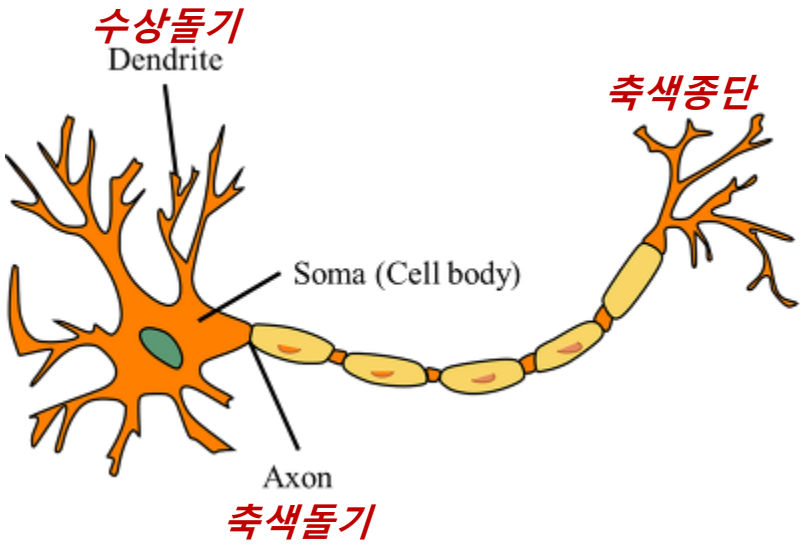
Chap.4 인공 신경망 (Artificial Neural Network)

방 수 식 교수
(bang@tukorea.ac.kr)

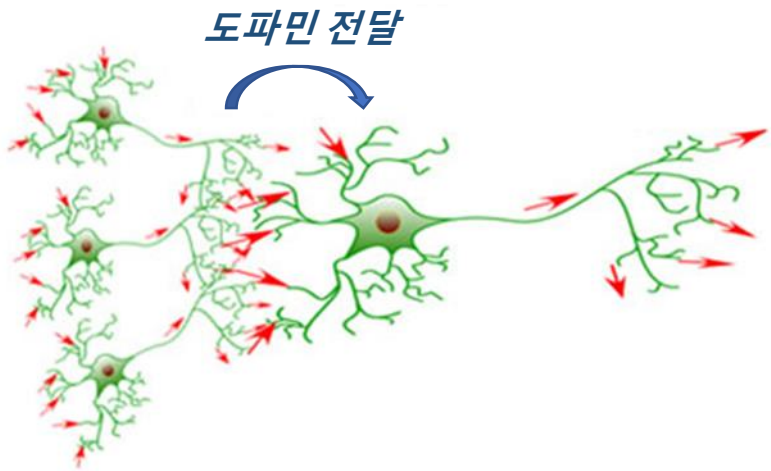
한국공학대학교 전자공학부

2024년도 1학기
머신러닝실습 & 인공지능설계실습1

신경망 (Neural Network)



Neuron

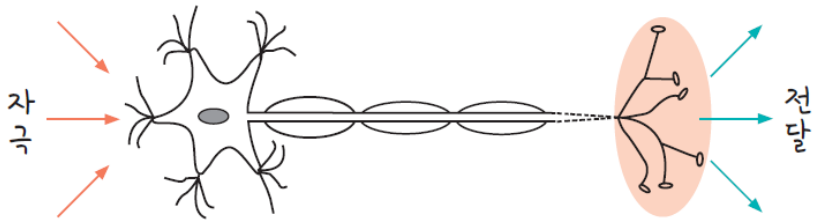


Neural Network in Brain

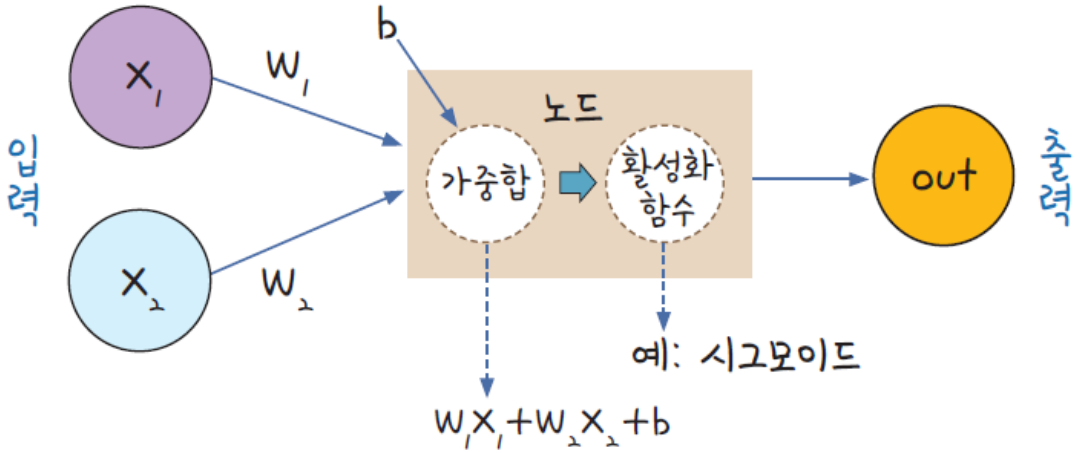
- 축색종단에서의 도파민(신경전달물질) 분비 여부
  - 전달된 신호의 강도에 따라 결정됨
  - 즉, 신호의 강도가 임계값을 넘으면 도파민 분비 ➔ **활성화**
  - 신호의 강도가 임계값을 넘지 않으면 도파민 분비되지 않음 ➔ **비활성화**

Artificial Neuron

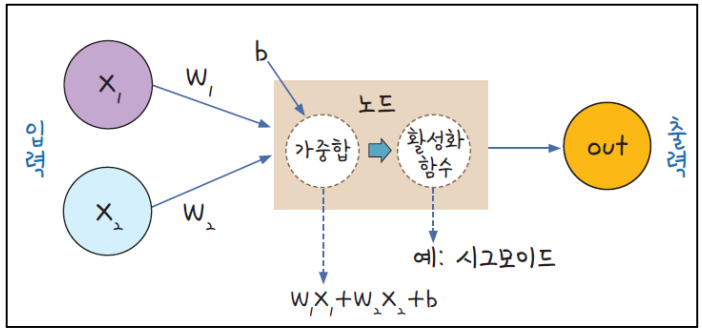
- **Neuron의 핵심 기능**
  - **전파(Propagation)** 기능
    - 신호(도파민)를 받아 전달
  - **활성화(Activation)** 기능
    - 신호의 강도에 따라 신호 활성화 유무를 결정

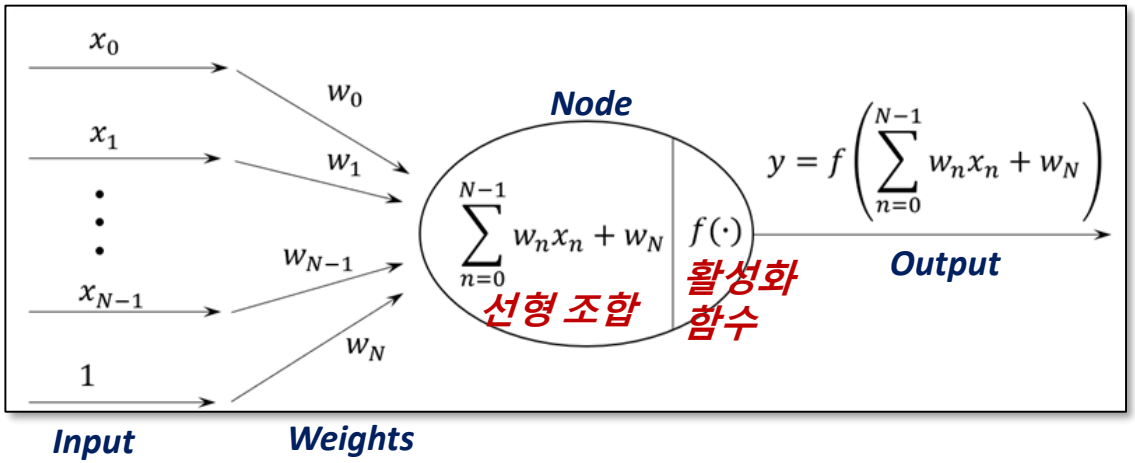


- **Artificial Neuron(인공 뉴런)**
  - Neuron 세포를 모방하여 설계



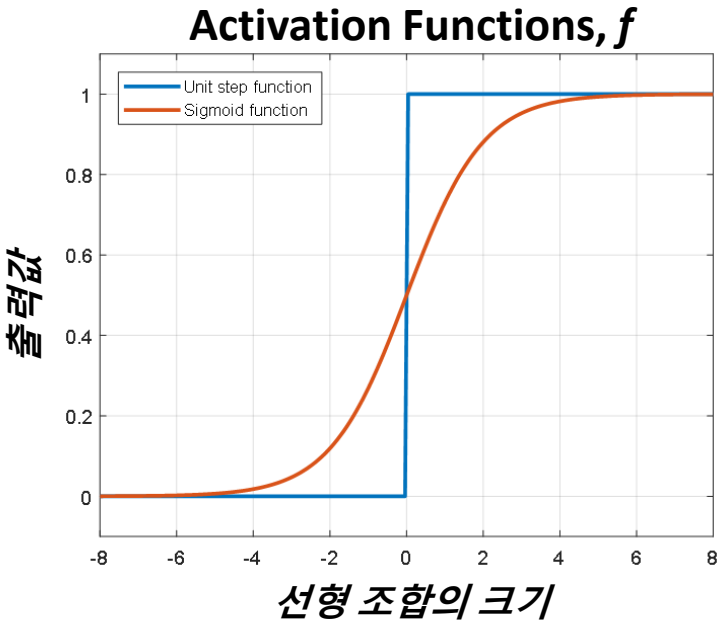
Artificial Neuron





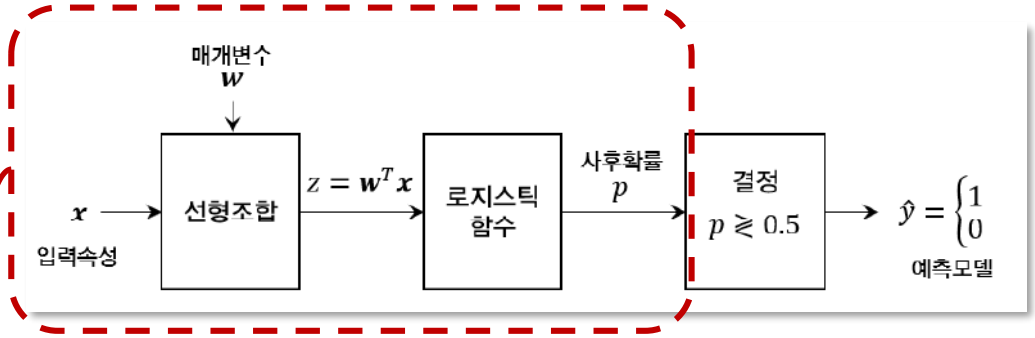
- **전파(Propagation)** 기능
  - 입력과 Weights 들의 선형 조합을 전파

- **활성화(Activation)** 기능
  - 선형 조합의 크기에 따라 다음 뉴런에 전달할지 여부를 결정

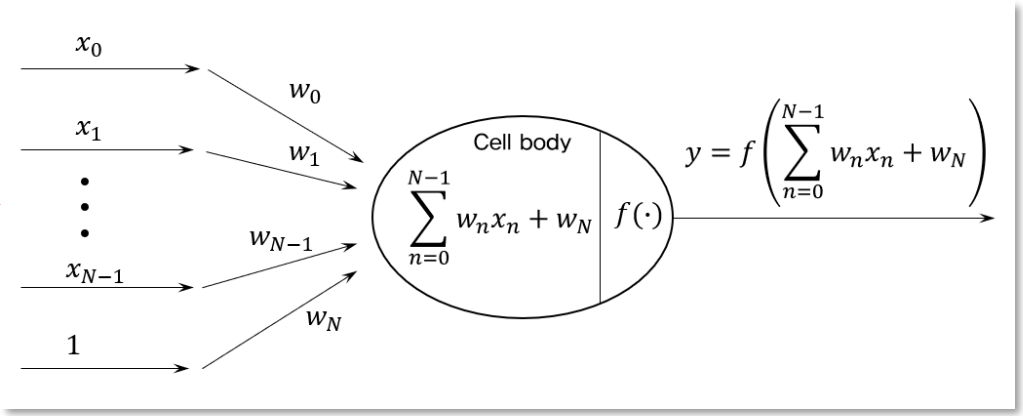


Logistic Regression과의 비교

- **Artificial Neuron의 관점에서의 Logistic Regression**
  - 일반적인 Logistic Regression 모델

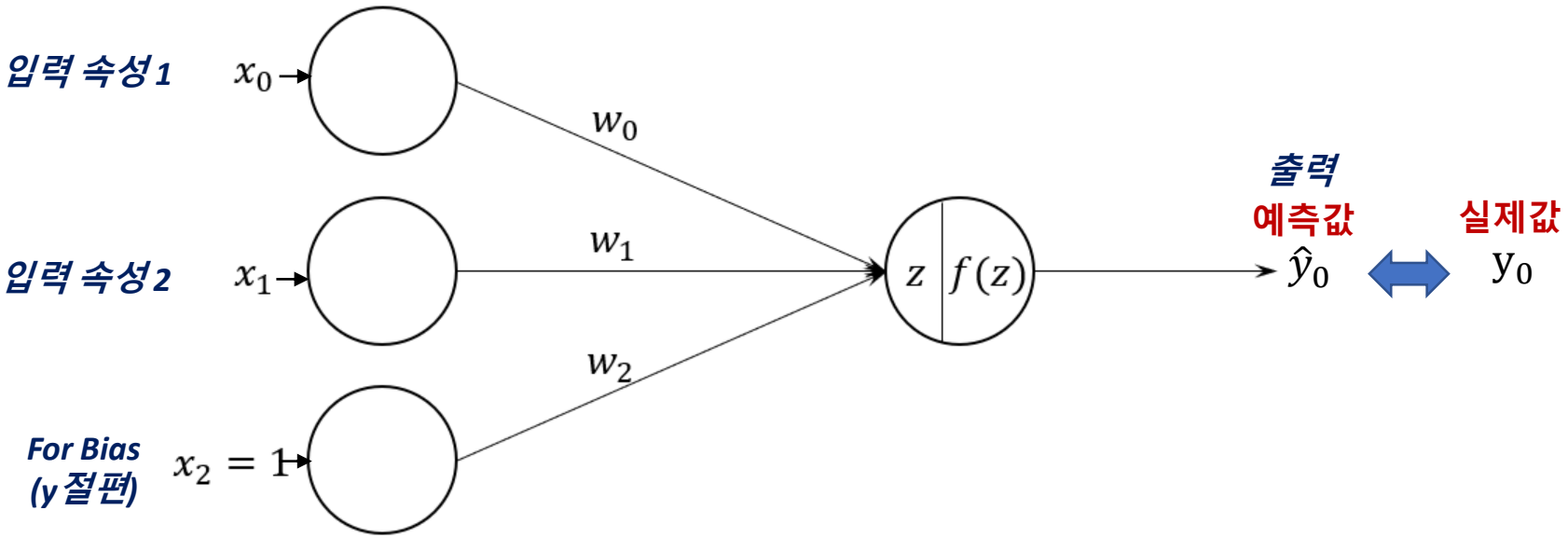


  - Sigmoid 함수를 Activation Function으로 사용하는 Artificial Neuron과 동일한 구조



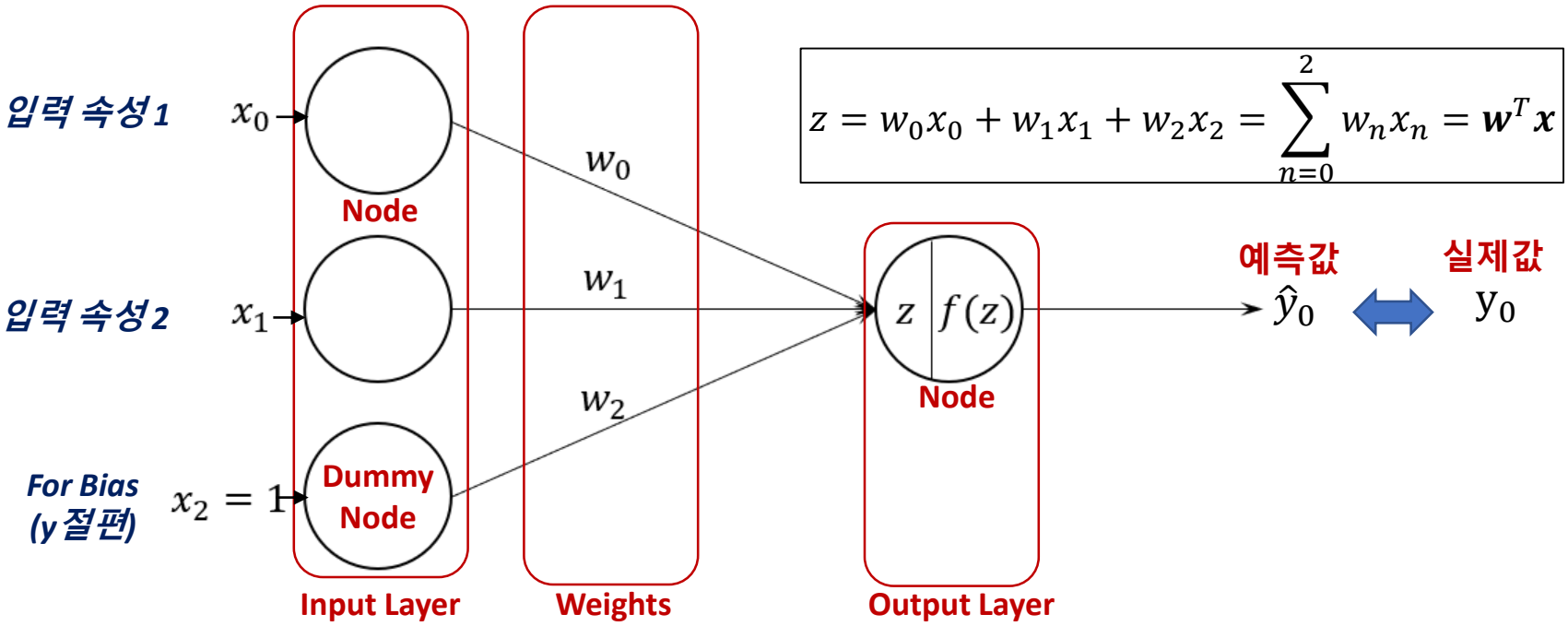
Perceptron

- **퍼셉트론 (Perceptron)**
  - 입력층과 출력층 만으로 구성된 Single-Layer Neural Network
    - Neural Network의 **기본 구성 단위**
    - Binary Classification 시스템 (출력이 0 or 1)
    - (예) 입력 속성이 2개인 Perceptron



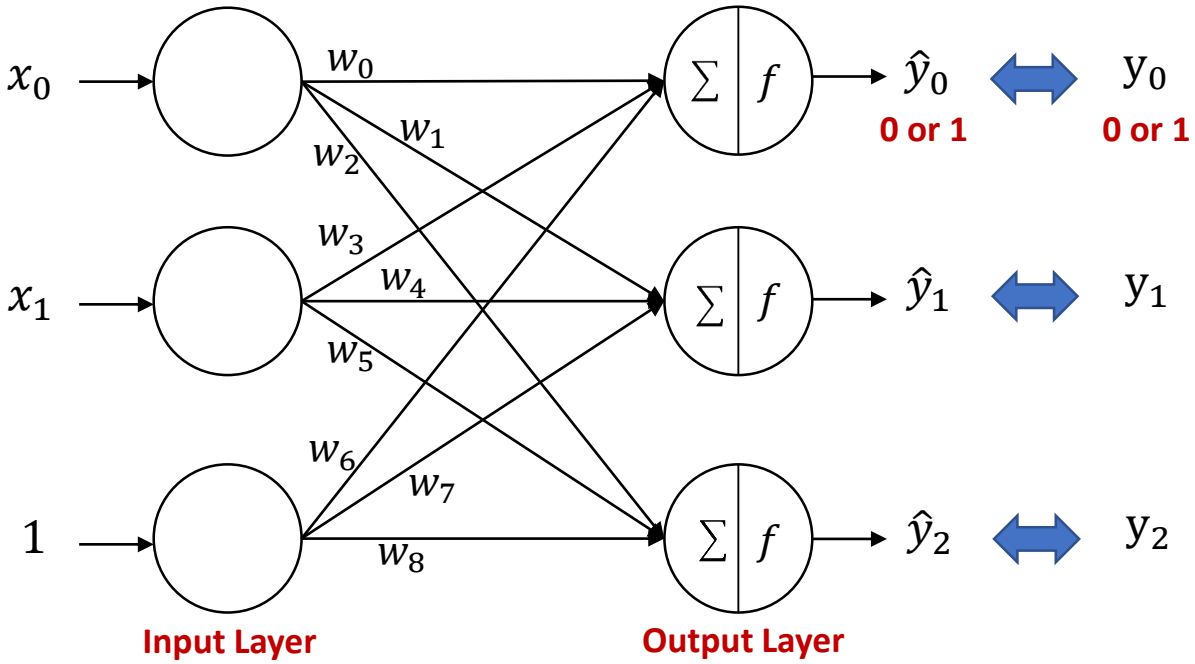
Perceptron

- **퍼셉트론 (Perceptron)**
  - Node: 해당 Node로 들어온 **모든 값을 더한 다음 Activation Function을 통과시킨 값을 출력**
    - Input node의 Activation Function: *f(x)=x* (별도표기하지 않음)
  - 화살표: 화살표 위에 적힌 weight만큼 **곱 연산**을 수행
    - 별도 표기가 없는 화살표의 weight는 1



분류해야하는 Class가 3개 이상이라면?

Multi-Class Classification in Single Layer

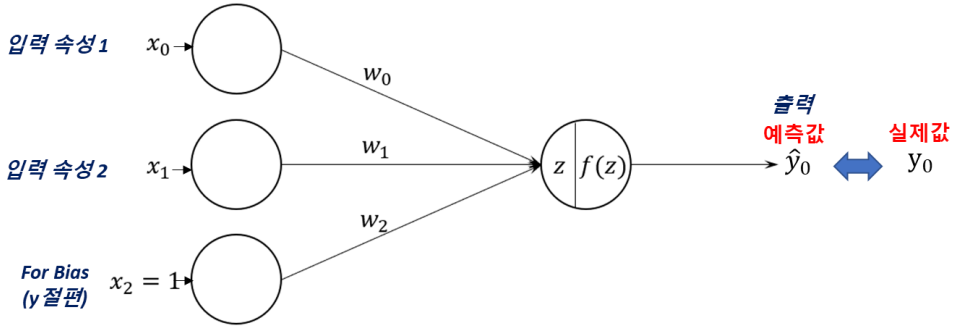


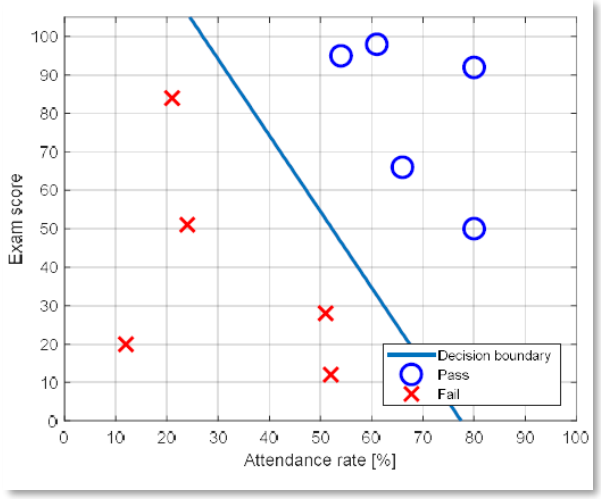
- **One-Hot Encoding (범주형 Class의 이진화)**
  - 확률론적 접근 + 기계가 인식 가능한 숫자
    - ✓ Binary Class: 음성 or 양성 => 0 or 1
    - ✓ Multi-Class: 개 or 고양이 or 말 => 100 or 010 or 001

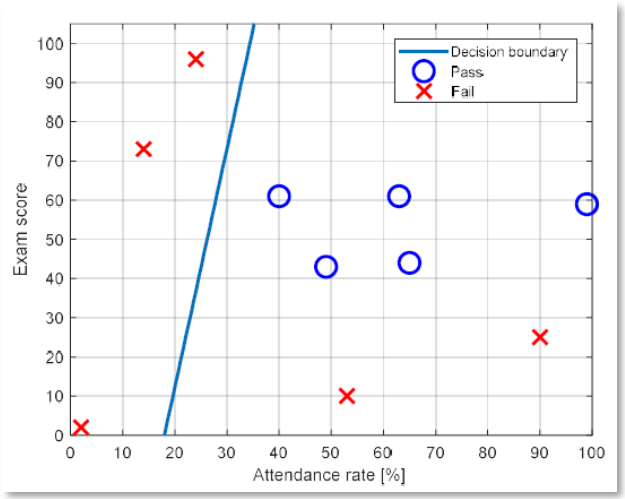
(y_0, y_1, y_2)

Single Layer Neural Network의 한계

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

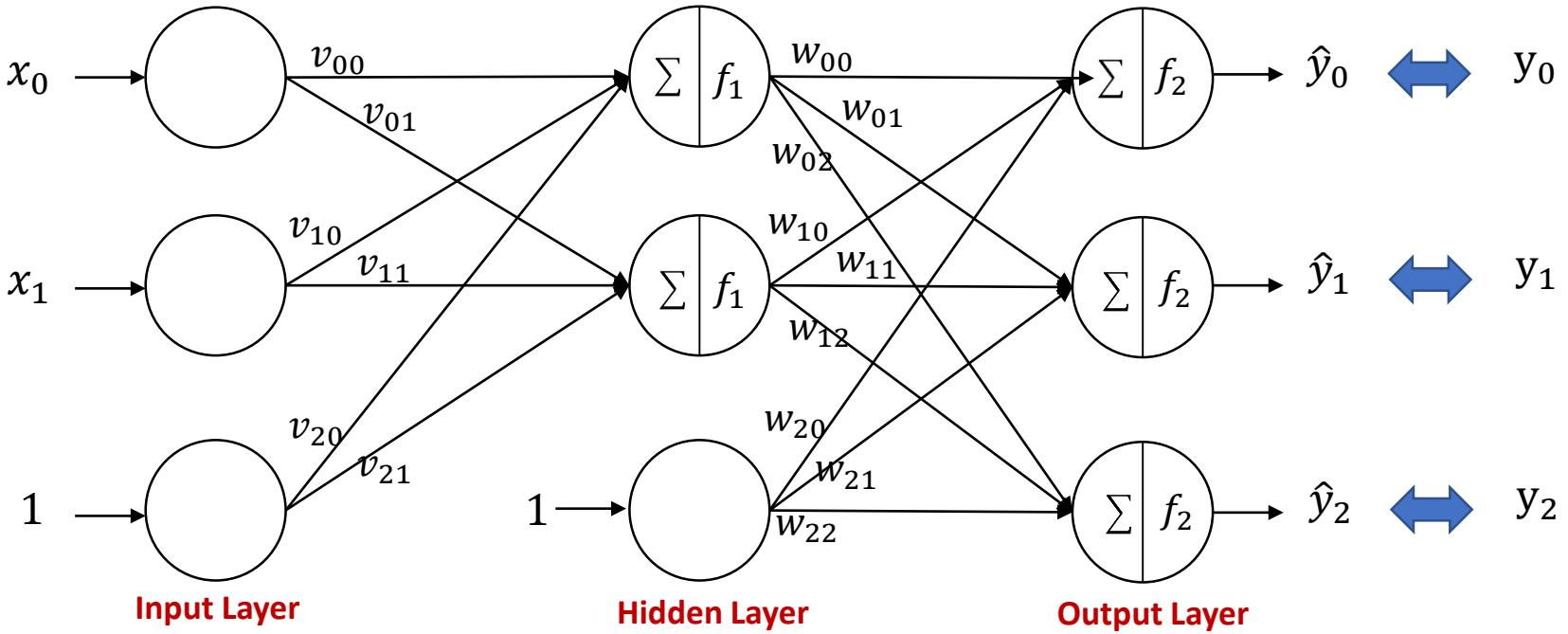






Multi-Layer Neural Network

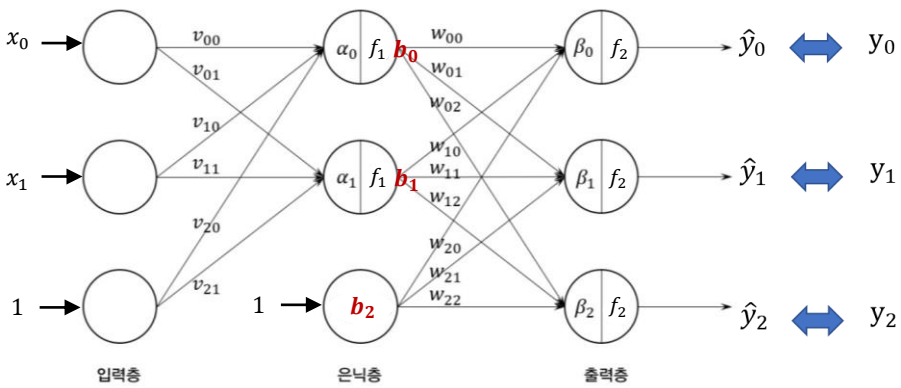
- **Two-Layer Neural Network**



- **Input layer (입력층)**
  - ➢ 2 Input: 일반노드 2개 + 더미노드 1개 *<= 데이터의 Input 속성 수에 의해 결정*
- **Hidden Layer (은닉층)**
  - ➢ 일반노드 2개 + 더미노드 1개 *<= 모델 설계자가 결정 (Hyper-parameter)*
- **Output Layer (출력층)**
  - ➢ 3-Class: 일반노드 3개 *<= 데이터의 Output class 수에 의해 결정*

Multi-Layer Neural Network

- **Node 별 입출력 계산**



Hidden Layer 입출력 관계

$$\alpha_0 = v_{00}x_0 + v_{10}x_1 + v_{20}x_2$$
$$\alpha_1 = v_{01}x_0 + v_{11}x_1 + v_{21}x_2$$

$x_2 = 1$

$$b_0 = f_1(\alpha_0)$$
$$b_1 = f_1(\alpha_1) \quad b_2 = 1$$

Output Layer 입출력

$$\beta_0 = w_{00}b_0 + w_{10}b_1 + w_{20}b_2 \qquad \hat{y}_0 = f_2(\beta_0)$$
$$\beta_1 = w_{01}b_0 + w_{11}b_1 + w_{21}b_2 \qquad \hat{y}_1 = f_2(\beta_1)$$
$$\beta_2 = w_{02}b_0 + w_{12}b_1 + w_{22}b_2 \qquad \hat{y}_2 = f_2(\beta_2)$$

Artificial Neural Network(ANN) 모델의 학습 목표

$\hat{y}_0 \approx y_0, \hat{y}_1 \approx y_1, \hat{y}_2 \approx y_2$를 만족하는 최적 매개변수를 찾는다.

Two-Layer Neural Network의 일반화

2-Input, 3-Class with 2-Hidden node

Dummy node는 일반적으로 카운팅하지 않음.

Hidden Layer 입출력

$x_2 = 1$

$$\alpha_0 = v_{00}x_0 + v_{10}x_1 + v_{20}x_2$$
$$\alpha_1 = v_{01}x_0 + v_{11}x_1 + v_{21}x_2$$

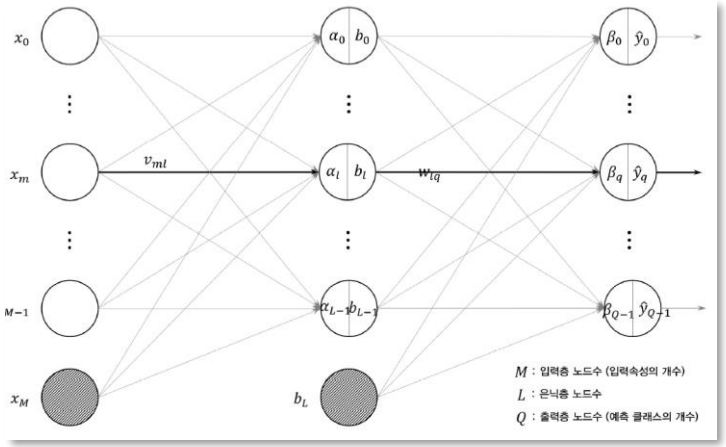
$$b_0 = f_1(\alpha_0)$$
$$b_1 = f_1(\alpha_1)$$
$$b_2 = 1$$

Output Layer 입출력

$$\beta_0 = w_{00}b_0 + w_{10}b_1 + w_{20}b_2$$
$$\beta_1 = w_{01}b_0 + w_{11}b_1 + w_{21}b_2$$
$$\beta_2 = w_{02}b_0 + w_{12}b_1 + w_{22}b_2$$

$$\hat{y}_0 = f_2(\beta_0)$$
$$\hat{y}_1 = f_2(\beta_1)$$
$$\hat{y}_2 = f_2(\beta_2)$$

M-Input, *Q*-Class with *L*-Hidden node



Two-Layer Neural Network의 일반화

2-Input, 3-Class with 2-Hidden node

Dummy node는 일반적으로 카운팅하지 않음.

Hidden Layer 입출력

$x_2 = 1$

$$\alpha_0 = v_{00}x_0 + v_{10}x_1 + v_{20}x_2$$
$$\alpha_1 = v_{01}x_0 + v_{11}x_1 + v_{21}x_2$$

$$b_0 = f_1(\alpha_0)$$
$$b_1 = f_1(\alpha_1)$$
$$b_2 = 1$$

Output Layer 입출력

$$\beta_0 = w_{00}b_0 + w_{10}b_1 + w_{20}b_2$$
$$\beta_1 = w_{01}b_0 + w_{11}b_1 + w_{21}b_2$$
$$\beta_2 = w_{02}b_0 + w_{12}b_1 + w_{22}b_2$$

$$\hat{y}_0 = f_2(\beta_0)$$
$$\hat{y}_1 = f_2(\beta_1)$$
$$\hat{y}_2 = f_2(\beta_2)$$

M-Input, *Q*-Class with *L*-Hidden node

Hidden Layer 입출력

$x_M = 1$

$$\alpha_l = v_{0l}x_0 + v_{1l}x_1 + \cdots + v_{Ml}x_M = \sum_{m=0}^{M} v_{ml}x_m$$

$$l = 0, 1, \cdots, L-1$$

$$b_l = f_1(\alpha_l) = f_1\left(\sum_{m=0}^{M} v_{ml}x_m\right) \qquad b_L = 1$$

Output Layer 입출력

$$\beta_q = w_{0q}b_0 + w_{1q}b_1 + \cdots + w_{lq}b_l + w_{Lq}b_L$$
$$= \sum_{l=0}^{L} w_{lq}b_l, \qquad q = 0, 1, \cdots, Q-1$$

$$\hat{y}_q = f_2(\beta_q) = f_2\left(\sum_{l=0}^{L} w_{lq}b_l\right)$$

Two-Layer Neural Network의 일반화

- 1개의 데이터에 대해: *M*-Input, Q-Class with *L*-Hidden node 연산의 행렬화

Hidden Layer 입력

$$\alpha_l = v_{0l}x_0 + v_{1l}x_1 + \cdots + v_{Ml}x_M = \sum_{m=0}^{M} v_{ml}x_m \qquad x_M = 1$$

$$l = 0, 1, \cdots, L-1$$

Hidden Layer 출력

$$b_l = f_1(\alpha_l) = f_1\left(\sum_{m=0}^{M} v_{ml}x_m\right) \qquad b_L = 1$$

$$\begin{aligned} b_0 &= f_1(\alpha_0) \\ b_1 &= f_1(\alpha_1) \\ &\vdots \\ b_l &= f_1(\alpha_l) \\ &\vdots \\ b_{L'} &= f_1(\alpha_{L'}) \\ b_L &= 1 \end{aligned} \quad \Rightarrow \quad \boldsymbol{b} = \begin{bmatrix} f_1(\alpha_0) \\ f_1(\alpha_1) \\ \vdots \\ f_1(\alpha_l) \\ \vdots \\ f_1(\alpha_{L'}) \\ 1 \end{bmatrix}$$

(L+1 by 1)

$L' = L - 1$ 일 때,

$$\begin{aligned} \alpha_0 &= v_{00}x_0 + v_{10}x_1 + \cdots + v_{M0}x_M \\ \alpha_1 &= v_{01}x_0 + v_{11}x_1 + \cdots + v_{M1}x_M \\ &\vdots \\ \alpha_l &= v_{0l}x_0 + v_{1l}x_1 + \cdots + v_{Ml}x_M \\ &\vdots \\ \alpha_{L'} &= v_{0L'}x_0 + v_{1L'}x_1 + \cdots + v_{ML'}x_M \end{aligned}$$

L개

$$\underset{\boldsymbol{\alpha}\ (L \text{ by } 1)}{\begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_l \\ \vdots \\ \alpha_{L'} \end{bmatrix}} = \underset{\boldsymbol{v}\ (L \text{ by } M+1)}{\begin{bmatrix} v_{00} & v_{10} & \cdots & v_{M0} \\ v_{01} & v_{11} & \cdots & v_{M1} \\ \vdots & & \ddots & \vdots \\ v_{0l} & v_{1l} & \ddots & v_{Ml} \\ \vdots & & \ddots & \vdots \\ v_{0L'} & v_{1L'} & \cdots & v_{ML'} \end{bmatrix}} \underset{\boldsymbol{x}\ (M+1 \text{ by } 1)}{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_M \end{bmatrix}}$$

$$\boldsymbol{\alpha} = \boldsymbol{v}\boldsymbol{x}$$

Two-Layer Neural Network의 일반화

- 1개의 데이터에 대해: *M*-Input, Q-Class with *L*-Hidden node 연산의 행렬화

Output Layer 입출력

$$\beta_q = w_{0q}b_0 + w_{1q}b_1 + \cdots + w_{lq}b_l + w_{Lq}b_L$$

$$= \sum_{l=0}^{L} w_{lq}b_l\,, \qquad q = 0, 1, \cdots, Q-1$$

$$\hat{y}_q = f_2(\beta_q) = f_2\left(\sum_{l=0}^{L} w_{lq}b_l\right)$$

$Q' = Q - 1$ 일 때,

$$\underset{\boldsymbol{\beta}\ (Q\ \text{by}\ 1)}{\begin{bmatrix}\beta_0\\ \beta_1\\ \vdots\\ \beta_q\\ \vdots\\ \beta_{Q'}\end{bmatrix}} = \underset{\boldsymbol{w}\ (Q\ \text{by}\ L+1)}{\begin{bmatrix} w_{00} & w_{10} & \cdots & w_{L0}\\ w_{01} & w_{11} & \cdots & w_{L1}\\ \vdots & & \ddots & \vdots\\ w_{0q} & w_{1q} & \ddots & w_{Lq}\\ \vdots & & \ddots & \vdots\\ w_{0Q'} & w_{1Q'} & \cdots & w_{LQ'}\end{bmatrix}} \underset{\boldsymbol{b}\ (L+1\ \text{by}\ 1)}{\begin{bmatrix} f_1(\alpha_0)\\ f_1(\alpha_1)\\ \vdots\\ f_1(\alpha_l)\\ \vdots\\ f_1(\alpha_{L'})\\ 1\end{bmatrix}}$$

$$\boldsymbol{\beta} = \boldsymbol{w}\mathbf{b}$$

$$\underset{(Q\ \text{by}\ 1)}{\hat{\boldsymbol{y}} = \begin{bmatrix}\hat{y}_0\\ \hat{y}_1\\ \hat{y}_2\\ \vdots\\ \hat{y}_q\\ \vdots\\ \hat{y}_{Q-1}\end{bmatrix}} = \begin{bmatrix} f_2(\beta_0)\\ f_2(\beta_1)\\ f_2(\beta_2)\\ \vdots\\ f_2(\beta_l)\\ \vdots\\ f_2(\beta_{Q-1})\end{bmatrix}$$

Two-Layer Neural Network의 일반화

- **N개의 데이터**

데이터 번호	입력	출력
	$x_0,\cdots,x_m,\cdots,x_{M-1}$	$y_0,\cdots,y_q,\cdots,y_{Q-1}$
0	$x_{00},\cdots,x_{m0},\cdots,x_{(M-1)0}$	$y_{00},\cdots,y_{q0},\cdots,y_{(Q-1)0}$
⋮	⋮	⋮
n	$x_{0n},\cdots,x_{mn},\cdots,x_{(M-1)n}$	$y_{0n},\cdots,y_{qn},\cdots,y_{(Q-1)n}$
⋮	⋮	⋮
$N-1$	$x_{0(N-1)},\cdots,x_{m(N-1)},\cdots,x_{(M-1)(N-1)}$	$y_{0(N-1)},\cdots,y_{q(N-1)},\cdots,y_{(Q-1)(N-1)}$

❖ **n번째 데이터에 대해서 (n은 0부터 N-1까지의 정수)**

$\boldsymbol{\alpha}$ *(L by 1)* $\boldsymbol{v}$ *(L by M+1)* $\boldsymbol{x}$ (M+1 by 1)

$$\begin{bmatrix}\alpha_{0n}\\ \alpha_{1n}\\ \vdots\\ \alpha_{ln}\\ \vdots\\ \alpha_{L'n}\end{bmatrix} = \begin{bmatrix} v_{00} & v_{10} & \cdots & v_{M0}\\ v_{01} & v_{11} & \cdots & v_{M1}\\ \vdots & & \ddots & \vdots\\ v_{0l} & v_{1l} & \ddots & v_{Ml}\\ \vdots & & \ddots & \vdots\\ v_{0L'} & v_{1L'} & \cdots & v_{ML'}\end{bmatrix} \begin{bmatrix}x_{0n}\\ x_{1n}\\ \vdots\\ x_{Mn}\end{bmatrix}$$

❖ *Training 관점: n-1 번째 데이터로부터 업데이트 된 weights*

Chap.4 실습 - 1주차

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

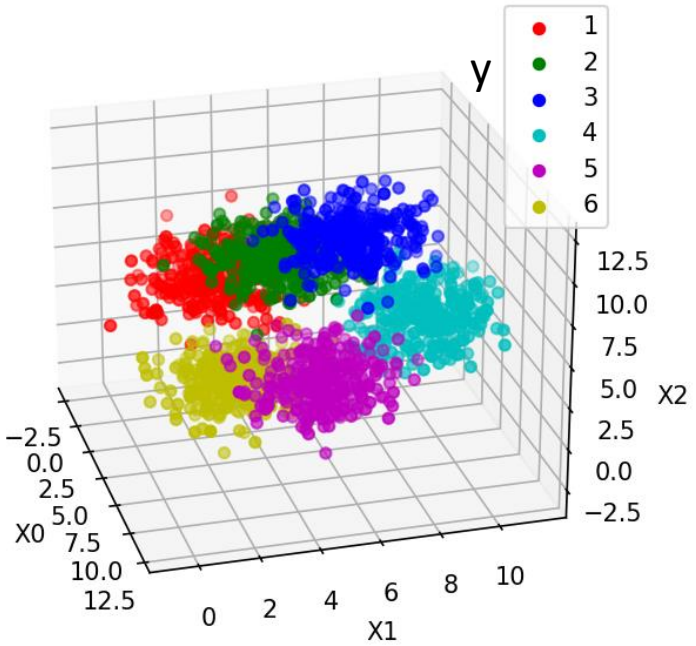
1) One-Hot Encoding 구현
   - 데이터에서의 y값(target) 으로부터 분류 할 Class가 몇 개인지 Check
   - 각 Class에 대해 One-Hot 표현으로 변환
   - “NN_data.csv”에 적용

2) Two-Layer Neural Network 구현
   - 데이터에서의 Input 속성 수 및 Output Class 수를 자동으로 Check하는 기능
   - Hidden layer의 Node 수를 자유롭게 설정하는 기능
   - Input 속성 수, Output Class 수, Hidden Node 수로부터 Weight Matrix 생성 및 초기화
   - “NN_data.csv”에 적용: 랜덤 초기화 된 Weight Matrix로부터 $\hat{y}$ 값 도출

3) Accuracy 함수 구현
   - $\hat{y}$ 값에서 0.5를 기준으로 0 or 1로 변환
   - 전체 Training set에 대해서 $\hat{y} = y$ 인 데이터 개수 count => 정확도 계산

Chap.4 실습 데이터

- 각 Class별 300개씩 총 1800개



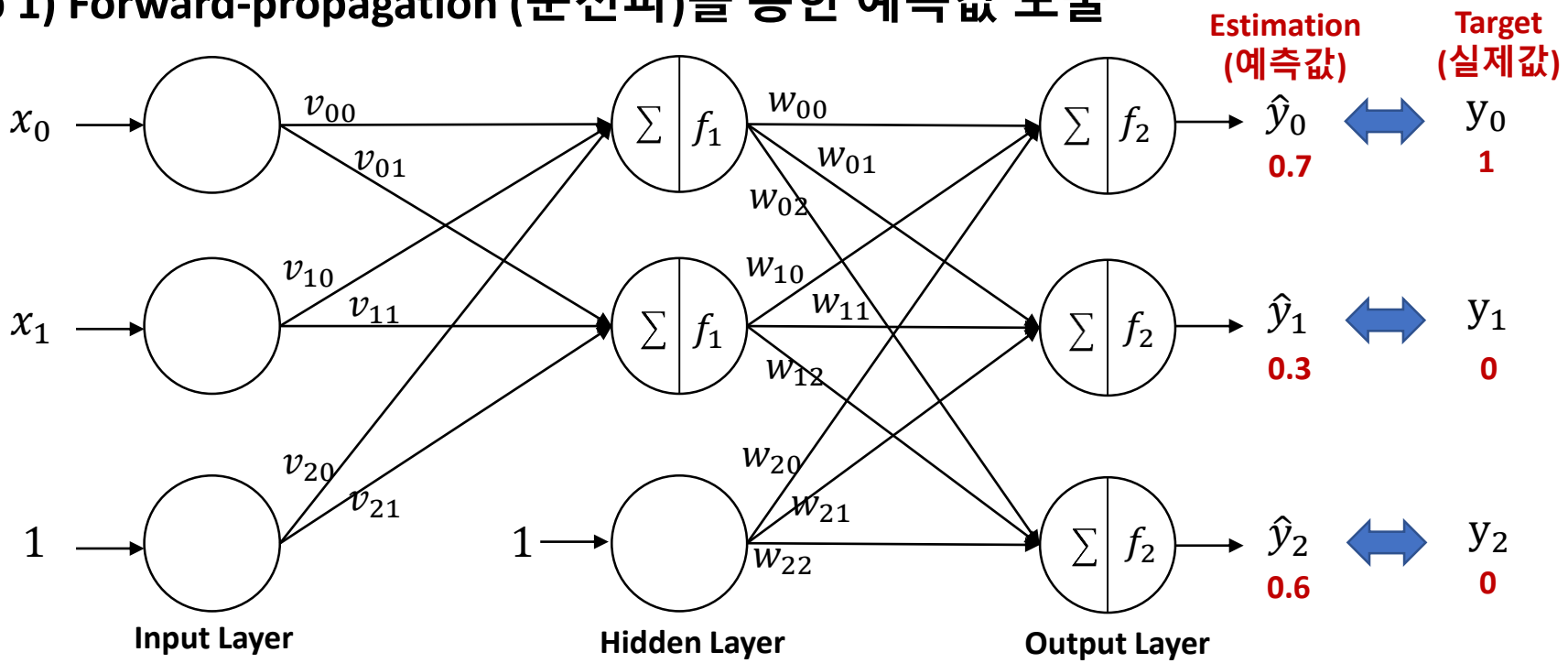
1) Class1 : (x0,x1,x2)=(1,3,5)을 중심으로 노이즈가 추가된 데이터
2) Class2 : (x0,x1,x2)=(3,5,7)을 중심으로 노이즈가 추가된 데이터
3) Class3 : (x0,x1,x2)=(5,7,9)을 중심으로 노이즈가 추가된 데이터
4) Class4 : (x0,x1,x2)=(7,9,5)을 중심으로 노이즈가 추가된 데이터
5) Class5 : (x0,x1,x2)=(9,5,3)을 중심으로 노이즈가 추가된 데이터
6) Class6 : (x0,x1,x2)=(5,3,1)을 중심으로 노이즈가 추가된 데이터

Neural Network 모델의 학습 알고리즘

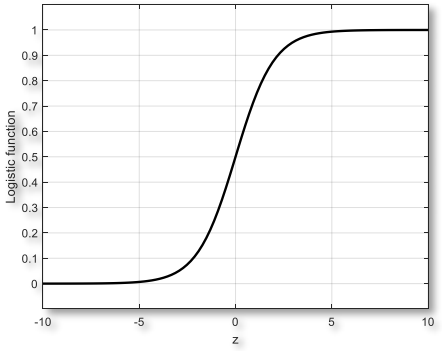
- **Error Backpropagation (오차 역전파) 알고리즘**
  - 경사하강법 기반
  - 개별 훈련 데이터에 대해서 알고리즘 적용
  - 설정한 Cost Function로부터 모델의 예측값과 실측값 간의 Error 값으로부터 역으로 전파되면서 Weight를 업데이트함
    - ✓ 본 강의자료에서는 Activation Function은 Sigmoid, Cost Function은 MSE 기준으로 설명

Error Back-Propagation 알고리즘

Step 1) Forward-propagation (순전파)를 통한 예측값 도출



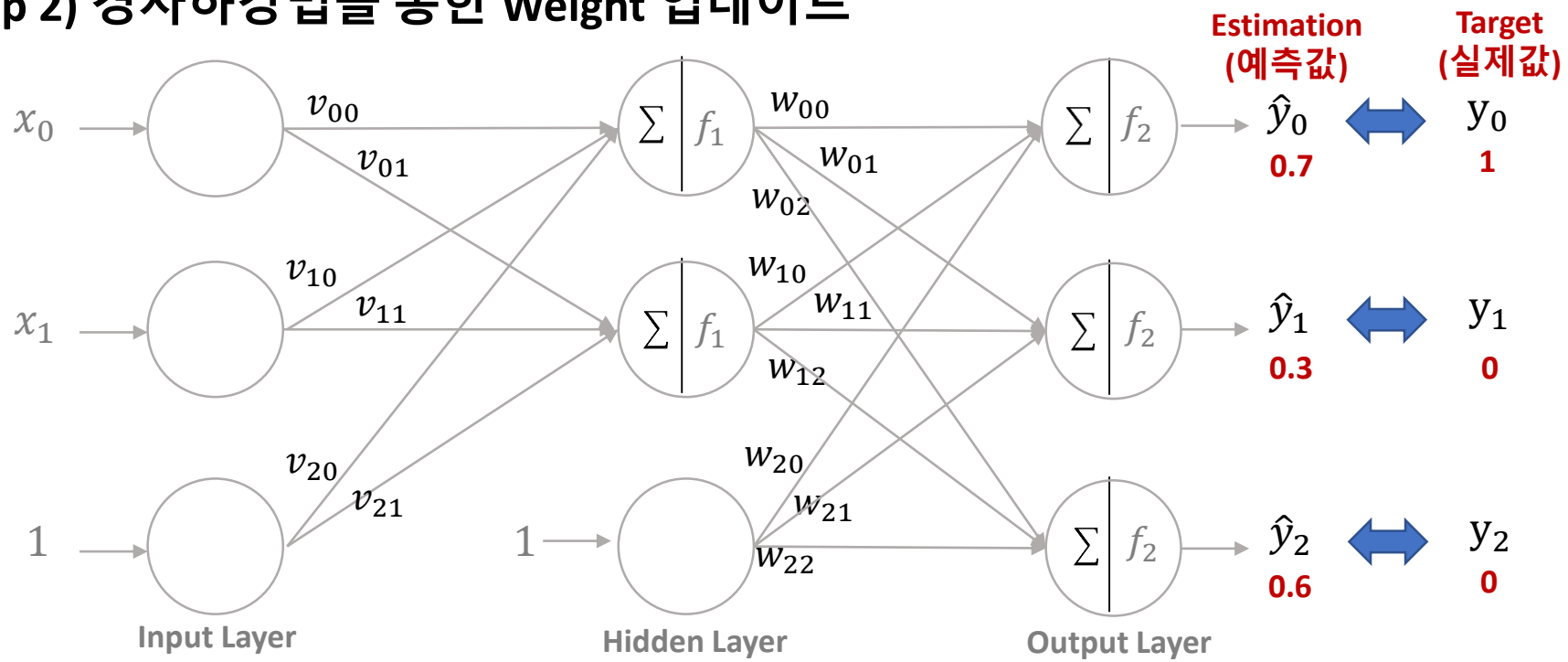
활성화함수 f_1과 f_2는 *Sigmoid 함수로 선택하였다고 가정*



- 예측값 계산 과정: Forward-propagation (순전파)
- Weight 업데이트 과정: Back-propagation(역전파)

Error Back-Propagation 알고리즘

Step 2) 경사하강법을 통한 Weight 업데이트



❖ **Cost Function을 MSE로 설정하였을 때, n번째 데이터에 대해서**

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

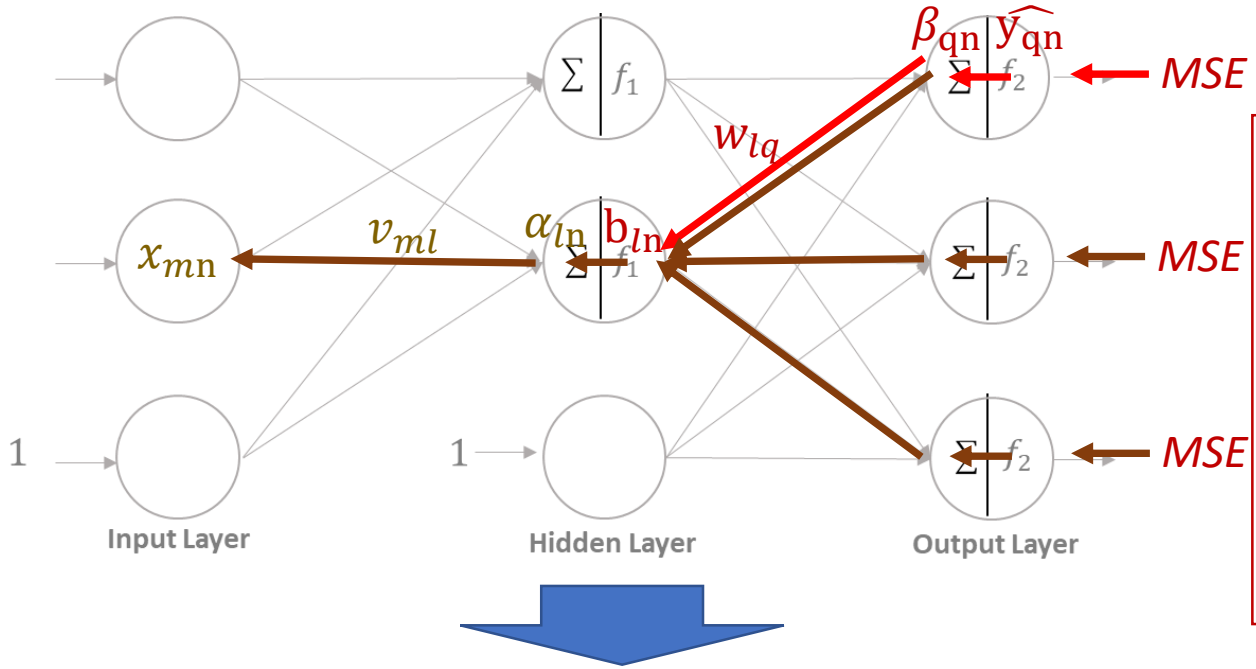
$$w_{lq} \leftarrow w_{lq} - \eta \frac{\partial}{\partial w_{lq}} \epsilon_{MSE}(n)$$

$$\epsilon_{MSE}(n) = \sum_{q=0}^{Q-1} \left(\hat{y}_{qn} - y_{qn} \right)^2$$

일단, 특정 weight에 대한 편미분 고려

Error Back-Propagation 알고리즘

Step 2-1) 경사하강법에서의 도함수 계산



Forward-propagation 에 사용한 weight (Old)

$$v_{ml}^{New} \leftarrow v_{ml}^{Old} - \eta \frac{\partial}{\partial v_{ml}} \epsilon_{MSE}(n)$$

$$w_{lq}^{New} \leftarrow w_{lq}^{Old} - \eta \frac{\partial}{\partial w_{lq}} \epsilon_{MSE}(n)$$

$$\frac{\partial}{\partial w_{lq}} \epsilon_{MSE}(n) = \frac{\partial \epsilon_{MSE}(n)}{\partial \hat{y}_{qn}} \cdot \frac{\partial \hat{y}_{qn}}{\partial \beta_{qn}} \cdot \frac{\partial \beta_{qn}}{\partial w_{lq}}$$

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

Error Back-Propagation 알고리즘

- $\frac{\partial}{\partial w_{lq}}\epsilon_{MSE}(n)$ 구하기

$$\frac{\partial}{\partial w_{lq}}\epsilon_{MSE}(n) = \frac{\partial \epsilon_{MSE}(n)}{\partial \hat{y}_{qn}} \cdot \frac{\partial \hat{y}_{qn}}{\partial \beta_{qn}} \cdot \frac{\partial \beta_{qn}}{\partial w_{lq}}$$

1 항 2 항 3 항

우변 1항

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

$$= \frac{\partial}{\partial \hat{y}_{qn}} \left\{ (\hat{y}_{0n} - y_{0n})^2 + \cdots + (\hat{y}_{qn} - y_{qn})^2 + \cdots + (\hat{y}_{(Q-1)n} - y_{(Q-1)n})^2 \right\}$$

$$= 2(\hat{y}_{qn} - y_{qn})$$

우변 2항

$$\frac{\partial \hat{y}_{qn}}{\partial \beta_{qn}} = \frac{\partial}{\partial \beta_{qn}} f(\beta_{qn}) = f(\beta_{qn})\left(1 - f(\beta_{qn})\right) = \hat{y}_{qn}(1 - \hat{y}_{qn})$$

(참고) 시그모이드 함수의 미분

$$f'(x) = f(x)(1 - f(x))$$

Error Back-Propagation 알고리즘

- $\frac{\partial}{\partial w_{lq}}\epsilon_{MSE}(n)$ 구하기

$$\frac{\partial}{\partial w_{lq}}\epsilon_{MSE}(n) = \frac{\partial \epsilon_{MSE}(n)}{\partial \hat{y}_{qn}} \cdot \frac{\partial \hat{y}_{qn}}{\partial \beta_{qn}} \cdot \frac{\partial \beta_{qn}}{\partial w_{lq}}$$

우변 3항

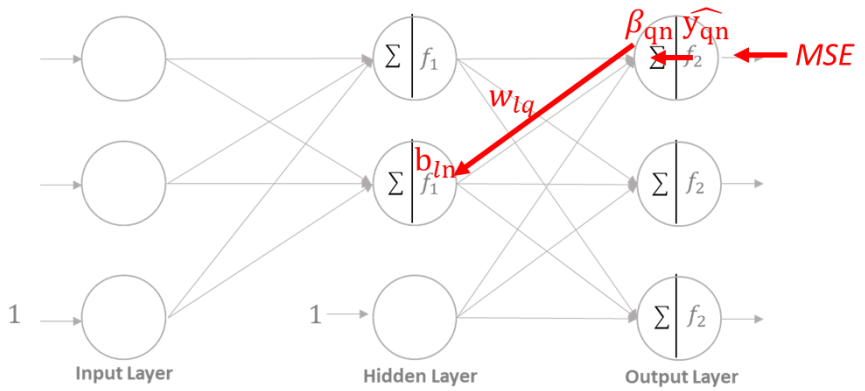
$$\frac{\partial \beta_{qn}}{\partial w_{lq}} = \frac{\partial}{\partial w_{lq}} \sum_{j=0}^{L} w_{jq} b_{jn} = \frac{\partial}{\partial w_{lq}} \left(w_{0q} b_{0n} + \cdots + w_{lq} b_{ln} + \cdots w_{Lq} b_{Ln} \right) = b_{ln}$$

최종

$$\frac{\partial \epsilon_{MSE}(n)}{\partial w_{lq}} = \frac{\partial \epsilon_{MSE}(n)}{\partial \hat{y}_{qn}} \cdot \frac{\partial \hat{y}_{qn}}{\partial \beta_{qn}} \cdot \frac{\partial \beta_{qn}}{\partial w_{lq}}$$

$$= 2\left(\hat{y}_{qn} - y_{qn}\right) \hat{y}_{qn}\left(1 - \hat{y}_{qn}\right) b_{ln}$$

$$= \delta_{qn} b_{ln}$$



- *Forward-propagation 기준*
  - w_{lq} *통과* ***전*** *입력값 =>* b_{ln}
  - w_{lq} *통과* ***후*** *지나게 되는 모든 Node*
    => 출력층 q 번째 Node

δ_{qn}: q번째 output node에 대해,
2*output*(output - target)*(1-output)

Error Back-Propagation 알고리즘

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

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

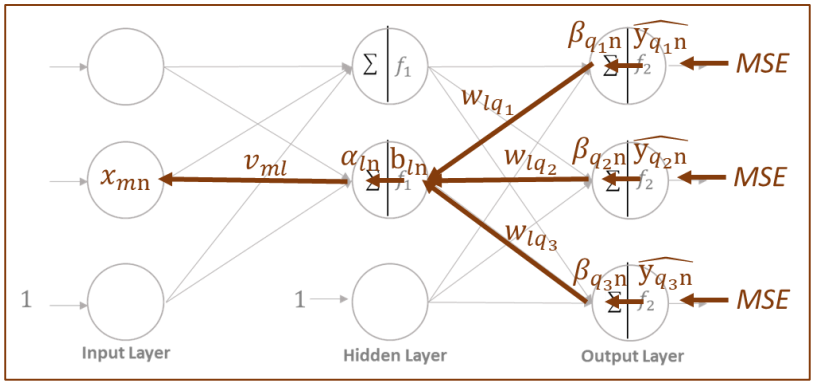
$$= 2 \sum_{q=0}^{Q-1} (\hat{y}_{qn} - y_{qn}) \frac{\partial}{\partial b_{ln}} f\left(\sum_{j=0}^{L} w_{jq} b_{jn} \right)$$

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

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

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

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



l 번째 Hidden Node와 연결된 weight들과 Output Node를 고려하게 됨.

Error Back-Propagation 알고리즘

- $\frac{\partial}{\partial v_{ml}}\epsilon_{MSE}(n)$ 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 2항

$$\frac{\partial b_{ln}}{\partial \alpha_{ln}} = \frac{\partial}{\partial \alpha_{ln}} f(\alpha_{ln}) = f(\alpha_{ln})\big(1 - f(\alpha_{ln})\big) = b_{ln}(1 - b_{ln})$$

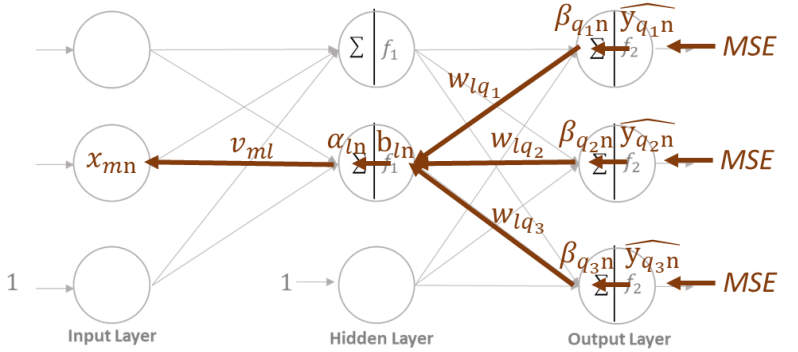
우변 3항

$$\frac{\partial \alpha_{ln}}{\partial v_{ml}} = \frac{\partial}{\partial v_{ml}} \sum_{j=0}^{M} v_{jl} x_{jn} = \frac{\partial}{\partial v_{ml}} (v_{0l}x_{0n} + \cdots + v_{ml}x_{mn} + \cdots + v_{Ml}x_{Mn}) = x_{mn}$$

최종

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

$$= \sum_{q=0}^{Q-1} \delta_{qn} w_{lq}\, b_{ln}(1 - b_{ln}) x_{mn}$$



- v_{ml} 통과 **전** 입력값 => x_{mn}
- v_{ml} 통과 **후** 지나게 되는 모든 Node
  => l 번째 Hidden node, 모든 Output Node

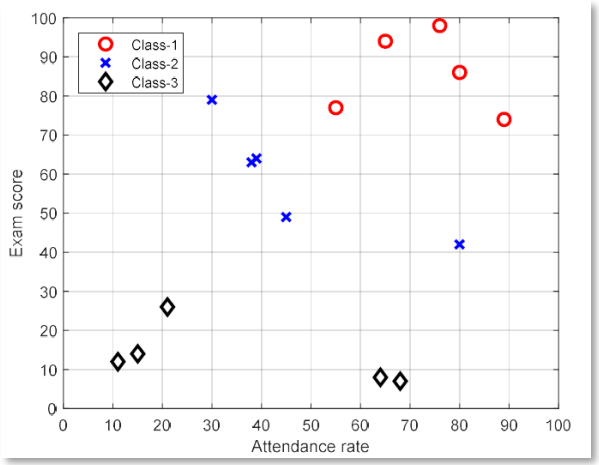
Neural Network 모델의 학습 알고리즘

- **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 $\hat{y}_{qn}$ 계산
     - (역전파) 실제값(Target)과 예측값(Output) 간의 오차로부터 Weight 업데이트 => 주의사항: Input layer에 가까운 weight부터 업데이트
5) 업데이트 된 weight로부터 accuracy 계산
6) 사용자가 입력한 epoch 수 만큼 3~4) 반복

Two-Layer Neural Network 적용 예시

- 데이터
  - 2입력 3클래스 분류 데이터
    - 입력 속성: 출석률, 시험성적
    - 출력 속성: 학점 (A, B, C)



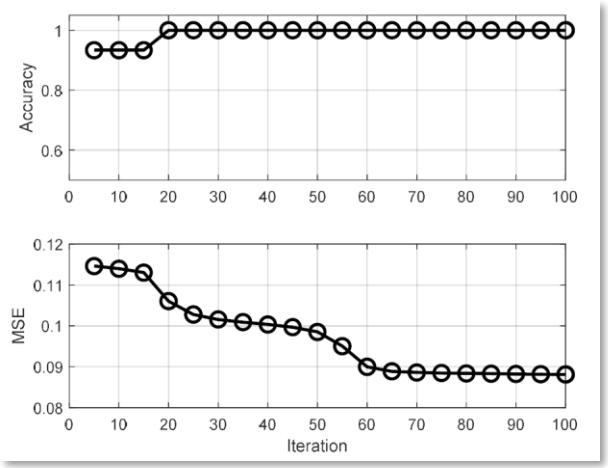
데이터 번호	입력		출력
	출석률	시험성적	학점
0	76	98	A
1	65	94	A
2	80	86	A
3	89	74	A
4	55	77	A
5	30	79	B
6	39	64	B
7	38	63	B
8	45	49	B
9	80	42	B
10	68	7	C
11	64	8	C
12	21	26	C
13	15	14	C
14	11	12	C

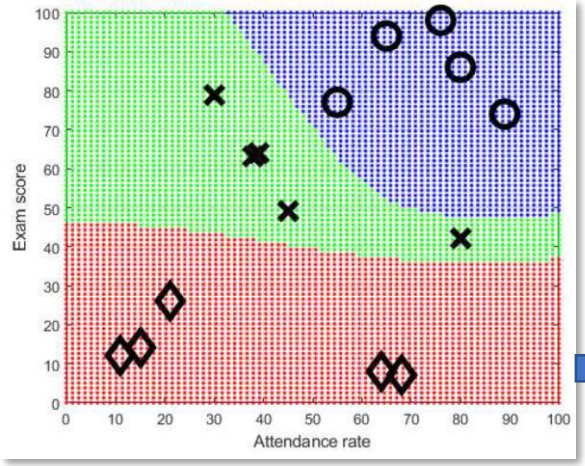
One-Hot Encoding

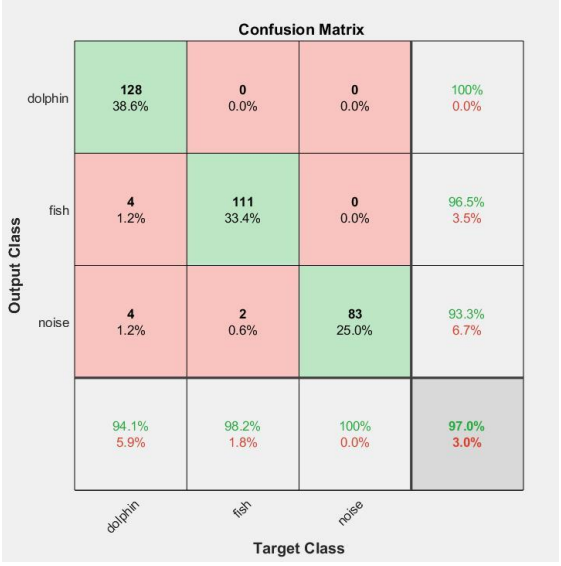
데이터 번호	입력		출력		
	출석률	시험성적	y_0	y_1	y_2
0	76	98	1	0	0
1	65	94	1	0	0
2	80	86	1	0	0
3	89	74	1	0	0
4	55	77	1	0	0
5	30	79	0	1	0
6	39	64	0	1	0
7	38	63	0	1	0
8	45	49	0	1	0
9	80	42	0	1	0
10	68	7	0	0	1
11	64	8	0	0	1
12	21	26	0	0	1
13	15	14	0	0	1
14	11	12	0	0	1

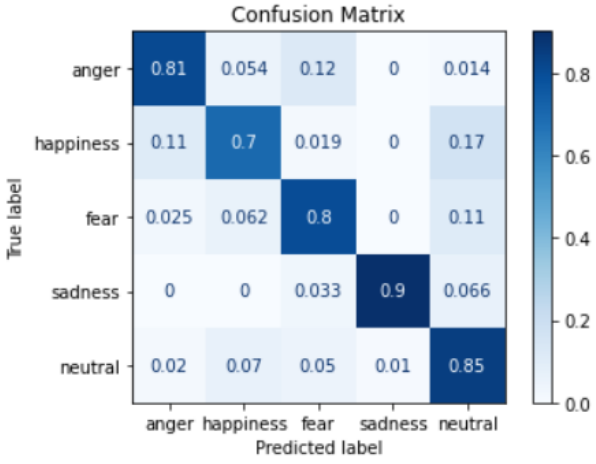
Two-Layer Neural Network 적용 예시

- 학습 성공 시,







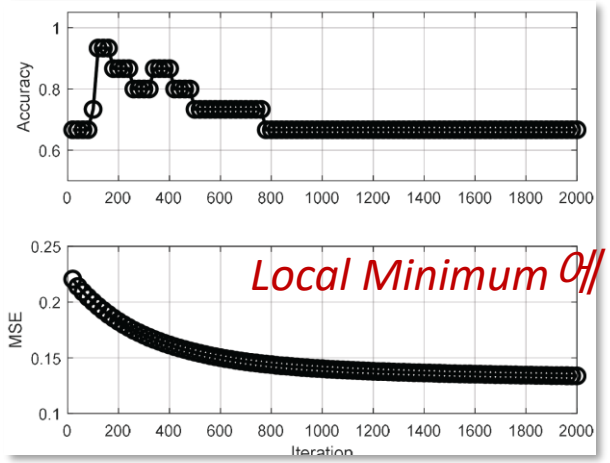


Confusion Matrix로 나타내 보자

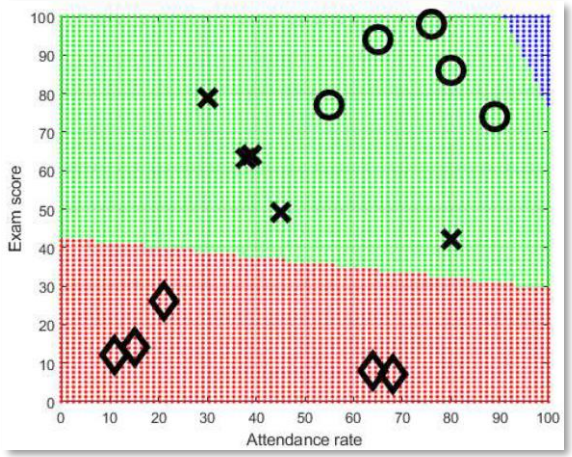
Confusion Matrix (모델 성능을 표현하는 가장 보편적 방법)

Two-Layer Neural Network 적용 예시

- 학습 실패 시,



Local Minimum 에 빠짐



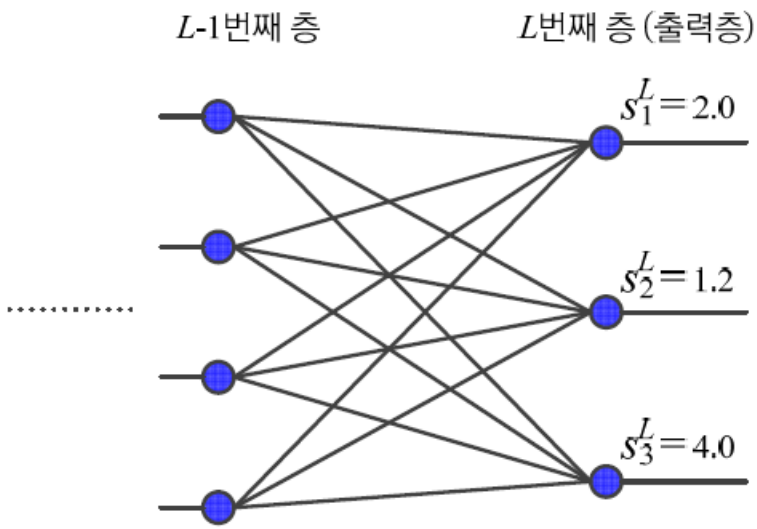
위의 결과를 Confusion Matrix로 나타내 보자

Softmax Function

- **다중 분류를 위한 Softmax Function**

$$o_j = \frac{e^{s_j}}{\sum_{i=1,c} e^{s_i}}$$

  - 동작 예시
    - softmax는 max를 모방(출력 노드의 중간 계산 결과 s_i^L에서 최댓값은 더욱 활성화하고 작은 값은 억제
    - 모두 더하면 1이 되어 확률 모방



로지스틱 시그모이드	max	softmax
0.8808	0.0	0.1131
0.7685	0.0	0.0508
0.9820	1.0	0.8360

Softmax Function

- **Softmax를 출력함수로 사용할 때의 Cost Function**
  - **Cross-Entropy**

❖ 이진분류 =>

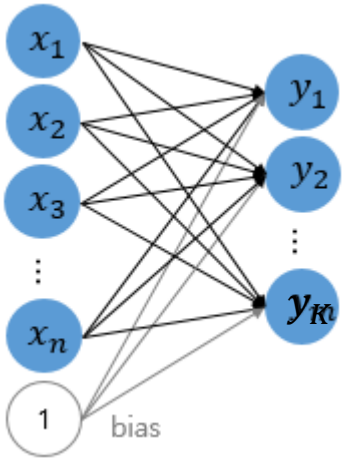
$$\epsilon_{CEE} = -\frac{1}{N}\sum_{n=0}^{N-1}\{y_n \ln p_n + (1-y_n)\ln(1-p_n)\}$$

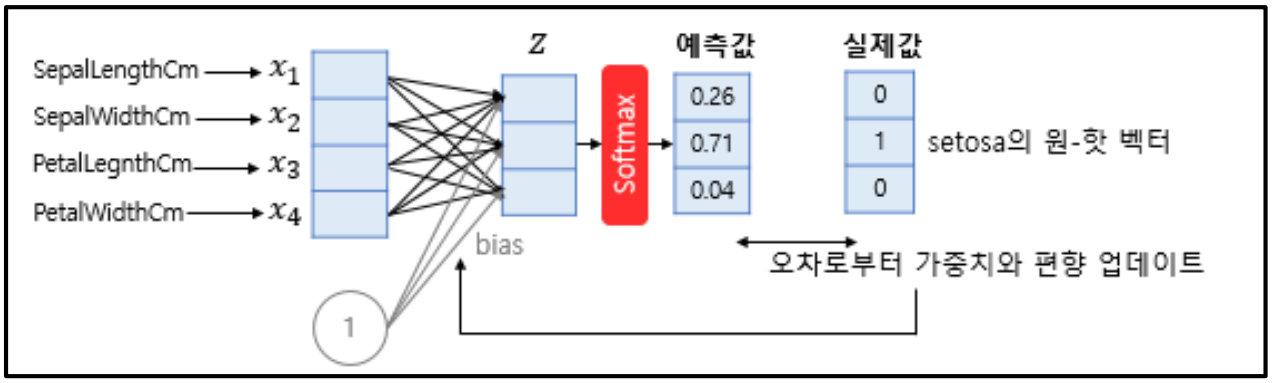
$$\{y_n \ln p_n + (1-y_n)\ln(1-p_n)\}$$

$$= \{y_{0,n}\ln p_{0,n} + (y_{1,n})\ln(p_{1,n})\}$$

$$= \sum_{k=0}^{1} y_{k,n} ln p_{k,n} \Rightarrow \sum_{k=0}^{K-1} y_{k,n}\ln(p_{k,n})$$

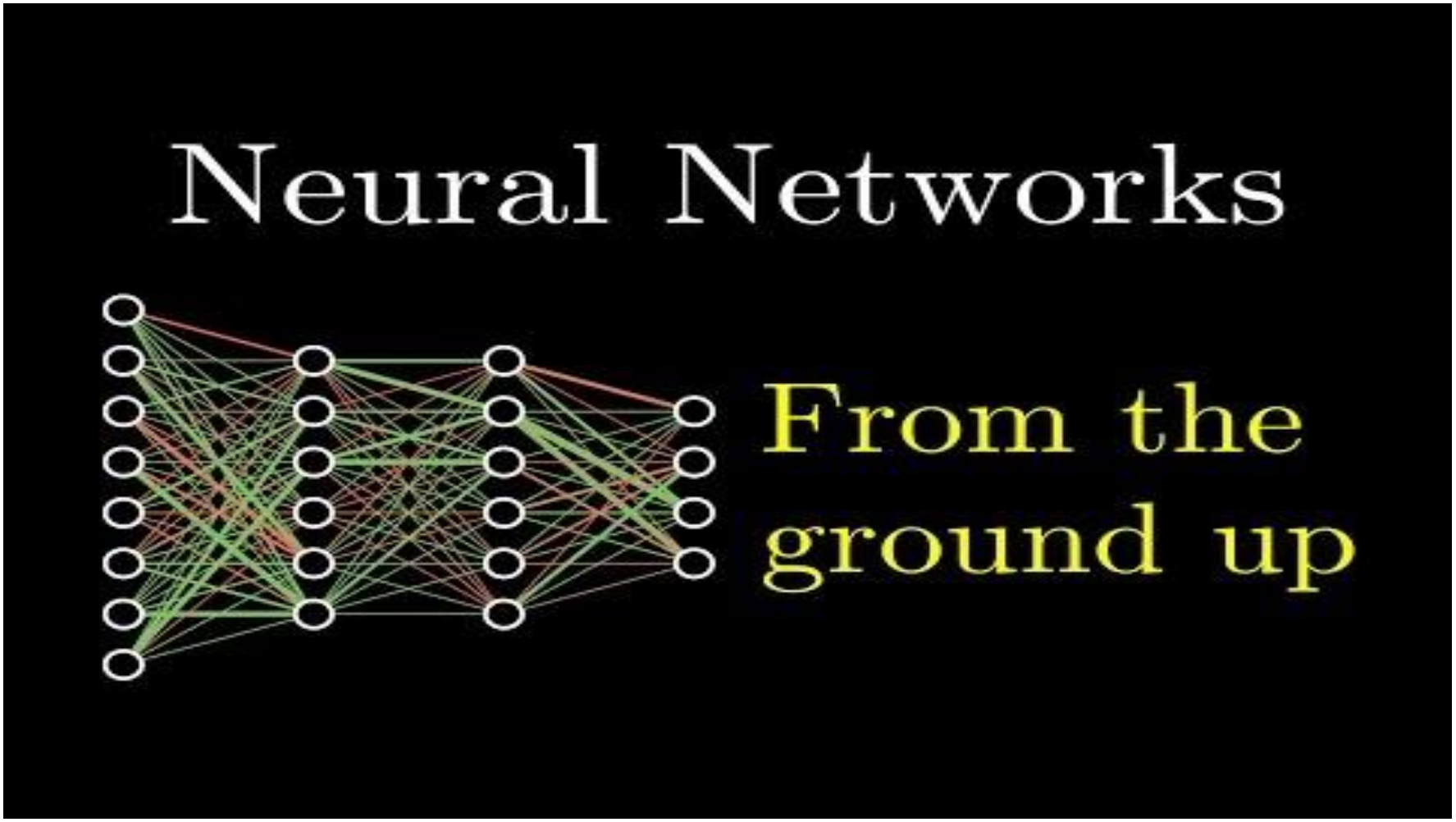
$$\epsilon_{CEE} = -\frac{1}{N}\sum_{n=0}^{N-1}\sum_{k=0}^{K-1} y_{i,n}\ln(p_{i,n})$$



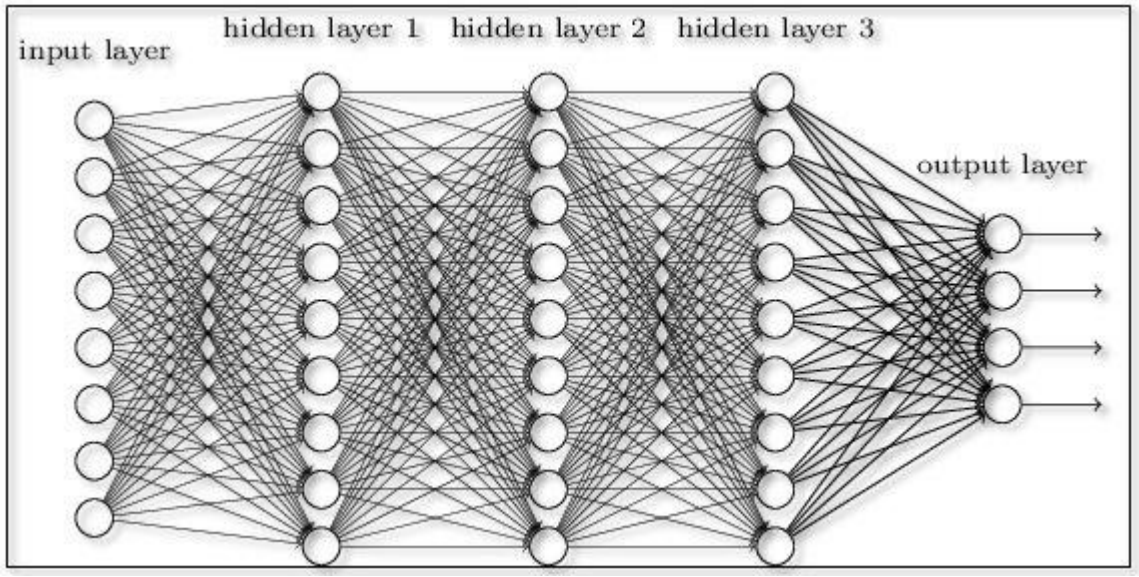


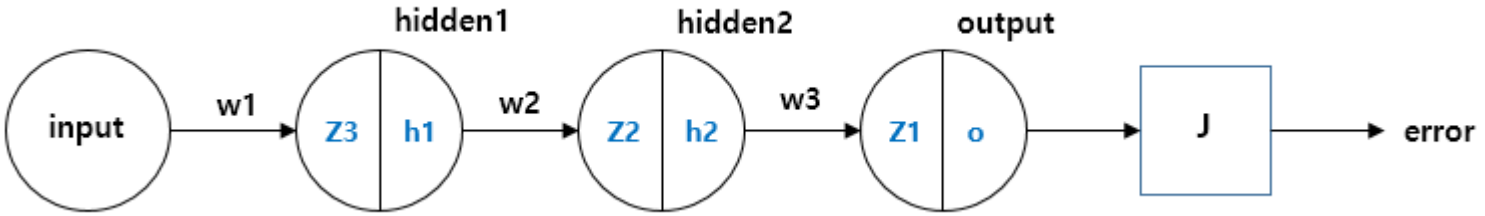
ANN 정리

https://www.youtube.com/watch?v=aircAruvnKk



[참고] Deep Learning으로의 확장

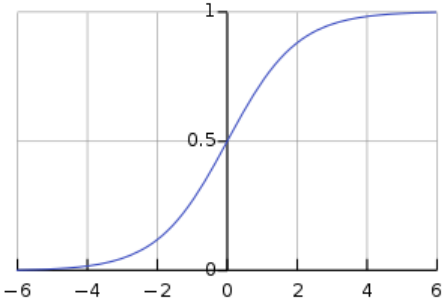




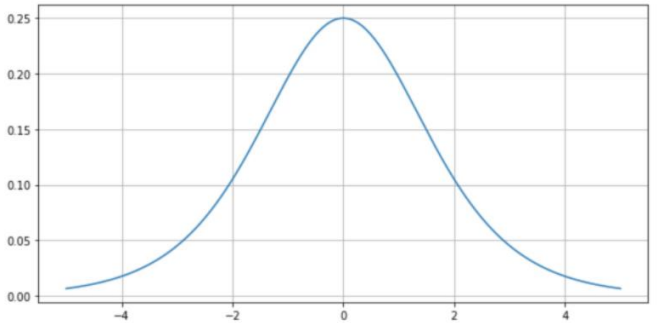
Node의 활성화 함수가 모두 동일하다고 했을 때,
w1을 update하기 위해서는 활성화 함수의 도함수를 3번 곱한 값이 들어감.

Deep Learning으로의 확장

Sigmoid 함수



Sigmoid의 도함수



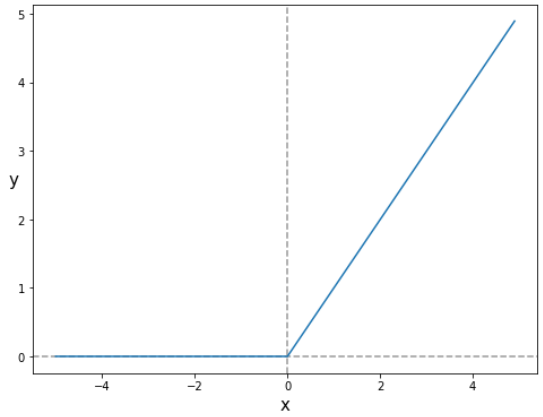
"Gradient Vanishing Problem"

도함수를 곱할수록 값이 작아져
Weight Update가 잘 이뤄지지 않음.
=> Hidden Layer가 많을수록 불리

ReLU 함수

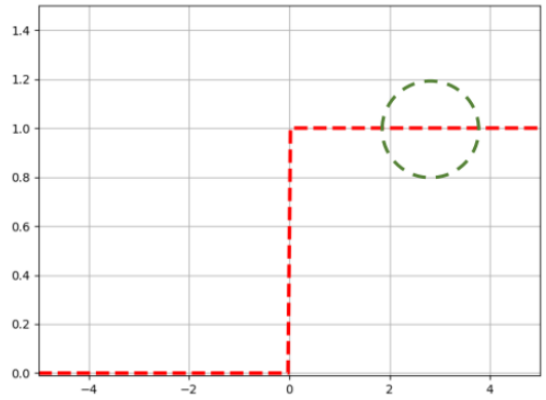
Gradient Vanishing 문제를
해결하기 위해 제안된 활성화 함수

ReLU



$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \qquad f(x) = max(0, x)$$

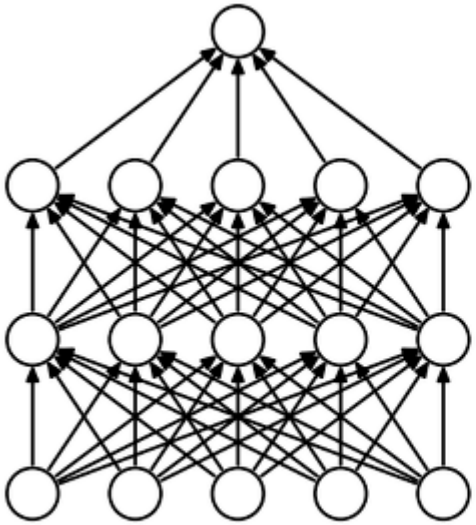
dReLU(x)/dx



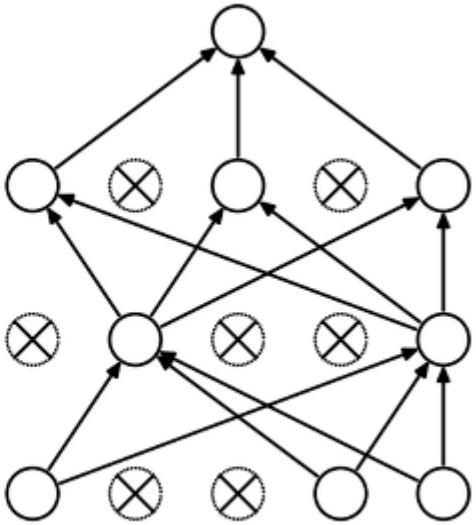
Hidden Layer가 많아도,
즉, Neural Network가 Deep해져도
Weight를 업데이트 할 수 있음

Dropout

- **Dropout (드롭아웃)**
  - 신경망의 일부 Node를 랜덤으로 비활성화하며 훈련시키는 기법
  - Regularization (Overfitting을 방지하기 위해 사용)



(a) Standard Neural Net



(b) After applying dropout.

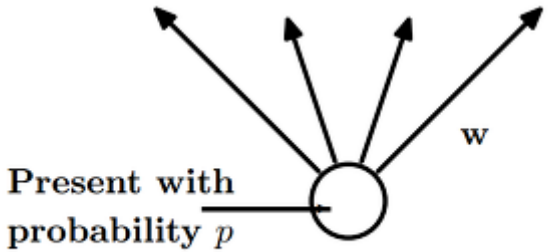
각 node가 존재할 확률 p

통상 알려진 default *입력층: 0.8* *은닉층: 0.5*

출력층은?

Dropout

- **Dropout에서의 Training 단계**



(a) At training time

l번째 은닉층의 j번째 노드의 연산:

$$z_j^l = \tau_l(s_j^l)$$

이때 $s_j^l = \mathbf{u}_j^l \mathbf{z}^{l-1}$

$\Rightarrow$

드롭아웃 적용:

$$z_j^l = \tau_l(s_j^l)$$

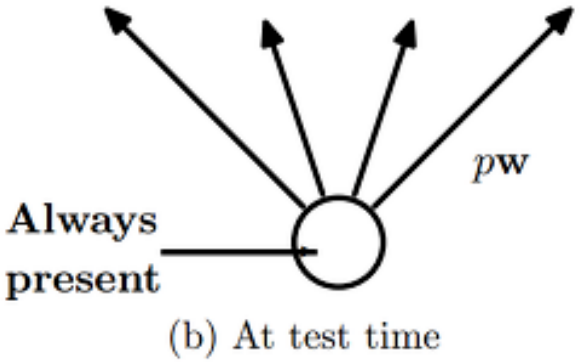
이때 $\begin{cases} \tilde{\mathbf{z}}^{l-1} = \mathbf{z}^{l-1} \odot \boldsymbol{\pi}^{l-1} \\ s_j^l = \mathbf{u}_j^l \tilde{\mathbf{z}}^{l-1} \end{cases}$

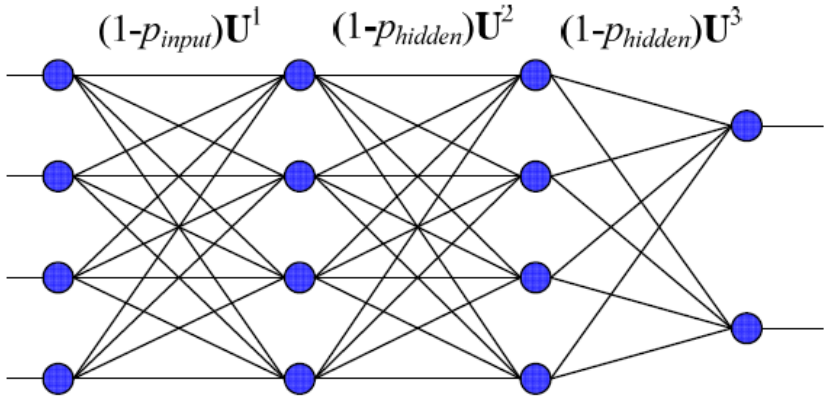
Boolean 배열 $\boldsymbol{\pi}$에 노드 제거 여부를 표시

Boolean 배열: 배열 내부에 True or False만 존재

Dropout

- **Dropout에서의 Test 단계**
  - 가중치에 생존 비율(p)을 곱하여 전방 계산
  - 학습 과정에서 가중치가 생존 비율(p) 만큼만 참여했기 때문





Chap.4 실습 - 2주차

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

1) Error Back-Propagation 알고리즘 구현
   - Chap. 4 강의자료 24 page에 나와있는 알고리즘을 사용자 지정함수로 구현

2) Two-Layer Neural Network “Training”
   - “NN_data.csv”를 7:3 으로 Training set :Test set 분할
   - Training set과 1)에서 구현한 알고리즘으로부터 Two-Layer Neural Network를 Training
   - Epoch에 따른 Accuracy 및 MSE 그래프 도출
   - Hyper-parameter tuning을 통해 최적화

3) Two-Layer Neural Network “Test”
   - 2)에서 도출한 Two-Layer Neural Network에 Testing set 삽입
   - Confusion Matrix 도출