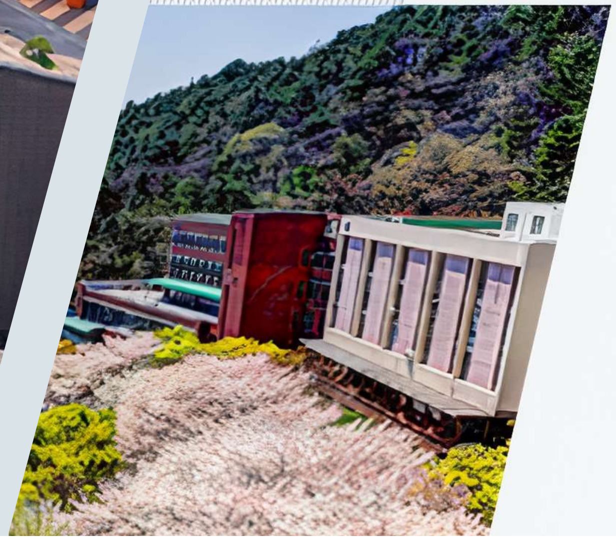
Dong-A Univ. (ISPL)

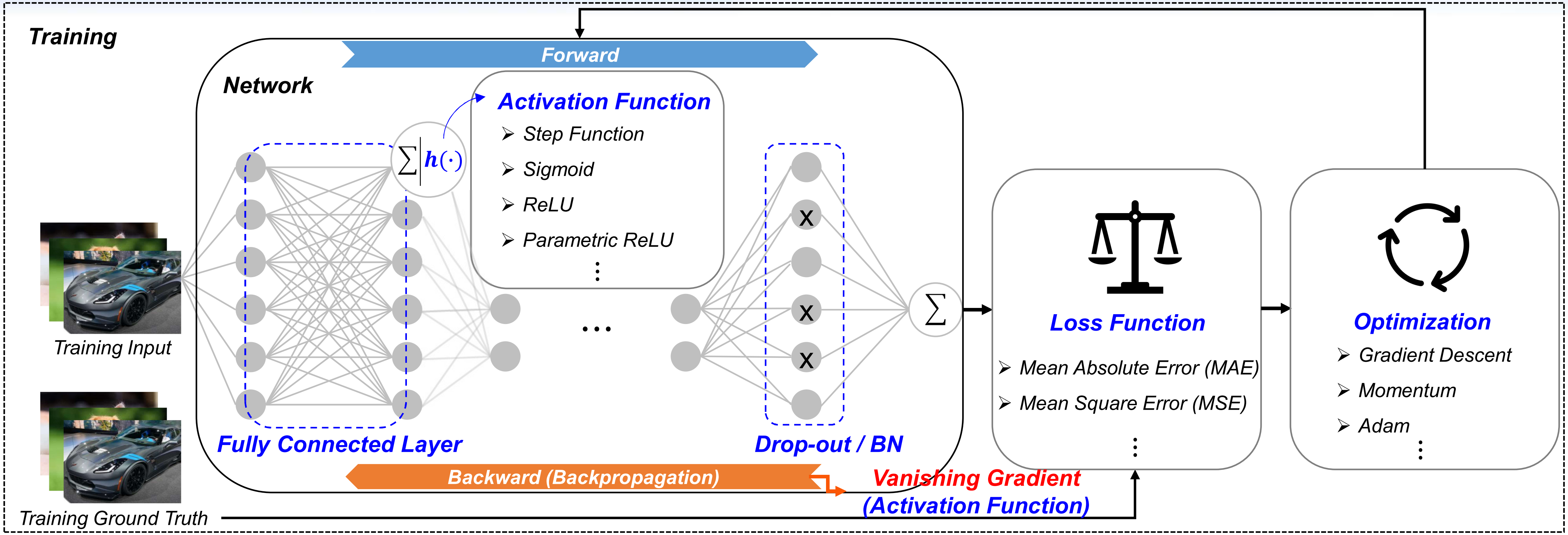
Overview of Multi Layer Perceptron

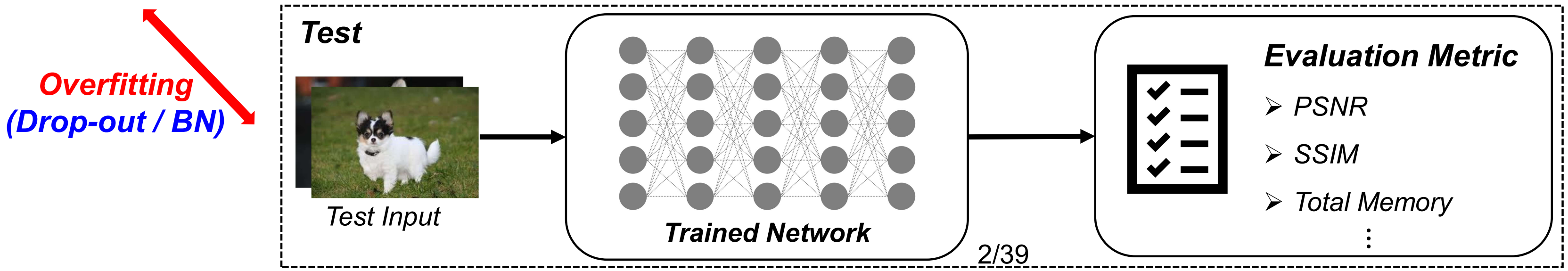
동아대학교
DONG-A UNIVERSITY

컴퓨터AI공학부 AI학과
2024년 1학기 인공지능



Overall Architecture of Deep Learning



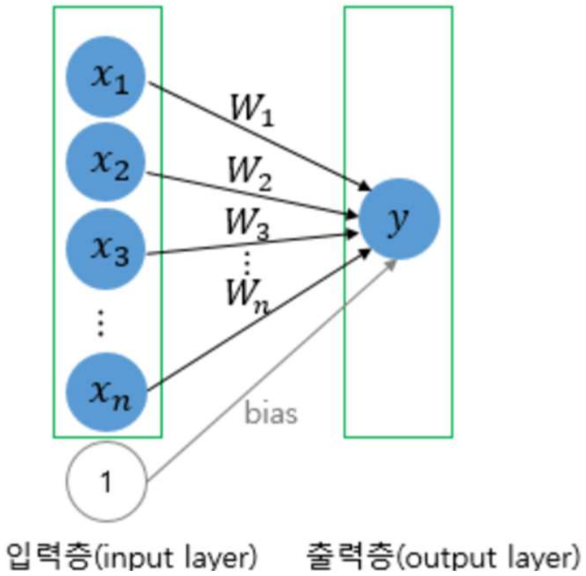


- ReLU: Rectified Linear Unit
- Adam: Adaptive Moment Estimation
- PSNR: Peak Signal-to-Noise Ratio
- SSIM: Structural Similarity Index Measure

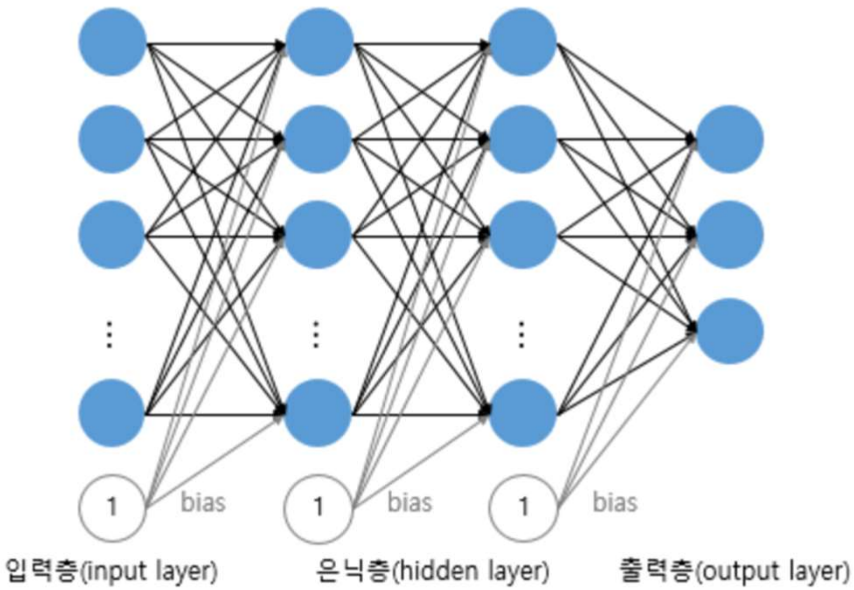
동아대학교 DONG-A UNIVERSITY

Multi Layer Perceptron

- **다층 퍼셉트론 (Multi Layer Perceptron, MLP)**
  - 여러 개의 층으로 구성된 perceptron 모델
  - 입력층, 은닉층, 출력층으로 구성됨



Perceptron 구조 예시

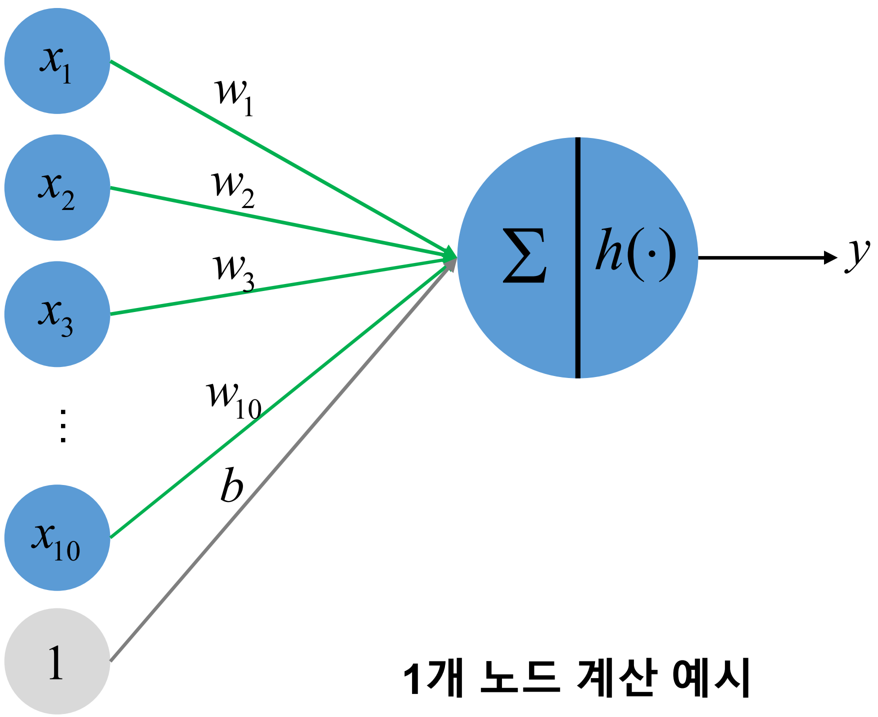


Multi Layer Perceptron (MLP) 예시

Multi Layer Perceptron

- **다층 퍼셉트론 (Multi Layer Perceptron, MLP)**
  - 각 입력(x)에 대응되는 weight(w), 1개의 노드에 입력되는 bias(b) 존재
  - 가중합으로 얻어진 결과치에 활성화 함수(h) 적용

X = Input Data
W^T = Weight
b = Bias
h = Activation Function



1개 노드 계산 예시

$$y = h(w_1x_1 + w_2x_2 + \ldots + w_{10}x_{10} + b)$$

$$\boxed{y = h(W^TX + b)}$$

$$W = \begin{pmatrix} w_1 \\ w_2 \\ \cdots \\ w_{10} \end{pmatrix}, \qquad X = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_{10} \end{pmatrix}$$

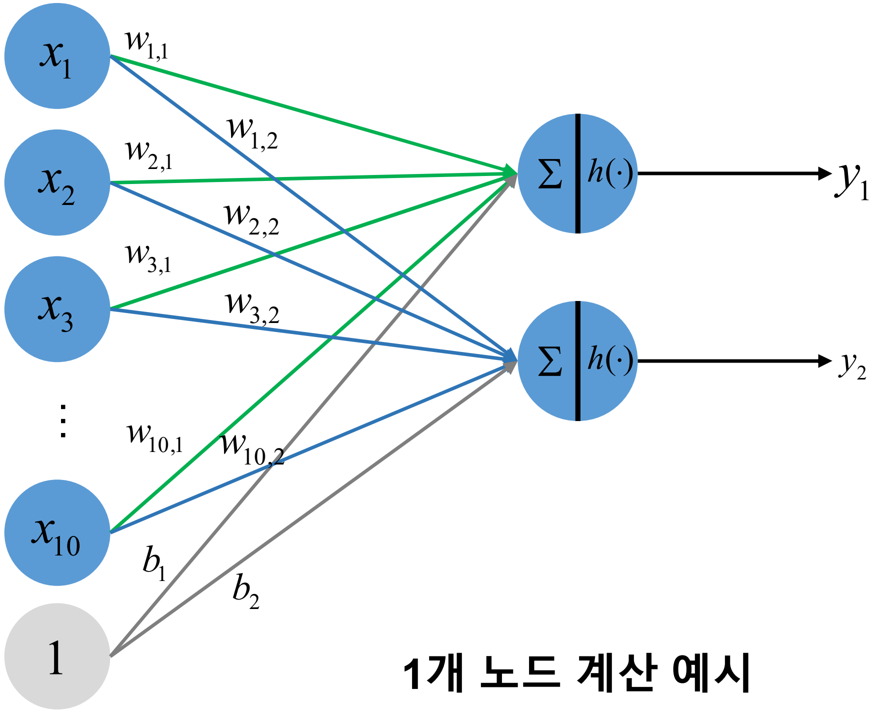
$$W^T = \begin{pmatrix} w_1 & w_2 & \ldots & w_{10} \end{pmatrix}$$

동아대학교 DONG-A UNIVERSITY

Multi Layer Perceptron

- **다층 퍼셉트론 (Multi Layer Perceptron, MLP)**
  - 각 입력(x)에 대응되는 weight(w), 1개의 노드에 입력되는 bias(b) 존재
  - 가중합으로 얻어진 결과치에 활성화 함수(h) 적용

X = Input Data
W^T = Weight
b = Bias
h = Activation Function



1개 노드 계산 예시

$$y_1 = h(w_{1,1}x_1 + w_{2,1}x_2 + \ldots + w_{10,1}x_{10} + b_1)$$

$$y_2 = h(w_{1,2}x_1 + w_{2,2}x_2 + \ldots + w_{10,2}x_{10} + b_2)$$

$$Y = h(W^T X + \boldsymbol{b})$$

❖ 1-layer perceptron 계산

$$W = \begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ \cdots & \cdots \\ w_{10,1} & w_{10,2} \end{pmatrix}, \qquad X = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_{10} \end{pmatrix}$$

$$W^T = \begin{pmatrix} \boxed{w_{1,1} \quad w_{2,1} \quad \cdots \quad w_{10,1}} \\ \boxed{w_{1,2} \quad w_{2,2} \quad \cdots \quad w_{10,2}} \end{pmatrix}$$

Multi Layer Perceptron

- **다층 퍼셉트론 (Multi Layer Perceptron, MLP)**
  - 각 입력(x)에 대응되는 weight(w), 1개의 노드에 입력되는 bias(b) 존재
  - 가중합으로 얻어진 결과치에 활성화 함수(h) 적용

X = Input Data
W^T = Weight
b = Bias
h = Activation Function

$$Y = h(W^T X + \boldsymbol{b})$$

❖ 1-layer perceptron 계산

$$= h\left(\underbrace{\begin{pmatrix} w_{1,1} & w_{2,1} & \cdots & w_{10,1} \\ w_{1,2} & w_{2,2} & \cdots & w_{10,2} \end{pmatrix}}_{\mathbf{2 \times 10}} \cdot \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_{10} \end{pmatrix}}_{\mathbf{10 \times 1}} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)$$

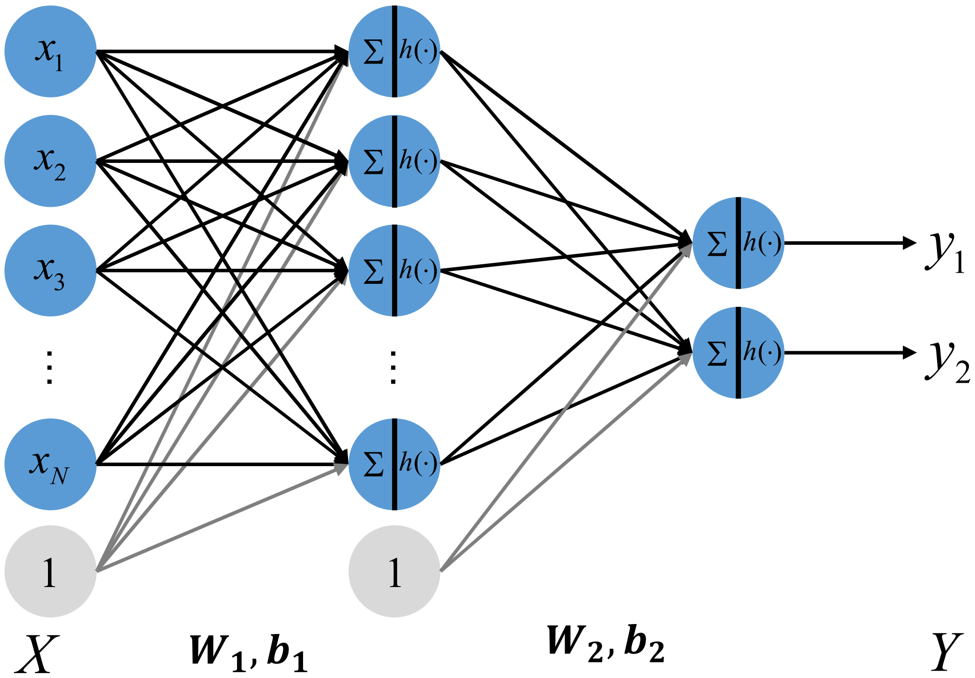
$$= \begin{pmatrix} h\left(w_{1,1}x_1 + w_{2,1}x_2 + \ldots + w_{10,1}x_{10} + b_1\right) \\ h\left(w_{1,2}x_1 + w_{2,2}x_2 + \ldots + w_{10,2}x_{10} + b_2\right) \end{pmatrix} = \underbrace{\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}}_{\mathbf{2 \times 1}}$$

동아대학교 DONG-A UNIVERSITY

Multi Layer Perceptron

- **다층 퍼셉트론 (Multi Layer Perceptron, MLP)**
  - 각 입력(x)에 대응되는 weight(w), 1개의 노드에 입력되는 bias(b) 존재
  - 가중합으로 얻어진 결과치에 활성화 함수(h) 적용

X = Input Data
W^T = Weight
b = Bias
h = Activation Function

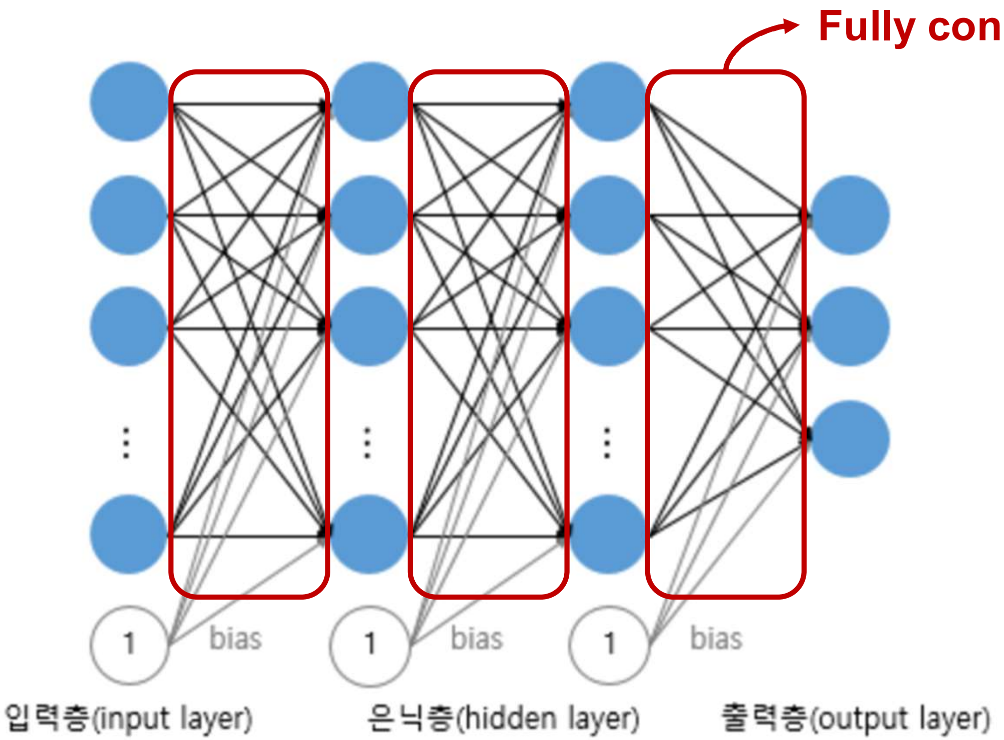


$$Y = h(W_2^T h(W_1^T X + \boldsymbol{b_1}) + \boldsymbol{b_2})$$

❖ 2-layer perceptron 계산

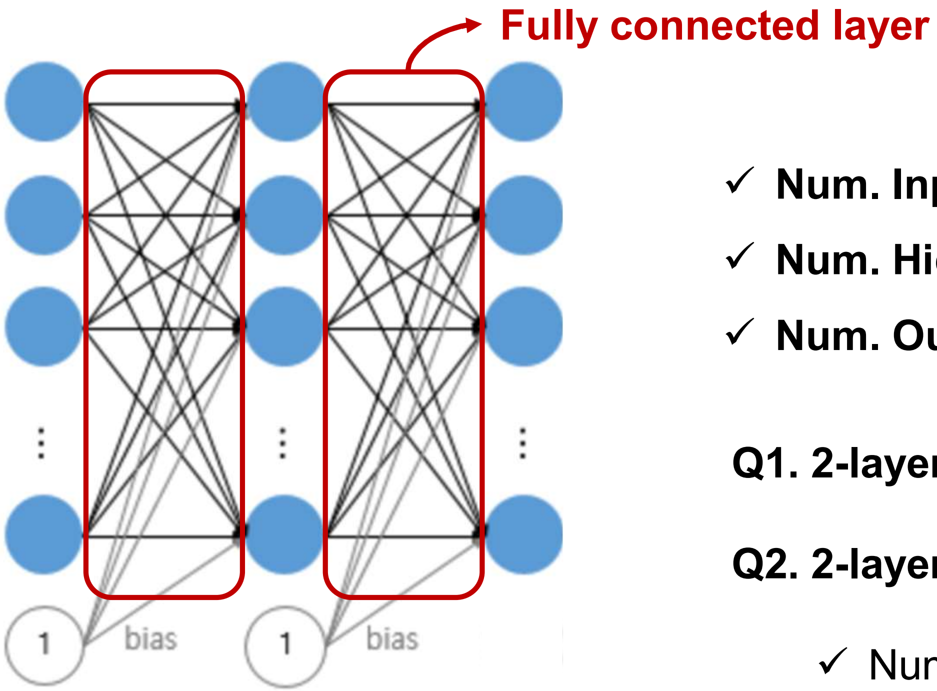
Multi Layer Perceptron - Fully Connected Layer

- **전 결합 계층 (Fully Connected layer, FC layer)**
  - 각 층별로 모든 노드가 연결되어 weight를 가지는 계층을 FC layer라고 함



Multi Layer Perceptron - Fully Connected Layer

- **전 결합 계층 (Fully Connected layer, FC layer)**
  - 각 층별로 모든 노드가 연결되어 weight를 가지는 계층을 FC layer라고 함



- ✓ **Num. Input nodes = 10**
- ✓ **Num. Hidden nodes = 10**
- ✓ **Num. Output nodes = 20**

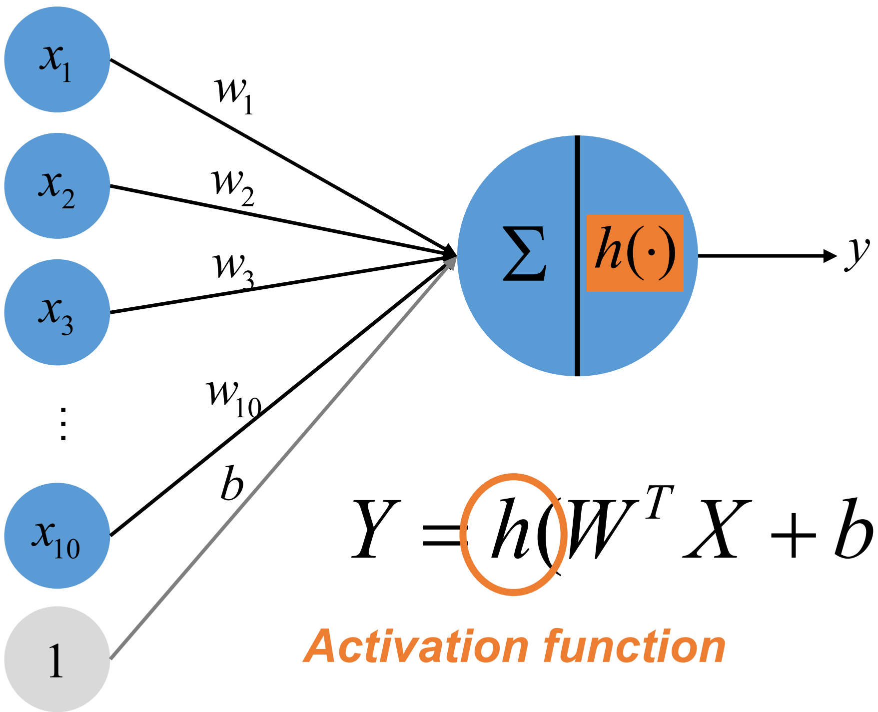
Q1. 2-layer perceptron에서 가지는 weight (w) 개수?

Q2. 2-layer perceptron 가 가지는 bias (b) 개수?

- ✓ Num. weight ➔ 300
- ✓ Num. bias ➔ 30

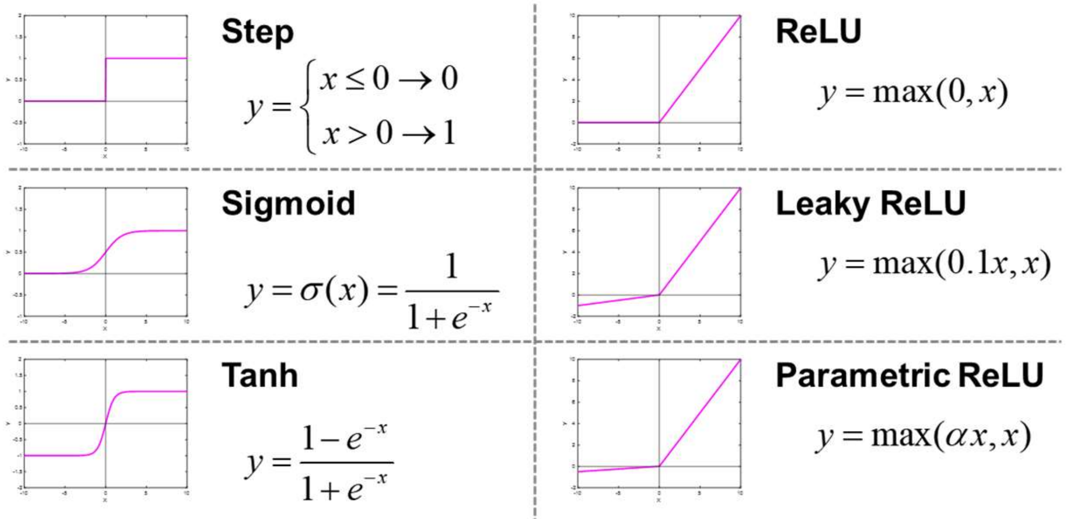
Multi Layer Perceptron - Activation Function

- **활성화 함수 (Activation function)**
  - 입력 값들의 가중 합을 통해 노드의 활성화 여부를 판단하는 함수



$$Y = h(W^T X + b)$$

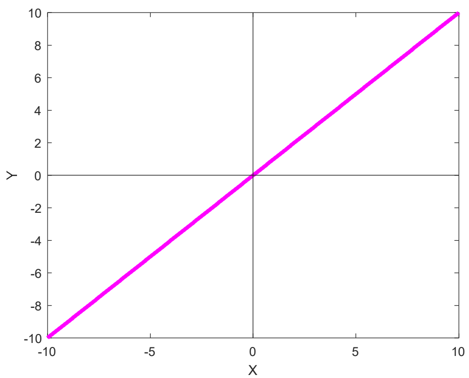
Activation function



동아대학교 DONG-A UNIVERSITY

Multi Layer Perceptron - Activation Function

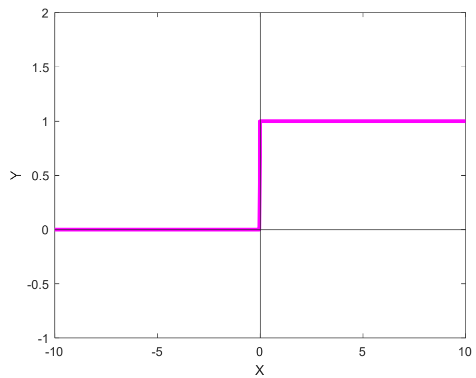
- **Linear function**
  - 입력 값을 그대로 출력으로 내보내는 함수
  - Regression (회귀)문제의 출력층에서 주로 사용됨



$$y = x$$

Linear function 출력 결과

- **Step function**
  - 0 이하이면 0, 0보다 크면 1을 출력
  - 역전파를 통한 학습 불가능 (미분불가)
  - 이진분류시 출력층에서 사용하기 적합함

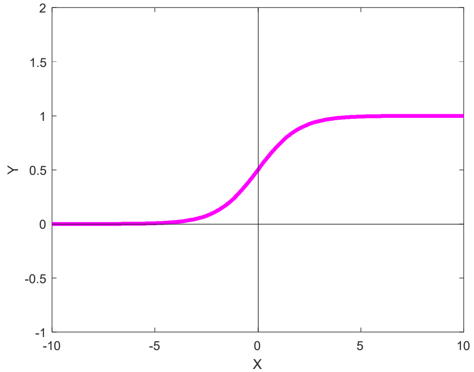


$$y = \begin{cases} x \le 0 \rightarrow 0 \\ x > 0 \rightarrow 1 \end{cases}$$

Tanh function 출력 결과

Multi Layer Perceptron - Activation Function

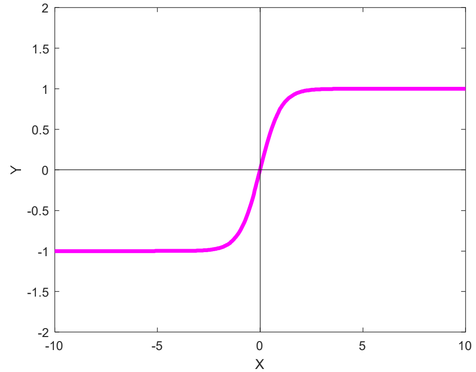
- **Sigmoid function**
  - 0~1 사이 실수를 출력 → 확률로 해석 가능
  - 기울기가 발생하지 않는 지점이 존재함
  - exp 연산의 속도가 느림



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

Sigmoid function 출력 결과

- **Hypergolic tangent (tanh) function**
  - -1~1 사이 실수를 출력
  - Sigmoid 함수보다 출력의 범위가 큼
  - Sigmoid 함수보다 최대 기울기가 큼

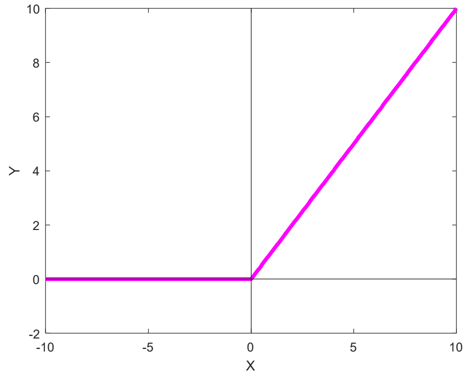


$$y = \frac{1-e^{-x}}{1+e^{-x}}$$

Tanh function 출력 결과

Multi Layer Perceptron - Activation Function

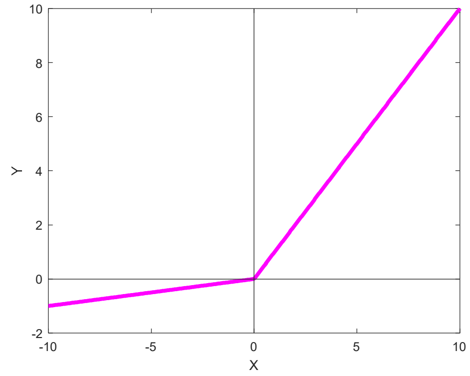
- **Rectified Linear Unit (ReLU) function**
  - 0 이하일때 0, 0 보다 크면 값을 그대로 출력
  - 단순한 연산으로 학습 속도가 빠름
  - 입력 값이 0이하인 경우 기울기는 항상 0



$$y = \max(0, x)$$

ReLU function 출력 결과

- **Leaky ReLU function**
  - 0 이하일 때 0.1을 곱한 값이 출력됨
  - ReLU 함수의 음수 구간 기울기 소실 문제를 보완



$$y = \max(0.1x, x)$$

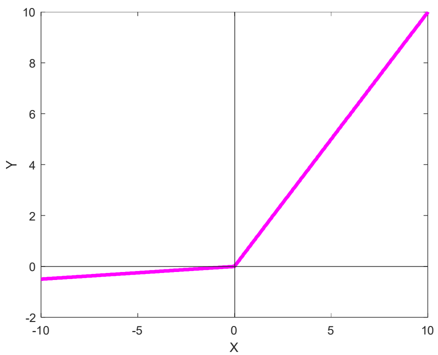
Leaky ReLU function 출력 결과

Multi Layer Perceptron - Activation Function

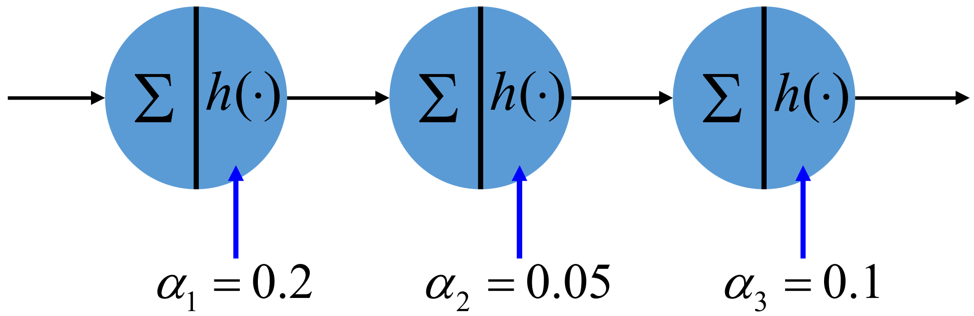
- **Parametric ReLU (PReLU) function**
  - Leaky ReLU와 유사
  - 음수 구간 기울기를 결정하는 p는 학습 가능한 파라미터

$$y = \max(\alpha x, x)$$

Trainable parameter



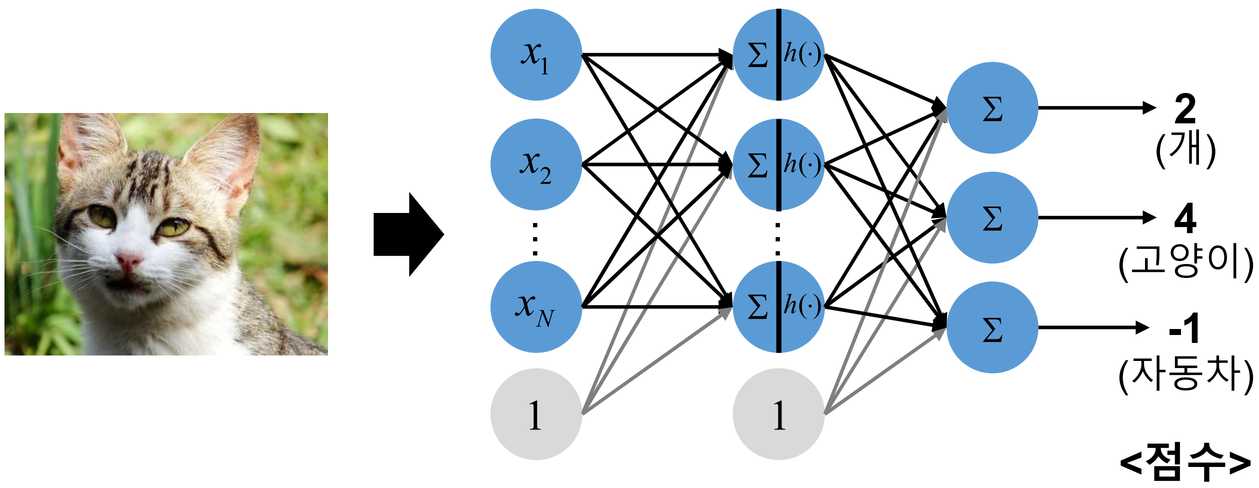
Parametric ReLU function 출력 결과



Perceptron의 PReLU 사용 예시

- **Softmax function**
  - 여러 개의 입력을 받아 각각의 확률 값으로 출력
  - 0~1 사이 실수를 출력하고, 모든 출력의 합은 1 (확률로 해석)
  - Multi-label classification 모델의 출력층에 주로 사용됨

$$y = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

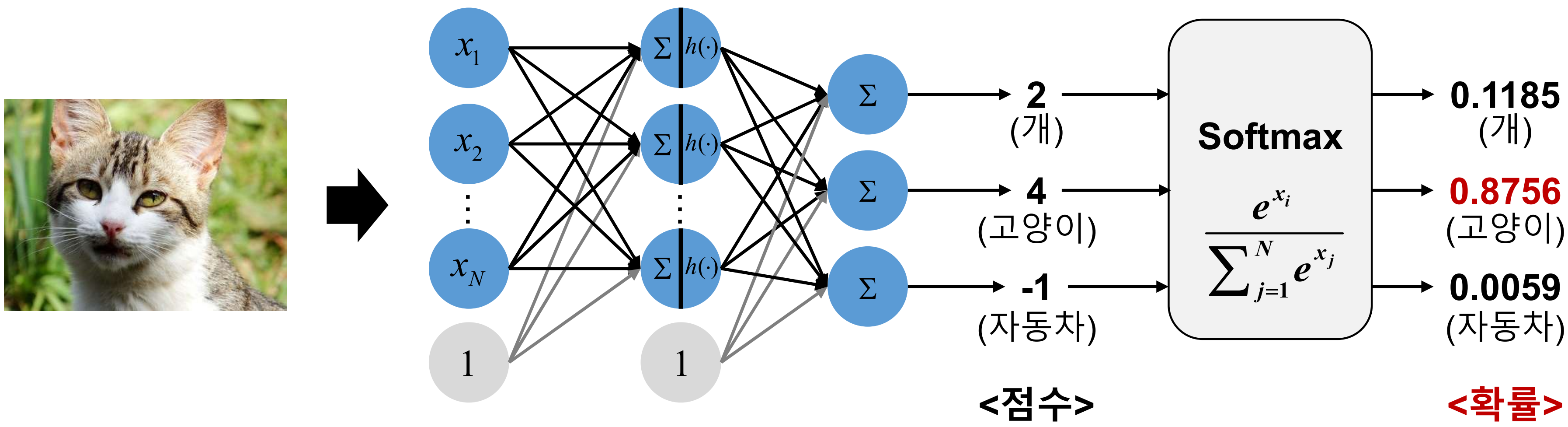


Softmax 함수 사용 예시 (개/고양이/자동차 분류)

Multi Layer Perceptron - Activation Function

- **Softmax function**
  - 여러 개의 입력을 받아 각각의 확률 값으로 출력
  - 0~1 사이 실수를 출력하고, 모든 출력의 합은 1 (확률로 해석)
  - Multi-label classification 모델의 출력층에 주로 사용됨

$$y = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

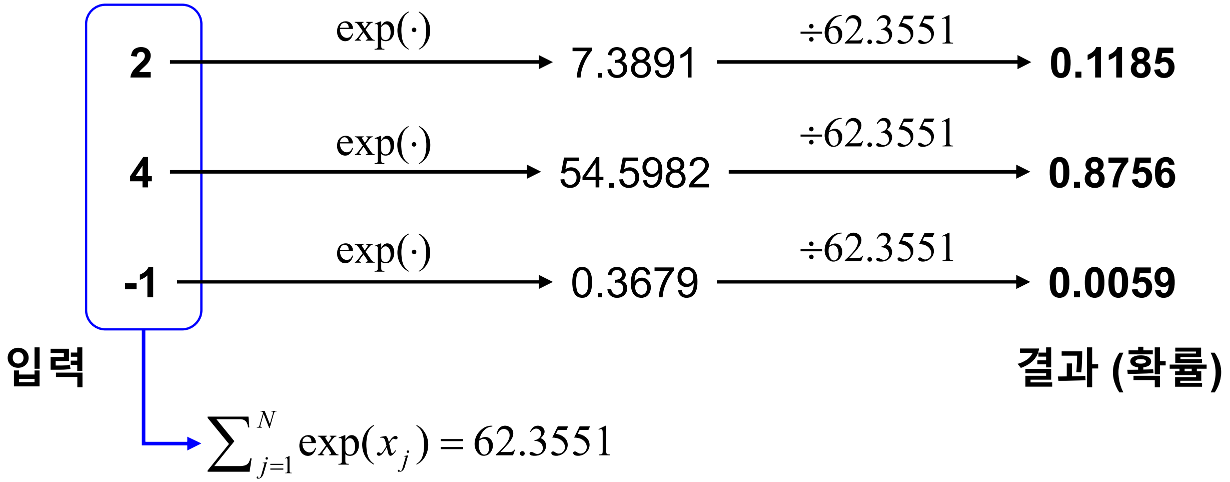


Softmax 함수 사용 예시 (개/고양이/자동차 분류)

Multi Layer Perceptron - Activation Function

- **Softmax function**
  - 여러 개의 입력을 받아 각각의 확률 값으로 출력
  - 0~1 사이 실수를 출력하고, 모든 출력의 합은 1 (확률로 해석)
  - Multi-label classification 모델의 출력층에 주로 사용됨

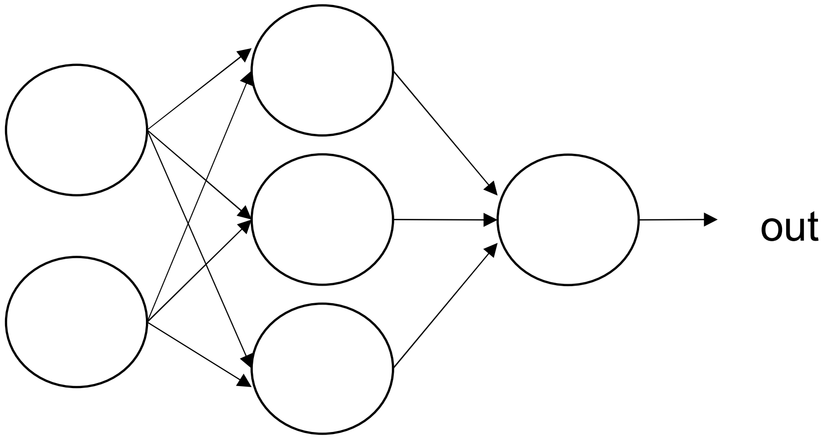
$$y = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$



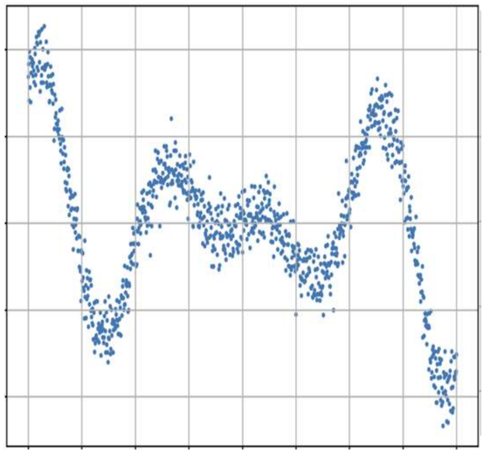
Softmax 함수 계산 예시

Multi Layer Perceptron - Activation Function

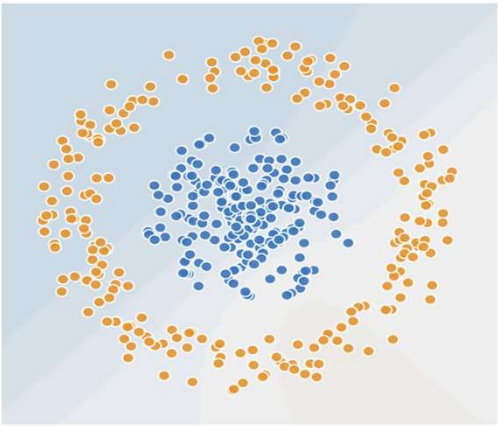
- **활성화 함수를 비선형 함수로 사용하는 이유**
  - 문제의 형상이 비선형 형태로 되어있는 경우



<Multi-Layer Perceptron>



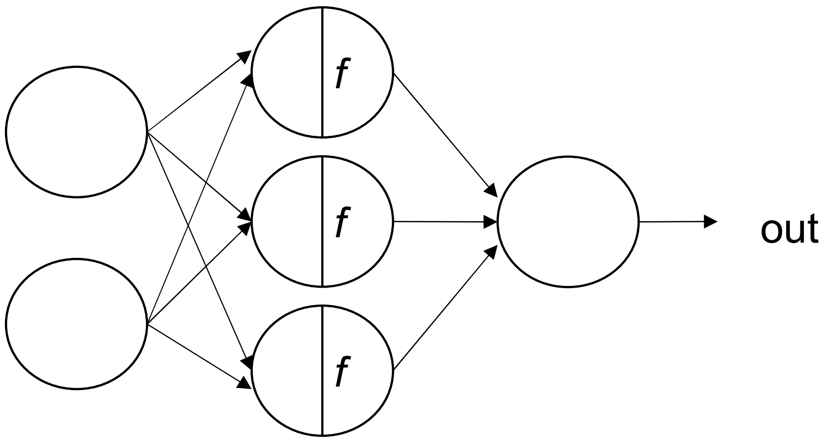
<Regression Problem>



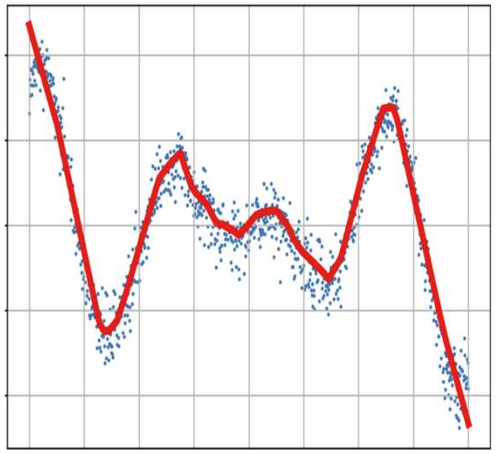
<Classification Problem>

Multi Layer Perceptron - Activation Function

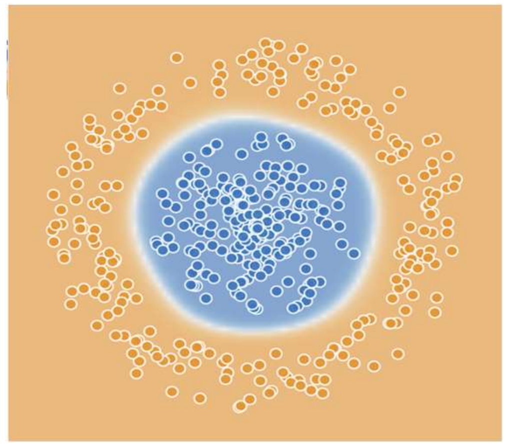
- **활성화 함수를 비선형 함수로 사용하는 이유**
  - 문제의 형상이 비선형 형태로 되어있는 경우 → 비선형 함수를 거쳐 표현 및 해결 가능



<Multi-Layer Perceptron>



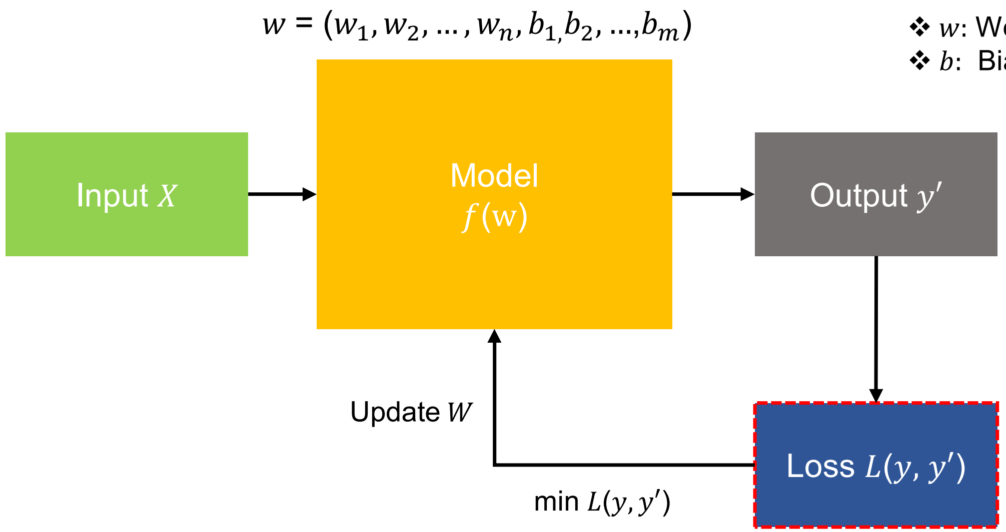
< Regression Problem>



<Classification Problem>

Multi Layer Perceptron – Loss Function

- **Loss function: 학습 모델이 얼마나 잘못 예측하고 있는지는 표현하는 지표**
  - 값이 낮을수록 모델이 정확하게 예측했다고 해석할 수 있음



❖ w: Weight
❖ b: Bias

✓ **평균 절대 오차 (Mean Absolute Error, MAE)**

$$MAE(y, y') = \frac{1}{N}\sum_{i=1}^{N} | y_i - y_i' |$$

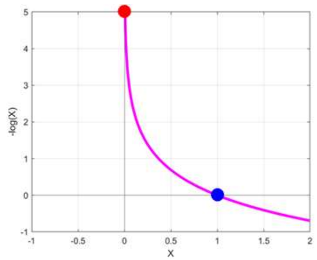
❖ y: 정답 값
❖ y': 예측 값

✓ **평균 제곱 오차 (Mean Squared Error, MSE)**

$$MSE(y, y') = \frac{1}{N}\sum_{i=1}^{N} (y_i - y_i')^2$$

✓ **교차 엔트로피 오차 (Cross Entropy Error, CEE)**

$$CEE(y, y') = -\sum_{i=1}^{N} y_i \times \log(y_i')$$



➢ 정답 예측 확률이 1인 경우 CEE = 0
➢ 정답 예측 확률이 0인 경우 CEE = 무한

Multi Layer Perceptron – Loss Function

- **Loss function: 학습 모델이 얼마나 잘못 예측하고 있는지는 표현하는 지표**
  - 값이 낮을수록 모델이 정확하게 예측했다고 해석할 수 있음
  - Ex. Cross Entropy Error (CEE) 계산 방법

$$\mathrm{CEE}(y, y') = -\sum_{i=1}^{N} y_i \times \log(y_i')$$

- ❖ y: 정답 값
- ❖ y': 예측 값

	0	1	2	3	4	5	6	7	8	9		
	0	0	**1**	0	0	0	0	0	0	0	정답 값 (y, one-hot)	
Model A의 예측 결과	0	0	**0.8**	0	0	0	**0.1**	0	**0.1**	0	예측 확률 (y')	CEE = 0.2231

$$\mathrm{CEE}(y, y') = -(1 \times \log(0.8)) = 0.2231$$

Multi Layer Perceptron – Loss Function

- **Loss function: 학습 모델이 얼마나 잘못 예측하고 있는지는 표현하는 지표**
  - 값이 낮을수록 모델이 정확하게 예측했다고 해석할 수 있음
  - Ex. Cross Entropy Error (CEE) 계산 방법

$$\mathrm{CEE}(y, y') = -\sum_{i=1}^{N} y_i \times \log(y_i')$$

❖ y: 정답 값
❖ y': 예측 값

	0	1	2	3	4	5	6	7	8	9		
	0	0	**1**	0	0	0	0	0	0	0	정답 값 (y, one-hot)	
Model A의 예측 결과	0	0	**0.8**	0	0	0	**0.1**	0	**0.1**	0	예측 확률 (y’)	CEE = 0.2231
Model B의 예측 결과	0	0	**0.2**	0	0	0	**0.2**	0	**0.6**	0	예측 확률 (y”)	CEE = 1.6094

$$\mathrm{CEE}(y, y'') = -(1 \times \log(0.2)) = 1.6094$$

동아대학교 DONG-A UNIVERSITY

Questions & Answers

Dongsan Jun (dsjun@dau.ac.kr)
Image Signal Processing Laboratory (www.donga-ispl.kr)
Dept. of AI
Dong-A University, Busan, Rep. of Korea