딥러닝팀

1팀

김재희
유경민
김주연
문서영
이수경

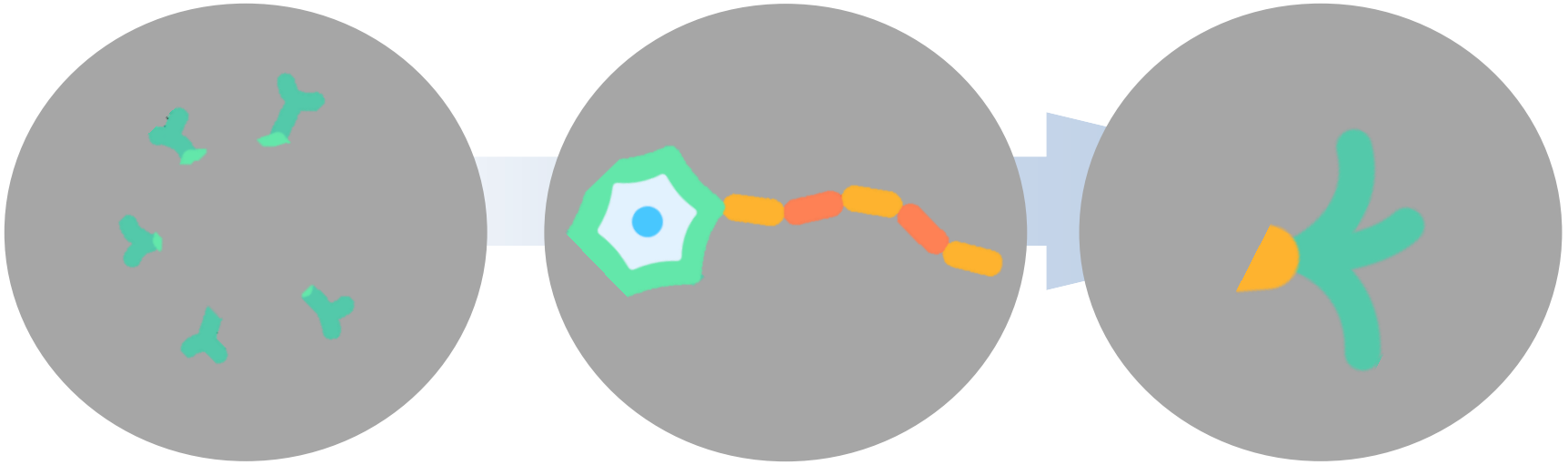
INDEX

1

딥러닝의 시작

1 딥러닝의 시작

● 신경세포의 구조



수상 돌기

정보를 전기신호 형태로 수신

신경세포체

정보의 강도를 판단, 전달할 정보 선별

축삭 돌기

전기 신호를 다음 신경세포에 전달

1 딥러닝의 시작

- 신경세포의 구조

단순한 동작을 하는 세포를 수억번 연결하면

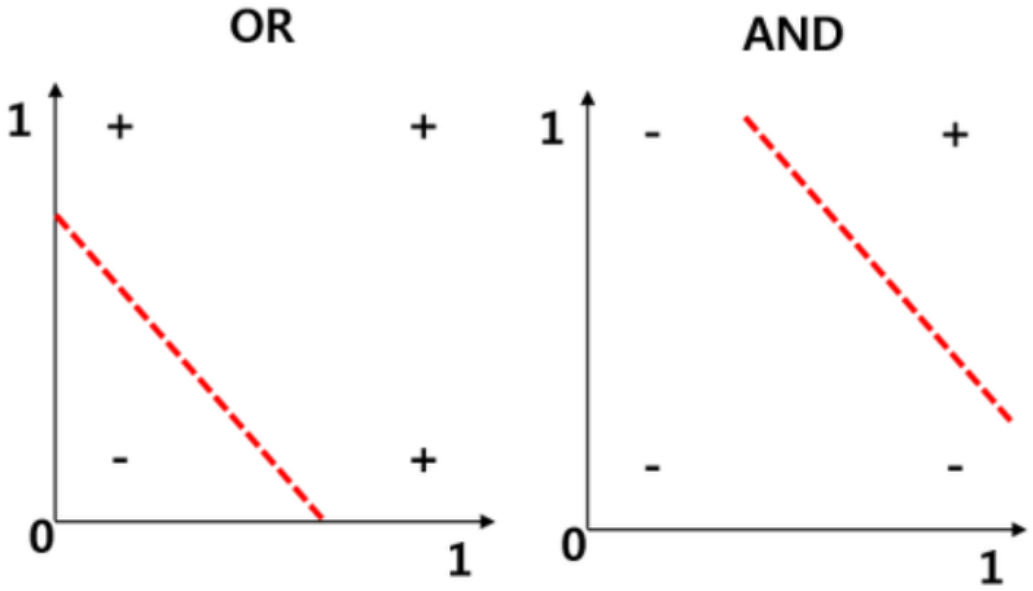
효율적이고 복잡한 의사결정이 가능하다는 사실!

신경세포 하나를 컴퓨터로 구현한 것을

퍼셉트론이라고 함!

수상 돌기	신경세포체	축삭 돌기
정보를 전기신호 형태로 **수신**	정보의 강도를 **판단**, 전달할 정보 **선별**	전기 신호를 다음 신경세포에 **전달**

● 퍼셉트론의 한계

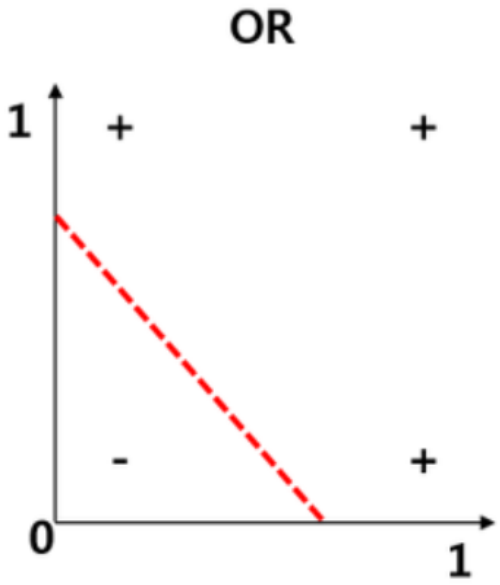


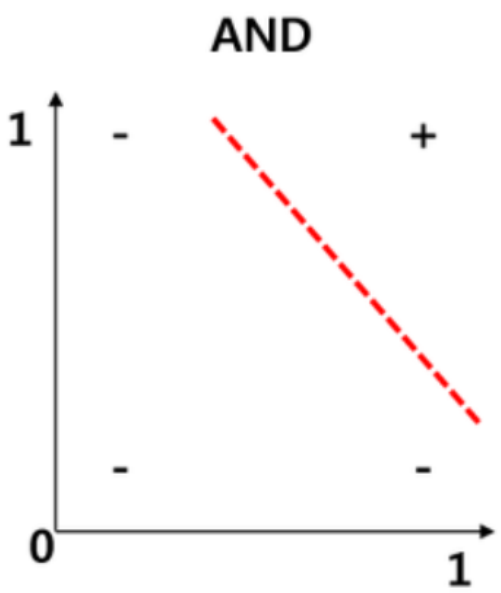
간단한 문제에서
나쁘지 않은 성능

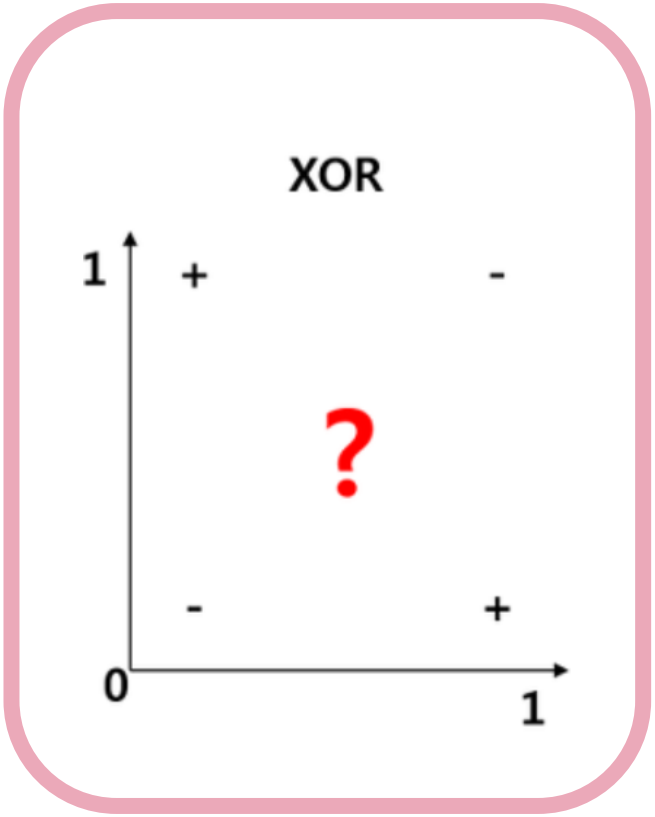
컴퓨터로 뇌세포 하나를 구현하는데 성공해,

좋은 모델로 성공할 것을 기대함

● 퍼셉트론의 한계





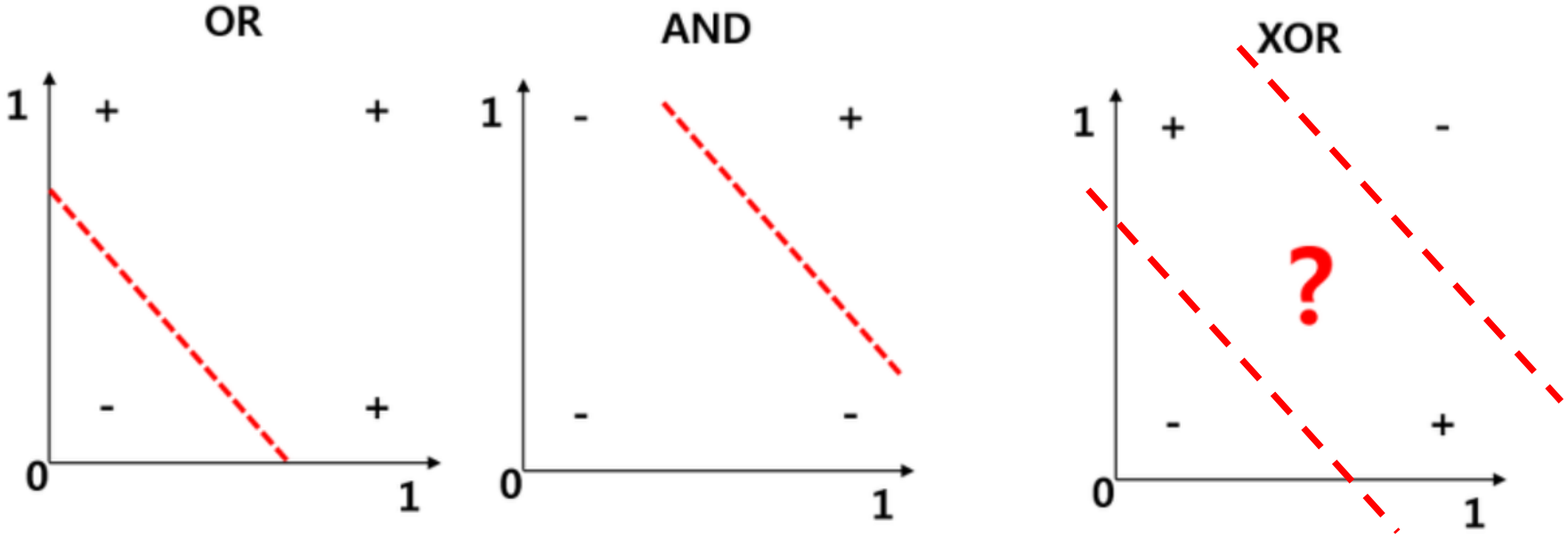


그러나, 복잡한 문제 해결 불가

EX) XOR

● 퍼셉트론의 한계

XOR 분류



두 회귀선을 이용한다면 해결가능하지만,

이를 하나의 퍼셉트론으로 묶을 방법을 찾지 못함

- 퍼셉트론의 구조

1) 입력값(x)

$$[\, x_1, x_2, \cdots, x_n \,]$$

데이터의 특징, 속성, 변수에 해당하며

벡터 형태로 입력 받는다.

- 퍼셉트론의 구조

2) 가중치(w, b)

$$[\, w_1, w_2, \cdots, w_n \,, \mathrm{b} \,]$$

가중치(w): 각 인풋에 대해 일정량의 중요도 부여
*가중치가 크다 = 해당 인풋이 해당 노드에 반영이 많이 된다

편향(b): 선형 회귀의 편향과 동일한 역할을 수행

- 퍼셉트론의 구조

3) 합계값(u)

$$u = \sum_{k=0}^{n} (x_k w_k) + b$$

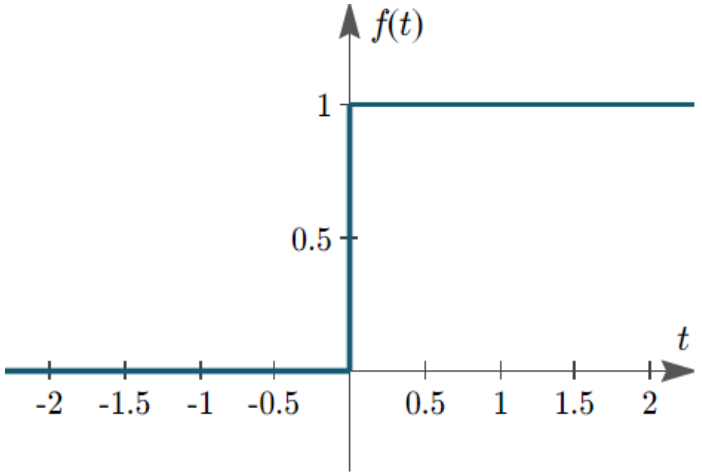
각 입력값(x), 가중치(w)의 곱을 합하여

하나의 합계값(u)으로 만든다

- 퍼셉트론의 구조

4) 활성화 함수

〈Step Function〉

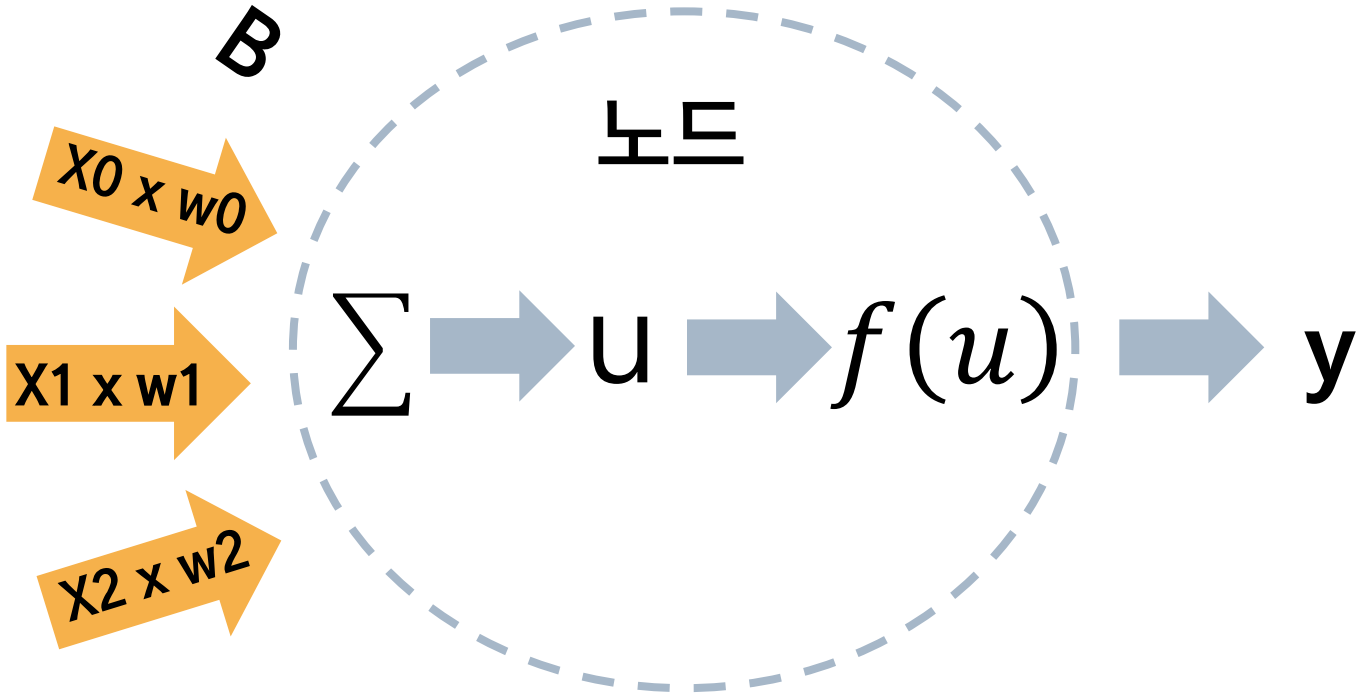


각 노드의 입력값을 출력값으로 변환해주는 활성화 함수로,

해당 출력값은 다음 노드에 전달된다

ex) sigmoid, Relu, Identity, Softmax

● 퍼셉트론의 구조



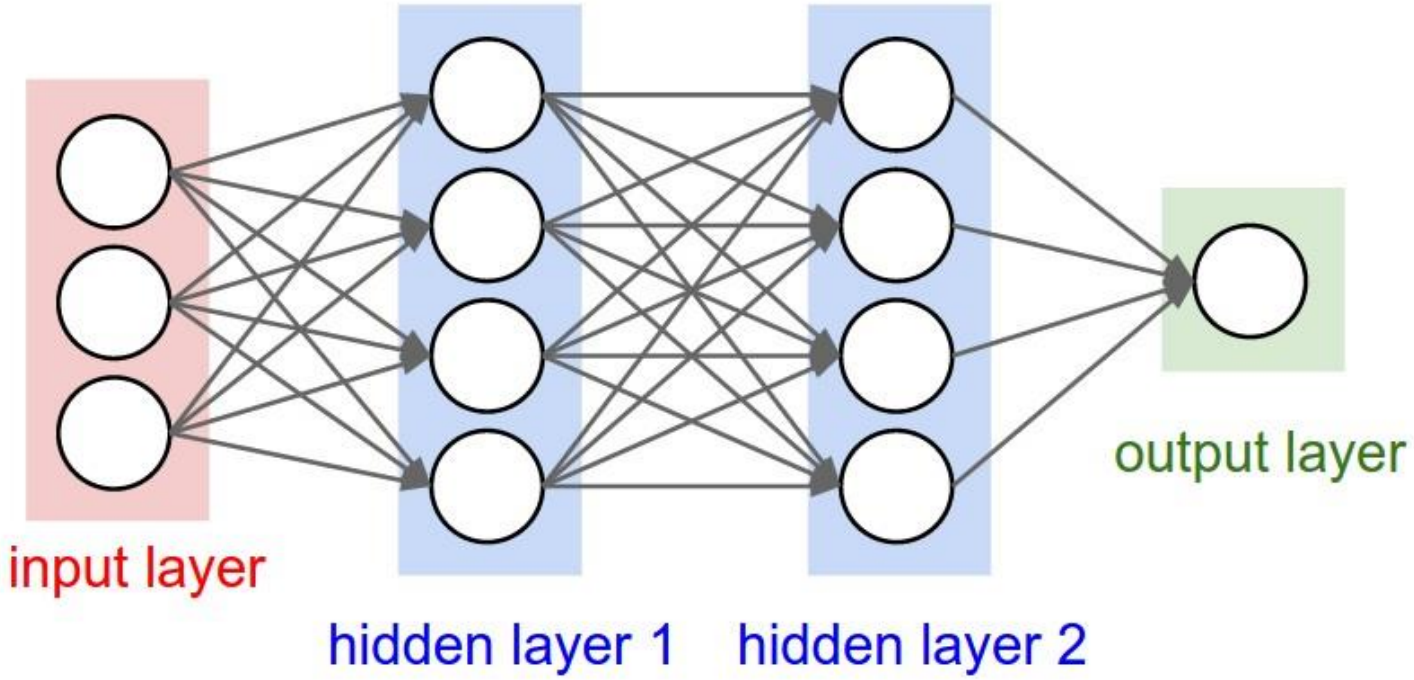
입력x가중치를 모두 더한 값을 활성함수에 대입

신경세포와 동일하다

2

다층 퍼셉트론

- **다층 퍼셉트론이란?**



퍼셉트론 여러 개를 엮어서 하나의 층을 만들고,

다시 여러 층을 엮어서 만들어진 모델

● 활성화 함수

Activation Function

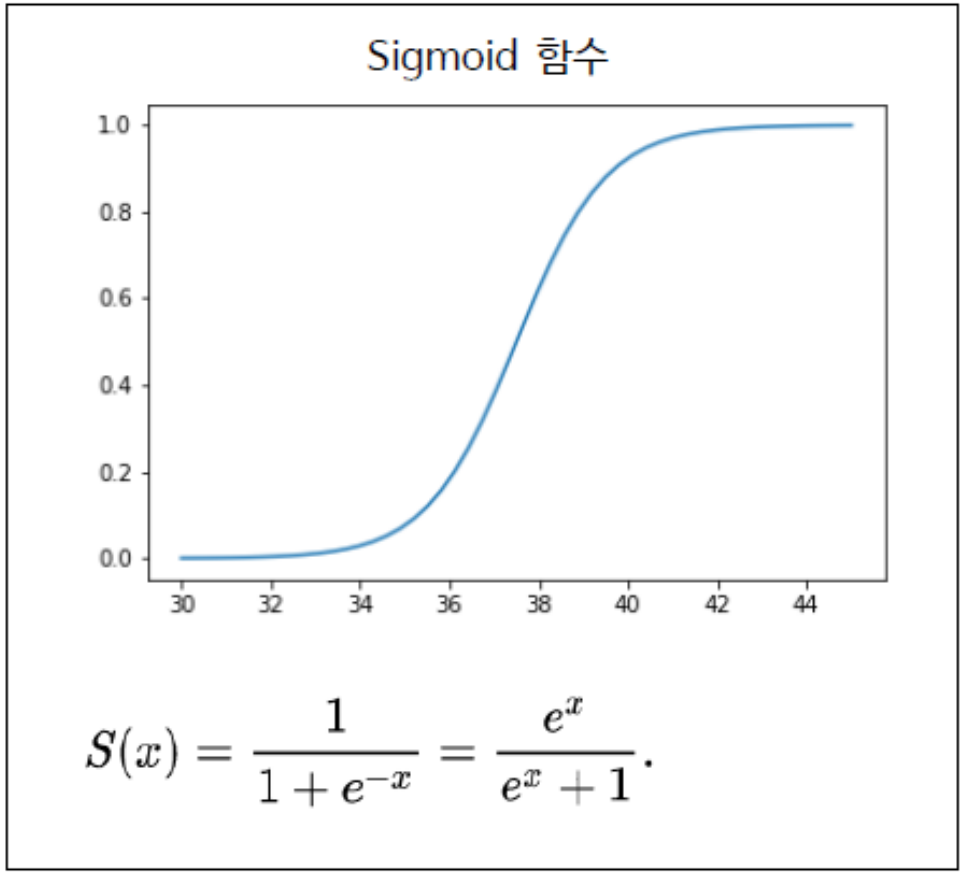
: 각 노드의 입력값을 출력값으로 변환해주는 함수로

활성화 함수를 통해 나온 출력 값이 다음 노드에 전달

예) sigmoid, Relu, Identity, Softmax

- 활성화 함수

Sigmoid Function



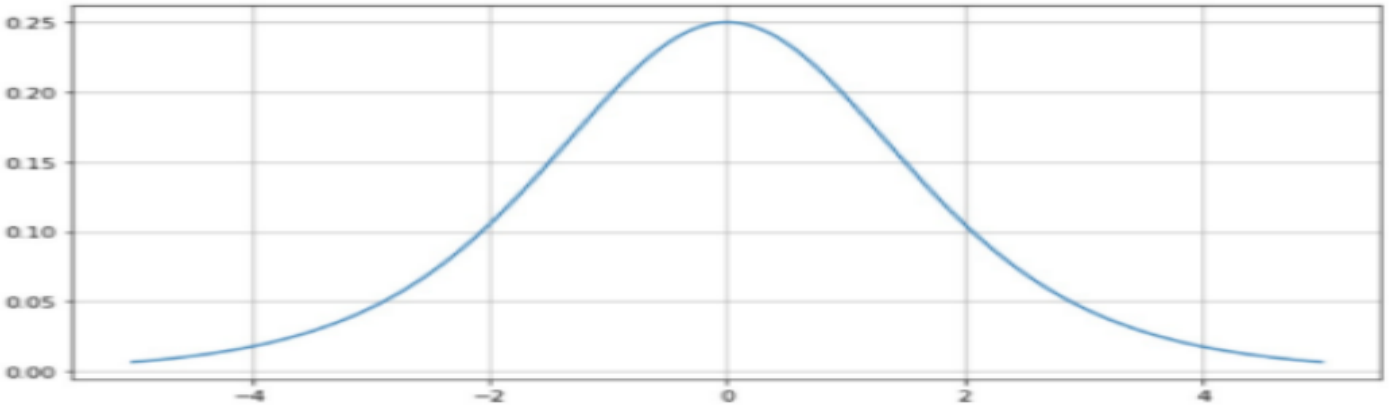
$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{e^{x}+1}.$$

모든 값이 0부터 1사이의 출력값을 갖는 함수로 기본적인 활성화 함수

기존의 함수와 달리 미분 가능한 함수로 역전파가 가능

- 활성화 함수

Sigmoid Function

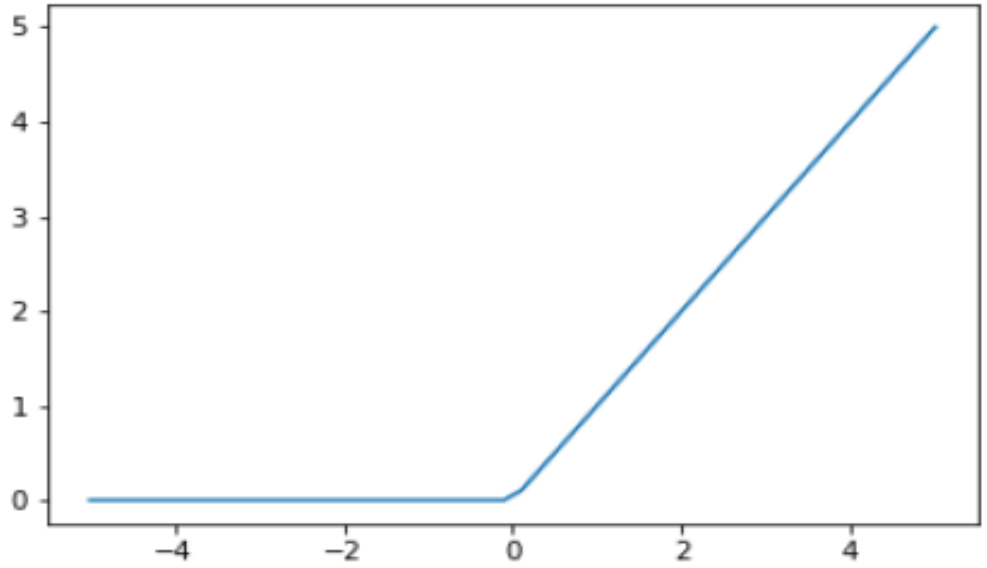


시그모이드 함수의 미분식을 시각화 한 것

그러나, 미분의 최대값이 0.25이기 때문에 곱할 수록 급격하게 값이 작아지는 형태를 띄어 Vanishing Gradient이라는 역전파 시 기울기 소실문제가 발생

- 활성화 함수

ReLU

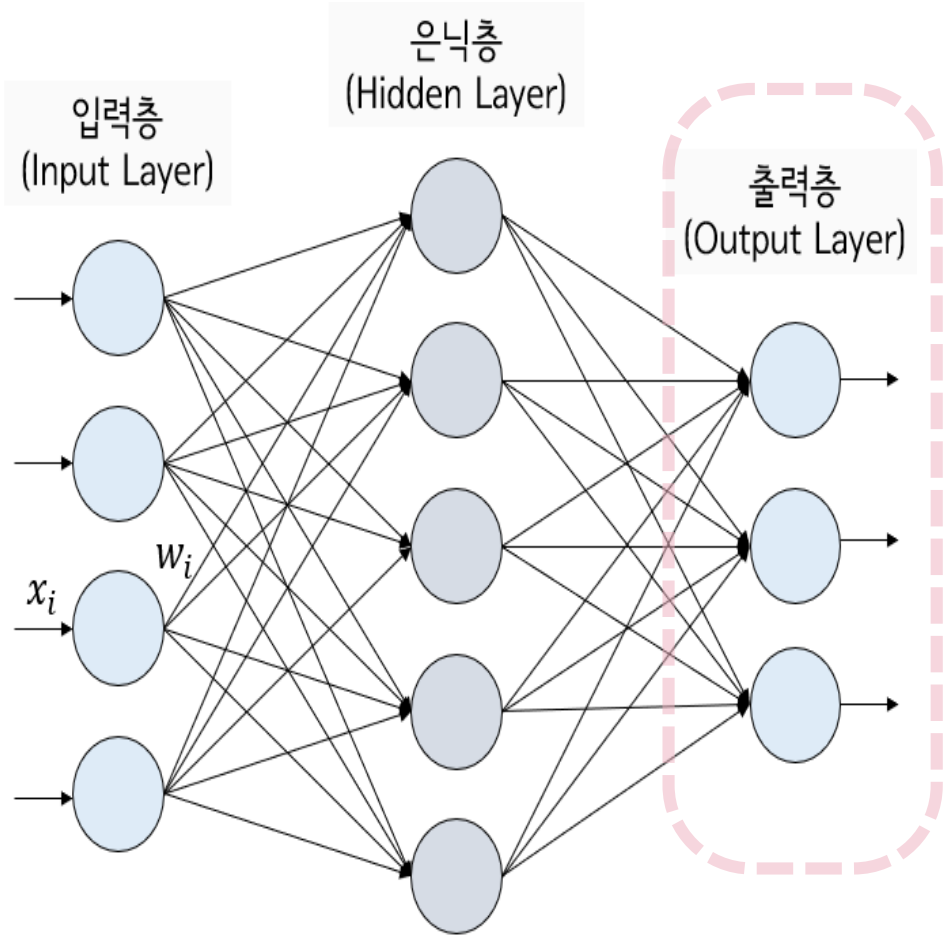


미분 값이 0 또는 1로 나오기 때문에 기울기 소실 문제로부터 Sigmoid Function보다 자유로움

$$y = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

- 활성화 함수

Identity Function



: 항등함수(y=x)

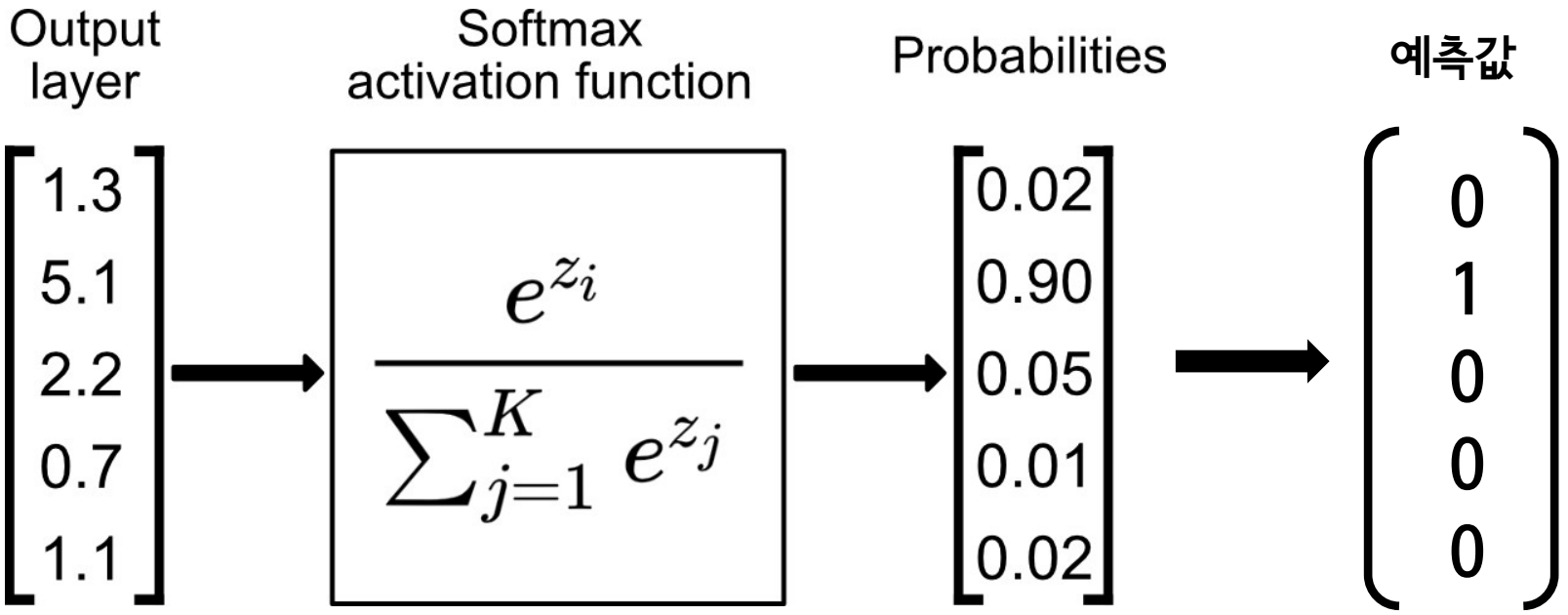
회귀 문제에서

마지막 출력층에 사용

합계값인 u값이 바로 출력

- 활성화 함수

Softmax



다중 분류 문제에서 출력층에 사용하며, 합계값인 u값을 (0,1)사이의 확률값으로 전환하여 각 label에 속할 확률을 출력함

● 손실함수

손실함수

: Loss Function

데이터에 대한 예측값과 실제 관측 값을 비교하는 함수

예측값과 실제 관측값을 비교해야 역전파를 사용할 수 있음

회귀문제

SSE

(Sum of Squares for Error)

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

분류문제

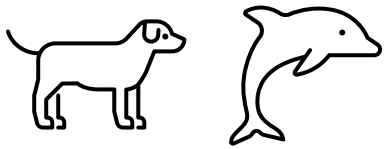
Cross Entropy

● 손실함수

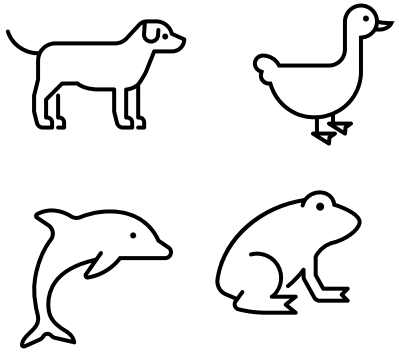
Cross Entropy

: 예측값의 분포가 실제값의 분포와

얼마나 차이가 나는지를 보여주는 함수



이진분류



다중분류

● 손실함수

이진분류

: 출력층의 노드가 1개여서

실제 값이 0일 때와 1일 때의 로그 값을 따로 만듦

$$Loss = \underbrace{(Y)(-log(Y_{pred}))}_{\text{Y=1일 때 남음 / Y=0일 때 소실}} + \underbrace{(1-Y)(-log(1-Y_{pred})}_{\text{Y=0일 때 남음 / Y=1일 때 소실}}$$

● 손실함수

다중분류

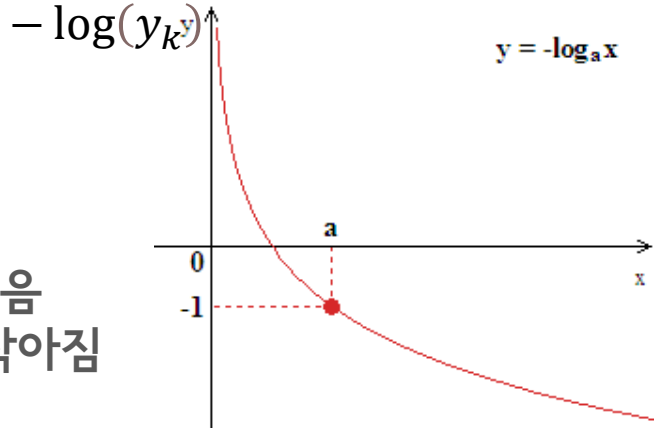
$$E = \sum_k t_k(-\log(y_k))$$

: 출력층의 노드가 3개 이상인 경우

실제 값(t_k) 예측 값(y_k)

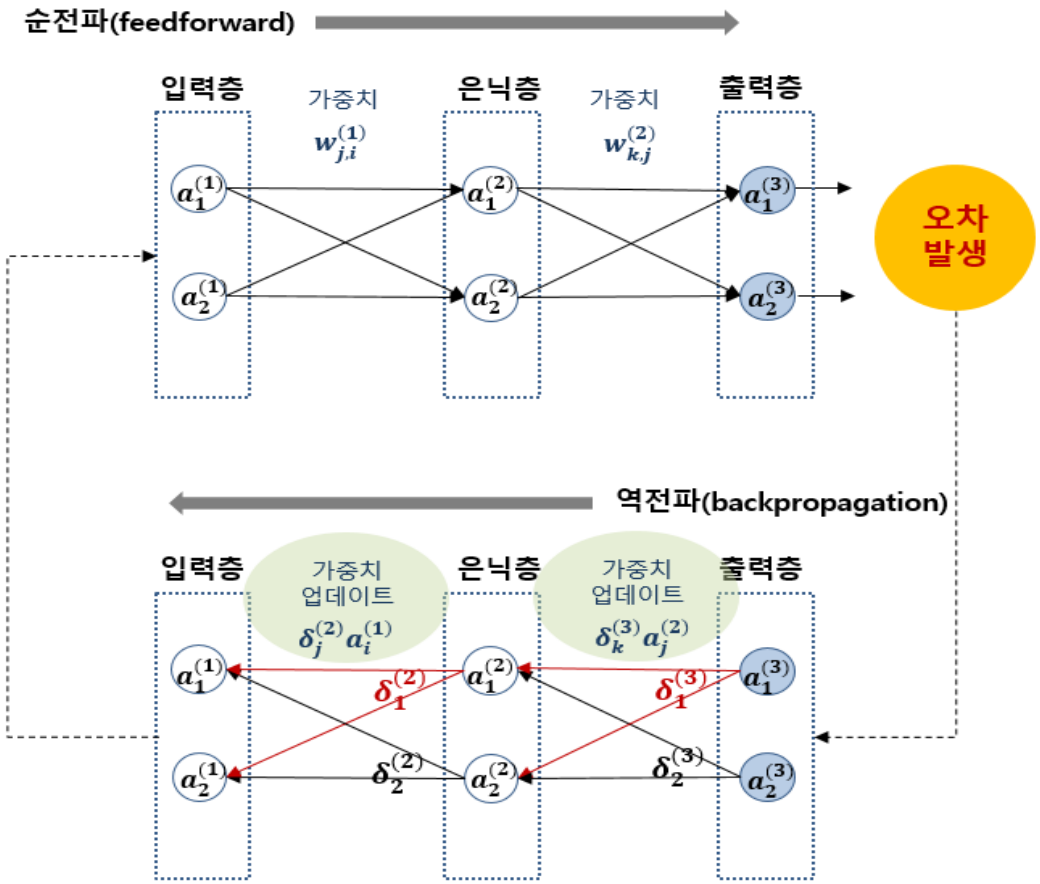
$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \times \begin{bmatrix} 0.02 \\ 0.90 \\ 0.05 \\ 0.01 \\ 0.02 \end{bmatrix} \rightarrow$$

t_k가 1인 값들만 남음
y_k가 클수록 E가 작아짐



2 다층 퍼셉트론

● 역전파



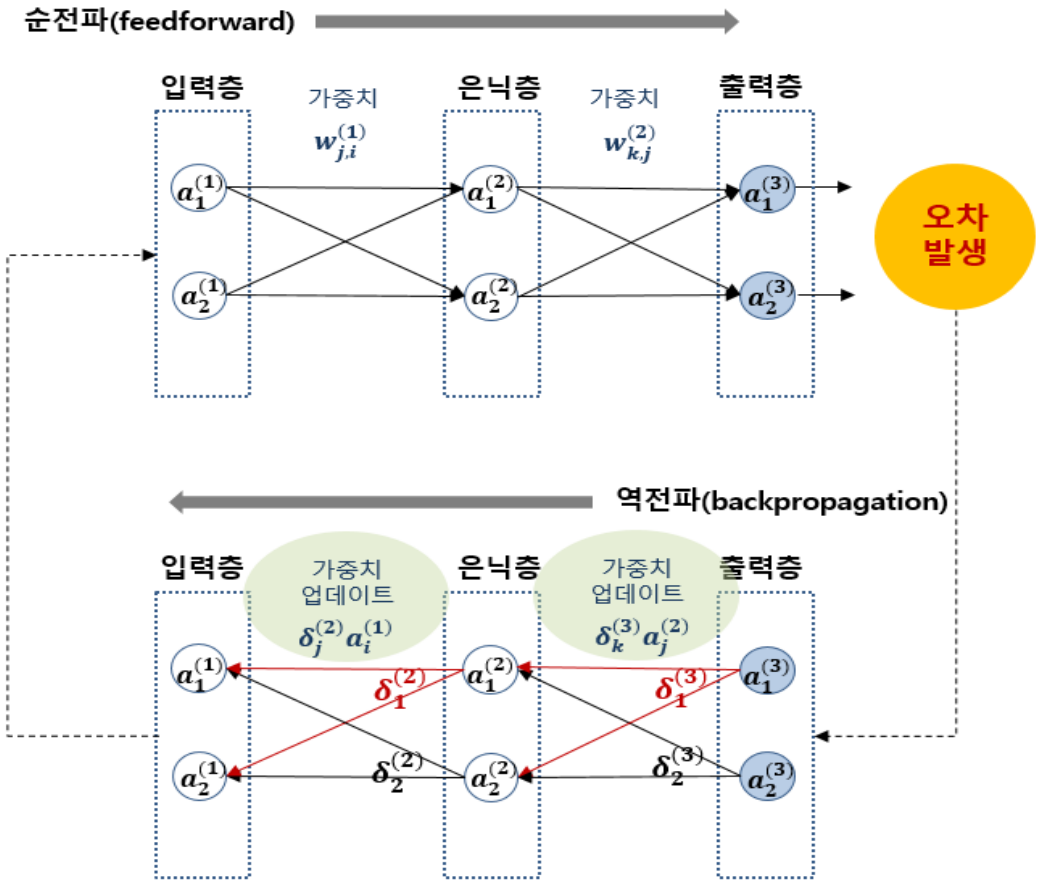
배경

모델의 학습은 파라미터(w, b)가 최소의 Loss값을 갖도록 조정되는 과정임

그런데, 다층 퍼셉트론은 증명을 통해 최적의 파라미터의 기준과 값을 구할 수 없음

2 다층 퍼셉트론

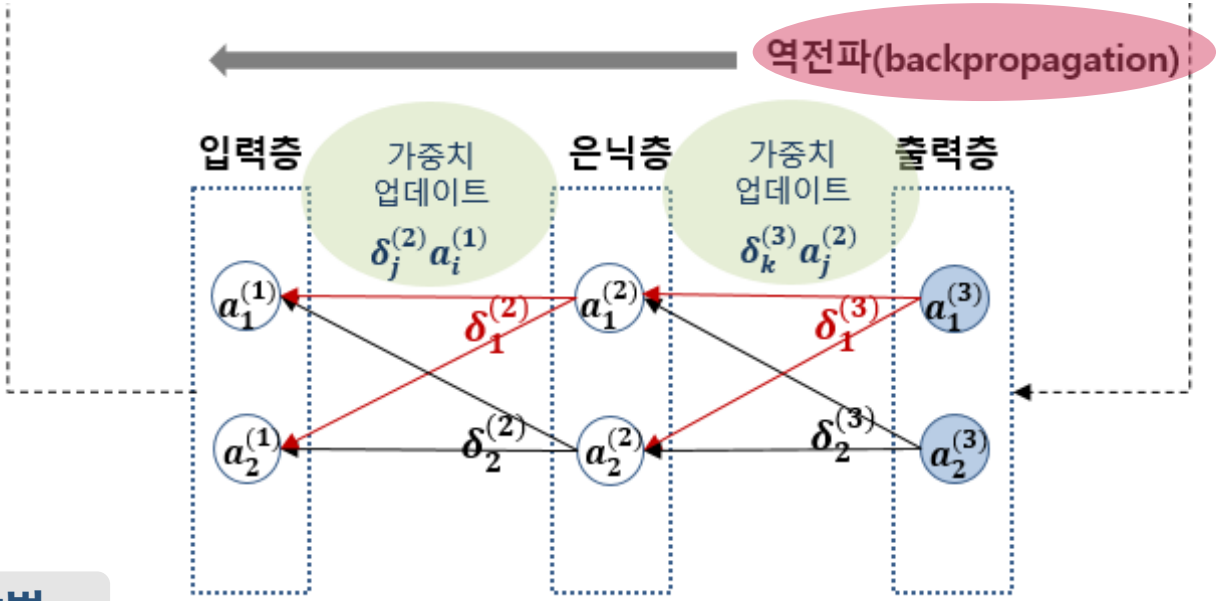
● 역전파



가중치가 오차에서 차지하는 비중에 따라 가중치 업데이트가 일어나는 데, 그 차지하는 비중의 계산이 역전파를 통해 이루어짐

2 다층 퍼셉트론

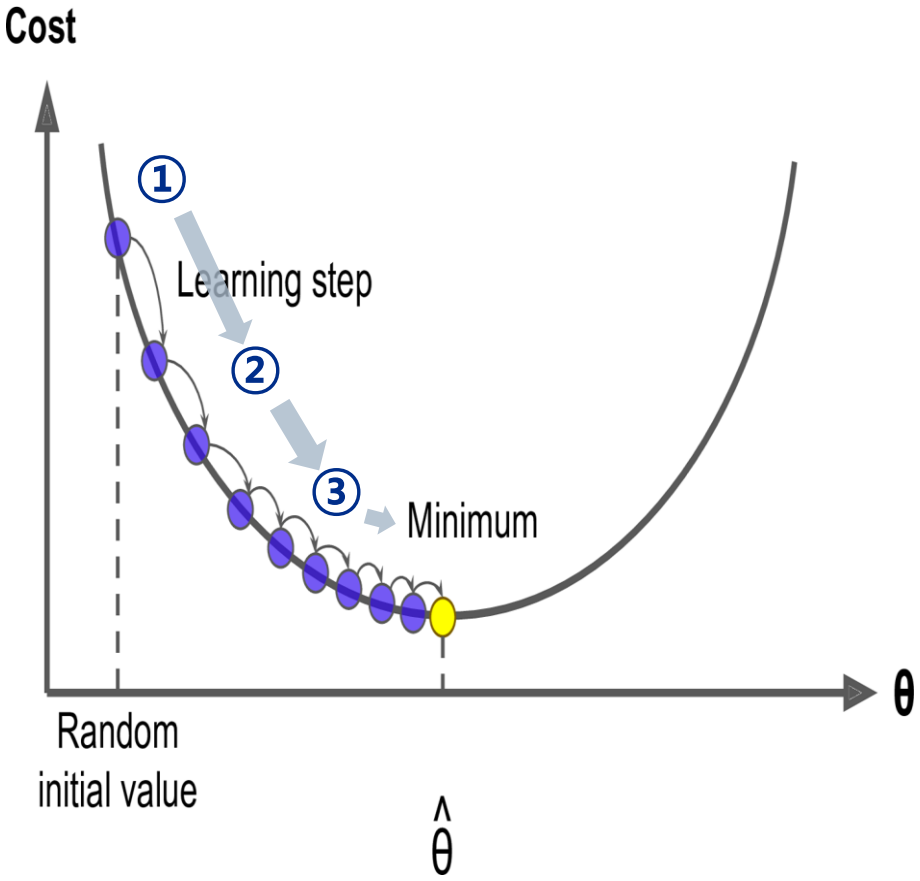
● 역전파



계산 방법

1) 표준 정규분포를 이루는 무작위의 값을 초기 가중치로 설정
2) 입력값들을 순전파 시켜 예측값을 산출
3) 예측값과 y값을 비교하여 오차 계산
4) 오차를 이용하여 모든 식을 미분하여 오차에 가중치가 차지하는 비중 계산

● 최적화 (Optimizer)

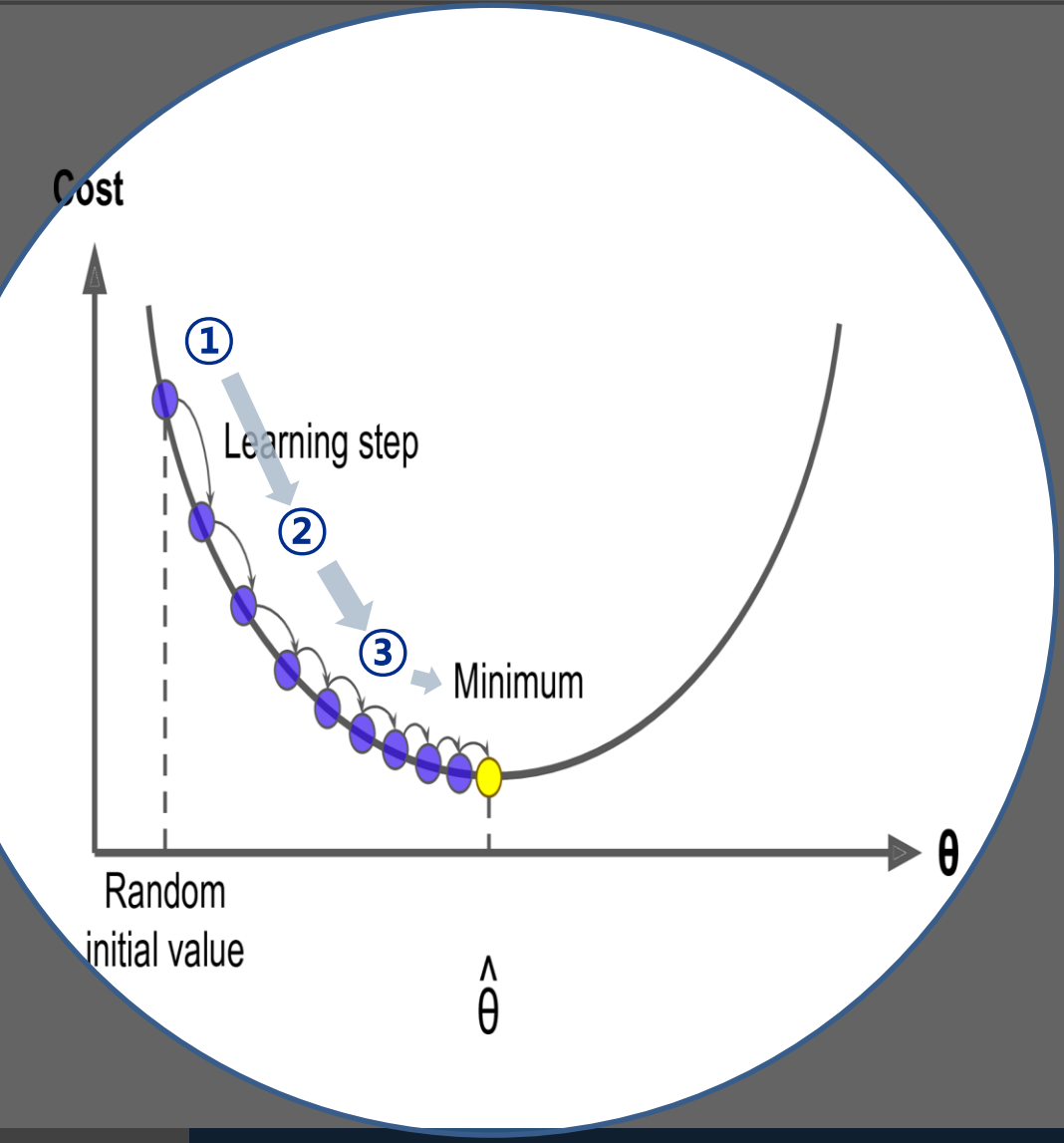


역전파를 이용해 각 파라미터(가중치; w,b)들의 최적의 값을 찾아가는 과정

그러나, 최적해를 찾을 수 있는 w가 있다는 것이 증명이 안됨

그래서 '확률적 경사 하강 기법'을 사용함

- 최적화 (Optimizer)



역전파를 이용해 각 파라미터(가중치;

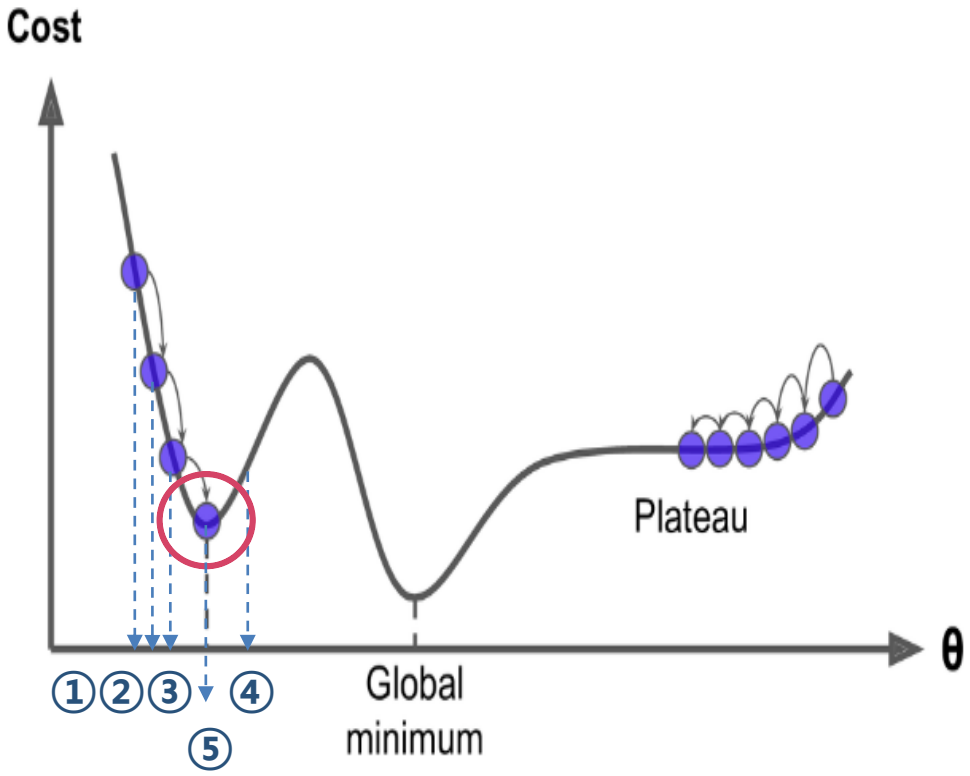
확률적 경사 하강 기법

최적의 값이 있을 것이라 예측되는
그러나, 최적해를 찾을 수 있는 w가 있
방향으로 조금씩 이동함
다는 것이 증명이 안됨

그래서 '확률적 하강 기법'을 사용함

- 최적화 (Optimizer)

국소 최적해

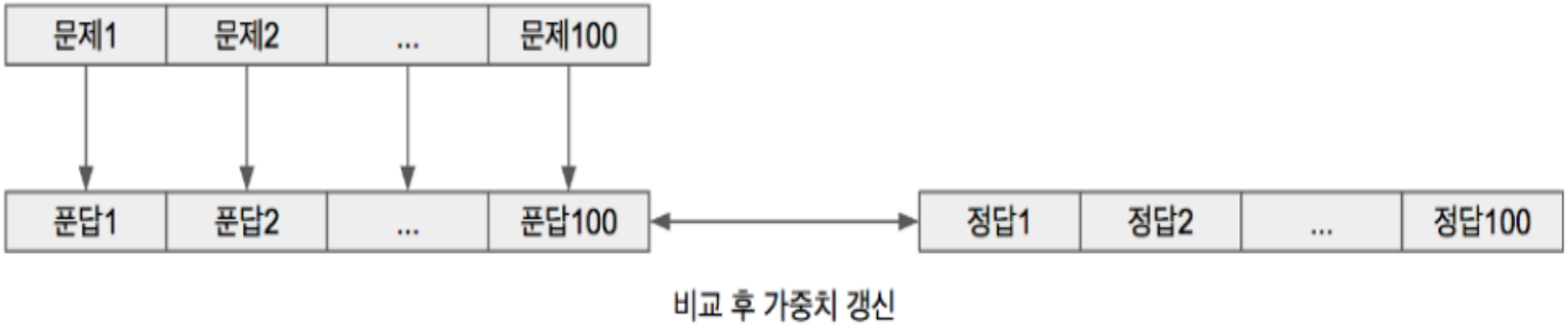


오른쪽 위치의 언덕을 넘어 해를 확인하지 않고, Local minimum을 손실함수를 최소화하는 가중치라 여기게 됨

- 미니 배치 (Mini-Batch)

Batch 학습

: 전체 데이터를 사용하여 학습(가중치 업데이트) 하는 방식



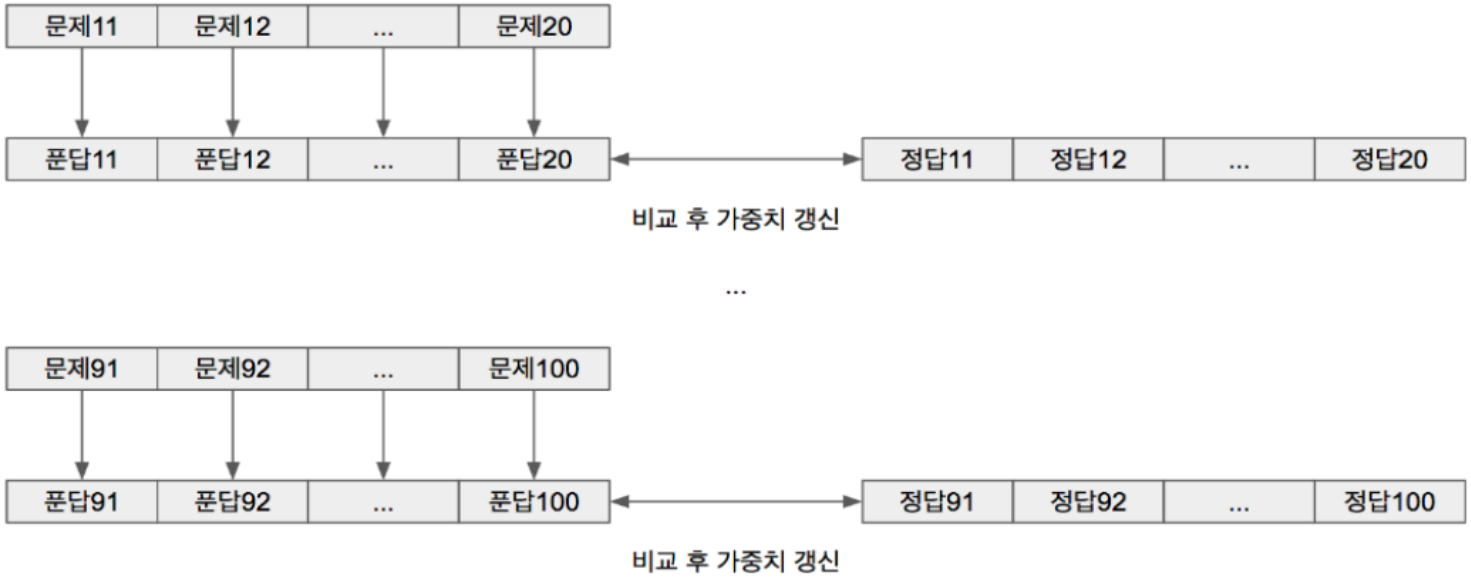
가중치가 최적화 될 때 까지 전체 데이터를 반복하기 때문에

학습 시간이 오래 걸림

- 미니 배치 (Mini-Batch)

Mini-Batch

: 전체 데이터 중 **Batch Size만큼만** 사용하여 학습을 진행



현실적이고 빠른 학습 가능

- 미니 배치 (Mini-Batch)

Mini-Batch

- **Batch size**: 미니 배치 학습에서 한번에 학습할 데이터 수
- **Epoch**: 전체 데이터셋에 대한 1회 학습
- **Iteration**: epoch를 나누어 실행하는 횟수

ex. 100 obs를 Batch size=5으로 학습 → 20 iteration

- 미니 배치 (Mini-Batch)

Mini-Batch

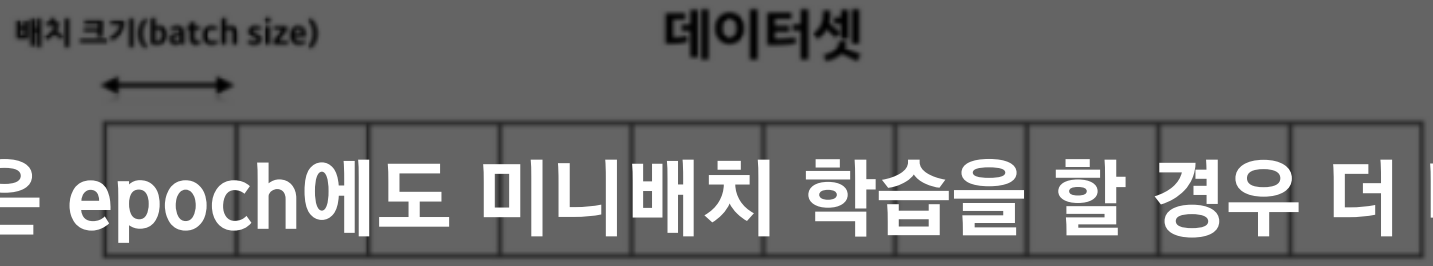
〈 ex. 1000 epoch로 학습할 경우 〉

- 배치학습 : 1000번 업데이트

- 미니 배치 10개로 학습(Iteration=10) : 10000(1000x10)번 업데이트

- 미니 배치 (Mini-Batch)

Mini-Batch



같은 epoch에도 미니배치 학습을 할 경우 더 빠른 속도로 최적의 가중치를 찾을 수 있음 !

〈 ex. 1000 epoch로 학습할 경우 〉

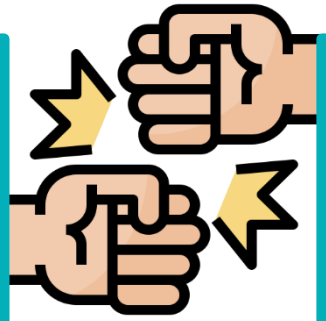
- 배치학습 : 1000번 업데이트

- 미니 배치 10개로 학습(Iteration=10) : 10000(1000x10)번 업데이트

- 미니 배치 (Mini-Batch)

배치 학습

- 느린 학습
- 가장 일반화된 모델



미니 배치 학습

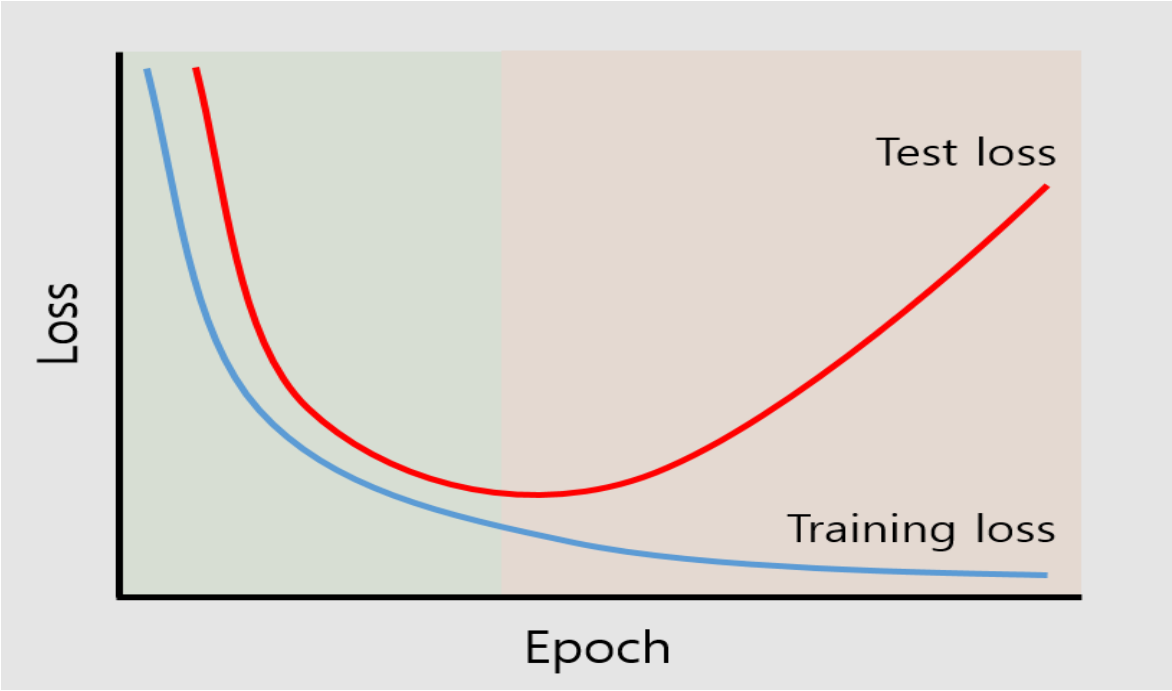
- 빠른 학습 가능
- 편향된 모델로 학습될 수 있음

3

모델 성능 향상시키기

3 모델 성능 향상시키기

- **오버피팅이란?**

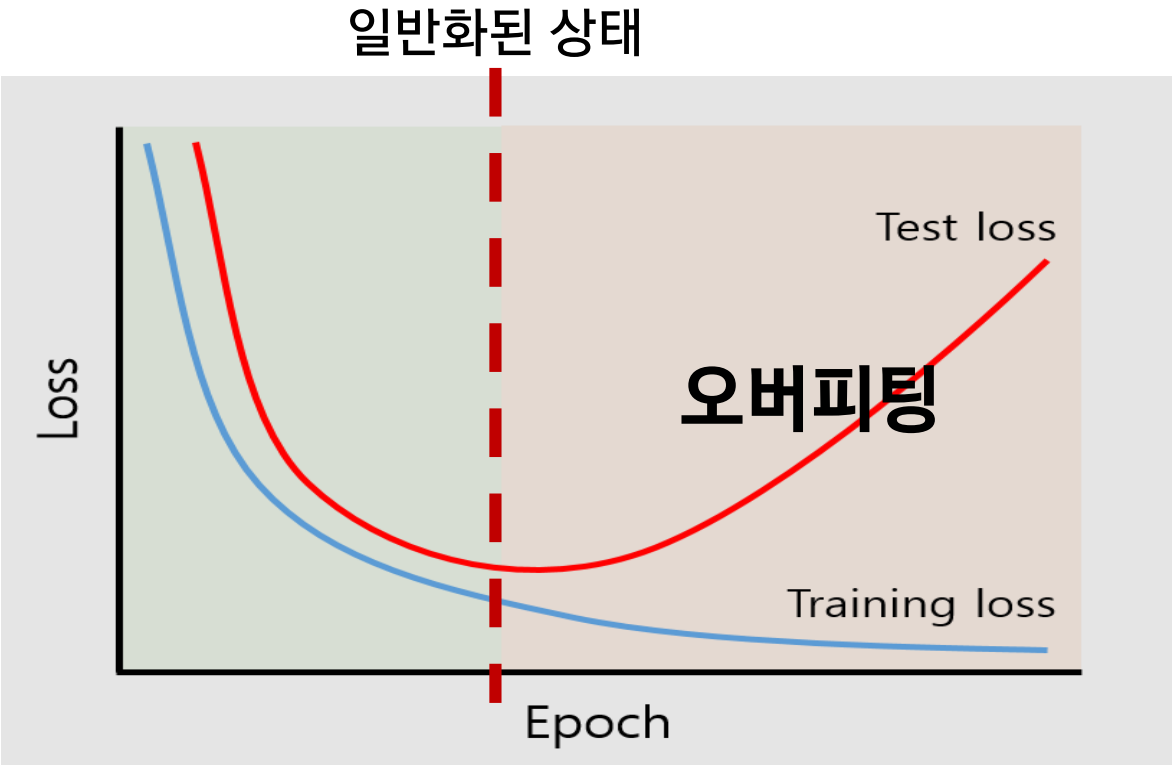


학습을 반복할수록 Train에 대한 성능은 향상되지만,

Test에 대한 성능은 일정 수준을 넘으면 저하되는 것

3 모델 성능 향상시키기

- **오버피팅이란?**



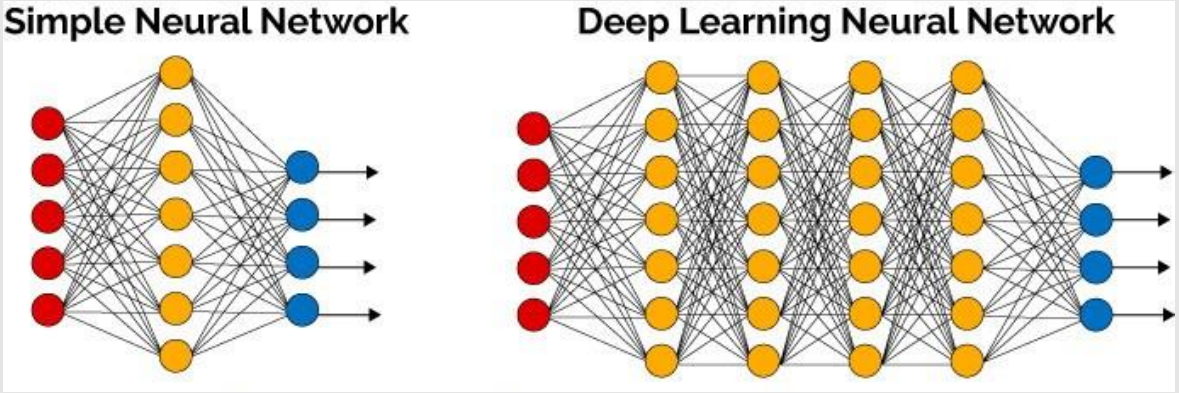
우리의 목표는 어떤 데이터를 주어도

균등한 성능을 내는 일반화된 모델을 만드는 것

● 오버피팅 피하기

데이터 수, 모델 구조

: 데이터 수가 많을수록, 모델 구조가 단순할수록 과적합 위험↓



- 데이터 수는 조절할 수 없기에 모델 구조를 조절
- 복잡한 문제의 경우 적절히 복잡한 모델을 사용해야함

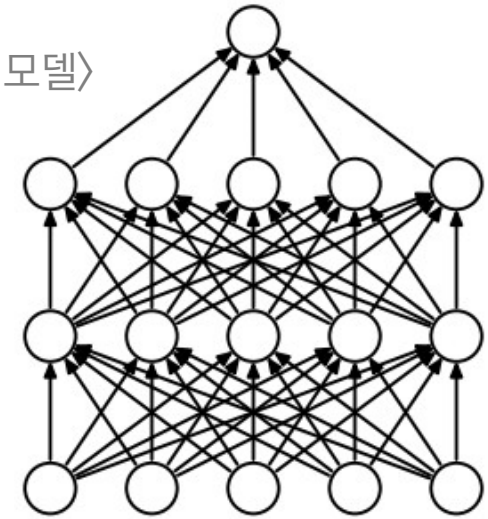
➡ 모델 구조를 바꾸어 가면서 비교해야 함.

● 오버피팅 피하기

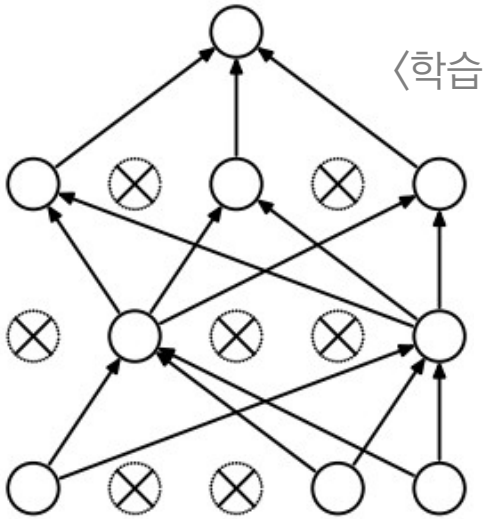
Dropout

: 모든 노드를 사용하지 않고, 일정 비율의 노드를 이용해 학습함

〈전체 모델〉



〈학습 시 모델〉



학습 시 모델구조는 단순하고,
전체 모델은 문제에 적합할 정도로 복잡해짐.

- 오버피팅 피하기

정규화

: 모델이 복잡해 질수록 손실함수의 값이 커지는 경향을 갖도록 만들어줌

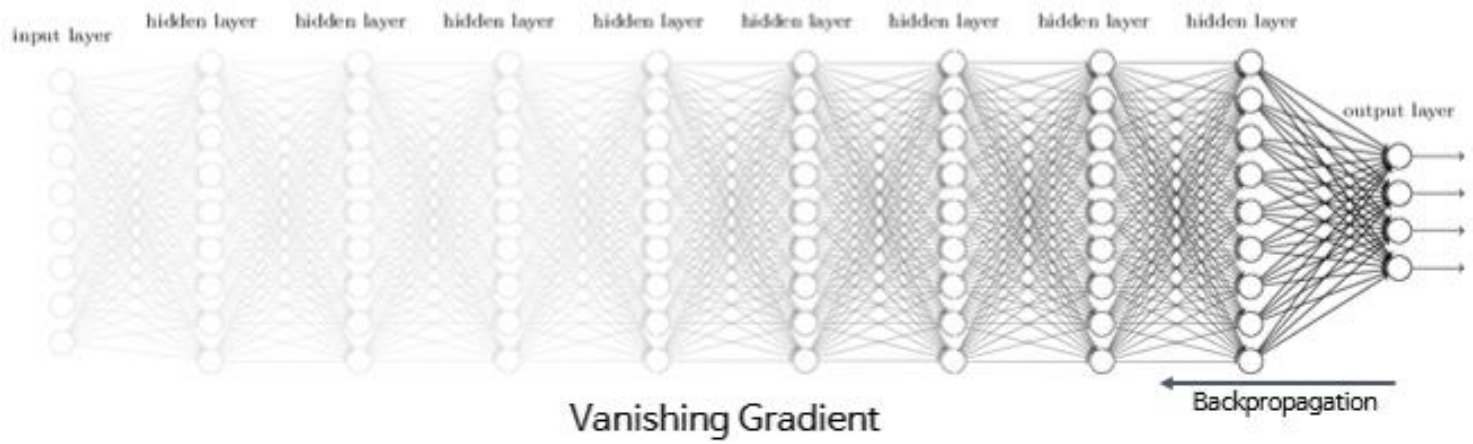
$$E = E_0 + \frac{\lambda}{2}\Sigma w^2$$

➡ 가중치들이 작아지는 경향을 가지게 됨.

- 중요한 문제

기울기 소실 문제

: Vanishing Gradient Problem

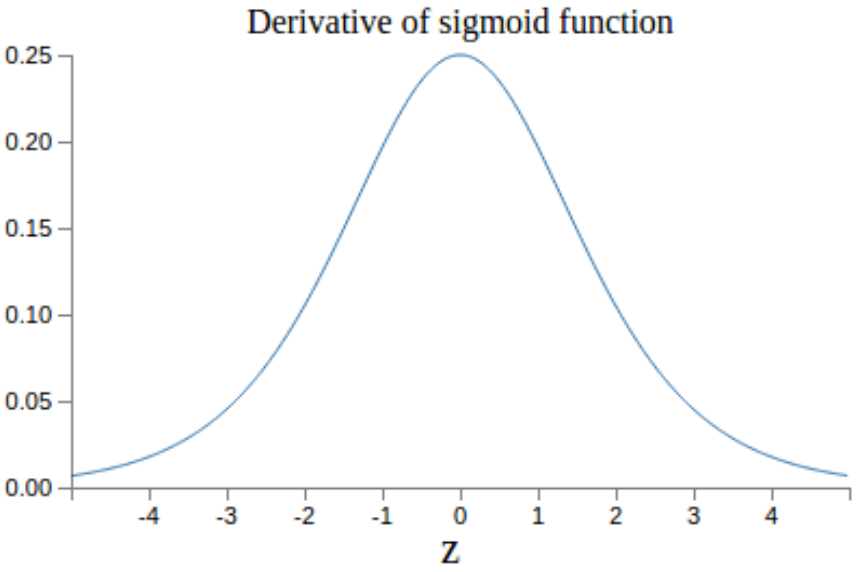


층이 깊은 심층신경망에서 역전파시 gradient가 입력층으로 전달됨에 따라 점점 작아져 가중치가 업데이트 되지 않는 것

● 중요한 문제

기울기 소실 문제

: Vanishing Gradient Problem



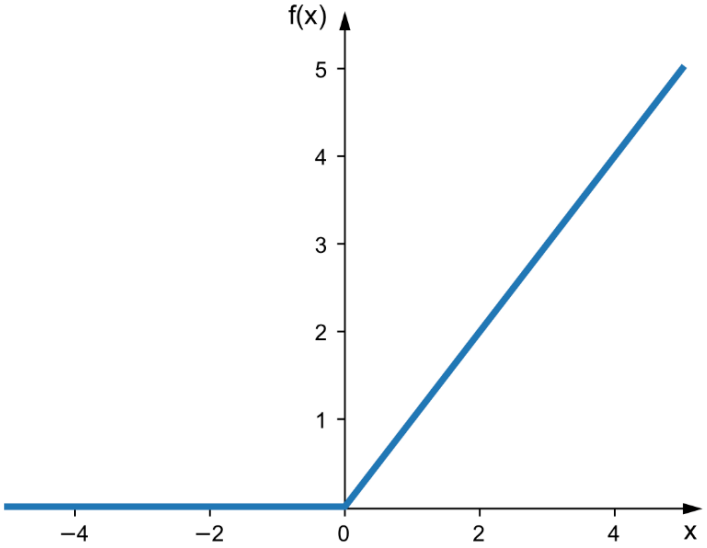
기존에 사용하던 sigmoid 함수의 미분 값은 최대 0.3이 되지 않았고,

작은 값이 수십 번 곱해지자 값이 결국 사라지게 됨.

- 중요한 문제

기울기 소실 문제

: Vanishing Gradient Problem



ReLU 함수를 사용하면 기울기가 1또는 0으로 기울기 소실 문제가 완화된다.

최근 ReLU외에도 다양한 활성화 함수가 존재!

- 중요한 문제

가중치 초기화 : Weight Initialization

Uniform distribution

$$r = \sqrt{\frac{3}{n_{in}}}\,[LeCun1998]$$

$$r = \sqrt{\frac{6}{n_{in} + n_{out}}}\,[Glorot2010]$$

$$r = \sqrt{\frac{6}{n_{in}}}\,[KaimingHe2015]$$

Gaussian distribution

$$r = \sqrt{\frac{1}{n_{in}}}\,[LeCun1998]$$

$$r = \sqrt{\frac{2}{n_{in} + n_{out}}}\,[Glorot2010]$$

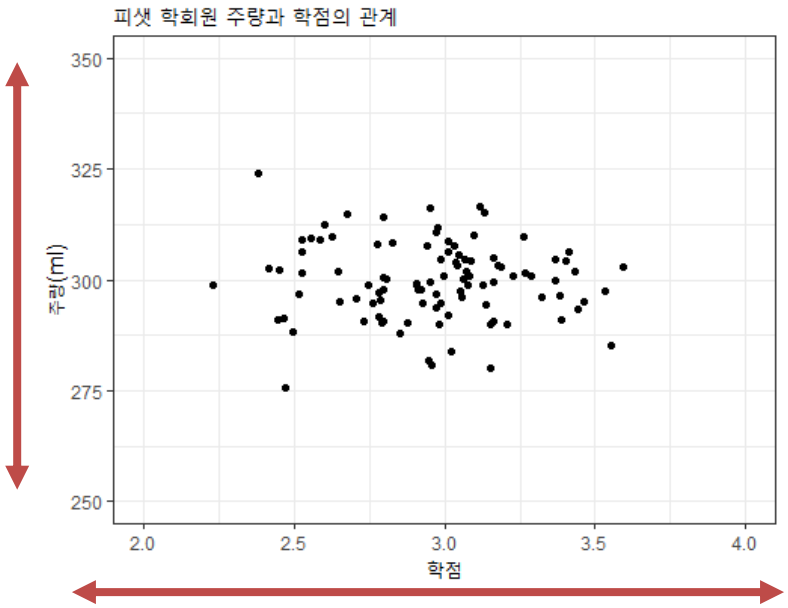
$$r = \sqrt{\frac{2}{n_{in}}}\,[KaimingHe2015]$$

가중치 초기화를 최적의 가중치 범위에 가깝게 하면 더욱 빠른 학습이 이뤄짐

∴ 위의 r을 이용하여 (-r, r) 범위로 초기화를 하는 것이 권장됨.

- 중요한 문제

Normalization



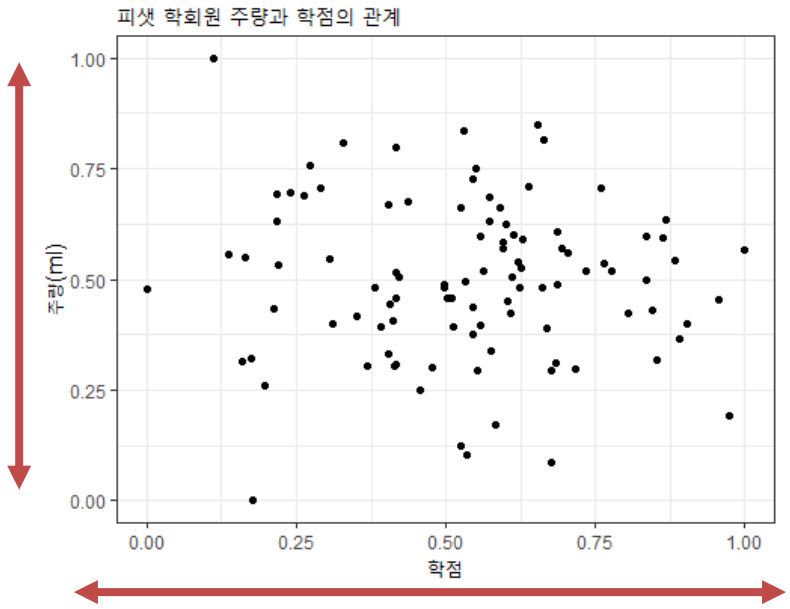
변수간 범위 차이가 많이 날 경우 모든 변수의 중요도가

동일하지 않게 취급되는 문제가 발생

- 중요한 문제

Normalization

〈 대표적인 Normalization 방법인 Min-Max Scaler 〉



$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

범위는 줄이면서 변수내의
관측치간 거리는 유지됨

4

딥러닝의 특징

4 딥러닝의 특징

1) 계산량이 아주 많다

Chain Rule을 이용 미분 곱셈을 반복하기에 계산량이 매우 많다.

또, 데이터셋이 1만건 이상일 때 효과적이기에 더욱 계산량이 많다.

2) 해석이 불가능하다

선형회귀의 경우 특정 변수와 y의 관계를 명확하게 확인할 수 있지만
딥러닝은 수많은 노드와 층으로 구성되어 있어 해석이 불가능하다.

3) 복잡한 비선형 관계를 잘 파악한다

$$f(x) = 2x \Rightarrow f\left(f\left(f(x)\right)\right) = 8x$$

굳이 세개의 식을 연결할 필요 없음

$$f(x) = 2x^2 - 2 \Rightarrow f\left(f\left(f(x)\right)\right) = 2(2(2x^2-2)^2-2)^2 - 2$$

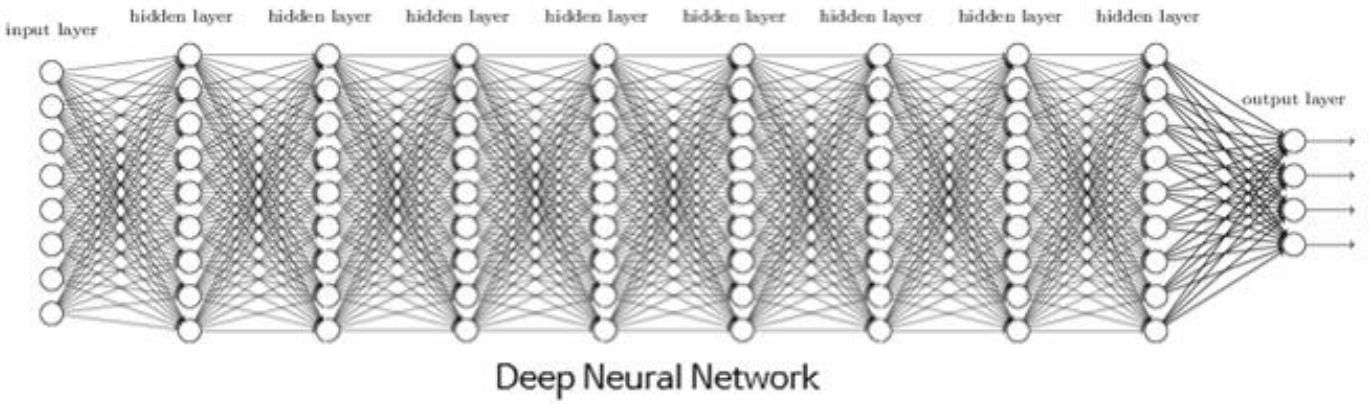
세개의 식을 연결하여 복잡한 비선형 관계를 표현할 수 있음

딥러닝은 비선형 활성화함수 f를 f(f(f(x)))로 연결하여 복잡한 비선형 관계를 표현한다. 비선형 관계의 장점은 조합에 따라 함수가 달라진다는 점이다.

4 딥러닝의 특징

3) 복잡한 비선형 관계를 잘 파악한다

➡ 기존에 사용할 수 없었던 비정형 데이터 사용 가능!

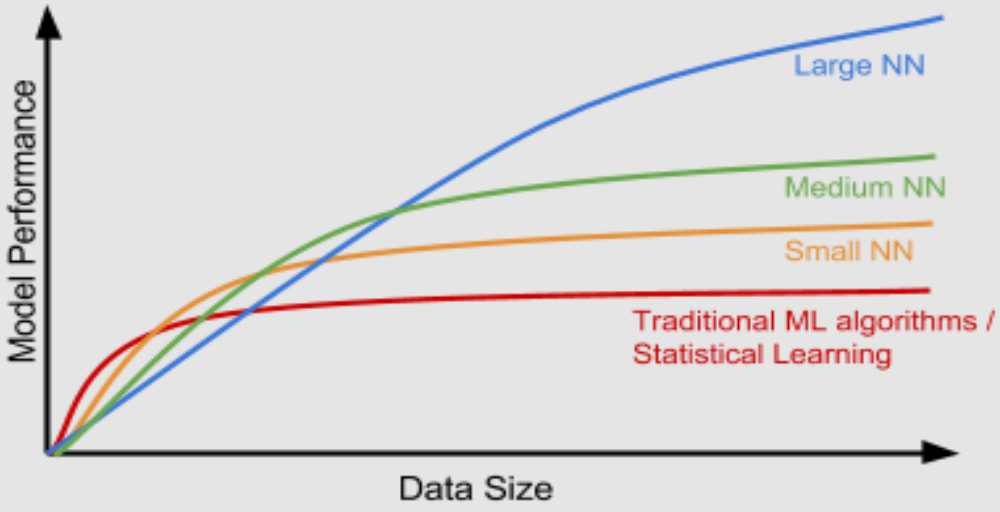


Deep Neural Network

Hidden Layer는 변수간 비선형 관계를 파악하여 조합을 통해 잠재적인 변수를 생성

Ex. 거래 중개인의 감정 변수를 거래일 날씨, 여의도 교통혼잡도로 추측

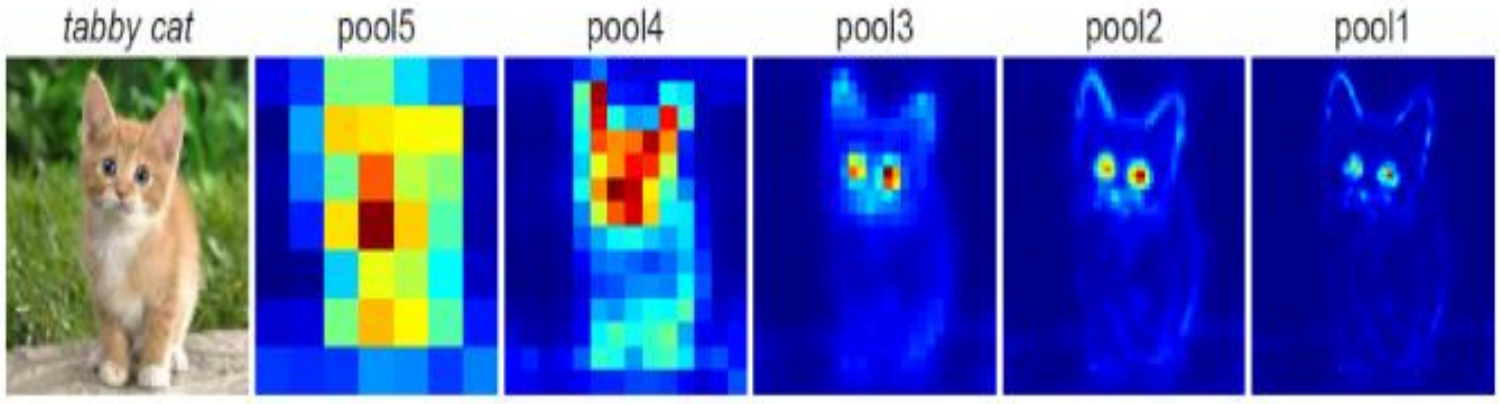
4) 많은 데이터를 필요로 한다



딥러닝 모델은 데이터의 다양성이 중요하다. 데이터의 수가 많으면 편향될 가능성이 적어지고, 일반화된 문제 해결이 가능해진다.

5) Feature Selection을 하지 않아도 된다

〈고양이 판별 문제의 heatmap〉



Heatmap을 통해 모델이 고양이를 판별하는데 중요하게 작용하는 귀의 형태, 눈과 코를 잘 선택했음을 알 수 있다. 학습 과정에서 불필요한 변수에 낮은 가중치를 부여하기에 변수 조정이 필요 없다.

6) 만능이 아니다

데이터가 적거나 문제가 복잡하지 않을 경우 딥러닝이 좋지 않을 수 있다. 하지만 비정형데이터를 잘 다루기 위해서는 딥러닝이 좋다 !

THANK YOU