NEURAL NETWORKS

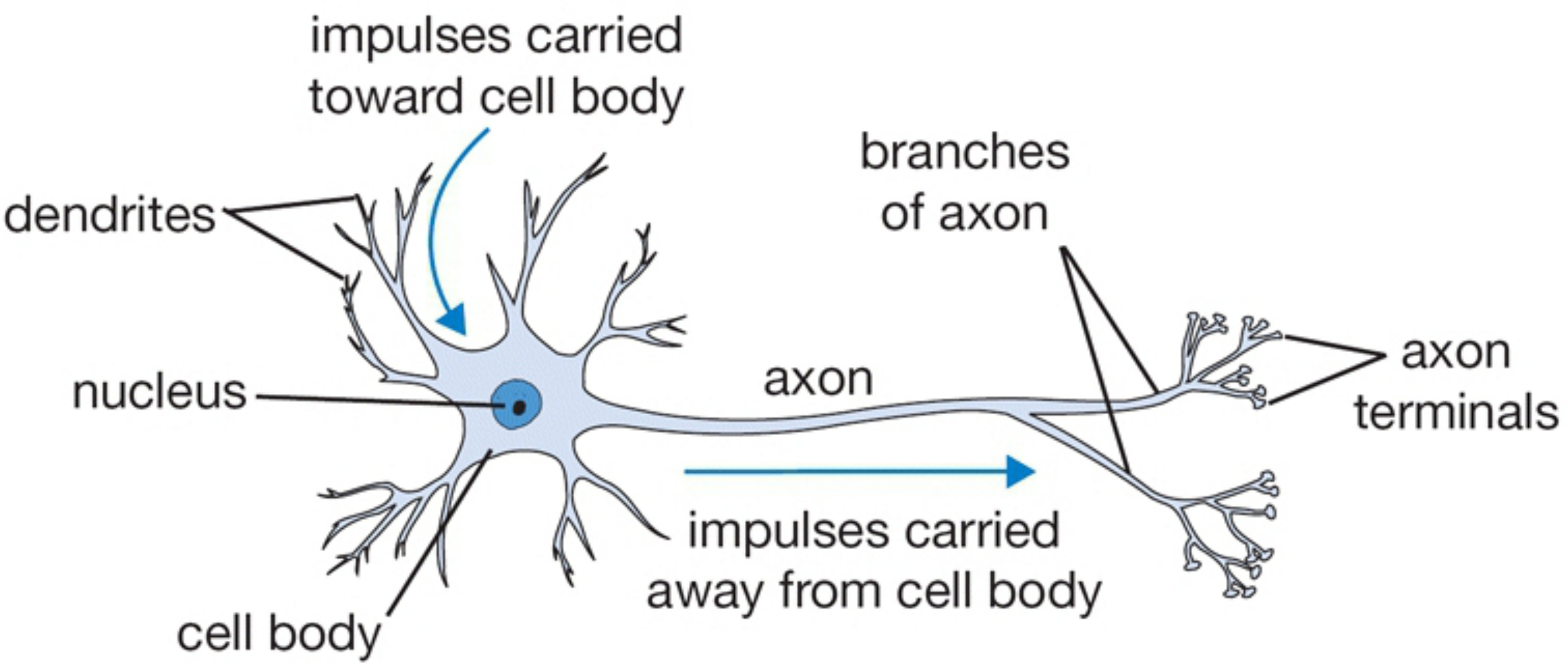
ARTIFICIAL NEURAL NETWORK

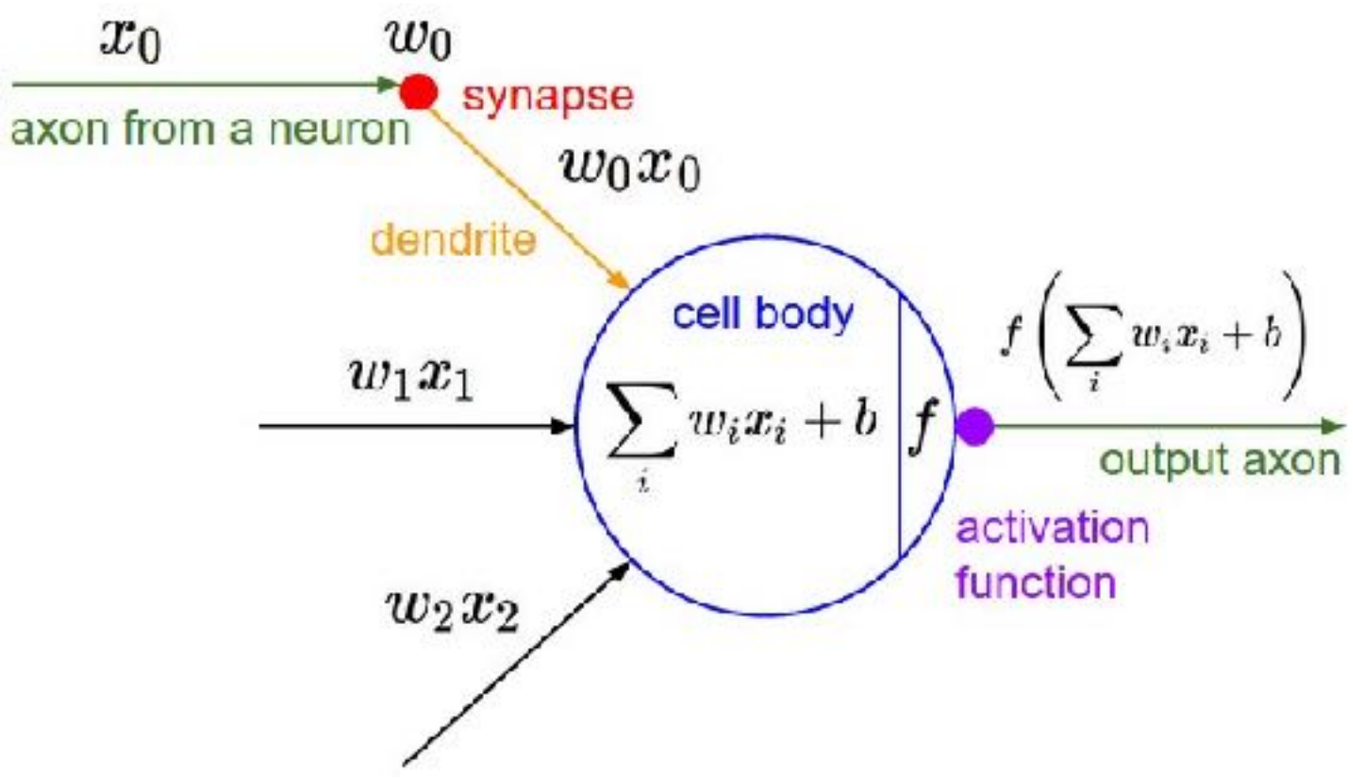
- 인간의 뇌 시스템을 모방

- 뉴런은 연결된 다른 뉴런의 '입력' 들을 받고, 그 결과가 '어느이상'의 자극이 될 시 이를 다른 뉴런에 전달

- 입력 : $XW^T + b$

- '어느이상' 자극의 기준 : Activation function f()

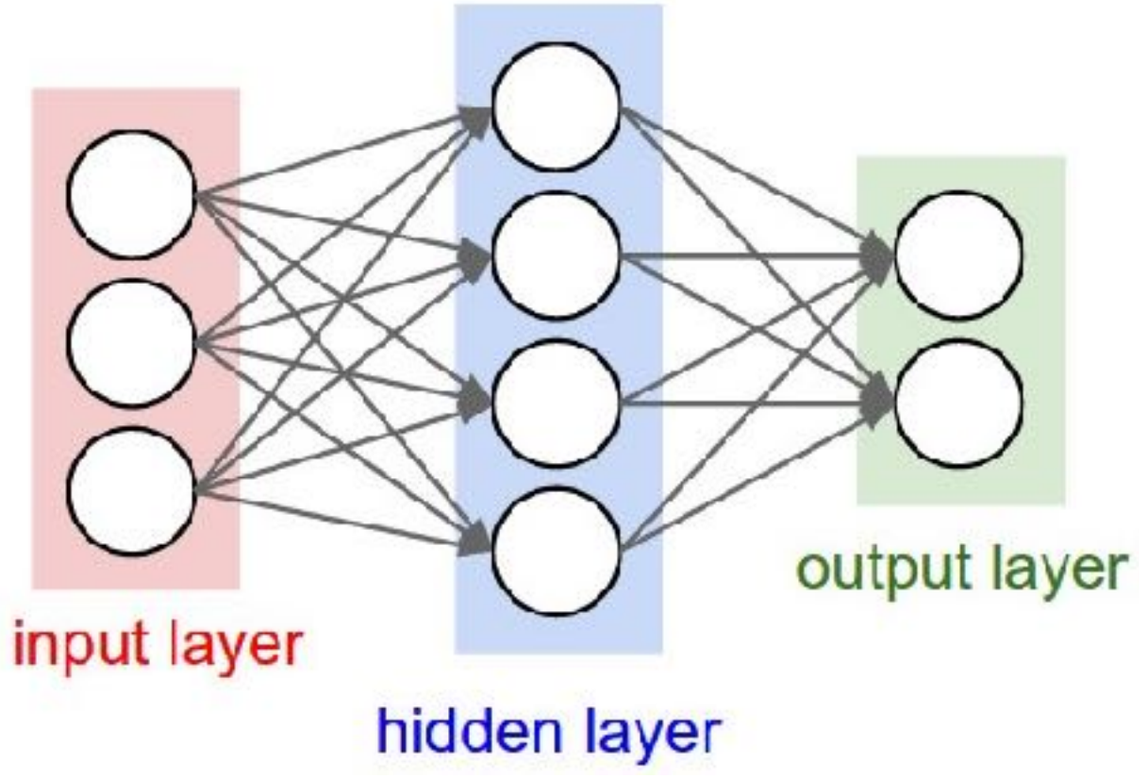




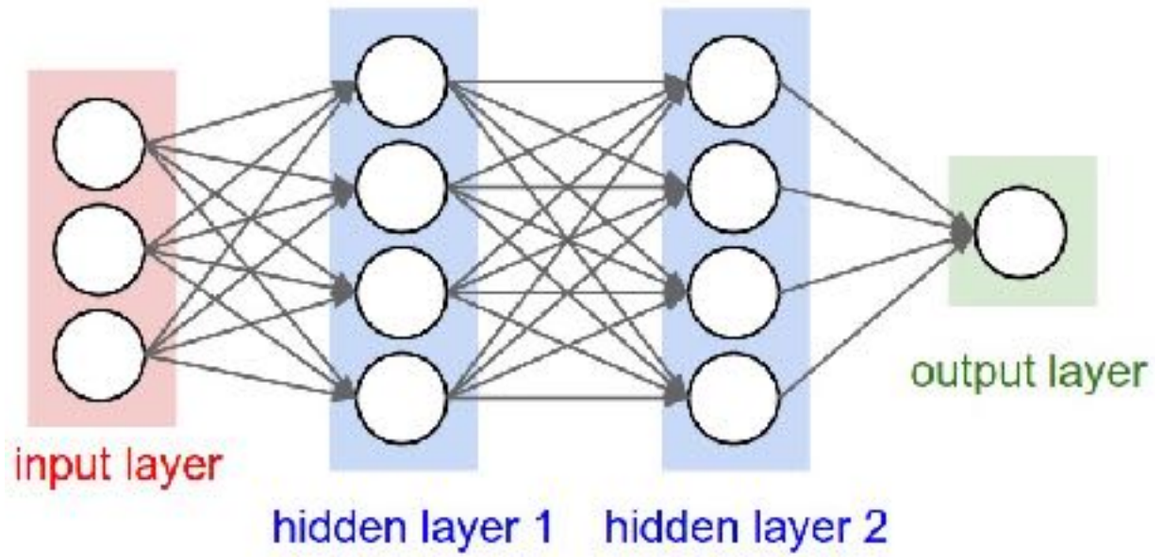
$$y = \phi(\sum_i w_i x_i + b)$$

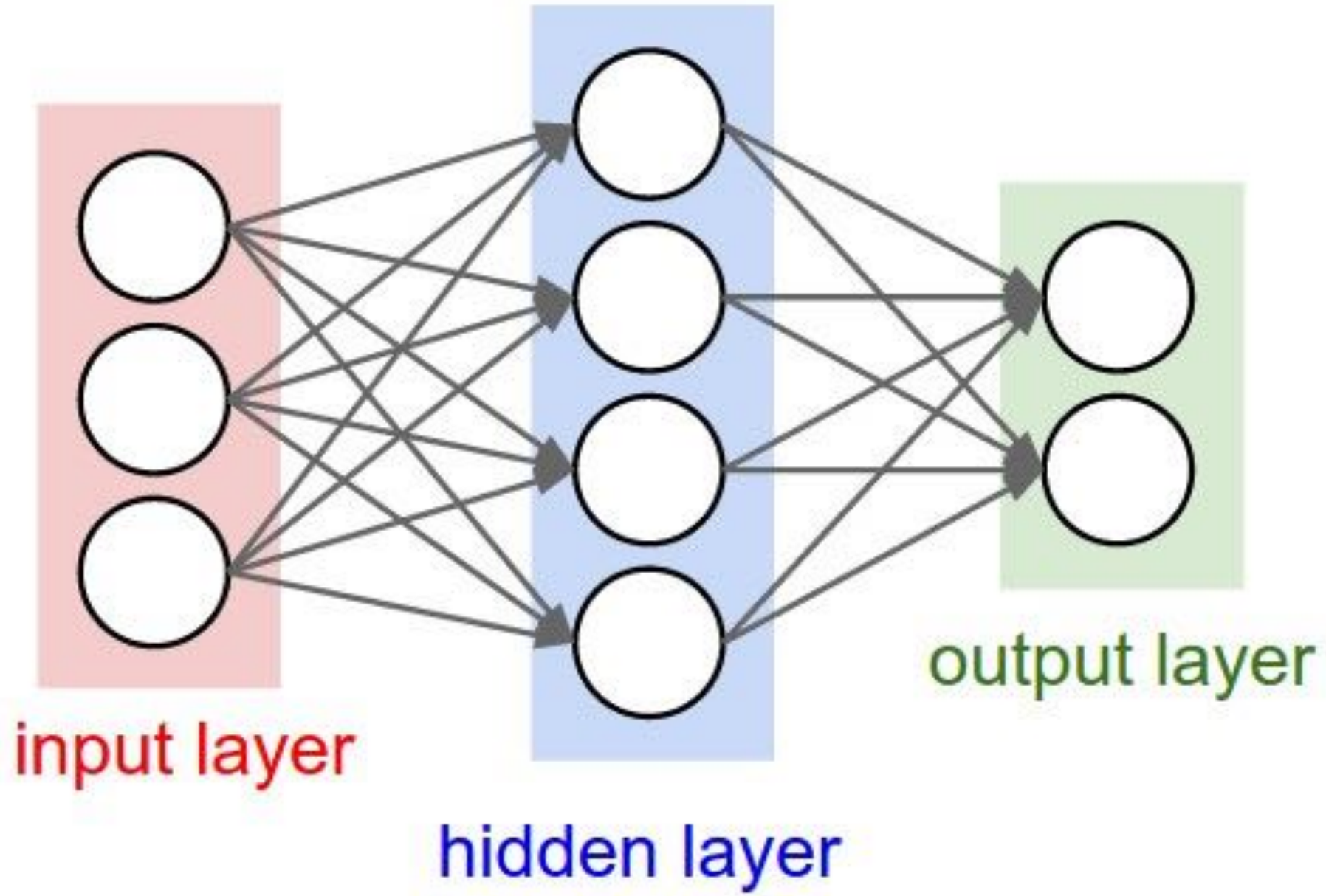
DEEP NEURAL NETWORK

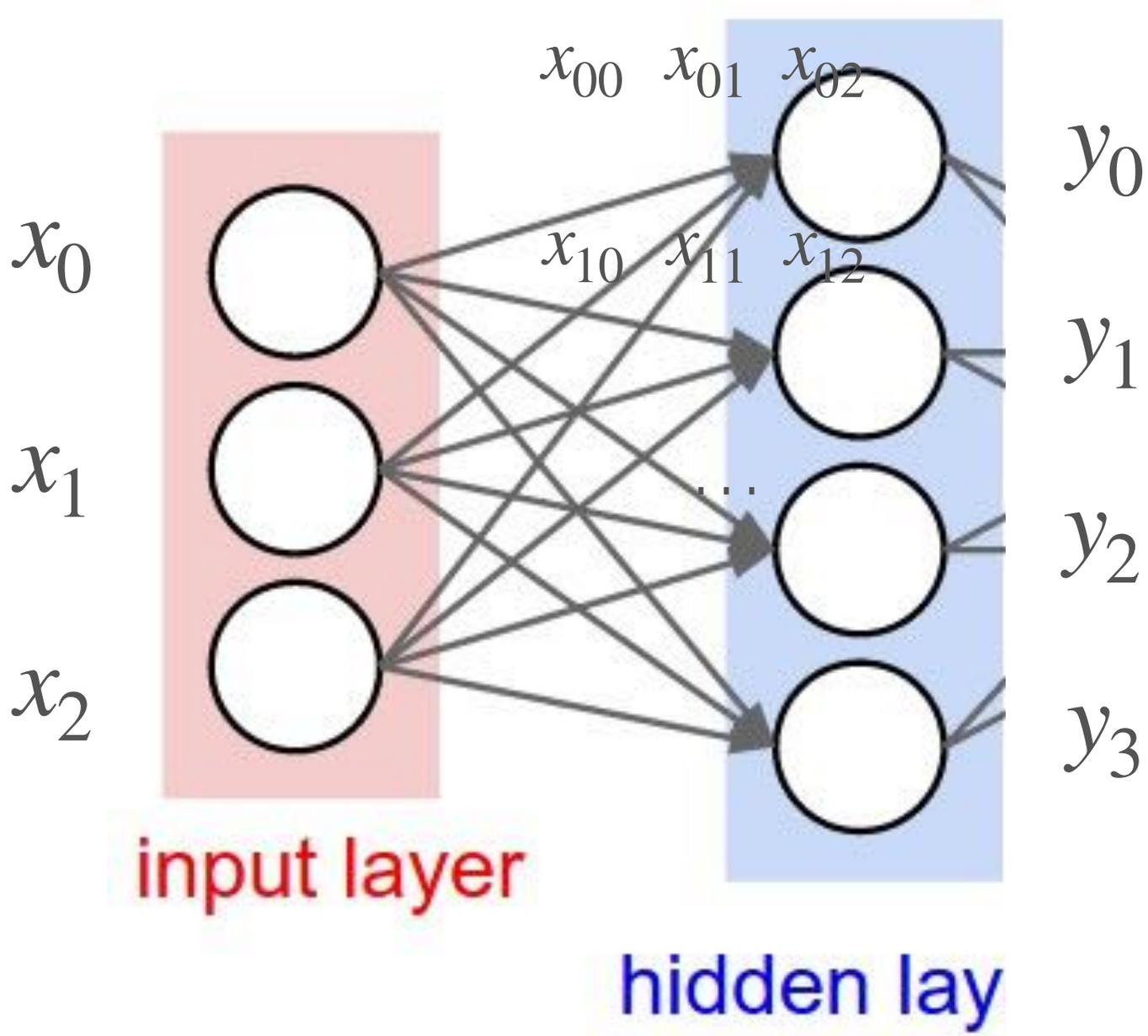
- 2개 이상의 hidden layer 를 가지면 DNN 이라 함
- 거의 실무에 활용되는 모든 NN 모델은 DNN 이라고 보면 됨

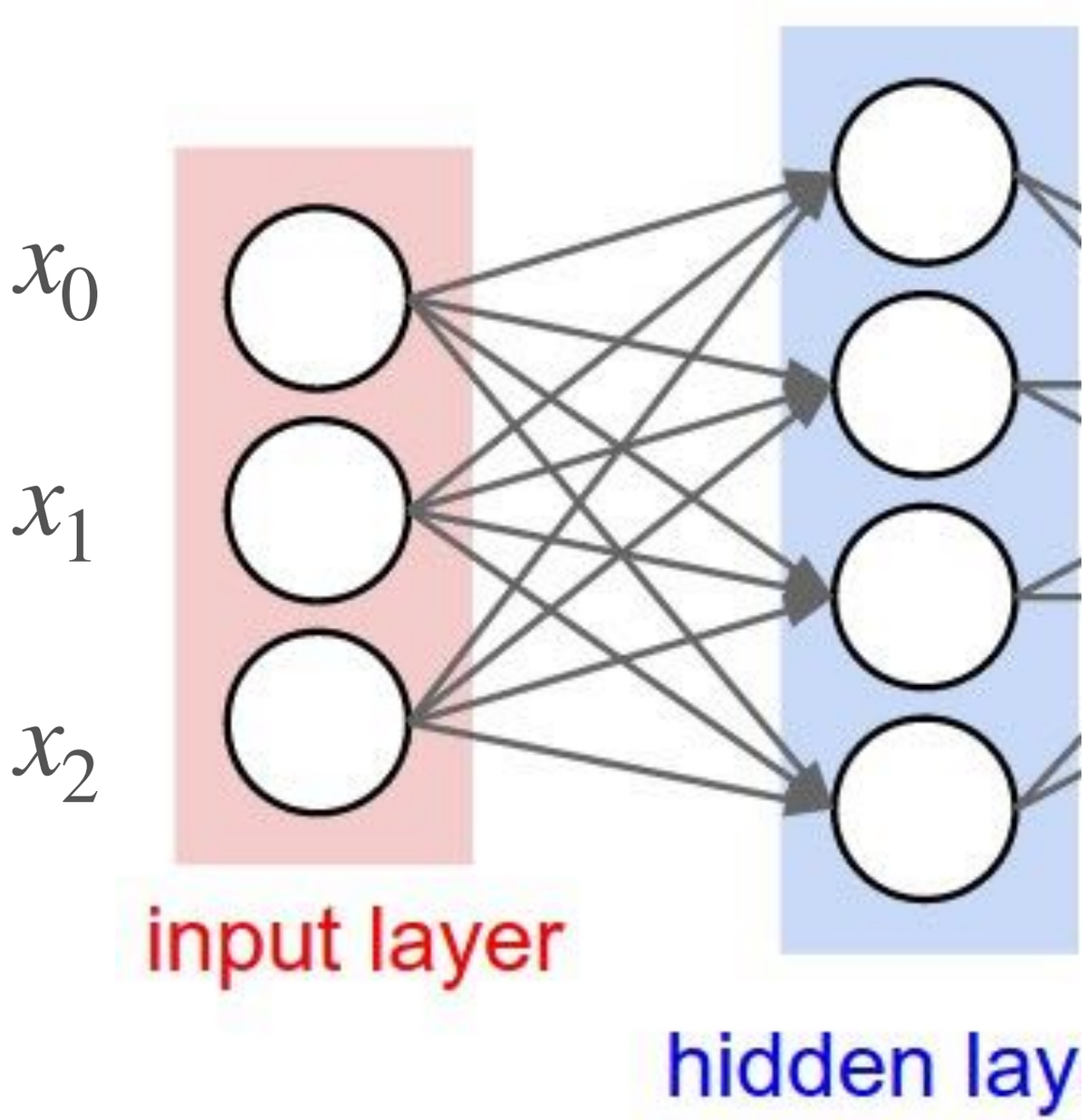


- 단순한 Linear 연산의 조합이 되지 않도록, activation function 단계가 있음




input layer
hidden layer
output layer


x_{00} x_{01} x_{02}
x_0
x_{10} x_{11} x_{12}
y_0
y_1
x_1
...
y_2
x_2
y_3
input layer
hidden lay



$$y_0 = \phi(\sum_{i=0}^{2} x_{0i} w_{0i})$$

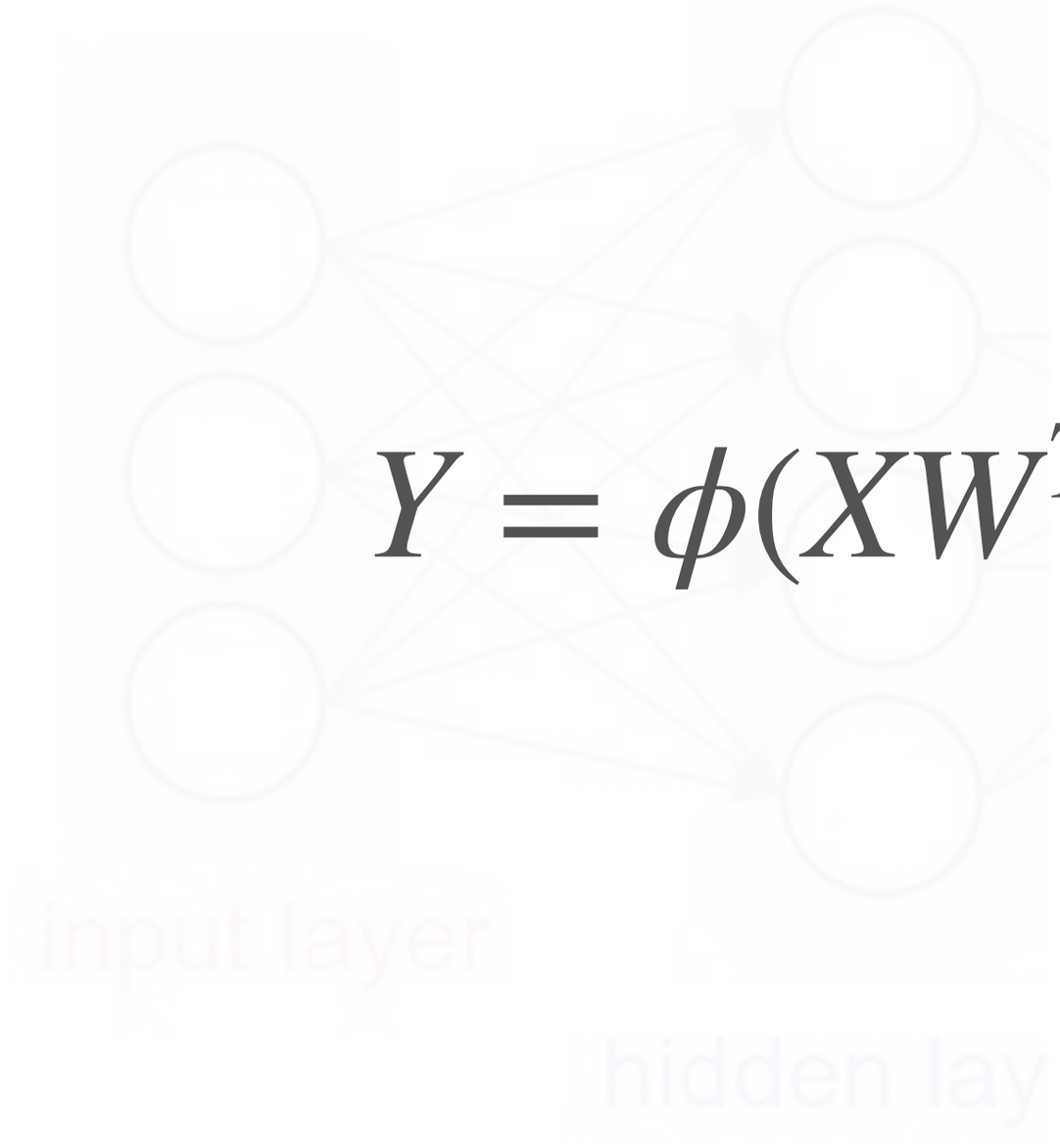
$$y_1 = \phi(\sum_{i=0}^{2} x_{1i} w_{1i})$$

...

$$
\begin{matrix}
Y & & X & & W^{\top} & & b \\
\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} & = \phi\Bigg(& \begin{bmatrix} x_{00} & x_{01} & x_{02} \\ x_{10} & x_{11} & x_{12} \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & x_{32} \end{bmatrix} & \begin{bmatrix} w_{00} & w_{10} & w_{20} & w_{30} \\ w_{01} & w_{11} & \cdots & \cdots \\ w_{02} & w_{12} & \cdots & \cdots \end{bmatrix} & + & \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \Bigg)
\end{matrix}
$$

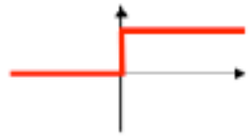
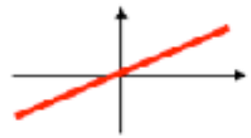
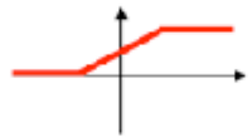
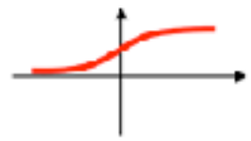
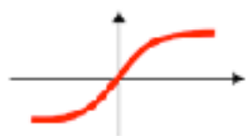
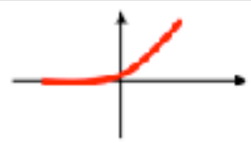
activation

$$Y = \phi(XW^T + b)$$

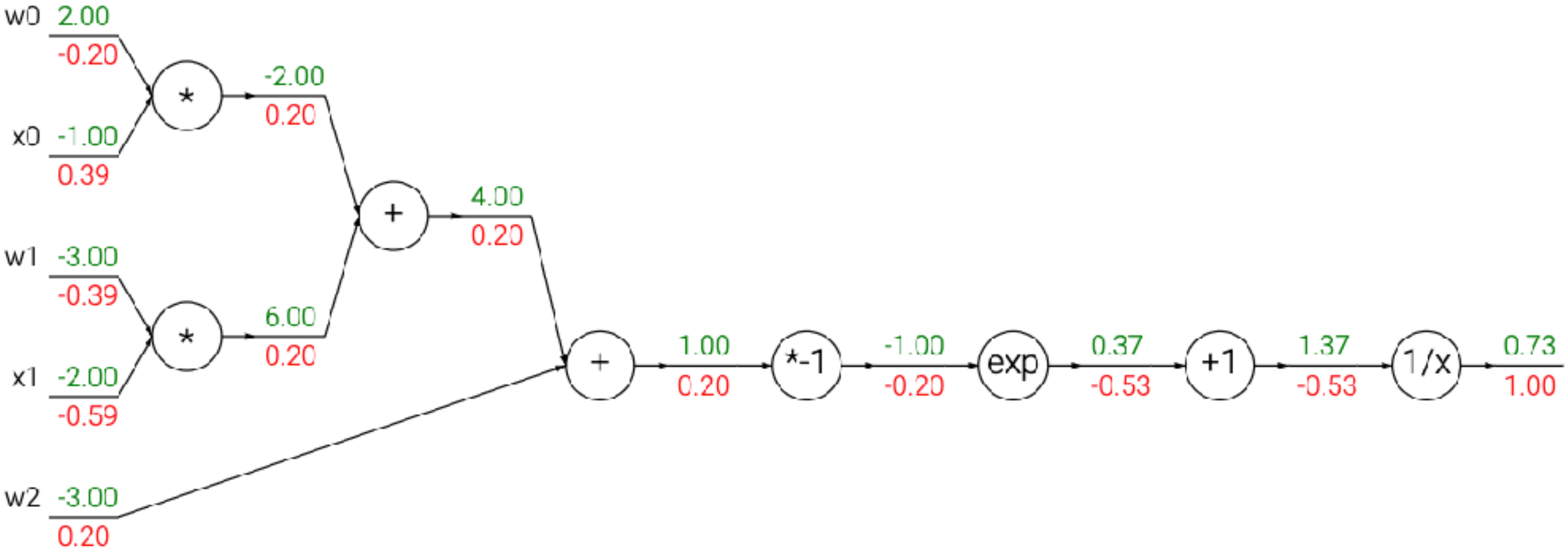


$$Y = \phi(XW^T + b)$$

MATRIX REPRESENTATION OF **SINGLE LAYER FORWARDING**

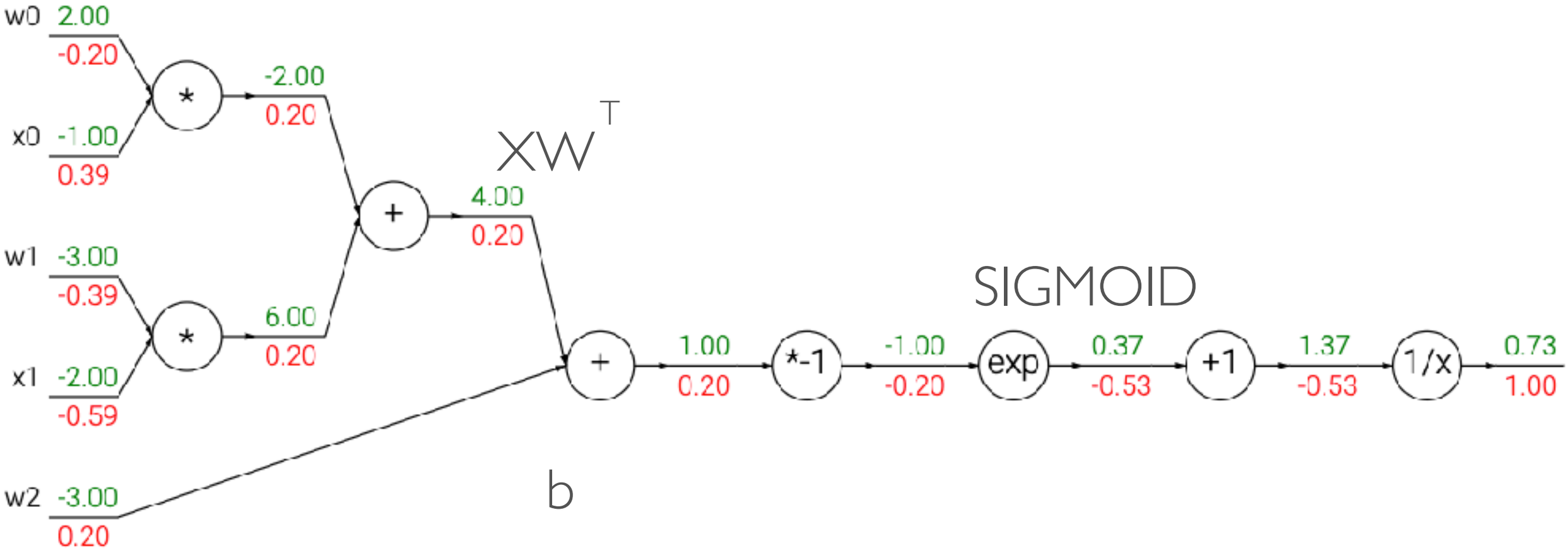
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0. \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

ACTIVATION FUNCTIONS



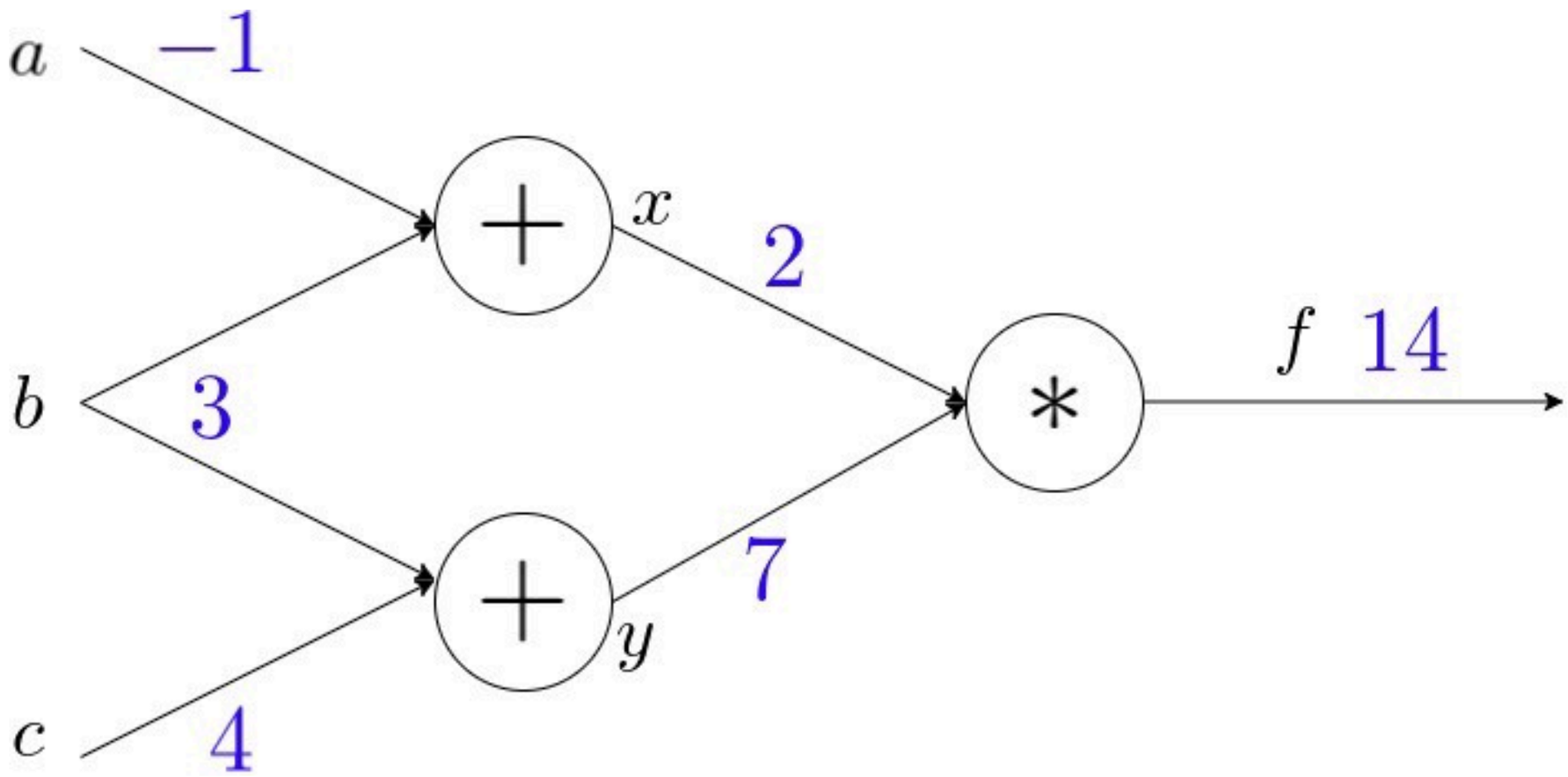
PROPAGATIONS

Forwards / Backwards

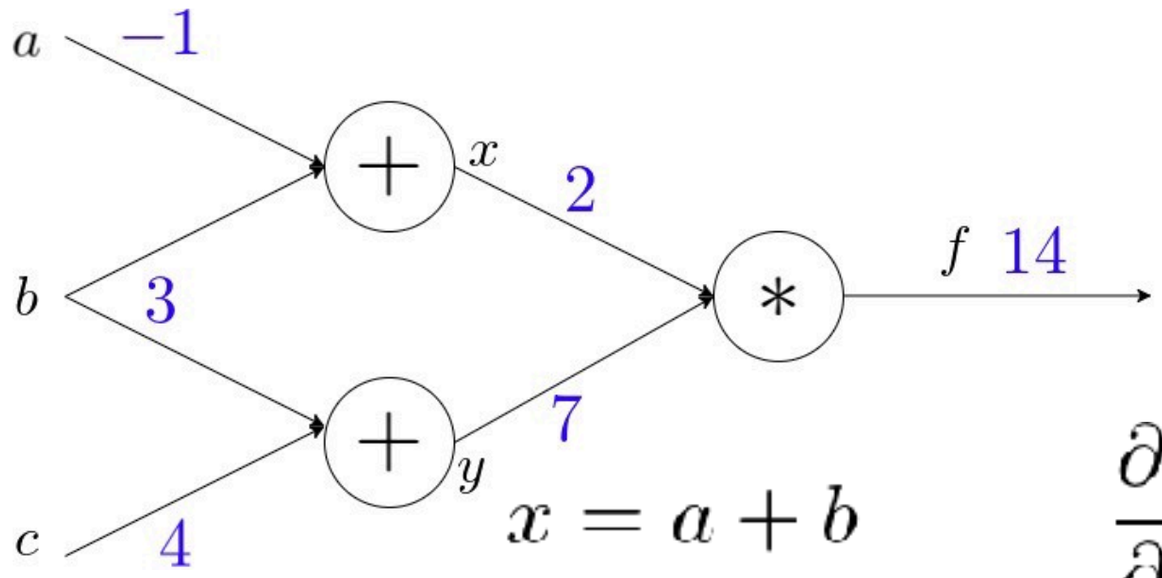

w0 2.00
-0.20
x0 -1.00
0.39
w1 -3.00
-0.39
x1 -2.00
-0.59
w2 -3.00
0.20
*
*
-2.00
0.20
6.00
0.20
+
4.00
0.20
XW^T
b
+
1.00
0.20
*-1
-1.00
-0.20
exp
0.37
-0.53
+1
1.37
-0.53
1/x
0.73
1.00
SIGMOID

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

THE CHAIN RULE



FORWARDS

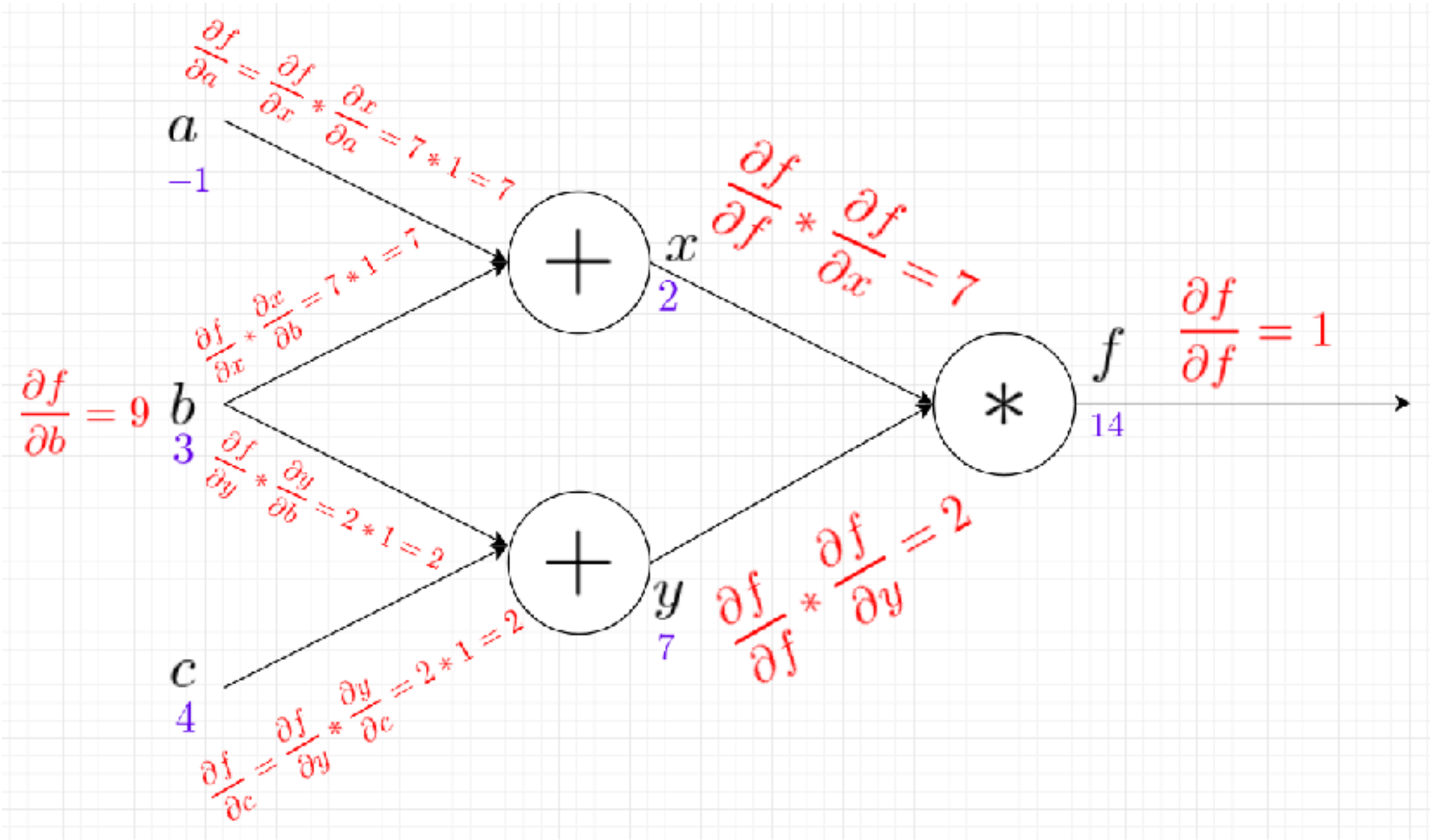


$$x = a + b \qquad \frac{\partial x}{\partial a} = 1 \qquad \frac{\partial x}{\partial b} = 1$$

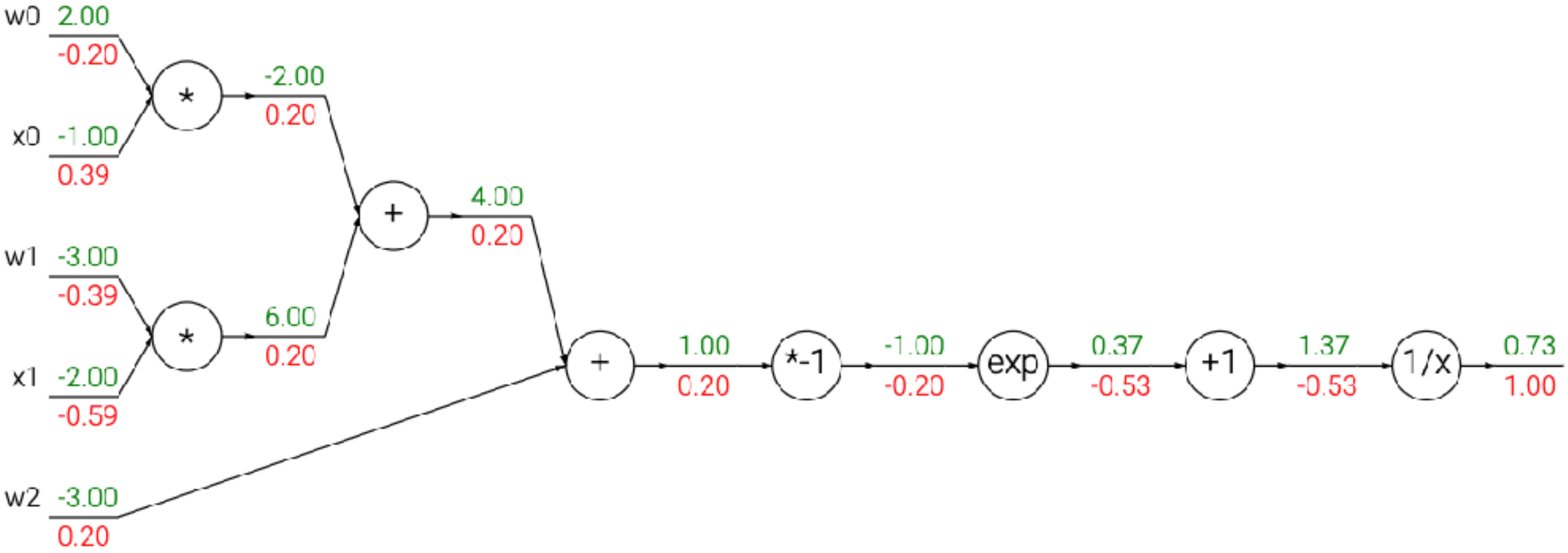
$$y = b + c \qquad \frac{\partial y}{\partial b} = 1 \qquad \frac{\partial y}{\partial c} = 1$$

$$f = x * y \qquad \frac{\partial f}{\partial x} = y \qquad \frac{\partial f}{\partial y} = x$$

GET LOCAL GRADIENTS


a
−1
b
3
c
4
x
2
y
7
f
14
+
+
*
$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial x} * \frac{\partial x}{\partial a} = 7 * 1 = 7$
$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial x} * \frac{\partial x}{\partial b} = 7 * 1 = 7$
$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial y} * \frac{\partial y}{\partial b} = 2 * 1 = 2$
$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial y} * \frac{\partial y}{\partial c} = 2 * 1 = 2$
$\frac{\partial f}{\partial f} * \frac{\partial f}{\partial x} = 7$
$\frac{\partial f}{\partial f} * \frac{\partial f}{\partial y} = 2$
$\frac{\partial f}{\partial f} = 1$
$\frac{\partial f}{\partial b} = 9$

BACKWARDS



NOW, DO IT URSELF :)

APPENDIX

VISUALIZED PLAYGROUND

- https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/playground-exercises?hl=ko

WHY MATRIX REPRESENTATION?

- TC of matmul : ~O(N^2.3xx)
  - https://en.wikipedia.org/wiki/Coppersmith%E2%80%93Winograd_algorithm
- **Massive** parallel executions on GPU
  - CPU : # of cores (~hundreds)
  - NVIDIA GPU : # of CUDA cores (>thousands)

REFERENCES

- http://cs231n.github.io/neural-networks-1/
- http://rasbt.github.io/mlxtend/user_guide/general_concepts/activation-functions/
- https://medium.com/spidernitt/breaking-down-neural-networks-an-intuitive-approach-to-backpropagation-3b2ff958794c
- https://deeplearningzerotoall.github.io/season2/