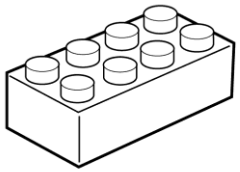
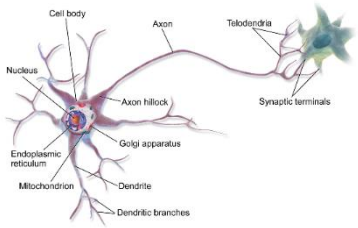
MACHINE LEARNING

Neural network을 중심으로

LINEAR PERCEPTRON

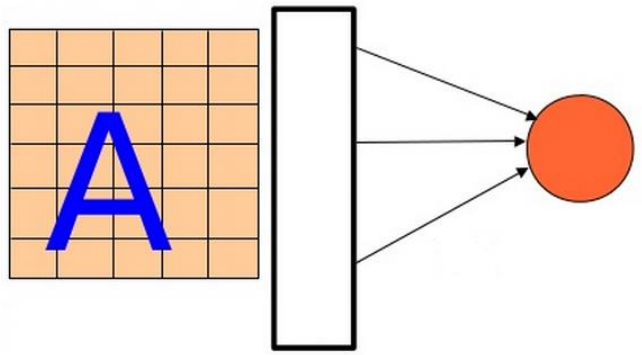
뉴런: 신경망의 기본 단위

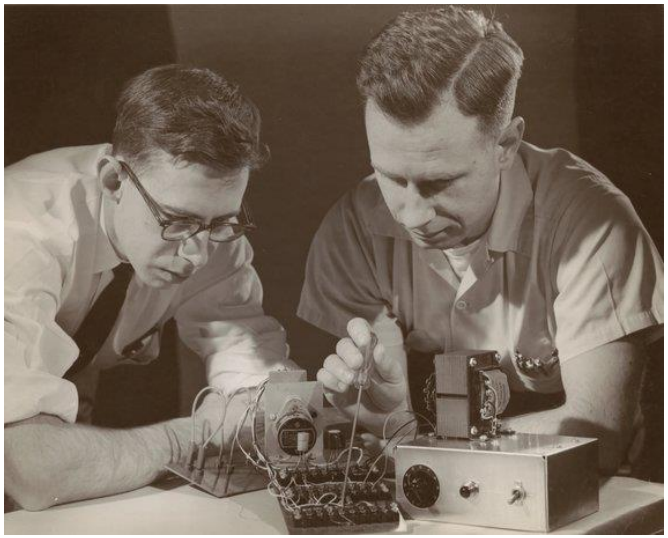



Cell body
Axon
Telodendria
Nucleus
Axon hillock
Synaptic terminals
Golgi apparatus
Endoplasmic reticulum
Mitochondrion
Dendrite
Dendritic branches

Basic model

- The first learning machine: the Perceptron (built in 1960)
- The perceptron was a linear classifier

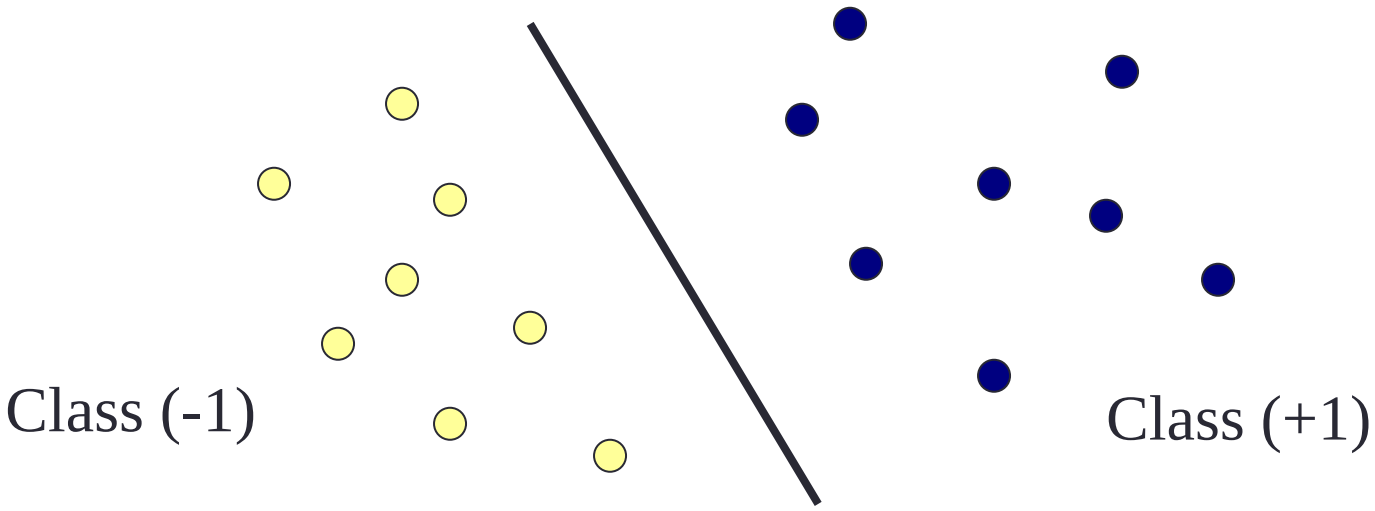




$$y = \text{sign}(\mathrm{w}^{\mathrm{T}}\mathrm{x} + \mathrm{b})$$

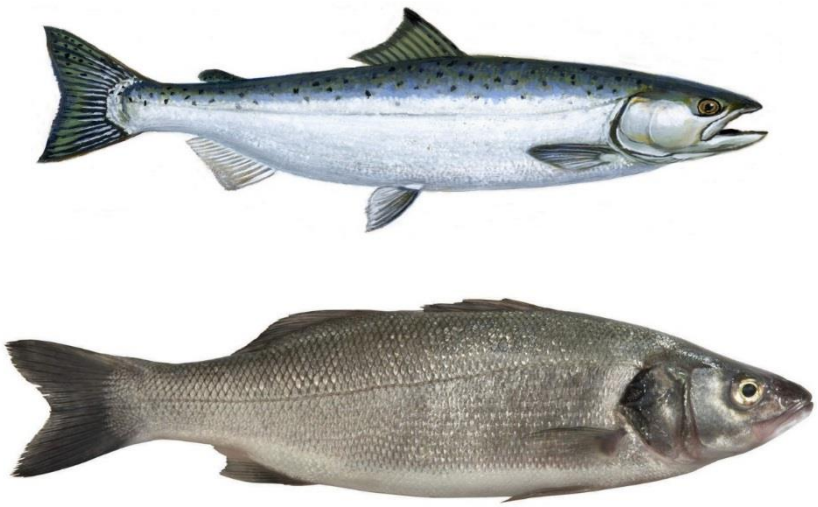
$$y = \begin{cases} +1 & \text{if } w_1x_1 + w_2x_2 + \cdots + w_nx_n + b > 0 \\ -1 & \text{otherwise} \end{cases}$$

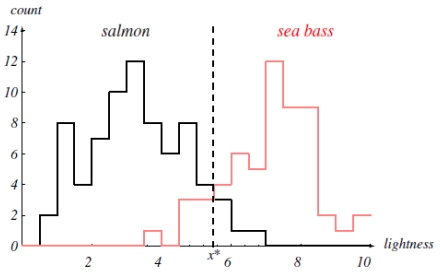
Linear Perceptron

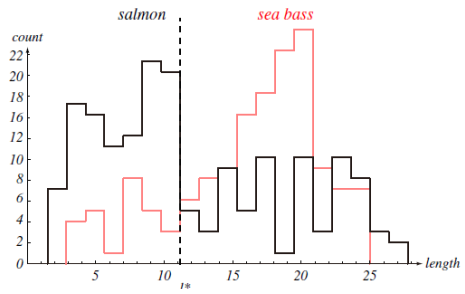


- The goal: Find the best line (or hyper-plane) to separate the training data.
  - In two dimensions, the equation of the line is given by a line:
    - $ax + by + c = 0$
  - A better notation for *n* dimensions: treat each data point and the coefficients as vectors. Then the equation is given by:
    - $\mathrm{w}^{\top}\mathrm{x} + \mathrm{b} = 0$

예시: 연어와 농어의 구별




count
salmon
sea bass
lightness
x*


salmon
sea bass
count
length
l*

예시: 연어와 농어의 구별

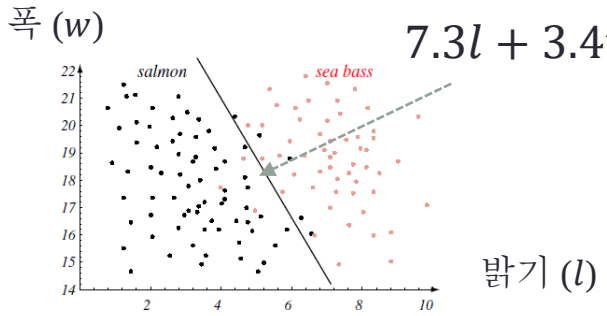
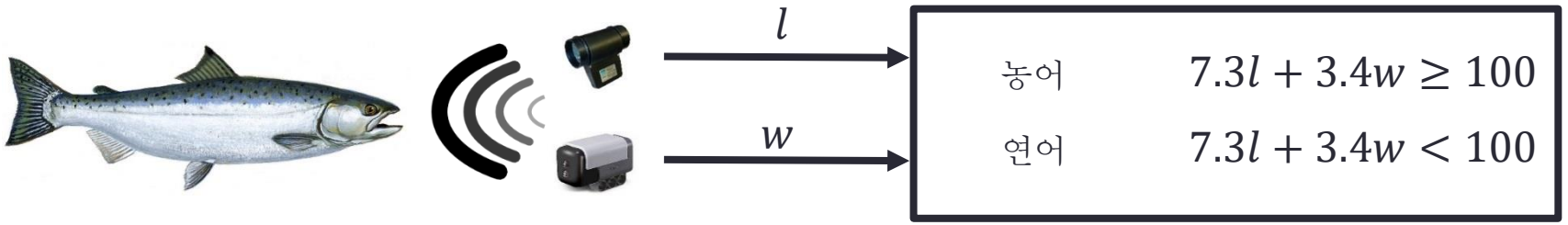
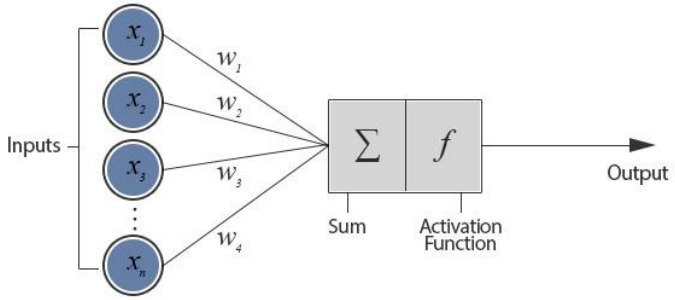
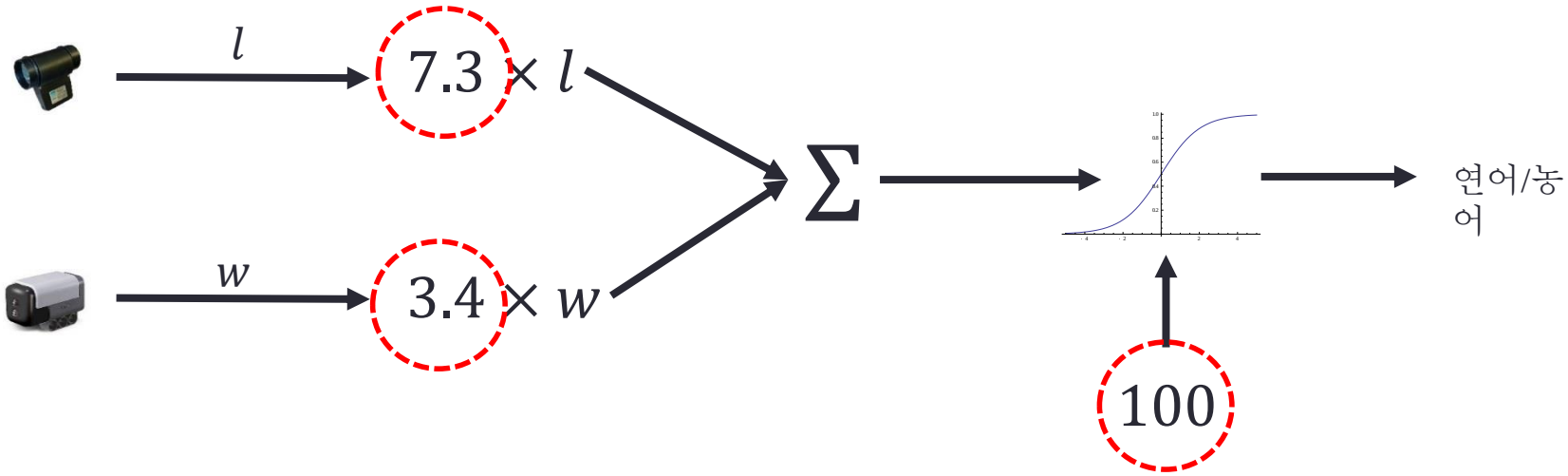


FIGURE 1.4. The two features of lightness and width for sea bass and salmon. The dark line could serve as a decision boundary of our classifier. Overall classification error on the data shown is lower than if we use only one feature as in Fig. 1.3, but there will still be some errors. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

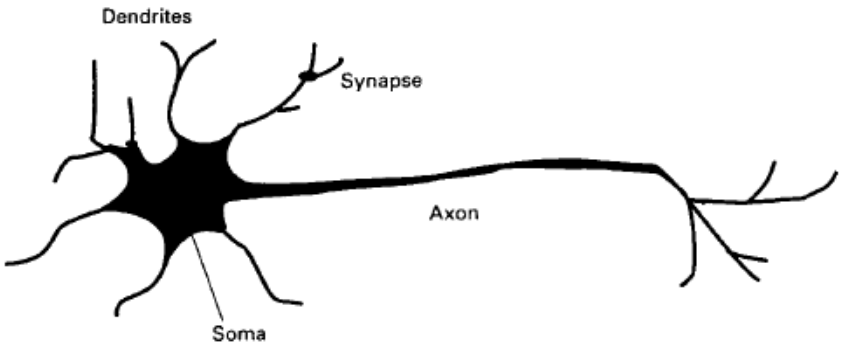


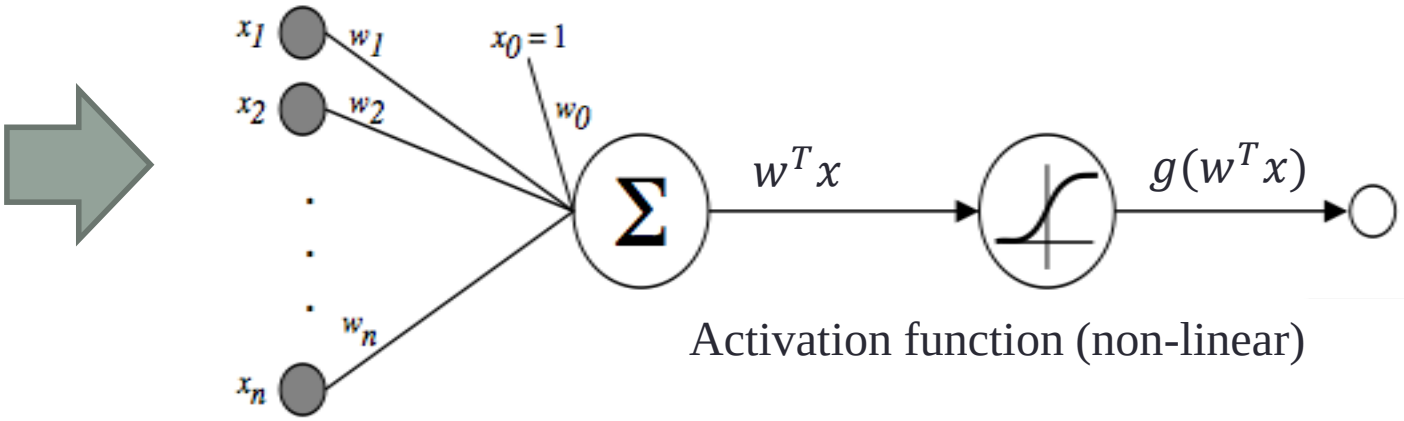
예시: 연어와 농어의 구별


Inputs
x_1
x_2
x_3
x_n
w_1
w_2
w_3
w_4
Σ
f
Sum
Activation Function
Output

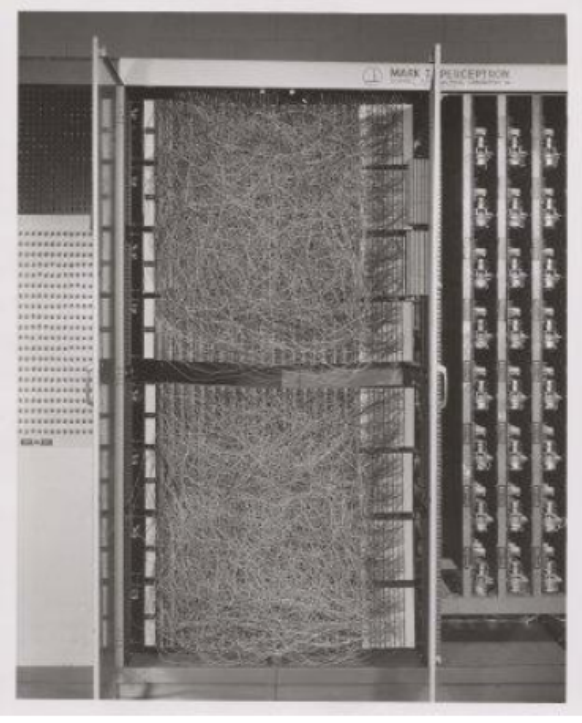

l
$7.3 \times l$
w
$3.4 \times w$
Σ
100
연어/농어

Artificial Neuron





Mark I Perceptron



- Frank Rosenblatt
- 400 pixel image input
- Weights encoded in potentiometers
- Weight updated by electric motors

The New York Times

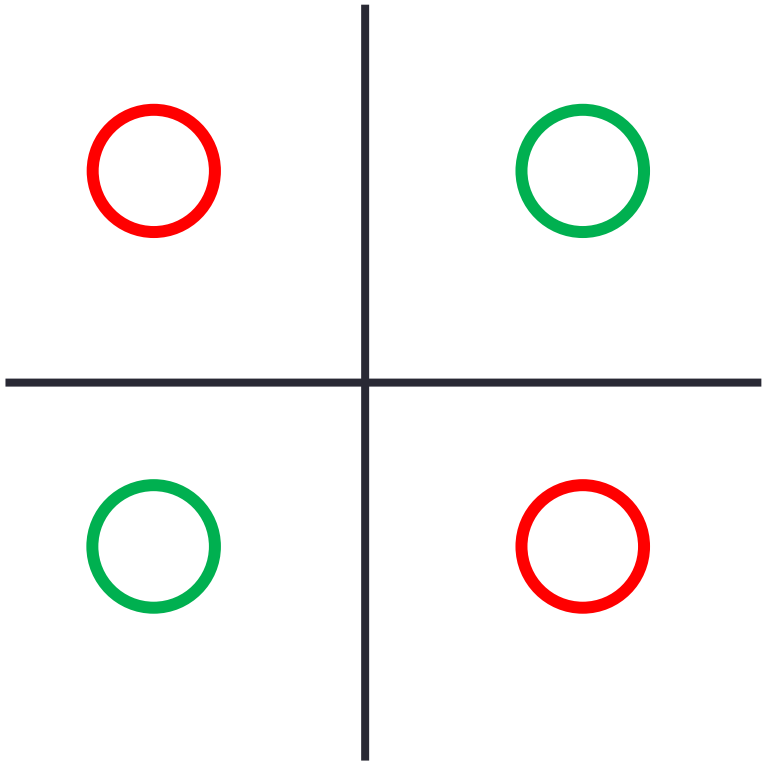
NEW NAVY DEVICE LEARNS BY DOING

July 8, 1958

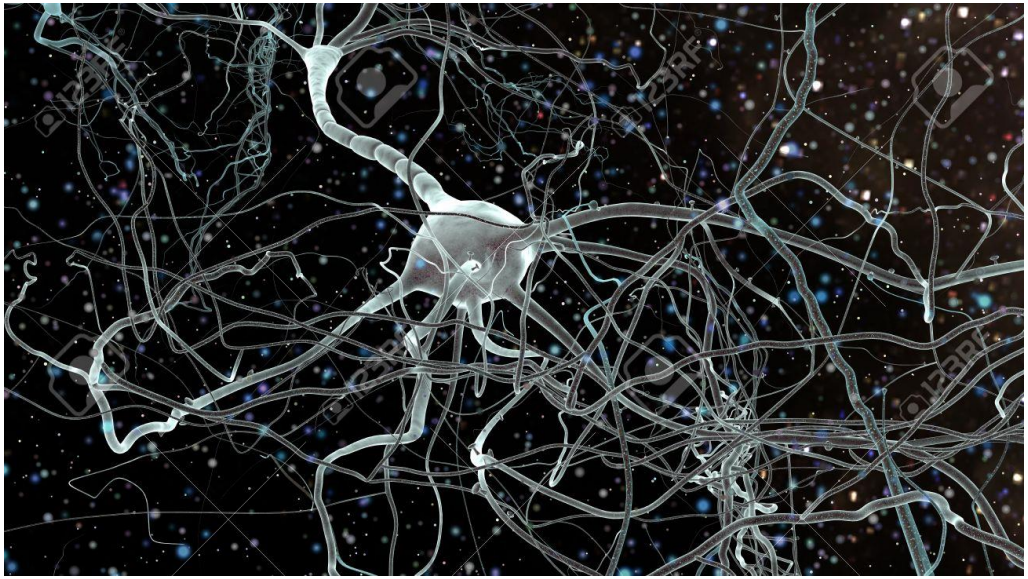
"The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence… Dr. Frank Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers"

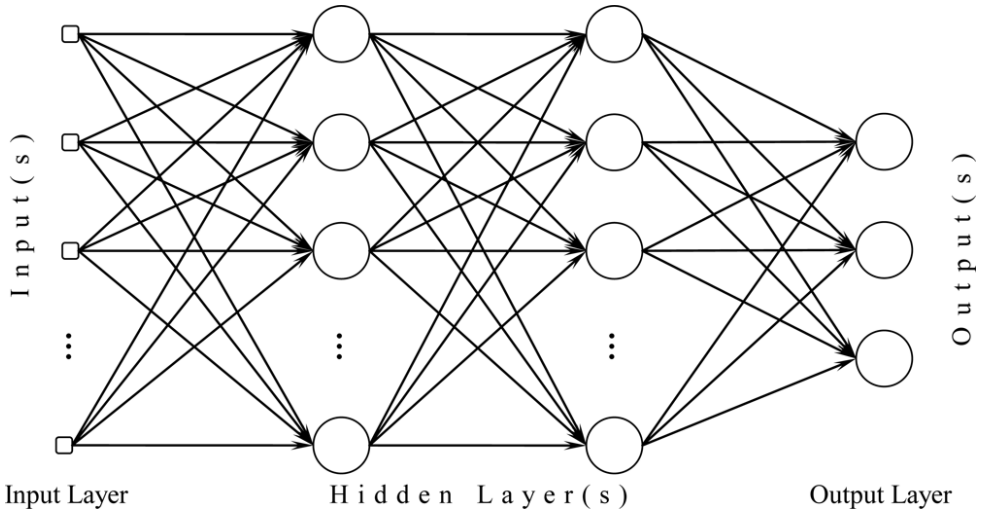
Artificial Neuron

- However, it cannot solve non-linearly-separable problems



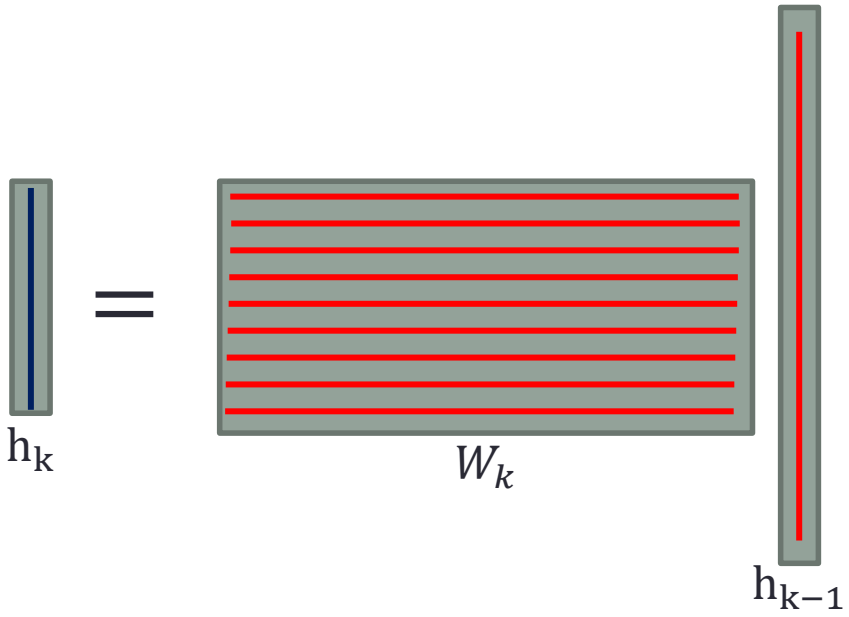
MULTI-LAYER PERCEPTRON




Input(s)
Output(s)
Input Layer
Hidden Layer(s)
Output Layer

Multi-layer Neural Network

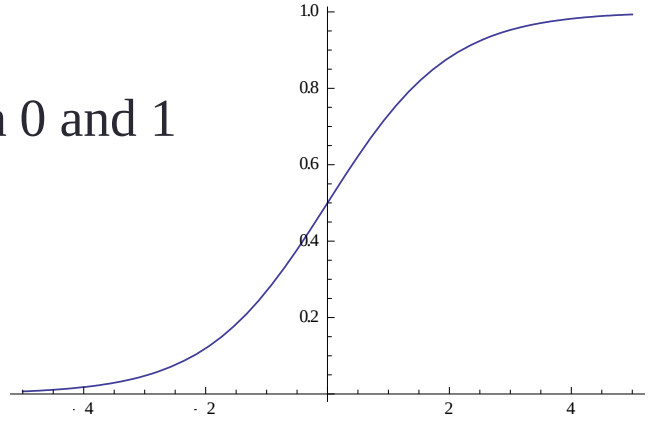
- 1st Layer
  - $\mathrm{h}_1 = g(W_1 \mathrm{x} + b_1)$
- 2nd Layer
  - $\mathrm{h}_2 = g(W_2 \mathrm{h}_1 + b_2)$
- …..
- Output layer
  - $\mathrm{o} = \mathrm{softmax}(\mathrm{W}_\mathrm{n} \mathrm{h}_{\mathrm{n}-1} + \mathrm{b}_\mathrm{n})$



Activation function $g(\cdot)$

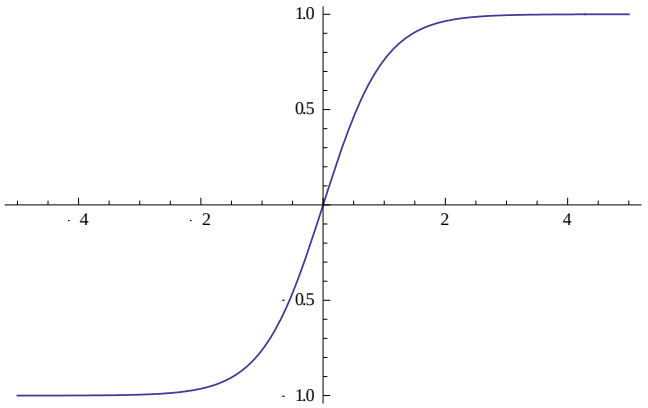
- Sigmoid activation function
  - Squashes the neuron's pre-activation between 0 and 1
  - Always positive/Bounded/Strictly increasing

$$g(x) = \frac{1}{1 + \exp(-x)}$$



- Hyperbolic tangent ("tanh") activation function
  - Squashes the neuron's pre-activation between -1 and 1
  - Bounded/Strictly increasing

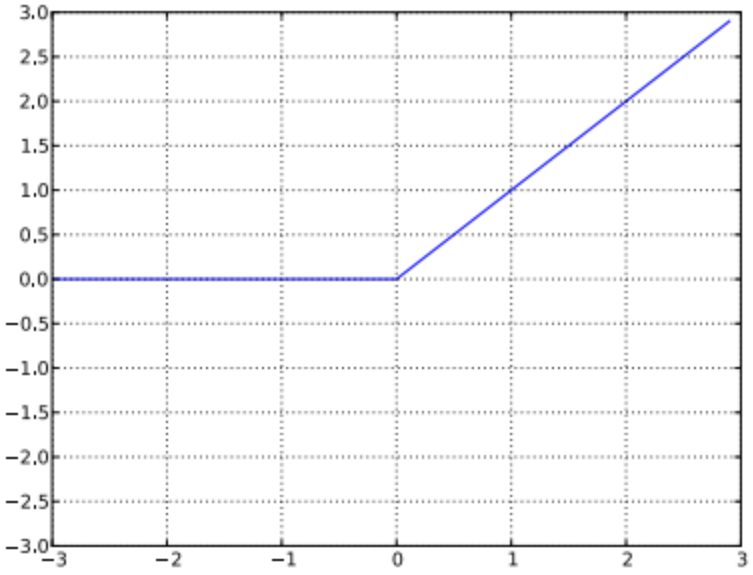
$$g(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$



Activation function $g(\cdot)$

- Rectified linear activation function (ReLU)
  - Bounded below by 0
  - Not upper bounded
  - Strictly increasing

$$g(a) = \text{rectlin}(a) = \max(0, a)$$



Soft-max activation function at the output

- For multi-class classification
  - We need multiple outputs (1 output per class)

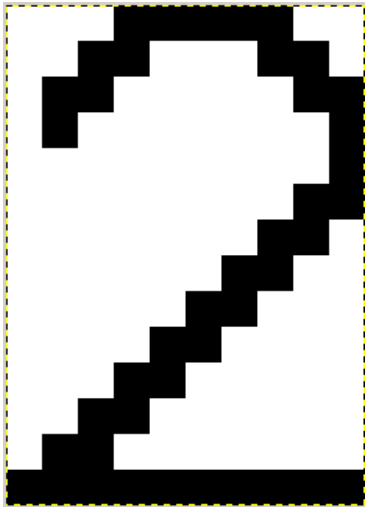
- We use the softmax activation function at the output

$$O(\mathbf{a}) = \text{softmax}(\mathbf{a}) = \begin{bmatrix} \frac{\exp(a_1)}{\sum_c \exp(a_c)} \\ \frac{\exp(a_2)}{\sum_c \exp(a_c)} \\ \vdots \\ \vdots \\ \frac{\exp(a_c)}{\sum_c \exp(a_c)} \end{bmatrix}$$

  - strictly positive
  - sums to one

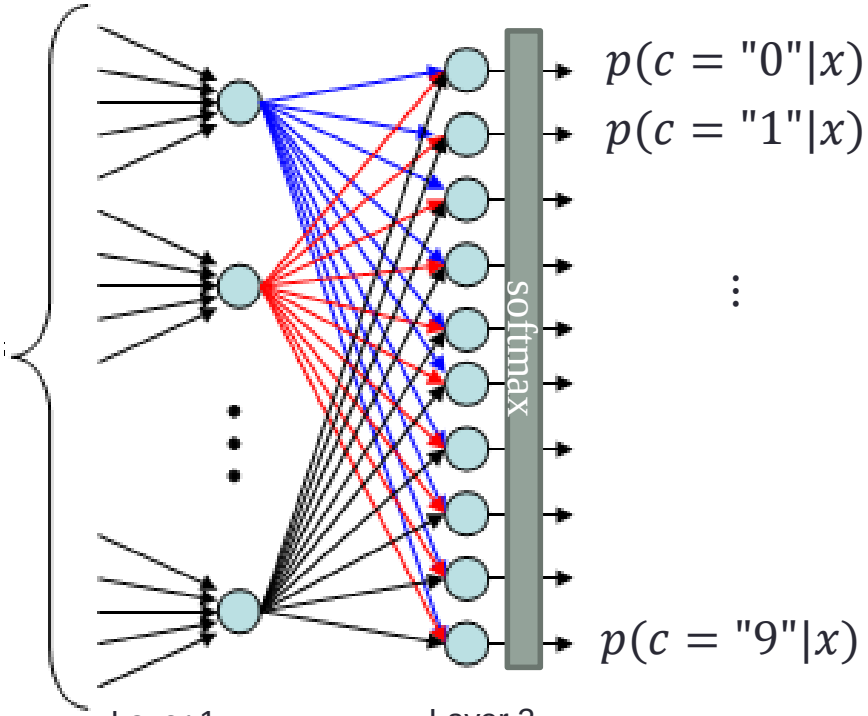
예시: $(a, b, c) \rightarrow \left(\frac{e^a}{e^a+e^b+e^c}, \frac{e^b}{e^a+e^b+e^c}, \frac{e^c}{e^a+e^b+e^c} \right)$

Example (character recognition example)



$x \in \{0,1\}^{10\times 14}$

140 inputs



Layer 1
with 12 perceptrons

$W_1 \in \Re^{12\times 140}$

$b_1 \in \Re^{12}$

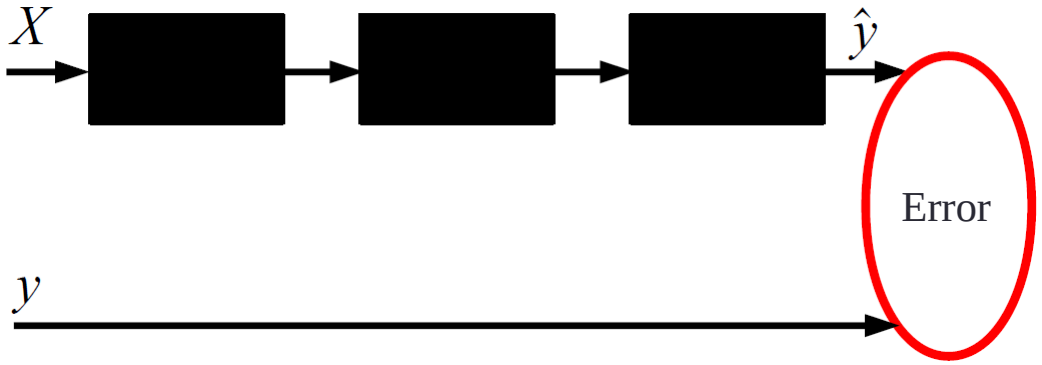
Layer 2
with 10 perceptrons
Each having 12 inputs

$W_2 \in \Re^{10\times 12}$

$b_2 \in \Re^{10}$

TRAINING OF MULTI-LAYER PERCEPTRON

Training: Loss function



- Cross entropy (classification)
  - $y, \hat{y} \in [0,1]^N, \sum_{i=1} y_i = 1, \sum_{i=1} \hat{y}_i = 1$
  - $L = -\sum y_i \log \hat{y}_i$

- Square Euclidean distance (regression)
  - $y, \hat{y} \in \Re^N$
  - $L = \frac{1}{2}\sum (y_i - \hat{y}_i)^2$

Cross Entropy (예시)

- Label:
  - $[y_1 \ y_2 \ y_3]$ = [1,0,0] : class 1
  - $[y_1 \ y_2 \ y_3]$ = [0,1,0] : class 2
  - $[y_1 \ y_2 \ y_3]$ = [0,0,1] : class 3

$$L = -\sum y_i \log \hat{y}_i$$

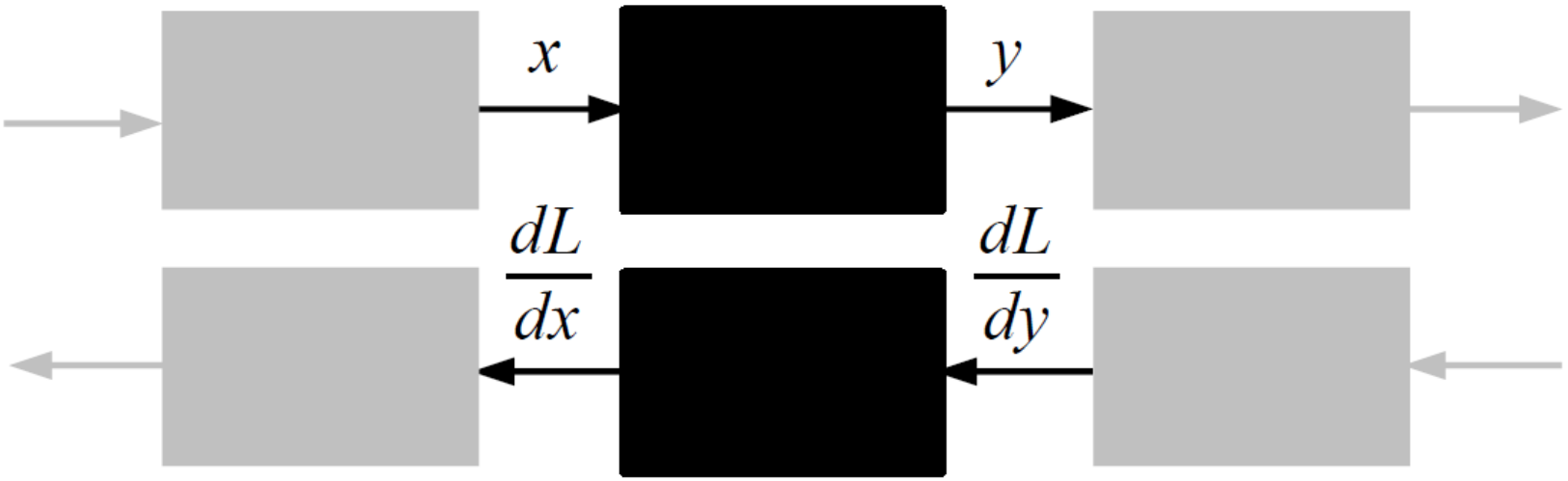
- 예시
  - Network output: $\hat{y} = [\hat{y}_1 \ \hat{y}_2 \ \hat{y}_3] = [0.3, 0.6, 0.1]$
    - Loss
      - If ground truth is class 1 (i.e., y = [1,0,0]) → $-\log 0.3 = 1.204$
      - If ground truth is class 2 (i.e., y = [0,1,0]) → $-\log 0.6 = 0.511$
      - If ground truth is class 3 (i.e., y = [0,0,1]) → $-\log 0.1 = 2.303$
  - Network output: $\hat{y} = [\hat{y}_1 \ \hat{y}_2 \ \hat{y}_3] = [0.01, 0.98, 0.01]$
    - Loss
      - If ground truth is class 1 (i.e., y = [1,0,0]) → $-\log 0.01 = 4.605$
      - If ground truth is class 2 (i.e., y = [0,1,0]) → $-\log 0.98 = 0.020$
      - If ground truth is class 3 (i.e., y = [0,0,1]) → $-\log 0.01 = 4.605$

Cross entropy (예시)

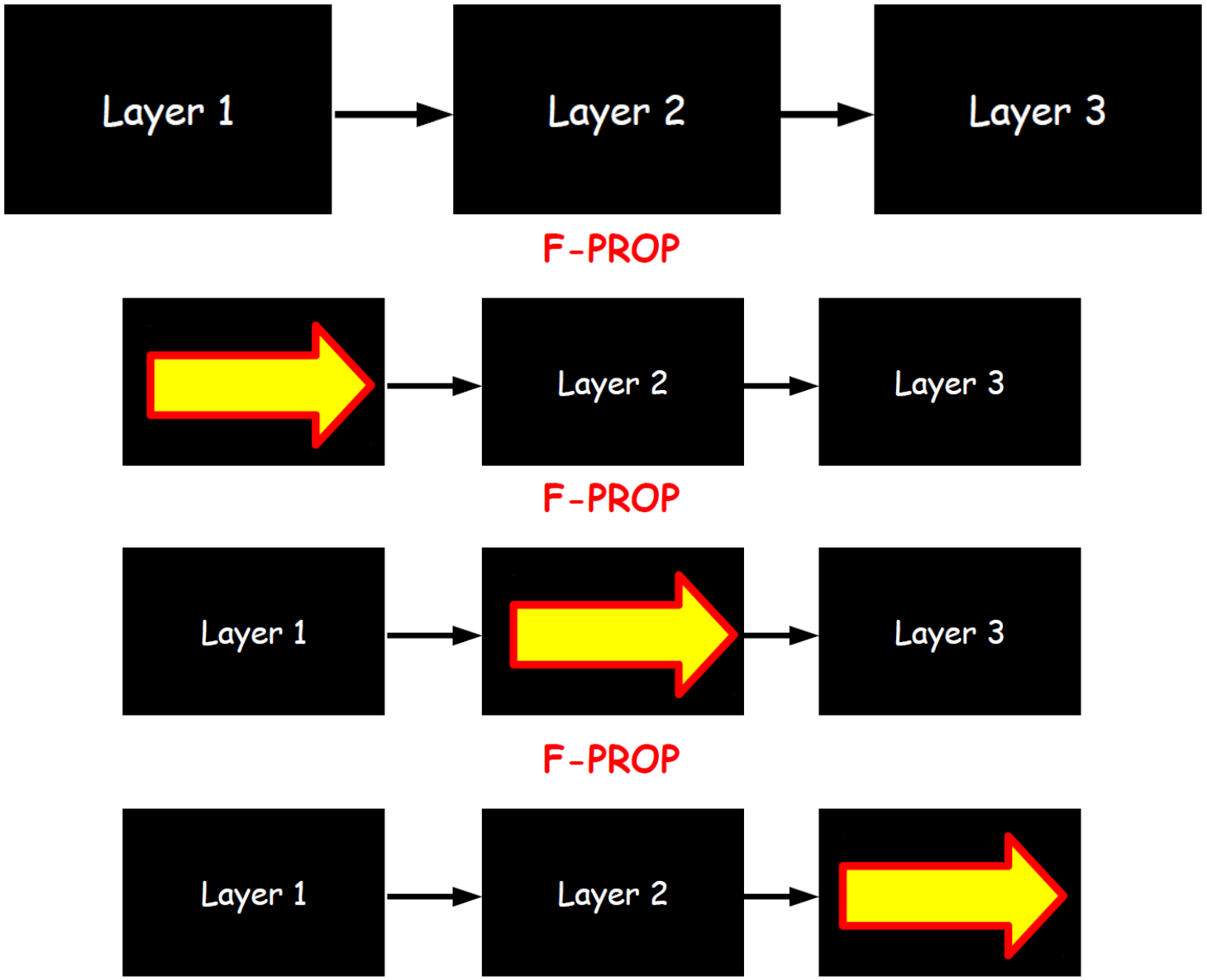
	If ground truth is class 1 (i.e., y = [1,0,0])	If ground truth is class 2 (i.e., y = [0,1,0])	If ground truth is class 3 (i.e., y = [0,0,1])
$[\hat{y}_1 \quad \hat{y}_2 \quad \hat{y}_3] = [0.3, 0.6, 0.1]$	$-\log 0.3 = 1.204$	$-\log 0.6 = 0.511$	$-\log 0.1 = 2.303$
$[\hat{y}_1 \quad \hat{y}_2 \quad \hat{y}_3] = [0.01, 0.98, 0.01]$	$-\log 0.01 = 4.605$	$-\log 0.98 = 0.020$	$-\log 0.01 = 4.605$

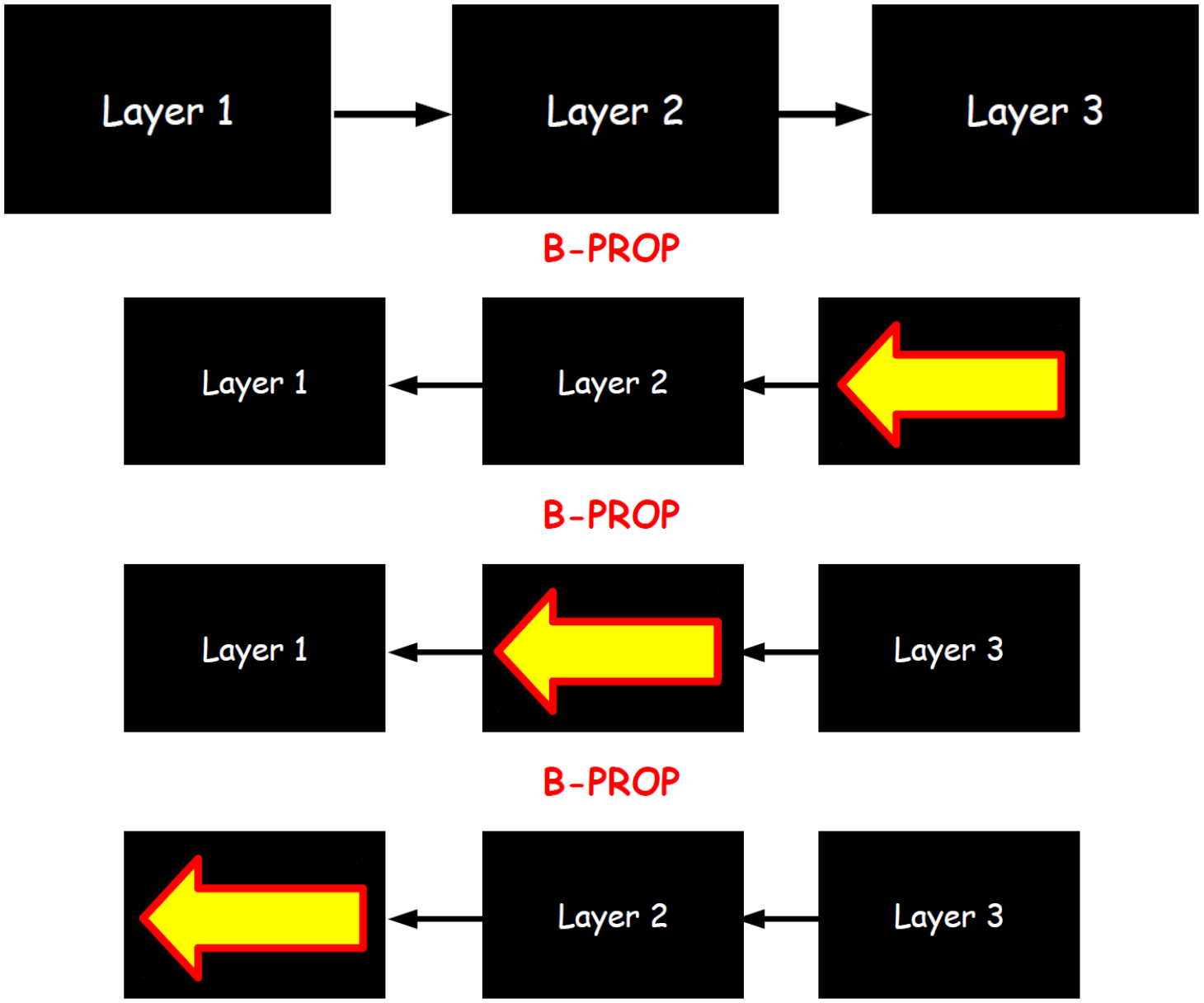
Forward/Backward propagation

- Chain rule

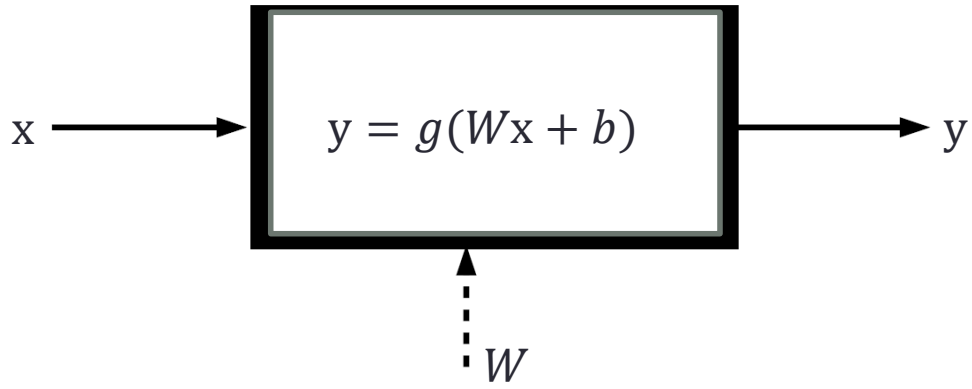


$$W^{new} = W^{old} - \eta \frac{dL}{dW}$$


Layer 1
Layer 2
Layer 3
F-PROP
Layer 2
Layer 3
F-PROP
Layer 1
Layer 3
F-PROP
Layer 1
Layer 2

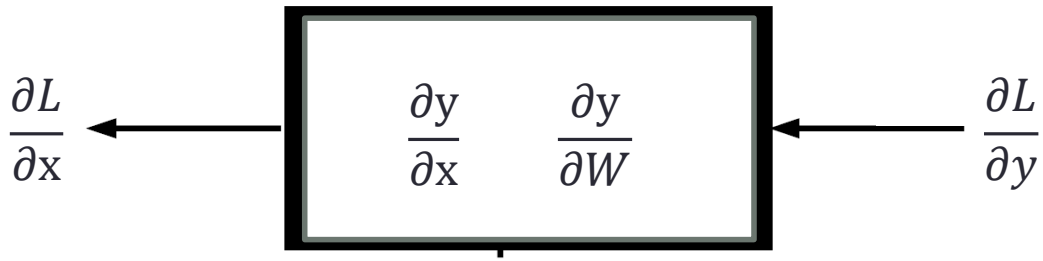

Layer 1
Layer 2
Layer 3
B-PROP
Layer 1
Layer 2
B-PROP
Layer 1
Layer 3
B-PROP
Layer 2
Layer 3

Forward/Backward propagation



Forward propagation

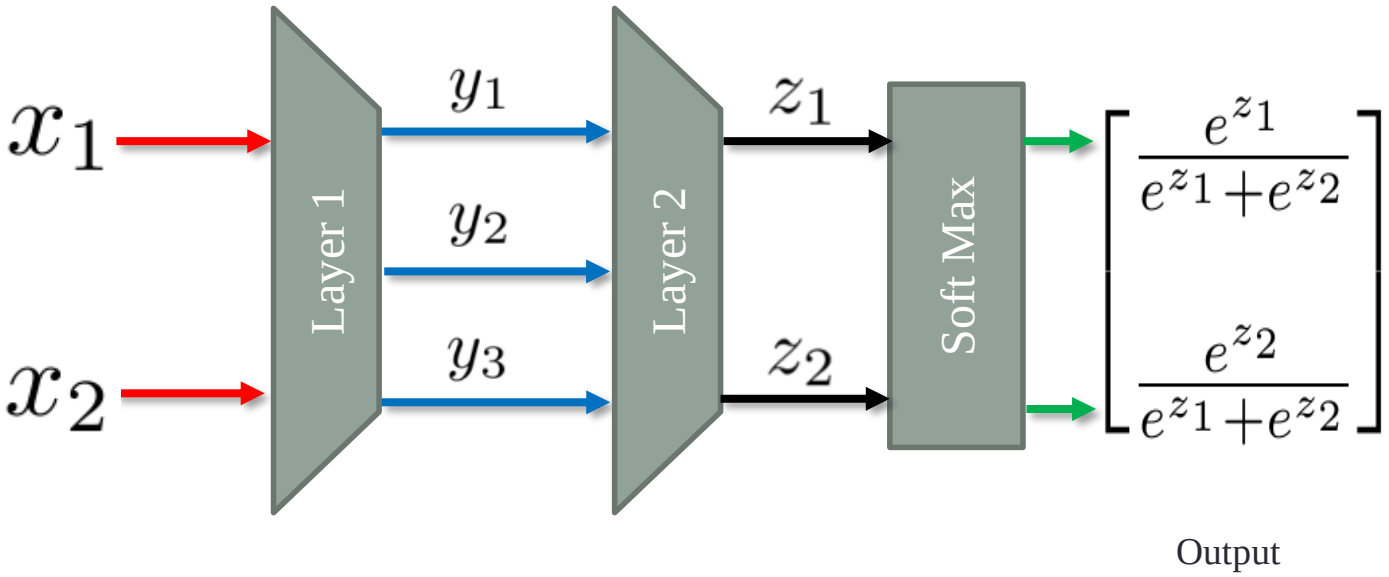
$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial x}$$



Backward propagation

FEED-FORWARD NEURAL NETWORK

Forward propagation



$$y_1 = \varphi(w_{11}x_1 + w_{12}x_2 + b_1)$$

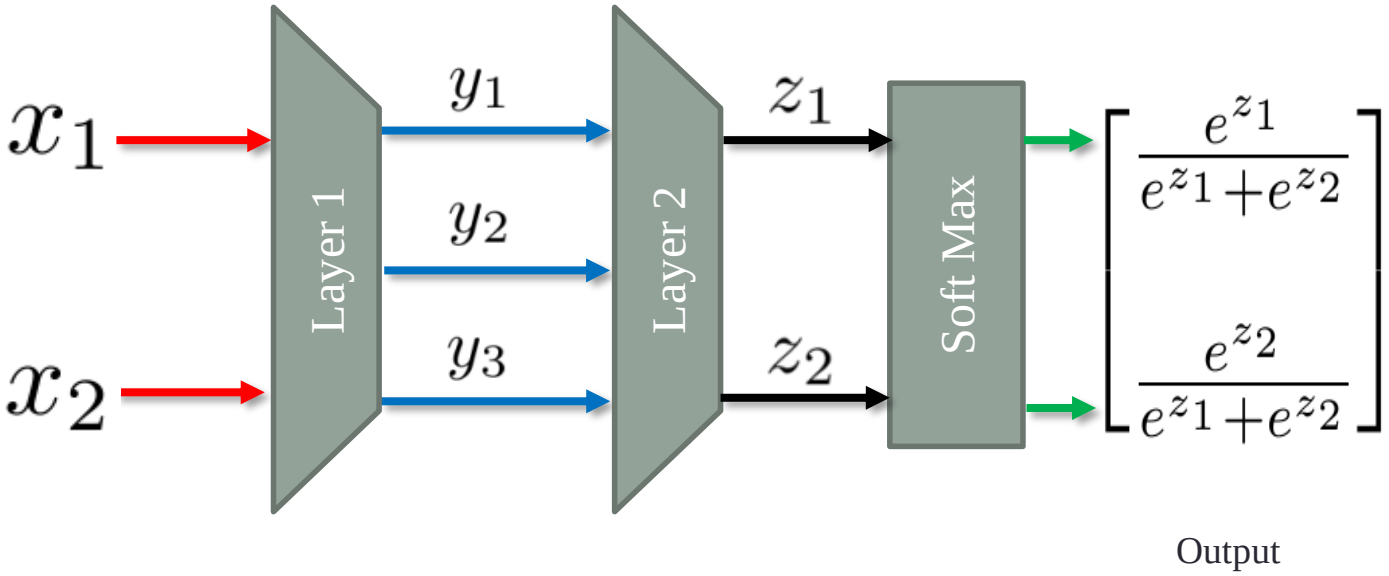
$$y_2 = \varphi(w_{21}x_1 + w_{22}x_2 + b_2)$$

$$y_3 = \varphi(w_{31}x_1 + w_{32}x_2 + b_3)$$

$$z_1 = u_{11}y_1 + u_{12}y_2 + u_{13}y_3 + c_1$$

$$z_2 = u_{21}y_1 + u_{22}y_2 + u_{23}y_3 + c_2$$

Forward propagation matrix repr.



$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \varphi\left(\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$