Chap.1 심층 신경망 (Deep Neural Network)

방 수 식 교수

(bang@tukorea.ac.kr)

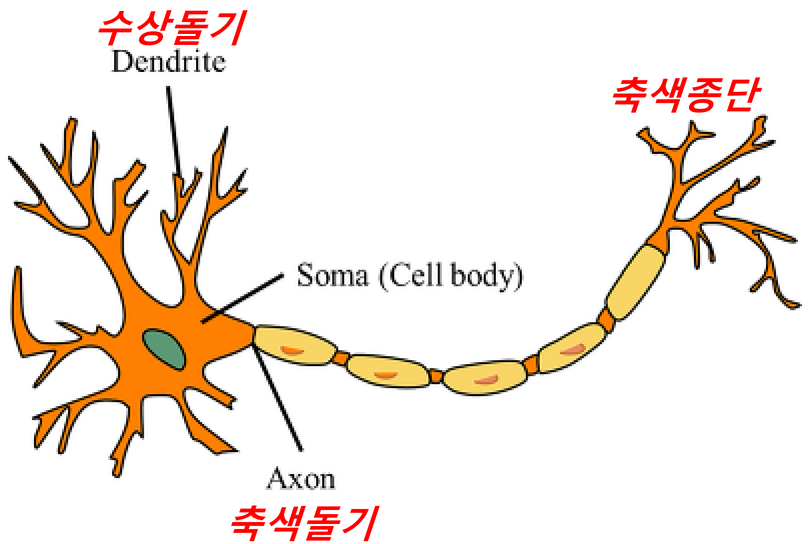
한국공학대학교 전자공학부

2025년도 2학기

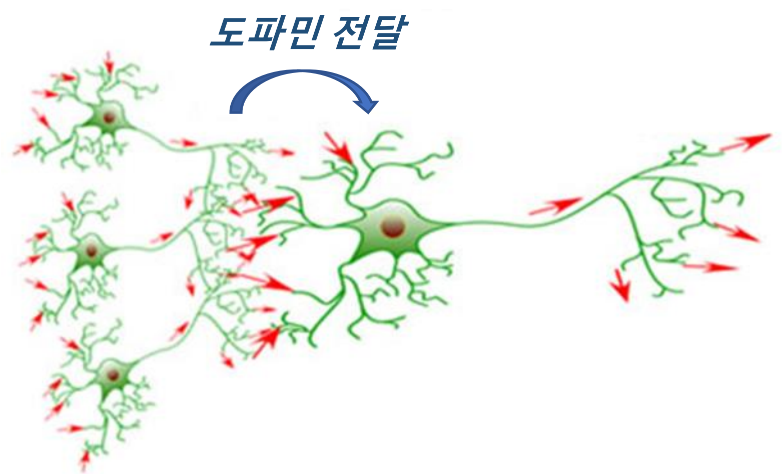
고급머신러닝실습 & 인공지능설계실습2

신경망 (Neural Network)

지능형 예측·진단 연구실
Intelligent Prognostics & Diagnostics Lab.



Neuron

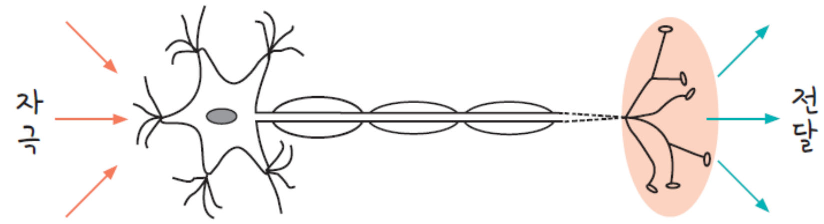


Neural Network in Brain

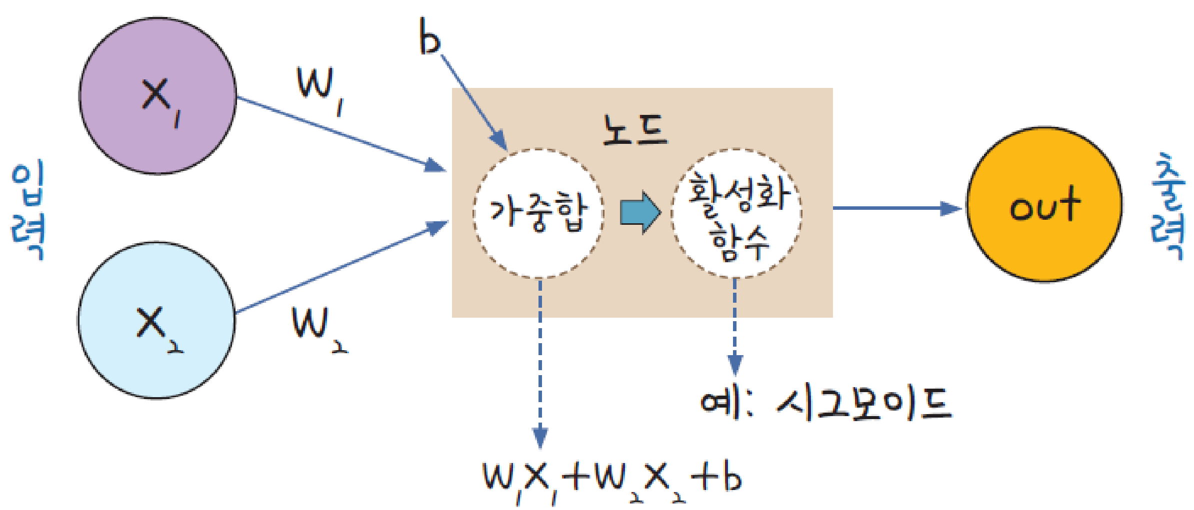
- 축색종단에서의 도파민(신경전달물질) 분비 여부
  - 전달된 신호의 강도에 따라 결정됨
  - 즉, 신호의 강도가 임계값을 넘으면 도파민 분비 ➔ **활성화**
  - 신호의 강도가 임계값을 넘지 않으면 도파민 분비되지 않음 ➔ **비활성화**

Artificial Neuron

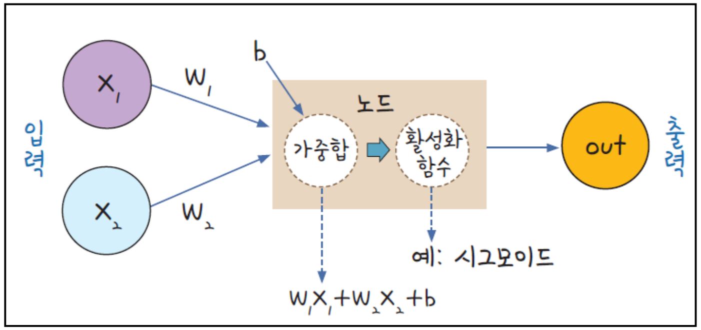
- **Neuron의 핵심 기능**
  - **전파(Propagation)** 기능
    - 신호(도파민)를 받아 전달
  - **활성화(Activation)** 기능
    - 신호의 강도에 따라 신호 활성화 유무를 결정

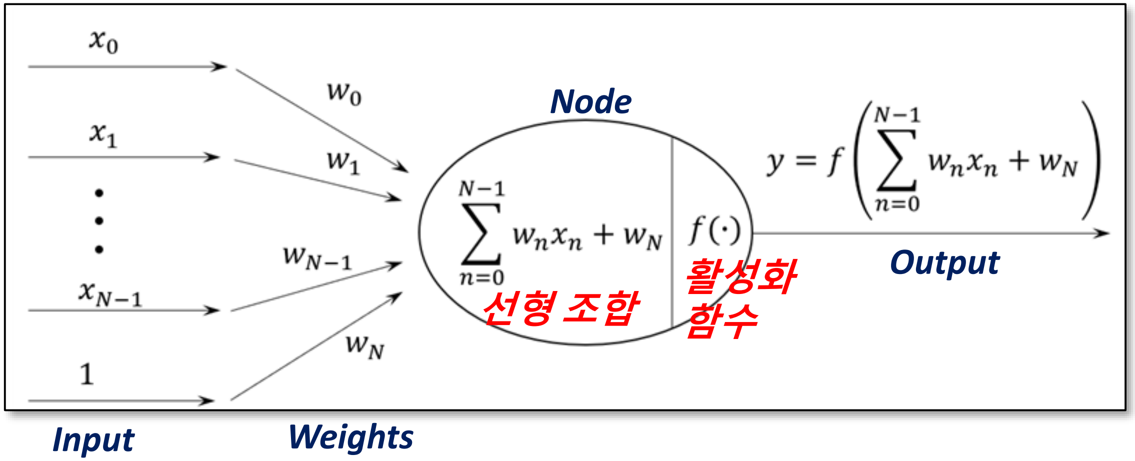


- **Artificial Neuron(인공 뉴런)**
  - Neuron 세포를 모방하여 설계



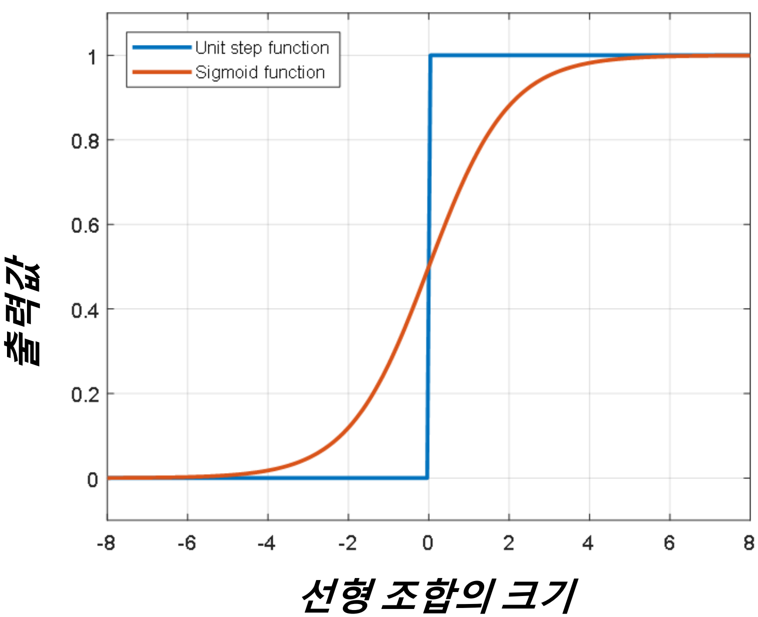
Artificial Neuron



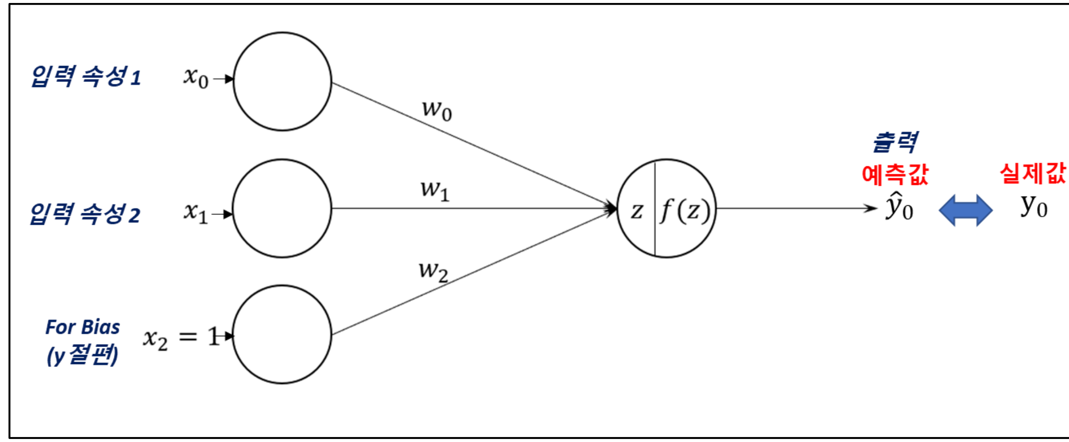


- **전파(Propagation)** 기능
  - 입력과 Weights 들의 선형 조합을 전파
- **활성화(Activation)** 기능
  - 선형 조합의 크기에 따라 다음 뉴런에 전달할 지 여부를 결정

Activation Functions, *f*



Perceptron

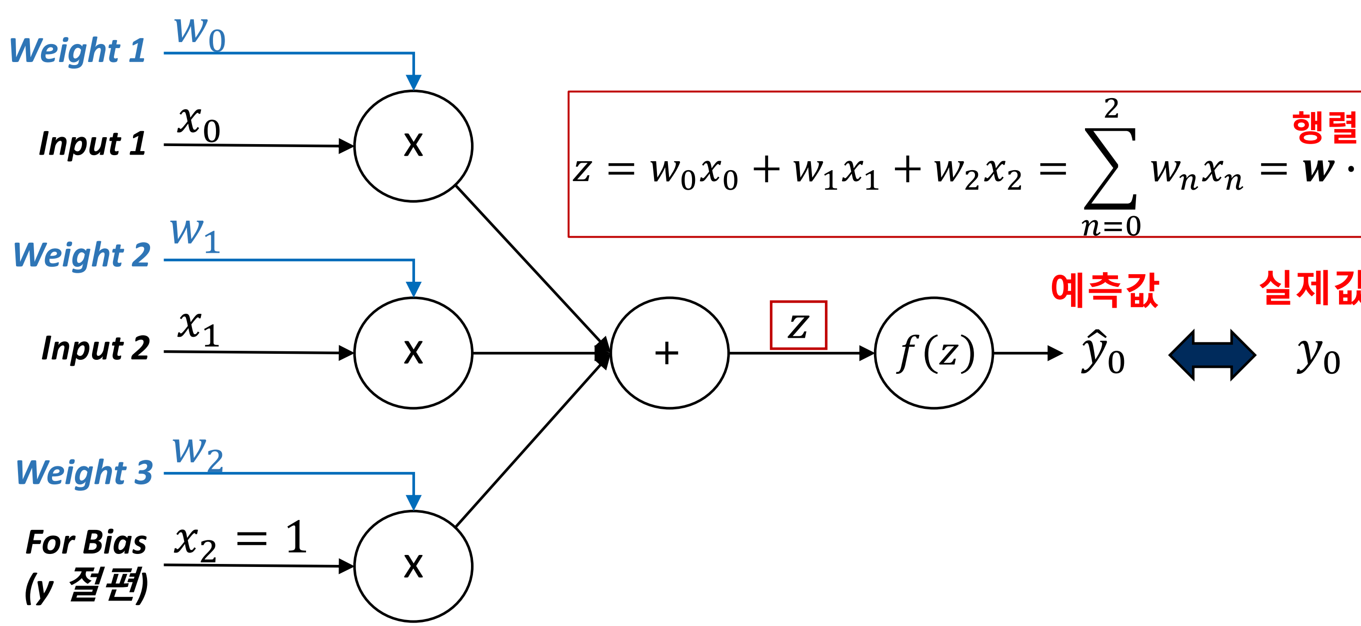


Weight, w

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

Input, x

$$[x_0 \quad x_1 \quad x_2]$$



행렬곱

$$z = w_0x_0 + w_1x_1 + w_2x_2 = \sum_{n=0}^{2} w_nx_n = \boldsymbol{w} \cdot \boldsymbol{x}$$

분류해야하는 Class가 3개 이상이라면?

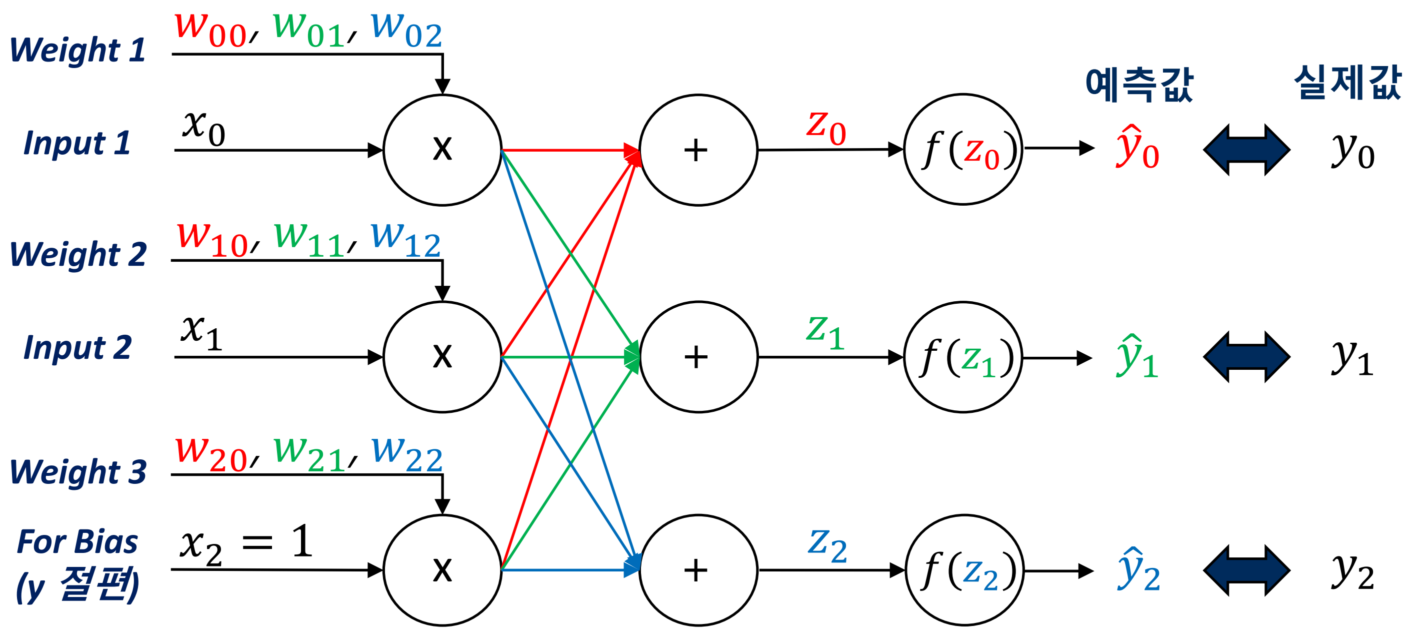
Multi-Class Classification in Single Layer

Input, x

$$[x_0 \quad x_1 \quad x_2]$$

Weight, w

$$\begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix}$$

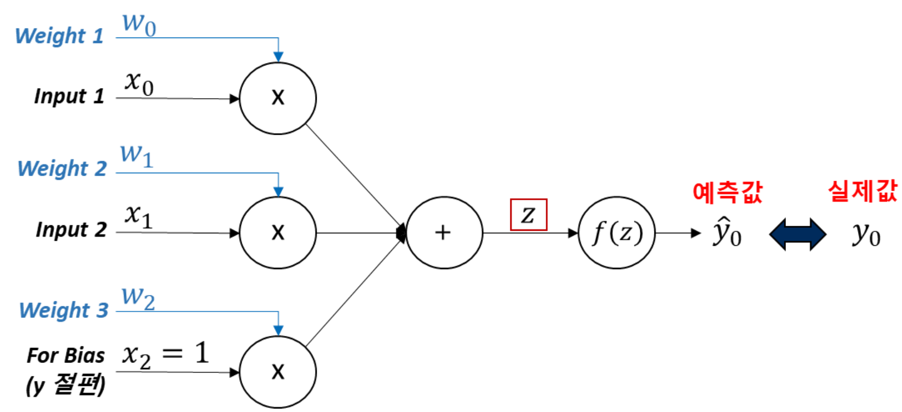


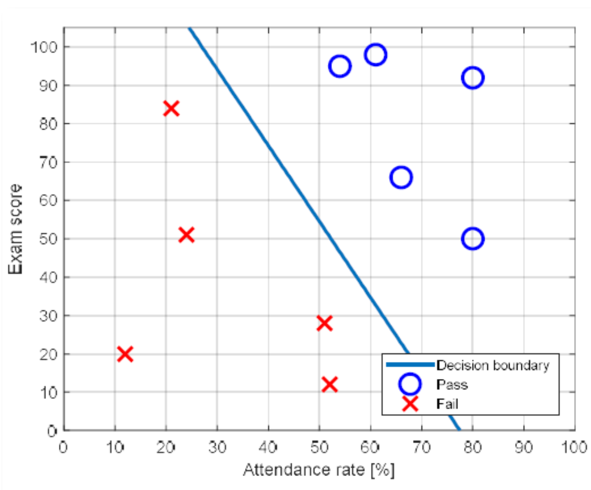
- **One-Hot Encoding (범주형 Class의 이진화)**
  - 확률론적 접근 + 기계가 인식 가능한 숫자
    - ✓ Binary Class: 음성 or 양성 => 0 or 1
    - ✓ Multi-Class: 개 or 고양이 or 말 => 100 or 010 or 001

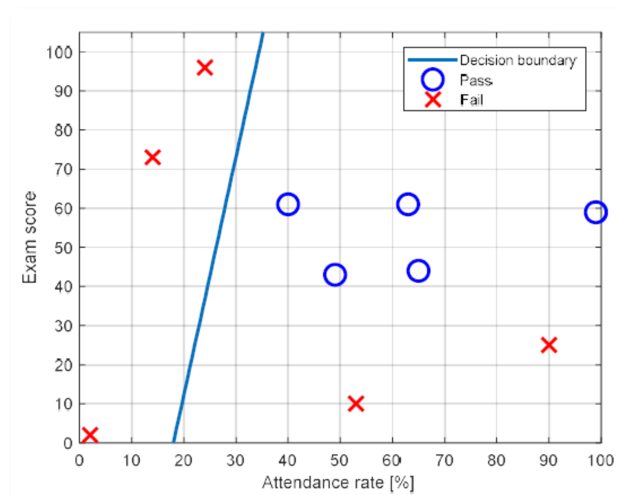
$$(y_0, y_1, y_2)$$

Single Layer Neural Network의 한계

- **Single Layer Neural Network의 한계**
  - 단층 신경망은 로지스틱 회귀의 성능과 동일
  - **<u>속성 공간의 "선형 분할"만 가능</u>**
    - ➢ **<u>Decision Boundary가 선형적</u>**







Multi-Layer Neural Network

Input, x

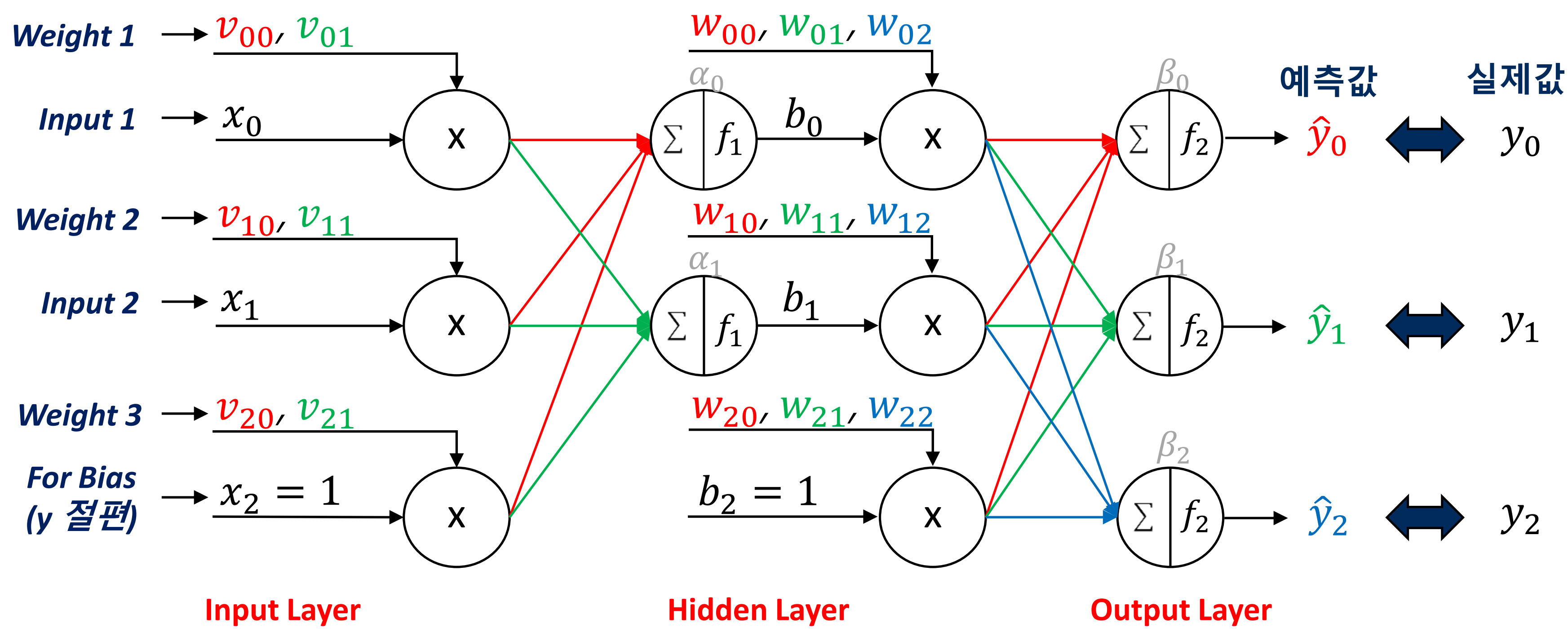
$$[x_0 \quad x_1 \quad x_2]$$

Weight, v

$$\begin{bmatrix} v_{00} & v_{01} \\ v_{10} & v_{11} \\ v_{20} & v_{21} \end{bmatrix}$$

Weight, w

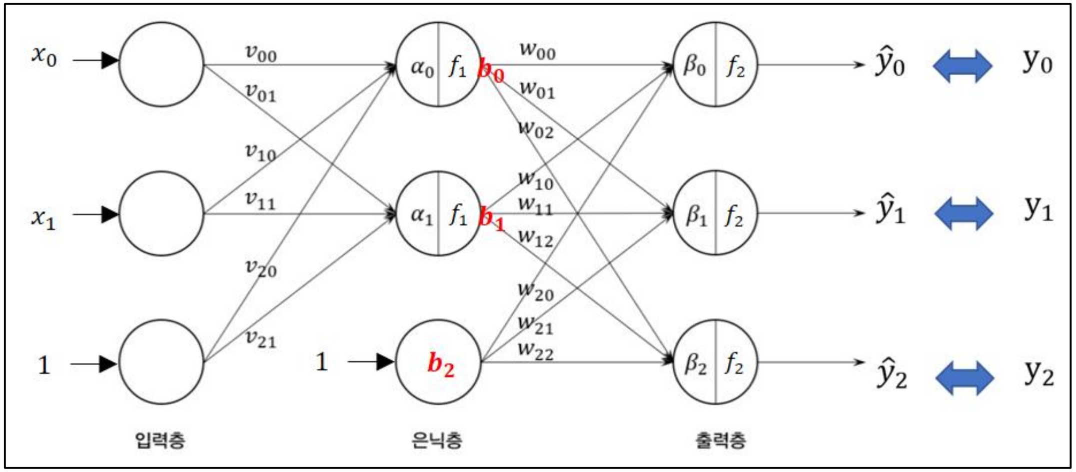
$$\begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix}$$



활성화 함수 이후 Bias 추가

$$[x_0 \quad x_1 \quad x_2] \cdot \begin{bmatrix} v_{00} & v_{01} \\ v_{10} & v_{11} \\ v_{20} & v_{21} \end{bmatrix} = [\,\alpha_0 \quad \alpha_1\,]\ ,\ [\,b_0 \quad b_1 \quad b_2\,] \cdot \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix} = [\,\hat{y}_0 \quad \hat{y}_1 \quad \hat{y}_2\,]$$

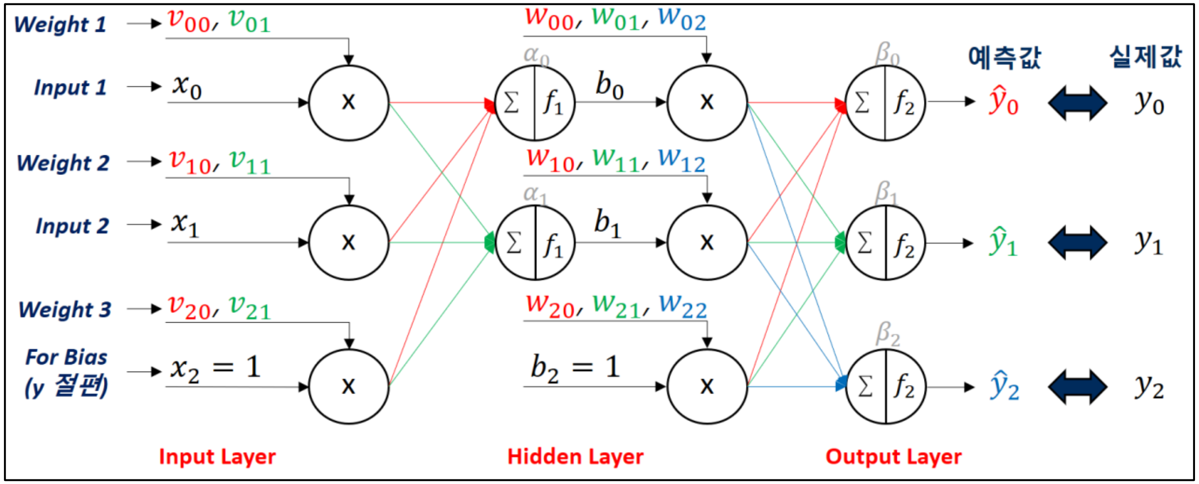
[참고] 인공신경망 설명 그림 변경



머신러닝실습에서 사용한
인공신경망 설명 그림

표현하는 방식이 다를 뿐,
같은 신경망을 표기한 것

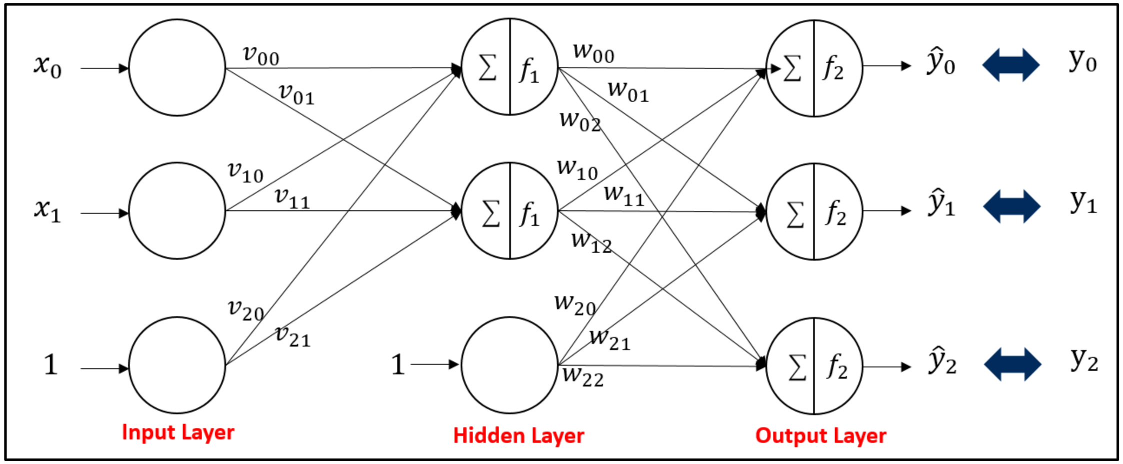
고급머신러닝실습에서 사용하는
인공신경망 설명 그림



Neural Network 모델의 학습 알고리즘

지능형 예측·진단 연구실
Intelligent Prognostics & Diagnostics Lab.

- **Error Backpropagation (오차 역전파) 알고리즘**
  - 경사하강법 기반
  - 개별 훈련 데이터에 대해서 알고리즘 적용
  - 설정한 Cost Function로부터 모델의 예측값과 실측값 간의 Error 값으로부터 역으로 전파되면서 Weight를 업데이트함
    - ✓ 본 강의자료에서는 Activation Function은 Sigmoid, Cost Function은 MSE 기준으로 설명



머신러닝 실습에서 사용한 인공신경망 설명 그림

$$\frac{\partial \epsilon_{MSE}(n)}{\partial w_{lq}} = 2(\hat{y}_{qn} - y_{qn})\,\hat{y}_{qn}(1-\hat{y}_{qn})\,b_{ln} = \delta_{qn} b_{ln}$$

$$\frac{\partial \epsilon_{MSE}(n)}{\partial v_{ml}} = \sum_{q=0}^{Q-1} \delta_{qn} w_{lq}\, b_{ln}(1-b_{ln}) x_{mn}$$

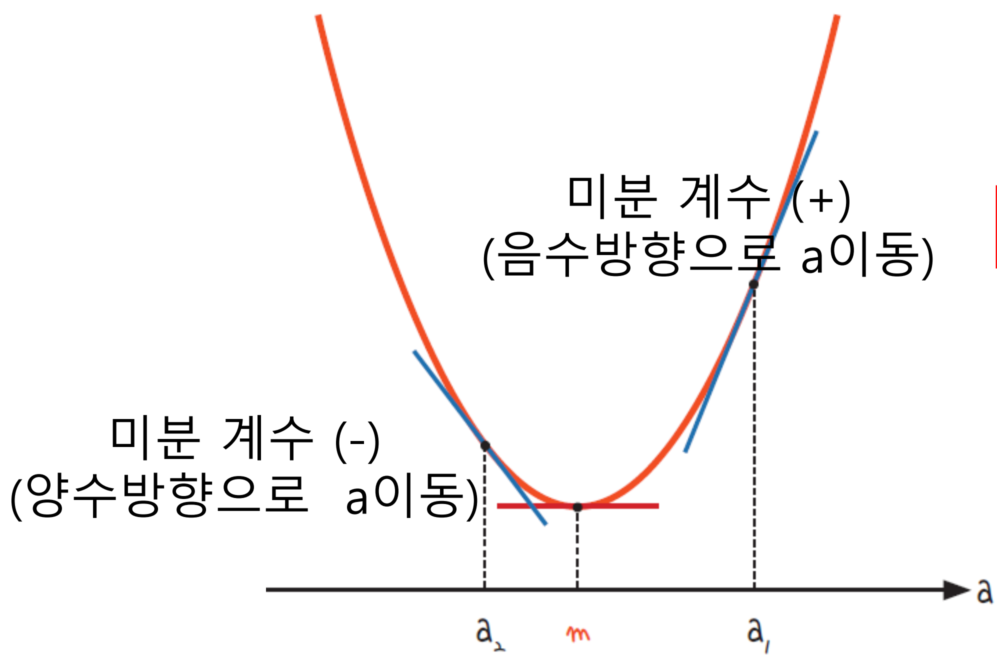
"머신러닝의 이해" 강의에서 도출한 Weight의 미분

인공신경망의 Hidden Layer를 늘리라고 한다면, 직접 만들 수 있나요?

[참고] 경사하강법

- 최소값에서 접선의 기울기에 해당하는 미분 계수는 0
  - ➢ **여러 번의 반복 (iteration)을 통해 미분 계수가 0이 되는 지점을 찾는다.**

- 미분 계수(Gradient)는 언제나 현재 위치에서 함수값이 커지는 방향을 가리키므로, **미분 계수의 반대 방향으로 이동**하면 최솟값을 찾을 수 있음
  - ➢ 즉 Gradient에 마이너스(-)를 취하여 더한다.



Learning Rate

$$\overset{\text{New}}{\boxed{w_0[t+1]}} = \overset{\text{Old}}{\boxed{w_0[t]}} - \boxed{\alpha}\frac{\partial}{\partial w_0}\epsilon_{MSE}\Big(\overset{\text{Old}}{\boxed{w_0, w_1}}\Big)$$

$$w_1[t+1] = w_1[t] - \alpha\frac{\partial}{\partial w_1}\epsilon_{MSE}(w_1, w_2)$$

Error Back-Propagation 알고리즘

지능형 예측·진단 연구실
Intelligent Prognostics & Diagnostics Lab.

- $\frac{\partial}{\partial v_{ml}} \epsilon_{MSE}(n)$ 구하기

$$\frac{\partial}{\partial v_{ml}} \epsilon_{MSE}(n) = \frac{\partial \epsilon_{MSE}(n)}{\partial b_{ln}} \cdot \frac{\partial b_{ln}}{\partial \alpha_{ln}} \cdot \frac{\partial \alpha_{ln}}{\partial v_{ml}}$$

우변 1항

$$\frac{\partial \epsilon_{MSE}(n)}{\partial b_{ln}} = \frac{\partial}{\partial b_{ln}} \sum_{q=0}^{Q-1} (\hat{y}_{qn} - y_{qn})^2 = 2 \sum_{q=0}^{Q-1} (\hat{y}_{qn} - y_{qn}) \frac{\partial \hat{y}_{qn}}{\partial b_{ln}}$$

$$= 2 \sum_{q=0}^{Q-1} (\hat{y}_{qn} - y_{qn}) \frac{\partial}{\partial b_{ln}} f\left(\sum_{j=0}^{L} w_{jq} b_{jn}\right)$$

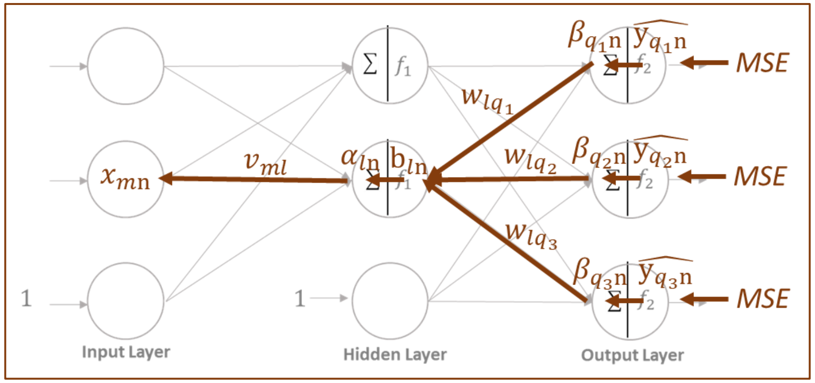
$$= 2 \sum_{q=0}^{Q-1} (\hat{y}_{qn} - y_{qn}) \frac{\partial}{\partial b_{ln}} f(w_{0q} b_{0n} + \cdots + w_{lq} b_{ln} + \cdots + w_{Lq} b_{Ln})$$

$$= 2 \sum_{q=0}^{Q-1} (\hat{y}_{qn} - y_{qn}) f\left(\sum_{j=0}^{L} w_{jq} b_{jn}\right)\left(1 - f\left(\sum_{j=0}^{L} w_{jq} b_{jn}\right)\right) w_{lq}$$

$$= 2 \sum_{q=0}^{Q-1} (\hat{y}_{qn} - y_{qn}) \hat{y}_{qn} (1 - \hat{y}_{qn}) w_{lq}$$

$$= \sum_{q=0}^{Q-1} \delta_{qn} w_{lq}$$

l 번째 Hidden Node와 연결된 weight들과 Output Node를 고려하게 됨.



인공신경망 Weight Update Code

- **NN_data 데이터 취득**

```python
import numpy as np
import pandas as pd

file_name ='NN_data.csv'
load_data = np.array(pd.read_csv(file_name, index_col = 0))

x_data = load_data[:,0:3]
y_data = load_data[:,-1]

N = len(load_data) # 총 데이터수
M = len(x_data.T) # 특징수 = 3
set_y = list(set(y_data))
Q =  len(set_y) # 클래스 수 = 3
```

- **One-Hot Encoding**

```python
#One_hot
y_target_set = np.zeros([N,Q])

for n in range(N):
    if y_data[n] == set_y[0]:
        y_target_set[n,0] = 1
    elif y_data[n] == set_y[1]:
        y_target_set[n,1] = 1
    elif y_data[n] == set_y[2]:
        y_target_set[n,2] = 1
```

y_target_set

	0	1	2
0	1	0	0
300	0	1	0
600	0	0	1

인공신경망 Weight Update Code

- **Forward propagation**

```
#forward_propagation
L = int(10*np.random.rand(1) + 2) #2~12 랜덤

ini_W = np.random.randn(Q,L+1) #W 초기화
ini_V = np.random.randn(L,M+1) #V 초기화


n=0 #n번째 데이터에 대해 수행
W = ini_W
V = ini_V

temp_x = x_data[n,:] #n번째 x 데이터
temp_x = np.concatenate((temp_x,1), axis = None) #bias 추가
temp_x = resize_col(temp_x) # 행렬곱을 위한 resize

temp_alpha = np.dot(V, temp_x) #행렬곱
temp_b = sigmoid(temp_alpha) #활성화 함수
temp_b = np.concatenate((temp_b,1), axis = None) #bias 추가
temp_b = resize_col(temp_b) # 행렬곱을 위한 resize

temp_beta = np.dot(W, temp_b) #행렬곱

y_hat = sigmoid(temp_beta) #활성화 함수
```

```
#resize 함수 (k,) --> (k,1)
def resize_col(input):
    output = np.resize(input, (len(input),1))
    return output
```

```
#시그모이드 함수 정의
def sigmoid(x):
    return 1/ (1 + np.exp(-x))
```

인공신경망 Weight Update Code

지능형 예측·진단 연구실
Intelligent Prognostics & Diagnostics Lab.

- **Back propagation**

```
#back_propagation
y_target = y_target_set[n,:] #n번째 y 데이터

y_target = resize_col(y_target) #resize
delta = 2 * (y_hat - y_target[n]) * y_hat * (1 - y_hat) #delta

learning_rate = 0.01

for l in range(L):
    for m in range(M+1):
        w_l = W[:,l]
        w_l = resize_col(w_l)
        b_l = temp_b[l]
        x_m = temp_x[m]

        diff_ml = np.sum(delta * w_l * b_l * (1-b_l) * x_m)

        v_ml_old = V[l,m]
        v_ml_new = v_ml_old - learning_rate * diff_ml
        V[l,m] = v_ml_new

for q in range(Q):
    for l in range(L+1):
        delta_q = delta[q]
        b_l = temp_b[l]

        diff_lq = delta_q * b_l
        w_lq_old = W[q,l]
        w_lq_new = w_lq_old - learning_rate * diff_lq
        W[q,l] = w_lq_new
```

$$\frac{\partial \epsilon_{MSE}(n)}{\partial w_{lq}} = 2\left(\hat{y}_{qn} - y_{qn}\right)\hat{y}_{qn}\left(1 - \hat{y}_{qn}\right) b_{ln} = \delta_{qn} b_{ln}$$

$$\frac{\partial \epsilon_{MSE}(n)}{\partial v_{ml}} = \sum_{q=0}^{Q-1} \delta_{qn} w_{lq}\, b_{ln}(1 - b_{ln}) x_{mn}$$

for문 사용하지 않고 하려면?

계층 수를 늘리려면?

Back-Propagation for DNN

지능형 예측·진단 연구실
Intelligent Prognostics & Diagnostics Lab.

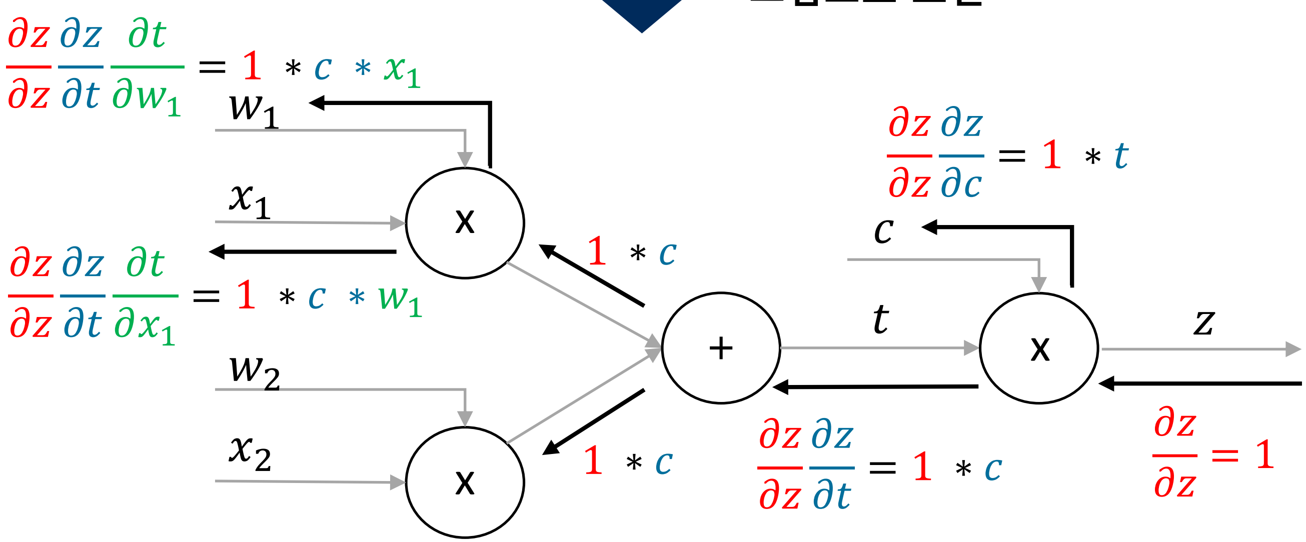
- **연쇄법칙의 이해**
  - 합성함수 : 여러 함수로 구성된 함수
  - 연쇄법칙 : 합성함수의 미분은 **각 함수의 미분의 곱**으로 나타낼 수 있음

ex) $z = c(w_1x_1 + w_2x_2)$ 일 때, x_1 에 대한 z의 미분값은?

$t =$

$z =$, $\frac{\partial t}{\partial x_1} =$, $\frac{\partial z}{\partial t} =$, $\frac{\partial z}{\partial x_1} =$

그림으로 표현

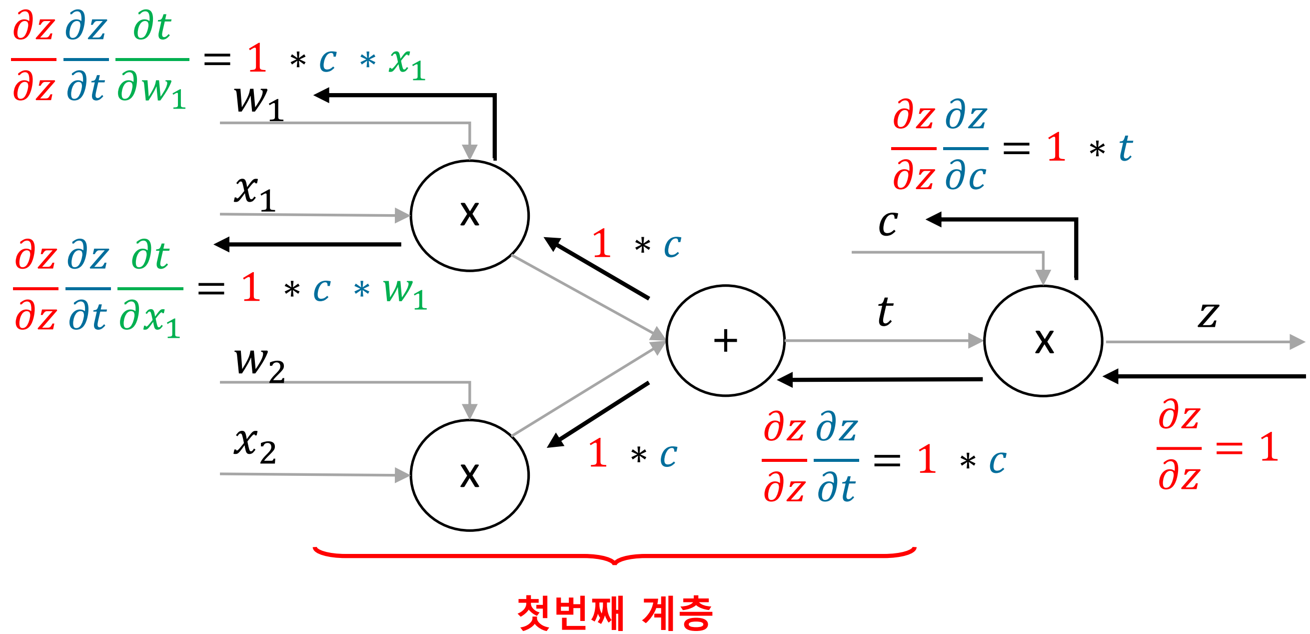


Back-Propagation for DNN

첫번째 계층 변수

$\mathbf{X} = [x_1, x_2]$

$$\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$



▪ 첫번째 계층 forward

$t = \mathrm{X} \cdot \mathrm{W}$

ex) X(2,3) · W(3,4) = output(2,4)

▪ 첫번째 계층 backward

legacy : Loss부터 직전까지의 미분값

$\mathrm{dX} = \mathrm{legacy} \cdot \mathrm{W}^{\mathrm{T}}$

$\mathrm{dW} = \mathrm{X}^{\mathrm{T}} \cdot \mathrm{legacy}$

$$\boxed{\mathrm{legacy} = \frac{\partial z}{\partial z}\frac{\partial z}{\partial t} = 1 * c}$$

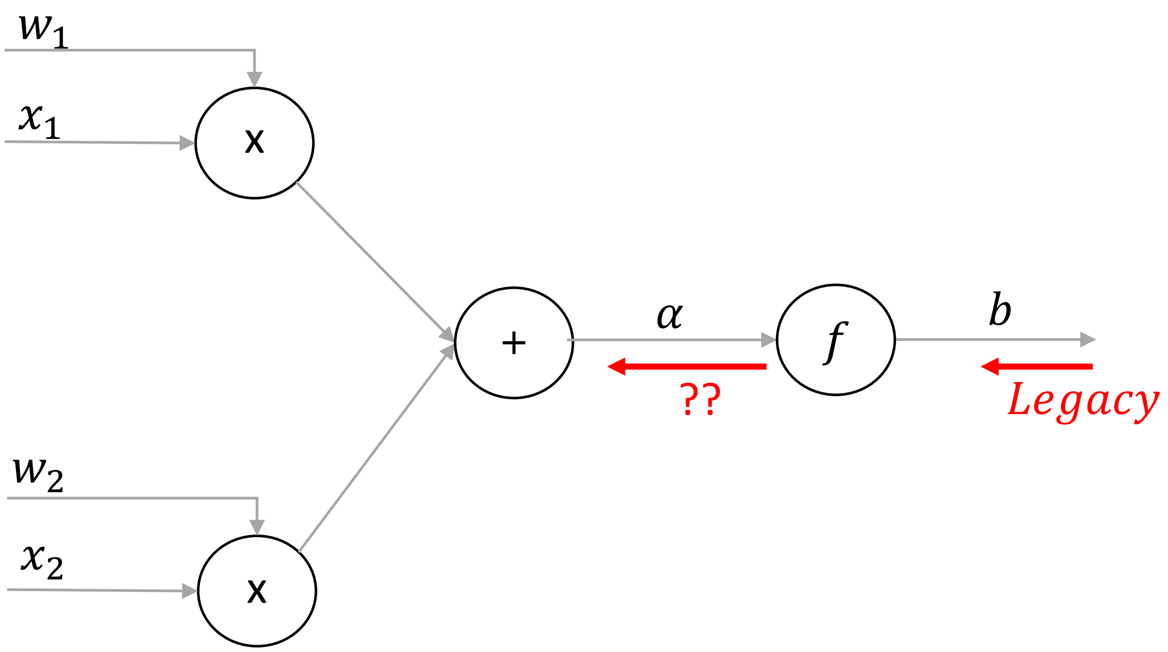
ex) legacy(2,4) · W^{T}(4,3) = dX(2,3)

ex) X^{T}(3,2) · legacy(2,4) = dW(3,4)

dW로 Weight 업데이트, dX는 다음 계층의 legacy로 동작

Back-Propagation for DNN

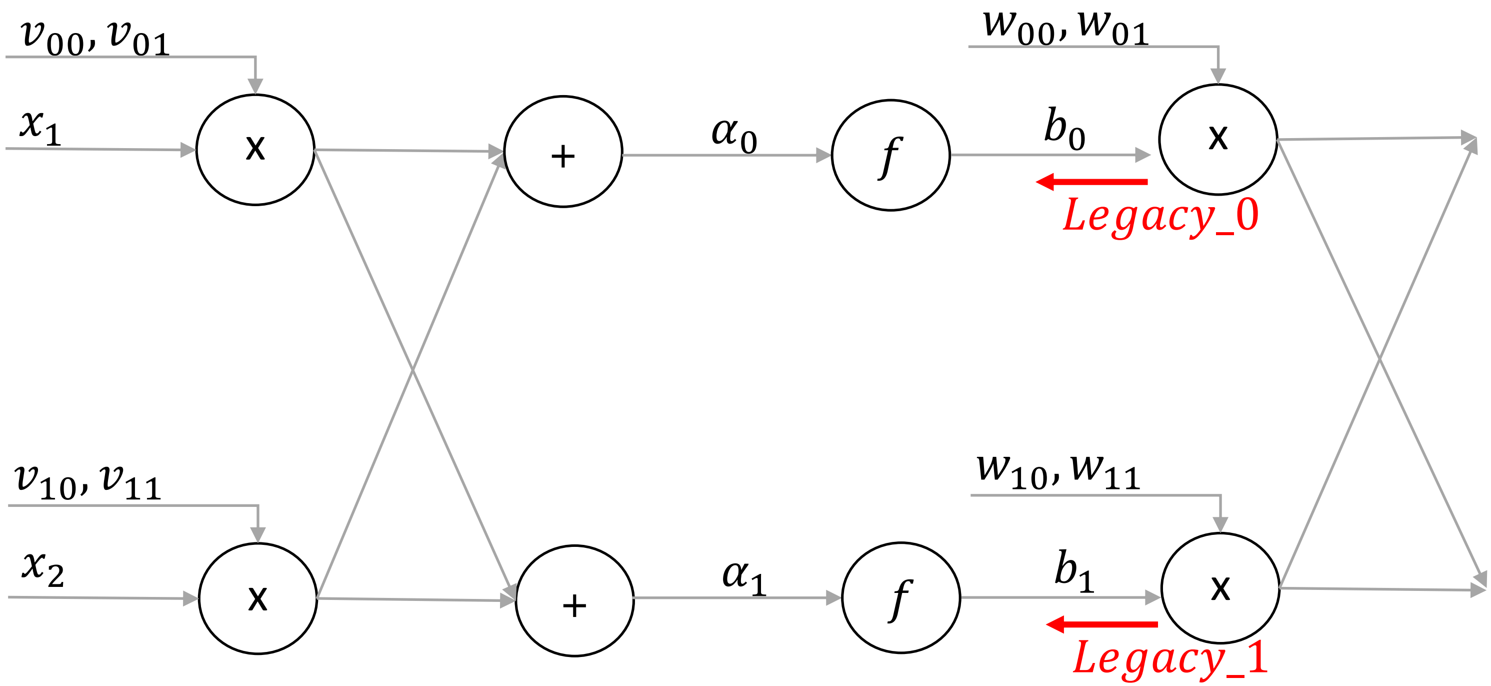
- 활성화 함수 추가



Back-Propagation for DNN

- 활성화 함수 + Node 추가

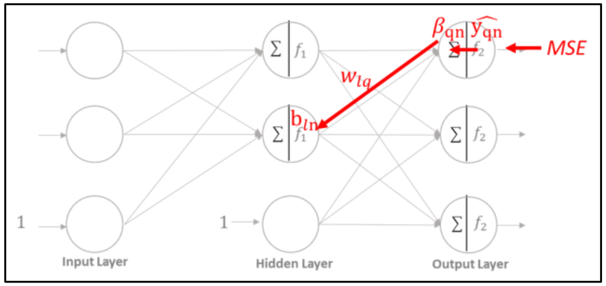
가중치가 3개 였다면?

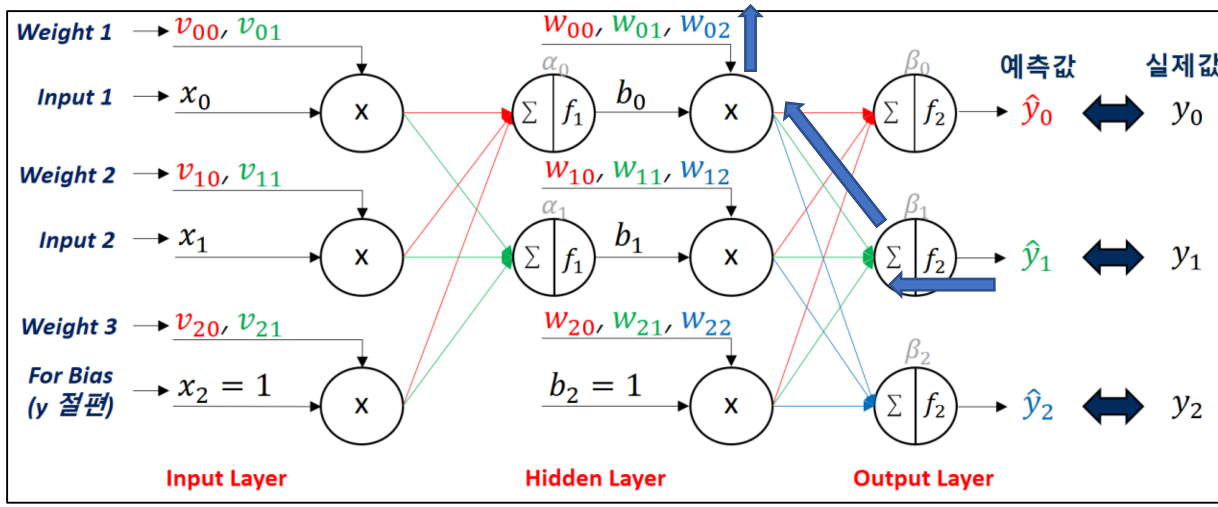


다계층 Error Back-Propagation

지능형 예측·진단 연구실
Intelligent Prognostics & Diagnostics Lab.

- **가중치 1개에 대해서 업데이트 예시**





- $\dfrac{\partial MSE}{\partial w_{01}} = \dfrac{\partial Loss}{\partial \widehat{y_1}} * \dfrac{\partial \widehat{y_1}}{\partial \beta_1} * \dfrac{\partial \beta_1}{\partial w_{01}}$
- $\dfrac{\partial MSE}{\partial \widehat{y_1}} = 2(\widehat{y_1} - y_1) = L1$
- $\dfrac{\partial MSE}{\partial \widehat{y_1}} * \dfrac{\partial \widehat{y_1}}{\partial \beta_1} = L1 \; * \widehat{y_1}(1 - \widehat{y_1}) = L2$
- $\dfrac{\partial MSE}{\partial \widehat{y_1}} * \dfrac{\partial \widehat{y_1}}{\partial \beta_1} * \dfrac{\partial \beta_1}{\partial w_{01}} = L2 \cdot b_0 = 2(\widehat{y_1} - y_1)\widehat{y_1}(1 - \widehat{y_1})b_0$

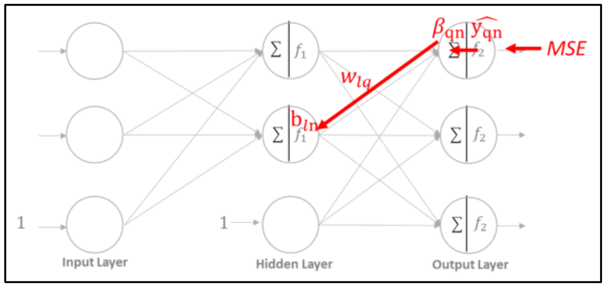
각 계층별 미분을 안다면,
연쇄법칙을 통해
쉽게 계층을 늘릴 수 있음

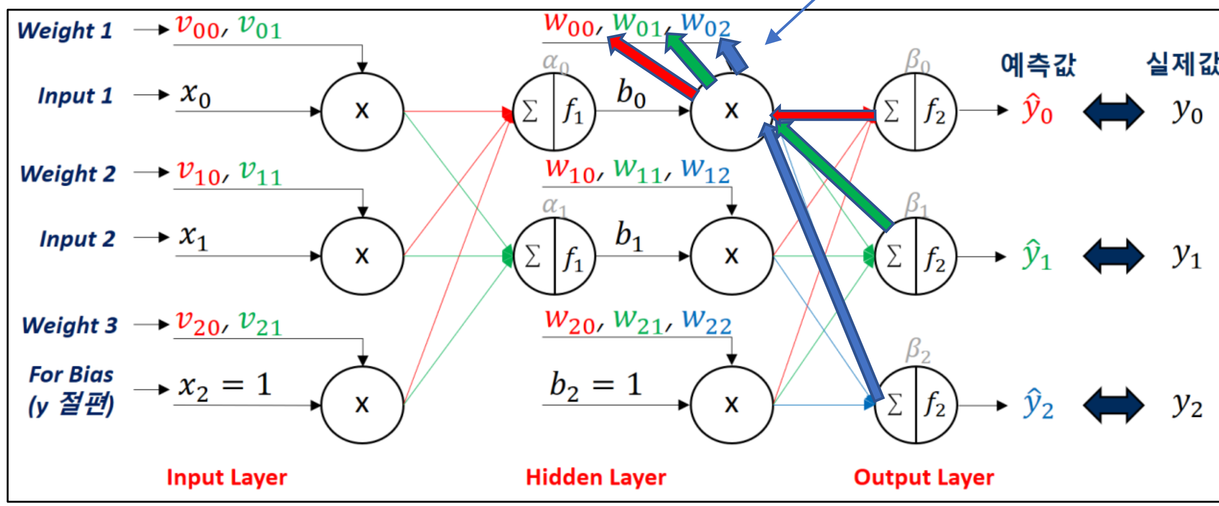
이전에 구한 식과 같음

$$\frac{\partial \epsilon_{MSE}(n)}{\partial w_{lq}} = 2\left(\hat{y}_{qn} - y_{qn}\right) \hat{y}_{qn}\left(1 - \hat{y}_{qn}\right) b_{ln}$$

다계층 Error Back-Propagation

$$\frac{\partial \boldsymbol{MSE}}{\partial \hat{\boldsymbol{y}}} * \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\beta}} * \frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{w_0}} = \boldsymbol{L2} * b_0 = 2(\hat{\boldsymbol{y}} - \boldsymbol{y})\hat{\boldsymbol{y}}(1 - \hat{\boldsymbol{y}}) * b_0$$





$$\boldsymbol{w_{new}} = \boldsymbol{w_{old}} - \alpha \frac{\partial \boldsymbol{MSE}}{\partial \boldsymbol{w_{old}}}$$

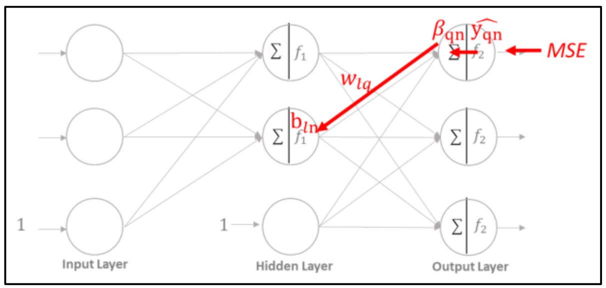
$$\frac{\partial \boldsymbol{MSE}}{\partial \boldsymbol{w_{old}}} = \boldsymbol{b^T} * 2(\hat{\boldsymbol{y}} - \boldsymbol{y})\hat{\boldsymbol{y}}(1 - \hat{\boldsymbol{y}})$$

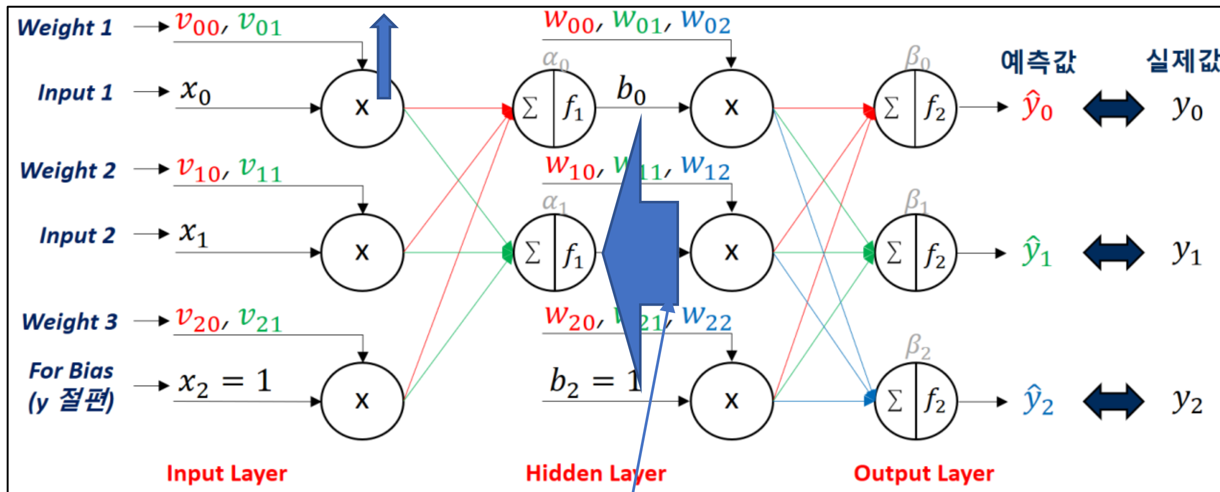
다계층 Error Back-Propagation

지능형 예측·진단 연구실
Intelligent Prognostics & Diagnostics Lab.

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{old} - \alpha \frac{\partial \boldsymbol{MSE}}{\partial \boldsymbol{w}}$$

$$\frac{\partial \boldsymbol{MSE}}{\partial \boldsymbol{w}_{old}} = \boldsymbol{b}^{T} * 2(\widehat{\boldsymbol{y}} - \boldsymbol{y})\widehat{\boldsymbol{y}}(1 - \widehat{\boldsymbol{y}})$$





$$\frac{\partial \boldsymbol{MSE}}{\partial \boldsymbol{b}} = 2(\widehat{\boldsymbol{y}} - \boldsymbol{y})\widehat{\boldsymbol{y}}(1 - \widehat{\boldsymbol{y}}) * \boldsymbol{w}_{old}^{\mathbf{T}}$$

Neural Network 모델의 학습 알고리즘

지능형 예측·진단 연구실
Intelligent Prognostics & Diagnostics Lab.

- **전체 알고리즘 적용 순서**
  1) 신경망 모델 설계 (**Hidden Layer 수**, Node 수, 활성화 함수, Learning Rate 등)
  2) 설계된 모델에 따른 Weight Matrix 생성 및 초기화
  3) N개의 훈련 데이터 Shuffle
  4) 0부터 N-1번째 데이터에 대해 순차적으로 오차 역전파 알고리즘 적용 (1 epoch)
     - **for** n번째 데이터
       - (순전파) n-1에서 업데이트 된 Weight로부터 예측값 $\hat{y}$ 계산
       - (역전파) 실제값(Target)과 예측값(Output) 간의 오차 계산
       - (역전파) 순전파의 반대 순서로 되돌아오며 미분값 계산 및 업데이트
  5) 업데이트 된 weight로부터 accuracy 계산
  6) 사용자가 입력한 epoch 수 만큼 3~4) 반복

각 실습문제에 대해 code(+주석), 그래프, 분석 내용 등은 필수적으로 포함시킬 것

실습 #1) Two-Layer Neural Network 구현

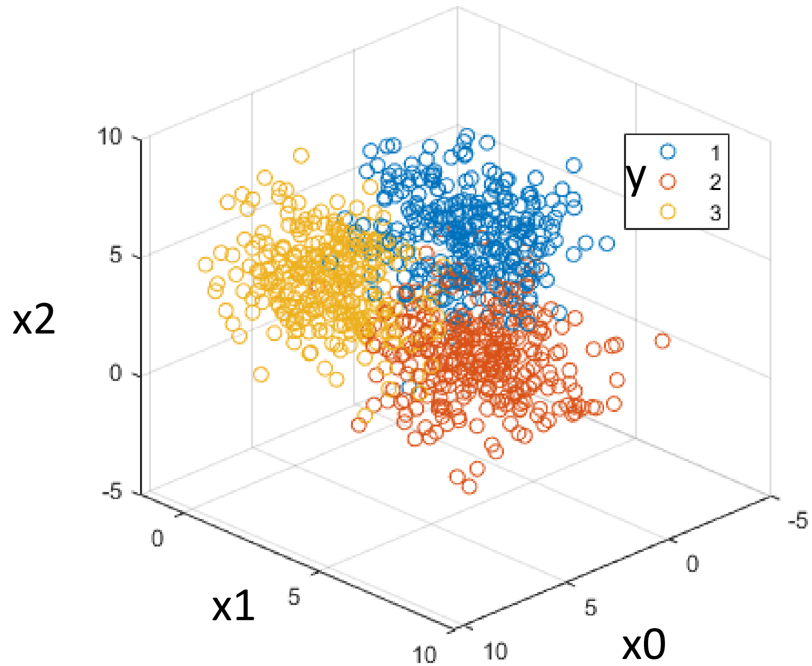
- 주어진 데이터의 y값을 One-Hot Encoding 하여 구현
- Hidden Layer의 Node 수는 5개로 고정
- Input 속성 수, Output Class 수, Hidden Node 수로부터 Weight Matrix 생성 및 초기화
- 모든 활성화 함수는 Sigmoid 사용
- <u>Weight 업데이트 시, 반복문(for, while 등)을 사용하지 않고 업데이트
데이터 개수에 따른 반복문은 사용 → (1회 업데이트에 한하여 사용X)</u>
- "NN_data.csv"에 적용: 랜덤 초기화 된 Weight Matrix로부터 $\hat{y}$ 값 도출
- $\hat{y}$값 중 가장 큰 값을 기준으로 One-Hot Encoding
- 전체 Training set에 대해서 $\hat{y} = y$ 인 데이터 개수 count => 정확도 계산

실습 #2) Deep Neural Network 구현

- 실습 #1)에서 Hidden Layer의 개수를 1개에서 2개로 증가
- Hidden Layer의 Node 수는 5개로 고정
- $\hat{y}$값에서 0.5를 기준으로 0 or 1로 변환
- 전체 Training set에 대해서 $\hat{y} = y$ 인 데이터 개수 count => 정확도 계산

Chap.1 실습 데이터

지능형 예측·진단 연구실
Intelligent Prognostics & Diagnostics Lab.



1) Class1 : (x0,x1,x2)=(2,4,6)을 중심으로 노이즈가 추가된 데이터
2) Class2 : (x0,x1,x2)=(4,6,2)을 중심으로 노이즈가 추가된 데이터
3) Class3 : (x0,x1,x2)=(6,2,4)을 중심으로 노이즈가 추가된 데이터

각 Class별 300개씩 총 900개