
딥러닝
박경규

딥러닝

인공신경망(Artificial Neural Network, ANN)

심층신경망(Deep Neural Network, DNN)

가중치(Weight)

활성화 함수(Activation Function)

손실함수(Loss Function)

딥러닝 학습방법(Deep Learning)

순전파(Forward Propagation)

오차역전파(Error Back Propagation)

경사하강법(Gradient Descent)

옵티마이저(Optimization Algorithm)

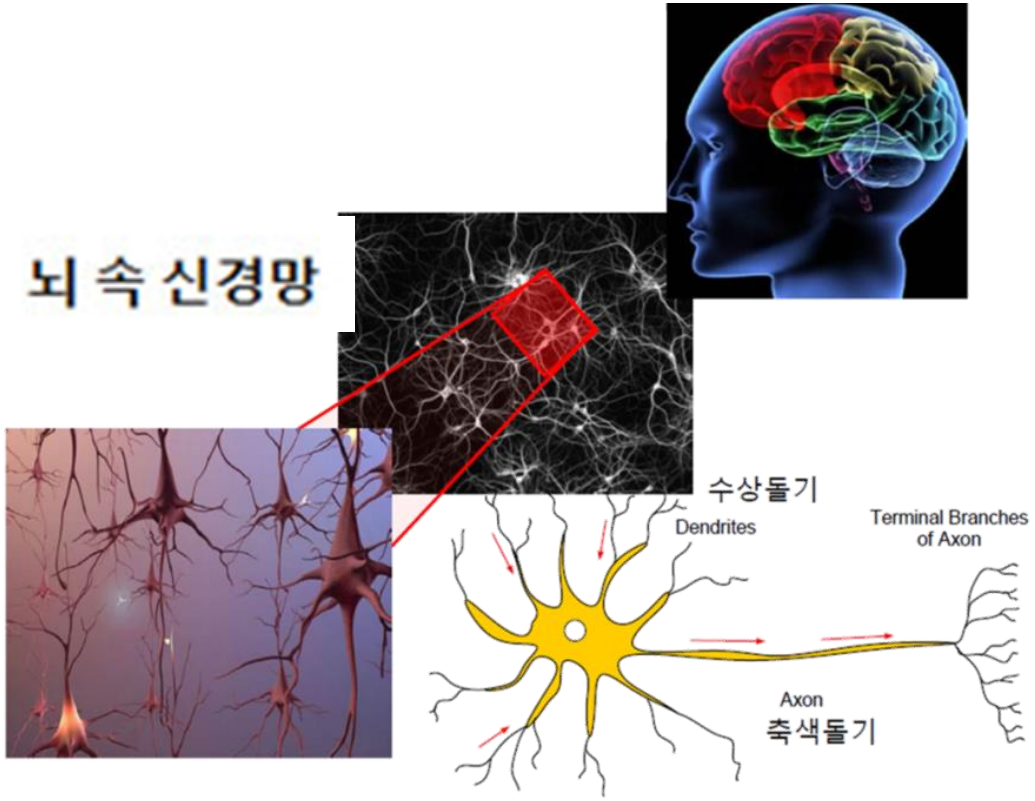
드롭아웃(Dropout)

미니배치(Mini Batch)

텐서(Tensor)

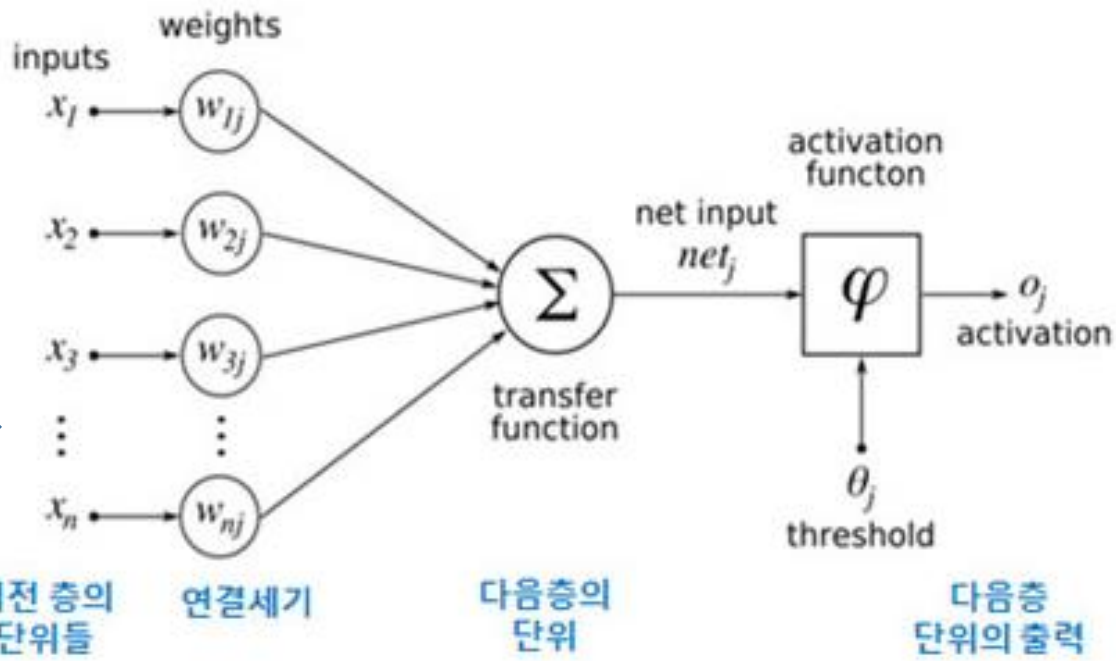
인공신경망(Artificial Neural Network, ANN)

- 뇌신경은 수많은 신경세포(뉴런, neuron)들이 연결되어 정보를 처리하고 전달합니다.
- 인공신경망은 뇌 신경계의 정보처리 구조를 모방하여 만든 계산 알고리즘입니다.
- 뇌 신경계와 같이 수많은 계산 함수를 연결하여 복잡한 정보를 처리하는 네트워크 구조입니다.

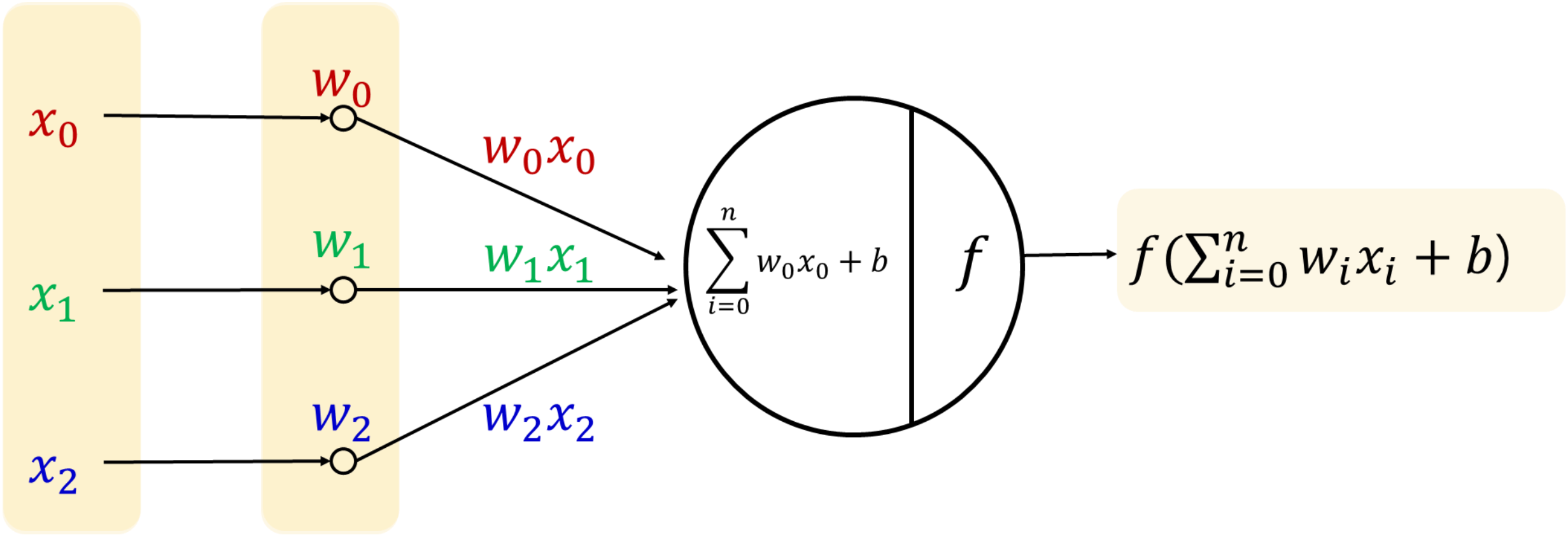


생물학적 뉴런(Neuron)

인공 뉴런(Neuron)

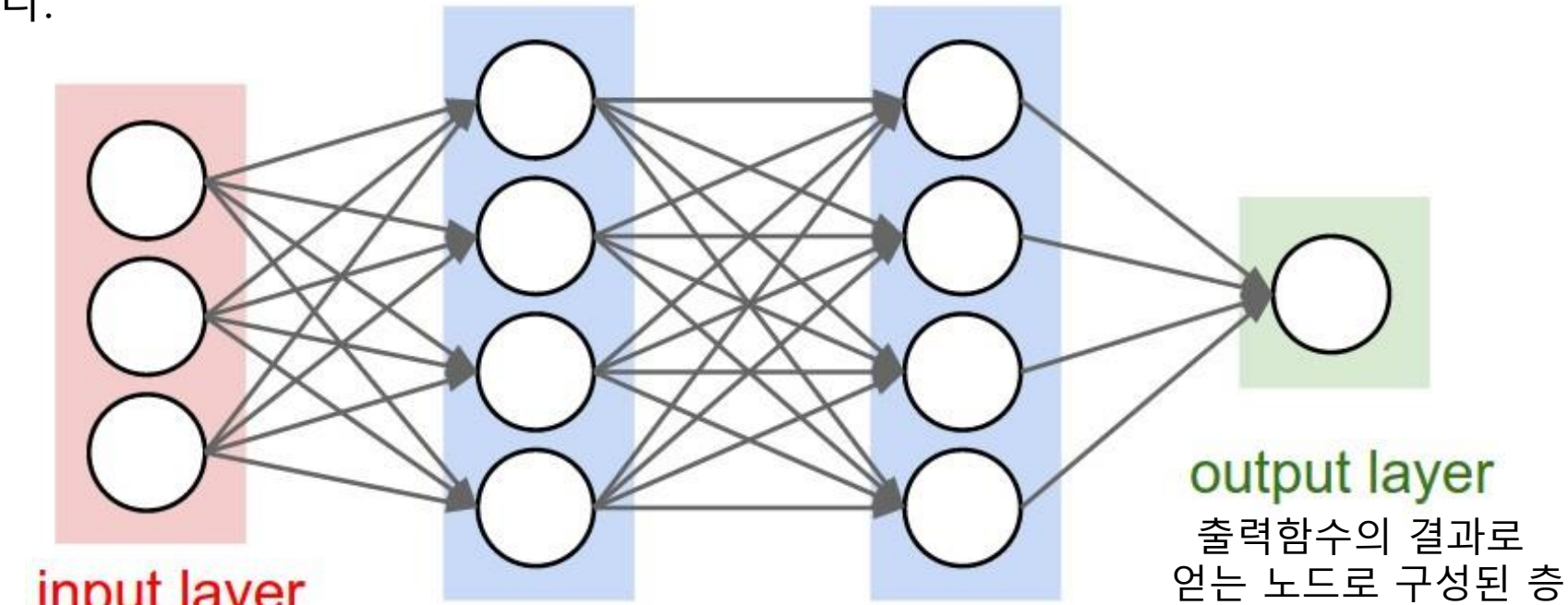


인공 뉴런 모델의 수학적 표현



심층신경망(Deep Neural Network, DNN)

- 딥러닝은 여러 층(layer)을 가진 인공신경망(Artificial Neural Network)을 사용하여 학습을 수행하는 것입니다.
- 심층신경망은 입력층과 출력층사이에 다수의 은닉층(hidden layer)을 포함하는 인공신경망입니다.
- 머신러닝에서는 비선형 분류를 위해 여러 trick을 사용하지만, DNN은 다수의 은닉층으로 비선형 분류가 가능해집니다.



심층신경망(Deep Neural Network, DNN)

■ input layer(입력층)

신경망의 첫 번째 레이어로서 입력 데이터를 수신합니다.

■ hidden layer(은닉층)

신경망에서 입력 레이어(특성)와 출력 레이어(예측) 사이에 위치하는 합성 레이어입니다.
신경망에 하나 이상의 히든 레이어가 포함될 수 있습니다.

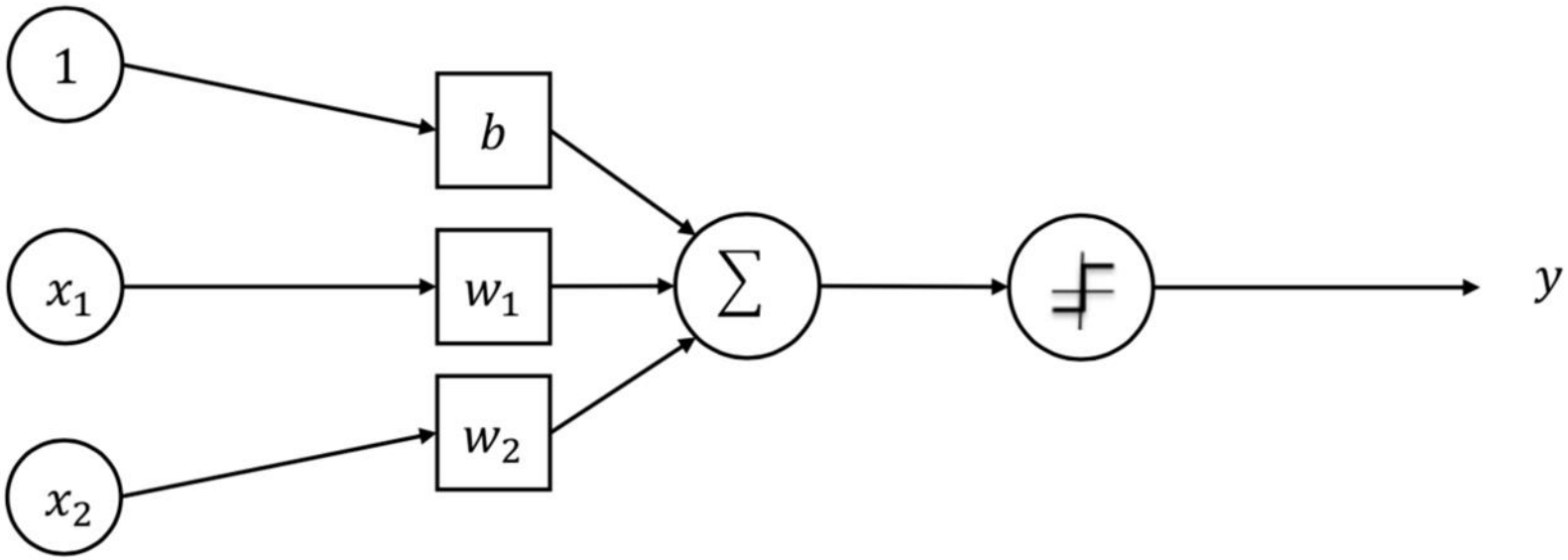
■ output layer(출력층)

신경망의 '최종' 레이어입니다. 이 레이어에 답이 포함됩니다.

■ Dense Layer(밀집층)

Fully Connected Layer(완전 연결층) 이나 Dense Layer(밀집층)라고 불리는 밀집 연결 층(Densely Connected Layer)

뉴런 Neuron

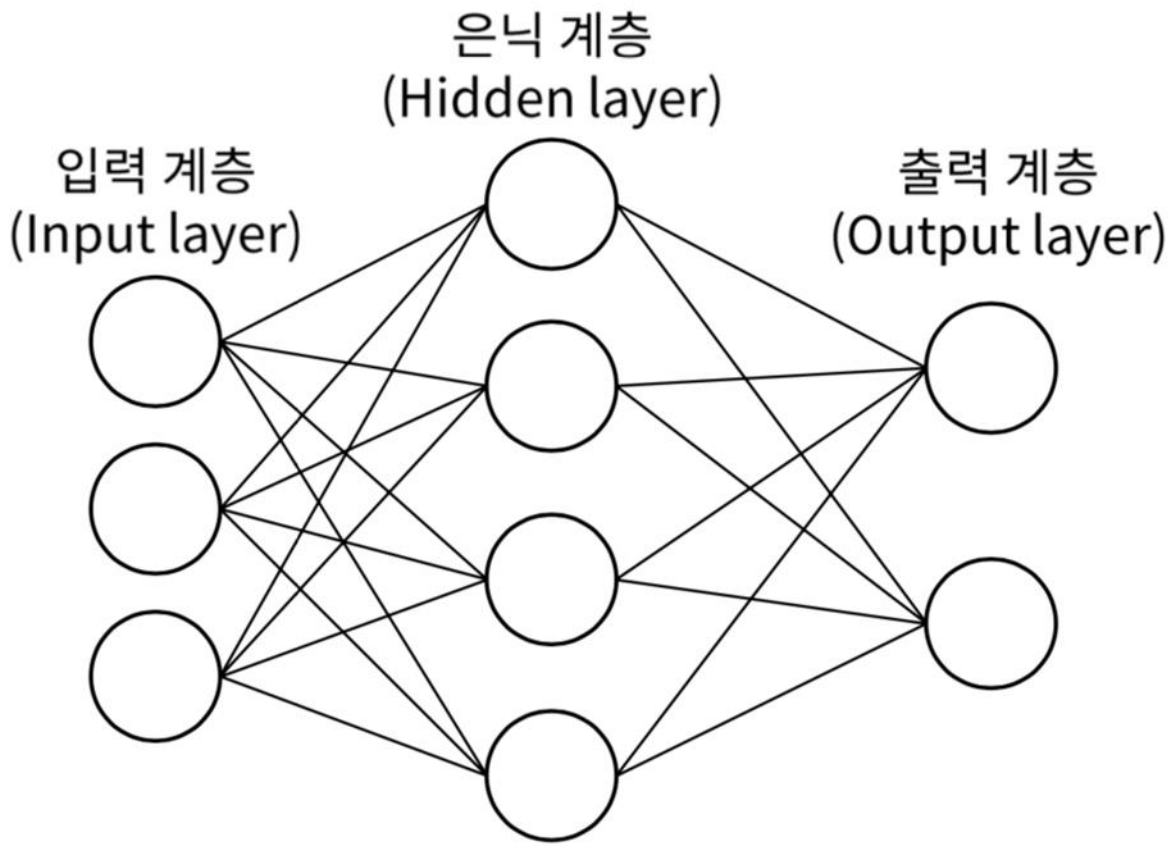


입력 (Input node) | 가중치와 편향 (Weights and bias) | 활성 함수 (Activation function) | 출력 (Output node)

신경망은 뉴런을 기본 단위로 하며, 이를 조합하여 복잡한 구조를 이룬다.

https://heung-bae-lee.github.io/2019/12/08/deep_learning_03/

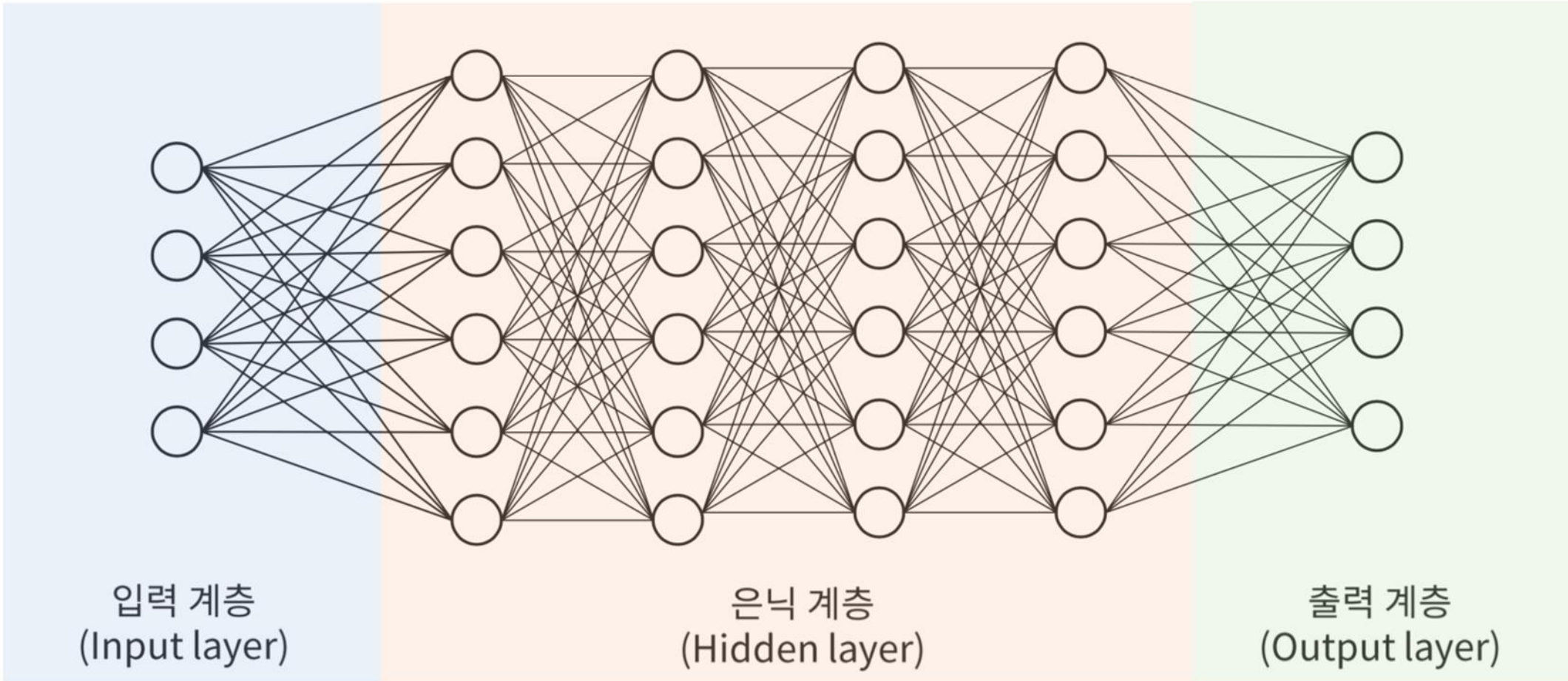
얕은 신경망 Shallow Neural Network



가장 단순하고 얕은(은닉 계층이 1개인) 신경망 구조를 얕은 신경망이라고 한다.

https://heung-bae-lee.github.io/2019/12/08/deep_learning_03/

심층 신경망 Deep Neural Network (DNN)

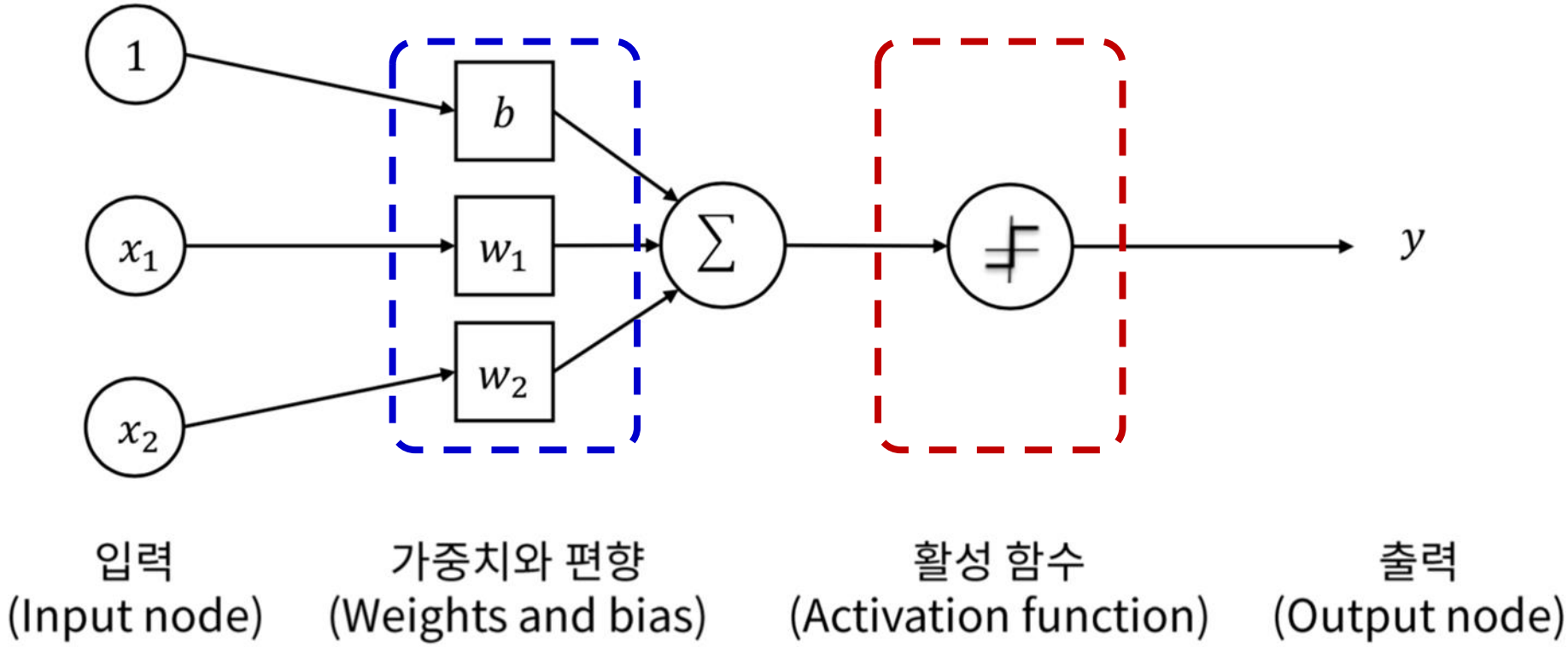


- 얕은 신경망보다 은닉 계층이 많은 신경망을 DNN이라고 부른다.

https://heung-bae-lee.github.io/2019/12/08/deep_learning_03/

가중치(Weight)

가중치는 입력값이 연산결과에 미치는 영향도를 조절하는 요소입니다.

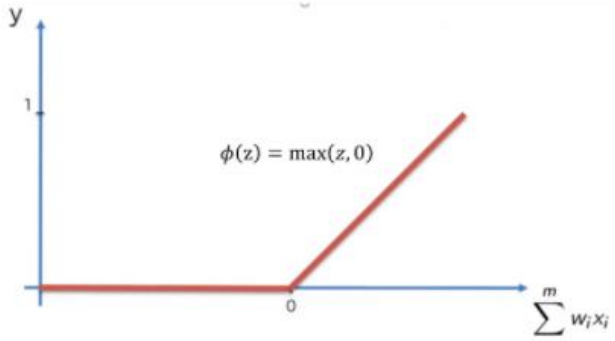


활성화 함수(Activation function)

입력값들의 수학적 선형결합을 다양한 형태의 비선형(또는 선형) 결합으로 변환하는 역할을 합니다.

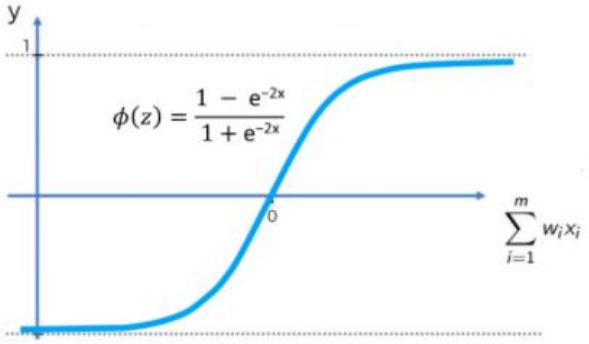
렐루 (ReLU)

입력이 양수일때는 x, 음수일 때는 0을 출력



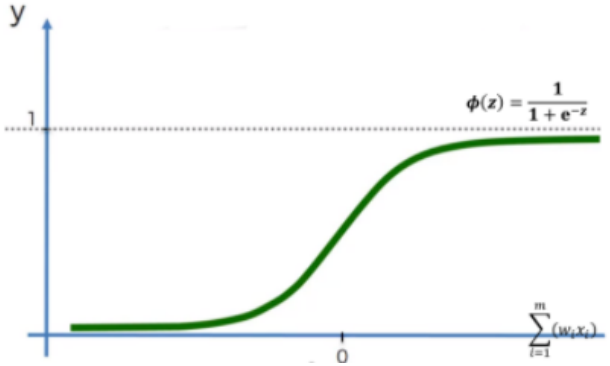
하이퍼볼릭 탄젠트 (Hyperbolic Tangent)

선형함수의 결과를 -1~1까지의 비선형 형태로 변경하는 함수



시그모이드 (Sigmoid)

0~1까지의 비선형 형태로 변경



소프트맥스 (Softmax)

입력값을 0~1 사이 출력이 되도록 정규화, 출력값들의 총합은 항상 1

$$\phi(z) = \frac{e^i}{\sum_{j=0}^{k} e^j} \quad \text{where } i=0,1,\ldots.k$$

https://muzukphysics.tistory.com/165

손실함수(Loss Function)

인공신경망 학습의 목적함수로 출력값(예측값)과 정답(실제값)의 차이를 계산합니다.

■ 회귀예측(Regression)

평균제곱오차 : Mean Squared Error

평균절대오차 : Mean Absolute Error

■ 이진분류(Binary Classification)

Binary Cross Entropy

■ 다중분류(Multi-class Classification)

Categorical Cross Entropy : label(target, 출력값)이 **원핫 벡터(One-Hot Vector)**

Sparce Categorical Cross Entropy : label(target , 출력값)이 정수(0, 1, 2, 3 또는 n)

원핫벡터(One-Hot Vector, One-Hot Encoding)

고유 값에 해당하는 칼럼에만 1을 표시하고 나머지 칼럼에는 0을 표시하는 방법입니다.

Human-Readable

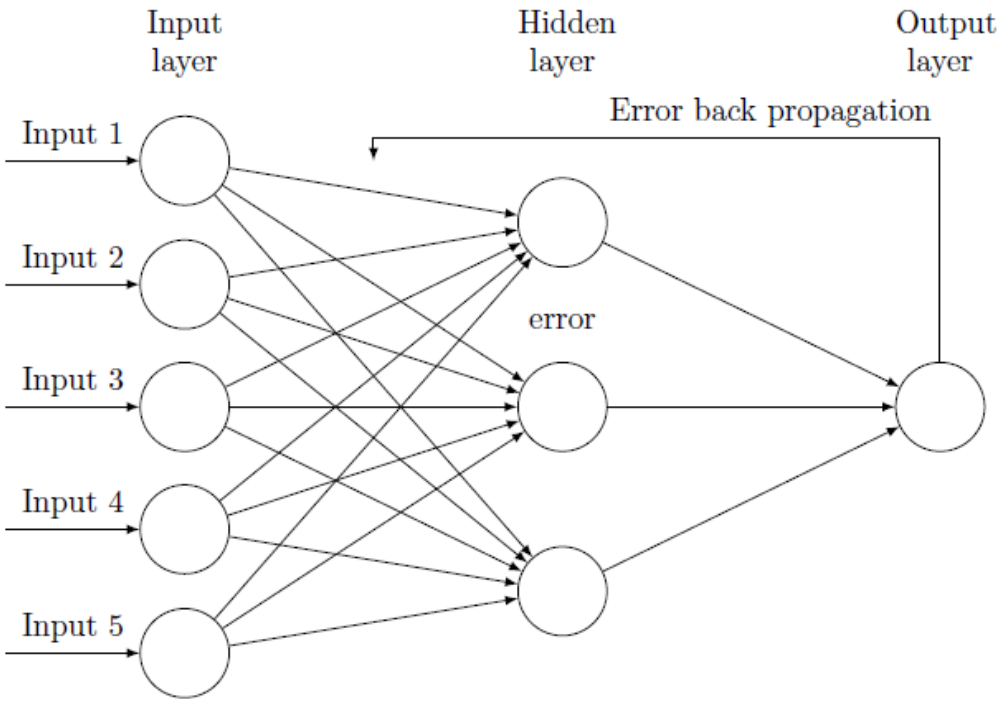
Pet
Cat
Dog
Turtle
Fish
Cat

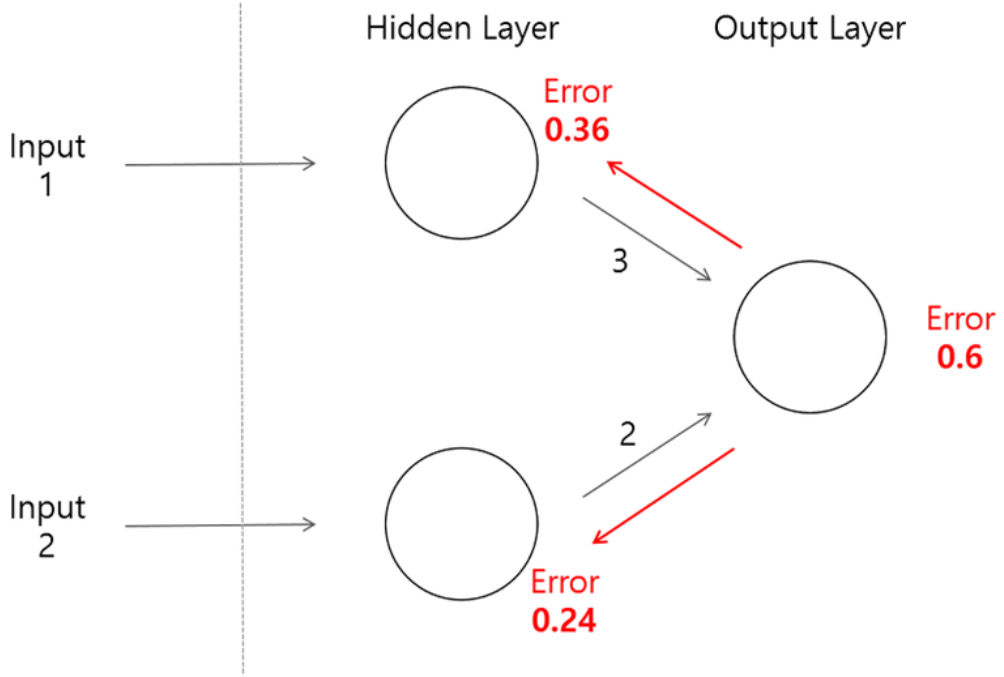
Machine-Readable

Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

딥러닝 학습방법

- 딥러닝 학습의 목표는 모델에 입력값을 넣었을 때의 출력값이 최대한 정답과 일치하게 하는 것입니다.
- 딥러닝 학습은 손실(Loss, Error)를 최소화 하는 인공신경망의 가중치(weight)와 편향(bias)을 찾는 과정입니다.
- 딥러닝 모델의 매개변수(weight, bias)를 무작위로 부여한 후, 반복학습(순전파-오차역전파)을 통해 모델의 출력값을 정답과 가깝게 되도록 매개변수(weight, bias)를 조금씩 조정합니다.
- 딥러닝 학습은 순전파(Forward Propagation)와 오차역전파(Error Back Propagation)의 반복으로 진행이 됩니다.

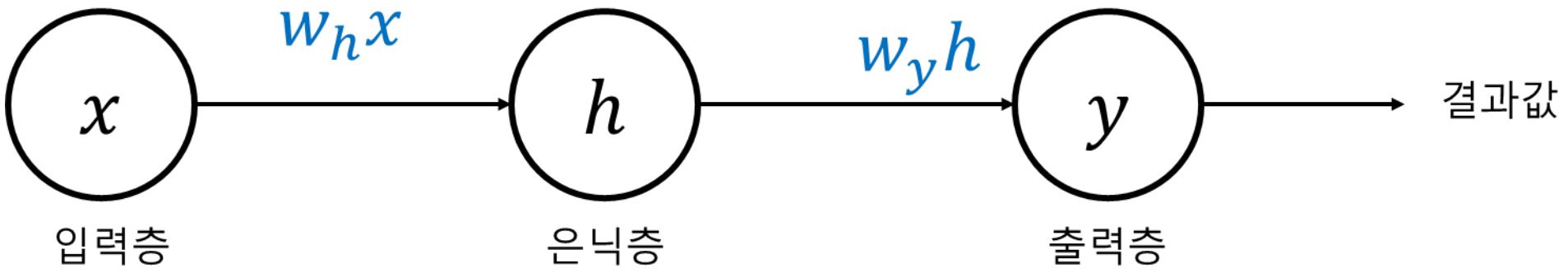




딥러닝 학습방법

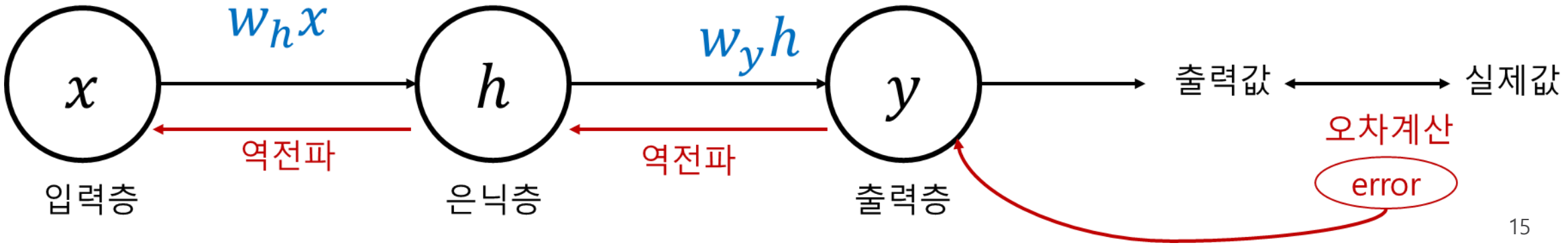
■ 순전파(Forward Propagation)

딥러닝 모델에 값을 입력해서 출력을 얻는 과정입니다.
순전파는 뉴럴 네트워크의 입력층부터 출력층까지 순서대로 변수들을 계산하고 저장하는 것입니다.



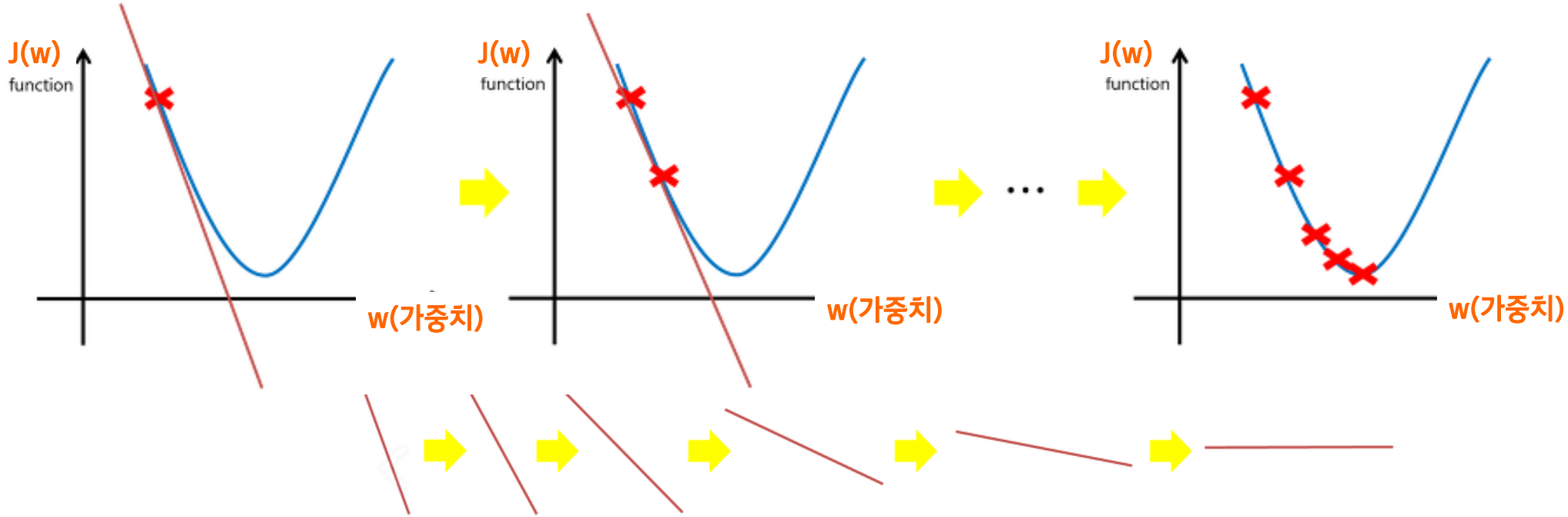
■ 오차역전파(Error Backpropagation)

실제값과 모델 결과값에서 오차를 구해서, 오차를 입력(input) 방향으로 보내서 가중치를 재업데이트 하는 과정입니다.



경사하강법(Gradient Descent)

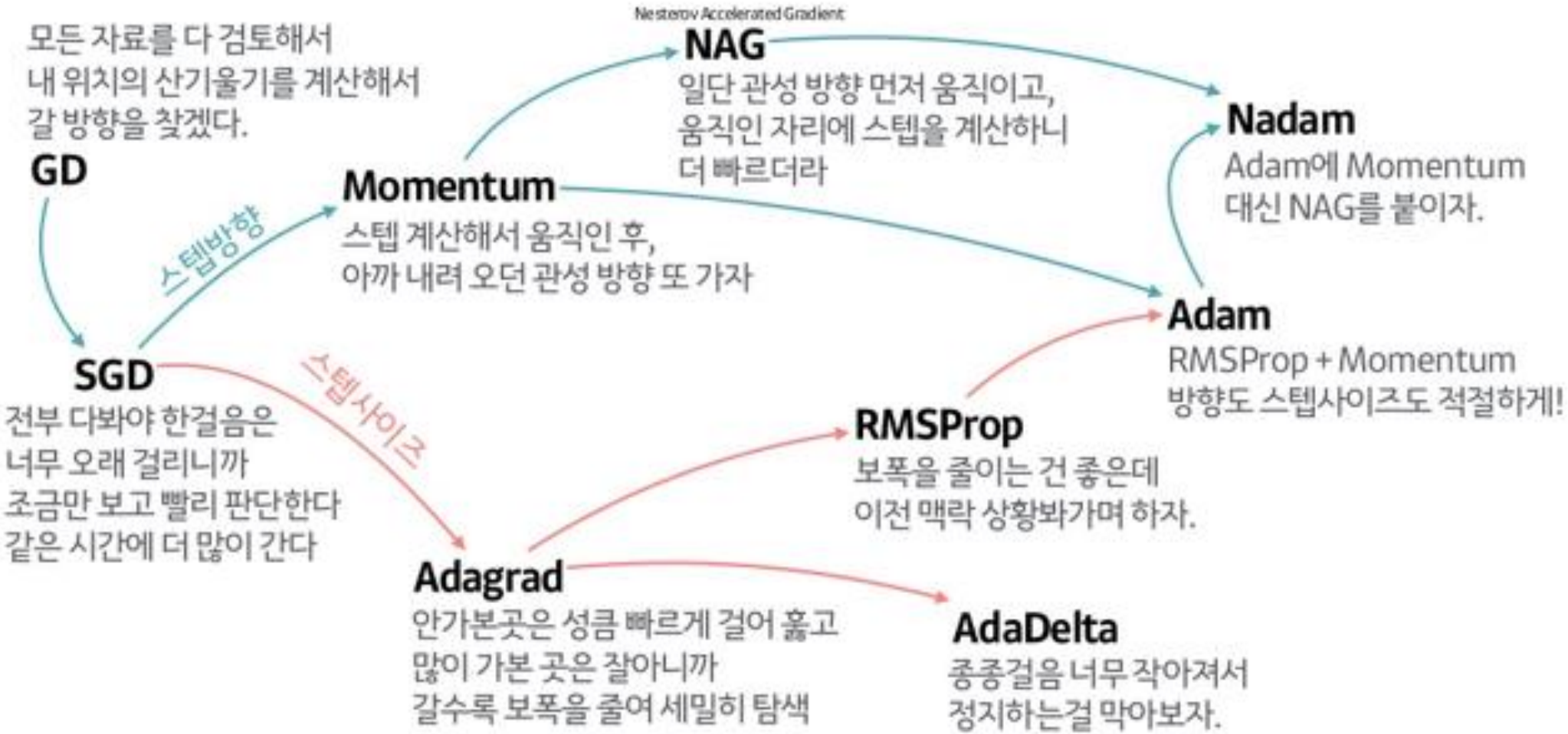
- 손실함수 J(w)는 가중치(w)의 함수로, 볼록함수 형태라면 미분으로 손실이 가장 작은 가중치를 찾을 수 있습니다.
- 하지만, 딥러닝에서는 손실함수가 복잡하고 계산량이 매우 크고, 미분이 0이 되는 값이 여러 개 존재하므로 미분만으로 최소값을 찾기 어려워 경사하강법(Gradient Descent)을 사용합니다.
- 경사하강법은 손실함수의 현 가중치에서 기울기를 구해서 손실(Loss)을 줄이는 방향으로 업데이트 해 나갑니다.



참고 : https://angeloyeo.github.io/2020/08/16/gradient_descent.html
https://angeloyeo.github.io/2020/08/16/gradient_descent.html

옵티마이저(Optimization Algorithm)

손실함수를 최소화하는 방향으로 가중치를 갱신하는 알고리즘입니다.



https://www.slideshare.net/yongho/ss-79607172

옵티마이저(Optimization Algorithm)

■ 손실함수(Loss Function)

손실함수는 신경망 학습의 목적으로(목적함수) 모델의 출력값(예측값)과 정답(실제값)의 차이를 계산합니다.

■ 최적화(Optimization)

딥러닝 모델의 매개변수(weight, bias)를 조절해서 손실함수의 값을 최저로 만드는 과정으로
경사하강법(Gradient Descent)이 대표적입니다.

■ 경사하강법(gradient descent)

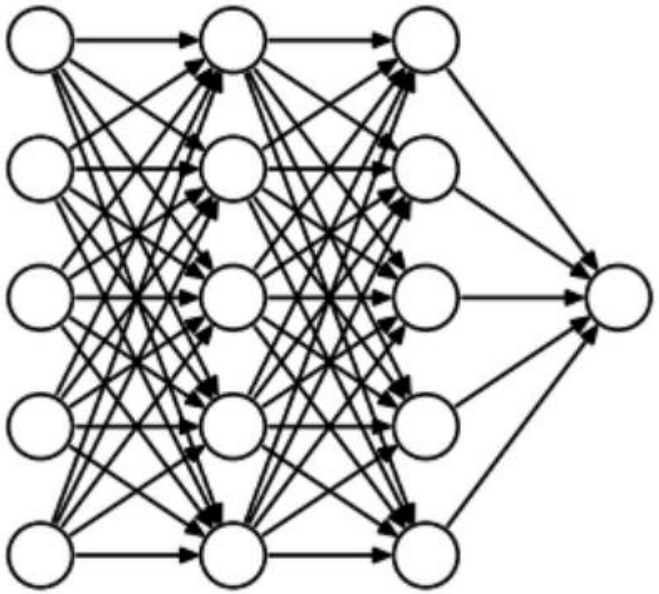
학습 데이터의 조건에 따라 모델의 매개변수를 기준으로 손실의 경사를 계산하여 손실을 최소화하는 기법입니다.
쉽게 설명하면, 경사하강법은 매개변수를 반복적으로 조정하면서 손실을 최소화하는 가중치와 편향의
가장 적절한 조합을 점진적으로 찾는 방식입니다.

■ 경사(gradient)

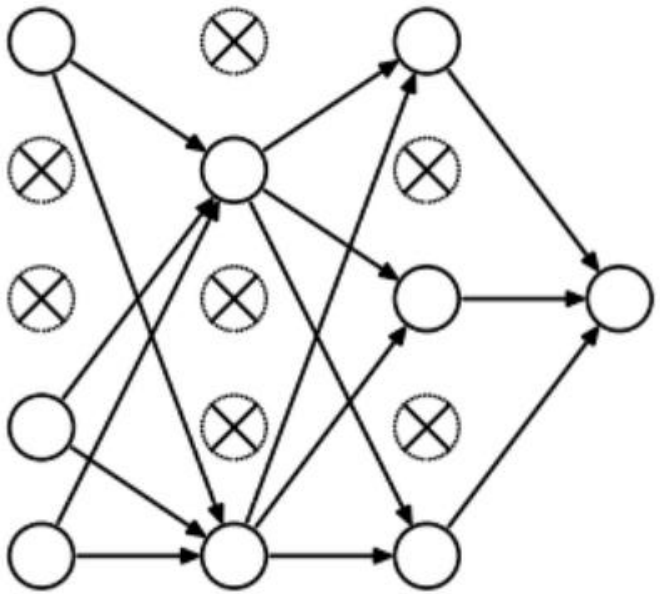
모든 독립 변수를 기준으로 한 편미분의 벡터입니다. 머신러닝에서 경사는 모델 함수의 편미분의 벡터입니다.

드롭아웃(Dropout)

Hidden Layer의 일부 유닛이 동작하지 않게 하여 overfitting(과적합)을 막는 방법입니다.



일반 심층신경망



Dropout이 적용된 심층신경망

■ 과적합(overfitting)

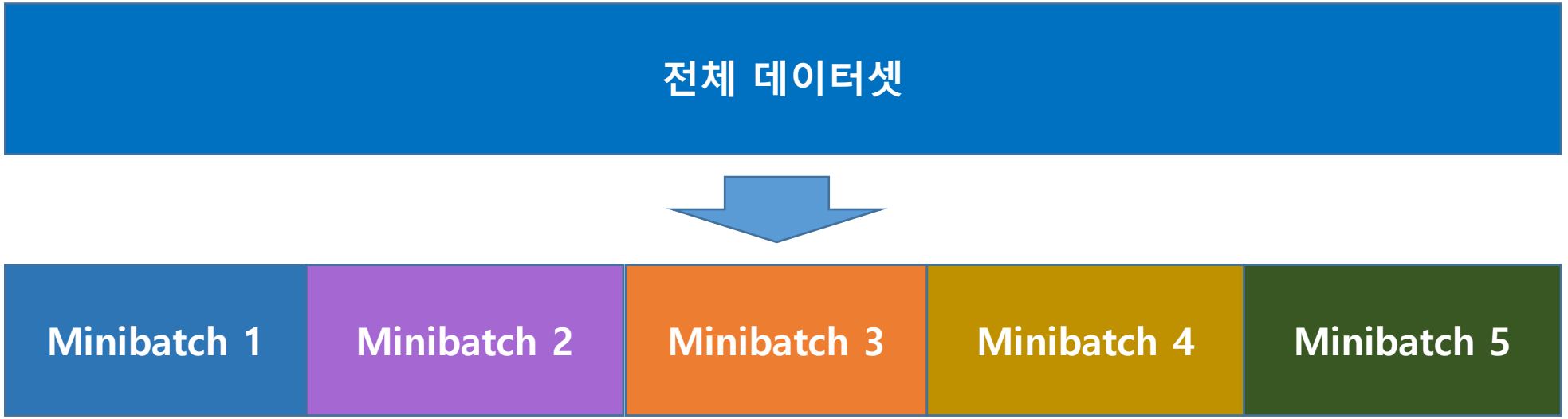
생성된 모델이 학습 데이터와 지나치게 일치하여 새 데이터를 올바르게 예측하지 못하는 경우입니다.

■ 일반화(generalization)

모델학습에 사용된 데이터가 아닌 이전에 접하지 못한 새로운 데이터에 대해 올바른 예측을 수행하는 능력을 의미합니다.

미니배치(Mini Batch)

데이터가 수십만개 이상이라면 전체 데이터를 더 작은 단위로 나누어서 학습하는 방법입니다.



1 Epoch(에폭) : 전체 데이터셋을 한번 학습

1 iteration(이터레이션) : 1회 학습

Minibatch : 데이터셋을 batch size 크기로 나누어서 학습

ex) 총 데이터가 1,000,000개, batch size가 1,000 이면,

1 iteration = 1,000개 데이터에 대해서 학습

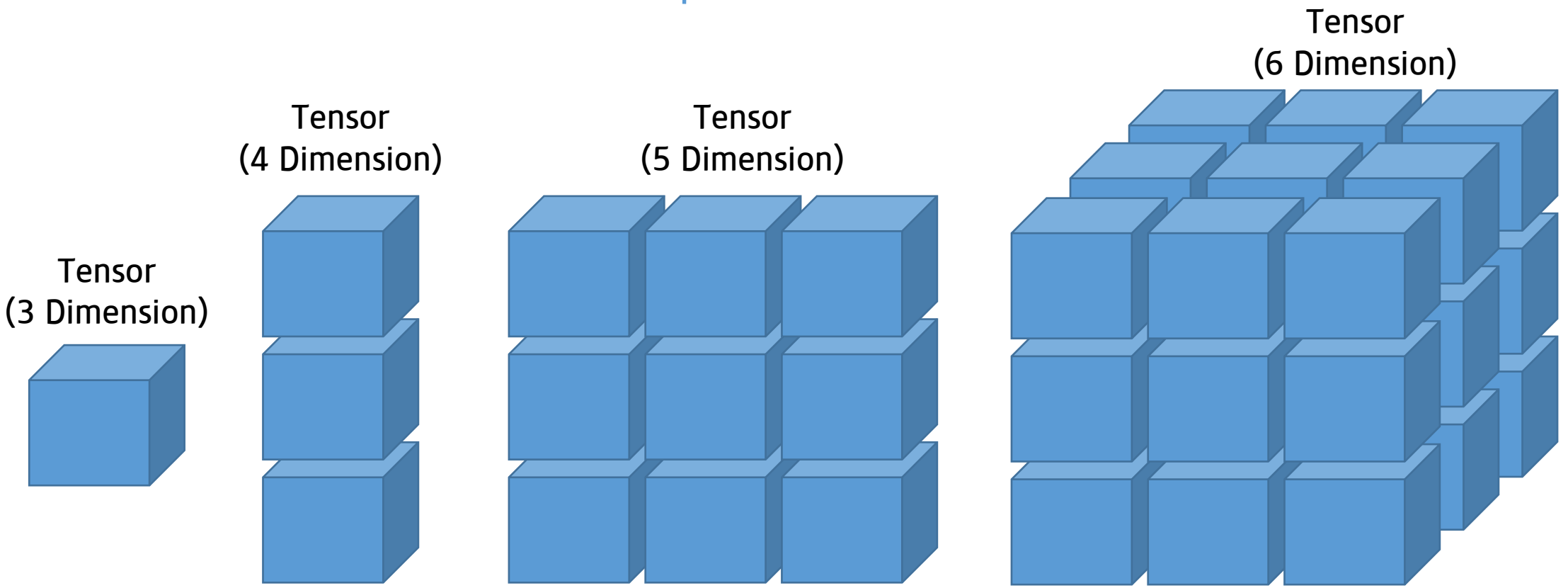
1 Epoch = 1,000,000/batch size = 1,000 iteration

텐서(Tensor)

Scalar
(0 Dimension)

Vector
(1 Dimension)

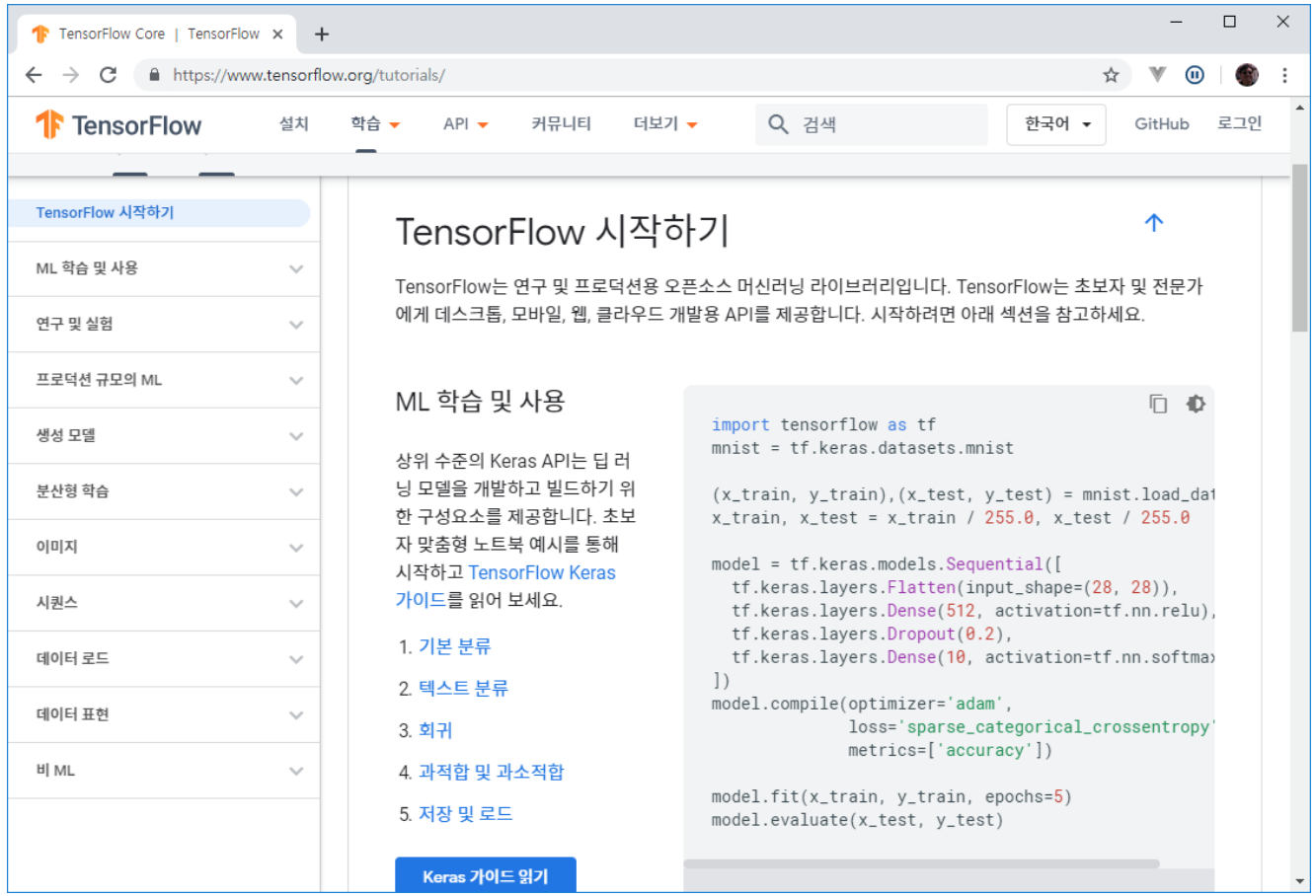
Matrix
(2 Dimension)



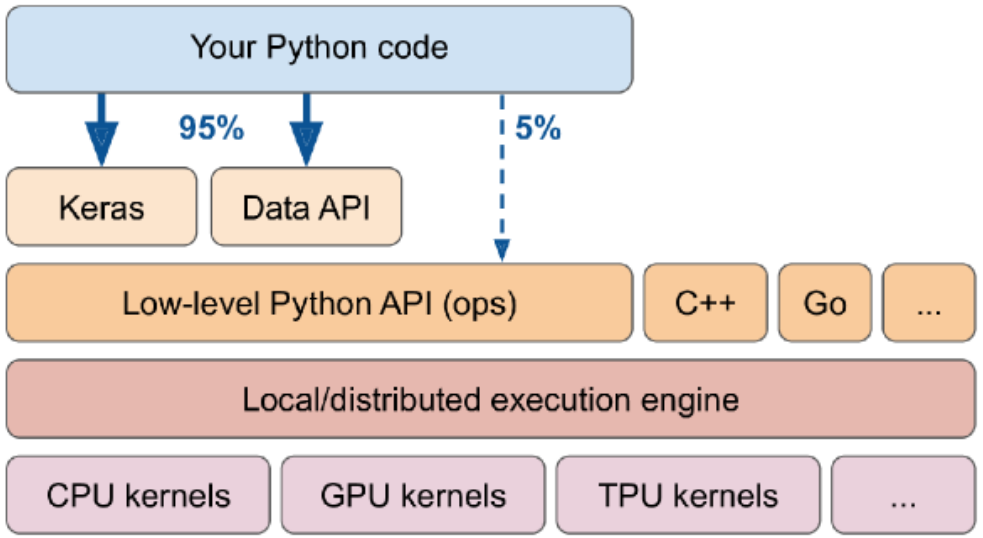
TensorFlow

딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 기능을 제공해주는 프레임워크입니다.

https://www.tensorflow.org/tutorials/



텐서플로 구조



Google TensorFlow

TF2 모델 구성 방법

1. Sequential Model

Sequential Model은 직관적이며 간결하여 딥러닝을 처음 접하는 사용자에게 적합합니다.

■ 방법 1

```
# Passing a list of layers to the constructor
model = Sequential([
    Dense(5, activation='relu', input_shape=(4,)),
    Dense(10, activation='relu'),
    Dense(3, activation='softmax'),
])
```

■ 방법 2

```
# Adding layer via add() method
model = Sequential()
model.add(Dense(5, activation='relu', input_shape=(4,)))
model.add(Dense(10, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

https://bit.ly/3fFiswv

TF2 모델 구성 방법

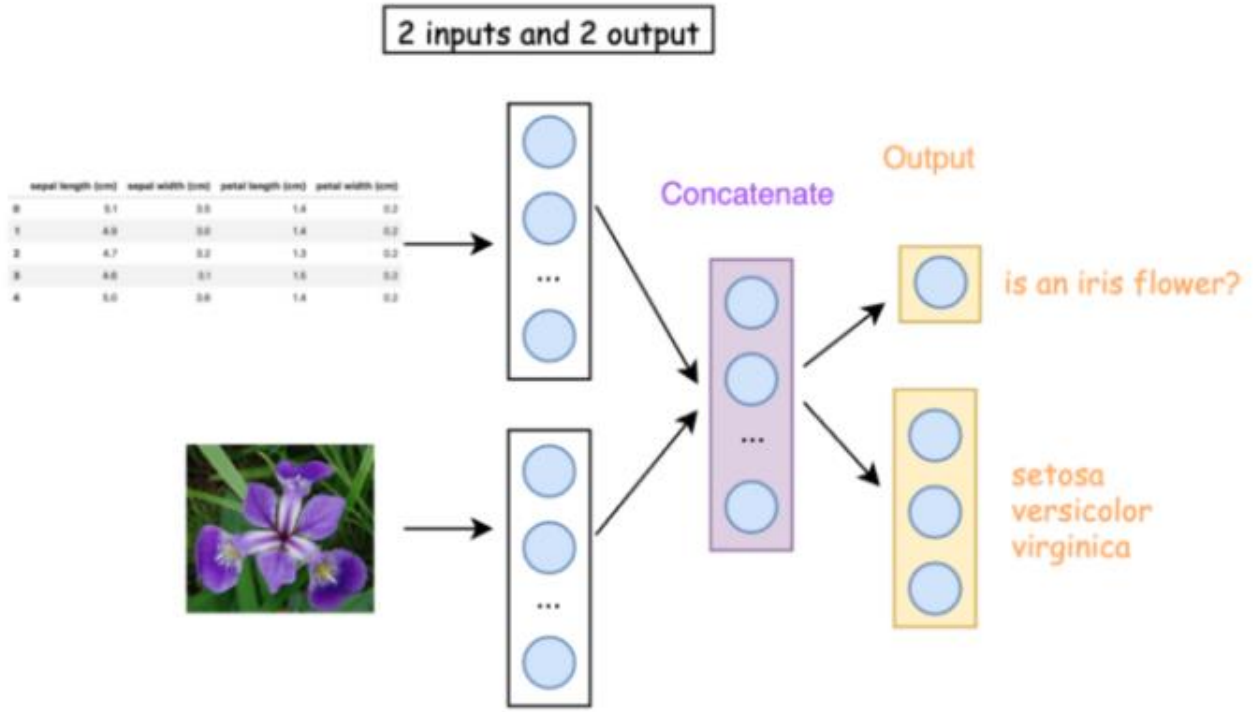
2. Functional API

입력과 다중 출력 등 복잡한 모델을 정의할 수 있습니다.

```
# This returns a tensor
inputs = Input(shape=(4,))


# A layer instance is callable on a tensor, a
x = Dense(5, activation='relu')(inputs)
x = Dense(10, activation='relu')(x)
outputs = Dense(3, activation='softmax')(x)


# This creates a model that includes
# the Input layer and three Dense layers
model = Model(inputs=inputs, outputs=outputs)
```



https://bit.ly/3fFiswv

TF2 모델 구성 방법

3. Model Subclassing

가장 자유롭게 모델을 구축할 수 있는 방법입니다.

```
class CustomModel(Model):

    def __init__(self, **kwargs):
        super(CustomModel, self).__init__(**kwargs)
        self.dense1 = Dense(5, activation='relu', )
        self.dense2 = Dense(10, activation='relu')
        self.dense3 = Dense(3, activation='softmax')

    def call(self, inputs):
        x = self.dense1(inputs)
        x = self.dense2(x)
        return self.dense3(x)


my_custom_model = CustomModel(name='my_custom_model')
```

https://bit.ly/3fFiswv

AI 모델 구현

DNN.ipynb

① 라이브러리 임포트(import)

② 데이터 가져오기(Loading the data)

③ 탐색적 데이터 분석(Exploratory Data Analysis)

④ 데이터 전처리(Data PreProcessing) : 데이터타입 변환, Null 데이터 처리, 누락데이터 처리, 카테고리 데이터, 더미특성 생성, 특성 추출 (feature engineering) 등

⑤ 훈련/테스트 데이터 분할(Train Test Split)

⑥ 데이터 정규화(Normalizing the Data)

⑦ 모델 개발(Creating the Model)

⑧ 모델 성능 평가(Evaluating Model Performance)

AI 모델 학습 데이터

데이터파일 : churn_data.csv

customerID	gender	SeniorCitizen	TotalCharges	Churn
7590-VHVEG	Female	0	29.85	No
5575-GNVDE	Male	0	1889.5	No
3668-QPYBK	Male	0	108.15	Yes
7795-CFOCW	Male	0	1840.75	No
9237-HQITU	Female	0	151.65	Yes
9305-CDSKC	Female	0	820.5	Yes
1452-KIOVK	Male	0	1949.4	No
6713-OKOMC	Female	0	301.9	No

- customerID: 고객ID
- gender: 고객 성별
- SeniorCitizen: 고객이 노약자인가 아닌가
- Partner: 고객에게 파트너가 있는지 여부(결혼 여부)
- Dependents: 고객의 부양 가족 여부
- tenure: 고객이 회사에 머물렀던 개월 수
- PhoneService: 고객에게 전화 서비스가 있는지 여부
- MultipleLines: 고객이 여러 회선을 사용하는지 여부
- InternetService: 고객의 인터넷 서비스 제공업체
- OnlineSecurity: 고객의 온라인 보안 여부
- OnlineBackup: 고객이 온라인 백업을 했는지 여부
- DeviceProtection: 고객에게 기기 보호 기능이 있는지 여부
- TechSupport: 고객이 기술 지원을 받았는지 여부
- StreamingTV: 고객이 스트리밍TV을 가지고 있는지 여부
- StreamingMovies: 고객이 영화를 스트리밍하는지 여부
- Contract: 고객의 계약기간
- PaperlessBilling: 고객의 종이 없는 청구서 수신 여부(모바일 청구서)
- PaymentMethod: 고객의 결제 수단
- MonthlyCharges: 매월 고객에게 청구되는 금액
- TotalCharges: 고객에게 청구된 총 금액
- Churn: 고객 이탈 여부 (label, y)

심층신경망(DNN, Deep Neural Network) 구현

■ 라이브러리 임포트

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
```

■ 데이터 로드

```
df = pd.read_csv('churn_data.csv')
```

심층신경망 구현

■ 데이터 분석

```
df.info()
df.isnull().sum()
df.describe().transpose()
df.corr()['MonthlyCharges'][:-1].sort_values().plot(kind='bar')
sns.pairplot(df)
```

■ 데이터 전처리

```
df.drop('customerID', axis=1, inplace=True)
df['TotalCharges'].replace([' '], ['0'], inplace=True)
df['TotalCharges'] = df['TotalCharges'].astype(float)
df['Churn'].replace(['Yes', 'No'], [1, 0], inplace=True)
```

심층신경망 구현

■ 데이터 전처리

```
cols = ['gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',
'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
'PaperlessBilling', 'PaymentMethod']

dummies = pd.get_dummies(df[cols], drop_first=True)
df = df.drop(cols, axis=1)
df = pd.concat([df, dummies], axis=1)

# df = pd.get_dummies(df)
# cols = list(df.select_dtypes('object').columns)
```

심층신경망 구현

■ 훈련/테스트 데이터셋 분할

```
from sklearn.model_selection import train_test_split

X = df.drop('Churn', axis=1).values

y = df['Churn'].values

X_train, X_test, y_train, y_test = train_test_split(
                                                    X, y,
                                                    test_size=0.3,
                                                    random_state=42)

X_train.shape
[Out] (4930, 30)

y_train.shape
[Out] (4930,)
```

심층신경망 구현

■ 데이터셋 정규화/스케일링(Normalizing/Scaling)

```
from sklearn.preprocessing import MinMaxScaler


scaler = MinMaxScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

심층신경망 구현

■ 모델 구성

```
model = Sequential()

# input layer
model.add(Dense(64, activation='relu', input_shape=(30,)))

# hidden layer
model.add(Dense(64, activation='relu'))

# hidden layer
model.add(Dense(32, activation='relu'))

# output layer
model.add(Dense(1, activation='sigmoid'))
```

이 모델은 회귀모델인가요?
이진분류모델인가요?
다중분류모델인가요?

심층신경망 구현

■ 모델 구성 - 과적합 방지

```
model = Sequential()
# input layer
model.add(Dense(128, activation='relu', input_shape=(30,)))
# hidden layer
model.add(Dropout(0.3))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.2))
# output layer
model.add(Dense(1, activation='sigmoid'))
```

심층신경망 구현

■ 회귀예측

```
model.add(Dense(1))
```

■ 이진분류

```
model.add(Dense(1, activation='sigmoid'))
```

sigmoid : 선형함수의 결과를 0~1까지의 비선형 형태로 변형하기 위한 함수

■ 다중분류

```
model.add(Dense(3, activation='softmax'))
```

softmax : 입력받은 값을 0~1 사이 출력이 되도록 정규화하여 출력값들의 총합이 항상 1 되는 특성을 가진 함수

심층신경망 구현

■ 회귀예측

컴파일 : 모델 학습과정을 설정하는 단계

```
model.compile(optimizer='adam', loss='mse')
```

■ 이진분류

```
model.compile(optimizer='adam', loss='binary_crossentropy',
              metrics=['accuracy'])
```

■ 다중분류

출력값이 one-hot encoding

```
model.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['accuracy'])
```

출력값이 integer(정수)

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

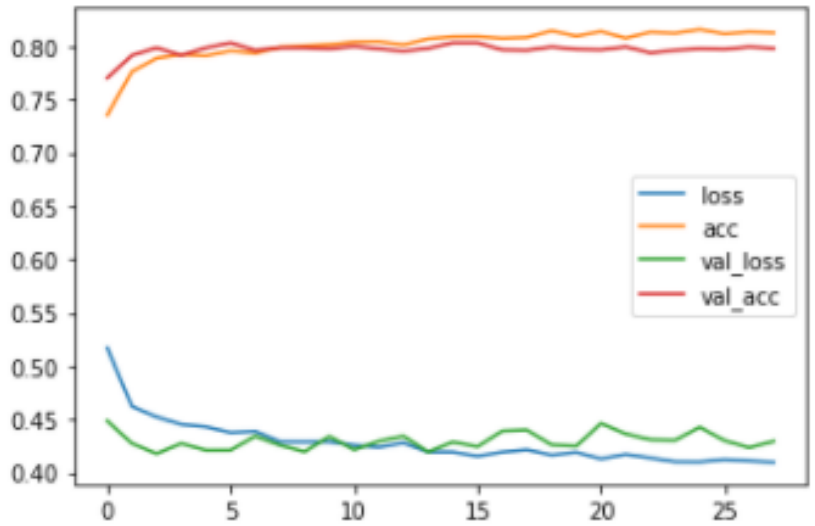
심층신경망 구현

■ 모델 학습(훈련)

```
model.fit(X_train, y_train,
          validation_data=(X_test, y_test), epochs=20, batch_size=10)
```

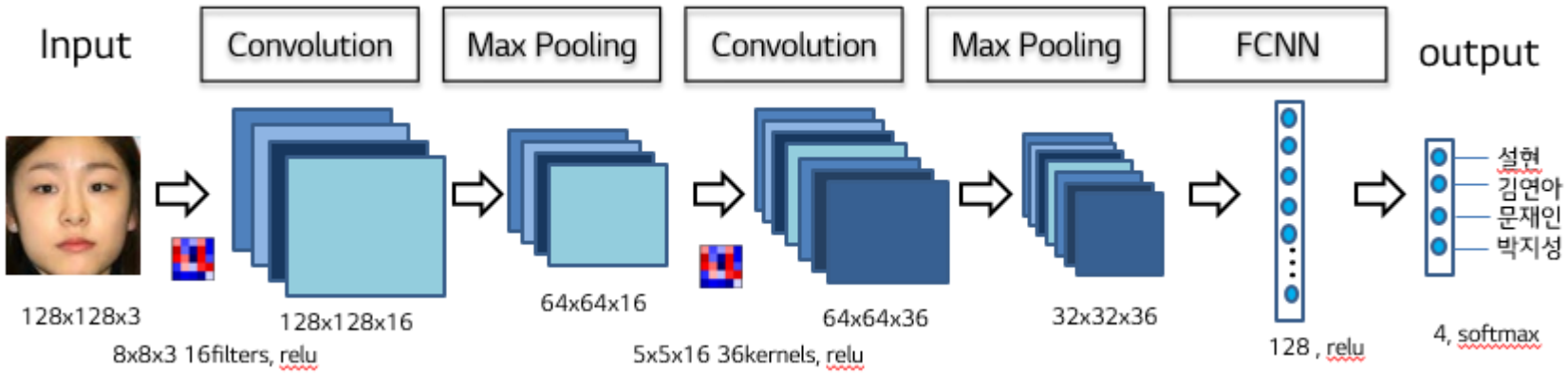
■ 학습과정 시각화

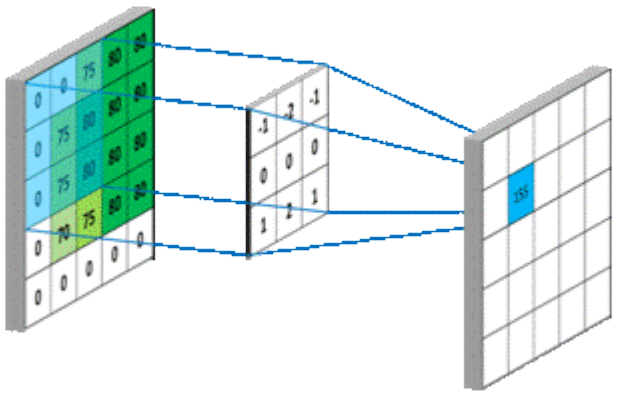
```
losses = pd.DataFrame(model.history.history)
losses[['loss','val_loss']].plot()
```



합성곱 신경망(CNN, Convolutional Neural Network)

사람의 시각 피질 메커니즘에 영감을 받아 설계된 이미지, 영상등을 인식하는 신경망 모델
Convolution층에서는 각 filter가 입력 이미지의 픽셀 전체를 차례로 훑고 지나가며
linear combination을 진행하고 Feature Map을 구성합니다.



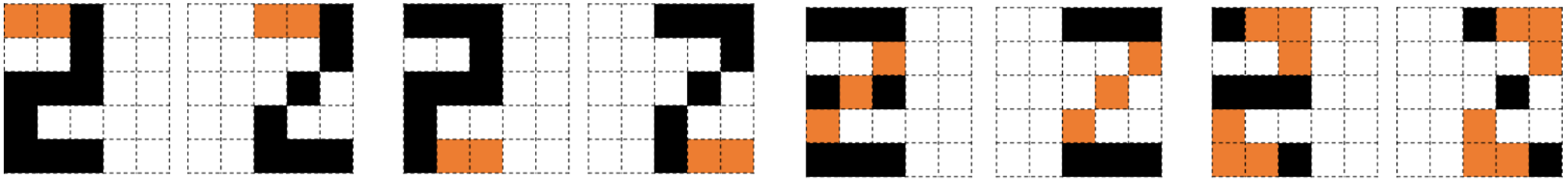


https://wonwooddo.tistory.com/47

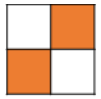
합성곱 신경망

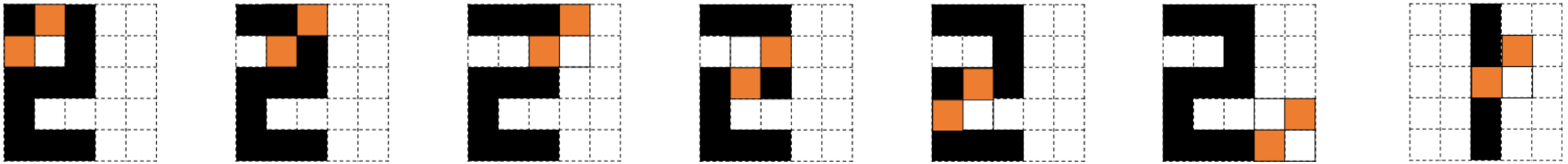
CNN은 뇌가 사물을 구별하듯 생김새 정보로 사물을 학습하고 구별해 낸다.

■ 숫자 2에서 공통적으로 얻을 수 있는 생김새 정보



■ CNN은 어떻게 특징을 찾아 내는가?





대각선 필터는 숫자 2로부터 두 곳의 대각선 특징을 감지하지만, 숫자 1에서는 대각선 특징을 발견하지 못한다.

합성곱 신경망

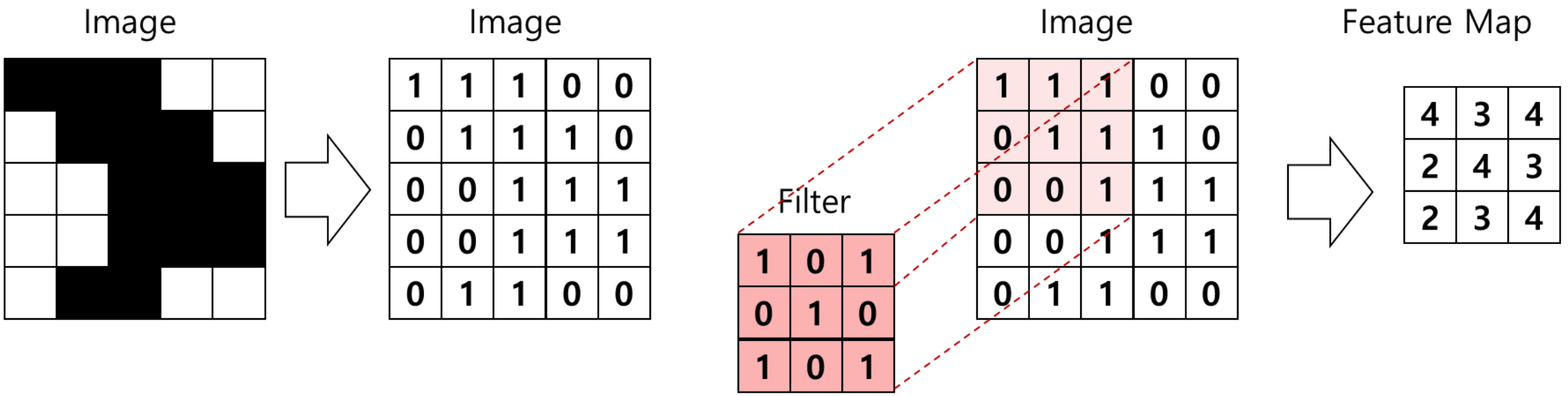


Image
Image
Image
Feature Map
Filter
1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 1
0 1 1 0 0
1 0 1
0 1 0
1 0 1
4 3 4
2 4 3
2 3 4

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

4		

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

4	3	

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

4	3	4

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

4	3	4
2		

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

4	3	4
2	4	

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

4	3	4
2	4	3

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

4	3	4
2	4	3
2		

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

4	3	4
2	4	3
2	3	

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

4	3	4
2	4	3
2	3	4

합성곱 신경망

Padding

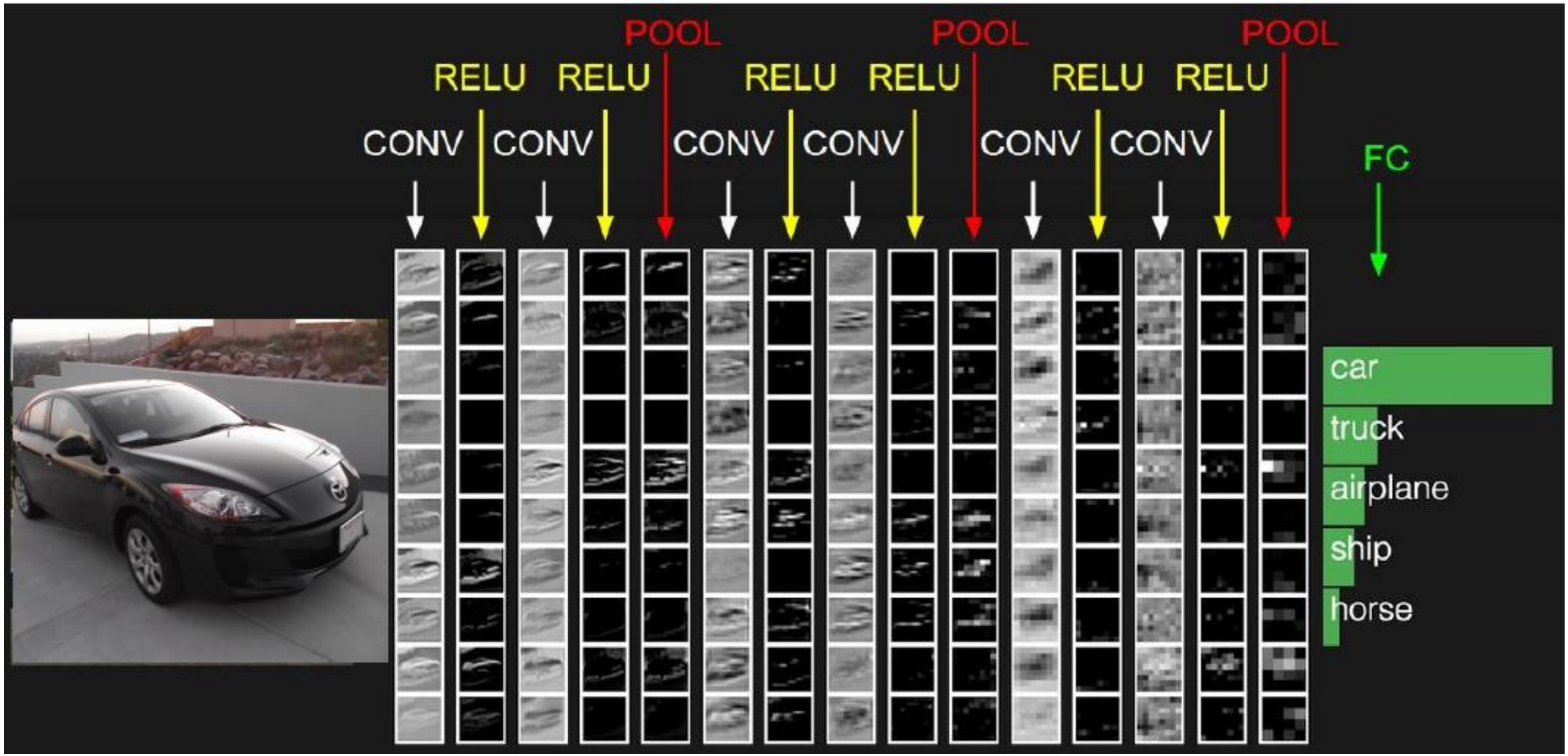
패딩(padding)을 1만큼 적용한 이미지 데이터 행렬

0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	1	1	1	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0

Pooling

1	4	**8**	3
6	**9**	2	1
11	13	6	7
8	**19**	**8**	2

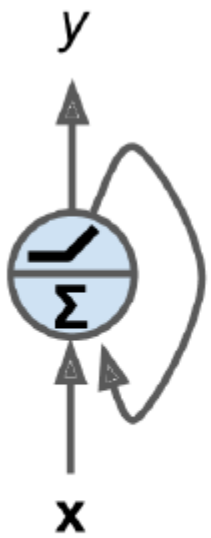
합성곱 신경망


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

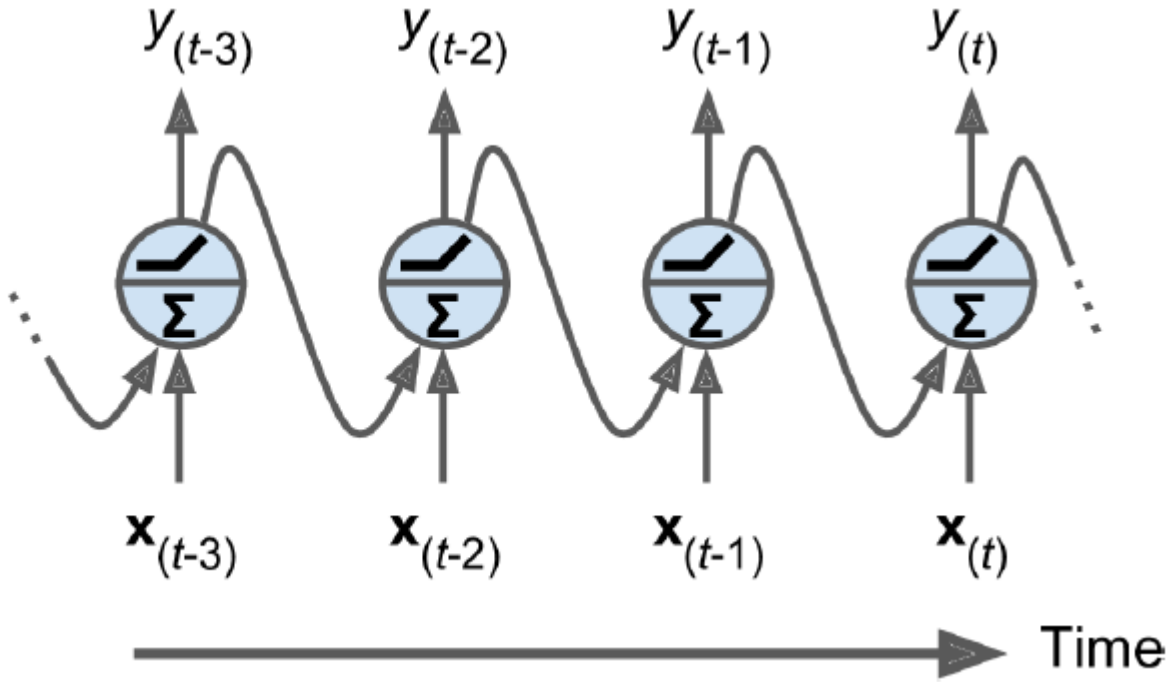
순환 신경망(RNN, Recurrent Neural Network)

순환신경망은 고정 길이 입력이 아닌 임의 길이를 가진 순차데이터(Sequence)를 다룰 수 있습니다.
순환신경망은 시계열 데이터를 분석해서 미래값을 예측하고 문장, 오디오를 입력으로 받아 자동번역,
자연어처리에 유용합니다.

■ 순환뉴런



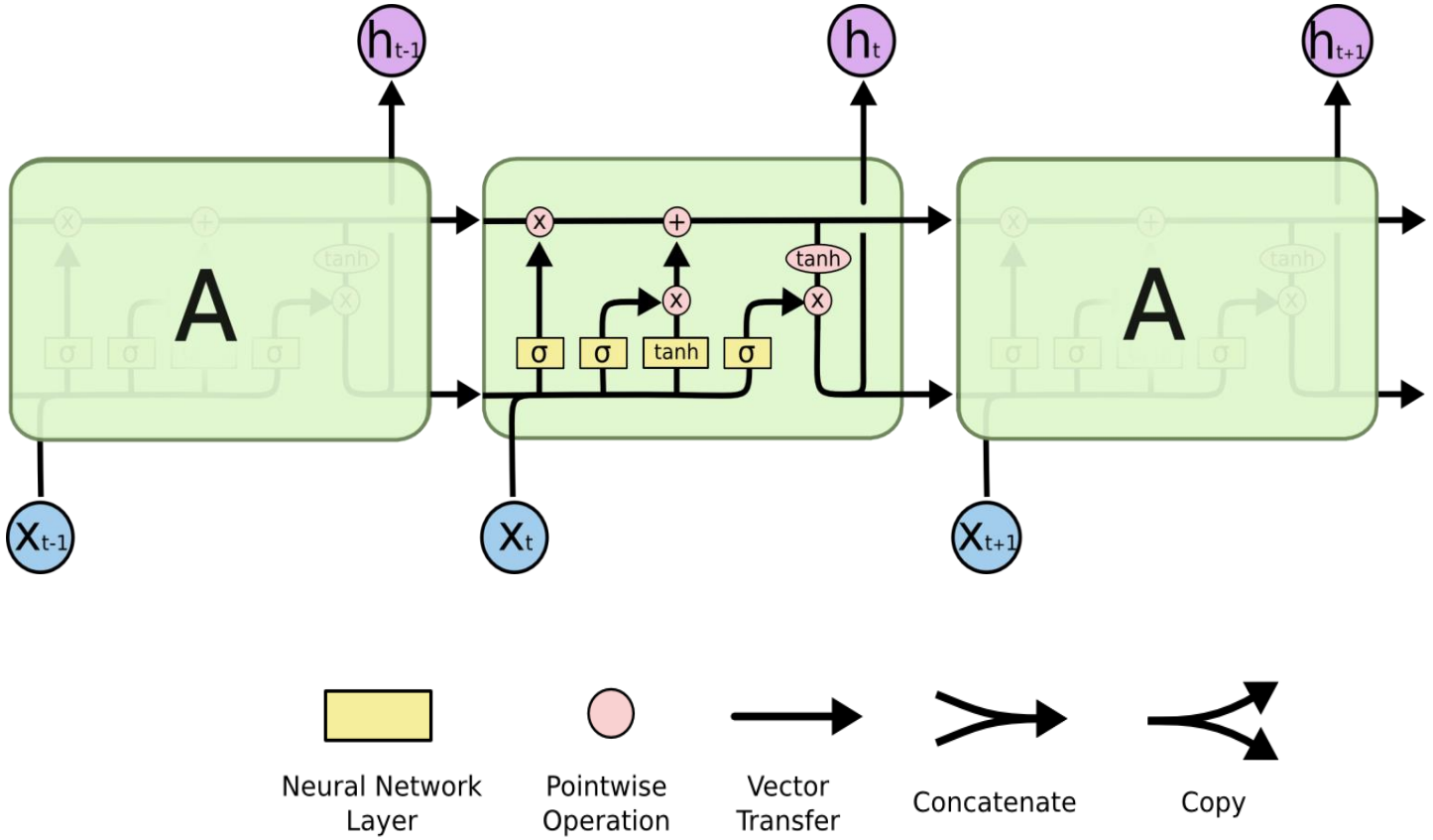
■ 순환뉴런을 타임 스텝으로 펼친 모습



출처 : 도서, 핸즈온 머신러닝 2판

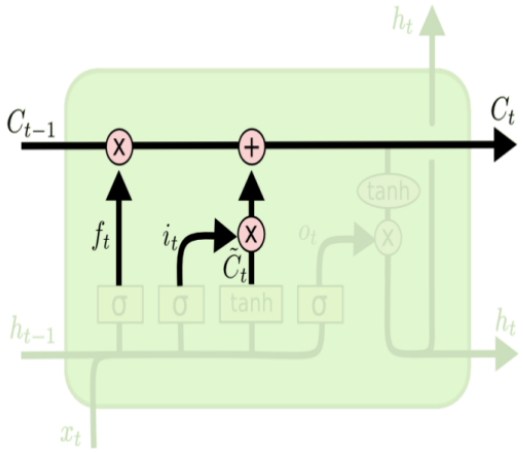
순환 신경망 – LSTM(Long-Short Term Memory)

LSTM 네트워크는 장기적인 종속성을 학습할 수 있는 특수한 종류의 RNN입니다.
LSTM은 Forget gate, Input gate, Output gate를 통한 정보전이 및 전파를 제어합니다.



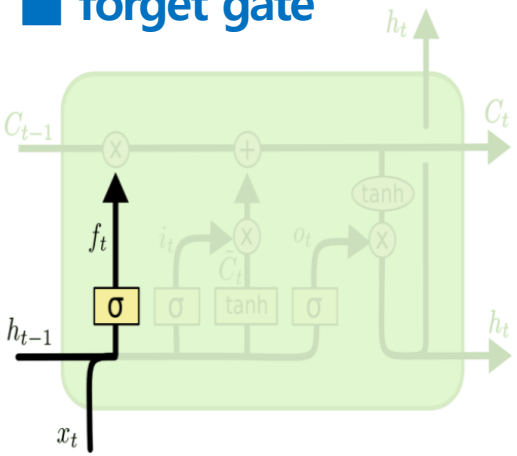
순환 신경망 – LSTM(Long-Short Term Memory)

■ Cell State(장기 상태)



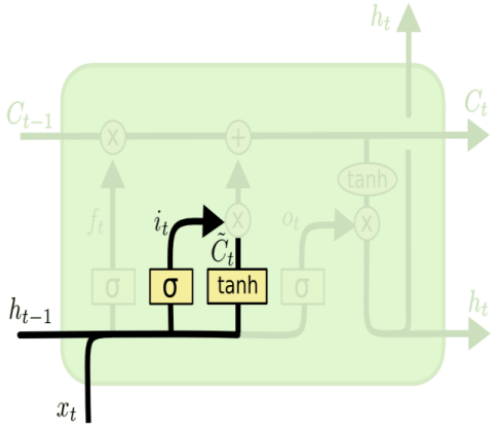
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

■ forget gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

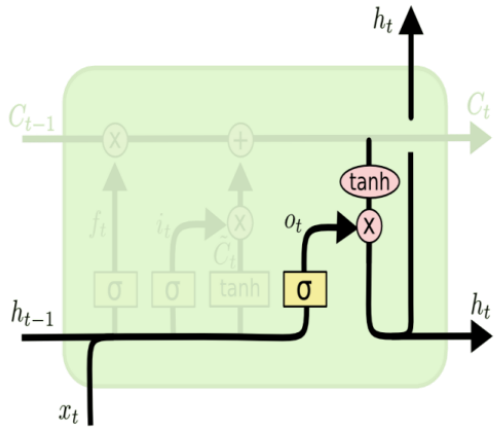
■ input gate



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

■ output gate, hidden state(단기 상태)



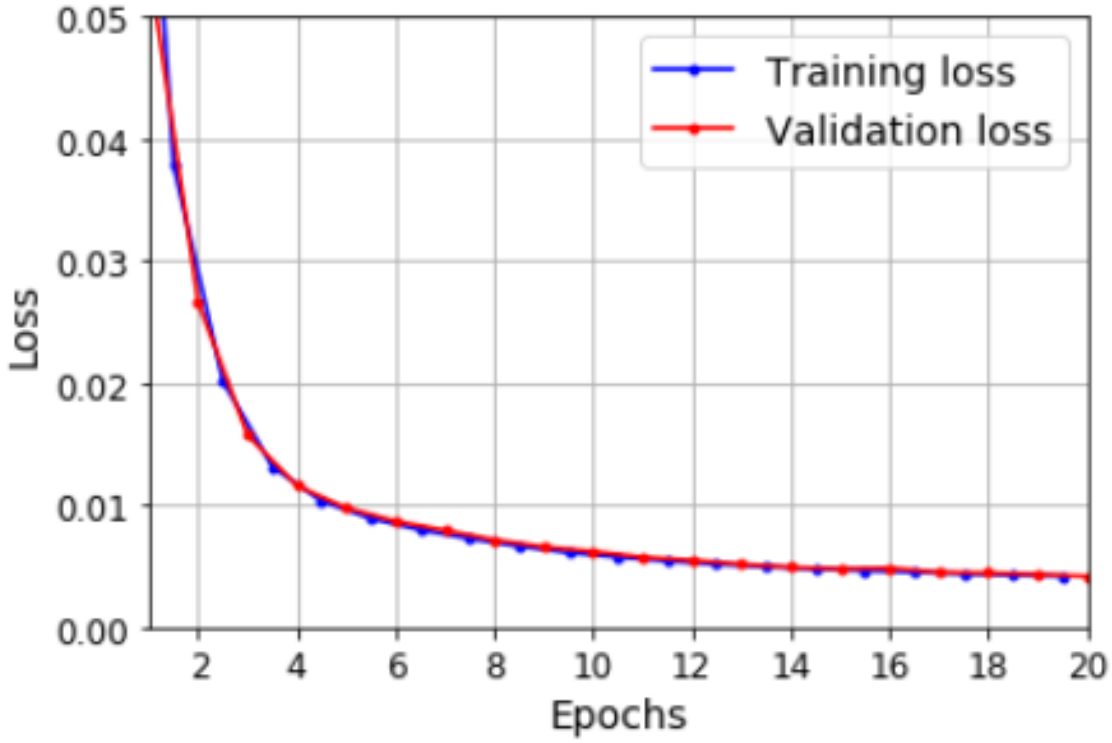
$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] \; + \; b_o\right)$$

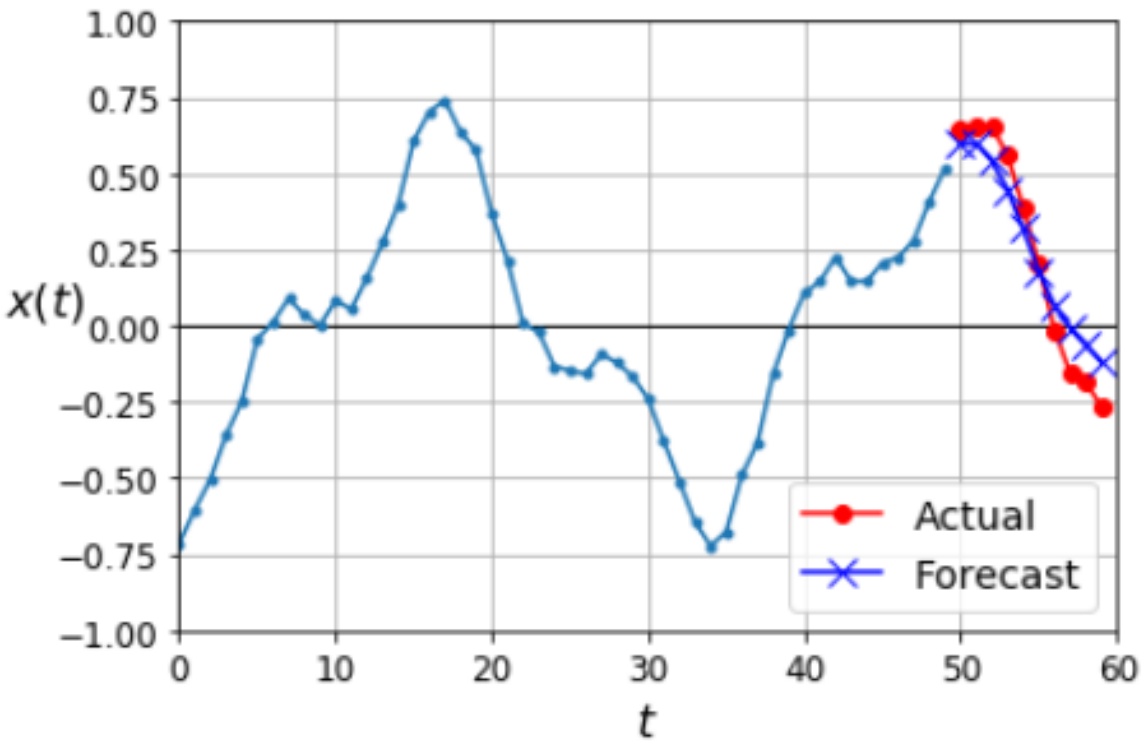
$$h_t = o_t * \tanh\left(C_t\right)$$

순환신경망 예측 모델

https://github.com/rickiepark/handson-ml2/blob/master/15_processing_sequences_using_rnns_and_cnns.ipynb

Run in Google Colab

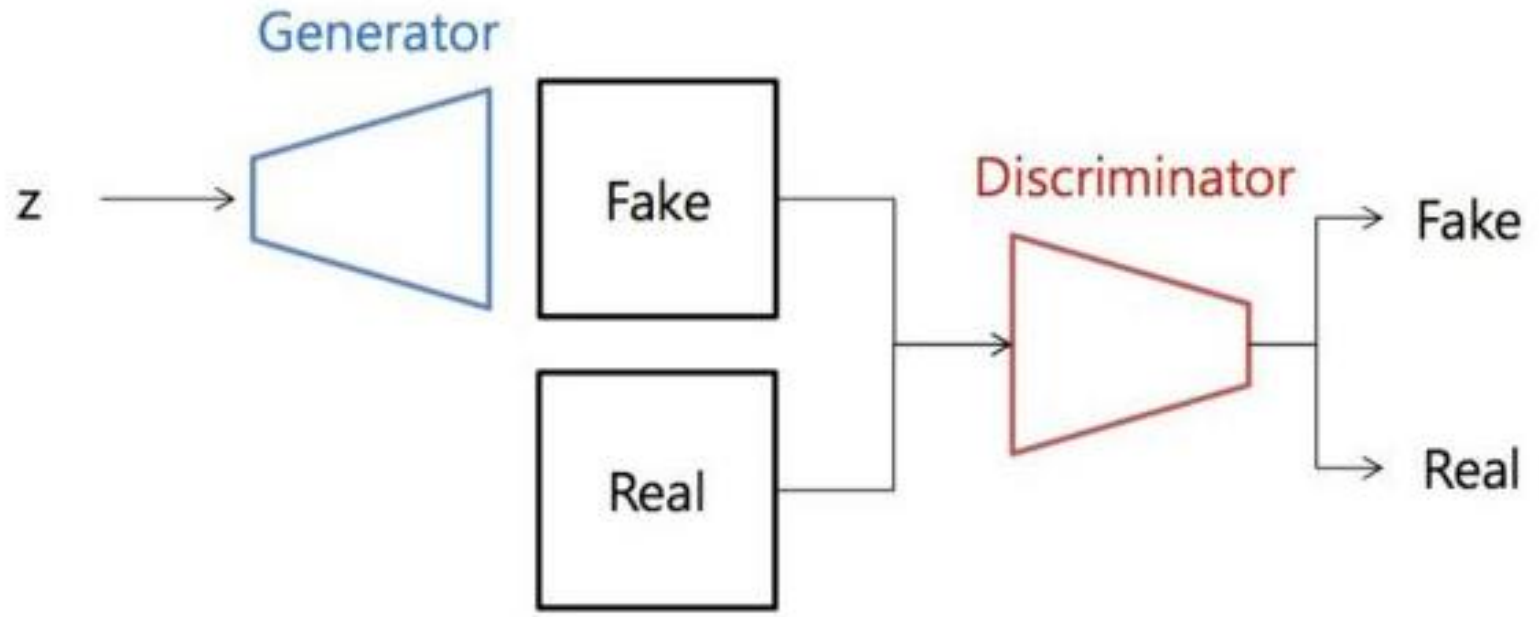




생성적 적대 신경망(GAN , Generative Adversarial Network)

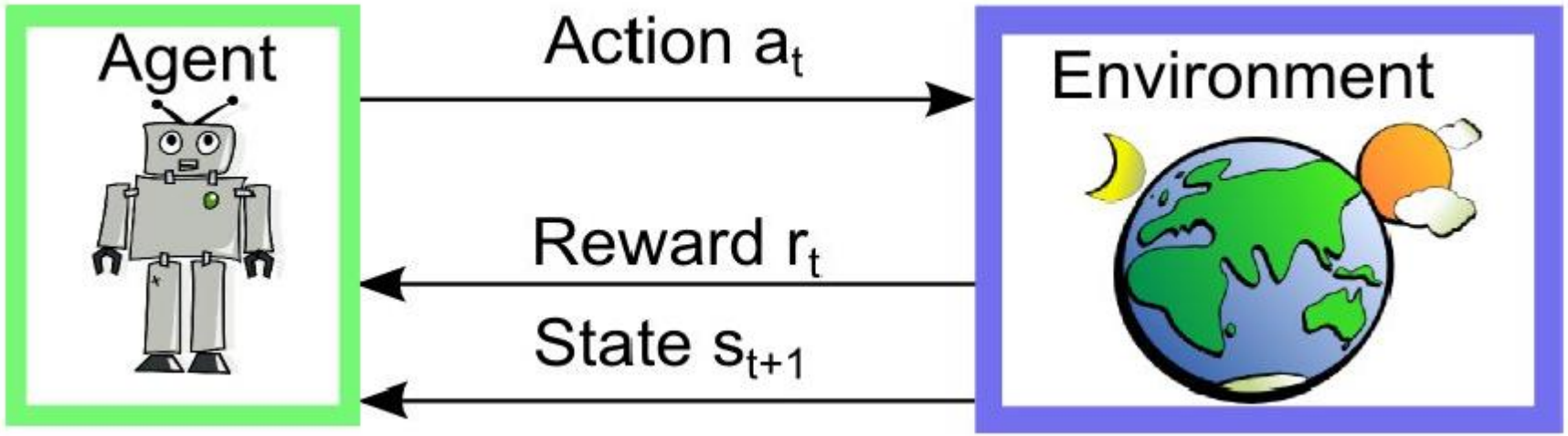
- 생성자(Generator)와 판별자(Discriminator)가 경쟁(Adversarial)하며 데이터를 생성(Generative)하는 모델
- 2014년 이안 굿펠로우(Ian Goodfellow)와 동료들이 심층신경망으로 새로운 이미지는 합성하는 방법 발표

  https://papers.nips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf



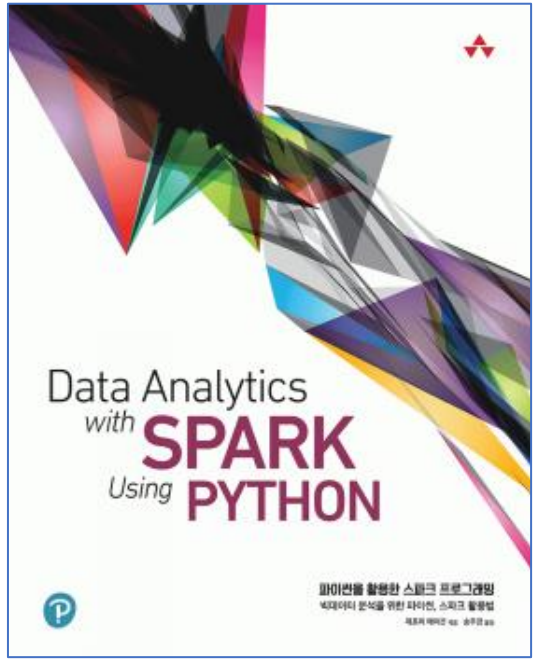
강화학습(RL , Reinforcement Learning)

- RL : Reinforcement Learning,
- 어떤 환경(Environment) 안에서 정의된 에이전트(Agent)가 현재의 상태(State)를 인식하여
  선택 가능한 행동(Action)들 중 보상(Reward)을 최대화하는 행동 혹은 생동 순서를 선택하는 방법



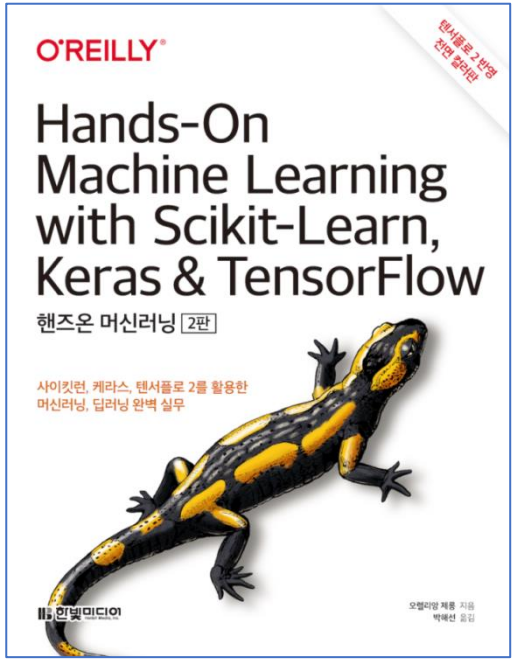
추천 도서

https://bit.ly/2SfLcme



https://bit.ly/3tbyWQf

https://bit.ly/36Ltr2X



https://wikidocs.net/

WikiDocs

https://tutorials.pytorch.kr/

파이토치(PyTorch) 튜토리얼

https://wikidocs.net/book/2788

PyTorch로 시작하는 딥러닝 입문


Q&A

수고하셨습니다. 감사합니다.

kgpark88@gmail.com