

The background of the slide is a vibrant blue digital theme. It features a glowing sphere on the right side, composed of binary code (0s and 1s) that recedes into the distance, creating a sense of depth. A bright light source behind the sphere creates a lens flare effect. The overall aesthetic is high-tech and futuristic.

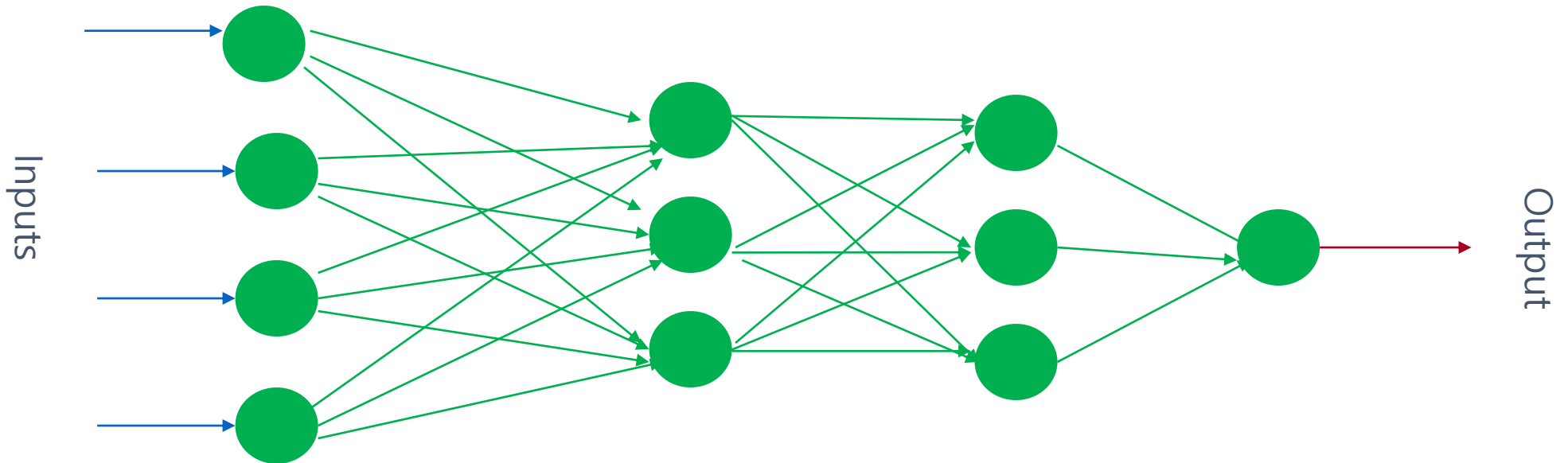
DL모형 리뷰

2022

Part I. Artificial Neural Network

I. 개요 및 현황

- Artificial Neural Network(ANN, 인공신경망)
 - 데이터를 입력받아 변환하여 원하는 결과로 출력: 각각 Input Layer, Hidden Layer, Output Layer로 구성
 - 예측 성능이 우수
 - 모형을 통한 추론은 어려움

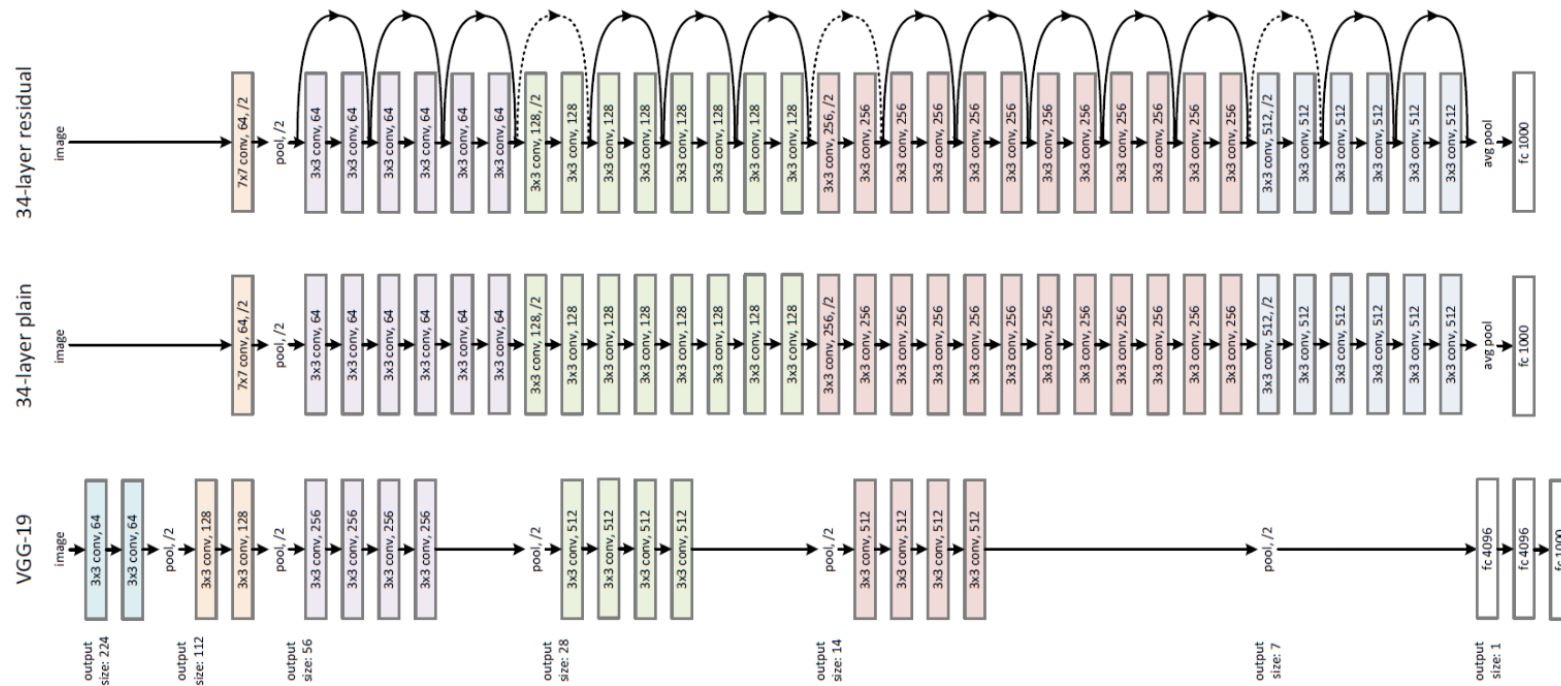


I. 개요 및 현황

인공신경망+깊이

- Hidden layer가 2개 이상인 NN(Neural Network)을 Deep Neural Network(DNN)
- DNN에서의 학습을 딥러닝이라고 부름

비선형 변환과정들

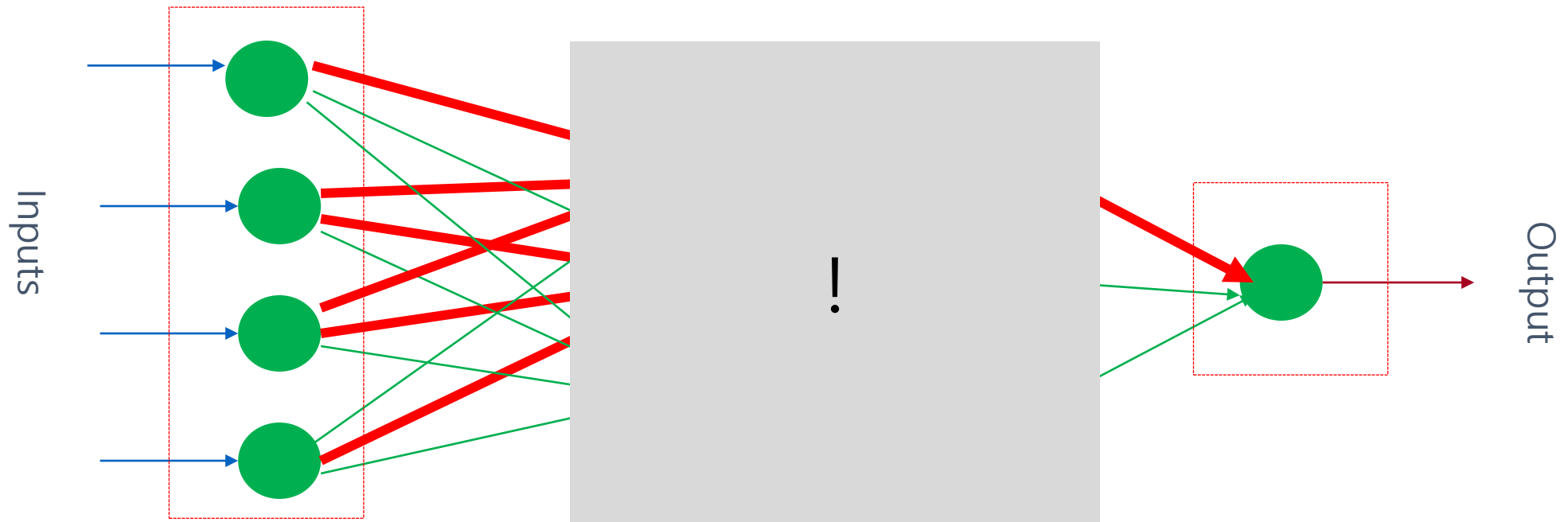


RESNET

I. 개요 및 현황

- 딥러닝의 특성

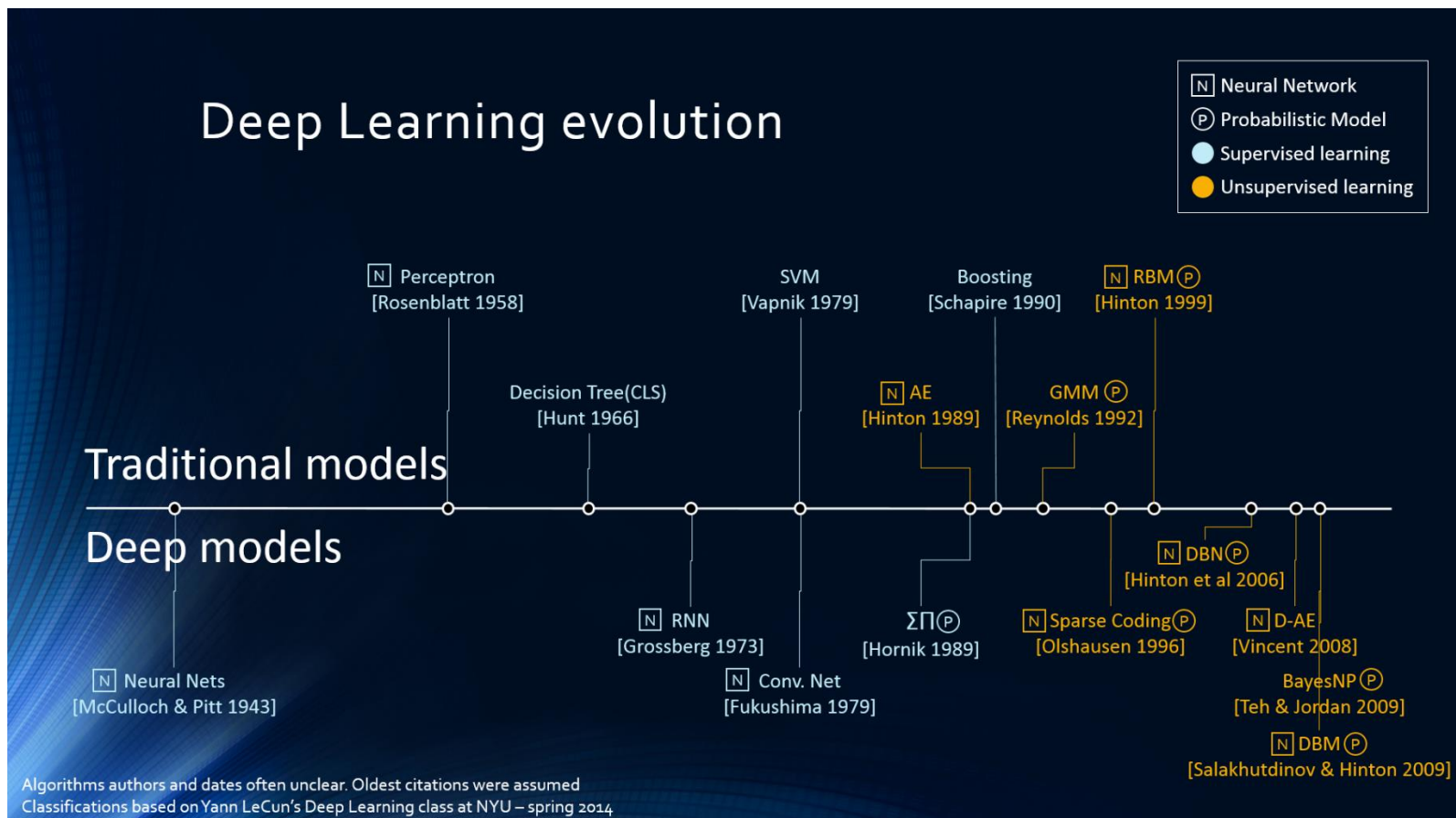
- 다수의 은닉층 = 계산량



I. 개요 및 현황

• 딥러닝 알고리즘의 발전

- 기존 ANN의 난제들을 해결
- CNN, RNN, LSTM, Transformer, BERT 등으로 발전 중



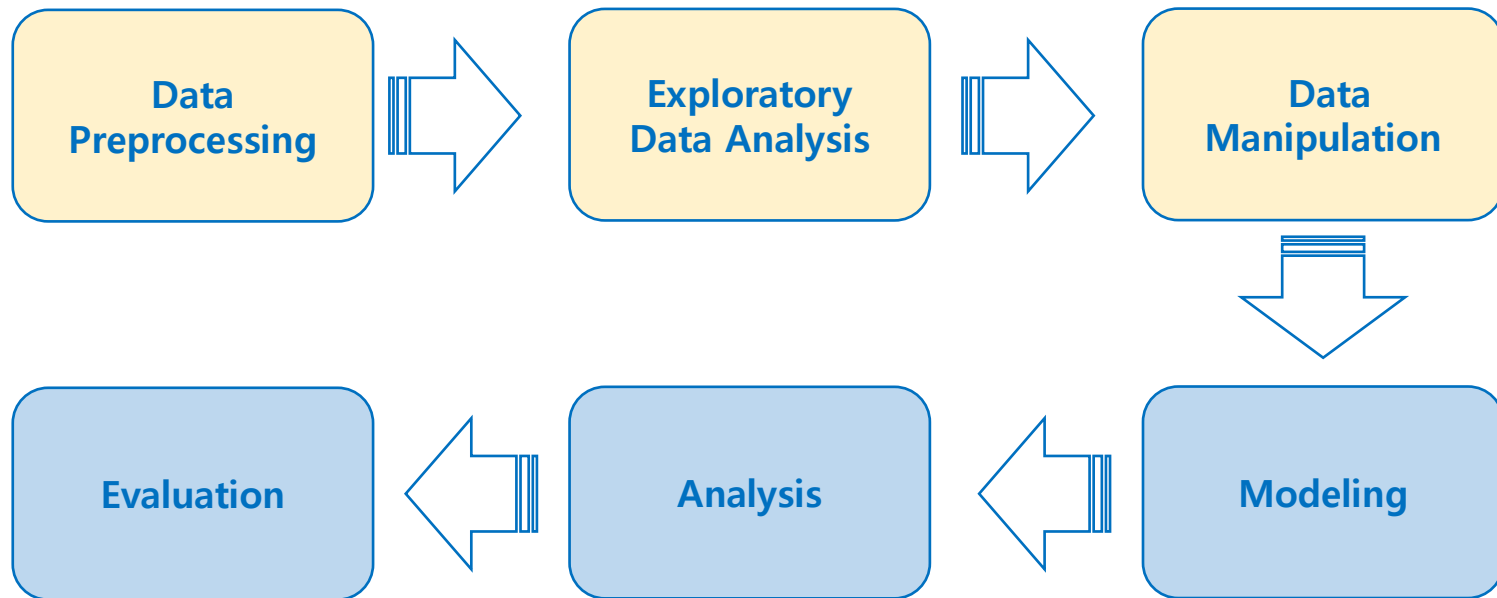
II. Deep VS Shallow Learning

변수

:관측된 개체들이 갖는 특성 또는 속성으로, 다양한 값을 갖을 수 있음, Feature라고도 지칭

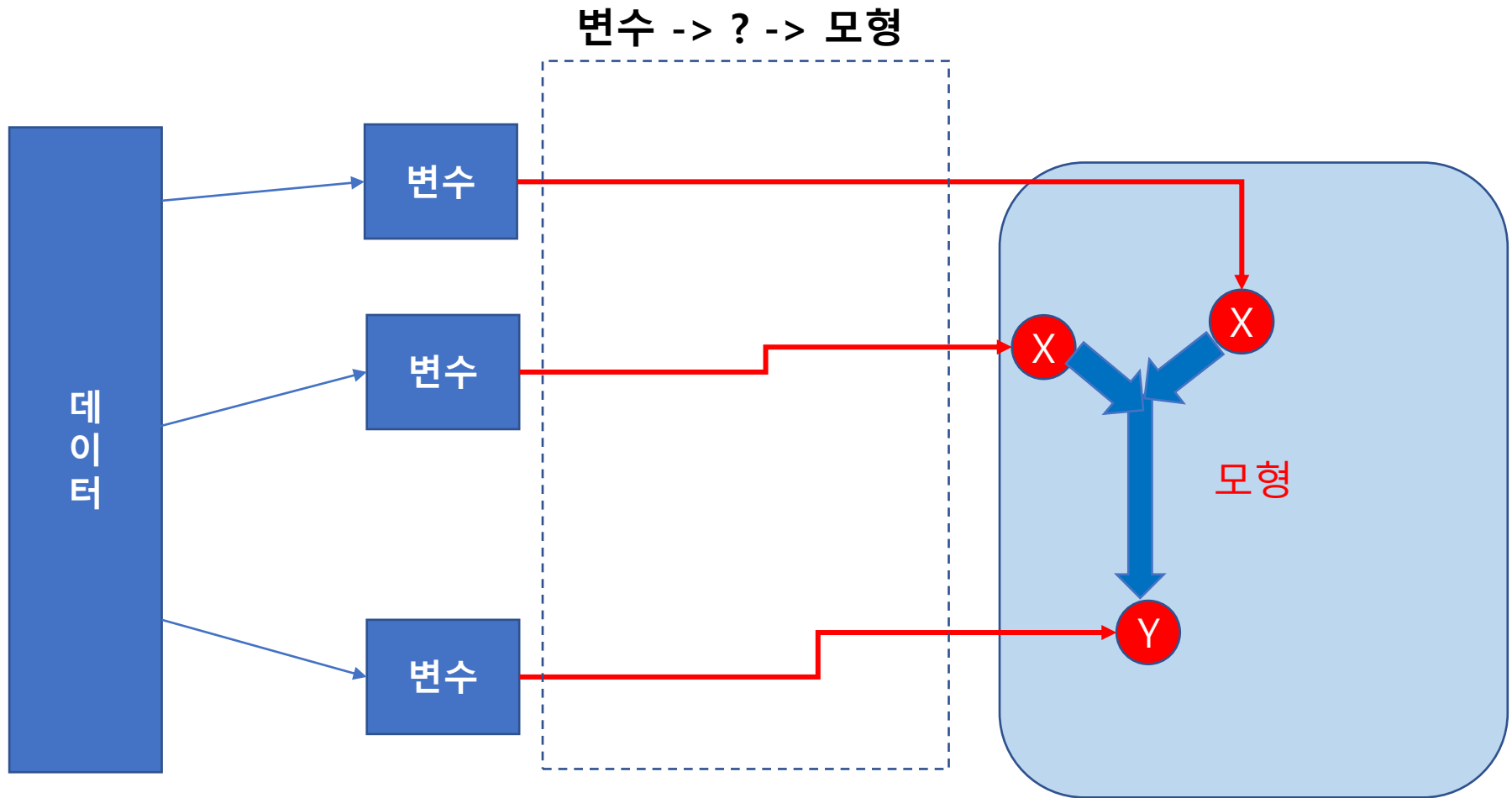
모형

: 분석 목적에 맞게 데이터에서 인사이트를 찾아내는 적절한 기법이나 알고리즘



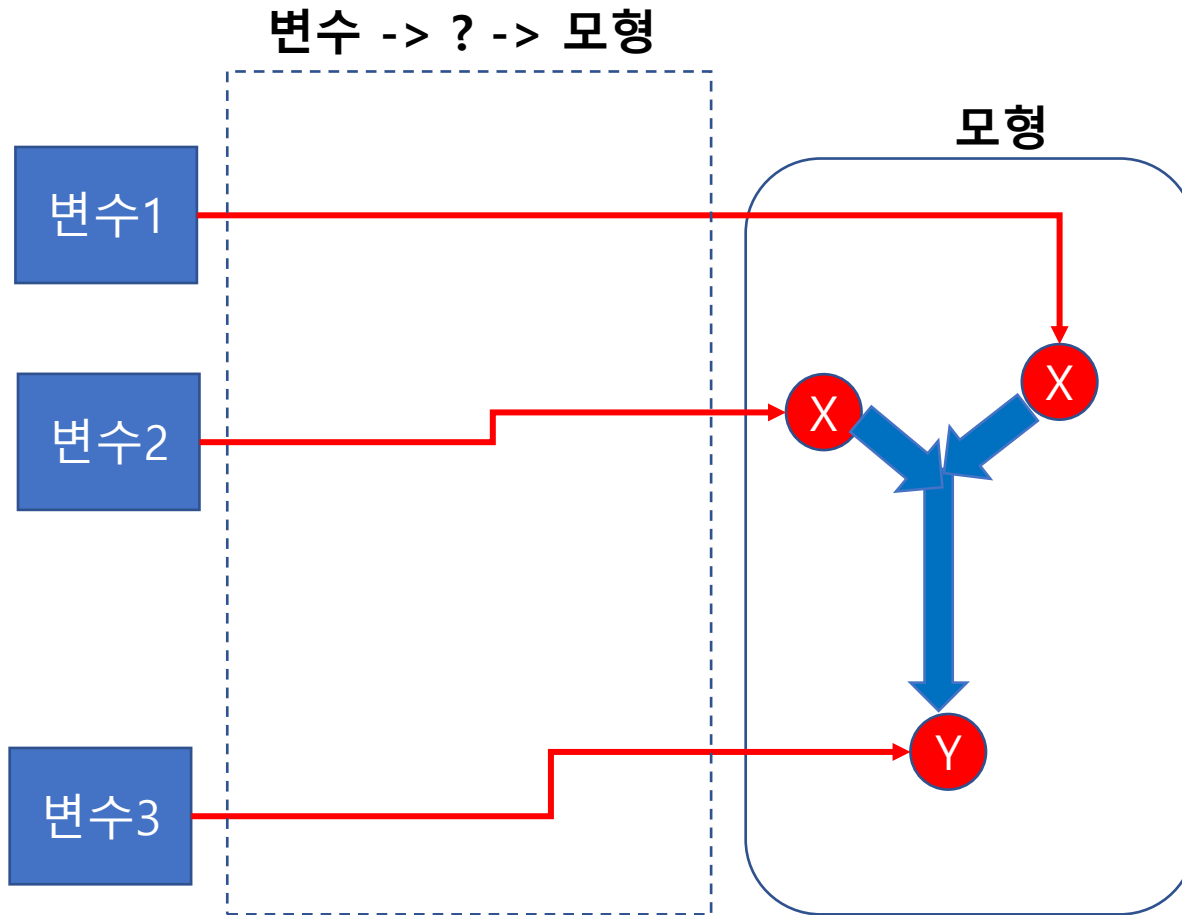
II. Deep VS Shallow Learning

변수에서 모형으로



II. Deep VS Shallow Learning

Shallow Learning?

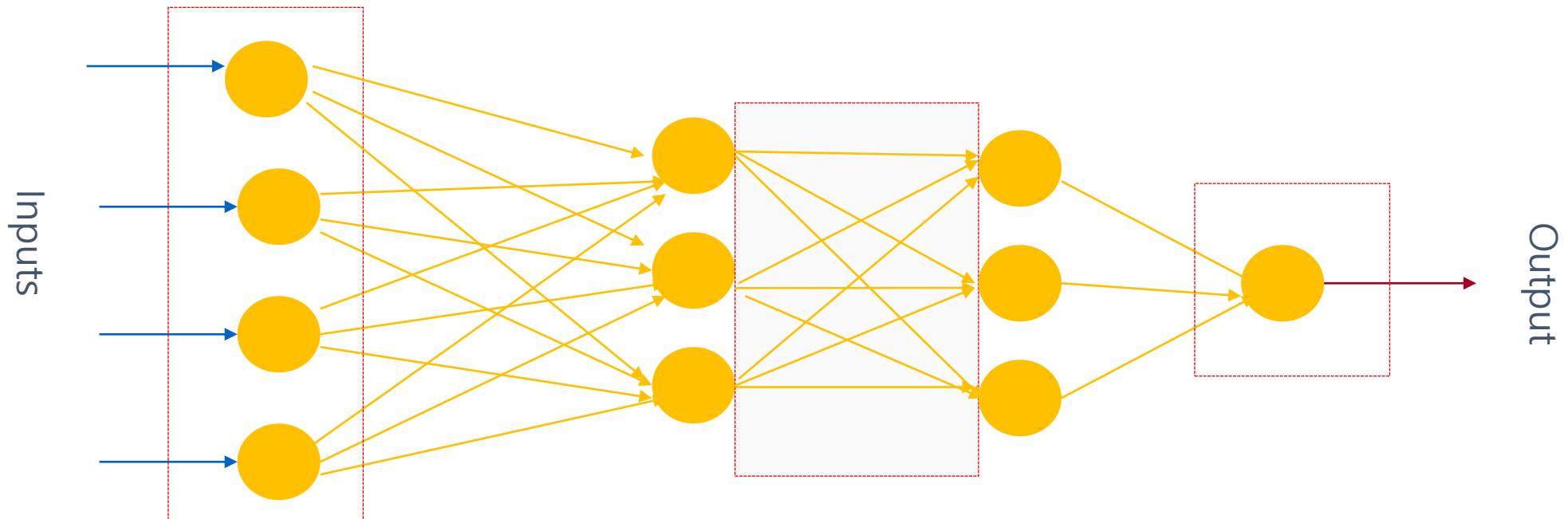


Shallow Learning:
주어진 변수를 별도로 다수의 비
선형 변환없이 그대로 사용해서
모델링

III. 인공신경망 구성

- **Artificial Neural Network(ANN, 인공신경망)**

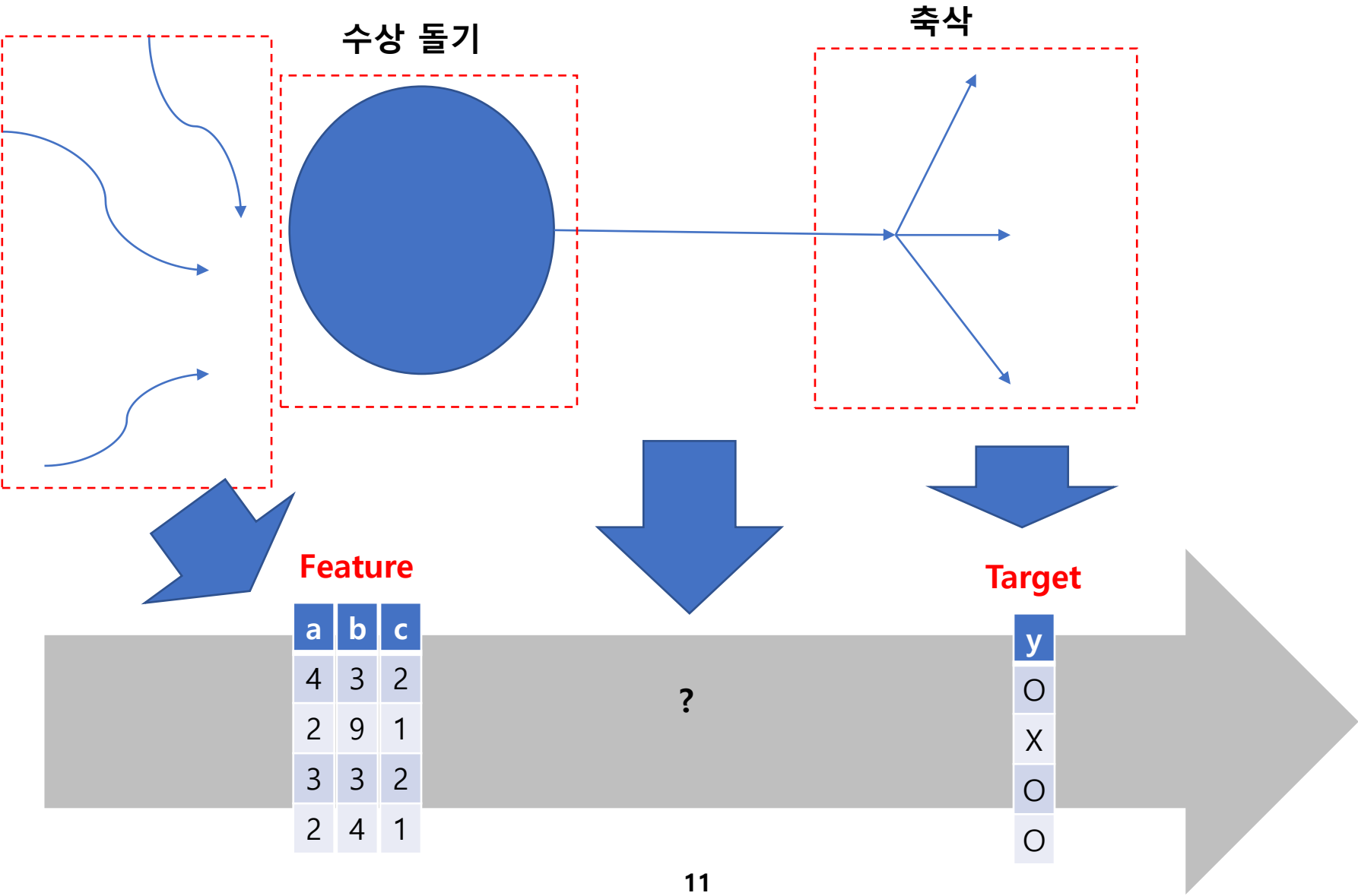
- 생물학의 신경망(동물의 중추신경계중 특히 뇌의 뉴런)을 모사한 학습 알고리즘
- 뉴런을 모방한 노드들이 각각 Input Layer, Hidden Layer, Output Layer로 구분되며 데이터를 입력받아 변환하여 원하는 결과로 출력하는 네트워크를 구축
- 예측 성능이 우수하다고 알려진 반면, 모델을 직관적으로 이해하기가 어려움



노드 간의 연결

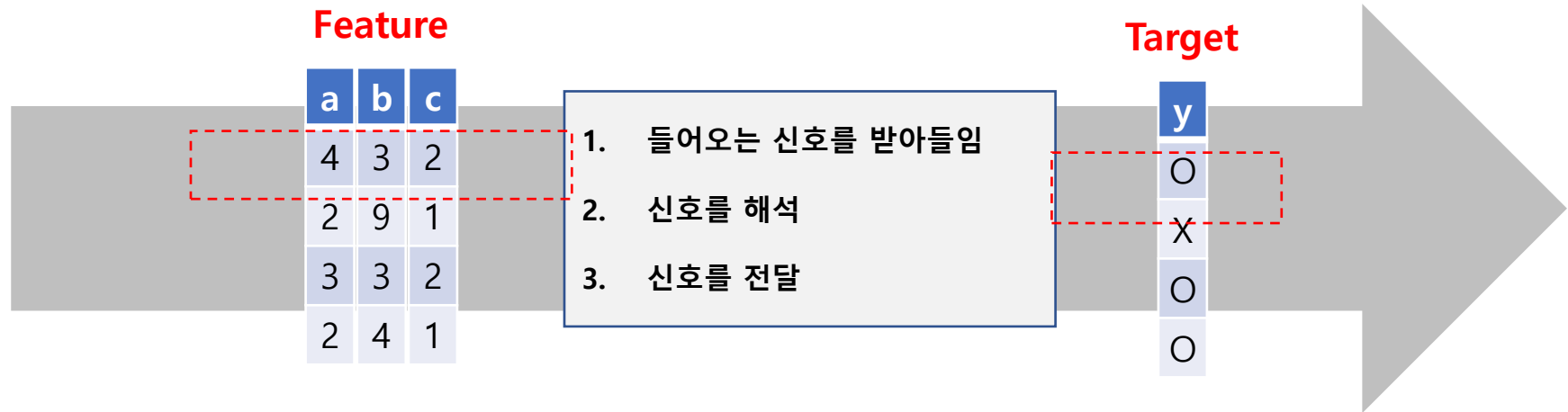
III. 인공신경망 구성

뉴런?



III. 인공신경망 구성

퍼셉트론: 신경세포와 같이 들어오는 신호를 바탕으로, Target을 계산



4, 3, 2



O
범주

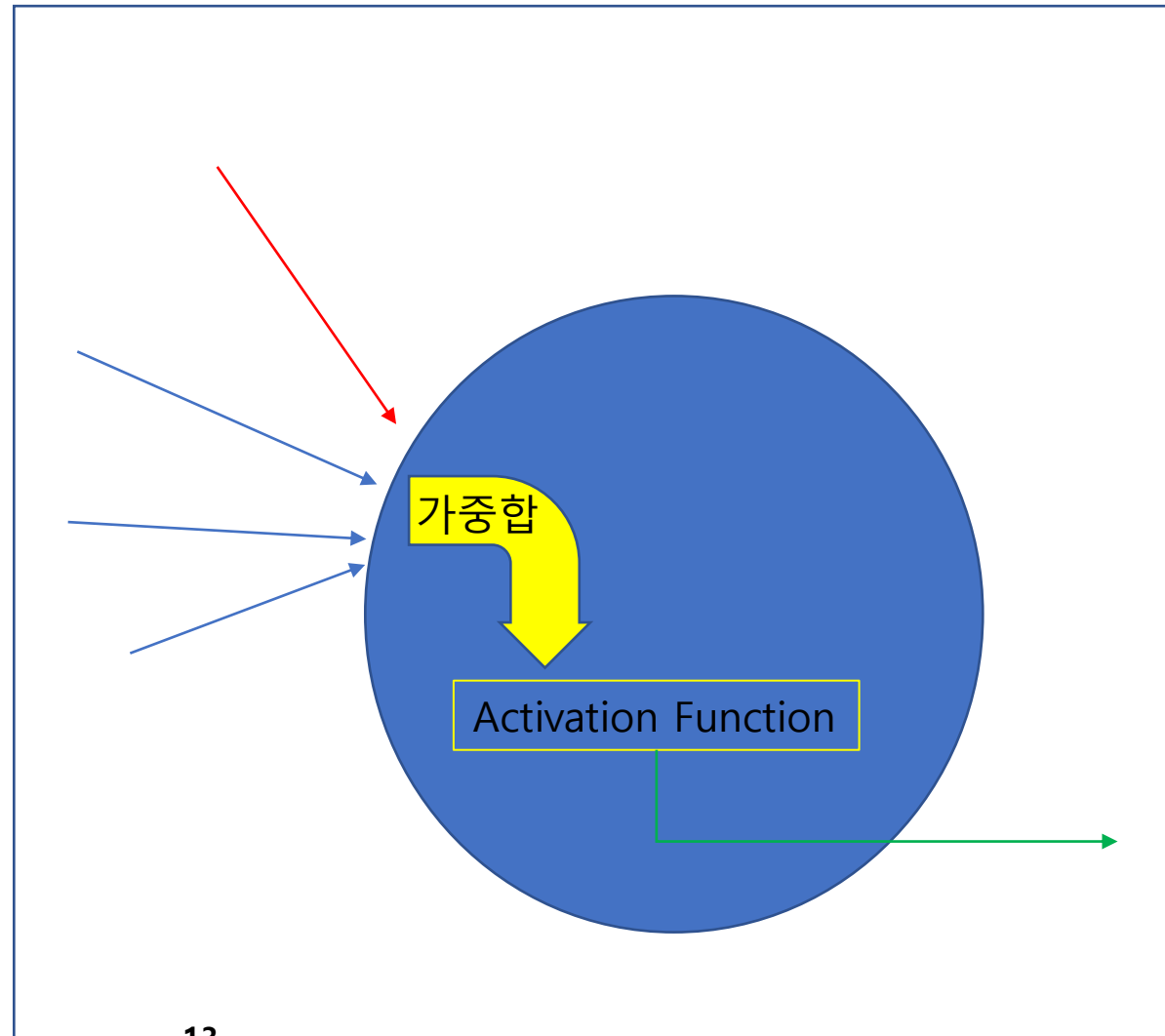
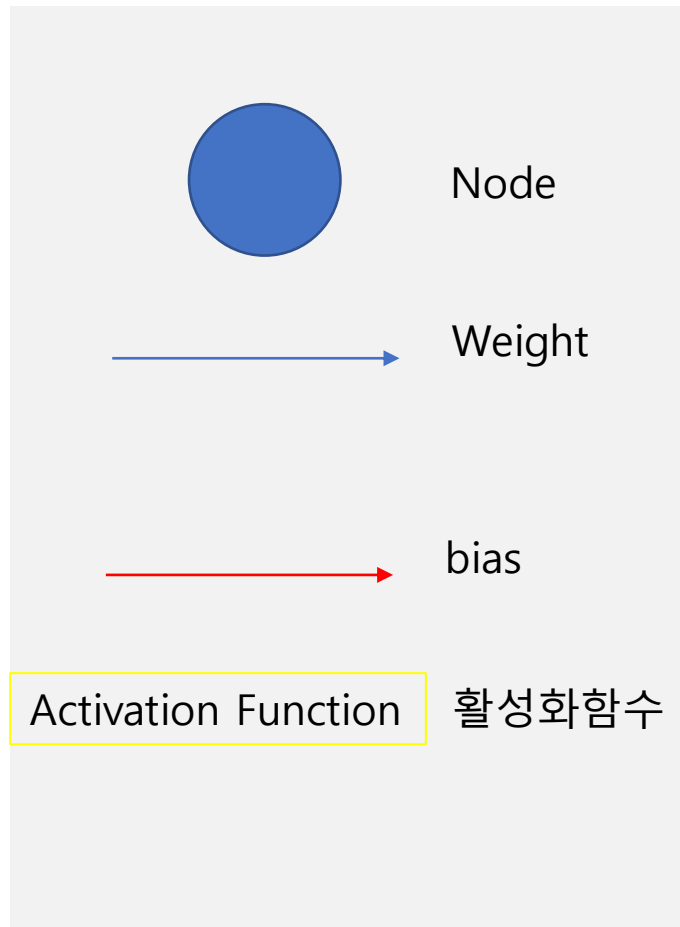
$$(4 \times ?) + (3 \times ?) + (2 \times ?)$$



만약 이 값이 어떤 수 보다 크면 O, 아니면 X

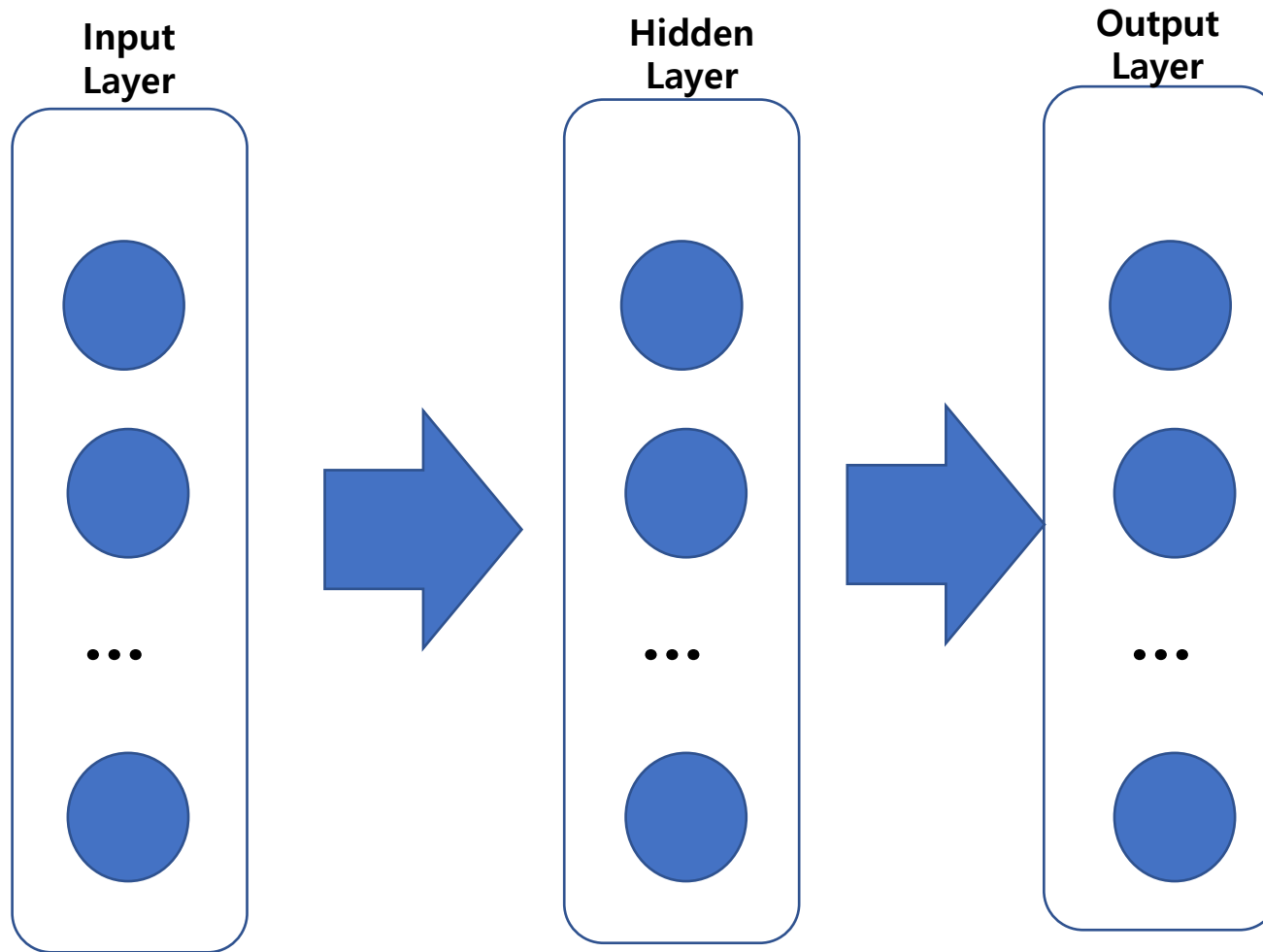
III. 인공신경망 구성

인공신경망 구성 요소



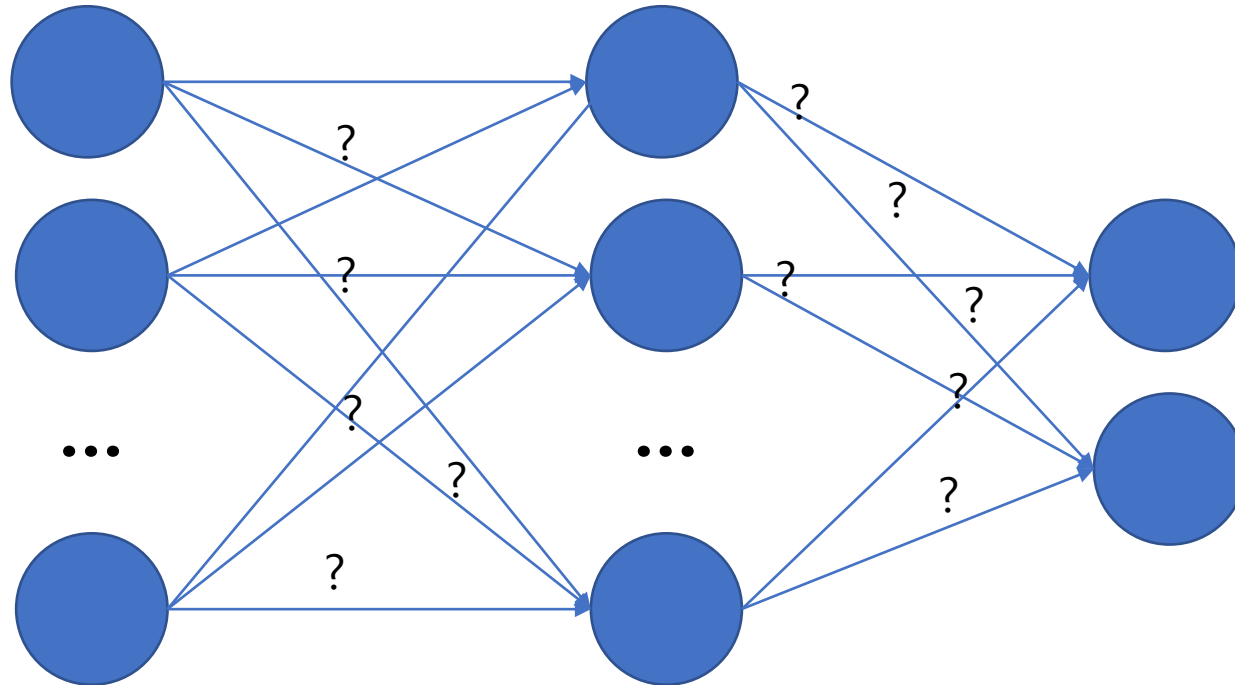
III. 인공신경망 구성

인공신경망 구성 요소



III. 인공신경망 구성

인공신경망: 가중치의 발견

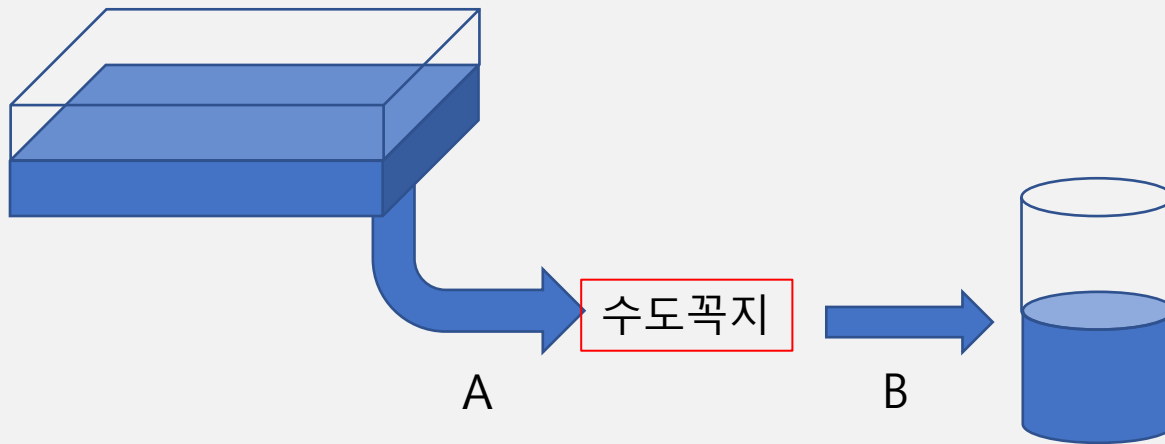


IV. 가중치와 행렬표현

가중치는?

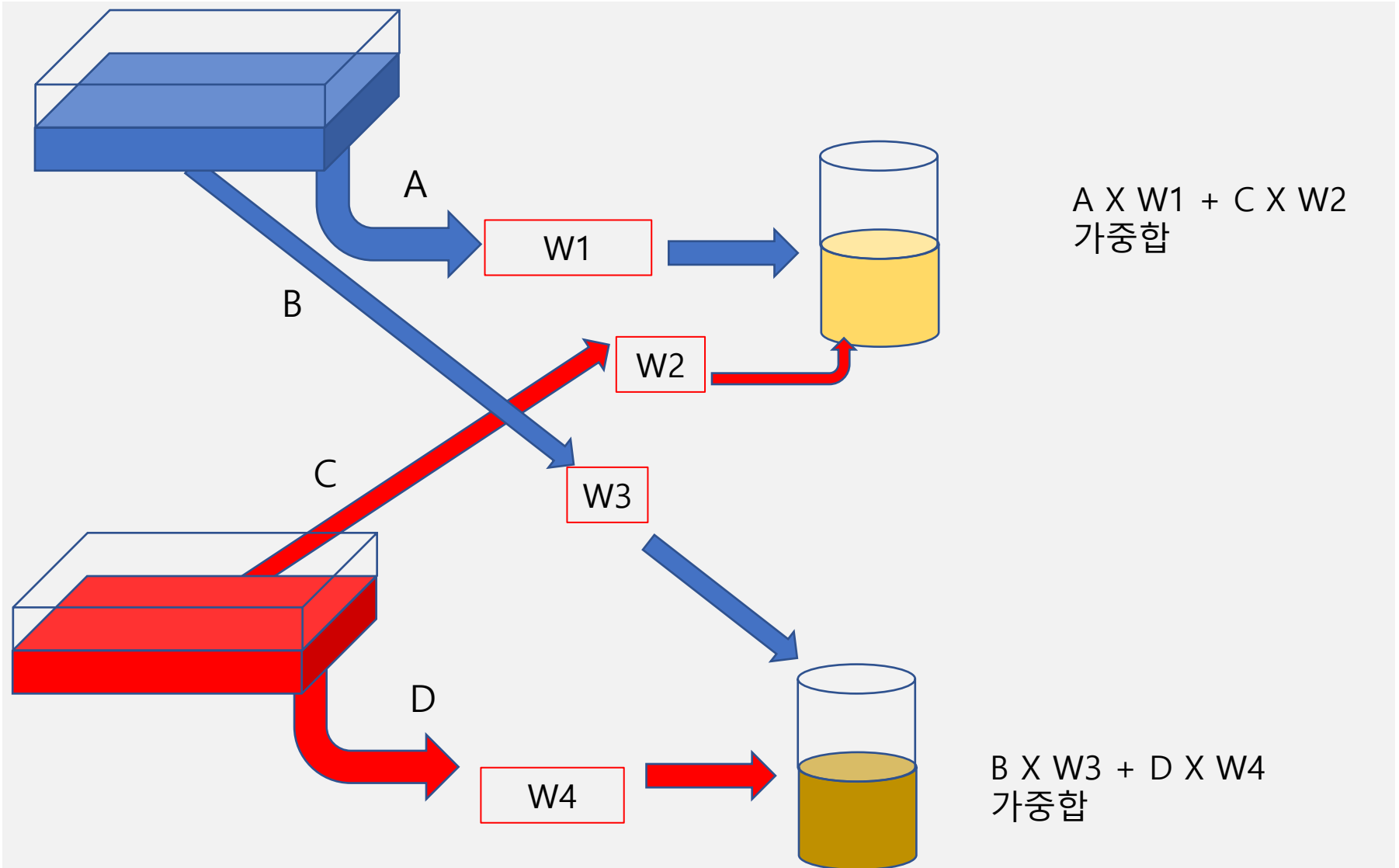
Weight, 가중치

$B = A \times \text{"수도꼭지를 통해 흘려보내는 정도"}$



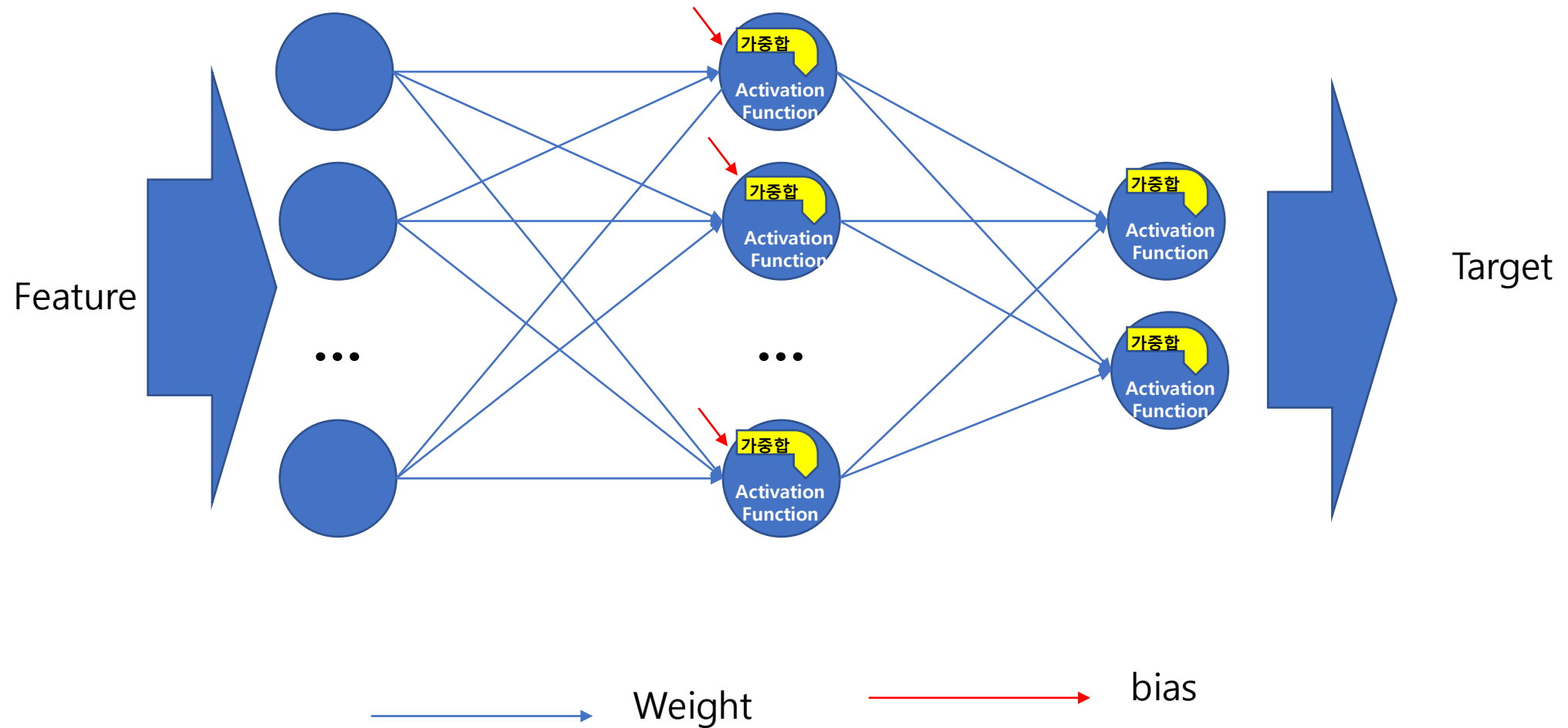
IV. 가중치와 행렬표현

가중치는?



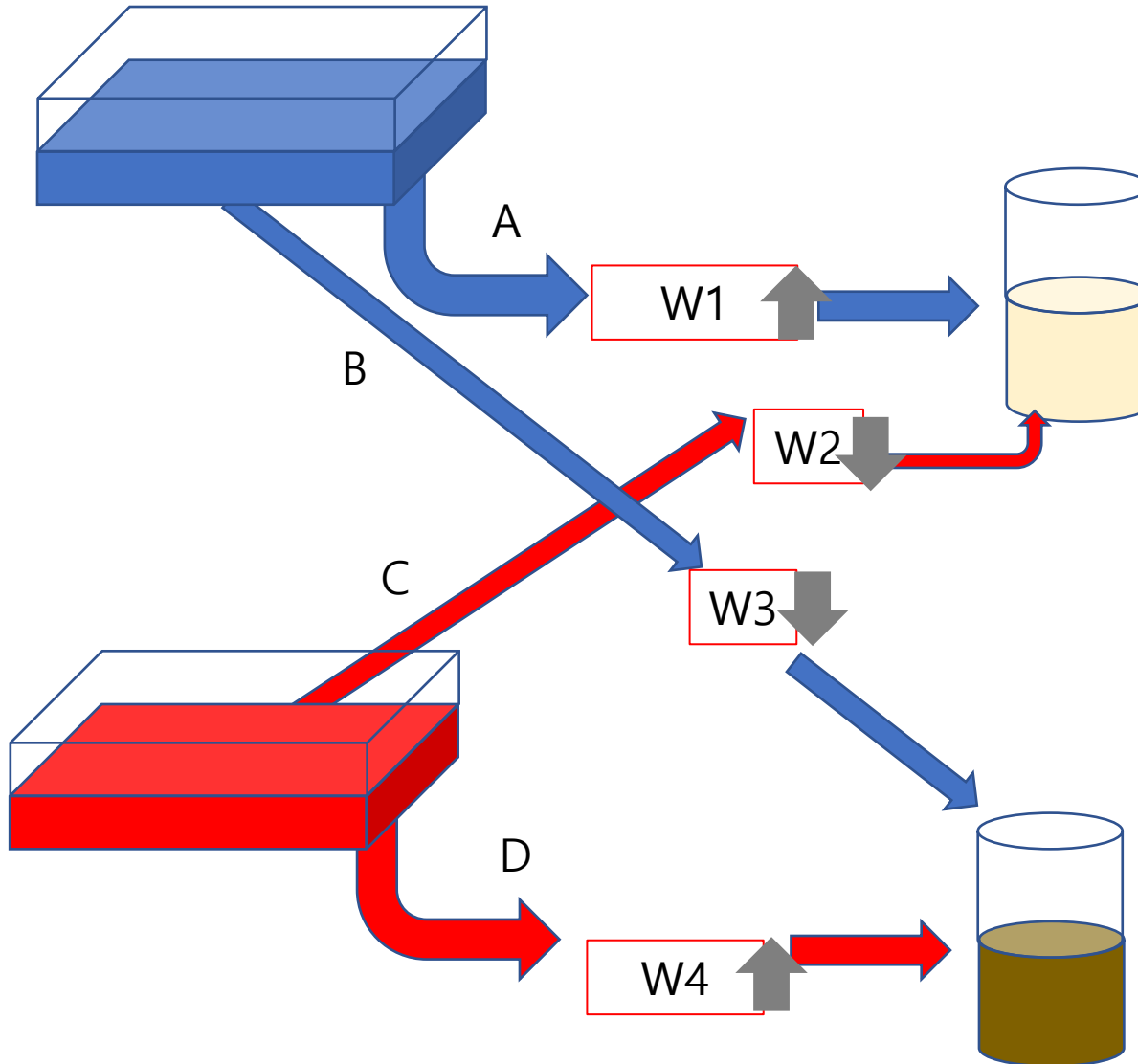
IV. 가중치와 행렬표현

인공신경망과 가중치



IV. 가중치와 행렬표현

가중치 조정!



원하는 색상이 아닌 경우,
"수도꼭지"를 조절!

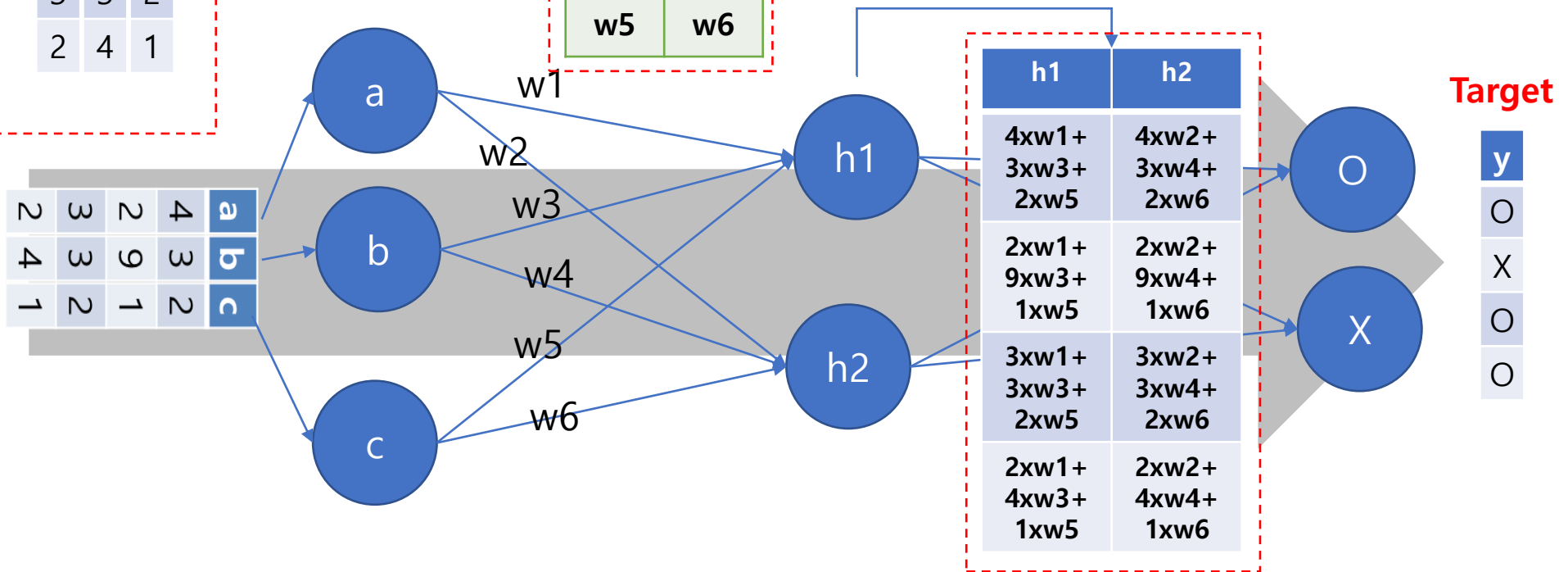
IV. 가중치와 행렬표현

인공신경망을 행렬로 표현하기!

Feature

a	b	c
4	3	2
2	9	1
3	3	2
2	4	1

w1	w2
w3	w4
w5	w6



IV. 가중치와 행렬표현

인공신경망의 가중합

Feature

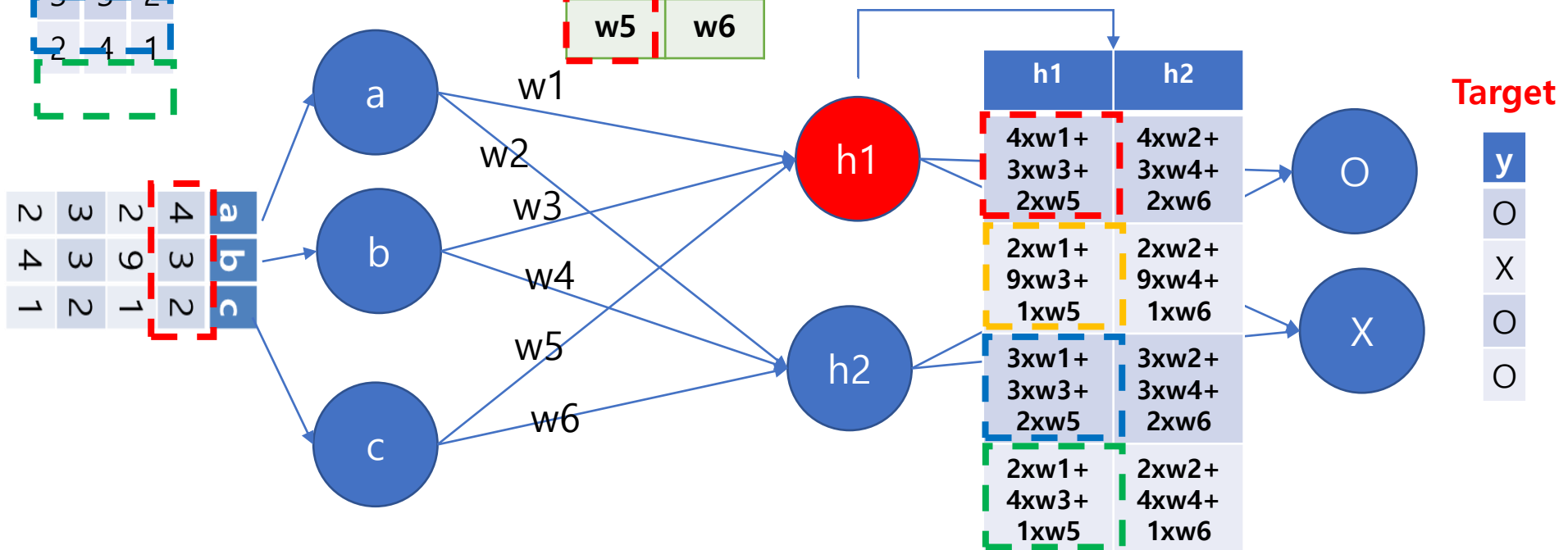
a	b	c
4	3	2
2	9	1
3	3	2
2	4	1

2	3	2	4	a
4	3	9	3	b
1	2	1	2	c

인공신경망 가중합 다시보기: 인공신경망의 가중합이란
행렬곱 결과의 한 원소에 대한 계산과 동일

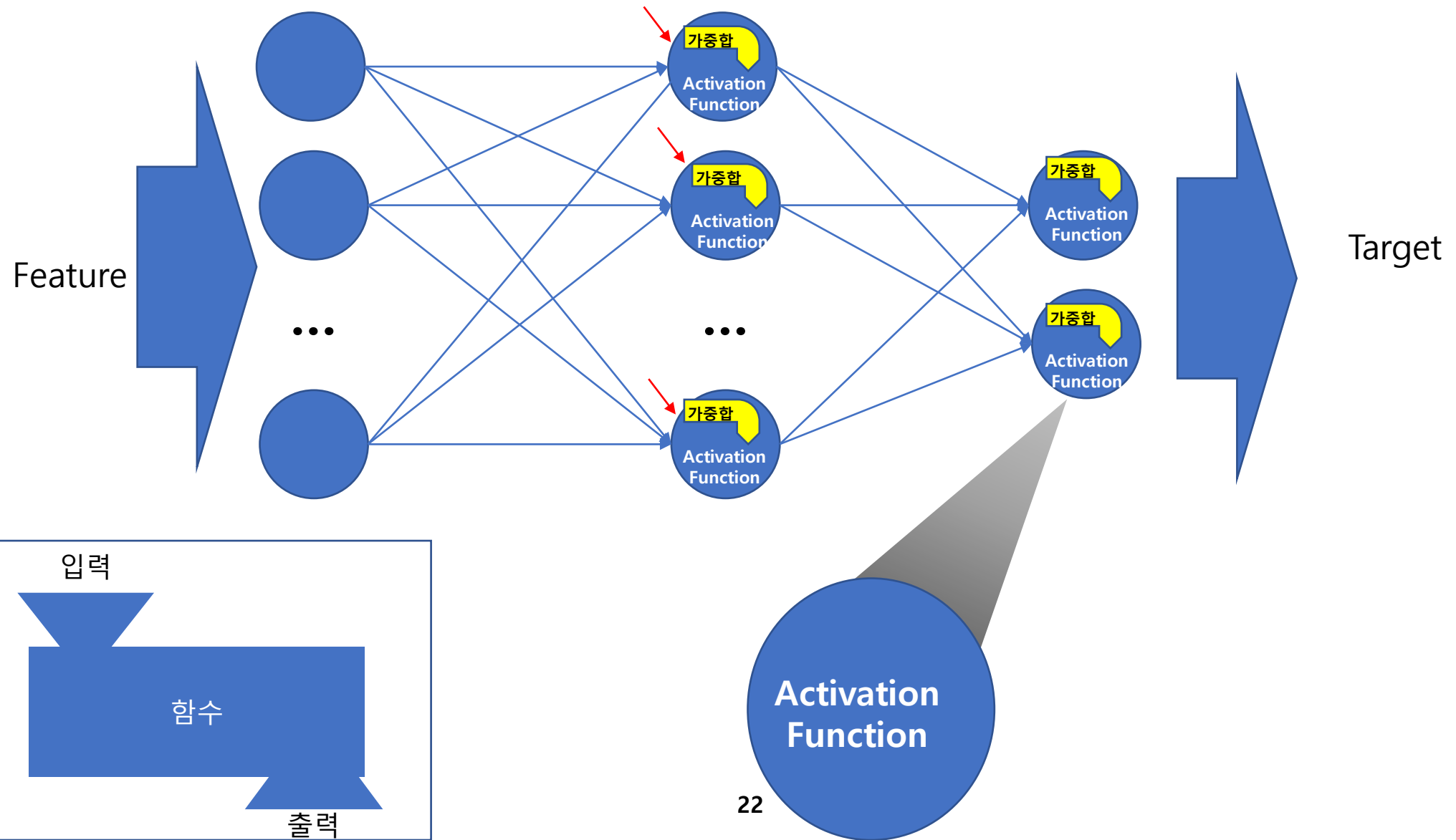
w1	w2
w3	w4
w5	w6

$$4 \times w1 + 3 \times w3 + 2 \times w5$$



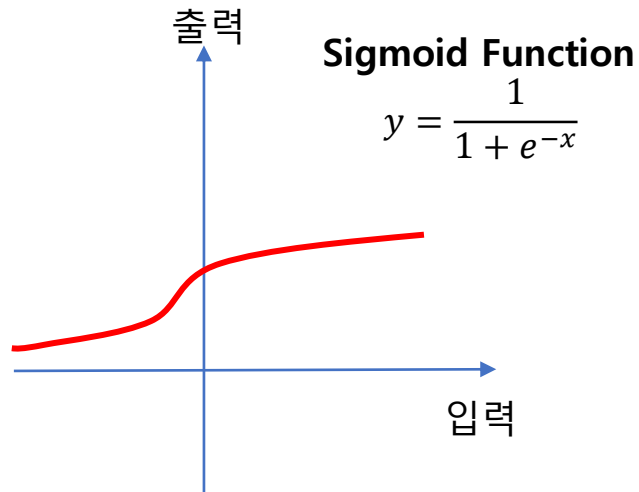
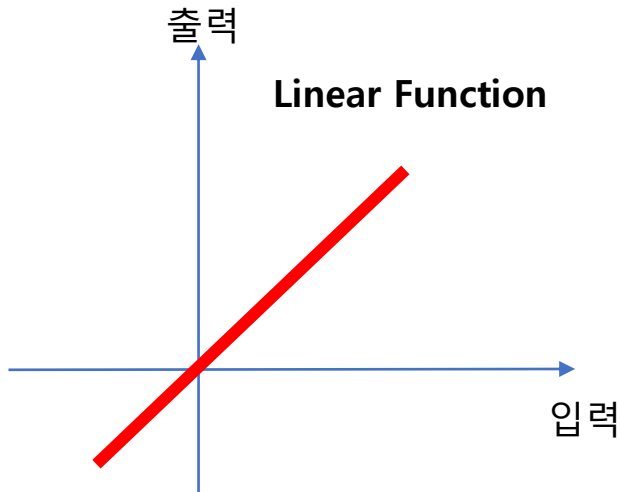
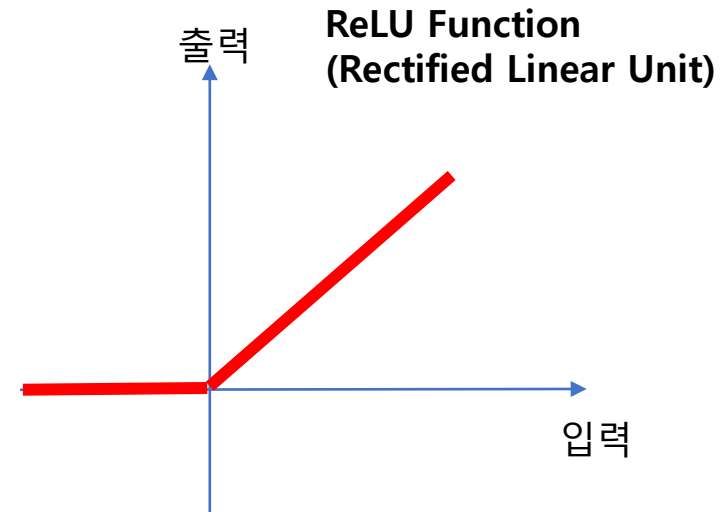
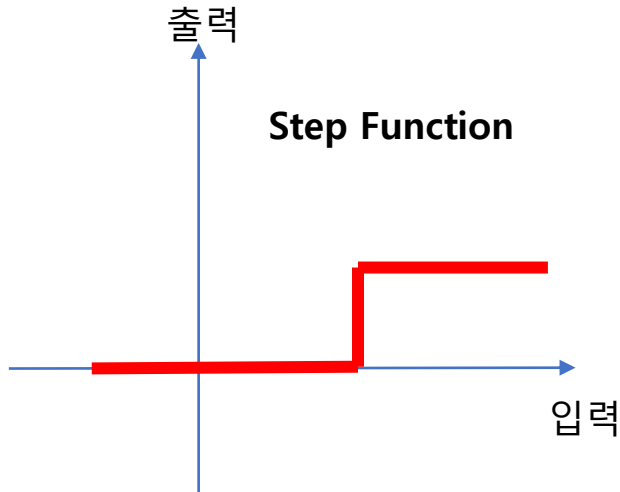
V. 활성화 함수와 가중치 업데이트

활성화 함수



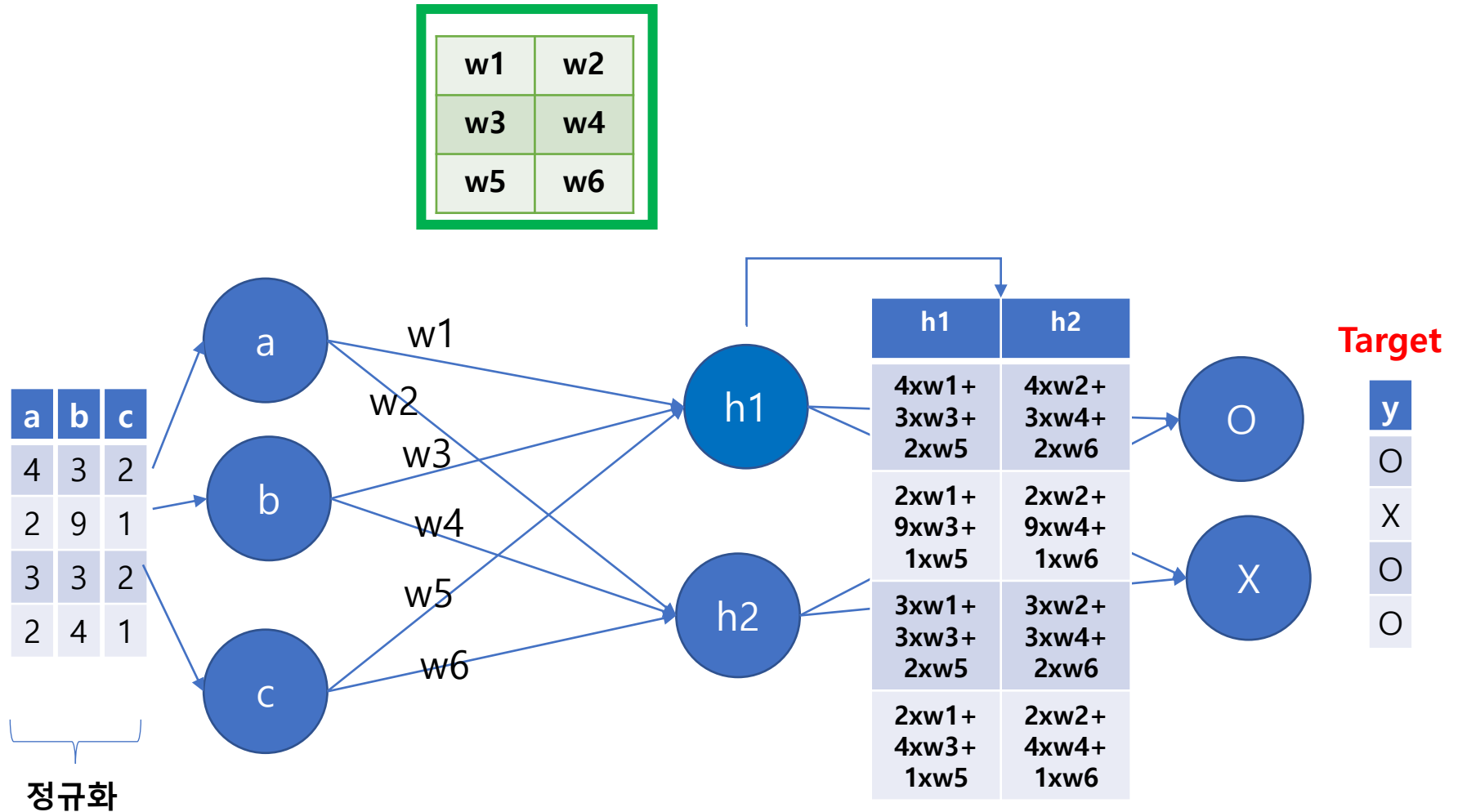
V. 활성화 함수와 가중치 업데이트

다양한 활성화 함수



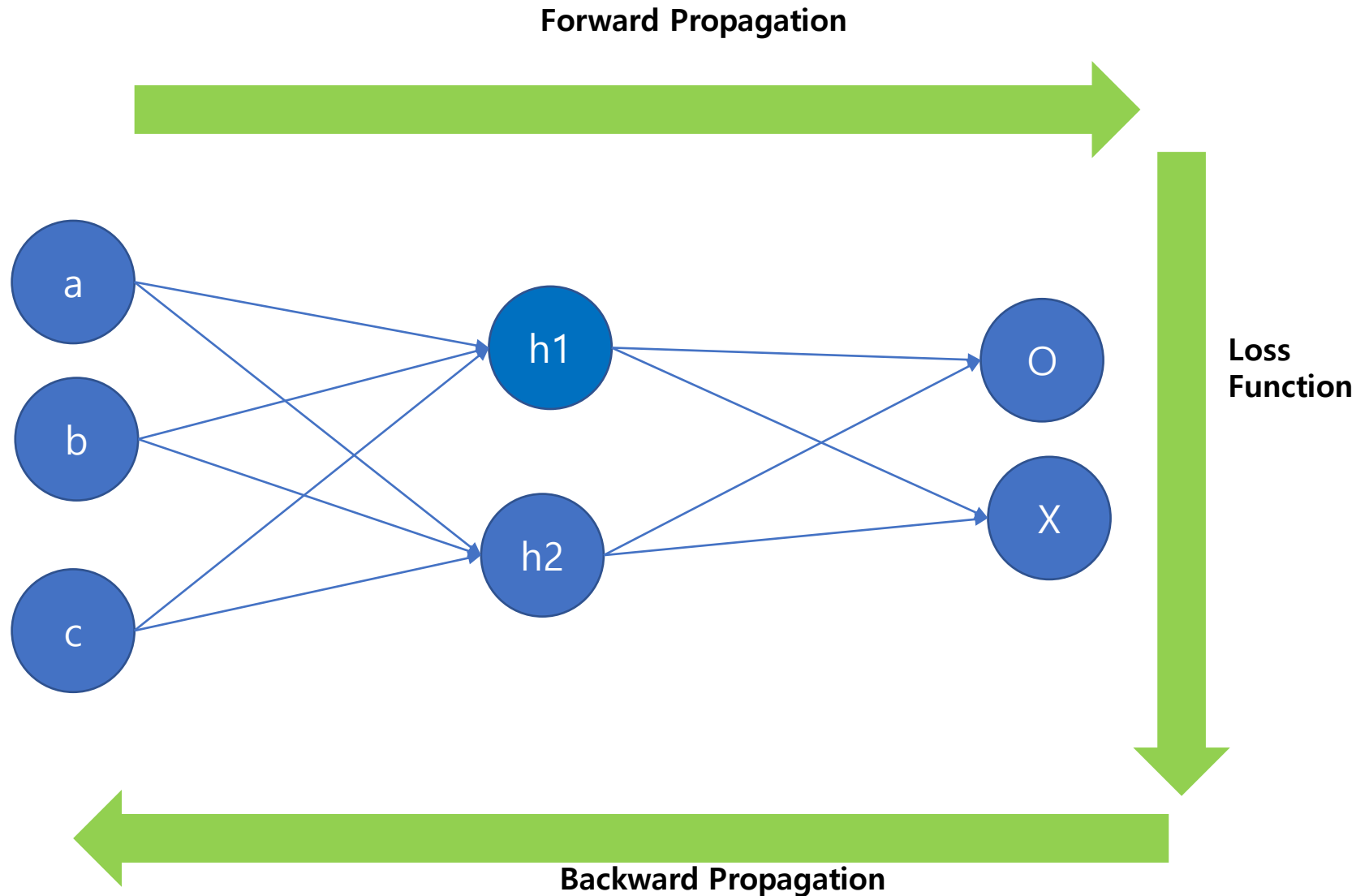
V. 활성화함수와 가중치 업데이트

가중치: 처음엔 Random (0~1)



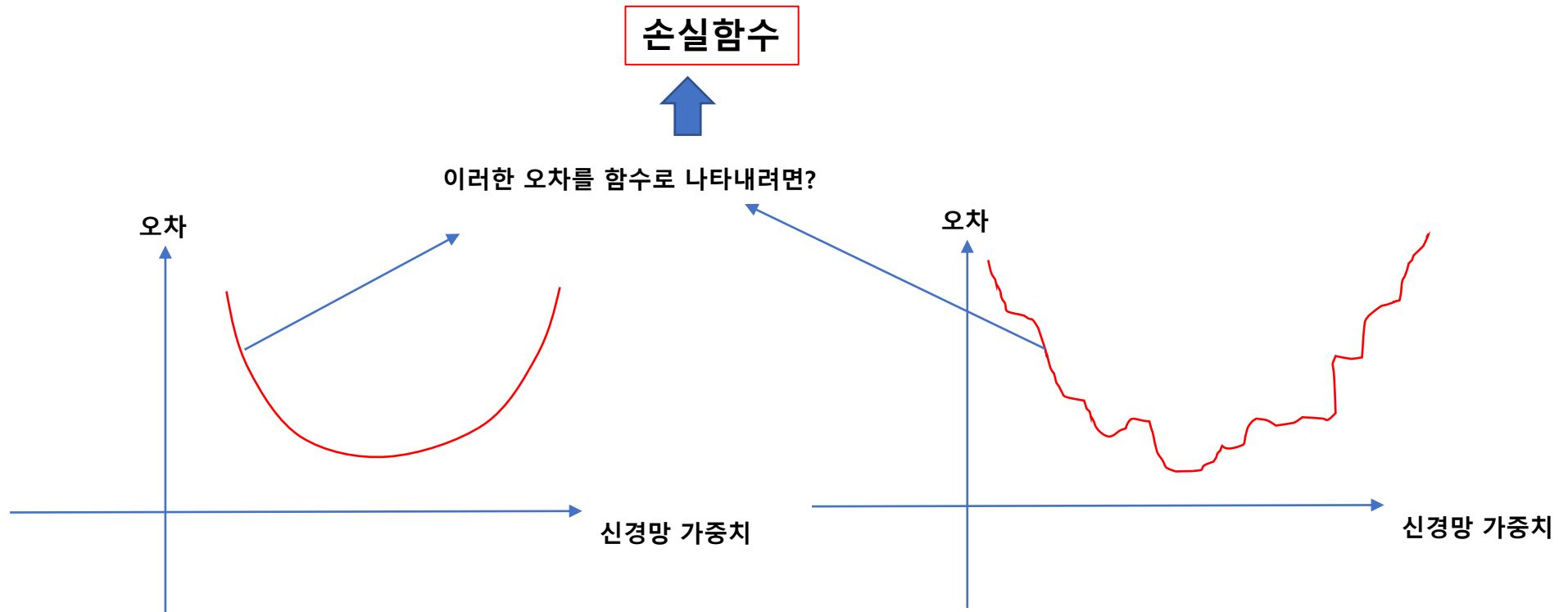
V. 활성화함수와 가중치 업데이트

Epoch(에포크): Forward Propagation + Back Propagation



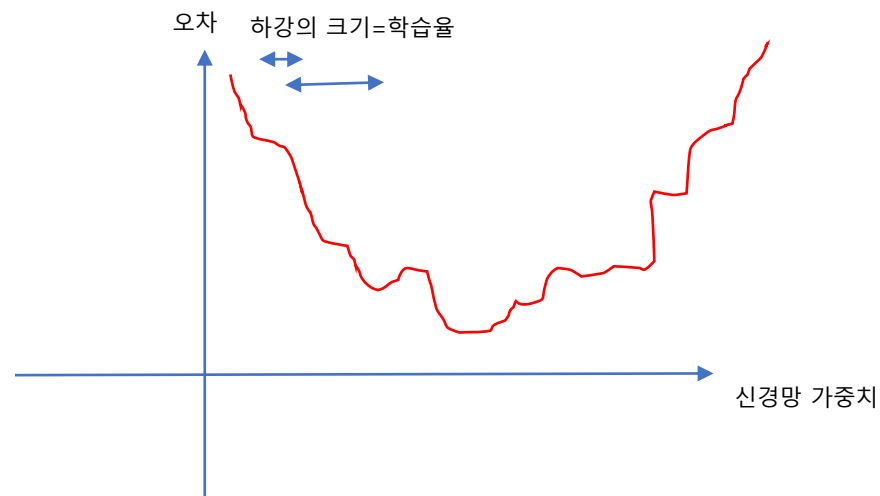
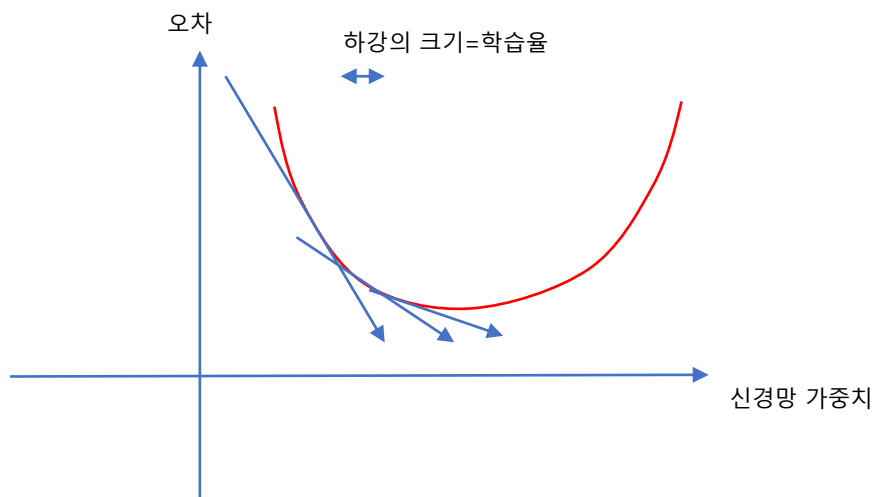
VI. 역전파 알고리즘

오차에 대한 함수



VI. 역전파 알고리즘

- 신경망 가중치가 입력이고 오차가 출력인 함수 대상
- 경사하강법: 오차의 최소화
- 학습율: 하강의 크기



VI. 역전파 알고리즘

학습률(Learning Rate)

가중치 조정과 학습률

$$\text{새로운 } w_{jk} = \text{이전 } w_{jk} - \alpha \times \frac{\partial E}{\partial w_{jk}}$$

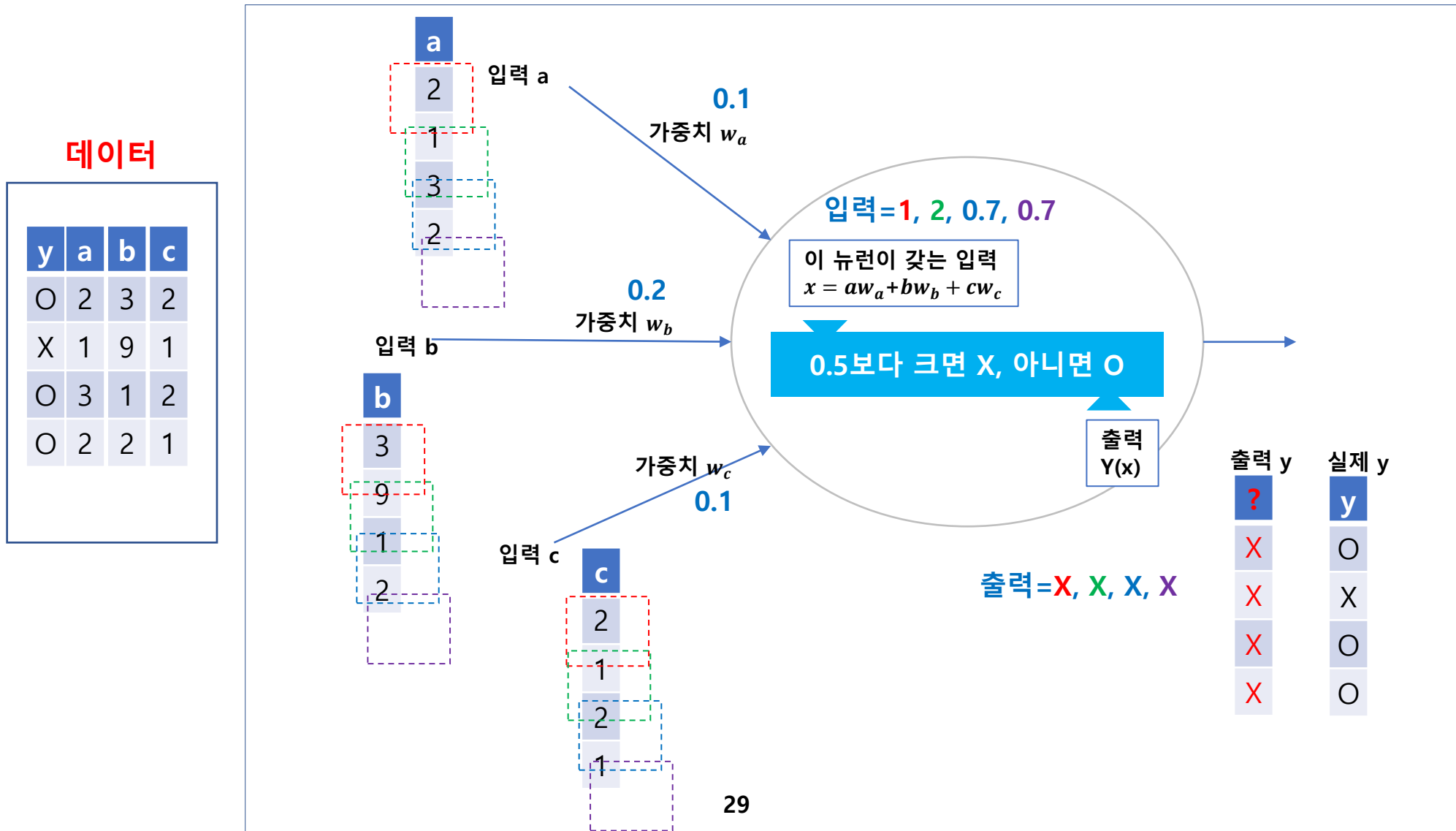


- 이전 가중치에서 오차의 변화율을 빼주기
 - 양의 기울기(오차 증가시킴)이면 이전 가중치에서 빼주어 영향을 덜 받게 하고, 음의 기울기(오차 감소)이면 이전 가중치를 더 크게 해주는 효과(-의-)
- α 는 가중치 변화하는 정도를 조정:학습률

VI. 역전파 알고리즘

ANN Step by Step

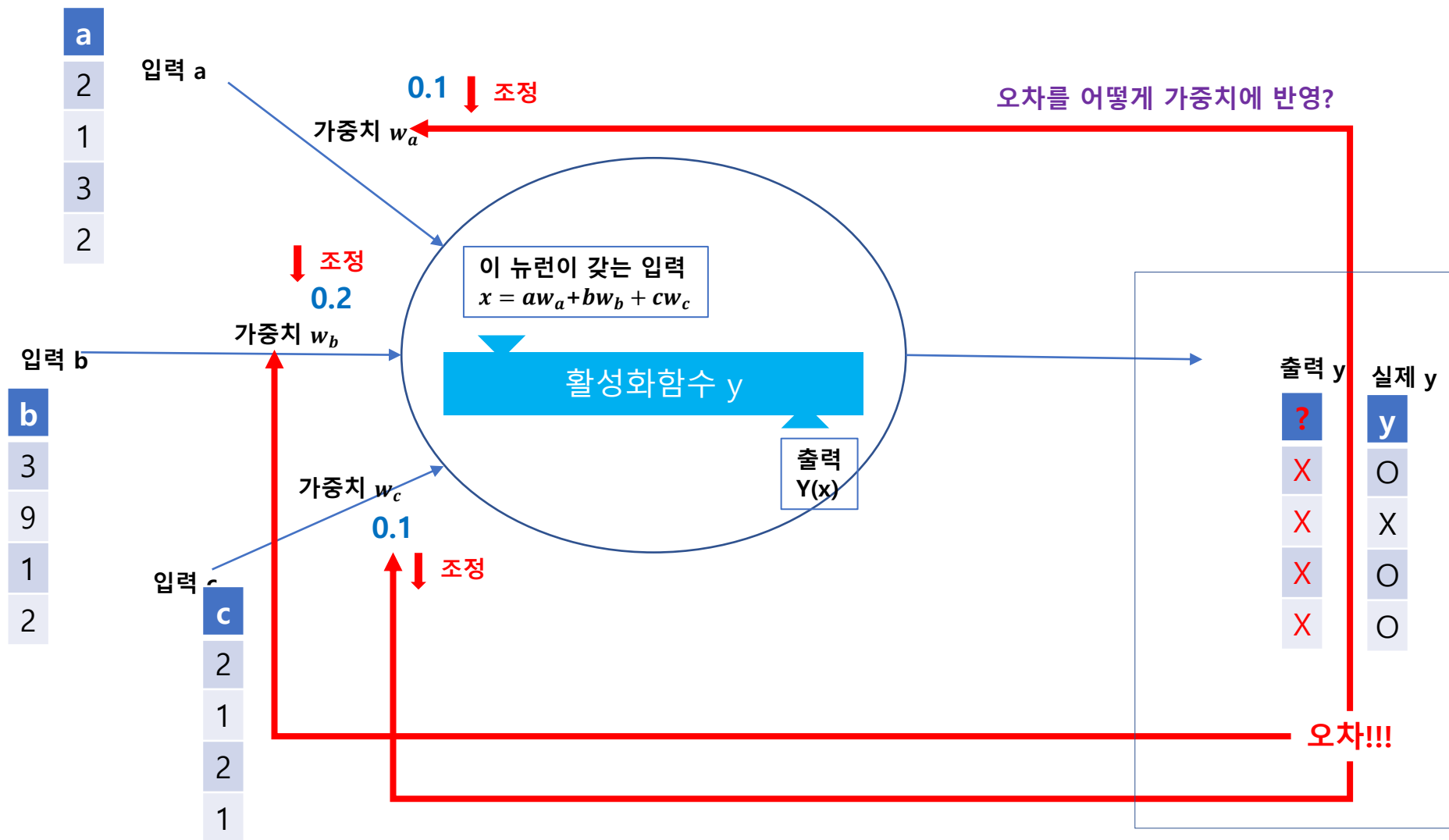
인공신경망 1단계: 전파!



VI. 역전파 알고리즘

ANN Step by Step

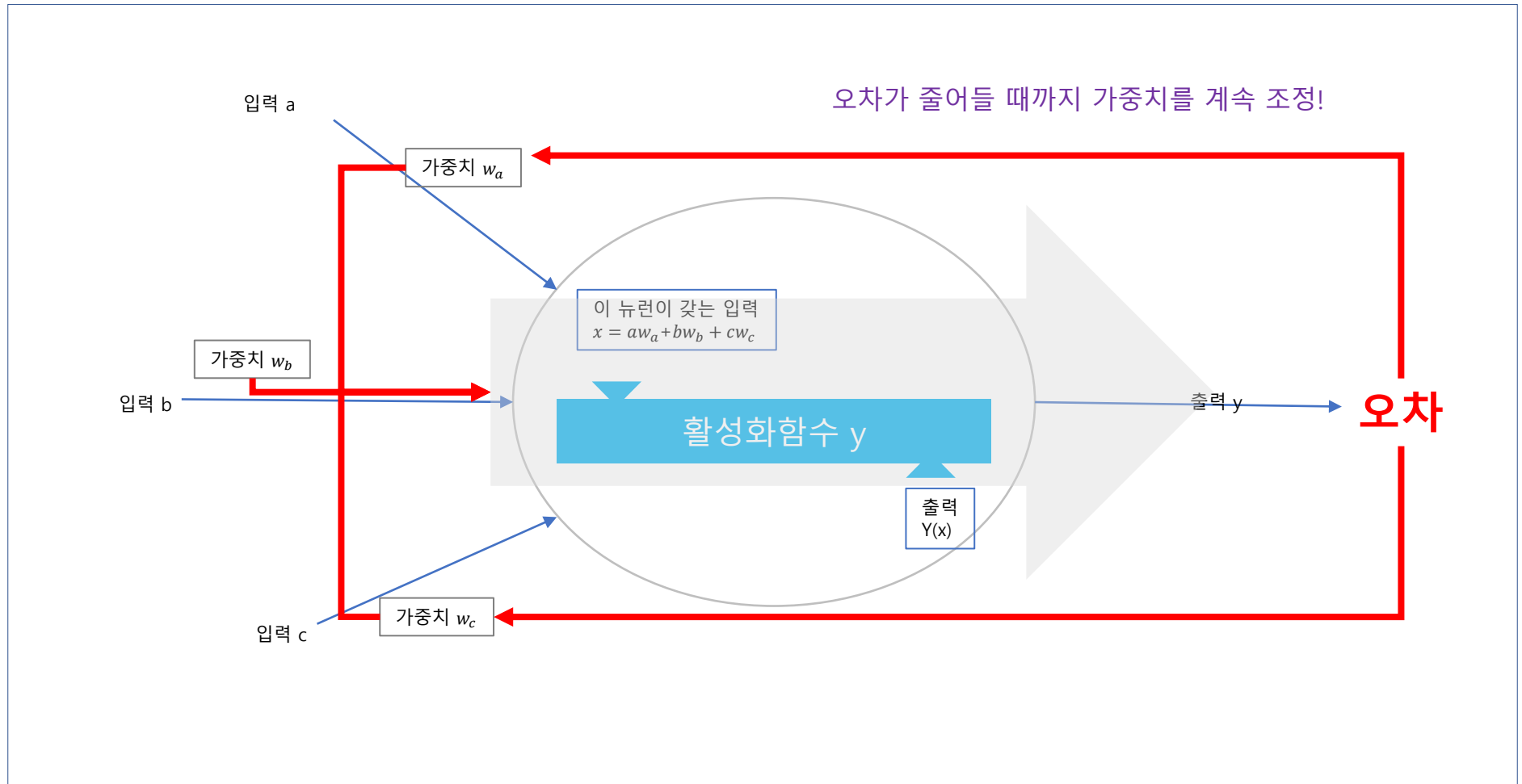
인공신경망 2단계: 오차의 역전파!



VI. 역전파 알고리즘

ANN Step by Step

잘 할 때까지 반복!

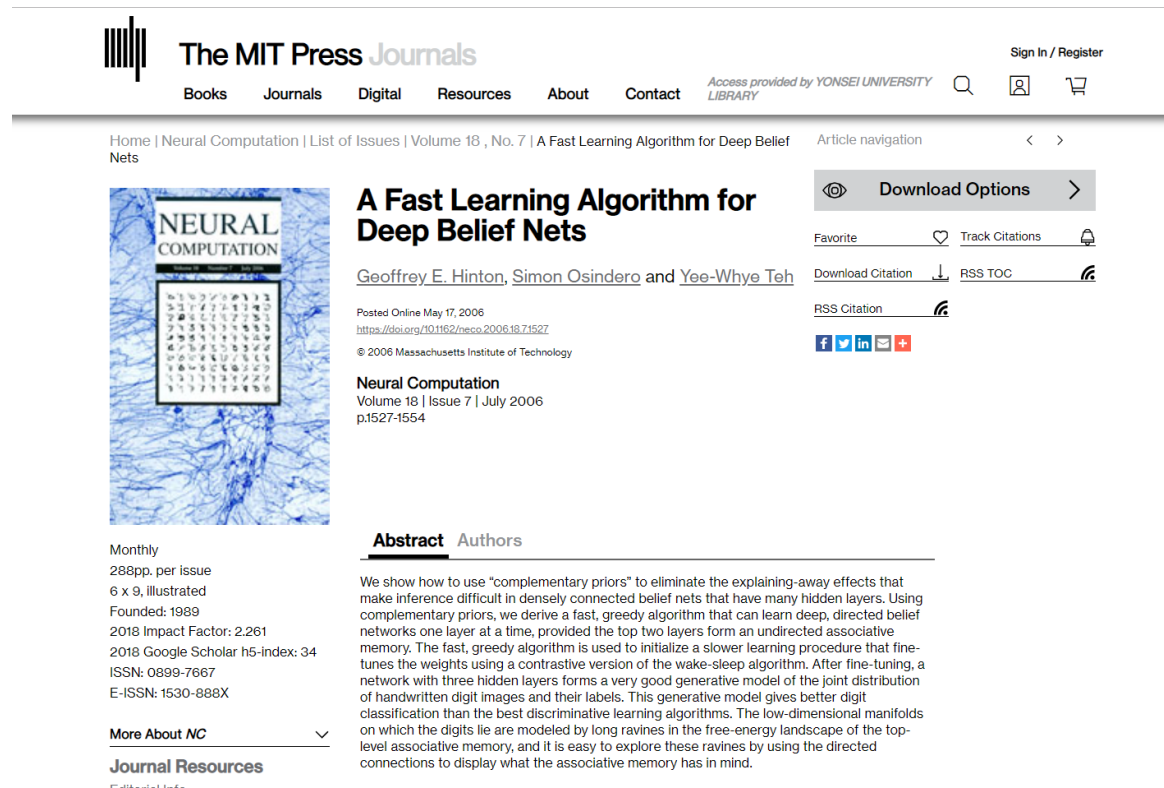


Part II. Deep Learning

I. 딥러닝 알고리즘의 특징

• 딥러닝 알고리즘의 등장!

- 기존 다층신경망의 역전파 알고리즘의 극복: 은닉층이 많아질 수록 성능 저하의 이슈가 해결
- ANN 이슈들을 해결
- Convolution NN, Auto-encoder, Recurrent NN 등이 많이 활용



The MIT Press Journals

Books Journals Digital Resources About Contact

Access provided by YONSEI UNIVERSITY LIBRARY

Sign In / Register

Home | Neural Computation | List of Issues | Volume 18, No. 7 | A Fast Learning Algorithm for Deep Belief Nets

Article navigation < >

A Fast Learning Algorithm for Deep Belief Nets

Geoffrey E. Hinton, Simon Osindero and Yee-Whye Teh

Posted Online May 17, 2006
<https://doi.org/10.1162/neco.2006.18.7.1527>
© 2006 Massachusetts Institute of Technology

Neural Computation
Volume 18 | Issue 7 | July 2006
p.1527-1554

Abstract Authors

We show how to use "complementary priors" to eliminate the explaining-away effects that make inference difficult in densely connected belief nets that have many hidden layers. Using complementary priors, we derive a fast, greedy algorithm that can learn deep, directed belief networks one layer at a time, provided the top two layers form an undirected associative memory. The fast, greedy algorithm is used to initialize a slower learning procedure that fine-tunes the weights using a contrastive version of the wake-sleep algorithm. After fine-tuning, a network with three hidden layers forms a very good generative model of the joint distribution of handwritten digit images and their labels. This generative model gives better digit classification than the best discriminative learning algorithms. The low-dimensional manifolds on which the digits lie are modeled by long ravines in the free-energy landscape of the top-level associative memory, and it is easy to explore these ravines by using the directed connections to display what the associative memory has in mind.

Monthly
288pp. per issue
6 x 9, illustrated
Founded: 1989
2018 Impact Factor: 2.261
2018 Google Scholar h5-index: 34
ISSN: 0899-7667
E-ISSN: 1530-888X

More About NC

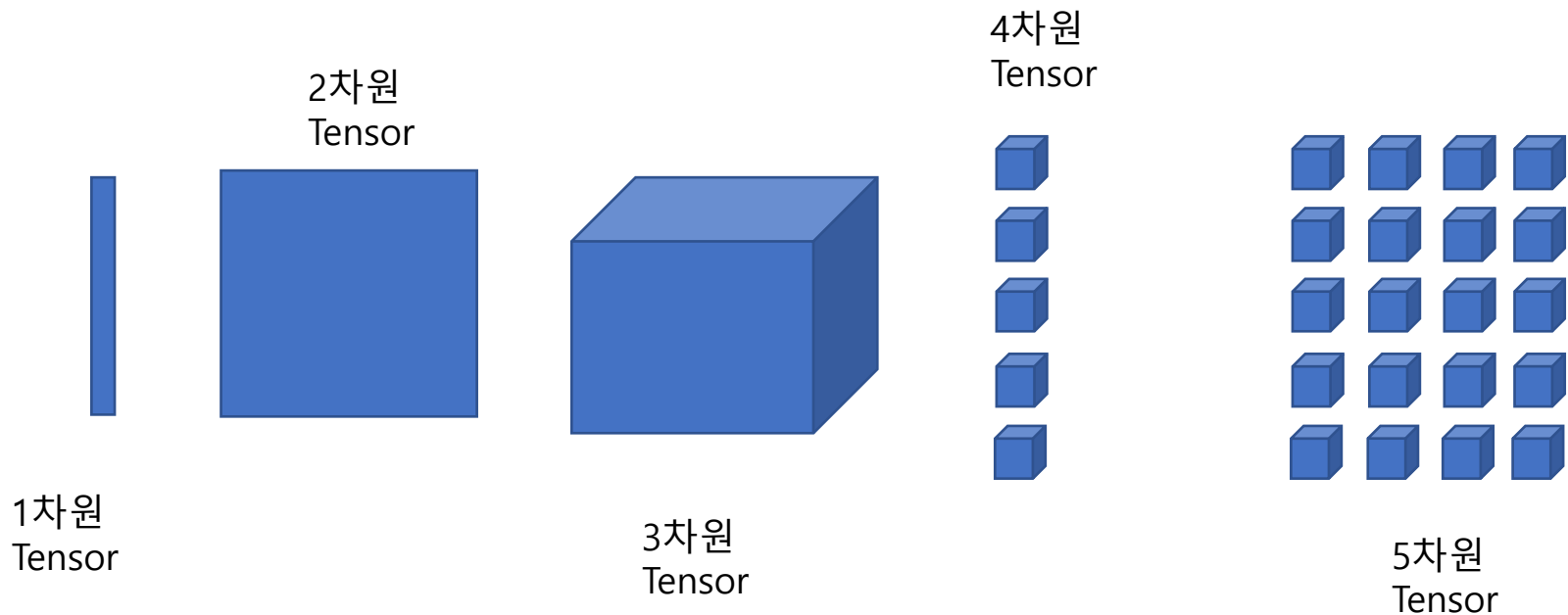
Journal Resources

다층신경망의 성능 문제의 해결방법이
2006년 Hinton의 "A fast learning algorithm for deep belief nets"를 통해 제시되면서, '딥러닝'으로 주목

I. 딥러닝 알고리즘의 특징

• 딥러닝의 기본 자료구조: Tensor

- 다차원 배열이나 리스트
- Rank(차원의 수), shape(행과 열), type(값의 타입)으로 구분

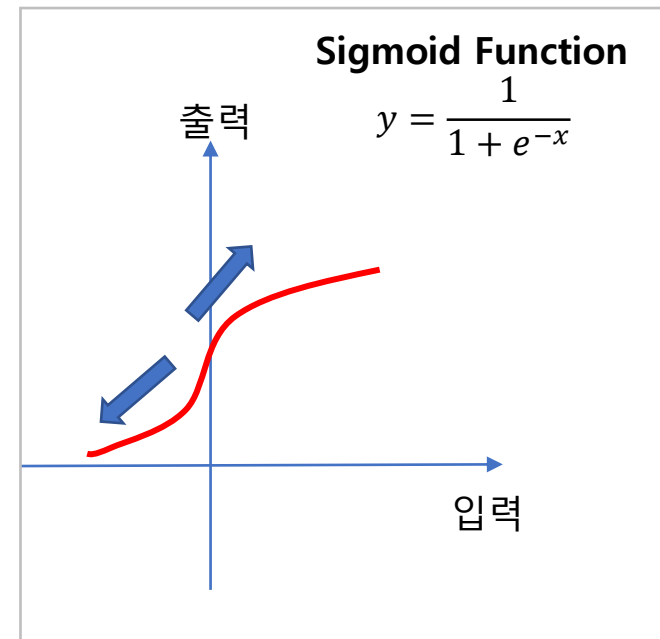
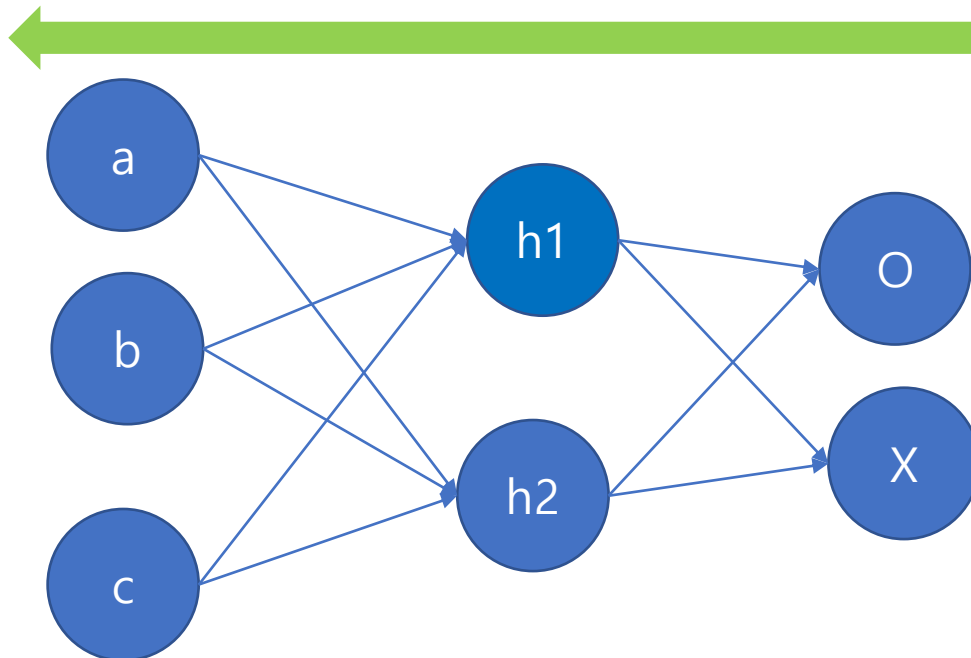


I. 딥러닝 알고리즘의 특징

Vanishing Gradient

- *Gradient?* 미분된 함수의 기울기
- *Gradient*는 역전파(*Back propagation*) 과정에 사용: 기존 활성화 함수(*Sigmoid*)를 거치면 원래의 값이 현저히 작아짐(작은 x 는 더 작은 y 로 변환, 큰 x 는 더 큰 y 로 변환)

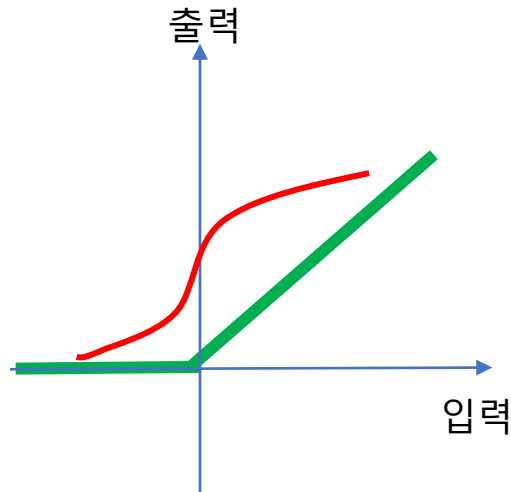
인공신경망의 두 번째 겨울 (1986-2006)



I. 딥러닝 알고리즘의 특징

ReLU 활성화함수의 등장 Rectified Linear Unit

2006년 Jeffrey Hinton, "Vanishing Gradient의 문제는 Sigmoid 활성화 함수 때문!"



Sigmoid Function

$$y = \frac{1}{1 + e^{-x}}$$

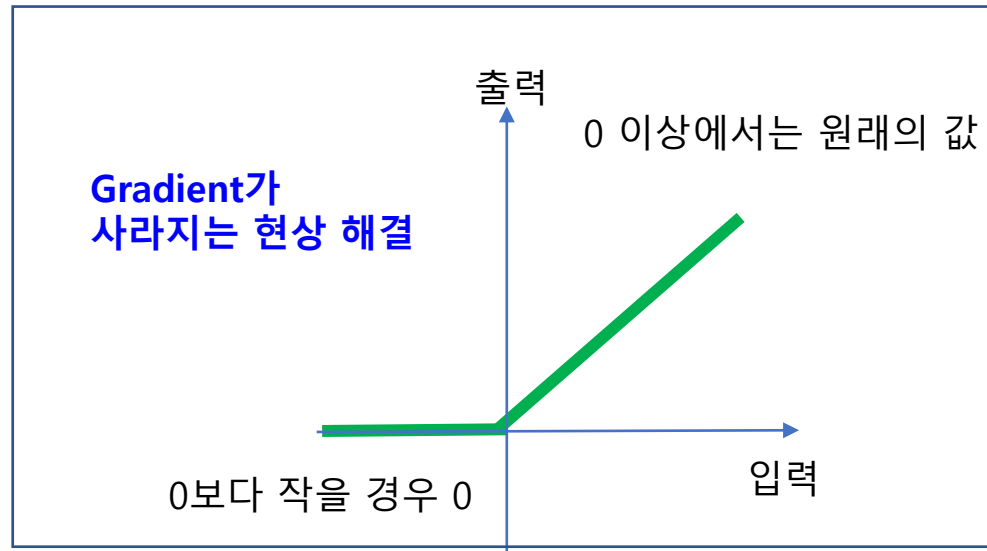


**ReLU Function
(Rectified Linear Unit)**

I. 딥러닝 알고리즘의 특징

ReLU 활성화 함수를 통한 인공신경망의 개선

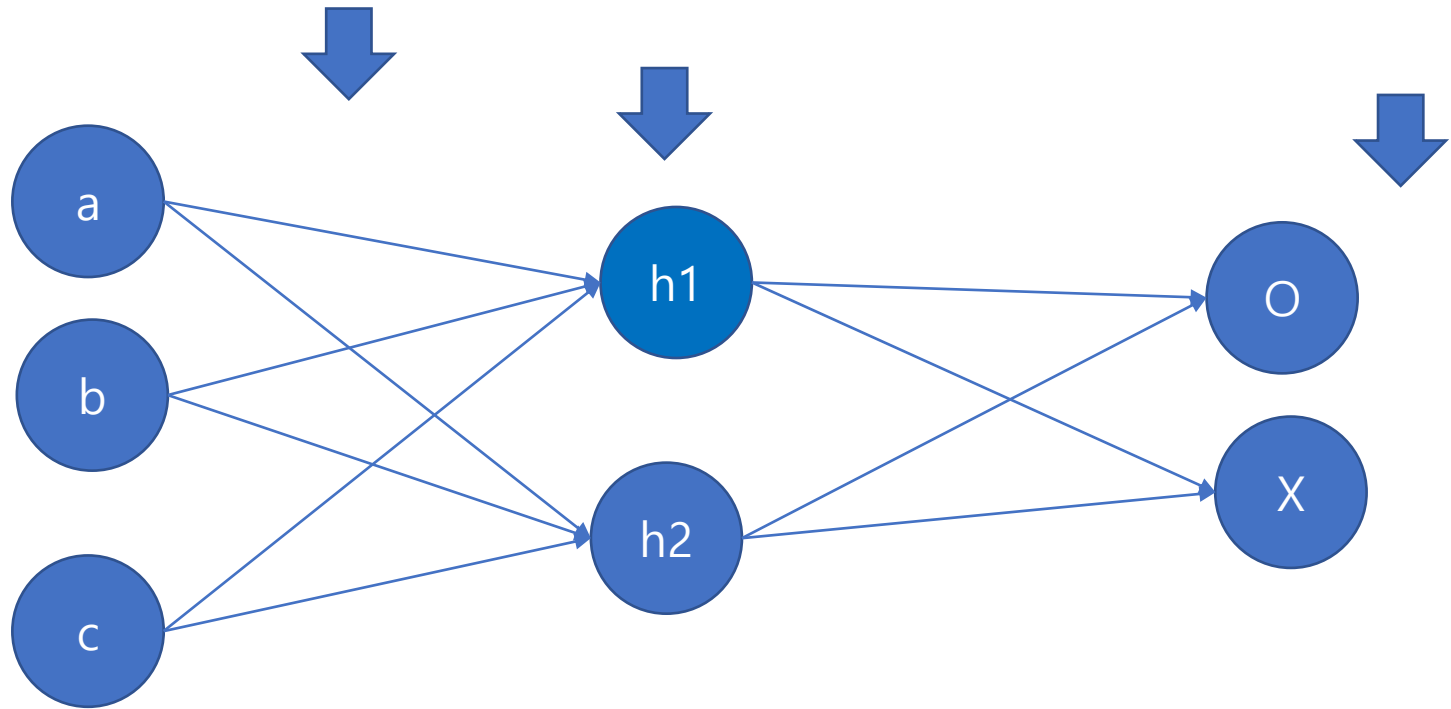
- 여러 은닉층에 잘 작동!
- 계산 비용 낮추고 정확도 증가
- ReLU도 다양한 변형이 가능!



II. 딥러닝 파라미터의 이해

인공신경망의 Parameter 혹은 Hyper Parameter

은닉층의 수, 노드의 수, 학습율 등



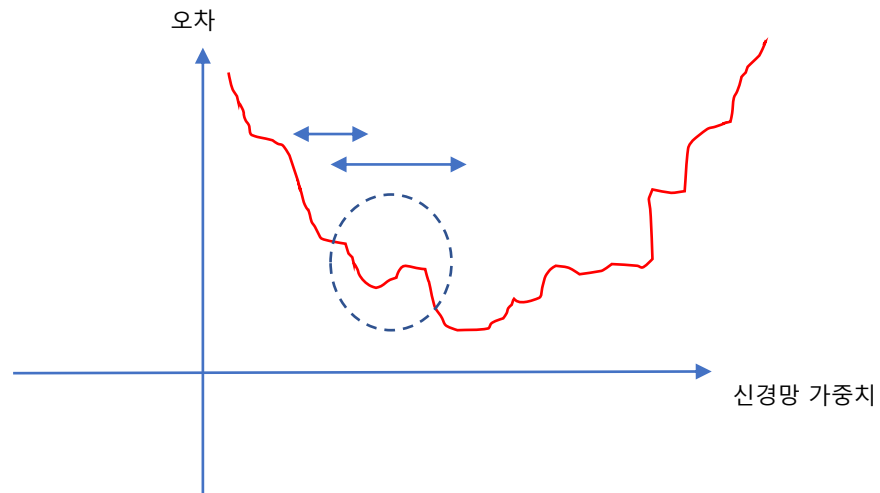
II. 딥러닝 파라미터의 이해

최적화 알고리즘과 학습율

학습율(Learning rate): 가장 중요한 하이퍼파라미터!

너무 크게 잡힌 경우!
오차가 크다!

너무 작게 잡힌 경우!
최적이 아닌 가중치를 계산한다!



II. 딥러닝 파라미터의 이해

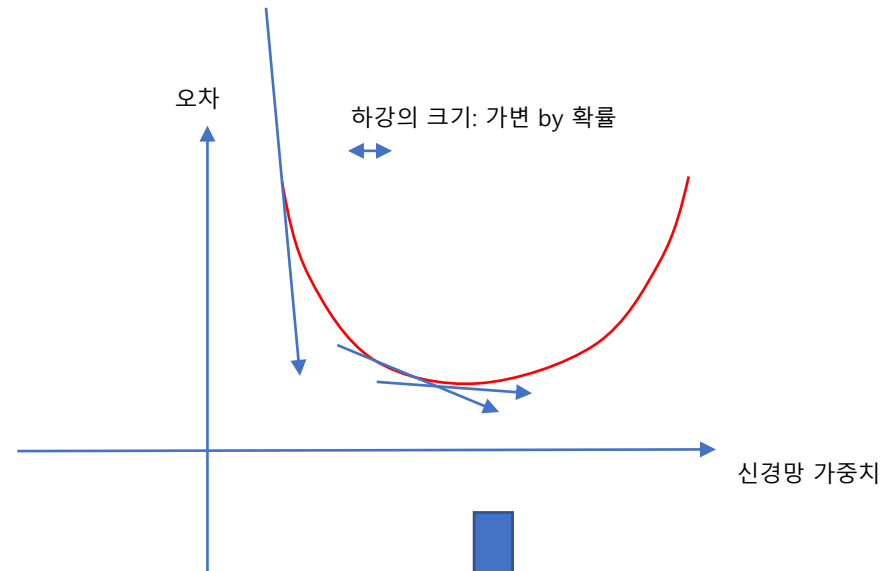
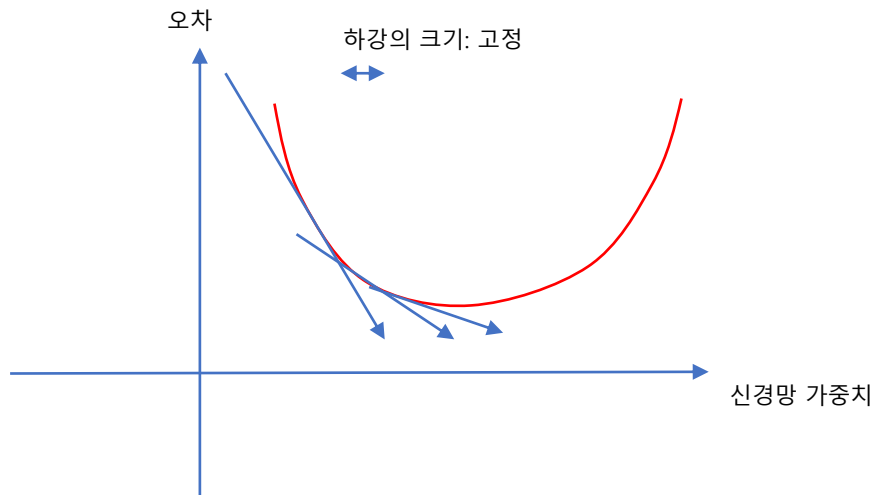
최적화 알고리즘과 학습율

- 최적화 알고리즘
 - Gradient Descent
 - Stochastic Gradient Descent
 - Momentum(관성 고려)
 - [Adam\(Momentum+RMSProp\)](#)
 - Adagrad(진행될 수록 변화 정도를 줄임)
 - RMSProp(상황을 보가며 변화정도 조정)
 - [Adam\(Momentum+RMSProp\)](#)

III. 경사 하강과 미니배치

- 경사하강 (Stochastic Gradient Descent):

- 함수의 기울기를 작게하면서 최소값을 구할때 까지 반복
- 최소값을 발견! 계산시간 소요!



- 확률적 경사하강 (Stochastic Gradient Descent)

- 경사하강의 계산시간이 오래걸리는 점을 보완하기 위해 일부 데이터로 최소값을 발견

III. 경사 하강과 미니배치

배치(Batch)?

공정에서의 묶음 단위

=

전체 데이터

경사하강을 통한 최적의 가중치!

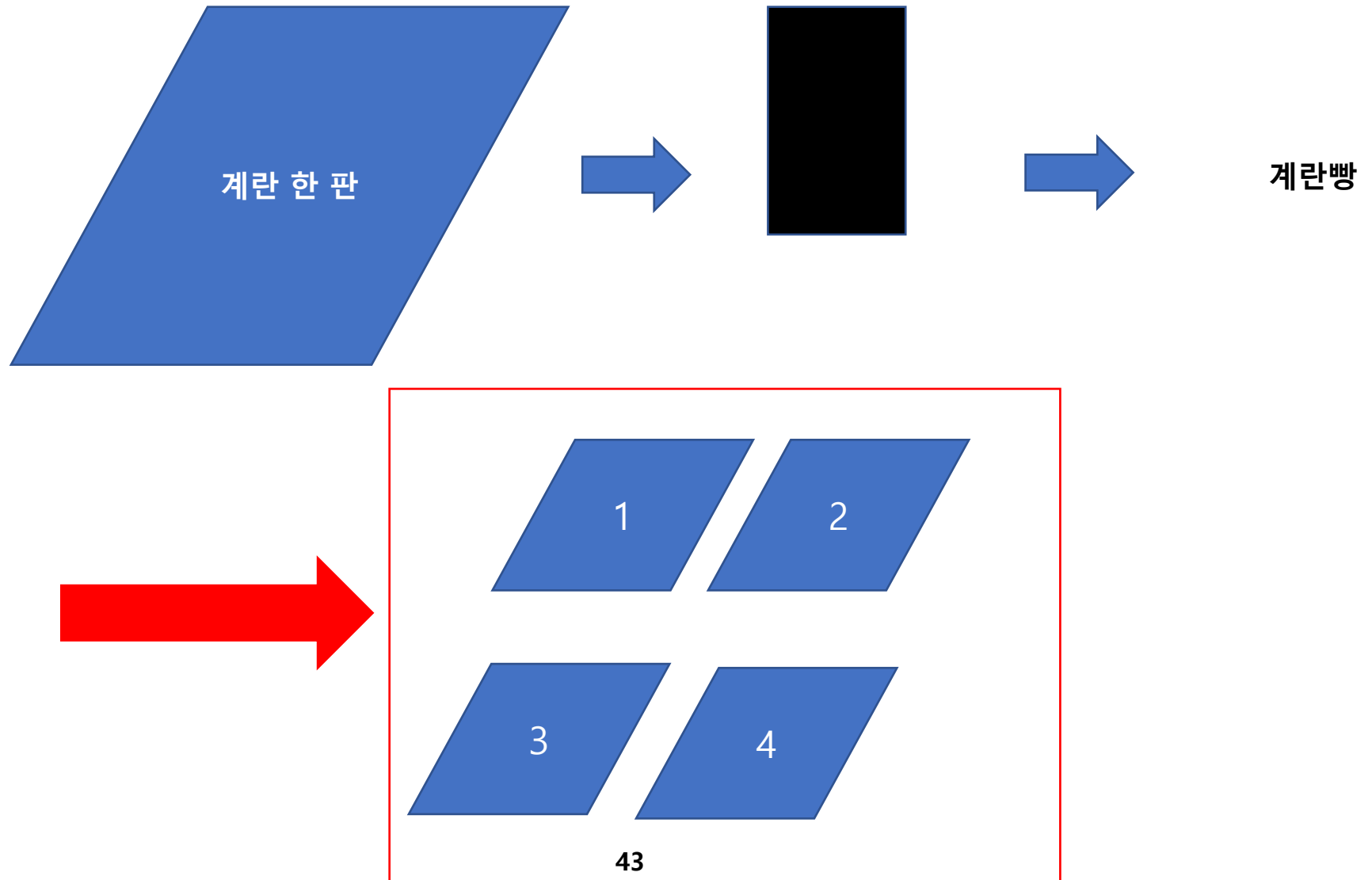


배치방식의 장점:

- 개별로 처리하는 것보다는 배치로 처리하는 것이 효율적!
- 안정적인 최적의 값을 발견할 수 있음

III. 경사 하강과 미니배치

배치(Batch)가 큰 경우: 미니배치(Mini Batch)!



III. 경사 하강과 미니배치

미니배치(Mini Batch)의 특징



Batch: 전체 데이터

Mini Batch: 여러 개로 나누기

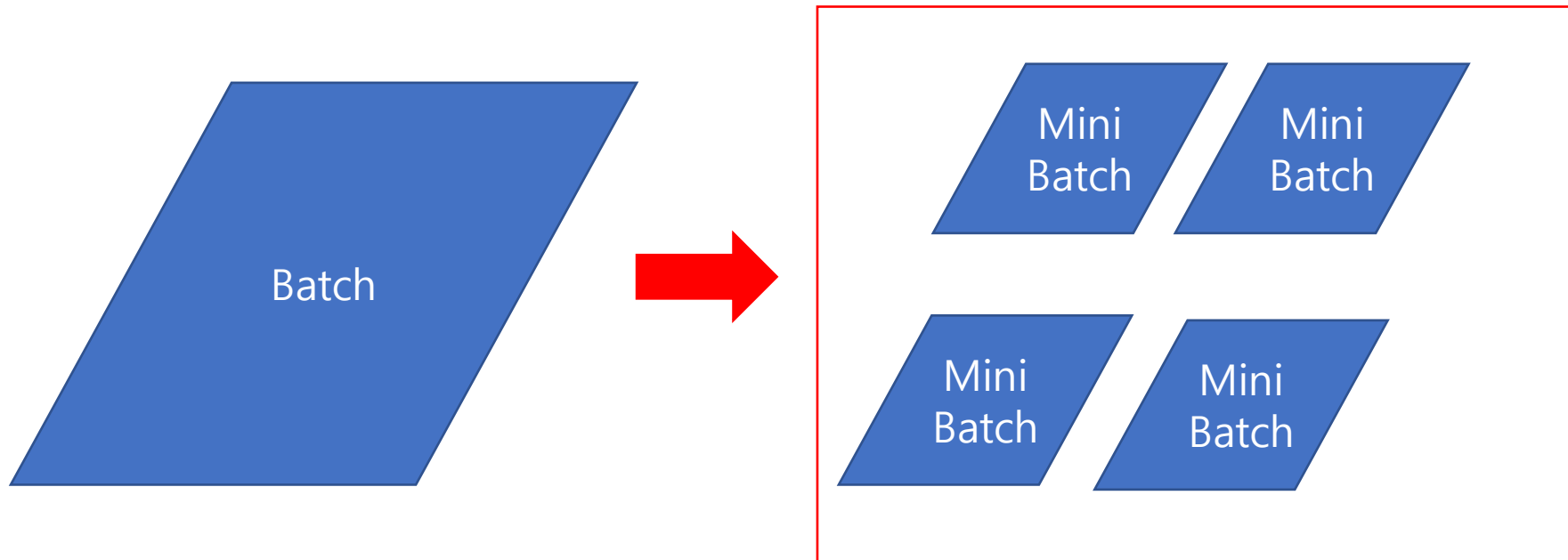
각각에 대한 경사하강 적용

- 배치의 장점을 계승
- 배치 사용 시 소요 시간 > 미니 배치 사용 시 소요 시간
- 여전히 전체 데이터를 학습!

III. 경사 하강과 미니배치

Iteration

- 배치 크기: 각 미니배치에 포함된 데이터의 수



- 전체 데이터 학습->1 에포크

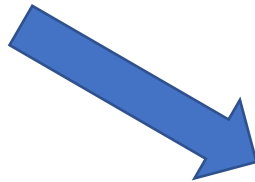
- Iteration: 한 에포크에서 미니배치수를 통한 학습의 횟수, 여기서는 4

IV. 과적합과 드롭아웃

딥러닝과 과적합

주어진 Training Data에만 잘 맞도록 모델링된 것을 Over Fitting(과적합)이라고 하며, Testing Data나 실제 적용 시 오차가 많이 발생할 수 있음

- 은닉층을 통한 입력 데이터의 비선형 변환
- 우수한 성능
- 과적합의 발생



- 차라리 정말 많은 데이터
- 가중치 제한(Regularization)
- 드롭아웃

IV. 과적합과 드롭아웃

Drop Out

인공신경망의 학습에서 *Random*하게 은닉층의 특정 노드들을 사용하지 않는 것을 의미하며, *Feature* 및 은닉층 표현을 풍부하게 하여 성능을 개선할 수 있음



- 인공신경망의 Layer의 일부 Weight만 학습에 사용
- 무작위로 특정 노드의 값을 0으로 지정!

IV. 과적합과 드롭아웃

Drop Out: 인공신경망의 Feature표현이 풍부

Drop Out Rate: 0~1

예: Drop Out rate, 0.7

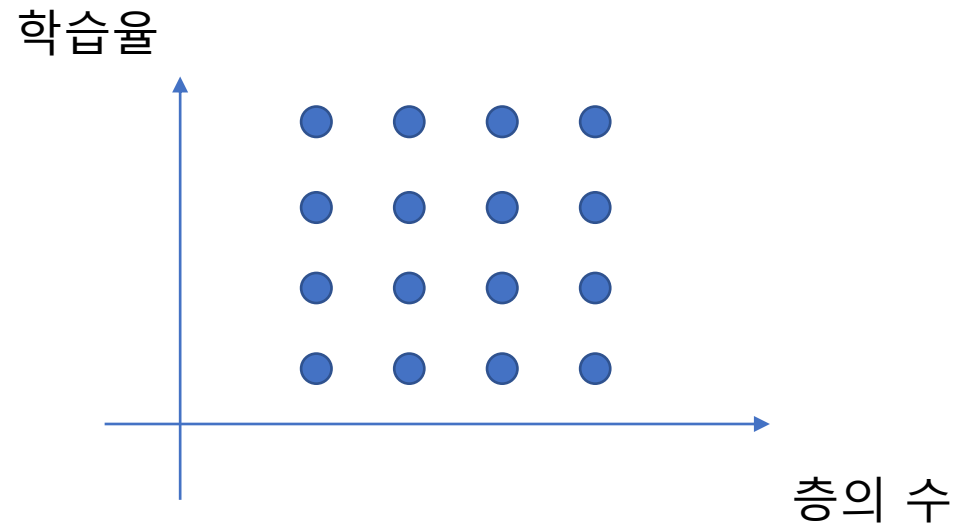
각 weight가 Drop out 될 확률 70%로 지정

V. 딥러닝 튜닝

Grid Search?

모든 가능한 하이퍼파라미터의 조합을 체계적으로 하나씩 탐색하는 방식

예: 2개의 파라미터의 조합

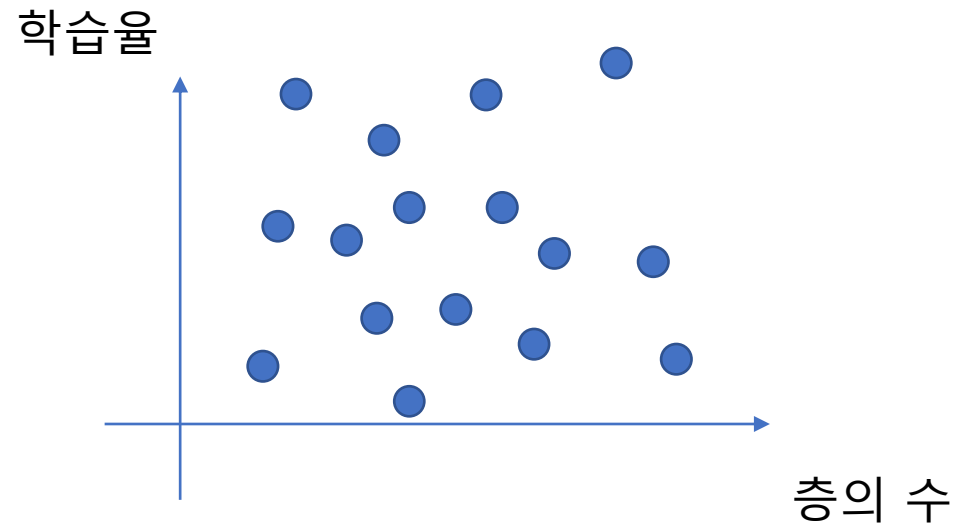


V. 딥러닝 튜닝

Random Search?

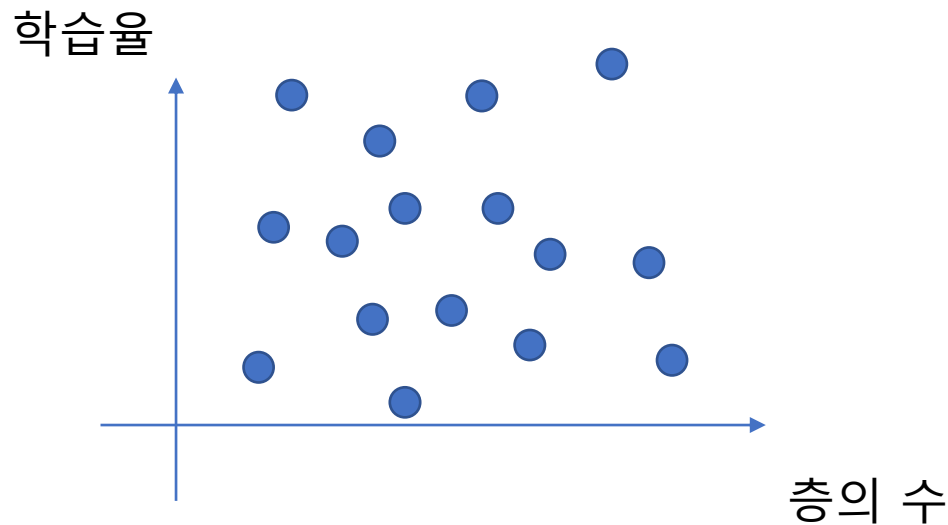
Random하게 하이퍼파라미터의 조합을 탐색

예: Random하게 생성되는, 파라미터 2개에 의한 값의 조합



Grid Search보다 우수할 수도!

Random Search?



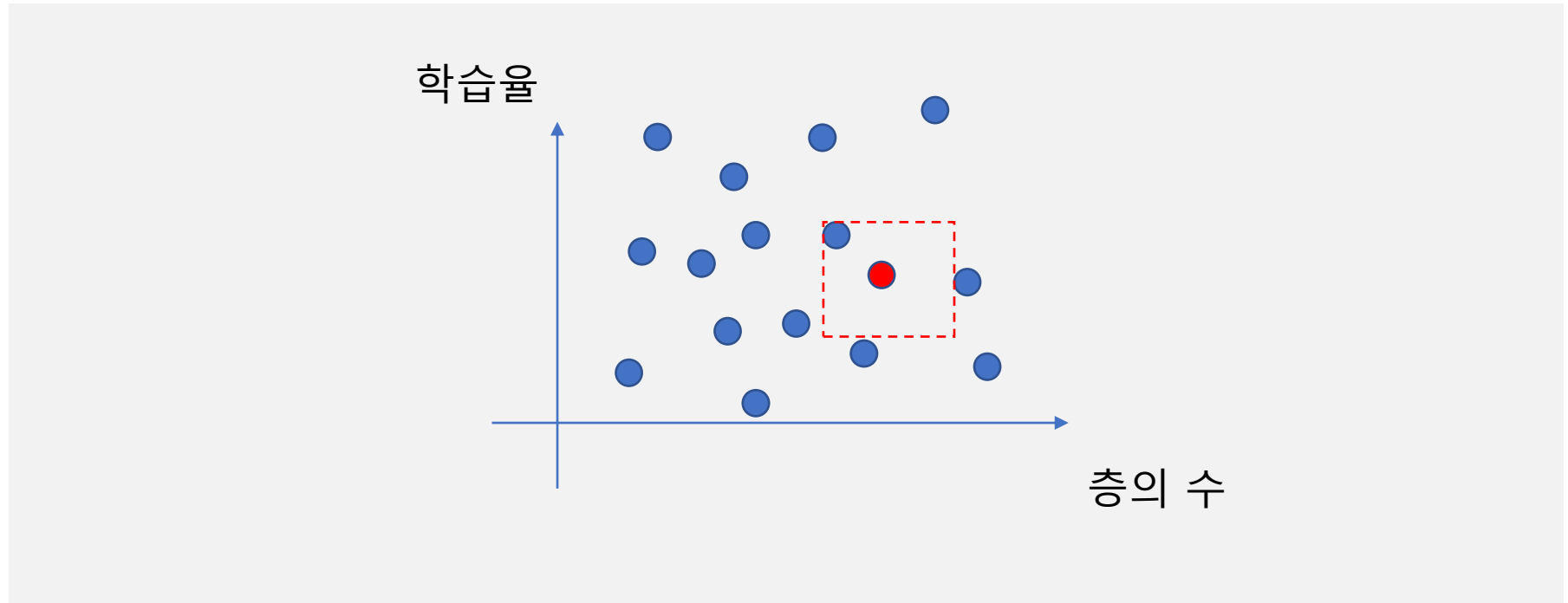
Don't use Grid!
from Andrew Ng

- 왜? 인공지능망에서는 어떤 파라미터가
- 성능에 어느 정도 영향을 주는지를 알 수 없음
- 중요한 하이퍼파라미터 관점에서 먼저 탐색!

예: 두 조합 중 학습율이 있다면, 학습율 우선으로 고려,
이후 전체 조합의 개수 산출, 다른 파라미터 값을 검색

V. 딥러닝 튜닝

정밀화 접근!



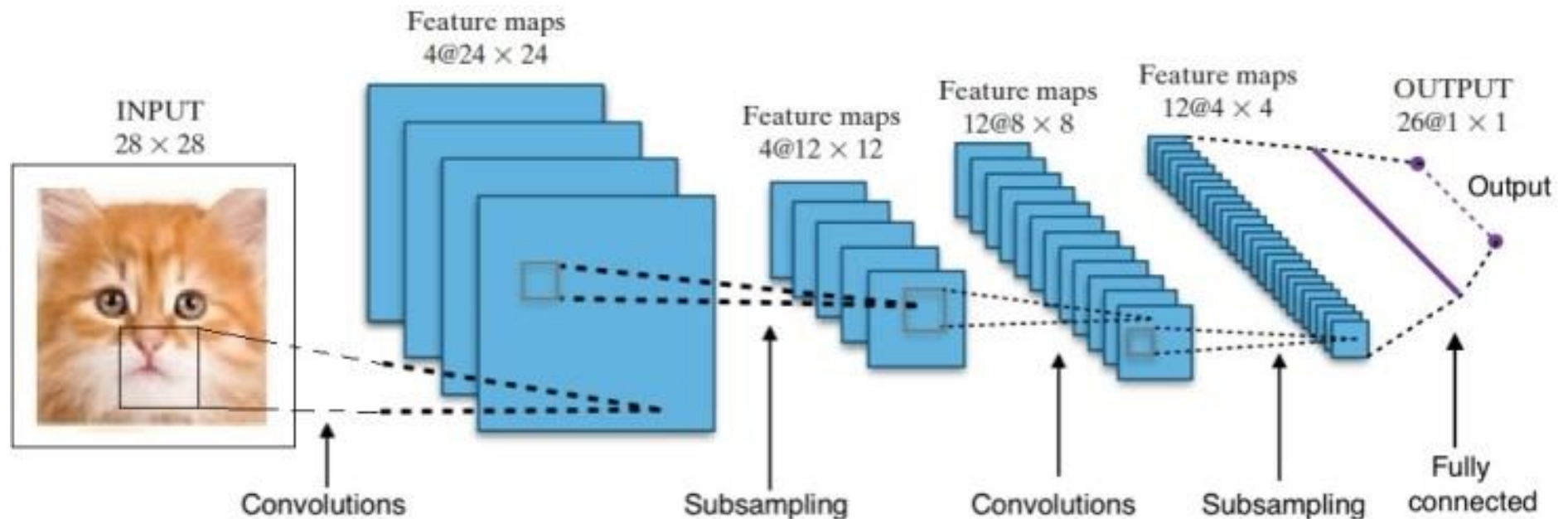
Random한 탐색에서 발견한 최적의 하이퍼파라미터 주변을 더 탐색!

VII. 이미지 분류를 위한 CNN

1. 이미지 데이터: 디지털 이미지의 값을 2차원 수치로 표현
2. CNN: 이미지 분류를 위한 딥러닝 기법으로 합성곱 신경망이라고도 하며 이미지 분류에 잘 활용됨
3. CNN 과정: 필터를 스트라이드하는 Convolution과정과, 이 결과에서 Feature값을 줄이는 Pooling 과정을 거쳐 이미지를 인공신경망에 적용
4. CNN 단계별 절차: CNN은 입력 이미지의 공간정보를 유지한 상태에서 Convolution과 Pooling을 반복한 후 인공신경망을 통해 분류하는 과정을 거침
5. 주요 하이퍼 파라미터: Convolution의 필터의 수, 필터의 크기, Padding, Stride 등이 CNN의 주요 하이퍼 파라미터임.
6. Convolution과 Pooling의 과정: Convolution과정을 통해서는 필터의 수 만큼 Feature Map이 생성되고, Pooling을 통해서는 주요 Feature들이 강화 됨.

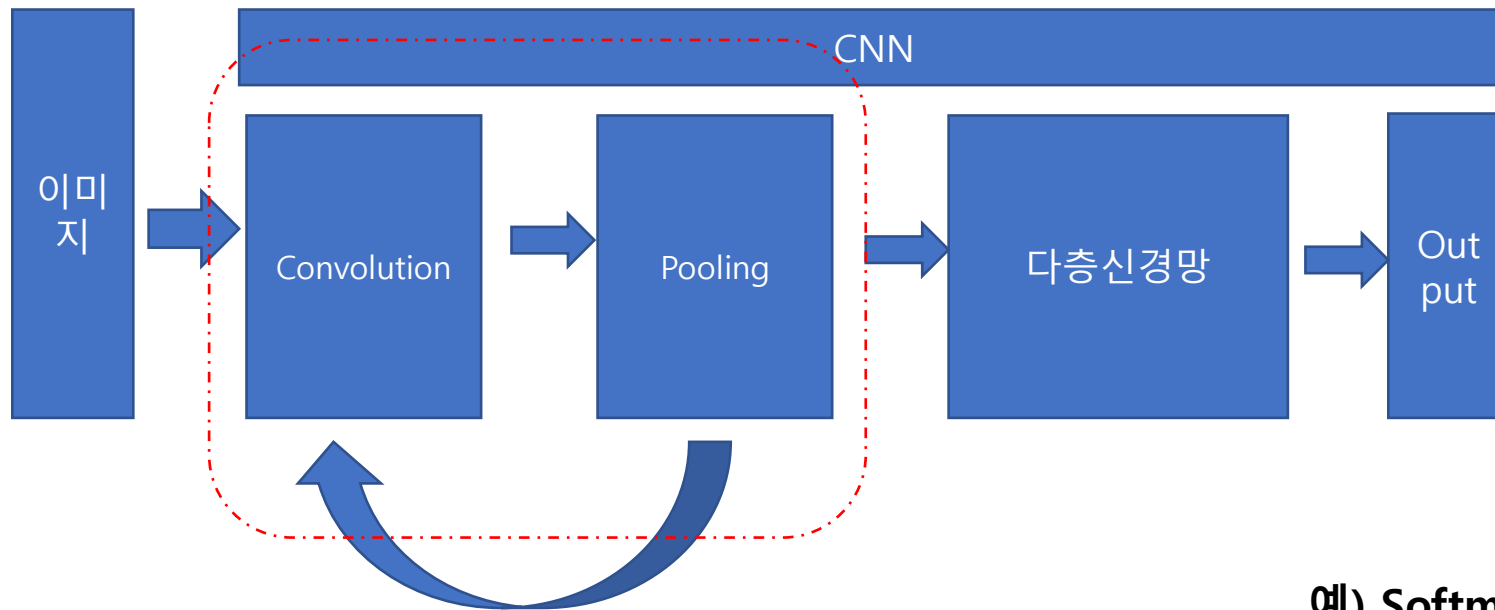
VII. 이미지 분류를 위한 CNN

Convolution Neural Network 합성곱 신경망 이미지 인식을 위한 인공신경망



VII. 이미지 분류를 위한 CNN

CNN의 주요 단계



예) Softmax 활성화함수
:출력을 0~1사이 값
:출력의 합은 1

VII. 이미지 분류를 위한 CNN

CNN 주요 하이퍼 파라미터

- Filter: 원래 입력 이미지에서 주요 Feature를 탐지하는 역할, Feature Map이 나오게 됨
- Convolution Filter의 수: 원래 이미지로 부터 Filter의 수 만큼 다른 방식으로 Feature Map을 탐지
- Padding 여부: Padding은 입력 데이터보다 출력이 작아지는 것을 방지, 주로 원래 이미지 주변을 0으로 채워 Filter가 이미지 첫 시작점부터 작동할 수 있도록 함.
- Stride: Filter가 움직이는 간격

VII. 이미지 분류를 위한 CNN

멀티채널과 단일채널

멀티채널(칼라): $R+G+B$

단일채널(흑백)

- 생성되는 Convolution결과의 수
- 흑백: Filter의 수와 같은 Feature Map
- 칼라(R,G,B인 경우): Filter의 수와 같은 Feature Map X 3

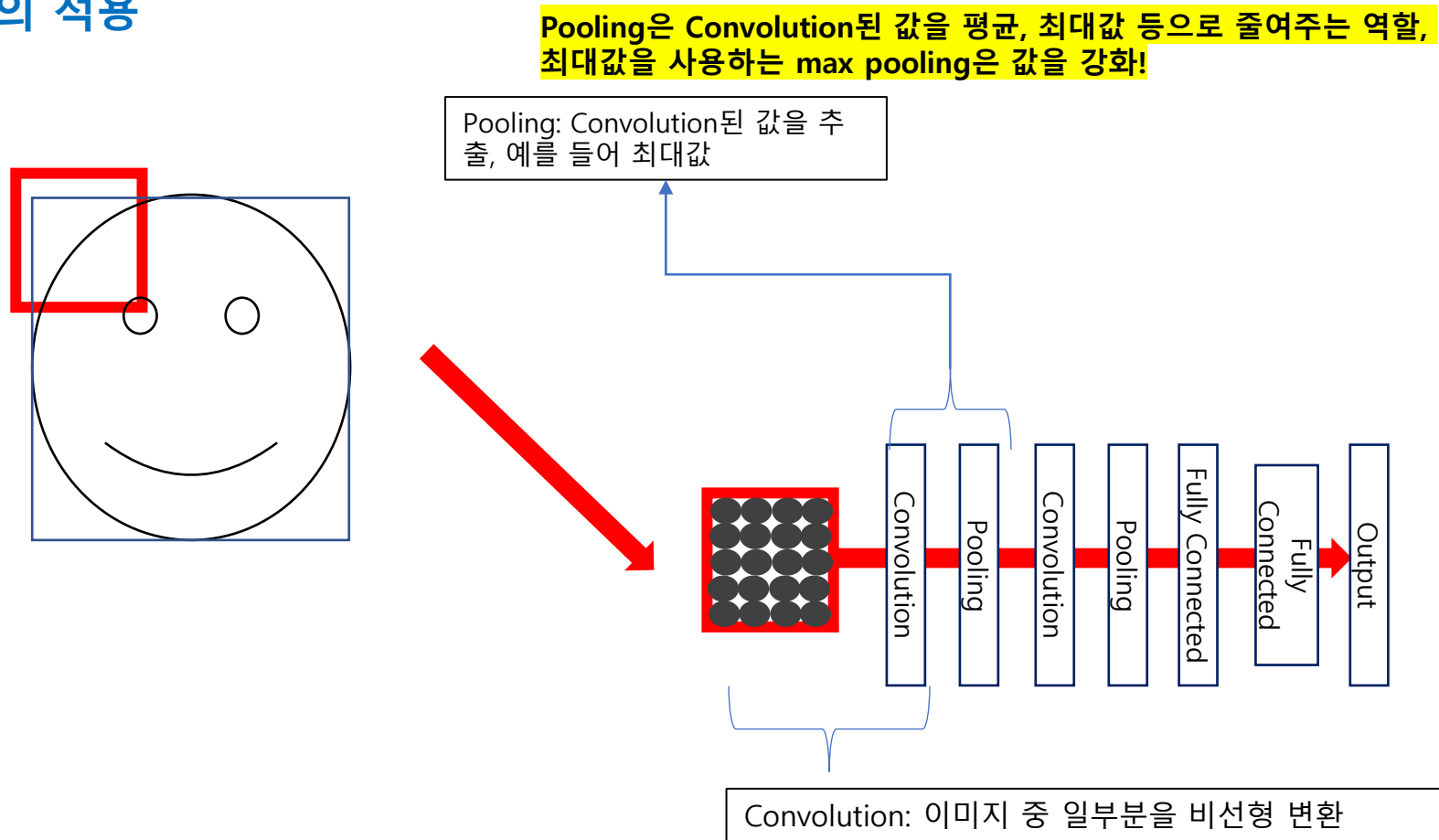
VII. 이미지 분류를 위한 CNN

Convolution과 Pooling

- Convolution:
 - 이미지의 주요 Feature를 다양한 관점의 Filter에 따라 추출 하이퍼파라미터를 지정
- Pooling:
 - Feature Map에서 특정 Feature를 강조
 - Pooling은 하이퍼파라미터 없음
 - Pooling을 해도 채널의 수는 동일하지만, 행렬 크기는 감소

VII. 이미지 분류를 위한 CNN

Filter의 적용



Convolution은 필터를 통해 이미지 일부를 새로운 값으로 계산

VII. 이미지 분류를 위한 CNN

CNN의 특징은!

- 입력 데이터의 공간 정보를 유지 = 2차원(또는 3차원)입력데이터를 사용
- Feature Map을 사용 = Feature Map은 입력데이터, 은닉층으로 전달되고, 은닉층에서 계산되는 값을 의미
 - **합성곱(Convolution) 연산을 수행!**

VII. 이미지 분류를 위한 CNN

Convolution 과정

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Convolution!

4		



4	3	4
2	4	3
2	3	4



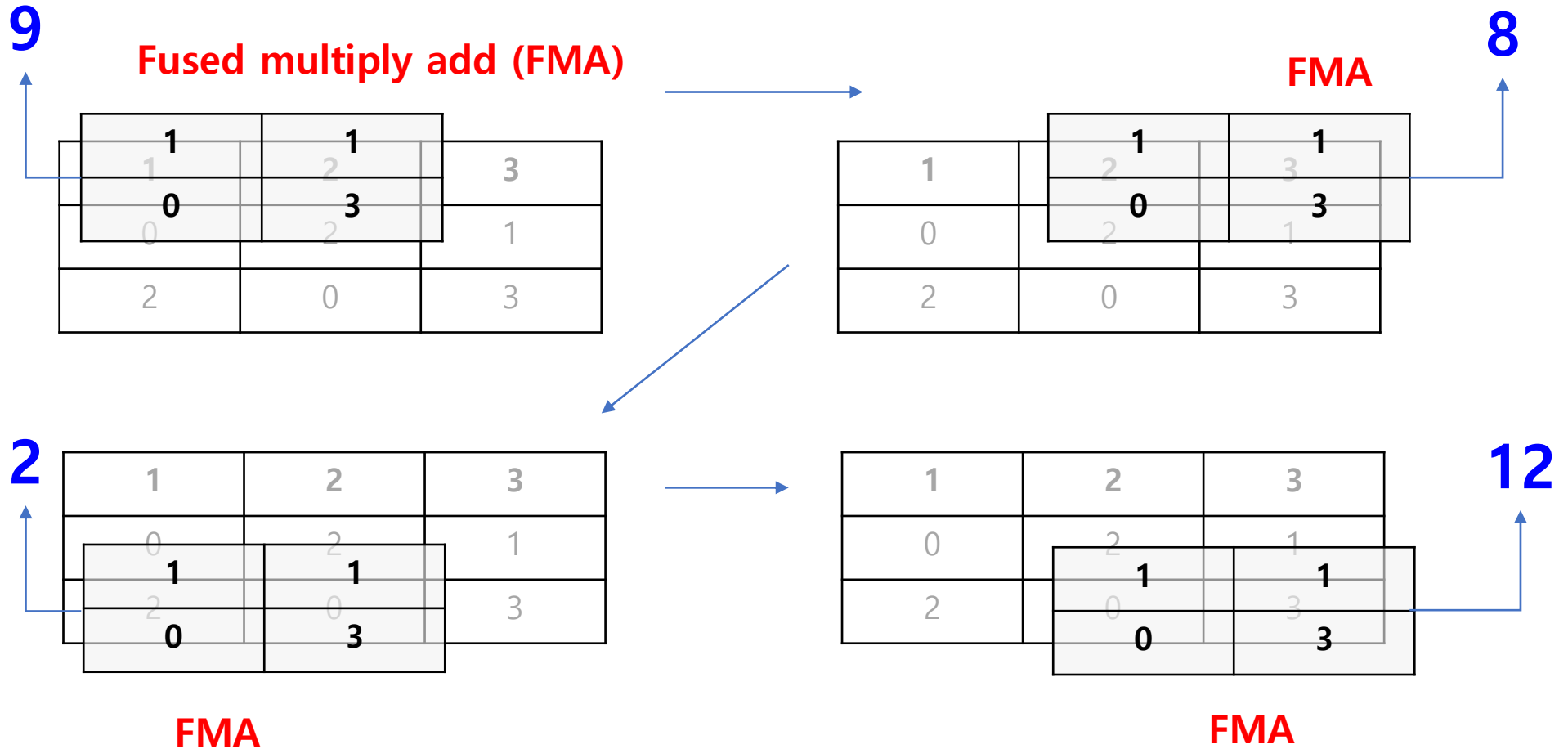
Max pooling

4

VII. 이미지 분류를 위한 CNN

Convolution

한 칸씩 움직임=Stride의 크기

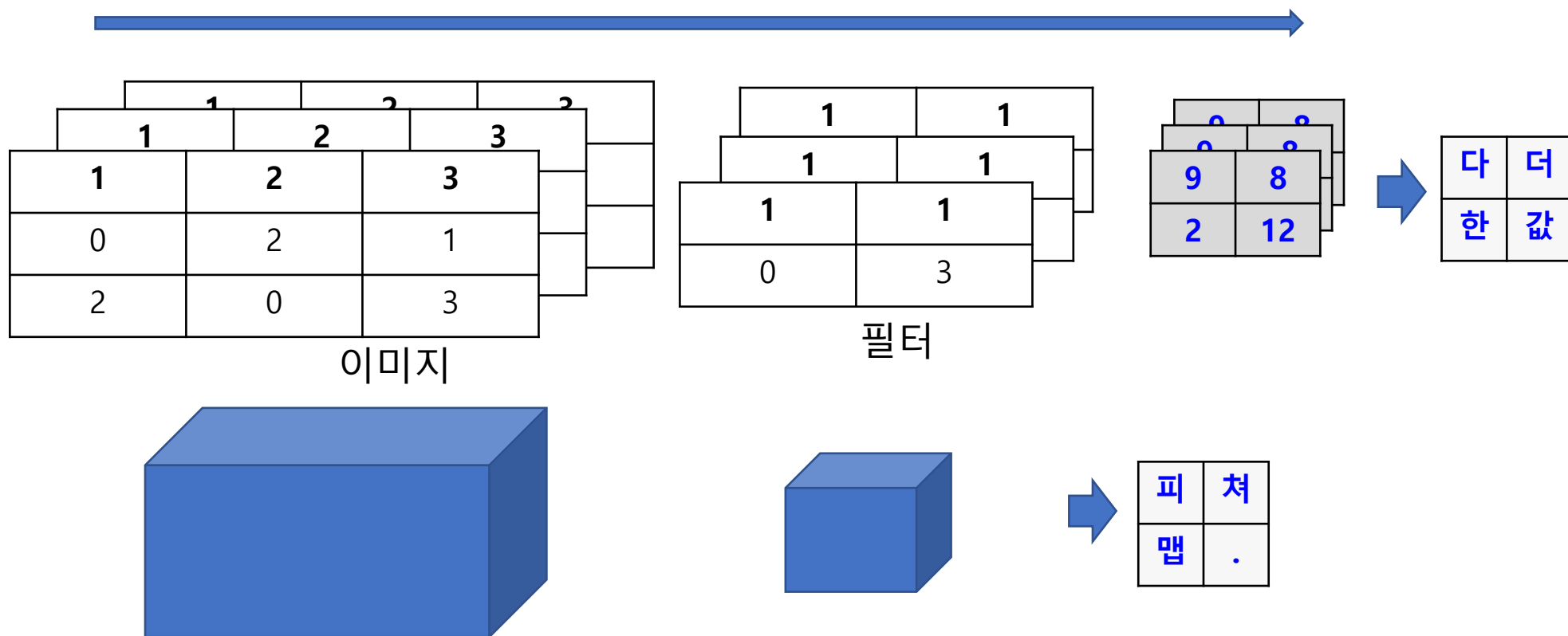


VII. 이미지 분류를 위한 CNN

3차원 데이터의 합성곱

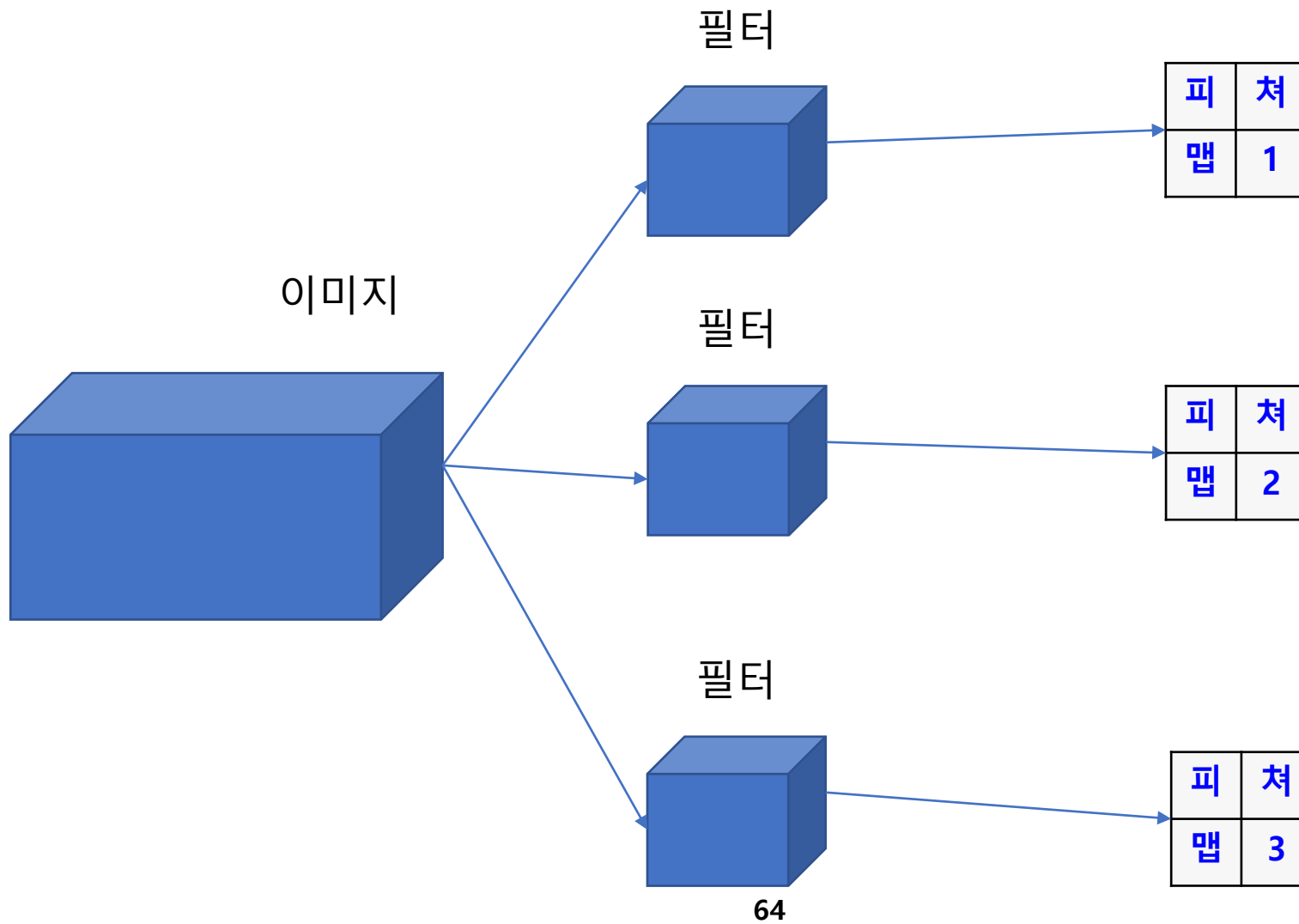
어떤 필터로 보느냐에 따라 대상이 다르게 처리됨
Convolution: 이미지 처리의 필터(=Kernel) 연산을 의미

채널별로 합성곱 연산



VII. 이미지 분류를 위한 CNN

3차원 데이터의 합성곱: 여러 필터!



VII. 이미지 분류를 위한 CNN

Pooling: 2차원 공간을 줄여주는 연산

- Pooling 절차: Convolution의 결과들의 공간을 줄여주며 주요 값을 더 강조할 수 있음
- Pooling의 여러 방법: Max Pooling과 Average Pooling등이 있으며, 이미지 처리에서는 Max Pooling을 많이 사용함
- Pooling의 특징: 하이퍼파라미터 학습이 없고, 입력값의 변화에 강건한 특징이 있음

1	2	3
0	2	1
2	0	3

이 피쳐맵을 줄여보기!

VII. 이미지 분류를 위한 CNN

Pooling: 2차원 공간을 줄여주는 연산: 2X2 영역별로 최대값을 구하기

2

1	2	3
0	2	1
2	0	3

3

1	2	3
0	2	1
2	0	3

2

1	2	3
0	2	1
2	0	3

3

1	2	3
0	2	1
2	0	3

VII. 이미지 분류를 위한 CNN

Pooling: 2차원 공간을 줄여주는 연산: 2X2 영역별로 최대값을 구하기

스트라이드: 1

2

1	2	3
0	2	1
2	0	3

3

1	2	3
0	2	1
2	0	3

2

1	2	3
0	2	1
2	0	3

3

1	2	3
0	2	1
2	0	3

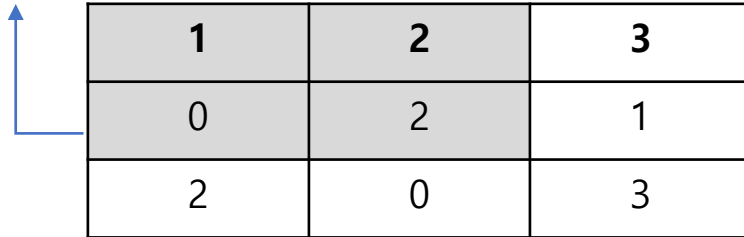
2X2 Max Pooling

VII. 이미지 분류를 위한 CNN

Pooling: 2차원 공간을 줄여주는 연산: 2X2 영역별로 최대값을 구하기

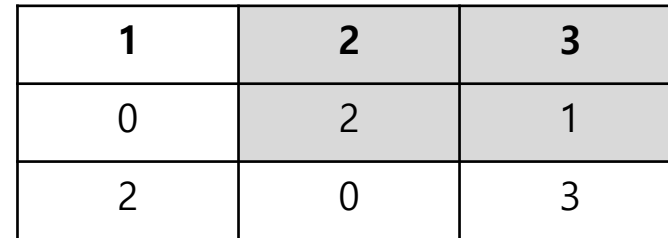
스트라이드: 1

1.25



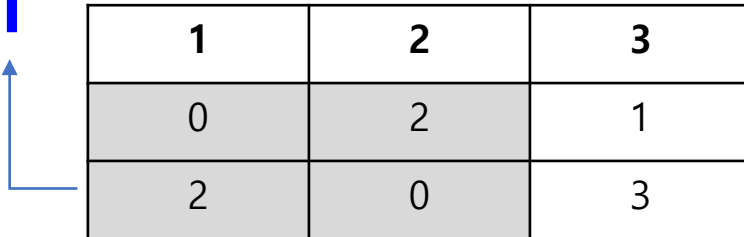
1	2	3
0	2	1
2	0	3

2



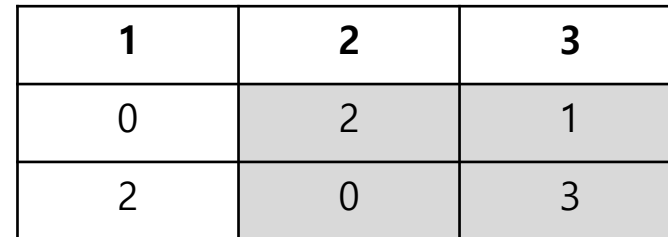
1	2	3
0	2	1
2	0	3

1



1	2	3
0	2	1
2	0	3

1.5

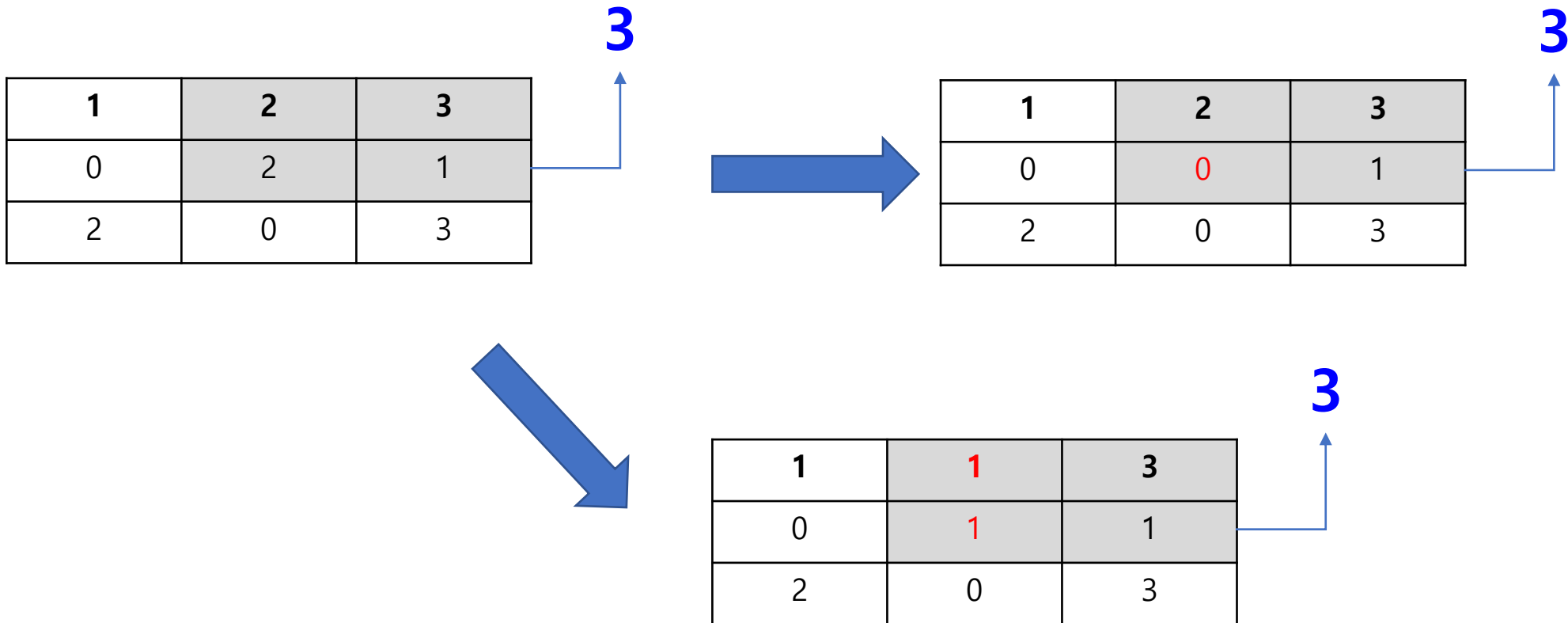


1	2	3
0	2	1
2	0	3

2X2 Average Pooling

VII. 이미지 분류를 위한 CNN

Pooling의 강건함: 입력값의 변화를 Pooling이 흡수, Robust한 결과를 제공

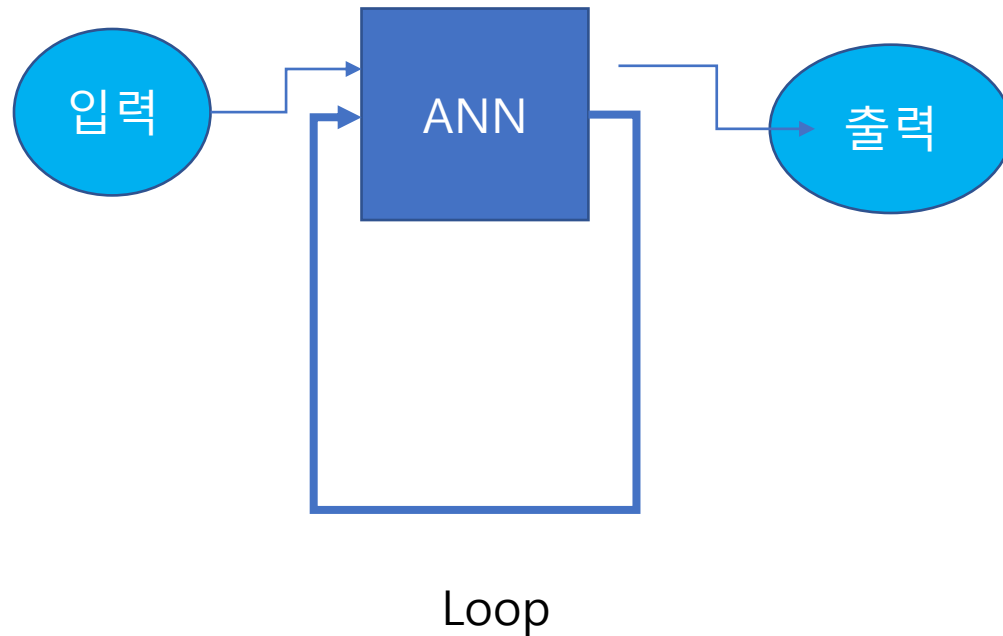


2X2 Max Pooling

이미지 처리에서는 주로
Max Pooling 사용!

- 순환신경망(Recurrent Neural Networks)

- Recurrent neural networks은 신경망 내에서 loop가 있고, 정보를 보존하게 함
- 신경망: 입력->은닉층(비선형변환)->출력
- 순환 신경망: 입력->은닉층(비선형변환)+Loop->출력

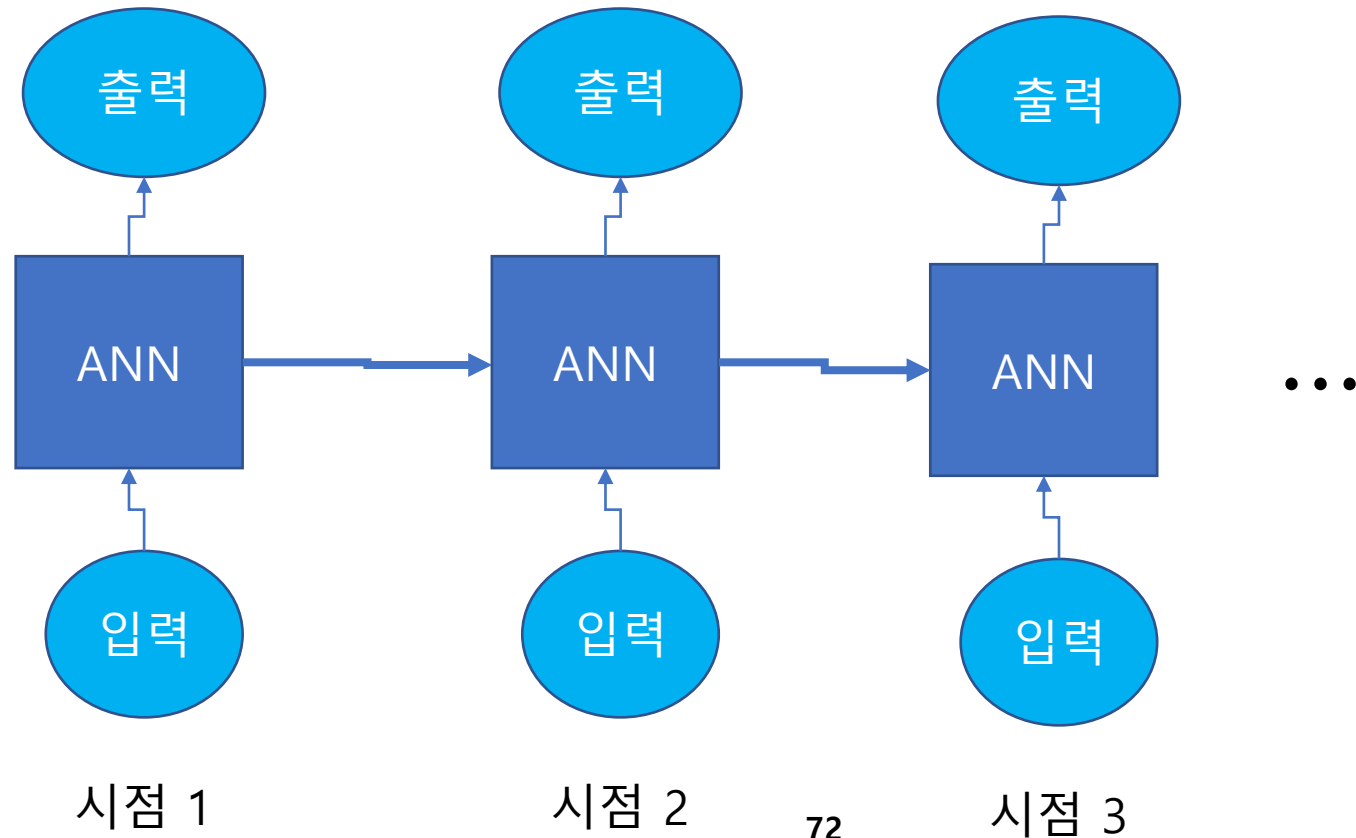


RNN

- 현재 정보와 이전 정보의 Gap
- 현재의 계산에 최근의 정보만 필요할 수 있음: 다음 단어 예측 시 이전 단어 기반
 - “the birds are in the sky” : the는 앞에 in을 사용하여, are는 앞의 birds를 사용하여...
 - 현재의 정보와 이전 정보의 gap은 작음
 - 학습 시 이전 단계에 대한 Dependency가 많은 자연어 처리에 잘 활용

- 순환신경망(Recurrent Neural Networks)

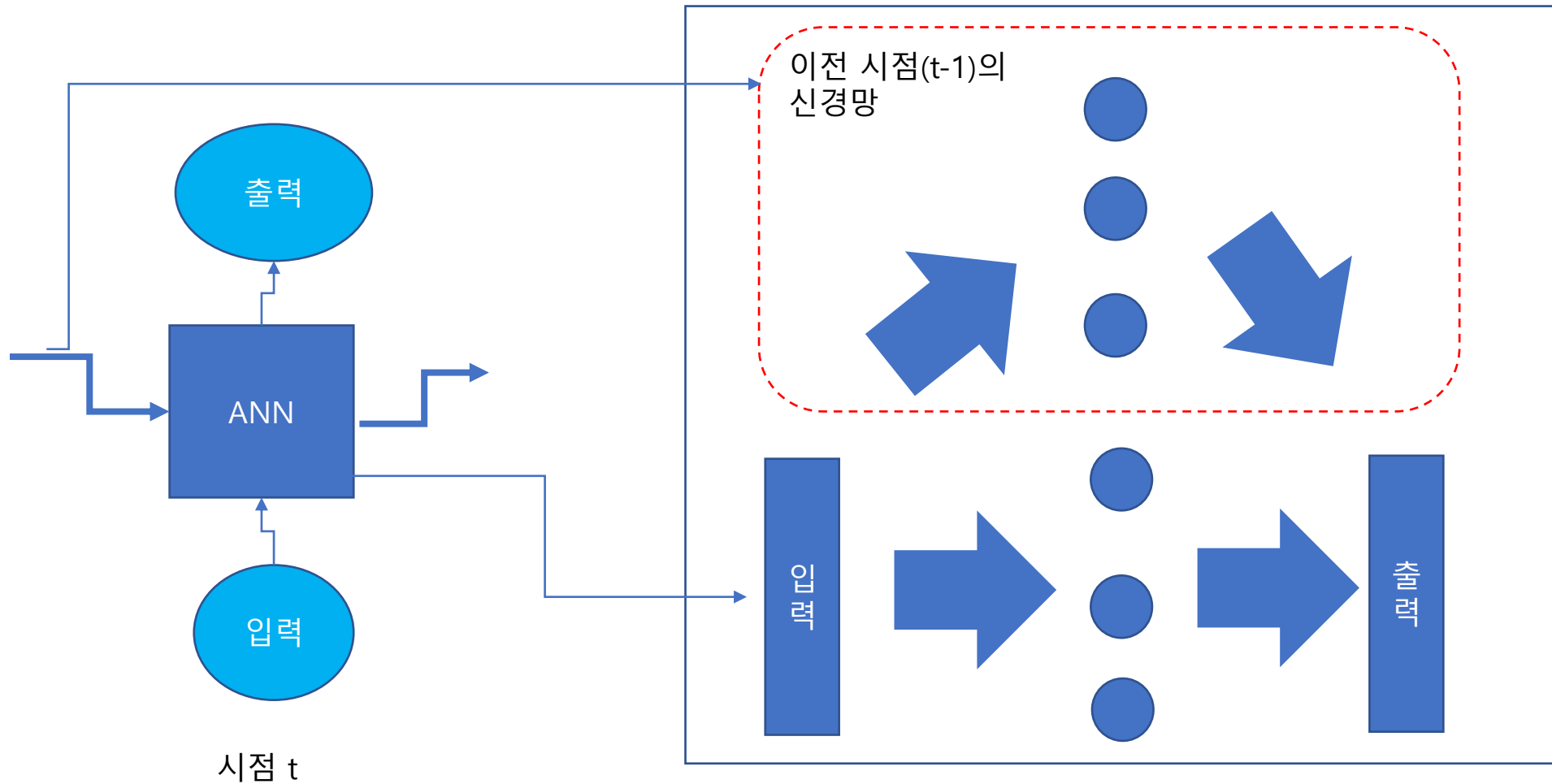
- Recurrent neural networks은 신경망 내에서 loop가 있고, 정보를 보존하게 함
- Loop?



VIII. RNN 소개

- 순환신경망(Recurrent Neural Networks)

- 이전 단계의 정보가 현재 단계의 계산에 활용



VIII. RNN 소개

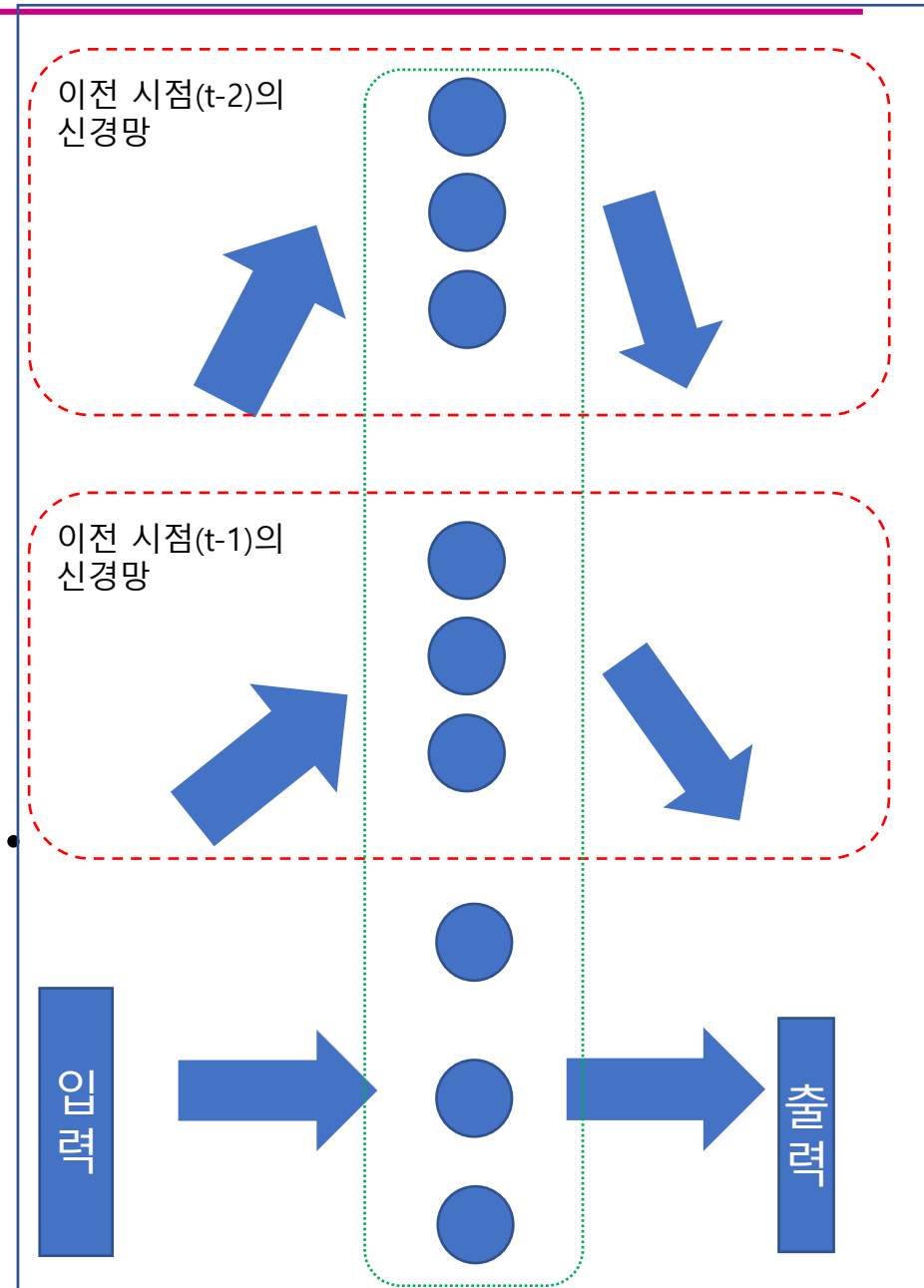
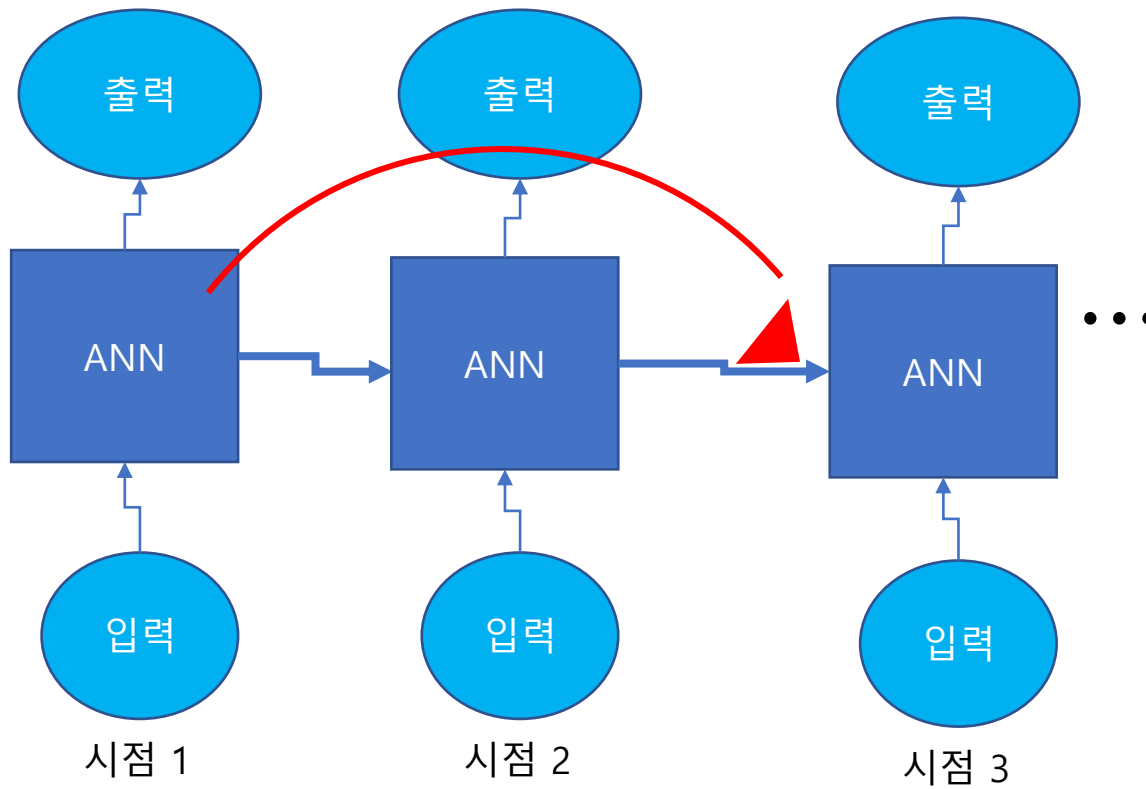
➤ LSTM?

➤ gap이 큰 경우? "long-term dependencies"

➤ RNN의 한 종류인 LSTM을 활용

➤ LSTM(Long Short Term Memory) 신경망

- LSTM은 long-term dependency를 해결, 긴 기간 동안 정보를 기억하는 방식
- 이전 정보와 현재 정보 연산

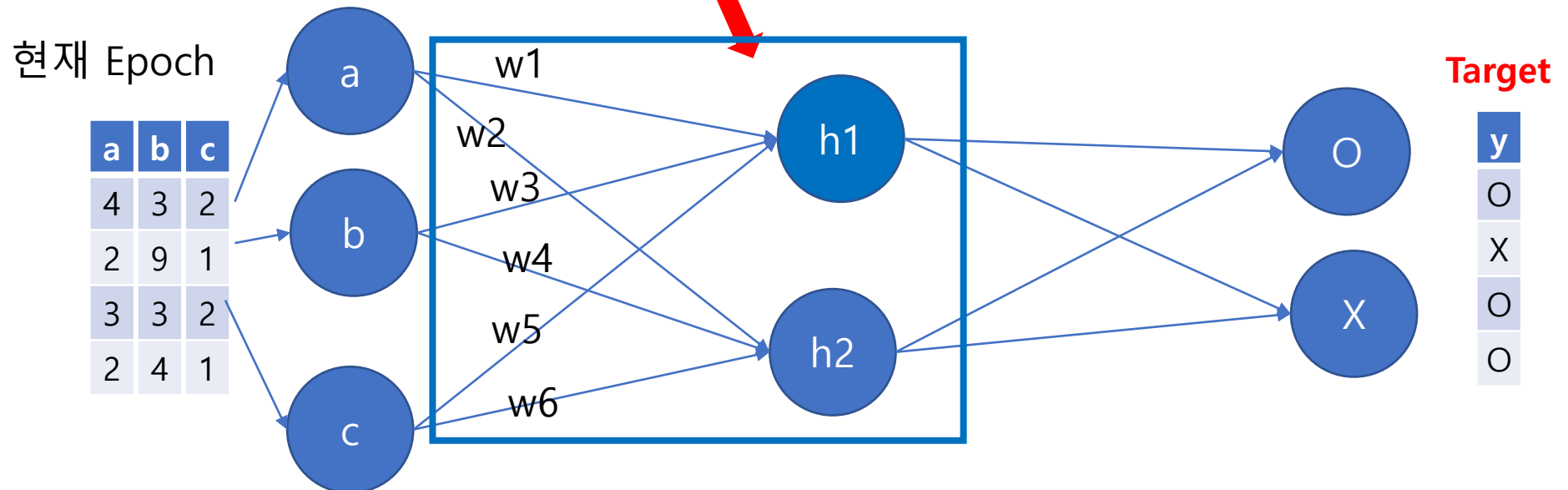
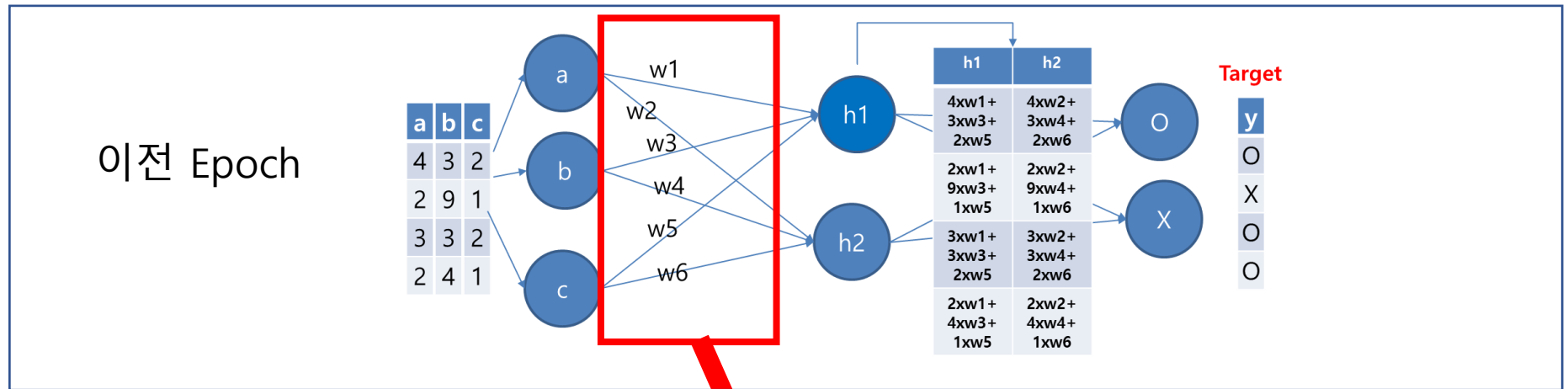


VIII. RNN 소개-특징

1. Epoch별 인공신경망의 학습: 인공신경망의 최적 가중치를 찾는 학습은 여러 차례 Epoch를 거치게 됨
2. 순환신경망(Recurrent Neural Network) 개요: RNN은 인공신경망의 학습에서 이전 Epoch의 가중치를 현재 Epoch의 학습에 활용하는 특징을 가짐

VIII. RNN 소개-특징

이전 Epoch의 가중치를 사용하는 ANN?



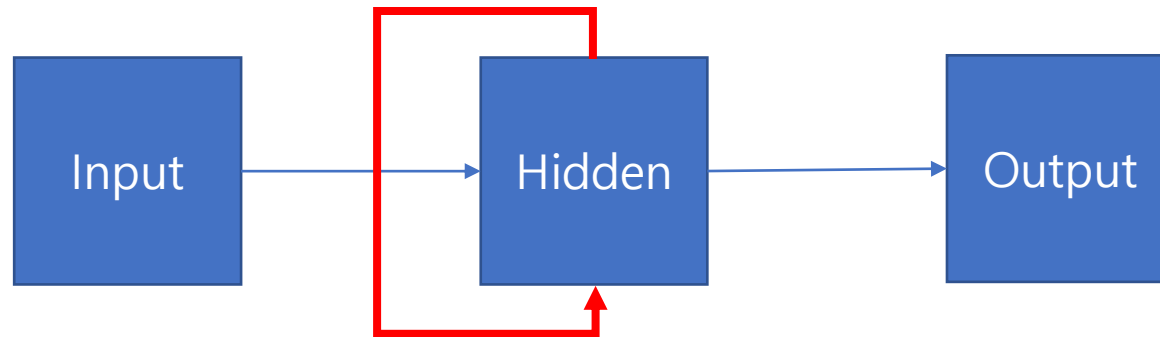
VIII. RNN 소개-특징

순환 신경망: 단일 계층을 반복적으로 계산, 체인구조의 연결

Recurrent Neural Network

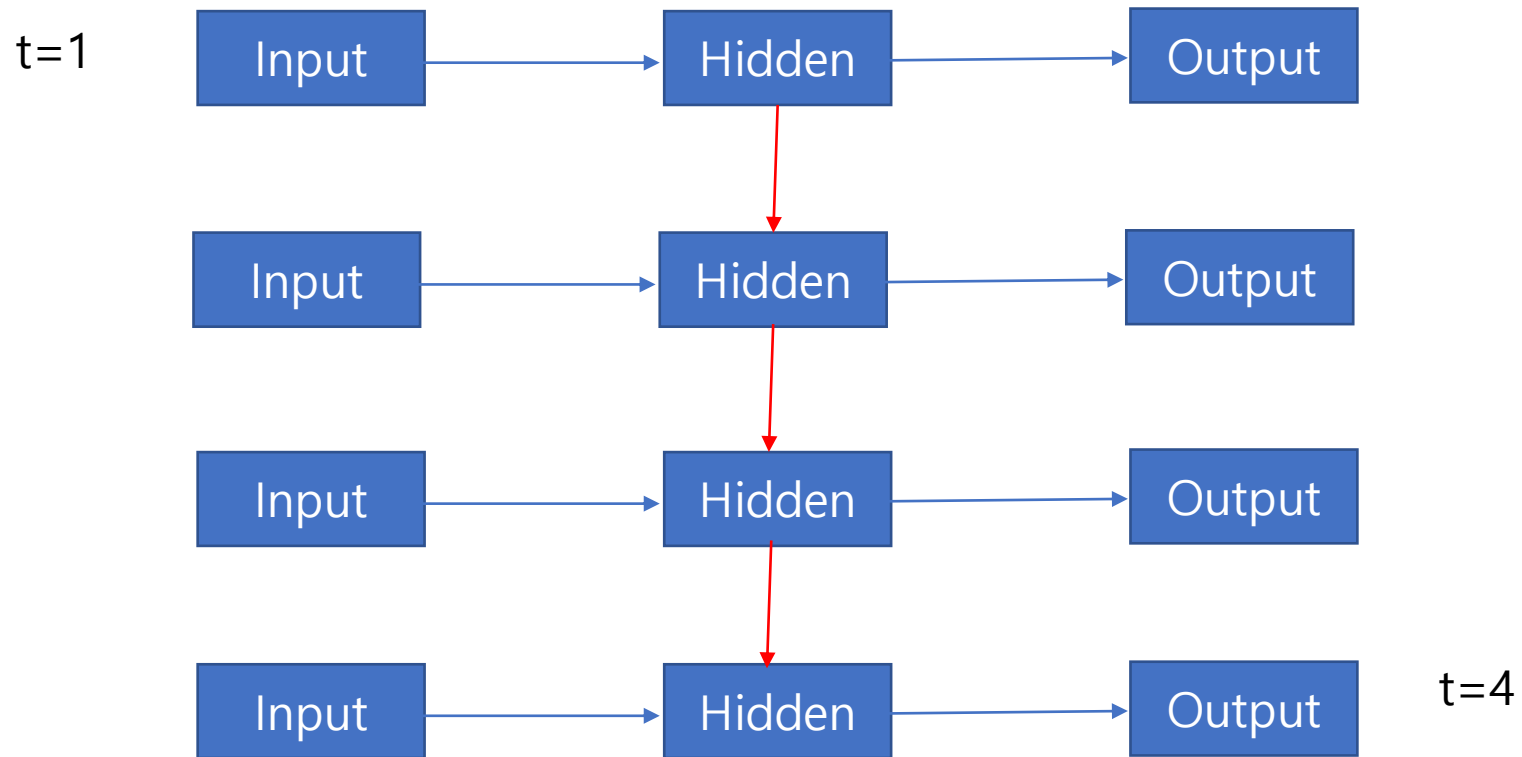
- 입력데이터의 순서적인 측면을 고려
- Sequence를 나타내는 입력데이터에 유리!
- 예: 시계열, 문장 등 자연어, 손글씨, 음성 신호, 센서 데이터 등

순환 신경망 Recurrent Neural Network



메모리셀(Memory Cell)

순환 신경망 Recurrent Neural Network



Memory!

RNN: Sequence Model

입력과 출력을 시퀀스 단위로 처리!

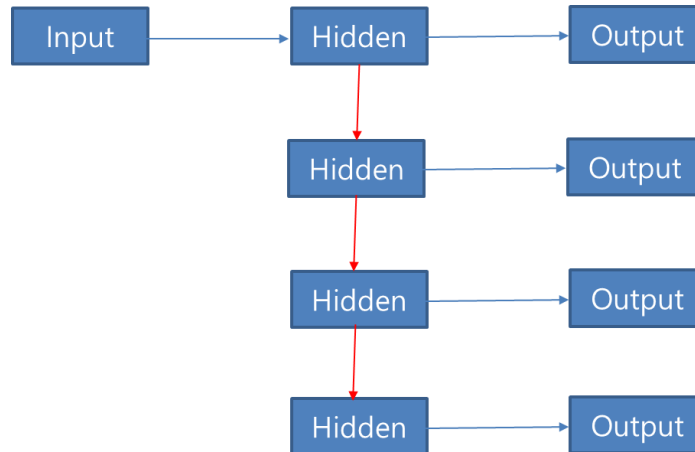
예: 한글 문장이 입력, 번역 결과인 영문 문장이 출력인 시퀀스!

- Long Term Dependency: 일반 RNN이 갖는 단점으로, RNN 학습 시 참고해야 하는 입력과 현재 출력의 위치 차이(time step 차이)가 큰 경우 연결이 어려워지고 성능이 저하
- 텍스트와 RNN: 텍스트는 순차적인 특성을 갖는 데이터여서 RNN을 통한 모델링에 많이 적용되고 있음
- LSTM과 언어모델: RNN의 Long Term Dependency를 개선한 LSTM과 RNN 언어모델등이 많이 활용됨

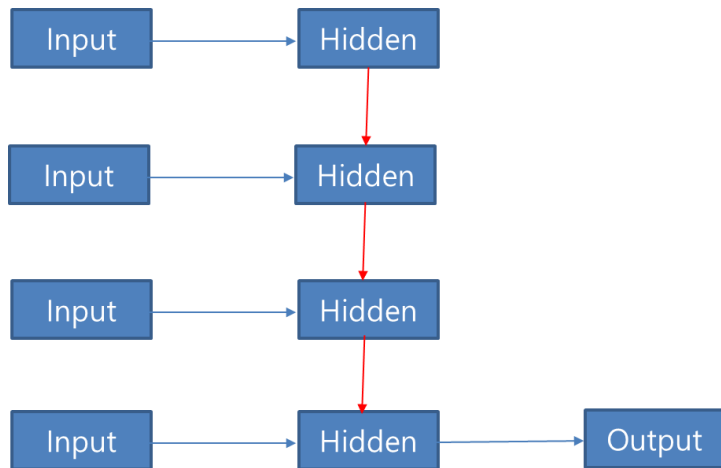
VIII. RNN 소개-특징

RNN: One to Many, Many to One, Many to Many

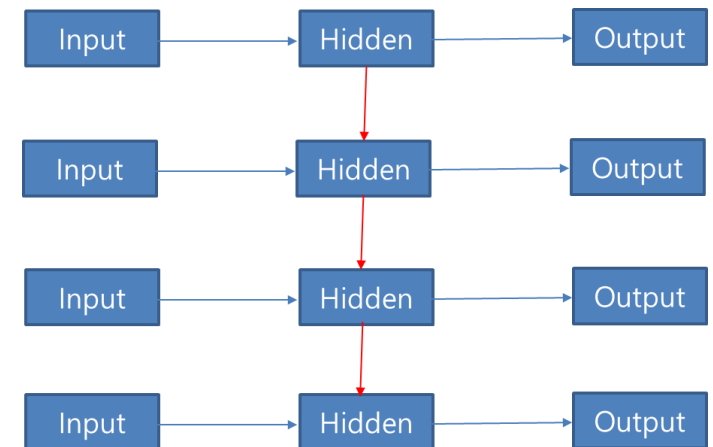
One to Many



Many to One



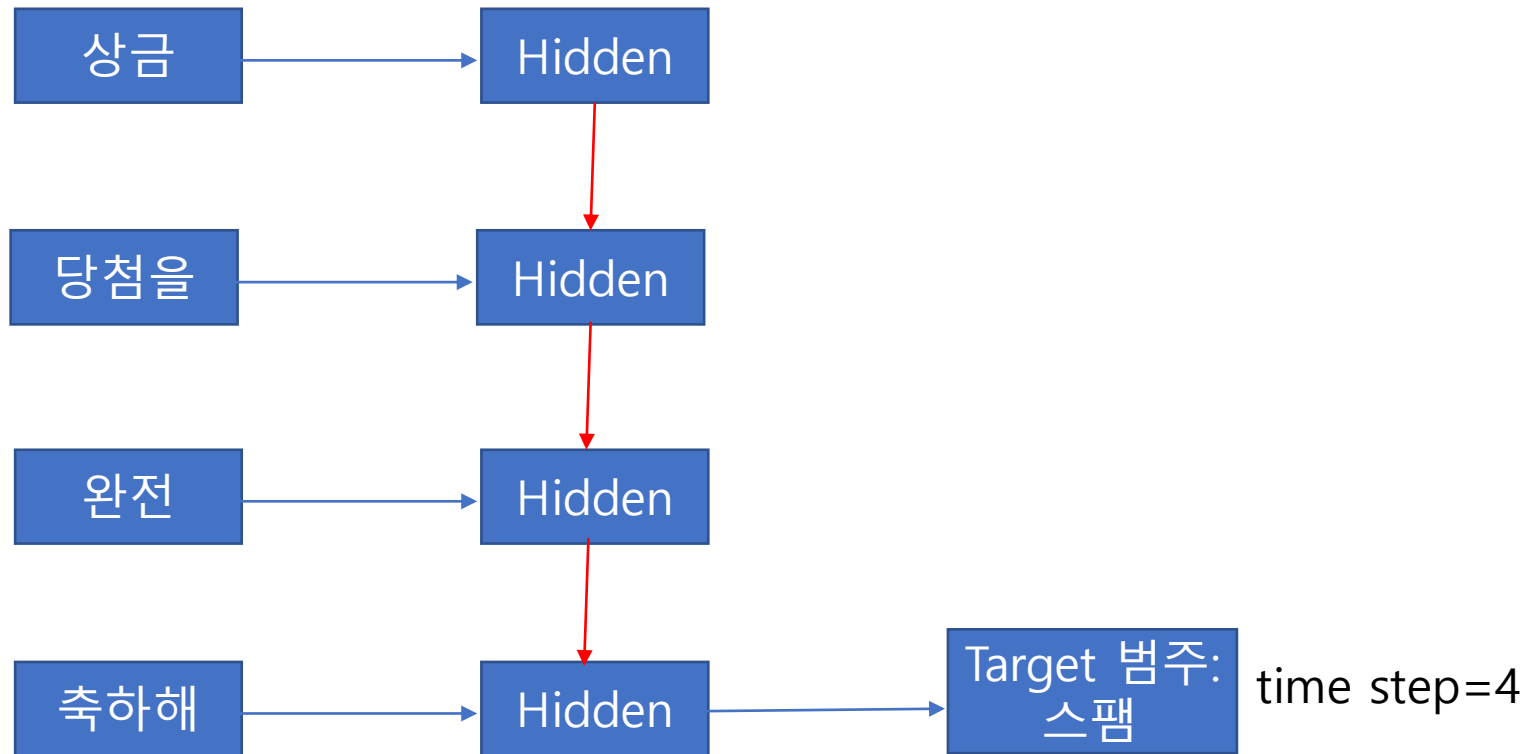
Many to Many



VIII. RNN 소개-특징

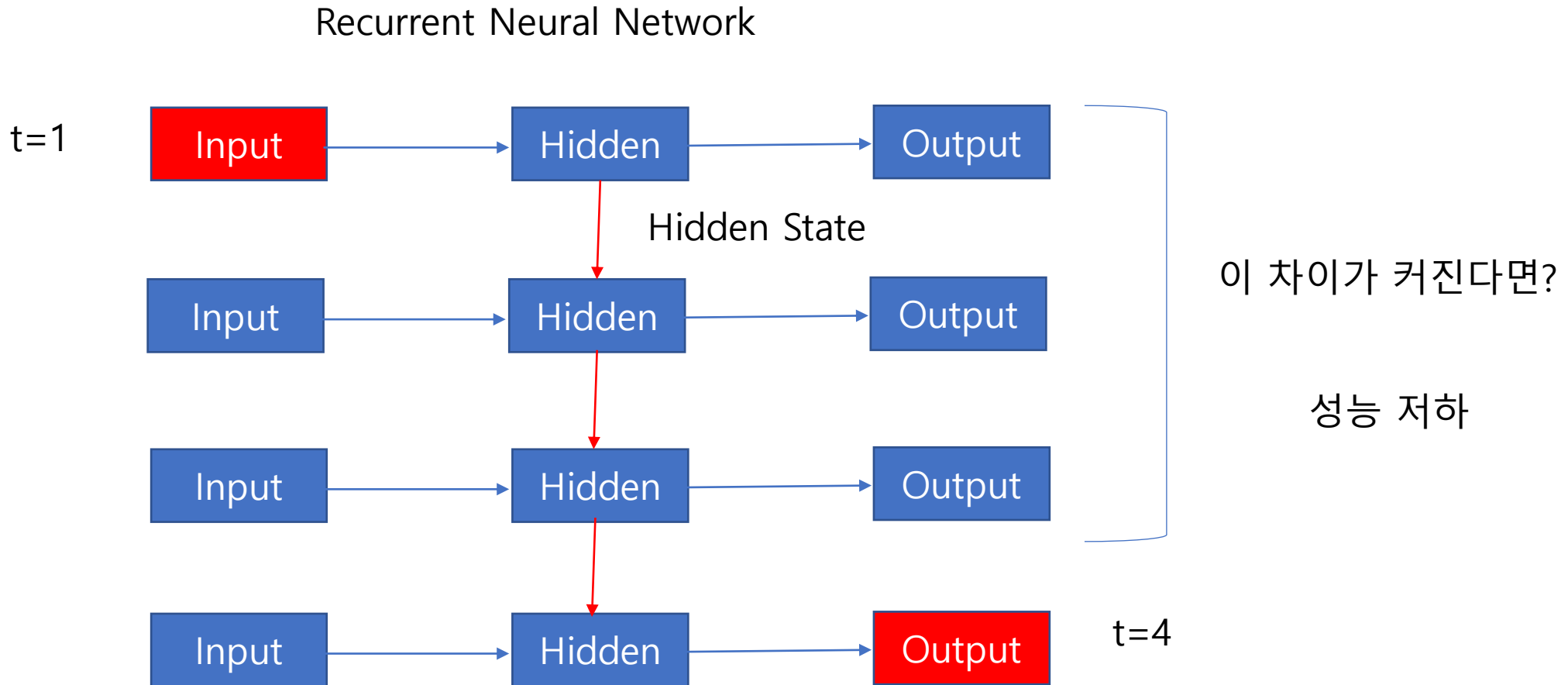
RNN과 텍스트 분류

Recurrent Neural Network (Many to One)



VIII. RNN 소개-특징

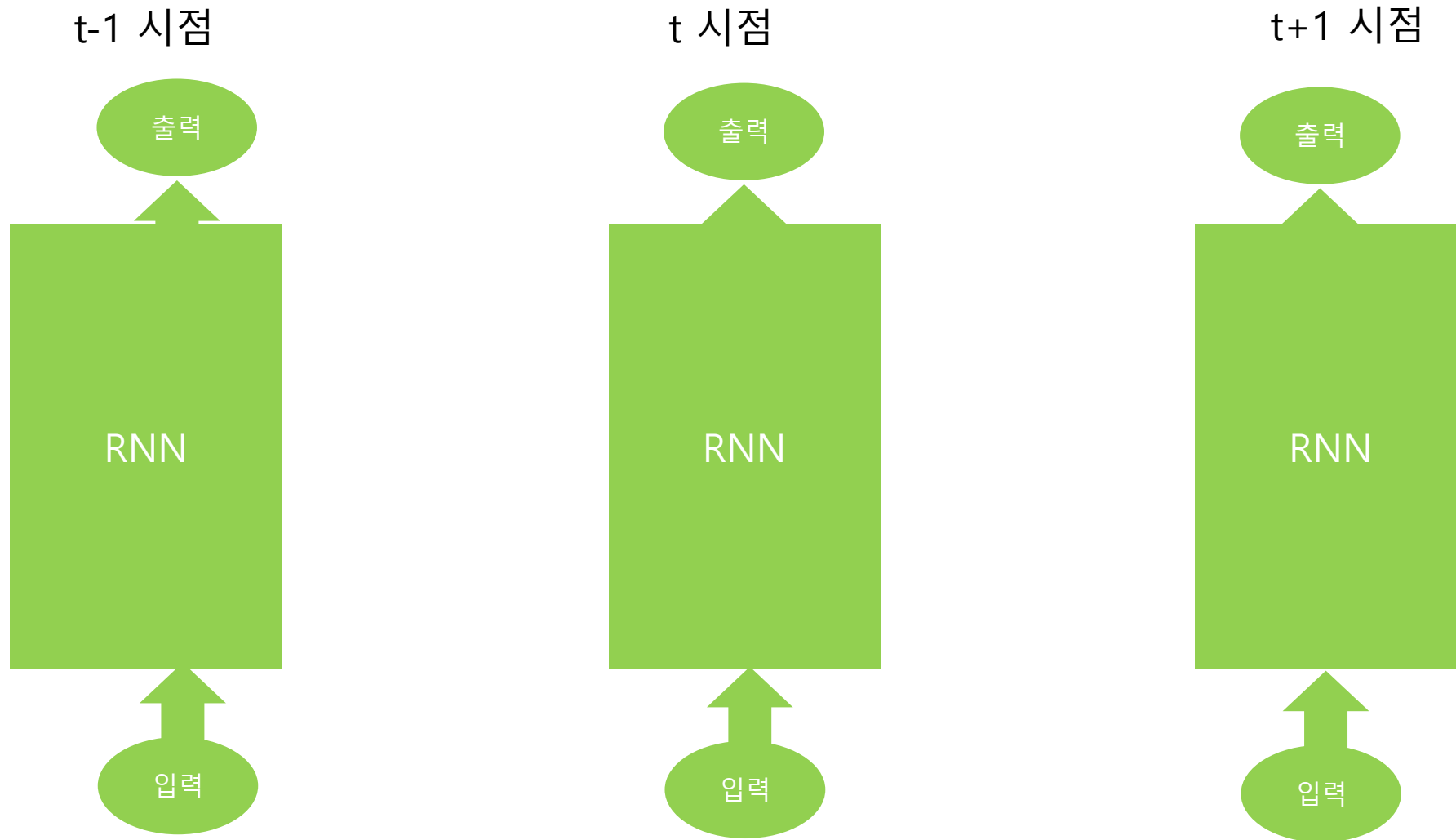
RNN의 장기의존성(Long Term Dependency)



VIII. RNN 소개-특징

RNN과 장기 의존성

시차가 커질수록 Vanishing Gradient!



VIII. RNN 소개-특징

기본 RNN: Vanilla RNN

Vs

Output이 긴 시점 전의 입력에 의존하는 경우, RNN 성능의 이슈

LSTM: Long Short Term Memory

은닉층의 메모리 셀에 입력/망각/출력 게이트를 추가
효율적으로 메모리를 관리
불필요한 메모리는 삭제

“Hidden State(메모리 셀에서 다음 시점에 정보를 제공)의 방식이 복잡,
Cell State의 도입으로 긴 시퀀스 처리 효과적”

QnA