

# Deep Learning 개요

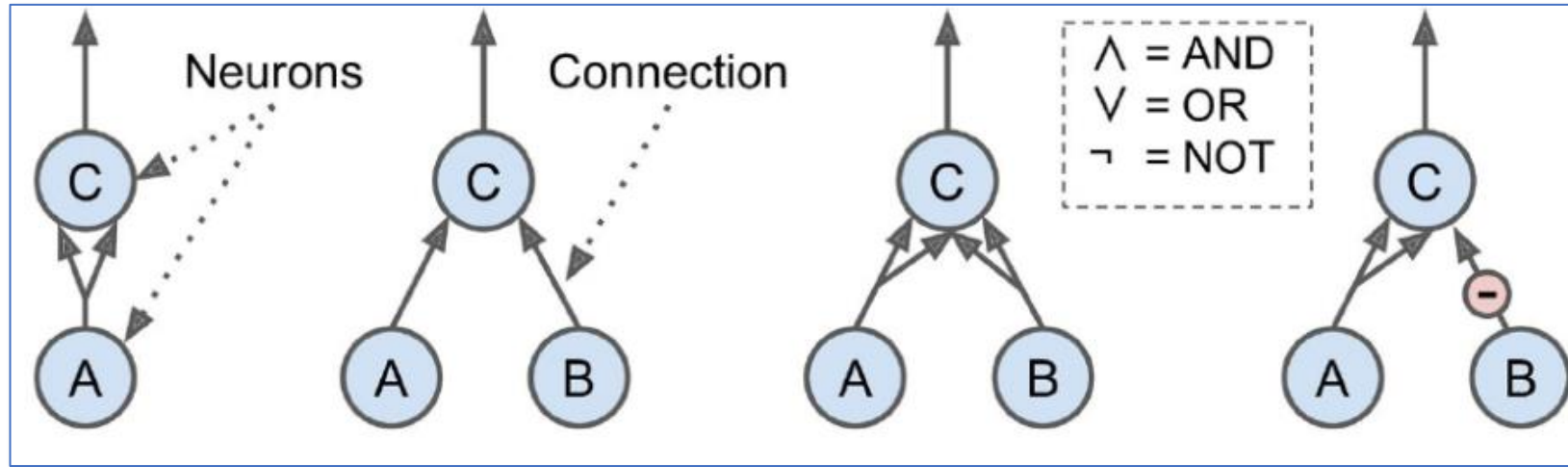


- 딥러닝은 신경망 모델이다!!!
  - 그 이유는?
  - 예전에 완성된 신경망 모델이 그대로 사용된다.
    - 부족했던 PC 성능이 급격히 향상되었다.
    - 부족했던 데이터가 이제는 넘쳐나는 시대로 바뀌었다.
    - 예전의 신경망을 구현, 검증하는 것이 가능해 졌다.
    - 구현해보니 안되는 줄 알았던 모델이 아주 잘 돌더라!!
    - 하는 김에 크기를 엄청나게 키워보자 → 역시 잘 되더라 → 딥러닝!!!
  - 좀 과격한 표현이지만 대체로 맞는 말임



- 신경세포의 간단하고 효과적인 처리 방식에 착안해 구현된 머신 러닝 알고리즘(모델)의 한 종류
- 신경세포의 형태와 동작을 극도로 단순화 시킨 뉴런 모델을 다수 연결하여 망, 즉 네트워크를 만든 모델  
( = 다량의 뉴런(Neuron)들이 층(Layer)으로 연결되어 간단한 계산과 연결 방식을 통해 복잡한 문제를 해결하는 모델 )
- 뉴런의 동작 방식은 컴퓨터 프로그램의 방식에 비해 다양한 장점을 지님

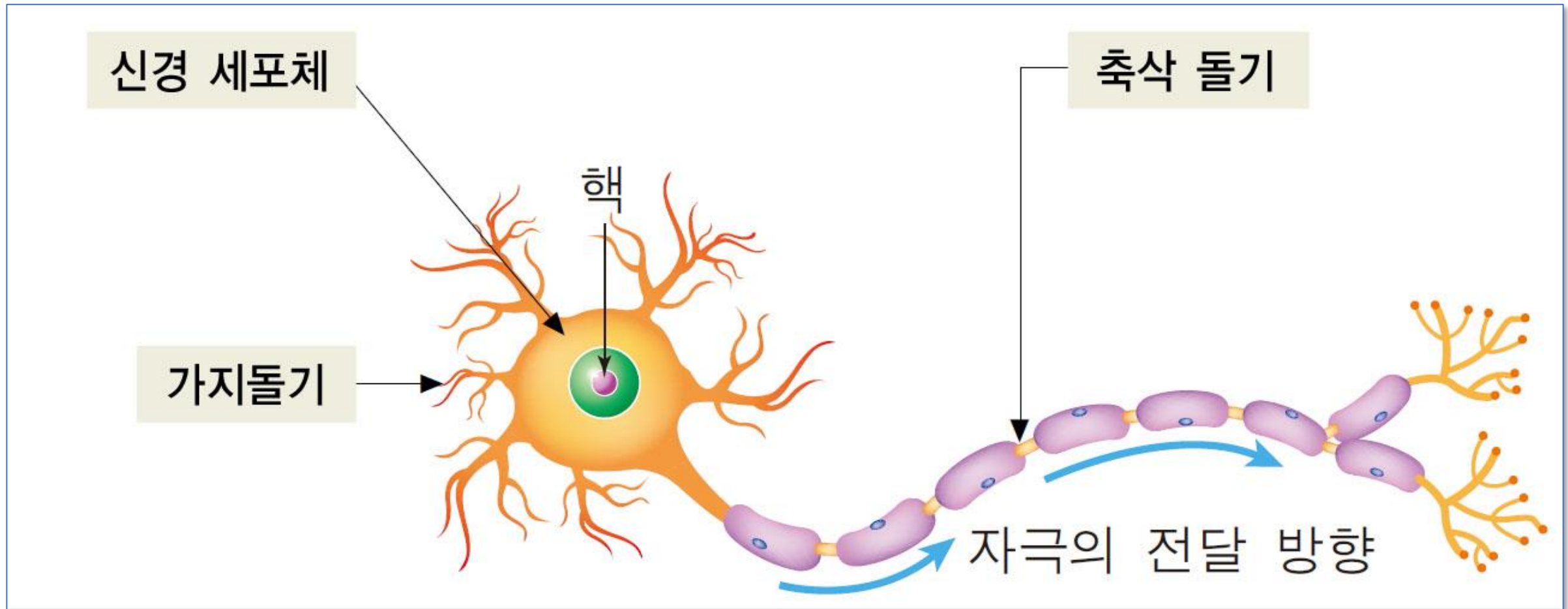
- 1943년 워런 맥컬록, 월터 피트의 최초의 신경망 모델이 시초



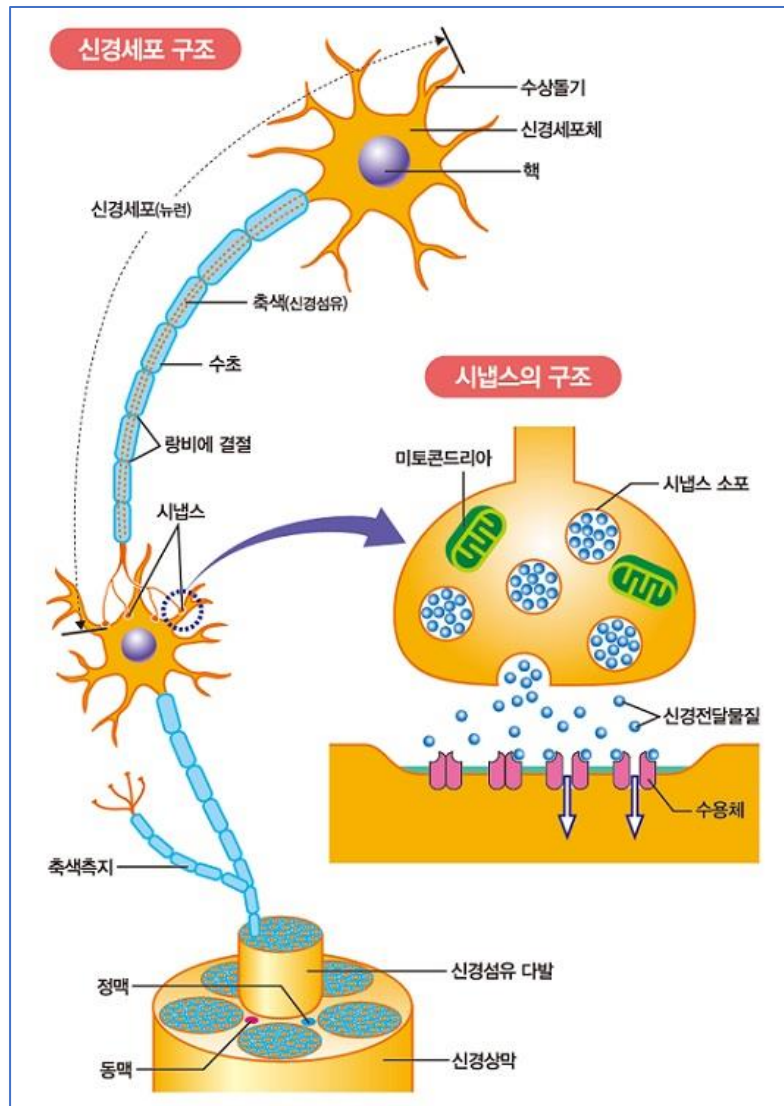
입력되는 데이터들의  
연결형태에 따라서  
결과가 결정된다

- 헵의 규칙이 신경망 모델의 동작을 정의하는 기반이 됨
  - 시냅스의 앞과 뒤에서 동시에 신경세포가 흥분할 때, 해당 시냅스의 효율이 강화된다.

# 신경세포의 구조



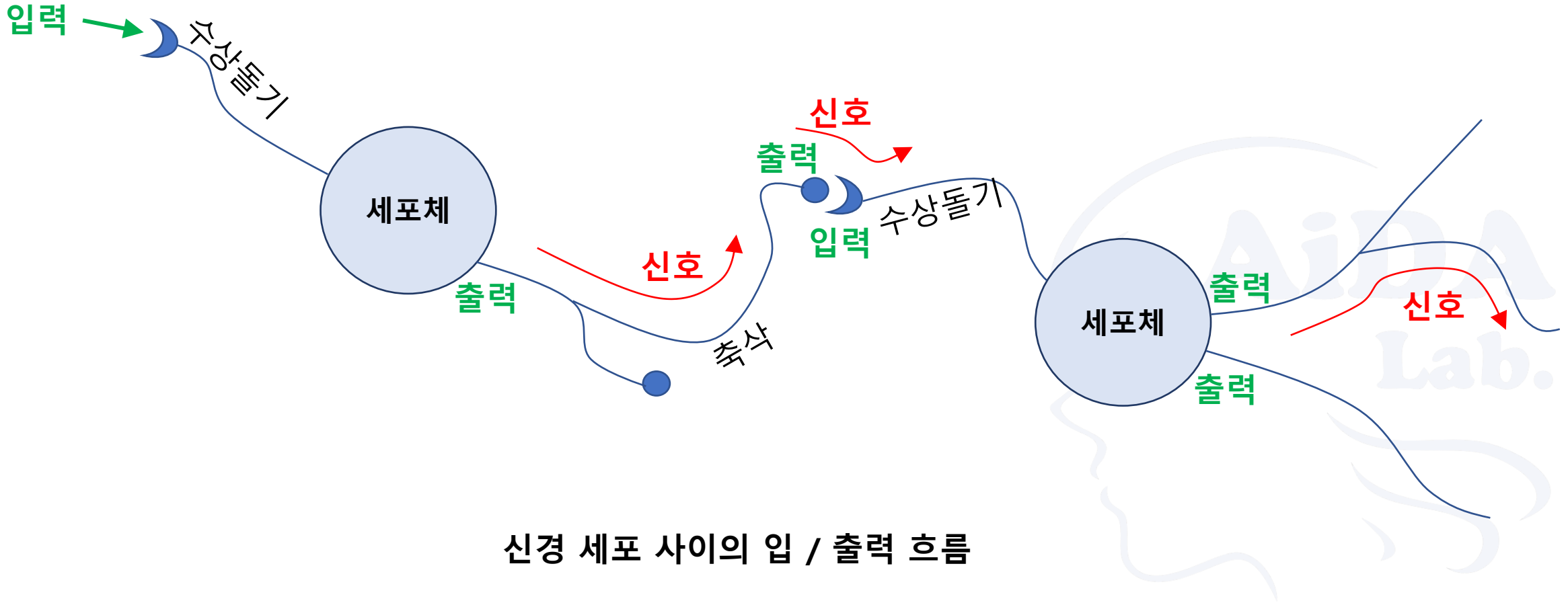
# 신경세포의 연결 형태



- 신호의 전달은 전기로 이루어짐
- 신경세포의 말단에는 시냅스가 존재
- 시냅스 사이의 신호 전달은 신경전달물질이라는 화학물질을 통해 전달됨
- 신경전달물질
  - 세로토닌
  - 도파민
  - 엔돌핀
  - 아드레날린 등..

- 신경체계를 구성하는 수많은 신경세포들
- 다양한 감각기관을 통하여 (전기)신호를 발생, 전달
- 각 신경세포는 수많은 시냅스를 통해 신호를 전달 받음
- 전달 받은 신호는 대체로 무시하지만.. 동시에 전달된 신호의 합이 임계값을 넘으면 활성화(발산, 흥분한다 라고 표현함)
- 활성화 된 신경세포는 활성화 패턴에 따라 신경전달물질 분비
- 이웃 신경세포는 신경전달물질을 수용하면서 이온화 작용, 화학작용을 통하여 전기 신호 발생
- 처리 단계 반복

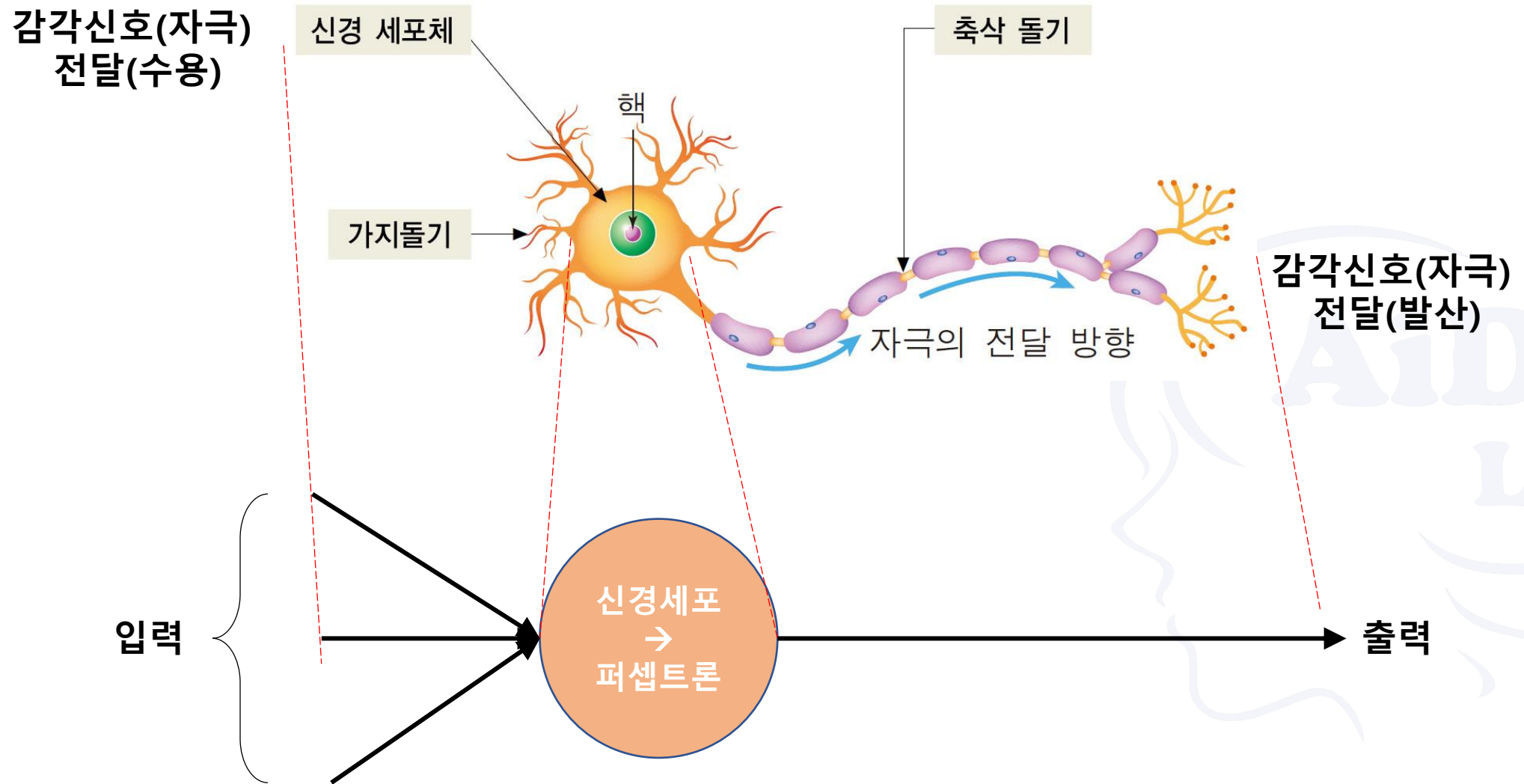
# 신경세포에서의 데이터 흐름



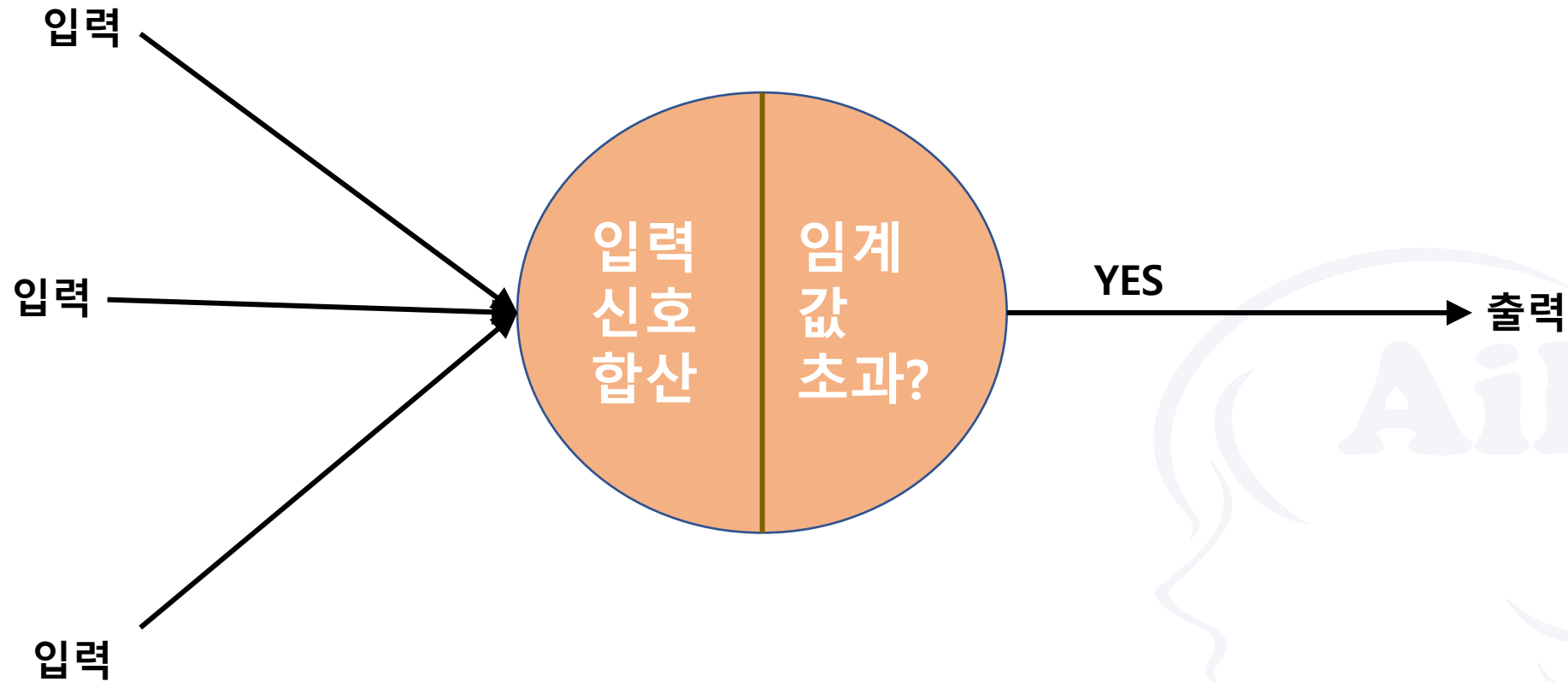
신경 세포 사이의 입 / 출력 흐름



# 신경세포, 신경망을 어떻게 단순화하였나?



# 신경세포를 어떻게 단순화하였나?

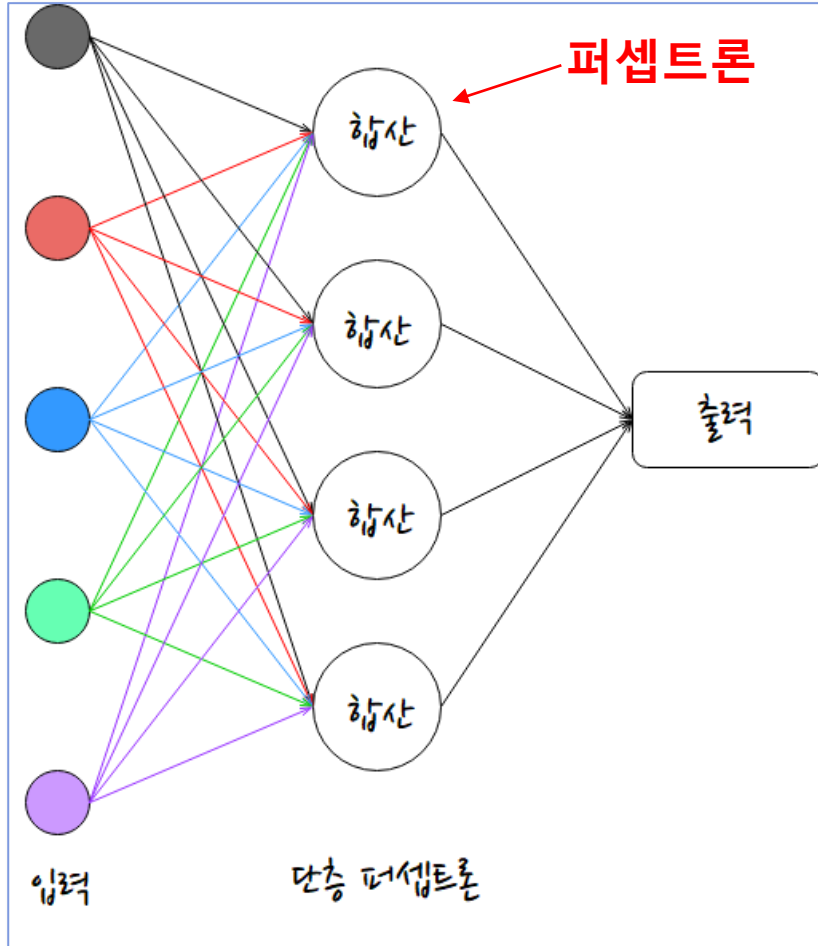


퍼셉트론 모델의 형태

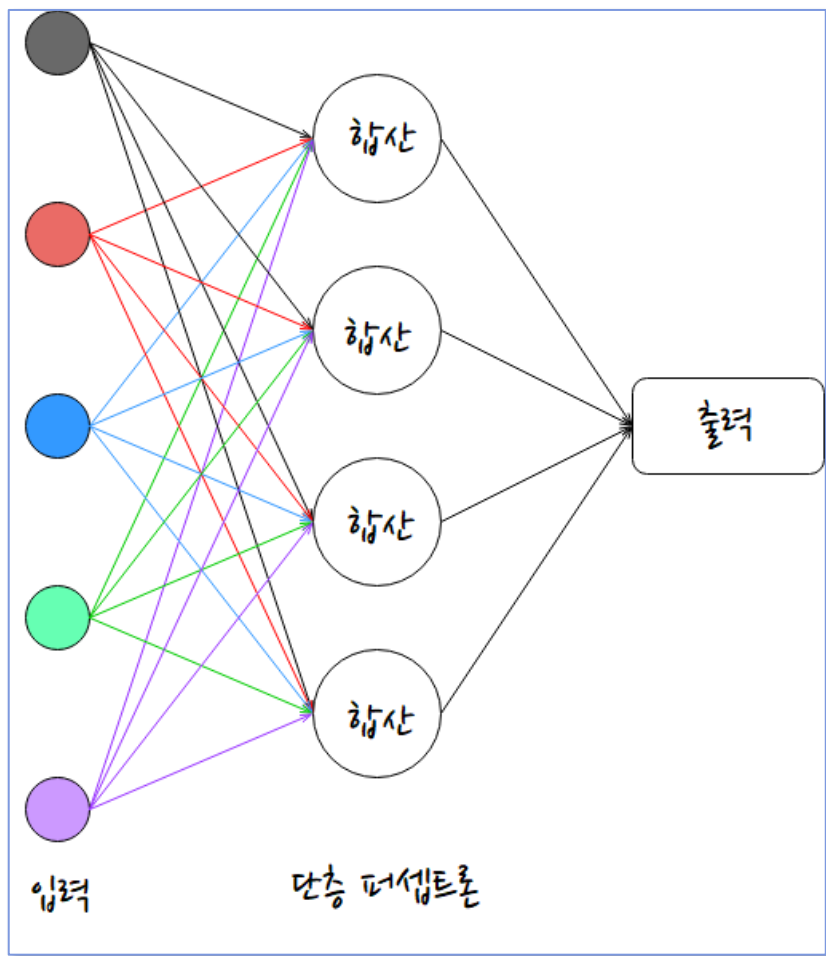
# 퍼셉트론 모델의 문제점? 한계?

- 퍼셉트론 모델은 실제 신경세포의 형태를 극도로 단순화한 모델
- 전기신호의 전달, 합산, 발산을 모델링 함
- 그러나... 신경세포는 전기신호만 전달하는 것이 아님
- 신경세포의 활성화는 전기신호의 합산으로 이루어지더라도 해당 신호를 전달할 때는 다양한 신경전달물질이 활동함
- 동일하게 전기신호는 전달되더라도 신경전달물질의 종류에 따라 사람의 반응은 달라짐  
→ 이런 부분은 전혀 반영되지 않은 모델
- 그러나 기능의 모방을 목표로 하는 약 인공지능의 시점에서는 충분히 활용가치가 높은 모델

## • 단층 퍼셉트론(SLP, Single Layer Perceptron)

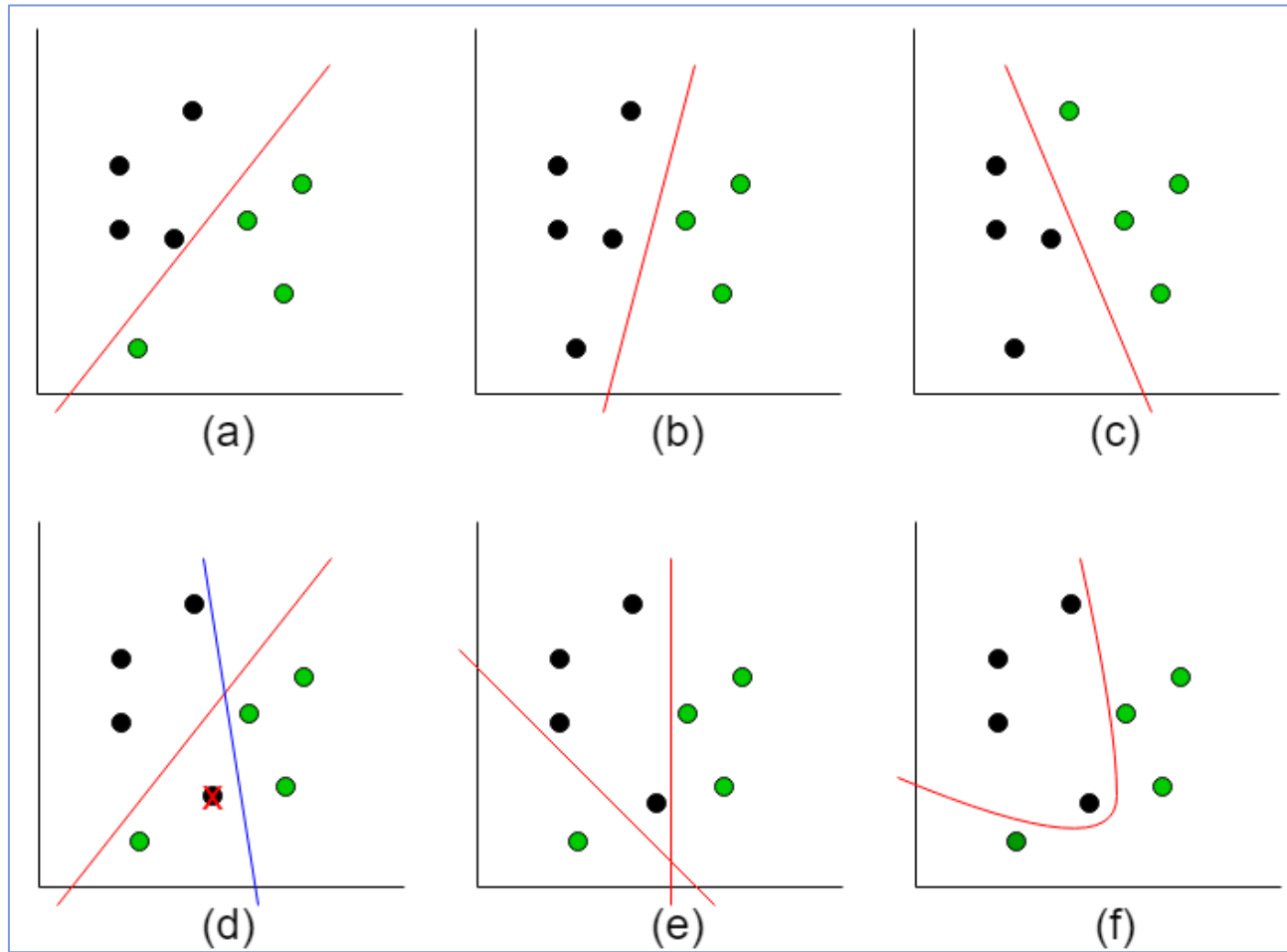


- 다수의 퍼셉트론이 하나의 층을 이루고 있는 형태
- 센서 데이터 등 다양한 데이터를 각 퍼셉트론의 입력으로 전달
- 각 퍼셉트론은 입력된 데이터를 모아서 합산
- 합산 결과가 임계 값을 넘으면 1, 넘지 않으면 0 출력
- 입력층에서 각 퍼셉트론으로 진행되는 통로에는 가중치 적용 (가중치는 모든 통로가 각각 다르게 적용될 수 있음)
- 왼쪽 그림에서 4개의 퍼셉트론이 각각 1, 0, 0, 1 이라는 결과를 낸다면 최종 출력은 1001 이라는 2진수 값이 나오는 형태



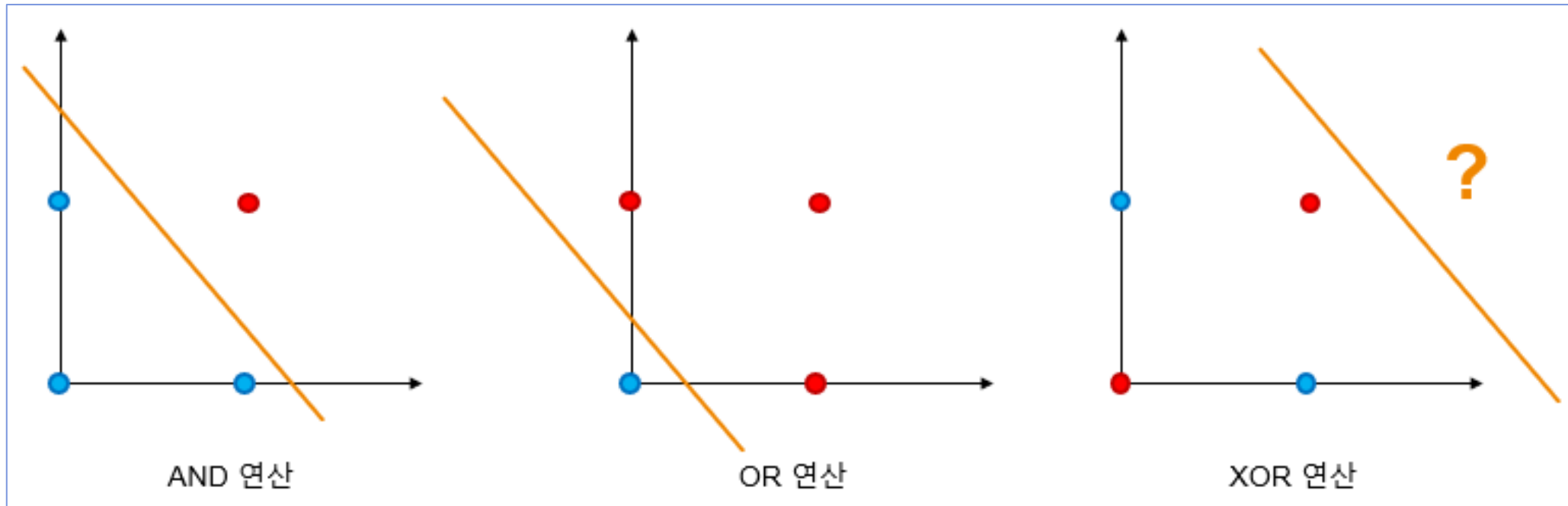
- 단층 퍼셉트론의 구조를 보면...
  - 가중치를 변경할 수 있는 방법이 없다  
→ 학습이라는 개념이 없다
  - 한 번 생성된 후에는 아무런 변형이 없는 단순한 분류 알고리즘에 불과함
  - 한 층의 변경가능한 퍼셉트론만 존재  
→ 1개의 선을 그어 분리 가능한 패턴만 분류 가능

- 직선으로 데이터 분류하기

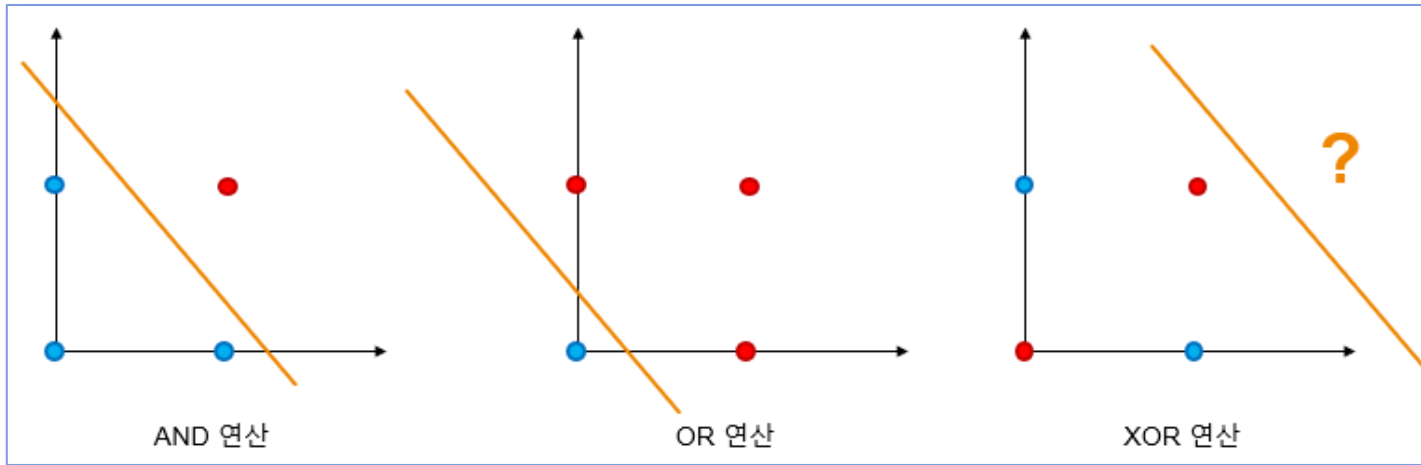


XOR 불가능 문제도  
여기에서 확인됨

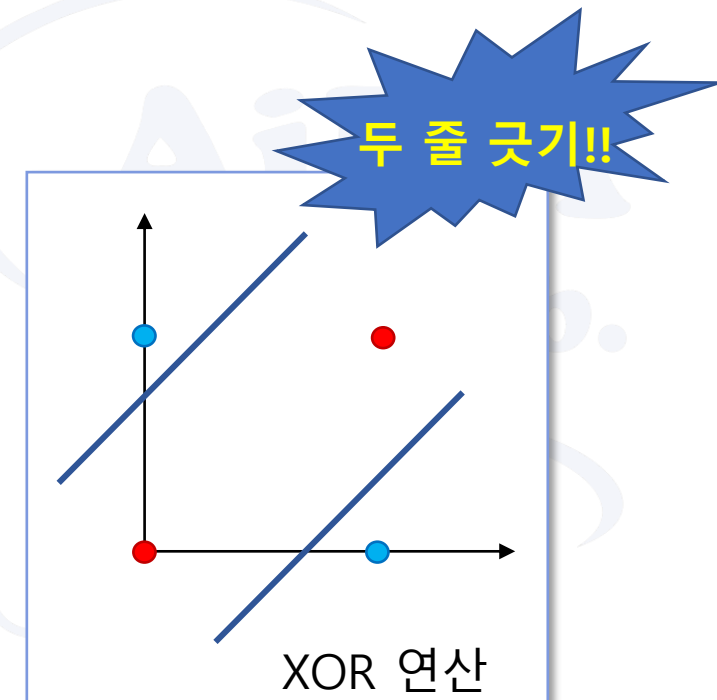
- XOR 문제



- XOR 문제

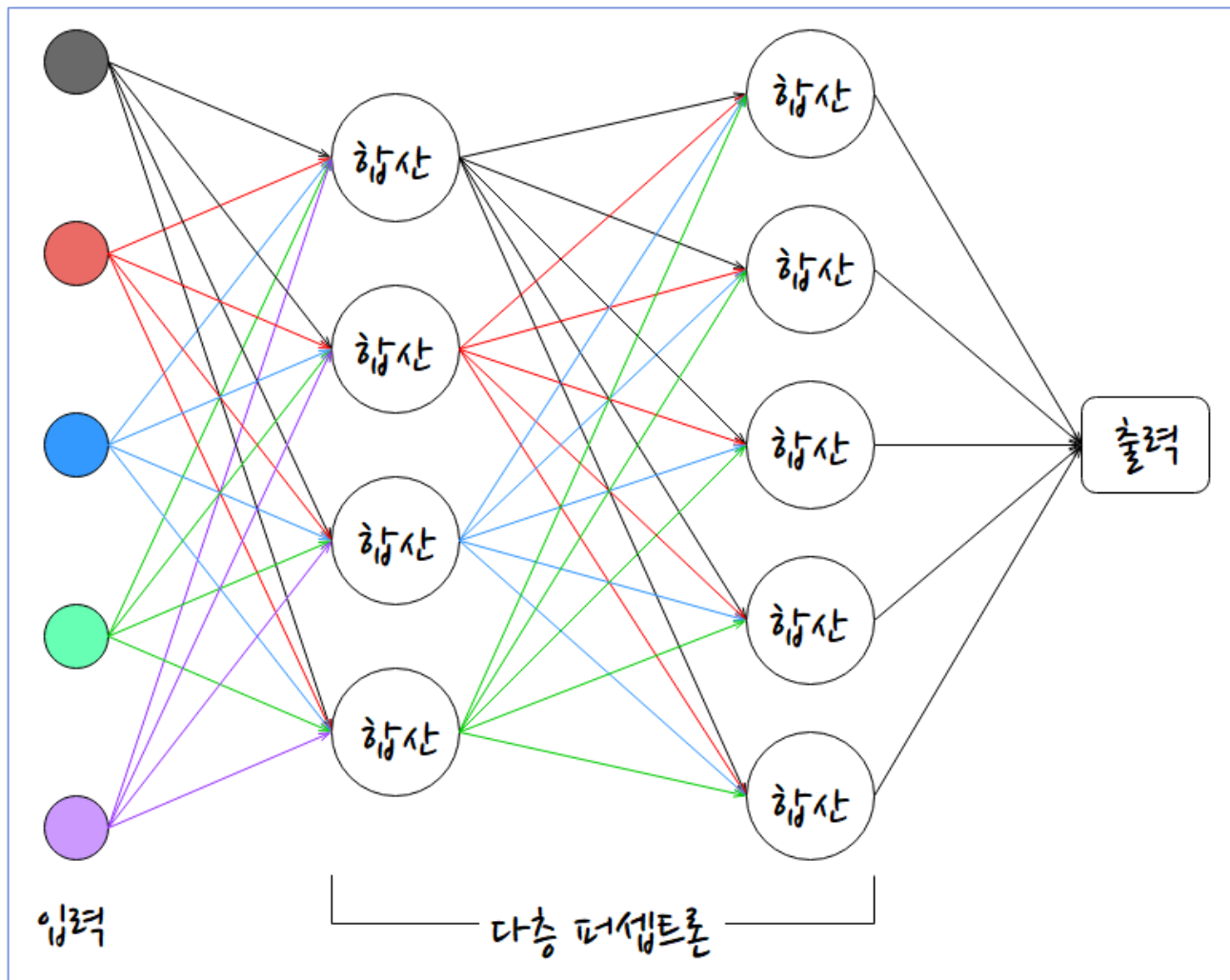


그런데 이렇게 하면?

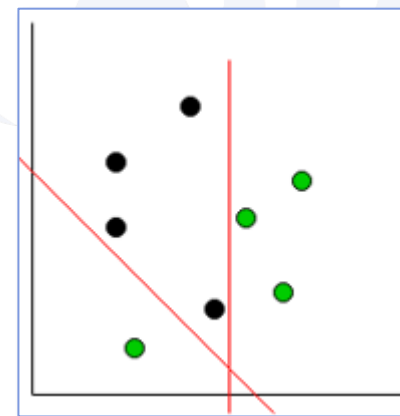




# 다층 퍼셉트론(Multi Layer Perceptron)



단층 퍼셉트론의  
가장 큰 문제 해결



# 다층 퍼셉트론(Multi Layer Perceptron)

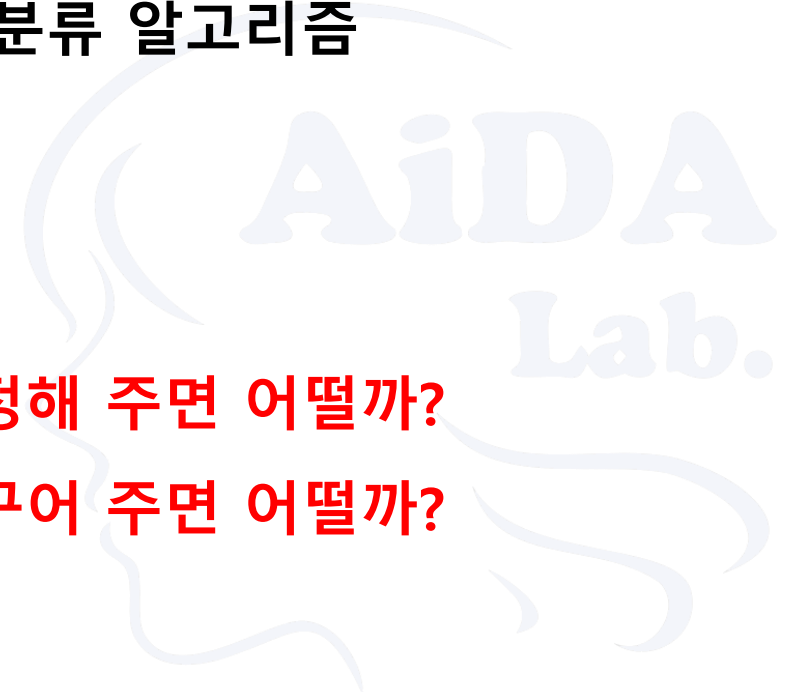


- 또 문제점

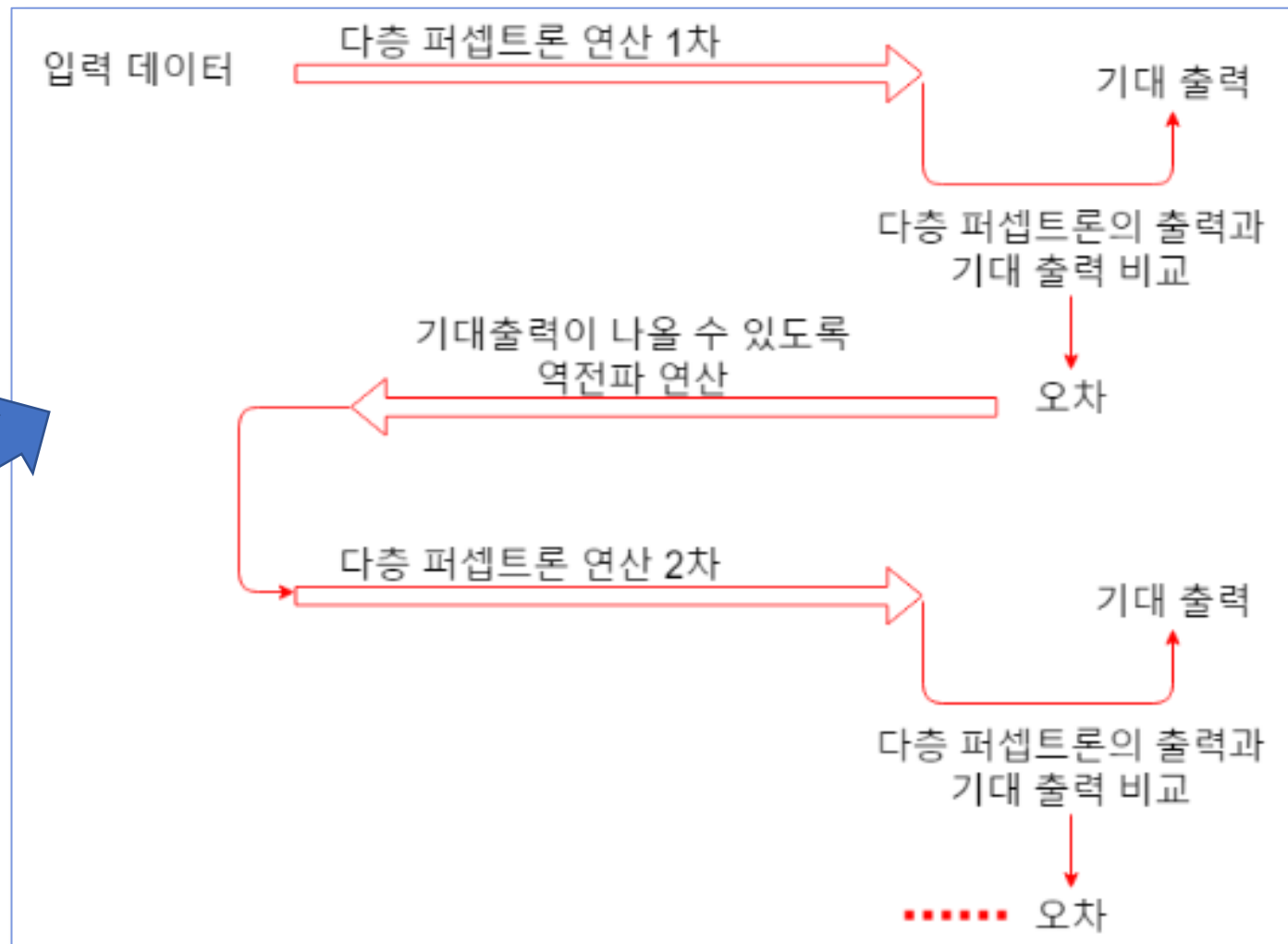
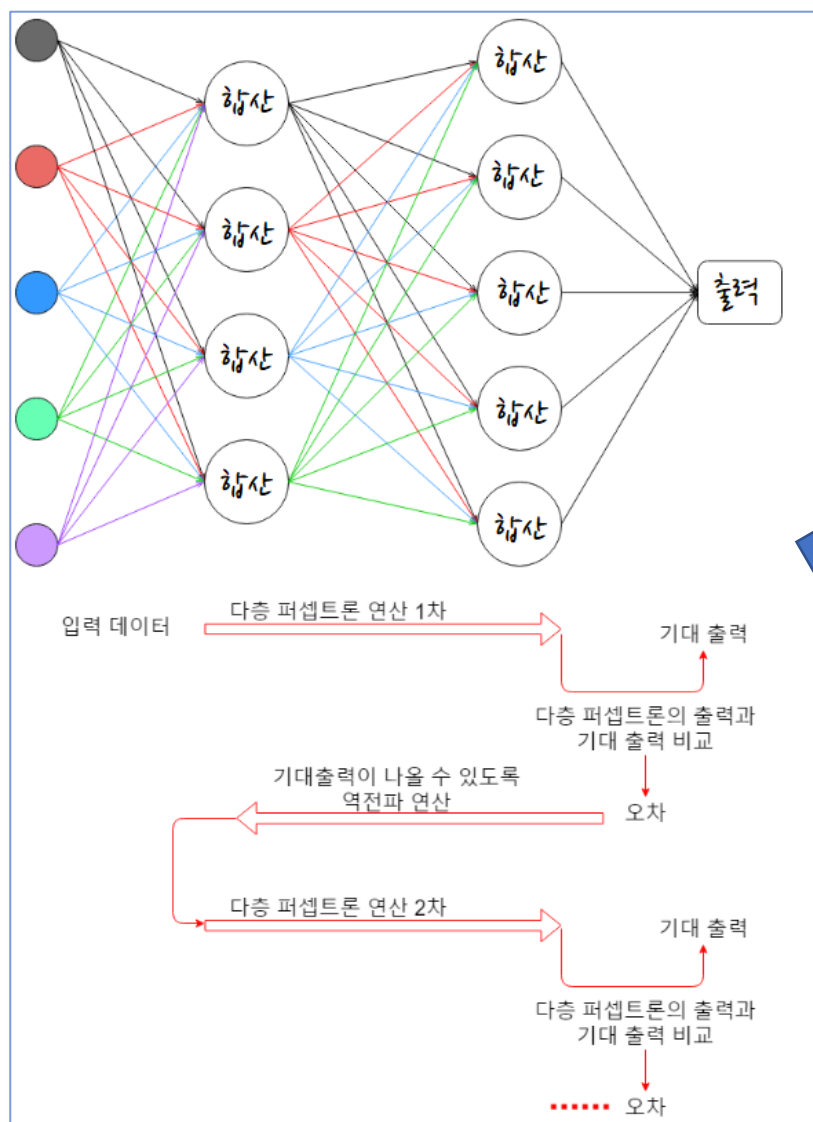
- 단층 퍼셉트론과 마찬가지로 각 통로의 가중치를 변경할 방법이 없다
- 한 번 생성되면 변경 불가능한 분류 알고리즘
- 역시 학습의 개념이 없다     인공지능이 아닌 단순한 분류 알고리즘

- 해결책은?

- 수행할 때마다 예전 데이터를 들고 와서 가중치를 수정해 주면 어떨까?
- 그럼 아예 앞뒤로 왔다 갔다 반복하면서 가중치를 바꾸어 주면 어떨까?
- Back Propagation (역전파) 알고리즘 제안



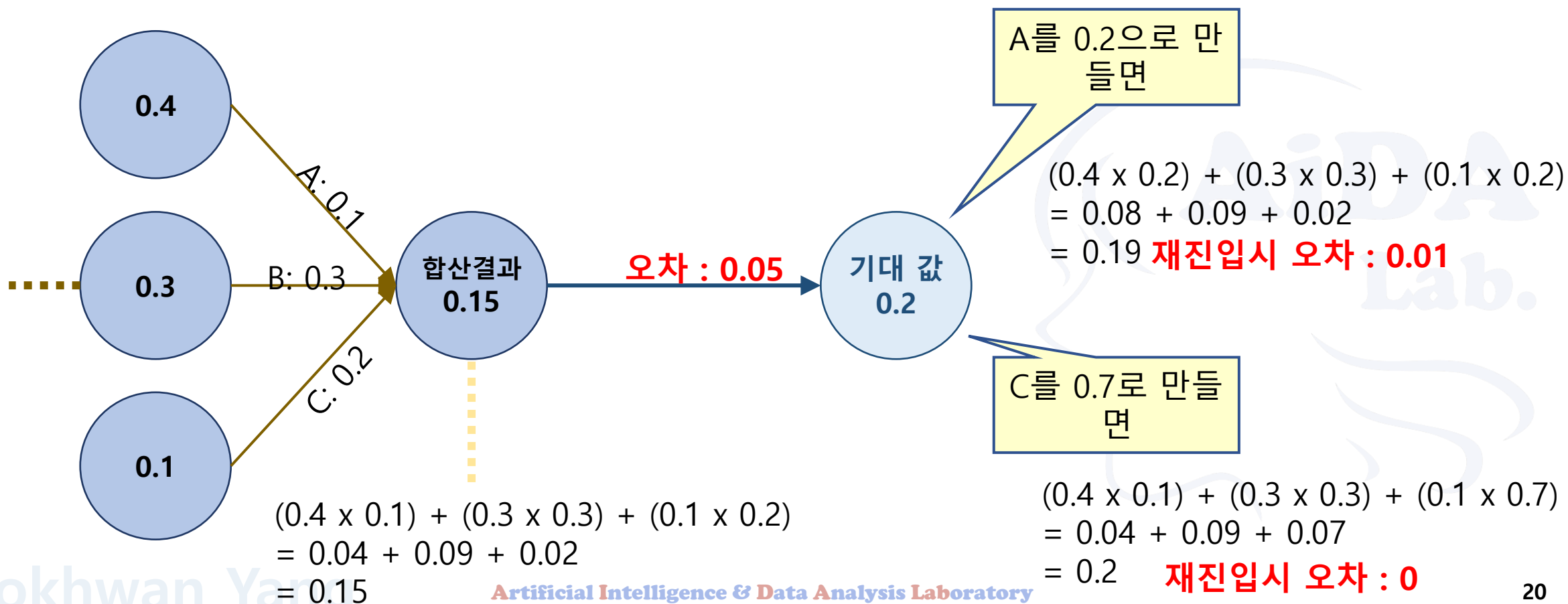
# Back Propagation (역전파) 알고리즘



# Back Propagation (역전파) 알고리즘

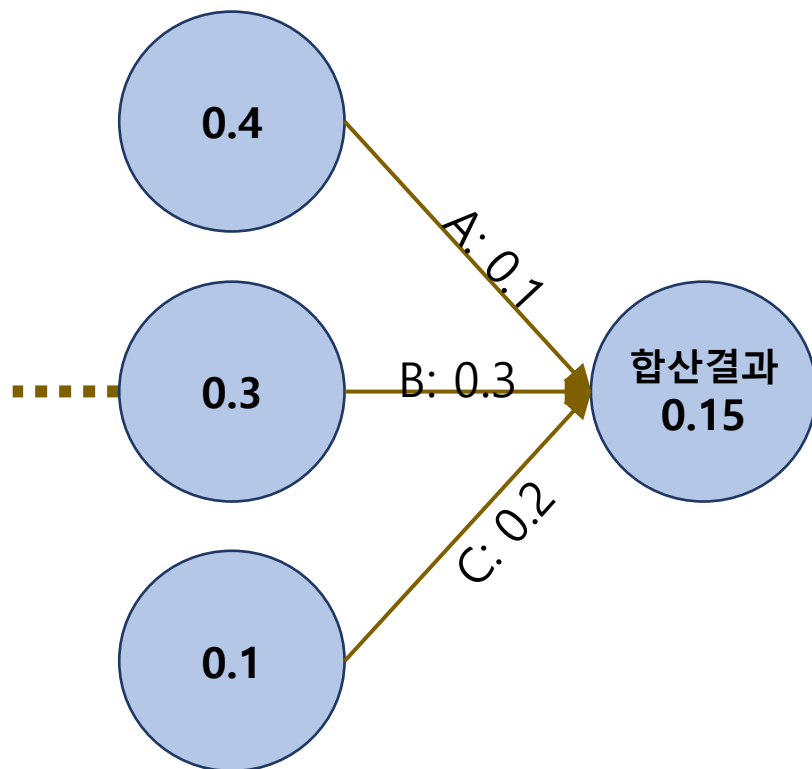
## • 역전파 시 어떻게 가중치를 조절하는가?

- 조절 내용: **은닉층을 거친 결과값**과 **기대한 결과값**의 오차를 줄이는 방향으로 수정



- 역전파 시 어떻게 가중치를 조절하는가?

- 조절 방법

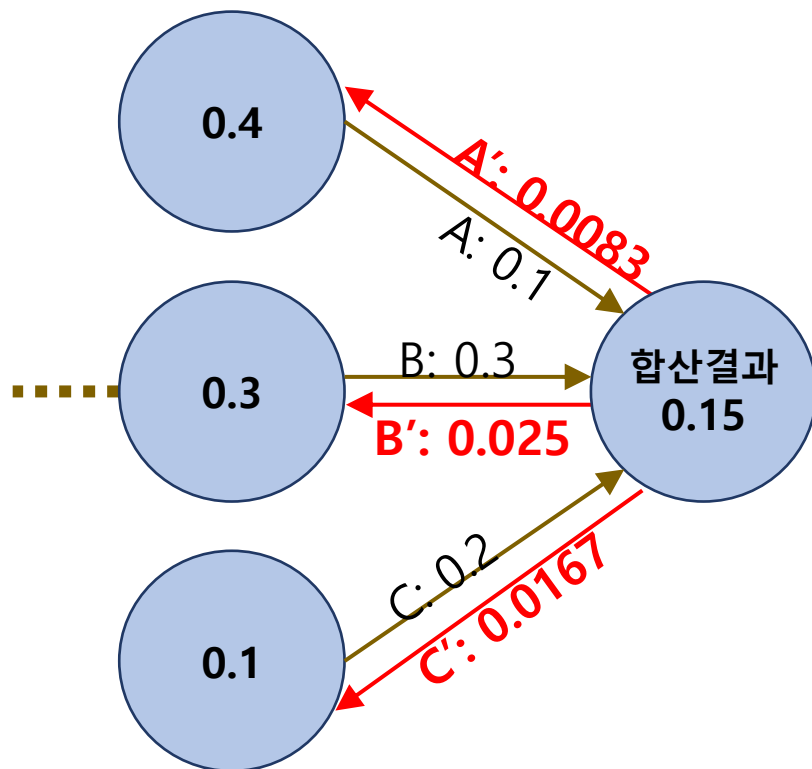


오차 값 0.05를 줄이기 위해서

- 가중치의 비율은 1 : 3 : 2
- 오차 0.05를 1 : 3 : 2 으로 나눈다
- $A' = 0.0083$
- $B' = 0.025$
- $C' = 0.0167$
- $A' + B' + C' = 0.05$

- 역전파 시 어떻게 가중치를 조절하는가?

- 조절 방법

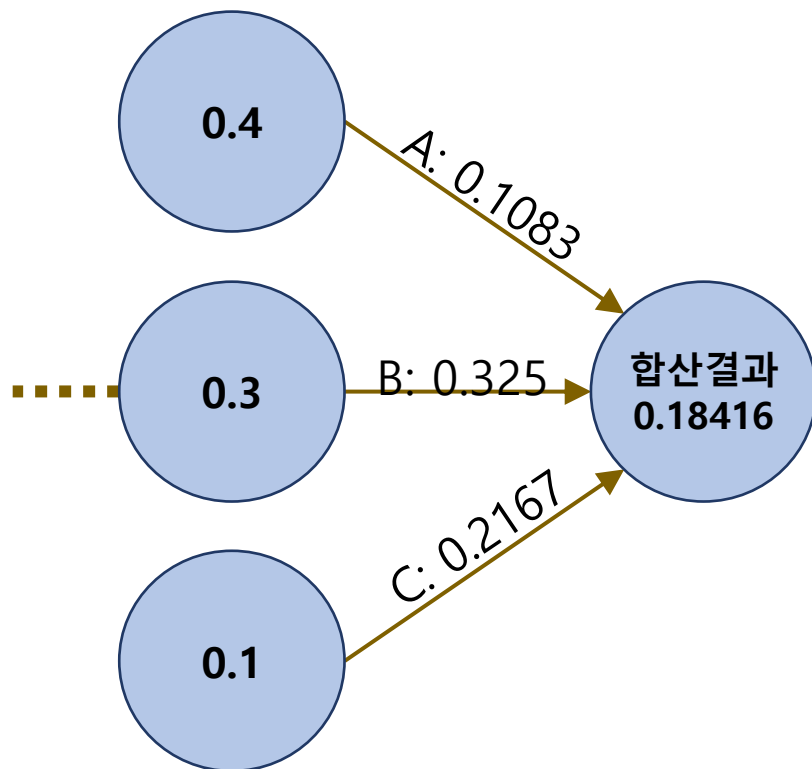


되돌려준 후 반영한다

- $A = 0.1 + 0.0083 = 0.1083$
- $B = 0.3 + 0.025 = 0.325$
- $C = 0.2 + 0.0167 = 0.2167$

- 역전파 시 어떻게 가중치를 조절하는가?

- 조절 방법



## 반영 후 다시 계산

- $A = 0.1 + 0.0083 = 0.1083$
- $B = 0.3 + 0.025 = 0.325$
- $C = 0.2 + 0.0167 = 0.2167$
- $0.4 \times A + 0.3 \times B + 0.1 \times C$   
 $= 0.04332 + 0.0975 + 0.04334 = 0.18416$   
→ 기대 값 0.2와 비교하여 → 오차 0.01584  
→ 기존의 오차 0.05보다 줄어듦

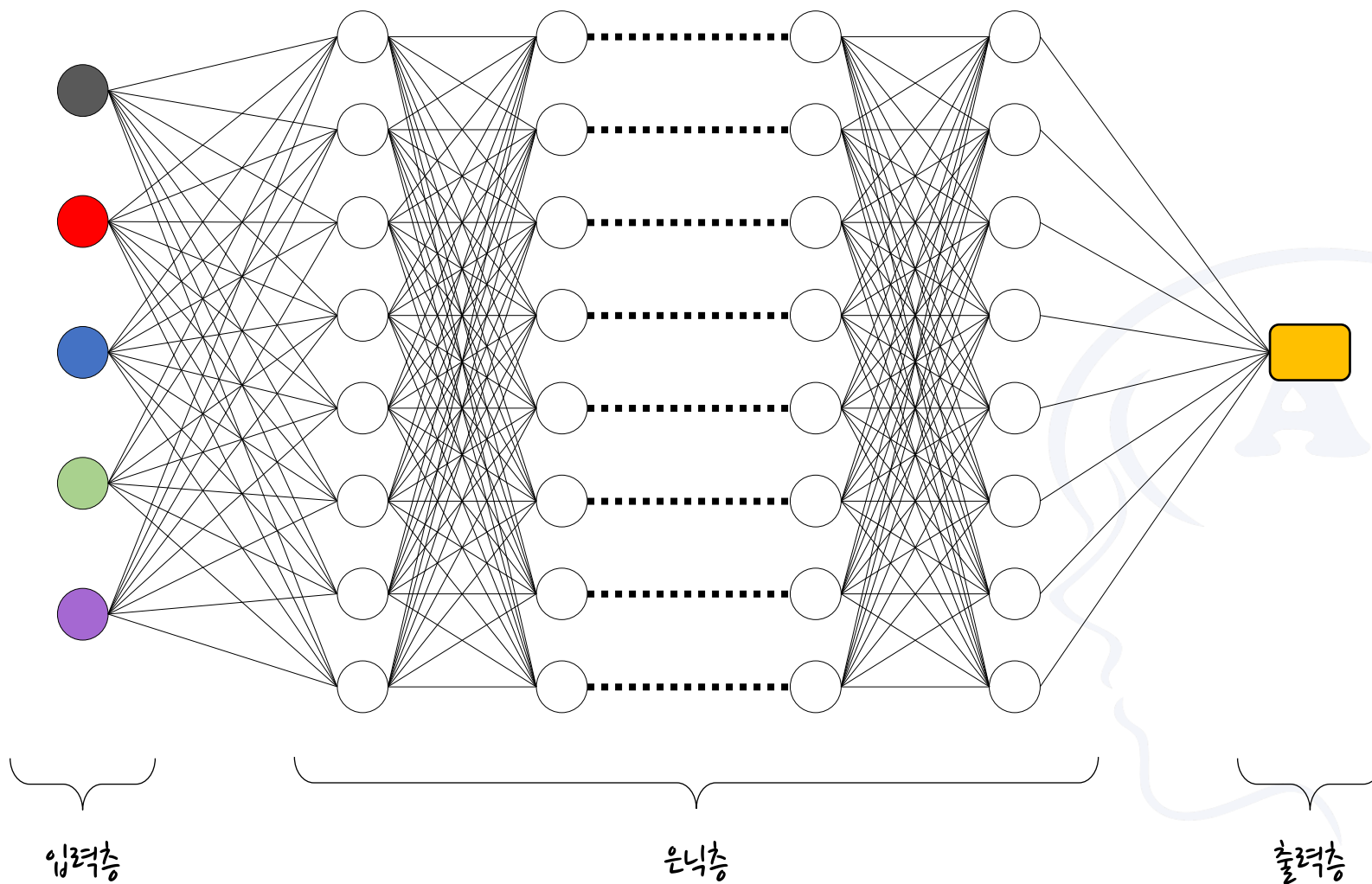
# Back Propagation (역전파) 알고리즘



- Back Propagation (역전파) 알고리즘을 구현한 다층 신경망에서
- 그 층을 훨씬 많이 만들어서
- 수 많은 분류 작업을 수행할 수 있게 한다면?
- 다층 신경망(Multilayer Neural Network)
  - 심층 신경망(Deep Neural Network) 으로 진화
- 심층 신경망을 이용한 학습 모델 = 딥 러닝(Deep Learning)

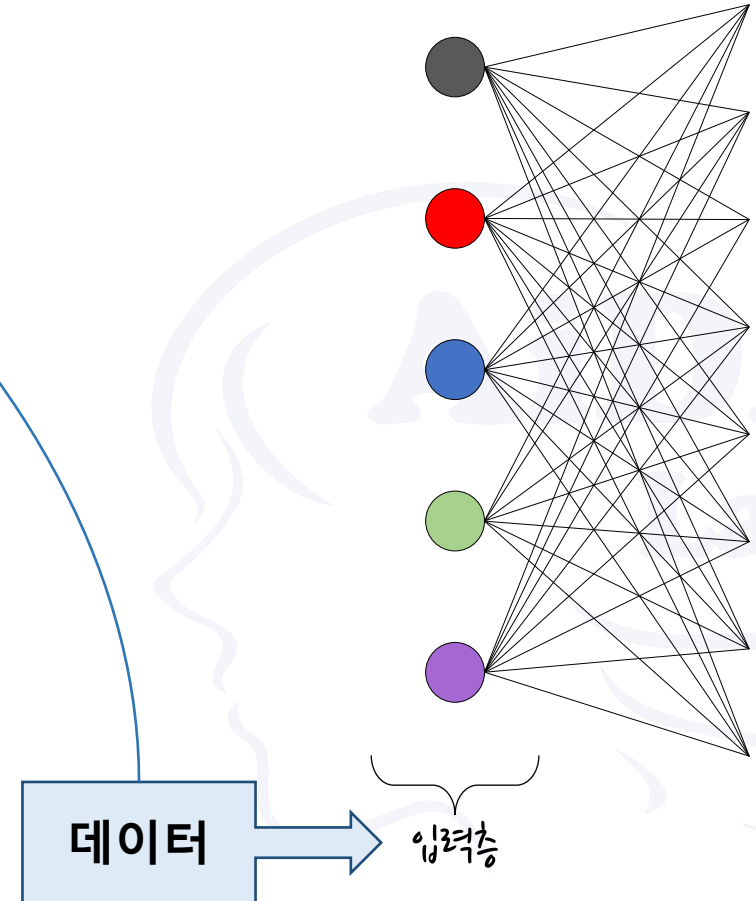


# 가장 기본적인 딥러닝 모델: 전체 구조



# 가장 기본적인 딥러닝 모델: 입력층

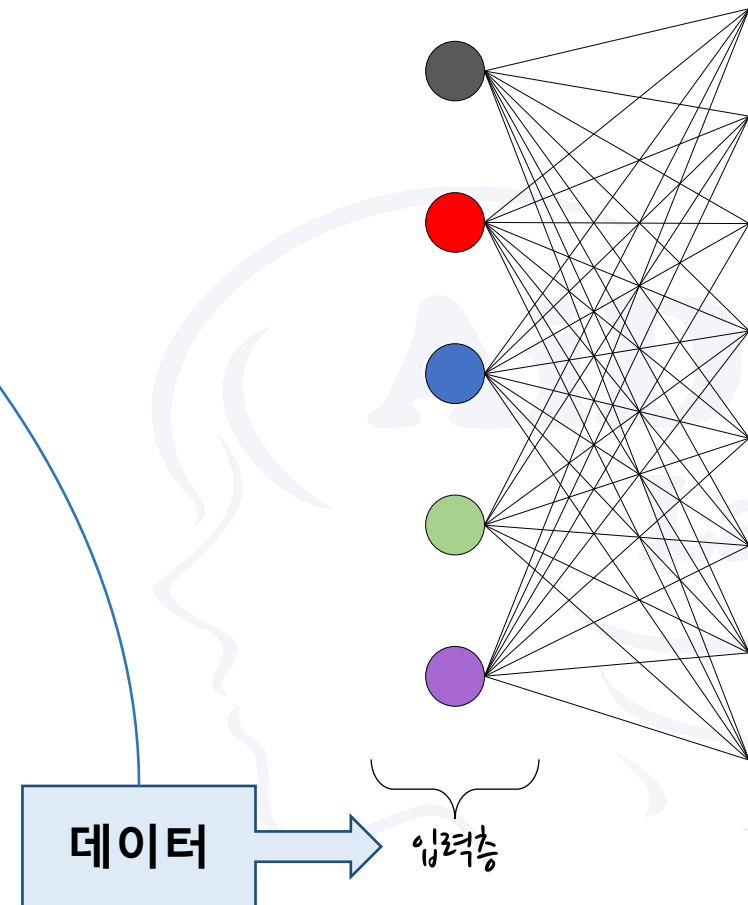
어떤 데이터들이 입력되는가?



# 가장 기본적인 딥러닝 모델: 입력층

어떤 데이터들이 입력되는가?

- 이미지(사진) 데이터
- 동영상 데이터
- 센서 데이터
- 주식 데이터
- 기상관측 데이터
- 등...
  
- SNS 데이터
- Web Scraping 데이터
- 등...



# 가장 기본적인 딥러닝 모델: 입력층

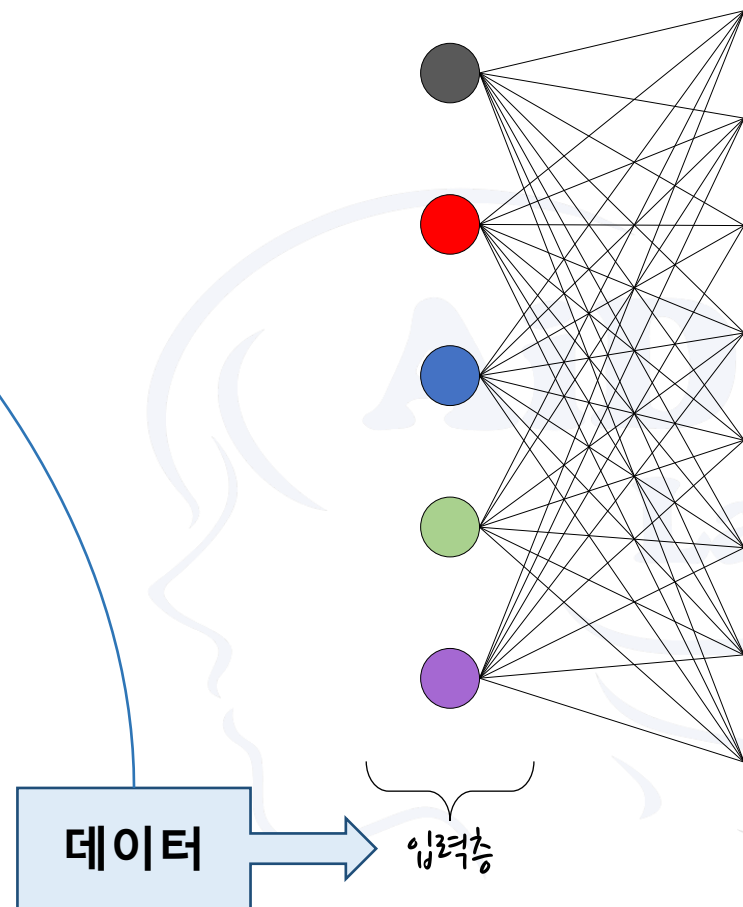
어떤 데이터들이 입력되는가?

- 이미지(사진) 데이터
- 동영상 데이터
- 센서 데이터
- 주식 데이터
- 기상관측 데이터
- 등...

수치 데이터

- SNS 데이터
- Web Scraping 데이터
- 등...

문자(열) 데이터 → 수치화



# 가장 기본적인 딥러닝 모델: 입력층

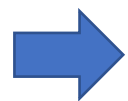
어떤 데이터들이 입력되는가?

- 이미지(사진) 데이터
- 동영상 데이터
- 센서 데이터
- 주식 데이터
- 기상관측 데이터
- 등...

수치 데이터

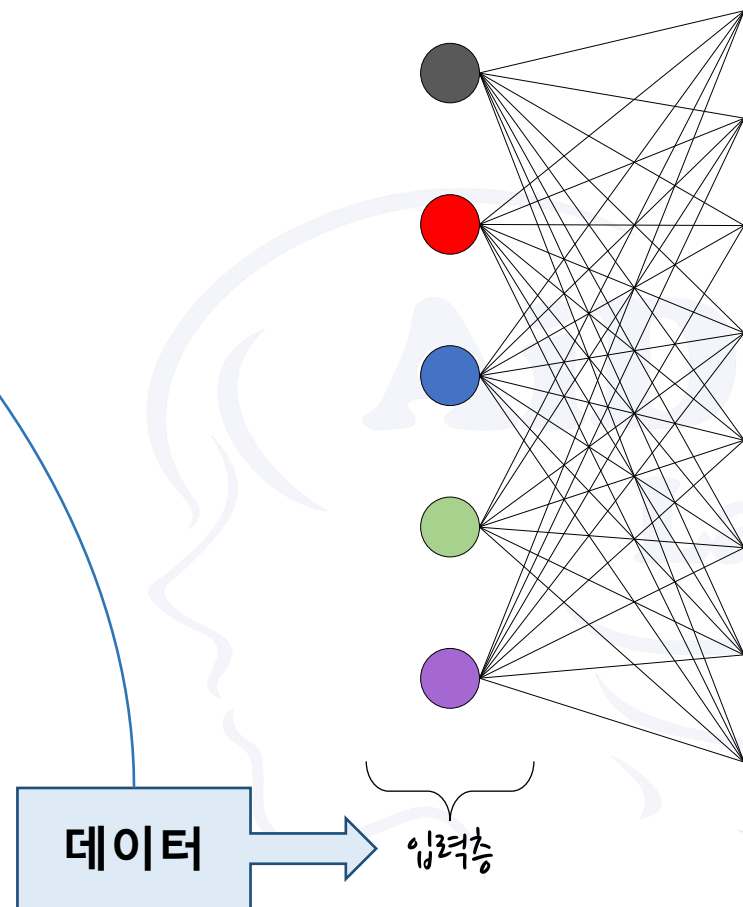
- SNS 데이터
- Web Scraping 데이터
- 등...

문자(열) 데이터 → 수치화



딥러닝에서 사용되는 데이터는

기본적으로 수치 데이터



# 가장 기본적인 딥러닝 모델: 입력층

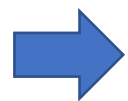
어떤 데이터들이 입력되는가?

- 이미지(사진) 데이터
- 동영상 데이터
- 센서 데이터
- 주식 데이터
- 기상관측 데이터
- 등...

수치 데이터

- SNS 데이터
- Web Scraping 데이터
- 등...

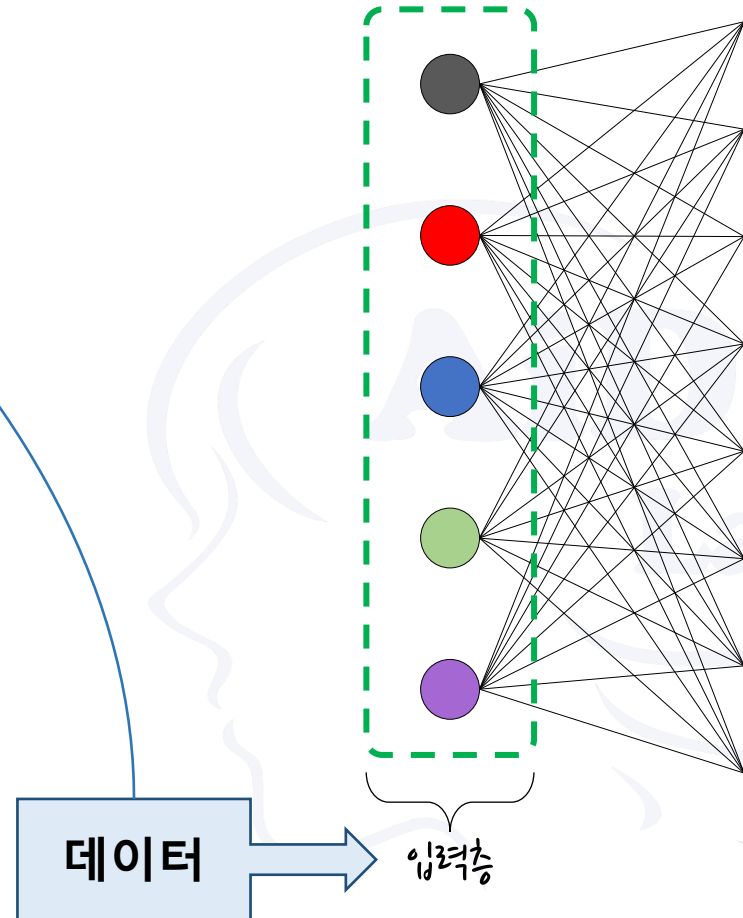
문자(열) 데이터 → 수치화



딥러닝에서 사용되는 데이터는

기본적으로 수치 데이터

각 노드에 하나씩 입력  
→ 한 줄로 이어진 데이터



# 가장 기본적인 딥러닝 모델: 입력층

어떤 데이터들이 입력되는가?

- 이미지(사진) 데이터
- 동영상 데이터
- 센서 데이터
- 주식 데이터
- 기상관측 데이터
- 등...

수치 데이터

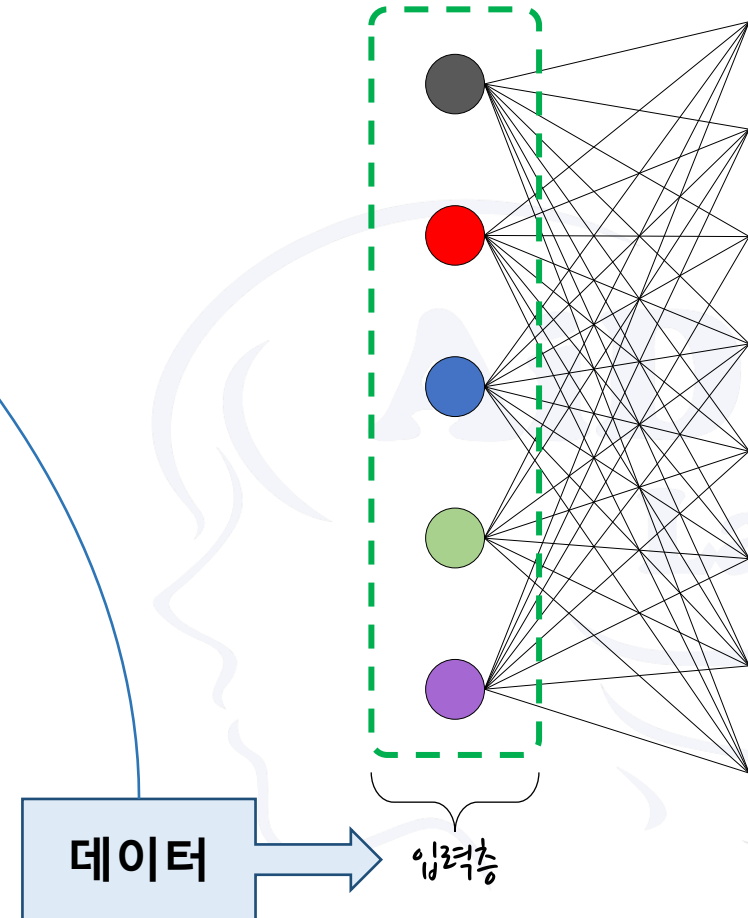
- SNS 데이터
- Web Scraping 데이터
- 등...

문자(열) 데이터 → 수치화

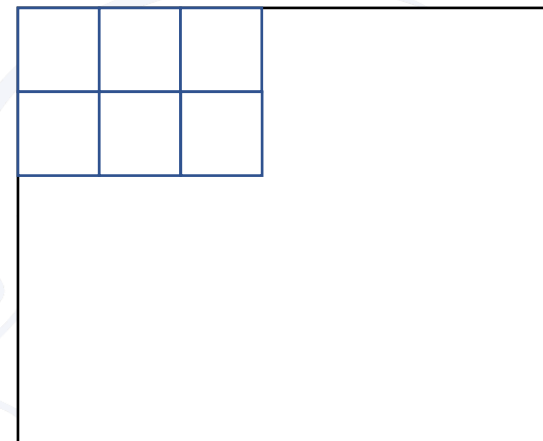
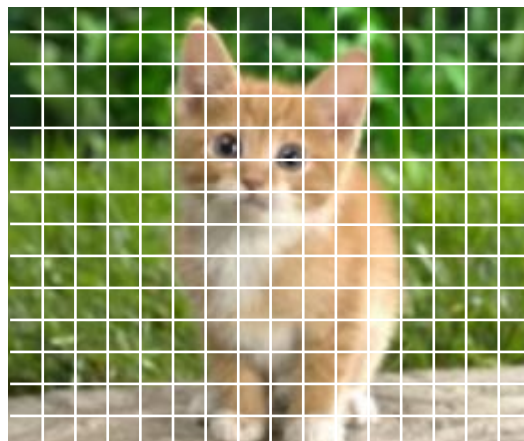
➡ 딥러닝에서 사용되는 데이터는  
기본적으로 수치 데이터

➡ 1차원 배열로 이루어진 수치 데이터

각 노드에 하나씩 입력  
→ 한 줄로 이어진 데이터



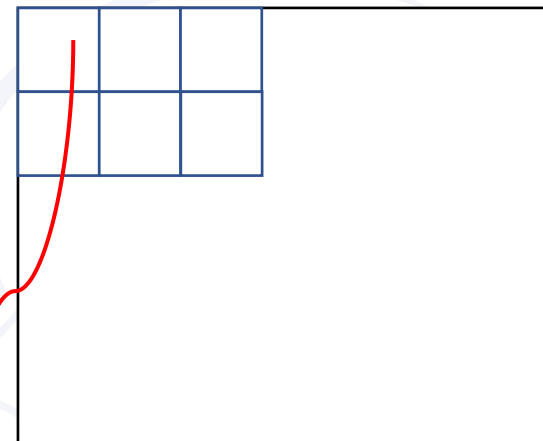
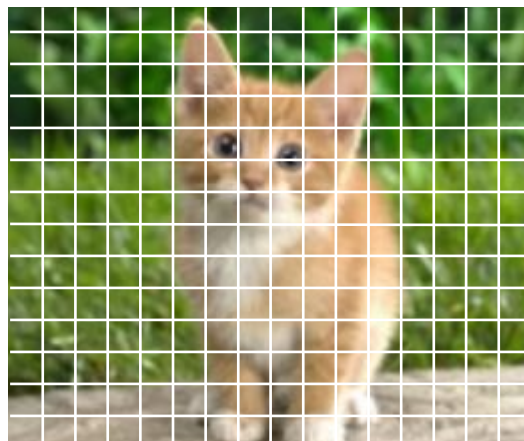
- 이미지 / 영상 데이터가 왜 수치 데이터인가?
  - 이미지 데이터는 색깔을 가진 수많은 점이 가로x세로 크기의 2차원 배열 속에 모인 데이터





# 가장 기본적인 딥러닝 모델: 입력층

- 이미지 / 영상 데이터가 왜 수치 데이터인가?
  - 이미지 데이터는 색깔을 가진 수많은 점이 가로x세로 크기의 2차원 배열 속에 모인 데이터

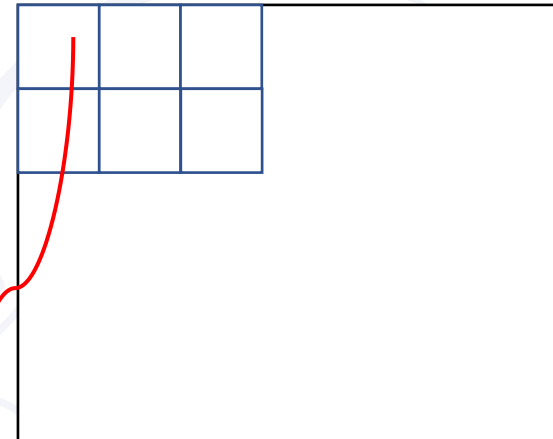
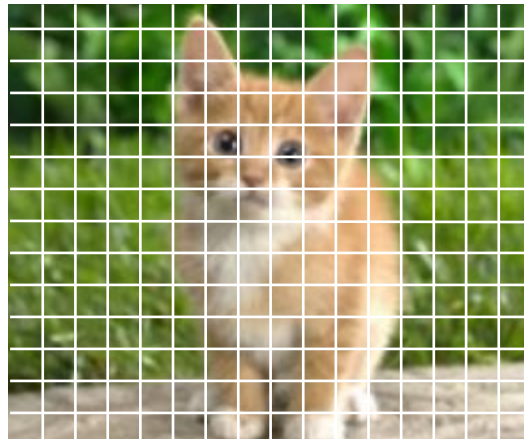


32832 = #008040 = #00 #80 #40 = R G B =

# 가장 기본적인 딥러닝 모델: 입력층

- 이미지 / 영상 데이터가 왜 수치 데이터인가?

- 이미지 데이터는 색깔을 가진 수많은 점이 가로x세로 크기의 2차원 배열 속에 모인 데이터



➡ 배열 데이터를 1차원으로 변환하여 입력 데이터로 사용  $32832 = \#008040 = \#00 \#80 \#40 =$  R G B

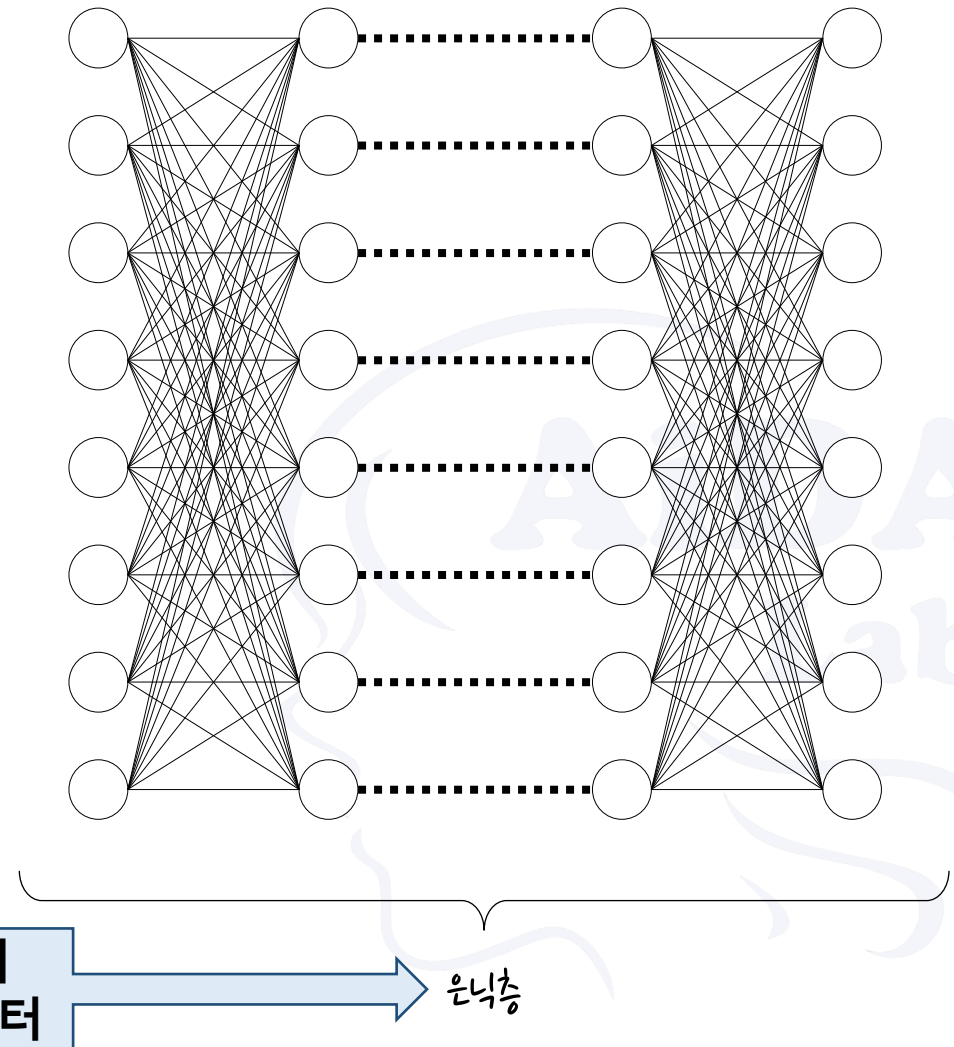
➡ 동영상 데이터는 다수의 이미지가 순서대로 연결된 것

# 가장 기본적인 딥러닝 모델: 은닉층

데이터는 어떻게 전달되는가?

- 입력층에서 입력된 데이터는
- 각 데이터가 첫번째 층의 모든 뉴런에
- 동일하게 전달됨
- 각 층의 모든 뉴런이 가진 데이터는
- 다음 층의 모든 뉴런에
- 동일하게 전달됨

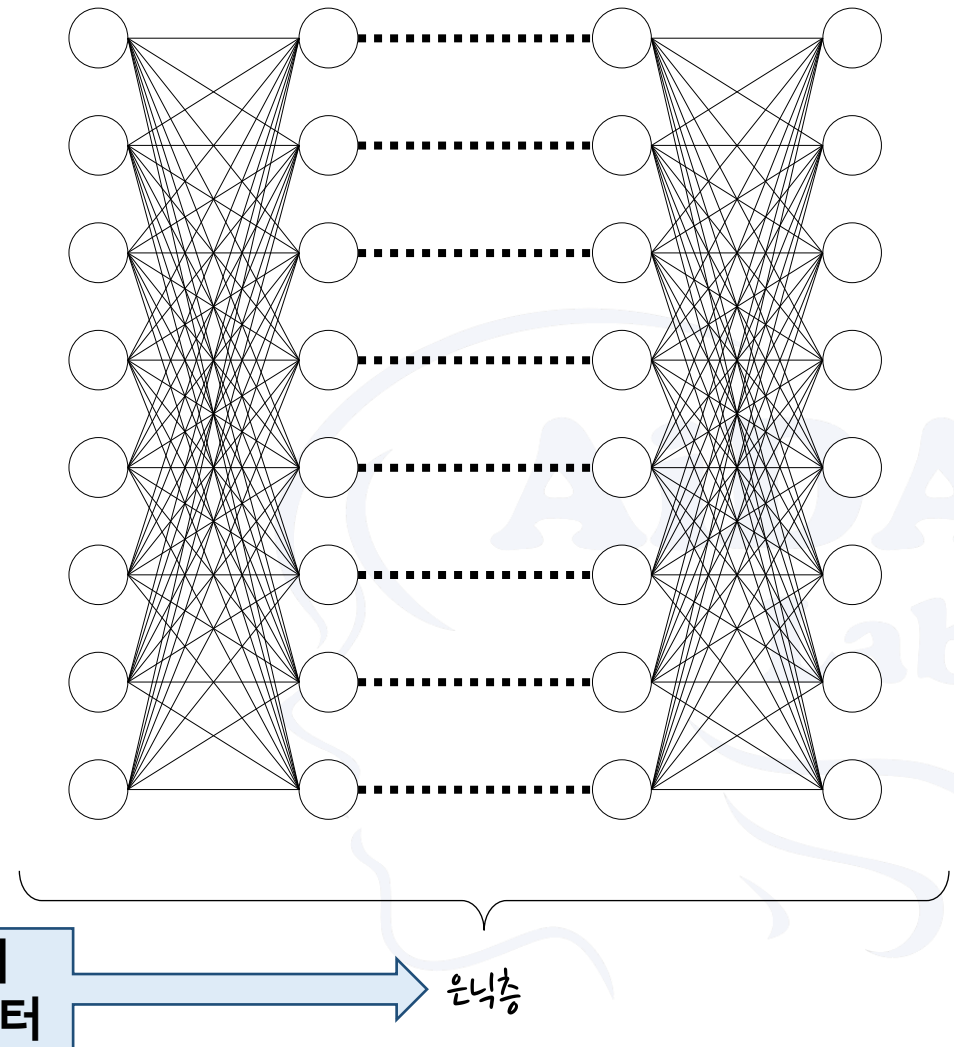
실제의 신경망에서는 신경전달물질이 퍼져 나가는 범위 안의 신경세포에만 신호가 전달됨 (모델링의 한계)



# 가장 기본적인 딥러닝 모델: 은닉층

각 뉴런에 전달된 데이터는 어떻게 가공되는가?

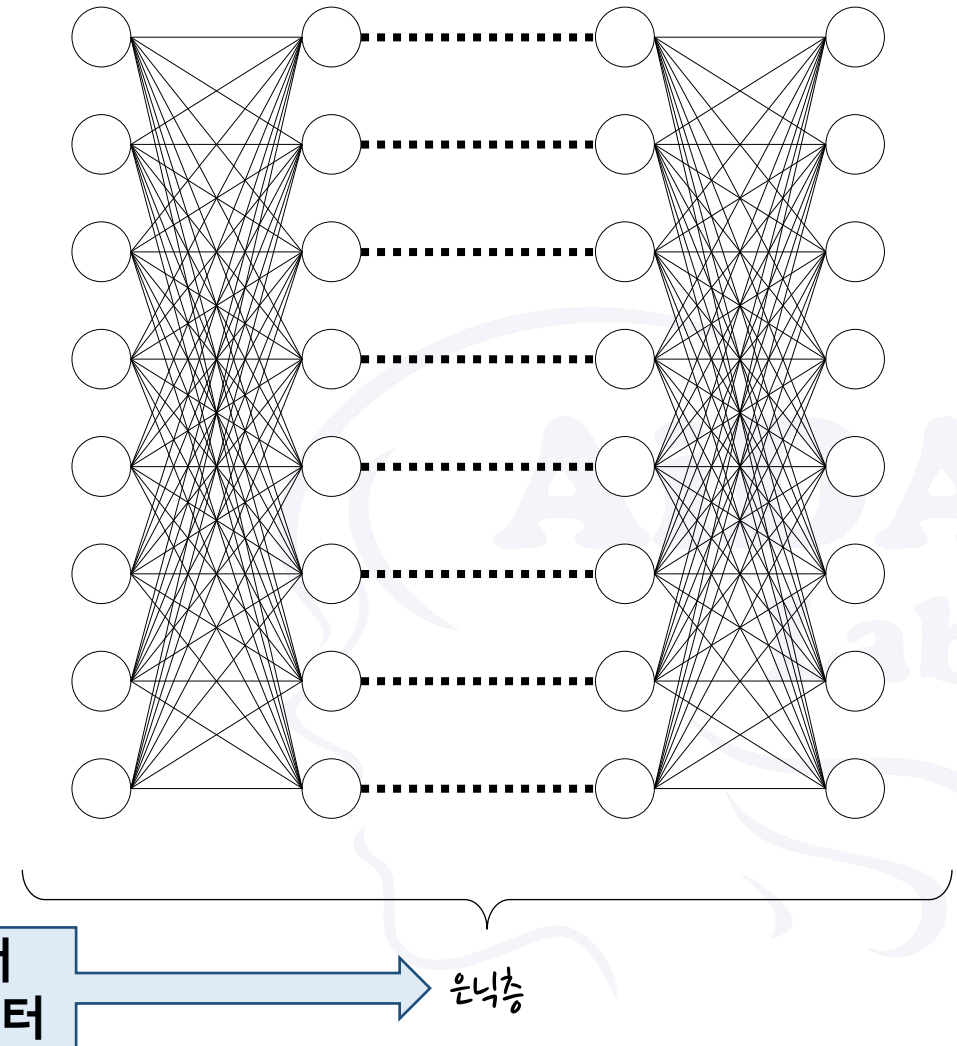
- 각 층의 뉴런은
- 입력되는 모든 데이터를
- 다 더한 후(합산)
- 합산 결과를 활성화 함수에 적용하고
- 활성화 함수(F1) 적용 결과를
- 다음 층의 뉴런으로 전달



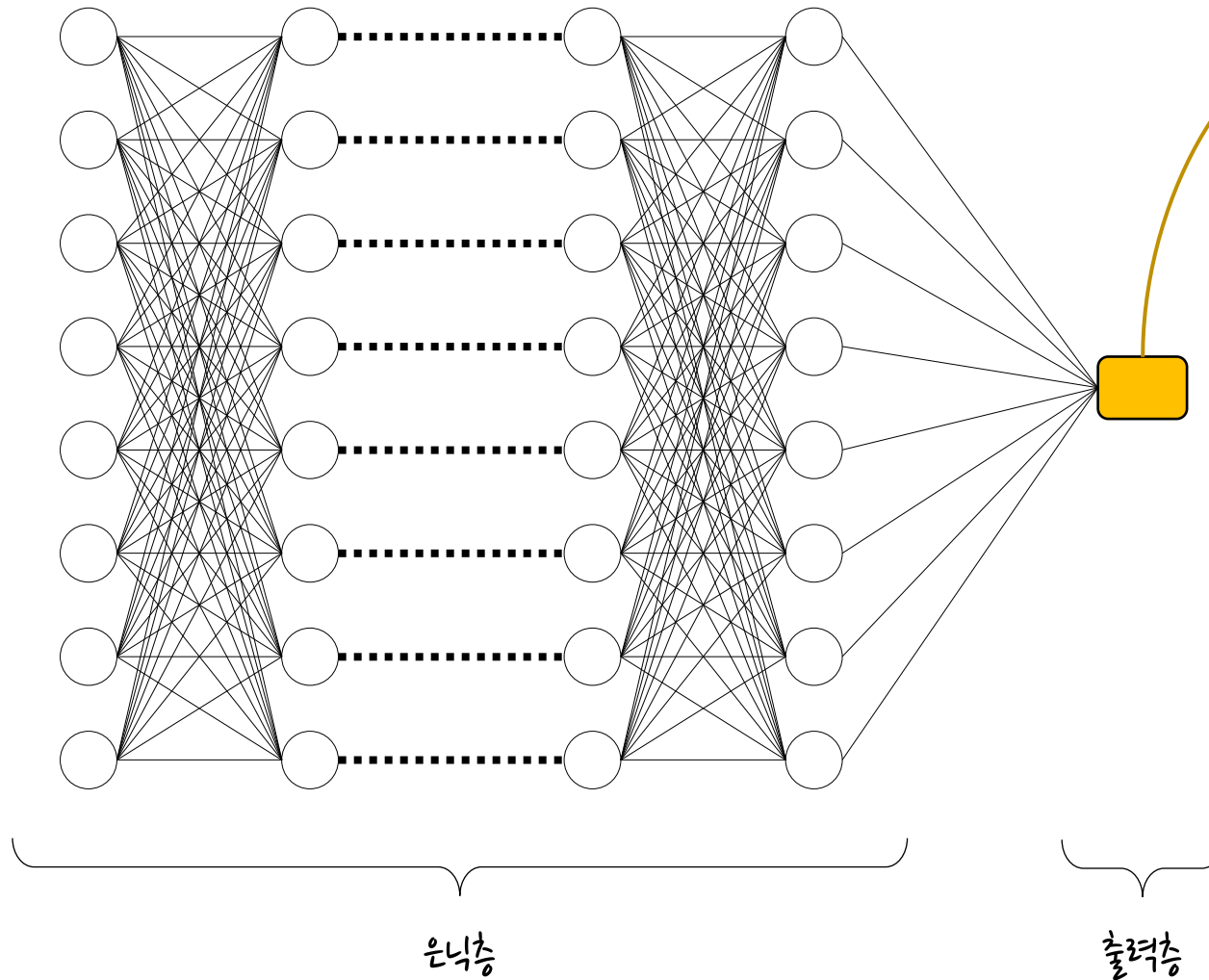
# 가장 기본적인 딥러닝 모델: 은닉층

은닉층은 데이터에 어떤 작업을 하는가?

- 각 층에 입력된 데이터는
  - 다음 층의 뉴런으로 가는 통로에 설정된 가중치를
  - 현재의 뉴런이 가진 데이터에 곱하여
  - 그 결과를 다음 층의 뉴런으로 전달
- 
- 모든 층의 데이터 이동에서
  - 가중치의 곱셈과 입력 값의 합산은
  - 동일하게 적용됨



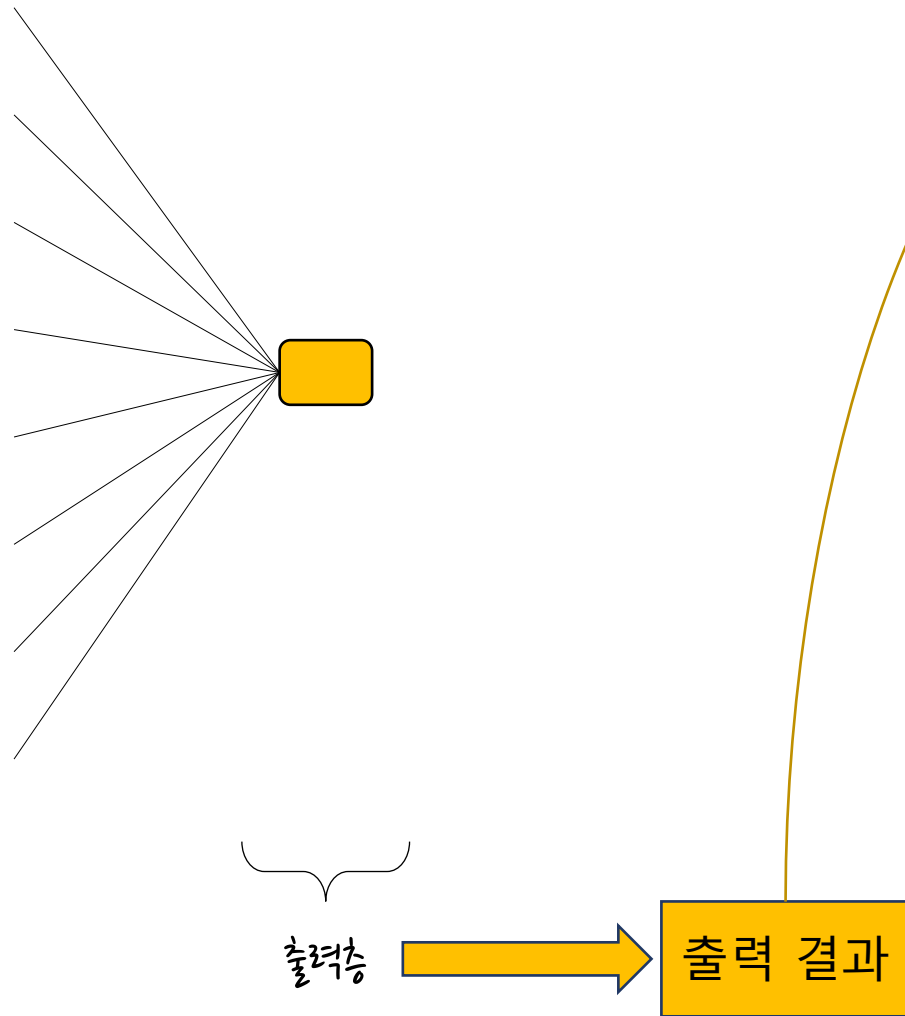
# 가장 기본적인 딥러닝 모델: 은닉층



은닉층과 출력층은 함께 어떤 작업을 하는가?

- 은닉층의 마지막 층의 뉴런들은
  - 자기에게 입력된 데이터를
  - 합산 후
  - 활성화 함수(F1)를 적용하고
  - 적용 결과를
  - 출력층의 출력 뉴런으로 전달
- 
- 출력 뉴런은
  - 전달된 모든 데이터를
  - 순서대로 나열하여(벡터화)
  - 활성화 함수(F2)를 적용하고
  - 적용 결과(벡터값)를
  - 결과로서 출력함

# 가장 기본적인 딥러닝 모델: 출력층

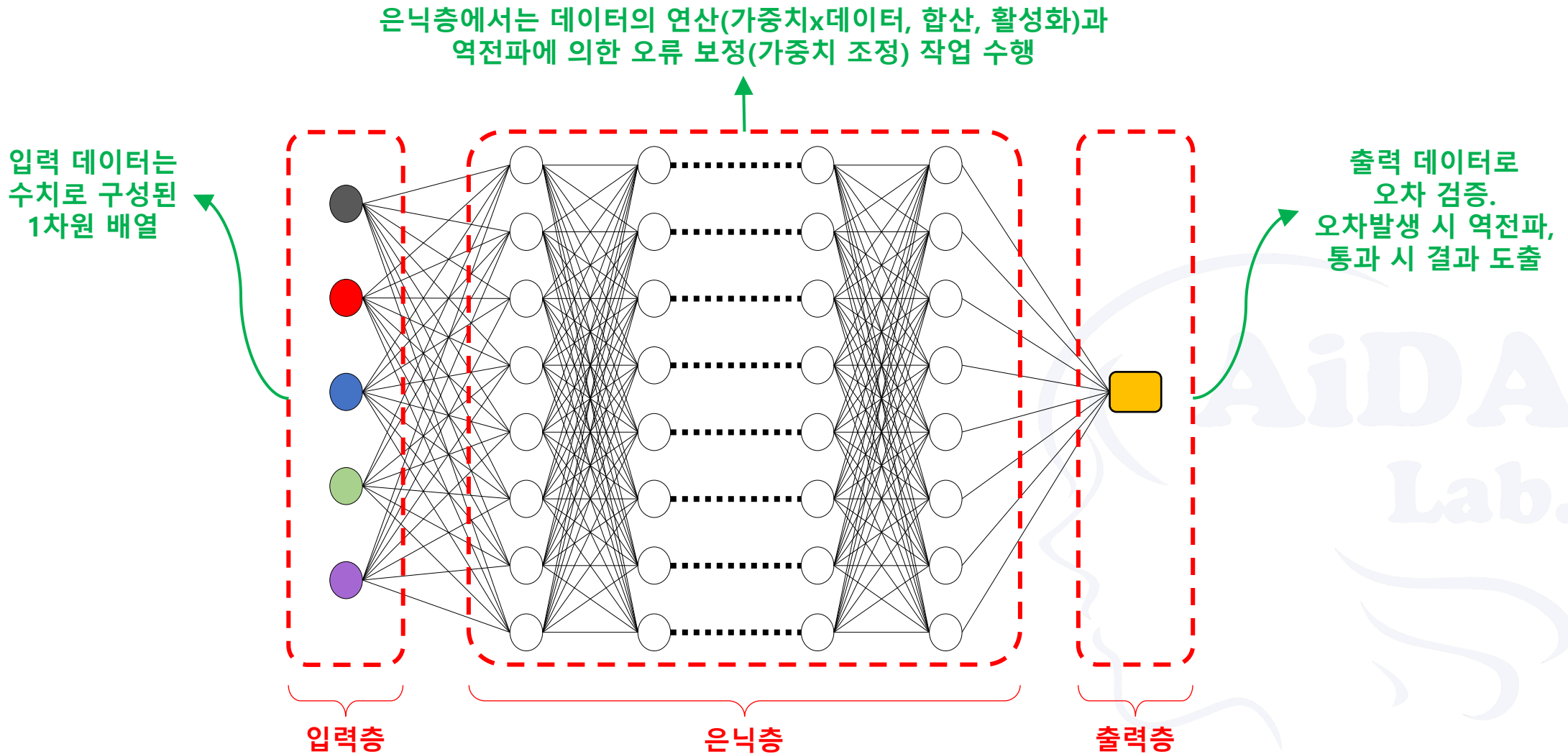


출력 층에서는 어떤 결과가 나오는가?

- 출력 뉴런은
  - 전달된 모든 데이터를
  - 순서대로 나열하여(벡터화)
  - 활성화 함수(F2)를 적용하고
  - 적용 결과(벡터값)를
  - 결과로서 출력함
- 
- 출력층의 결과는
  - 미리 입력된 기대 결과값과 비교하여(지도학습)
  - 일치하면 → 학습 완료
  - **일치하지 않으면 → 진행의 반대 방향으로 다시 전달**



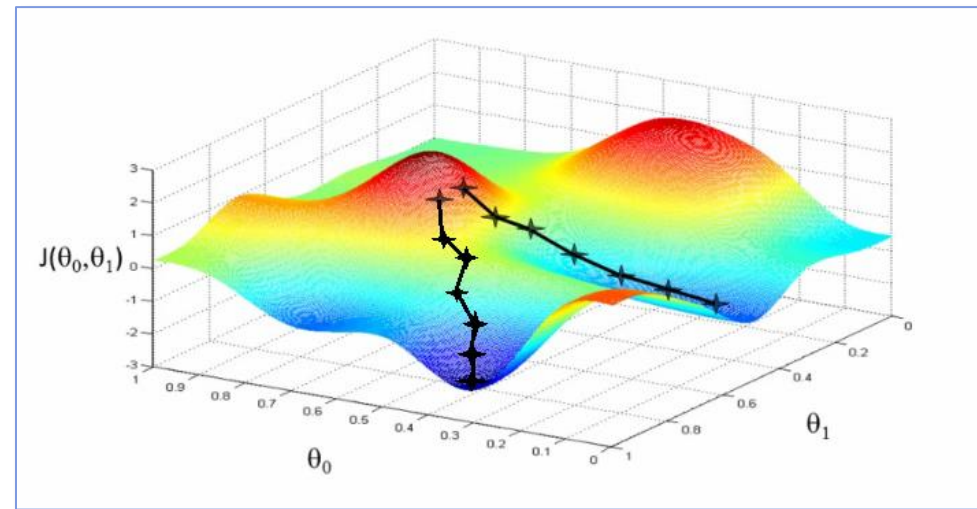
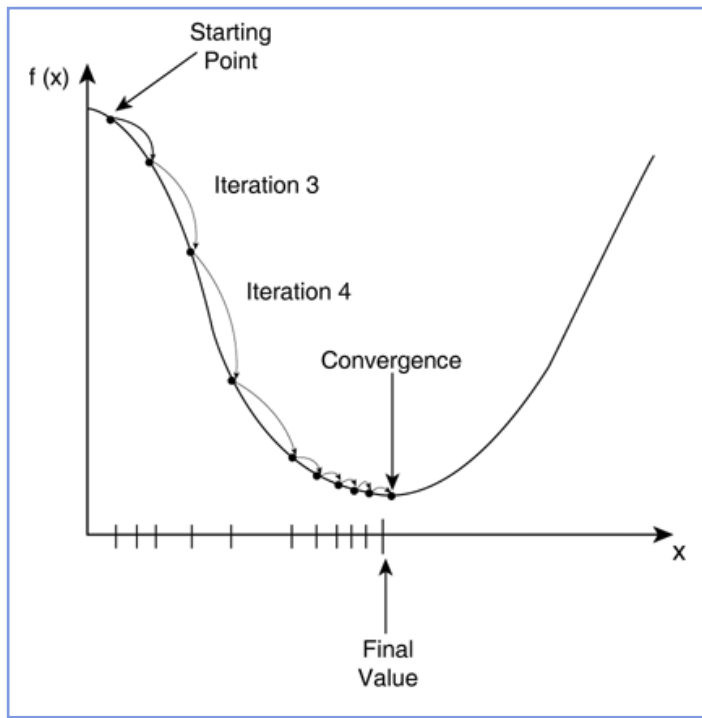
# 딥러닝 기본 모델





- 가장 많이, 기본적으로 사용되는 가중치 조정 방법
  - 경사 하강법 (Gradient Descent)

초기값에 따라  
최저점이  
달라질 수 있다

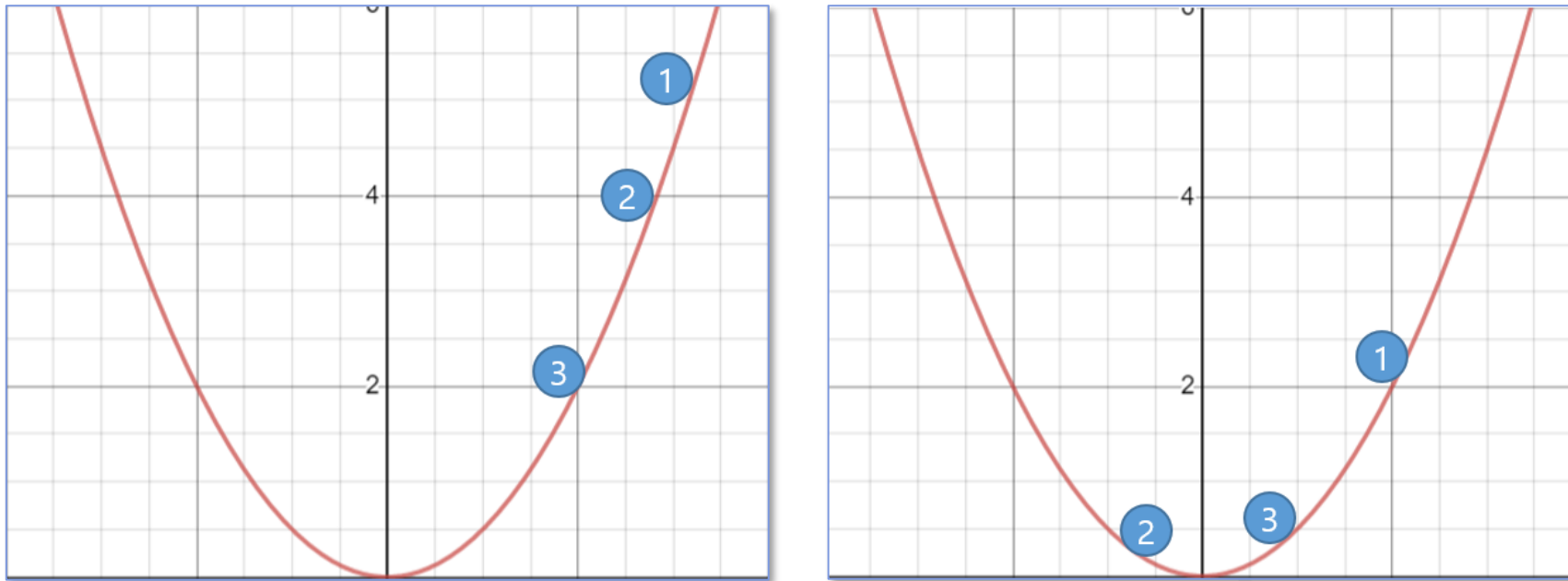


최저점은 하나가 아니다

이런 이유로...  
같은 목적과 같은 데이터로 학습하더라도  
모든 학습된 모델의 내부(가중치 집합)는 모두 다름

- 그 외의 가중치 조절 방법들..

- SGD, Momentum, AdaGrad, RMSprop, Adam 등



공이 굴러가듯이 움직인다

어떻게 굴러가는지, 어디에서 튀어 오르는지, 관성의 적용을 어떻게 받는지... 등

이런 특징들을 활용 또는 참고하여 새로운 방식을 고려할 수도 있다

- 가중치 계산 방법으로 사용할 수 없는 것은?
  - 딥러닝 모델에서의 가중치 계산, 조정은 미분의 개념을 기반으로 움직임
  - 부드럽게 이어지지 않고 뾰족하거나 각진 형태로 인하여 미분이 불가능한 그라프의 형태를 가지는 계산 모델은 사용할 수 없음



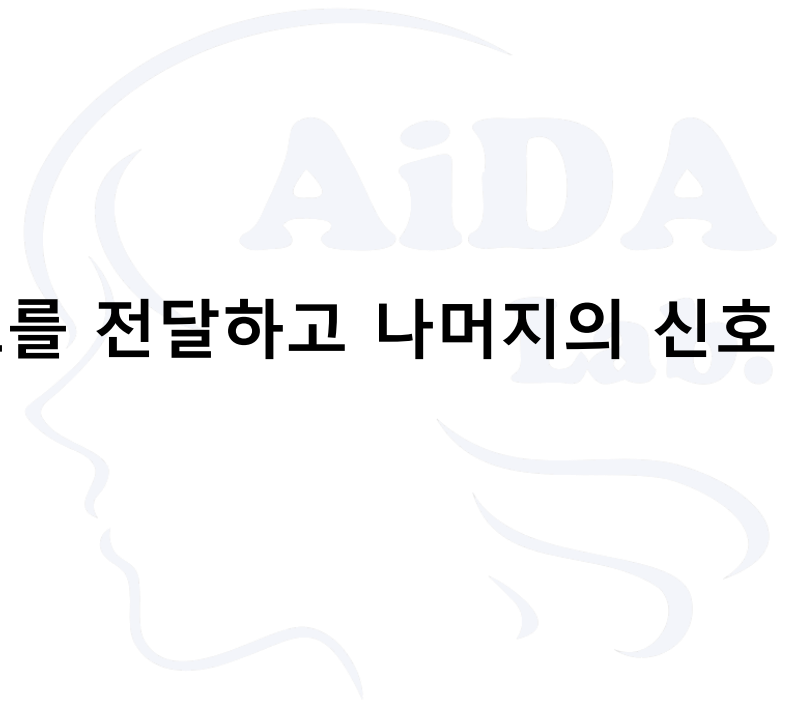
- 그럼 여기서... 각 퍼셉트론에서 임계 값을 넘으면 활성화는 어떻게 하나?
- 활성화 함수(Activation Function)
  - 신경망을 구성하는 각 퍼셉트론에서 임계 값을 넘었을 때 출력을 처리하는 함수
  - 함수의 정의
    - 입력: 이전 층의 디바이스 또는 퍼셉트론들로부터 전달되는 데이터
    - 함수의 동작: 입력 값의 합산 + 합산결과와 임계 값의 비교 + 출력 결정  
(활성화 조건)
    - 출력: 퍼셉트론 층의 연산 결과값. 다음 층의 뉴런에 대한 입력 또는 최종 층의 출력

- 그런데 활성화 함수는 왜 필요한가?
  - 피부, 눈과 같은 감각기관이 어떤 자극을 받아 신호를 발생시키면
  - 그 신호는 축삭을 통해서 이동하고
  - 축삭의 말단에 있는 시냅스를 거쳐
  - 다음 뉴런으로 전달



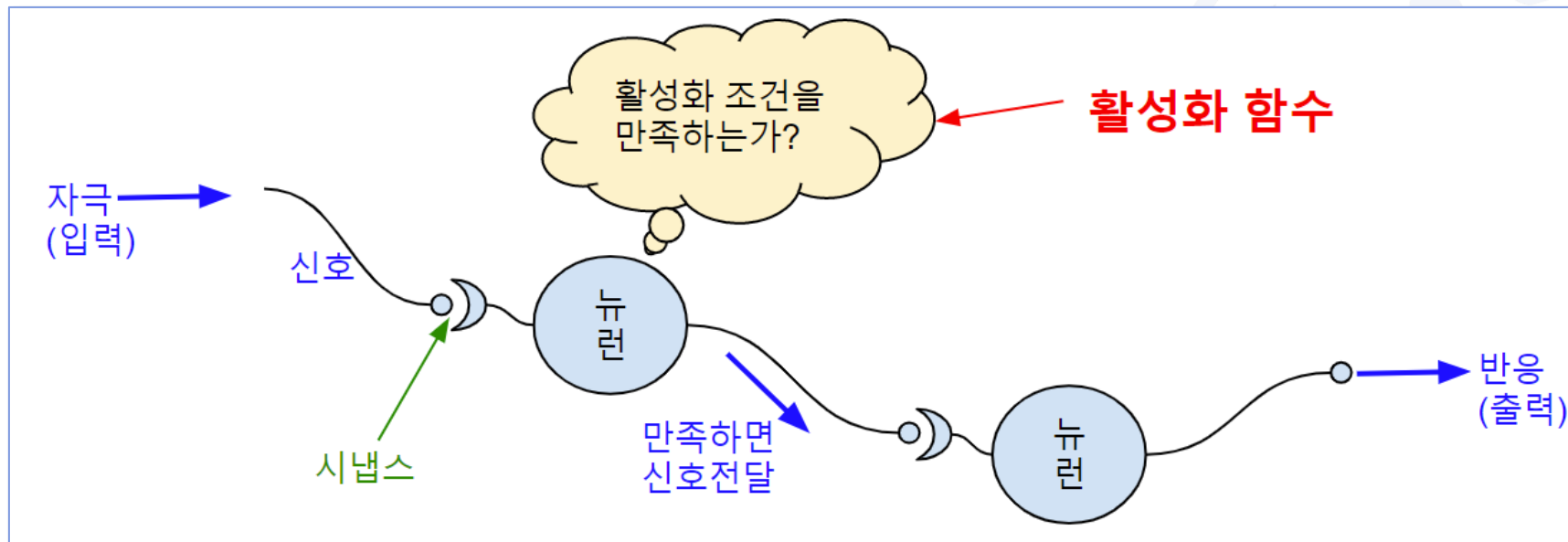
- 그런데!!!

- 전달되는 모든 신호(아주 미세한 신호부터 강한 신호까지)를 모두 다음 뉴런으로 전달한다면?
  - 생활 자체가 어려워짐
  - 매우 비 효율적
- 우리 몸에서 반응할 필요가 있는 수준까지만 신호를 전달하고 나머지의 신호는 무시한다!!
  - 진화의 결과
  - 이 기준을 모델에 반영한 것이 활성화 함수

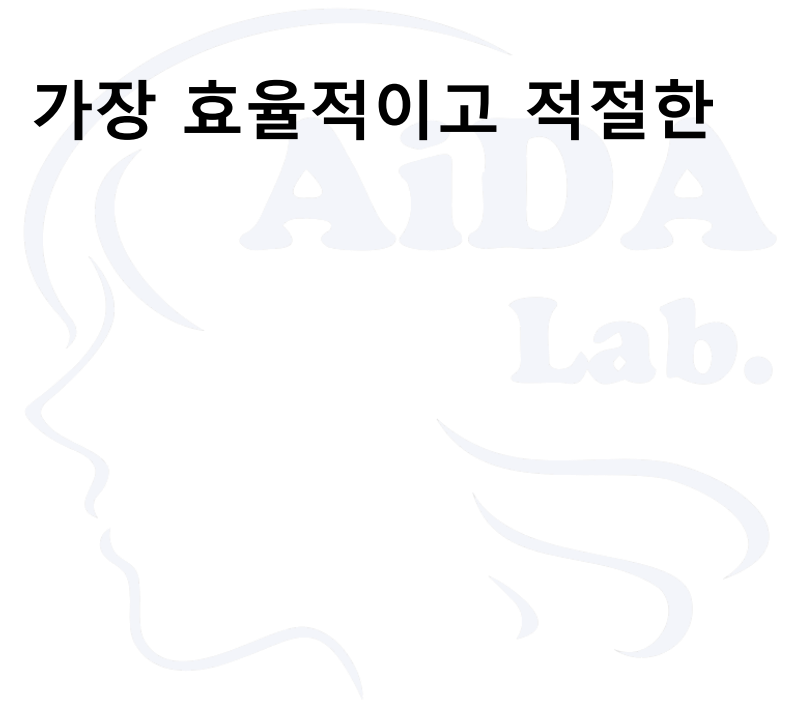


- 즉, 활성화 함수란

- 뉴런의 신호 흐름을 모델링 할 때 각각의 뉴런에 제한을 걸어 둔 것
- 활성화 함수에서 적용한 기준에 따라 조건을 만족하는 경우에만 다음 뉴런으로 신호 전달



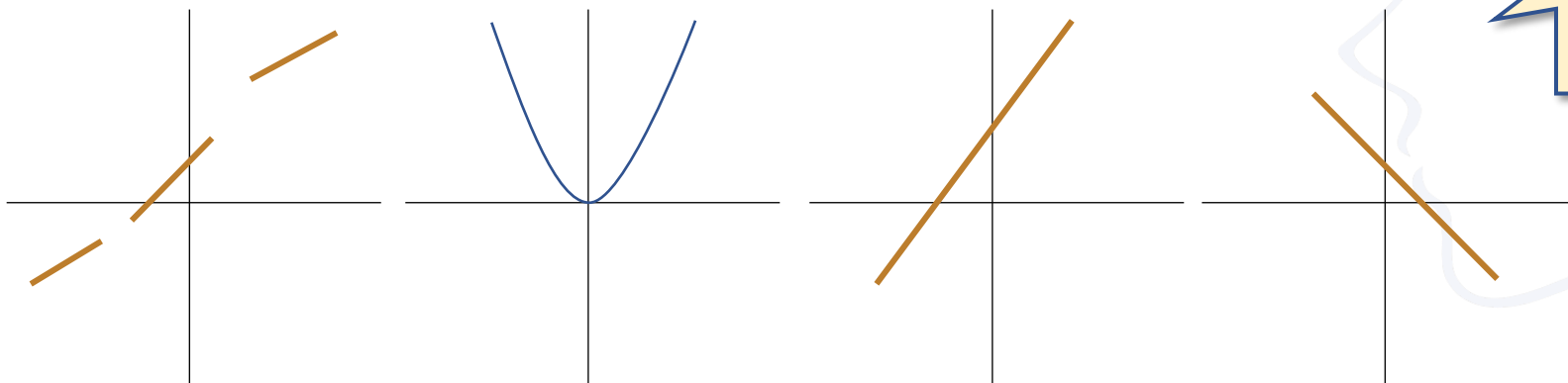
- 그럼 활성화 함수는 왜 그렇게 많은 형태가 존재하는가?
  - 우리는 아직 우리의 뇌와 신경들이 어떻게 동작하고 서로 어우러지는지 정확하게 알지 못함
  - 따라서 우리가 AI로 해결하고자 하는 문제에 맞게 가장 효율적이고 적절한 활성화 함수를 계속 연구, 개발하여 활용하는 것





## • 활성화 함수의 조건

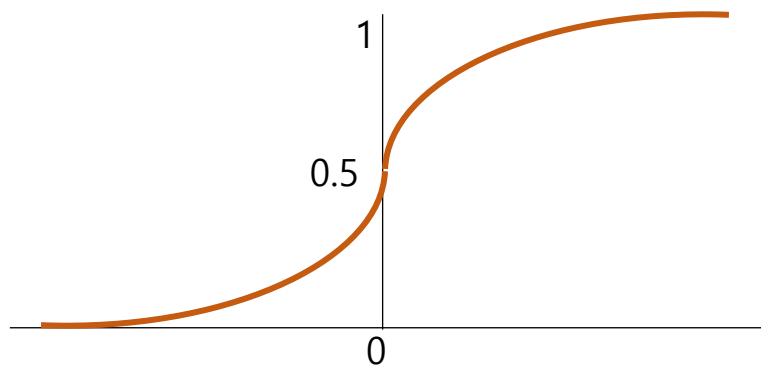
- 정의역(함수에 입력 가능한 값의 범위, 집합) 안에서 연속이며 무한해야 한다
- 단조 함수여야 한다 (방향을 바꾸지 않아야 한다)
- 비선형 함수여야 한다.
- 계산 효율이 좋아야 한다.



이런 함수들은  
부적합

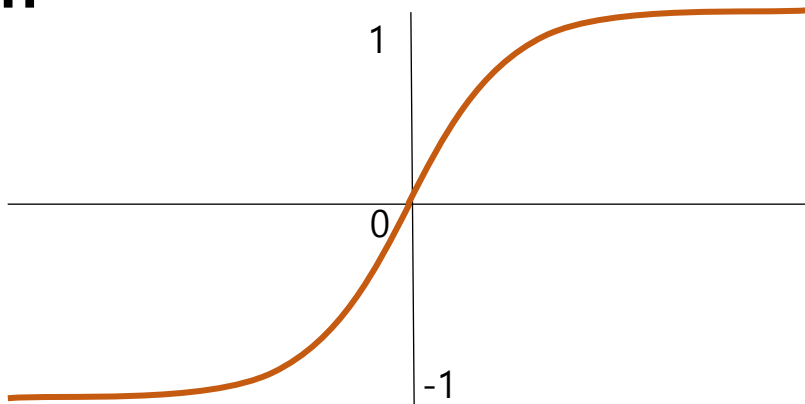
- 표준 은닉 활성화 함수

- Sigmoid



가장 많이 사용되어 왔고  
가장 중요한 활성화 함수  
(활성화 함수의 기본 형태)

- tanh



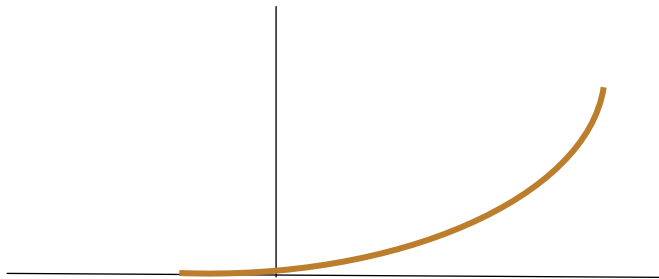
은닉층에서는 sigmoid 보다 tanh 함수가 더 좋음  
음의 상관관계도 지원

- 표준 출력 계층 활성화 함수
  - 신경망의 목적에 따라 최선의 선택이 달라진다
    - 일반 데이터값 예측 → 활성화 함수 미적용
    - 서로 무관한 항목에 대한 예/아니오 확률 예측 → sigmoid
    - 여러 가능성 중 하나의 확률 예측 → softmax
- 최근 가장 많이 쓰이는 활성화 함수
  - Relu



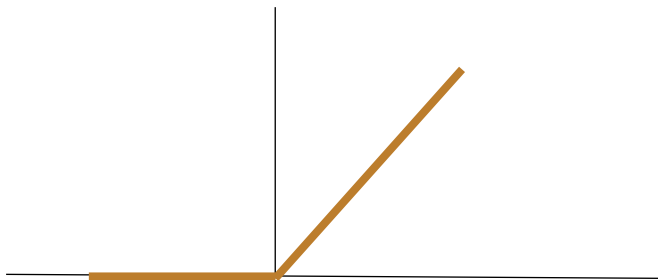
## • 표준 출력 계층 활성화 함수

### • Softmax



- k개의 값이 존재할 때 각각의 값의 편차를 확대시켜 큰 값은 상대적으로 더 크게, 작은 값은 상대적으로 더 작게 만든 후, 정규화 시키는 함수
- Softmax 함수를 거친 k개의 값의 총합은 1이 됨
- 지수증가를 기반으로 하는 함수

### • Relu (Rectified Linear Unit, 정류된 선형 유닛)



- 데이터가 0보다 작으면 무조건 0
- 데이터가 0보다 크면 입력값 그대로!! (주로 선형증가의 형태가 된다)

- 출력층에서는 왜 오차를 측정하는가?
  - 신경망의 목적은 정확한 예측 결과를 얻는 것
    - 분류를 위한 모델은? 분류 역시 어떤 클래스가 가장 잘 일치할 것인가..를 예측, 계산하여 그 값이 가장 큰 것을 선택하는 것이므로 동일하다고 볼 수 있음
- 예측을 한 후에는 얼마나 잘 예측했는가 평가해야 함
  - 평가 방법으로 오차의 측정을 사용 → 가장 간단하고 쉬운 방법이므로
- 특히 가중치의 조정은 미분과 관련이 있다는 것은  
→ **오차의 값은 양수만 사용**해야 한다는 의미

- 오차의 값은 왜 양수만 사용하는가?
  - 미분은 거리, 넓이를 이용한 개념 → 거리 또는 넓이는 음수가 없음
- 어떻게 양수만으로 오차를 처리할 것인가?
  - 실제로 오차를 측정하면 양수, 음수 모두 나올 수 있지만 각 값을 거리의 개념으로 바꿔서 사용
  - 절대값, 제곱 등을 이용하여 양수로 변환함

- 손실 함수 (Loss Function)

- 출력 값과 정답(기대 값)의 오차를 정의하는 함수
- 손실 함수는 데이터의 특성에 따라 변형, 새로 제안해서 사용 가능

- 종류

- 평균 제곱 오차 (MSE, Mean Squared Error)

- 가장 많이 사용됨. 출력 값과 기대 값의 차이를 제곱하여 평균한 값

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

- 큰 오차는 더욱 크게, 작은 오차는 더욱 작게 → **처리할 때에는 큰 오차에 더 집중**

(전반적인 성능 향상에 더 좋음)

- 교차 엔트로피 오차 (Cross Entropy Error)

- 범주형 데이터의 분류에 주로 사용

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

- 왜 손실함수를 사용하는가?
  - 학습의 궁극적인 목적은 높은 정확도를 끌어내는 매개변수를 찾는 것
  - 왜 “정확도”라는 지표를 놔두고 “손실함수의 값”이라는 우회적인 방법을 사용하는가?





## • 왜 손실함수를 사용하는가?

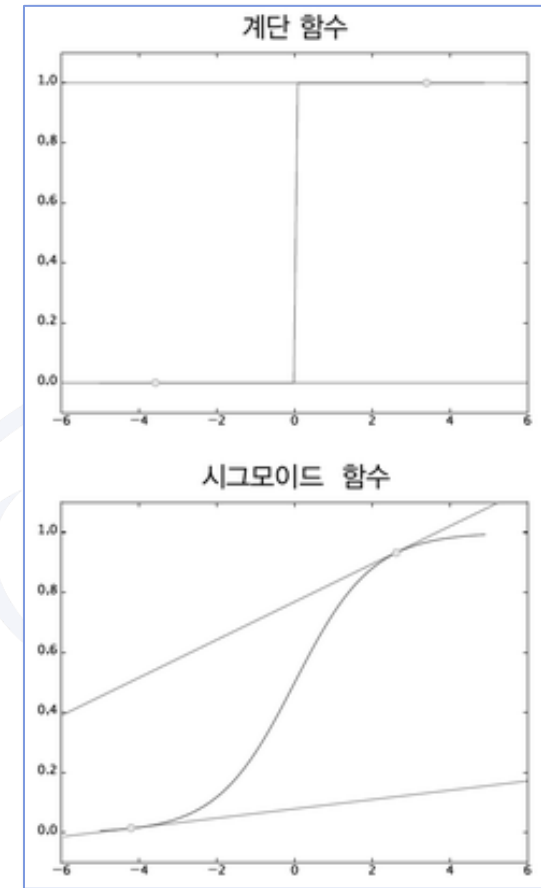
신경망 학습에서의 미분의 역할을 생각해 보면...

- 신경망 학습에서는 최적의 매개변수를 탐색할 때, 손실함수의 값을 가능한 작게 만드는 매개변수의 값을 찾음
- 이때 매개변수의 미분을 계산하고, 그 미분 값을 단서로 매개변수의 값을 서서히 갱신하는 과정을 반복함
  - 손실함수의 미분 값이 음수  $\rightarrow$  가중치 매개변수를 양의 방향으로 변화시켜 손실함수의 값을 줄일 수 있다.
  - 손실함수의 미분 값이 양수  $\rightarrow$  가중치 매개변수를 음의 방향으로 변화시켜 손실함수의 값을 줄일 수 있다.
  - 손실함수의 미분 값이 0  $\rightarrow$  어느 쪽으로도 움직이지 않으므로 갱신이 멈춘다.

- 정확도를 지표로 삼지 않는 이유
  - 미분 값이 대부분의 장소에서 0이 되어 매개변수를 갱신할 수 없다.
  - 정확도는 매개변수의 작은 변화에는 거의 반응을 보이지 않거나, 갑자기 변화한다.
  - 활성화 함수로 '계단 함수'가 아닌 '시그모이드 함수'를 사용하는 이유도 같다.

- 정확도를 지표로 삼지 않는 이유

- 계단함수는 대부분의 장소에서 기울기가 0 이지만, 시그모이드 함수의 기울기(접선)는 0이 아니다.
- 계단 함수는 한순간만 변화를 일으키지만, 시그모이드 함수의 미분은 연속적으로 변한다.
- 즉 시그모이드 함수의 미분은 어느 장소에서도 0이 되지 않는다.
- 이는 신경망 학습에서 중요한 성질로, 기울기가 0이 되지 않는 덕분에 신경망이 올바르게 학습할 수 있다.



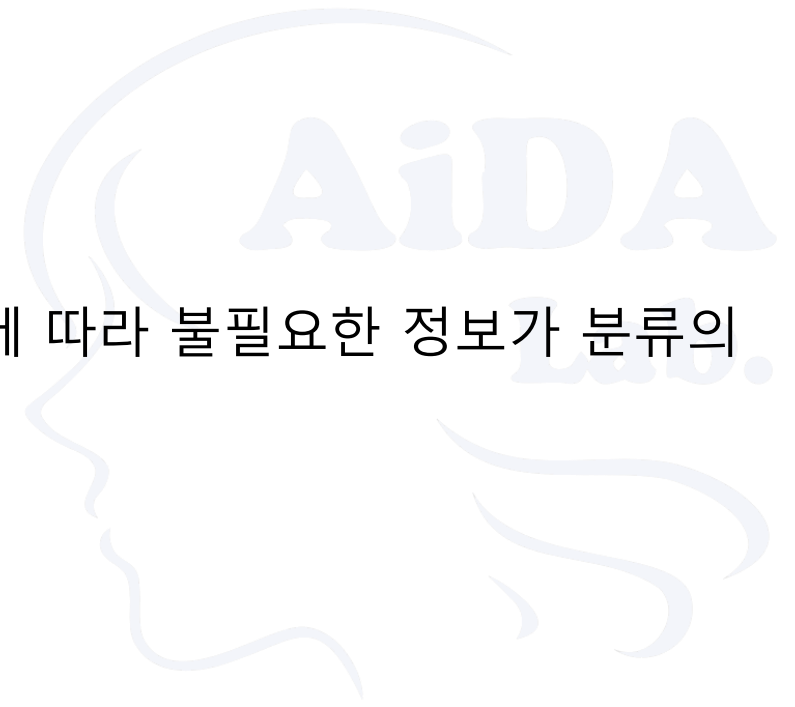
- **과적합(Over Fitting)**

- 주어진 데이터로 학습을 너무 많이 하면 오히려 역효과!!

- 학습에 입력된 데이터는 완벽에 가깝게 처리함
    - 학습에 입력되지 않은 데이터는 제대로 처리되지 않음

- **원인: 잡음 데이터 (대부분)**

- 불필요한 정보가 많이 포함된 데이터로 학습이 반복됨에 따라 불필요한 정보가 분류의 기준에 포함되어 버리는 것이 원인



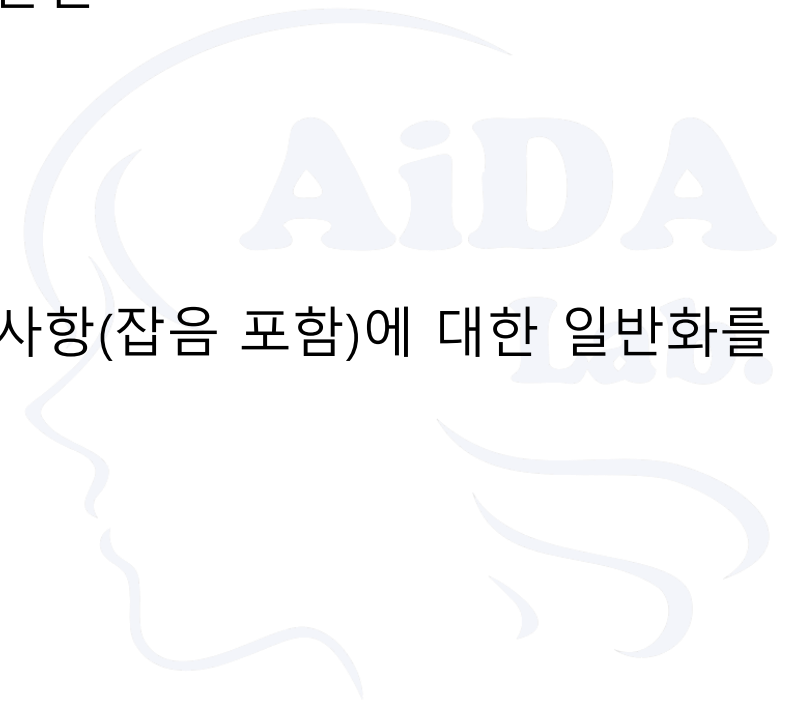
- **과적합(Over Fitting)의 해결 방안**

- **조기 종료**

- 적당한 선에서 학습을 종료시킴 → 데이터의 정규화와 관련

- **정규화 (데이터를 일반화 시키기)**

- 필요한 신호는 학습하고 잡음은 제거하는 효과
  - 모델의 학습 난이도를 높임으로써 학습 데이터의 세부 사항(잡음 포함)에 대한 일반화를 활용하도록 하는 기법의 일부로 사용됨

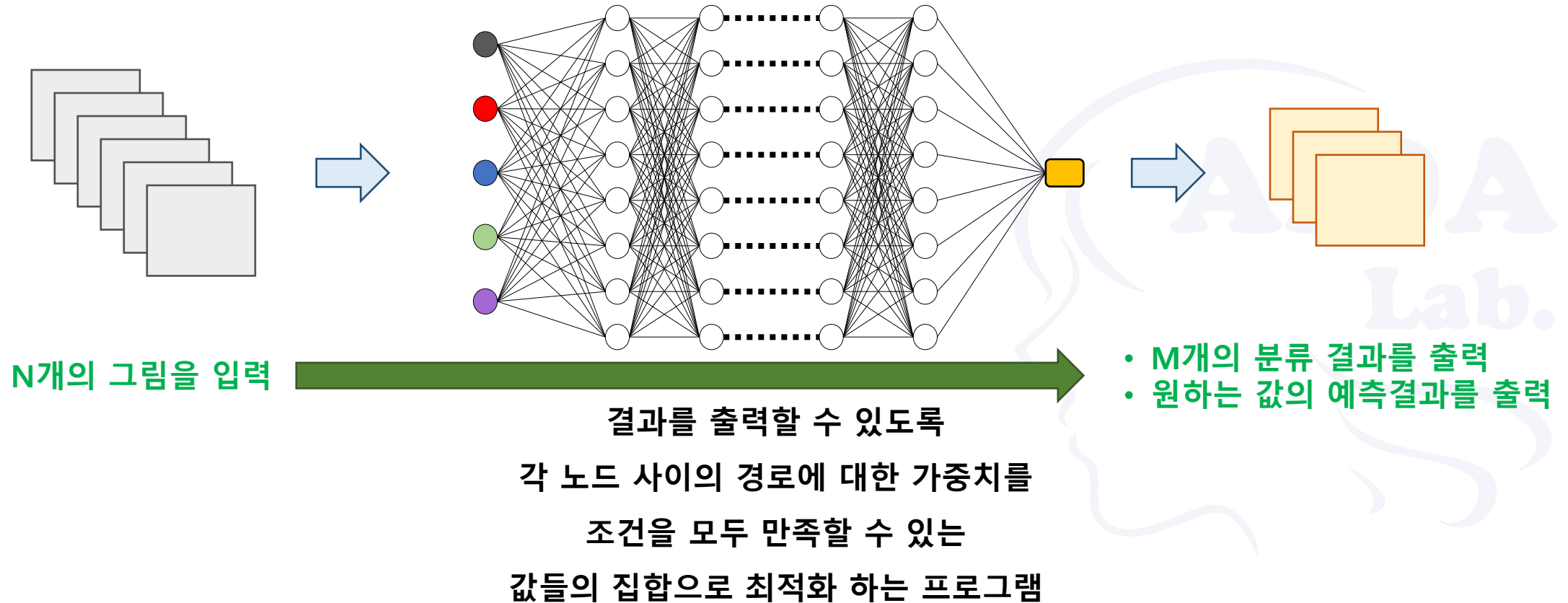


- 과적합(Over Fitting)의 해결 방안

- Drop Out

- 학습 중에 무작위로 선택한 뉴런을 0으로 설정  
→ 군데군데 망의 연결고리를 잘라 내어 대형 신경망이 소형 신경망처럼 동작하게 만듦
- 소형 신경망에서는 과적합이 거의 발생하지 않음 (표현능력이 협소하기 때문)
- 대형 신경망(딥러닝 모델)을 Drop Out을 통해 소형 신경망처럼 동작하게 하여 과적합 발생률을 떨어뜨리는 방법

- 엄밀하게 따지면 딥러닝이란...
- 다수의 노드에 적용되는 최적화 프로그램이다.



- 그렇다면 이것은 인공지능이 아니지 않나?
  - 애초에 인간의 두뇌 역시 다양한 입력의 결과를 올바르게 출력하기 위한 최적화 과정을 처리함
  - 많이 사용될 수록 굽어지고 민감해지는 각 신경세포 간의 시냅스 결합 강도를
  - 뉴런 사이의 가중치로 모델링하여 구현한 것일 뿐!!

