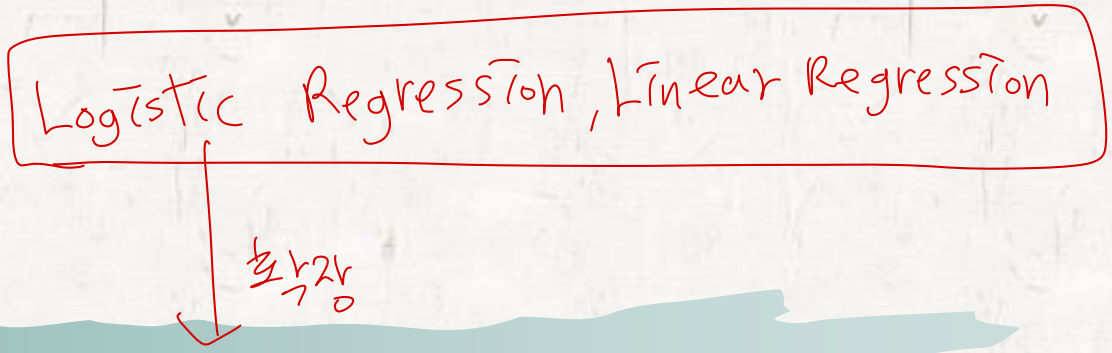
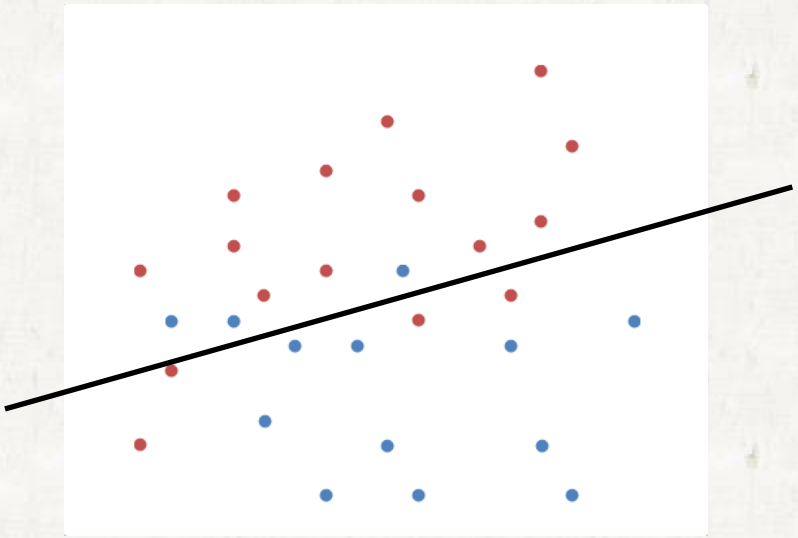


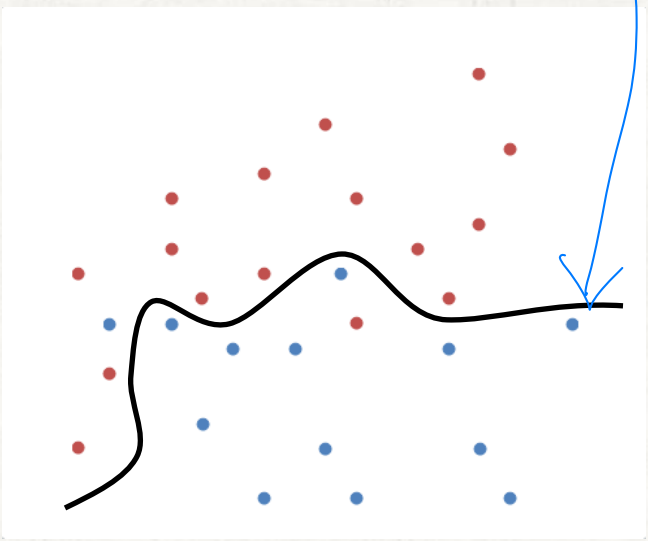
Neural Networks

Logistic Regression

- Logistic Regression
  - Linear classifier
- How can I generate non-linear boundary?

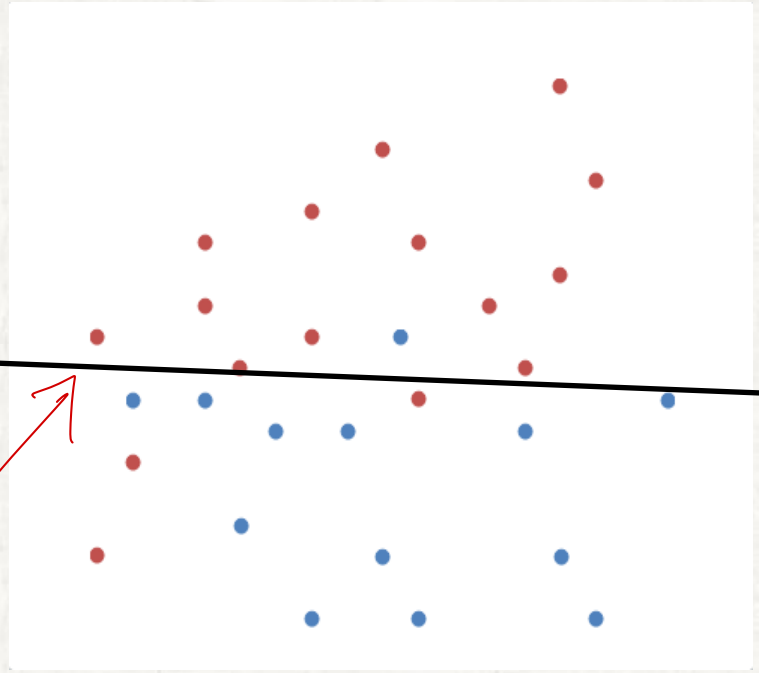
Find multiple LR's





Logistic Regression

- Linear Boundary



$$f(x_1, x_2) = w_1 x_1 + w_2 x_2 + w_0 \quad h(x_1, x_2) = \frac{1}{1 + \exp(-f(x_1, x_2))}$$

x_0 (= 1)

Logistic Regression

⇒ for given (x_1, x_2)

① evaluate $f(x_1, x_2)$

$s = w_1 x_1 + x_2 w_2 + w_0$

② evaluate h

$h = \frac{1}{1+e^{-s}}$

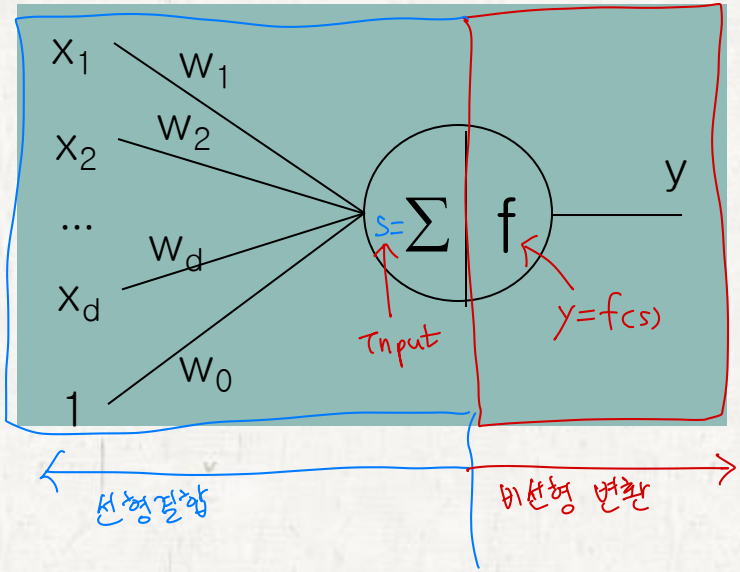
③ Decision

Red if $h \geq 0.5$

Blue if $h < 0.5$

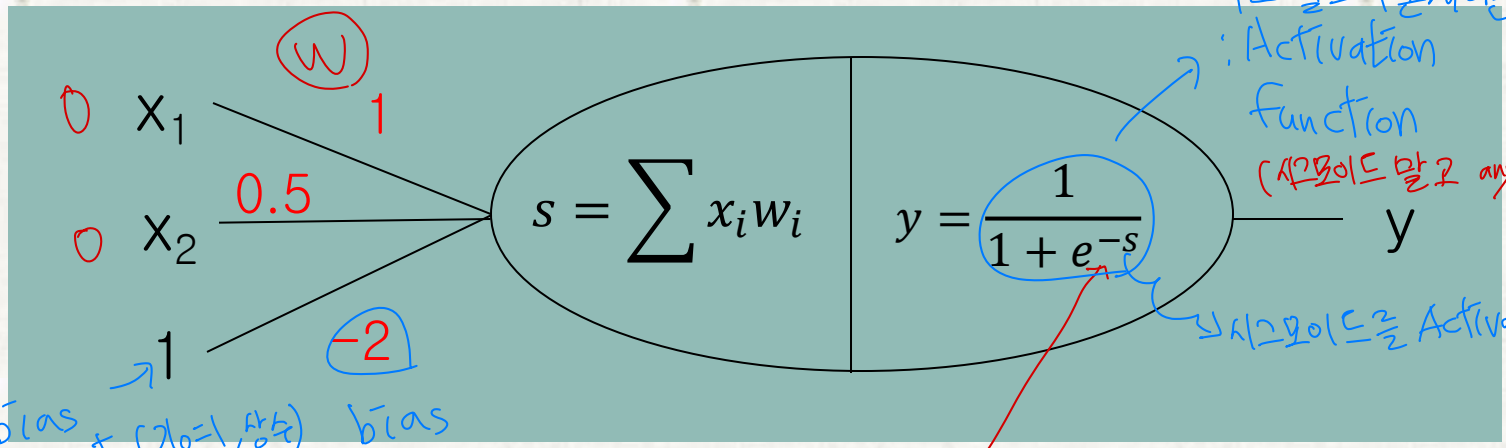
Logistic Regression

- Two Steps for Evaluation
  - Linear combination of inputs: $s = \mathbf{wx} = \sum_{i=0}^{d} w_i x_i$ = $w_0 + w_1 x_1 + w_2 x_2$
  - Nonlinear transform of s: $y = \frac{1}{1 + \exp(-s)}$
- Graphical Representation



Logistic Regression

- Graphical Representation of Logistic Regression

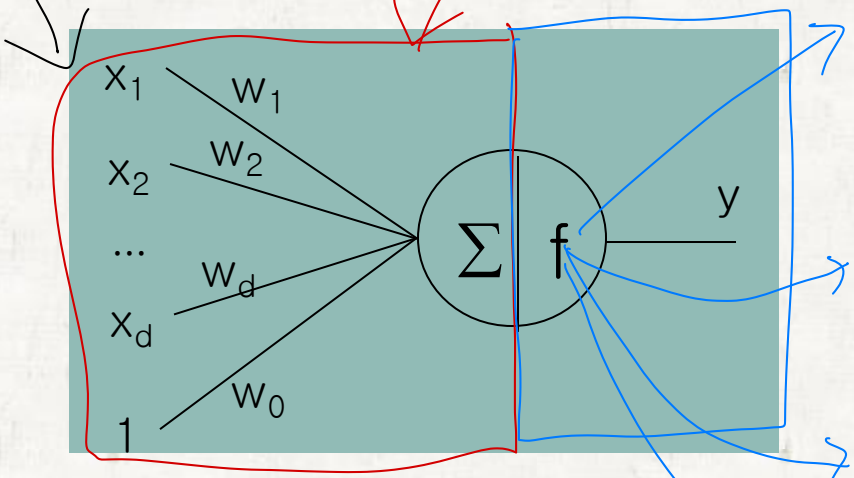


x_1	x_2	$\Sigma = s$	y
0	0	$0 \cdot 1 + 0 \cdot 0.5 + 1 \cdot (-2) = -2$	0.119
−0.5	3	$(-0.5) \cdot 1 + 3 \cdot 0.5 + 1 \cdot (-2) = -1$	0.269
0.5	1	$0.5 \cdot 1 + 1 \cdot 0.5 + 1 \cdot (-2) = -1$	0.269
2	1	$2 \cdot 1 + 1 \cdot 0.5 + 1 \cdot (-2) = 0.5$	0.622

Logistic Regression

- Another Name: Perceptron
  - Linear combination of inputs: $s = \mathbf{wx} = \sum_{i=0}^{d} w_i x_i$
  - Activation Function:

many options



Sigmoid $y = \dfrac{1}{1+\exp(-s)}$ ⇒ Logistic Reg = Linear Classifier

Hard limit $y = \begin{cases} 1 & \text{if } s \geq 0 \\ 0 & \text{otherwise} \end{cases}$

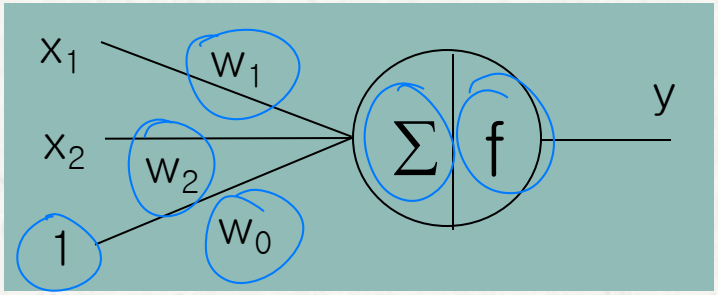
ReLU $y = \begin{cases} s & \text{if } s \geq 0 \\ 0 & \text{otherwise} \end{cases}$

optimization 안 하면 의미 없다

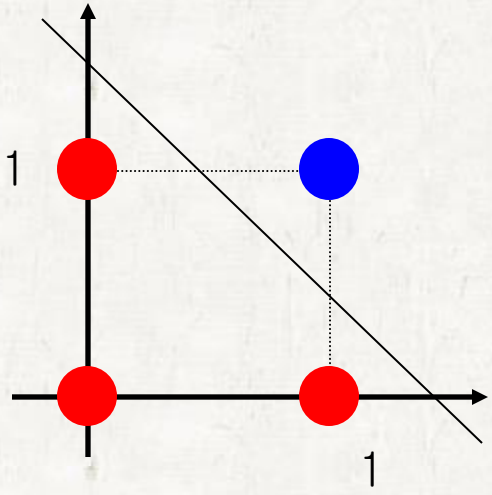
Linear $y = s$ ⇒ Linear Reg

Perceptron

- What a perceptron can do
  - AND operation



w1=1.0, w2=1.0, w0=−1.5

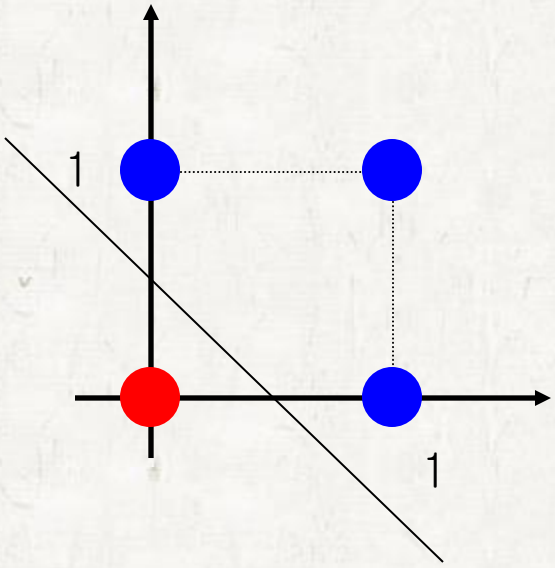


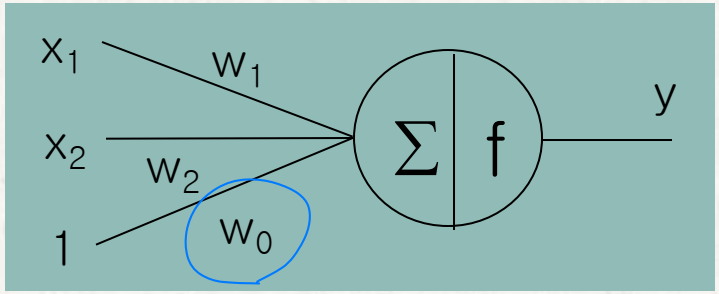
x_1	x_2	$s=\Sigma$	y		
			Sig.	H.L.	ReLU
0	0	−1.5	0.18	0	0
0	1	−0.5	0.38	0	0
1	0	−0.5	0.38	0	0
1	1	0.5	0.62	1	0.5

AND

Perceptron

- What a perceptron can do
  - OR operation



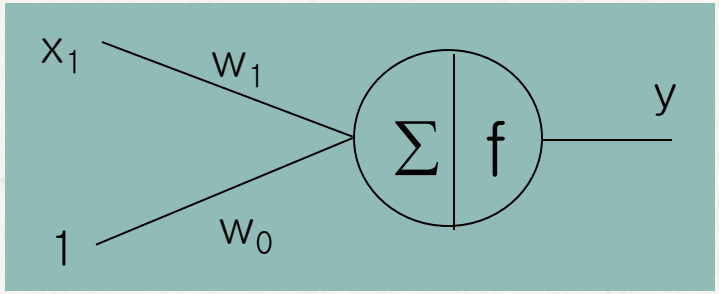


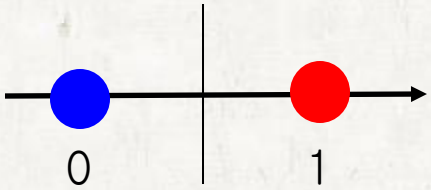
w1=1.0, w2=1.0, w0=−0.5

x_1	x_2	Σ	y		
			Sig.	H.L.	ReLU
0	0	−0.5	0.38	0	0
0	1	0.5	0.62	1	0.5
1	0	0.5	0.62	1	0.5
1	1	1.5	0.82	1	1.5

Perceptron

- What a perceptron can do
  - NOT operation





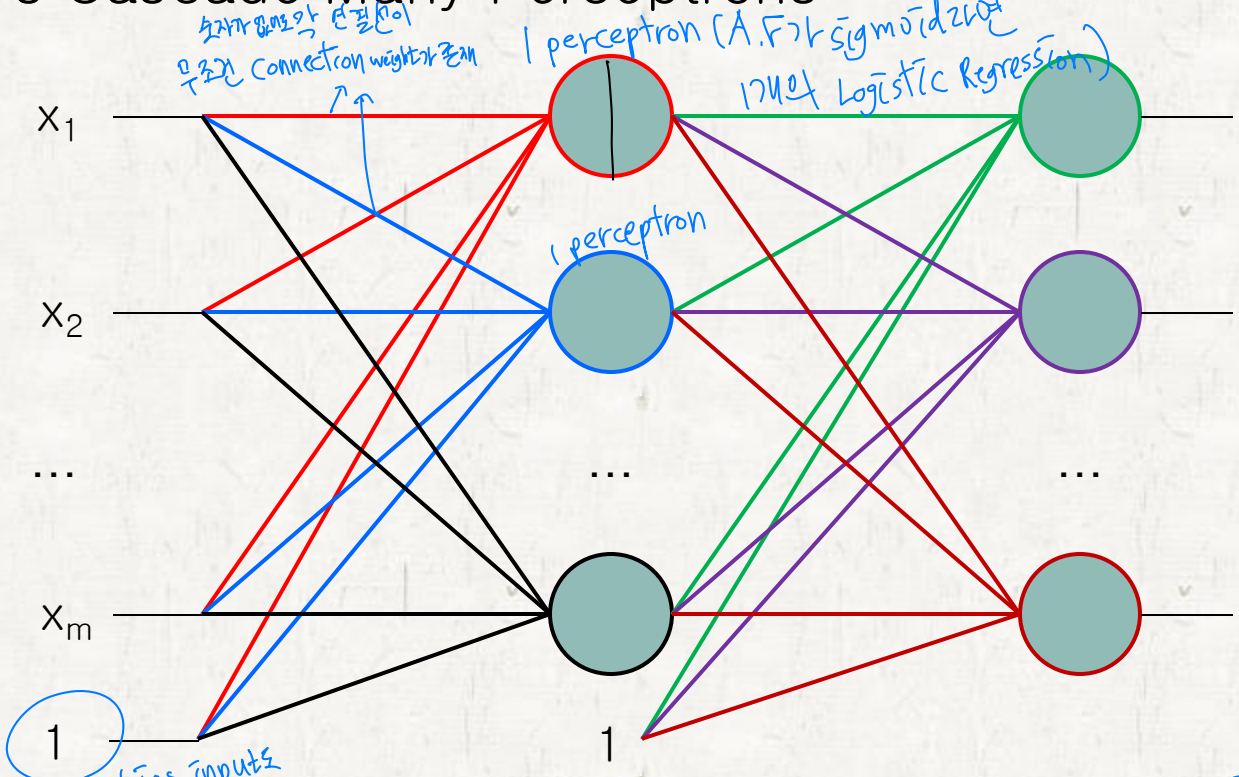
w1=−1.0, w0=0.5

x_1	Σ	y		
		Sig.	H.L.	ReLU
0	0.5	0.62	1	0.5
1	−0.5	0.38	0	0

Not

Neural Network

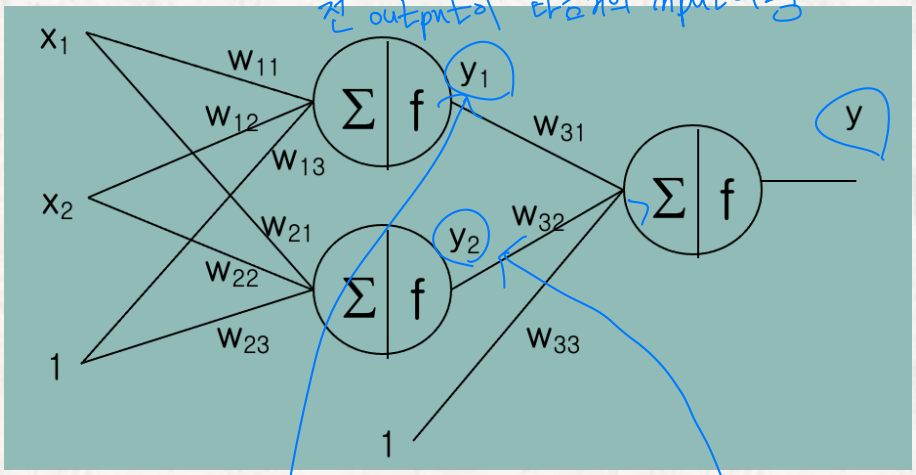
- Let's Cascade Many Perceptrons



Neural Network

- Example: XOR operation (Hard limit AF) = Activation function

$w_{11}=1.0,\ w_{12}=1.0,\ w_{13}=-1.5$ $w_{21}=1.0,\ w_{22}=1.0,\ w_{23}=-0.5$ $w_{31}=-1.0,\ w_{32}=1.0,\ w_{33}=-0.5$



전 output이 다음의 input이됨

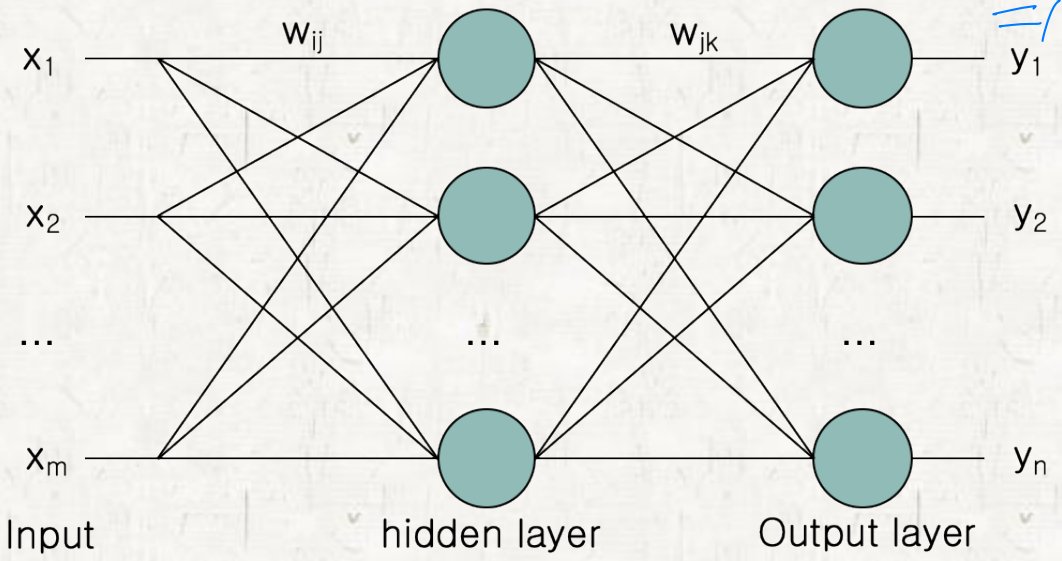
x_1	x_2	Σ	y_1
0	0	−1.5	0
0	1	−0.5	0
1	0	−0.5	0
1	1	0.5	1

x_1	x_2	Σ	y_2
0	0	−0.5	0
0	1	0.5	1
1	0	0.5	1
1	1	1.5	1

y_1	y_2	Σ	y
0	0	−0.5	0
0	1	0.5	1
0	1	0.5	1
1	1	−0.5	0

Neural Network

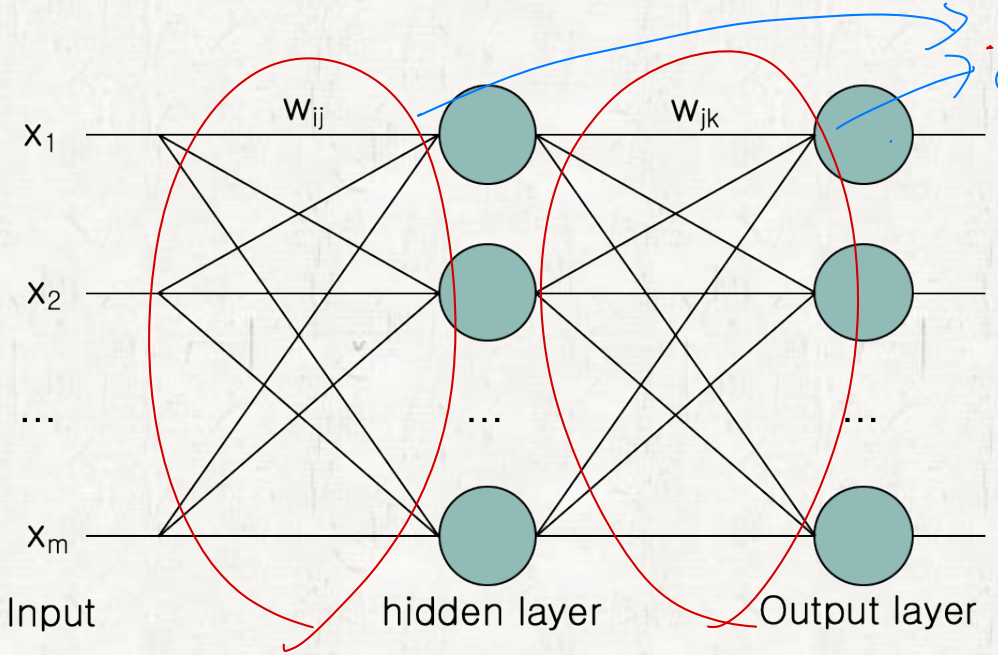
- What Can It Do?



- Everything a digital computer can do!!
- What? Prove it

Neural Network

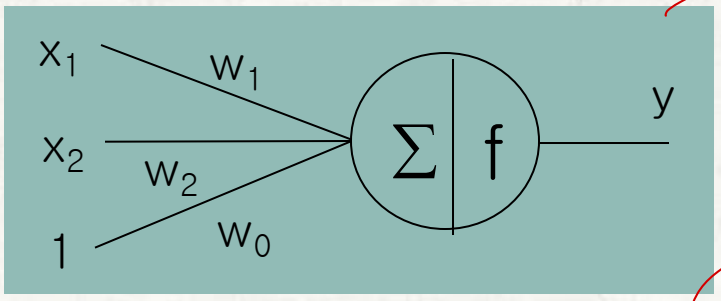
- Digital Computer? = NN
  - Do I have to make a program? yes (∵ 디지털 컴퓨터니까)
  - Does it have programs? Where?
  - How do I make a program?



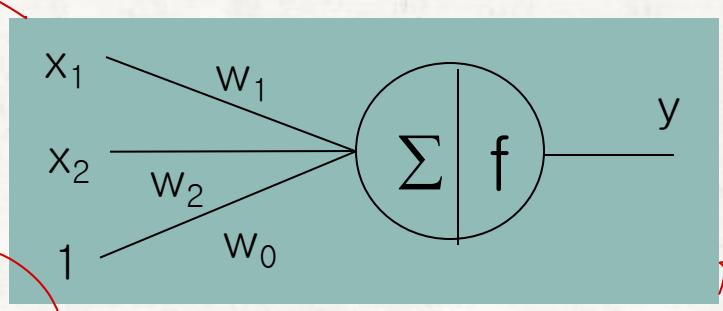
connection weight를 다르게 준 여러개의 Stock Index Predictor, dog/cat 분류기 GPT 등등이 있음 (겉은 N.N이지만 C.W만 달라도 기능이 완전 아예 달라진다)

Neural Network

- Digital Computer?



w1=1.0, w2=1.0, w0=−1.5



w1=1.0, w2=1.0, w0=−0.5

x_1	x_2	Σ	H.L.
0	0	−1.5	0
0	1	−0.5	0
1	0	−0.5	0
1	1	0.5	1

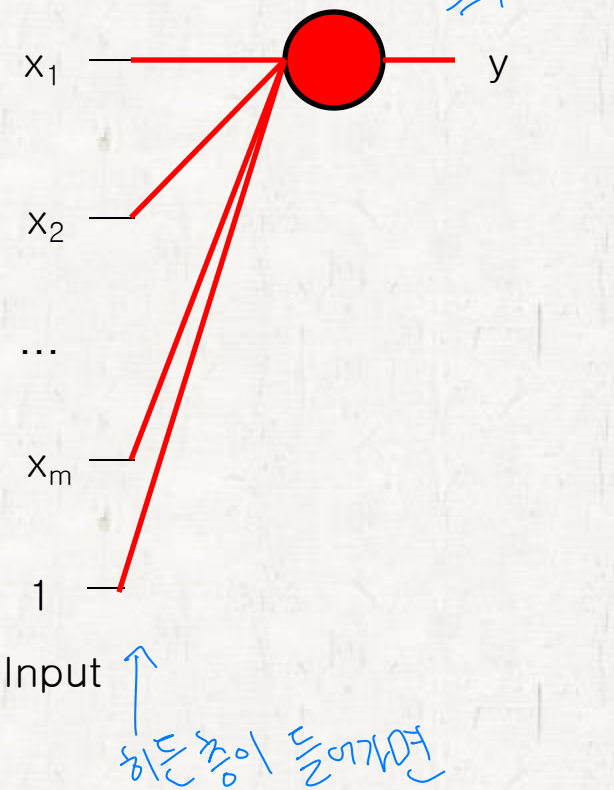
x_1	x_2	Σ	H.L.
0	0	−0.5	0
0	1	0.5	1
1	0	0.5	1
1	1	1.5	1

Neural Network

- Linear Model vs Neural Network

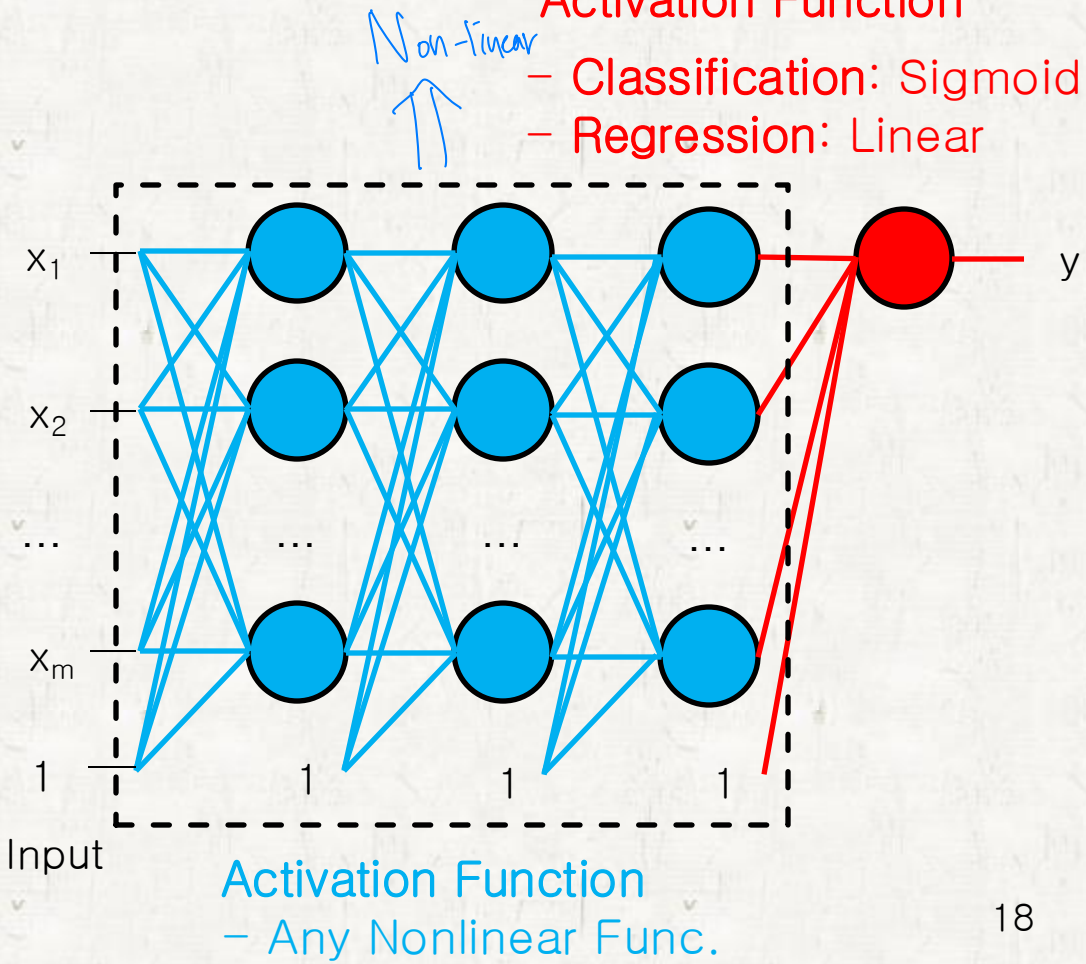
Activation Function

- **Classification**: Sigmoid
- **Regression**: Linear



Activation Function

- **Classification**: Sigmoid
- **Regression**: Linear

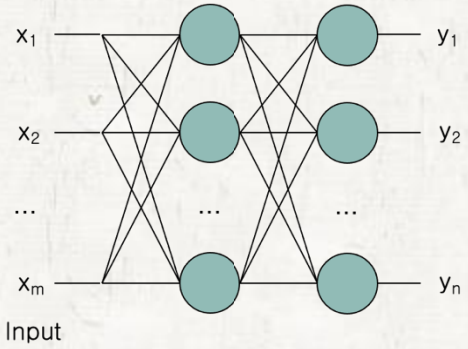


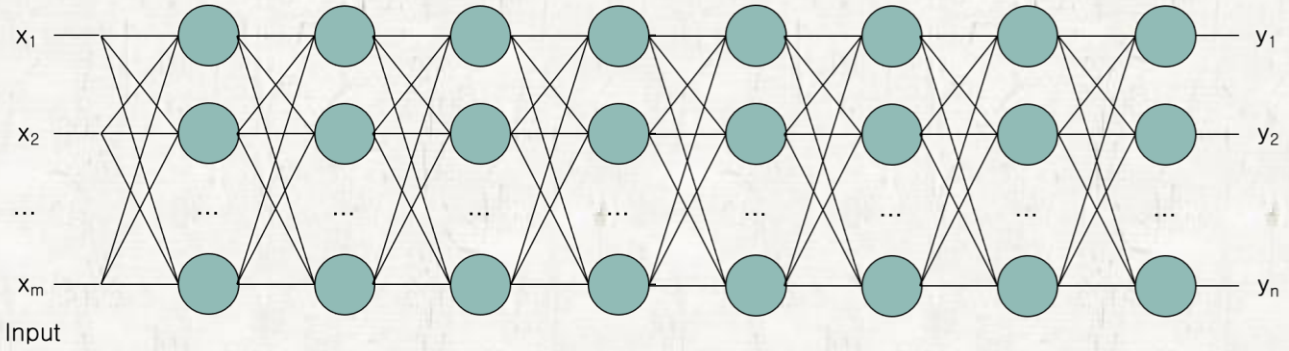
Activation Function

- Any Nonlinear Func.

Neural Network

- Why deep?





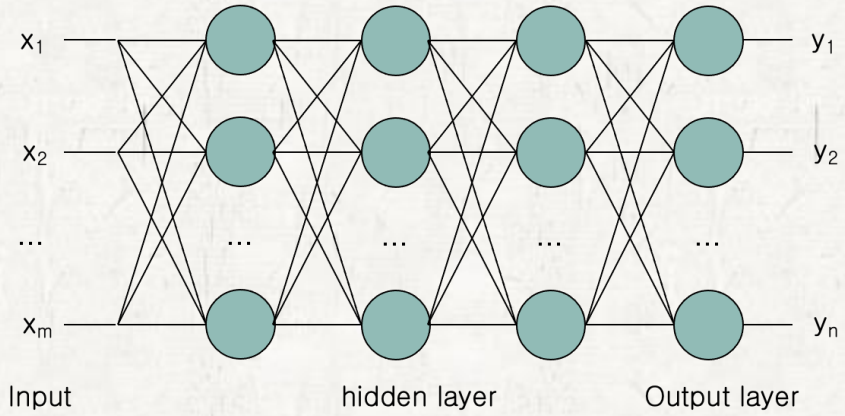
Neural Network Design

- Neural Network Structure
  - How many inputs?
  - How many outputs?
  - How many hidden layers?
  - How many nodes in hidden layers?

by your problem

by you

you have to decide structure of NN



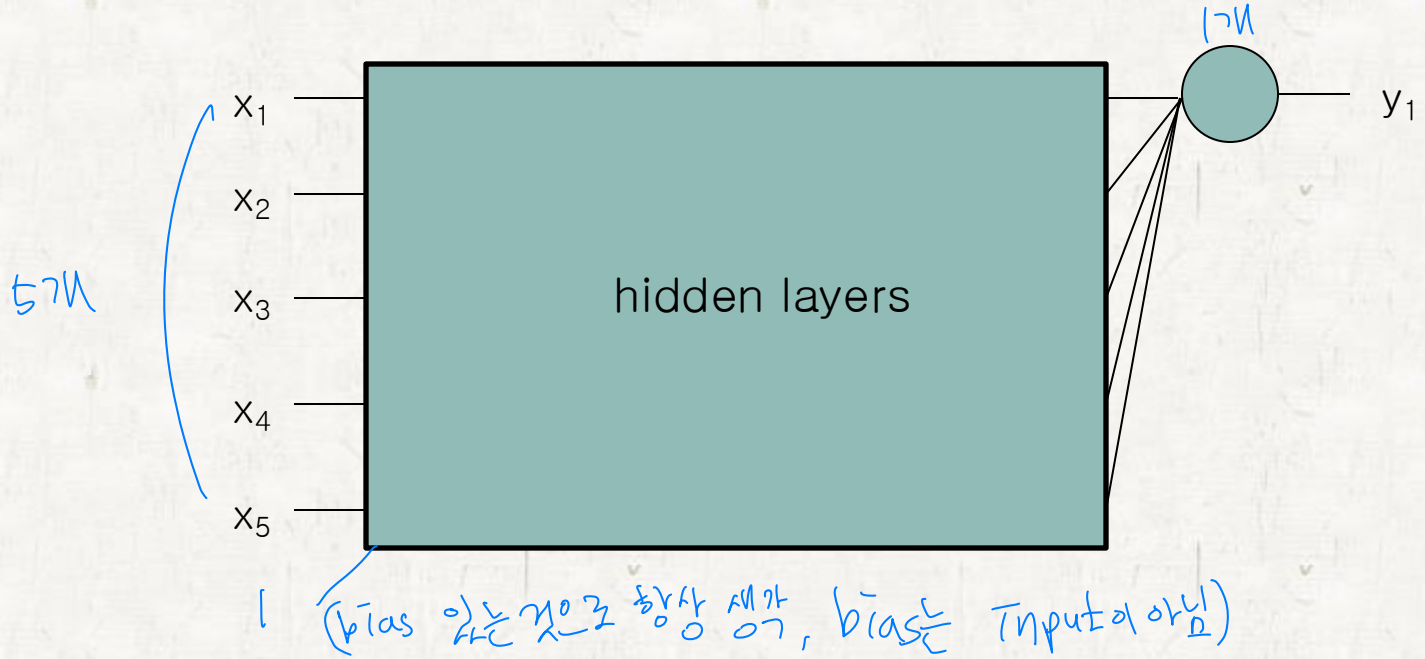
Neural Network Design

- How many inputs? How many outputs?
  - Stock Index Prediction

(2500,2550, 2530, 2540, 2550) -> 2600

(2400,2410, 2420, 2430, 2440) -> 2450

(2470,2460, 2450, 2470, 2480) -> 2470



Neural Network Design

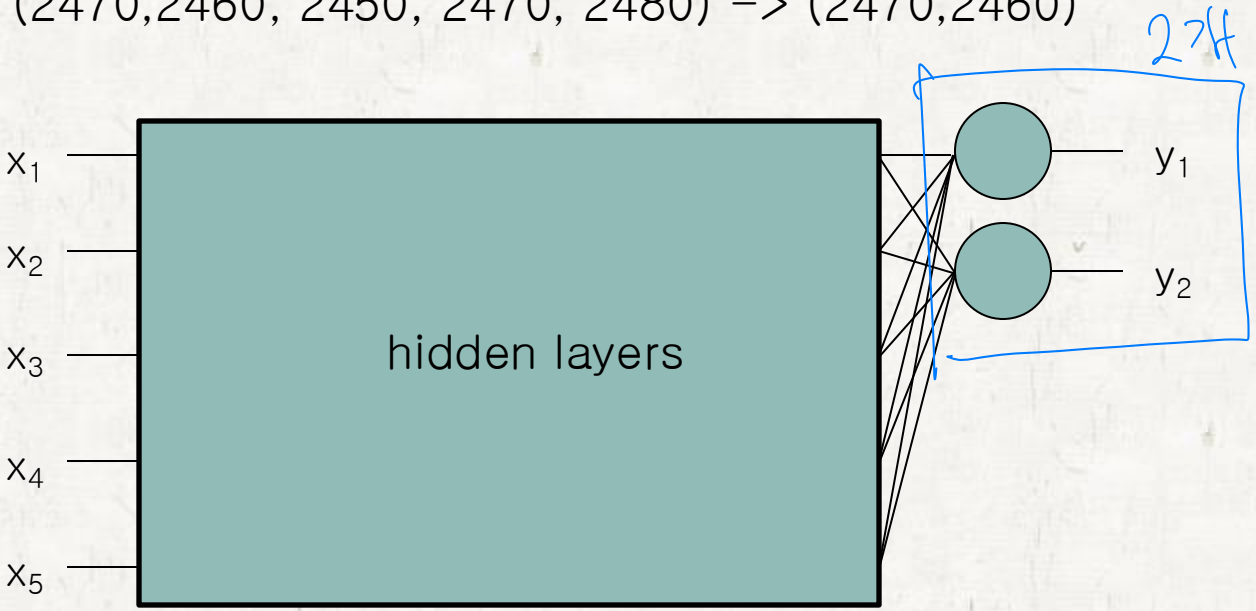
- How many inputs? How many outputs?
  - Stock Index Prediction

2 Next days

(2500,2550, 2530, 2540, 2550) -> (2600,2580)

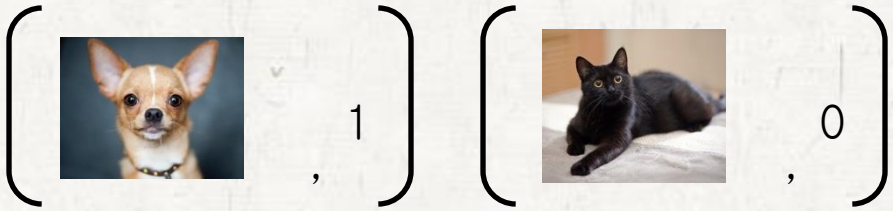
(2400,2410, 2420, 2430, 2440) -> (2450,2460)

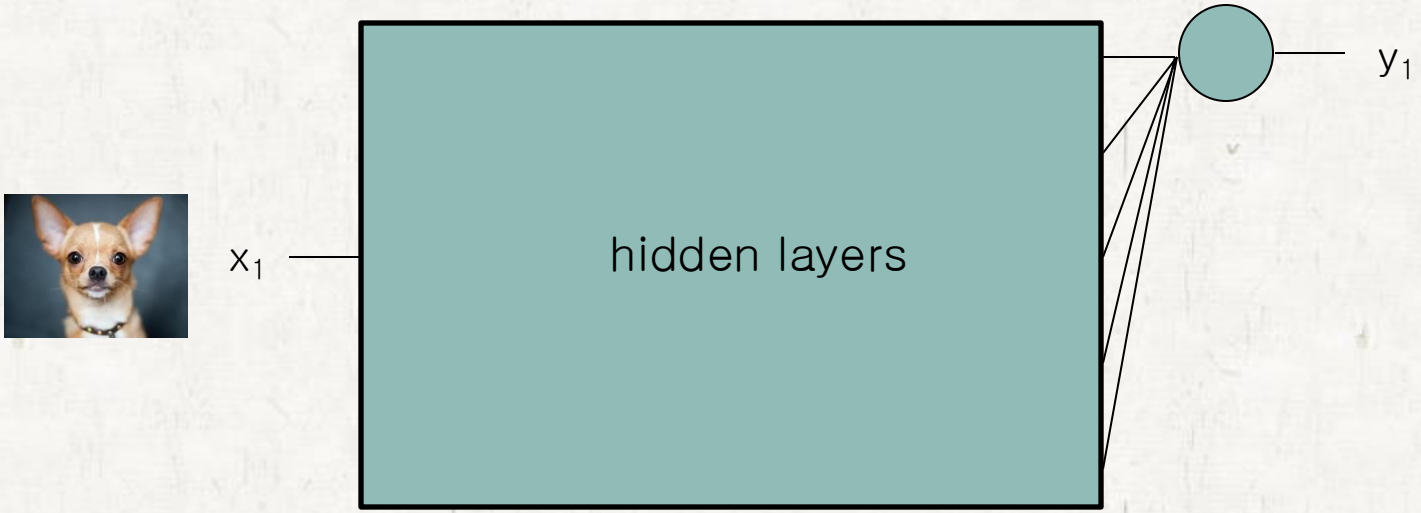
(2470,2460, 2450, 2470, 2480) -> (2470,2460)



Neural Network Design

- How many inputs? How many outputs?
  - Image Classification

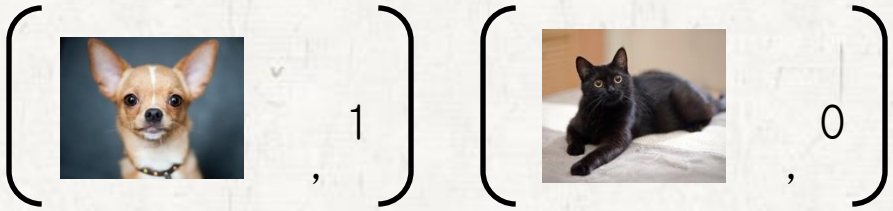


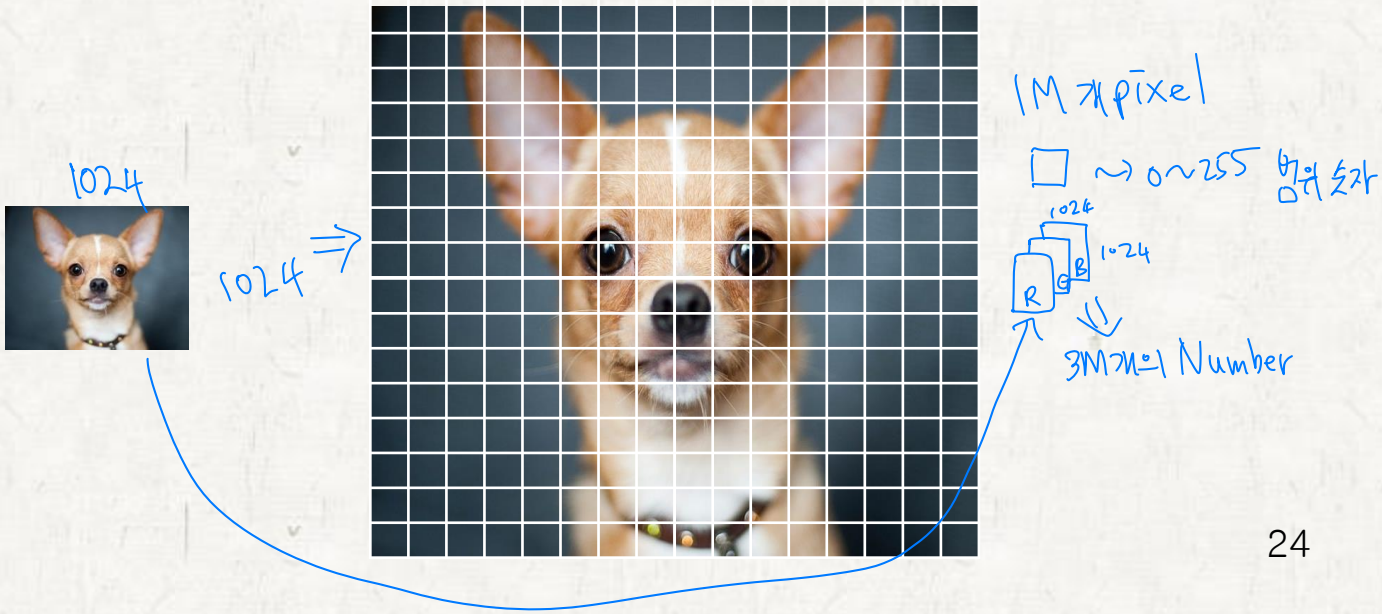


Neural Network Design

- How many inputs? How many outputs?
  - Image Classification

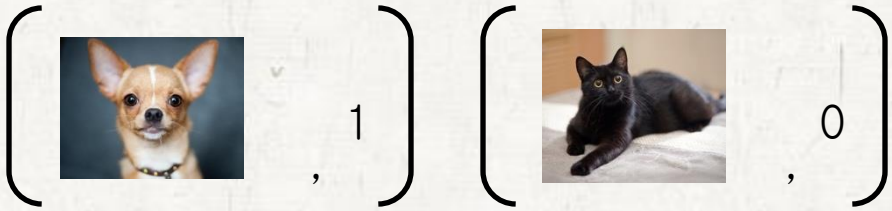
픽셀 단위로 나뉘서 입력 값으로 인식

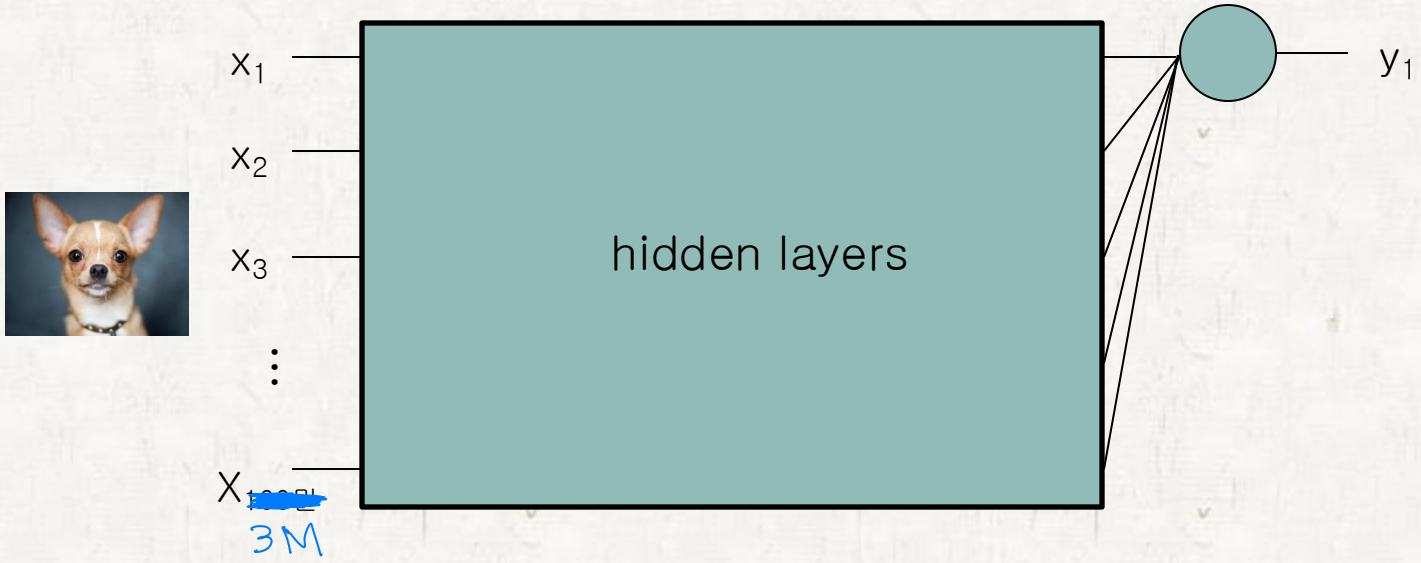




Neural Network Design

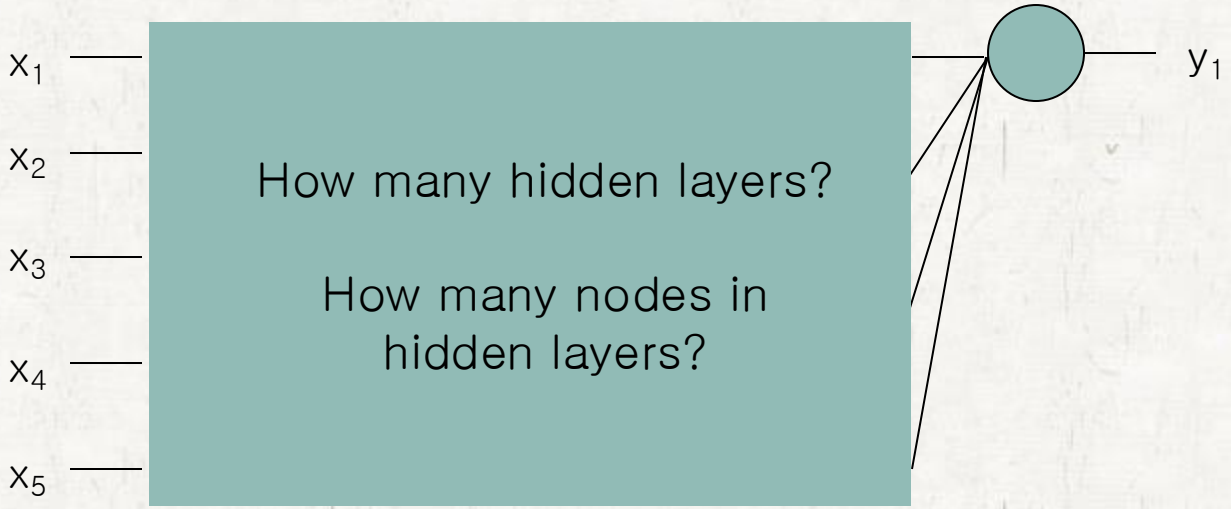
- How many inputs? How many outputs?
  - Image Classification





Neural Network Design

- How many hidden layers?
  How many nodes in hidden layers?
  - Well... it's up to you. But generally,
  - For simpler problems, use fewer layers and fewer nodes.
  - For more complex problems, use more layers and more nodes.



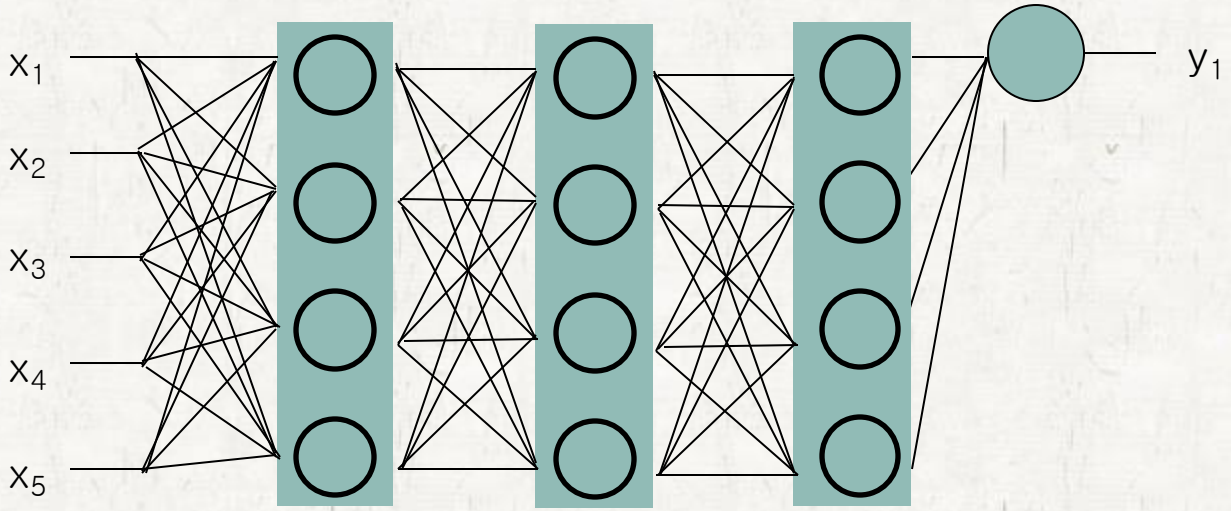
Neural Network Design

- How many hidden layers?
  How many nodes in hidden layers?
  - Stock Index Prediction

    (2500,2550, 2530, 2540, 2550) -> 2600

    (2400,2410, 2420, 2430, 2440) -> 2450

    (2470,2460, 2450, 2470, 2480) -> 2470



Neural Network Design

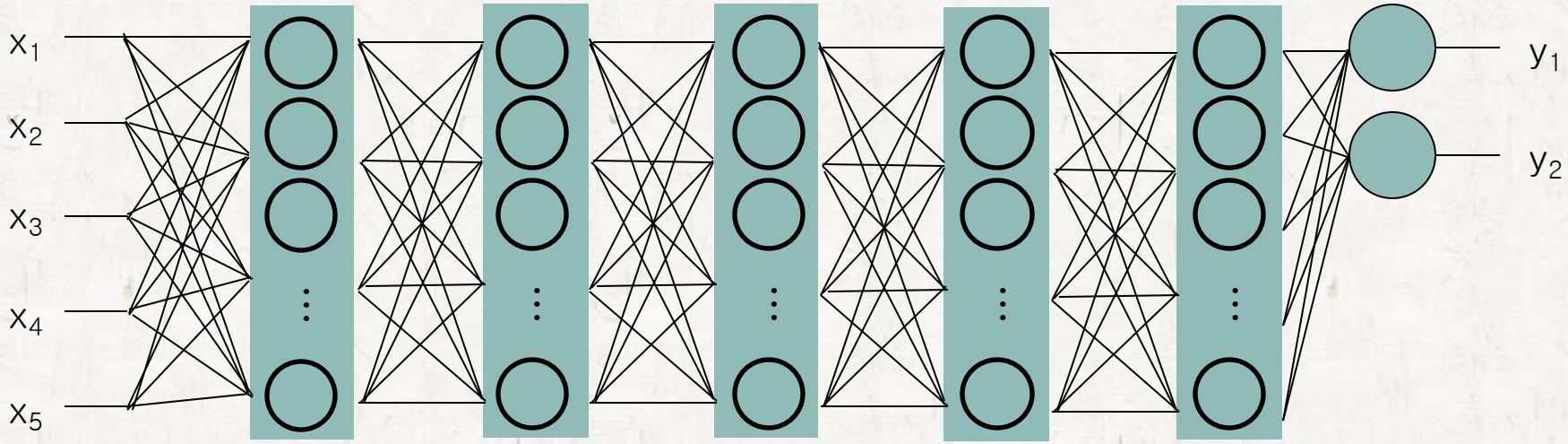
- How many hidden layers?
  How many nodes in hidden layers?
  - Stock Index Prediction

2 days 구하는게 어려우니 더 많은 층/노드수

(2500,2550, 2530, 2540, 2550) -> (2600,2580)

(2400,2410, 2420, 2430, 2440) -> (2450,2460)

(2470,2460, 2450, 2470, 2480) -> (2470,2460)



Neural Network Design

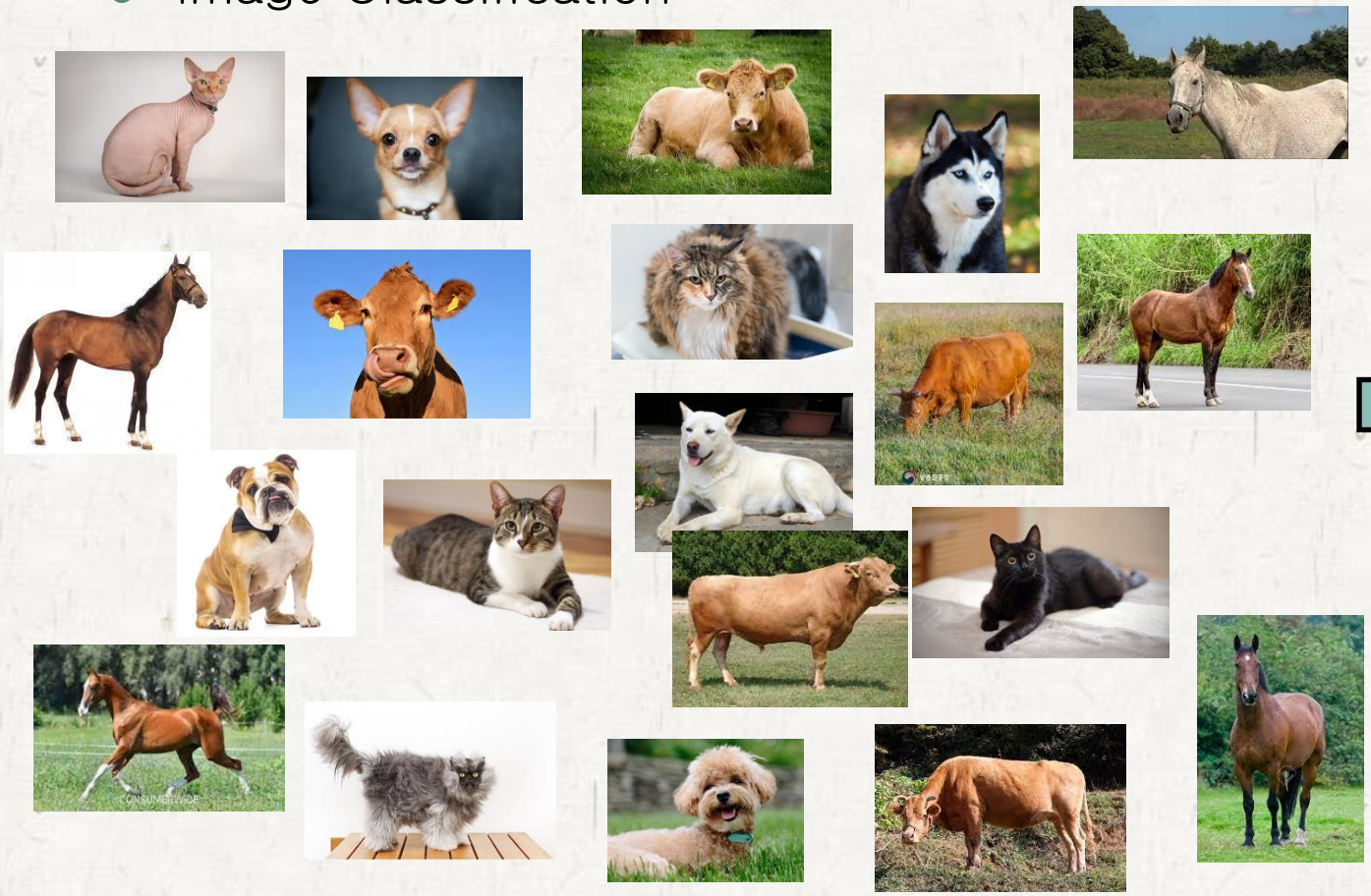
- How many hidden layers?
  How many nodes in hidden layers?
  - Image Classification

Dog

Cat

Neural Network Design

- How many hidden layers?
  How many nodes in hidden layers?
  - Image Classification



2classification보다 더 복잡함

Dog

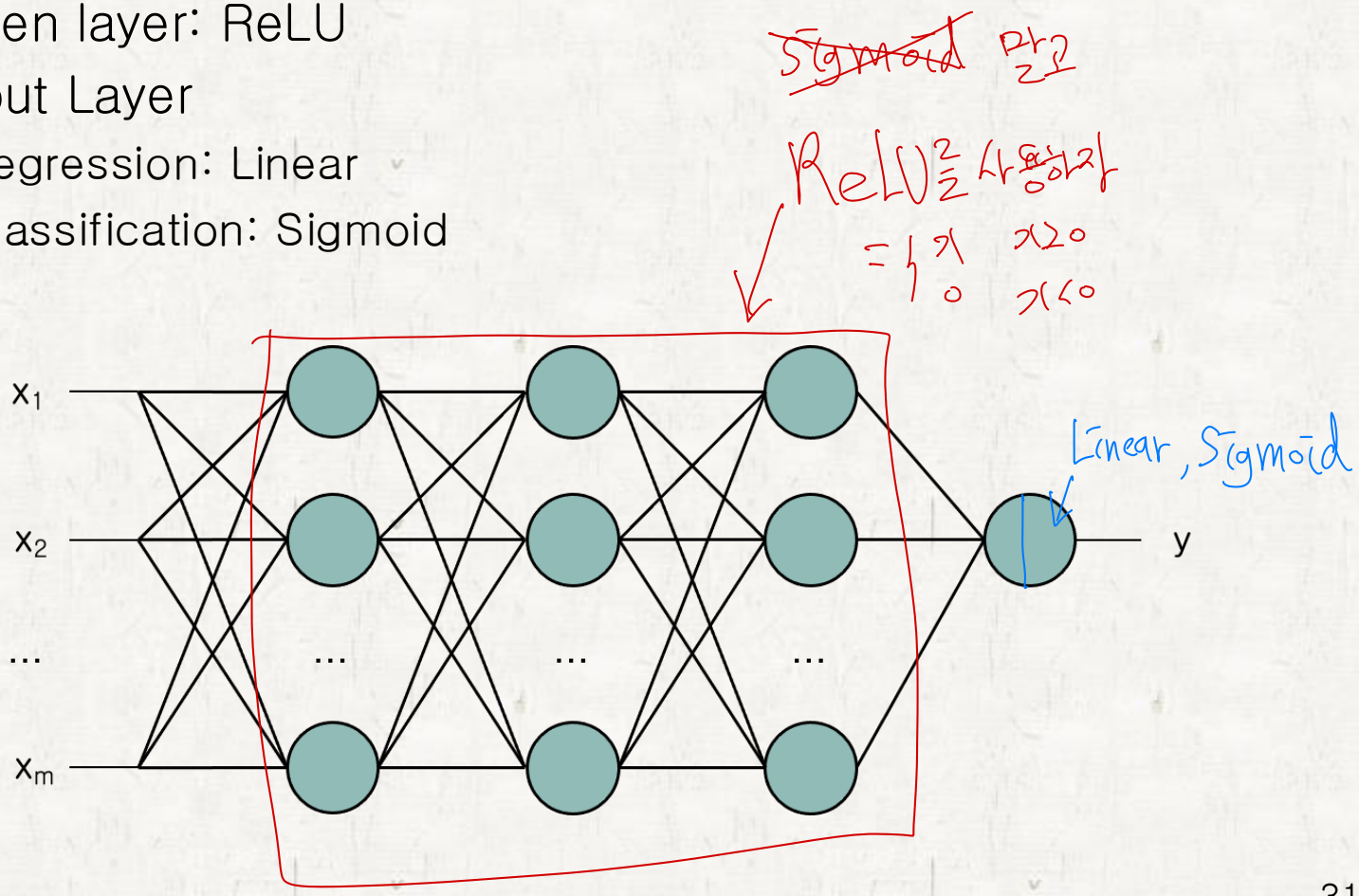
Cat

Horse

Cow

Neural Network Design

- Activation function
  - Hidden layer: ReLU
  - Output Layer
    - Regression: Linear
    - Classification: Sigmoid



Neural Network

- History of Neural Network