Machine Learning & Scikit-Learn

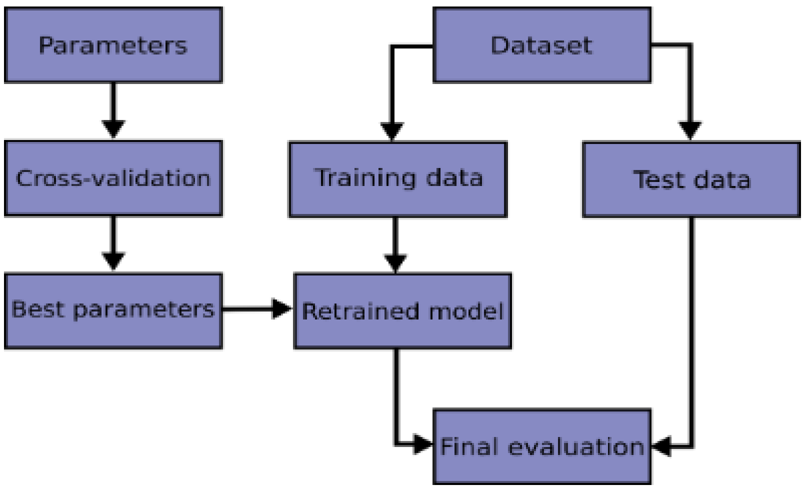
Day6. 파이프라인

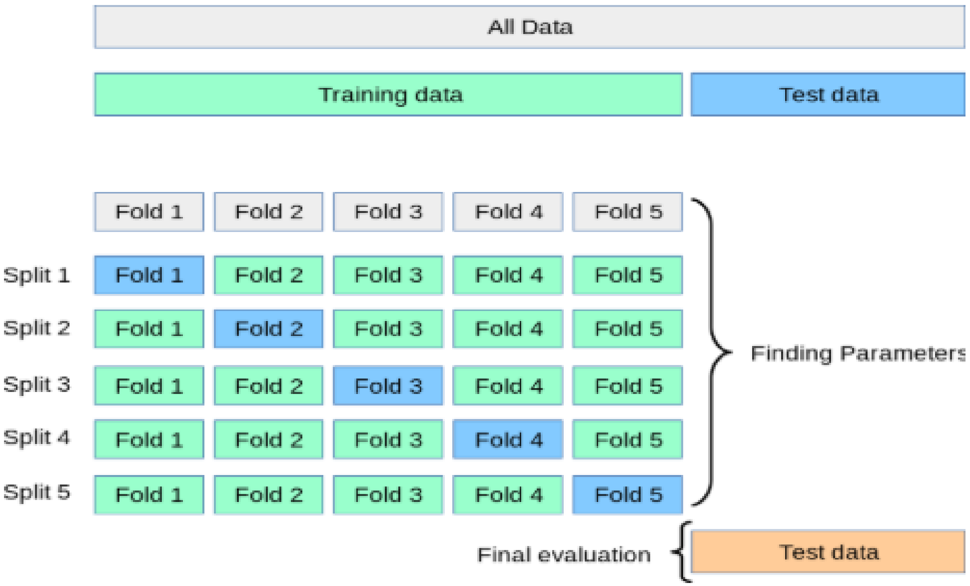
#1

파이프라인

cross_val_score ( ) 함수

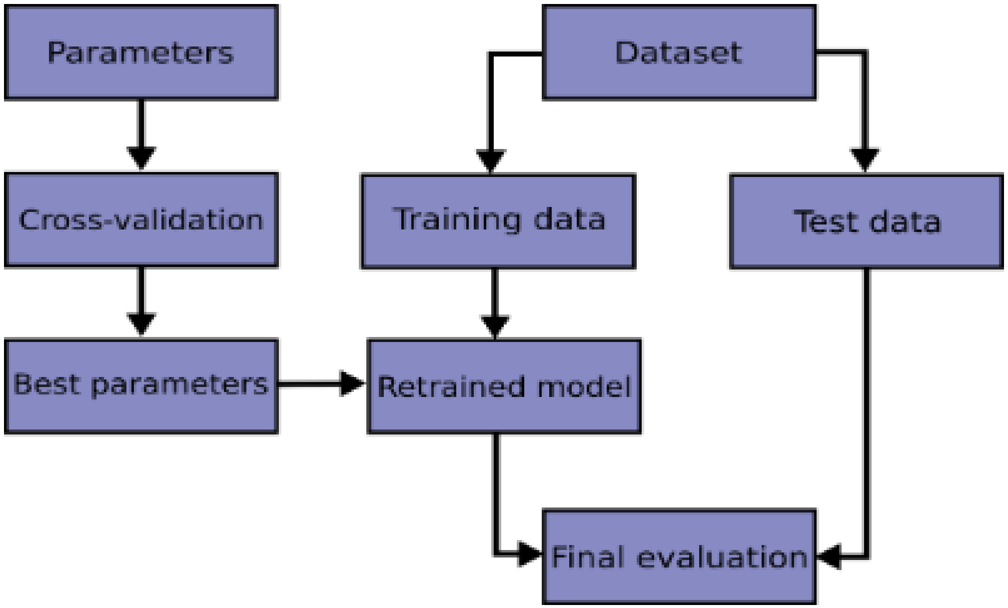
- cross_val_score(Estimator, X, y) 함수





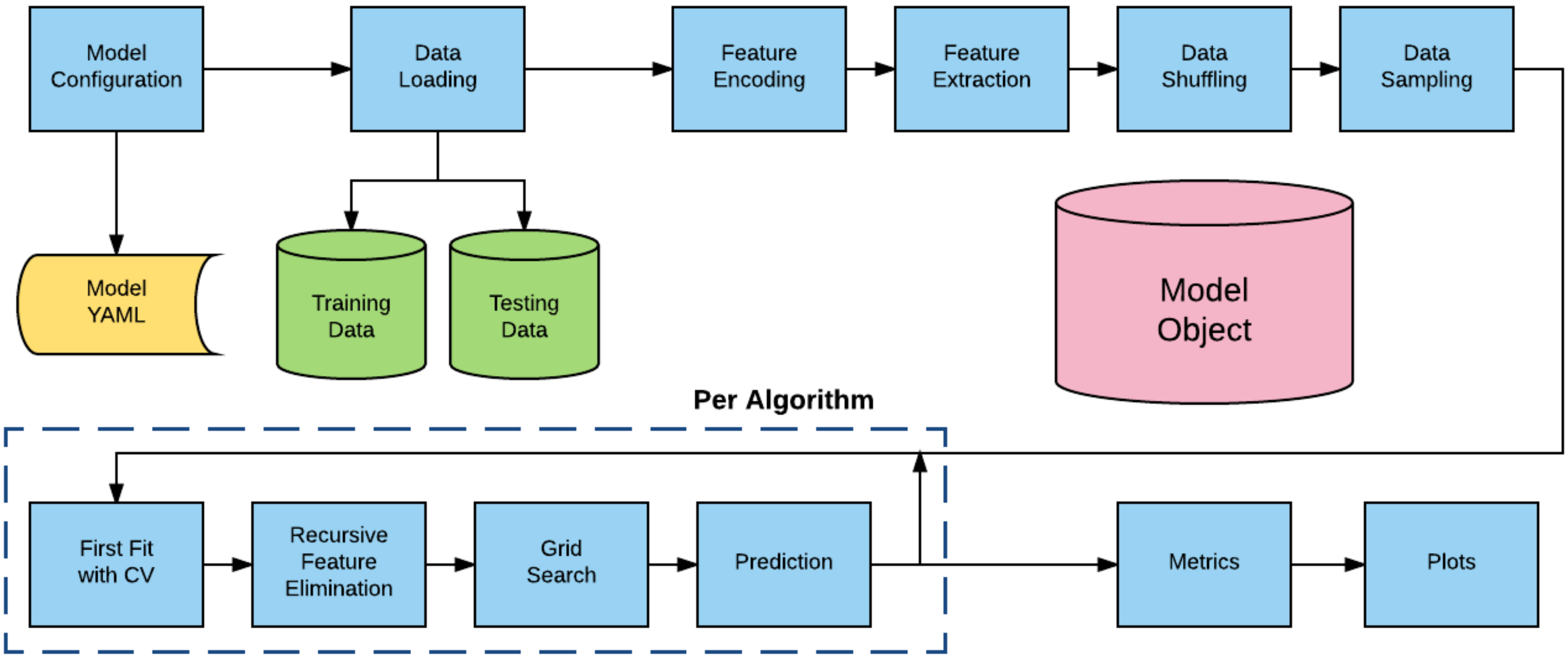
GridSearchCV () 함수

- GridSearchCV(Estimator, param_grid) 함수
  - Estimator에 param_grid 의 값을 적용하여 성능 평가하여 가장 성능이 우수한 파라메터를 가지는 estimator를 선택함



파이프 라인

- 파이프 라인
  - 문제 해결에 가장 적합한 알고리즘과 파라메터를 찾는 과정



함수 chain()

Iterator	Arguments	Results	Example
`chain()`	p, q, ...	p0, p1, ... plast, q0, q1, ...	`chain('ABC', 'DEF') --> A B C D E F`

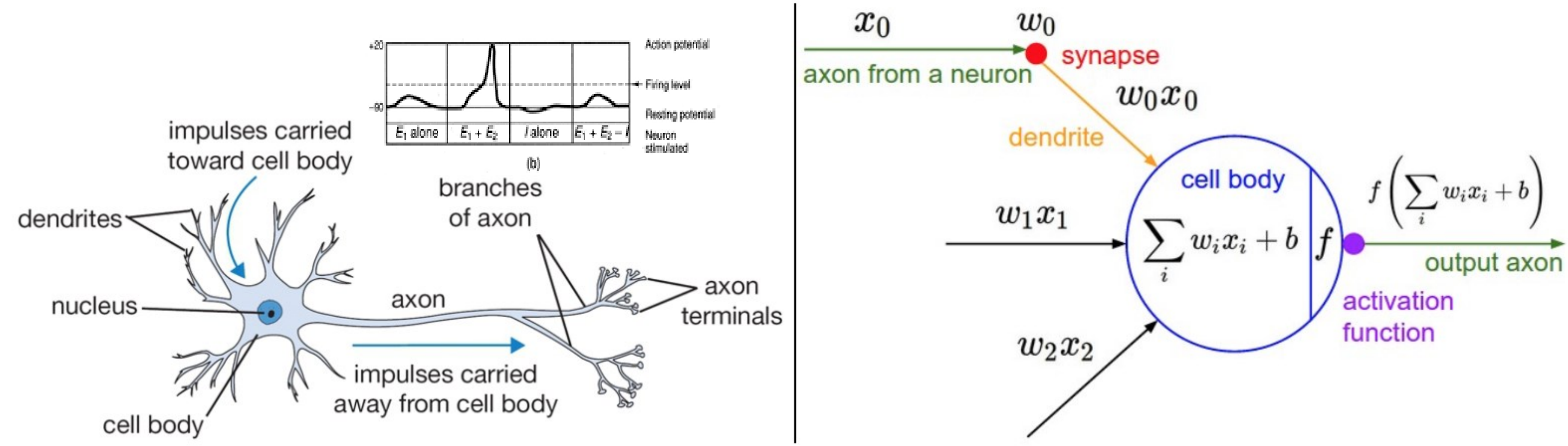
#2

Artificial Neural Network

추천 링크

https://youtu.be/aircAruvnKk

Artificial Neuron



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Unit step	g(z) =	1 if z ≥ 0 -1 otherwise.
	g(z) =	1 if z ≥ 0 0 otherwise.
Linear	g(z) =	z
Logistic (sigmoid)	g(z) =	1 / (1 + exp(-z))
Hyperbolic tangent (sigmoid)	g(z) =	$\frac{\exp(2z) - 1}{\exp(2z) + 1}$

. . .

Perceptron

Perceptron

- Perceptron
  - ✓ 논리 연산

NOT

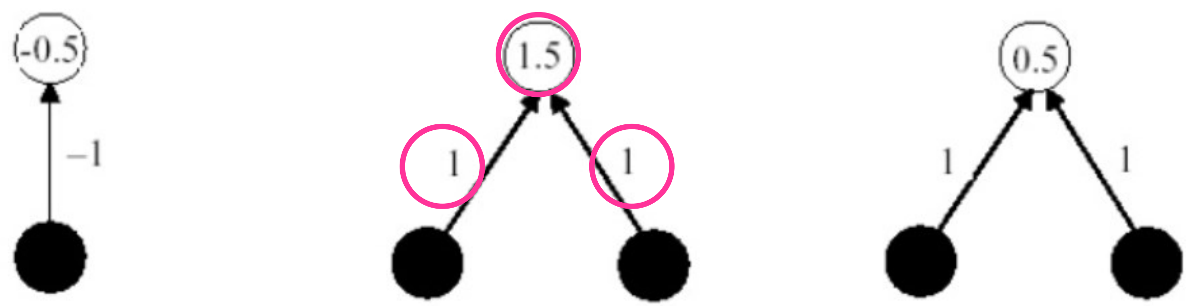
in	*out*
0	1
1	0

AND

in_1	in_2	*out*
0	0	0
0	1	0
1	0	0
1	1	1

OR

in_1	in_2	*out*
0	0	0
0	1	1
1	0	1
1	1	1

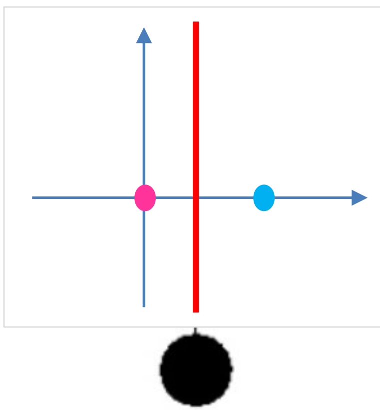


Perceptron

- Perceptron은 선형 분류기

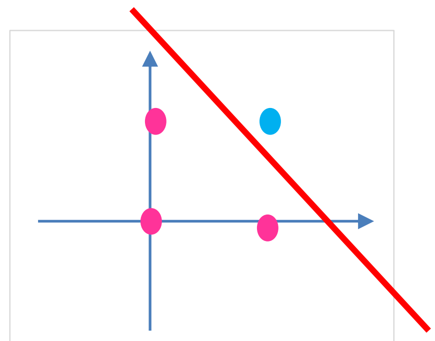
NOT

in	out
0	1
1	0



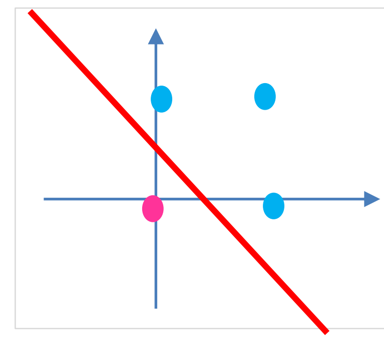
AND

in_1	in_2	out
0	0	0
0	1	0
1	0	0
1	1	1



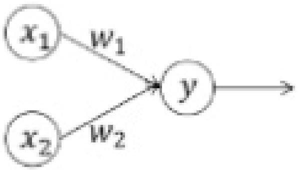
OR

in_1	in_2	out
0	0	0
0	1	1
1	0	1
1	1	1

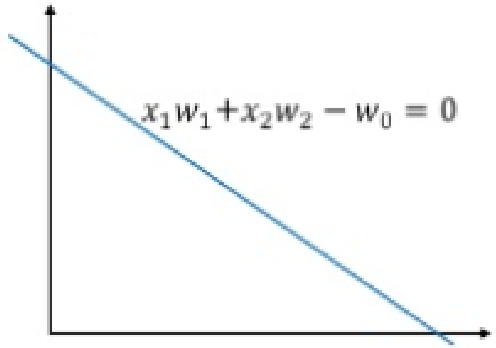


Artificial Neural Network

- Perceptron

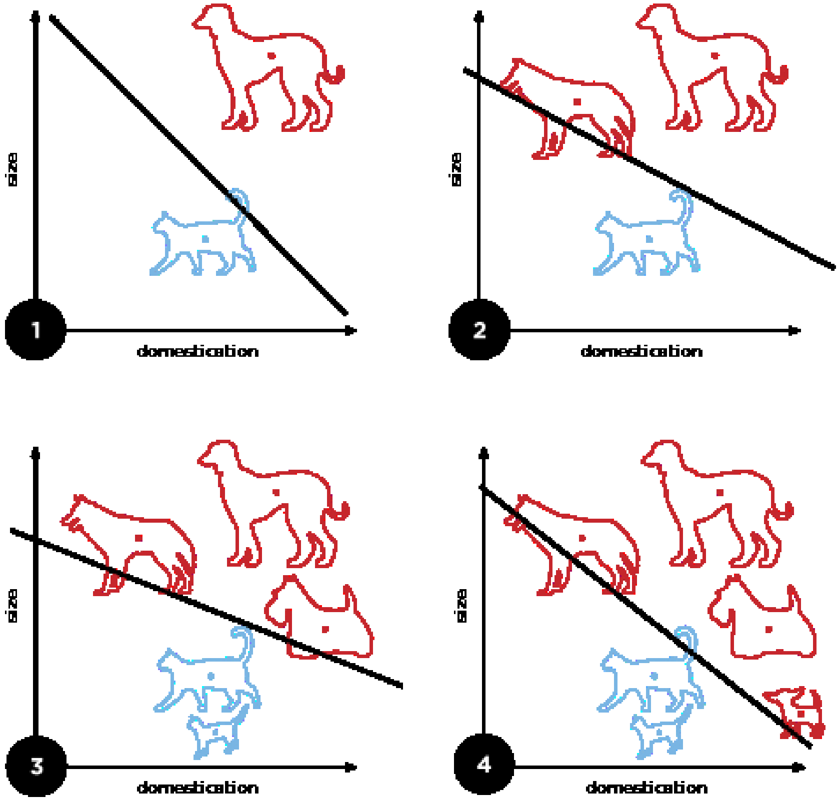


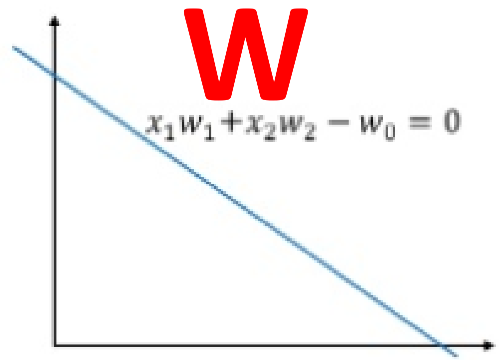
$$y = \phi_{step}\ (x_1 w_1 + x_2 w_2 - w_0)$$



Perceptron

- Perceptron 학습의 목표





Perceptron

- Perceptron 학습 규칙(= delta rule)
  - 에러가 최소가 되는 w를 구함
  - Cost function

$$E = \sum_j \frac{1}{2}(t_j - y_j)^2.$$

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial\left(\frac{1}{2}(t_j - y_j)^2\right)}{\partial w_{ji}} = \frac{\partial\left(\frac{1}{2}(t_j - y_j)^2\right)}{\partial y_j}\frac{\partial y_j}{\partial w_{ji}}$$

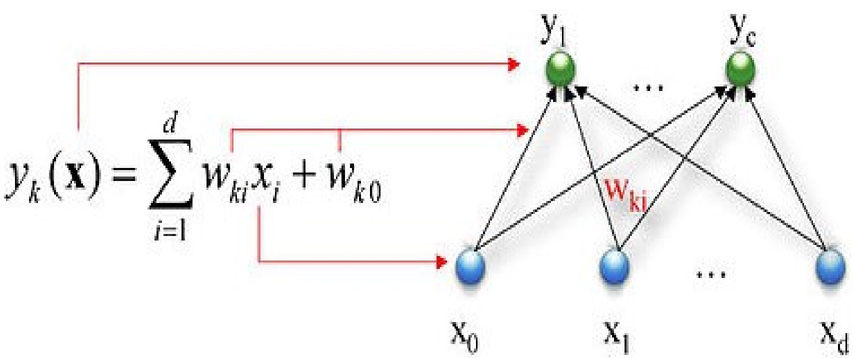
$$= -(t_j - y_j)\frac{\partial y_j}{\partial w_{ji}} = -(t_j - y_j)\,x_i,$$

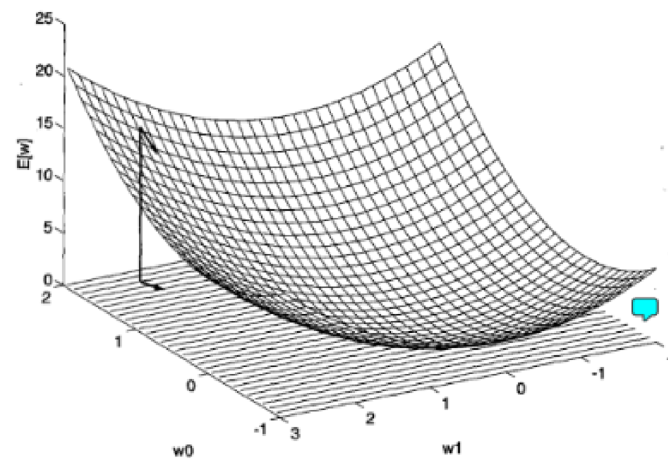
$$\boxed{\frac{\partial x_i w_{ji}}{\partial w_{ji}} = x_i,}$$

  - Weight update between input neuron I and output neuron j

$$W_{ji}(t+1) = W_{ji}(t) - \eta\Delta W_{ji}(t)$$

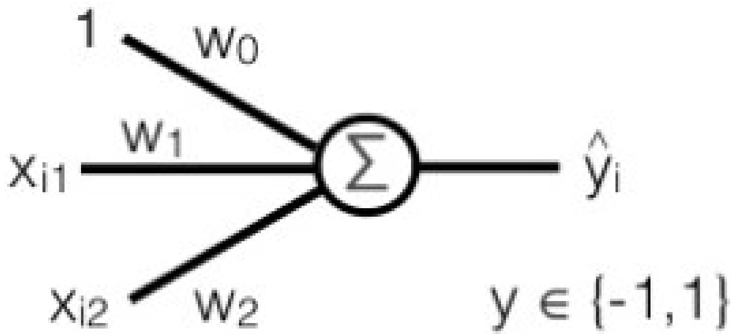
$$= W_{ji}(t) + \eta(t_j - y_j)x_i$$

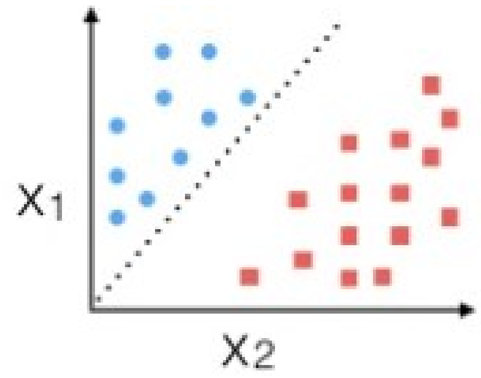




Perceptron

- Perceptron 학습 규칙





$$\hat{y} = \mathbf{w}^T\mathbf{x} = w_0 + w_1x_1 + w_2x_2$$

$$\hat{y}_i \begin{cases} 1 \text{ if } w^Tx_i \geq \theta \\ -1 \text{ otherwise} \end{cases}$$

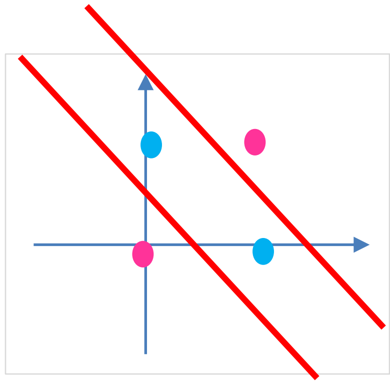
w_j = weight
x_i = training sample
y_i = desired output
$\hat{y}_i$ = actual output
t = iteration step
η = learning rate
θ = threshold (here 0)

update rule:
$$w_j(t+1) = w_j(t) + \eta(y_i - \hat{y}_i)x_i$$

until
$t+1$ = max iter
or error = 0

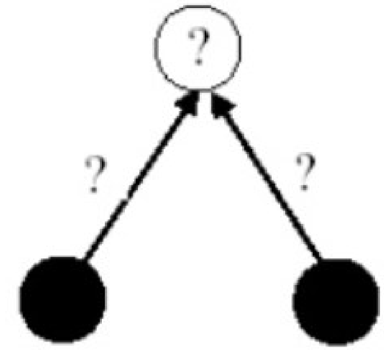
Perceptron

- Perceptron의 문제점: OXR 연산

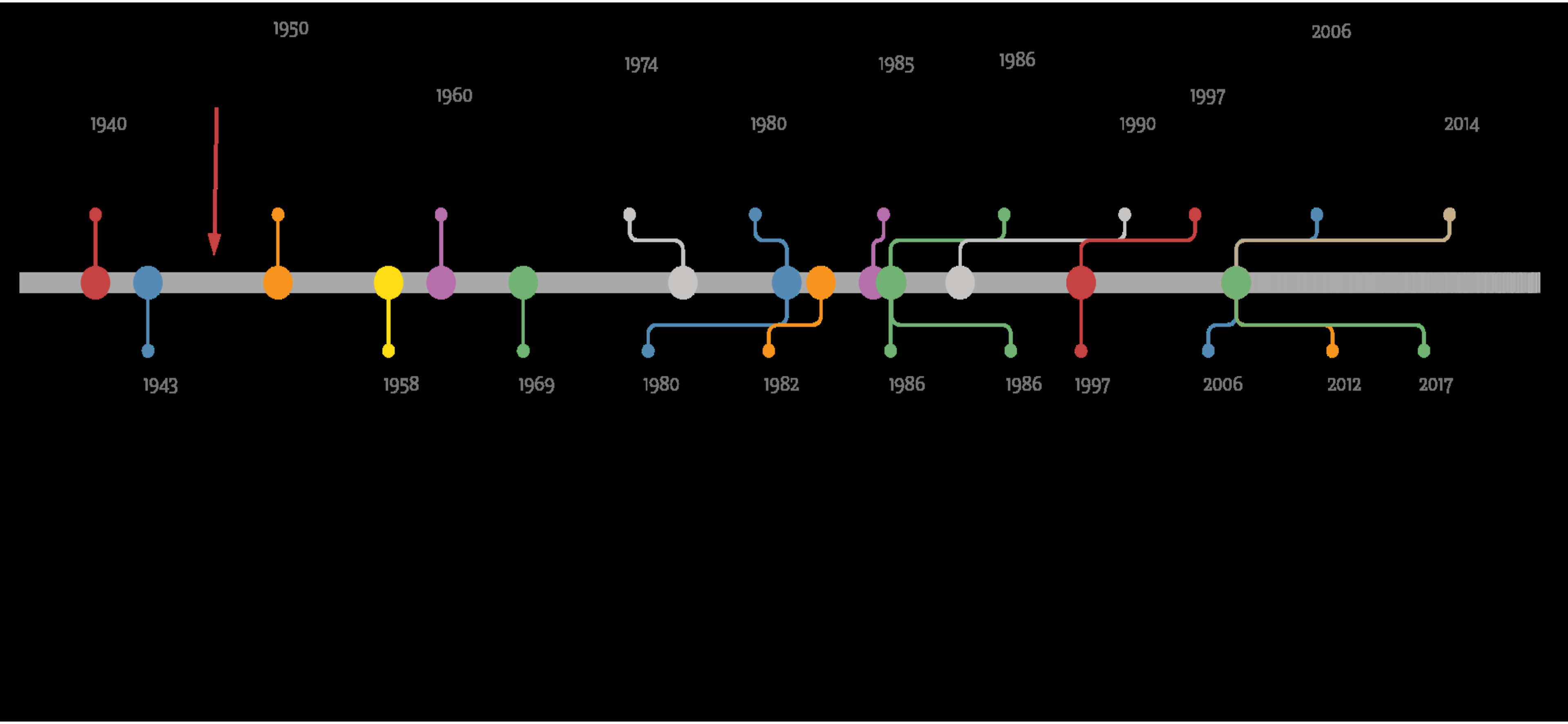


XOR

in_1	in_2	out
0	0	0
0	1	1
1	0	1
1	1	0



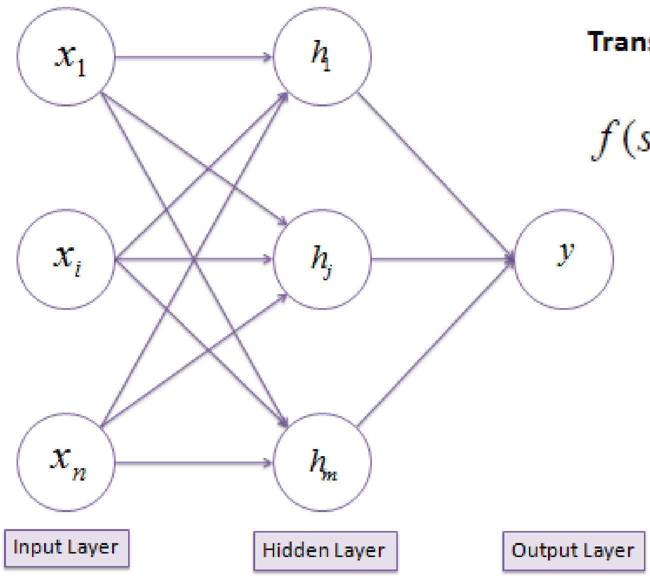
Deep learning time line



Multi layered Perceptron

MLP

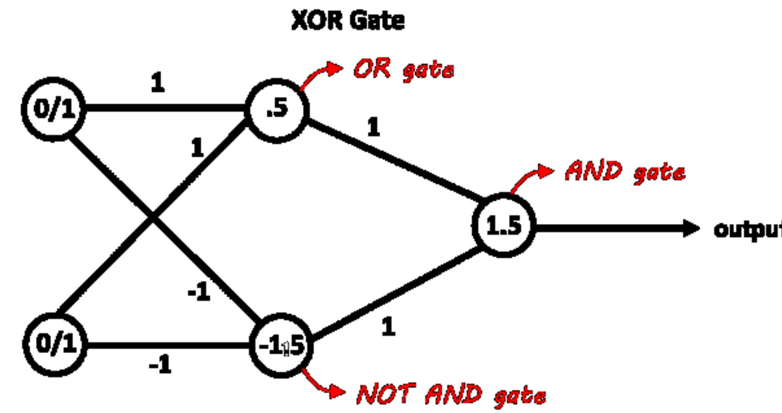
- MLP 구조



Transformation

$$f(s) = \frac{1}{1+e^{-s}}$$

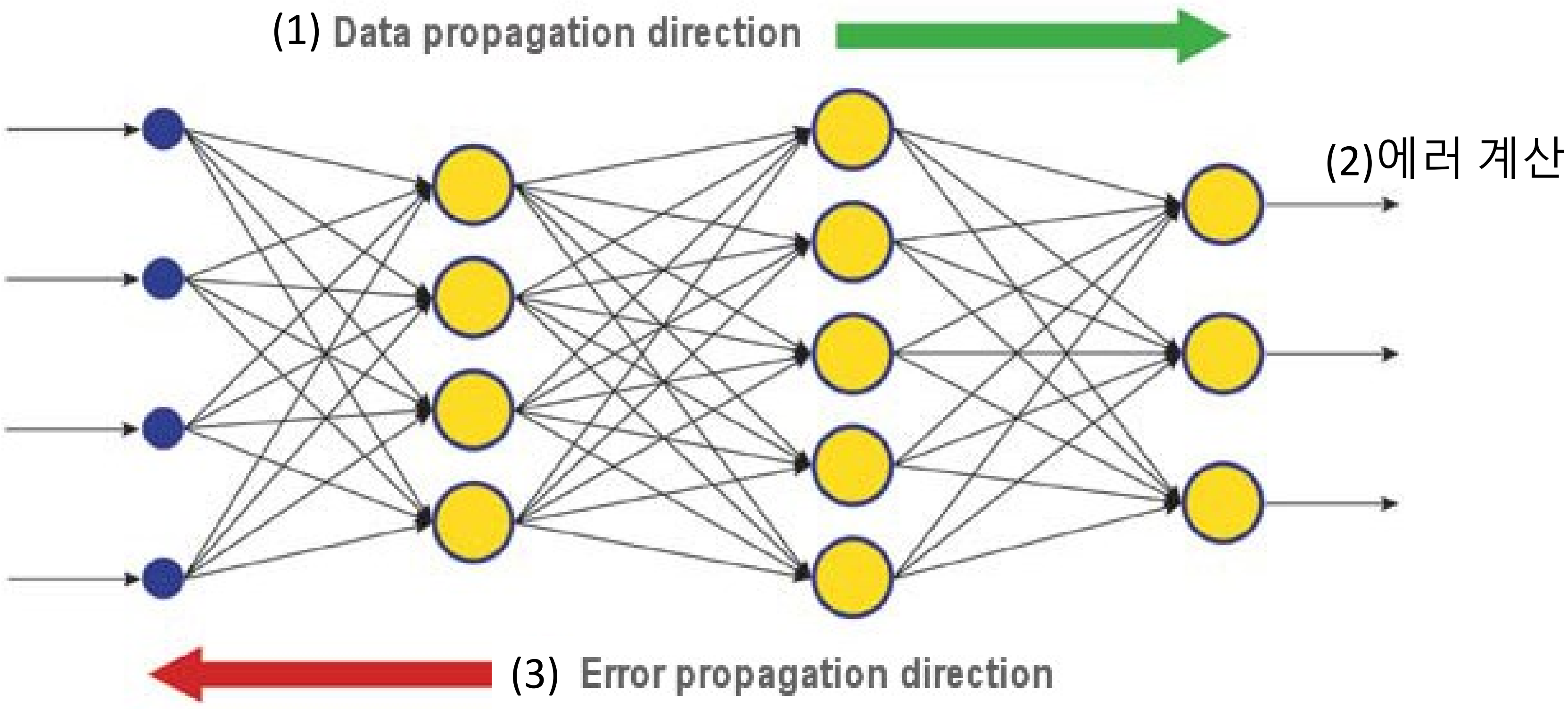
$$s = \sum w \cdot x$$



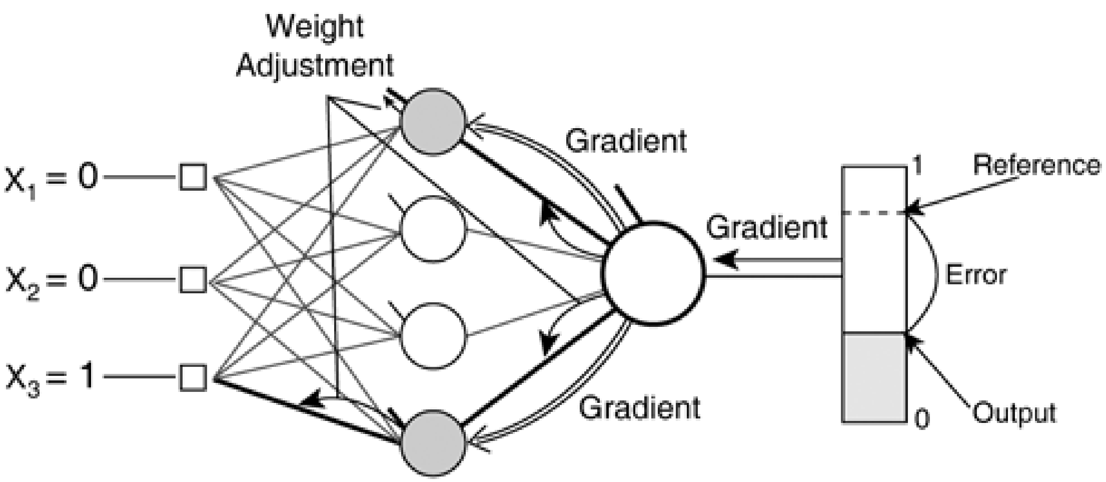
X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

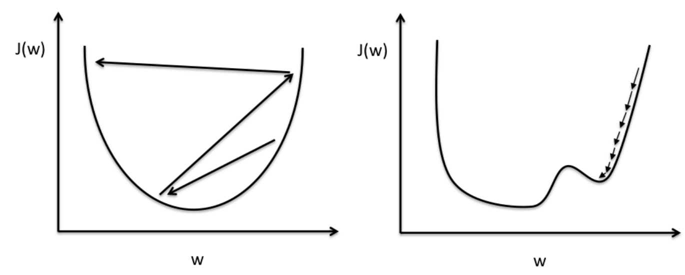
MLP

- MLP의 학습


(1) Data propagation direction
(2)에러 계산
(3) Error propagation direction

- 학습 규칙:오류 역전파 학습(EBP, Error Back Propagation)





Large learning rate: Overshooting.

Small learning rate: Many iterations until convergence and trapping in local minima.

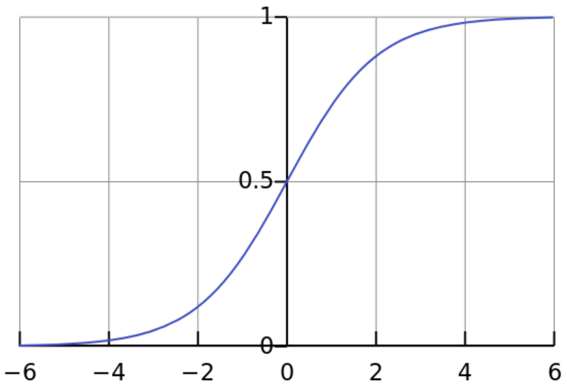
$$\delta_j = \begin{cases} \sigma'(\zeta_j)(t_j - y_j) & \text{if j is an output unit} \\ \sigma'(\zeta_j)\sum_k \delta_k w_{jk} & \text{if j is a hidden unit} \end{cases}$$

MLP

- Activation functions

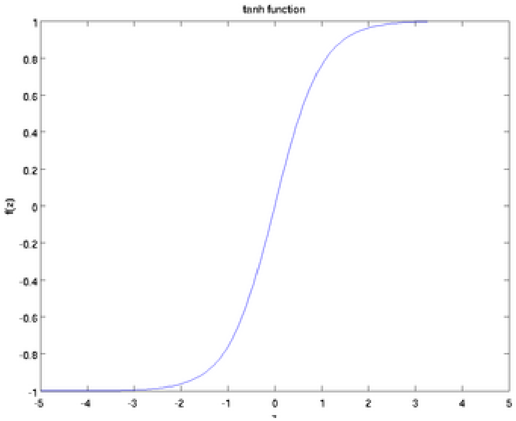
Sigmoid function

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



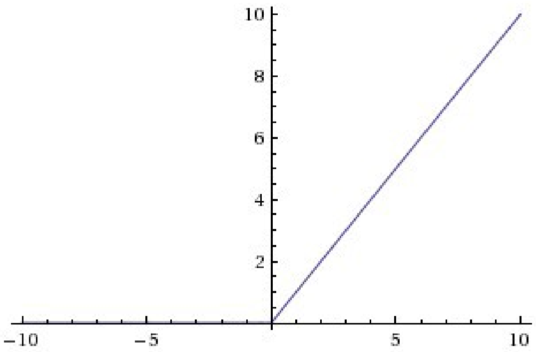
tanh function

$$tanh(x) = \frac{2}{1+e^{-2x}} - 1$$



ReLu function

$$A(x) = max(0,x)$$



Scikit-learn을 활용한 MLP

- 필기체 숫자 인식에 적용

(1) 자료 준비

```
from sklearn.datasets import load_digits
```

```
X, y = load_digits(return_X_y=True)
print(len(X))
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                   test_size = 0.33)
```

```
1797
```

Scikit-learn을 활용한 MLP

- 필기체 숫자 인식에 적용

(2) 모델 import와 인스턴스화

```
from sklearn.neural_network import MLPClassifier
```

```
mlp = MLPClassifier(solver='sgd', activation='logistic',
                    hidden_layer_sizes=(30, 10),
                    random_state=1,max_iter=2000)
```

Scikit-learn을 활용한 MLP

- 필기체 숫자 인식에 적용

(3) 학습 데이터로 학습

```
mlp.fit(X_train,y_train)
```

```
C:\Users\user\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_pe
ing: Stochastic Optimizer: Maximum iterations (2000) reached and the optimiza
  % self.max_iter, ConvergenceWarning)
```

```
MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
              beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(30, 10), learning_rate='constant',
              learning_rate_init=0.001, max_iter=2000, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=1, shuffle=True, solver='sgd', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False)
```

Scikit-learn을 활용한 MLP

- 필기체 숫자 인식에 적용

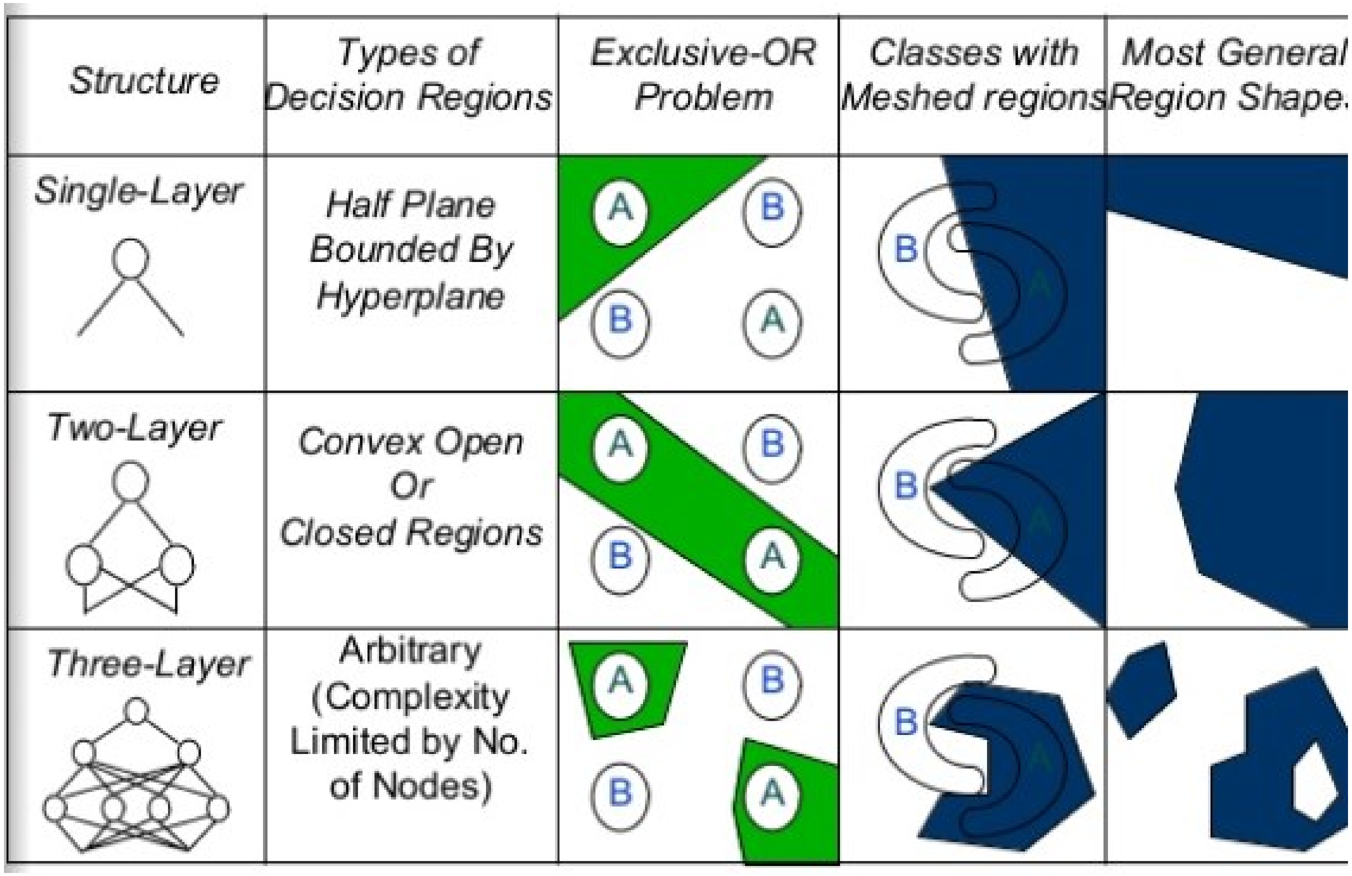
(4) 성능 평가

```
mlp.score(X_train, y_train)
```

0.9891936824605154

```
mlp.score(X_test, y_test)
```

0.9579124579124579


Structure
Types of Decision Regions
Exclusive-OR Problem
Classes with Meshed regions
Most General Region Shapes
Single-Layer
Half Plane Bounded By Hyperplane
Two-Layer
Convex Open Or Closed Regions
Three-Layer
Arbitrary (Complexity Limited by No. of Nodes)
A
B

Q & A