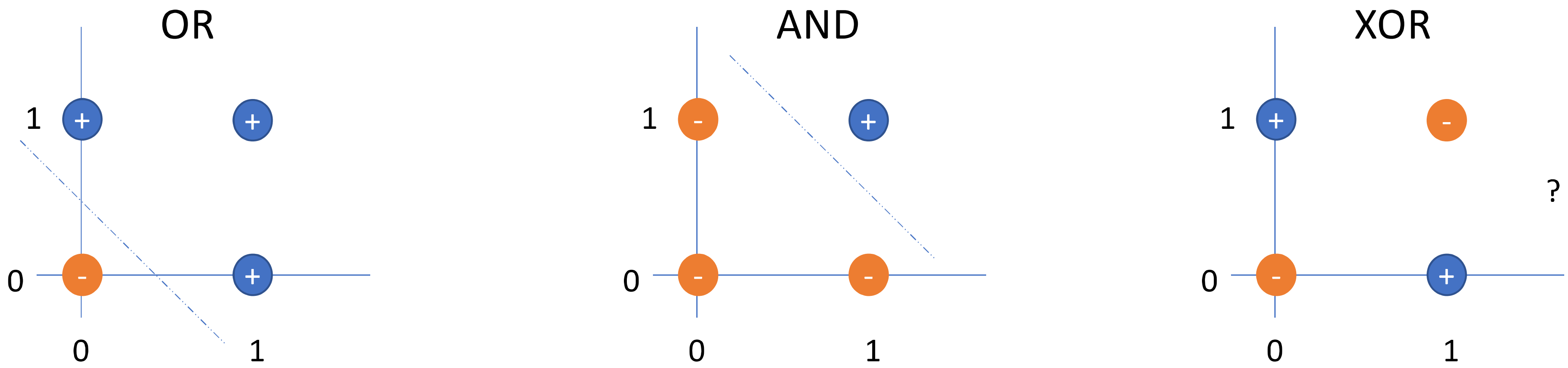
Beginning of the Artificial Neural Network



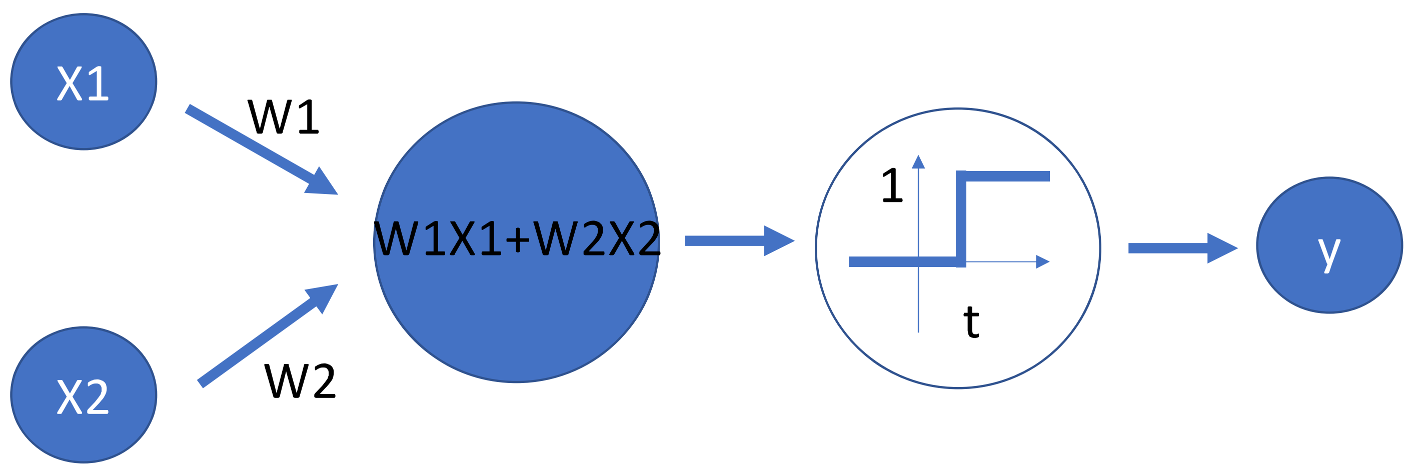
Lemon-Grape classifier

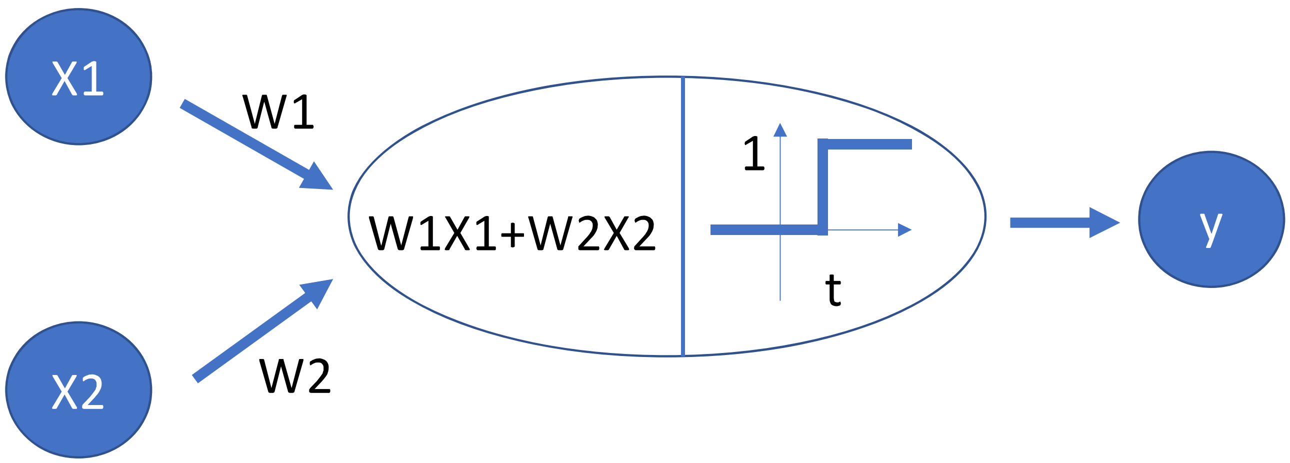
 → f() → $\begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix}$ → Grape

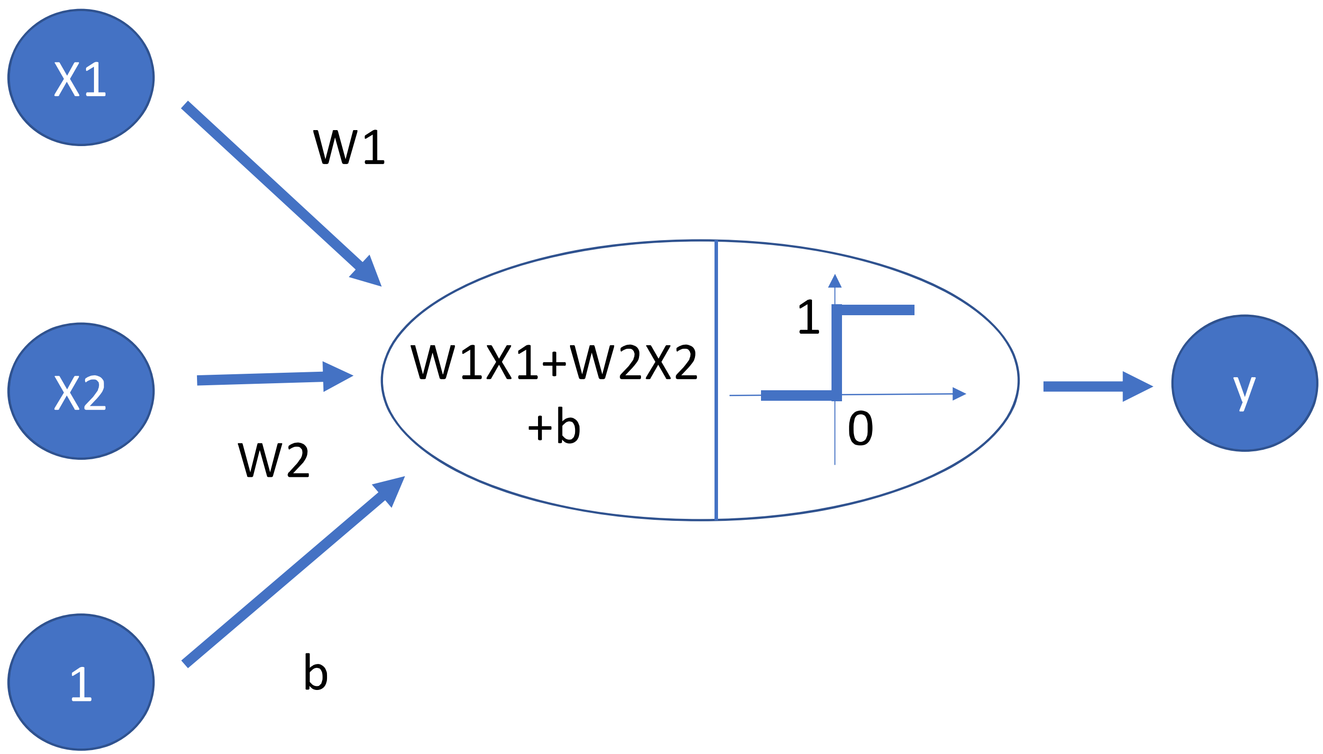
 → f() → $\begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}$ → Lemon

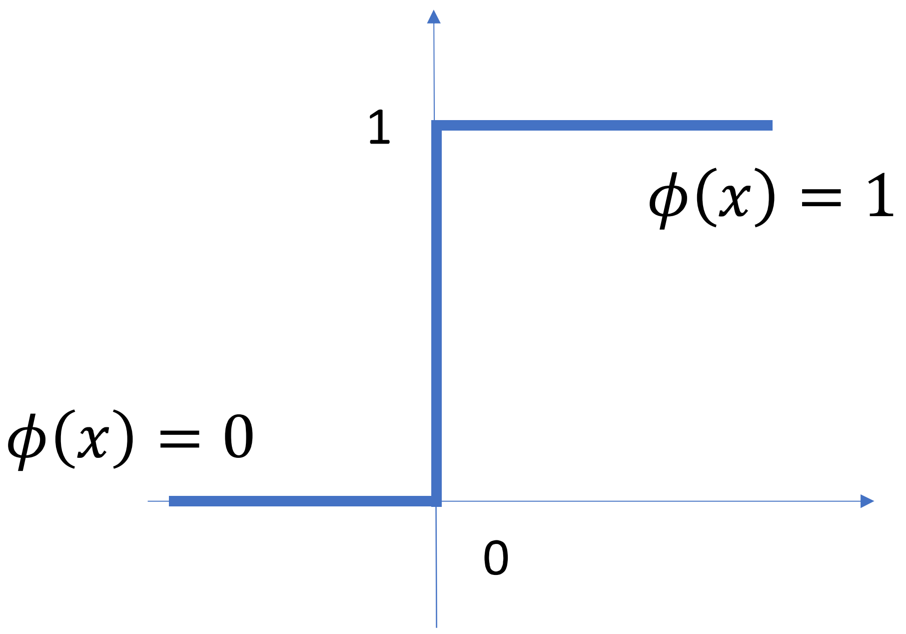
Perceptron, the basic of deep learning

- Perceptron is the minimum unit of the neural networks

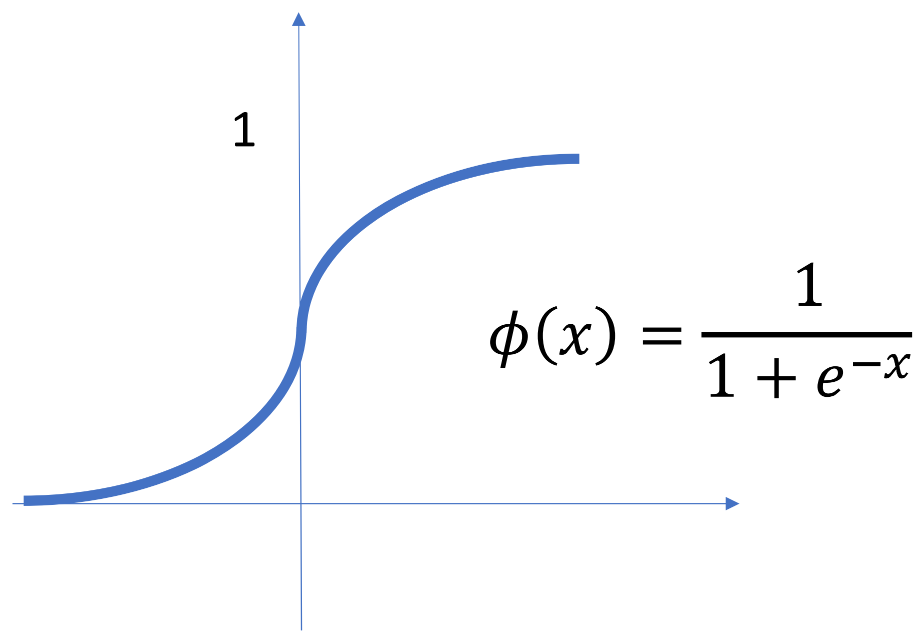



X1
X2
W1
W2
W1X1+W2X2
1
t
y


X1
X2
1
W1
W2
b
W1X1+W2X2
+b
1
0
y

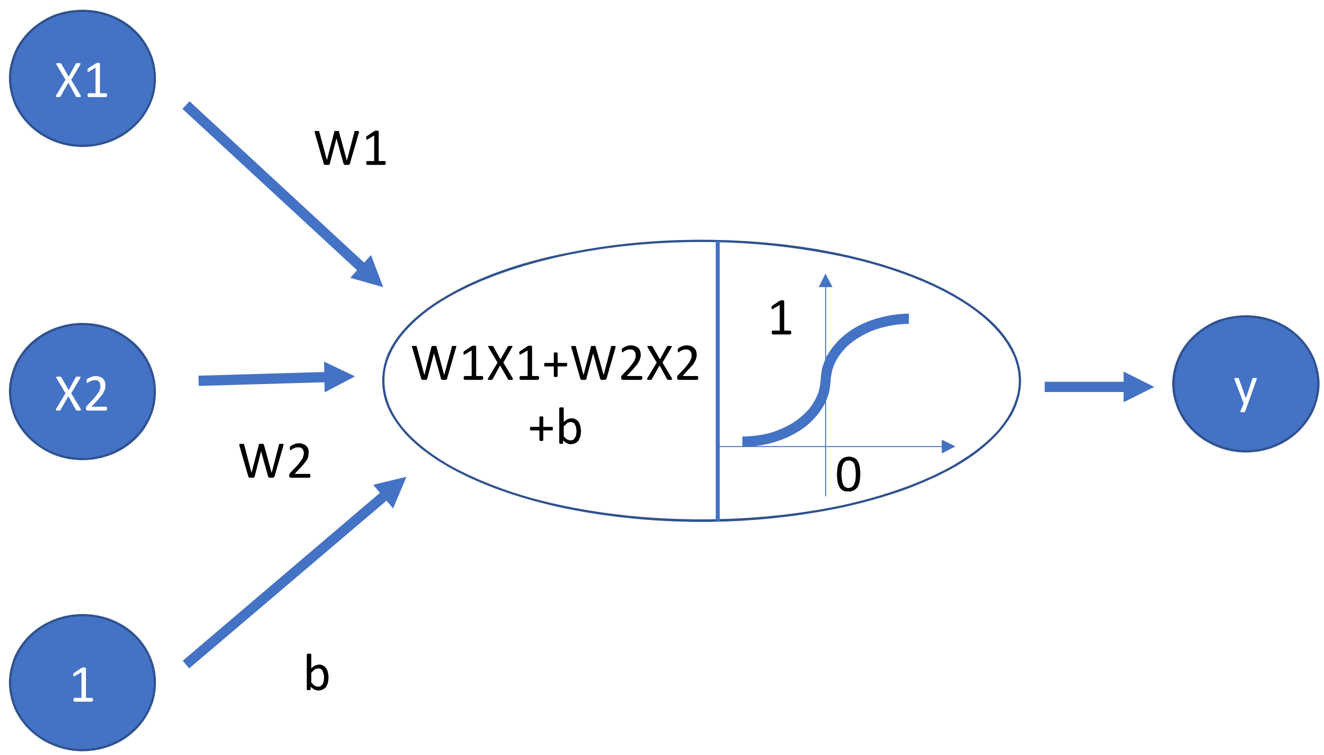


Step function



Sigmoid function

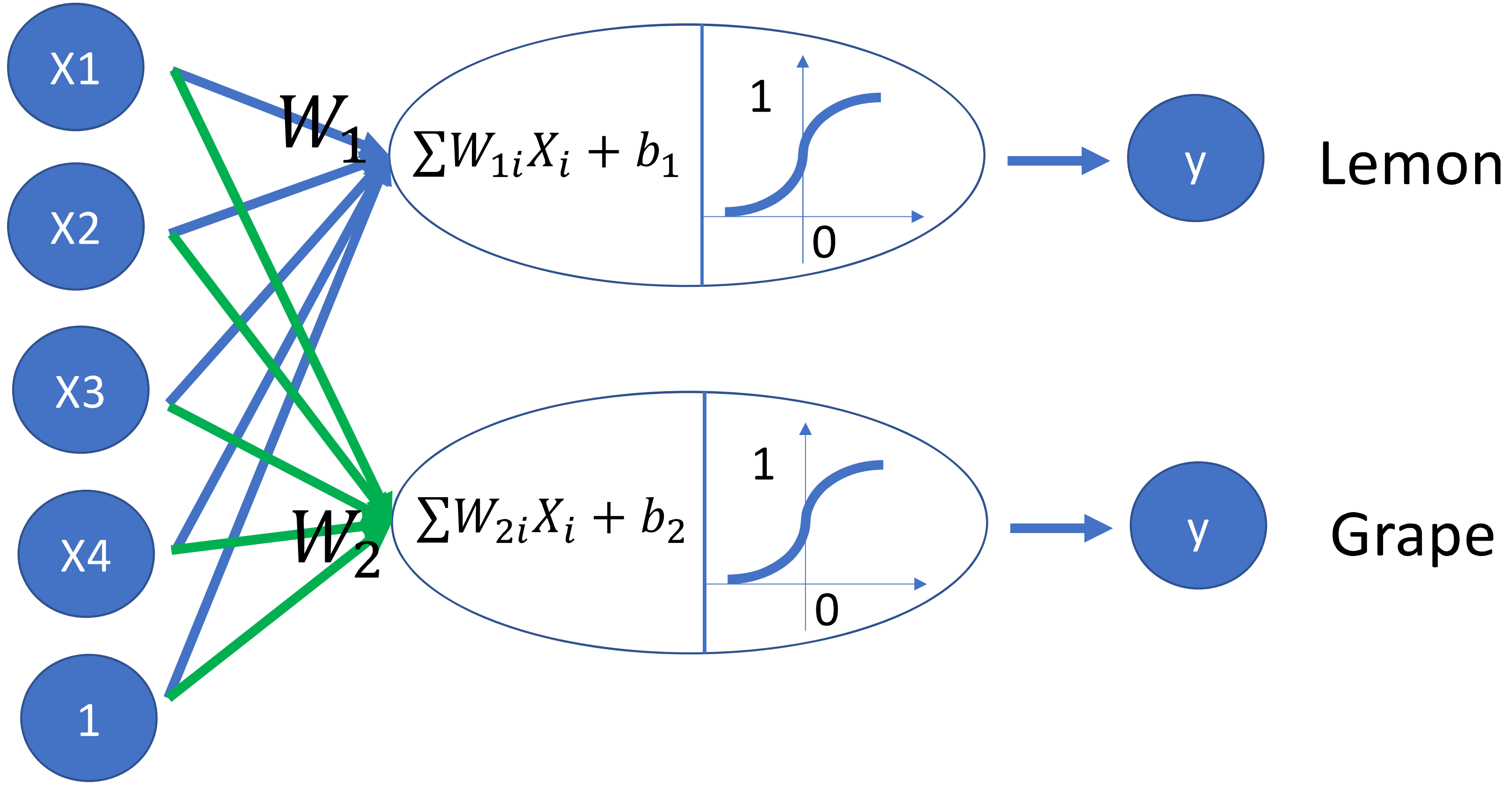
Perceptron with sigmoid activation



$$\begin{pmatrix} 2 & 1 & -3 & 3 \\ 1 & -3 & 1 & 3 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \\ 5 \\ 1 \end{pmatrix} + \begin{pmatrix} 3 \\ 3 \end{pmatrix} = \begin{pmatrix} -2 \\ 4 \end{pmatrix}$$

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

$$\begin{pmatrix} 0.119 \\ 0.982 \end{pmatrix}$$

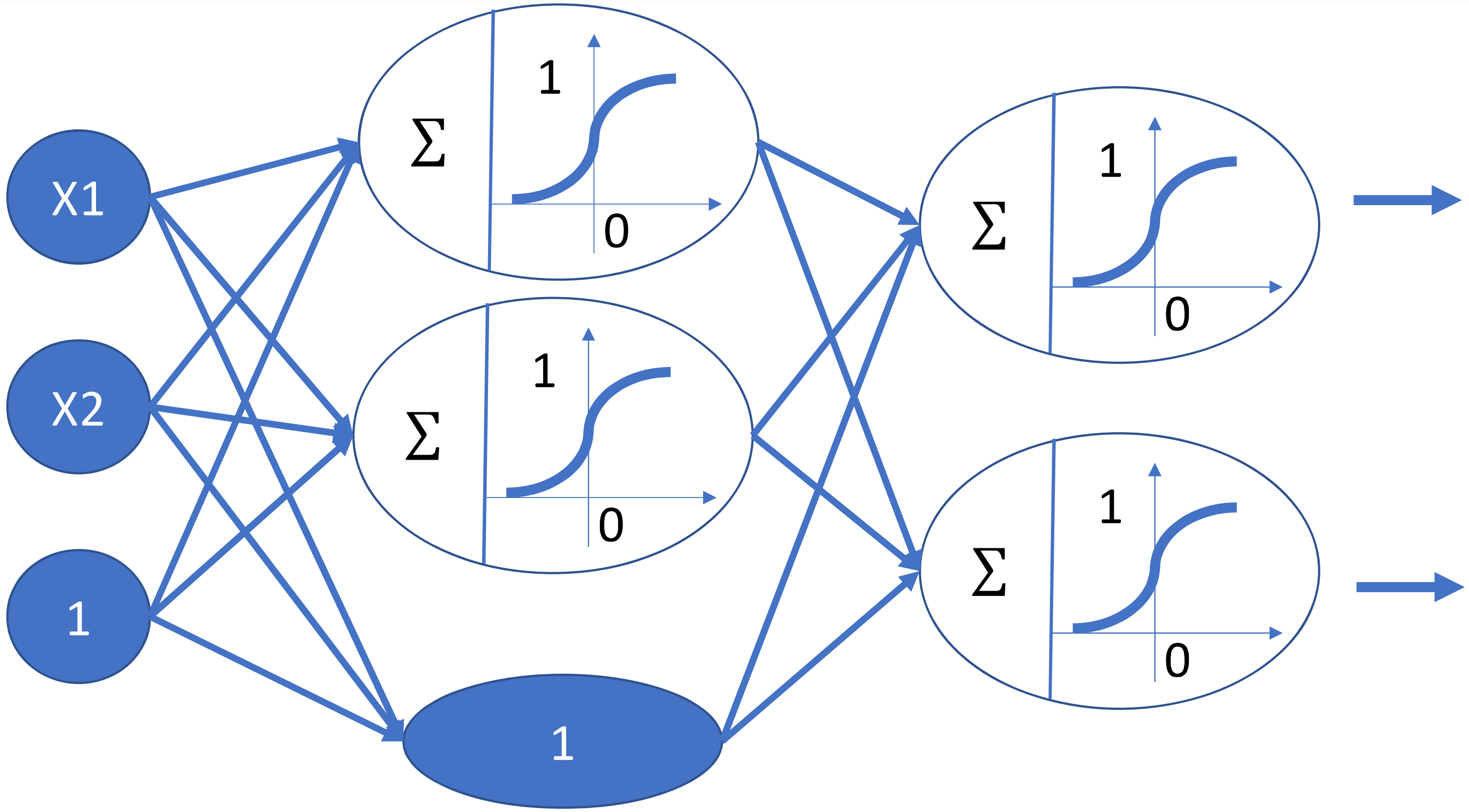


Exercise: Make perceptron

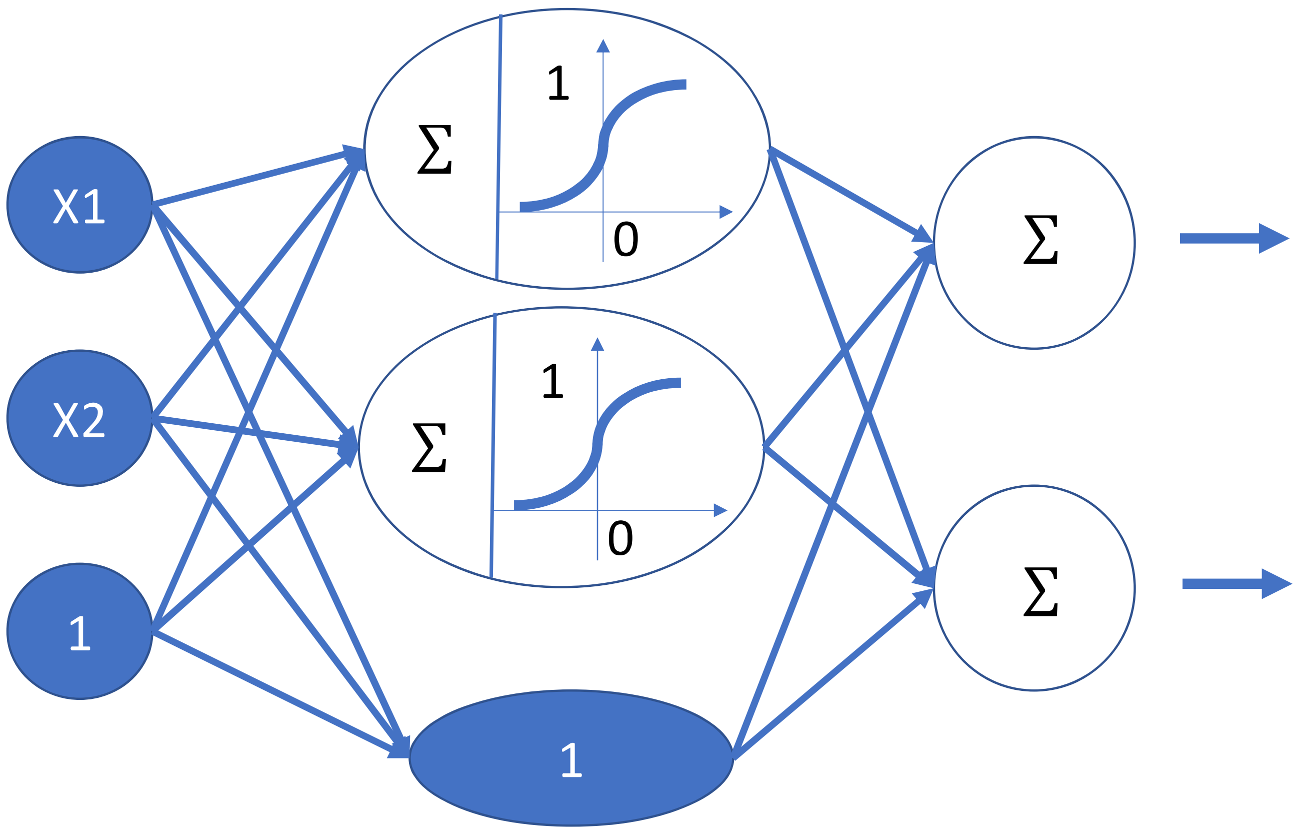
```
import numpy as np
 # input layer
Input_data =
np.array([[2,3],[5,1]])
x = input_data.reshape(-1)

# weight and bias
w1 = np.array([2,1,-3,3])
w2 = np.array([1,-3,1,3])
b1 = 3
b2 = 3
```

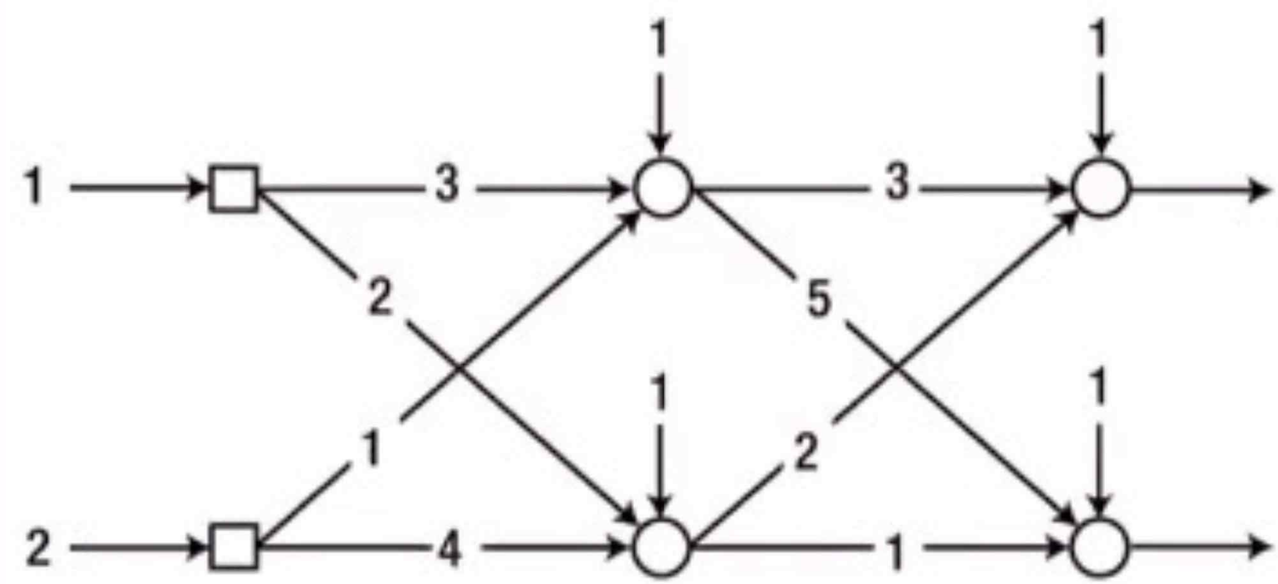
```
# sum of weight
W = np.array([w1,w2])
b = np.array([b1,b2])
weight_sum = np.dot(W,x) + b

#output layer
output = 1/(1+np.exp(-weight_sum))
```

The structure of the Artificial Neural Network



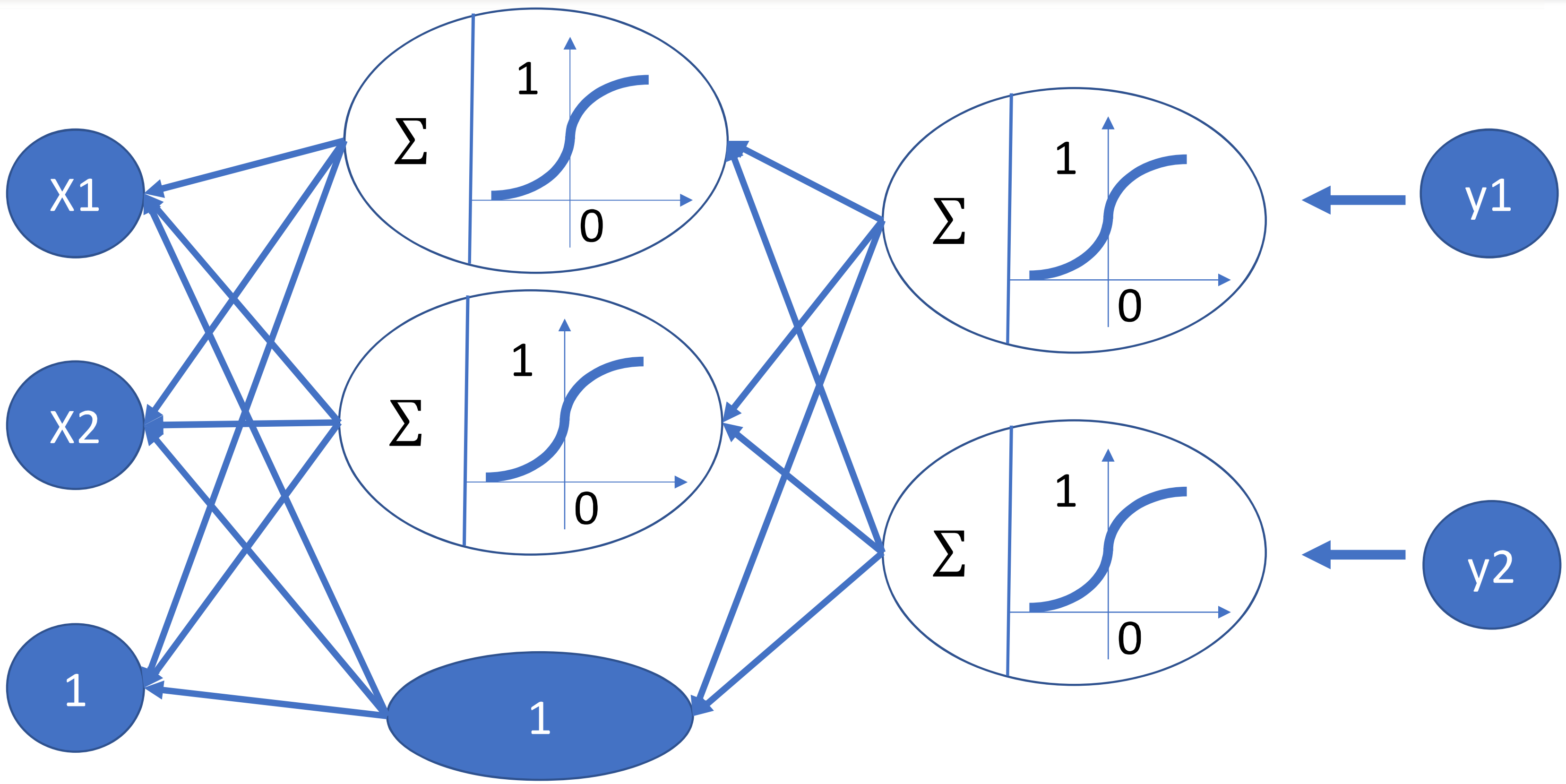
The structure of the Artificial Neural Network

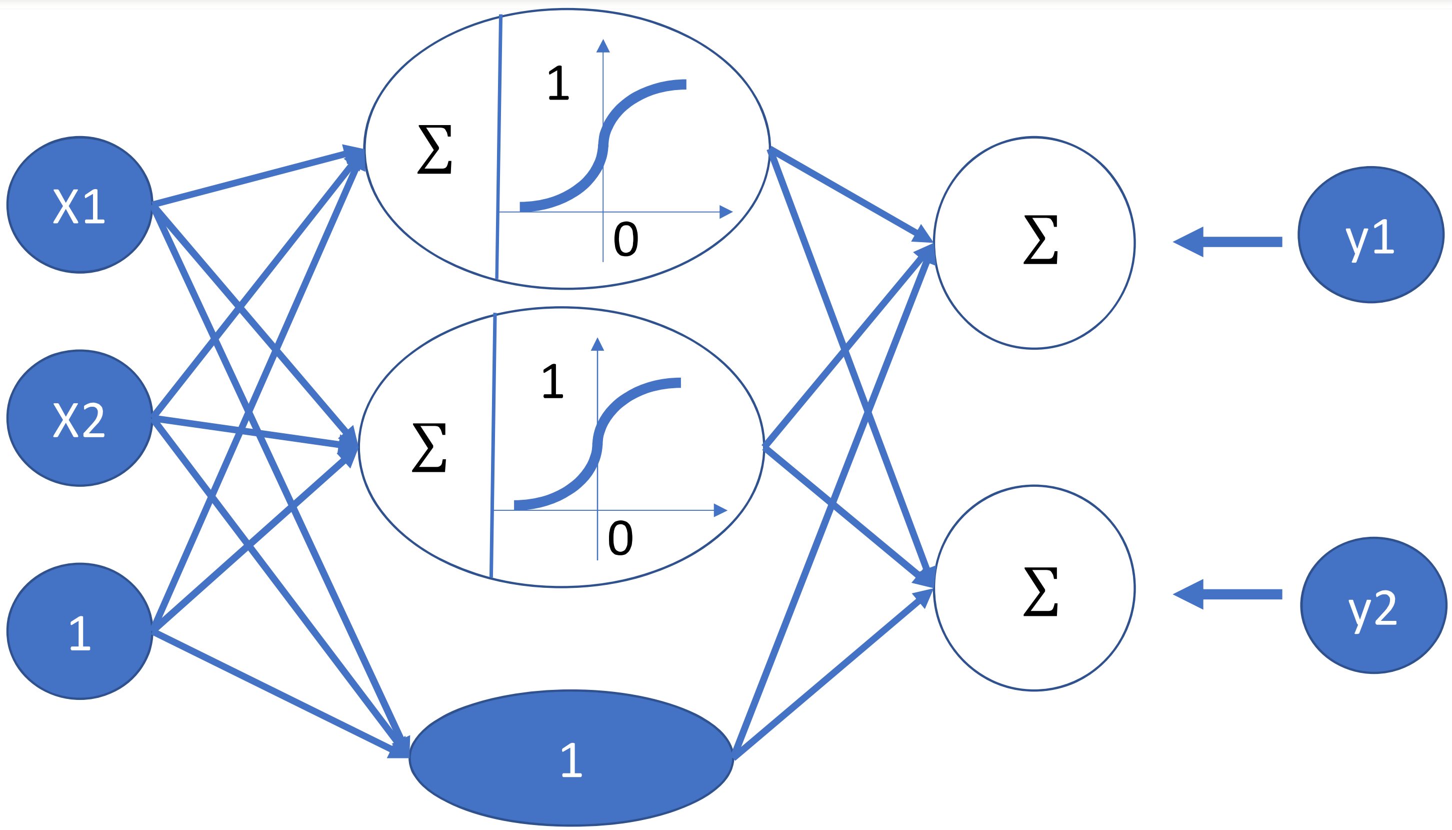




1
2
3
2
1
4
1
1
3
5
2
1
1
1

Error back propagation

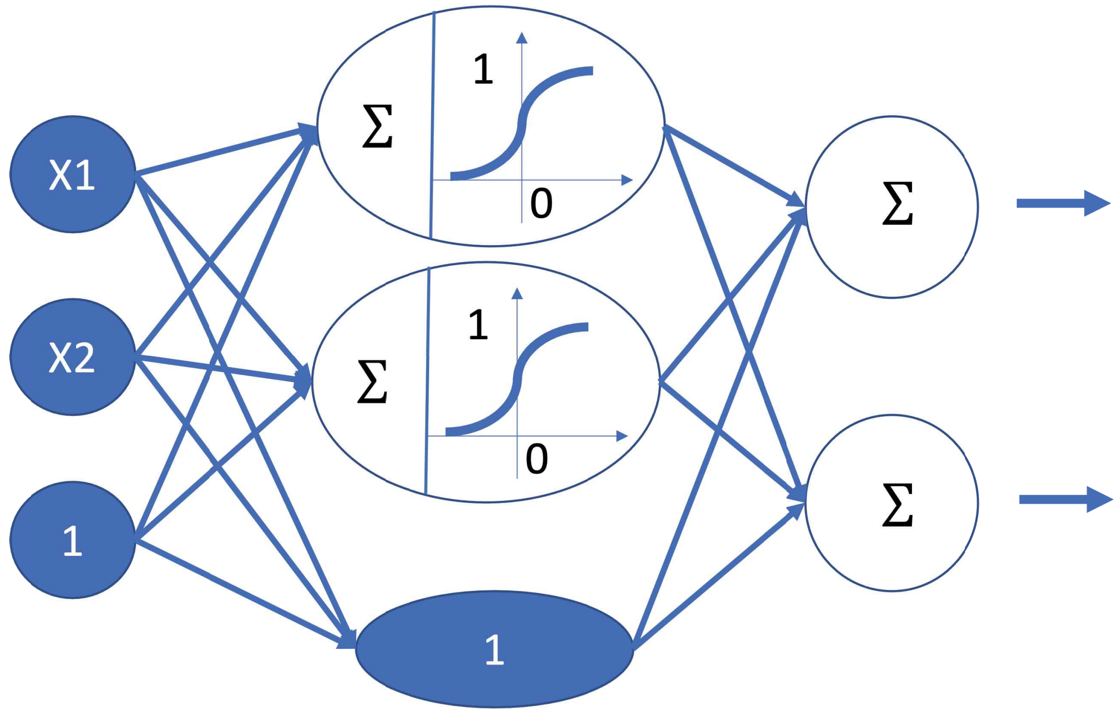



X1
X2
1
Σ
1
0
Σ
1
0
1
Σ
Σ
y1
y2

가중치 행렬 W와 입력데이터 x, 그것들의 곱의 결과를 h, h의 활성함수 값을 s, 최종 가중치 행렬 Wo, 그리고 최종 결과값을 y라고 정의하면 다음과 같이 나타낼 수 있다.

$$W*x=h \rightarrow S=\sigma(h) \rightarrow Wo*S=y$$

여기서 x는 n차원 벡터일 경우, 그리고 히든 노드의 개수를 m개로 설정할 경우, W는 $m \times n$차 행렬, h와s는 m차원 벡터가 된다. 결과치 y의 차원이 k차원일 경우, Wo는 $k \times m$차원 행렬이 된다.



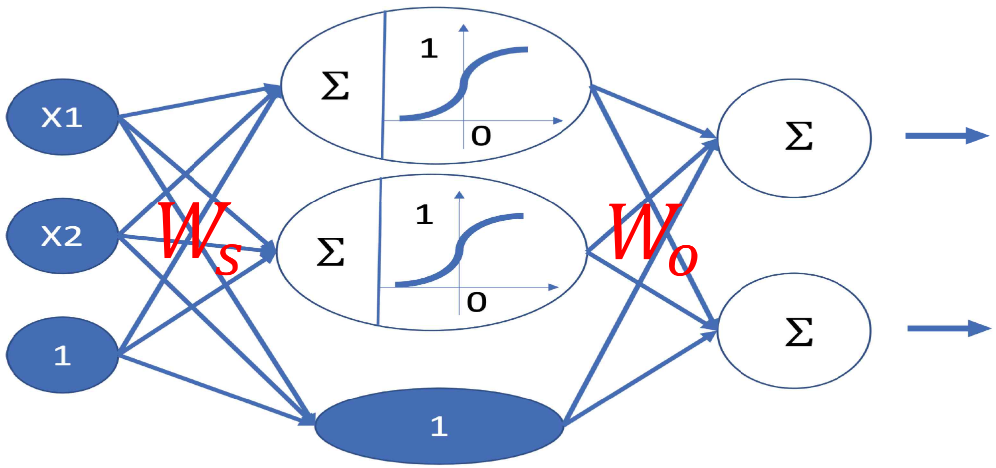
Batch 모드는 데이터셋을 적당한 분량으로 할당하여 해당 분량만큼 오류역전파를 계산하여 가중치를 업데이트 한다.

$$\Delta w_{ij} = \frac{1}{N}\sum_{k=1}^{N} \Delta w_{ij}(k)$$

w는 주어진 인공신경망의 가중치이고, 데이터의 개수가 N일 경우이다.

Stochastic 모드 방식의 업데이트 방식은 각 데이터에서 오류역전파를 계산하여 가중치를 곧바로 업데이트 한다.

$$\Delta w_{ij} = \Delta w_{ij}(k)$$



$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} W_s^{11} & W_s^{12} \\ W_s^{21} & W_s^{22} \\ W_s^{31} & W_s^{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$s_1 \rightarrow h(s_1)$$
$$s_2 \rightarrow h(s_2)$$
$$s_3 \rightarrow h(s_3)$$

$$\begin{bmatrix} o_1 \\ o_2 \end{bmatrix} = \begin{bmatrix} W_o^{11} & W_o^{12} & W_o^{13} \\ W_o^{21} & W_o^{22} & W_o^{23} \end{bmatrix} \begin{bmatrix} h(s_1) \\ h(s_2) \\ h(s_3) \end{bmatrix}$$

연쇄미분을 통한 오류역전파 계산은 행렬형식과 for-loop 형식으로 각각 계산해 볼 수 있다.

행렬형식으로 계산할 경우 cpu기준 더 빠른 계산을 진행해 볼 수 있으며, for-loop형식을 적용할 경우, 직관적으로 손쉽게 프로그래밍 할 수 있어서 작은 규모의 데이터를 학습할 경우 인공신경망의 계산 방식을 이해하는데 도움이 될 수 있다.

이해를 직관적으로 돕기위하여 y가 단순히 2차원 벡터라고 가정하면 손실함수 L는 다음과같이 표현할 수 있다.

$$L = \frac{1}{2}\sum_{k=1}^{N}\left|\left|o - y\right|\right|^2$$

$$= \frac{1}{2}\sum_{k=1}^{N}\left|\left|\begin{pmatrix} o_1 \\ o_2 \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}\right|\right|^2$$

$$= \frac{1}{2}\sum_{k=1}^{N}\left((o_1 - y_1)^2 + (o_2 - y_2)^2\right)$$

이제 y의 각 인덱스를 i로 설정하고 오류역전파를 구성하는 연쇄미분들을 구성하여 보자.

$$\frac{\partial L}{\partial o_i} = o_i - y_i$$

$$\frac{\partial o_i}{\partial s_j} = w_o^{ij} h(s_j)\left[1 - h(s_j)\right]$$

$$\frac{\partial s_j}{\partial w_s^{jk}} = x_k$$

$$\frac{\partial o_i}{\partial w_o^{ij}} = h(s_j)$$

$$\rightarrow \frac{\partial s_j}{\partial w_s^{jk}} \frac{\partial o_i}{\partial s_j} \frac{\partial L}{\partial o_i} = \frac{\partial L}{\partial w_s^{jk}}$$

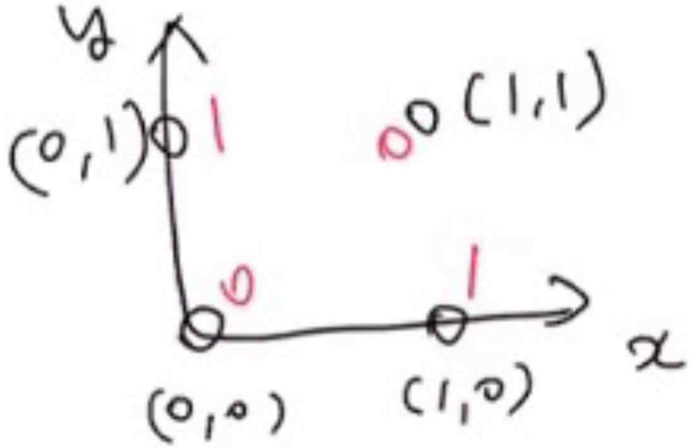
$$\frac{\partial o_i}{\partial w_o^{ij}} \frac{\partial L}{\partial o_i} = \frac{\partial L}{\partial w_o^{ij}}$$

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} W_s^{11} & W_s^{12} \\ W_s^{21} & W_s^{22} \\ W_s^{31} & W_s^{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\begin{matrix} s_1 \rightarrow h(s_1) \\ s_2 \rightarrow h(s_2) \\ s_3 \rightarrow h(s_3) \end{matrix}$$

$$\begin{bmatrix} o_1 \\ o_2 \end{bmatrix} = \begin{bmatrix} W_o^{11} & W_o^{12} & W_o^{13} \\ W_o^{21} & W_o^{22} & W_o^{23} \end{bmatrix} \begin{bmatrix} h(s_1) \\ h(s_2) \\ h(s_3) \end{bmatrix}$$

<XOR 문제>



XOR : fuction on domain(XY)

$$\begin{cases} f(0,0)=0 \\ f(1,0)=1 \\ f(0,1)=1 \\ f(1,1)=0 \end{cases}$$