

데이터 과학 기초

06

인공신경망

경북대학교 배준현 교수
(joonion@knu.ac.kr)



06. 인공지능경망

- 인공지능경망: ANN, *Artificial Neural Network*
 - 사람의 뇌가 동작하는 방식을 그대로 흉내 내어 만든 수학적 모델
 - 뉴런과 시냅스: *neuron* and *synapse*
 - 사람의 뇌는 뉴런(신경세포)들이 서로 연결되어 다른 뉴런들과 상호작용
 - 입력으로 받은 전기 신호를 적당히 처리하여 다른 뉴런에 전달
 - 신호를 전달하려면 입력으로 받은 전기 신호의 합이 일정 수준을 넘어야 함



■ 생물학적 뉴런:

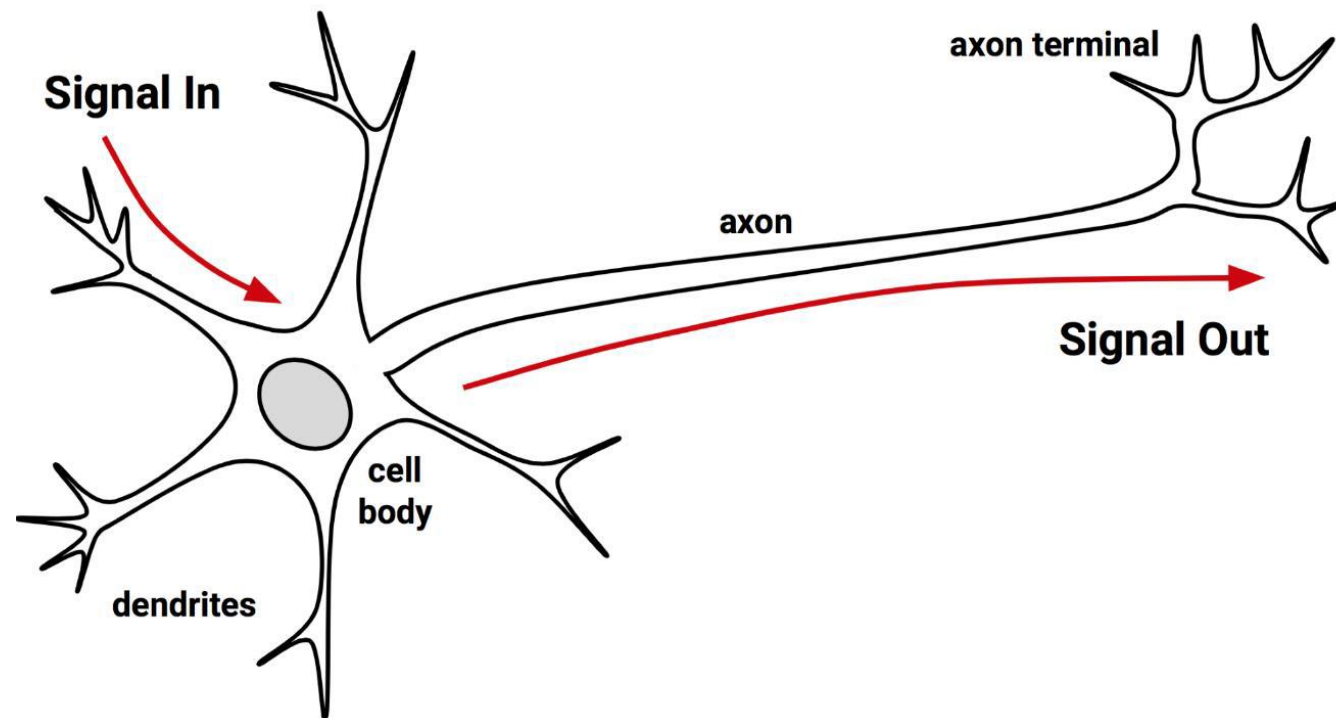


Image Source: Brett Lantz, Machine Learning with R, 3/e.



■ 인공적(수학적)인 뉴런:

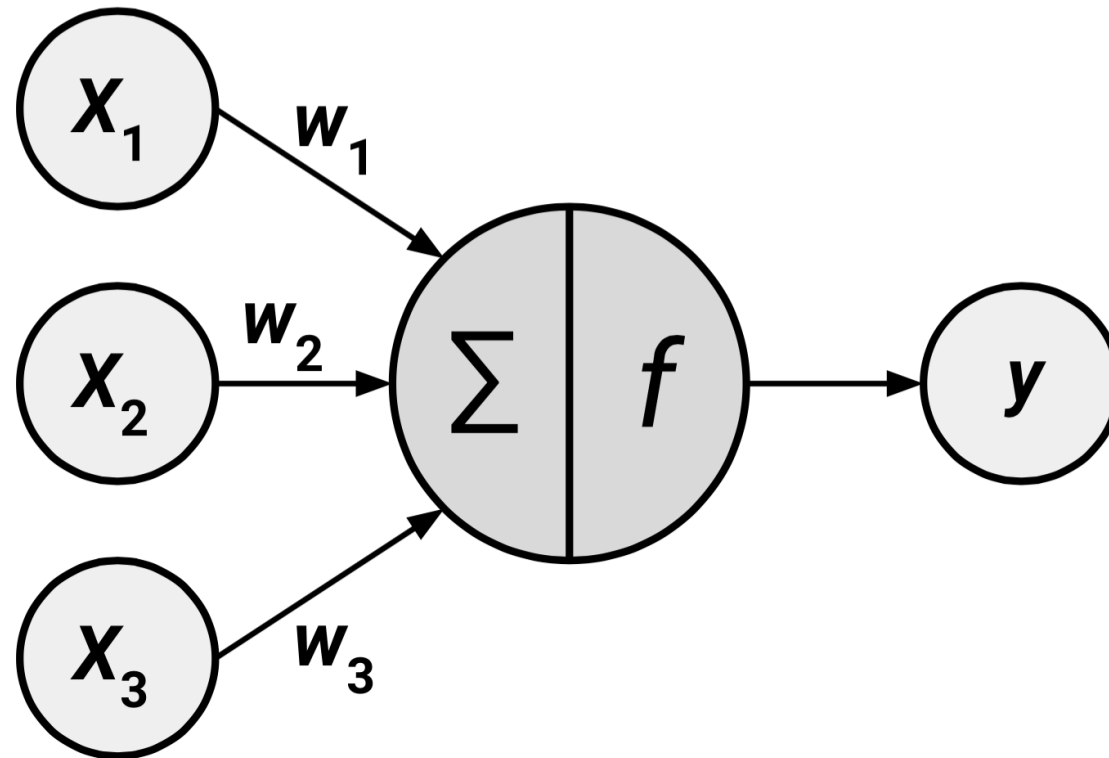


Image Source: Brett Lantz, Machine Learning with R, 3/e.



06. 인공지능망

■ 퍼셉트론: *Perceptron*

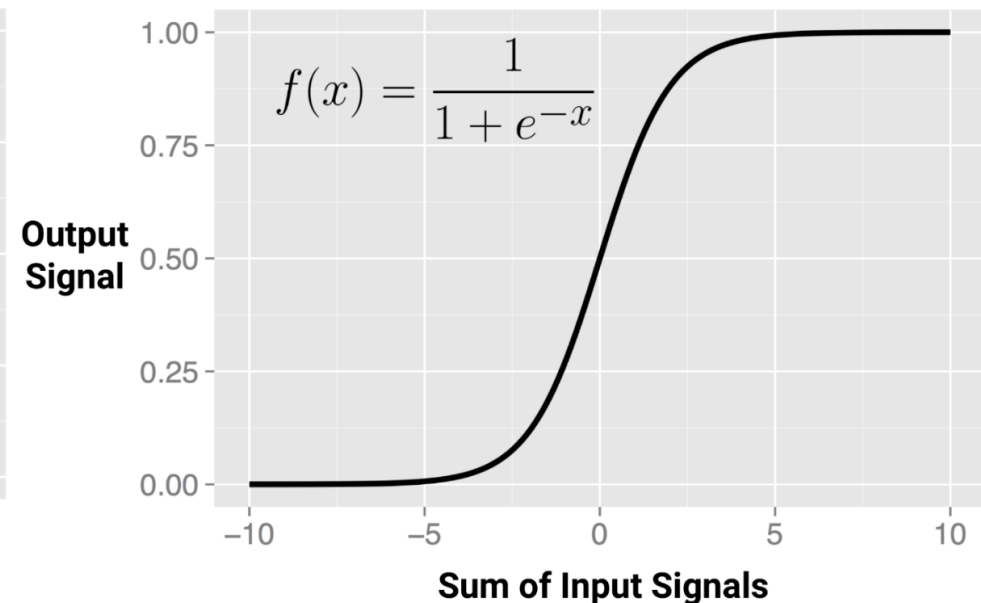
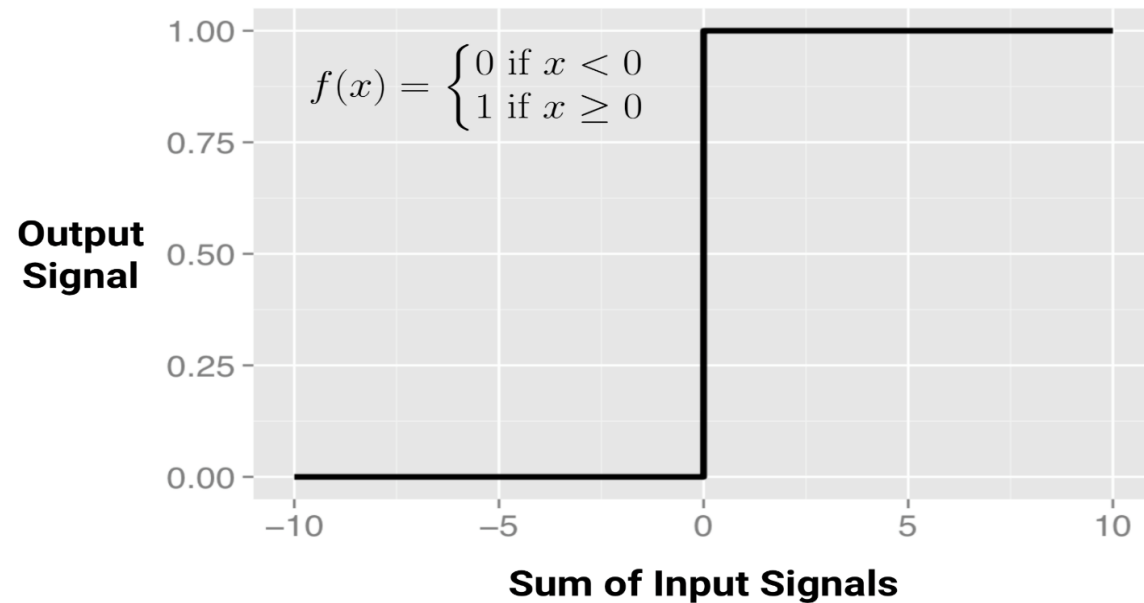
- 뉴런의 동작 방식을 모방하여 만든 수학적 모델
- 입력값: x_1, x_2, \dots, x_n
- 가중치: w_1, w_2, \dots, w_n
 - 입력값의 합을 구할때, 해당 입력값을 강화하거나 약화하기 위해 곱하는 값
 - $y = f(x_1, x_2, \dots, x_n) = w_1x_1 + w_2x_2 + \dots w_nx_n$
- 활성화 함수: *Activation Function*
 - y 의 값이 임계값(threshold)보다 크면 1, 아니면 0을 출력하는 함수
 - $$g(y) = \begin{cases} 1 & y > threshold \\ 0 & y \leq threshold \end{cases}$$
- 출력값: $y = 1$ 이면 다음 퍼셉트론으로 전달하고, $y = 0$ 이면 전달하지 않음



06. 인공지능망

■ 활성화 함수: *Activation Function*

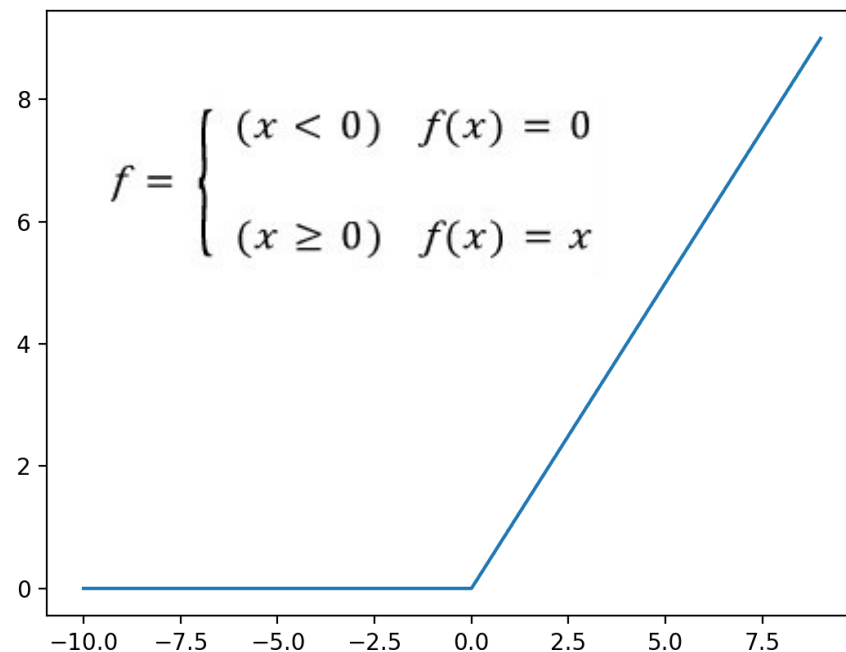
- 입력신호의 총합을 바탕으로 출력값을 0 또는 1로 결정해 주는 함수
- 시그모이드 함수: 미분가능(*differentiable*)한 성질을 가짐





■ 활성화 함수: ReLU 함수

- 교정된 선형 단위 함수: *Rectified* Linear Unit
- 시그모이드 함수의 경사 손실 현상: *Vanishing Gradient*
 - 경사하강법으로 최적의 파라미터를 찾을 때 기울기가 0에 가까워짐
- ReLU: 0보다 큰 값은 출력값과 같아지도록 함 (경사 손실을 줄여줌)



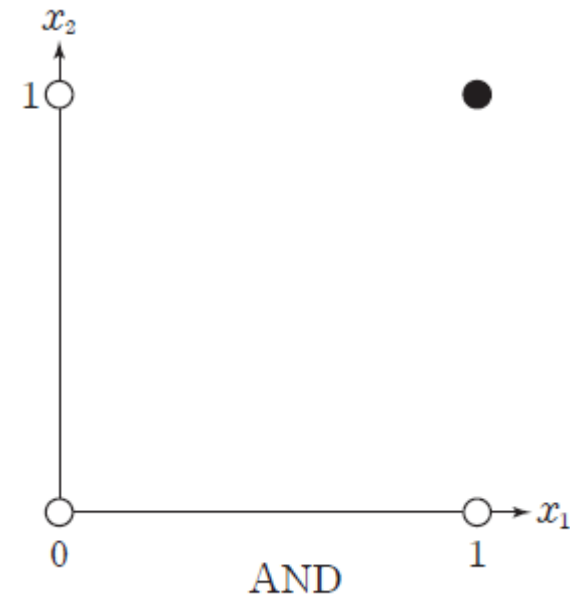


06. 인공지능망

■ 퍼셉트론으로 AND 논리회로 만들기:

▼ AND 진리표

입력		연산자	출력
x_1	x_2		y
0	0	AND	0
0	1		0
1	0		0
1	1		1



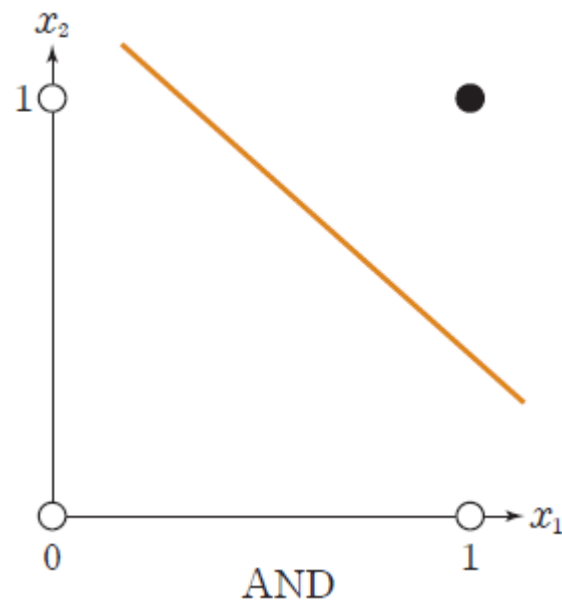
▲ AND 연산의 결과

그림 출처: 수학과 함께 하는 AI 기초, EBS



06. 인공지능망

$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq (\text{임계값})) \\ 1 & (w_1x_1 + w_2x_2 > (\text{임계값})) \end{cases}$$



▼ 퍼셉트론식을 적용한 AND 진리표

입력		퍼셉트론	출력
x_1	x_2	$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq (\text{임계값})) \\ 1 & (w_1x_1 + w_2x_2 > (\text{임계값})) \end{cases}$ $(w_1, w_2, \text{임계값}) = (0.2, 0.2, 0.3)$	y
0	0	$0 \times 0.2 + 0 \times 0.2 \leq 0.3$	0
0	1	$0 \times 0.2 + 1 \times 0.2 \leq 0.3$	0
1	0	$1 \times 0.2 + 0 \times 0.2 \leq 0.3$	0
1	1	$1 \times 0.2 + 1 \times 0.2 > 0.3$	1

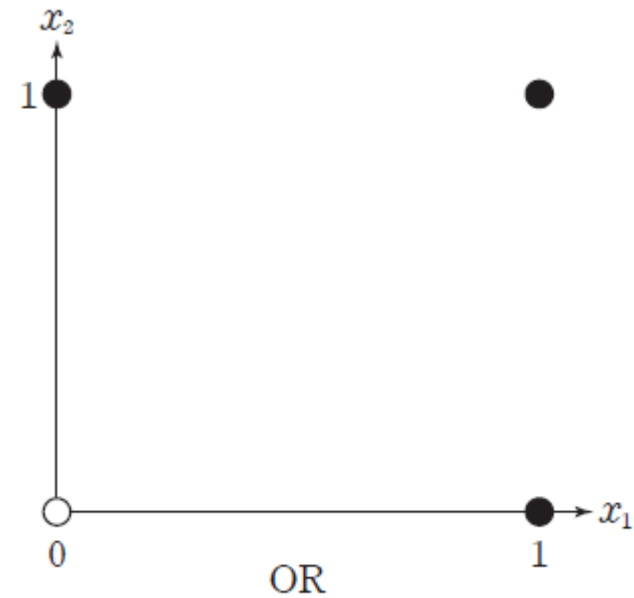


06. 인공지능망

■ 퍼셉트론으로 OR 논리회로 만들기:

▼ OR 진리표

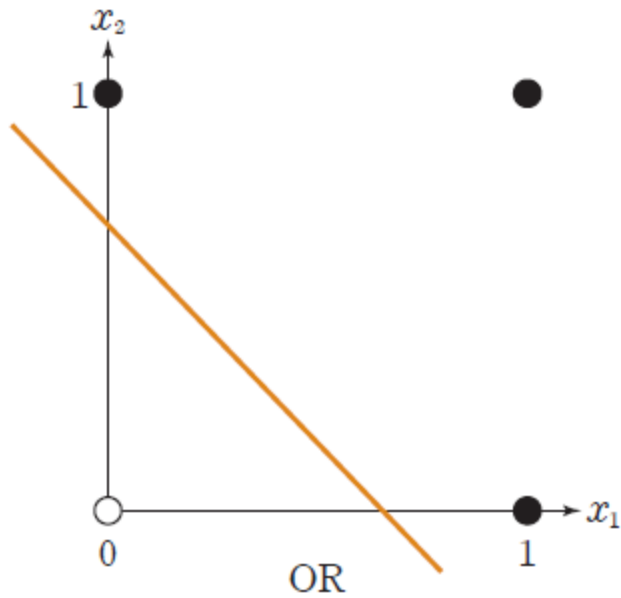
입력		연산자	출력
x_1	x_2		y
0	0	OR	0
0	1		1
1	0		1
1	1		1



▲ OR 연산의 결과



06. 인공지능망



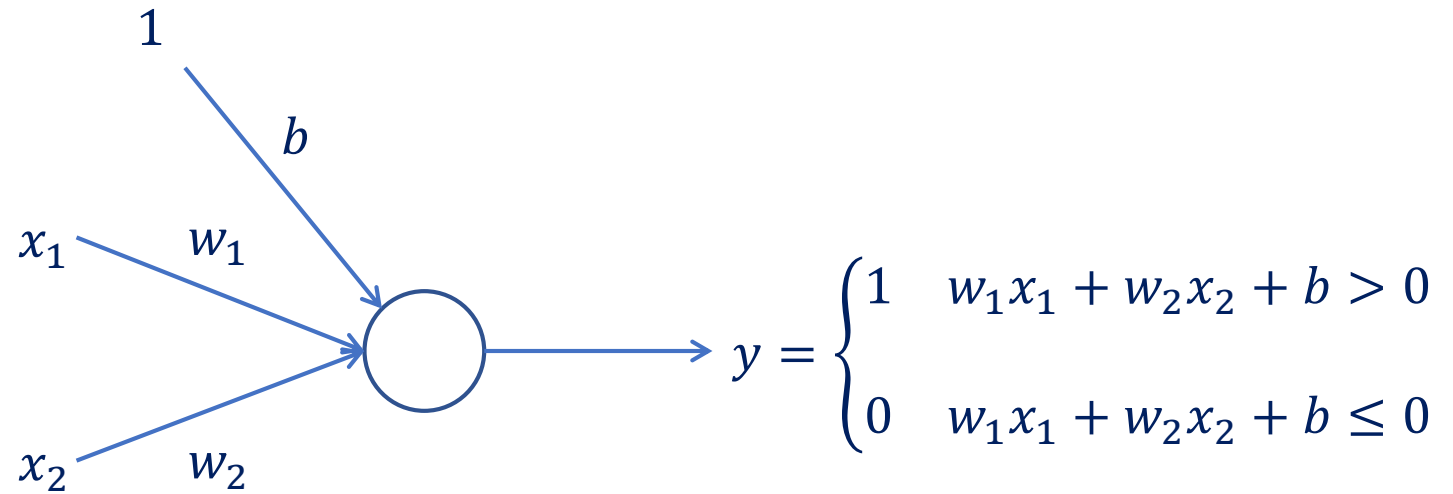
▲ OR 연산 결과를 분류하는 퍼셉트론

▼ OR 진리표

입력		퍼셉트론	출력
x_1	x_2	$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq (\text{임계값})) \\ 1 & (w_1x_1 + w_2x_2 > (\text{임계값})) \end{cases}$ $(w_1, w_2, \text{임계값}) = (0.3, 0.3, 0.2)$	y
0	0	$0 \times 0.3 + 0 \times 0.3 \leq 0.2$	0
0	1	$0 \times 0.3 + 1 \times 0.3 > 0.2$	1
1	0	$1 \times 0.3 + 0 \times 0.3 > 0.2$	1
1	1	$1 \times 0.3 + 1 \times 0.3 > 0.2$	1

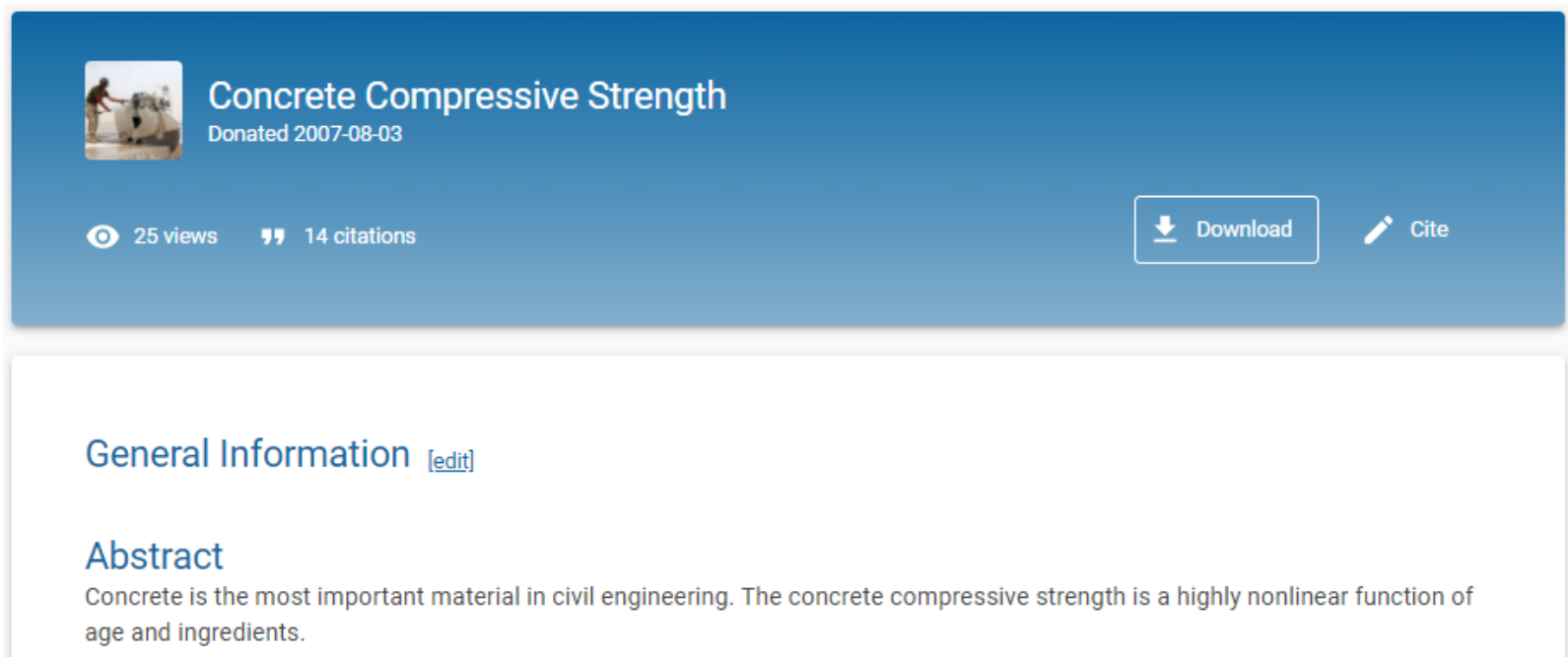
06. 인공지능망

■ 퍼셉트론의 구성:



06. 인공지능경망

- 콘크리트 내압 강도 데이터셋: concrete.csv
 - UCI M/L Repository: Concrete Compressive Strength Dataset
 - <https://archive-beta.ics.uci.edu/ml/datasets/165>



The screenshot shows the UCI Machine Learning Repository page for the 'Concrete Compressive Strength' dataset. The header is blue with a small image of a person working with concrete. The title 'Concrete Compressive Strength' is displayed, along with the date 'Donated 2007-08-03'. Below the title, it shows '25 views' and '14 citations'. There are buttons for 'Download' and 'Cite'. The 'General Information' section is visible, with an '[edit]' link. The 'Abstract' section follows, stating: 'Concrete is the most important material in civil engineering. The concrete compressive strength is a highly nonlinear function of age and ingredients.'

Data & Source Code: Brett Lantz, Machine Learning with R, 3/e.

06. 인공지능경망

■ 데이터 탐색:

- 9개의 변수, 1030개의 관측값
- 종속변수:
 - strength (numeric): 시멘트의 내압 강도
- 독립변수: 8개 (모두 numeric)
 - cement: 시멘트, slag: 슬랙, ash: 재, water: 물,
 - superplastic: 고성능 감수제, coarseagg: 굵은 골재, fineagg: 가는 골재,
 - age: 숙성 시간



06. 인공지능망

■ R: neuralnet

```
# original source:  
# https://github.com/PacktPublishing/Machine-Learning-with-R-Third-Edition  
  
#install.packages("neuralnet")  
library(neuralnet)  
  
df <- read.csv("./concrete.csv")  
str(df)  
  
normalize <- function(x) {  
  return((x - min(x)) / (max(x) - min(x)))  
}  
df <- as.data.frame(lapply(df, normalize))
```



06. 인공지능경망

```
> str(df)
'data.frame': 1030 obs. of 9 variables:
 $ cement      : num  141 169 250 266 155 ...
 $ slag        : num  212 42.2 0 114 183.4 ...
 $ ash         : num  0 124.3 95.7 0 0 ...
 $ water       : num  204 158 187 228 193 ...
 $ superplastic: num  0 10.8 5.5 0 9.1 0 0 6.4 0 9 ...
 $ coarseagg   : num  972 1081 957 932 1047 ...
 $ fineagg     : num  748 796 861 670 697 ...
 $ age         : int   28 14 28 28 28 90 7 56 28 28 ...
 $ strength    : num  29.9 23.5 29.2 45.9 18.3 ...
```




06. 인공지능망

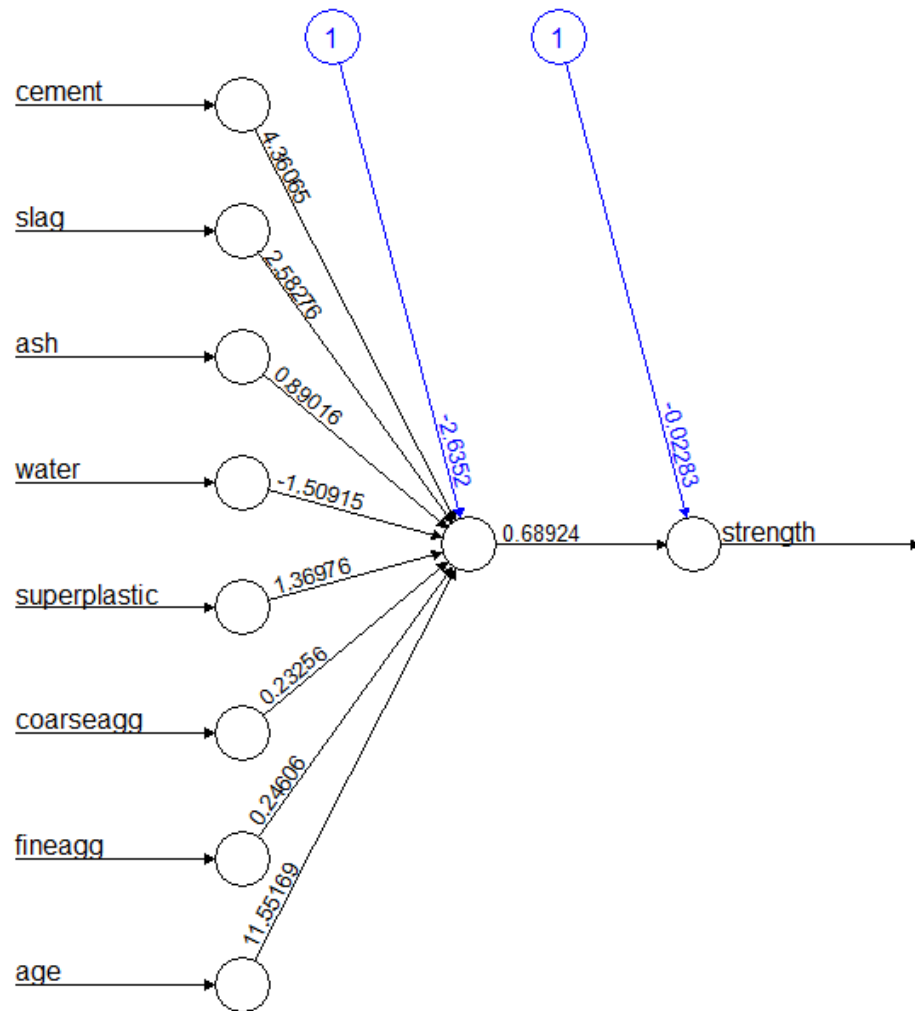
```
model <- neuralnet(strength ~ ., data = df)
summary(model)
head(model$response)
plot(model)

results <- compute(model, df[1:8])
summary(results)
df$predict <- results$net.result
head(df[, 9:10])

cor(df$strength, df$predict)
```



06. 인공지능망



Error: 6.910303 Steps: 5252

```
> head(df[, 9:10])
      strength  predict
1 0.3433412 0.2155125
2 0.2638595 0.2592632
3 0.3349944 0.2933244
4 0.5421702 0.2421023
5 0.1988290 0.2909476
6 0.2433038 0.4997745
> cor(df$strength, df$predict)
      [,1]
[1,] 0.8306087
```



06. 인공지능망

■ 선형 회귀와 ANN의 성능 비교:

```
model <- lm(strength ~ ., data=df)
model
df$linear <- predict(model, data=df)
head(df[, 9:11])
```

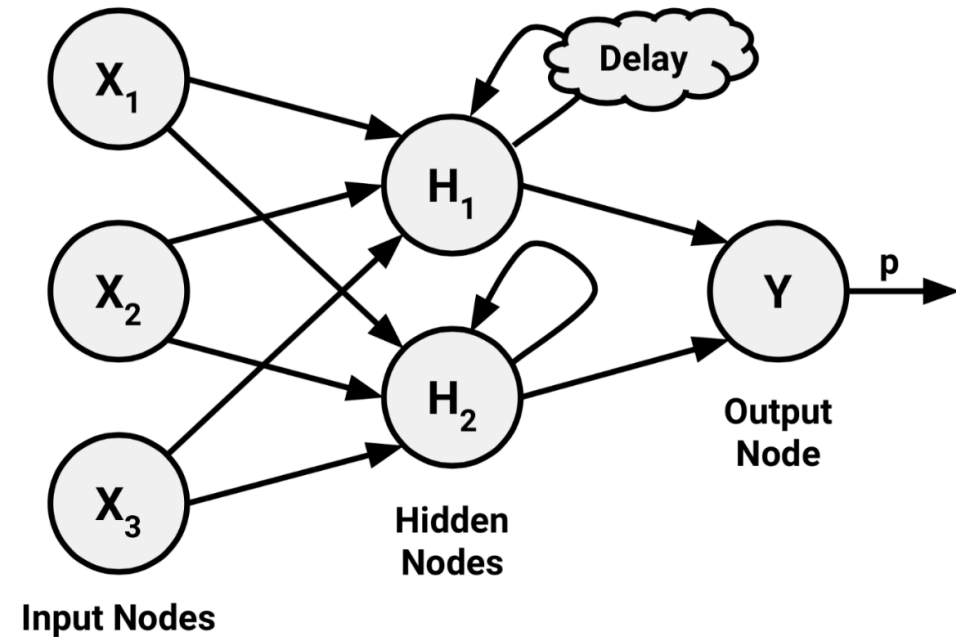
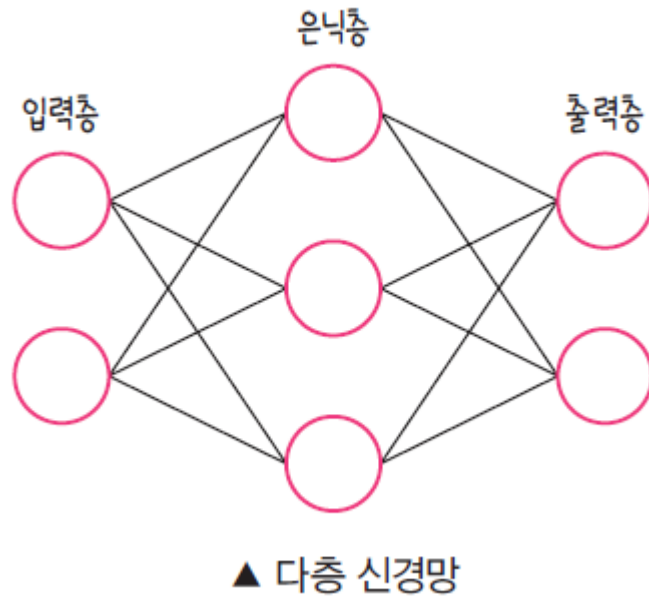
```
cor(df$strength, df$linear)
```

```
> head(df[, 9:11])
      strength    predict    linear
1 0.3433412 0.2155125 0.2062624
2 0.2638595 0.2592632 0.2767071
3 0.3349944 0.2933244 0.2838415
4 0.5421702 0.2421023 0.2121498
5 0.1988290 0.2909476 0.2830656
6 0.2433038 0.4997745 0.4210968
> cor(df$strength, df$linear)
[1] 0.8505233
```



06. 인공지능망

■ 다층 퍼셉트론: MLP, *Multi-Layer Perceptron*





06. 인공지능망

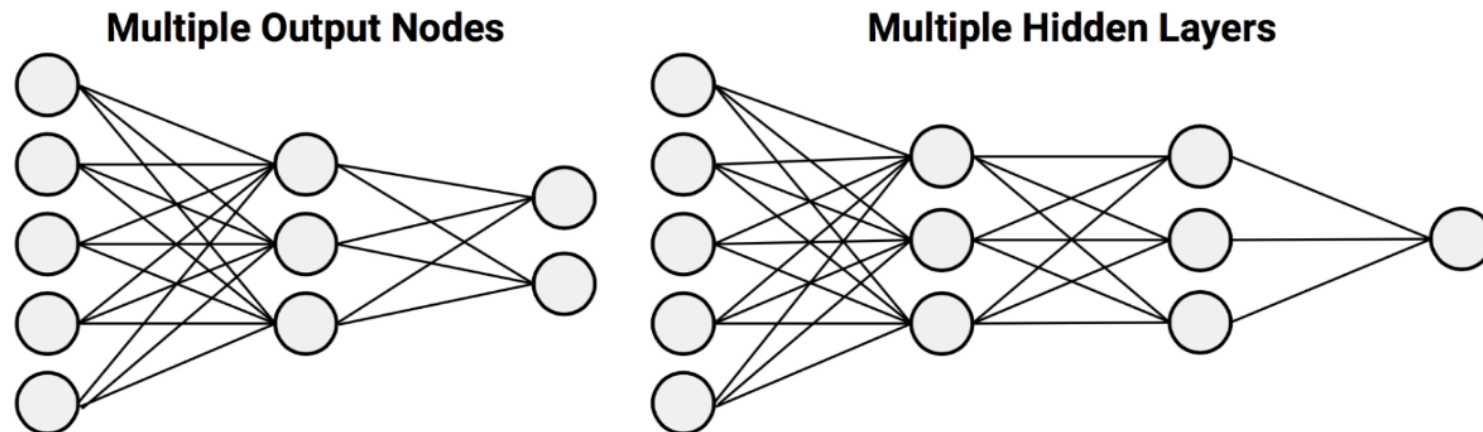
■ 다층 퍼셉트론의 구성:

- 입력층: *Input* Layer
 - 신호가 입력되는 곳
- 은닉층: *Hidden* Layer
 - 입력층의 신호들을 모아서 출력층으로 신호를 전달하는 곳
 - 노드가 많을수록 더 결과가 정확하겠지만, 계산량도 많아짐
- 출력층: *Output* Layer
 - 은닉층으로부터 받은 값을 출력해 주는 곳



06. 인공지능망

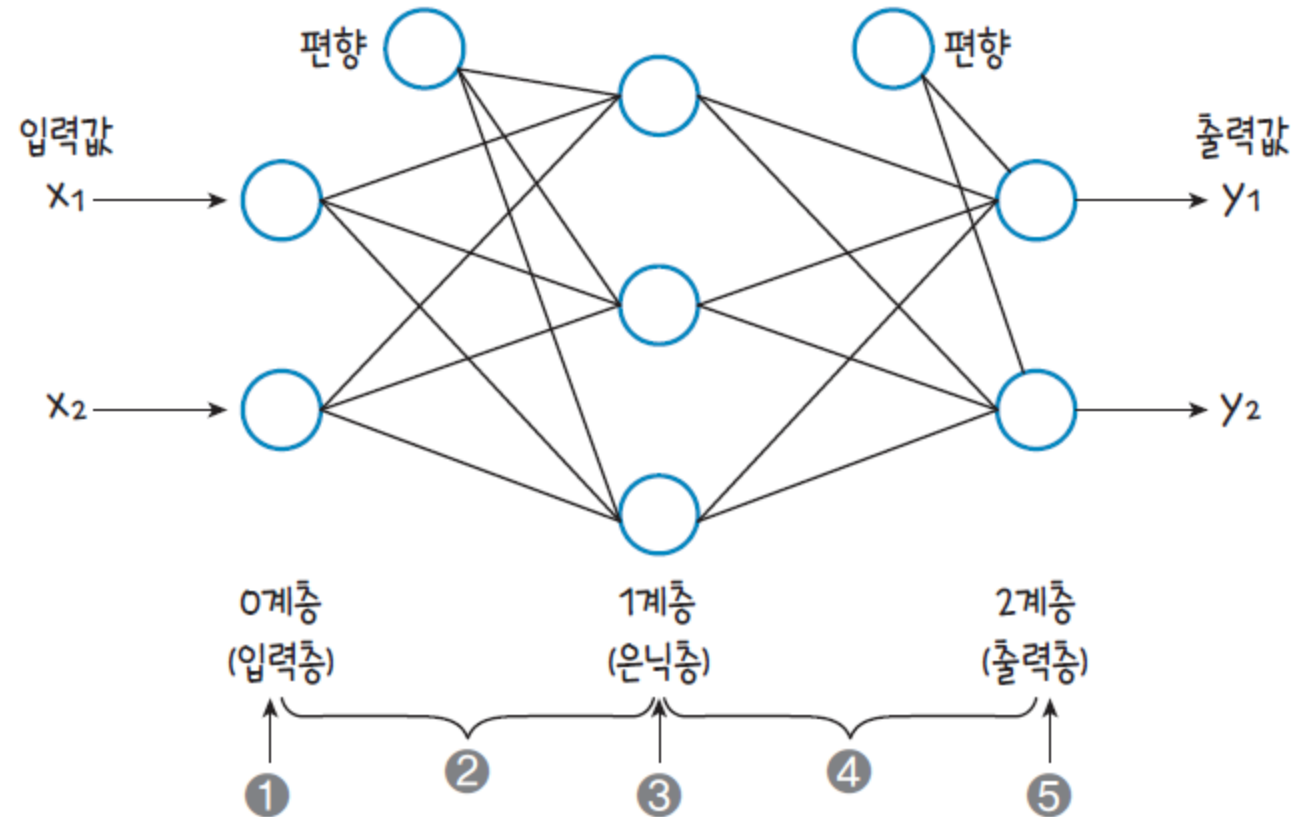
- 인공지능망(ANN)의 네트워크 토폴로지: Network *Topology*
 - 몇 개의 계층으로 구성할 것인가? 은닉층은 여러 개 있을 수 있다.
 - 네트워크의 각 계층별로 몇 개의 노드를 둘 것인가?
 - 정보는 순방향으로만 흐를 것인가, 역방향으로만 흐를 것인가?





06. 인공지능망

- 인공지능망의 순전파: *Feed Forward*
 - 정보의 흐름이 입력층에서 출력층까지 **순방향**으로만 진행





06. 인공지능망

x_n : 입력층의 n 번째 노드에 최초 입력되는 값 x

y_n : 출력층의 n 번째 노드에서 최종 출력되는 값 y

a_n^k : k 번째 계층의 n 번째 노드에 입력된 신호의 합 a

z_n^k : k 번째 계층의 n 번째 노드에서 출력되는 신호 z

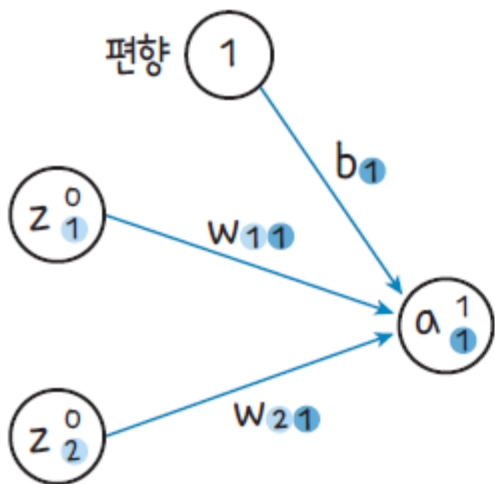
w_{mn} : 이전 계층의 m 번째 노드에서 다음 계층의 n 번째 노드로 신호 전달 시 적용되는 가중치 w

b_n : 다음 계층의 n 번째 노드로 신호 전달 시 적용되는 편향 b



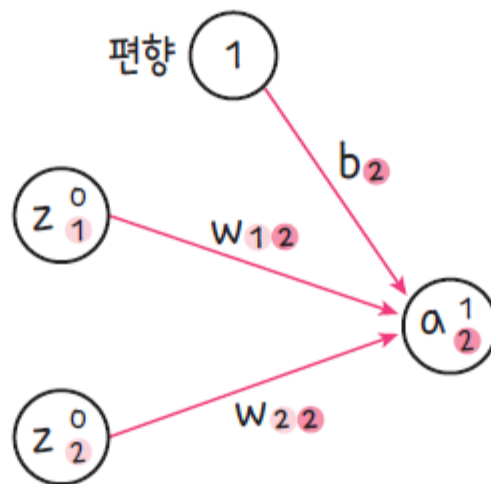
06. 인공지능망

은닉층의 첫 번째 노드



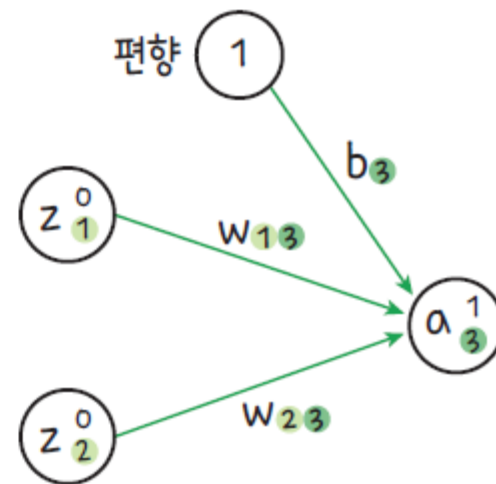
$$a_1^1 = w_{11}z_1^0 + w_{21}z_2^0 + b_1$$

은닉층의 두 번째 노드



$$a_2^1 = w_{12}z_1^0 + w_{22}z_2^0 + b_2$$

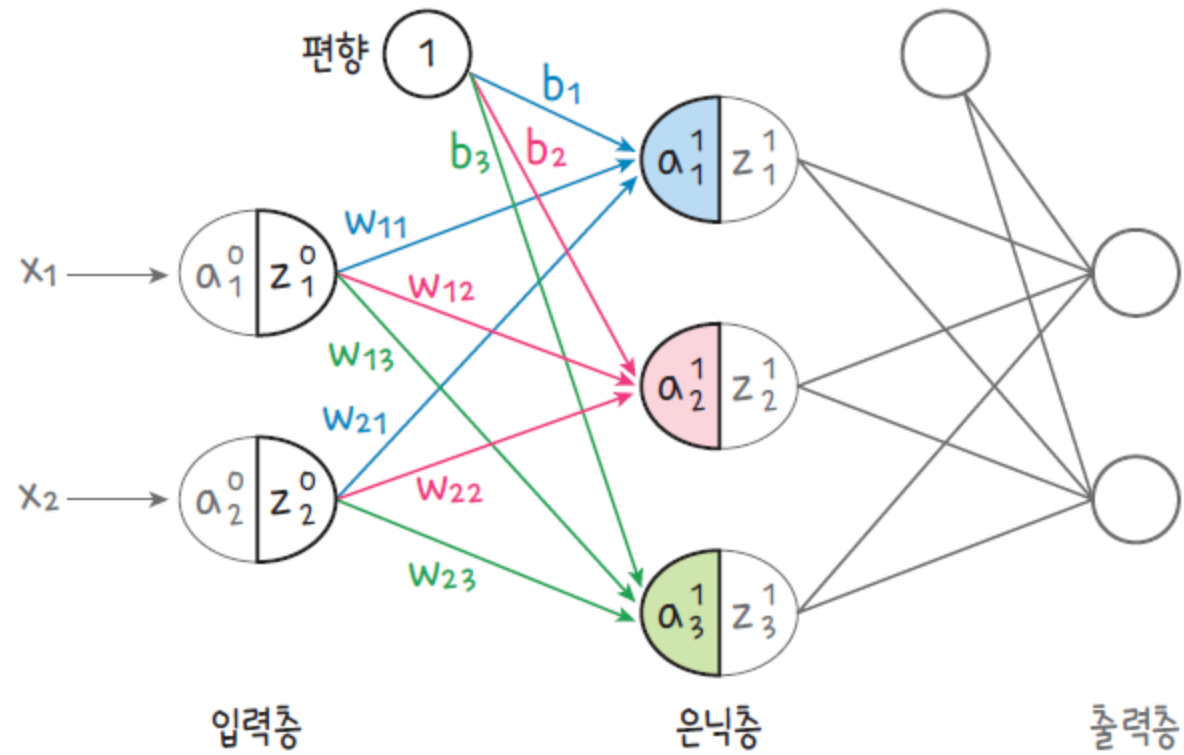
은닉층의 세 번째 노드



$$a_3^1 = w_{13}z_1^0 + w_{23}z_2^0 + b_3$$

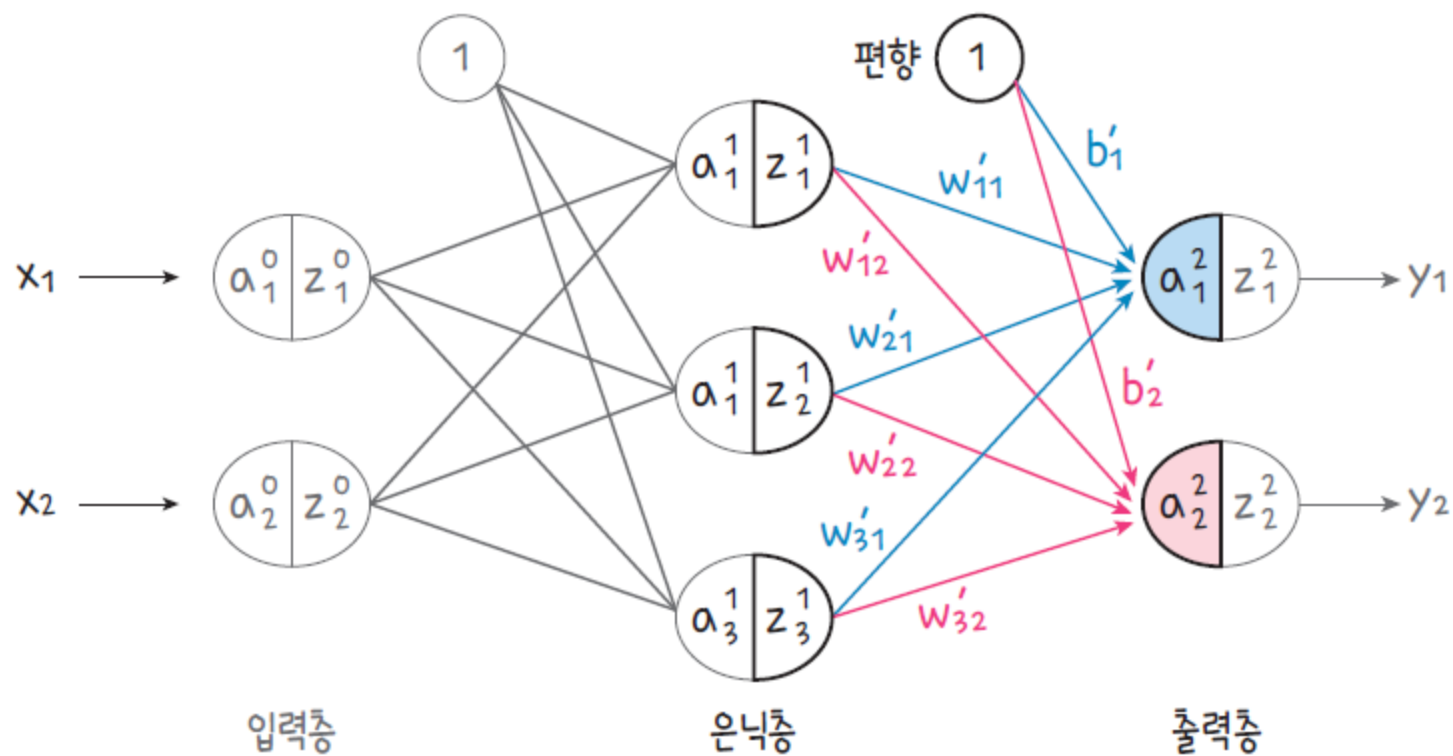


06. 인공지능망





06. 인공지능망





06. 인공지능경망

■ 다층 퍼셉트론 적용:

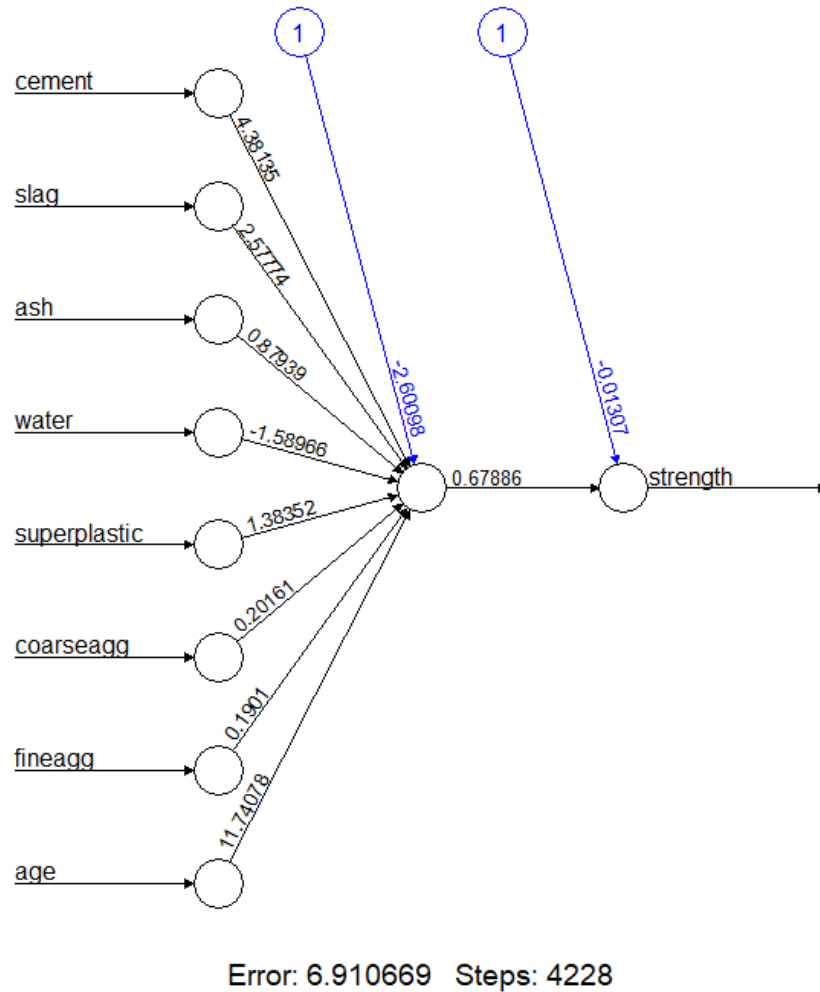
```
df <- df[, 1:9]
model.2 <- neuralnet(strength ~ ., data = df, hidden = 5)
plot(model.2)

results.2 <- compute(model.2, df[, 1:8])
df$predict <- results.2$net.result
head(df[, 9:10])

cor(df$strength, df$predict)
```



06. 인공지능망



```
> head(df[, 9:10])
      strength  predict
1 0.3433412 0.3189175
2 0.2638595 0.2891970
3 0.3349944 0.3666726
4 0.5421702 0.4082607
5 0.1988290 0.3268313
6 0.2433038 0.3144627
> cor(df$strength, df$predict)
      [,1]
[1,] 0.946084
```



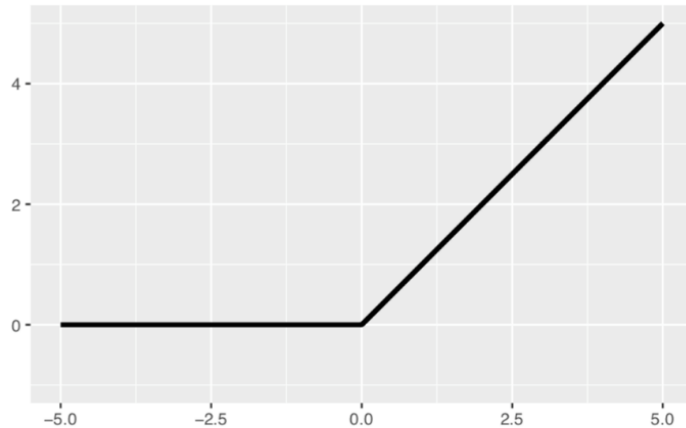
06. 인공지능망

■ 활성화 함수의 선택:

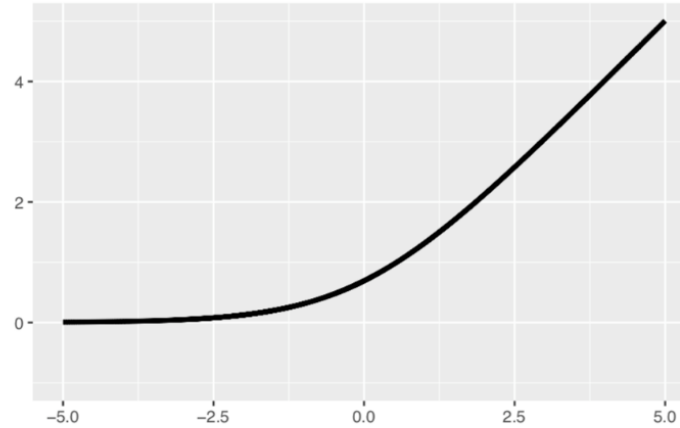
- 소프트플러스 함수: ReLU 함수를 미분가능하도록 변형

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

Rectifier (ReLU)



SoftPlus





06. 인공지능망

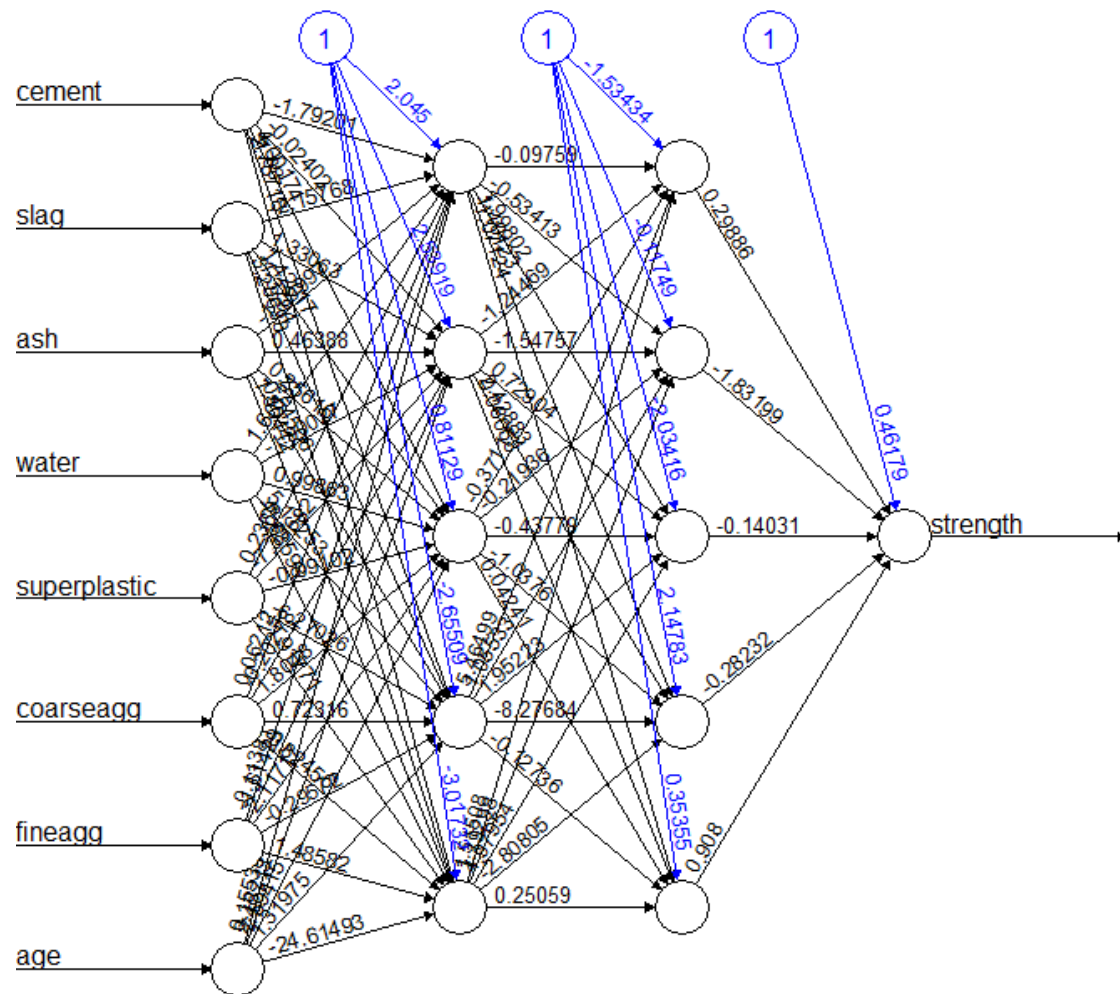
```
df <- df[, 1:9]
softplus <- function (x) {
  log(1 / (1 + exp(x)))
}
model.3 <- neuralnet(strength ~ ., data = df,
                     hidden = c(5, 5),
                     act.fct = softplus)

plot(model.3)

results.3 <- compute(model.3, df[, 1:8])
df$predict <- results.3$net.result
head(df[, 9:10])
```



06. 인공지능망



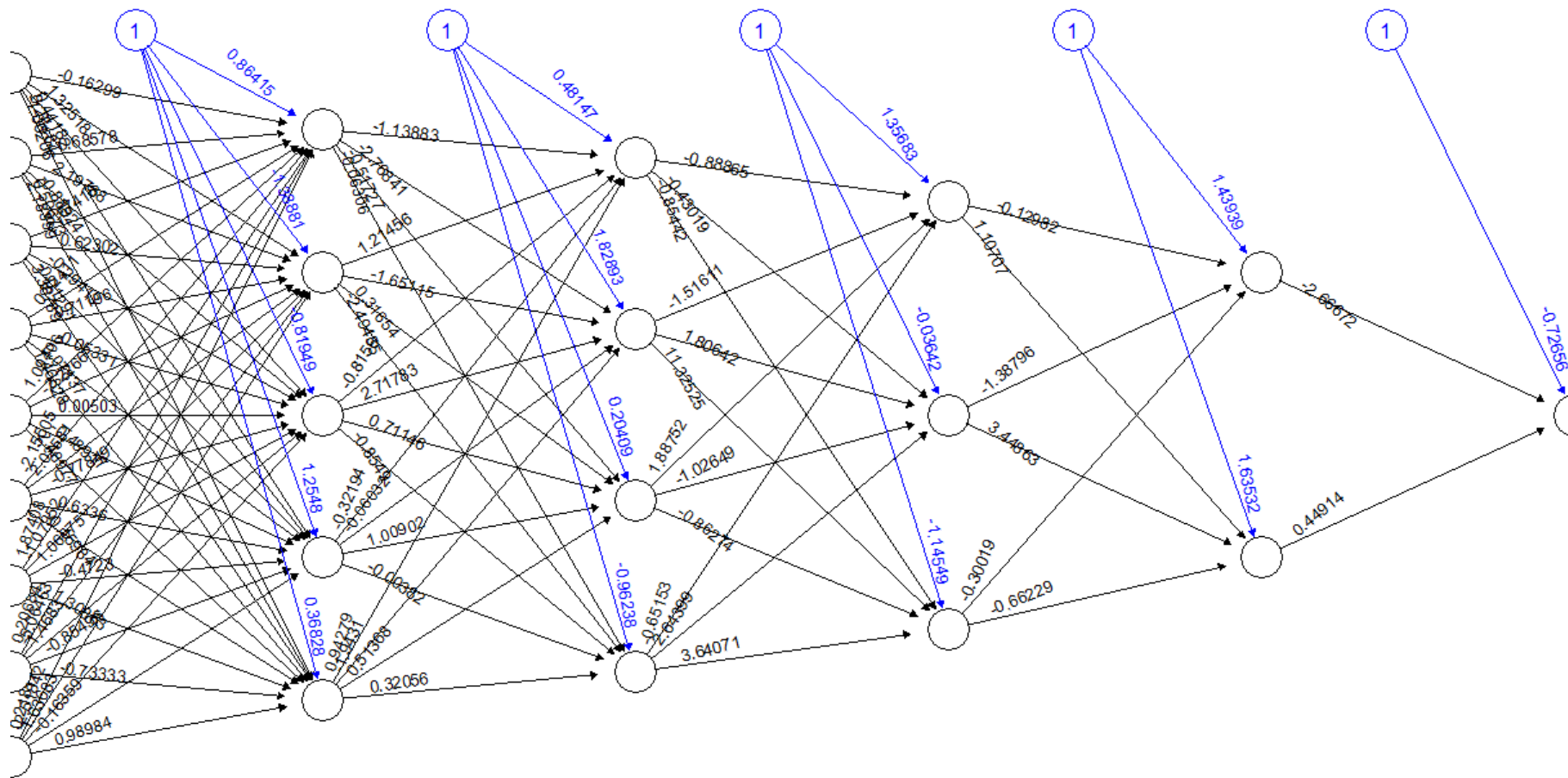
Error: 0.766272 Steps: 20428

```
> head(df[, 9:10])
      strength predict
1 0.3433412 0.3199803
2 0.2638595 0.3150599
3 0.3349944 0.2987671
4 0.5421702 0.5031613
5 0.1988290 0.3052765
6 0.2433038 0.3205856
> cor(df$strength, df$predict)
      [,1]
[1,] 0.9696667
```




06. 인공지능망

```
model.3 <- neuralnet(strength ~ ., data = df,
  hidden = c(5, 4, 3, 2))
```



Error: 0.065972 Steps: 1510

06. 인공지능경망

■ 강화 학습과 딥 러닝:

- 강화 학습: *Reinforcement Learning*
 - 행동(*action*)에 따른 보상(*reward*)을 줌으로써,
 - 보상을 극대화하기 위해 동적으로 학습을 하며 **행동을 조정함**
- 딥 러닝: *Deep (Reinforcement) Learning*
 - DNN(*Deep Neural Network*)을 이용한 강화 학습 방법
 - DNN = MLP + Backpropagation



06. 인공지능망

■ 역전파: *Back-propagation*

- 출력값과 실제값의 오차(error)를 역으로 전달하여 학습하는 방법

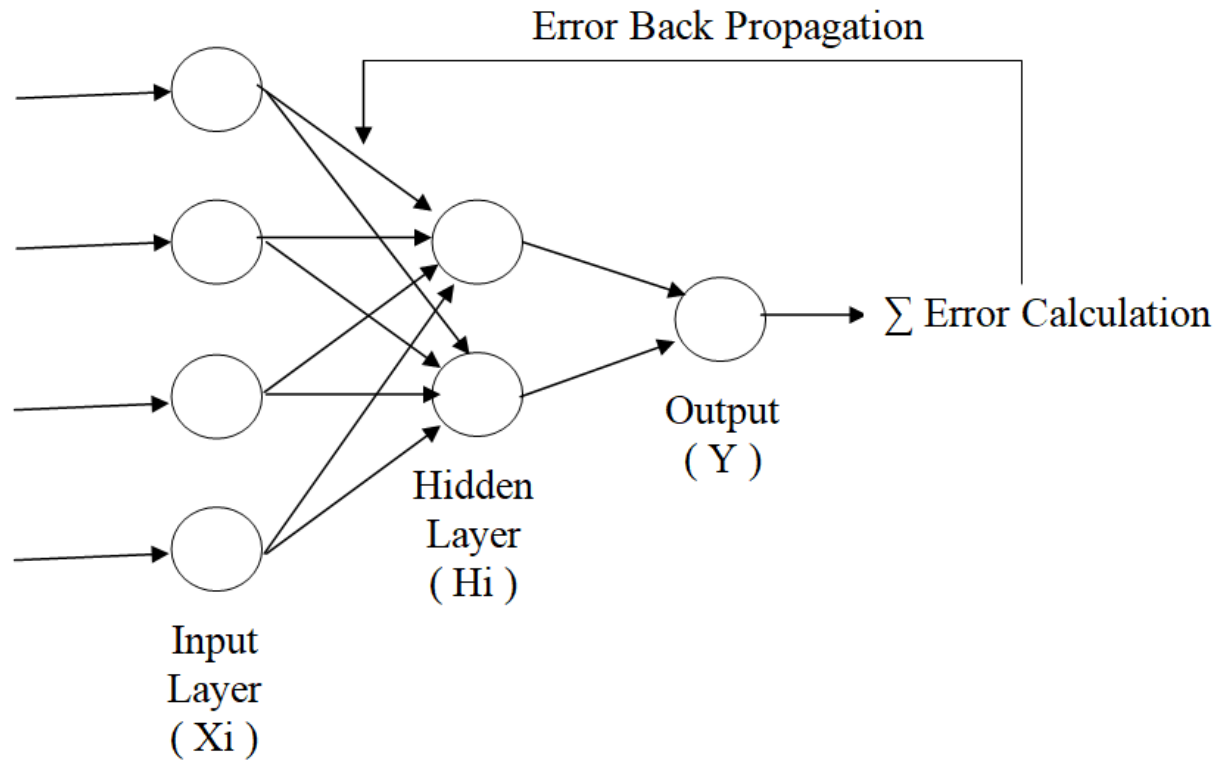
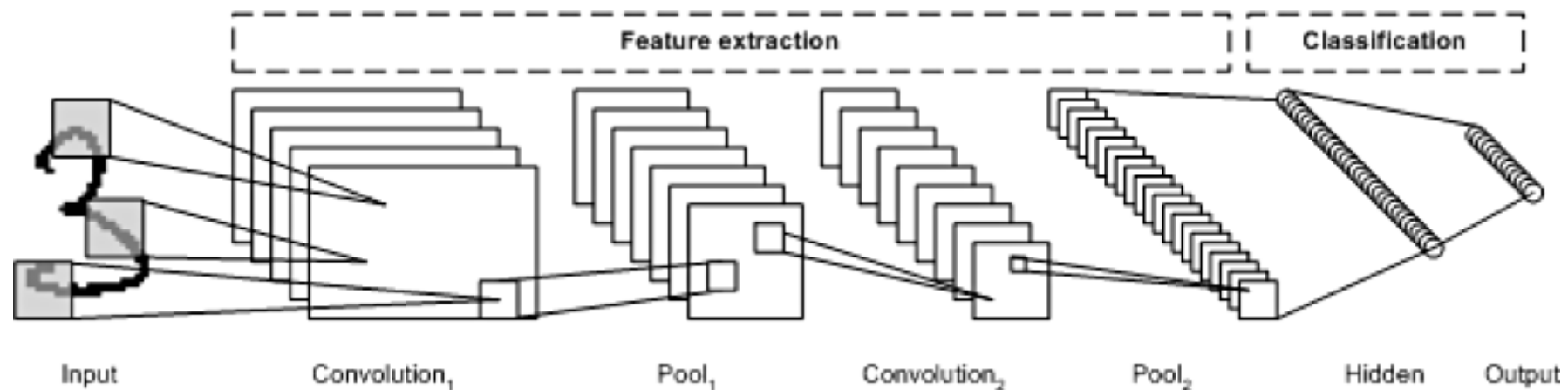


Image Source: https://www.researchgate.net/figure/Sample-Error-Back-Propagation_fig4_322332639



06. 인공지능망

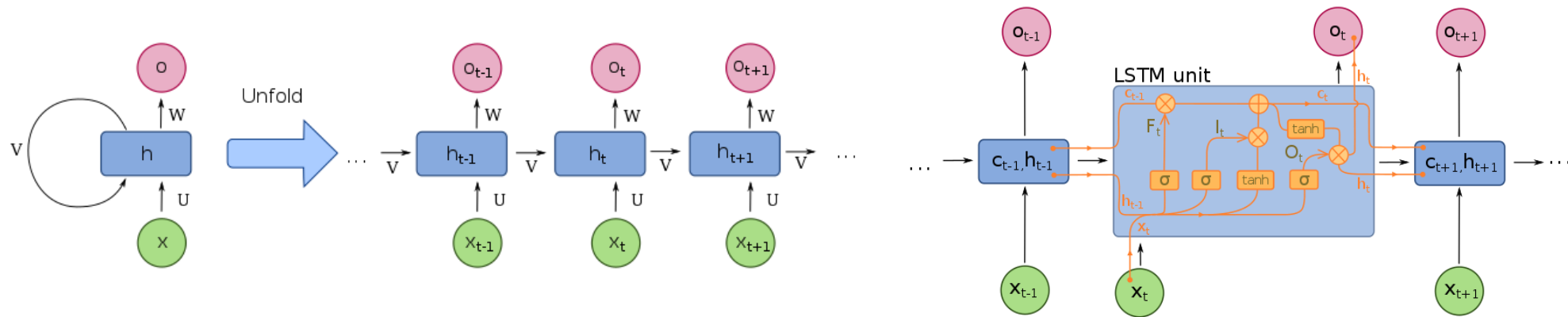
- 합성곱 신경망: CNN, *Convolutional Neural Network*
 - 주로 이미지 인식에 많이 사용되는 심층 신경망
 - 합성곱층과 풀링층을 반복적으로 조합해서 다층 신경망을 구성
 - Convolutional Layer: 입력 이미지에 필터를 적용하여 활성 함수를 반영
 - Pooling Layer: 합성곱층의 출력 크기를 줄이거나 특정 데이터를 강조





06. 인공지능망

- 순환 신경망: RNN, *Recurrent Neural Network*
 - 필기체 인식이나 음성 인식과 같은 시계열 응용에 많이 사용되는 심층 신경망
 - 순환적 구조를 가지고 있어서 신경망 내부에 상태를 저장할 수 있게 해 줌
 - LSTM: Long Short-Term Memory
 - 망각 게이트를 추가해서 장기 기억과 단기 기억을 가질 수 있음



06. 인공지능경망

- **가중치와 편차의 강화 학습:**
 - **SGD**: Stochastic Gradient Descent
 - **L-BFGS-B**: Limited-memory BFGS extended
 - **Adam**: Adaptive Moment Estimation

06. 인공지능경망

■ 가중치와 편차의 학습 방법

- L-BFGS-B: Limited-memory BFGS extended
 - 비선형 최적화 알고리즘: nonlinear optimization
 - L-BFGS: quasi-Newton 방법인 BFGS의 Limited Memory 버전
- Adam: Adaptive Moment Estimation
 - AdaGrad(Adaptive Gradient): 경사 하강을 할 때 step size를 다르게 설정
 - Momentum: 경사 하강을 하는 과정에 일종의 관성을 주는 방법
 - Adam: 위 두 가지를 동시에 적용하는 최적화 방법

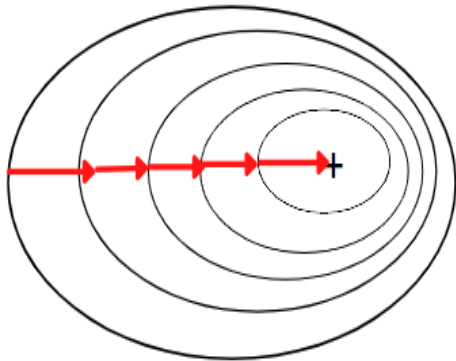


06. 인공지능망

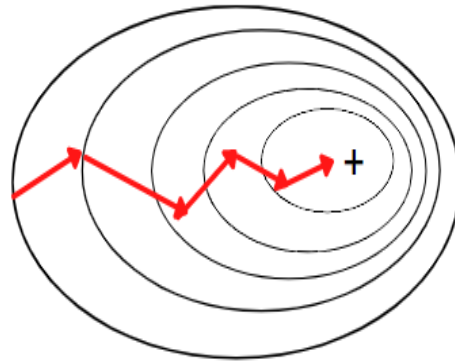
■ 확률적 경사 하강법: Stochastic Gradient Descent

- Batch GD: 손실 함수를 계산할 때 전체 훈련 데이터셋(batch)을 사용
- Mini-Batch GD: 전체 데이터 대신 *mini-batch*로 손실 함수 계산
- Stochastic GD: Mini-Batch에 확률을 도입해서 손실 함수 계산

Batch Gradient Descent



Mini-Batch Gradient Descent



Stochastic Gradient Descent

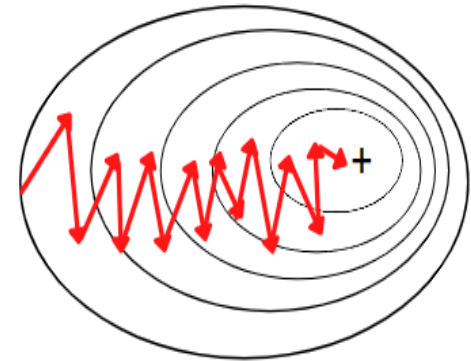


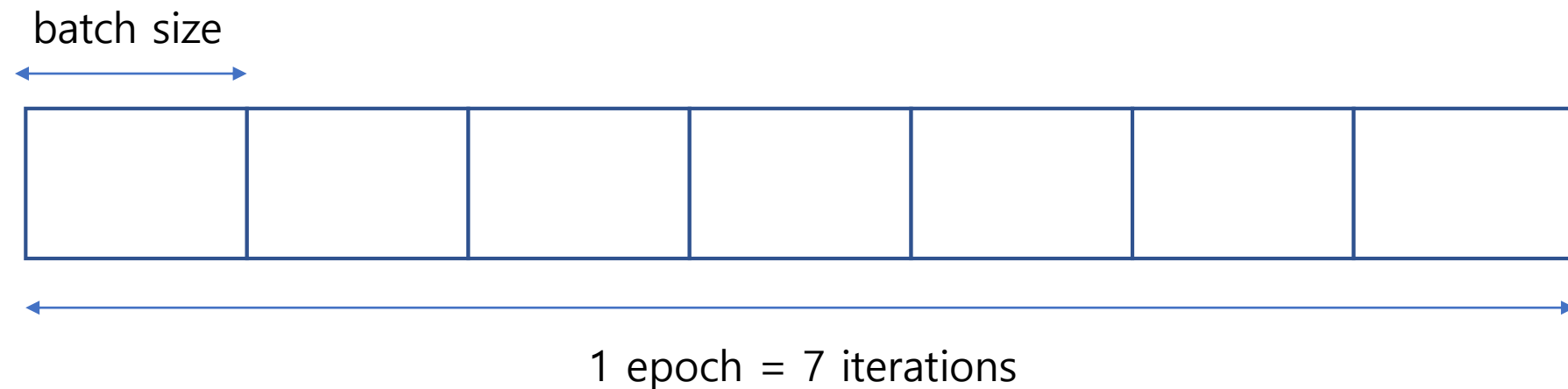
Image Source: <https://laptrinhx.com/understanding-optimization-algorithms-3818430905/>



06. 인공지능망

■ 배치와 에포크: *Batch* and *Epoch*

- 미니-배치: Mini-Batch
 - 전체 훈련 데이터셋을 여러 개의 작은 batch로 나눔
- 에포크: Epoch
 - 전체 훈련 데이터셋이 순전파/역전파를 통해 한 번 학습을 완료함
- 반복: Iteration
 - 한 번의 epoch에서 각 batch 별로 학습





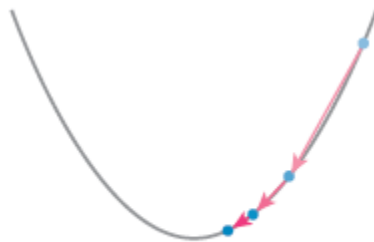
06. 인공지능망

■ 학습률: *Learning Rate*

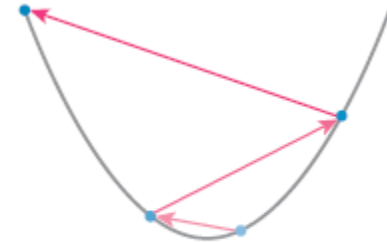
- 경사 하강을 할 때 움직이는 점의 보폭(step size)을 결정하는 상수
- 학습률의 크기에 비례하여 각 epoch마다 보폭을 조절함
 - 학습률이 작을수록 학습 시간이 길어지지만, 정확도는 높아짐
 - 학습률이 높을수록 학습 시간은 줄어들지만, 정확도가 떨어짐



▲ 학습률의 크기가 지나치게 작은 경우



▲ 학습률의 크기가 적당한 경우

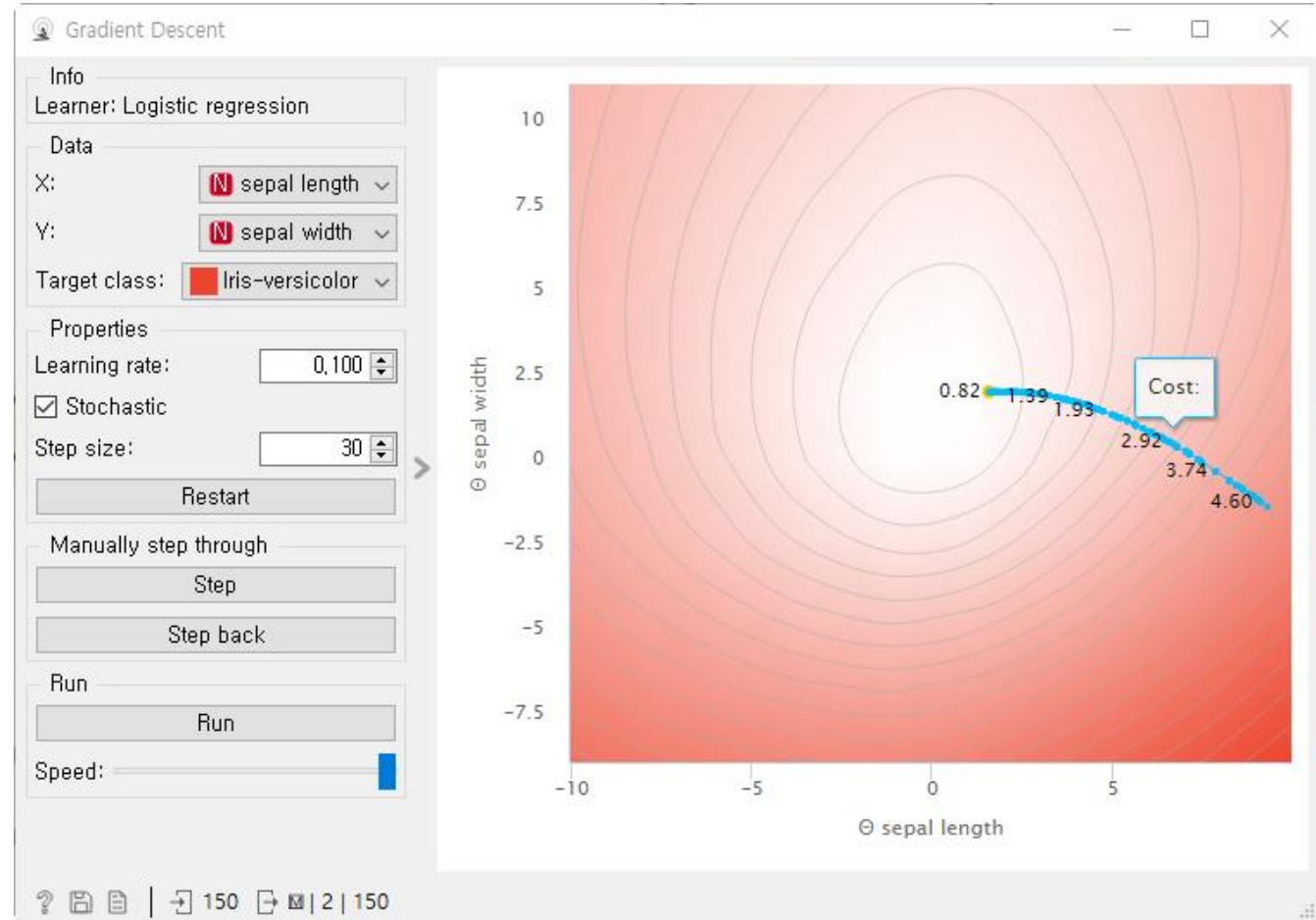


▲ 학습률의 크기가 지나치게 큰 경우



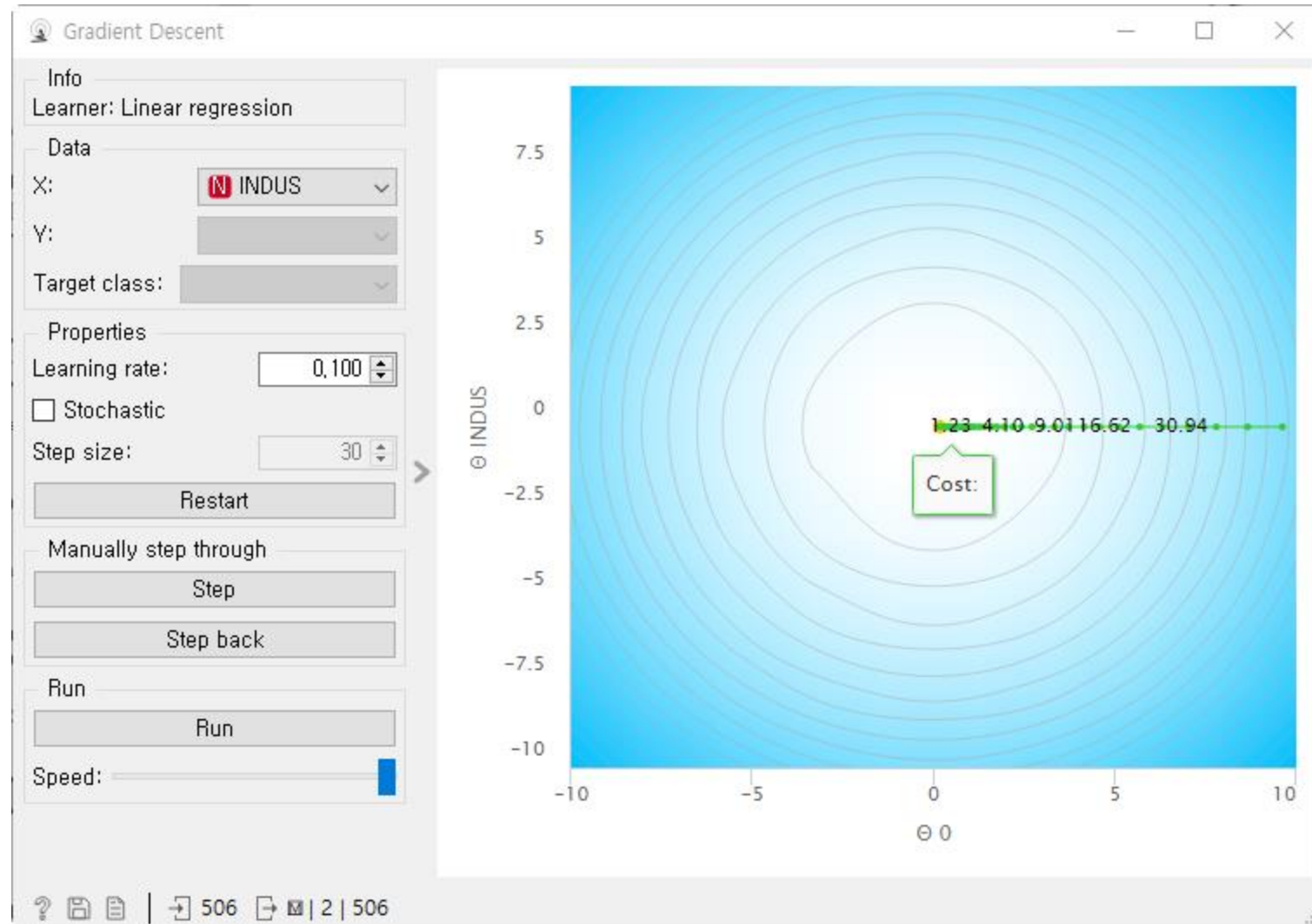
06. 인공지능망

■ Orange: Gradient Descent





06. 인공지능망





06. 인공지능망

- Orange: Neural Network
 - 개와 고양이의 이미지를 구분하는 이진 분류기 만들기



cats



dogs



cats



dogs



cats



dogs



cats



dogs



cats



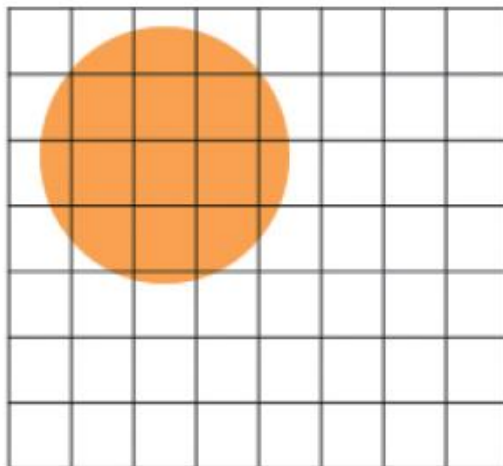
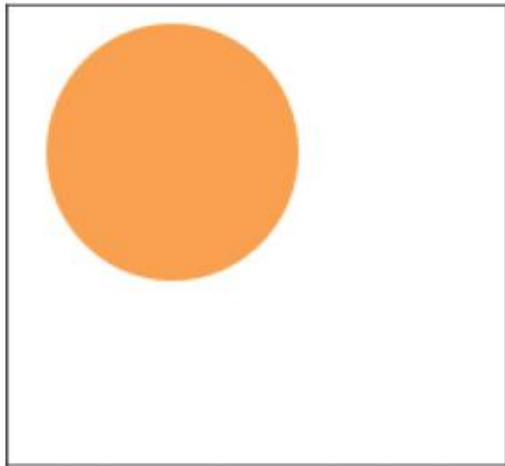
dogs



06. 인공지능망

■ 이미지의 데이터 표현:

- 비트맵 이미지: BMP (Bitmap)
 - 각 픽셀을 하나의 정수로 표현
- 컬러의 표현: 색의 3원소: RGB (Red, Greed, Blue)
 - 색의 원소별로 8비트씩: 한 픽셀당 24비트 필요



0	64	64	64	0	0	0	0	
64	0	171	171	171	0	0	0	0
64	171	0	255	255	255	0	0	0
64	171	255	255	255	255	255	0	0
0	171	255	255	255	255	255	0	0
0	0	255	255	255	255	255	0	0
0	0	255	255	255	255	255	0	0
0	0	0	0	255	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0



06. 인공지능경망

■ 이미지간의 유사도 측정:

- 이미지 임베딩: Image *Embedding*
 - 임베딩: 구조화되어 있지 않은 비정형 데이터로부터 데이터의 특징을 추출
 - 이미지 임베딩: 이미지 데이터의 특성을 반영한 벡터를 생성하는 것
- 개와 고양이 이미지 임베딩:
 - 개와 고양이의 이미지에서 각 픽셀간의 패턴을 특성으로 찾아내기
 - 찾아낸 특성을 기반으로 각 이미지간의 유사도를 측정



06. 인공지능망

■ 훈련용/시험용 데이터 준비: catsanddogs.zip

dataset

catsanddogs

test

training

cats

dogs



cat.4990.jpg



dog.4003.jpg



cat1.jpg



cat2.jpg



cat3.jpg



cat4.jpg



cat5.jpg



dog1.jpg



dog2.jpg



dog3.jpg



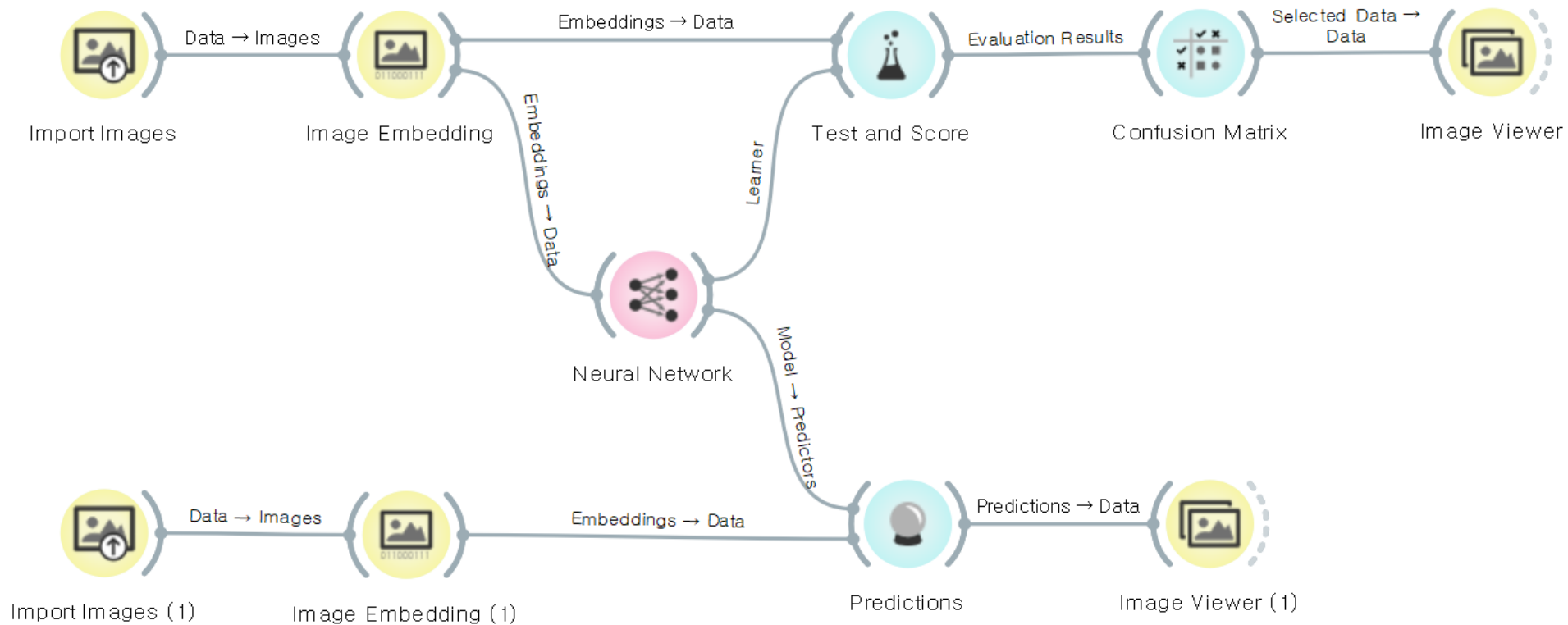
dog4.jpg



dog5.jpg

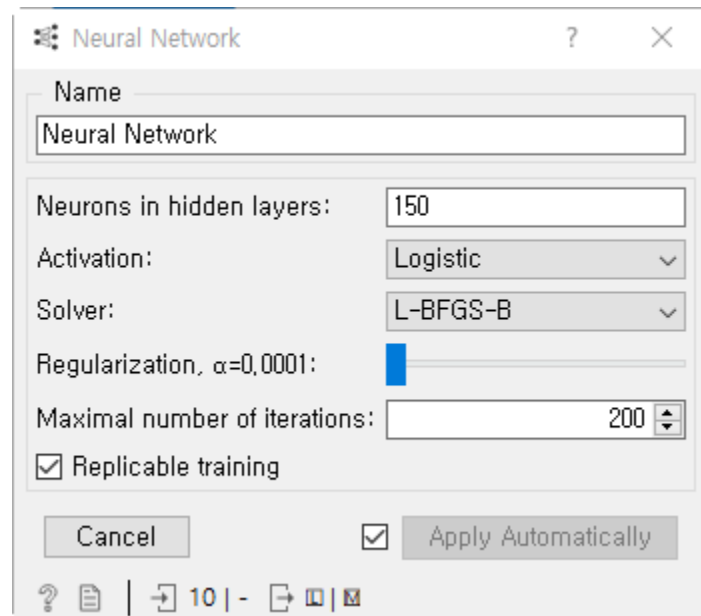
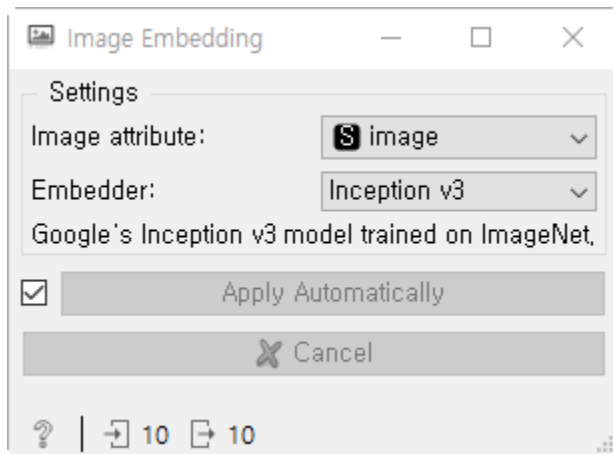


06. 인공지능망



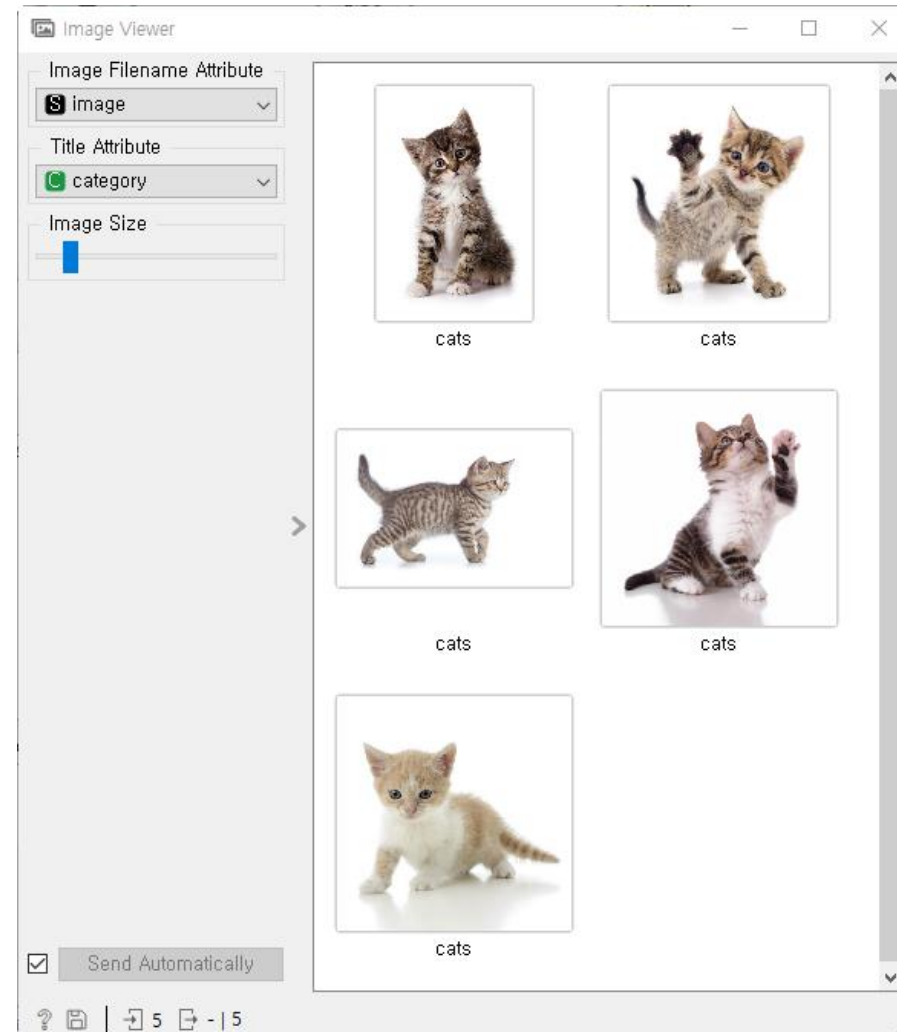
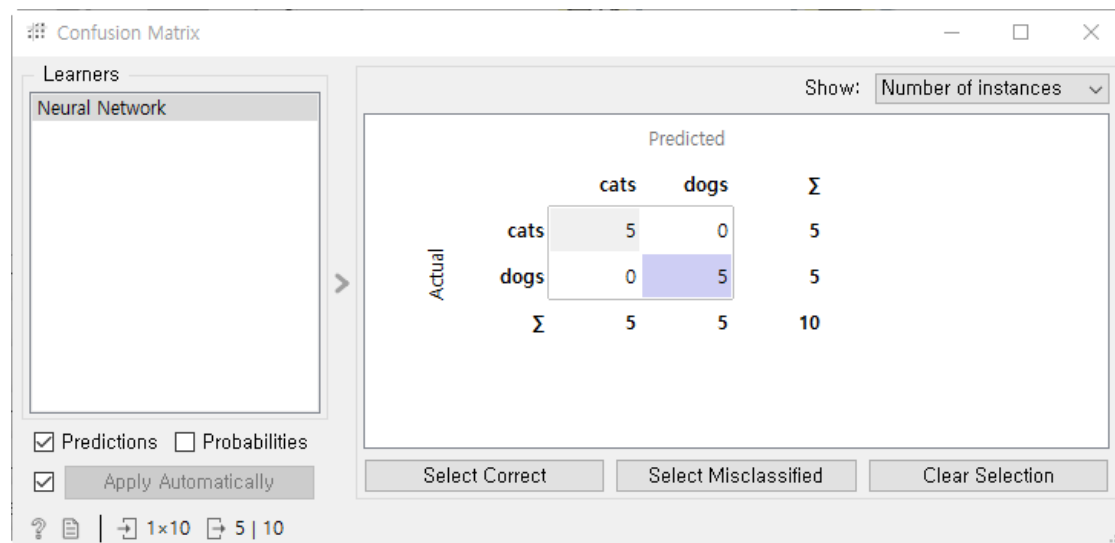


06. 인공지능경망





06. 인공지능망





06. 인공지능망

■ MNIST 데이터셋:

- The MNIST dataset of handwritten digits:
 - <http://yann.lecun.com/exdb/mnist/>
 - 0에서 9까지의 숫자를 손으로 쓴 이미지의 모음
 - 흑백 이미지: 픽셀당 8비트, 해상도: $28 \times 28 = 784$ 픽셀
 - 총 7만장의 데이터: 이 중에서 600개만 추출 (60×10개)

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

06. 인공지능경망

■ 출력층의 구성:

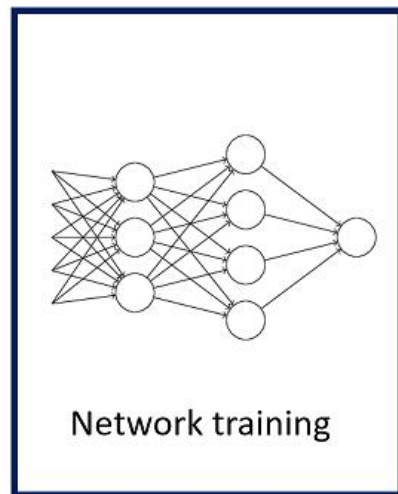
- 원-핫-인코딩: One-Hot-Encoding
 - 데이터를 수많은 0과 한 개의 1의 값으로 구분하는 인코딩 방법
 - 범주형 데이터의 범주가 3개라면 3차원 벡터로 표현
 - IRIS: setosa=(1, 0, 0), versicolor=(0, 1, 0), virginica=(0, 0, 1)



06. 인공지능망

0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 5 5 5 5 5 5 5 5 5 5 5 5 5 5
 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 9 9 9 9 9 9 9 9 9 9 9 9 9 9

Data & Labels



0
1
2
3
4
5
6
7
8
9



06. 인공지능망

■ 입력층의 구성:

- 픽셀 하나당 하나의 입력: 총 784개의 입력값

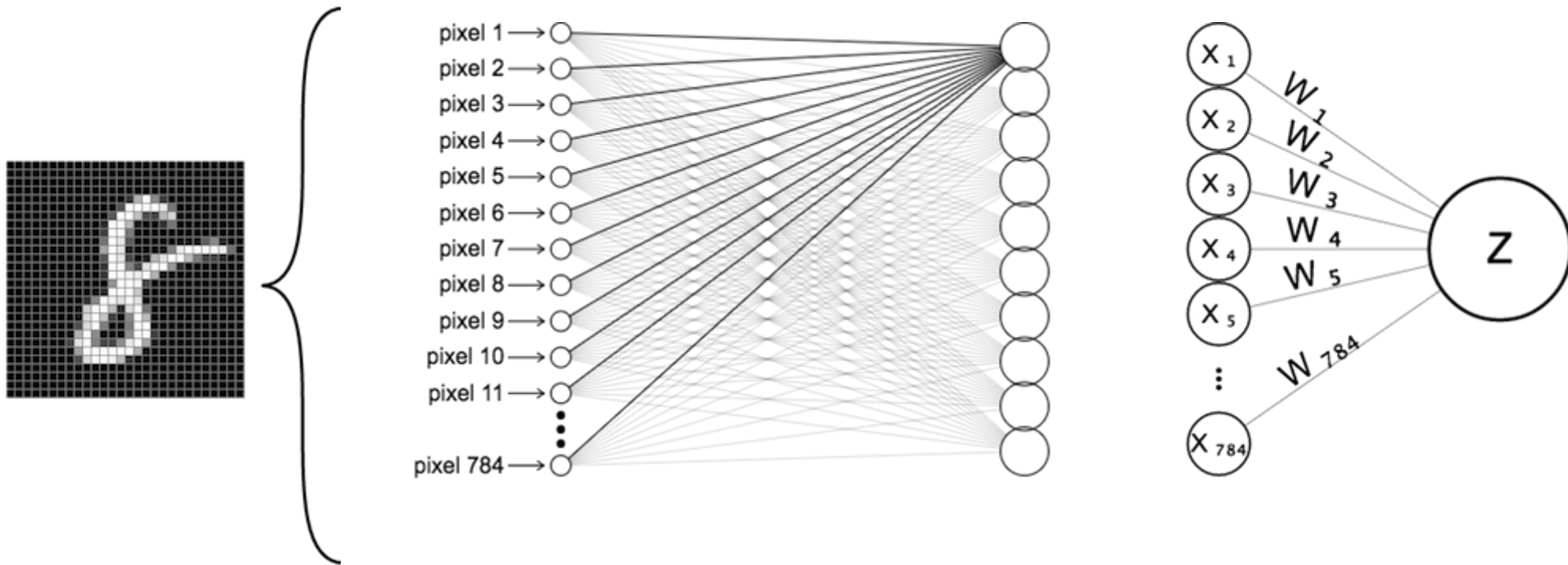


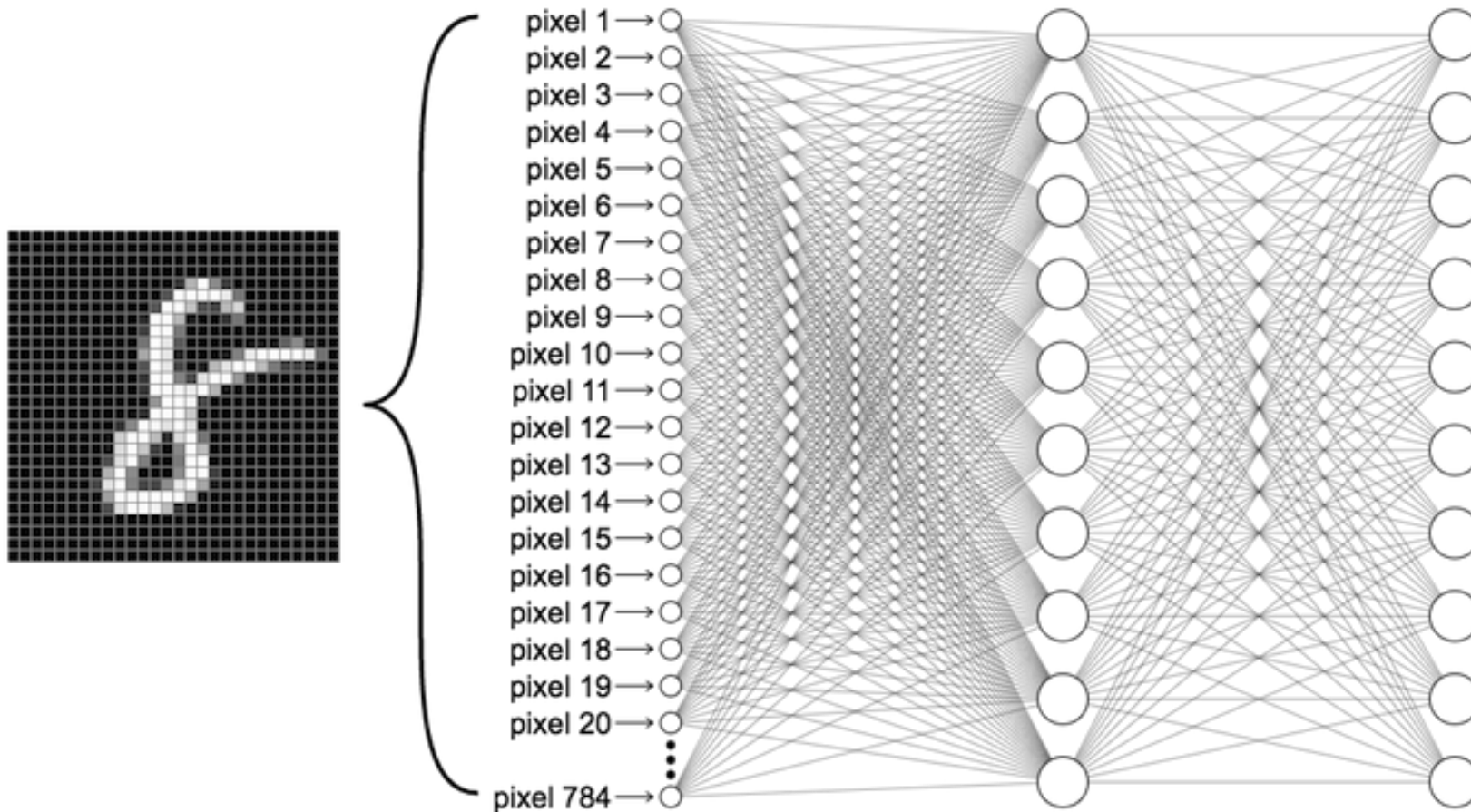
그림 출처: https://ml4a.github.io/ml4a/ko/looking_inside_neural_nets/



06. 인공지능신경망

■ 신경망의 구성:

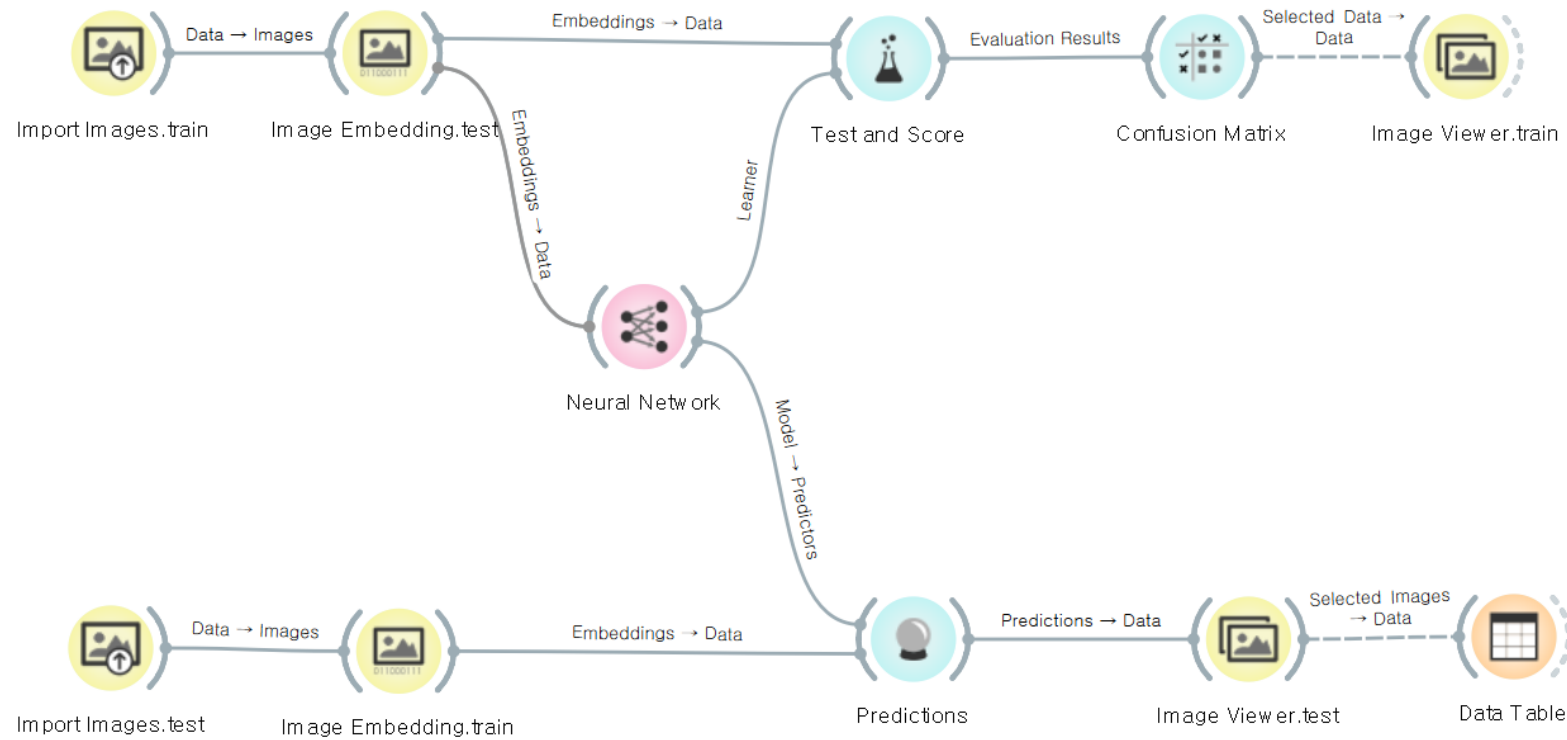
- DNN을 구성하는 각종 파라미터 설정: 뉴런의 수, 활성화 함수, 학습 방법 등





06. 인공지능망

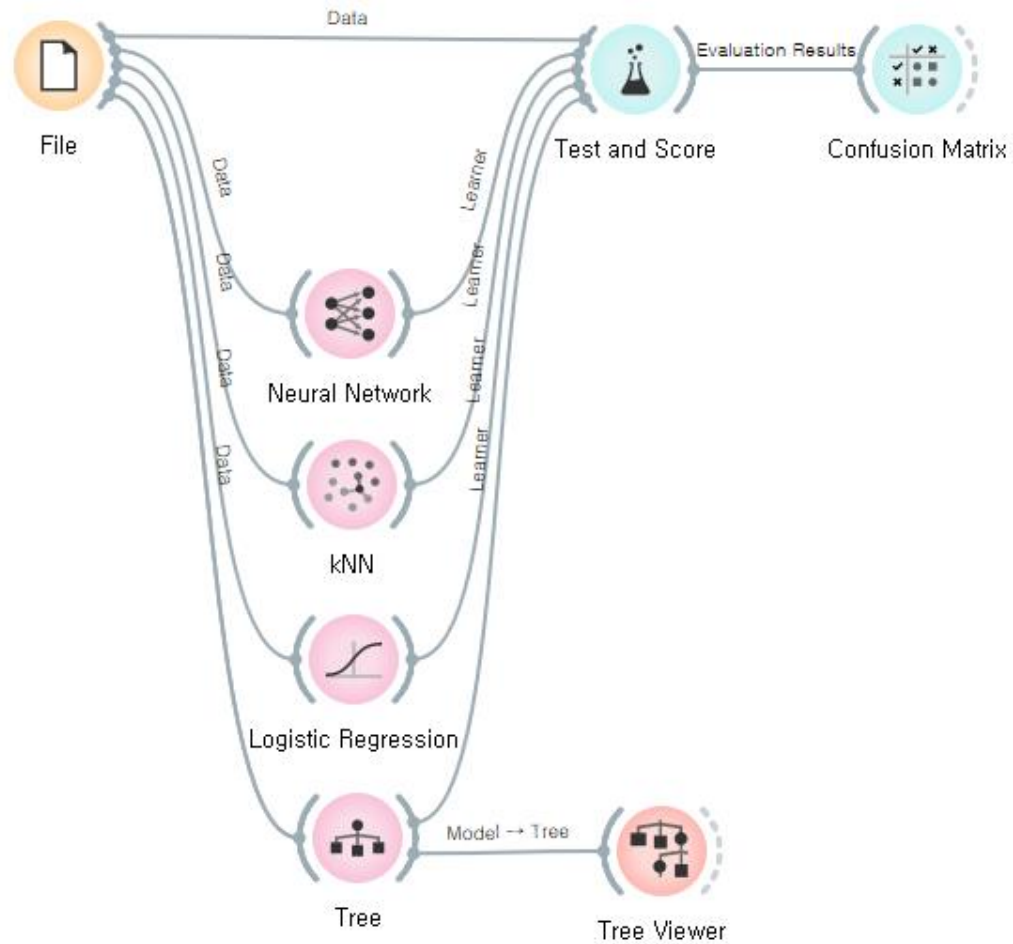
■ Neural Network으로 필기체 인식:





06. 인공지능망

■ 다양한 분류기의 성능 비교:





06. 인공지능망

Test and Score

Sampling

☒ Cross validation
 Number of folds: 20
☐ Stratified

☐ Cross validation by feature

☐ Random sampling
 Repeat train/test: 10
 Training set size: 25 %
☐ Stratified

☐ Leave one out
☐ Test on train data
☐ Test on test data

Target Class

Model Comparison

☐ Negligible difference: 0.1

Evaluation Results

Model	AUC	CA	F1	Precision	Recall
kNN	0.992	0.967	0.967	0.968	0.967
Tree	0.961	0.933	0.934	0.936	0.933
Neural Network	0.983	0.953	0.953	0.953	0.953
Logistic Regression	0.987	0.960	0.960	0.960	0.960

Model Comparison by AUC

	kNN	Tree	Neural Net...	Logistic Re...
kNN		0.946	0.607	0.547
Tree	0.054		0.049	0.028
Neural Network	0.393	0.951		0.448
Logistic Regression	0.453	0.972	0.552	

Table shows probabilities that the score for the model in the row is higher than that of the model in the column. Small numbers show the probability that the difference is negligible.

? | 150 | - | 150 | 4x150

Any Questions?

