

01_딥러닝 기초

딥러닝 라이브러리:
텐서플로, 케라스,파이토치

1. Numpy 사용하여 딥러닝 기본구조 학습
2. 딥러닝의 알고리즘 학습
: GPT, CNN, RNN,L STN 등등

텐서: 3차원배열

- 딥러닝이란?

- > 인공지능

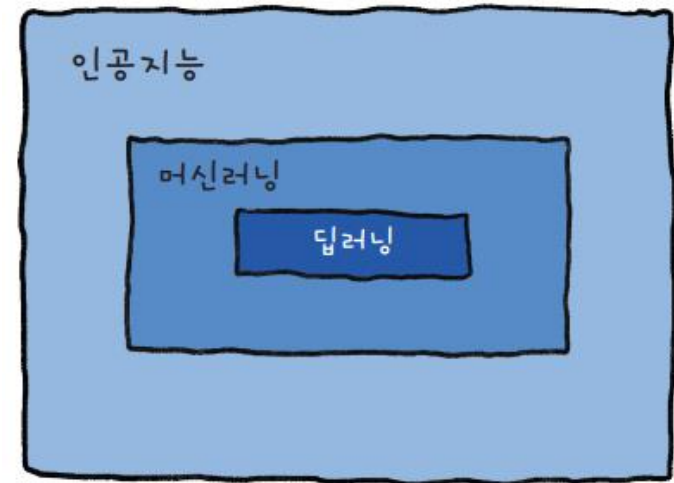
: 문제를 인식하고 해결하는 능력인 지능을 구현하는 기술

- > 머신러닝

: 기계 스스로 학습하여 지능을 습득하는 기술

- > 딥러닝

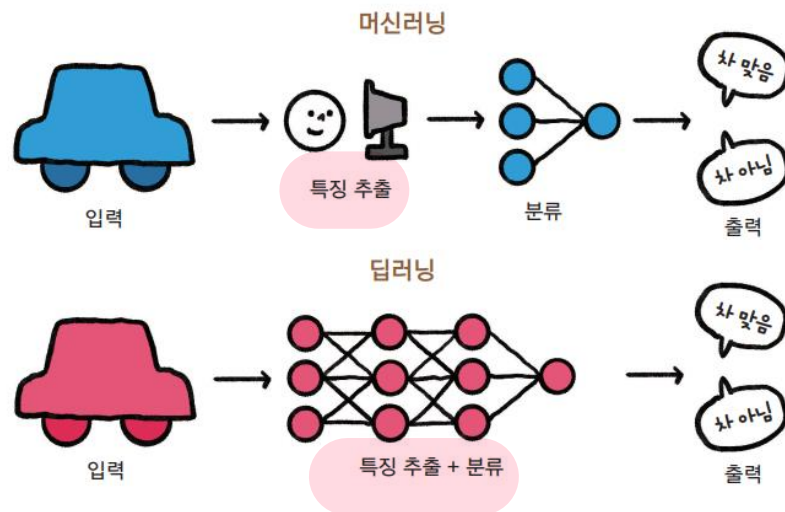
: 생체 신경망을 모방해서 만든
인공 신경망을 이용하여 복잡한
데이터 관계를 찾아내는 기법



• 딥러닝의 장점과 한계

> 장점

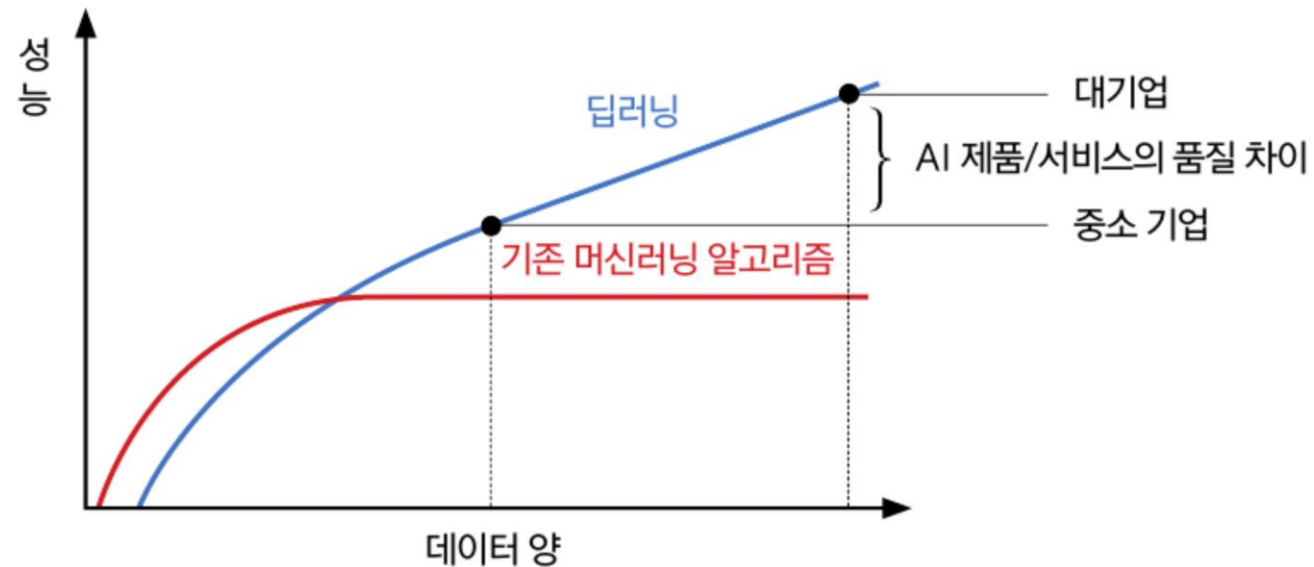
- 함수를 근사하는 능력이 뛰어나다.
- 특징을 자동으로 추출한다.
- 모델의 확장성이 뛰어나다.
- 머신러닝보다 좋은 성능을 보인다.



• 딥러닝의 장점과 한계

> 한계

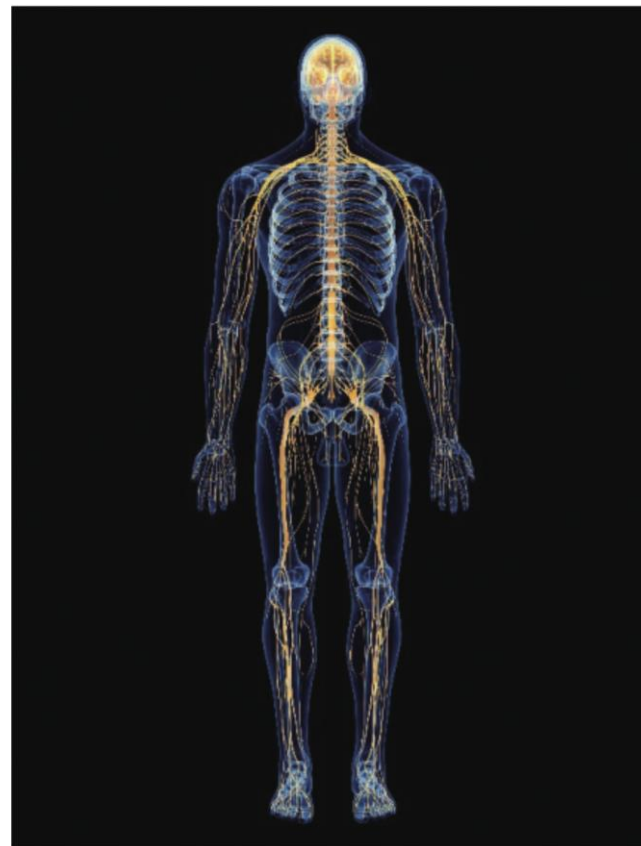
- 머신러닝 모델보다 상대적으로 많은 데이터가 필요하다. 단, 메모리 문제 없음
- 훈련을 위한 시간과 비용이 많이 든다.
- 인공 신경망 모델은 오류를 파악하기가 어렵다.



• 인공 신경망의 탄생

> 뇌, 신경망을 사용한 지능 학습

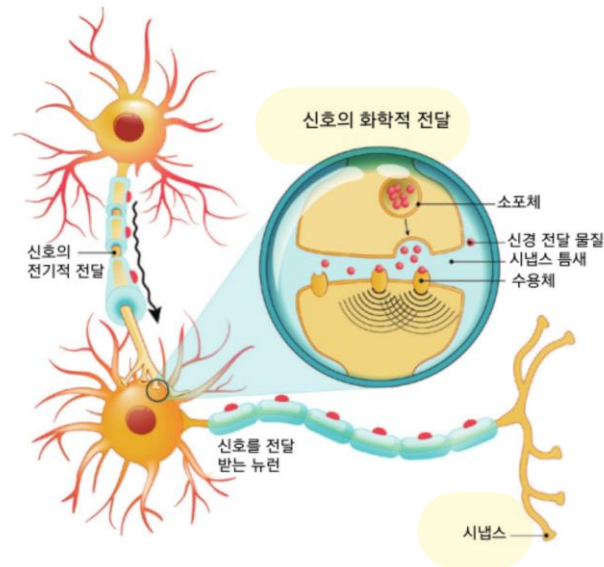
- 경험을 토대로 인간은 학습을 하고 처리를 한다.
- 경험은 뇌에 저장되고, 뇌의 기능 단위는 뉴런이다.
- 생체 신경망으로 이루어진 신체는 자극(오감)을 뇌에 전달하고, 뇌는 기억으로 저장된 경험을 활용해서 새로운 자극에 대해 판단하여 판단 결과를 운동 기관에 명령하여 반응한다.



• 인공 신경망의 탄생

> 생체 신경망과 뉴런의 신호 전달 과정

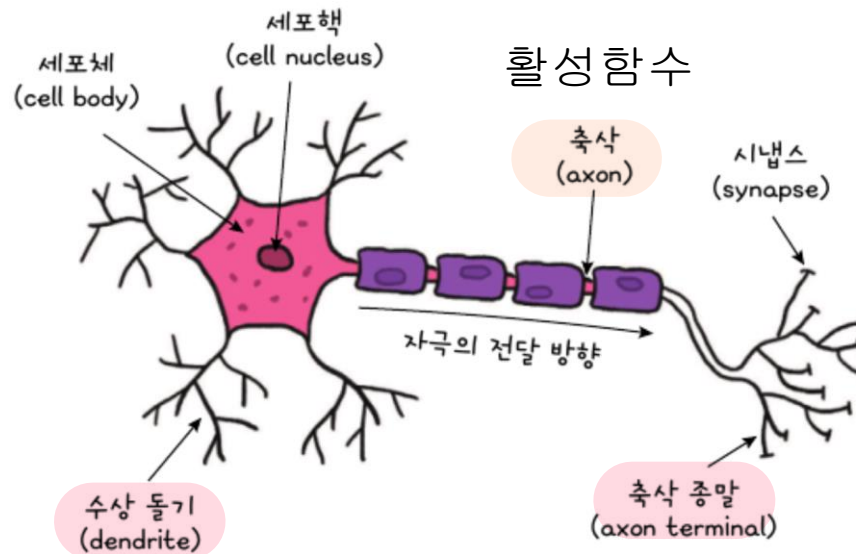
- 외부에서 자극이 들어오는 순간 전기 신호로 변환된다.
- 변환된 전기 신호는 뉴런을 통과할 때, 이온 통로를 통해 전기 신호 형태로 전달한다.
- 다음 뉴런으로 신호를 전달할 때는 화학적인 방법을 통해 전기 신호를 변환한다.



• 인공 신경망의 탄생

> 뉴런의 구조

- 수상 돌기를 통해 신호를 전달받는다.
- 받은 신호는 세포체로 모이고 특정 수가 넘으면 전기 신호로 변환하여 시냅스로 전달한다.
- 시냅스에서는 전기 신호를 화학 물질로 변환하여 다음 뉴런으로 전달한다.
- 전기 신호를 전달하는 활동 상태에 있으면 뉴런이 활성화된다고 한다.



• 딥러닝의 역사

> 인공지능의 연대기

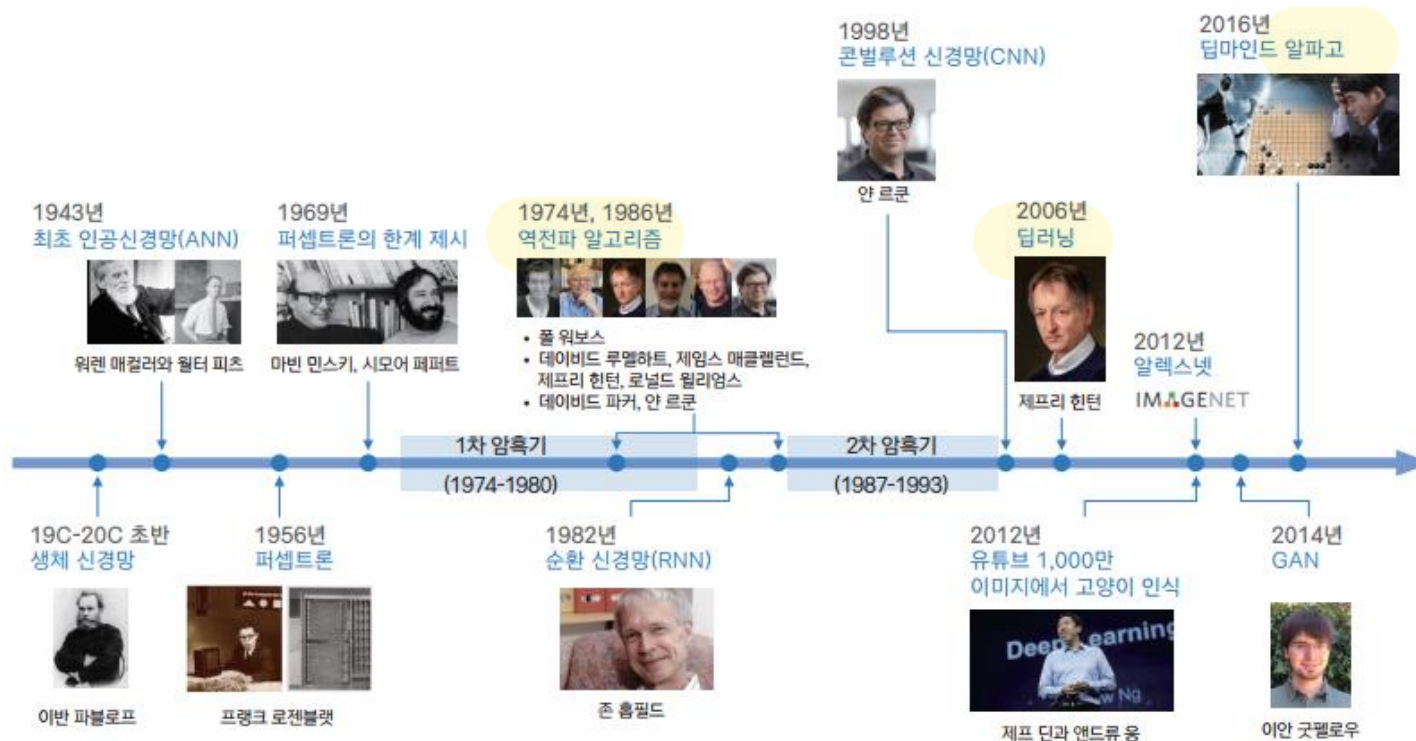


그림 1-14 인공 신경망의 70년 역사

• 딥러닝의 역사

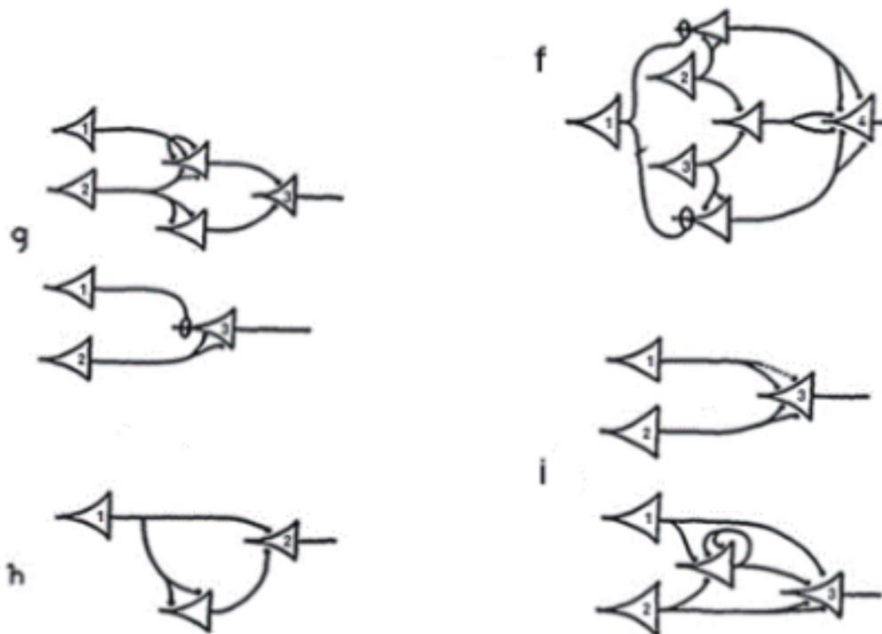
> 최초의 인공 신경망 : 매컬러-피츠 모델

- 매컬러, 피츠는 최초의 인공 신경망을 구성한 사람이다.
- 이들이 정의한 인공 신경망 모델은 활성, 비활성으로 구분하는 이진 뉴런이다.
- 생체 뉴런과 같이 신호가 전달되고 특정 임계치를 넘어야 신호가 발화한다.
- 이진 뉴런으로 구성된 신경망은 임의의 논리 연산과 산술 연산이 가능하다는 것을 보여주고 뇌가 잠재적으로 매우 강력한 연산 장치임을 증명하고자 했다.

• 딥러닝의 역사

> 최초의 인공 신경망 구조

- 이진 뉴런으로 구성된 신경망 모델은 현대 신경망 구조의 시초가 되었고 아직까지 이런 구조를 유지하고 있다.



• 딥러닝의 역사

> 학습하는 인공 신경망 : 퍼셉트론

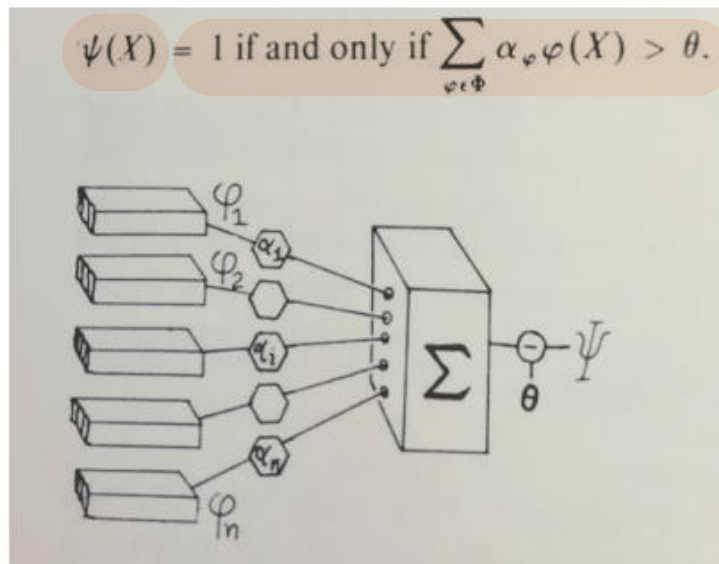
- 매컬러-피츠 모델은 학습 과정없이 연산을 하기에 문제에 따라 구조를 수정해야 했다.
- 이를 해결하기 위해 프랭크 로젠블랫은 헵의 학습 가설에 따라 인공 신경망이 스스로 문제에 맞춰 학습하는 모델인 퍼셉트론을 개발했다.
- 퍼셉트론의 학습 알고리즘은 새로운 입력에 대한 오차가 발생하면 뉴런의 연결 강도를 조절하는 방식이다.

• 딥러닝의 역사

012

> 퍼셉트론의 구조

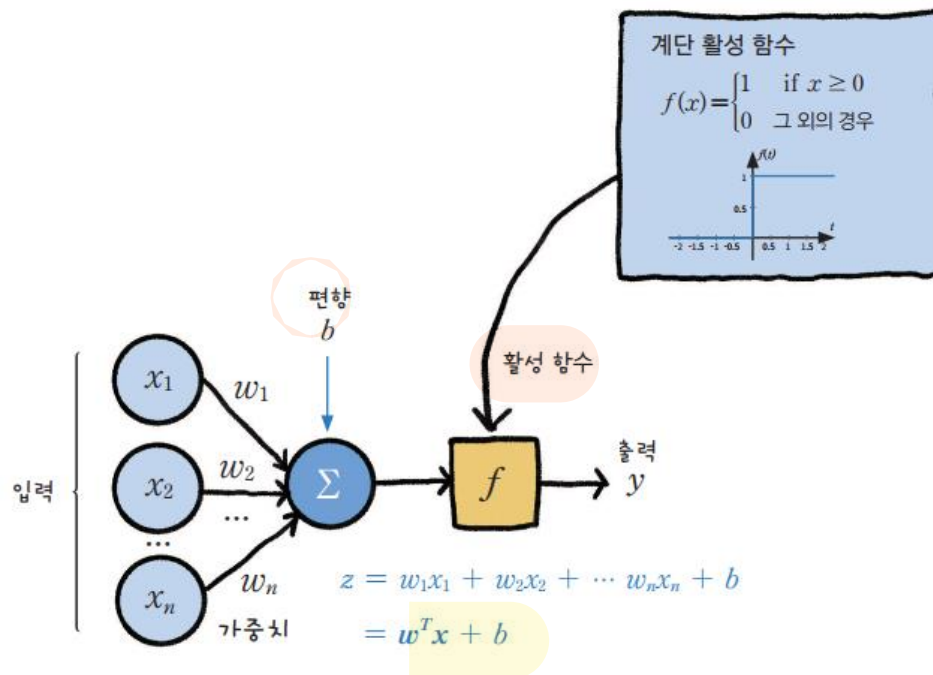
- 퍼셉트론은 입력 데이터가 들어오면 가중치와 곱해서 가중 합산을 하며 그 결과가 0보다 크면 1을 출력하고 그렇지 않으면 0을 출력한다.
- 0과 1로 나타내는 함수를 활성화 함수라고 부른다.



• 딥러닝의 역사

> 퍼셉트론의 구조

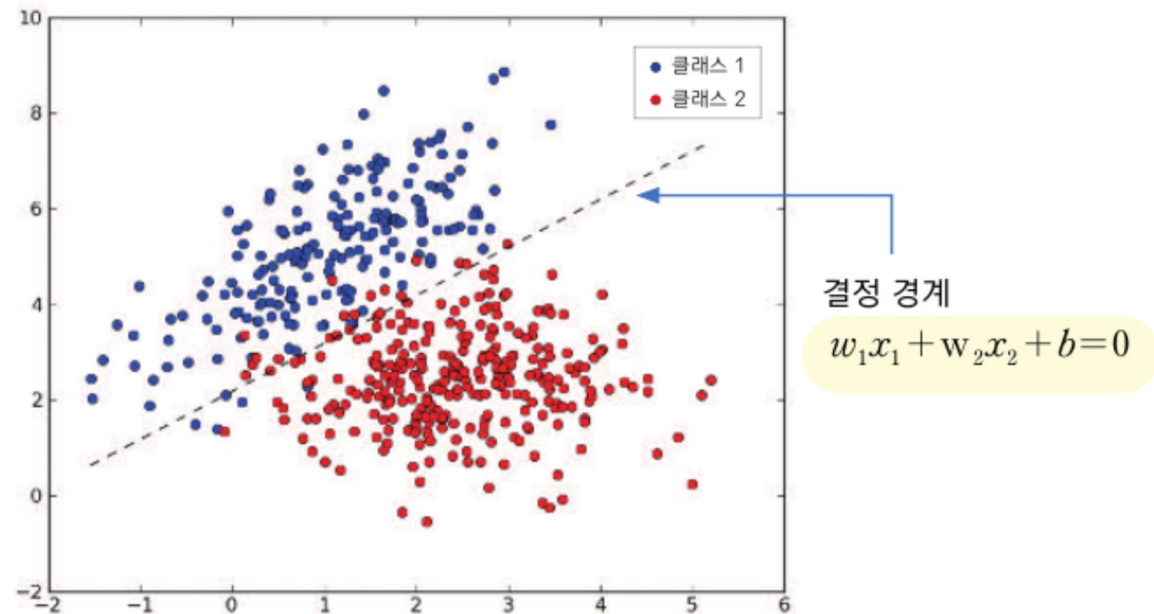
- 이 구조는 현재의 인공 뉴런까지 변하지 않고 그대로 유지되고 있다.
- 단, 활성화 함수는 다양하게 변화해왔다.



• 딥러닝의 역사

> 퍼셉트론의 구조

- 단일 퍼셉트론은 두 종류의 클래스를 직선으로 분류하는 선형 분류기로, 입력 데이터와 가중치의 합산 식은 두 클래스를 분류하는 결정 경계를 이룬다

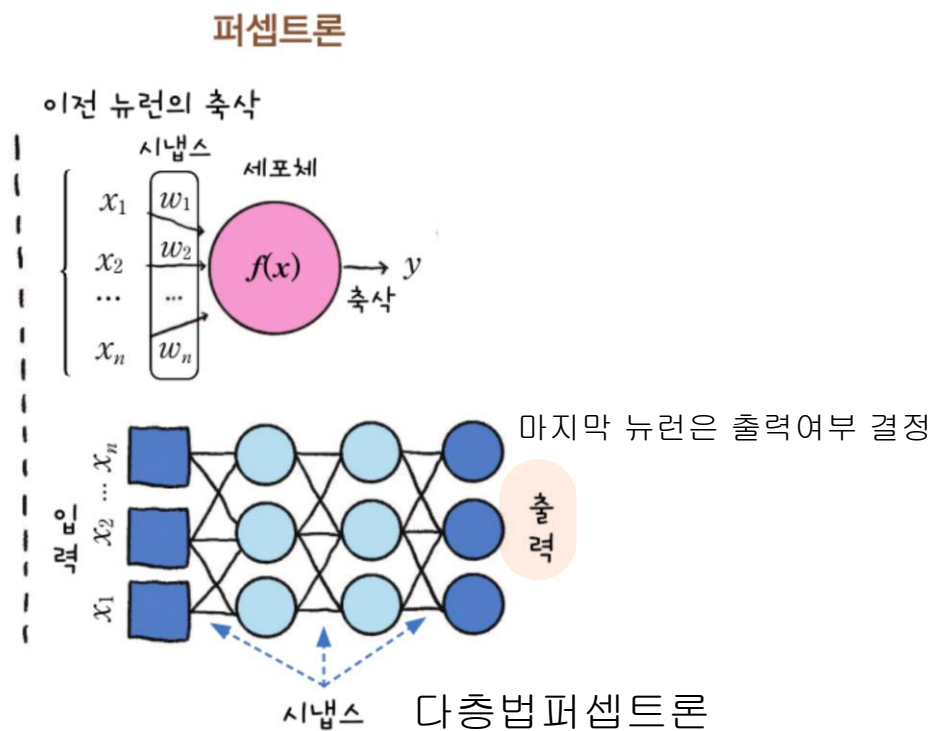
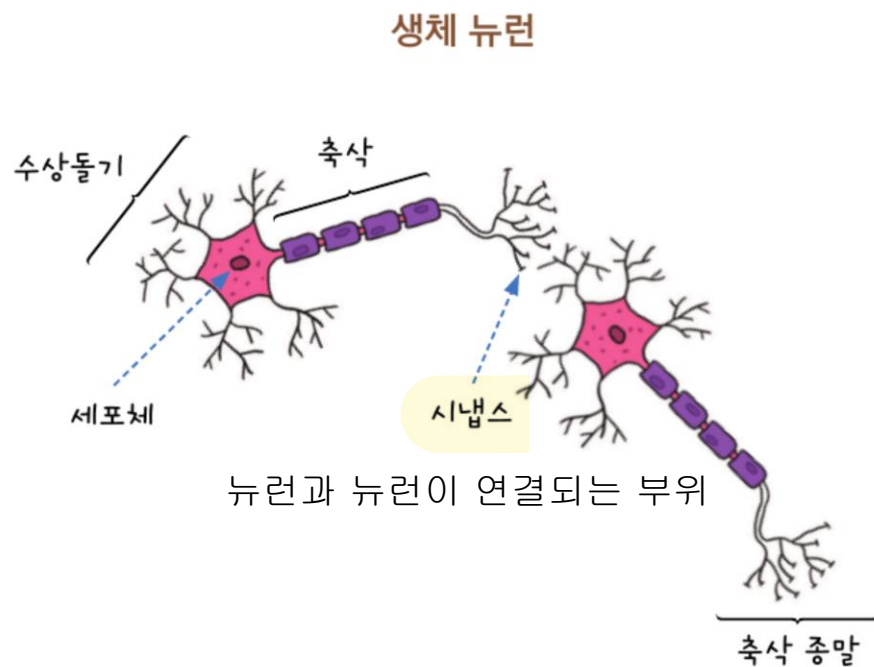


• 딥러닝의 역사

015

> 생체 뉴런과 퍼셉트론 비교

- 퍼셉트론은 생체 뉴런을 모방해서 만들었기에 구조가 똑같다.



• 딥러닝의 역사

016

> 생체 뉴런과 퍼셉트론 비교

- 구조를 정리하면 다음과 같다.

	퍼셉트론	생체 신경망
입력	$\mathbf{x}^T = (x_1, x_2, \dots, x_n)$	이전 뉴런이 발화한 신호
가중치	$\mathbf{w}^T = (w_1, w_2, \dots, w_n)$	시냅스의 연결 강도
입력 데이터와 가중치의 곱	$w_i x_i (i=1, 2, \dots, n)$	시냅스의 연결 강도에 따라 신호가 강해지거나 약해지는 과정 가중치가 0에 가까울수록 약한 데이터
가중 합산	$z = \sum_{i=1}^n w_i x_i + b$	세포체에서 수상 돌기를 통해 들어온 신호를 모으는 과정
활성 함수	$f(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{그 외의 경우} \end{cases}$	세포체의 신호 발화 과정 화학신호를 발화신호로 바꾸는 과정: 이를 통해 전달여부 결정
출력	$f(\mathbf{x}) = f(\mathbf{w}^T \mathbf{x} + b)$	축삭을 따라 시냅스로 전달되는 과정

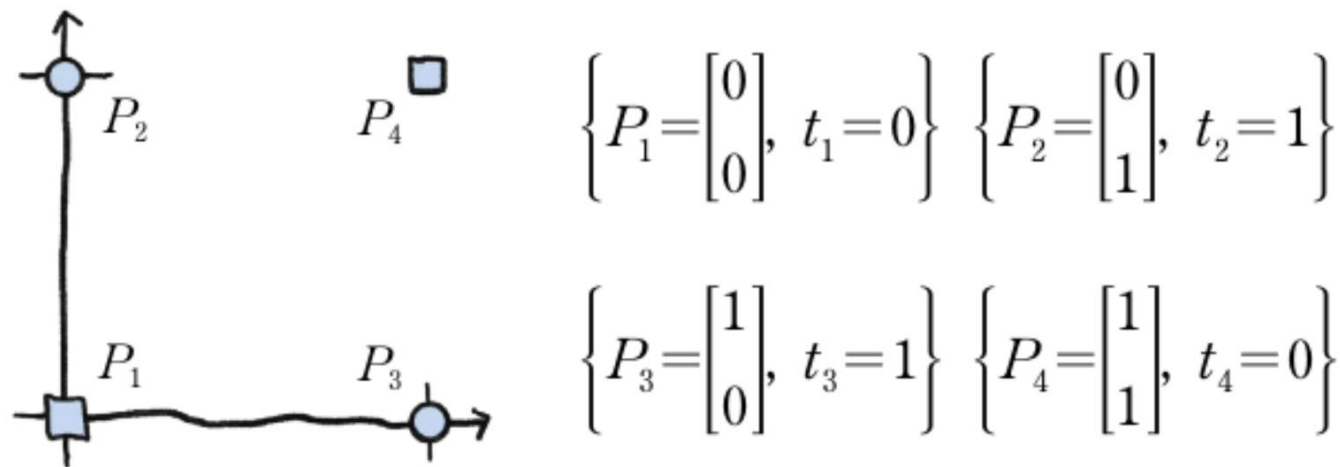
• 딥러닝의 역사

017

> 퍼셉트론의 한계

- 과학자들은 퍼셉트론이 비선형 문제를 풀 수 없다는 한계를 XOR 논리 연산을 통해 증명했다.
- P1, P4와 P2, P3를 분류하는 문제이다.

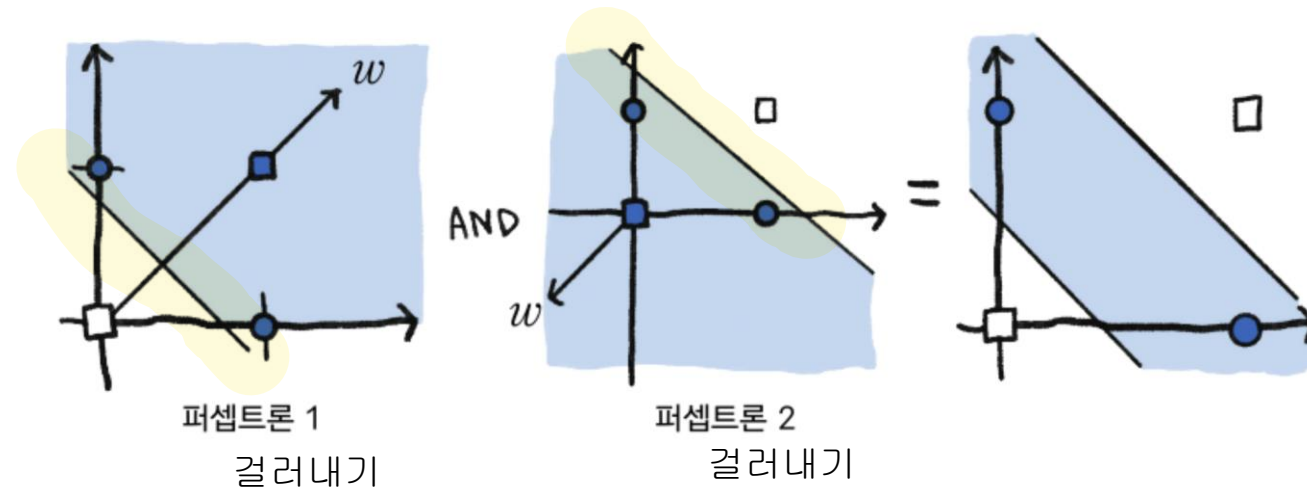
: 다층퍼셉트론으로 해결



• 딥러닝의 역사

> 퍼셉트론의 한계

- P2, P3를 따로 분류하려면 띠 모양의 영역이 참이 되어야 한다.



- 이러한 문제를 해결하기 위해선 퍼셉트론 간의 AND 연산을 표현해야하지만 퍼셉트론은 하나의 계층만 표현가능하기에 불가능했다.

• 딥러닝의 역사

> 퍼셉트론의 한계

- 이를 해결하기 위해서는 다층 퍼셉트론으로 확장하여야한다.
- 그러나 다층 퍼셉트론을 학습하는 방법은 찾지 못하고 퍼셉트론은 역사 속으로 사라졌다.

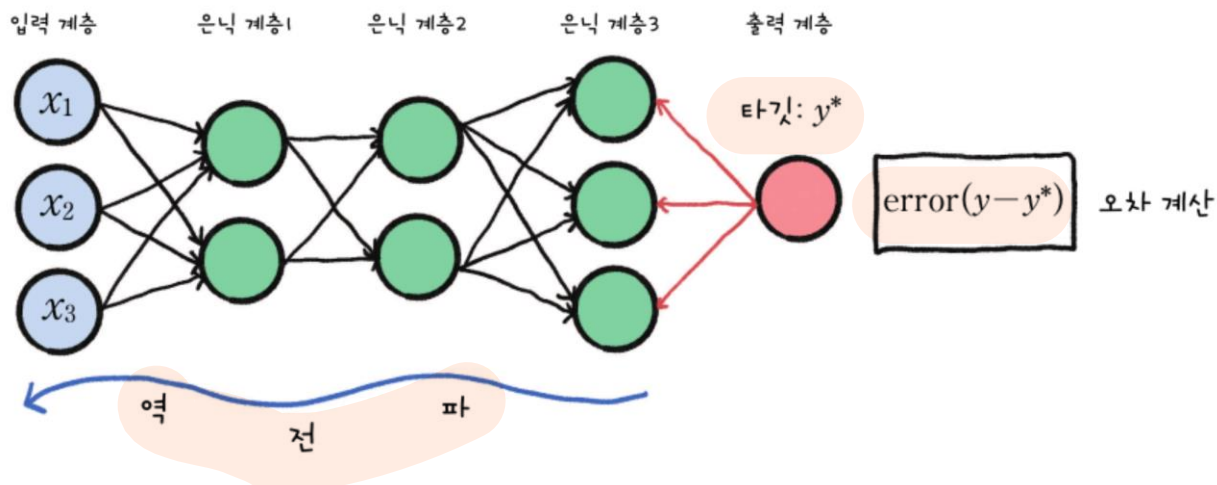
> 역전파 알고리즘

- 이후 다층 퍼셉트론을 학습시킬 수 있는 역전파 알고리즘이 개발되었다. 그러나 인공지능의 침체기로 알려지지 않고 오랜 시간이 지난 뒤에야 세상에 발표된다.

• 딥러닝의 역사

> 역전파 알고리즘

- 인공 신경망은 학습 과정에서 **출력과 정답의 오차를 최소화**하도록 최적화를 수행한다.
- 역전파 알고리즘은 신경망의 뉴런에 분산된 파라미터의 미분을 효율적으로 계산하기 위한 알고리즘이다.
- 출력 계층에서 입력 계층 방향으로 미분을 계산한다.
- 각 파라미터의 오차에 대한 기여도를 계산하여 오차를 최소화하는 방향으로 파라미터를 조정한다.

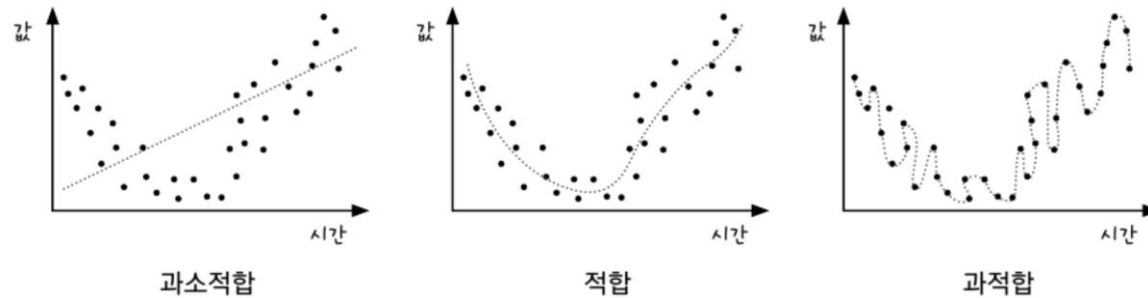


• 딥러닝의 역사

021

> 과적합과 기울기 손실

- 과적합은 학습 데이터에 대해 과도하게 학습되어 학습 데이터는 정확히 예측하지만 새로운 데이터는 예측정도가 떨어지는 상태를 말한다. 규제: L1.L2 사용
- 과적합의 주요 원인은 학습 데이터보다 모델의 파라미터 수가 많기 때문이다.
- 반대로 데이터의 분포를 반영 못하는 상태를 과소적합이라 한다.

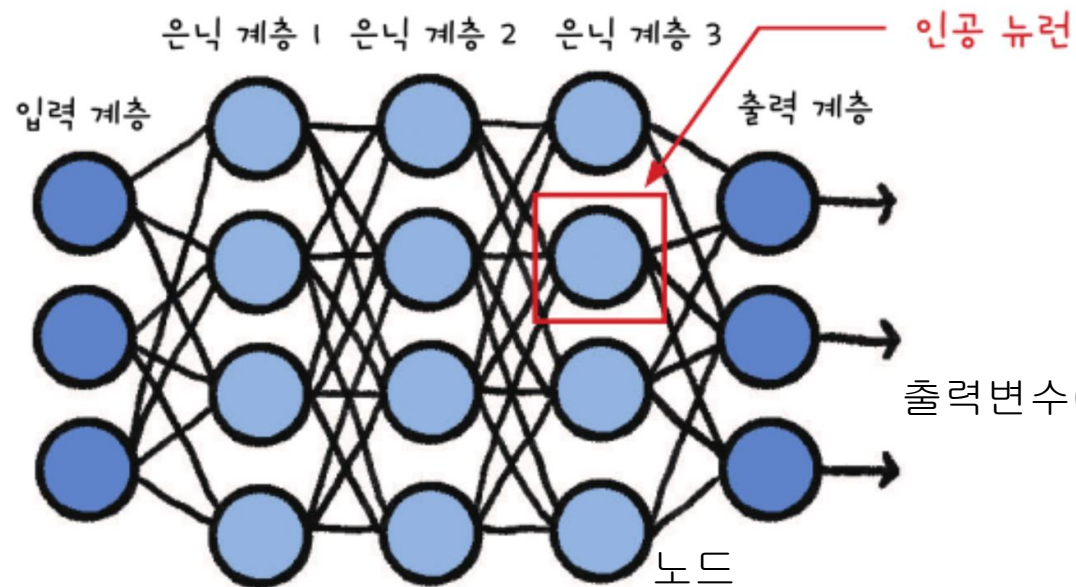


• 순방향 신경망

> 순방향 신경망의 구조

- 다음 그림과 같이 뉴런들이 모여 계층을 이루고 계층이 쌓여 전체 신경망을 이루는 구조를 말한다.
- 입력, 은닉, 출력 계층으로 구분된다.

노드

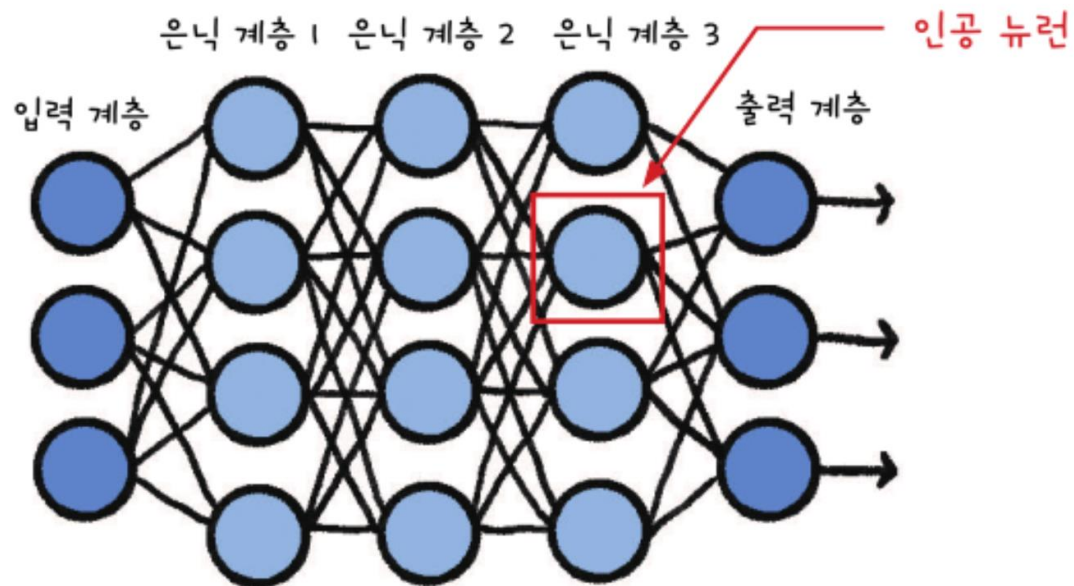


출력변수(y)가 여러개 가능(범주형 변수)

- 순방향 신경망

- > 순방향 신경망의 구조

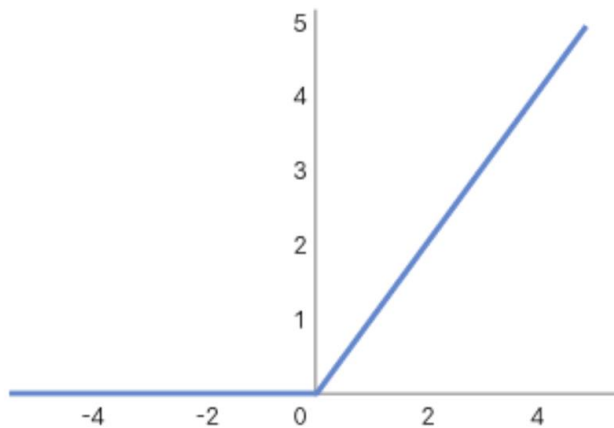
- 입력 계층 : 데이터를 입력받는 계층이다.
 - 은닉 계층 : 입력 받는 데이터의 특징을 추출한다.
 - 출력 계층 : 추출된 특징을 기반으로 추론한 결과를 출력한다.



• 순방향 신경망

> 활성화 함수

- 비선형 변환을 통해 특징을 추출한다.
- 다음은 기본 활성화 함수인 **ReLU**이다.
- 0보다 크면 그대로 값을 출력하고, 0보다 작거나 같으면 0 값을 출력하는 선형 함수이다.



$$f(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{그 외의 경우} \end{cases}$$


가중치를 0으로 만듦

계단 함수
속도 빠름

• 순방향 신경망

> 벡터 함수인 계층

- 계층은 벡터를 입력받고 출력하는 벡터 함수이다.
- 벡터의 크기는 뉴런의 수와 같다.
- 입력받는 뉴런이 N개 출력하는 뉴런이 M개라면 계층 가중치의 크기를 N x M 인 행렬로 정의된다.



목표:값구하기

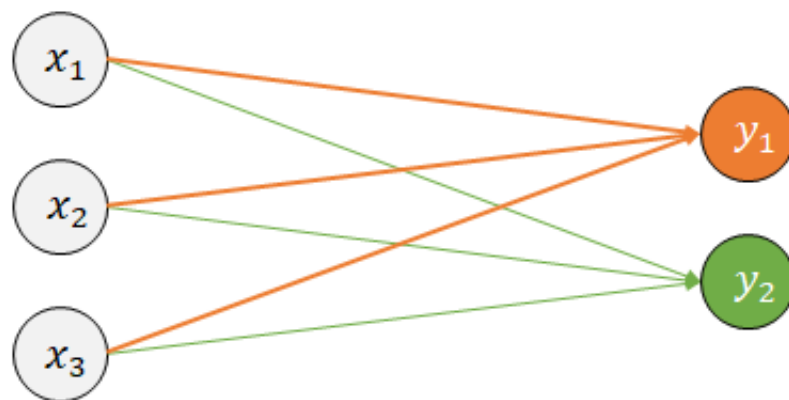
$$W = [w_1 \ w_2 \ \dots \ w_m] = \begin{bmatrix} w_{11} & w_{21} & \dots & w_{m1} \\ w_{12} & w_{22} & \dots & w_{m2} \\ \dots & \dots & \dots & \dots \\ w_{1n} & w_{2n} & \dots & w_{mn} \end{bmatrix},$$
$$\mathbf{b}^T = [b_1 \ b_2 \ \dots \ b_m]$$

파라미터: W, b

• 순방향 신경망

> 벡터 함수인 계층

- 다음과 같은 방식으로 행렬 연산을 한다.

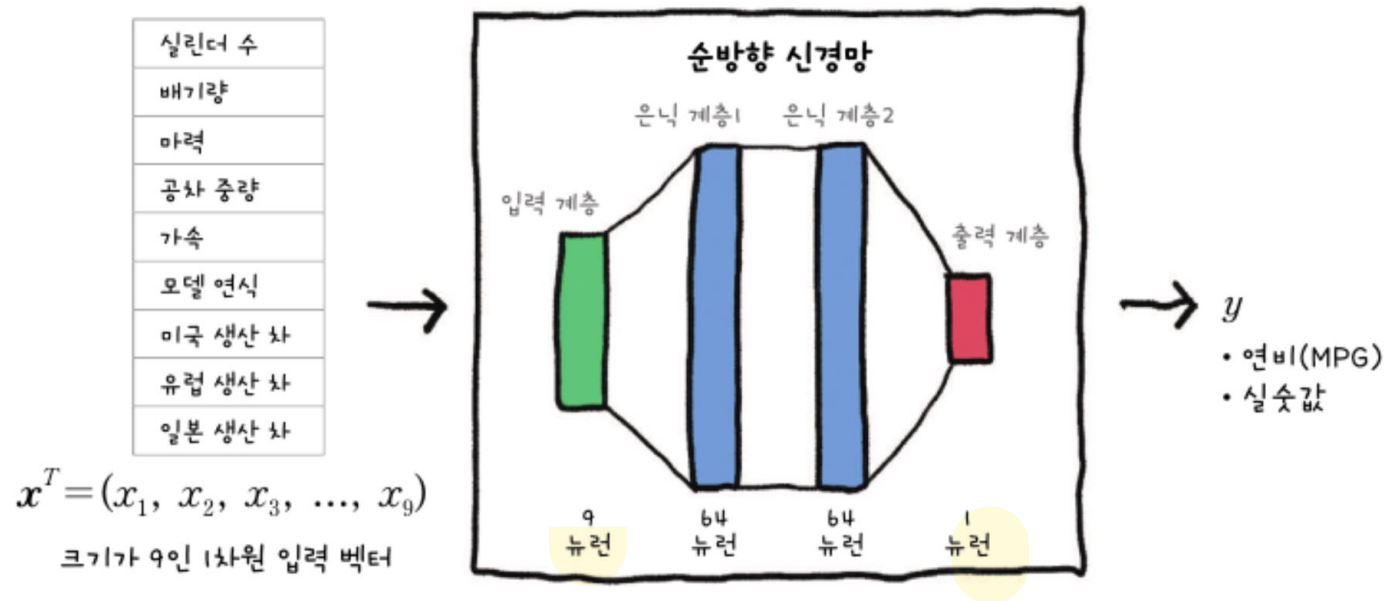


$$\begin{array}{l}
 \text{Sample \#1} \\
 \text{Sample \#2} \\
 \text{Sample \#3} \\
 \text{Sample \#4}
 \end{array}
 \begin{array}{c}
 X \\
 \begin{bmatrix} x_1 & x_2 & x_3 \\ x_1 & x_2 & x_3 \\ x_1 & x_2 & x_3 \\ x_1 & x_2 & x_3 \end{bmatrix}
 \end{array}
 \times
 \begin{array}{c}
 W \\
 \begin{bmatrix} w_1 & w_4 \\ w_2 & w_5 \\ w_3 & w_6 \end{bmatrix}
 \end{array}
 +
 \begin{array}{c}
 B \\
 \begin{bmatrix} b_1 & b_2 \end{bmatrix}
 \end{array}
 =
 \begin{array}{c}
 Y \\
 \begin{bmatrix} y_1 & y_2 \\ y_1 & y_2 \\ y_1 & y_2 \\ y_1 & y_2 \end{bmatrix}
 \end{array}$$

• 순방향 신경망

> 입력 계층

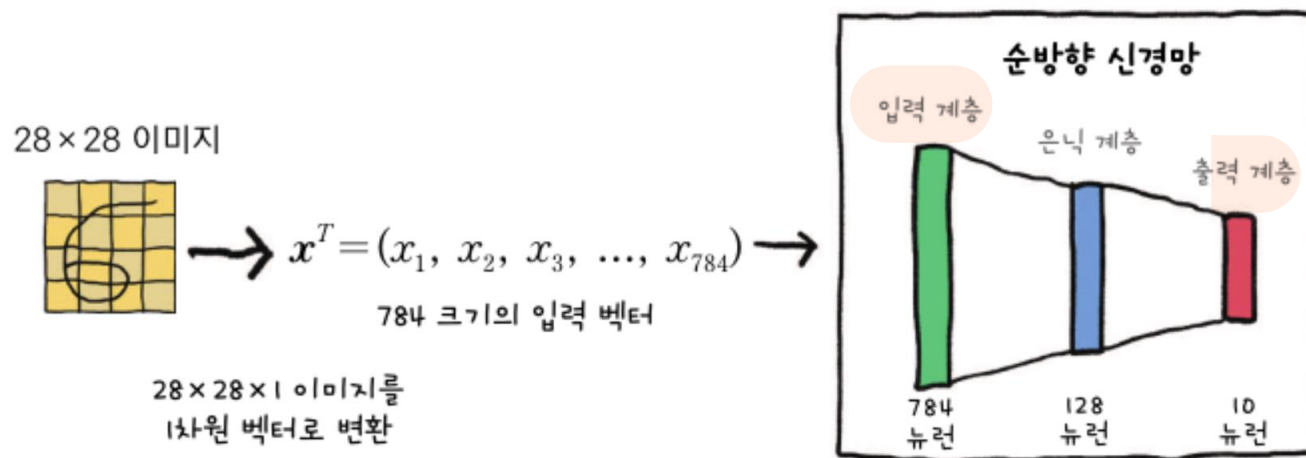
- 입력 데이터를 벡터 형태로 받아서 다음 계층에 전달한다.
- 입력 크기가 N개라면 N개의 뉴런으로 정의된다.



• 순방향 신경망

> 입력 계층

- 다음과 같이 1차원 벡터가 아닌 다차원 배열이라면 이를 1차원 벡터로 변환하여 입력 받는 계층을 구성한다.



입력뉴런갯수 미리 정해줘야

• 순방향 신경망

> 활성화 함수

- 은닉 계층을 설계할 때 선택하는 **활성화 함수**는 다양하다.

> 대표적인 활성화 함수

은닉층, 출력층에서 사용됨 / 시그모이드, 소프트맥스 함수는 출력층에 사용하는 것으로 정채짐

- 시그모이드 0-1 사이의 값으로 변경: 로지스틱회귀 : 아웃풋에서 이진분류에 사용됨
- 하이퍼볼릭 탄젠트 -1에서 1 사이

은닉 계층

- **ReLU** 음수일땐 0 양수일때 양수값대로 출력
- **Leaky ReLU** 0보다 작을때 가중치를 주어 0이되지는 않지만 작은수가 되도록
- **맥스아웃** 구간별로 별도의 함수 사용하도록 지정할수 있는 함수

스위스: 구글에서 만듦-렐루함수 단점 보완

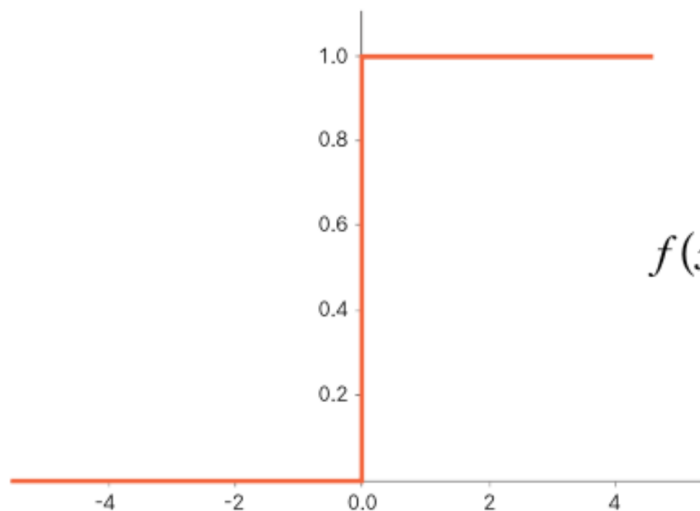
softmax 함수: 아웃풋에서 사용됨 -다중분류

• 순방향 신경망

030

> 계단 함수

- 매컬러-피츠 모델과 퍼셉트론에서 사용된 활성화 함수
- 0보다 크거나 같으면 1, 0보다 작으면 0을 출력한다.
- 계단 함수는 모든 구간에서 미분값이 0이기 때문에 역전파 알고리즘이 적용되지 않아서 현재는 사용하지 않는다.



$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{그 외의 경우} \end{cases}$$

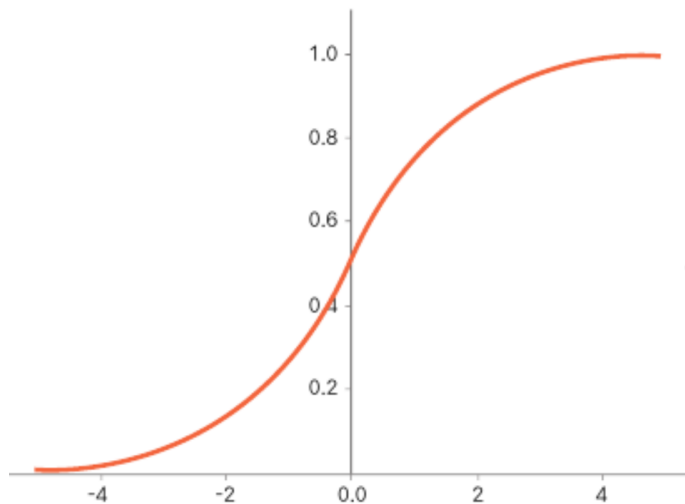
전체 구간에서 기울기가 0이므로 학습불가=미분불가

• 순방향 신경망

> 시그모이드 함수

- 계단 함수와 비슷하면서 미분이 가능한 함수이다.
- S 같은 형태의 함수로 0~1 사이의 값을 출력한다.
- 증가하는 함수로 모든 미분 값이 양수이다.

: 은닉층이 아니라 출력층에서 분류를 위해 사용

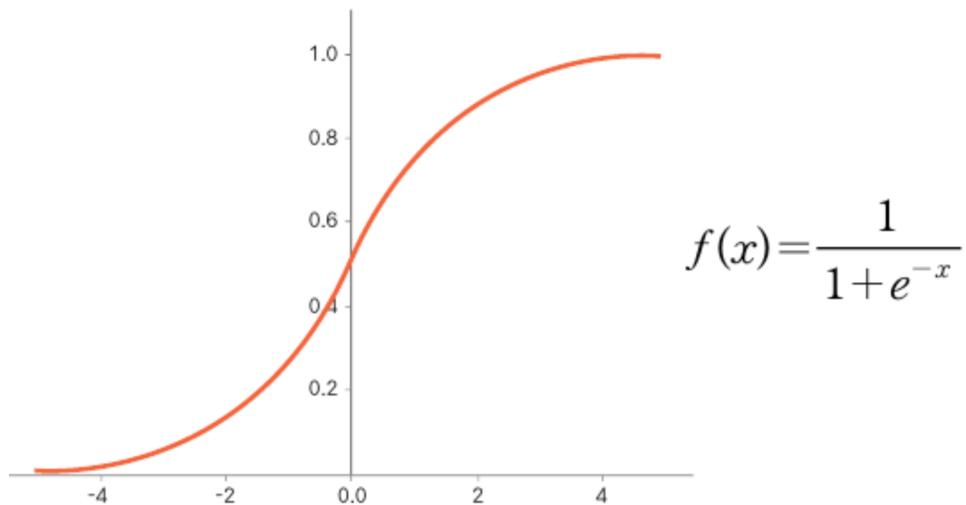


$$f(x) = \frac{1}{1 + e^{-x}}$$

• 순방향 신경망

> 시그모이드 함수의 한계

- 함수의 정의에 지수 함수가 포함되어 연산에 시간이 걸린다.
- 함수의 양끝의 기울기가 0이 되어 학습이 중단될 수 있다.
- 출력 값이 항상 양수로 학습 시, 좌우로 진동하면서 비효율적이다.

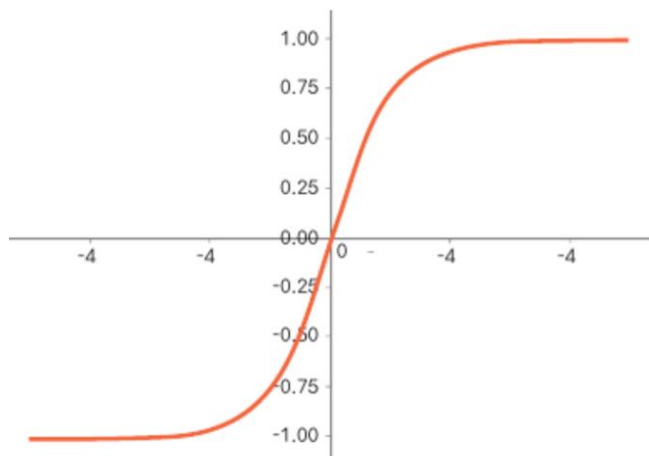


• 순방향 신경망

033

> 하이퍼볼릭 탄젠트 함수

- 시그모이드 함수와 유사하고 -1 ~ 1 사이의 값을 출력한다.
- 음수도 같이 출력하기에 시그모이드 함수가 비효율적으로 학습하는 것을 해결하고 사용했다.
- 그 외에는 시그모이드 함수와 같은 한계점을 가지고 있다.



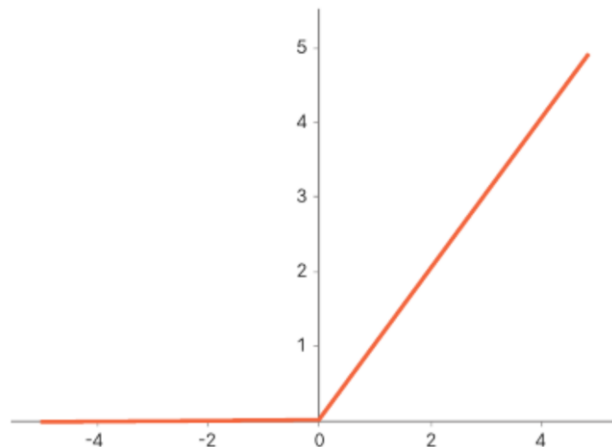
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

• 순방향 신경망

034

> **ReLU 함수** 가장 많이 사용한다.

- 0보다 큰 값이 들어오면 그대로 통과시키고 0보다 작은 값이 들어오면 0을 출력하는 함수이다.
- 가장 기본적으로 쓰는 활성화 함수이다.
- 연산이 필요 없기에 속도가 매우 빠르다.



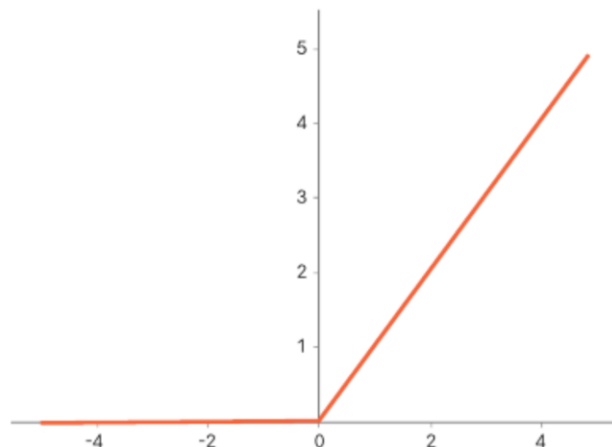
$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{그 외의 경우} \end{cases}$$

• 순방향 신경망

035

> ReLU 함수의 한계

- 죽은 ReLU는 뉴런이 계속해서 0을 출력하는 상태이다.
- 계속해서 0을 출력하면 기울기가 0이 되어 학습되지 않는다.
- 뉴런의 10 ~ 20%가 죽은 ReLU가 되면 학습에 문제가 될 수 있다.



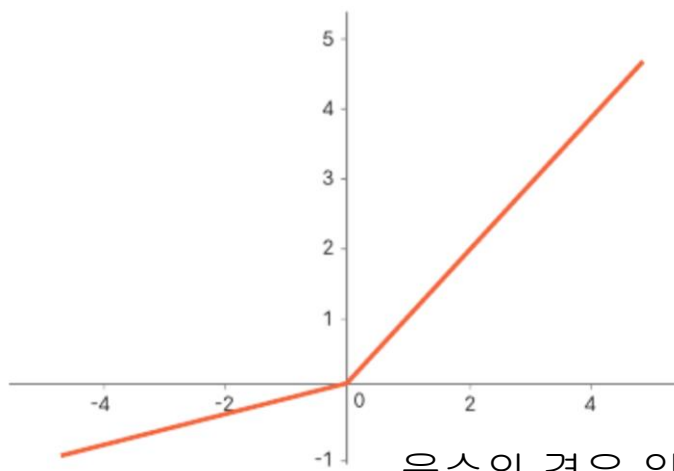
기울기가 0과 곱해지기에 값이 0이 됨.
원값을 알 수 없어 역전파 경우 오차 계산 불가
학습 안됨

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{그 외의 경우} \end{cases}$$

• 순방향 신경망

> Leaky ReLU

- 죽은 ReLU 문제를 해결하기 위해 사용한다.
- 음수 구간에 작은 기울기를 주어 학습을 진행할 수 있게 한다.
- 단! 기울기가 고정되어 있기에 최적의 성능을 내지 못할 수 있다.



$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0.01x & \text{그 외의 경우} \end{cases}$$

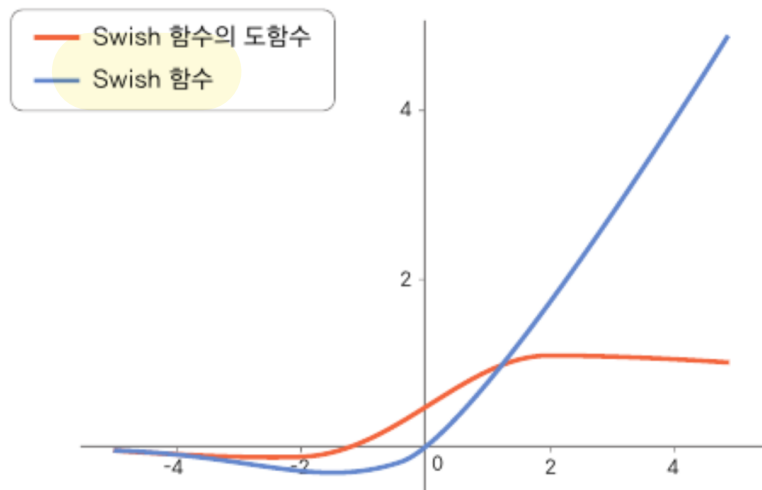
음수의 경우 일정 비율의 작은 수를 주어 0이 되지 않도록 함
끝으로 가면갈수록 음수값이 매우 커지면 학습에 방해가 됨

• 순방향 신경망

> Swish 함수

- ReLU의 문제를 해결하기 위해 구글에서 개발한 함수이다.
- 시그모이드 함수와 선형 함수의 곱으로 나타낸다.
- ReLU와 비슷하지만 음수 구간에서 바로 0 값이 아닌 음수 값을 출력하다가 0으로 수렴한다.

기울기가 고정되어있지 않음

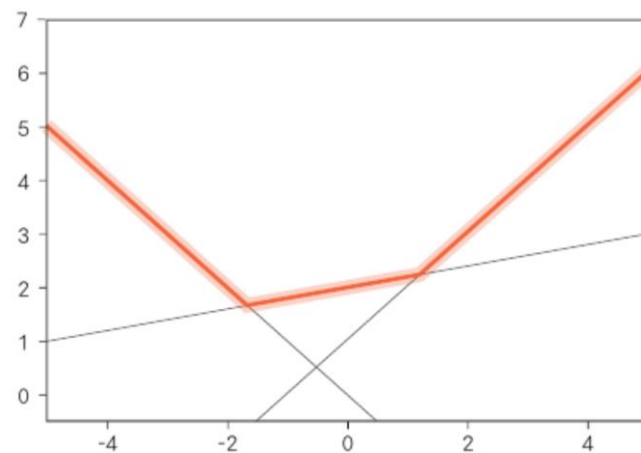


$$f(x) = x \cdot \sigma(x)$$

• 순방향 신경망

> 맥스아웃 함수

- 맥스아웃 함수는 구간 선형 함수로 뉴런에 최적화된 활성 함수를 학습을 통해 찾아낸다.
- 다음과 같이 뉴런별로 선형 함수를 학습한 뒤에 구간별로 활성 함수를 결정한다.



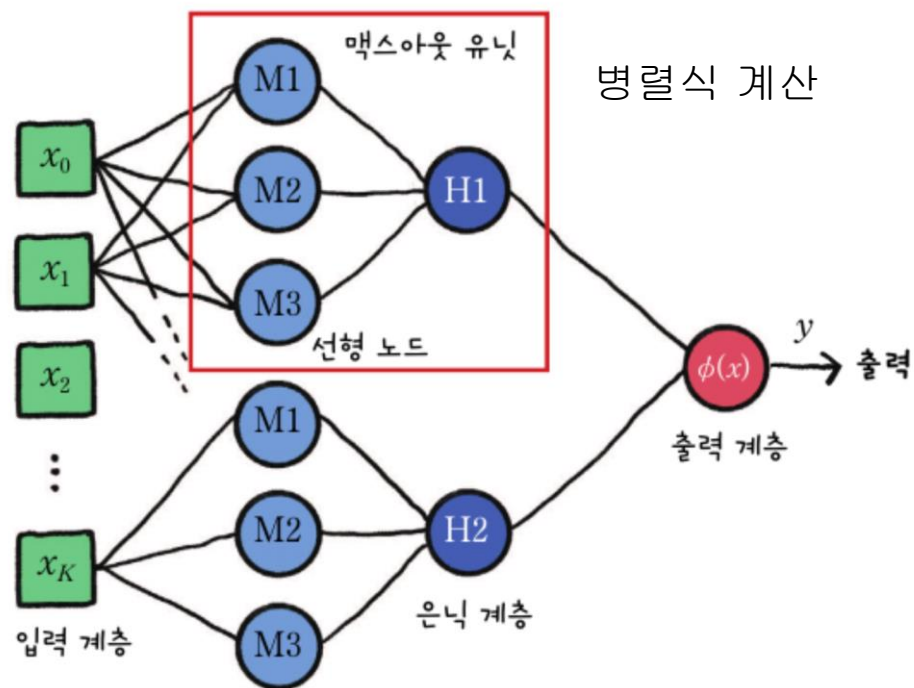
$$f(x) = \max (w_i^T x + b_i), i=1, 2, \dots, k$$

k : 선형구간의 개수

• 순방향 신경망

> 맥스아웃 함수

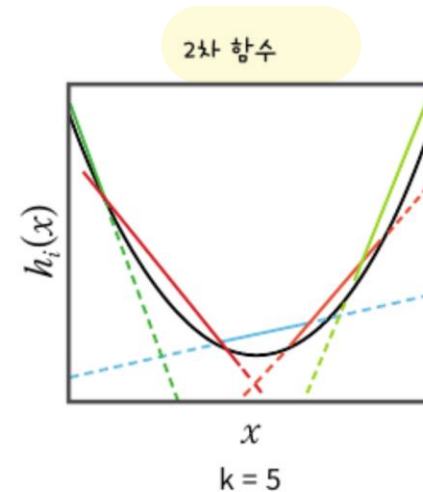
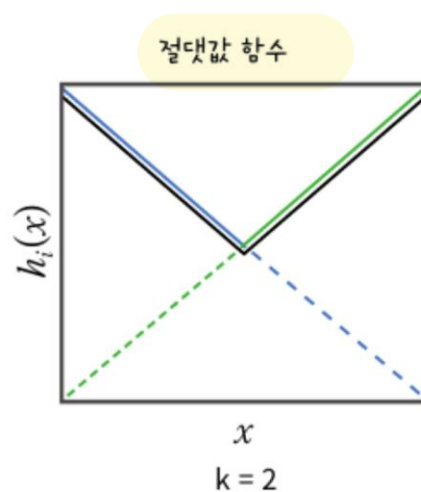
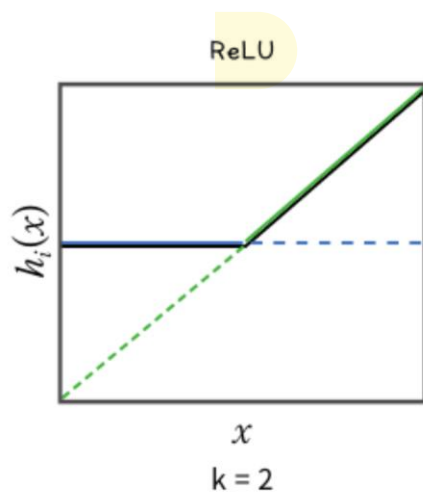
- 맥스 유닛은 구간별로 맥스아웃 유닛을 구성하여 신경망을 구성한다.



• 순방향 신경망

> 맥스아웃 함수

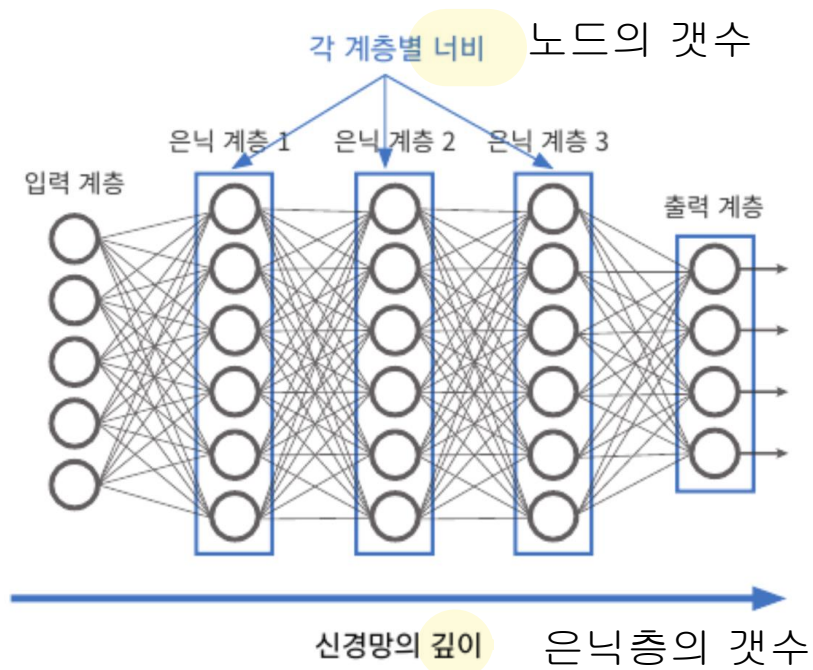
- 맥스 아웃은 선형 노드의 개수에 따라 다른 형태의 함수로 근사할 수 있다.



• 순방향 신경망

> 신경망 모델의 크기

- 신경망의 크기는 너비(Width)와 깊이(Depth)로 정해진다.
- 너비는 계층별 뉴런의 수를 뜻한다.
- 깊이는 신경망의 계층 수를 뜻한다.



이진분류, 회귀 경우는 1개값
그외 정하는대로

• 순방향 신경망

042

> **신경망 모델의 크기** 노드와 은닉계층의 갯수는 많을 수록 좋으나...

- 신경망의 크기는 데이터 간의 관계가 복잡할수록 뉴런의 수를 늘려야하고, 특징의 추상화 수준이 높을수록 계층의 수를 늘려야한다.
- 하지만 문제를 풀기 전에는 데이터에 잠재적 특징의 수나 관계의 복잡도, 추상화 수준을 가늠하기 어려워서 신경망의 크기를 결정하기 쉽지 않다.
- 적절한 모델의 크기를 찾기 위해서는 경험적으로 크기의 범위를 정하고 성능 분석을 통해 최적의 크기를 탐색해야 한다.

보통>
노드의 갯수는 노드의 2배수
은닉층은 2-3개
정해진 규칙이 없음

• 순방향 신경망

> 신경망 모델의 크기 탐색

- 모델의 크기를 탐색할 때는 그리드 서치나 랜덤 서치와 같은 탐색 방법을 이용한다.
- 그리드 서치는 등간격으로 파라미터를 설정하고, 랜덤 서치는 여러 파라미터를 랜덤하게 조합해서 설정한다.

