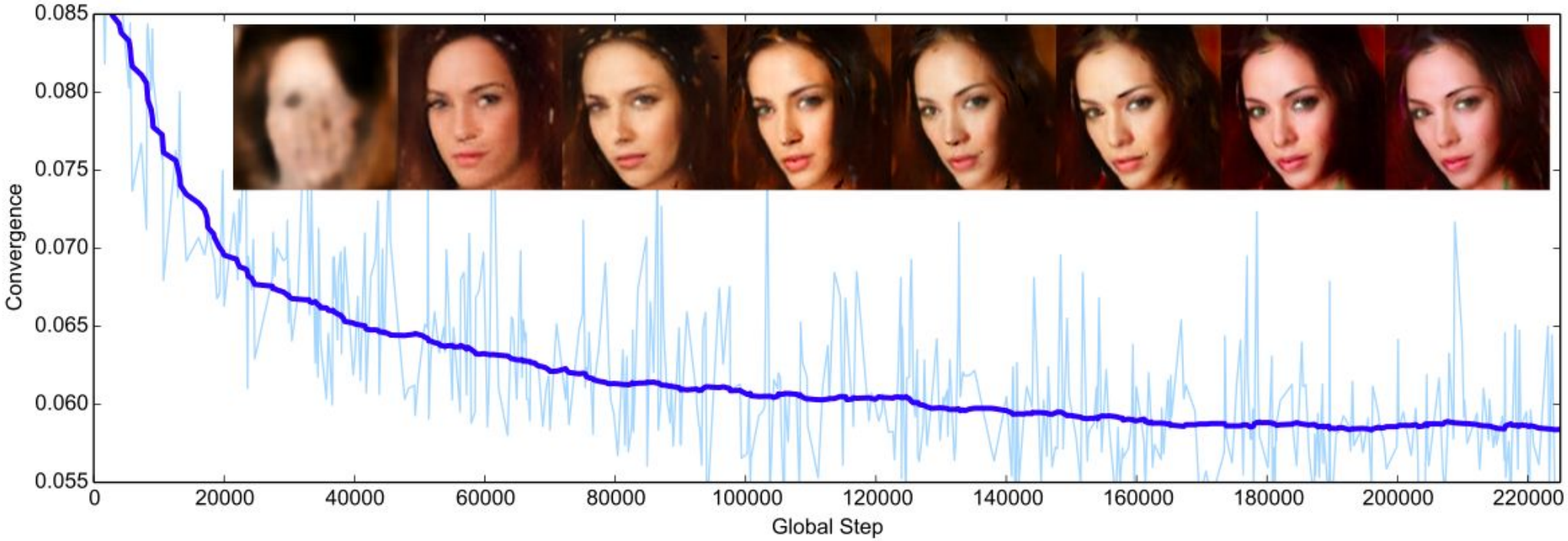

RBM 쌓기
RBM 쌓기
은닉층 3
은닉층 2
은닉층 1
가시층
RBM
RBM
RBM
RBM
RBM
RBM
Copyright © Gilbut, Inc. All rights reserved.



https://images.deepai.org/django-summernote/2019-06-03/e20ff932-4269-4bed-82fb-383b0f1ce96d.png

가중치(Weight)

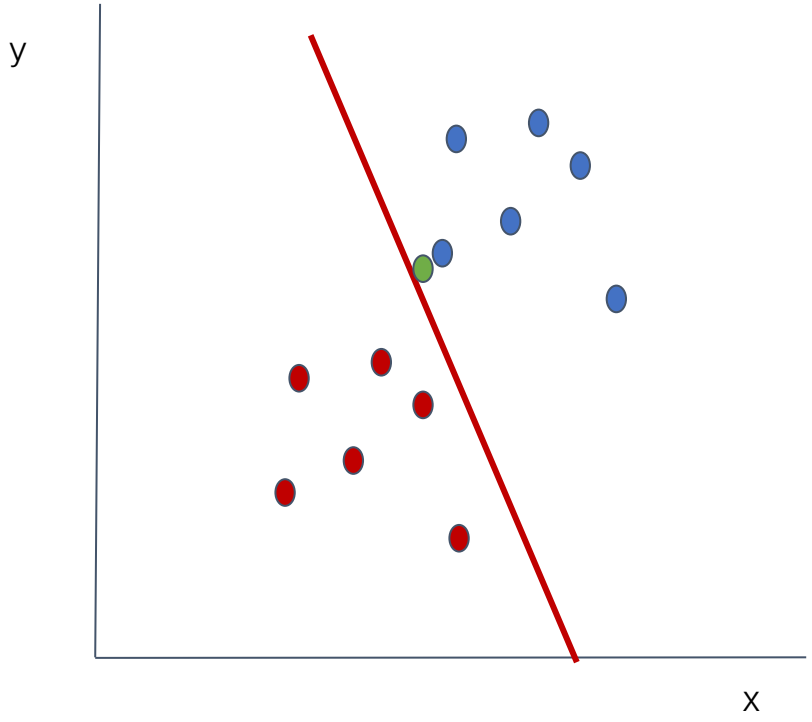
- 뉴런 학습시에 변하는 것
- 처음에는 초기화를 통해 랜덤한 값 넣고, 학습 과정에서 점차 일정한 값으로 수렴
- 학습이 잘 된다:
  - 좋은 가중치를 얻어서 원하는 출력에 점점 가까운 값을 얻고 있다



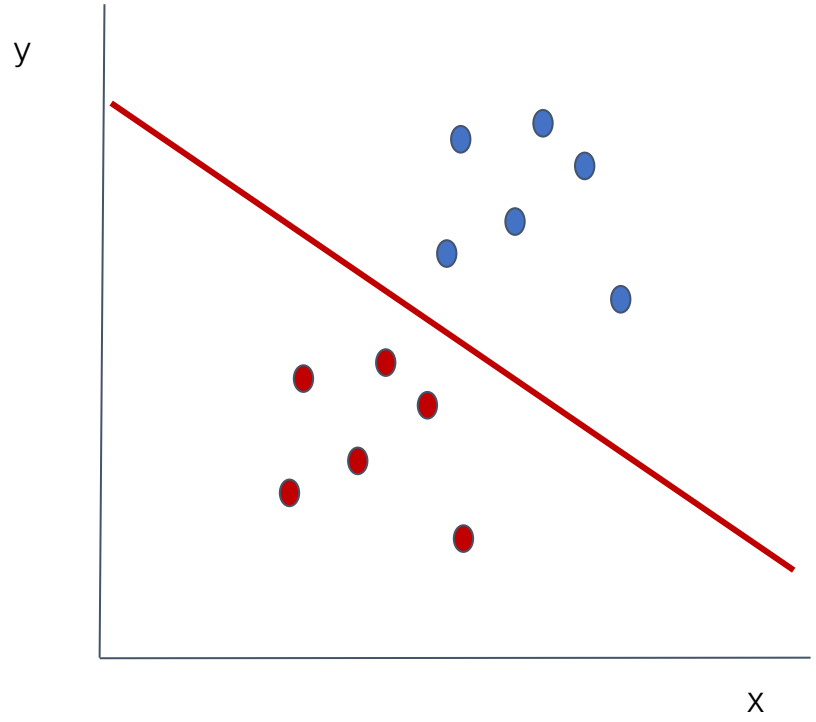
딥러닝 학습을 이용한 가상 인물의 생성 사례, https://github.com/artcg/BEGAN/blob/master/readme/conv_measure_vis.png

활성화 함수(Activation Function)

둘 중 더 좋은 결정 경계는 무엇일까요?

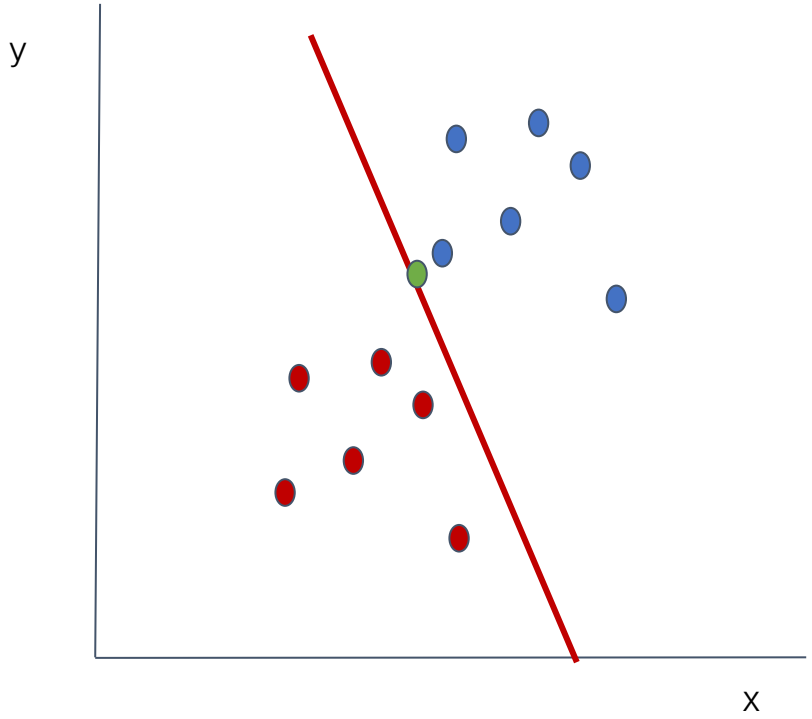


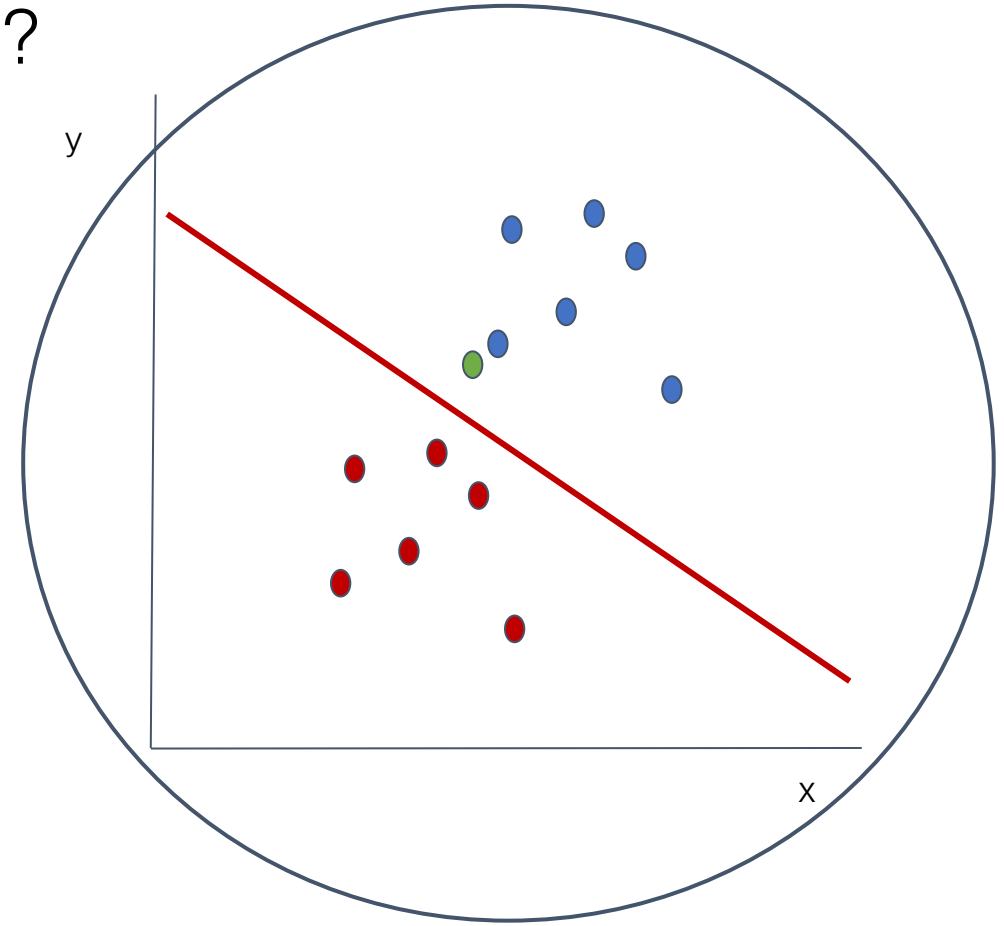
vs



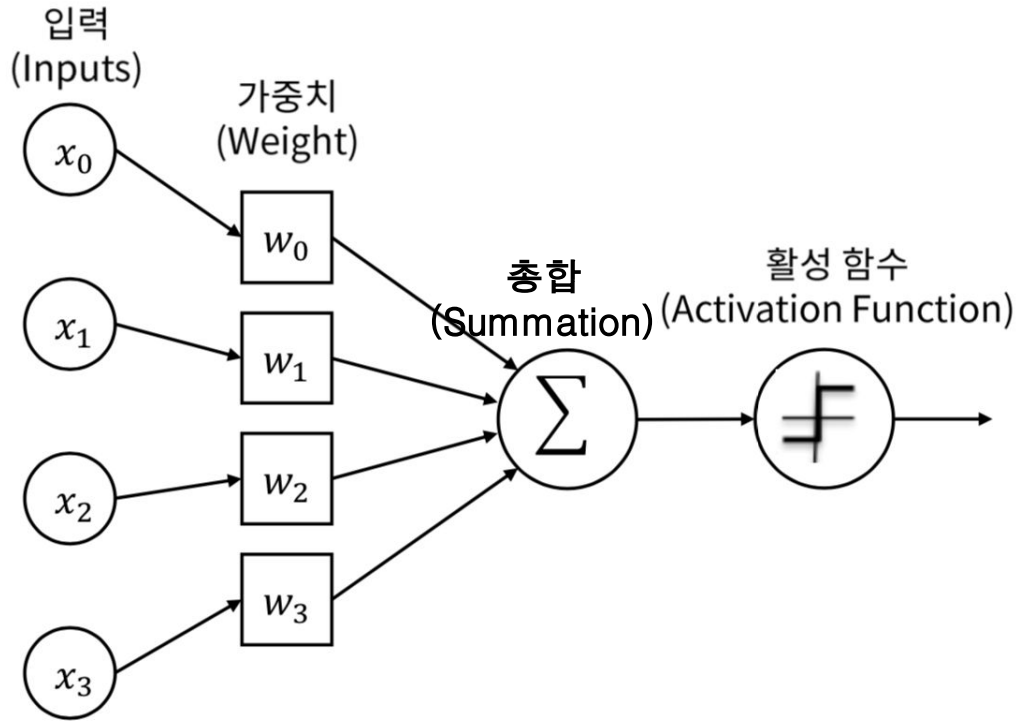
활성화 함수(Activation Function)

둘 중 더 좋은 결정 경계는 무엇일까요?

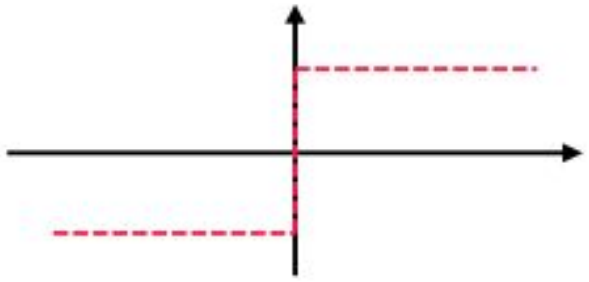




활성화 함수(Activation Function)



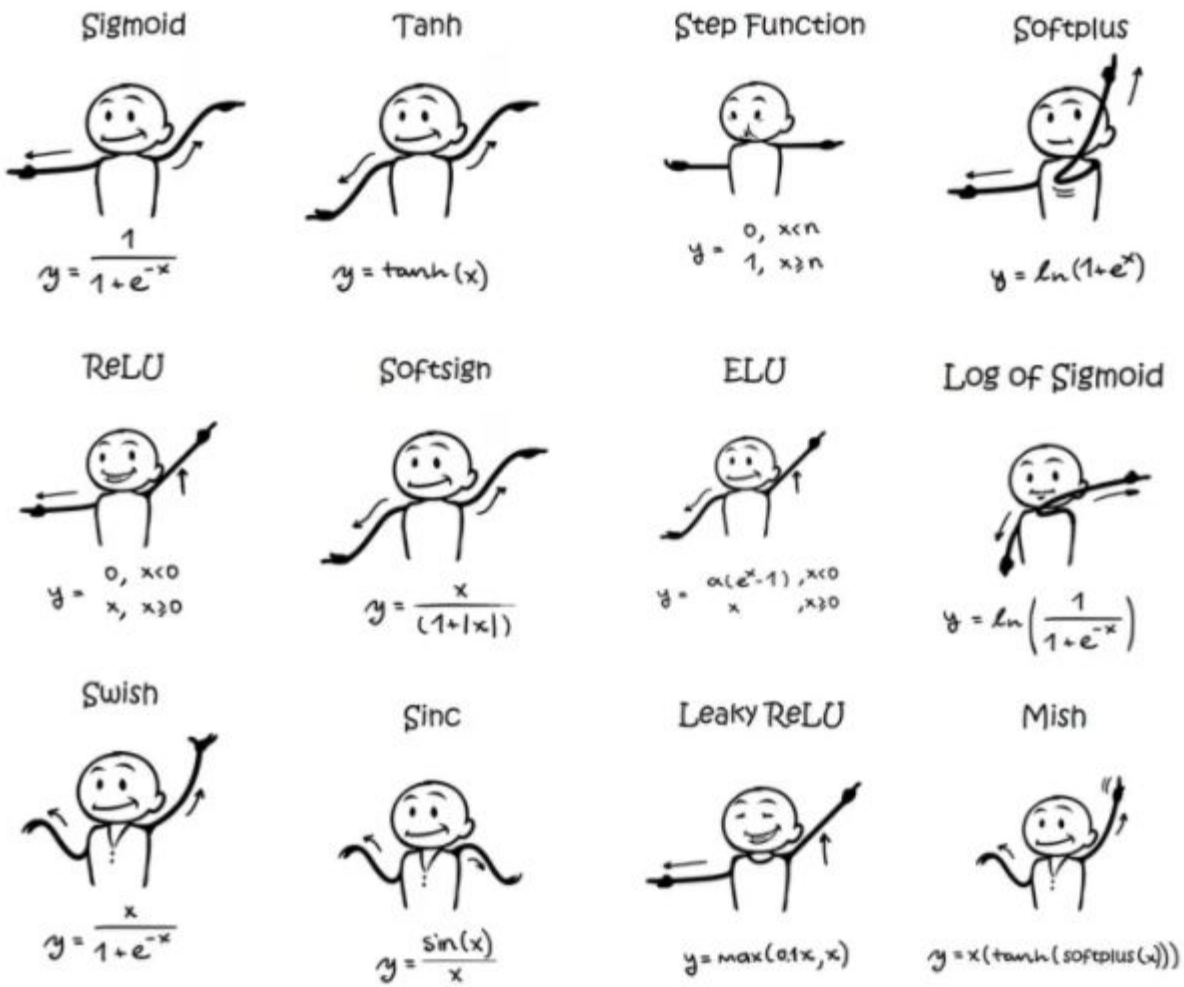
0/1만 구분할 뿐 거리를 신경쓰지 않았기 때문에 생긴 문제



$$sign(t) = \begin{cases} +1, & if\ t > 0 \\ -1, & otherwise \end{cases}$$

$$y = sign(\boldsymbol{x}^T \boldsymbol{w})$$

활성화 함수(Activation Function)

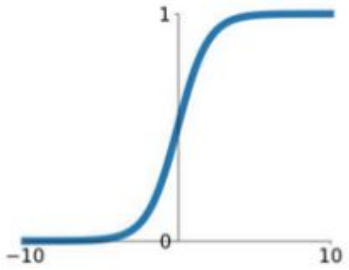


https://medium.com/analytics-vidhya/activation-functions-in-neural-network-55d1afb5397a

활성화 함수(Activation Function)

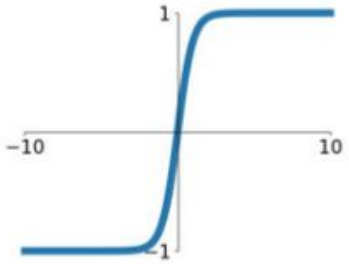
Sigmoid

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



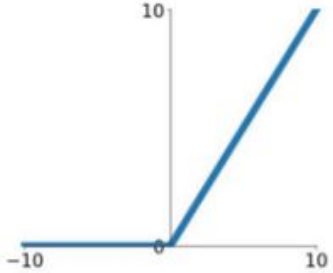
tanh

$$\tanh(x)$$



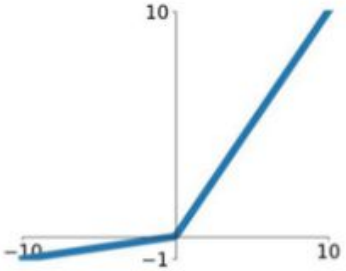
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

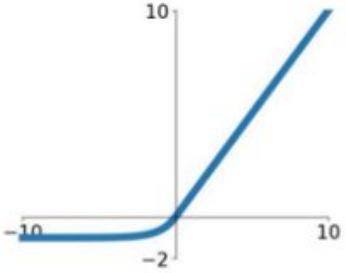


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



https://22-22.tistory.com/20

활성화 함수(Activation Function)

활성함수의 역할

- 입력을 활성화해서 다양한 출력을 구성
- 입력의 총합을 어떻게 활성화 해서 출력하는 지를 결정하는 함수
- 각 노드가 이전 노드들로부터 전달 받은 정보를 다음 노드에 얼마만큼 전달해 줄지를 결정
- 가중치 값을 학습할 때 에러가 적게 나도록 도와주는 함수

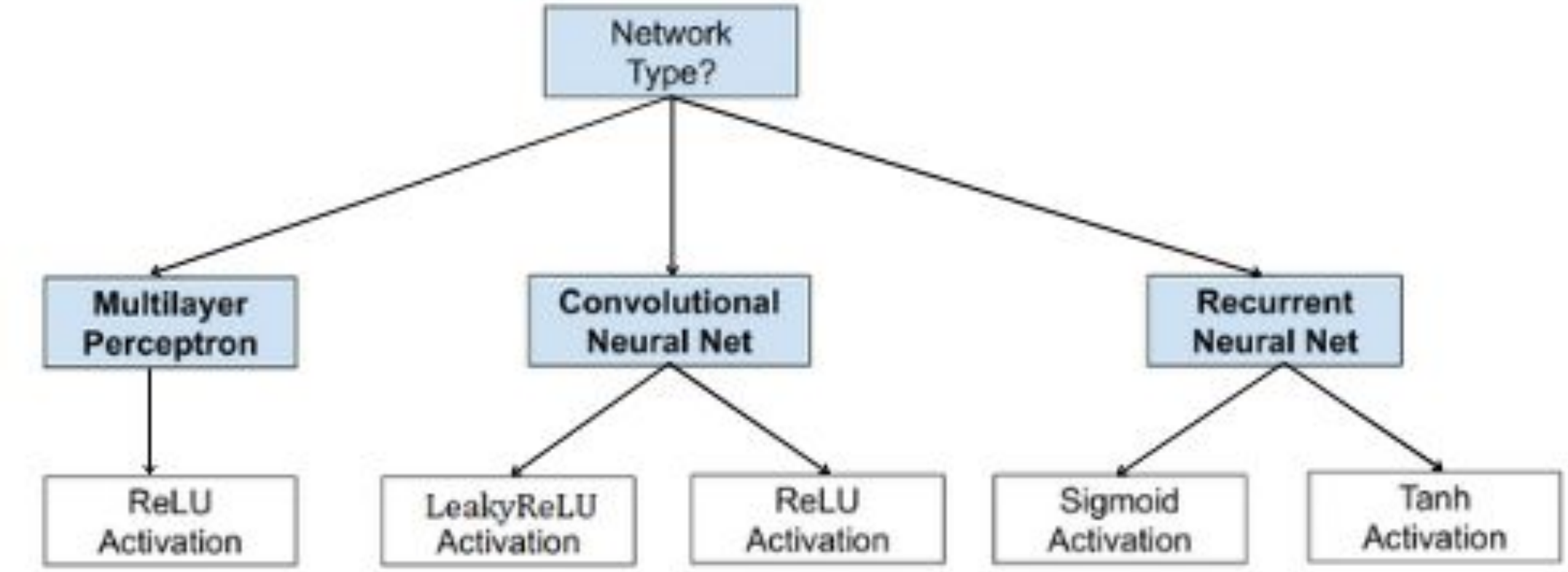
활성화 함수(Activation Function)

활성화함수로 선형 함수를 사용하면 안되는 이유

- 여러 층으로 된 모델을 하나의 층만으로도 나타낼 수 있게 되기 때문
  - 활성화함수가 선형 함수이면 여러 층의 레이어를 행렬의 곱으로 나타낼 수 있어 하나의 행렬과 같아짐

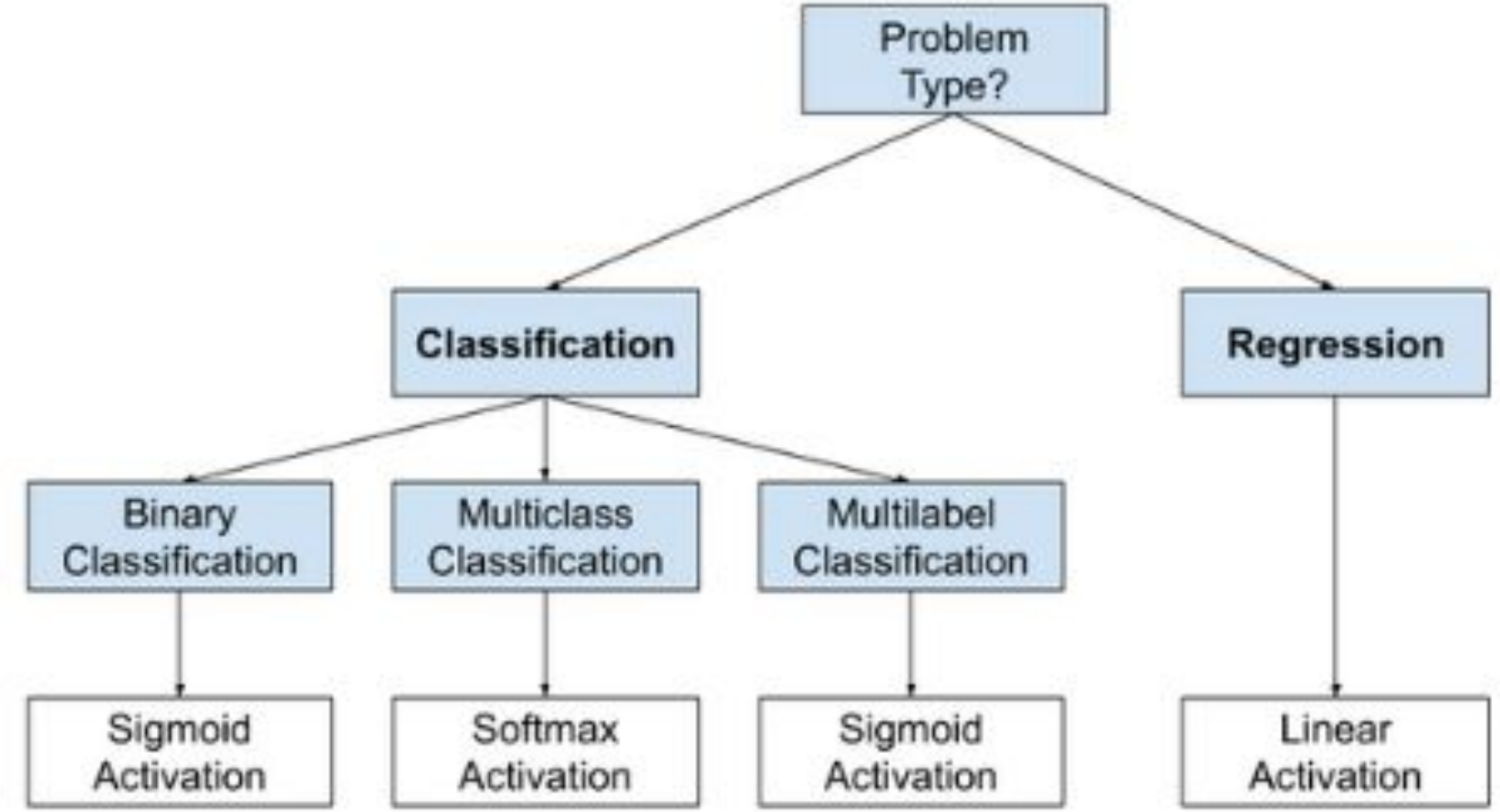
- vanishing gradient 문제
  - 역전파(backpropagation, 틀린정도를 미분한 것을 전달하는 것)에서 레이어가 깊어지며 업데이트가 사라지면서 underfitting 발생

Activation Functions in Hidden Layers



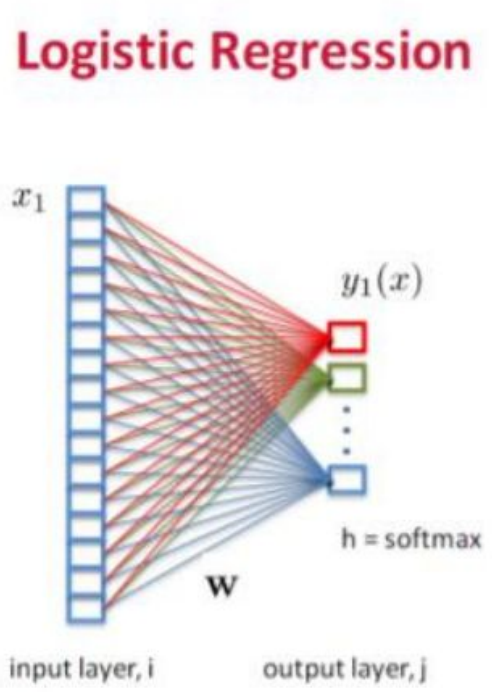
https://www.kaggle.com/general/212325

Activation Functions in Output Layers

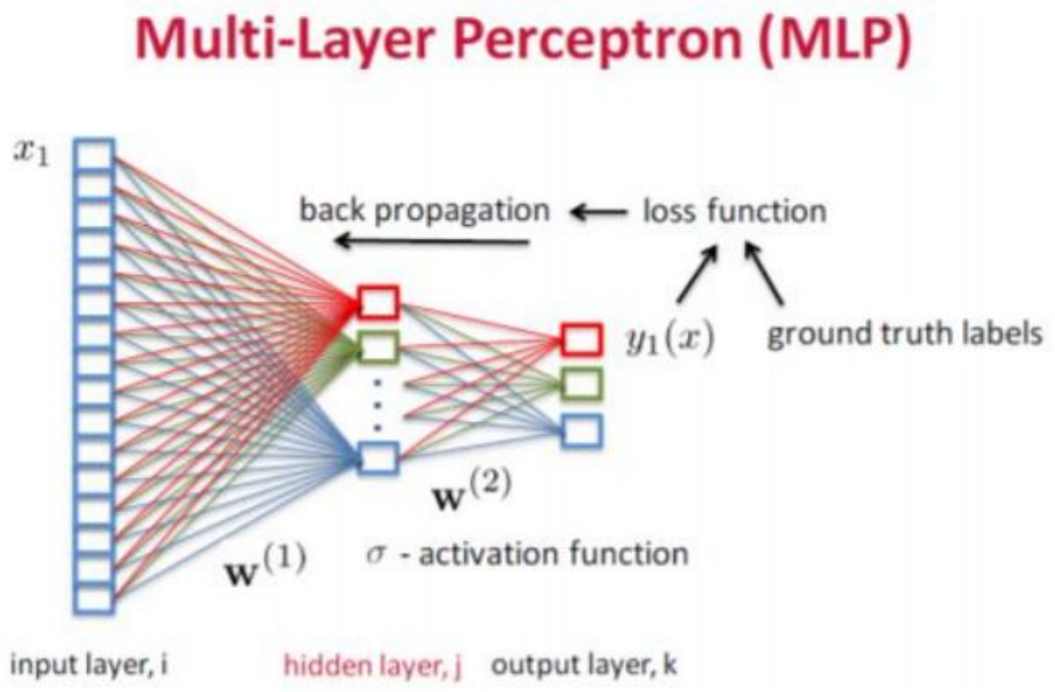


https://www.kaggle.com/general/212325

딥러닝



은닉층이 미존재하는 신경망



일반적인 신경망의 모습

• Full Connected Network – 다 연결하는 구조이기 때문

딥러닝

- 깊게 배운다: 연속된 층(Layer)에서 점진적으로 의미 있는 표현을 배움

  층 기반 표현 학습(Layered Representations Learning)

  계층적 표현 학습(Hierarchical Representations Learning)

- Deep : 데이터로부터 모델은 만드는 데 많은 층(deep)을 사용

  모델의 깊이 : 모델을 만드는 데 얼마나 많은 층을 사용했는가

  사람이 개입하지 않아도 자동으로 변화에 적응 연속된 층에서 의미있는 output을 제공해주고 input으로 유입

데이터

훈련데이터

학습 과정에 사용되는 데이터

딥러닝 네트워크의 가중치에 영향을 주는 데이터는 훈련 데이터뿐

검증 데이터

훈련 데이터로 학습할 때 일부 데이터를 떼어내서**검증** 데이터로 구성

검증 데이터 성적이 잘 나오지 않을 경우 학습을 중단 할 수 있음

학습을 언제 멈출지 결정하는데 좋은 판단 기준이 됨

테스트 데이터

학습 결과를 최종 **평가**하기 위한 데이터

훈련 : 검증 : 테스트 데이터 비율 = 60 : 20 : 20

Batch, Epoch

- Batch
  - batch의 사전적 의미에는 "집단, 무리; 한 회분; (일괄 처리를 위해) 함께 묶다" 등이 있음
  - 딥러닝에서 배치는 모델의 가중치를 한번 업데이트시킬 때 사용되는 샘플들의 묶음을 의미
  - 만약에 총 1000개의 훈련 샘플이 있는데, 배치 사이즈가 20이라면 20개의 샘플 단위마다 모델의 가중치를 한번씩 업데이트
  - 총 50번(=1000/20) 가중치가 업데이트
  - 하나의 데이터셋을 총 50개의 배치로 나눠서 훈련을 진행
- epoch
  - 사전적 의미는 "(중요한 사건, 변화들이 일어난) 시대"
  - 학습의 횟수를 의미
  - 1000개의 데이터가 존재할 경우 에포크가 10이고 배치 사이즈가 20이면, 가중치를 50번 업데이트하는 것을 총 10번
  - 반복
  - 각 데이터 샘플이 총 10번씩 사용되는 것으로 결과적으로 가중치가 총 500번 업데이트

Batch, Epoch


batch size
훈련셋
1 epoch
batch
2 epoch
batch
N epoch
batch
bskyvision.com
확대
샘플

데이터 정규화(Standardization) 필요성

- 실생활에서 얻는 데이터는 다양한 단위 보유
- 딥러닝에서는 이러한 데이터를 전처리해서 정규화 해야 학습 효율이 좋음

- 정규화
  - 각 데이터에서 평균값을 뺀 다음 표준편차로 나눔 (값의 범위(scale)를 0~1 사이의 값으로 바꾸는 것)
  - 데이터의 분포를 정규분포화하는 역할
  - 평균과 표준편차 산출시
    훈련 데이터의 평균과 표준편차를 구한 후 이것으로 테스트 데이터도 정규화
  - 학습 전에 scaling하는 것
  - 머신러닝에서 scale이 큰 feature의 영향이 비대해지는 것을 방지
  - 딥러닝에서 Local Minimal에 빠질 위험 감소(학습 속도 향상)