머신 러닝·딥러닝에 필요한 수학 기초 with 파이썬

Neural Network

인공신경망: 복잡한 입력과 출력의 관계를 표현하기

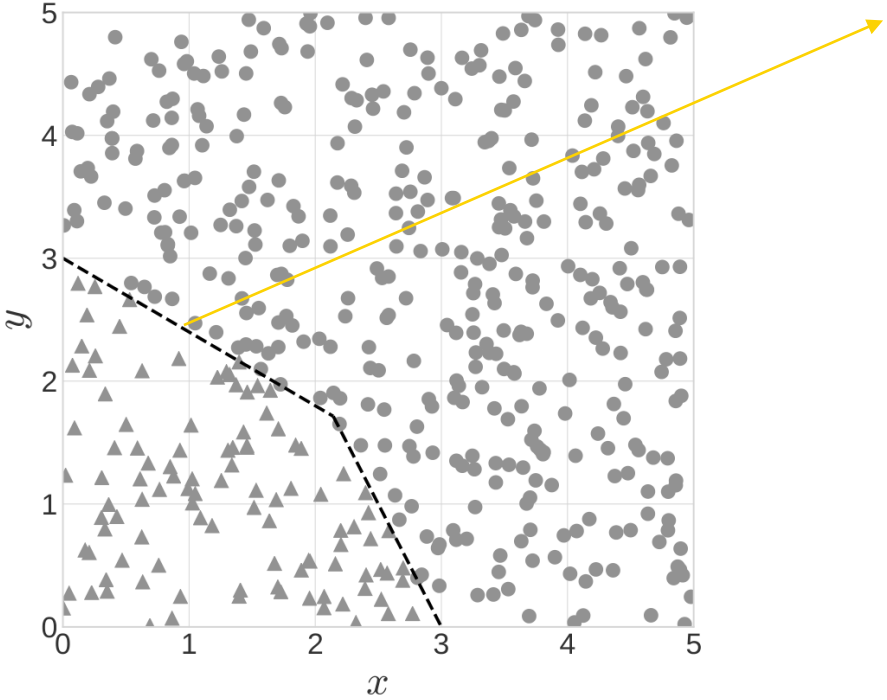
조준우
metamath@gmail.com

학습 개요

- 선형 분류기
- 선형 분류기의 결합
- 인공신경망 학습하기
- 인공신경망 미분하기(역전파 알고리즘)*

Sample Data

- 2차원 데이터



원과 삼각형을 나누는 비선형 결정 경계decision boundary

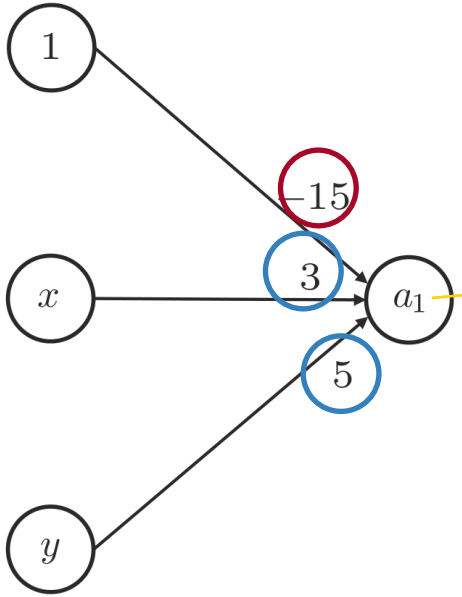
목표: 이 결정 경계를 그려주는 함수를 찾고 싶다.

선형 분류기

- 이변수 스칼라 함수 $z_1 = 3x + 5y - 15$

가중치weight

편향bias



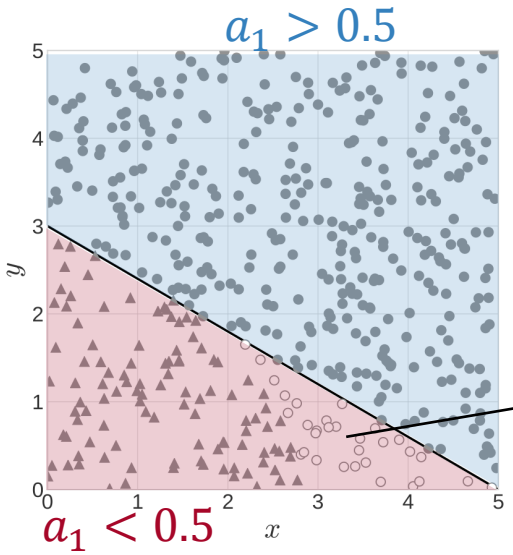
$$a_i = f\left(\sum_{j=1}^{D} w_j x_j\right)$$

비선형 활성화함수
non-linear activation func.

선형 분류기 결정경계

- 이변수 스칼라 함수 $z_1 = 3x + 5y - 15$ 3차원에 있는 평면

$3x + 5y - 15 = 0$ 2차원에 있는 직선



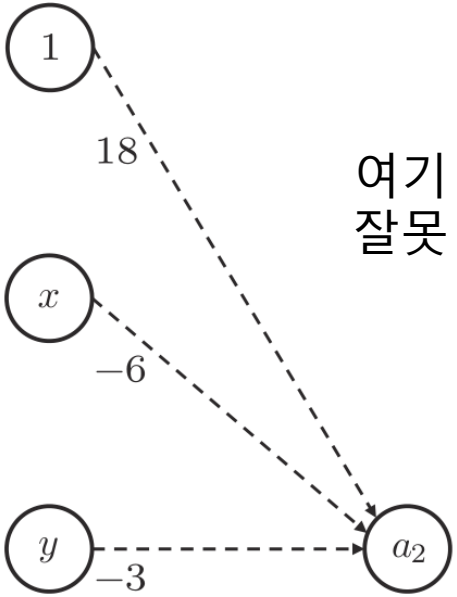
여기 원은 삼각형으로 잘못 분류

- 이 직선 위에 z_1은 모두 0
- a_1 으로 로지스틱 시그모이드 함수를 쓰면 이 직선 위에 a_1은 모두 0.5

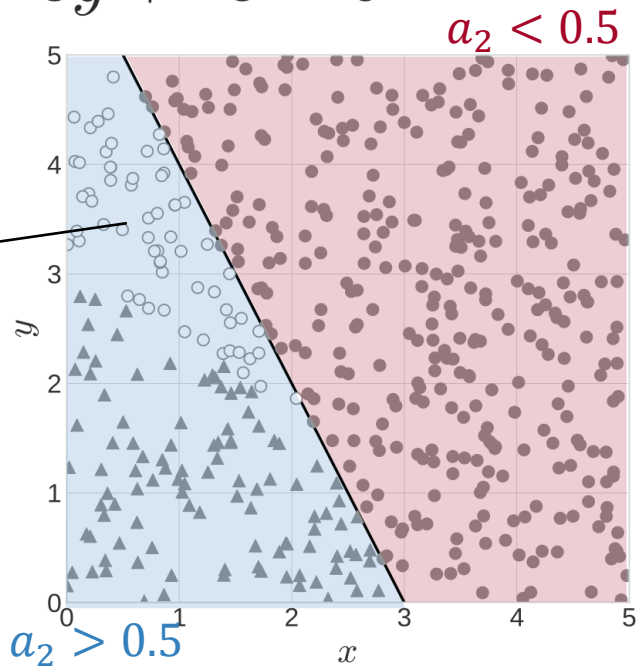
두번째 선형 분류기

- 이변수 스칼라 함수 $z_2 = -6x - 3y + 18$

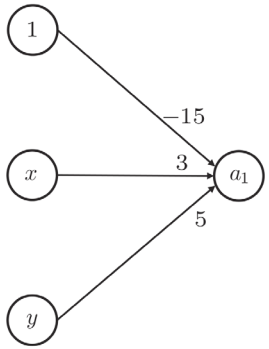
$$-6x - 3y + 18 = 0$$

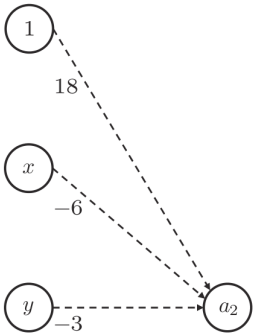


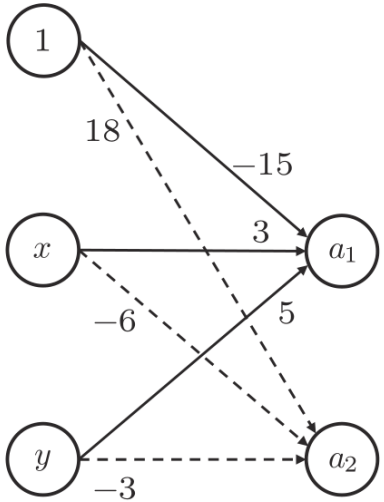
여기 원은 삼각형으로
잘못 분류



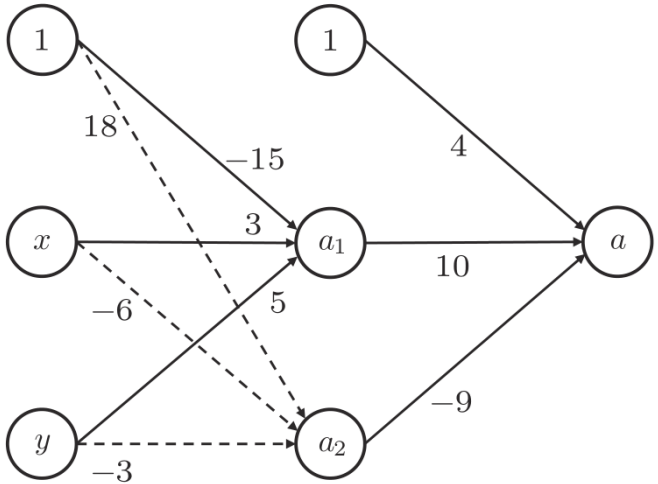
선형 분류기 결합



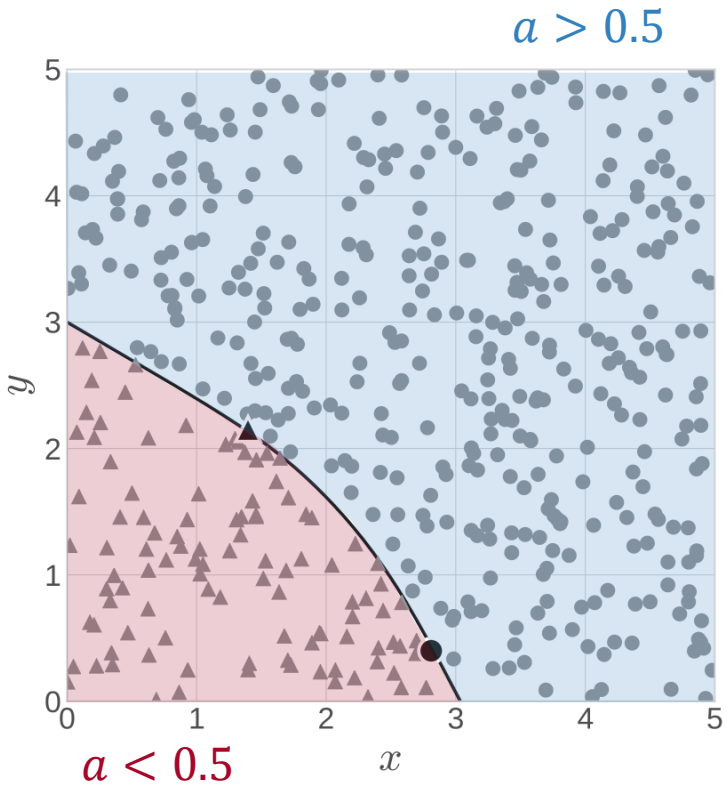


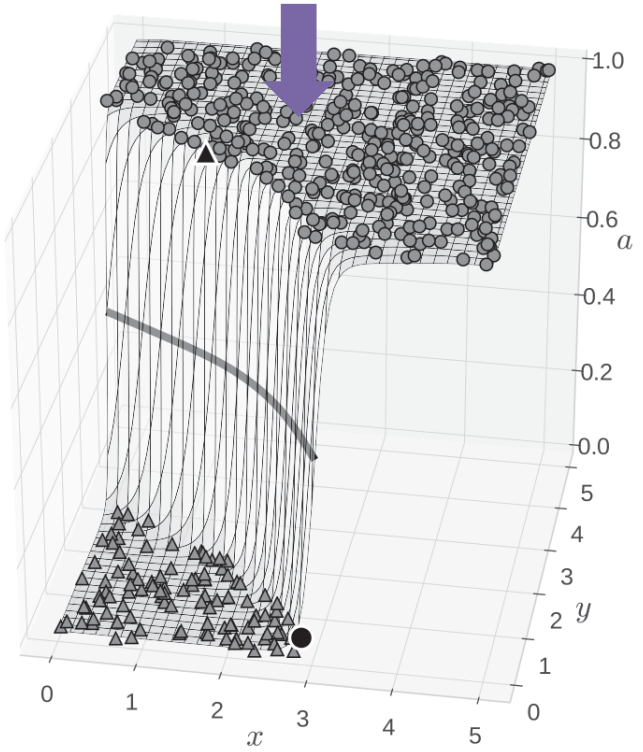


$$z = 10a_1 - 9a_2 + 4$$



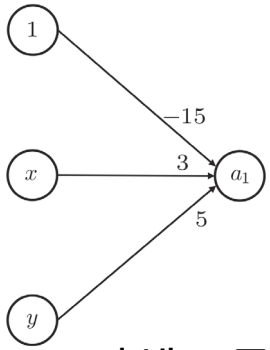
결합된 선형 분류기의 결정경계



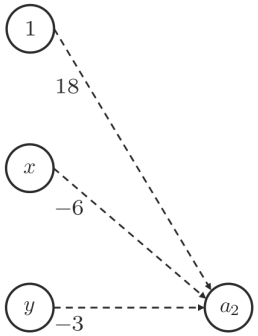


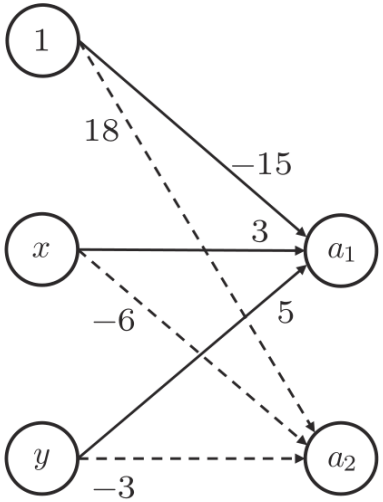
합성함수로 바라보기

- 두 선형 분류기의 결합→합성함수

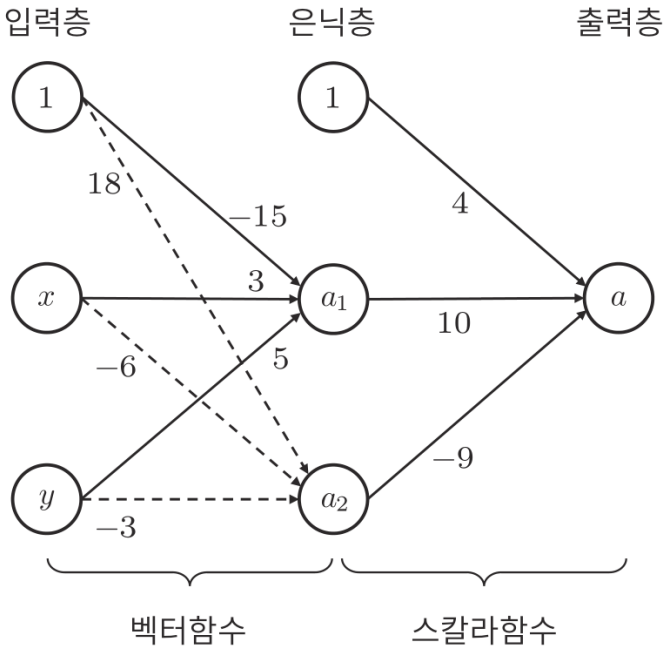


퍼셉트론



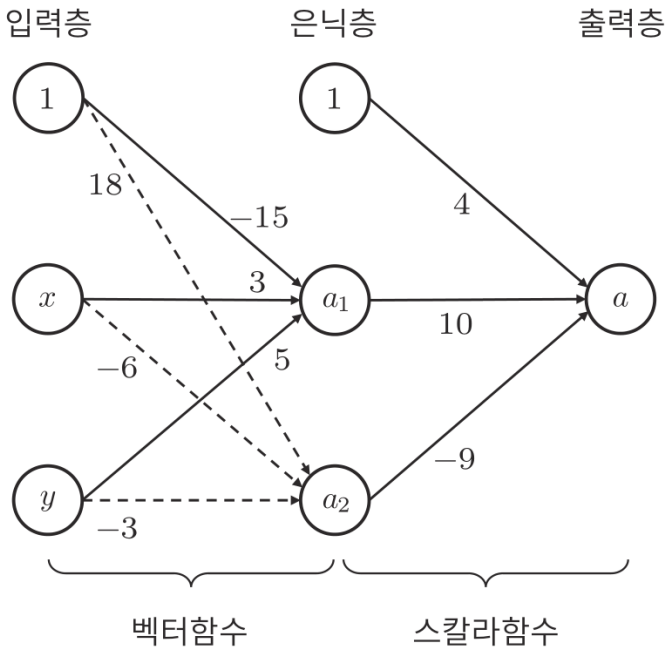


다층 퍼셉트론
MLP



가중치 표현

- 가중치와 편향의 행렬 표현

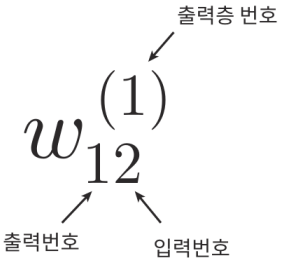


$$a_1 = 3x + 5y - 15$$

$$a_2 = -6x - 3y + 18$$

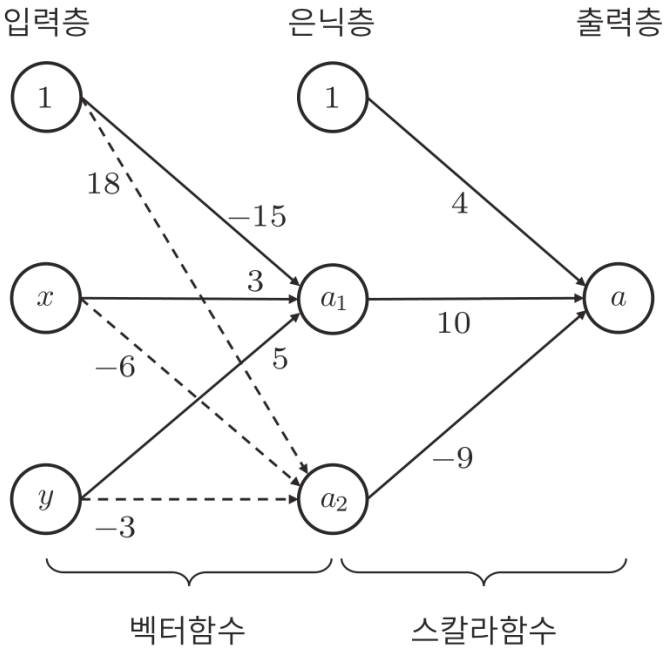
$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} -15 & 3 & 5 \\ 18 & -6 & -3 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix}$$

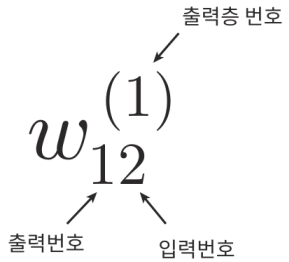
$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} = \begin{bmatrix} -15 & 3 & 5 \\ 18 & -6 & -3 \end{bmatrix}$$



가중치의 행렬 형태

- 가중치와 편향의 행렬 표현





$$\begin{bmatrix} b_1^{(1)} & w_{11}^{(1)} & w_{12}^{(1)} \\ b_2^{(1)} & w_{21}^{(1)} & w_{22}^{(1)} \\ b_1^{(2)} & w_{11}^{(2)} & w_{12}^{(2)} \end{bmatrix}$$

인공신경망 순전파

```
def network(X, W):
    """
    X : (N, D)
    W : (3, 3)
        [b^(1)_1, w^(1)_11, w^(1)_12]
        [b^(1)_2, w^(1)_21, w^(1)_22]
        [b^(2)_1, w^(2)_11, w^(2)_12]

    ret : (N,)

    D, H, A = 2, 2, 1
    """
    X = np.hstack( (np.ones(X.shape[0]).reshape(-1,1), X) ) # (N,D)->(N,D+1) # ❶
    Z1 = np.dot(W[:2,:], X.T) # (H,N)=(H,D+1)*(D+1,N) ❷
    A1 = sigmoid(Z1)          # (H,N)                  ❸

    A1 = np.vstack((np.ones(A1.shape[1]), A1)) # (H,N)->(H+1,N) ❹
    Z = np.dot(W[-1,:], A1)                    # (H+1,)*(H+1,N) ❺
    A = sigmoid(Z)                             # (N,)           ❻

    return A
```

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & y_N \end{bmatrix}$$

$$\begin{bmatrix} b_1^{(1)} & w_{11}^{(1)} & w_{12}^{(1)} \\ b_2^{(1)} & w_{21}^{(1)} & w_{22}^{(1)} \\ b_1^{(2)} & w_{11}^{(2)} & w_{12}^{(2)} \end{bmatrix}$$

```
def sigmoid(x):
    return 1 / (1+np.exp(-x))
```

순전파 테스트

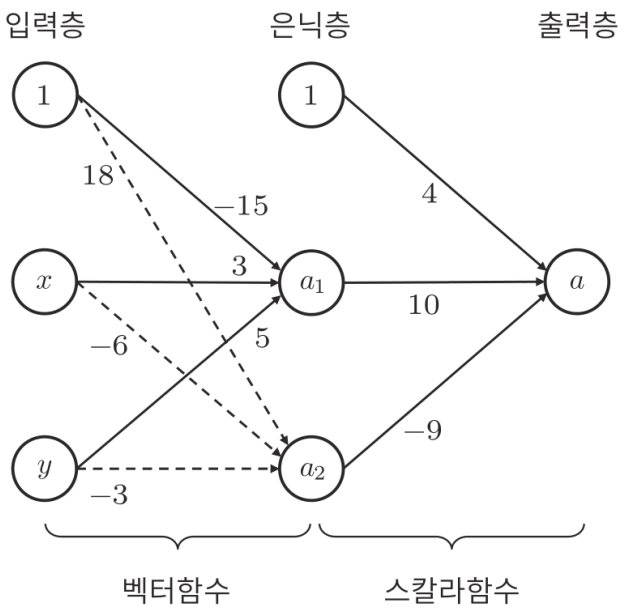
```
import numpy as np

W = np.array([ [-15, 3, 5], [18, -6, -3], [4, 10, -9] ]) #❶
pred = network(samples, W)                                #❷

pred[pred>=0.5] = 1                          #❸
pred[pred<0.5] = 0

result = pred==target                        #❹

np.size(result) - np.count_nonzero(result) #❺
#>>> 2
```



인공신경망=함수

```
np.random.seed(17)
W = np.random.randn(9)

W.reshape(3,3)
#>>> array([[ 0.2763, -1.8546,  0.6239],
            [ 1.1453,  1.0372,  1.8866],
            [-0.1117, -0.3621,  0.1487]])
pred = network(samples, W.reshape(3,3))

pred[pred>=0.5] = 1
pred[pred<0.5] = 0

result = pred==target
np.size(result) - np.count_nonzero(result)
#>>> 163
```

```
pred = network(samples, W.reshape(3,3))
```

주어지는 데이터 X

바꿔야 하는 변수 w

목적 함수

$$J(\mathbf{W};\mathbf{X},\mathbf{t}) = \frac{1}{2N}\sum_{i=1}^{N}(t_i - y_i)^2$$

주어지는 정답

```
pred = network(samples, W.reshape(3,3))
```

주어지는 데이터 X

우리가 바꿔야 하는
변수 w

줄여야 한다!!!

목적 함수 코드

```
def J(W, X, T):
    """
    W: 함숫값을 결정하는 변수, 가중치 (9,)
    X: 주어진 점 데이터 X, X: (N,D)
    T: 데이터에 대한 클래스 T, 0 또는 1, T: (N,)
    """
    N = X.shape[0]
    W = W.reshape(3,3)

    Y = network(X, W)
    return (1/(2*N)) * ((T-Y)**2).sum()

# 초기 상태에서 목적함숫값
J(W, samples, target)
#>>> 0.1263148192185165
```

$$J(\mathbf{W};\mathbf{X},\mathbf{t}) = \frac{1}{2N}\sum_{i=1}^{N}(t_i - y_i)^2$$

학습 하기

- 학습?
  - J를 가장 작게 하는 W를 찾는 것
  - 즉 J함수를 최적화 하는 것

- J함수를 미분?
  - 수치 미분으로 가능

```
from scipy import optimize

# https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fmin_
cg.html
W_star = optimize.fmin_cg(J, W, args=(samples, target),  gtol=1e-06)

#>>> Optimization terminated successfully.
         Current function value: 0.000003
         Iterations: 509
         Function evaluations: 20262
         Gradient evaluations: 1842
```

학습 결과

```
W_star = W_star.reshape(3,3)
W_star
#>>> array([[ 42.3795,  -9.3715, -13.6424],
           [-24.8334,   8.7549,   2.9808],
           [ 17.7471, -45.1586,  42.9089]])
```

- 미분 계수: 9개 (그래디언트 차원: 9)
  - 수치 미분 가능
- 만약 가중치가 10000개
  - 미분 계수 10000개
  - 수치 미분?
- 수치 미분 보다 빠르게 그래디언트를 구해야 최적화 가능
  - 자동 미분 사용