DAY 2

XOR 문제
Neural Network
역전파 알고리즘
NN Binary Classification
NN Regression

0. Day1 무엇을 배웠나요?

1. Deep Learning Overview

1. Perceptron
2. Neural Network
3. Convolution Neural Network

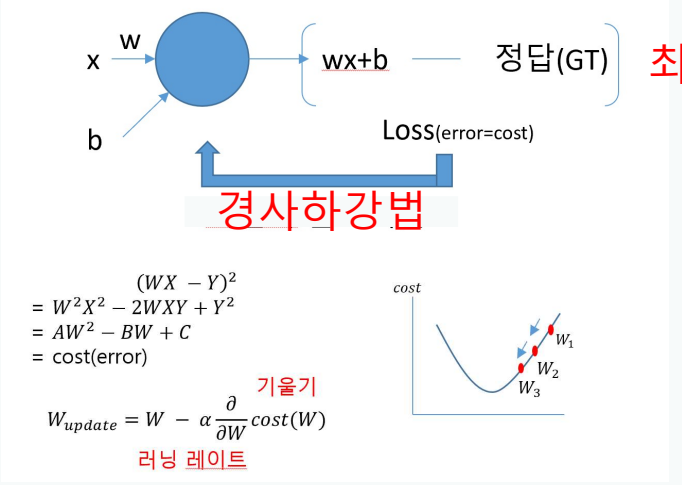
2. DeepLearning FrameWork

1. Tensorflow1.x : 그래프 기반
2. Tensorflow2.x : 즉시 모드

3. Mean Square Error Loss 함수

1. Loss 함수란 perceptron의 task를 지정
2. 정답과의 차이를 나타냄

4. Perceptron + 경사하강법



최종 얻는 값은 w 와 b

5. Regression

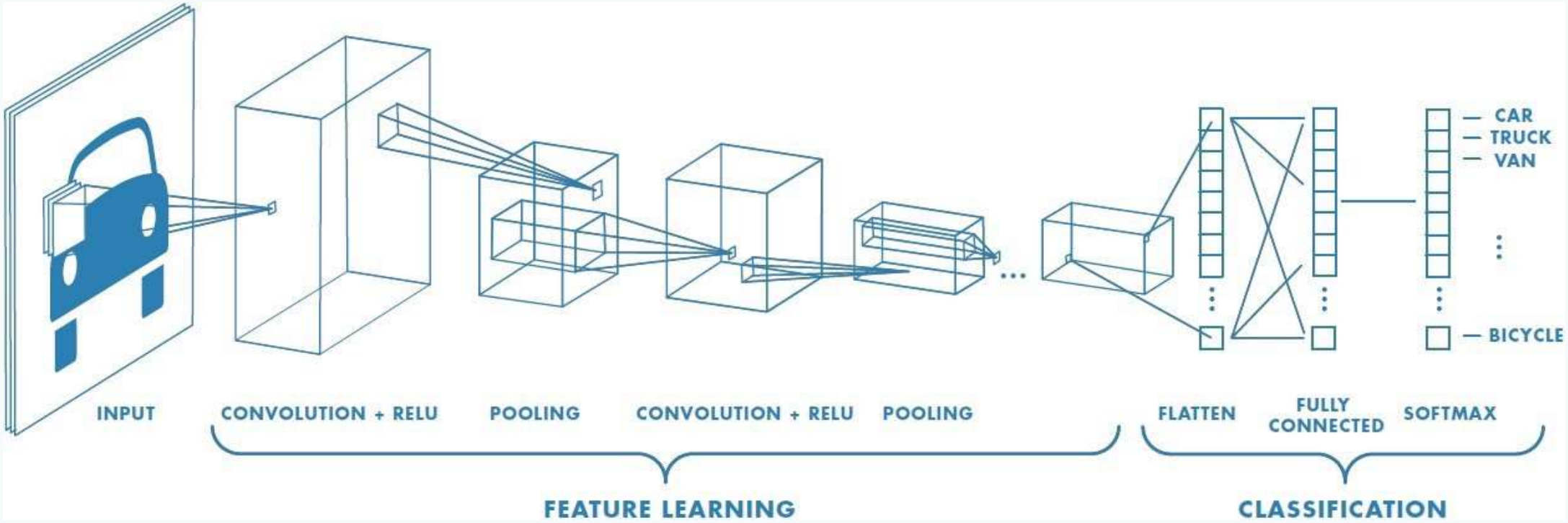
1. 값 예측 (ex : 여러 변수를 받아 하나의 값을 예측)
2. Loss : Mean Square Error

6. Classification(logistic regression)

1. Sigmoid 함수를 통해서 분류 학습
2. Loos : Cross Entropy

0. Day1 무엇을 배웠나요?

Deep Learning

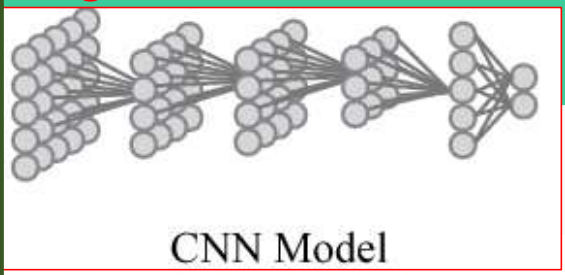


출처 : https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

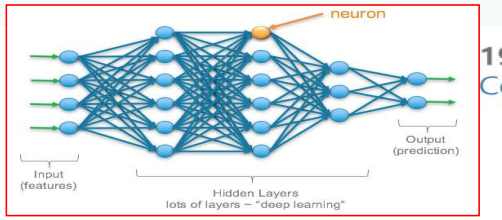
5. Day1 무엇을 배웠나요?

Deep Learning
신경망의 70년

Day4 학습 목표
regression, classification



Day2 학습 목표
regression, classification



1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

1998년
Convolution Neural Network (CNN)

얀 르쿤

2006년
Deep Learning

제프리 힌튼

2016년
DeepMind AlphaGo

Day6 학습 목표 : 심화

2012년
AlexNet
IMAGENET

Day3 학습 목표

1차 암흑기
(1974-1980)

2차 암흑기
(1987-1993)

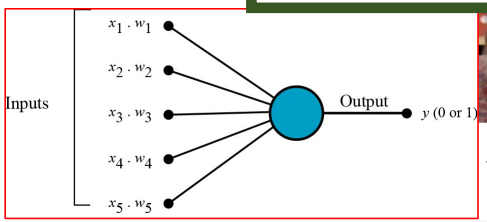
19C-20C 초반
생체 신경망

이반 파블로프

Day1의 학습 목표

1958년
Perceptron

로젠블랫



regression, classification

1982년
Recurrent Neural Network (RNN)

홉필드

Day5 학습 목표

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

2014년
Generative Adversarial Network (GAN)

이안 굿펠로우

진도상황에 따라 변경될 수 있음

0. Day1 무엇을 배웠나요?

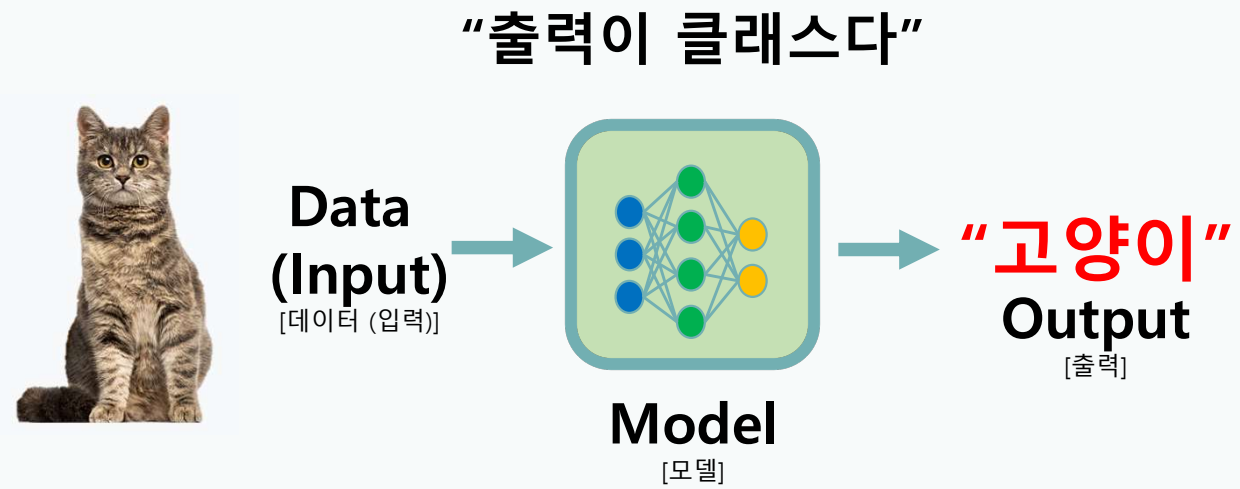
Supervised Learning
- Classification
- Regression
- etc.

Unsupervised Learning
- Clustering
- Anomaly Detection
- etc.

Semi-supervised Learning

기계학습 (Machine Learning) – Supervised

Classification (분류)



0. 어제 무엇을 배웠나요?

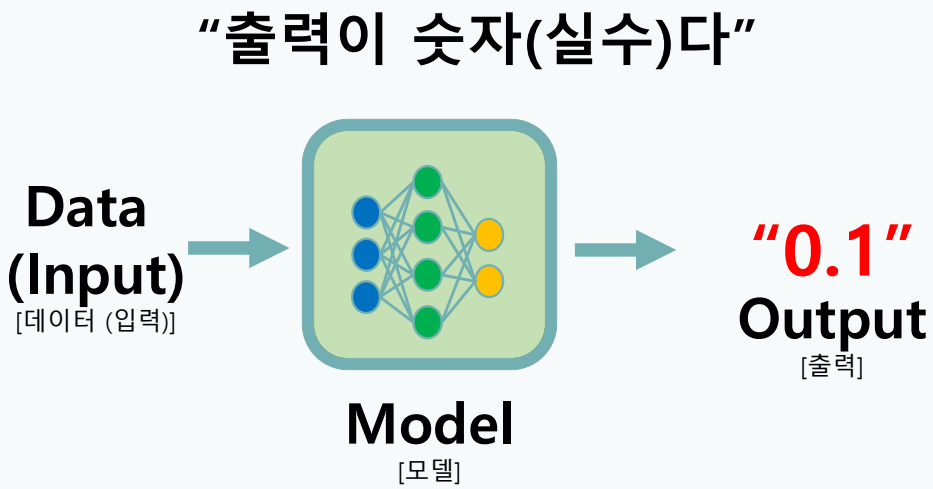
Supervised Learning
- Classification
- Regression
- etc.

Unsupervised Learning
- Clustering
- Anomaly Detection
- etc.

Semi-supervised Learning

기계학습 (Machine Learning) – Supervised

Regression (회귀)



0. 어제 무엇을 배웠나요?

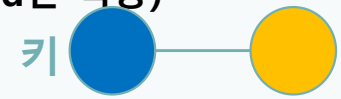
1. Linear Regression

$$\mathcal{L}_{mse} = \frac{1}{n}\sum_{i=1}^{n}(H(x_i) - y_i)^2$$

-> 문제점 : binary classification 에서 사용할 시 풀리지 않을때가 존재함.

---

2. Logistic Linear Regression (Sigmoid만 적용)



$$\mathcal{L}_{mse} = \frac{1}{n}\sum_{i=1}^{n}(\sigma(H(x_i)) - y_i)^2$$

-> 해결점 : 출력할때 sigmoid 함수를 통해서 출력 0~1사이 값 이며 커브피팅 효과
-> 문제점 : 기존 mse loss를 사용하면 두가지 문제가 있음 1. local min에 빠질 확률이 크고,
2. 시그모이드 미분값으로 w 학습이 잘안되거나 느림

---

3. Logistic Regression(Classification)

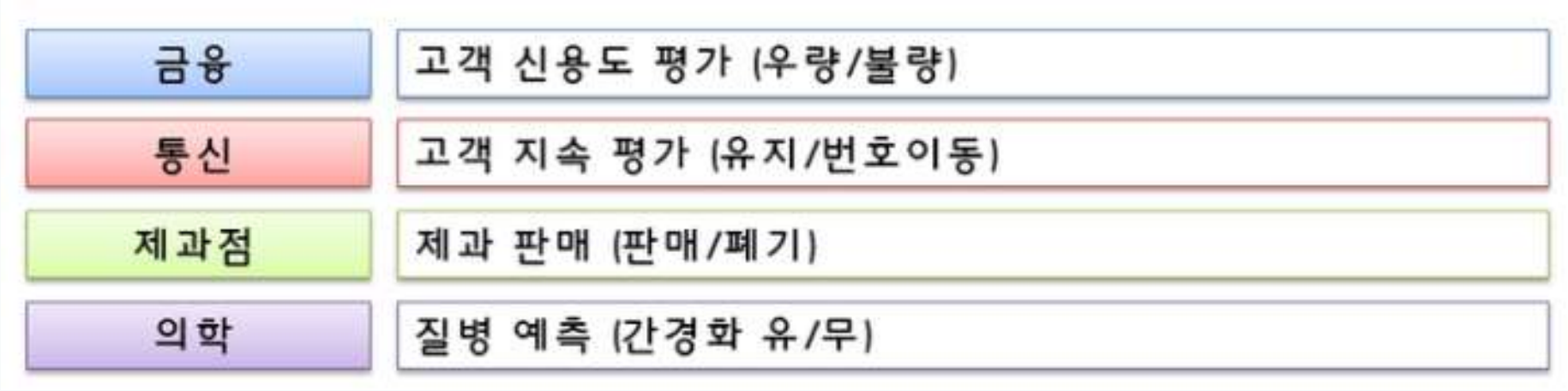
$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

-> 해결점 : mse loss 함수 대신 binary_crossentropy loss

0. 어제 무엇을 배웠나요?

로지스틱 회귀

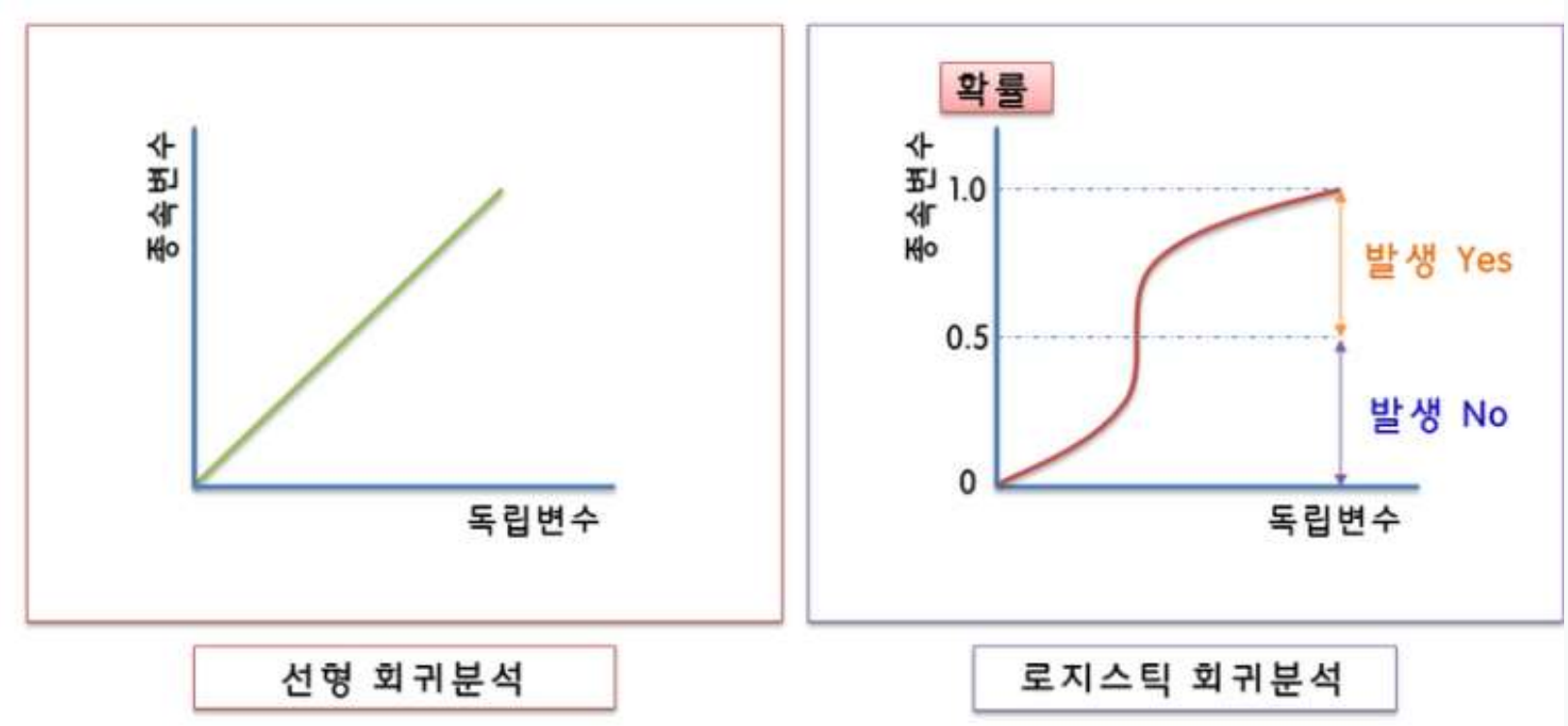
- 두개의 값(0,1) 만 가지는 종속 변수와 독립변수들 간의 인과관계를 로지스틱 함수를 이용하여 추정하는 통계기법
- 어떤 사전이 발생할지에 대한 직접 예측이 아니라 그 사건이 발생할 확률을 예측하는 것



독립변수 : 연구자가 의도적으로 변화시키는 변수
종속변수 : 연구자가 독립변수의 변화에 따라 어떻게 변하는지 알고 싶어하는 변수

0. 어제 무엇을 배웠나요?

로지스틱 회귀 모형



0. 어제 무엇을 배웠나요?

로지스틱 회귀 모형

$$\text{odds} = \frac{\text{사건 발생함}}{\text{사건 발생안함}} = \frac{p}{1-p} = e^{\beta_0 + \beta_1 X_1 + \cdots\cdots + \beta_i X_i}$$

지수함수로 표현 가능

- 오즈비는 확률과 관련된 의미로 p가 주어졌을 때, 사건이 발생할 확률이 발생하지 않을 확률에 비해 몇배 더 높은가 의미
- 예) 종속변수가 1=True, 0=False 라고 할때, P가 0.8 이라면 오즈비는 (0.8/(1-0.8))=4가 되고 이것은 성공이 될 확률이 실패가 될 확률보다 4배 높다는 의미

0. 어제 무엇을 배웠나요?

로지스틱 회귀 모형

$$\text{odds} = \frac{\text{사건 발생함}}{\text{사건 발생안함}} = \frac{p}{1-p} = e^{\beta_0 + \beta_1 X_1 + \cdots\cdots + \beta_i X_i}$$

자연지수 e를 ln으로 치환

$$\text{odds} = \ln\left(\frac{p}{1-p}\right) = \ln\left(e^{\beta_0 + \beta_1 X_1 + \cdots\cdots + \beta_i X_i}\right)$$

$$\text{OR (Y)} = \beta_0 + \beta_1 X_1 + \cdots\cdots + \beta_i X_i$$

다중 회귀분석의 회귀식과 동일

0. 어제 무엇을 배웠나요?

로지스틱 회귀 모형 loss 함수

로지시트 회귀모형

$$\text{cost} = (\sigma(\hat{y}) - y)^2$$

미분하면 $2\sigma(\hat{y})(\sigma(\hat{y}))'$

즉, 미분하면 1과 0에 수렴하는 부분은 굉장히 작은 값으로 나타내어 진다. 그러므로 학습이 느려짐.

$$\text{cost} = (\sigma(\hat{y}) - y)^2$$

$$\text{cost} = -[y ln\sigma(\hat{y}) + (1 - \text{y})\ln(1 - \sigma(\hat{y}))]$$

성질

1. 예측 값인 $\sigma(\hat{y})$은 0~1사이 값. 그러므로 항상 음의 값만 존재
2. $\sigma(\hat{y})$이 y에 가까운 값일 수록 0에 가까워 짐.

0. 어제 무엇을 배웠나요?

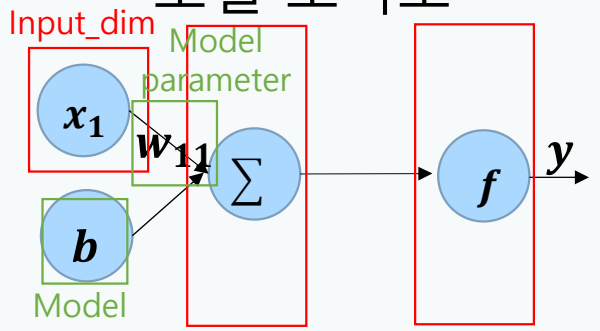
1. Data 생성 or DATA 읽기
2. DATA의 변수에 따른 차원 확인
3. PERCEPTRON의 입력 차원 맞추기
4. PERCEPTRON 생성
5. PERCEPTRON LOSS 함수 지정
6. 경사하강법 (LOSS 함수의 MIN을 찾기 위한 최적화 방법 선택)
7. 학습완료에 따른 결과 보기.

0. 어제 무엇을 배웠나요?

Single neuron model : perception logistic regression

```python
## data 선언
x_data = [[2.], [4.], [6.], [8.], [10.], [12.], [14.]]
y_data = [[0.], [0.], [0.], [1.], [1.], [1.], [1.]]
test_data= [[3.]]
test_data2= [[7.]]
test_data3= [[17.]]
test_data4= [[60.]]

## tf.keras를 활용한 perceptron 모델 구현.
model = tf.keras.Sequential() ## 모델 선언
model.add(tf.keras.layers.Dense(1, input_dim=1, activation='sigmoid')) # 선언된 모델에 add를통해 쌓아감.은닉층
model.summary()

# 모델 loss, 학습 방법 결정하기
optimizer=tf.keras.optimizers.SGD(learning_rate=0.007) ### 경사 하강법으로 global min 에 찾아가는 최적화 방법 선언.
loss=tf.keras.losses.binary_crossentropy
metrics=tf.keras.metrics.binary_accuracy

# 모델 컴파일하기 - 모델 및 loss 등 구조화한 모델을 컴퓨터가 동작 할수 있도록 변환
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=5000, batch_size=9)

# 결과를 출력합니다.
print(" test data [3.] 예측 값 : ", model.predict(test_data))
print(" test data [7.] 예측 값 : ", model.predict(test_data2))
print(" test data [17.] 예측 값 : ", model.predict(test_data3))
print(" test data [60.] 예측 값 : ", model.predict(test_data4))
```

모델 도식도



모델 summury

```
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
Train on 7 samples
Epoch 1/1000
```

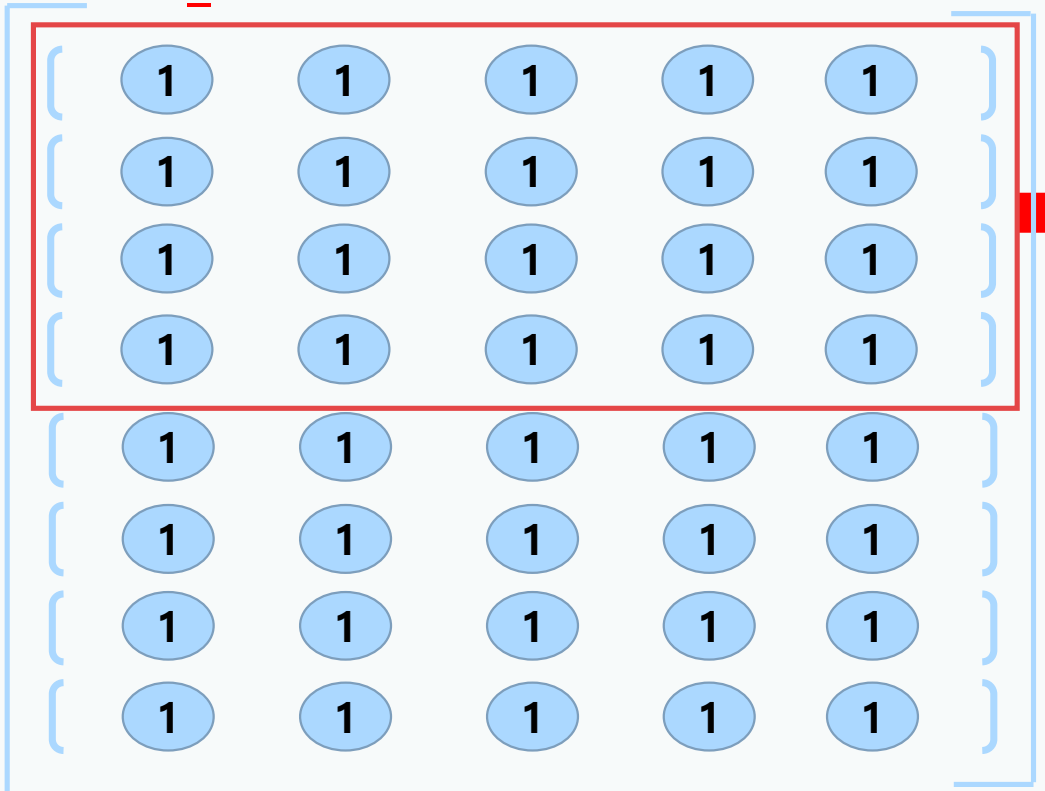
Batch_size

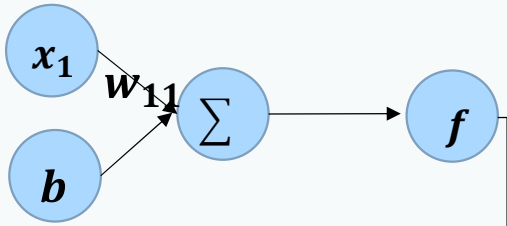
W11, b

practice : P_01_08_tensor_cross_entropy.ipynb

0. 어제 무엇을 배웠나요?

데이터 분리 학습 과정 살펴보기





모델 학습을 위한 loss

$\hat{y}$ y

loss 계산해서
모델 업데이트

$\hat{y}$ y

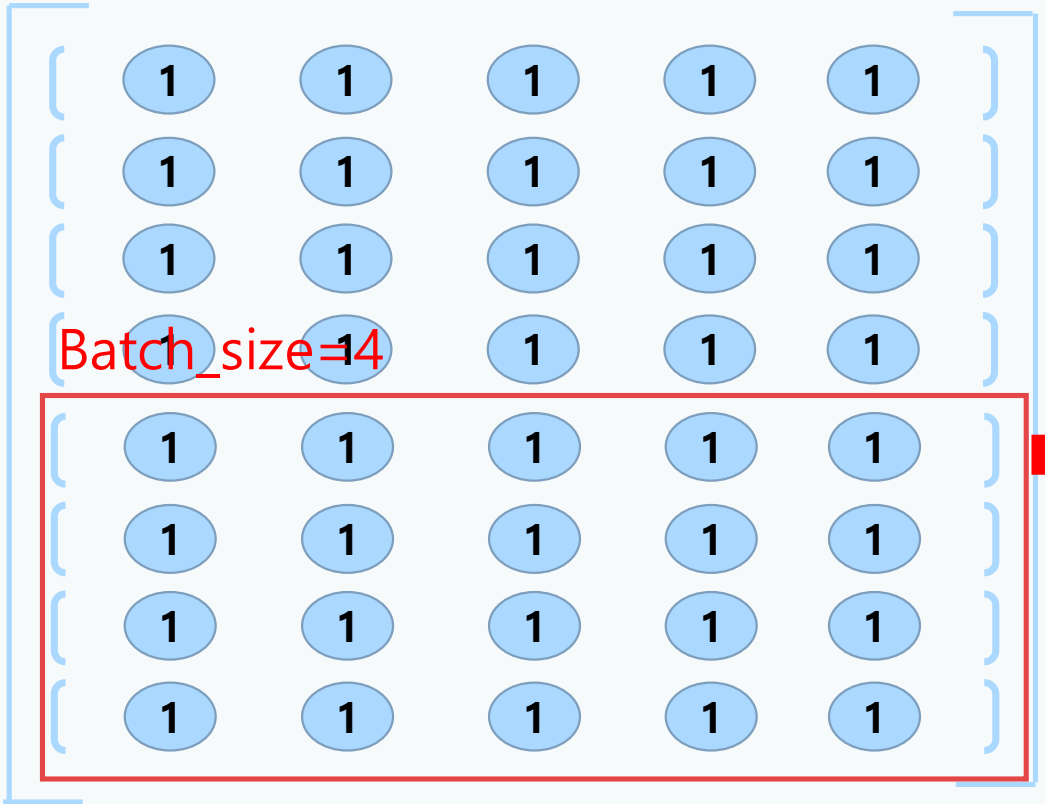
Metric 계산

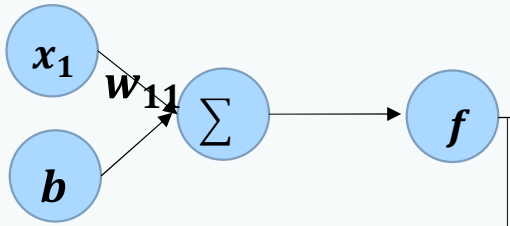
모델의 성능을 평가
하기 위한 지표

```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

0. 어제 무엇을 배웠나요?

데이터 분리 학습 과정 살펴보기





모델 학습을 위한 loss

$\hat{y}$ y

loss 계산해서
모델 업데이트

$\hat{y}$ y

Metric 계산

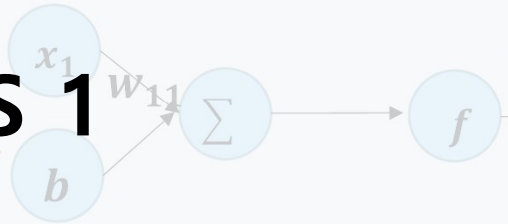
모델의 성능을 평가하기 위한 지표

```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

0. 어제 무엇을 배웠나요?

데이터 분리 학습 과정 살펴보기

EPOCHS 1



모델 학습을 위한 loss

loss 계산해서 모델 업데이트

Metric 계산

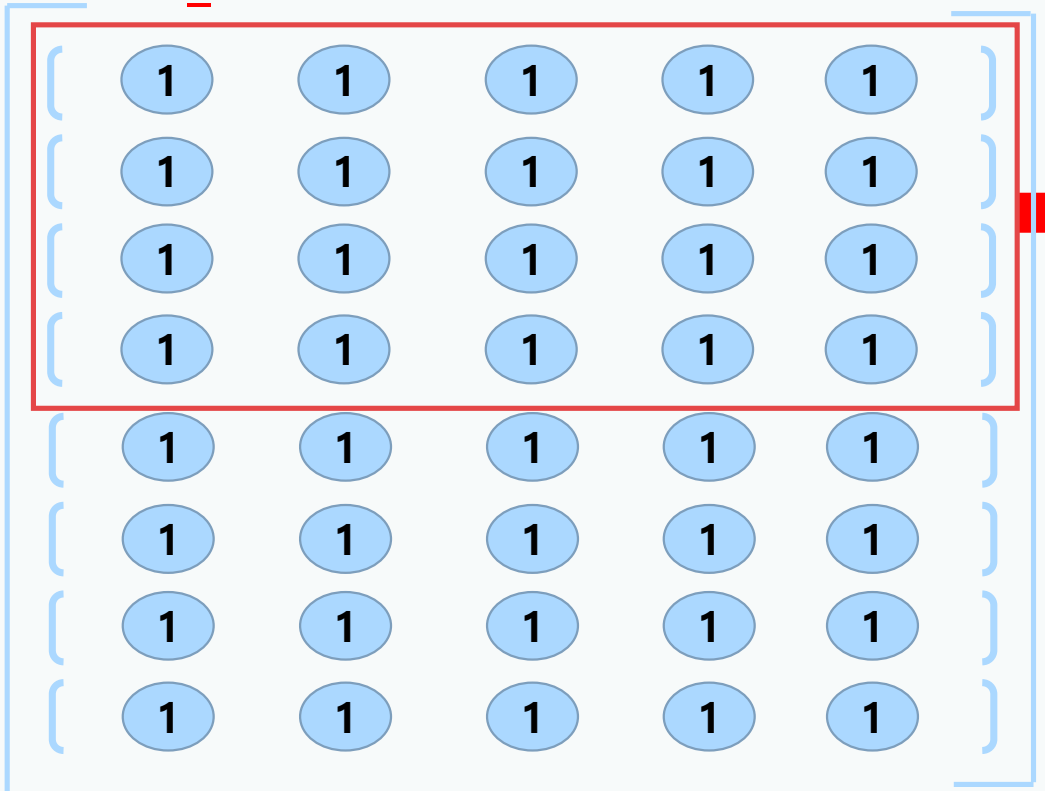
모델의 성능을 평가하기 위한 지표

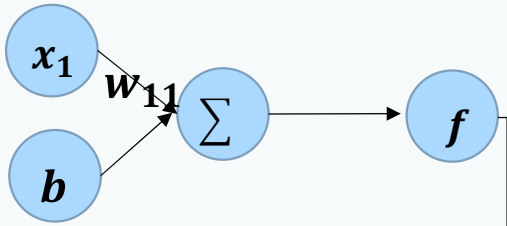
```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

0. 어제 무엇을 배웠나요?

데이터 분리 학습 과정 살펴보기

Batch_size=4





모델 학습을 위한 loss

$\hat{y}$ y

loss 계산해서
모델 업데이트

$\hat{y}$ y

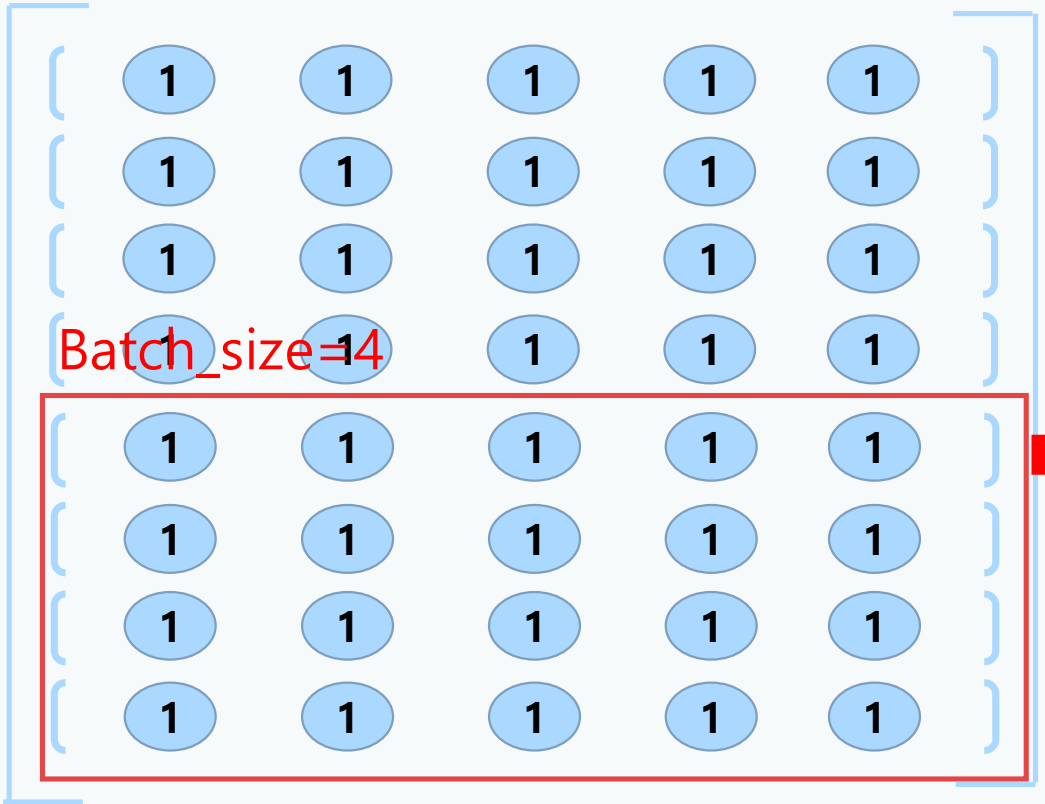
Metric 계산

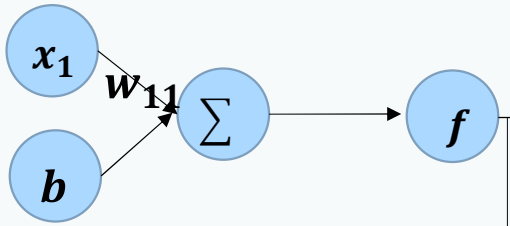
모델의 성능을 평가
하기 위한 지표

```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

0. 어제 무엇을 배웠나요?

데이터 분리 학습 과정 살펴보기





모델 학습을 위한 loss

$\hat{y}$ y

loss 계산해서
모델 업데이트

$\hat{y}$ y

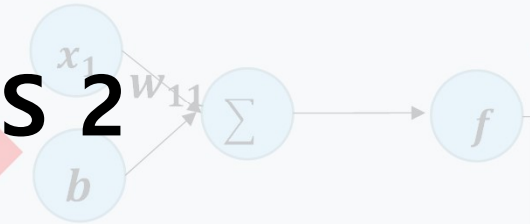
Metric 계산

모델의 성능을 평가하기 위한 지표

```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

0. 어제 무엇을 배웠나요?

데이터 분리 학습 과정 살펴보기



모델 학습을 위한 loss

loss 계산해서 모델 업데이트

Metric 계산

모델의 성능을 평가하기 위한 지표

```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

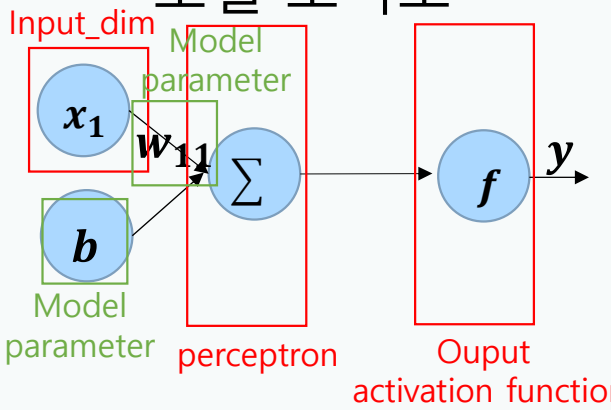
0. 어제 무엇을 배웠나요?

Single neuron model : perception logistic regression

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
Train on 7 samples
Epoch 1/1000
```

Batch_size

모델 도식도




```
x_data = [[2.], [4.], [6.], [8.], [10.], [12.], [14.]]
y_data = [[0.], [0.], [0.], [1.], [1.], [1.], [1.]]
```

```
x_data = [[2.], [4.], [6.], [8.], [10.], [12.], [14.]]
y_data = [[0.], [0.], [0.], [1.], [1.], [1.], [1.]]
```

데이터 입력 차원 : (4, 1)

Perceptron (node) (3, 1) Sigmoid

practice : P_01_08_tensor_cross_entropy.ipynb

0. 어제 무엇을 배웠나요?

Single neuron model : perception logistic regression

Activation + Cross Entropy loss 적용

공부 일수	합격 여부(y)
5	불합격 -> 0
30	불합격 -> 0
95	불합격 -> 0
100	불합격 -> 0
265	합격 -> 1
270	합격 -> 1
290	합격 -> 1
300	합격 -> 1
365	합격 -> 1

excersice : 01_08_tensor_cross_entropy.ipynb

0. 오늘 하루 동안 무엇을 하나요?

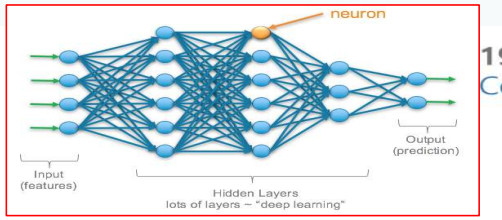
1. **XOR 문제**
   - AND GATE
   - OR GATE
   - XOR GATE
2. **Neural Network**
3. **역전파 알고리즘**
3. **Neural Network 실습**
   - NN Regression
   - Data normalization
   - Data 분리 하기
   - NN Binary classification

5. Perceptron Summary

Deep Learning
신경망의 70년

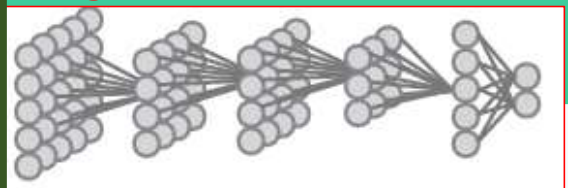
오늘의 학습 목표

regression, classification



Day4 학습 목표

regression, classification



1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

1998년
Convolution Neural Network (CNN)

얀 르쿤

2006년
Deep Learning

제프리 힌튼

2016년
DeepMind AlphaGo

Day6 학습 목표 : 심화

2012년
AlexNet
IMAGENET

Day3 학습 목표

1차 암흑기
(1974-1980)

2차 암흑기
(1987-1993)

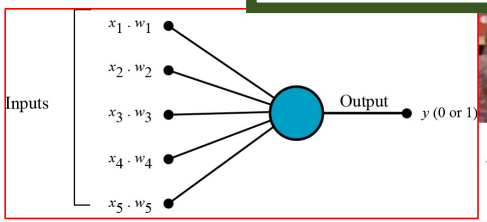
19C-20C 초반
생체 신경망

이반 파블로프

Day1 학습 목표

1958년
Perceptron

로젠블랫



regression, classification

1982년
Recurrent Neural Network (RNN)

홉필드

Day5 학습 목표

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

2014년
Generative Adversarial Network (GAN)

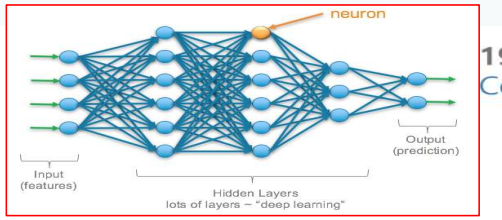
이안 굿펠로우

진도상황에 따라 변경될 수 있음

1. PERCEPTRON XOR 문제

오늘의 학습 목표

regression, classification



신경망의 70년

19C-20C 초반
생체 신경망

이반 파블로프

1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1956년
Perceptron

로젠블랫

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

1차 암흑기
(1974-1980)

1982년
Recurrent Neural Network (RNN)

존 홉필드

1998년
Convolution Neural Network (CNN)

얀 르쿤

2006년
Deep Learning

제프리 힌튼

2012년
AlexNet
IMAGENET

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

2014년
Generative Adversarial Network (GAN)

이안 굿펠로우

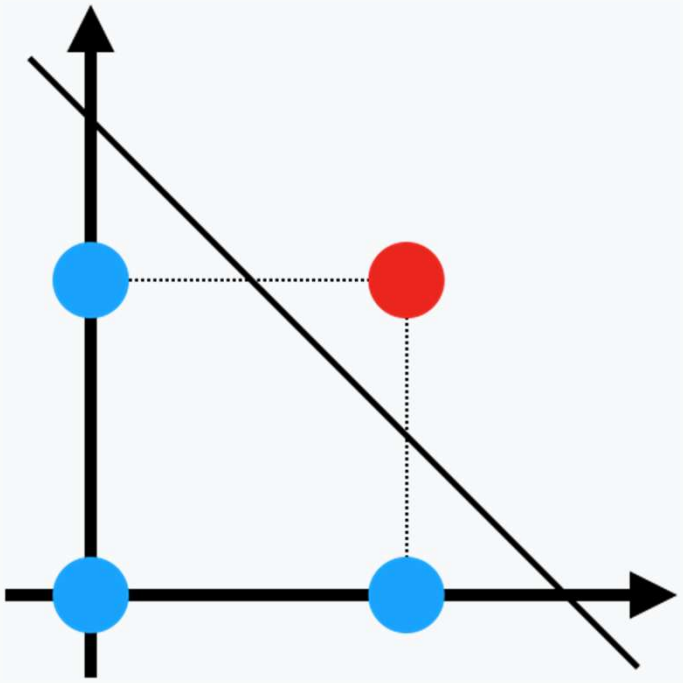
2016년
DeepMind AlphaGo

2차 암흑기
(1987-1993)

1. PERCEPTRON XOR 문제

- Perceptron – AND Gate

1956
perceptron



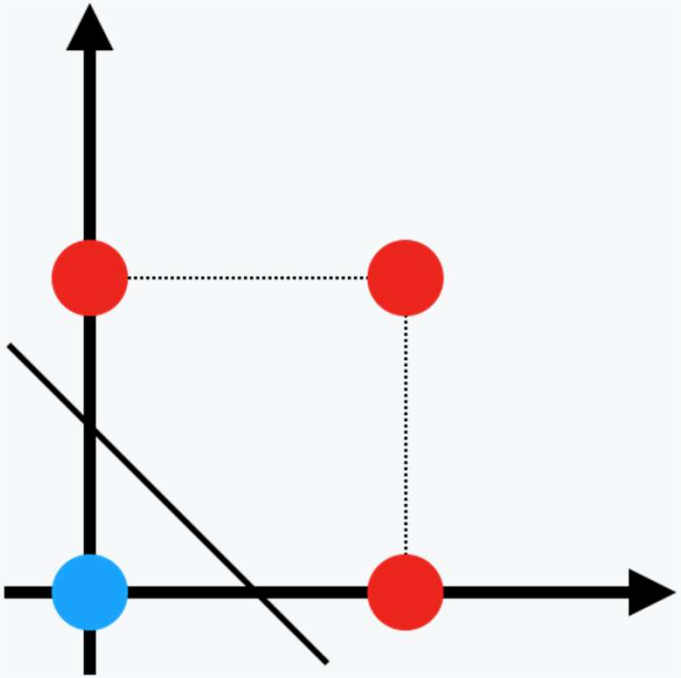
$$w_1 = 2,\ w_2 = 2,\ b = -3$$

x_1	x_2	$\sum$	y
0	0	-3	0
0	1	-1	0
1	0	-1	0
1	1	1	1

1. PERCEPTRON XOR 문제

- Perceptron – OR Gate

1956
perceptron



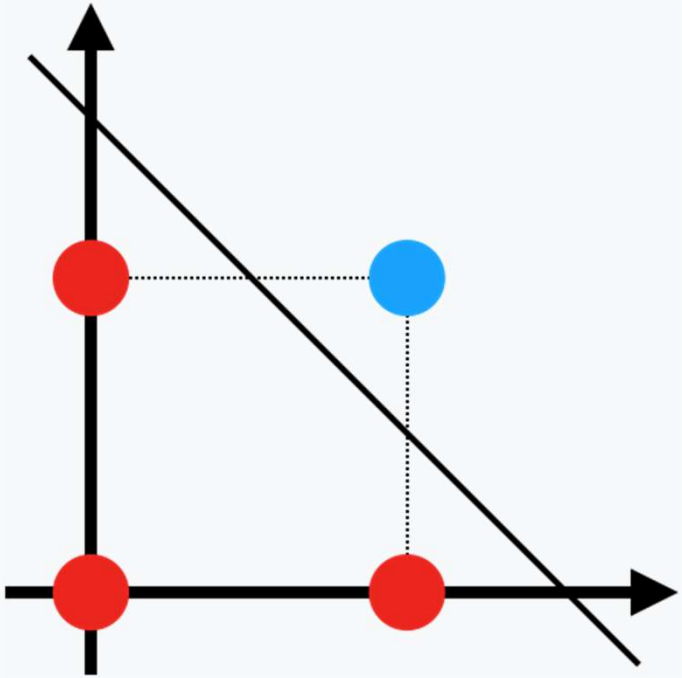
$$w_1 = 2,\ w_2 = 2,\ b = -1$$

x_1	x_2	$\sum$	y
0	0	-1	0
0	1	1	1
1	0	1	1
1	1	3	1

1. PERCEPTRON XOR 문제

- Perceptron – NAND Gate

1956
perceptron



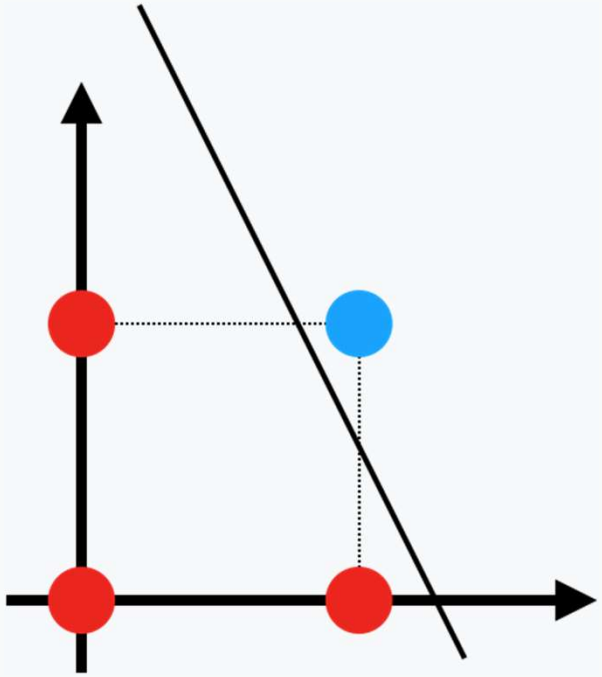
$$w_1 = -2,\ w_2 = -2,\ b = 3$$

x_1	x_2	$\sum$	y
0	0	3	1
0	1	1	1
1	0	1	1
1	1	-1	0

1. PERCEPTRON XOR 문제

- Perceptron – NAND Gate

1956
perceptron



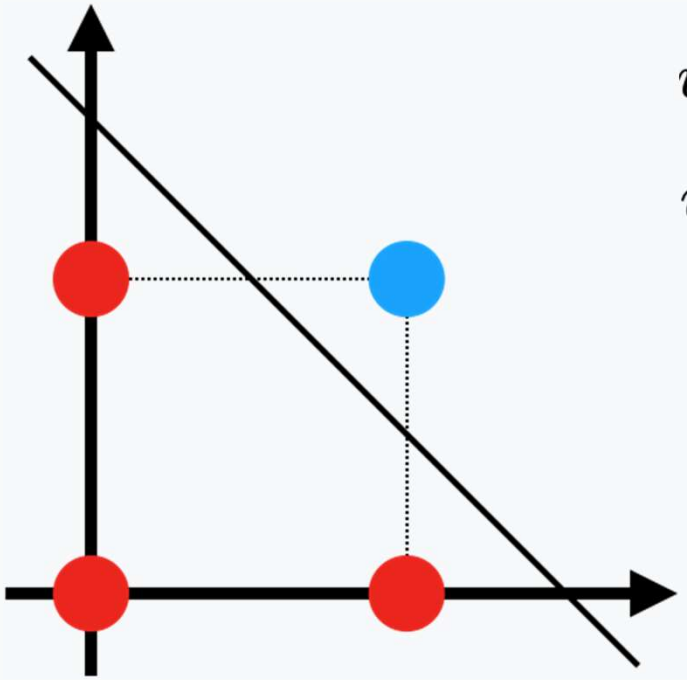
$$w_1 = -2,\ w_2 = -1,\ b = 2.5$$

x_1	x_2	$\sum$	y
0	0	2.5	1
0	1	1.5	1
1	0	0.5	1
1	1	-0.5	0

1. PERCEPTRON XOR 문제

- Perceptron – NAND Gate

1956
perceptron



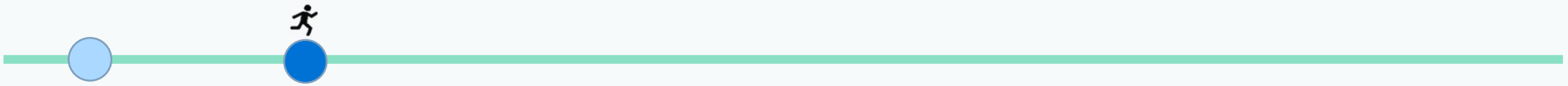
$$w_1 = -2,\ w_2 = -2,\ b = 3$$

$$w_1 = -2,\ w_2 = -1,\ b = 2.5$$

$$\vdots$$

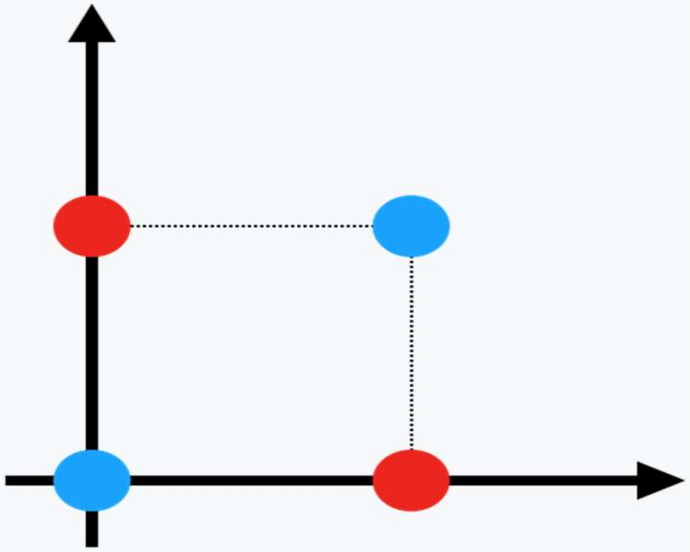
1. PERCEPTRON XOR 문제

- Perceptron – XOR 문제



1956
perceptron

1969
XOR problem



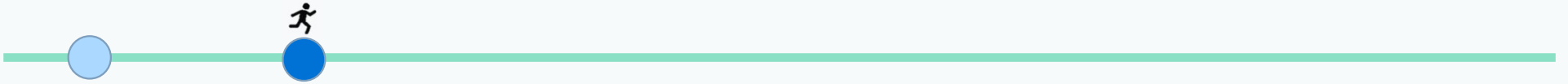
$w_1 =?,\ w_2 =?,\ b =?$

x_1	x_2	$\sum$	y
0	0		0
0	1		1
1	0		1
1	1		0

practice : P_02_00_XOR_problem.py

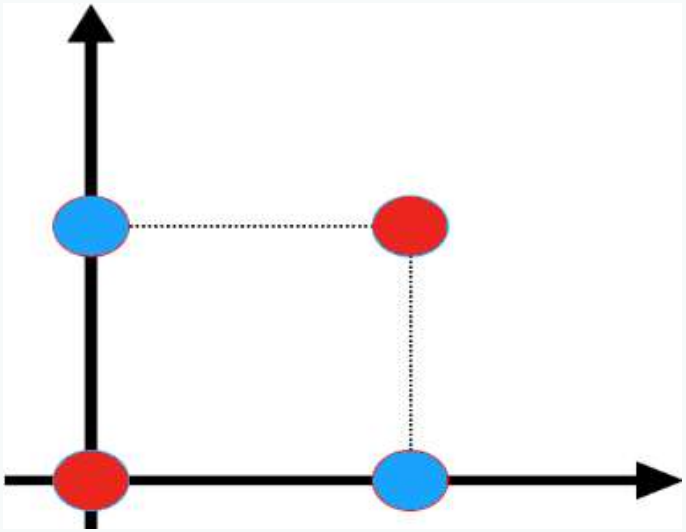
1. PERCEPTRON XOR 문제

- Perceptron – XOR 문제



1956
perceptron

1969
XOR problem



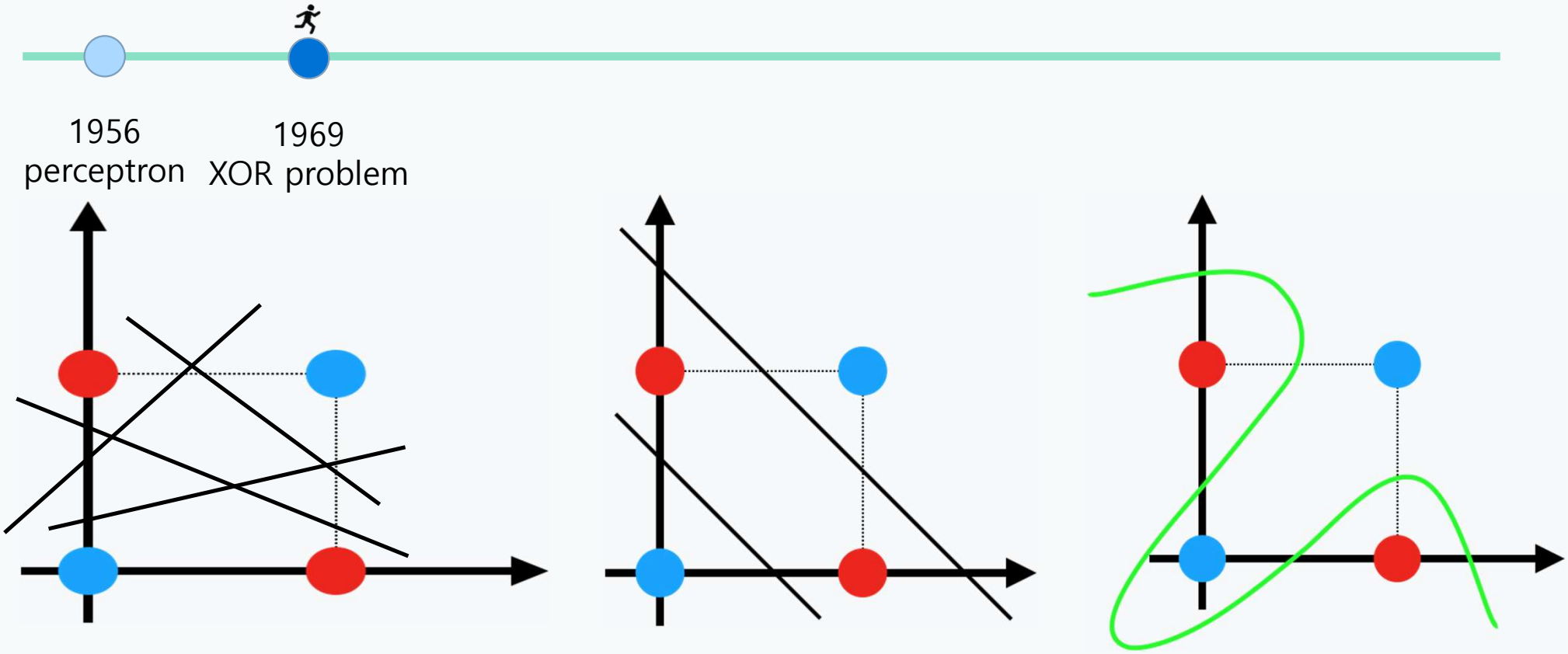
$w_1 = ?,\ w_2 = ?,\ b = ?$

x_1	x_2	$\sum$	y
0	1		1
1	1		0
1	1		0
0	1		1

execise : 02_00_XOR_problem.py

1. PERCEPTRON XOR 문제

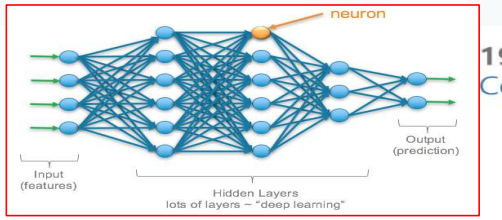
- Perceptron – XOR 문제



1. PERCEPTRON XOR 문제

오늘의 학습 목표

regression, classification



신경망의 70년

19C-20C 초반
생체 신경망

이반 파블로프

1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1956년
Perceptron

로젠블랫

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

1차 암흑기
(1974-1980)

1982년
Recurrent Neural Network (RNN)

존 홉필드

1998년
Convolution Neural Network (CNN)

얀 르쿤

2006년
Deep Learning

제프리 힌튼

2012년
AlexNet
IMAGENET

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

2014년
Generative Adversarial Network (GAN)

이안 굿펠로우

2016년
DeepMind AlphaGo

2차 암흑기
(1987-1993)

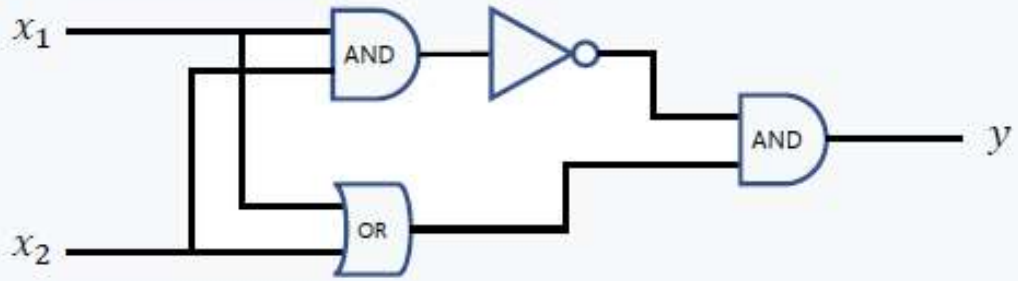
1. PERCEPTRON XOR 문제

- Perceptron – XOR 문제

1957 perceptron

1969 XOR problem

XOR 게이트



XOR 게이트는 AND 와 OR 게이트의 조합으로 만들 수 있구나

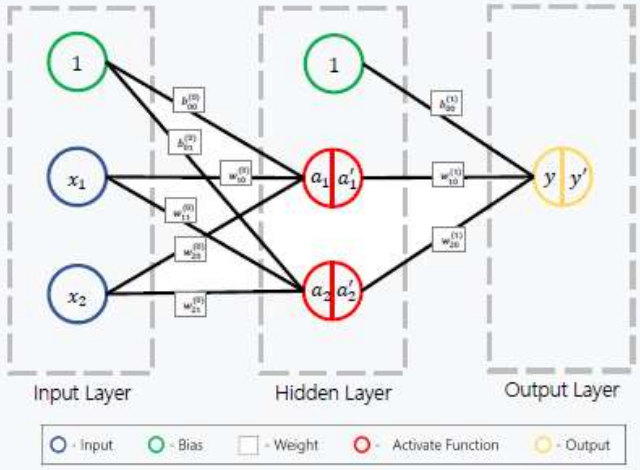
1. PERCEPTRON XOR 문제

- Perceptron – XOR 문제

1957 perceptron

1969 XOR problem

그럼 퍼셉트론 여러 개를 쌓아보자 : MLP



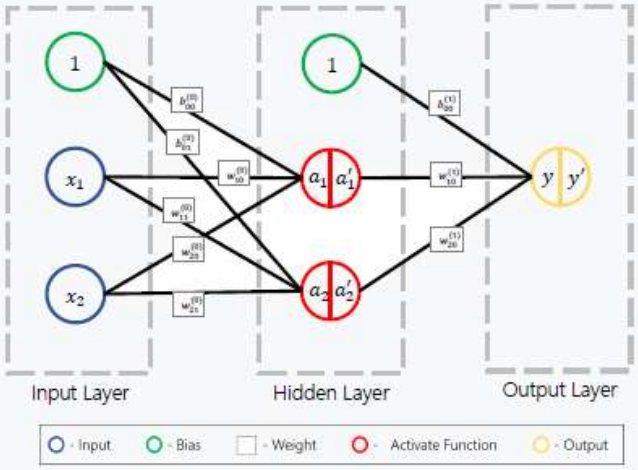
1. PERCEPTRON XOR 문제

- Perceptron – XOR 문제

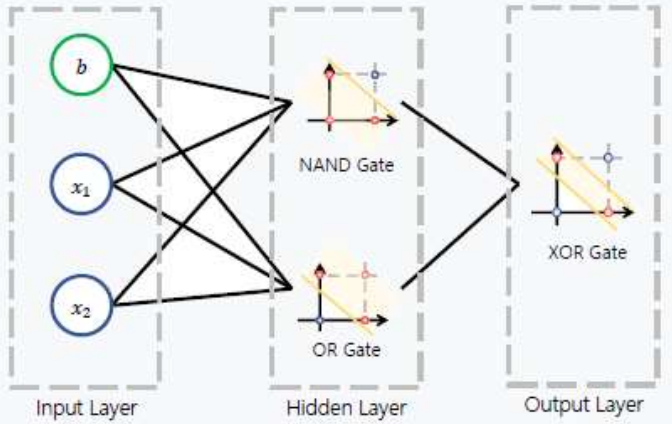
1957 perceptron

1969 XOR problem

그럼 퍼셉트론 여러 개를 쌓아보자 : MLP



MLP 로 XOR **게이트를 풀어보자**



MLP 로도 XOR 게이트를 만들어 낼 수 있다

1. PERCEPTRON XOR 문제

- Perceptron – XOR 문제

1957 perceptron

1969 XOR problem

$w_1 = -2,\ w_2 = -2,\ b = 3$

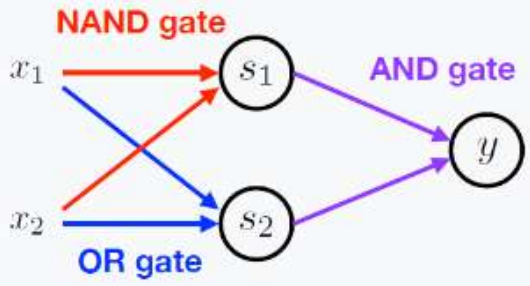
x_1	x_2	$\sum$	s_1
0	0	3	1
0	1	1	1
1	0	1	1
1	1	-1	0

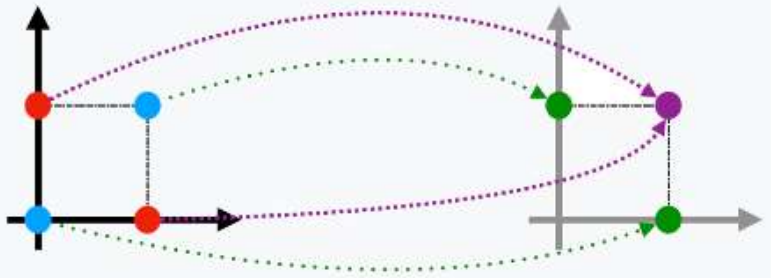
$w_1 = 2,\ w_2 = 2,\ b = -1$

x_1	x_2	$\sum$	s_2
0	0	-1	0
0	1	1	1
1	0	1	1
1	1	3	1

$w_1 = 2,\ w_2 = 2,\ b = -3$

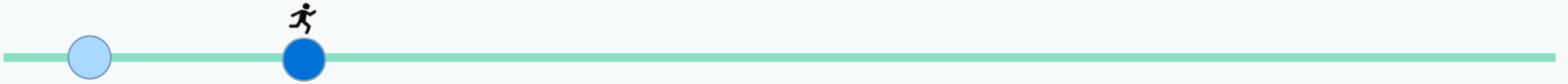
s_1	s_2	$\sum$	y
1	0	-1	0
1	1	1	1
1	1	1	1
0	1	-1	0





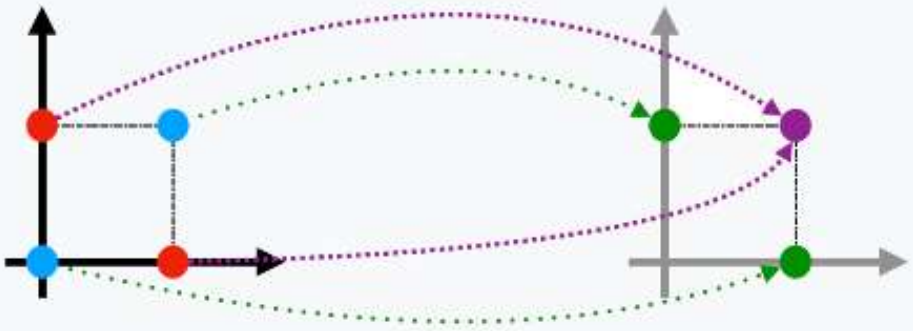
1. PERCEPTRON XOR 문제

- Perceptron – XOR 문제



1957
perceptron

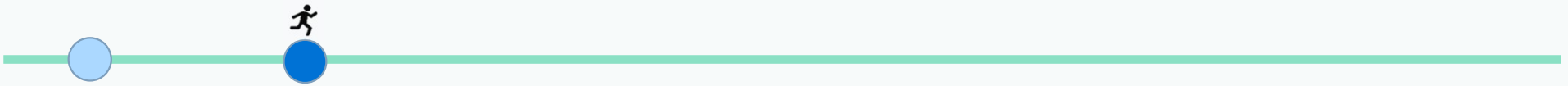
1969
XOR problem



to map space X to space Y

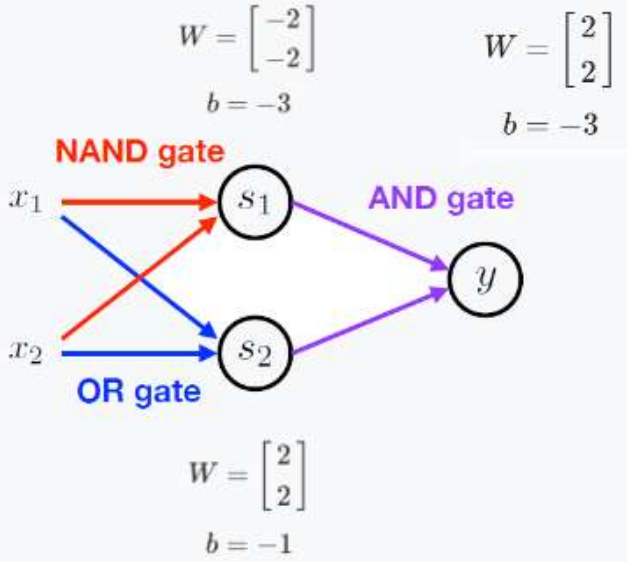
1. PERCEPTRON XOR 문제

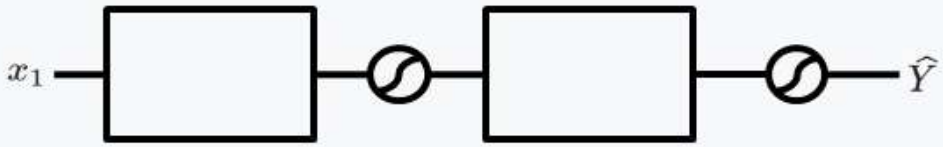
- Perceptron – XOR 문제



1957
perceptron

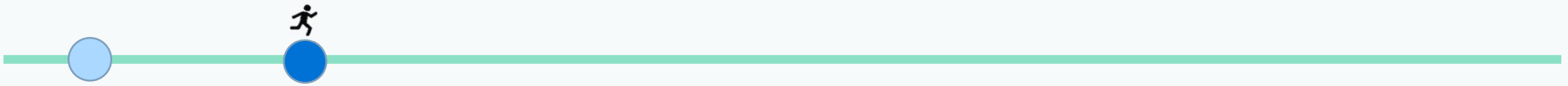
1969
XOR problem





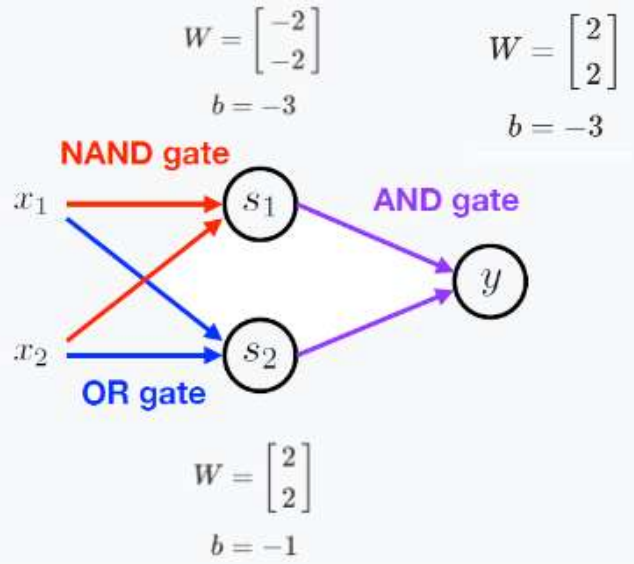
1. PERCEPTRON XOR 문제

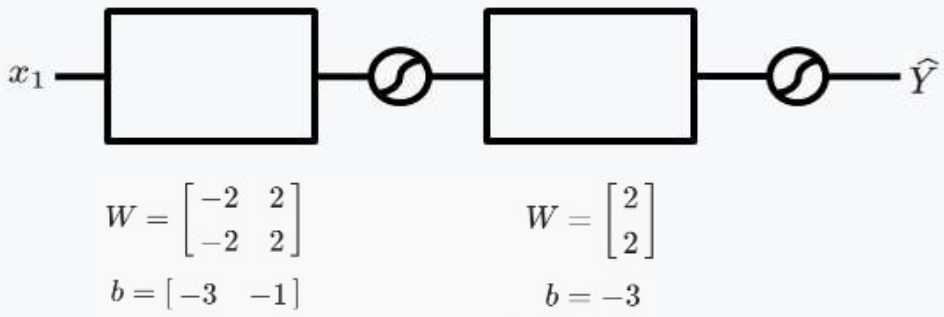
- Perceptron – XOR 문제



1957
perceptron

1969
XOR problem





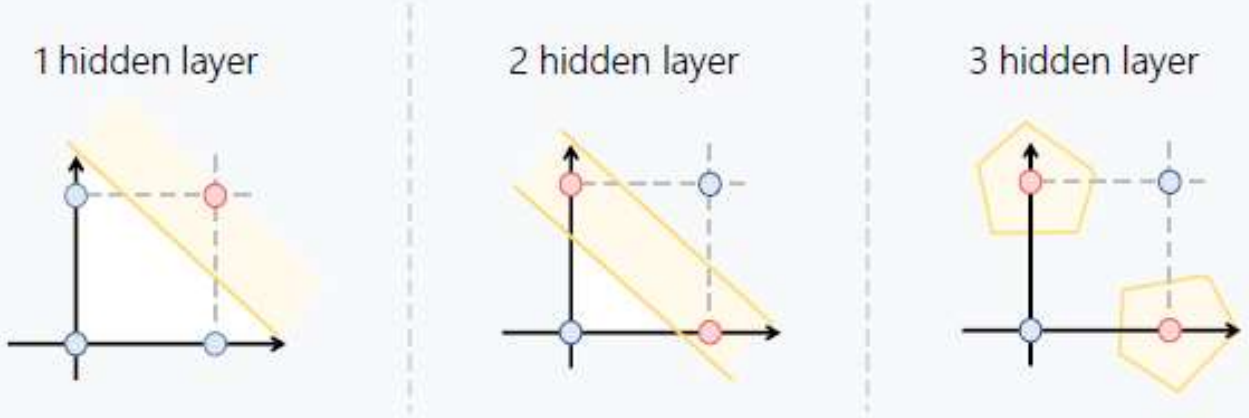
1. PERCEPTRON XOR 문제

- Perceptron – XOR 문제

1957 perceptron

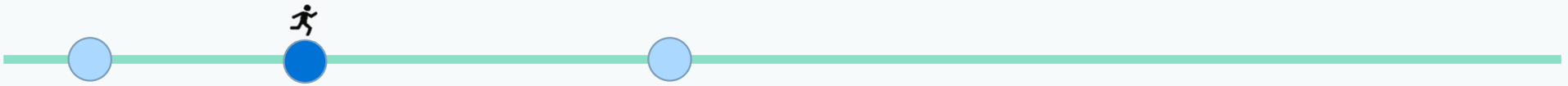
1969 XOR problem

Layer 의 개수에 따라 결정할 수 있는 영역



2. Neural Network

- Neural Network or Multi Layer Perceptron



1957
perceptron

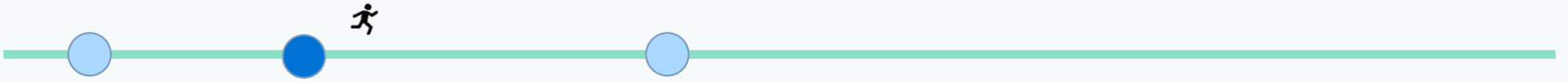
1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)

많은 perceptron을 쌓아서 비선형의 데이터를
선형 분리가 가능하도록 단순화 시킴

2. Neural Network

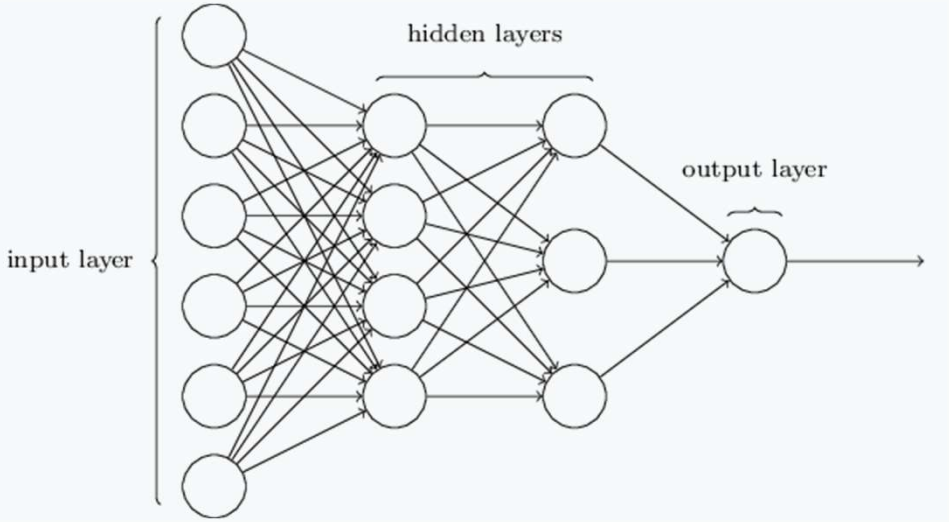
- Neural Network or Multi Layer Perceptron



1957
perceptron

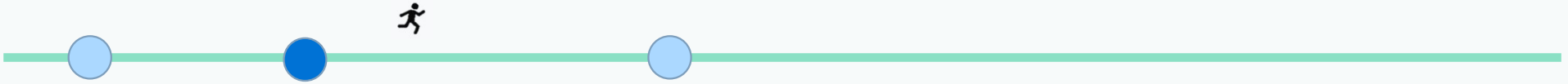
1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)



2. Neural Network

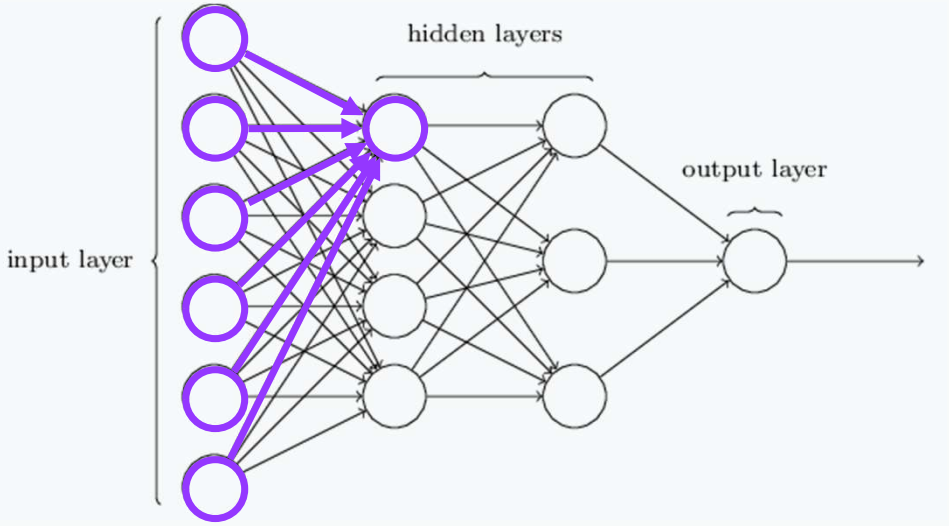
- Neural Network or Multi Layer Perceptron



1957 perceptron

1969 XOR problem

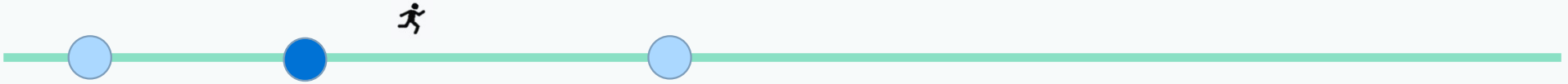
1986 Multi layer Perceptron (backpropagation)



$\mathbf{w}^1$: vectors

2. Neural Network

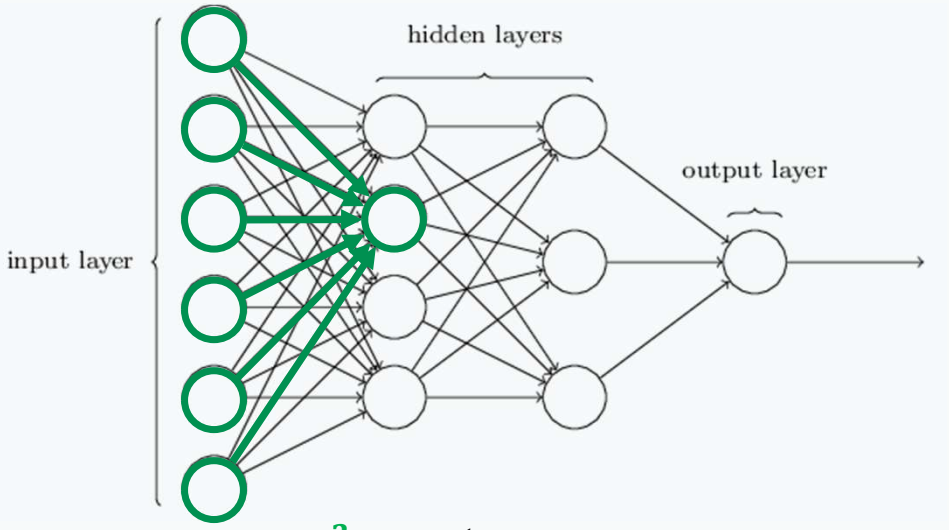
- Neural Network or Multi Layer Perceptron



1957 perceptron

1969 XOR problem

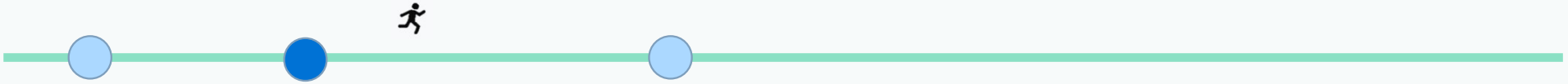
1986 Multi layer Perceptron (backpropagation)



w^2 : vectors

2. Neural Network

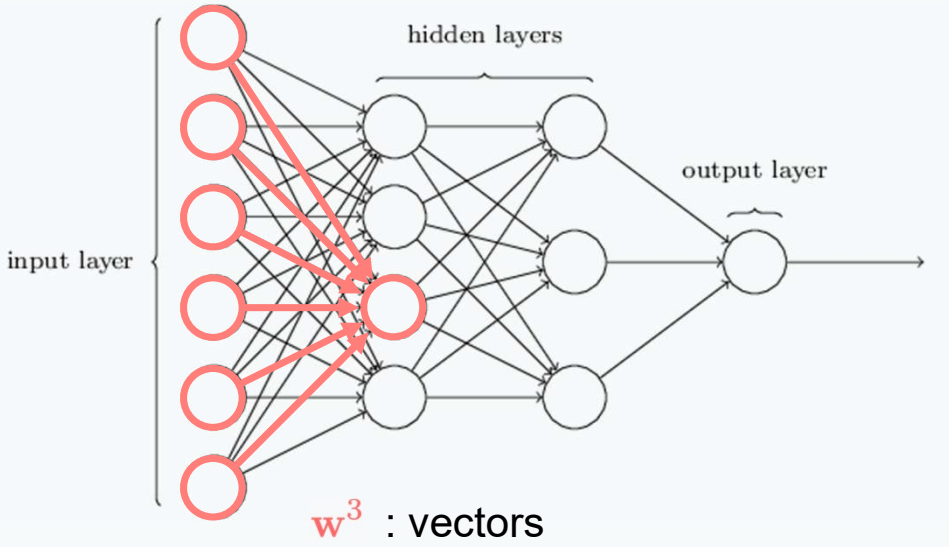
- Neural Network or Multi Layer Perceptron



1957 perceptron

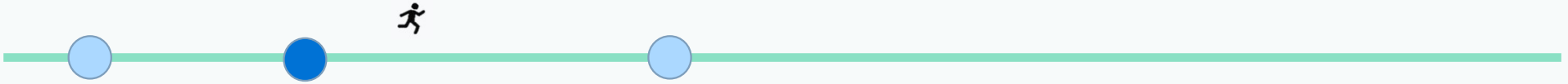
1969 XOR problem

1986 Multi layer Perceptron (backpropagation)



2. Neural Network

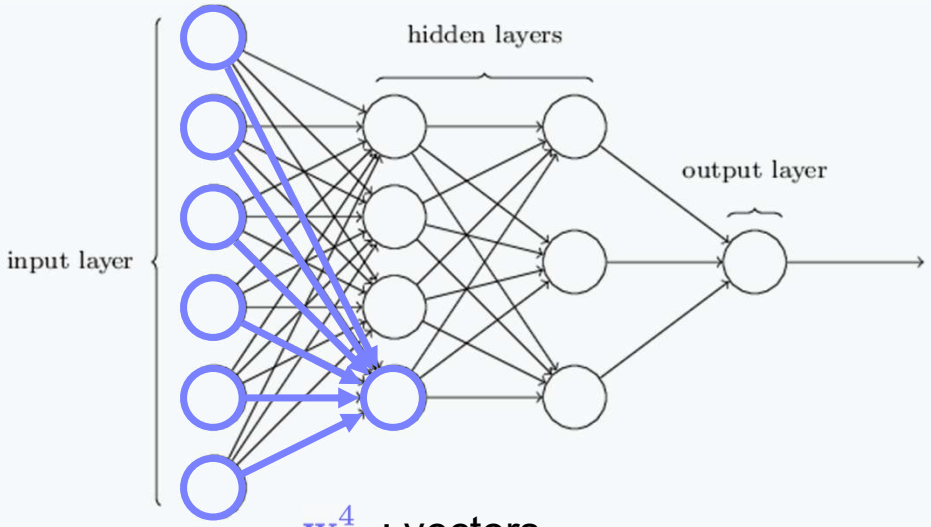
- Neural Network or Multi Layer Perceptron



1957 perceptron

1969 XOR problem

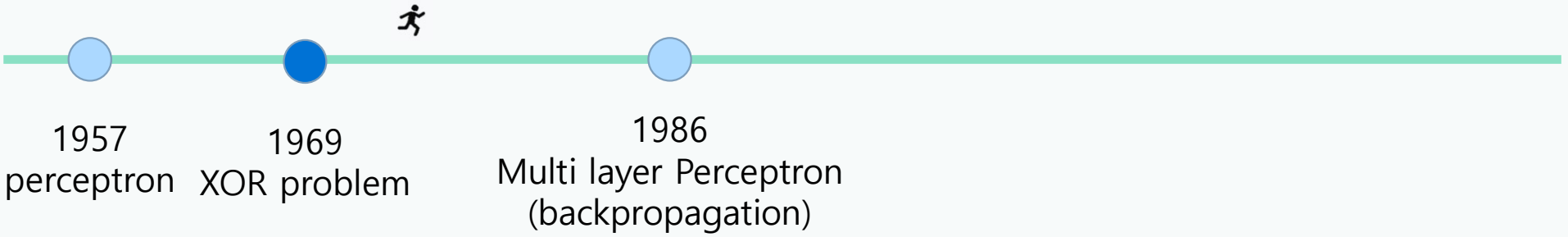
1986 Multi layer Perceptron (backpropagation)

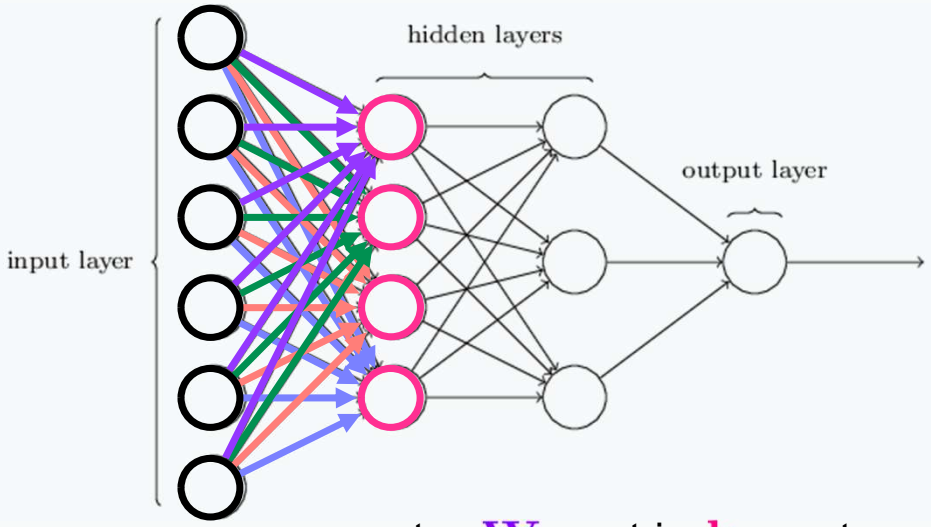


w^4 : vectors

2. Neural Network

- Neural Network or Multi Layer Perceptron





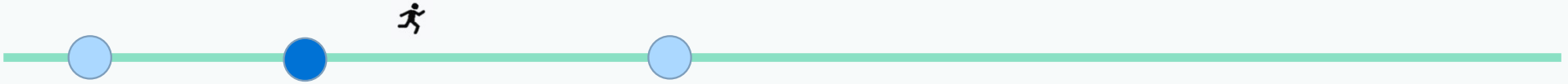
Input Layer의 경우
데이터를 전달하는 역할

$\mathbf{x}$: vector $\mathbf{W}$: matrix $\mathbf{h}$: vector

$$\mathbf{h} = f_{\text{act}}\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right)$$

2. Neural Network

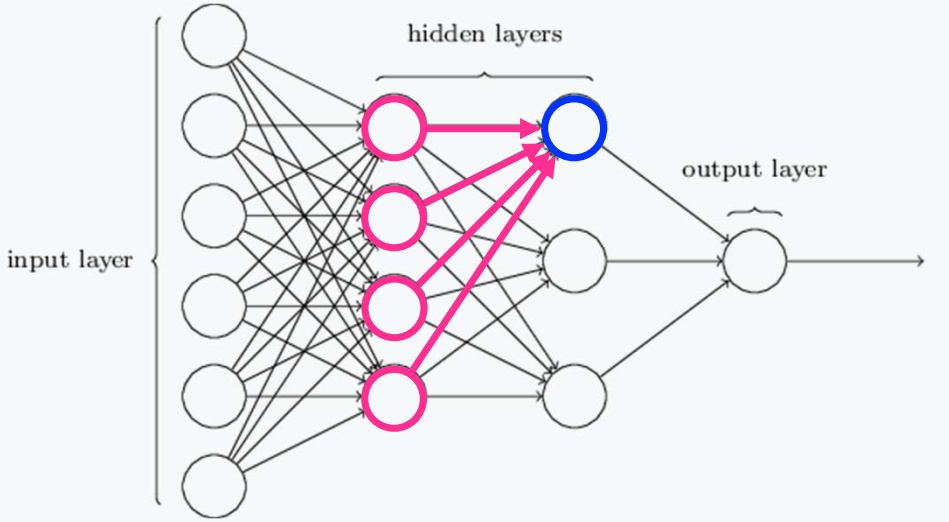
- Neural Network or Multi Layer Perceptron



1957
perceptron

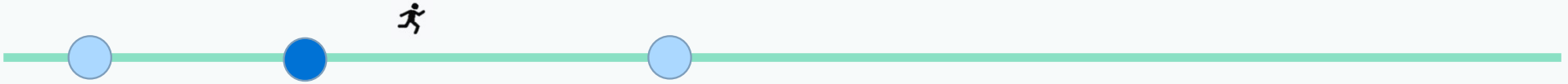
1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)



2. Neural Network

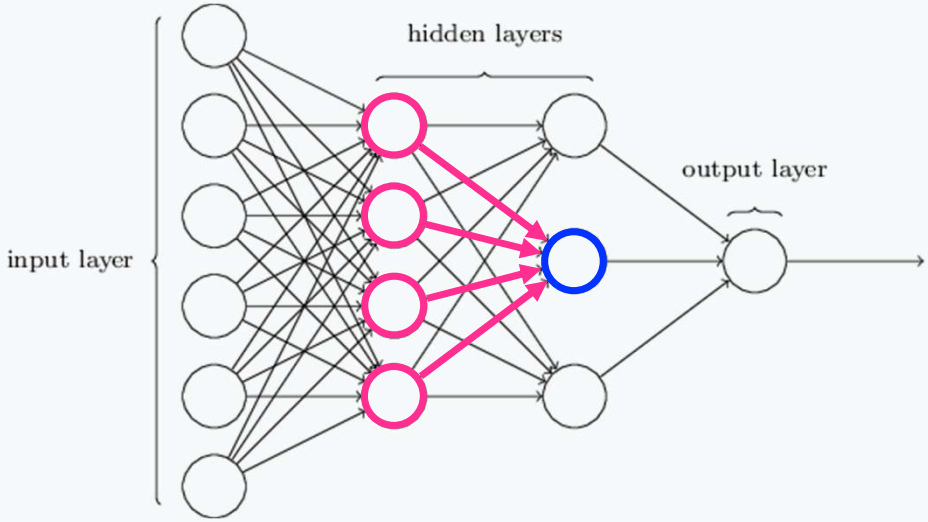
- Neural Network or Multi Layer Perceptron



1957
perceptron

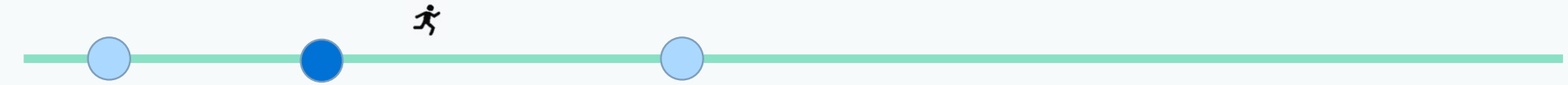
1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)



2. Neural Network

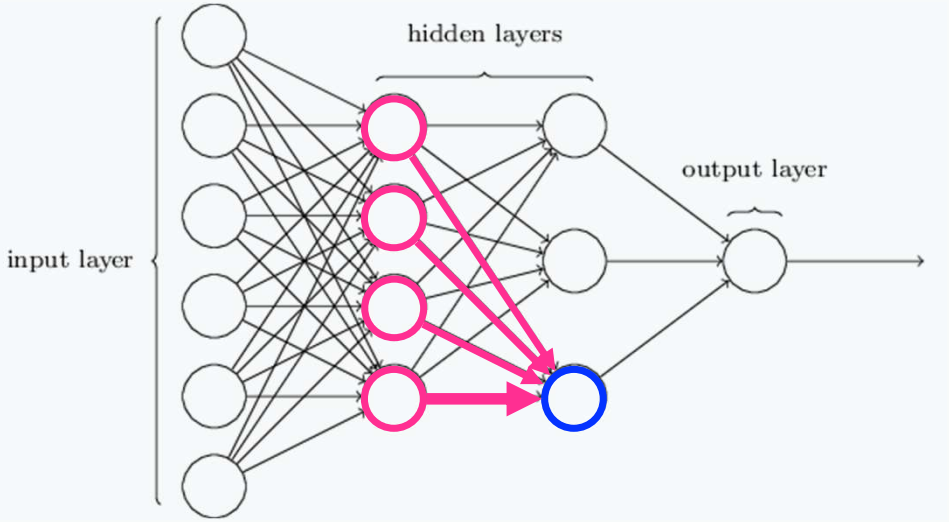
- Neural Network or Multi Layer Perceptron



1957
perceptron

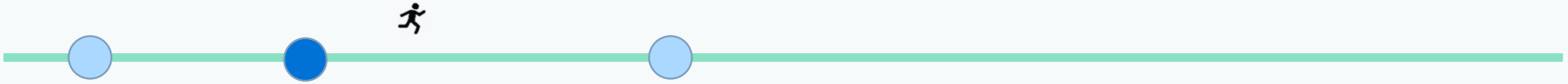
1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)



2. Neural Network

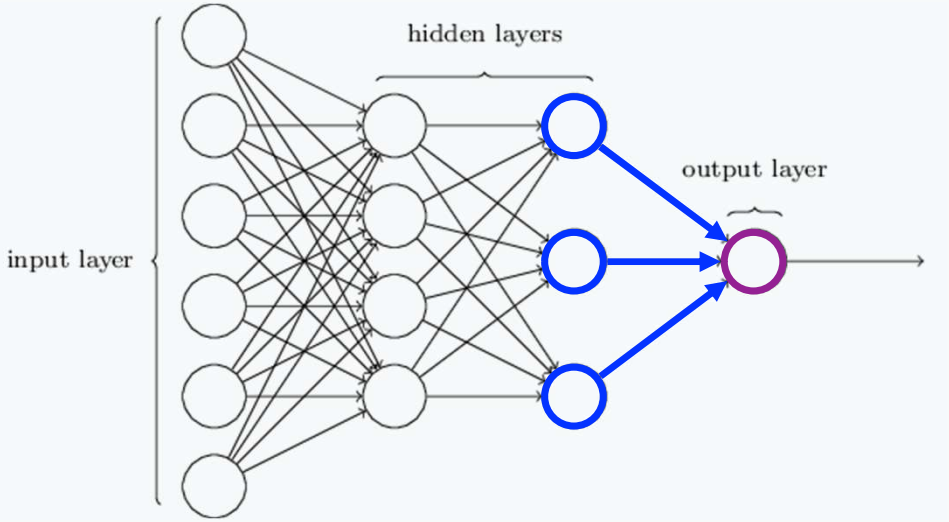
- Neural Network or Multi Layer Perceptron



1957
perceptron

1969
XOR problem

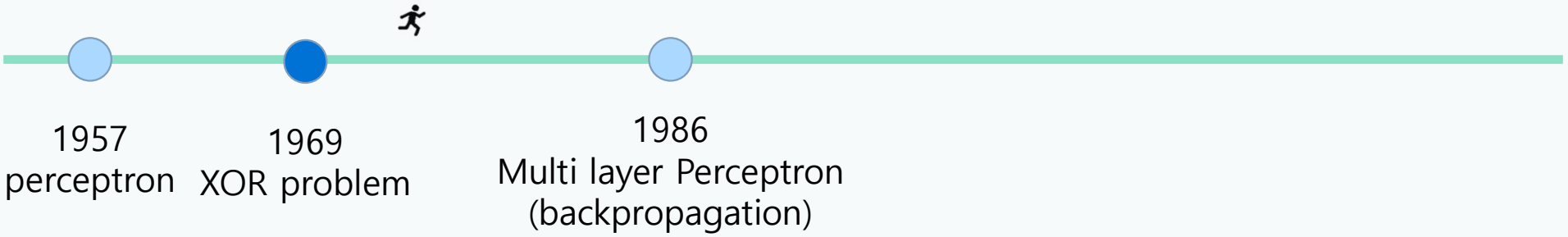
1986
Multi layer Perceptron
(backpropagation)



Hidden layer의 경우
비선형 데이터를 선형 분류가 가능하도록
representation 해주는 역할

2. Neural Network

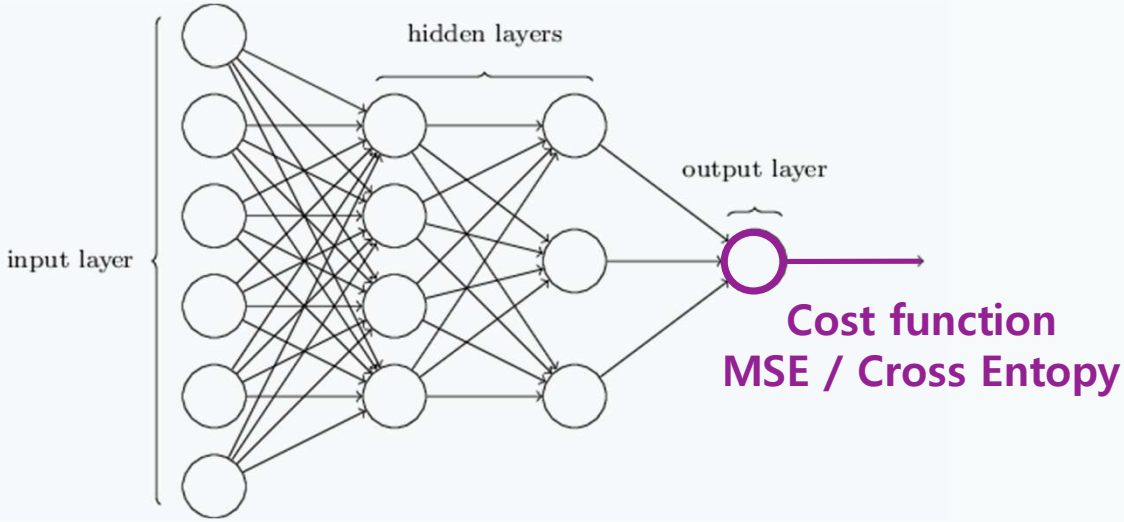
- Neural Network or Multi Layer Perceptron



1957
perceptron

1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)

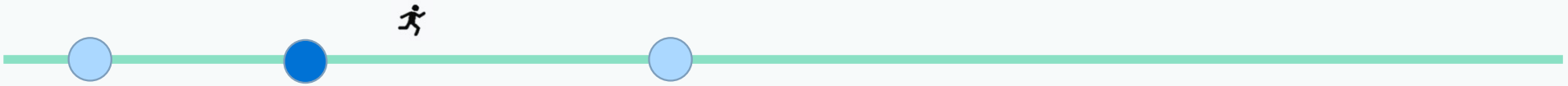


Cost function
MSE / Cross Entopy

Output layer은
기능을 정의해주는 역할

2. Neural Network

- Neural Network or Multi Layer Perceptron

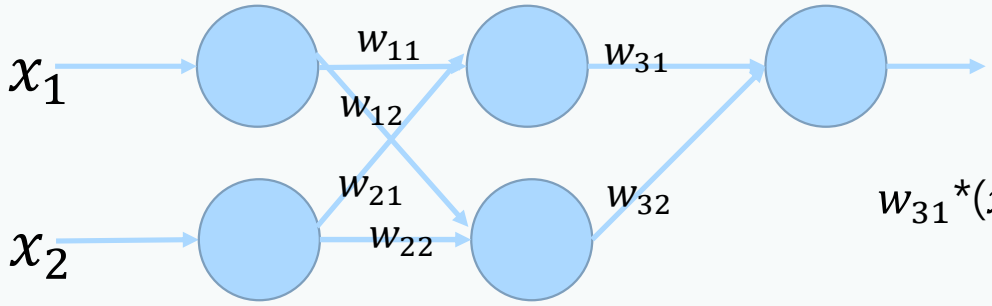


1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

$x_1 * w_{11} + x_2 * w_{21}$



$x_1 * w_{12} + x_2 * w_{22}$

$$w_{31}*(x_1 * w_{11} + x_2 * w_{21}) + w_{32}*(x_1 * w_{12} + x_2 * w_{22})$$

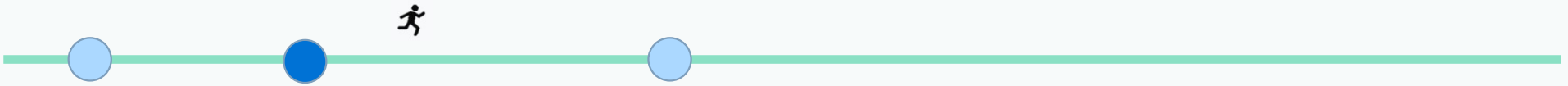
$$= w_{11} * w_{31} * x_1 + w_{21} * w_{31} * x_2 +$$

$$w_{12} * w_{32} * x_1 + w_{22} * w_{32} * x_2$$

$$= W1 * x_1 + W2 * x_2$$

2. Neural Network

- Neural Network or Multi Layer Perceptron



1957
perceptron

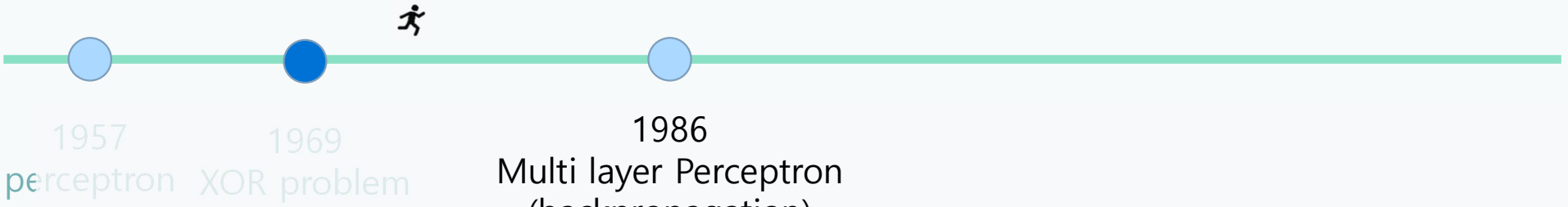
1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)

결국 perceptron을 여러 개 쌓아도
하나의 perceptron과 같은 효과

2. Neural Network

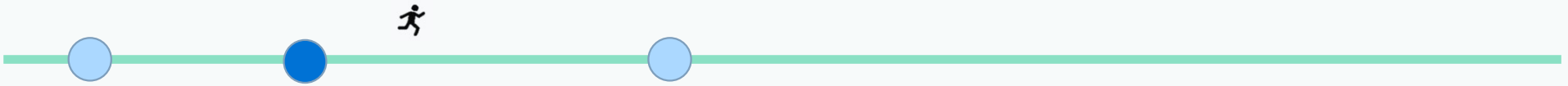
- Neural Network or Multi Layer Perceptron



어떻게 하면 해결이 가능할까요?

2. Neural Network

- Neural Network or Multi Layer Perceptron

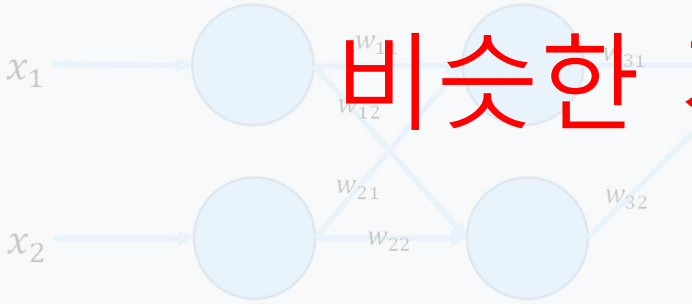


1957
perceptron

1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)

$x_1 * w_{11} + x_2 * w_{21}$



$x_1 * w_{21} + x_2 * w_{22}$

비슷한 개념을 배웠습니다

$w_{31}*(x_1 * w_{11} + x_2 * w_{12}) + w_{32}*(x_1 * w_{21} + x_2 * w_{22})$

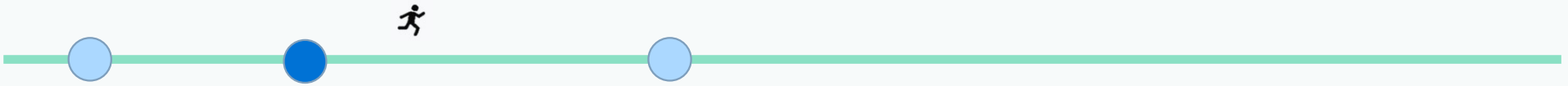
$= w_{11} * w_{31} * x_1 + w_{12} * w_{31} * x_2 +$

$w_{21} * w_{32} * x_1 + w_{22} * w_{32} * x_1$

$= W1 * x_1 + W2 * x_2$

2. Neural Network

- Neural Network or Multi Layer Perceptron

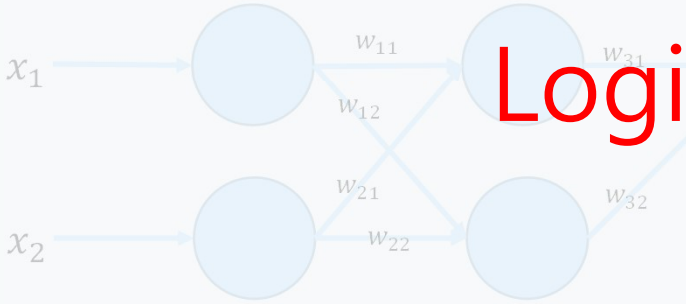


1957
perceptron

1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)

$x_1 * w_{11} + x_2 * w_{21}$



$x_1 * w_{21} + x_2 * w_{22}$

Logistic Regression

$w_{31}*(x_1 * w_{11} + x_2 * w_{12}) + w_{32}*(x_1 * w_{21} + x_2 * w_{22})$

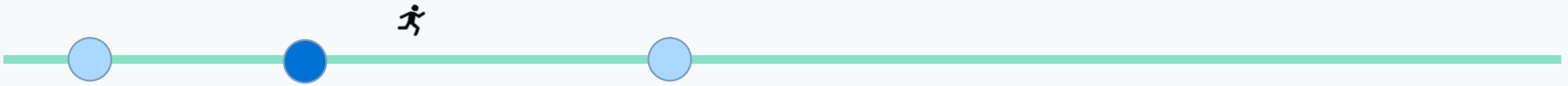
$= w_{11} * w_{31} * x_1 + w_{12} * w_{31} * x_2 +$

$w_{21} * w_{32} * x_1 + w_{22} * w_{32} * x_1$

$= W1 * x_1 + W2 * x_2$

2. Neural Network

- Neural Network or Multi Layer Perceptron



1957
perceptron

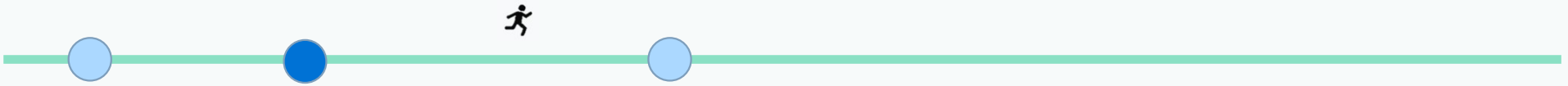
1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)

perceptron이 perceptron에게 전달 할때
비선형으로 전달하자!

2. Neural Network

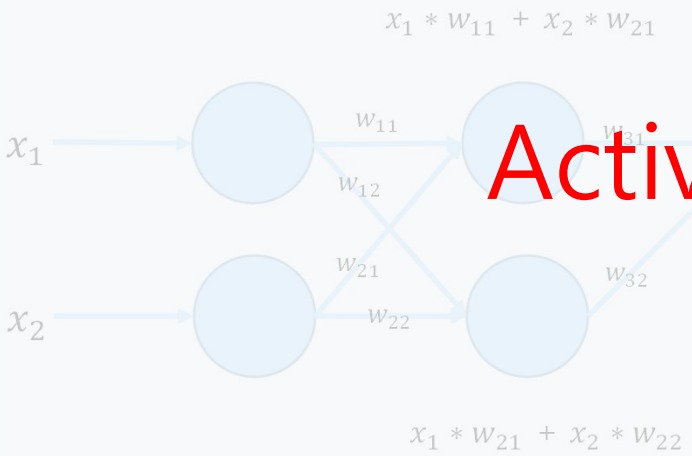
- Neural Network or Multi Layer Perceptron



1957
perceptron

1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)

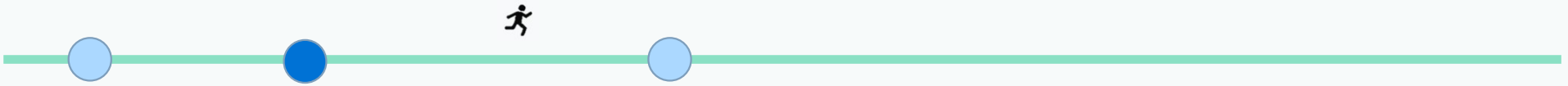


Activation Function

$$w_{31}*(x_1 * w_{11} + x_2 * w_{12}) + w_{32}*(x_1 * w_{21} + x_2 * w_{22})$$
$$= w_{11} * w_{31} * x_1 + w_{12} * w_{31} * x_2 +$$
$$w_{21} * w_{32} * x_1 + w_{22} * w_{32} * x_1$$
$$= W1 * x_1 + W2 * x_2$$

2. Neural Network

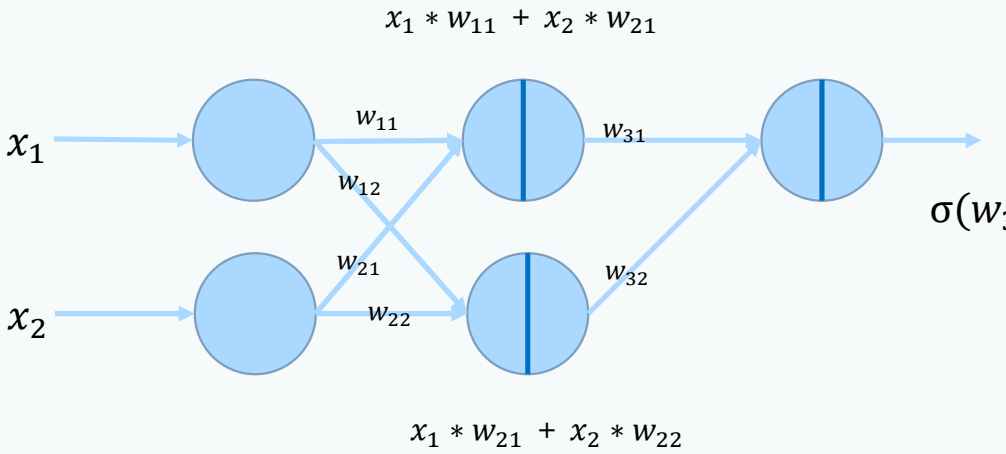
- Activation Function



1957 perceptron

1969 XOR problem

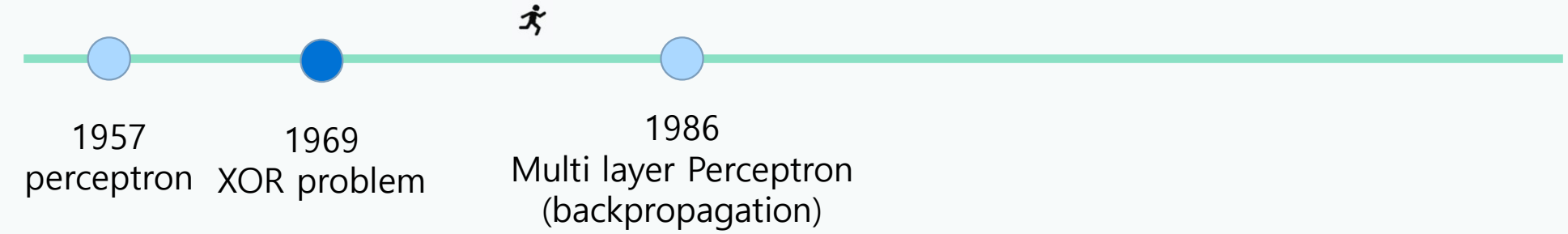
1986 Multi layer Perceptron (backpropagation)



$$\sigma(w_{31}{}^{*}\sigma(x_1 * w_{11} + x_2 * w_{21})) + \sigma(w_{31}{}^{*}\sigma(x_1 * w_{21} + x_2 * w_{22}))$$

2. Neural Network

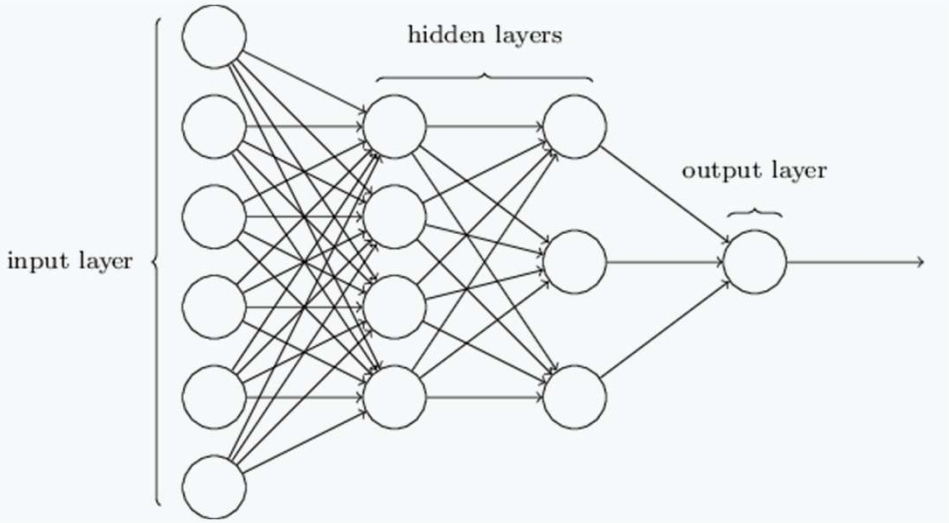
- Activation Function



1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)



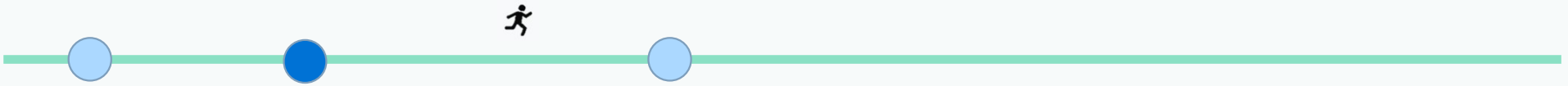
$$\hat{\mathbf{y}} = f(\mathbf{x}; \mathbf{w}^1, \cdots, \mathbf{w}^L, \mathbf{b}^1, \cdots, \mathbf{b}^L)$$
$$= \sigma(\mathbf{w}^L \sigma(\mathbf{w}^{L-1} \cdots \sigma(\mathbf{w}^1 \mathbf{x} + \mathbf{b}^1) \cdots) + \mathbf{b}^L)$$

activation function:
sigmoid function

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

2. Neural Network

- Activation Function

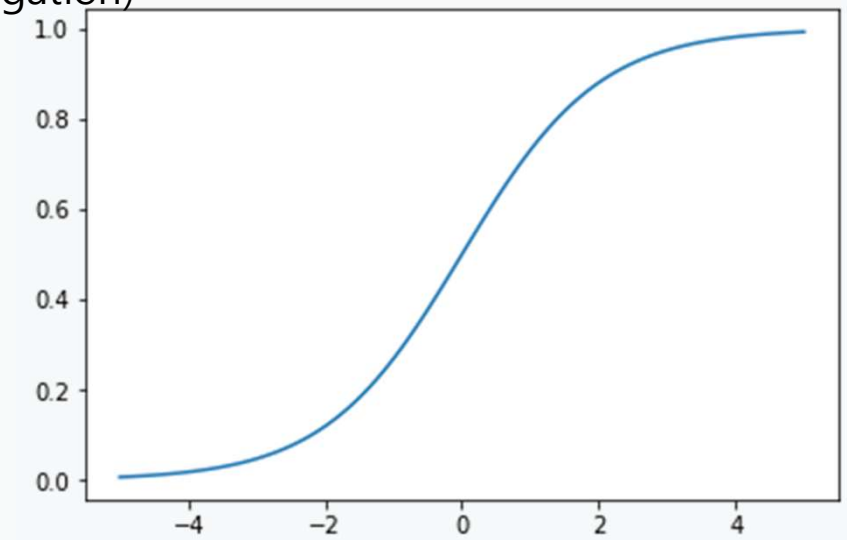


1957
perceptron

1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)

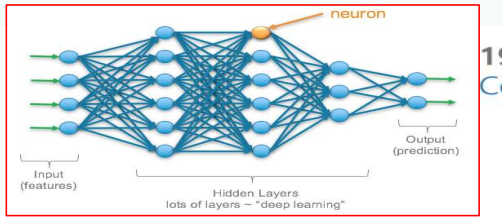
$$\sigma(z) \equiv \frac{1}{1+e^{-z}}$$



2. Neural Network

오늘의 학습 목표

regression, classification



신경망의 70년

1943년
최초 인공신경망

워렌 맥컬록과 월터 피츠

1969년
Perceptron 한계 제시

마빈 민스키, 시모어 페퍼트

1974년, 1986년
역전파 알고리즘

- 폴 워보스,
- 데이비드 루멜하트, 제임스 맥클레랜드, 제프리 힌튼, 로날드 윌리암스
- 데이비드 파커, 얀 르쿤

1998년
Convolution Neural Network (CNN)

얀 르쿤

2006년
Deep Learning

제프리 힌튼

2012년
AlexNet
IMAGENET

2016년
DeepMind AlphaGo

1차 암흑기
(1974-1980)

2차 암흑기
(1987-1993)

19C-20C 초반
생체 신경망

이반 파블로프

1956년
Perceptron

로젠블랫

1982년
Recurrent Neural Network (RNN)

존 홉필드

2012년
YouTube 1,000만 이미지에서 고양이 인식

제프 딘앤과 앤듀류 응

2014년
Generative Adversarial Network (GAN)

이안 굿펠로우

3. BACKPROPAGATION

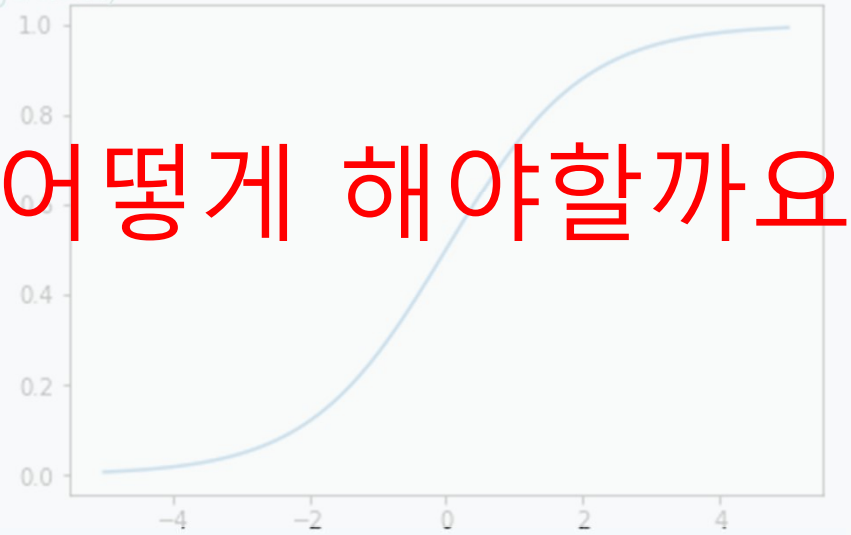
- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

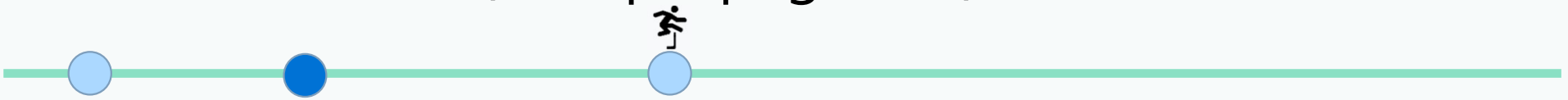
$$\sigma(z) \equiv \frac{1}{1+e^{-z}}$$



그런데 학습은 어떻게 해야할까요?

3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)



$$\sigma(z) \equiv \frac{1}{1+e^{-z}}$$

backpropagation

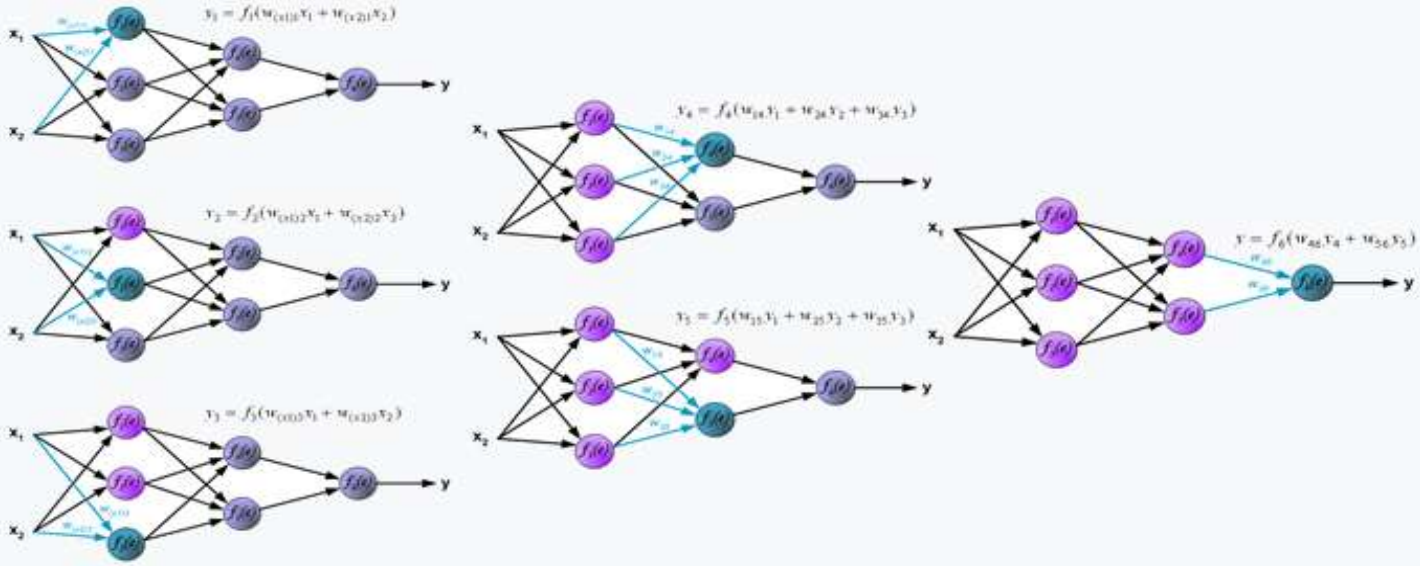
3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957
perceptron

1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)



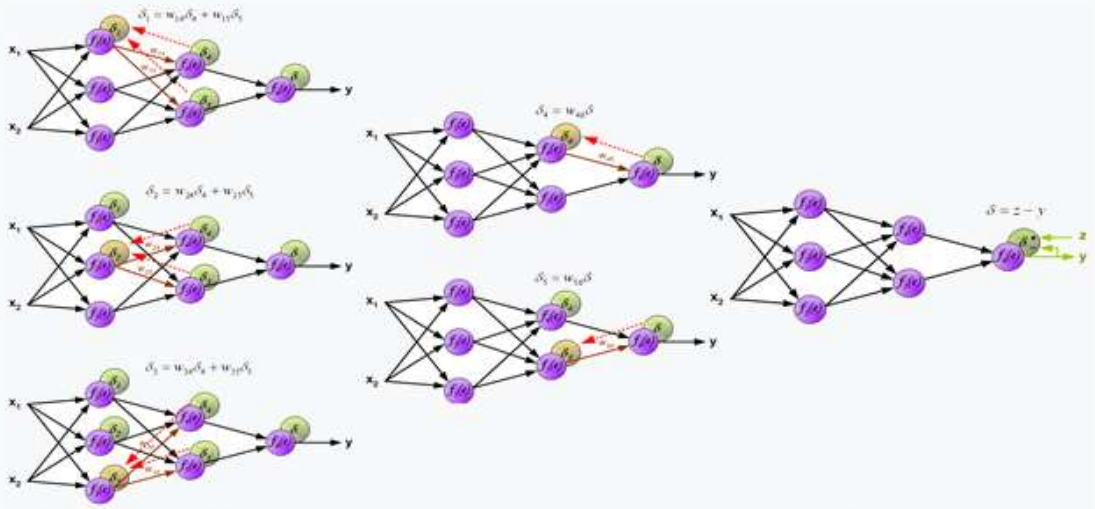
3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957
perceptron

1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)



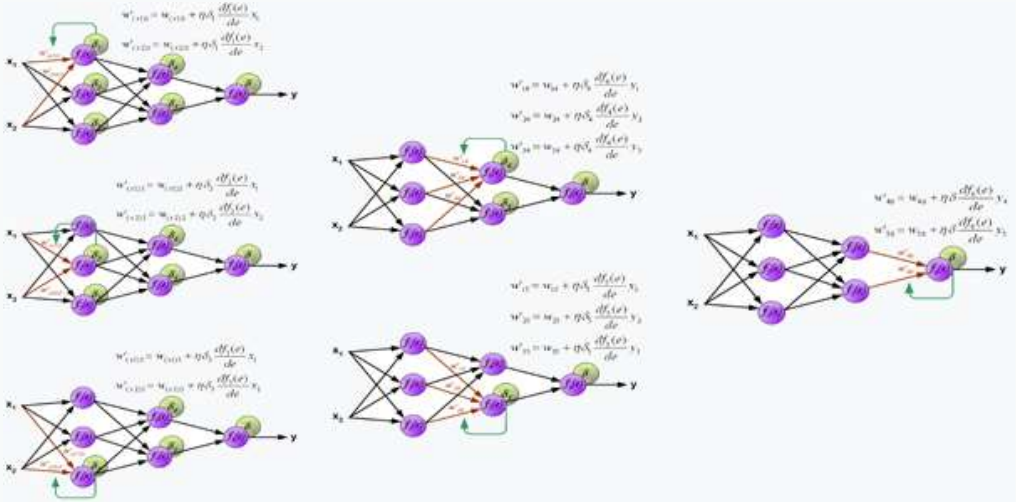
3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957
perceptron

1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)



input layer　　hidden layer　　output layer

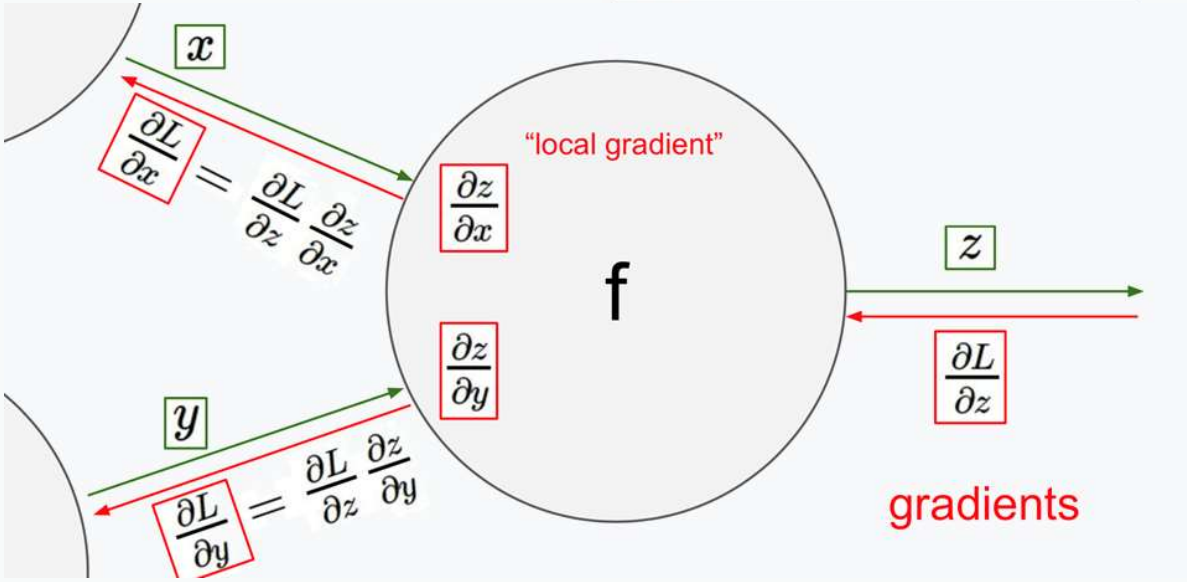
3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957 perceptron

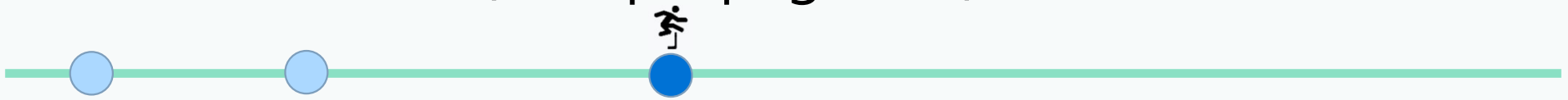
1969 XOR problem

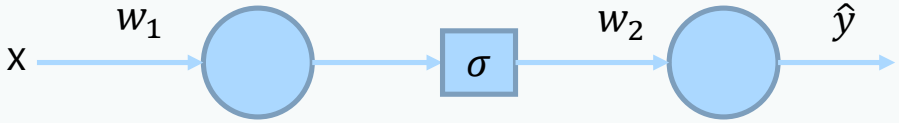
1986 Multi layer Perceptron (backpropagation)



3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)





$$h_1 = xw_1$$

$$h_2 = \sigma(h_1)$$

$$\hat{y} = w_2 h_2$$

$$L = \hat{y} - y$$

$$w_1 = w_1 - \alpha \Delta w_1$$

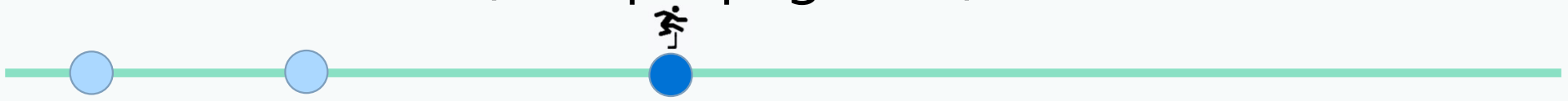
$$w_2 = w_1 - \alpha \Delta w_2$$

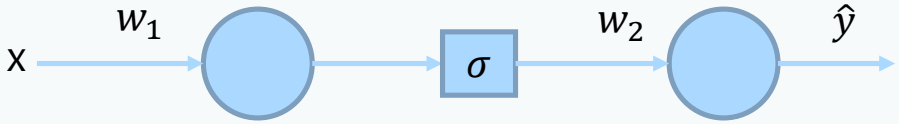
$$\Delta w_1 = \frac{\partial L}{\partial w_1}$$

$$\Delta w_2 = \frac{\partial L}{\partial w_2}$$

3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)





$$h_1 = xw_1$$

$$h_2 = \sigma(h_1)$$

$$\hat{y} = w_2 h_2$$

$$\boxed{L = \hat{y} - y}$$

$$w_1 = w_1 - \alpha \Delta w_1$$

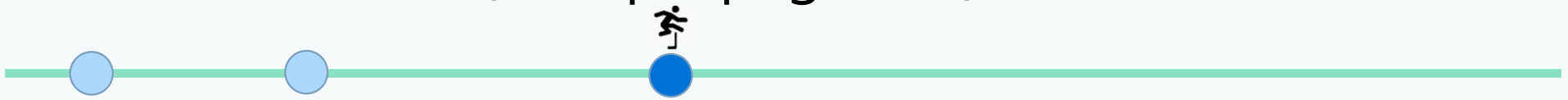
$$w_2 = w_1 - \alpha \Delta w_2$$

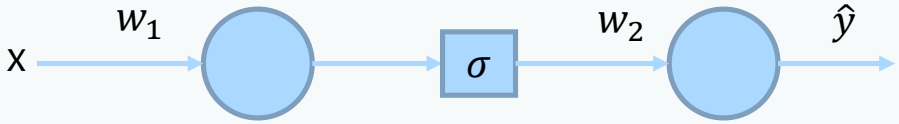
$$\Delta w_1 = \frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}}$$

$$\Delta w_2 = \frac{\partial L}{\partial w_2} =$$

3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)





$$h_1 = xw_1$$

$$h_2 = \sigma(h_1)$$

$$\hat{y} = w_2 h_2$$

$$L = \hat{y} - y$$

$$w_1 = w_1 - \alpha \Delta w_1$$

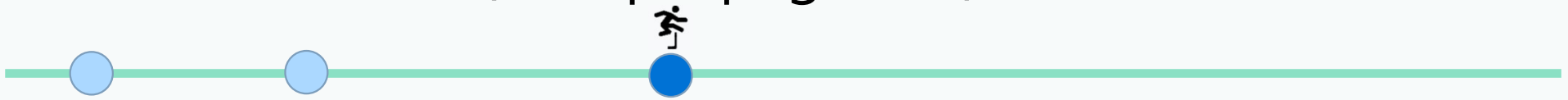
$$w_2 = w_1 - \alpha \Delta w_2$$

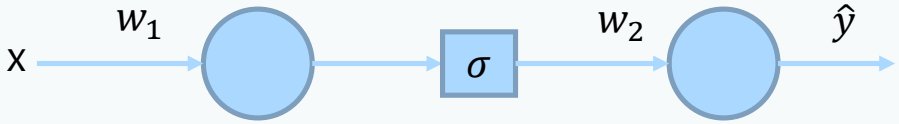
$$\Delta w_1 = \frac{\partial L}{\partial w_1} = \frac{\partial \hat{y}}{\partial h_2}\frac{\partial L}{\partial \hat{y}}$$

$$\Delta w_2 = \frac{\partial L}{\partial w_2} =$$

3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)





$$h_1 = xw_1$$

$$h_2 = \sigma(h_1)$$

$$\hat{y} = w_2 h_2$$

$$L = \hat{y} - y$$

$$w_1 = w_1 - \alpha \Delta w_1$$

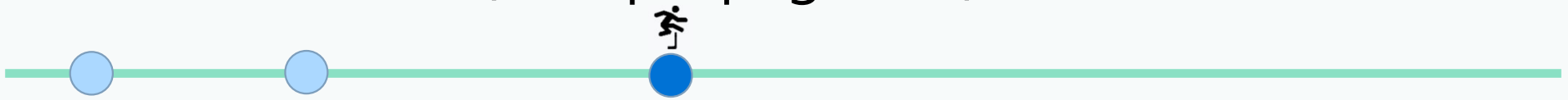
$$w_2 = w_1 - \alpha \Delta w_2$$

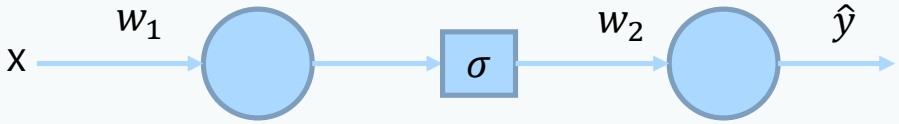
$$\Delta w_1 = \frac{\partial L}{\partial w_1} = \frac{\partial h_2}{\partial h_1}\frac{\partial \hat{y}}{\partial h_2}\frac{\partial L}{\partial \hat{y}}$$

$$\Delta w_2 = \frac{\partial L}{\partial w_2} =$$

3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)





$$h_1 = xw_1$$

$$h_2 = \sigma(h_1)$$

$$\hat{y} = w_2 h_2$$

$$L = \hat{y} - y$$

$$w_1 = w_1 - \alpha \Delta w_1$$

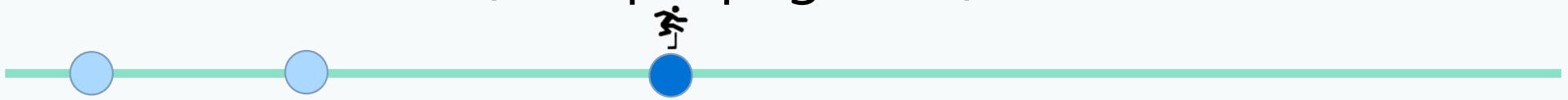
$$w_2 = w_1 - \alpha \Delta w_2$$

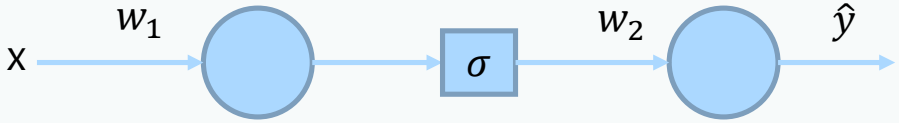
$$\Delta w_1 = \frac{\partial L}{\partial w_1} = \frac{\partial h_1}{\partial w_1}\frac{\partial h_2}{\partial h_1}\frac{\partial \hat{y}}{\partial h_2}\frac{\partial L}{\partial \hat{y}}$$

$$\Delta w_2 = \frac{\partial L}{\partial w_2} =$$

3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)





$$h_1 = xw_1$$

$$h_2 = \sigma(h_1)$$

$$\hat{y} = w_2 h_2$$

$$L = \hat{y} - y$$

$$w_1 = w_1 - \alpha \Delta w_1$$

$$w_2 = w_1 - \alpha \Delta w_2$$

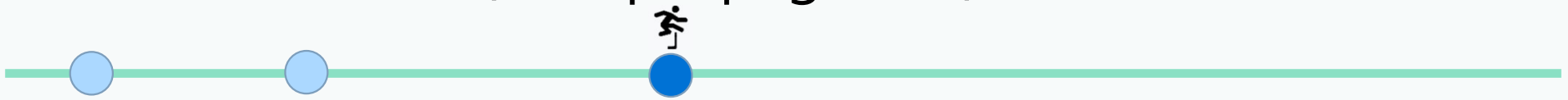
$$\Delta w_1 = \frac{\partial L}{\partial w_1} = \frac{\partial h_1}{\partial w_1}\frac{\partial h_2}{\partial h_1}\frac{\partial \hat{y}}{\partial h_2}\frac{\partial L}{\partial \hat{y}}$$

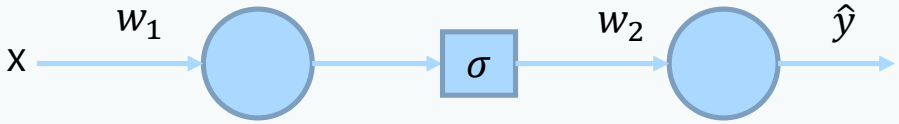
Chain Rule

$$\Delta w_2 = \frac{\partial L}{\partial w_2} =$$

3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)





$$h_1 = xw_1$$

$$h_2 = \sigma(h_1)$$

$$\hat{y} = w_2 h_2$$

$$\boxed{L = \hat{y} - y}$$

$$w_1 = w_1 - \alpha \Delta w_1$$

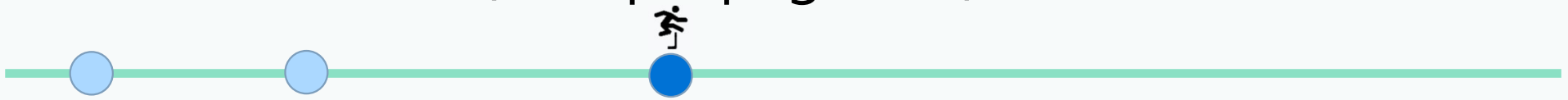
$$w_2 = w_1 - \alpha \Delta w_2$$

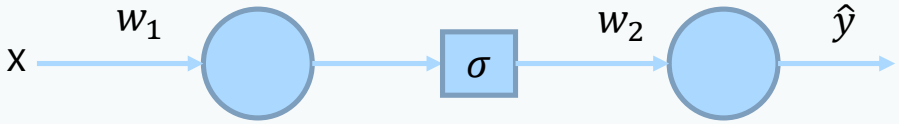
$$\Delta w_1 = \frac{\partial L}{\partial w_1} = \frac{\partial h_1}{\partial w_1}\frac{\partial h_2}{\partial h_1}\frac{\partial \hat{y}}{\partial h_2}\frac{\partial L}{\partial \hat{y}}$$

$$\Delta w_2 = \frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}}$$

3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)





$$h_1 = xw_1$$

$$h_2 = \sigma(h_1)$$

$$\hat{y} = w_2 h_2$$

$$L = \hat{y} - y$$

$$w_1 = w_1 - \alpha \Delta w_1$$

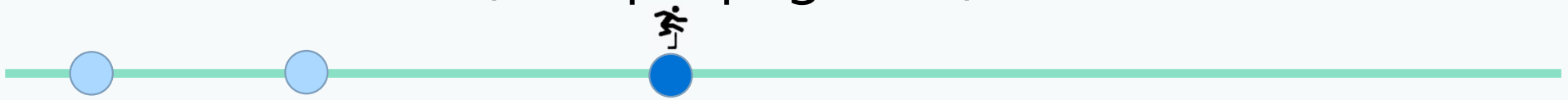
$$w_2 = w_1 - \alpha \Delta w_2$$

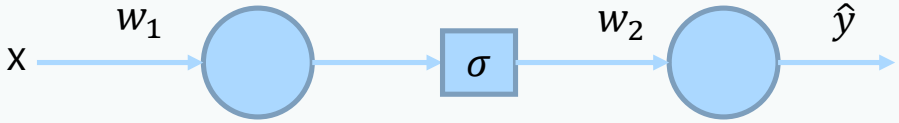
$$\Delta w_1 = \frac{\partial L}{\partial w_1} = \frac{\partial h_1}{\partial w_1}\frac{\partial h_2}{\partial h_1}\frac{\partial \hat{y}}{\partial h_2}\frac{\partial L}{\partial \hat{y}}$$

$$\Delta w_2 = \frac{\partial L}{\partial w_2} = \frac{\partial \hat{y}}{\partial w_2}\frac{\partial L}{\partial \hat{y}}$$

3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)





$$h_1 = xw_1$$

$$h_2 = \sigma(h_1)$$

$$\hat{y} = w_2 h_2$$

$$L = \hat{y} - y$$

$$w_1 = w_1 - \alpha \Delta w_1$$

$$w_2 = w_1 - \alpha \Delta w_2$$

$$\Delta w_1 = \frac{\partial L}{\partial w_1} = \frac{\partial h_1}{\partial w_1}\frac{\partial h_2}{\partial h_1}\frac{\partial \hat{y}}{\partial h_2}\frac{\partial L}{\partial \hat{y}}$$

$$\Delta w_2 = \frac{\partial L}{\partial w_2} = \frac{\cancel{\partial \hat{y}}}{\partial w_2}\frac{\partial L}{\cancel{\partial \hat{y}}}$$

Chain Rule

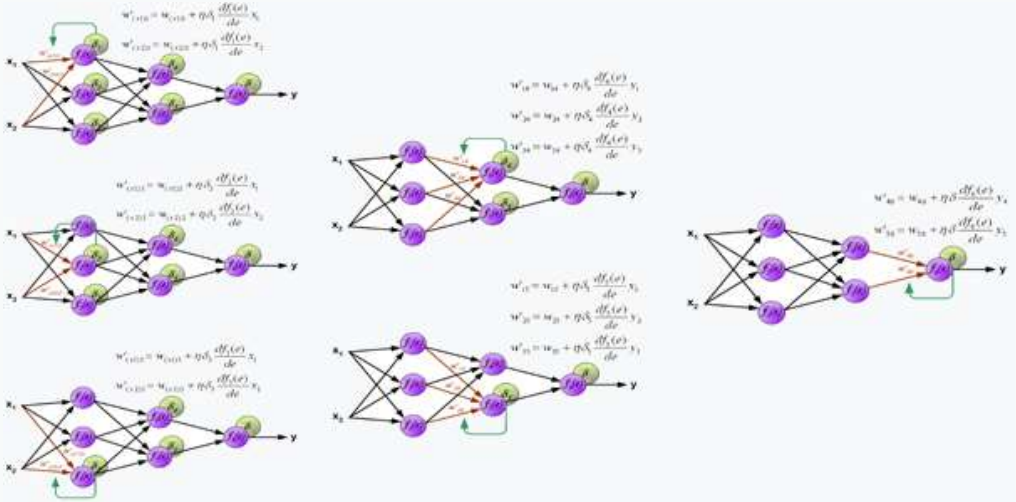
3. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957
perceptron

1969
XOR problem

1986
Multi layer Perceptron
(backpropagation)



input layer　　hidden layer　　output layer

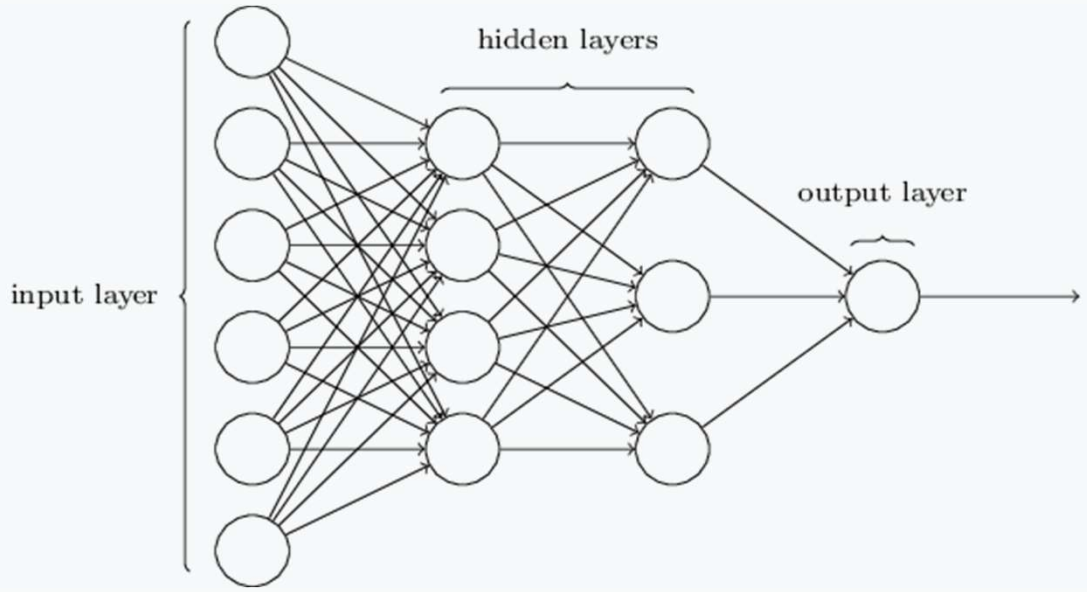
4. Neural Network 실습

- Neural Network 구조

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)



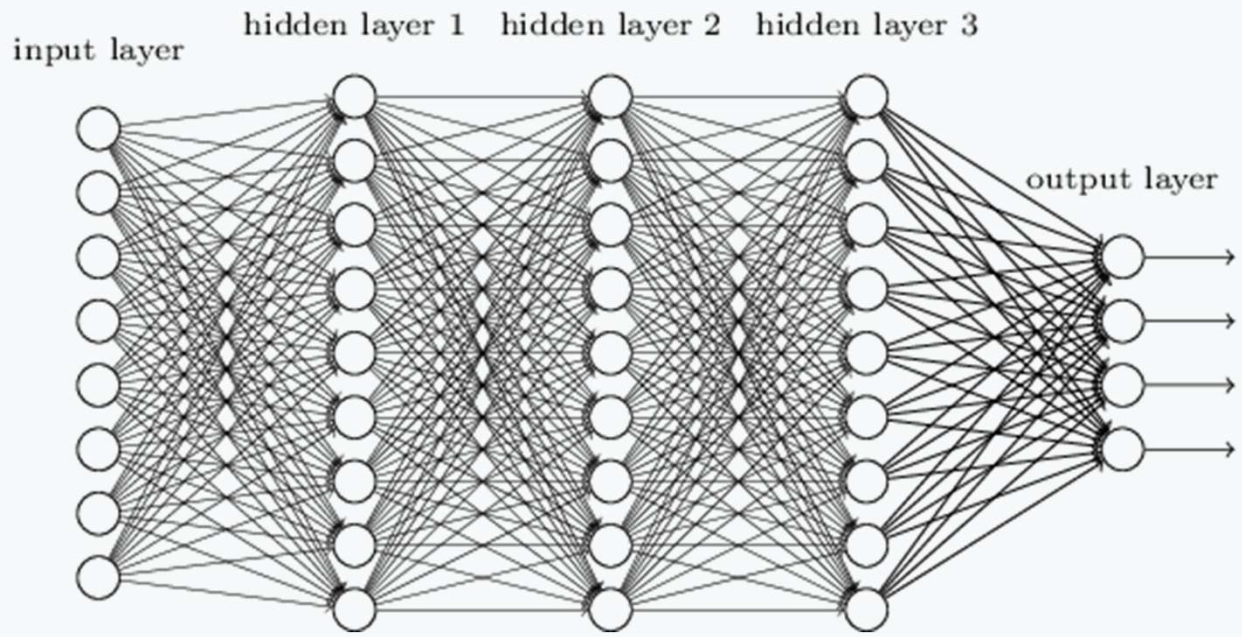
4. Neural Network 실습

- Neural Network 구조

1957 perceptron

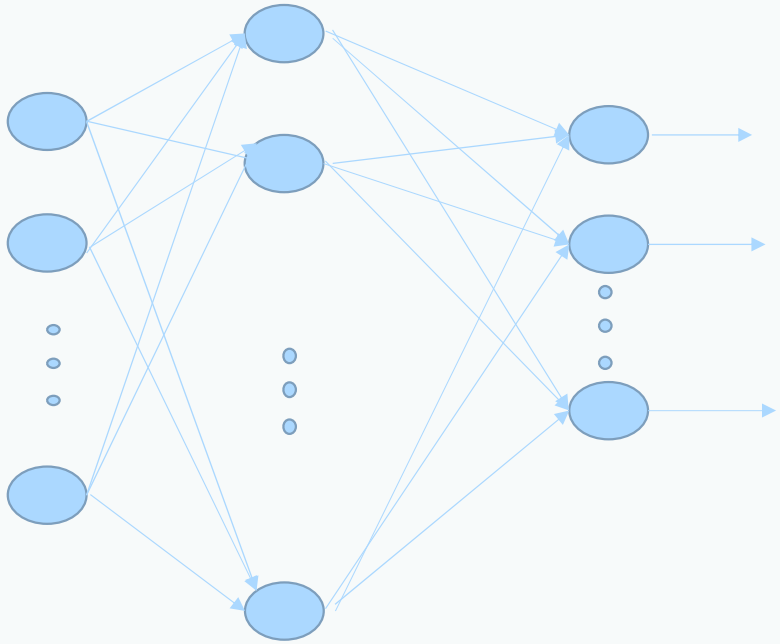
1969 XOR problem

1986 Multi layer Perceptron (backpropagation)



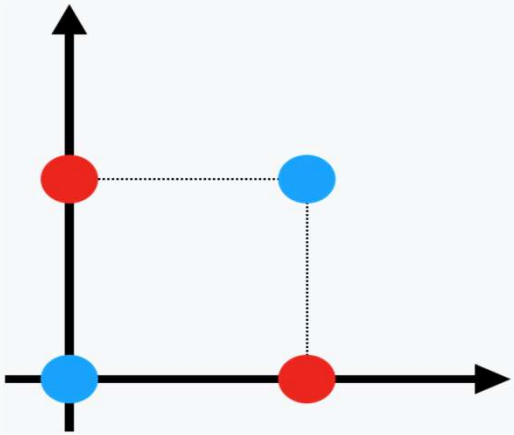
4. Neural Network 실습

- XOR Problem 해결해보기



입력층 은닉층 출력층

XOR 문제 적용



$w_1 =?,\ w_2 =?,\ b =?$

x_1	x_2	$\sum$	y
0	0		0
0	1		1
1	0		1
1	1		0

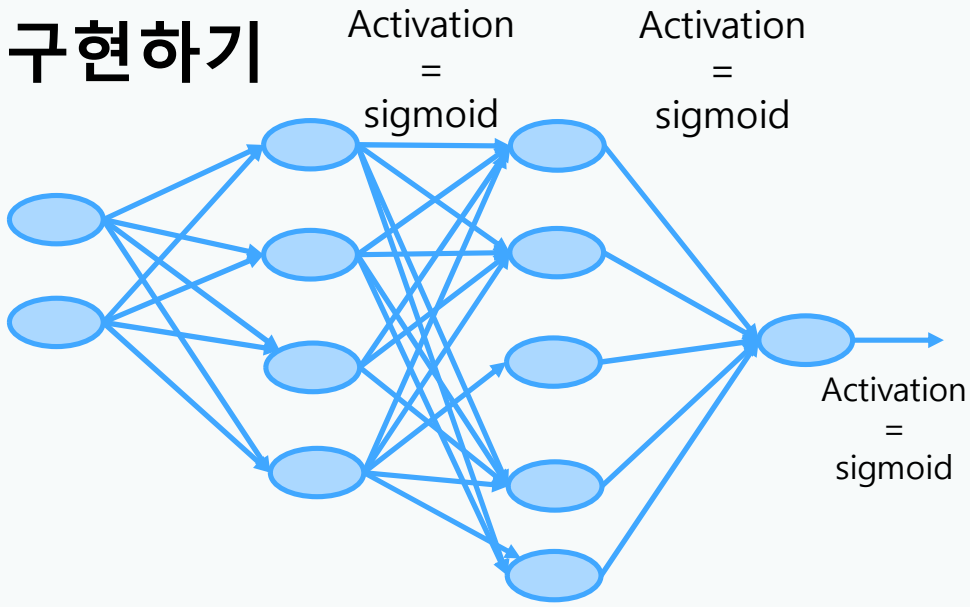
exercise_02_01_XOR_problem_solution.ipynb

practice : P_02_01_XOR_problem_solution.ipynb

4. Neural Network 실습

Neural Network 구현하기

```
## data 선언
x_data = [[0.,0.], [0.,1.], [1.,0.],[1.,1.]]
y_data = [[0.], [1.], [1.], [0.]]
test_data=[[0.5, 0.5]]


## tf.keras를 활용한 perceptron 모델 구현.
model = tf.keras.Sequential() ## 모델 선언
model.add(tf.keras.layers.Dense(4, input_dim=2, activation='sigmoid')) # 선언된 모
model.add(tf.keras.layers.Dense(5, activation='sigmoid'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid')) # 선언된 모델에 add를통해
model.summary()

# 모델 loss, 학습 방법 결정하기
optimizer=tf.keras.optimizers.SGD(learning_rate=0.5) ### 경사 하강법으로 global mi
loss=tf.keras.losses.binary_crossentropy  ## 예측값 과 정답의 오차값 정의.
metrics=tf.keras.metrics.binary_accuracy ### 학습하면서 평가할 메트릭스 선언언

# 모델 컴파일하기
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```



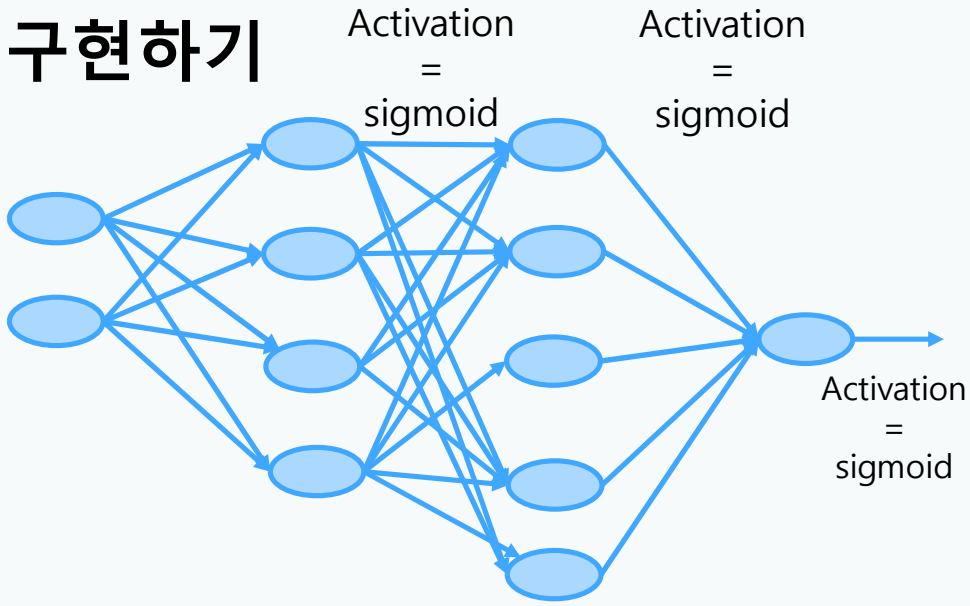
입력층	은닉층 1	은닉층 2	출력층
갯수 : 2	Perceptron (node) 4 Sigmoid	Perceptron (node) 5 Sigmoid	Perceptron (node) 1 Sigmoid

practice : p_02_01_XOR_problem_solution.ipynb

4. Neural Network 실습

Neural Network 구현하기

```
## tf.keras를 활용한 perceptron 모델 구현.
input_Layer1 = tf.keras.layers.Input(shape=(2,))
x = tf.keras.layers.Dense(4, activation='sigmoid')(input_Layer1)
x = tf.keras.layers.Dense(5, activation='sigmoid')(x)
Out_Layer= tf.keras.layers.Dense(1, activation='sigmoid')(x)
model = tf.keras.Model(inputs=[ input_Layer1 ], outputs=[Out_Layer])
model.summary()

# 모델 loss, 학습 방법 결정하기
optimizer=tf.keras.optimizers.SGD(lr=0.5) ### 경사 하강법으로 global m
loss=tf.keras.losses.binary_crossentropy  ## 예측값 과 정답의 오차값 정
metrics=tf.keras.metrics.binary_accuracy ### 학습하면서 평가할 메트릭스

# 모델 컴파일하기
model.compile(loss=loss, optimizer=optimizer, metrics=[metrics])

# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
# model.fit(x_data,y_data, epochs=1000,)
```



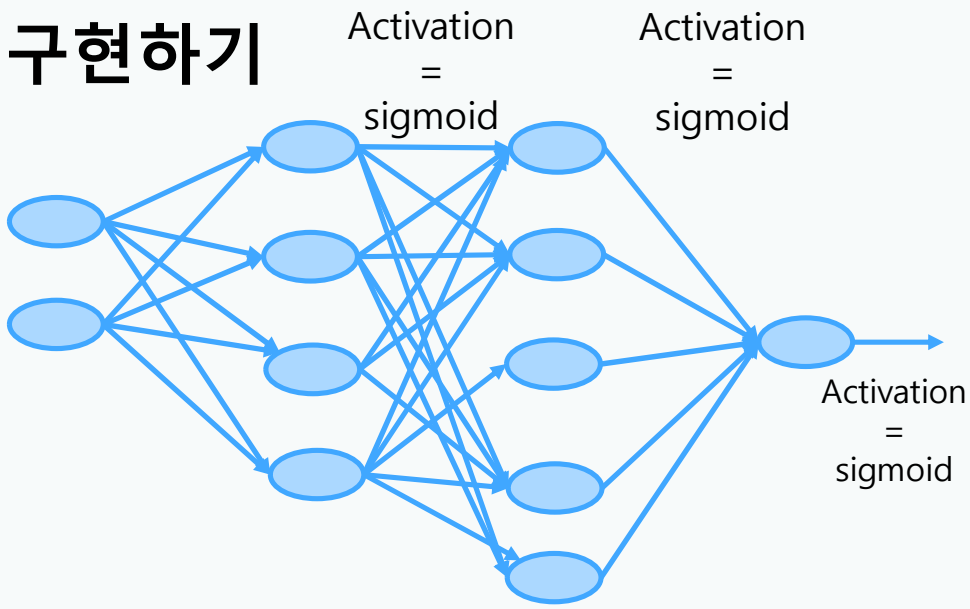
입력층	은닉층 1	은닉층 2	출력층
갯수 : 2	**Perceptron (node) 4 Sigmoid**	**Perceptron (node) 5 Sigmoid**	**Perceptron (node) 1 Sigmoid**

practice : p_02_01_XOR_problem_solution_v2.ipynb

4. Neural Network 실습

Neural Network 구현하기

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
dense (Dense)                (None, 4)                 12
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 25
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 6
=================================================================
Total params: 43
Trainable params: 43
Non-trainable params: 0
```



입력층
갯수 : 2

은닉층 1
Perceptron (node) 4
Sigmoid

은닉층 2
Perceptron (node) 5
Sigmoid

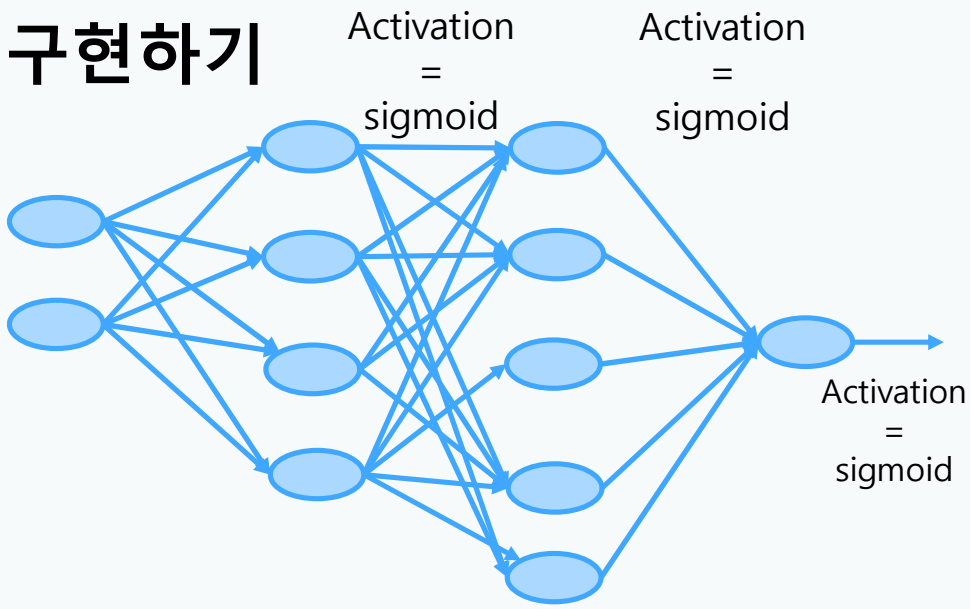
출력층
Perceptron (node) 1
Sigmoid

practice : p_02_01_XOR_problem_solution_v2.ipynb

4. Neural Network 실습

Neural Network 구현하기

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
dense (Dense)                (None, 4)                 12
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 25
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 6
=================================================================
Total params: 43
Trainable params: 43
Non-trainable params: 0
```



입력층
갯수 : 2

은닉층 1
Perceptron (node) 4
Sigmoid

은닉층 2
Perceptron (node) 5
Sigmoid

출력층
Perceptron (node) 1
Sigmoid

practice : p_02_01_XOR_problem_solution_v2.ipynb

4. Neural Network 실습

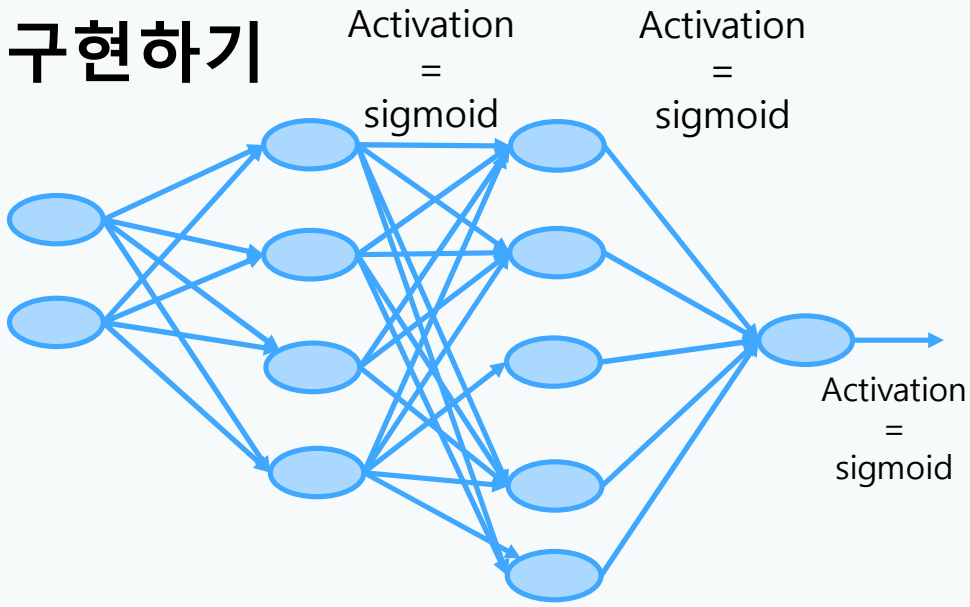
Neural Network 구현하기

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
dense (Dense)                (None, 4)                 12
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 25
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 6
=================================================================
Total params: 43
Trainable params: 43
Non-trainable params: 0
```

Batch Size

Batch Size

Batch Size



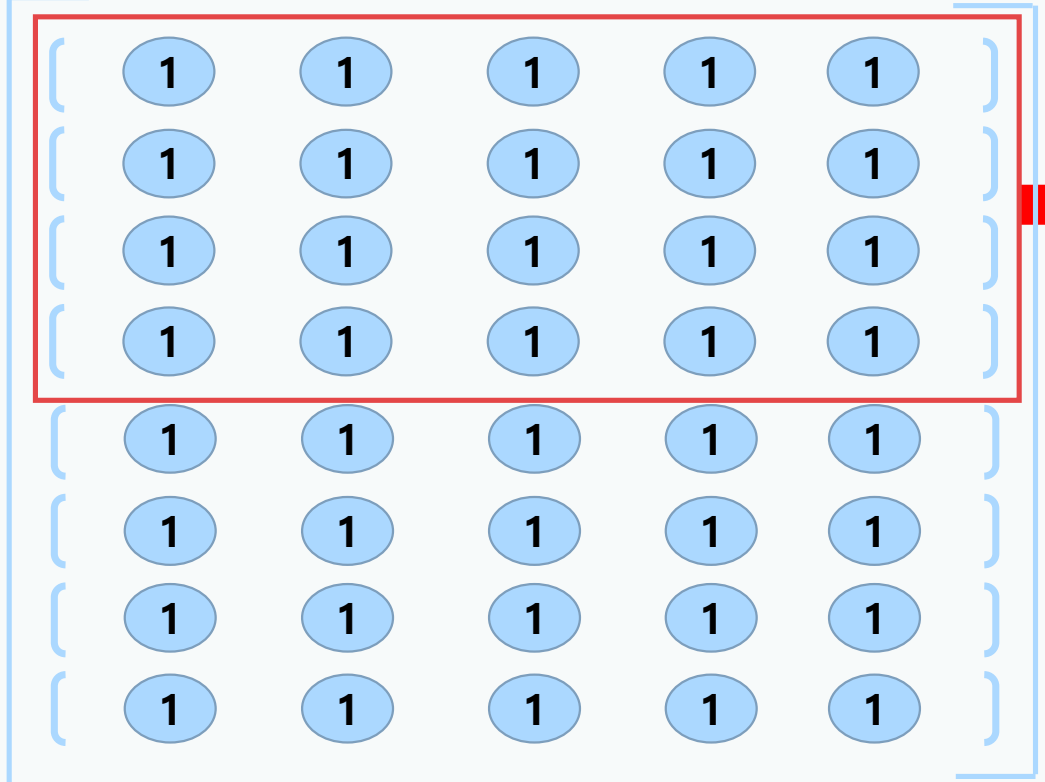
입력층	은닉층 1	은닉층 2	출력층
갯수 : 2	Perceptron (node) 4 Sigmoid	Perceptron (node) 5 Sigmoid	Perceptron (node) 1 Sigmoid

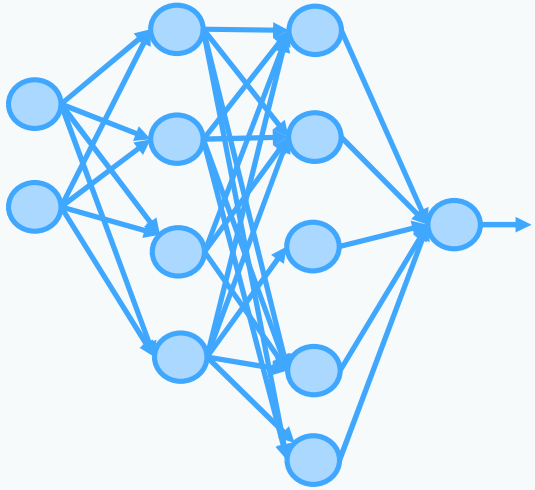
practice : p_02_01_XOR_problem_solution_v2.ipynb

4. Neural Network 실습

데이터 분리 학습 과정 살펴보기

Batch_size=4





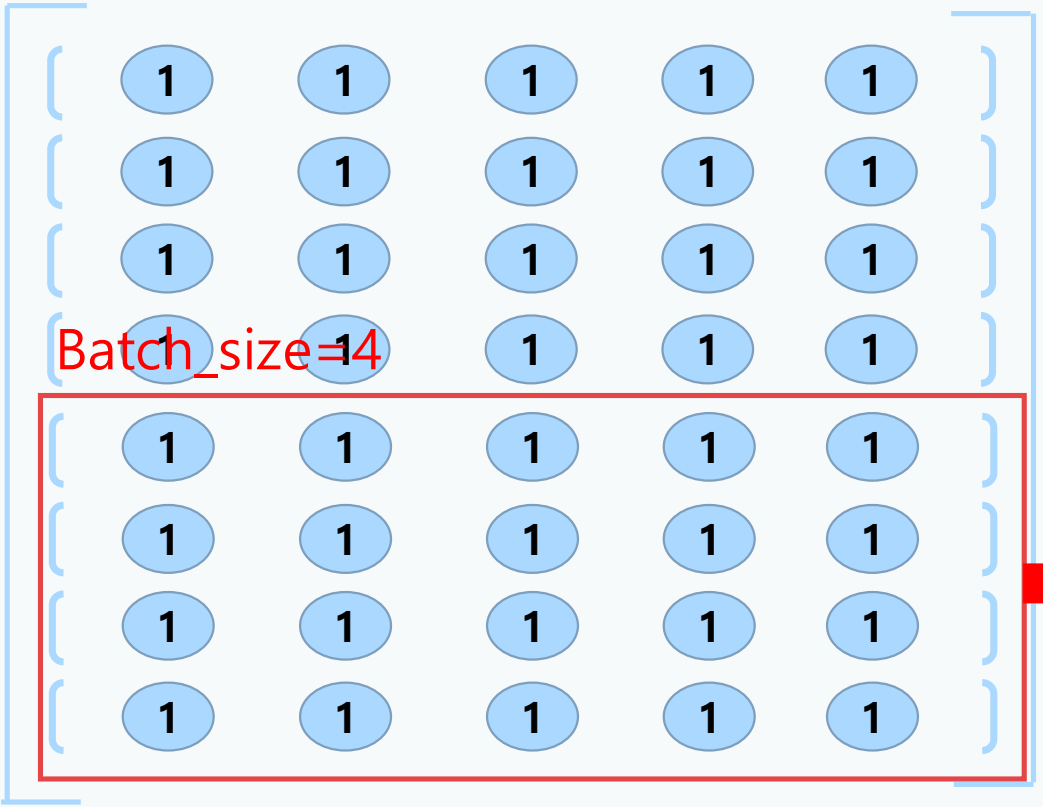
에러 계산해서
모델 업데이트

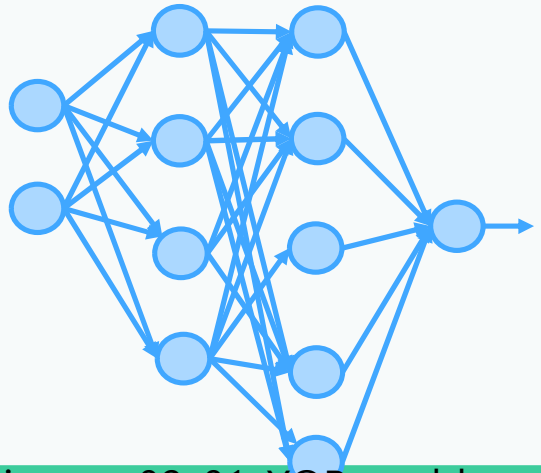
```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```


practice : p_02_01_XOR_problem_solution_v2.ipynb

4. Neural Network 실습

데이터 분리 학습 과정 살펴보기





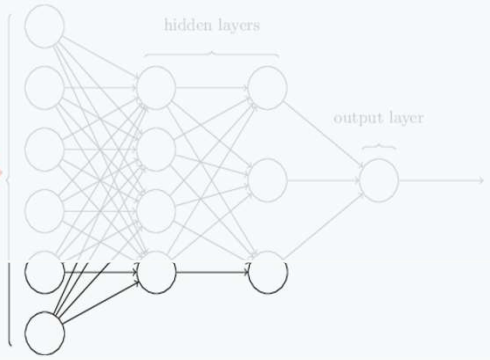
에러 계산해서
모델 업데이트

```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

practice : p_02_01_XOR_problem_solution_v2.ipynb

4. Neural Network 실습

데이터 분리 학습 과정 살펴보기



에러 계산해서
모델 업데이트

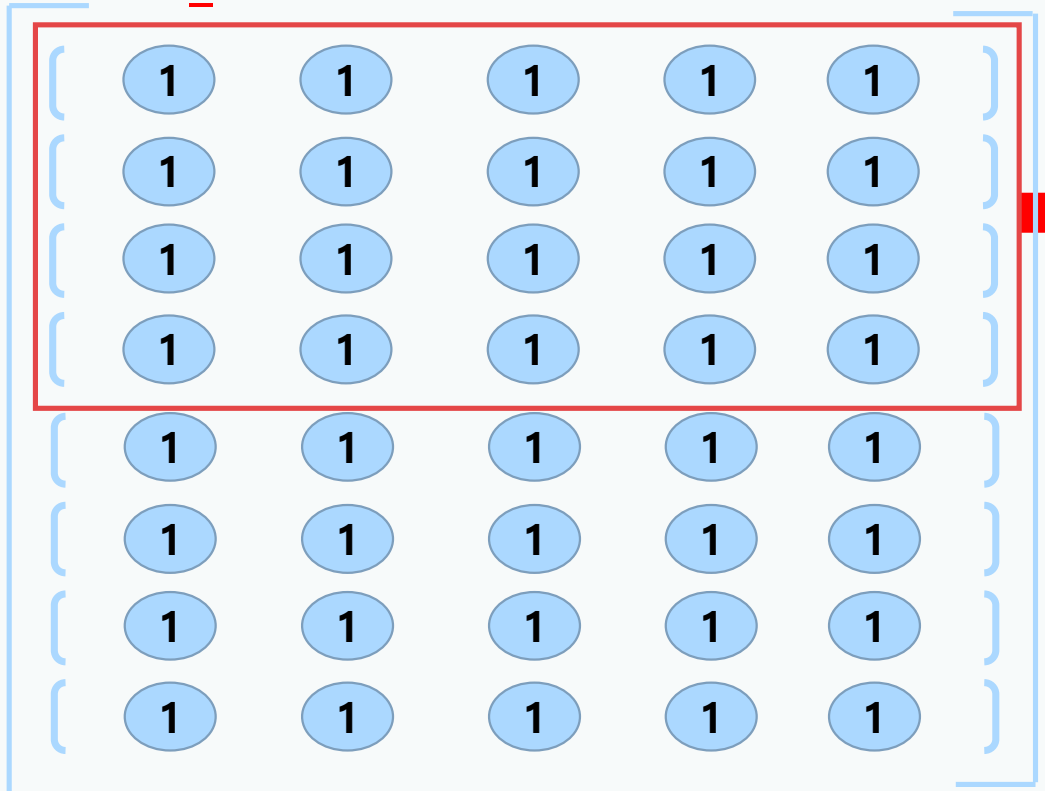
```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

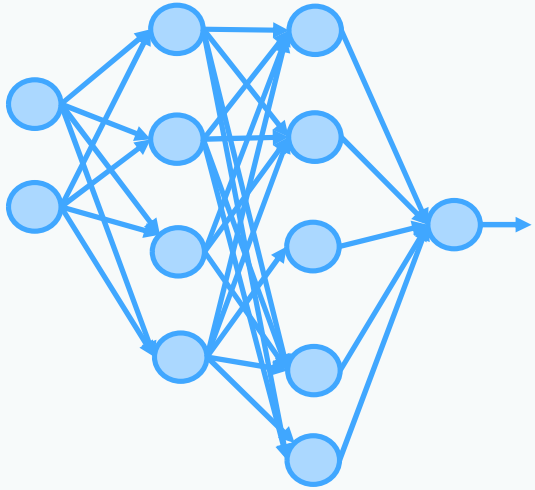

practice : p_02_01_XOR_problem_solution_v2.ipynb

4. Neural Network 실습

데이터 분리 학습 과정 살펴보기

Batch_size=4





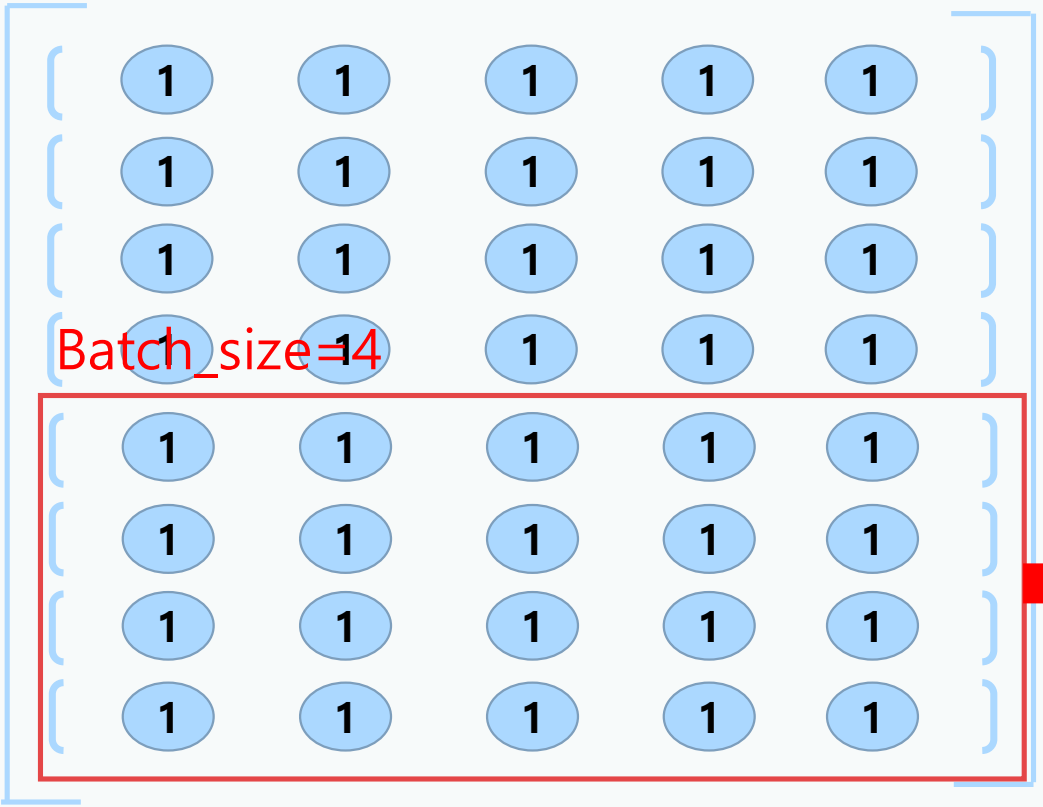
에러 계산해서
모델 업데이트

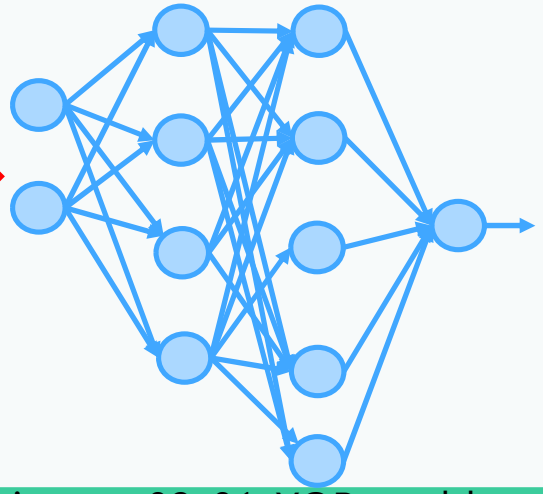
```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```


practice : p_02_01_XOR_problem_solution_v2.ipynb

4. Neural Network 실습

데이터 분리 학습 과정 살펴보기





에러 계산해서
모델 업데이트

```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```

practice : p_02_01_XOR_problem_solution_v2.ipynb

4. Neural Network 실습

데이터 분리 학습 과정 살펴보기


```
# 모델 동작하기
model.fit(x_data, y_data, epochs=2000, batch_size=4)
```


practice : p_02_01_XOR_problem_solution_v2.ipynb

5. Neural Network 실습

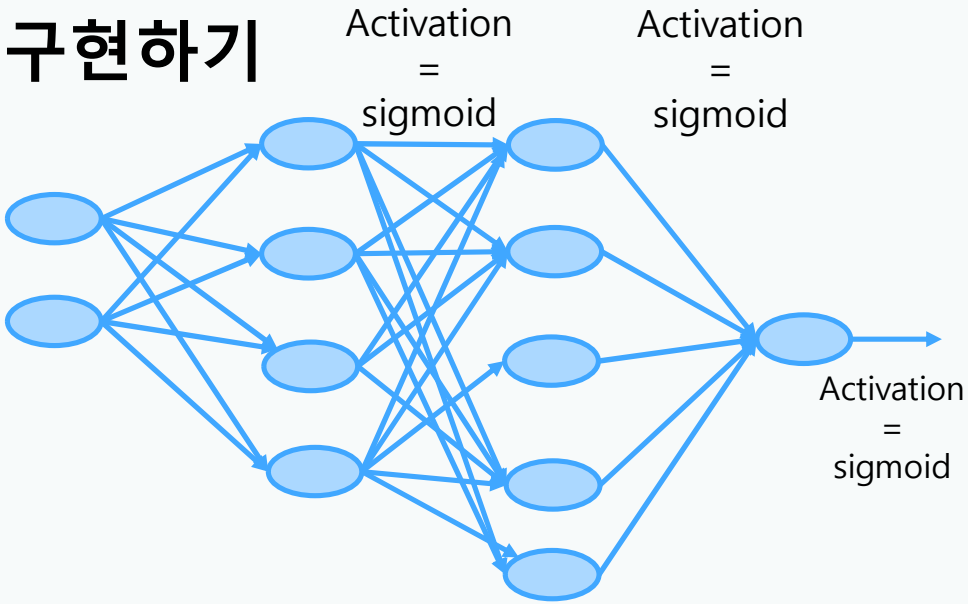
Neural Network 구현하기

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
dense (Dense)                (None, 4)                 12
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 25
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 6
=================================================================
Total params: 43
Trainable params: 43
Non-trainable params: 0
```

Batch Size

Batch Size

Batch Size



입력층
차원 :
(4, 2)

은닉층 1
Perceptron
(node)
(4, 4)
Sigmoid

은닉층 2
Perceptron
(node)
(4, 5)
Sigmoid

출력층
Perceptron
(node)
(4, 1)
Sigmoid

exercise_02_01_XOR_problem_solution.ipynb

practice : p_02_01_XOR_problem_solution_v2.ipynb

4. Neural Network 실습

여기서 잠깐!!

4. Neural Network 실습

Activation function
Sigmoid function

4. Neural Network 실습

- Logistic sigmoid
  - Perceptron

    출력을 0~1사이로 변환해 주는 함수로 사용.
    Cross entropy loss 함수에 사용.
    Binary classification에 활용
    Output Layer에 적용

- Neural Network
  - Hidden Layer

    Perceptron을 여러 개 쌓아서 비선형 으로 만들어 주기위해 사용
    Hidden Layer에 적용

  - Output Layer

    Regression / classification task를 위해 출력 activation function + loss 함수에 사용

4. Neural Network 실습

NN Regression

house_price_of_area.csv 파일을 가지고

Neural Network 를 학습하여

지역 집 가격 예측하는 모델 만들기

4. Neural Network 실습

NN Regression

house_price_of_area.csv 데이터를 살펴보겠습니다.

	x1	x2	x3	x4	x5	y
1	32	84.87	10	24.98	121.54	37.9
2	19.5	306.59	9	24.98	121.53	42.2
3	13.3	561.98	5	24.98	121.54	47.3
	…	…	…	…	…	…
415	6.5	90.45	9	24.97	121.54	63.9

샘플수 : 415
속성 : 1 (house age), Length2(distance station), Length3(number of convenience stores) 등 5개
예측값 : house_price_of_area

practice : p_02_02_house_price_of_area_prediction.ipynb

4. Neural Network 실습

```
2  import pandas as pd
3  import tensorflow as tf
4  import matplotlib.pyplot as plt
5
6  ## 데이터 읽어오기.
7  df = pd.read_csv("../dataset/house_price_of_unit_area.csv")
8  print(df.info())
9  print(df.head())
10
11 ## data copy
12 dataset=df.copy()
13
14 ## 데이터에서 label 데이터 추출
15 label_data=dataset.pop("house price of unit area")
16
17 # 모델의 설계
18 input_Layer = tf.keras.layers.Input(shape=(5,))
19 x = tf.keras.layers.Dense(50, activation='sigmoid')(input_Layer)
20 x= tf.keras.layers.Dense(100, activation='sigmoid')(x)
21 x= tf.keras.layers.Dense(300, activation='sigmoid')(x)
22 Out_Layer= tf.keras.layers.Dense(1, activation=None)(x)
23
24 model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
25 model.summary()
```

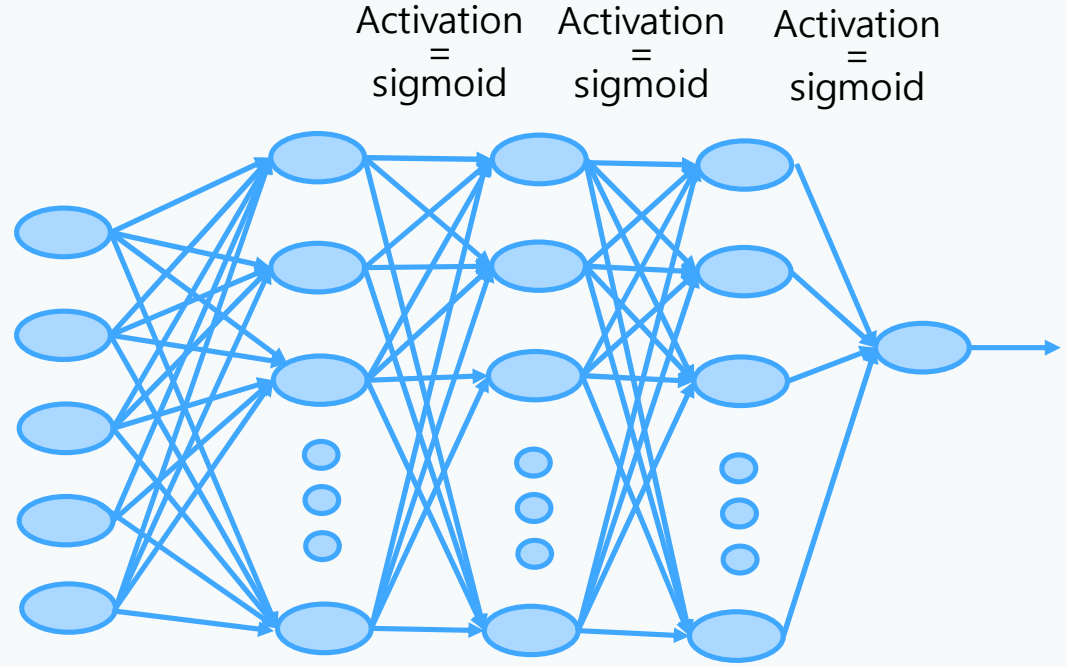
dataset	x1	x2	x3	x4	x5	y
1	32	84.87	10	24.98	121.54	37.9
2	19.5	306.59	9	24.98	121.53	42.2
3	13.3	561.98	5	24.98	121.54	47.3
	…	…	…	…	…	…
415	6.5	90.45	9	24.97	121.54	63.9

dataset

label_data

4. Neural Network 실습

```python
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt

## 데이터 읽어오기.
df = pd.read_csv("../dataset/house_price_of_unit_area.csv")
print(df.info())
print(df.head())

## data copy
dataset=df.copy()

## 데이터에서 label 데이터 추출
label_data=dataset.pop("house price of unit area")

# 모델의 설계
input_Layer = tf.keras.layers.Input(shape=(5,))
x = tf.keras.layers.Dense(50, activation='sigmoid')(input_Layer)
x= tf.keras.layers.Dense(100, activation='sigmoid')(x)
x= tf.keras.layers.Dense(300, activation='sigmoid')(x)
Out_Layer= tf.keras.layers.Dense(1, activation=None)(x)

model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()
```



입력층
갯수 : 5

은닉층 1
Perceptron (node) 50
Sigmoid

은닉층 2
Perceptron (node) 100
Sigmoid

은닉층 3
Perceptron (node) 300
Sigmoid

출력층
Perceptron (node) 1
Sigmoid

4. Neural Network 실습

NN Regression

집값 예측하기

BostonHousing.csv 파일을 가지고

Neural Network 를 학습하여

집값 예측하는 모델 만들기

4. Neural Network 실습

NN Regression

BostonHousing.csv 데이터를 살펴보겠습니다.

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
1	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
2	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
3	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
	...	...	...	...	...	...	...	...	...	...	...	...	...	...
506	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

샘플수 : 506
속성 : CRIM(범죄율), ZN(거주지역비율), INDUS(토지비율),CHAS(찰스강 경계위치),NOX(일산화질소값),RM(방의갯수) ... 등 총 13개
예측값 : MEDV(집값: 단위는 $1,000)

exersice : 02_02_house_prediction.ipynb

4. Neural Network 실습

NN Regression

성능이 잘 나오나요~?

4. Neural Network 실습

NN Regression

생각보다 성능이 잘 안나옵니다..

왜 일까요??

4. Neural Network 실습

NN Regression

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
1	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
2	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
3	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
	…	…	…	…	…	…	…	…	…	…	…	…	…	…
506	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

	Length1	Length2	Length3	Height	Width	Weight
1	23.2	25.4	30	11.52	4.02	242
2	24	26.3	31.2	12.48	4.3056	290
3	23.9	26.5	31.1	12.3778	4.6961	340
	…	…	…	…	…	…
159	13.8	15	16.2	2.9322	1.8792	19.9

4. Neural Network 실습

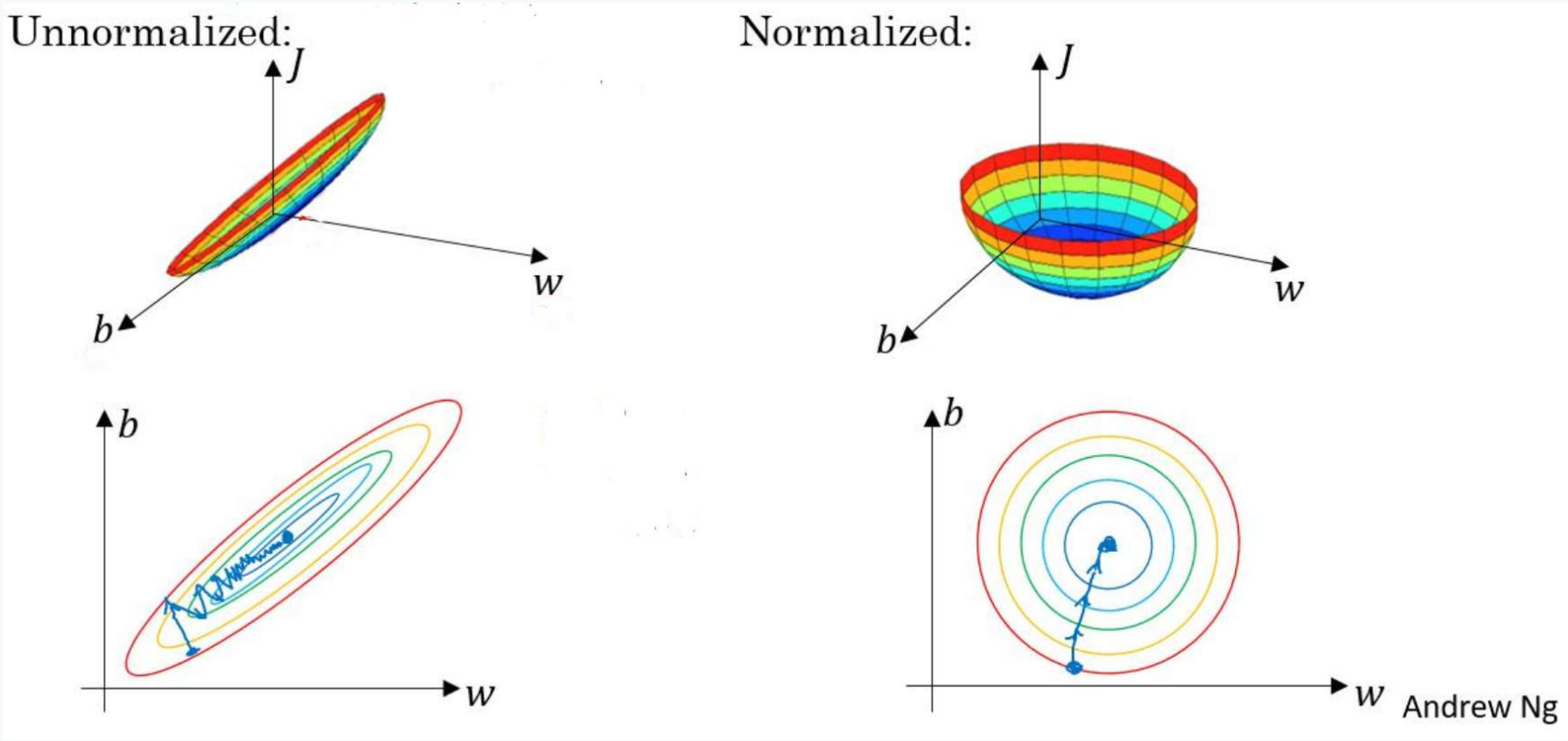
NN Regression

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
1	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
2	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
3	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
	...	...	...	...	...	...	...	...	...	...	...	...	...	...
506	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

	Length1	Length2	Length3	Height	Width	Weight
1	23.2	25.4	30	11.52	4.02	242
2	24	26.3	31.2	12.48	4.3056	290
3	23.9	26.5	31.1	12.3778	4.6961	340
	...	...	...	...	...	...
159	13.8	15	16.2	2.9322	1.8792	19.9

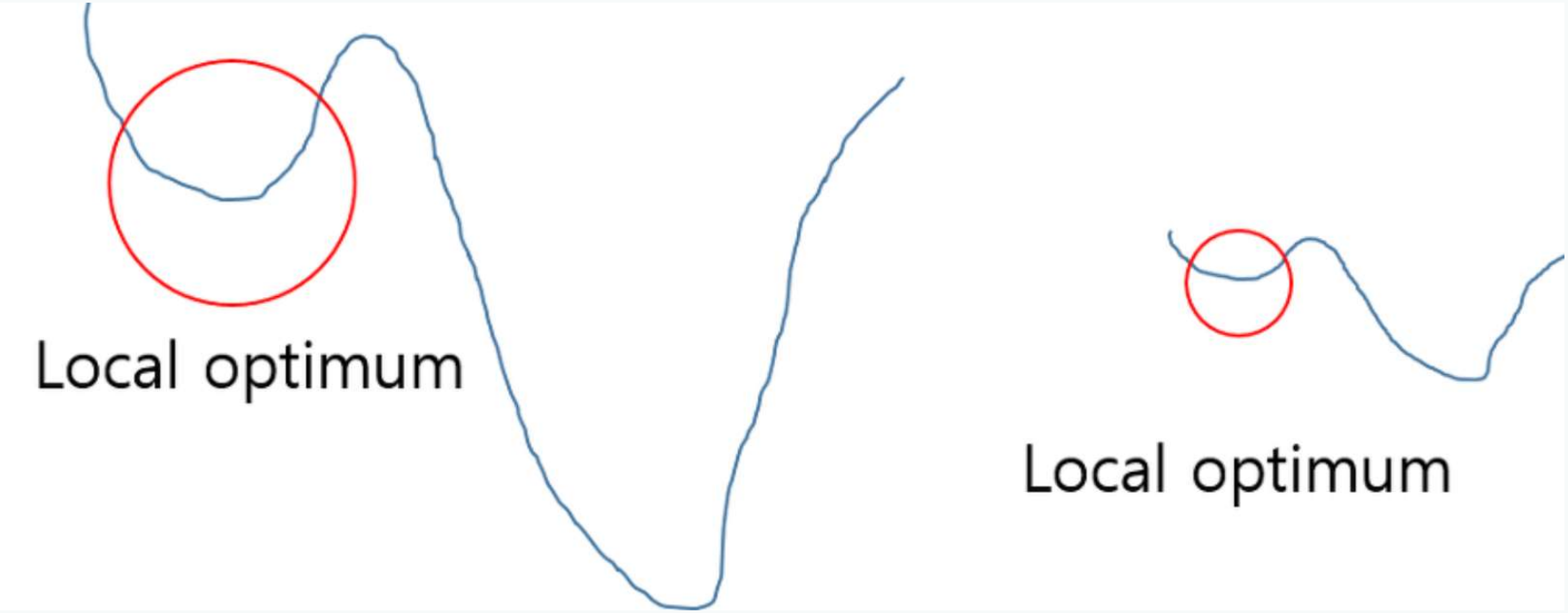
4. Neural Network 실습

NN Regression -normalization



4. Neural Network 실습

NN Regression



4. Neural Network 실습

Normalization 대표적인 방법

1. Min max : 일정범위(일반적으로 0~1) 사이로 scaling 함

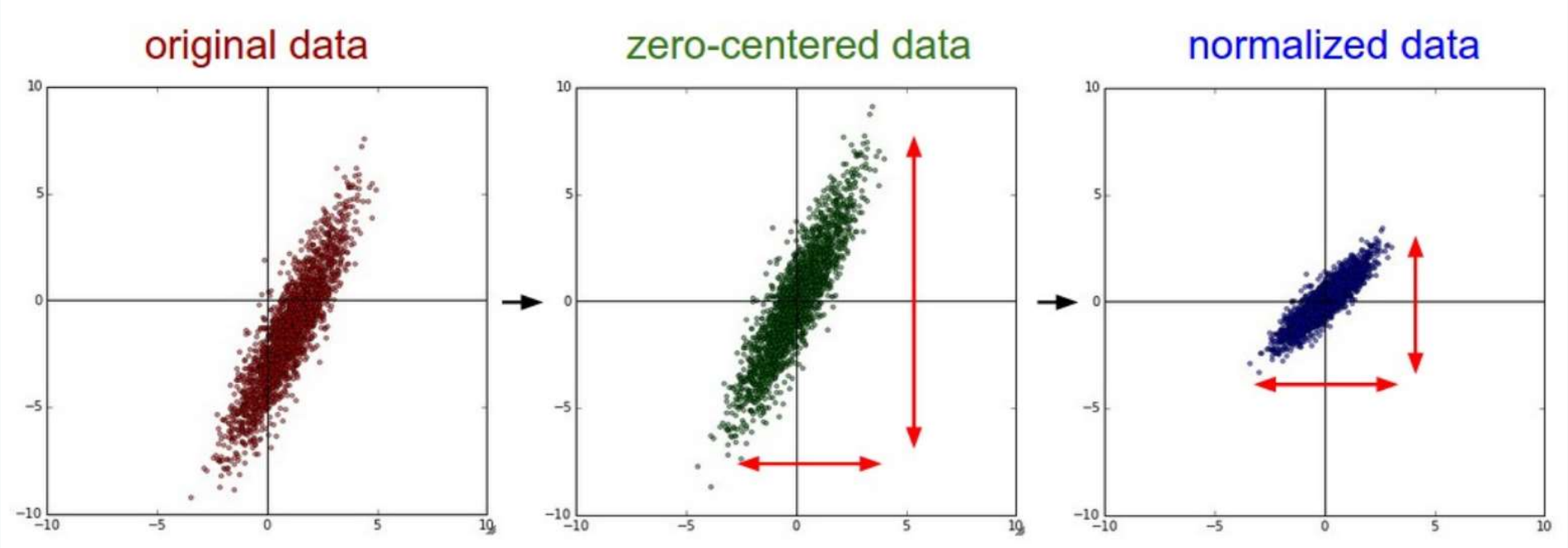
$$x = \frac{x - x_{min}}{x_{max} - x_{min}}$$

2. Standardization : mean 차감을 통해 zero-centered 화 해주고 std로 나누어 주어 데이터가 일정 범위안에 머무르게 함

$$x = \frac{x - x_{mean}}{x_{std}}$$

4. Neural Network 실습

Standardization **Normalization**



Standardization Normalization

Min Max Normalization

실습 하기

practice : P_02_03_normalization.ipynb
exersice : 02_03_normalization.ipynb

4. Neural Network 실습

입력과 정답분리

	A	B	C	D
1	1	10	100	2
2	2	20	200	2
3	3	30	300	2
	…	…	…	…
10	10	100	1000	2

practice : P_02_03_normalization.ipynb
exersice : 02_03_normalization.ipynb

4. Neural Network 실습

Normalization

	A	B	C	D
1	1	10	100	2
2	2	20	200	2
3	3	30	300	2
	…	…	…	…
10	10	100	1000	2

practice : P_02_03_normalization.ipynb
exersice : 02_03_normalization.ipynb

4. Neural Network 실습

```
5
6   ## 데이터 읽어오기.## 데이터 읽어오기.
7   raw_df = pd.read_csv("../dataset/test.csv")
8   print(raw_df.info())
9   print(raw_df.head())
10  dataset=raw_df.copy()
11
12  ## 데이터 분포도 확인하기.
13  sns.pairplot(dataset[["A","B","C","D"]], diag_kind="kde")
14  plt.show()
15
16  train_labels = dataset.pop("D")
17
18  ## 데이터의 min , max, mean, std 값 구하기.
19  dataset_stats = dataset.describe()
20  dataset_stats = dataset_stats.transpose()
21
22  ## data normalization
23  def min_max_norm(x):
24    return (x - dataset_stats['min']) / (dataset_stats['max'] - dataset_stats['min'])
25
26  def standard_norm(x):
27    return (x - dataset_stats['mean']) / dataset_stats['std']
28
29  min_max_norm_train_data = min_max_norm(dataset)
30  standard_norm_train_data = standard_norm(dataset)
31
```

Data 읽어오기

	A	B	C	D
1	1	10	100	2
2	2	20	200	2
3	3	30	300	2
	…	…	…	…
10	10	100	1000	2

practice : P_02_03_normalization.ipynb

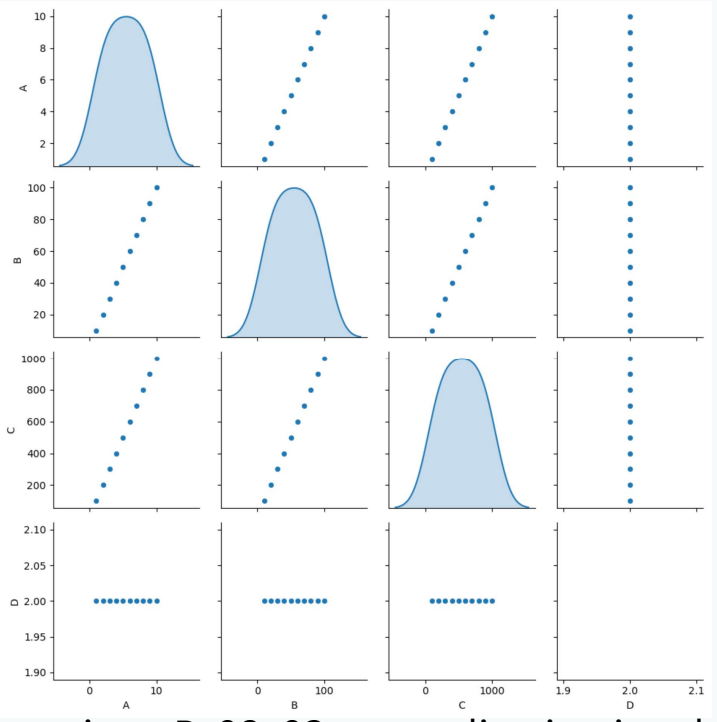
exersice : 02_03_normalization.ipynb

4. Neural Network 실습

```
5
6   ## 데이터 읽어오기.## 데이터 읽어오기.
7   raw_df = pd.read_csv("../dataset/test.csv")
8   print(raw_df.info())
9   print(raw_df.head())
10  dataset=raw_df.copy()
11
12  ## 데이터 분포도 확인하기.
13  sns.pairplot(dataset[["A","B","C","D"]], diag_kind="kde")
14  plt.show()
15
16  train_labels = dataset.pop("D")
17
18  ## 데이터의 min , max, mean, std 값 구하기.
19  dataset_stats = dataset.describe()
20  dataset_stats = dataset_stats.transpose()
21
22  ## data normalization
23  def min_max_norm(x):
24    return (x - dataset_stats['min']) / (dataset_stats['max'] - dataset_stats['min'])
25
26  def standard_norm(x):
27    return (x - dataset_stats['mean']) / dataset_stats['std']
28
29  min_max_norm_train_data = min_max_norm(dataset)
30  standard_norm_train_data = standard_norm(dataset)
31
```

Data 읽어오기

Data 분포도



practice : P_02_03_normalization.ipynb

exersice : 02_03_normalization.ipynb

4. Neural Network 실습

```
5
6   ## 데이터 읽어오기.## 데이터 읽어오기.
7   raw_df = pd.read_csv("../dataset/test.csv")
8   print(raw_df.info())
9   print(raw_df.head())
10  dataset=raw_df.copy()
11
12  ## 데이터 분포도 확인하기.
13  sns.pairplot(dataset[["A","B","C","D"]], diag_kind="kde")
14  plt.show()
15
16  train_labels = dataset.pop("D")
17
18  ## 데이터의 min , max, mean, std 값 구하기.
19  dataset_stats = dataset.describe()
20  dataset_stats = dataset_stats.transpose()
21
22  ## data normalization
23  def min_max_norm(x):
24    return (x - dataset_stats['min']) / (dataset_stats['max'] - dataset_stats['min'])
25
26  def standard_norm(x):
27    return (x - dataset_stats['mean']) / dataset_stats['std']
28
29  min_max_norm_train_data = min_max_norm(dataset)
30  standard_norm_train_data = standard_norm(dataset)
31
```

Data 읽어오기

Data 분포도

입력/정답분리

A	B	C	D
1	10	100	2
2	20	200	2
3	30	300	2
...	...	...	...
10	100	1000	2

practice : P_02_03_normalization.ipynb

exersice : 02_03_normalization.ipynb

4. Neural Network 실습

```
5
6   ## 데이터 읽어오기.## 데이터 읽어오기.
7   raw_df = pd.read_csv("../dataset/test.csv")
8   print(raw_df.info())
9   print(raw_df.head())
10  dataset=raw_df.copy()
11
12  ## 데이터 분포도 확인하기.
13  sns.pairplot(dataset[["A","B","C","D"]], diag_kind="kde")
14  plt.show()
15
16  train_labels = dataset.pop("D")
17
18  ## 데이터의 min , max, mean, std 값 구하기.
19  dataset_stats = dataset.describe()
20  dataset_stats = dataset_stats.transpose()
21
22  ## data normalization
23  def min_max_norm(x):
24    return (x - dataset_stats['min']) / (dataset_stats['max'] - dataset_stats['min'])
25
26  def standard_norm(x):
27    return (x - dataset_stats['mean']) / dataset_stats['std']
28
29  min_max_norm_train_data = min_max_norm(dataset)
30  standard_norm_train_data = standard_norm(dataset)
31
```

Data 읽어오기

Data 분포도

입력/정답분리

데이터 특성 값 구하기

	A	B	C	D
count	10.00000	10.000000	10.000000	10.0
mean	5.50000	55.000000	550.000000	2.0
std	3.02765	30.276504	302.765035	0.0
min	1.00000	10.000000	100.000000	2.0
25%	3.25000	32.500000	325.000000	2.0
50%	5.50000	55.000000	550.000000	2.0
75%	7.75000	77.500000	775.000000	2.0
max	10.00000	100.000000	1000.000000	2.0

practice : P_02_03_normalization.ipynb

exersice : 02_03_normalization.ipynb

4. Neural Network 실습

```
5
6   ## 데이터 읽어오기.## 데이터 읽어오기.
7   raw_df = pd.read_csv("../dataset/test.csv")
8   print(raw_df.info())
9   print(raw_df.head())
10  dataset=raw_df.copy()
11
12  ## 데이터 분포도 확인하기.
13  sns.pairplot(dataset[["A","B","C","D"]], diag_kind="kde")
14  plt.show()
15
16  train_labels = dataset.pop("D")
17
18  ## 데이터의 min , max, mean, std 값 구하기.
19  dataset_stats = dataset.describe()
20  dataset_stats = dataset_stats.transpose()
21
22  ## data normalization
23  def min_max_norm(x):
24    return (x - dataset_stats['min']) / (dataset_stats['max'] - dataset_stats['min'])
25
26  def standard_norm(x):
27    return (x - dataset_stats['mean']) / dataset_stats['std']
28
29  min_max_norm_train_data = min_max_norm(dataset)
30  standard_norm_train_data = standard_norm(dataset)
31
```

Data 읽어오기

Data 분포도

입력/정답분리

데이터 특성 값 구하기

데이터 특성 매트릭스 행/열 바꾸기

	A	B	C	D
count	10.00000	10.000000	10.000000	10.0
mean	5.50000	55.000000	550.000000	2.0
std	3.02765	30.276504	302.765035	0.0
min	1.00000	10.000000	100.000000	2.0
25%	3.25000	32.500000	325.000000	2.0
50%	5.50000	55.000000	550.000000	2.0
75%	7.75000	77.500000	775.000000	2.0
max	10.00000	100.000000	1000.000000	2.0

	count	mean	std	min	25%	50%	75%	max
A	10.0	5.5	3.027650	1.0	3.25	5.5	7.75	10.0
B	10.0	55.0	30.276504	10.0	32.50	55.0	77.50	100.0
C	10.0	550.0	302.765035	100.0	325.00	550.0	775.00	1000.0

practice : P_02_03_normalization.py
exersice : 02_03_normalization.py

4. Neural Network 실습

```
5
6   ## 데이터 읽어오기.## 데이터 읽어오기.
7   raw_df = pd.read_csv("../dataset/test.csv")
8   print(raw_df.info())
9   print(raw_df.head())
10  dataset=raw_df.copy()
11
12  ## 데이터 분포도 확인하기.
13  sns.pairplot(dataset[["A","B","C","D"]], diag_kind="kde")
14  plt.show()
15
16  train_labels = dataset.pop("D")
17
18  ## 데이터의 min , max, mean, std 값 구하기.
19  dataset_stats = dataset.describe()
20  dataset_stats = dataset_stats.transpose()
21
22  ## data normalization
23  def min_max_norm(x):
24    return (x - dataset_stats['min']) / (dataset_stats['max'] - dataset_stats['min'])
25
26  def standard_norm(x):
27    return (x - dataset_stats['mean']) / dataset_stats['std']
28
29  min_max_norm_train_data = min_max_norm(dataset)
30  standard_norm_train_data = standard_norm(dataset)
31
```

Data 읽어오기

Data 분포도

입력/정답분리

데이터 특성
값 구하기

데이터 특성
매트릭스 행/열 바꾸기

Min/max normalization , std normalization 함수선언

$$x = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad x = \frac{x - x_{mean}}{x_{std}}$$

exersice : 02_03_normalization.ipynb practice : P_02_03_normalization.ipynb

4. Neural Network 실습

```
5
6   ## 데이터 읽어오기.## 데이터 읽어오기.
7   raw_df = pd.read_csv("../dataset/test.csv")
8   print(raw_df.info())
9   print(raw_df.head())
10  dataset=raw_df.copy()
11
12  ## 데이터 분포도 확인하기.
13  sns.pairplot(dataset[["A","B","C","D"]], diag_kind="kde")
14  plt.show()
15
16  train_labels = dataset.pop("D")
17
18  ## 데이터의 min , max, mean, std 값 구하기.
19  dataset_stats = dataset.describe()
20  dataset_stats = dataset_stats.transpose()
21
22  ## data normalization
23  def min_max_norm(x):
24    return (x - dataset_stats['min']) / (dataset_stats['max'] - dataset_stats['min'])
25
26  def standard_norm(x):
27    return (x - dataset_stats['mean']) / dataset_stats['std']
28
29  min_max_norm_train_data = min_max_norm(dataset)
30  standard_norm_train_data = standard_norm(dataset)
31
```

Min/max normalization , std normalization 적용

	A	B	C
0	0.000000	0.000000	0.000000
1	0.111111	0.111111	0.111111
2	0.222222	0.222222	0.222222
3	0.333333	0.333333	0.333333
4	0.444444	0.444444	0.444444
5	0.555556	0.555556	0.555556
6	0.666667	0.666667	0.666667
7	0.777778	0.777778	0.777778
8	0.888889	0.888889	0.888889
9	1.000000	1.000000	1.000000

	A	B	C
0	-1.486301	-1.486301	-1.486301
1	-1.156012	-1.156012	-1.156012
2	-0.825723	-0.825723	-0.825723
3	-0.495434	-0.495434	-0.495434
4	-0.165145	-0.165145	-0.165145
5	0.165145	0.165145	0.165145
6	0.495434	0.495434	0.495434
7	0.825723	0.825723	0.825723
8	1.156012	1.156012	1.156012
9	1.486301	1.486301	1.486301

exersice : 02_03_normalization.ipynb

practice : P_02_03_normalization.ipynb

4. Neural Network 실습

NN Regression

앞에서 실습하였던 지역 집값 예측 실습 코드에

normalization을 적용해 보겠습니다.

practice : P_02_04_ house_price_of_area_prediction_with_normalization.ipynb

4. Neural Network 실습

NN Regression

```
## 데이터 읽어오기.## 데이터 읽어오기.
raw_df = pd.read_csv("../dataset/house_price_of_unit_area.csv")
print(raw_df.info())
print(raw_df.head())

## data copy
dataset=raw_df.copy()
```

Pandas로 데이터 읽어오기

```
## data 분포도 확인
sns.pairplot(dataset[["house age", "distance to the nearest MRT station", "number of convenience
plt.show()
```

Data 분포도 확인하기.

```
## label data 추출
label_data=dataset.pop("house price of unit area")
```

데이터 추출 하여 입력/정답 데이터로 분리

```
## 입력 데이터 열 별로 min, max, mean, std 구하기.
dataset_stats = dataset.describe()
dataset_stats = dataset_stats.transpose()
```

데이터 normalization을 위해
값 구하기

```
## data normalization
def min_max_norm(x):
  return (x - dataset_stats['min']) / (dataset_stats['max'] - dataset_stats['min'])

def standard_norm(x):
  return (x - dataset_stats['mean']) / dataset_stats['std']

## 데이터 normalization 하기.
normed_train_data=standard_norm(dataset)
```

데이터 normalization

practice : P_02_04_ house_price_of_area_prediction_with_normalization.ipynb

4. Neural Network 실습

NN Regression

앞에서 실습하였던 집값 예측 실습 코드에

normalization을 적용해 주세요

exersice : 02_04_house_prediction_with_normalization.ipynb

4. Neural Network 실습

NN Regression

성능이 잘 나오나요?

4. Neural Network 실습

Neural Network

당연히 잘 나옵니다.

4. Neural Network 실습

Neural Network

왜 잘 나올까요??

4. Neural Network 실습

4. Neural Network 실습

Neural Network – 학습

4. Neural Network 실습

Neural Network – 학습

3월
모의고사

4월
모의고사

5월
모의고사

6월
모의고사

7월
모의고사

8월
모의고사

9월
모의고사

10월
모의고사

수능

4. Neural Network 실습

Neural Network – 학습

3월
모의고사

4월
모의고사

5월
모의고사

6월
모의고사

7월
모의고사

8월
모의고사

9월
모의고사

10월
모의고사

수능

4. Neural Network 실습

Neural Network – 학습



3월
모의고사

4월
모의고사

5월
모의고사

6월
모의고사

7월
모의고사

8월
모의고사

9월
모의고사

10월
모의고사

수능

4. Neural Network 실습

Neural Network – 학습



3월
모의고사



4월
모의고사

5월
모의고사

6월
모의고사

7월
모의고사

8월
모의고사

9월
모의고사

10월
모의고사

수능

4. Neural Network 실습

Neural Network – 학습

3월
모의고사

4월
모의고사

5월
모의고사

6월
모의고사

7월
모의고사

8월
모의고사

9월
모의고사

10월
모의고사

수능

4. Neural Network 실습

Neural Network – 학습



3월
모의고사



4월
모의고사



5월
모의고사

6월
모의고사

7월
모의고사

8월
모의고사

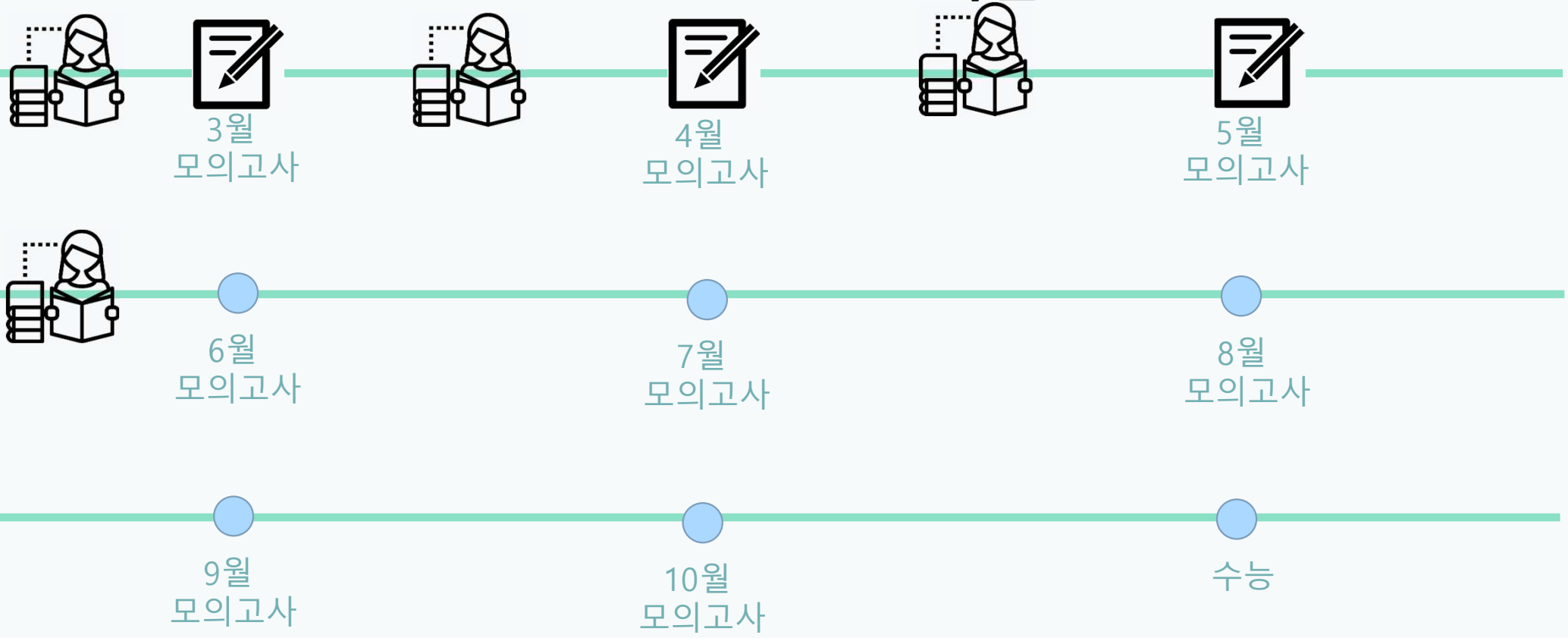
9월
모의고사

10월
모의고사

수능

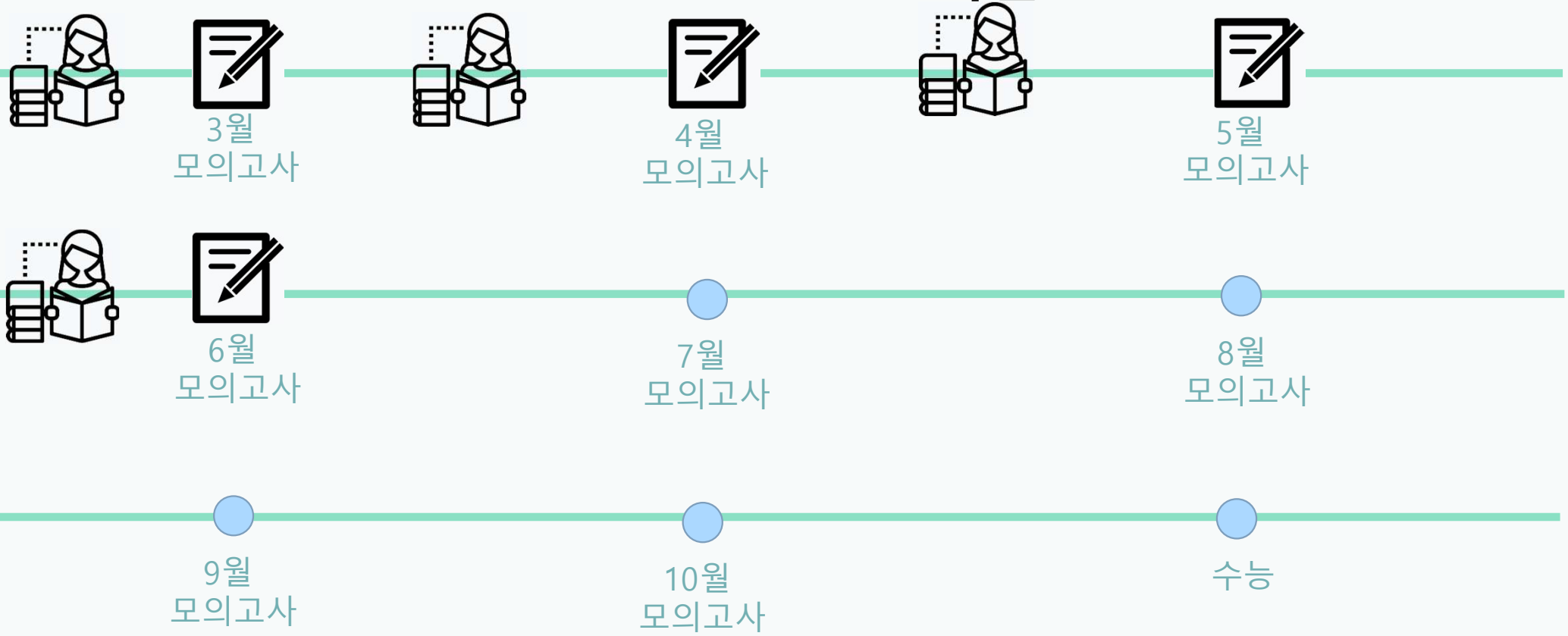
4. Neural Network 실습

Neural Network – 학습



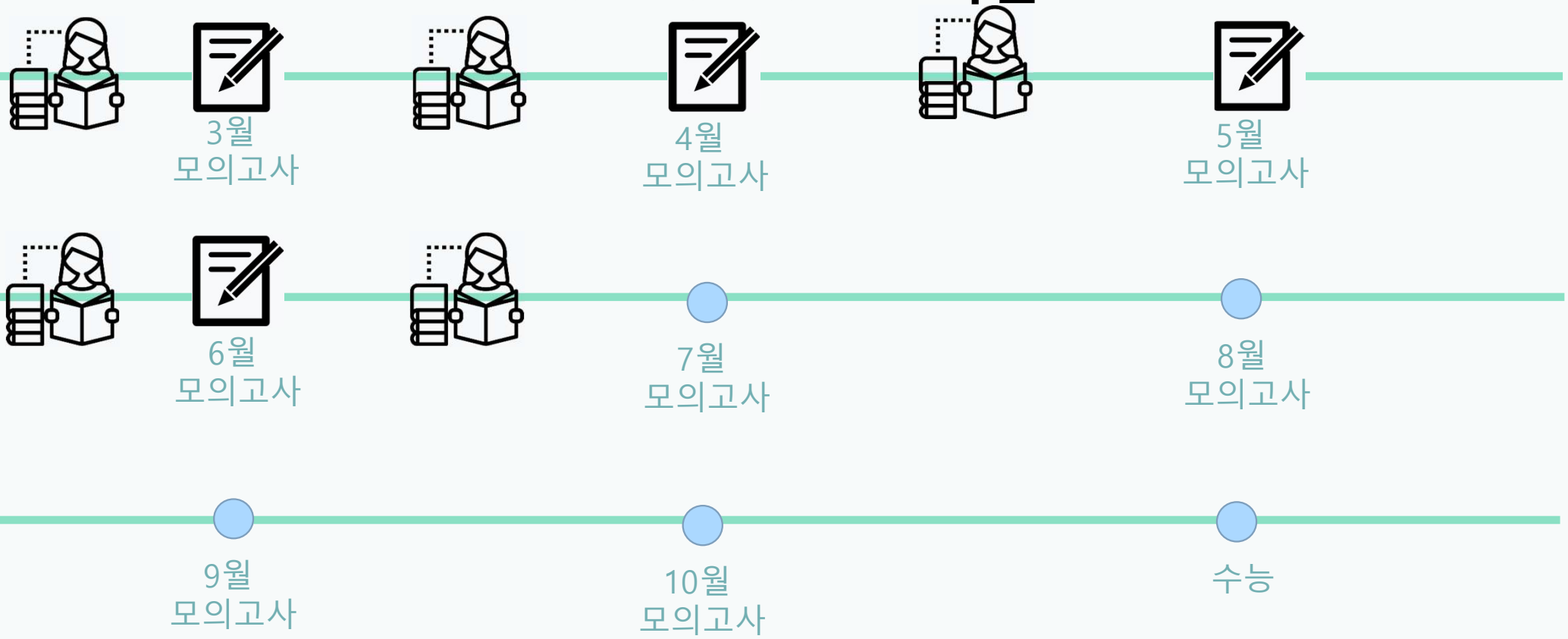
4. Neural Network 실습

Neural Network – 학습



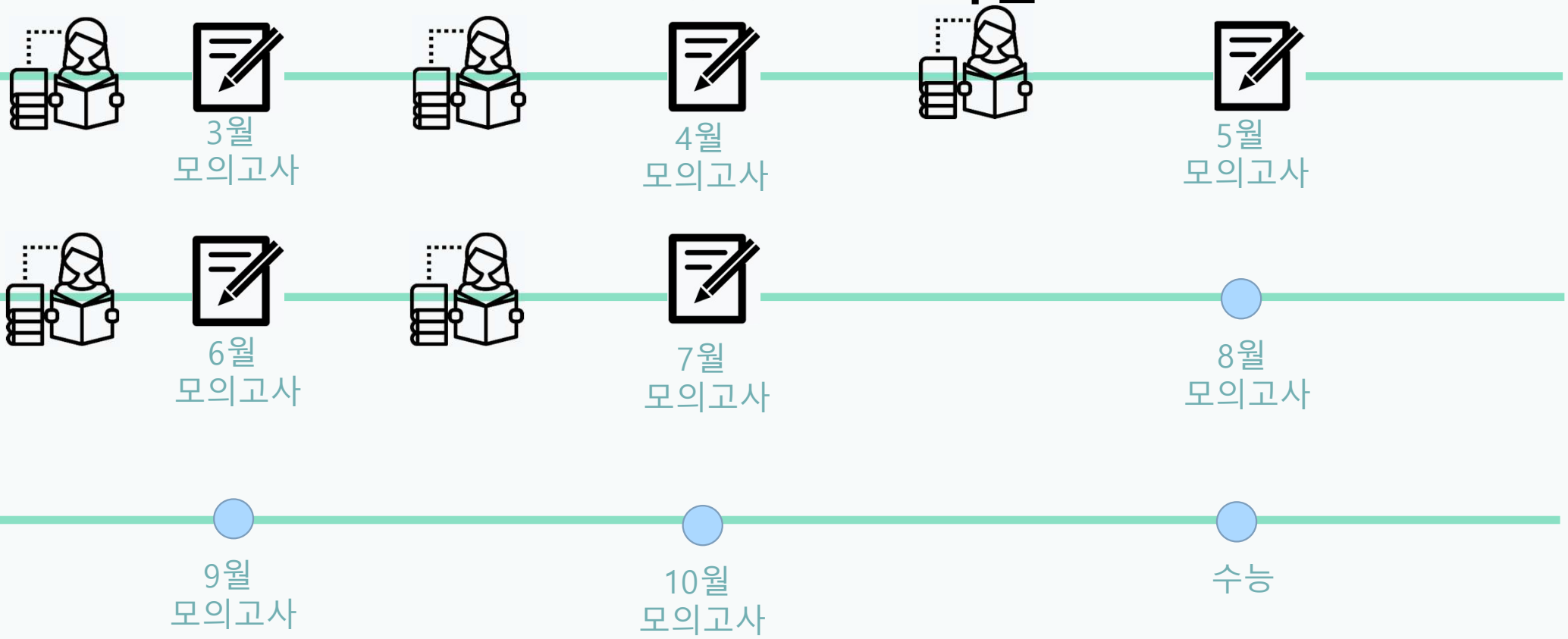
4. Neural Network 실습

Neural Network – 학습

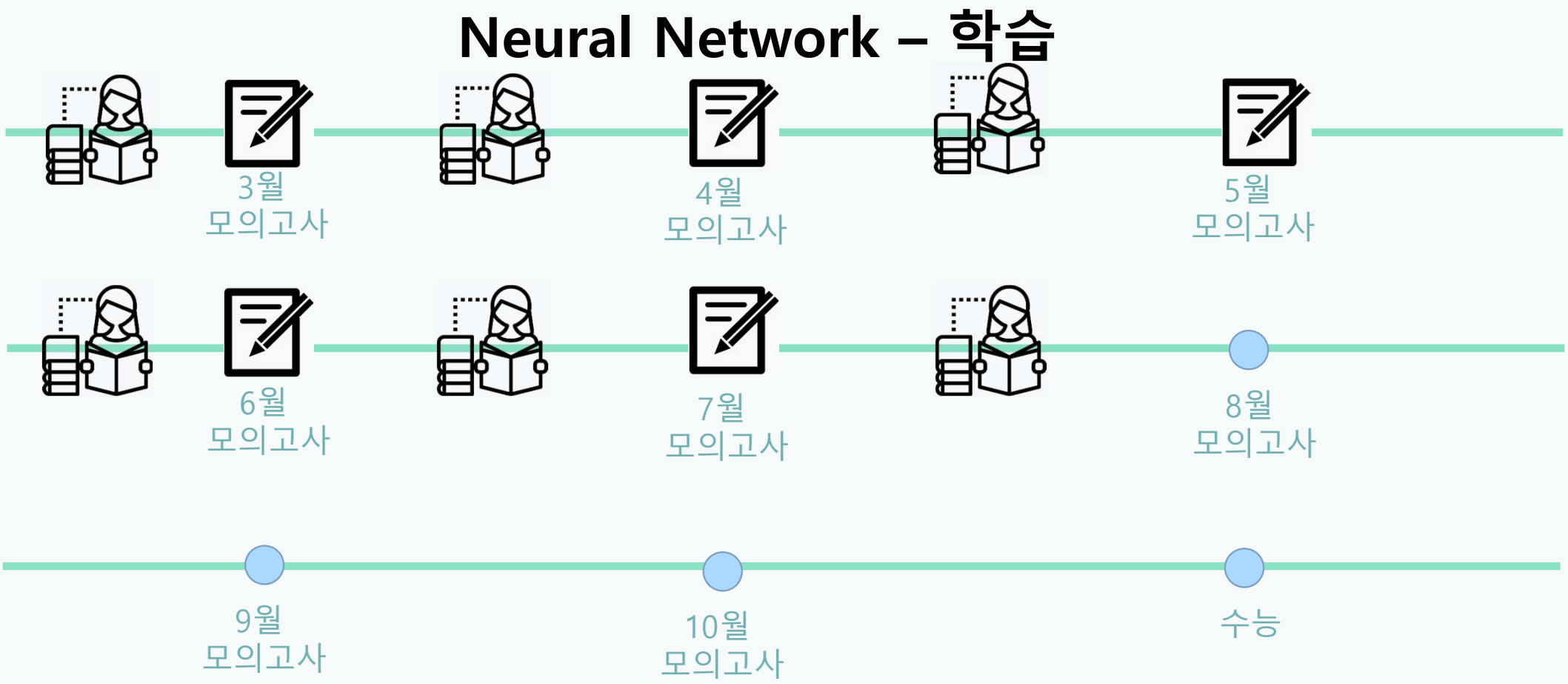


4. Neural Network 실습

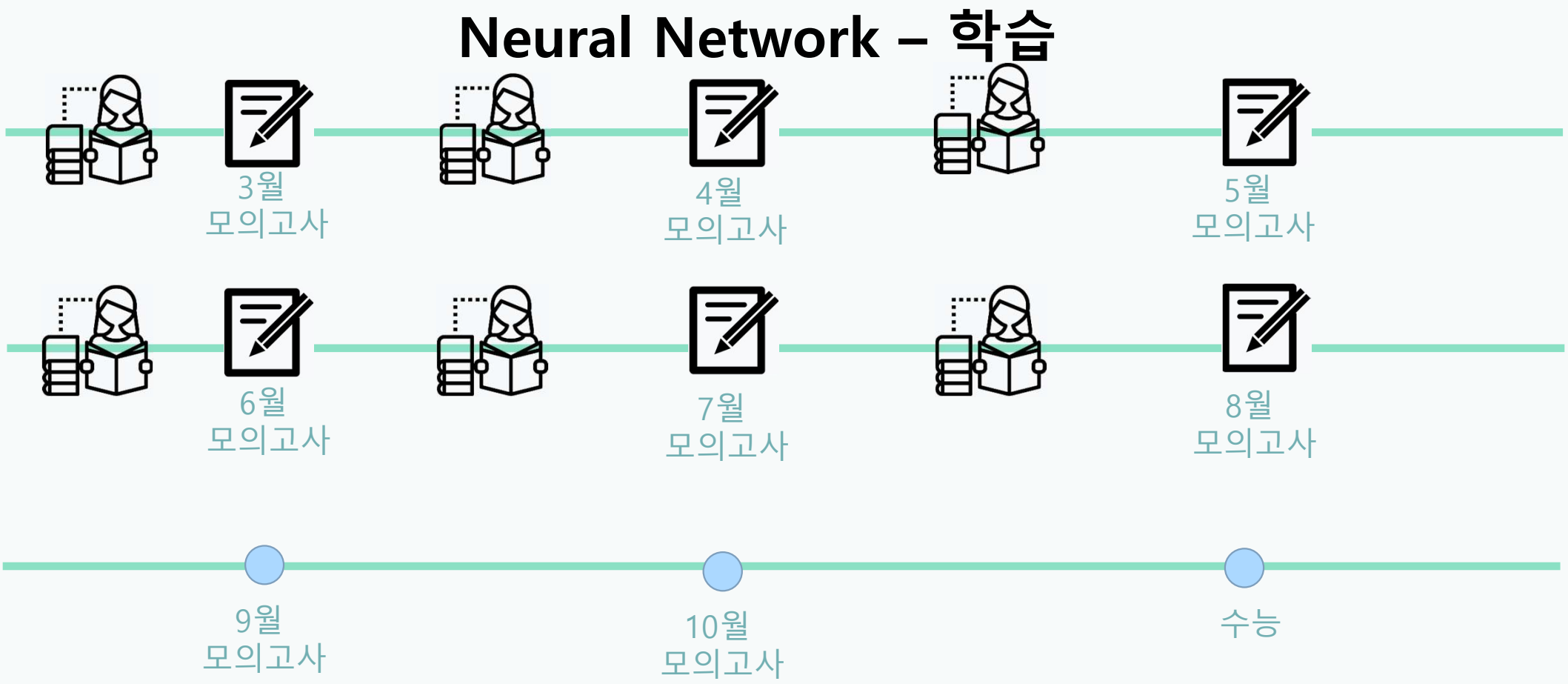
Neural Network – 학습



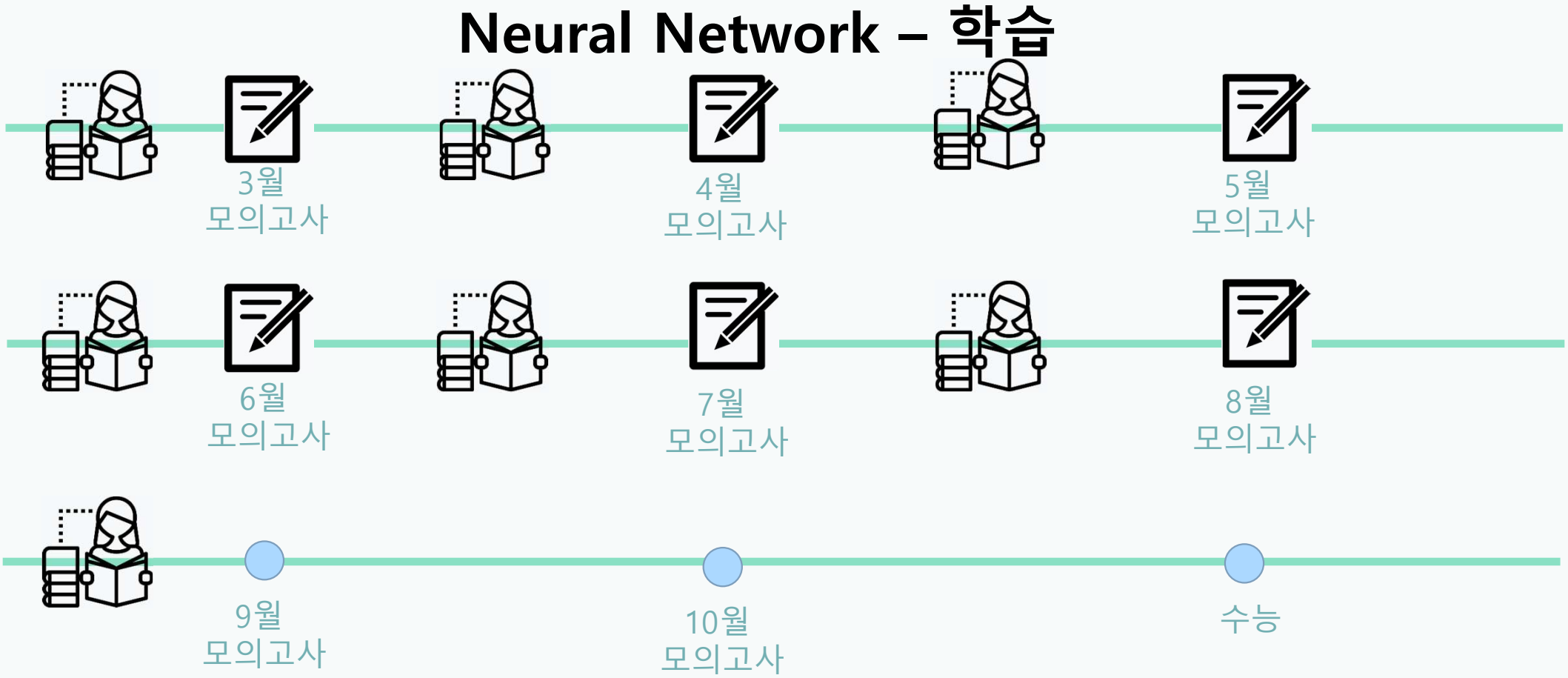
4. Neural Network 실습



4. Neural Network 실습

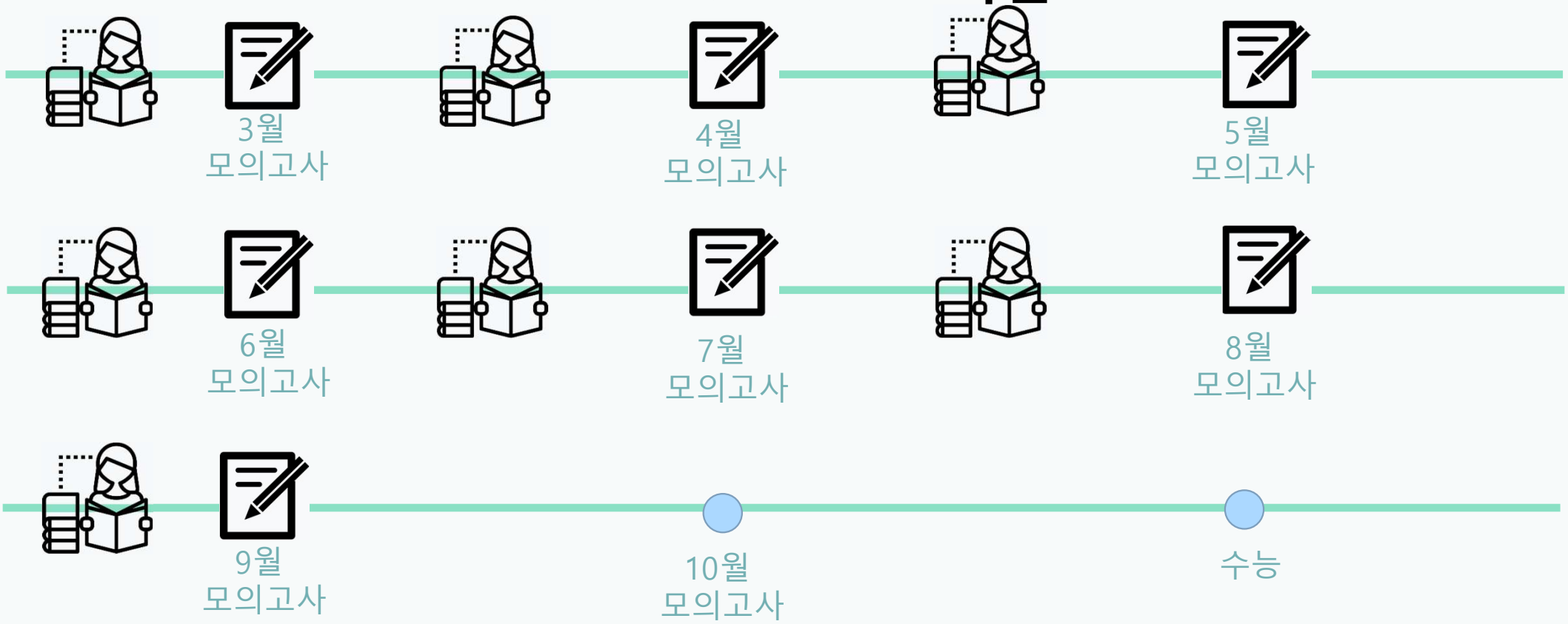


4. Neural Network 실습



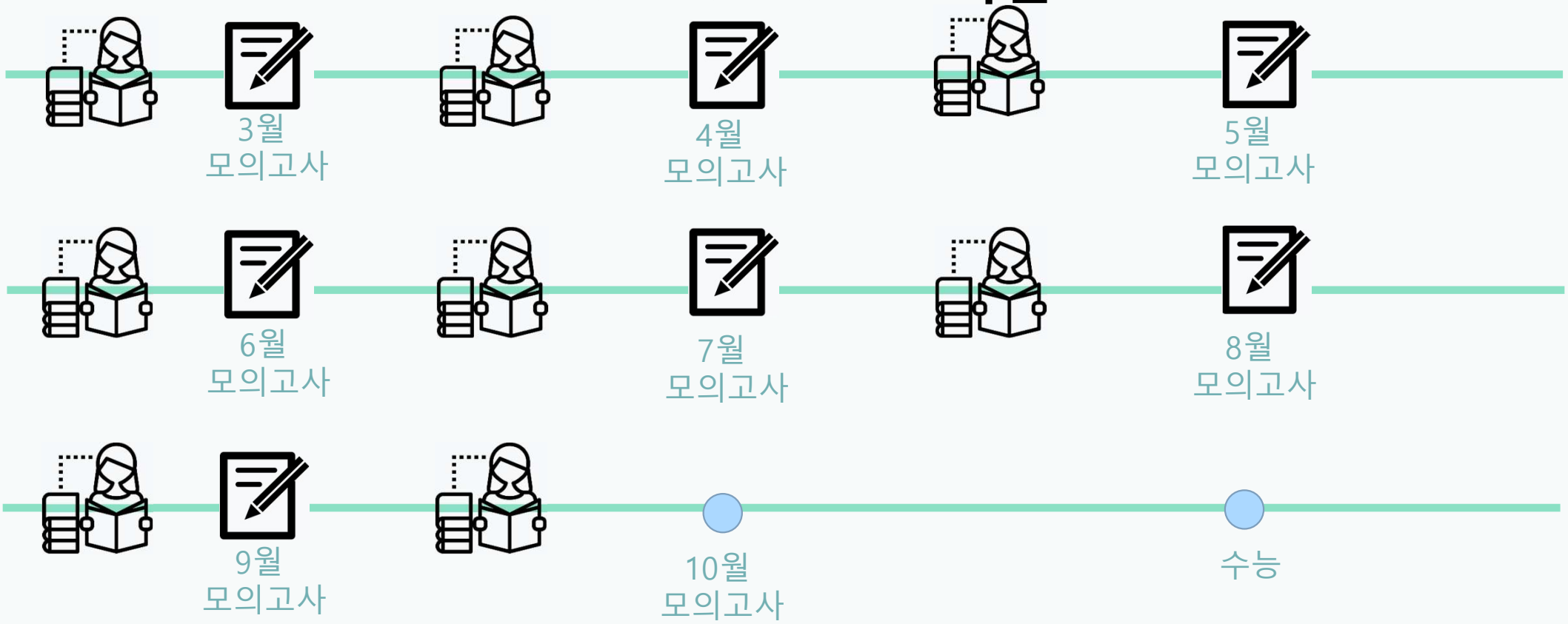
4. Neural Network 실습

Neural Network – 학습



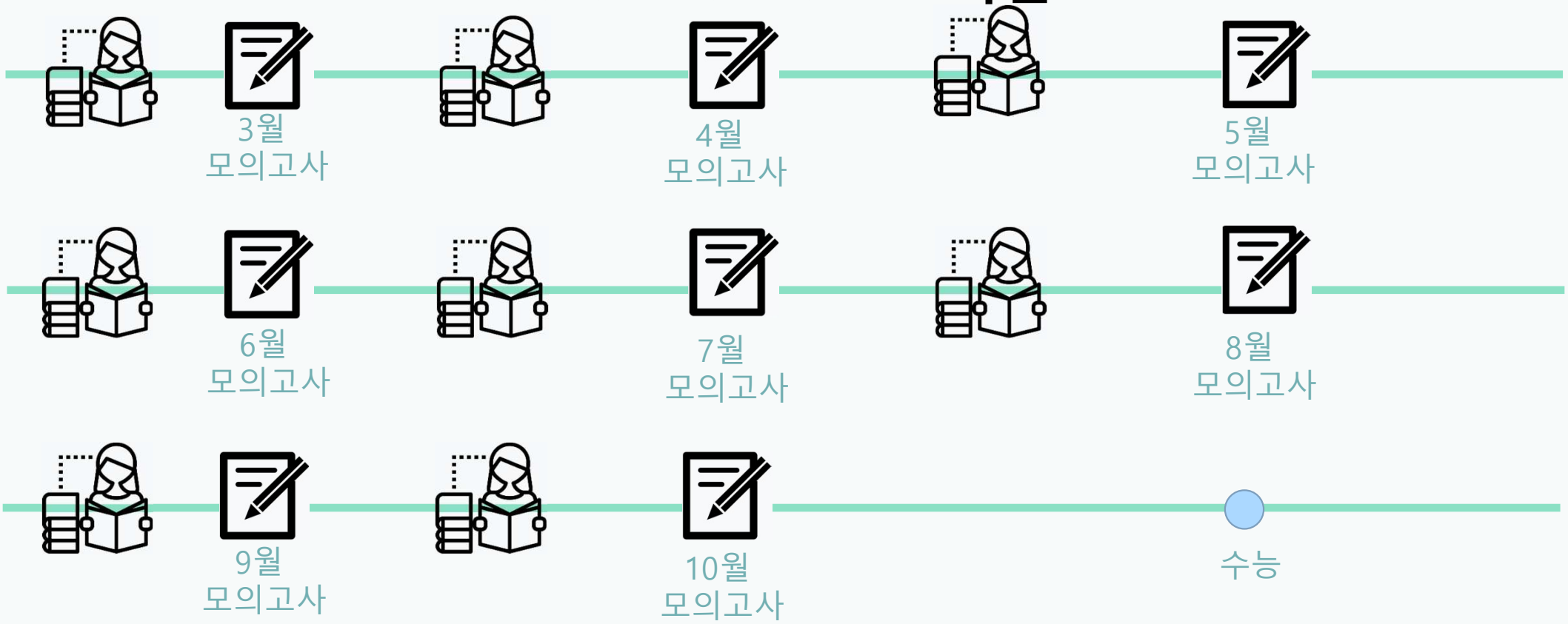
4. Neural Network 실습

Neural Network – 학습



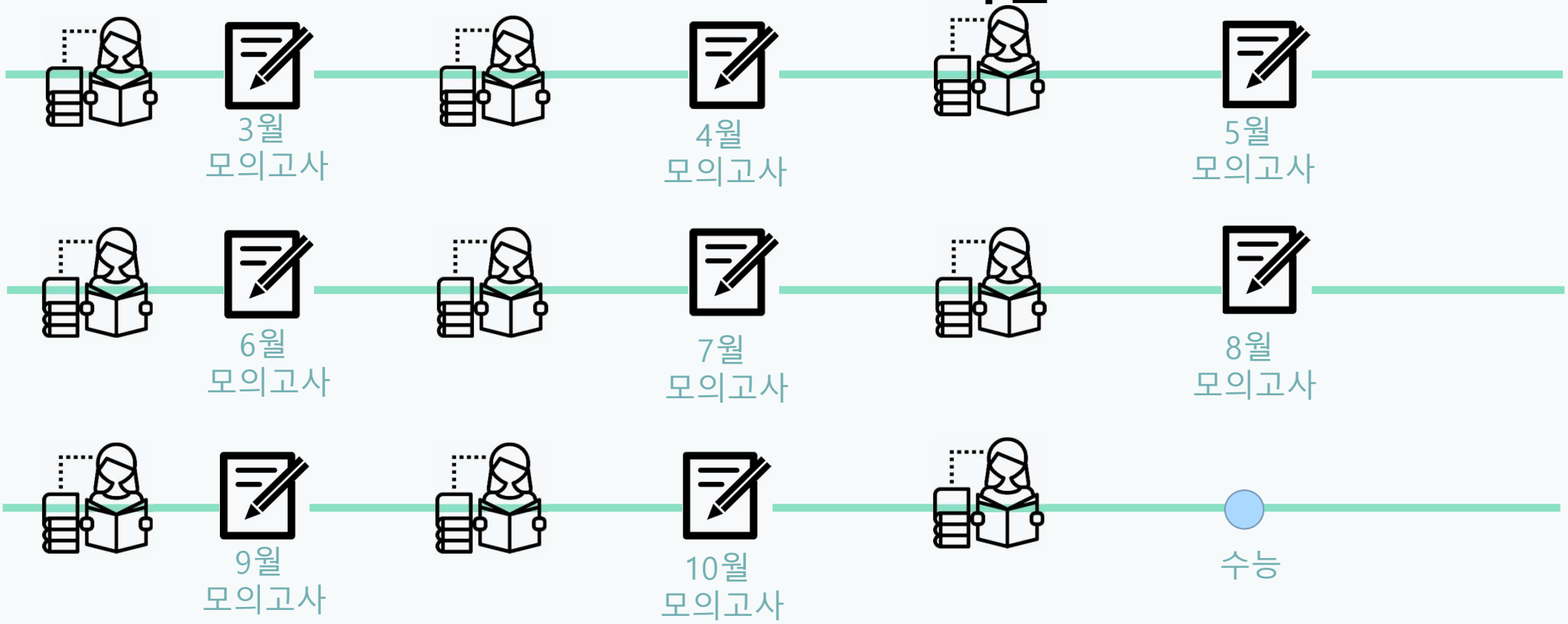
4. Neural Network 실습

Neural Network – 학습



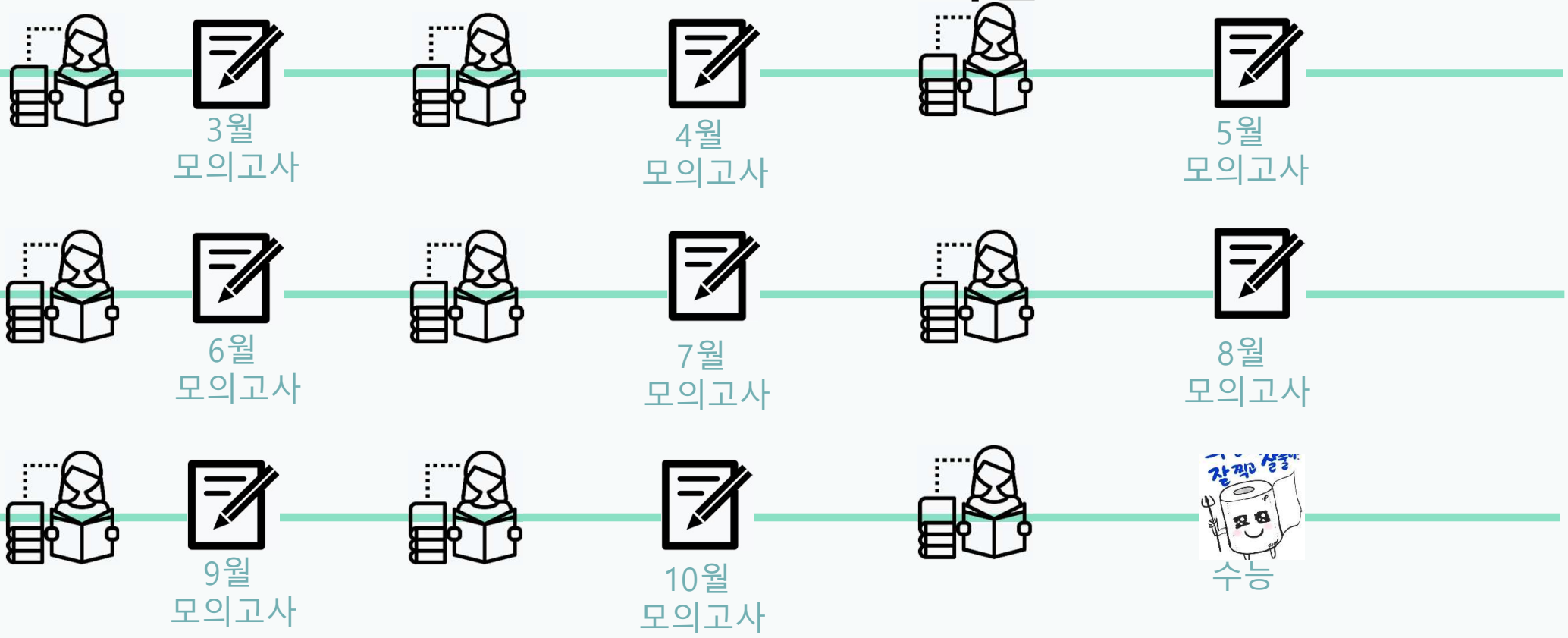
4. Neural Network 실습

Neural Network – 학습



4. Neural Network 실습

Neural Network – 학습



4. Neural Network 실습

DATA 나누기

학습
(train)

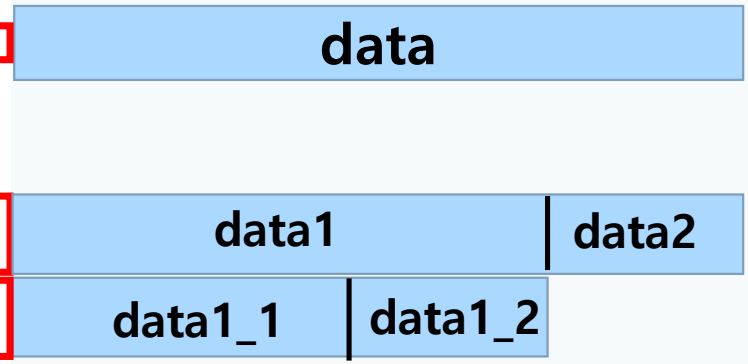
검증
(validation)

시험
(test)

4. Neural Network 실습

DATA 나누기

```
19  ## 데이터의 min , max, mean, std 값 구하기.
20  dataset_stats = dataset.describe()
21  dataset_stats.pop("D")
22  dataset_stats = dataset_stats.transpose()
23
24  train_labels = dataset.pop("D")
25
26  ## data normalization
27  def min_max_norm(x):
28    return (x - dataset_stats['min']) / (dataset_stats['max'] - dataset_stats['min'])
29
30  def standard_norm(x):
31    return (x - dataset_stats['mean']) / dataset_stats['std']
32
33  min_max_norm_train_data = min_max_norm(dataset)
34  standard_norm_train_data = standard_norm(dataset)
35
36  print("min max : ")
37  print(min_max_norm_train_data)
38  print("standard : ")
39  print(standard_norm_train_data)
40
41  X_train1, X_test, Y_train1, Y_test = \
42    train_test_split(standard_norm_train_data, train_labels, test_size=0.2,shuffle=False)
43
44  X_train, X_val, Y_train, Y_val = \
45    train_test_split(X_train1, Y_train1, test_size=0.2,shuffle=False)
46
47  print("x train")
48  print(X_train)
49  print("label train")
50  print(Y_train)
```



practice : P_02_05_data_division.py

Exersice : 02_05_data_division.py

4. Neural Network 실습

DATA 나누기 적용 코드 리뷰

```
## data copy
dataset=df.copy()

## label data 추출
ori_Y = dataset.pop("house price of unit area")

# train / val / test 데이터 분리
X_train1, X_test, Y_train1, Y_test = train_test_split(dataset, ori_Y, test_size=0.3, shuffle=False)
X_train, X_valid, Y_train, Y_valid = train_test_split(X_train1, Y_train1, test_size=0.2, shuffle=Fa
```

DATA를 TRAIN, VALIDATAION, TEST 로 나누기

```
# 모델의 설정
input_Layer = tf.keras.layers.Input(shape=(5,))
x = tf.keras.layers.Dense(50, activation='sigmoid')(input_Layer)
x= tf.keras.layers.Dense(100, activation='sigmoid')(x)
x= tf.keras.layers.Dense(300, activation='sigmoid')(x)
Out_Layer= tf.keras.layers.Dense(1, activation=None)(x)

model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()

loss=tf.keras.losses.mean_squared_error
optimizer=tf.keras.optimizers.SGD(lr=0.00004)
metrics = tf.keras.metrics.mean_squared_error ## regression이므로 평가 지표는 MSE가 된다.
model.compile(loss=loss,
              optimizer=optimizer,
              metrics=[metrics])
```

모델 만들기

```
result=model.fit(X_train, Y_train, epochs=2000, batch_size=100,validation_data=(X_valid,Y_valid))
```

학습하기

학습 데이터 넣기

검증 데이터 넣기

```
print("\n Test rmse: %.4f" % (model.evaluate(X_test, Y_test)[1]))
```

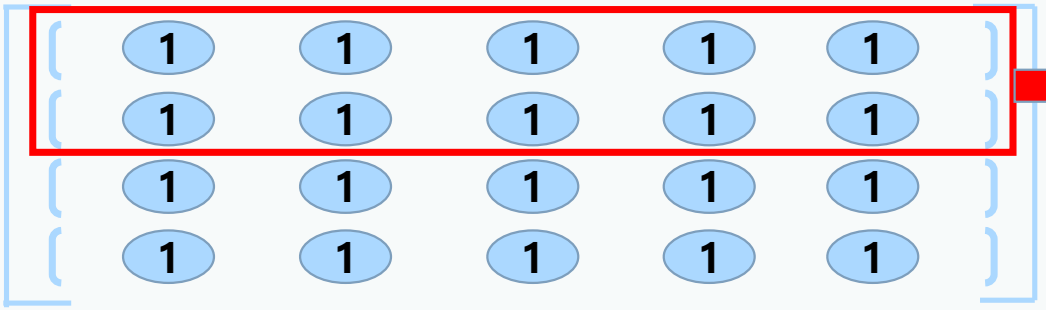
평가할 때 테스트 데이터

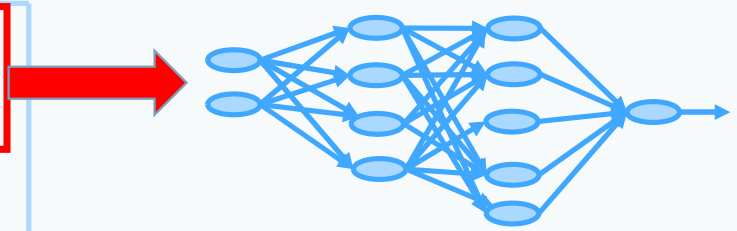
practice : P_02_06_house_price_of_area_prediction_data_div.py

4. Neural Network 실습

데이터 분리 학습 과정 살펴보기

TRAIN DATA (BATCH SIZE 2)





에러 계산해서
모델 업데이트

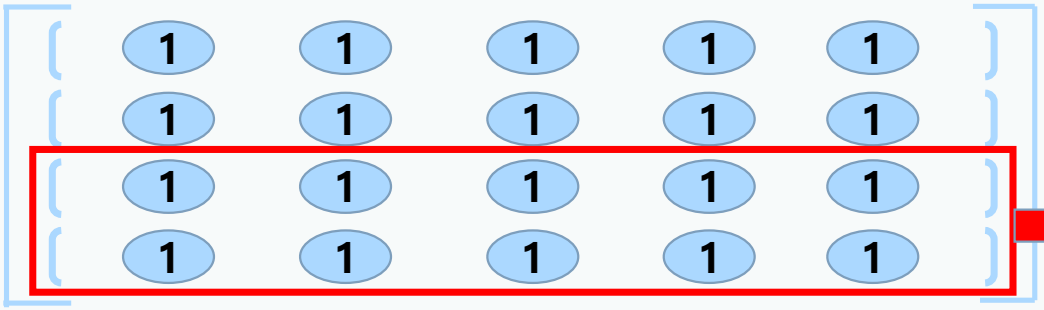
VALIDATION DATA(BATCH SIZE 2)

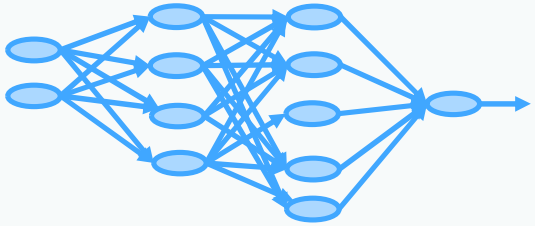
TEST DATA

4. Neural Network 실습

데이터 분리 학습 과정 살펴보기

TRAIN DATA (BATCH SIZE 2)





에러 계산해서
모델 업데이트

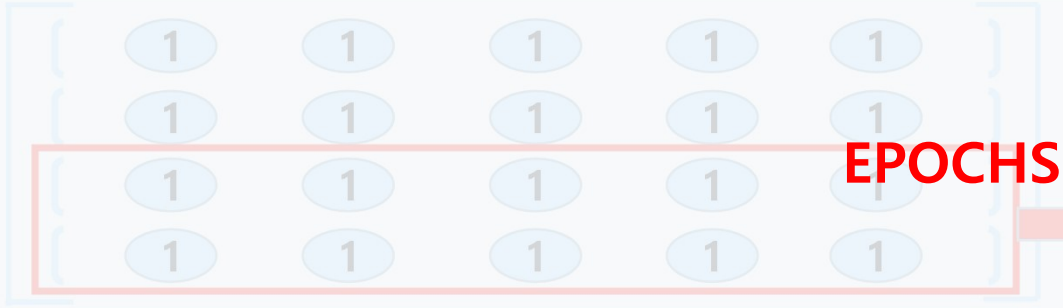
VALIDATION DATA(BATCH SIZE 2)

TEST DATA

4. Neural Network 실습

데이터 분리 학습 과정 살펴보기

TRAIN DATA (BATCH SIZE 2)



EPOCHS 1 완료

에러 계산해서
모델 업데이트

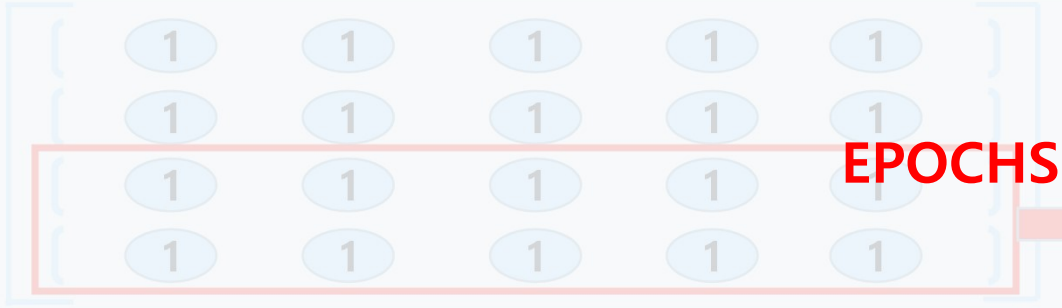
VALIDATION DATA(BATCH SIZE 2)

TEST DATA

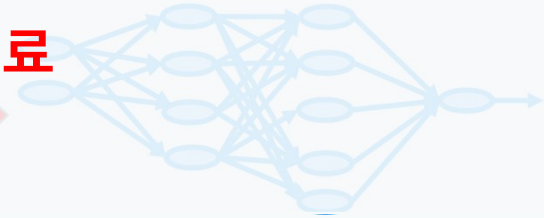
4. Neural Network 실습

데이터 분리 학습 과정 살펴보기

TRAIN DATA (BATCH SIZE 2)

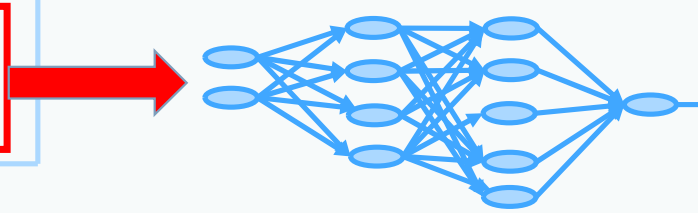


EPOCHS 1 완료



에러 계산해서
모델 업데이트

VALIDATION DATA(BATCH SIZE 2)



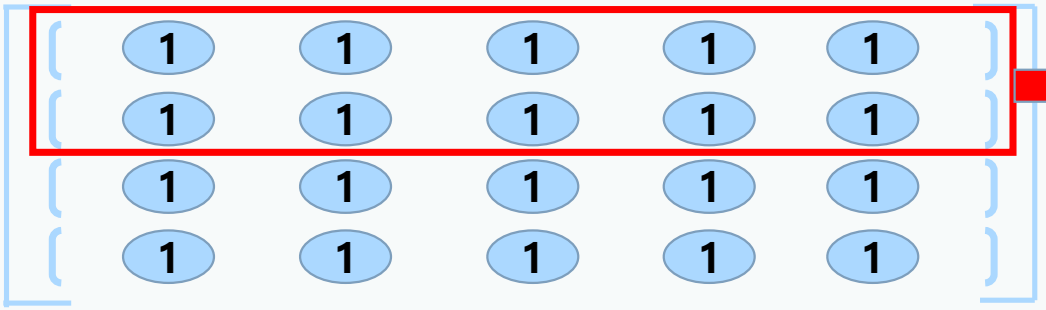
에러만 계산
모델 업데이트는 안함

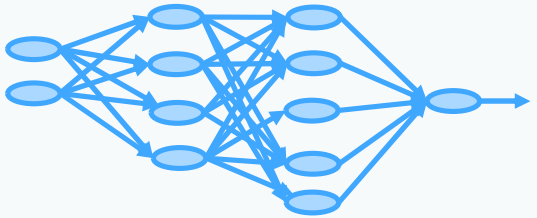
TEST DATA

4. Neural Network 실습

데이터 분리 학습 과정 살펴보기

TRAIN DATA (BATCH SIZE 2)





에러 계산해서
모델 업데이트

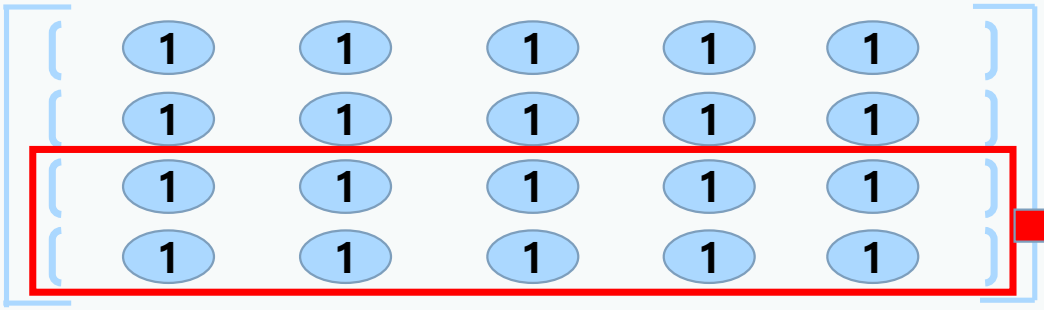
VALIDATION DATA(BATCH SIZE 2)

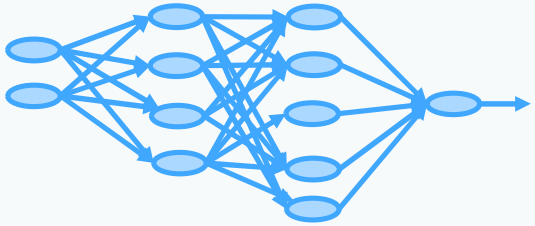
TEST DATA

4. Neural Network 실습

데이터 분리 학습 과정 살펴보기

TRAIN DATA (BATCH SIZE 2)





에러 계산해서
모델 업데이트

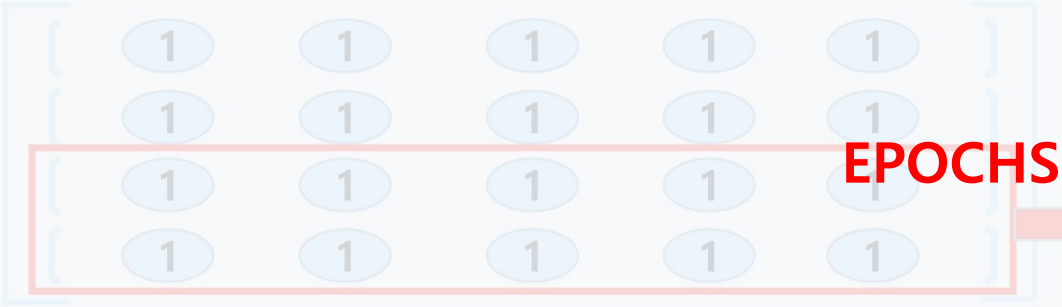
VALIDATION DATA(BATCH SIZE 2)

TEST DATA

4. Neural Network 실습

데이터 분리 학습 과정 살펴보기

TRAIN DATA (BATCH SIZE 2)



EPOCHS 2 완료

에러 계산해서
모델 업데이트

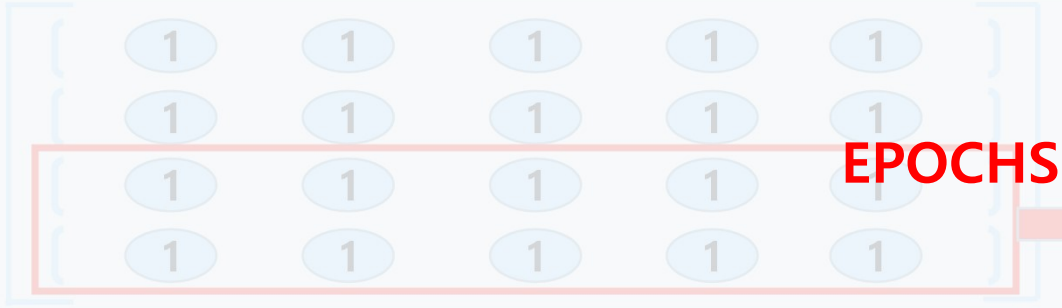
VALIDATION DATA(BATCH SIZE 2)

TEST DATA

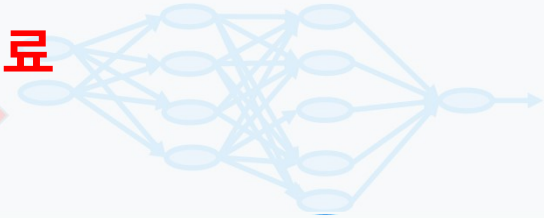
4. Neural Network 실습

데이터 분리 학습 과정 살펴보기

TRAIN DATA (BATCH SIZE 2)

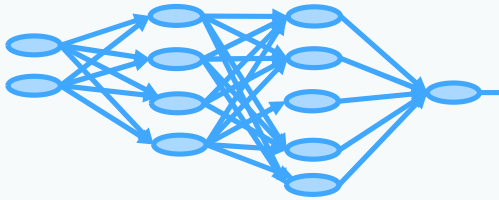


EPOCHS 2 완료



에러 계산해서
모델 업데이트

VALIDATION DATA(BATCH SIZE 2)

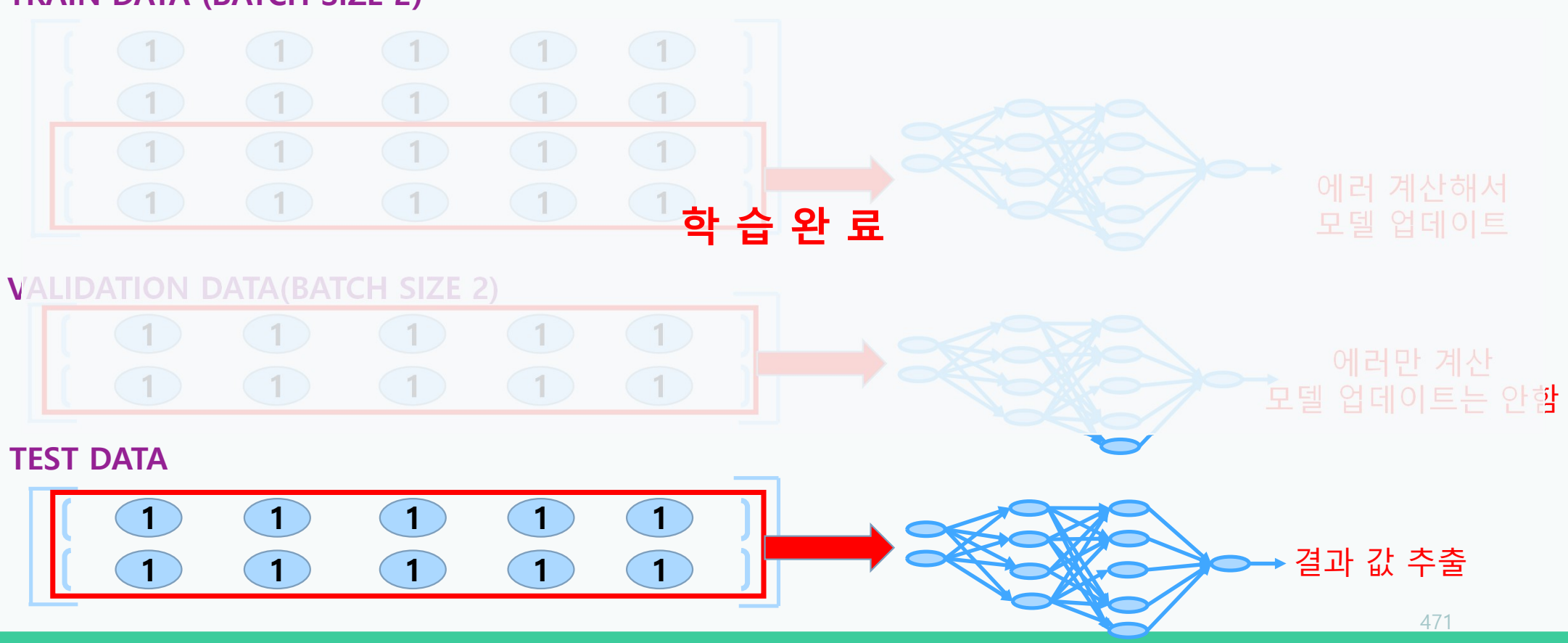


에러만 계산
모델 업데이트는 안함

TEST DATA

4. Neural Network 실습

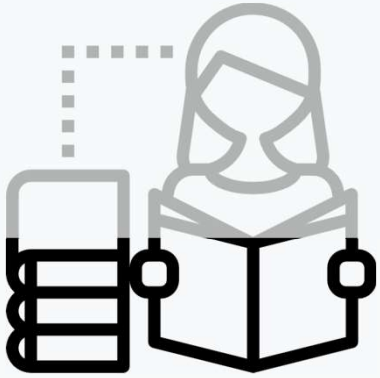
데이터 분리 학습 과정 살펴보기



4. Neural Network 실습

DATA 나누기 적용

보스턴 집값 예측 모델에 적용하기!

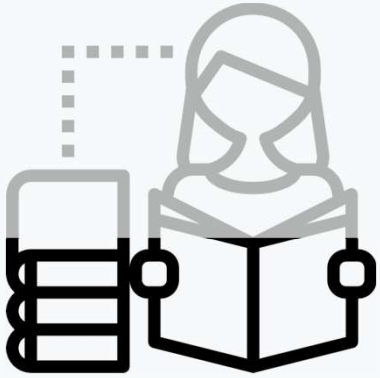


exersice : 02_06_house_prediction_data_div.ipynb

4. Neural Network 실습

DATA 나누기 적용+Normalization

지역 집 값 예측 모델에 적용하기!



practice : P_02_07_house_price_of_area_prediction_data_div_with_normal.py

4. Neural Network 실습

DATA 나누기 + Normalization 적용 코드 리뷰

```
## 데이터 읽어오기.## 데이터 읽어오기.
raw_df = pd.read_csv("../dataset/house_price_of_unit_area.csv")
print(raw_df.info())
print(raw_df.head())
dataset=raw_df.copy()

## data 분포도 확인
sns.pairplot(dataset[["house price",
plt.show()

## 데이터 열 별로 min, max, mean, std 구하기.
dataset_stats = dataset.describe()
dataset_stats.pop("house price of unit area")
dataset_stats = dataset_stats.transpose()
label_data=dataset.pop("house price of unit area")

## data normalization
def min_max_norm(x):
    return (x - dataset_stats['min']) / (dataset_stats['max'] - dataset_stats['min'])

def standard_norm(x):
    return (x - dataset_stats['mean']) / dataset_stats['std']

normed_train_data = standard_norm(dataset)

# 전체 데이터에서 학습 데이터와 테스트 데이터 분리
X_train1, X_test, Y_train1, Y_test = train_test_split(normed_train_data, label_data, test_size=0.1)
X_train, X_valid, Y_train, Y_valid = train_test_split(X_train1, Y_train1, test_size=0.1)

# 모델의 설정
input_Layer = tf.keras.layers.Input(shape=(5,))
x = tf.keras.layers.Dense(100, activation='sigmoid')(input_Layer)
x= tf.keras.layers.Dense(150, activation='sigmoid')(x)
x= tf.keras.layers.Dense(100, activation='sigmoid')(x)
Out_Layer= tf.keras.layers.Dense(1, activation=None)(x)

model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()
```

DATA NORMALIZATION을 위한 파라미터 구하기

DATA NORMALIZATION

DATA를 TRAIN, VALIDATAION, TEST 로 나누기

모델 만들기

```
loss=tf.keras.losses.mse
optimizer=tf.keras.optimizers.SGD(lr=0.002)
## regression이므로 평가 지표는 MSE가 된다.
model.compile(loss=loss,
              optimizer=optimizer,
              metrics=[tf.keras.metrics.mean_squared_error])

history=model.fit(X_train, Y_train, epochs=1000, batch_size=100, validation_data=(X_valid,Y_valid))
```

학습하기

학습 데이터 넣기

검증 데이터 넣기

평가할 때 테스트 데이터

```
print("\n Test rmse: %.4f" % (model.evaluate(normed_test_data, Y_test)[1]))
```

```
## 10샘플 데이터를 넣어서 예측값을 출력하여 정답값과 비교
for i in range(10):
    input_test_data=np.expand_dims(normed_test_data.values[i][:], axis=0)
    print("true : ", Y_test.values[i], "prediction: ", model.predict(input_test_data))
```

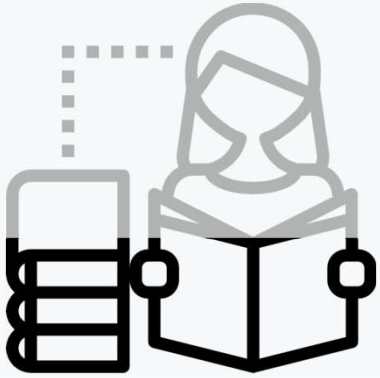
값 예측할 때 테스트 데이터

practice : P_02_07_house_price_of_area_prediction_data_div_with_normal.ipynb

4. Neural Network 실습

DATA 나누기 적용+Normalization

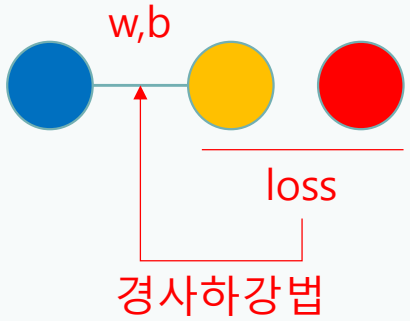
보스턴 집값 예측 모델에 적용하기!



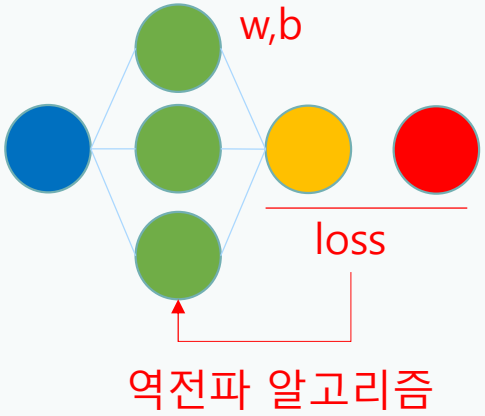
exersice : 02_07_house_prediction_data_div_with_normal.ipynb

5. Summary

perceptron



Neural Network



Loss

- MSE : Regression
- Cross Entropy : Logistic Regression(binary classification)

Data

- 현실과 유사한 형태, 분포도를 가진 데이터 구축이 필요
- 데이터 구축 시 학습/검증/테스트 데이터로 나누어 개발
- 데이터에서 normalization을 통해 성능 향상 할 수도 있음.

Neural Network

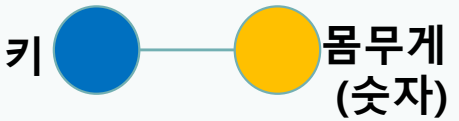
- Input Layer : Data input
- Hidden Layer : Data representation
- Output Layer : Output (즉, loss에 따라 task(출력)을 정함)

역전파 알고리즘

- 경사하강법 처럼 미분을 통해 delta_w, delta_b의 값을 구하는데 chain rule을 통해서 계산

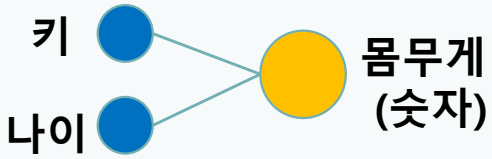
5. Summary

1. Linear Regression



$$\mathcal{L}_{mse} = \frac{1}{n}\sum_{i=1}^{n}(H(x_i) - y_i)^2$$

2. Multi-variable Linear Regression



$$\mathcal{L}_{mse} = \frac{1}{n}\sum_{i=1}^{n}(H(x_i) - y_i)^2$$

3. Logistic Regression(Classification)

내일 배울 내용입니다.

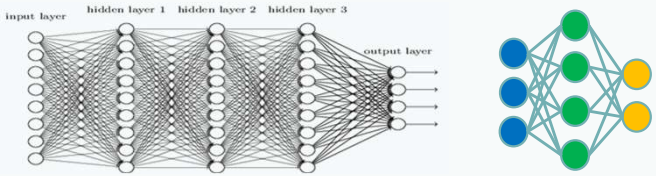
$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

4. Softmax Classifier

내일 배울 내용입니다.

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

5. Neural Network (Deep Learning)



오늘 하루 고생 하셨습니다.

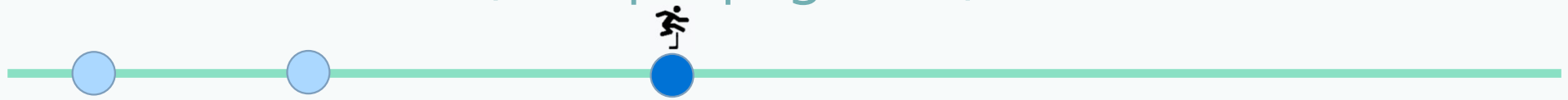
Q & A

참 고 자 료

- cs231n 강의
- 모두의 딥러닝 시즌 2
- 밑바닥 부터 시작하는 딥러닝 1, 2

Appendix. BACKPROPAGATION

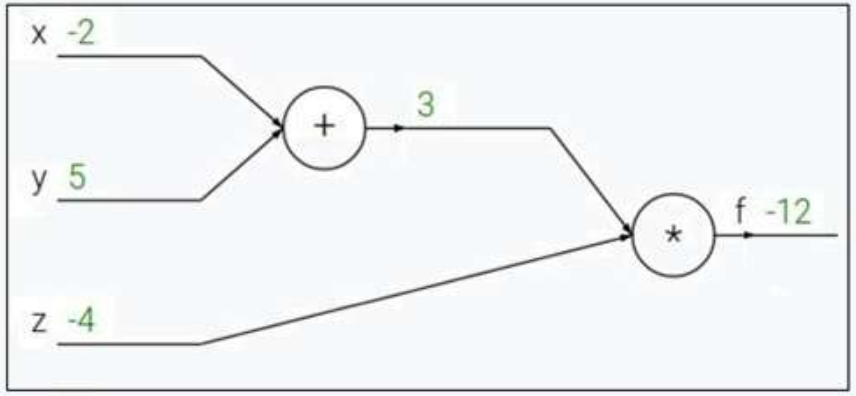
- 역전파 알고리즘(Backpropagation)



Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



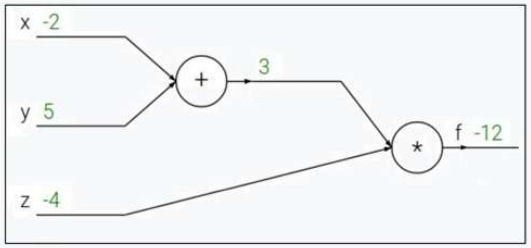
Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

Backpropagation: a simple example

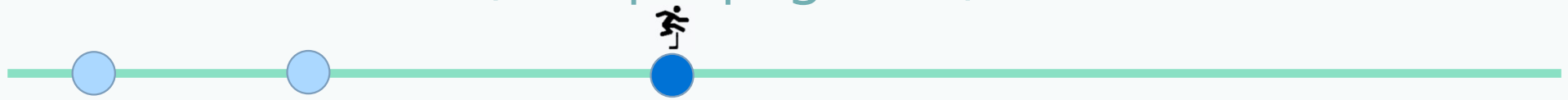
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)



Backpropagation: a simple example

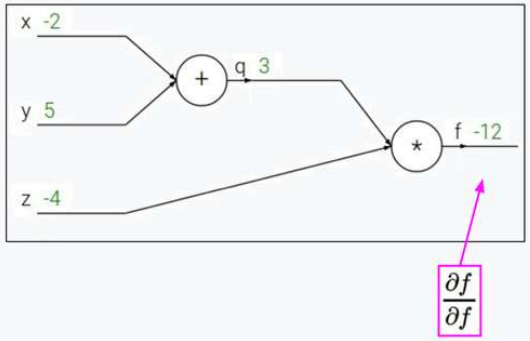
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

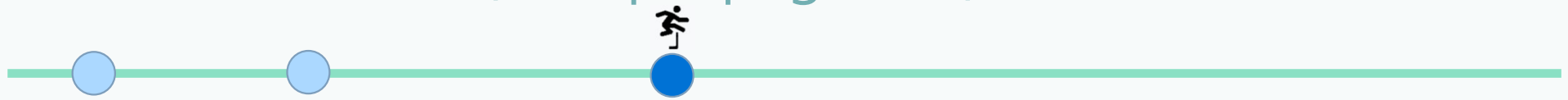
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)



Backpropagation: a simple example

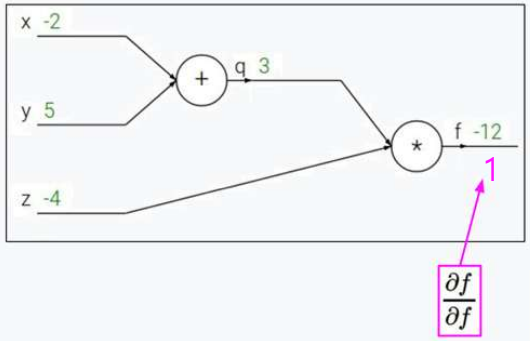
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

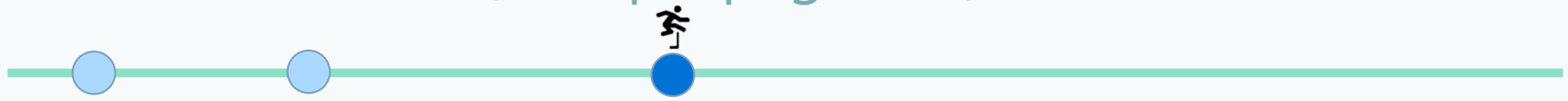
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)



Backpropagation: a simple example

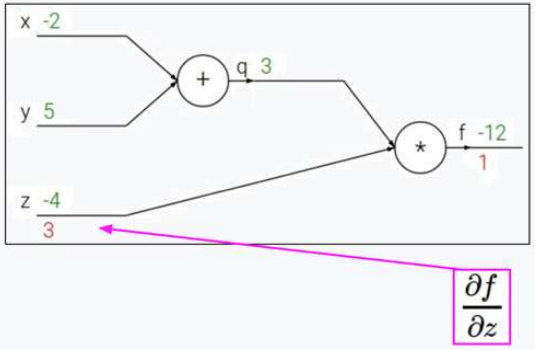
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Backpropagation: a simple example

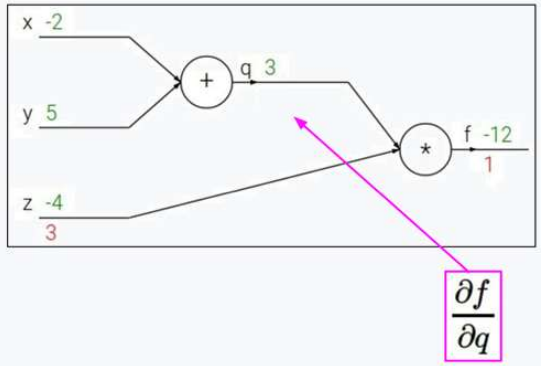
$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4

$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Backpropagation: a simple example

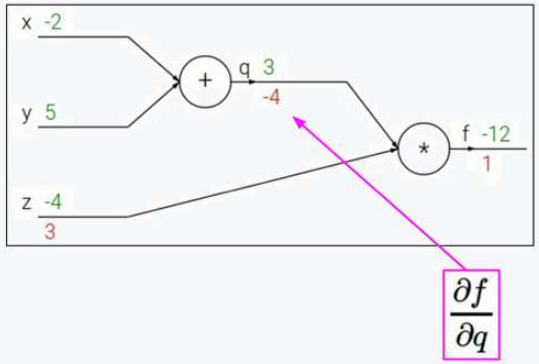
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

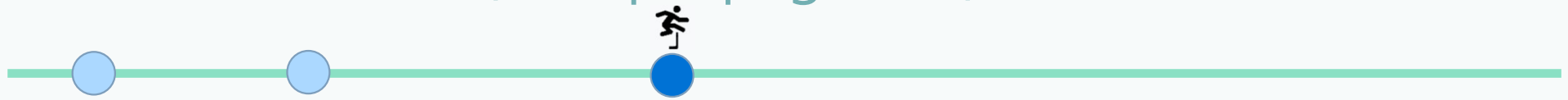
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)



Backpropagation: a simple example

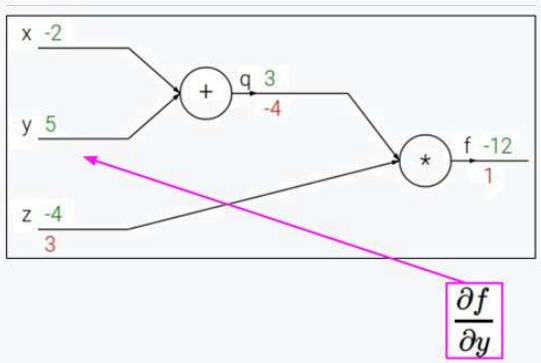
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Backpropagation: a simple example

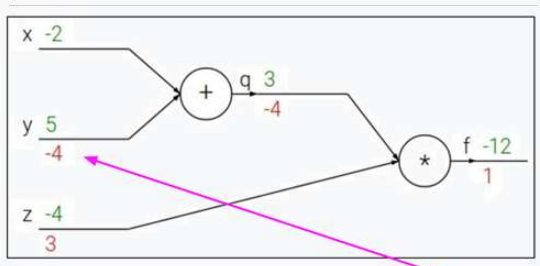
$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4

$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Backpropagation: a simple example

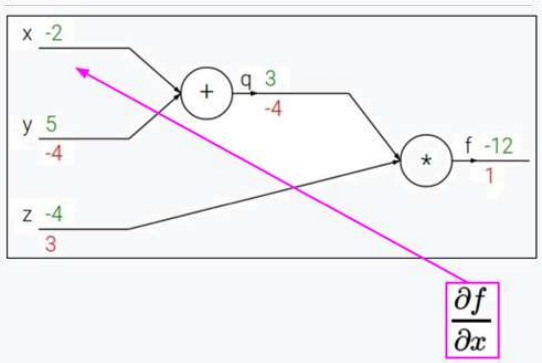
$f(x, y, z) = (x + y)z$

e.g. x = -2, y = 5, z = -4

$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$

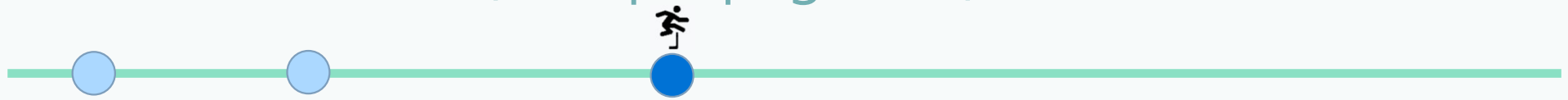
$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)



Backpropagation: a simple example

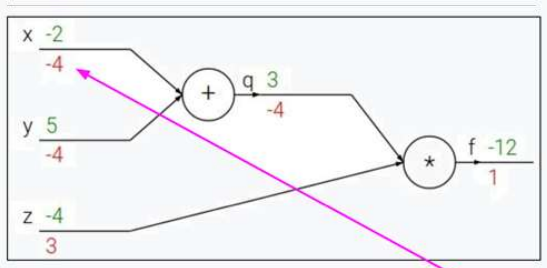
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

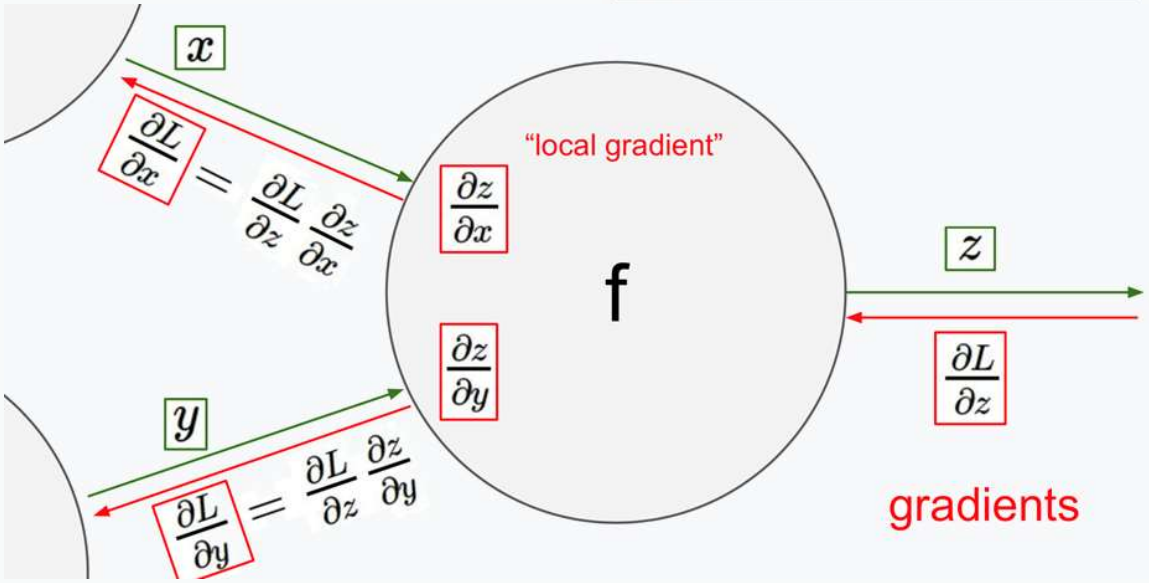
Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)



Appendix. BACKPROPAGATION

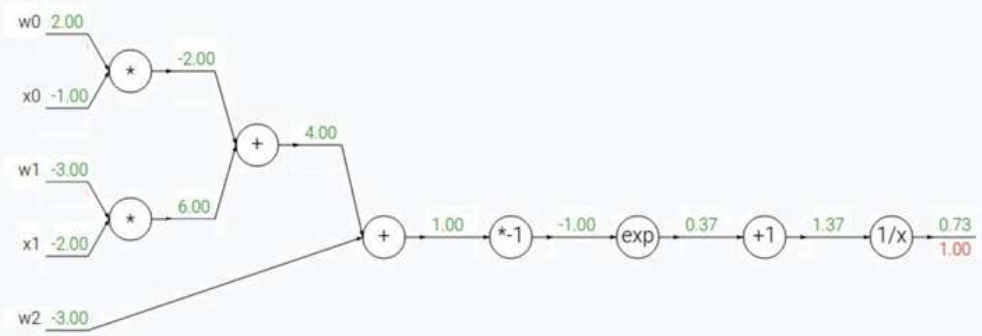
- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



Appendix. BACKPROPAGATION

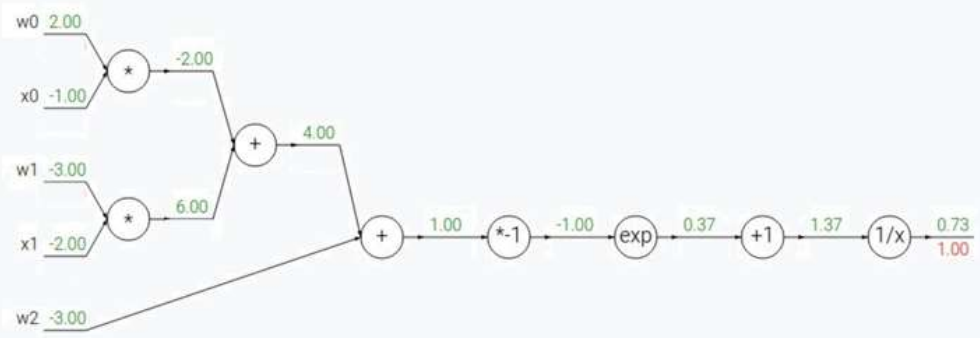
- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

Appendix. BACKPROPAGATION

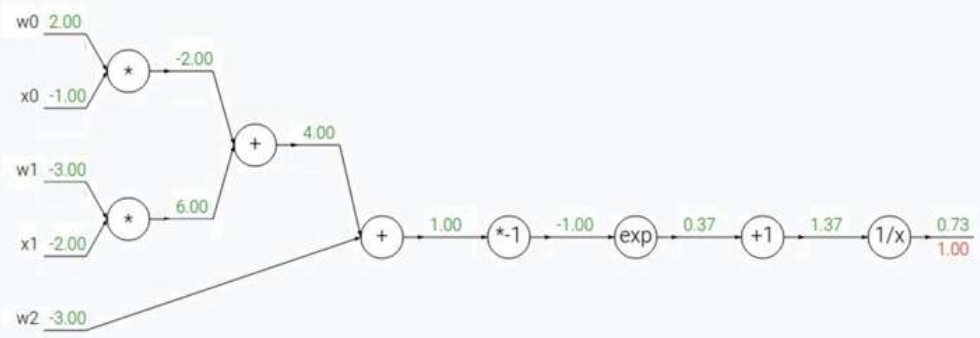
- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

Appendix. BACKPROPAGATION

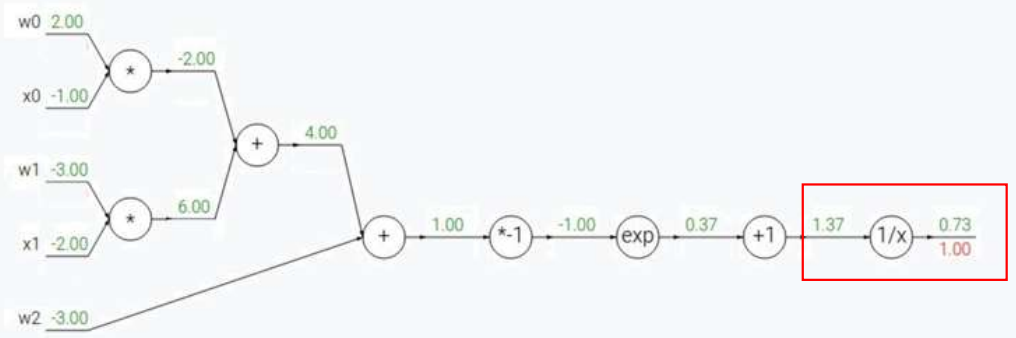
- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

Appendix. BACKPROPAGATION

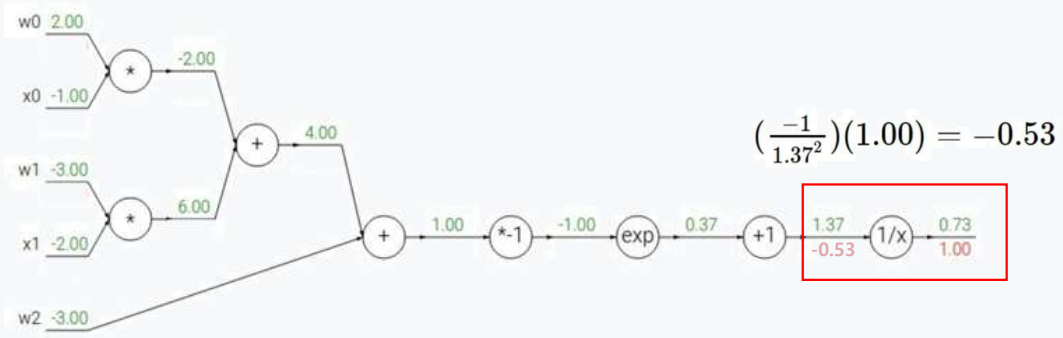
- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$(\frac{-1}{1.37^2})(1.00) = -0.53$

$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

Appendix. BACKPROPAGATION

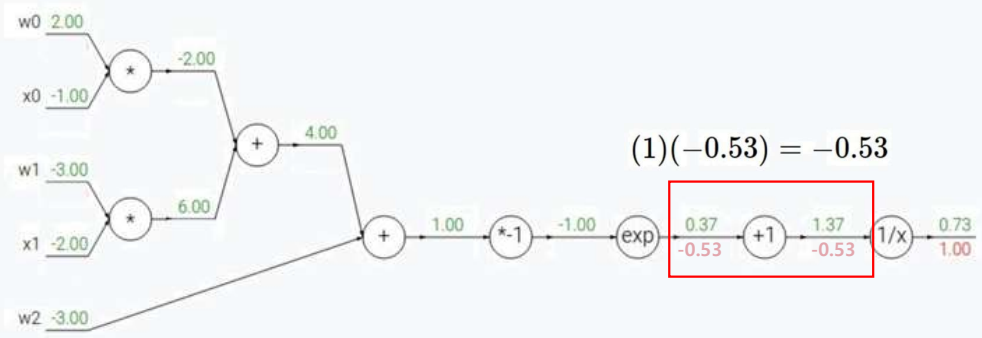
- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$(1)(-0.53) = -0.53$

$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

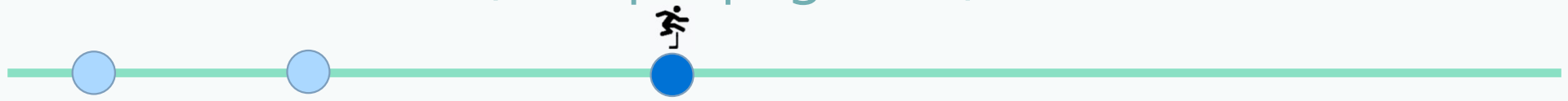
$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

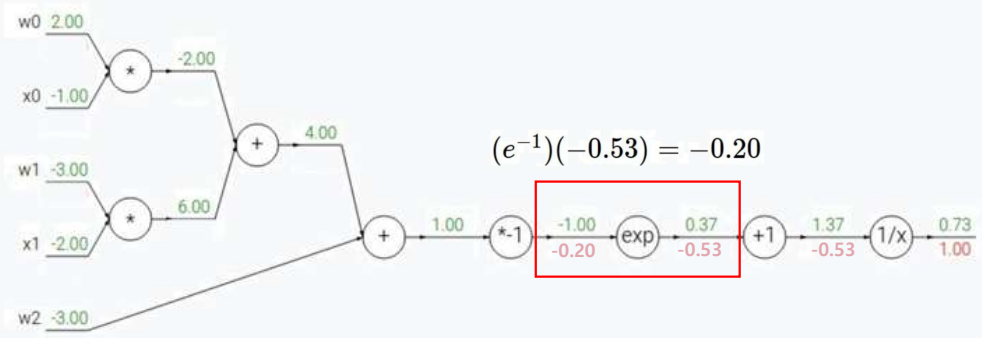
$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)



Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

Appendix. BACKPROPAGATION

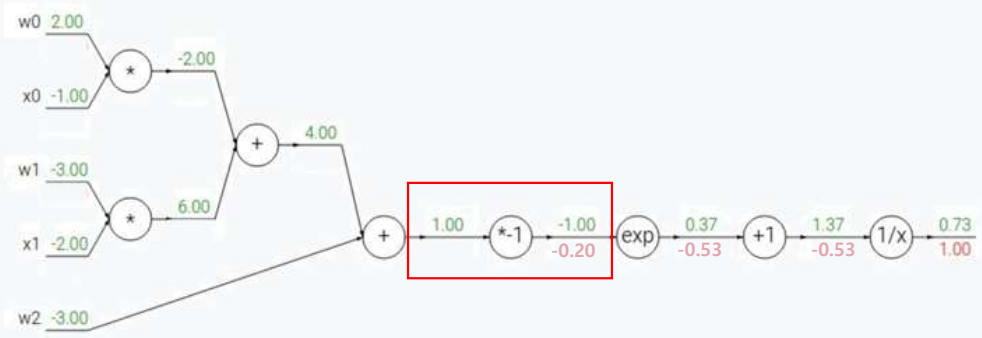
- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

Appendix. BACKPROPAGATION

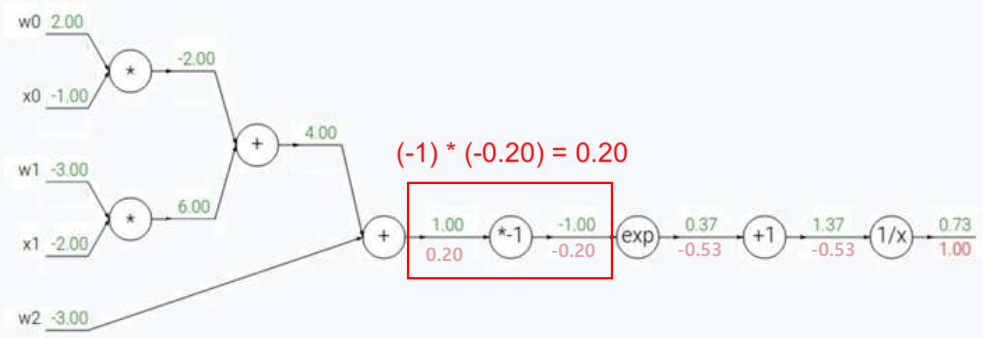
- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

Appendix. BACKPROPAGATION

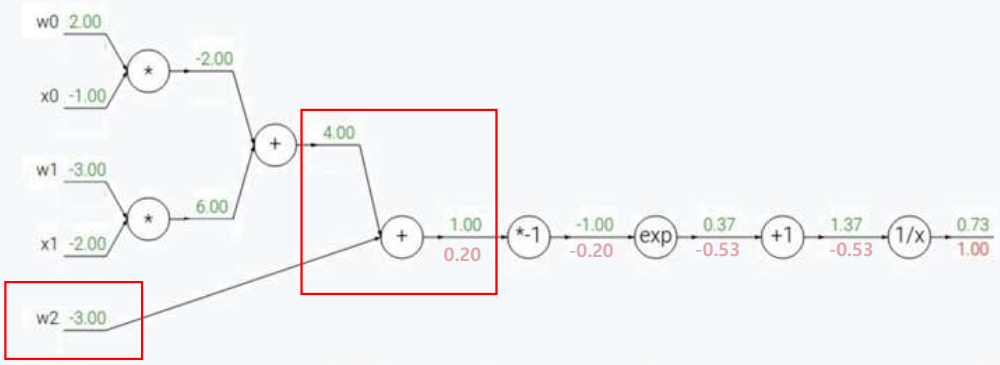
- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

Appendix. BACKPROPAGATION

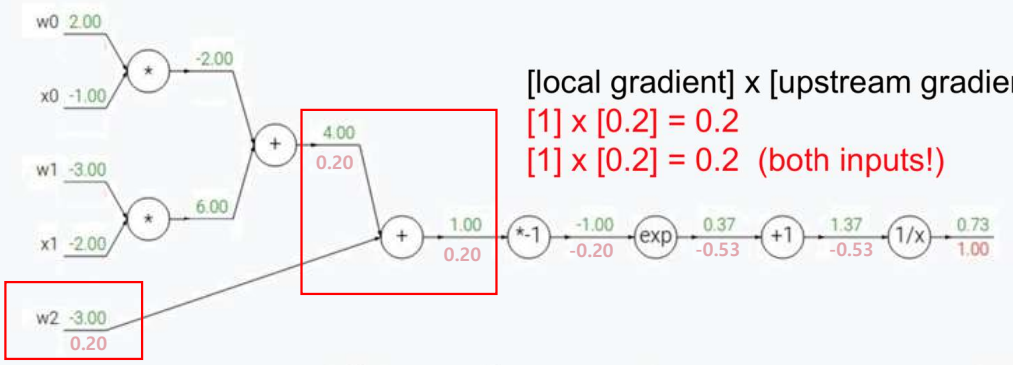
- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



[local gradient] x [upstream gradient]

[1] x [0.2] = 0.2

[1] x [0.2] = 0.2 (both inputs!)

$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

Appendix. BACKPROPAGATION

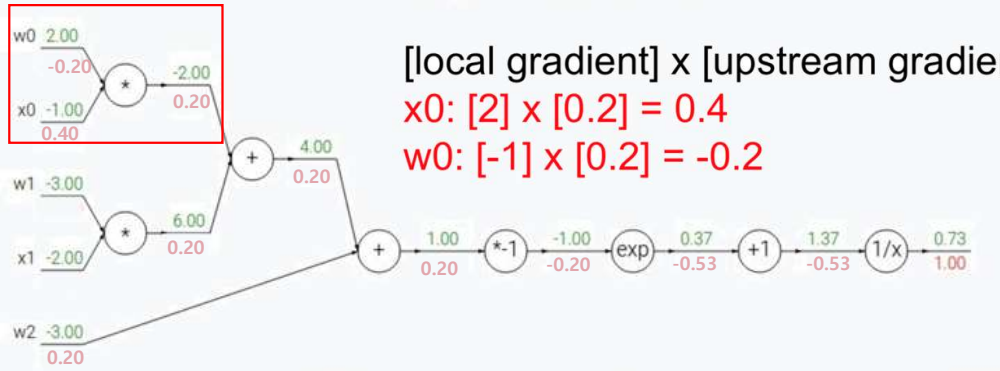
- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



[local gradient] x [upstream gradient]

x0: [2] x [0.2] = 0.4

w0: [-1] x [0.2] = -0.2

$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$

$f_a(x) = ax \rightarrow \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$

Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957 perceptron

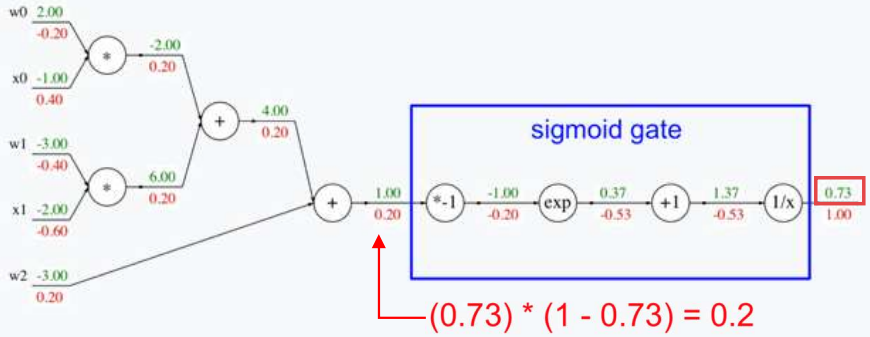
1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

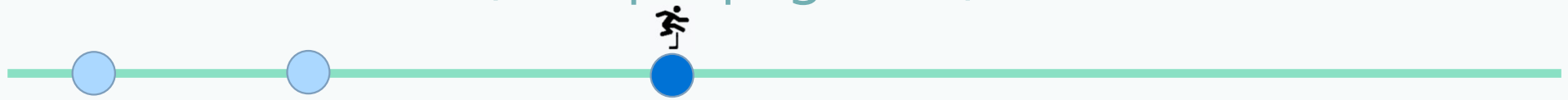
$$\sigma(x) = \frac{1}{1+e^{-x}}$$ sigmoid function

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\,\sigma(x)$$



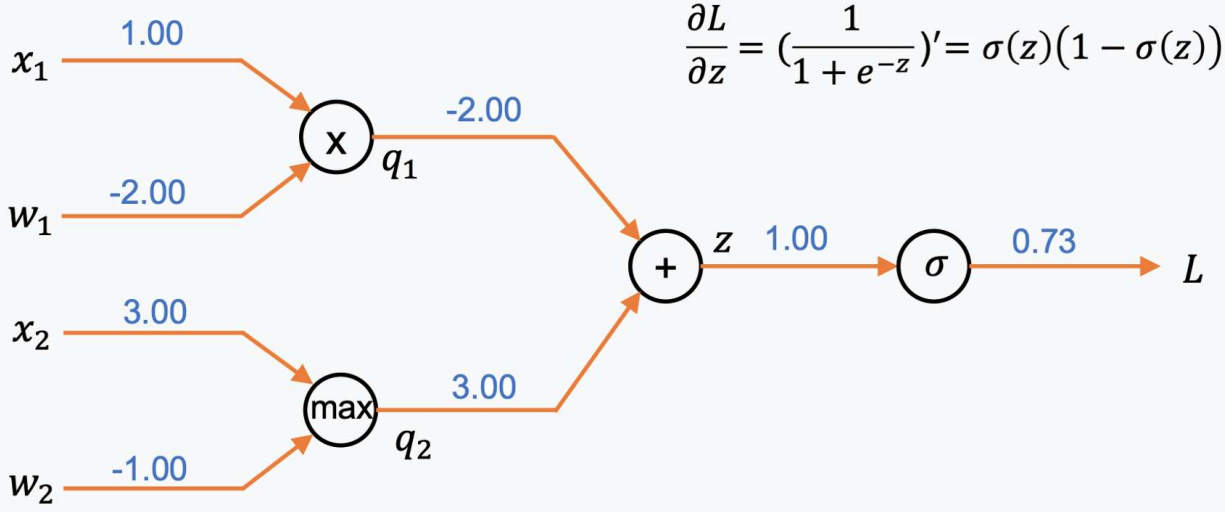
Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)



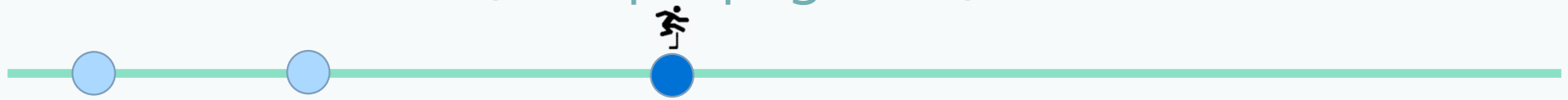
$$L = \sigma(z) = \frac{1}{1+e^{-z}}$$

$$\frac{\partial L}{\partial z} = \left(\frac{1}{1+e^{-z}}\right)' = \sigma(z)(1-\sigma(z))$$

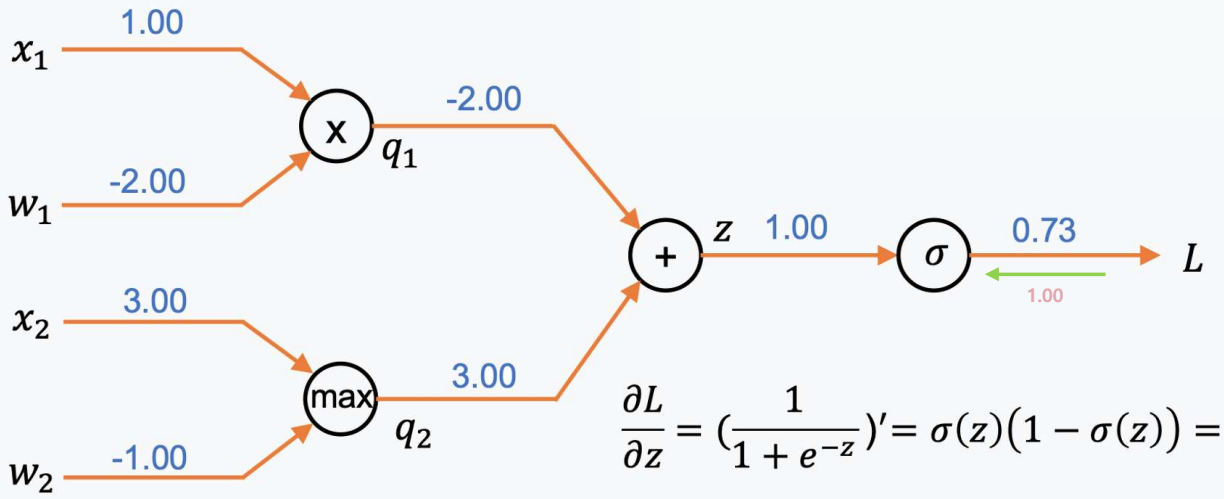


Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)



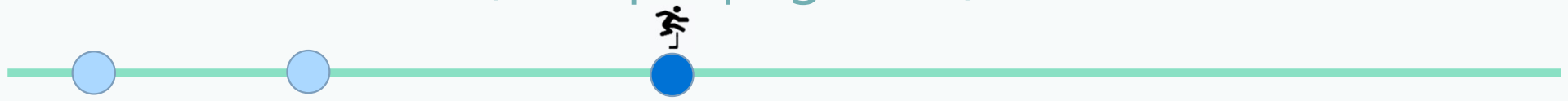
$$L = \sigma(z) = \frac{1}{1+e^{-z}}$$



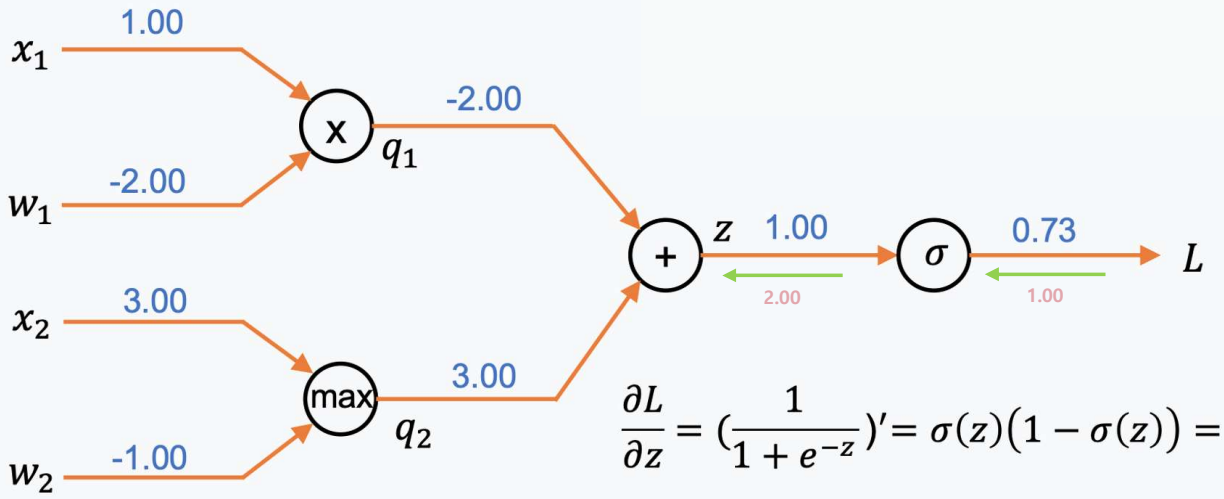
$$\frac{\partial L}{\partial z} = \left(\frac{1}{1+e^{-z}}\right)' = \sigma(z)\left(1-\sigma(z)\right) = 0.1971..$$

Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)



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



$$\frac{\partial L}{\partial z} = \left(\frac{1}{1+e^{-z}}\right)' = \sigma(z)\left(1-\sigma(z)\right) = 0.1971..$$

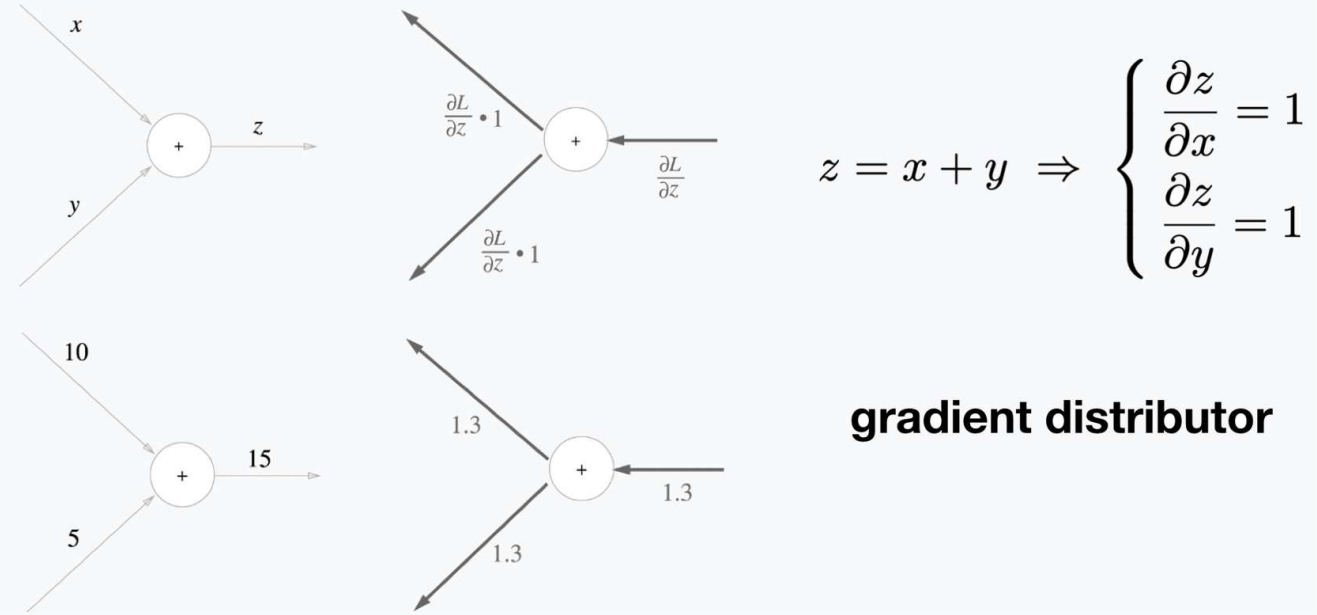
Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

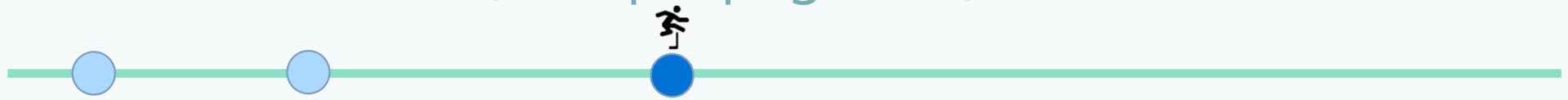


$$z = x + y \Rightarrow \begin{cases} \frac{\partial z}{\partial x} = 1 \\ \frac{\partial z}{\partial y} = 1 \end{cases}$$

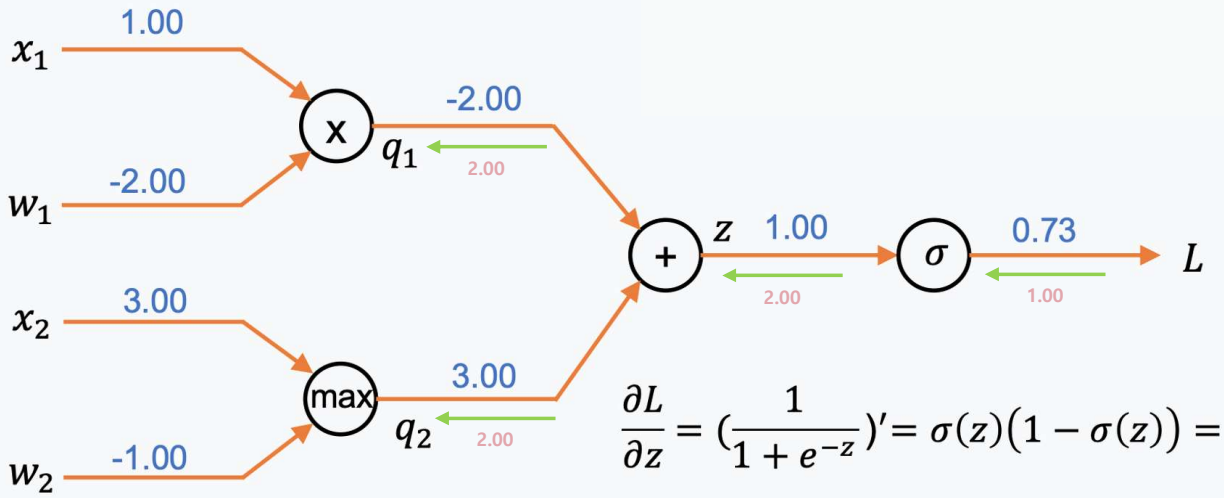
gradient distributor

Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)



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



$$\frac{\partial L}{\partial z} = \left(\frac{1}{1+e^{-z}}\right)' = \sigma(z)\left(1-\sigma(z)\right) = 0.1971..$$

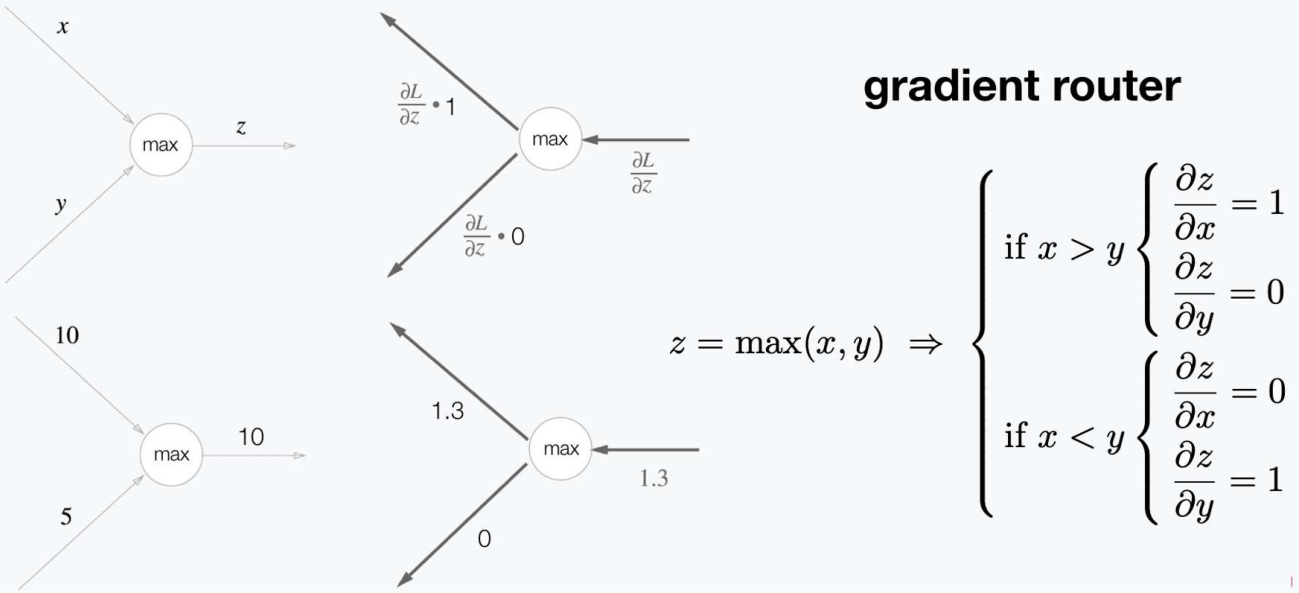
Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957 perceptron

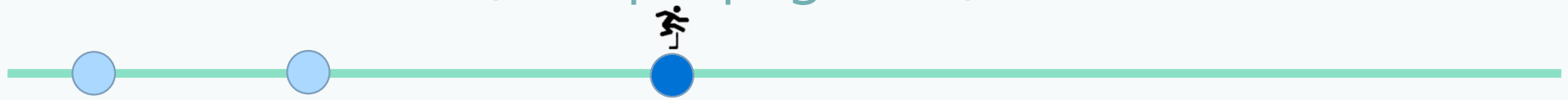
1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

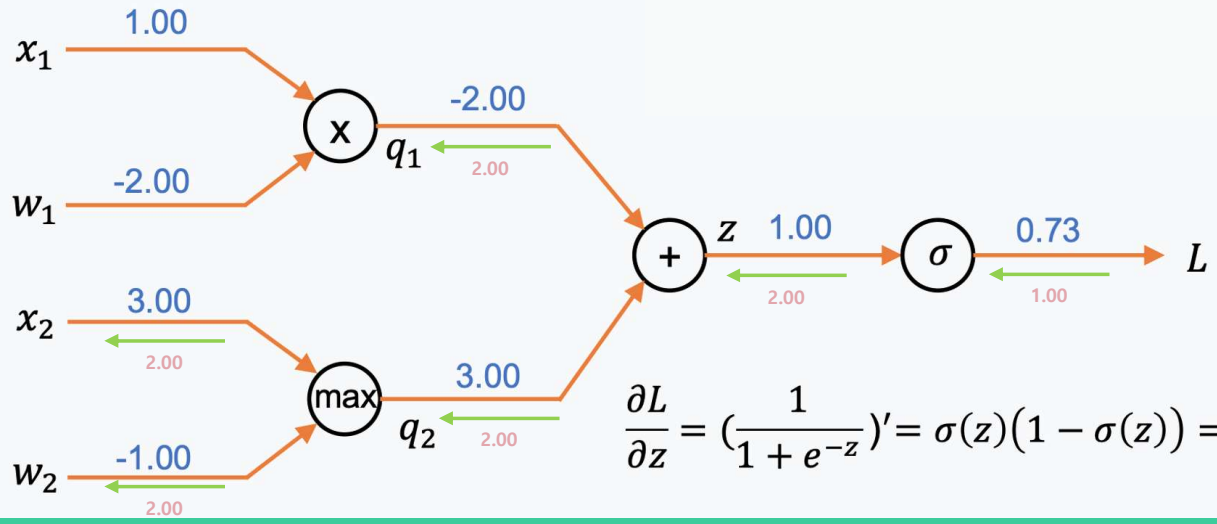


Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)



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



$$\frac{\partial L}{\partial z} = \left(\frac{1}{1+e^{-z}}\right)' = \sigma(z)\left(1-\sigma(z)\right) = 0.1971..$$

Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

$$\frac{\partial q_2}{\partial x_2} = \mathbf{1}(x_2 > w_2)$$

$$\frac{\partial L}{\partial x_2} = \frac{\partial L}{\partial q_2} \cdot \frac{\partial q_2}{\partial x_2} = 0.20 * 1.00 = 0.20$$

$$\frac{\partial q_2}{\partial w_2} = \mathbf{1}(w_2 > x_2)$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial q_2} \cdot \frac{\partial q_2}{\partial w_2} = 0.20 * 0.00 = 0.00$$

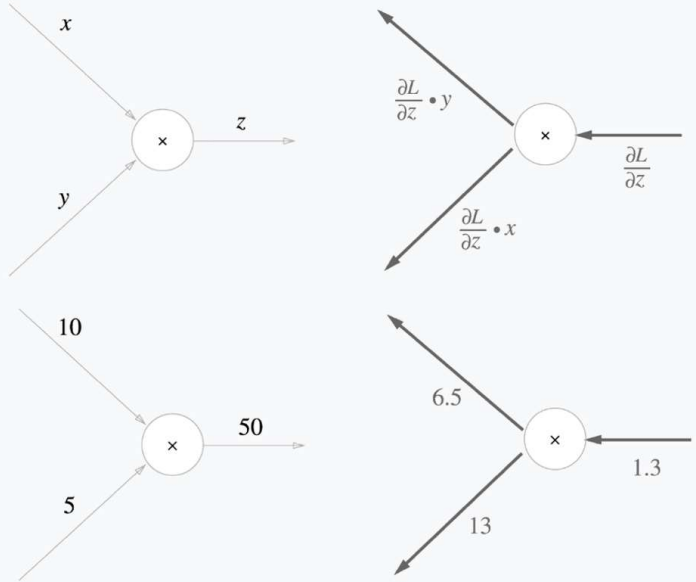
Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)

1957 perceptron

1969 XOR problem

1986 Multi layer Perceptron (backpropagation)

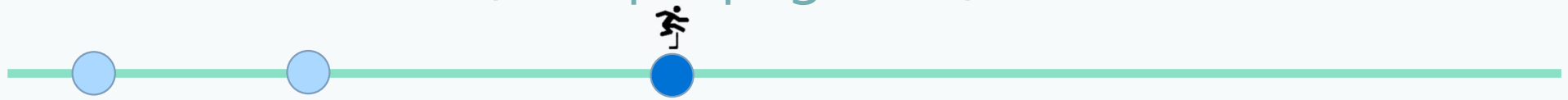


$$z = xy \Rightarrow \begin{cases} \frac{\partial z}{\partial x} = y \\ \frac{\partial z}{\partial y} = x \end{cases}$$

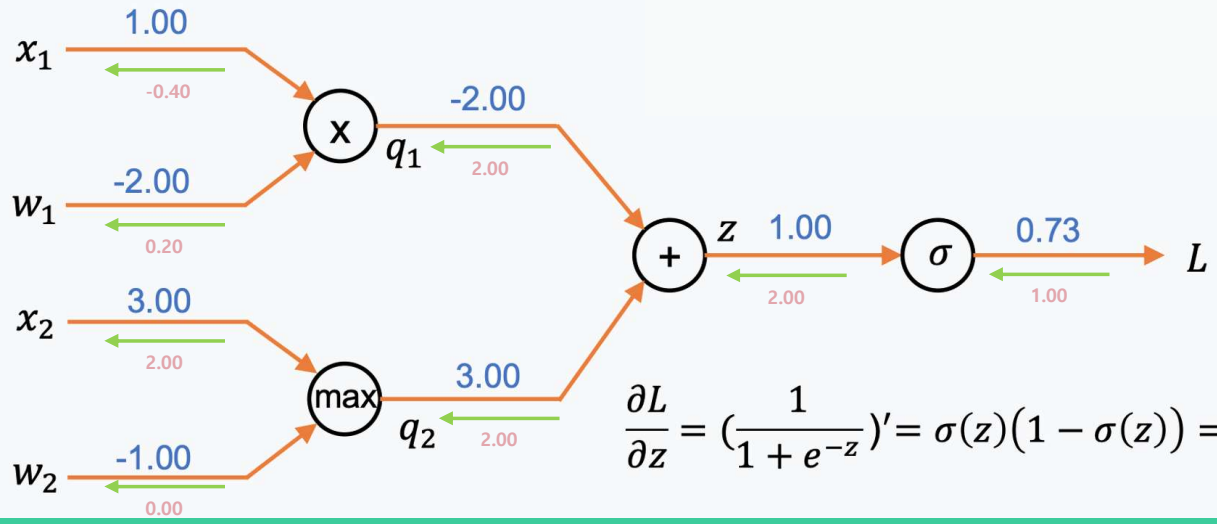
gradient switcher

Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)



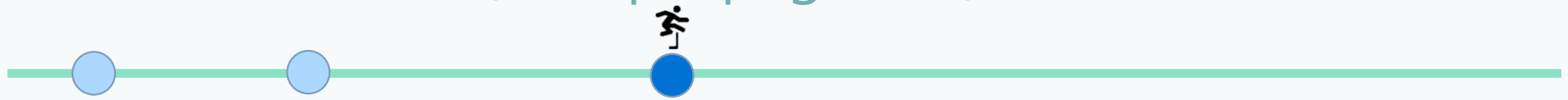
$$L = \sigma(z) = \frac{1}{1+e^{-z}}$$

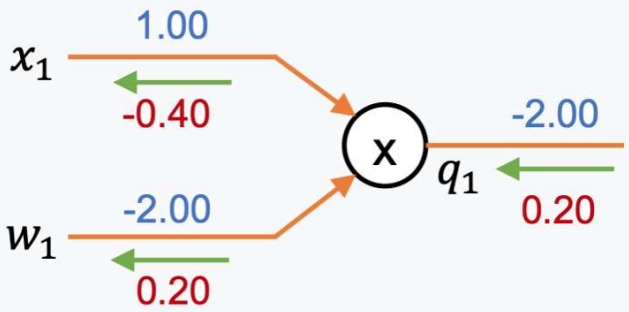


$$\frac{\partial L}{\partial z} = \left(\frac{1}{1+e^{-z}}\right)' = \sigma(z)\left(1-\sigma(z)\right) = 0.1971..$$

Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)





$$\frac{\partial q_1}{\partial x_1} = w_1$$

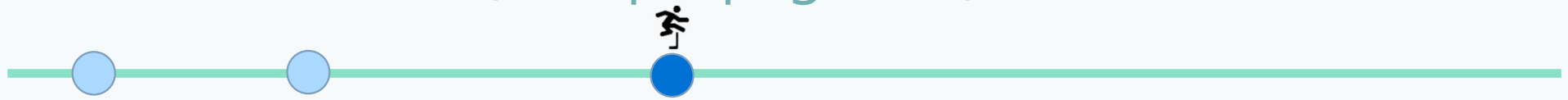
$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial q_1} \cdot \frac{\partial q_1}{\partial x_1} = 0.20 * (-2.00) = -0.40$$

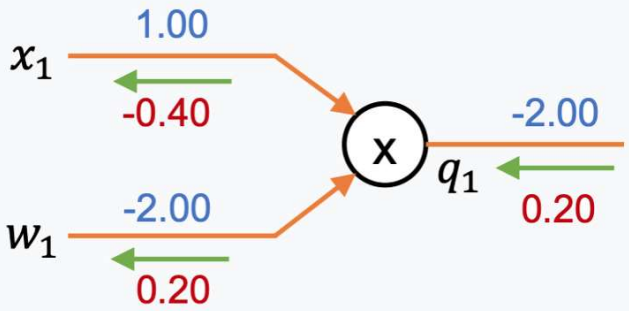
$$\frac{\partial q_1}{\partial w_1} = x_1$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial q_1} \cdot \frac{\partial q_1}{\partial w_1} = 0.20 * 1.00 = 0.20$$

Appendix. BACKPROPAGATION

- 역전파 알고리즘(Backpropagation)





$$\frac{\partial q_1}{\partial x_1} = w_1$$

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial q_1} \cdot \frac{\partial q_1}{\partial x_1} = 0.20 * (-2.00) = -0.40$$

$$\frac{\partial q_1}{\partial w_1} = x_1$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial q_1} \cdot \frac{\partial q_1}{\partial w_1} = 0.20 * 1.00 = 0.20$$