Deep Learning

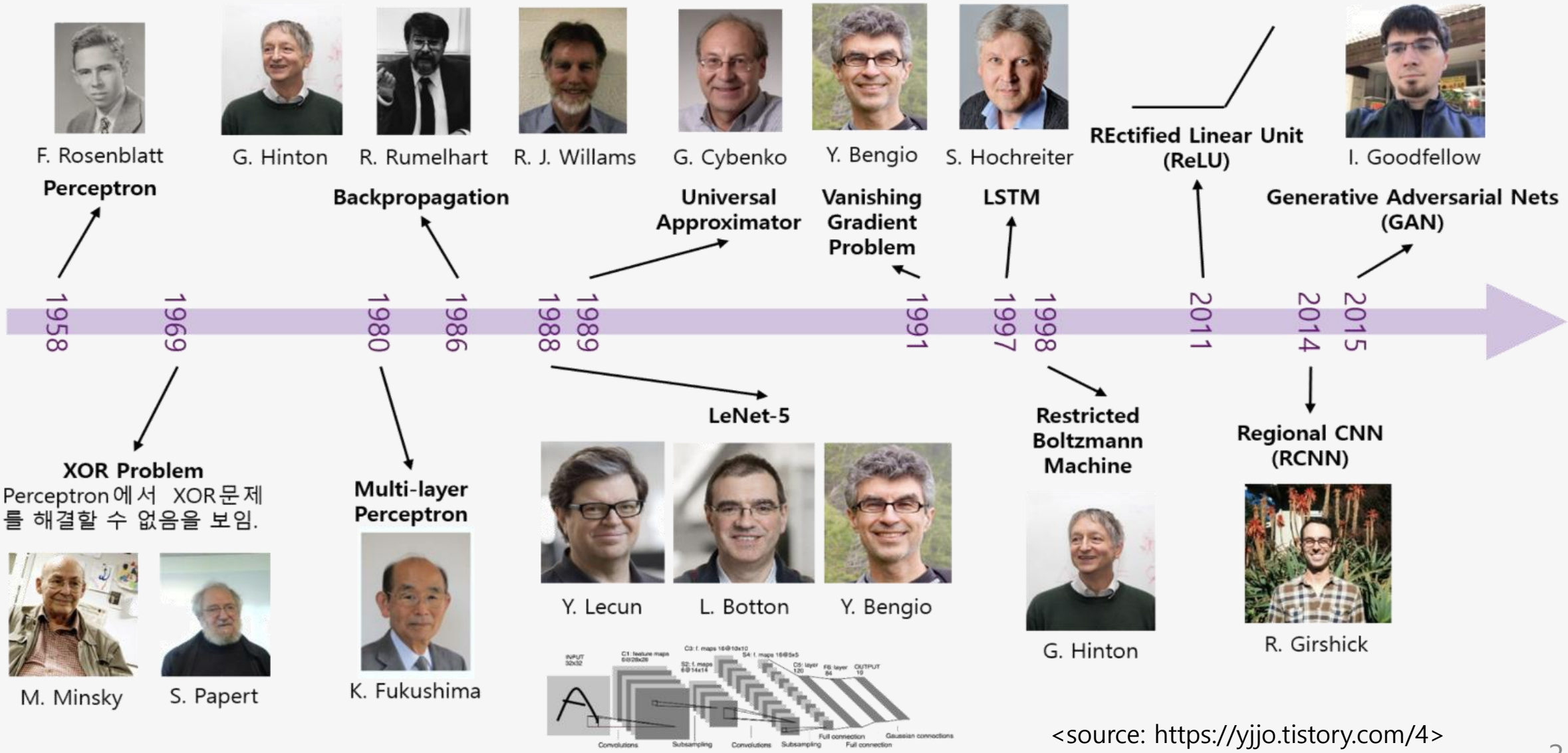
Summary

Outline

- Perceptron
  - Logic Gates Example
- Neural Network Structure
  - Activation Function, Node Inside, Softmax, One-hot Encoding
- Neural Network Training
  - Data Driven, Loss Function, Mini-batch, Gradient Decent
- Backpropagation

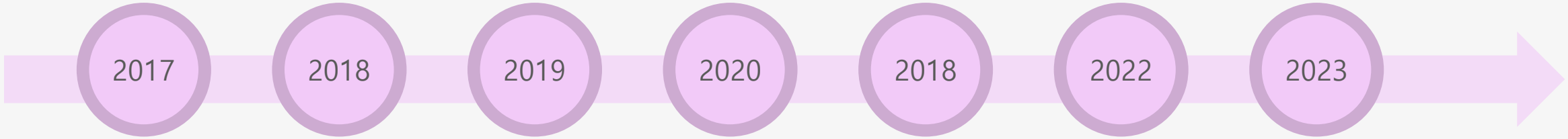
. . . TBA . . .

Brief History of Deep Learning



<source: https://yjjo.tistory.com/4>

Brief History of Deep Learning (or NLP?)



2017: Transformer
Google
"Attention is all you need"

2018: BERT
Devlin
•
GPT
OpenAI

2019: GPT-2
OpenAI

2020: GPT-3
OpenAI

2018: DALL-E
OpenAI

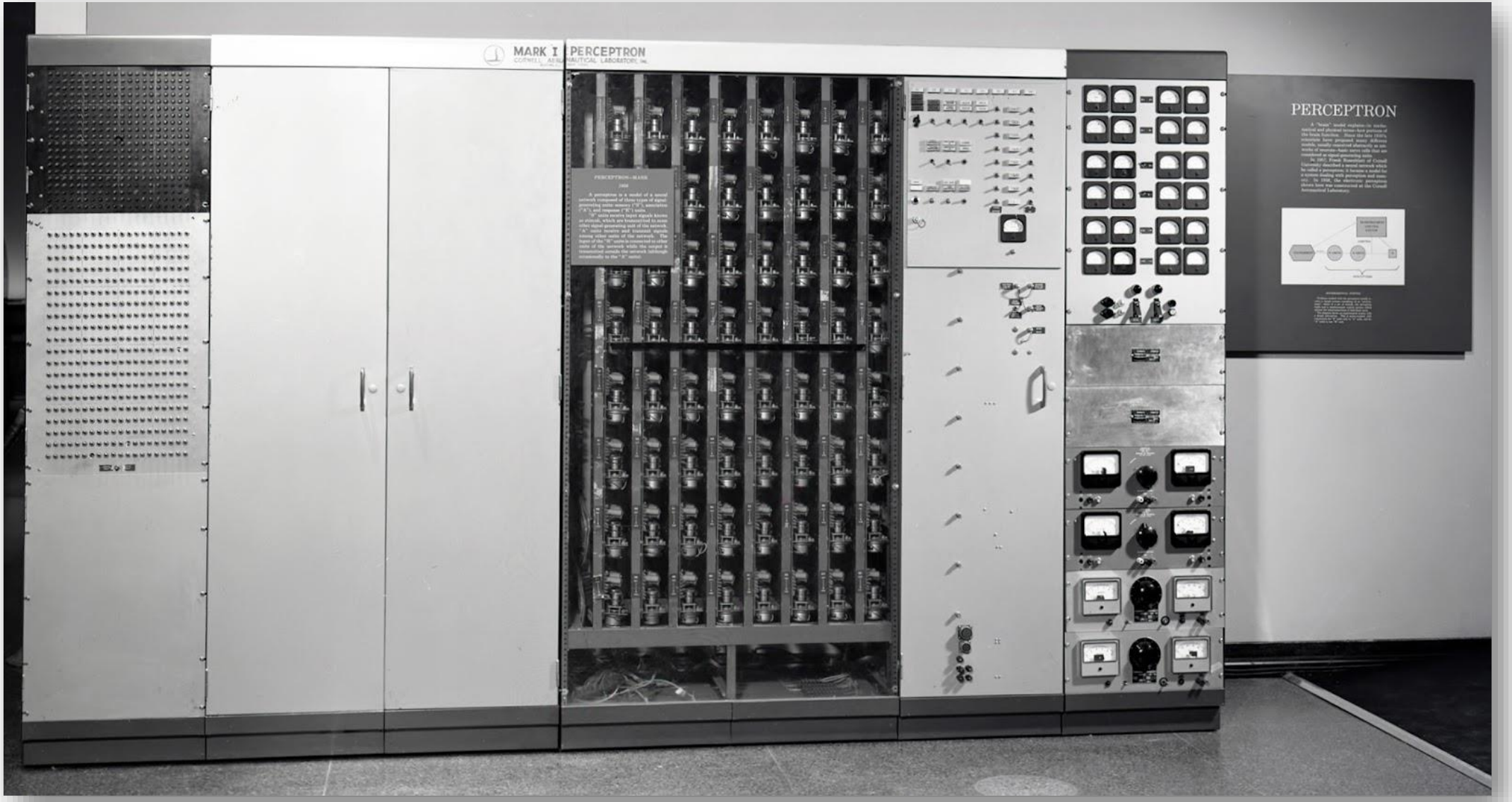
2022: ChatGPT
OpenAI
•
HuggingFace

2023: LLaMa
Meta

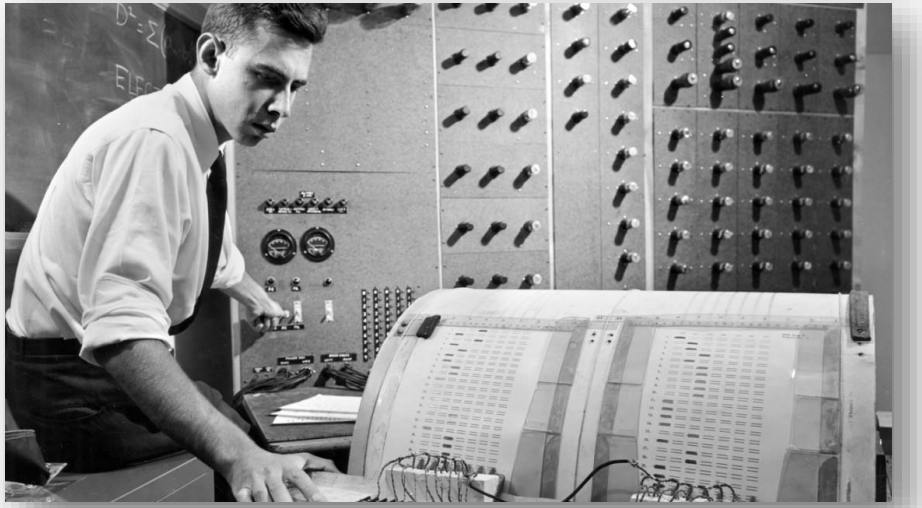
Perceptron

...an algorithm for supervised learning of binary classifiers...

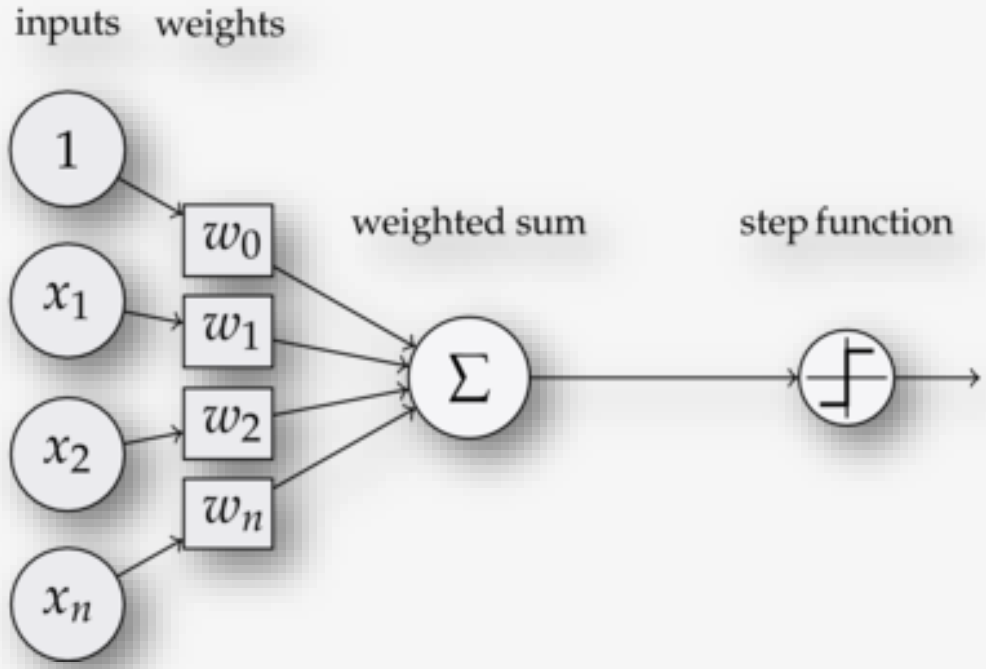
MARK 1 PERCEPTRON - 1958 at the Cornell Aeronautical Laboratory by ***Frank Rosenblatt***



MARK 1 PERCEPTRON - 1958 at the Cornell Aeronautical Laboratory by ***Frank Rosenblatt***

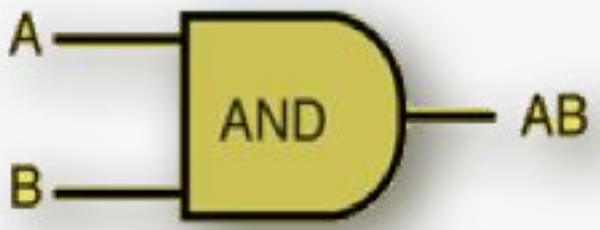


- 400 photocells (20x20 size)
- "neurons" connected randomly
- weights encoded in potentiometers가변저항
- learning performed by electric motors



<***Single*** Layer Perceptron>

AND Gate by Legacy Programming



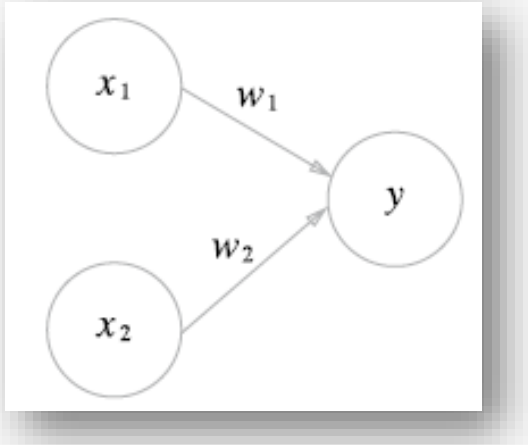
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

- Legacy Programming

```
# A, B, Out ∈ Boolean

function _AND_(A, B):
    if A=1 and B=1: Out=1
    else: Out=0
    return Out
```

AND Gate by Perceptron

- Perceptron



w1	w2	θ
?	?	?

- 원하는 답이 나오도록 w1, w2, θ를 잘 조정

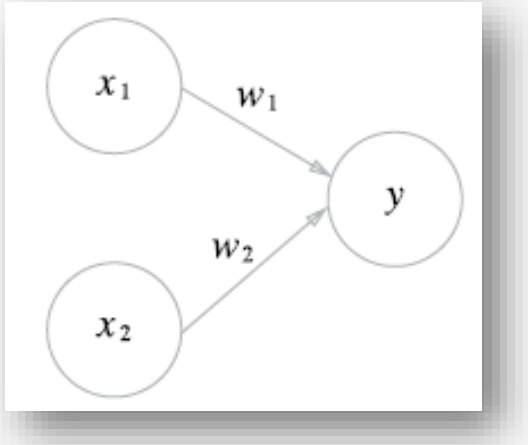
- 알고리즘

```
# x1, x2, y ∈ Boolean
# w1, w2, θ ∈ Integer

Σ = x1*w1 + x2*w2
y = f( Σ )

Function f( Σ ):
    if Σ > θ: return 1
    else: return 0
```

x1	x2	Σ	y
0	0	?	0
0	1	?	0
1	0	?	0
1	1	?	1

AND Gate by Perceptron

- Perceptron



w1	w2	θ
0.6	0.6	0.9

- 원하는 답이 나오도록 w1, w2, θ를 잘 조정 (무한히 많은 답 존재)

- 알고리즘

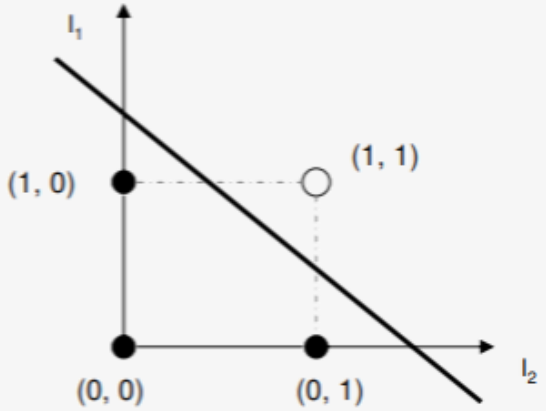
```
# x1, x2, y ∈ Boolean
# w1, w2, θ ∈ Integer

Σ = x1*w1 + x2*w2
y = f( Σ )

Function f( Σ ):
    if Σ > θ: return 1
    else: return 0
```

x1	x2	Σ	y
0	0	0	0
0	1	0.6	0
1	0	0.6	0
1	1	1.2	1

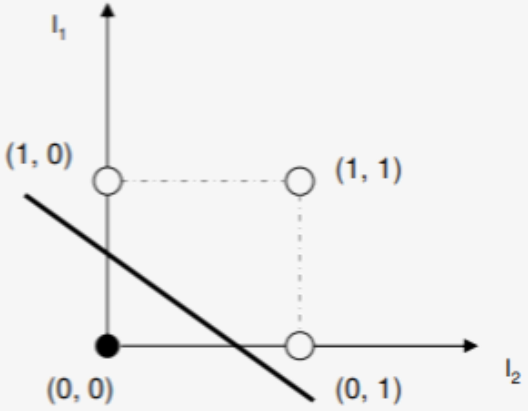
Limitation of Perceptron

- Good for AND, OR, NAND Gates, but XOR?

AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1

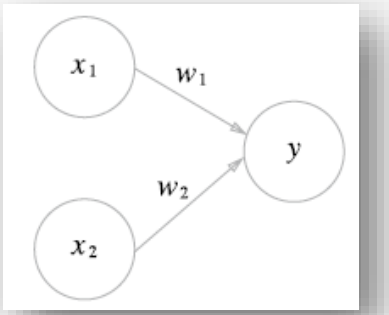


OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1

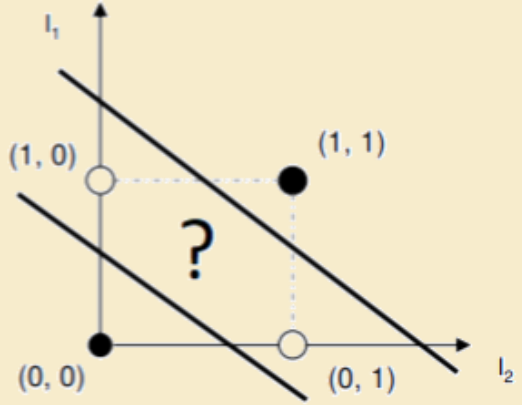


- Y는 x1, x2의 일차 함수

$$\Sigma = I_1 * w1 + I_2 * w2$$
$$y = f(\Sigma)$$



XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0



XOR Gate by Legacy Programming

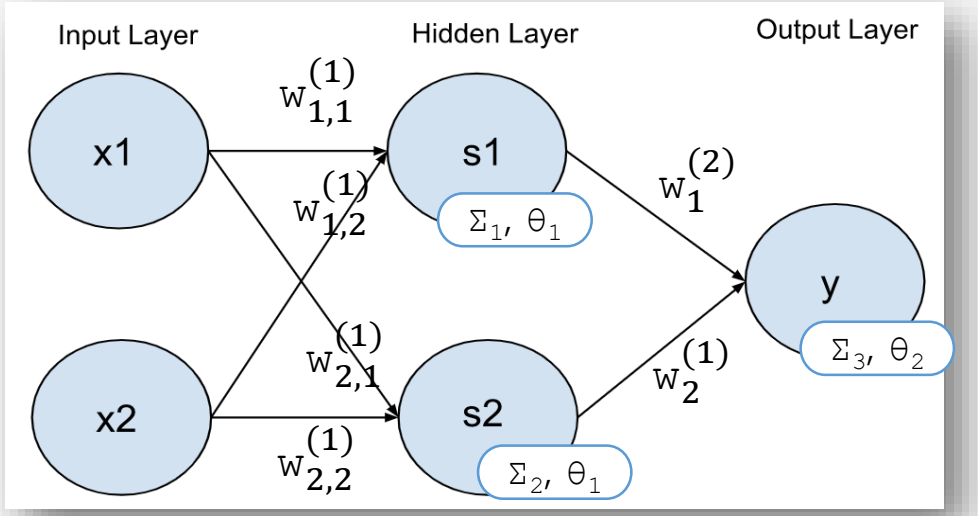
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

- Legacy Programming

```
# A, B, Out ∈ Boolean

function _XOR_(A, B):
    if A != B: Out=1
    else: Out=0
    return Out
```

XOR Gate by Perceptron

- Perceptron



w11	w12	w21	w22	θ1	w1	w2	θ2
?	?	?	?	?	?	?	?

- 원하는 답이 나오도록 w11, w12, w21, w22, θ1, w1, w2, θ2를 잘 조정

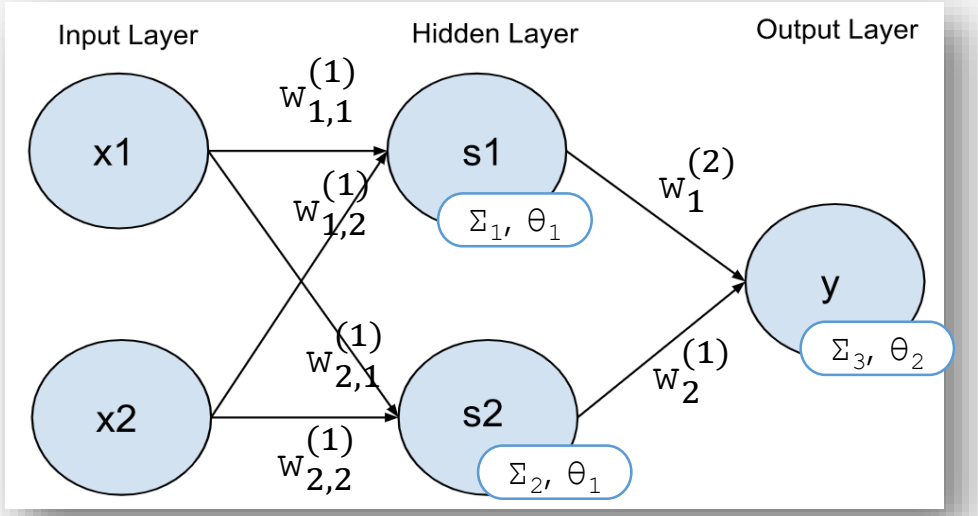
- 알고리즘

```
# s1, s2, y 노드 계산

Σ = x*w + x*w
z = f( Σ )

Function z( Σ ):
    if Σ > θ: return 1
    else: return 0
```

		Σ_1, θ_1		Σ_2, θ_1		Σ_3, θ_2	
x1	x2	Σ	s1	Σ	s2	Σ	y
0	0	?	?	?	?	?	0
0	1	?	?	?	?	?	1
1	0	?	?	?	?	?	1
1	1	?	?	?	?	?	0

XOR Gate by Perceptron

- Perceptron



w11	w12	w21	w22	θ1	w1	w2	θ2
0.6	-0.6	-0.6	0.6	0	1	1	0.5

- 원하는 답이 나오도록 w11, w12, w21, w22, θ1, w1, w2, θ2를 잘 조정 (무한히 많은 답 존재)

- 알고리즘

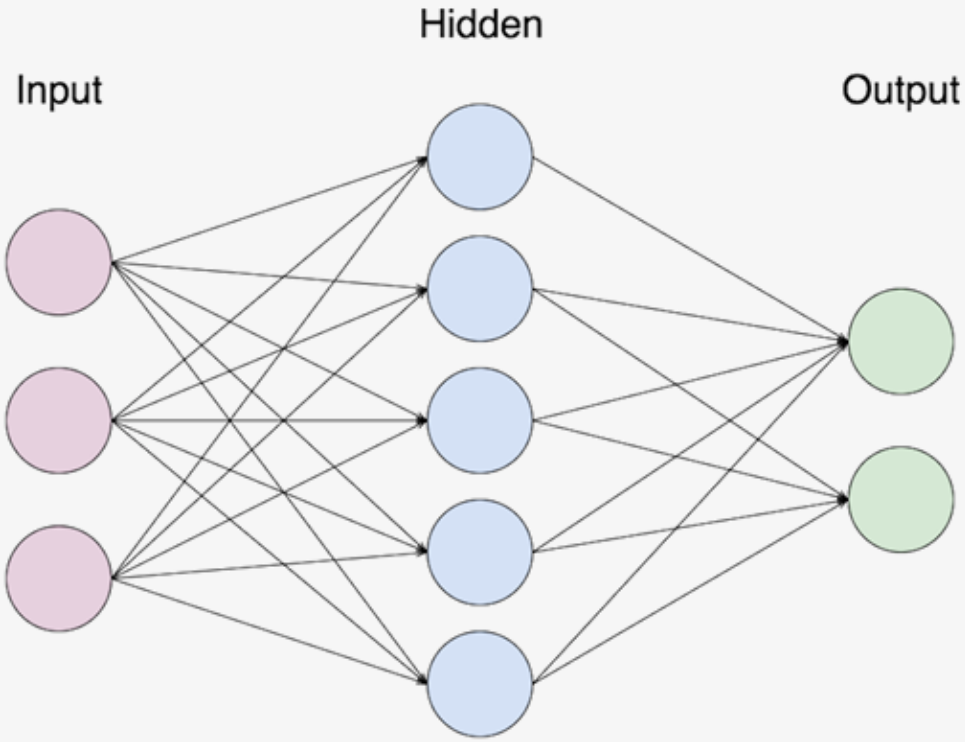
```
# s1, s2, y 노드 계산

Σ = x*w + x*w
z = f( Σ )

Function z( Σ ):
    if Σ > θ: return 1
    else: return 0
```

		Σ_1, θ_1		Σ_2, θ_1		Σ_3, θ_2	
x1	x2	Σ_1	s1	Σ_2	s2	Σ_3	y
0	0	0	0	0	0	0	0
0	1	-0.6	0	0.6	1	1	1
1	0	0.6	1	-0.6	0	1	1
1	1	0	0	0	0	0	0

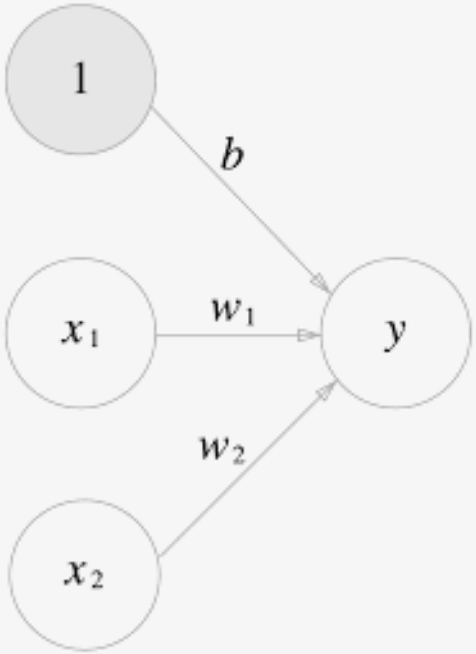
Neural Network Structure

...a network or circuit of artificial neurons...

Neural Network Components

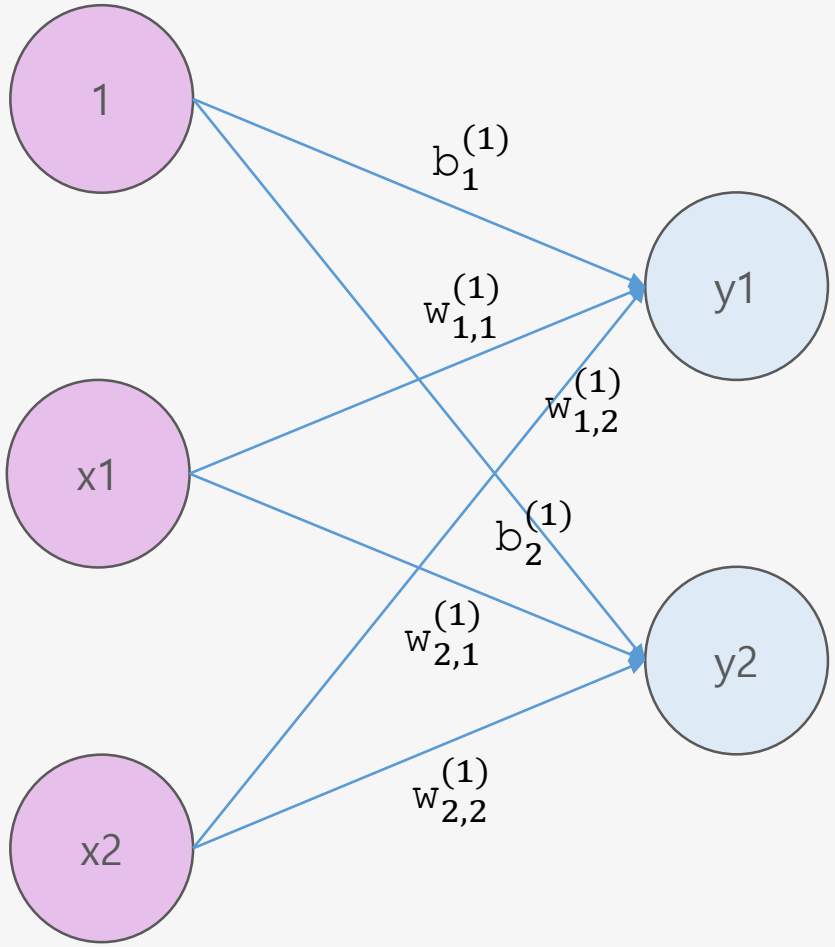


- 총 3층이지만 가중치를 갖는 층은 2개 뿐이기 때문에 '2층 신경망'이라고 부름



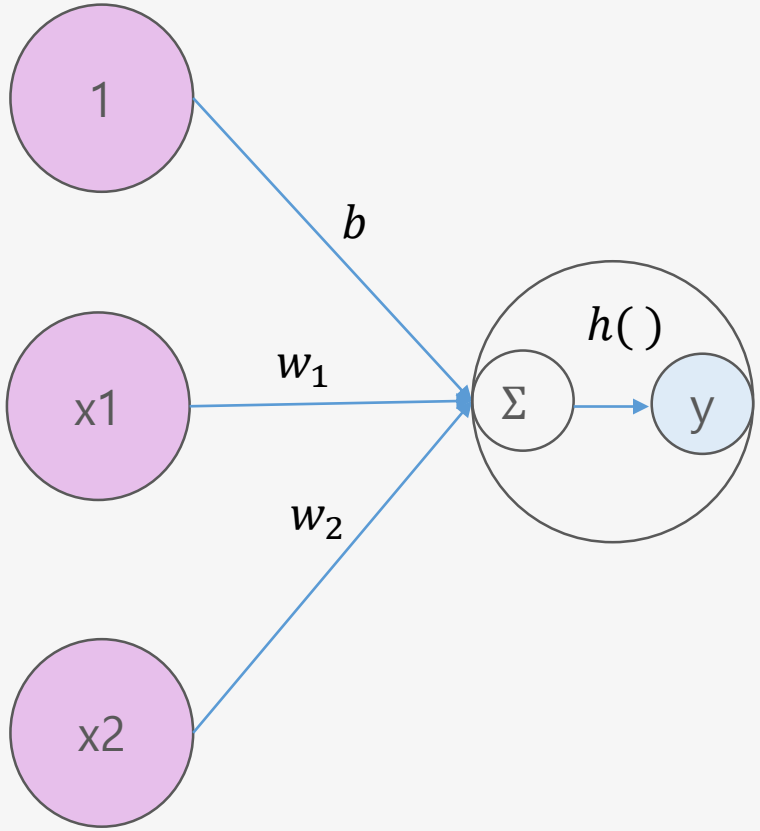
- 각 Layer 마다 Bias b가 default로 추가됨
- Bias는 뉴런이 얼마나 쉽게 활성화 되는가를 제어

Matrix Multiplication on Neural Network



$$\begin{pmatrix} y1 \\ y2 \end{pmatrix} = \begin{pmatrix} b_1^{(1)} & w_{1,1}^{(1)} & w_{1,2}^{(1)} \\ b_2^{(1)} & w_{2,1}^{(1)} & w_{2,2}^{(1)} \end{pmatrix} \begin{pmatrix} 1 \\ x1 \\ x2 \end{pmatrix}$$

Activation Function

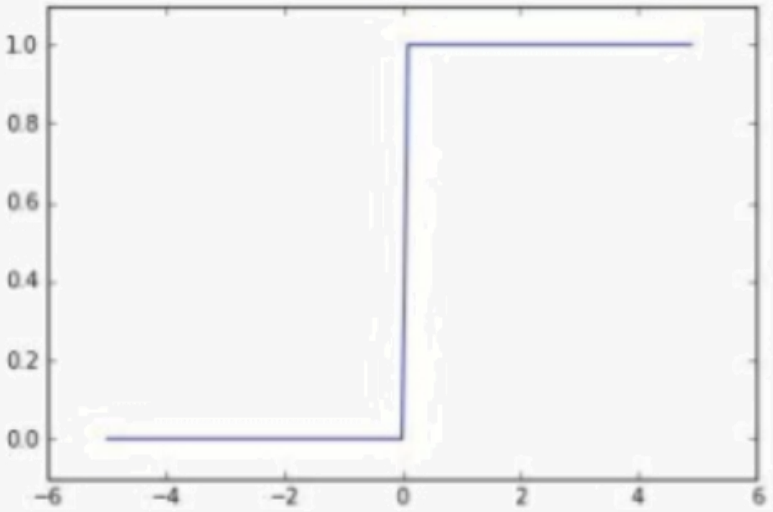


$$\Sigma = b + w_1 x_1 + w_2 x_2$$

$$y = h(\Sigma)$$

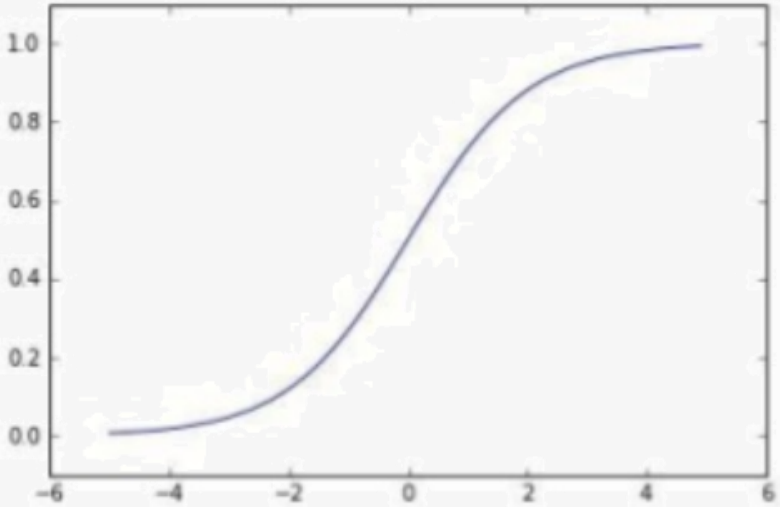
- $h( )$: Activation Function

Activation Function



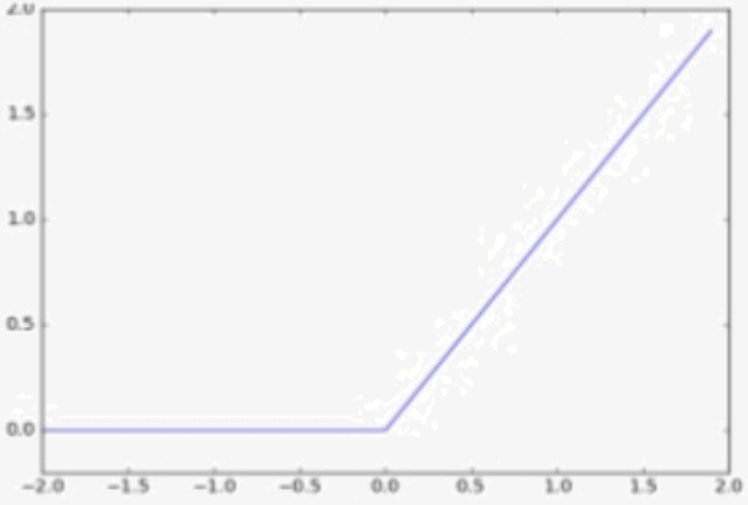
계단 함수

$$h(x) = \begin{cases} 0\ (x \le 0) \\ 1 (x > 0) \end{cases}$$



시그모이드 함수

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

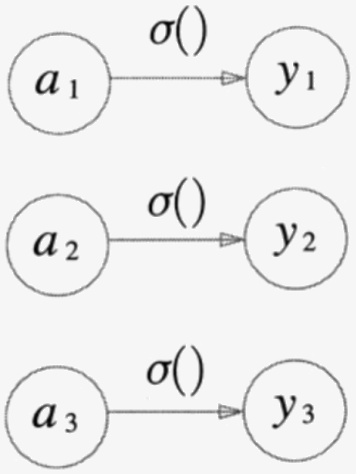


ReLU 함수

$$h(x) = \begin{cases} 0\ (x \le 0) \\ x (x > 0) \end{cases}$$

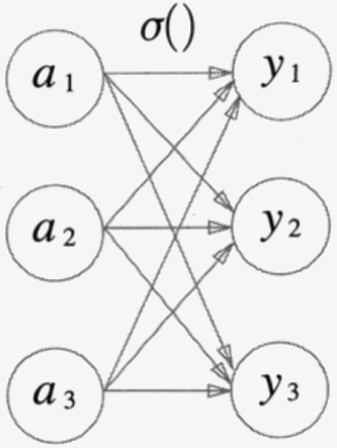
Output Layer

- Regression
  - Identity Function
  - 입력 데이터에 대해 수치적 결과 예측



- Output 계층의 노드별 결과값을 그대로 출력

- Classification
  - Softmax Function
  - 입력 데이터가 어느 부류에 속하는지 예측

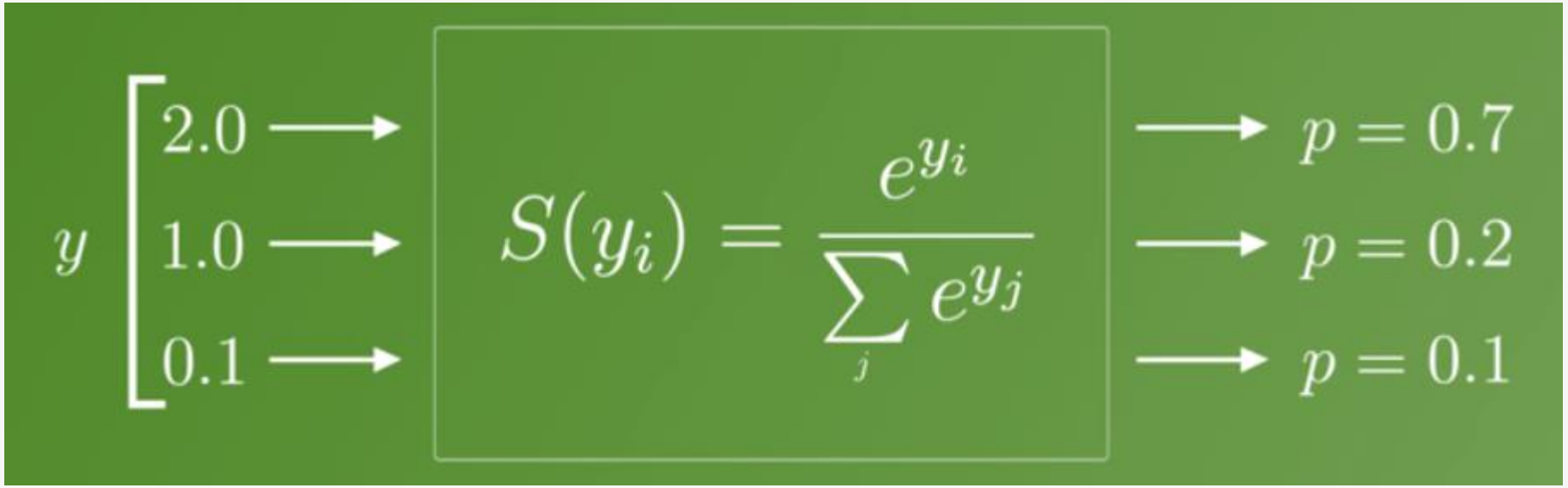


$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

- Output 계층의 전체 노드의 합에 대해 각 노드별 비율 값을 최종 출력

Output Layer - Softmax

- Output Score
- Softmax
- Probability



Output Layer – One-hot encoding

- 정답 데이터(Labeled Data) → One-hot Encoding
  -  → [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]

- Neural Net 계산

SoftMax 결과값	One-hot 정답지	<Meaning>
0.05	0	'0'
0.1	0	'1'
0.6	1	'2'
. . .	. . .	⋮
0.12	0	'9'