Neural Network
(Basic Ideas)

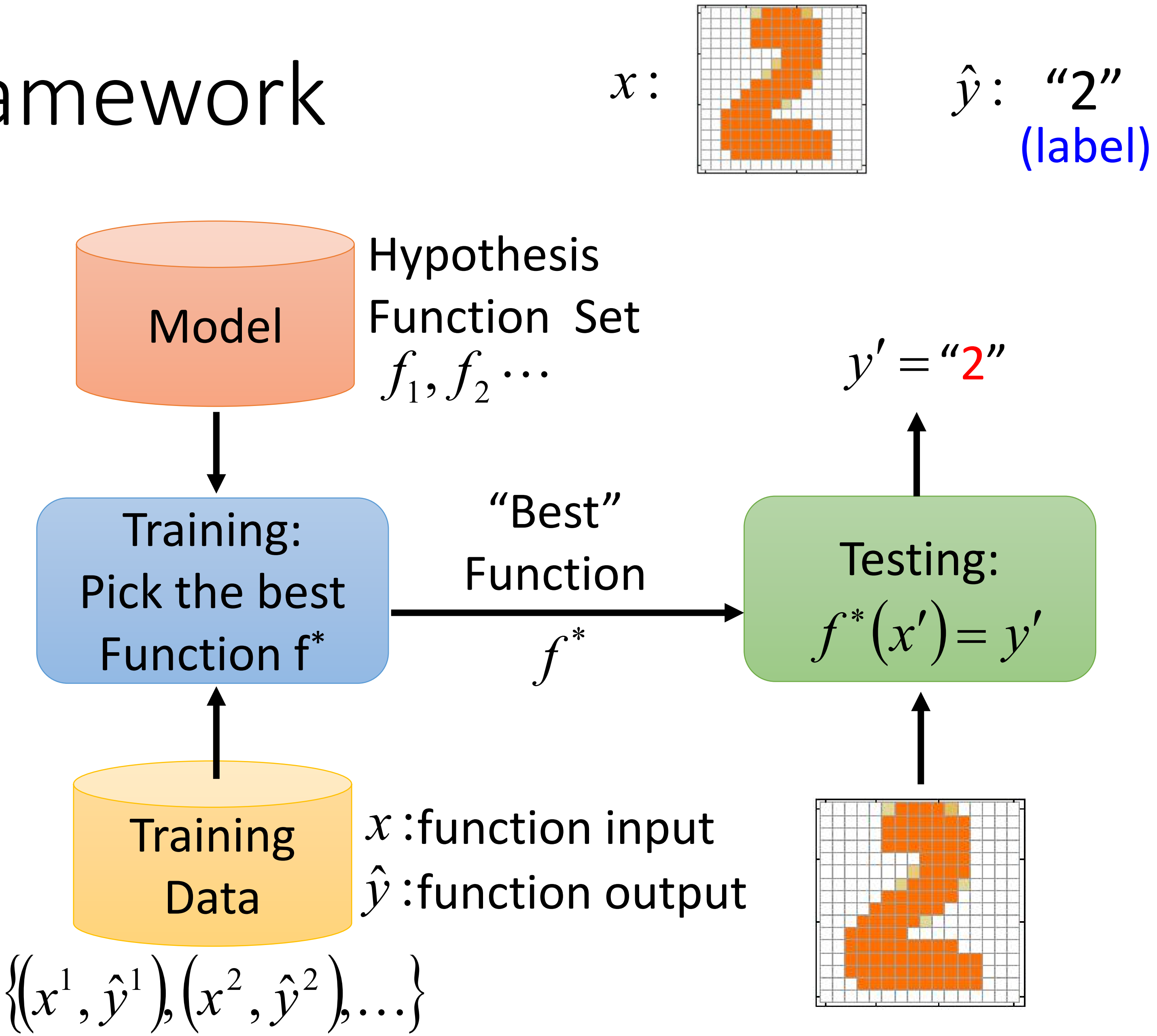
Machine Learning Steps

1. What is the model (function hypothesis set)?

2. What is the “best” function?

3. How to pick the “best” function?

Framework


$x:$
$\hat{y}:$ "2"
(label)
Model
Hypothesis
Function Set
$f_1, f_2 \cdots$
Training:
Pick the best
Function f*
"Best"
Function
f^*
Testing:
$f^*(x') = y'$
$y' =$ "2"
Training
Data
x :function input
$\hat{y}$:function output
$\{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \ldots\}$

Step 1: What is the function we are looking for?

- ***classification***

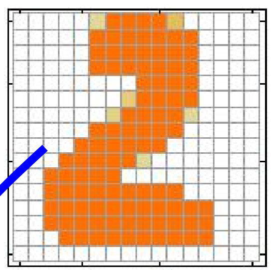
$$y = f(x) \quad \Rightarrow \quad f: R^N \to R^M$$

  - x: input object to be classified
  - y: class
  - ***Assume both x and y can be represented as fixed-size vector***
    - x is a vector with N dimensions, and y is a vector with M dimensions

Step 1: What is the function we are looking for?

- ***Handwriting Digit Classification*** $f: R^N \rightarrow R^M$

x: image



16 x 16

Each pixel corresponds to an element in the vector

$$\begin{bmatrix} 0 \\ 1 \\ \vdots \end{bmatrix}$$

1: for ink,
0: otherwise

16 x 16 = 256 dimensions

y: class

10 dimensions for digit recognition

“1”

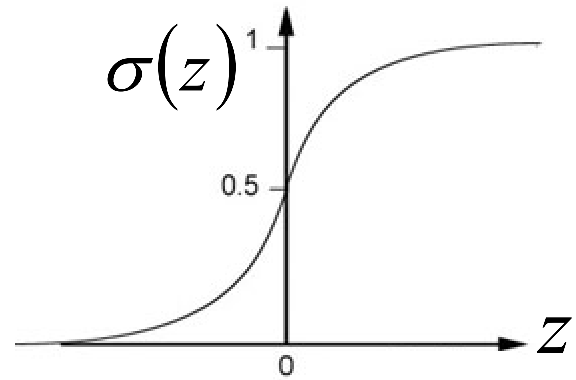
$$\begin{array}{c l} \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix} & \begin{array}{l} \text{“1”} \\ \text{“2”} \\ \text{“3”} \\ \end{array} \end{array}$$

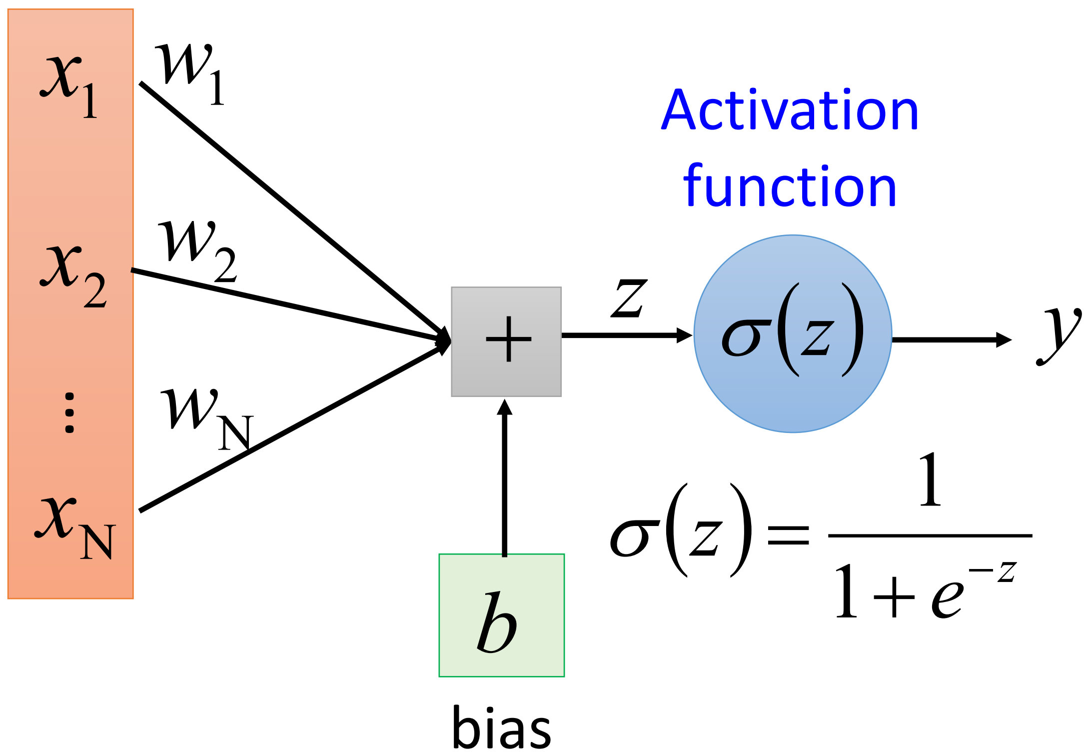
“2”

$$\begin{array}{c l} \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix} & \begin{array}{l} \text{“1”} \rightarrow \text{“1” or not} \\ \text{“2”} \rightarrow \text{“2” or not} \\ \text{“3”} \rightarrow \text{“3” or not} \\ \end{array} \end{array}$$

Single Neuron

$$f: R^N \to R$$

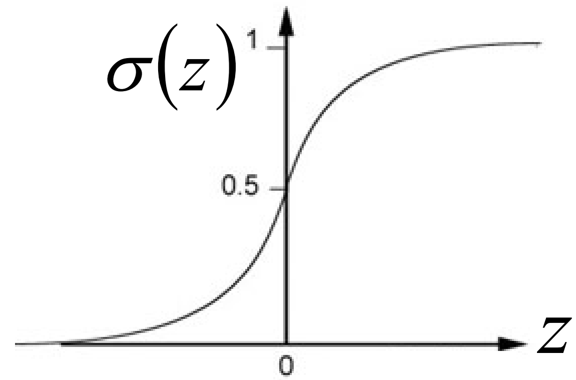


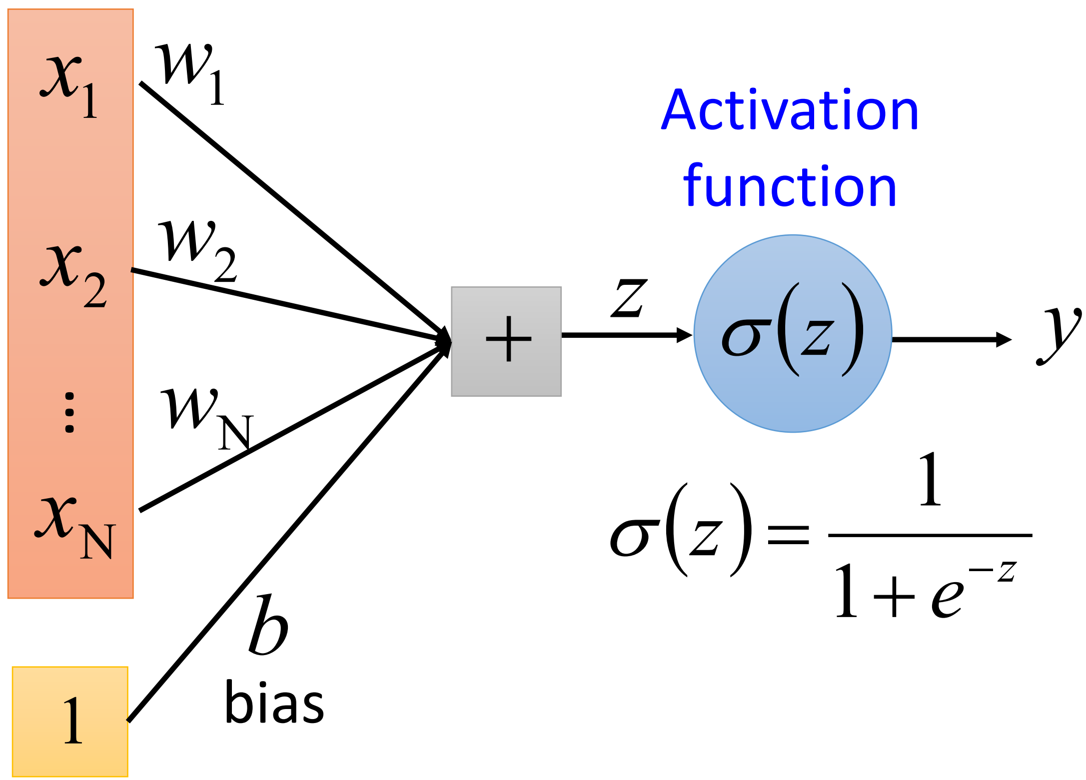


$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Single Neuron

$$f: R^N \rightarrow R$$

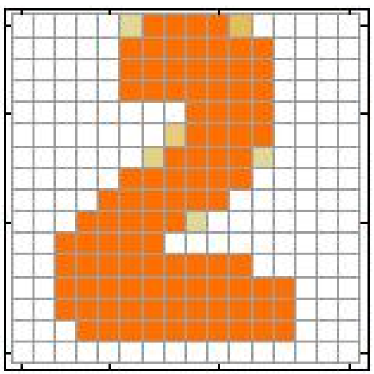


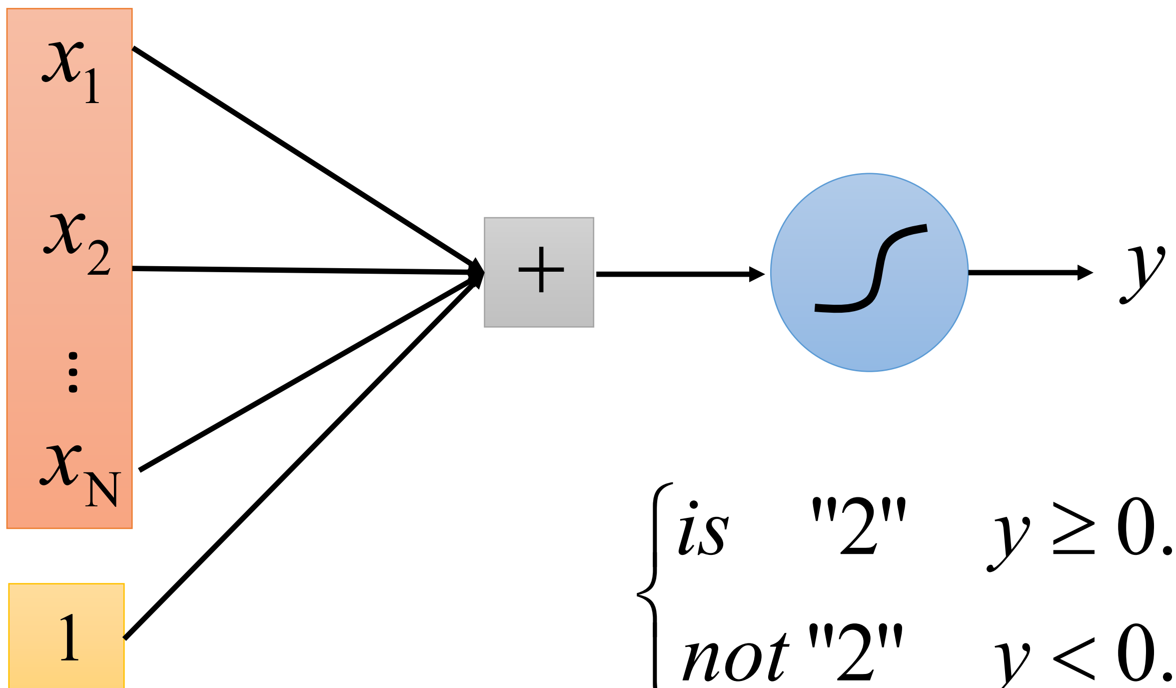


$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Single Neuron $f: R^N \to R$

- Single neuron can only do binary classification, cannot handle multi-class classification



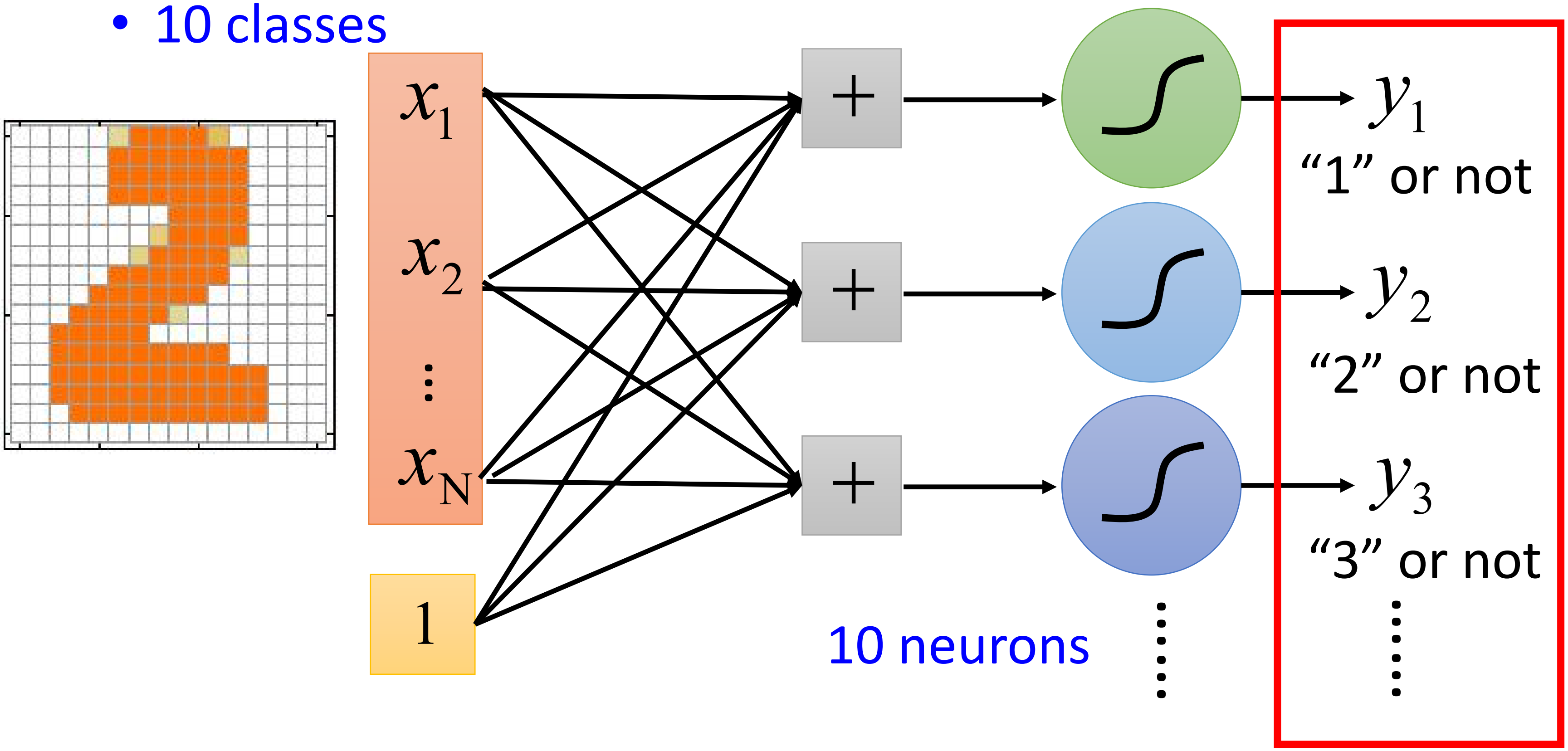


$$\begin{cases} is \quad "2" & y \geq 0.5 \\ not \; "2" & y < 0.5 \end{cases}$$

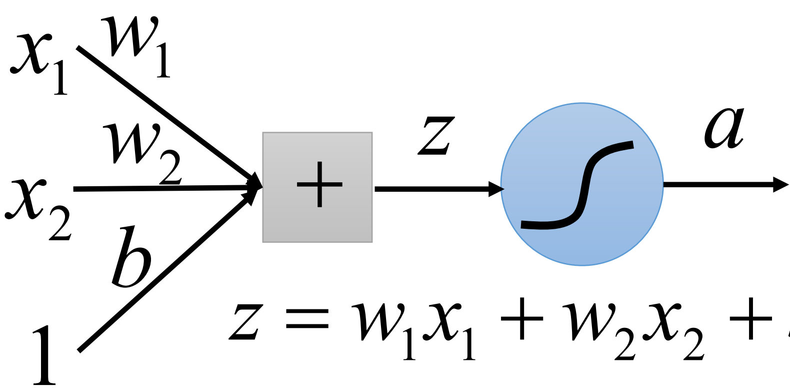
A Layer of Neuron $f: R^N \rightarrow R^M$

- Handwriting digit classification
  - Classes: “1”, “2”, …., “9”, “0”
  - 10 classes

If y_2 is the max, then the image is “2”.



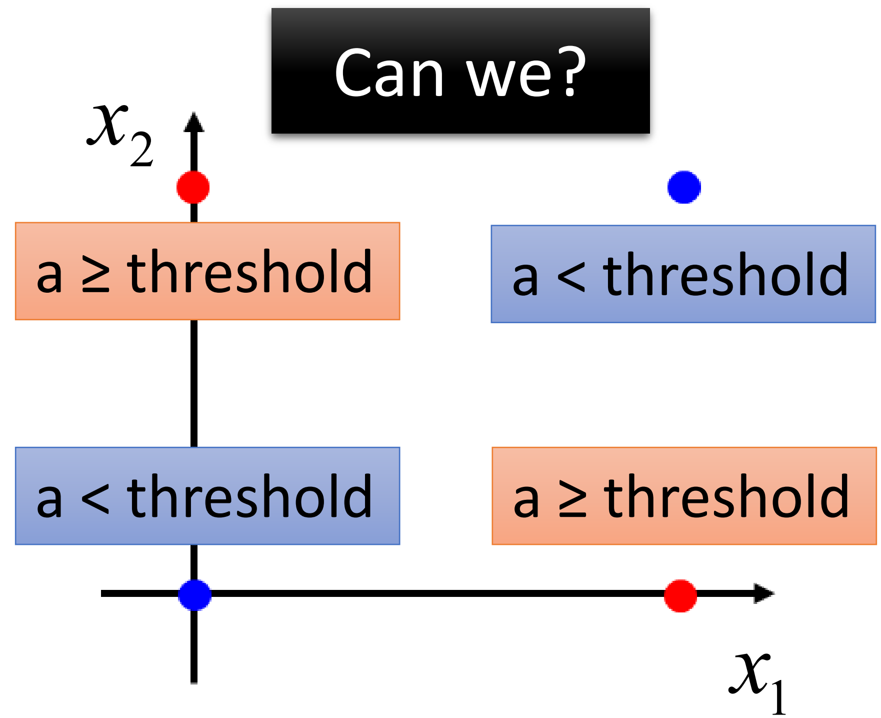
Limitation of Single Layer



$$z = w_1 x_1 + w_2 x_2 + b$$

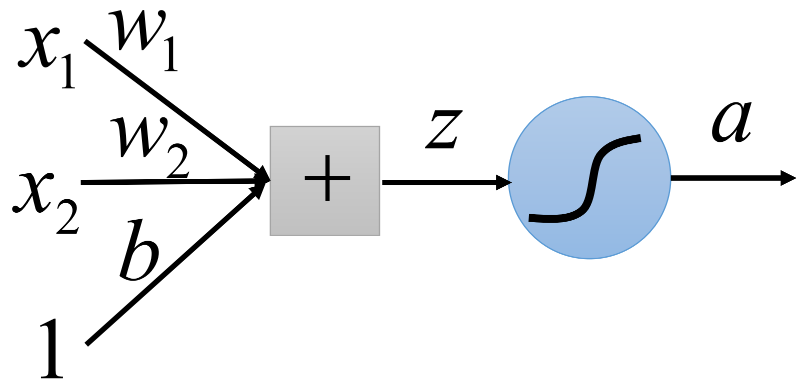
$$\begin{cases} yes & a \geq threshold \\ no & a < threshold \end{cases}$$

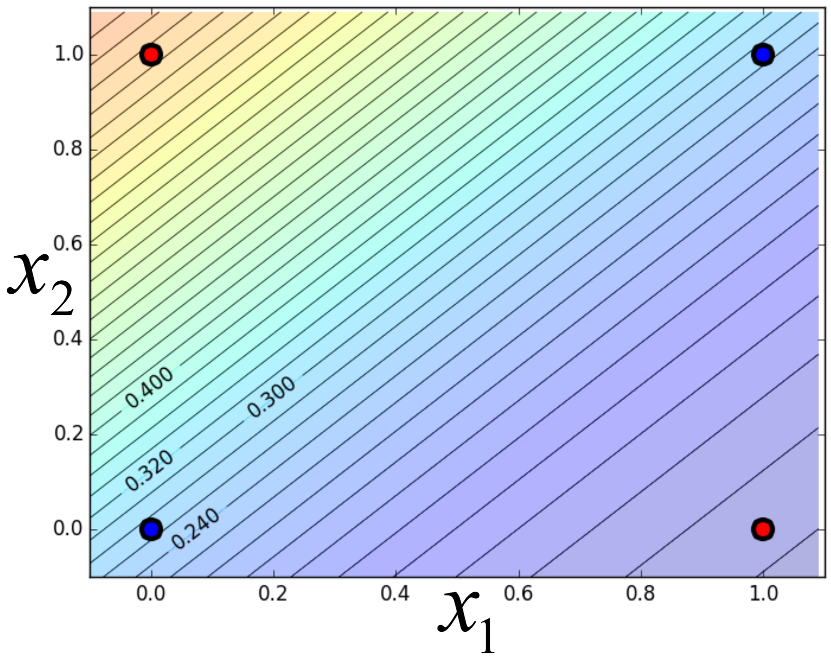
Input		Output
x_1	x_2	
0	0	No
0	1	Yes
1	0	Yes
1	1	No

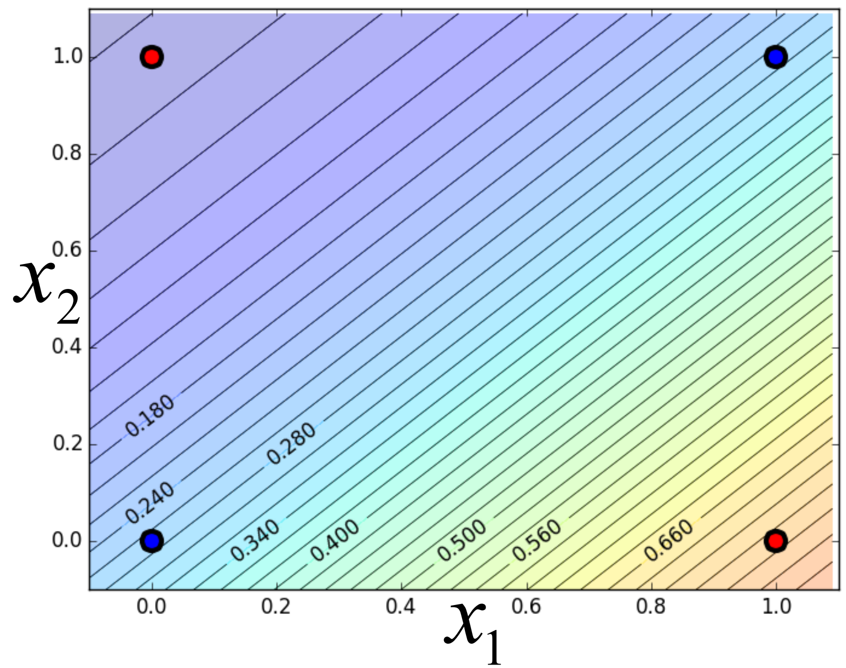


Limitation of Single Layer

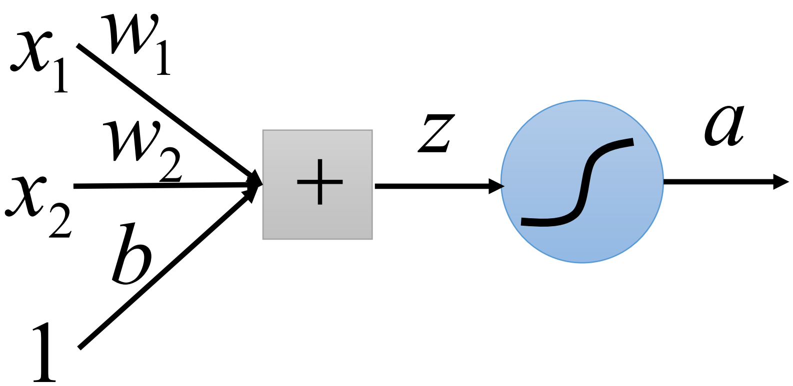
- No, we can't ……



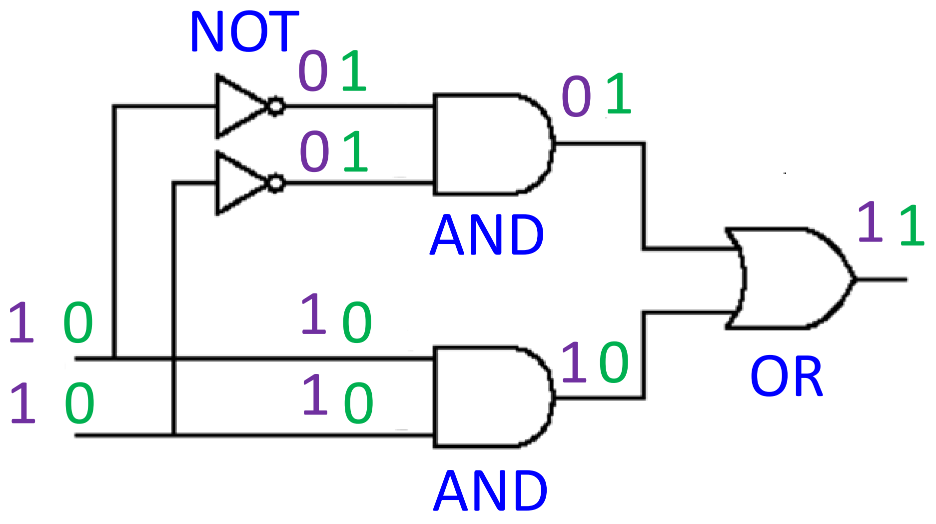


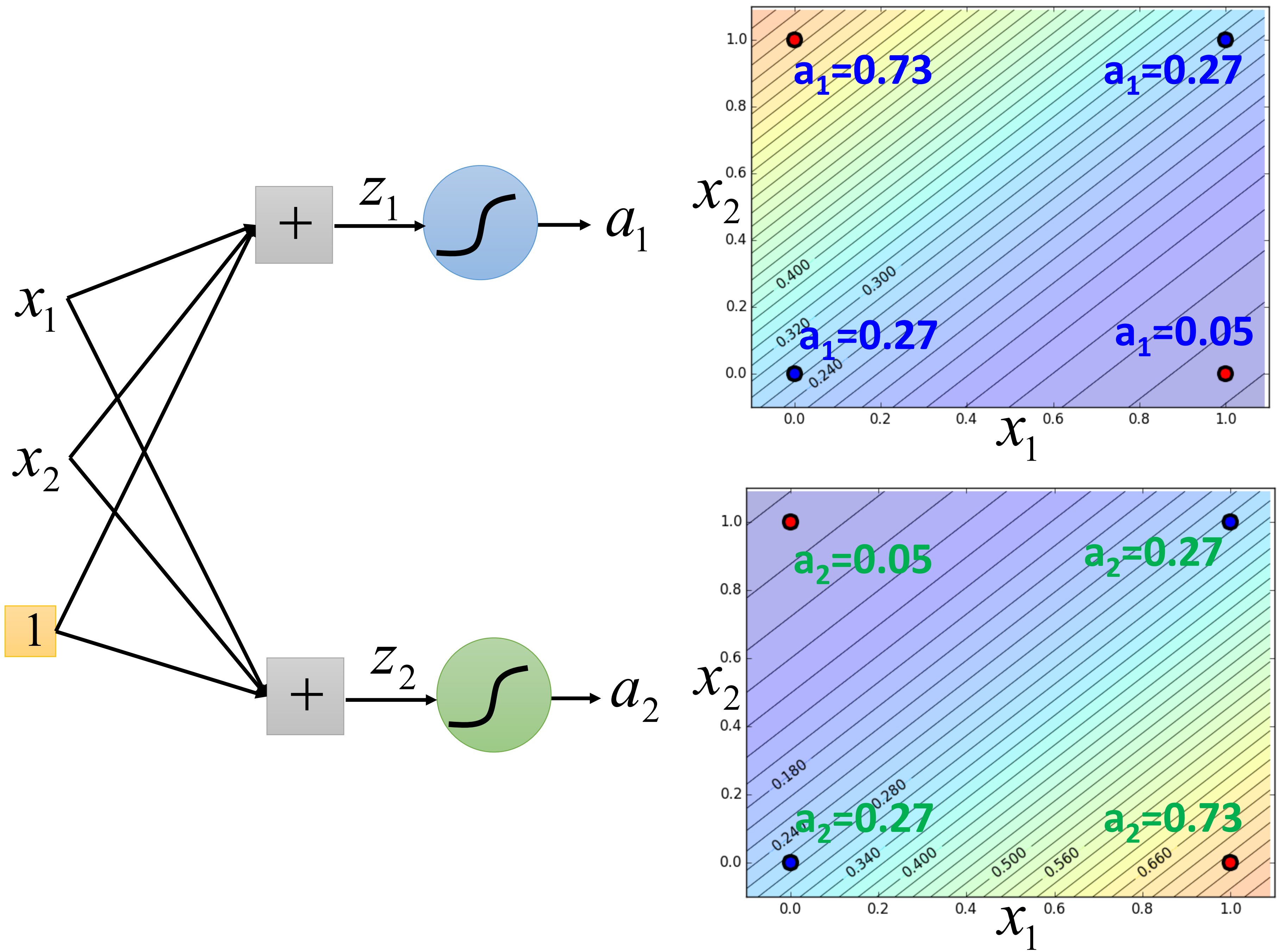


Limitation of Single Layer

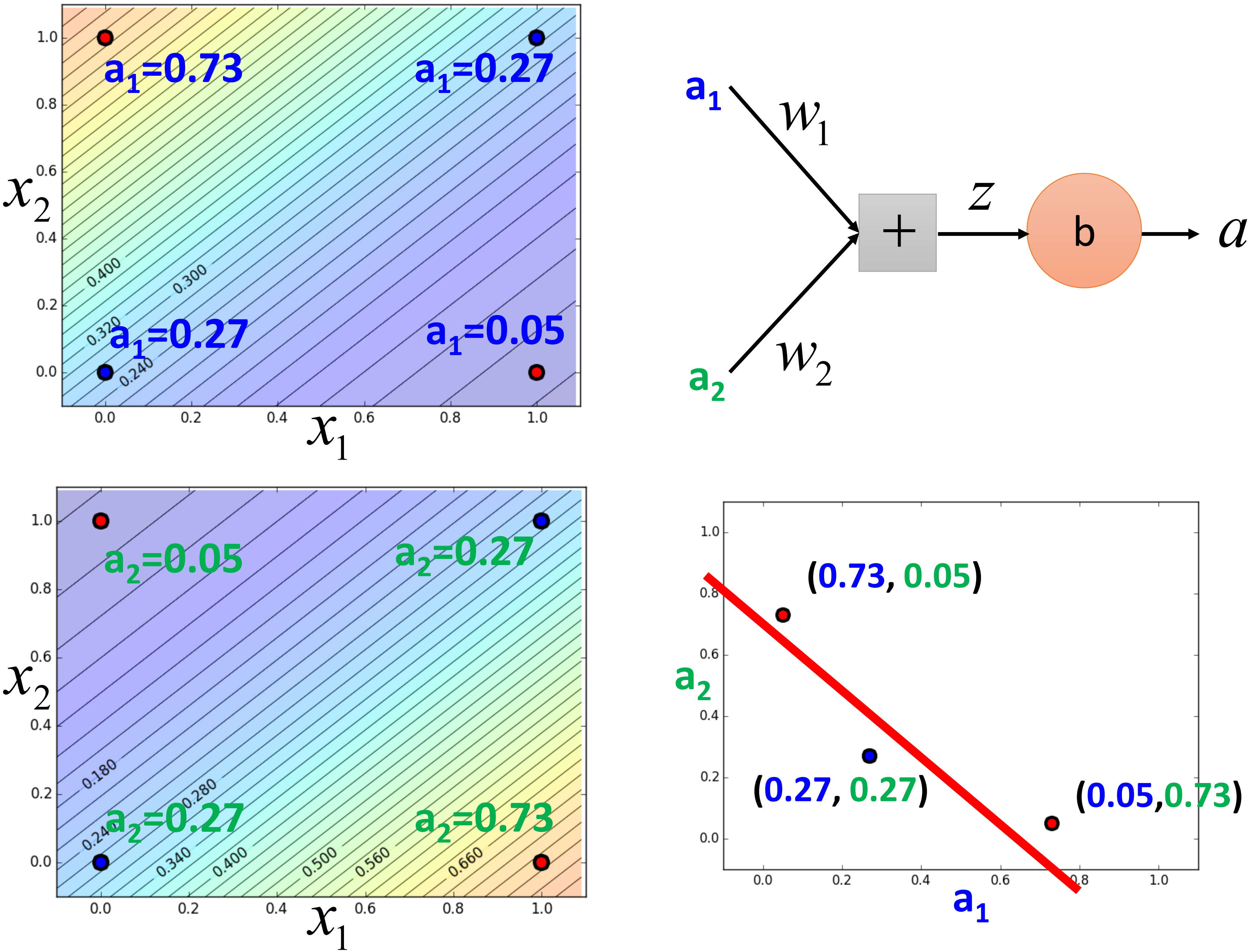


Input		Output
x_1	x_2	
0	0	No
0	1	Yes
1	0	Yes
1	1	No



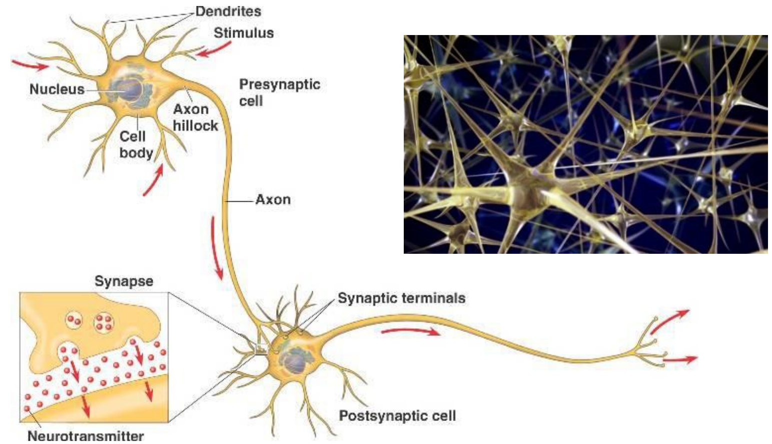


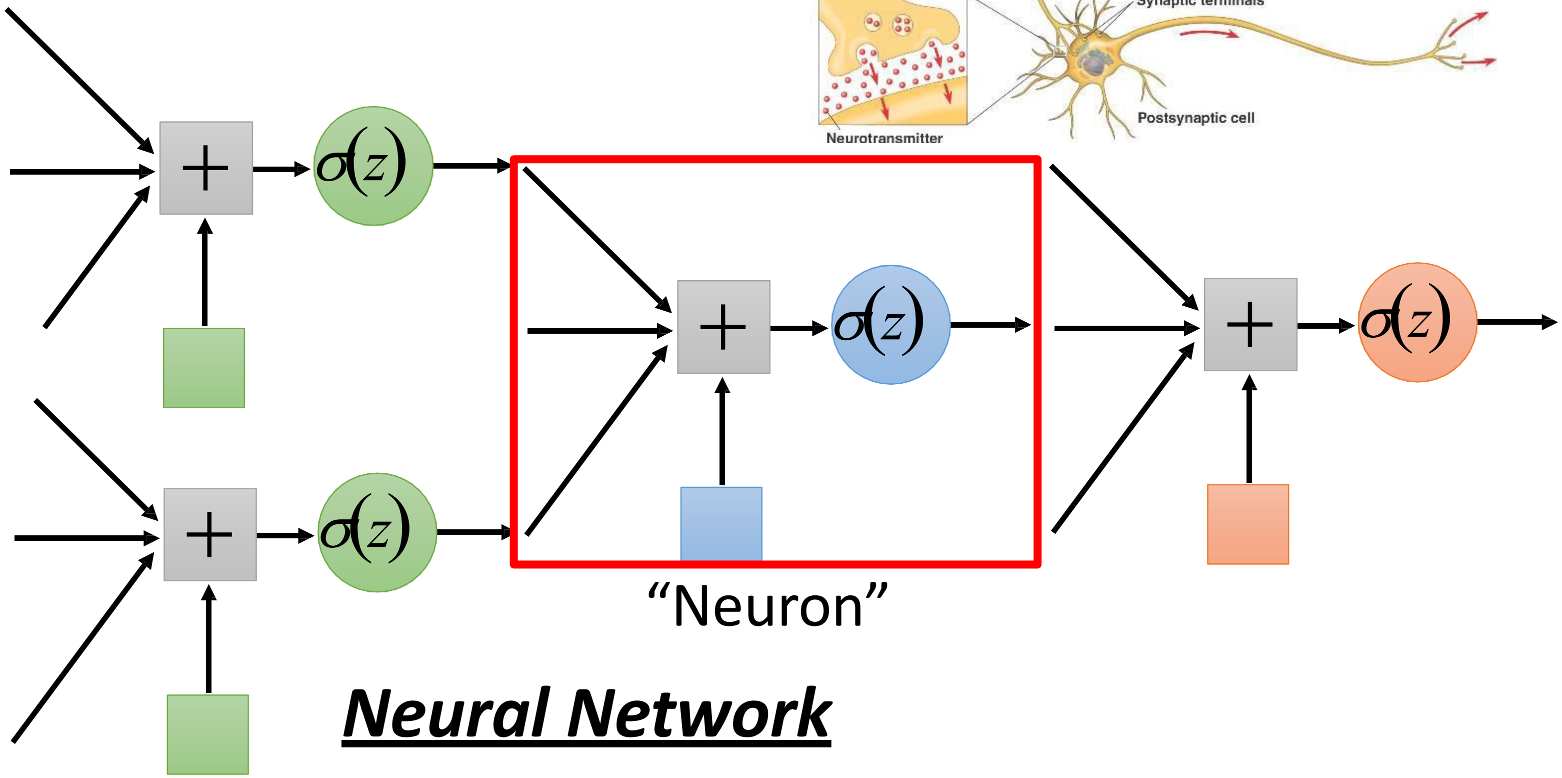
x_1
x_2
1
z_1
a_1
z_2
a_2
a_1=0.73
a_1=0.27
a_1=0.27
a_1=0.05
0.400
0.300
0.320
0.240
a_2=0.05
a_2=0.27
a_2=0.27
a_2=0.73
0.180
0.280
0.340
0.400
0.500
0.560
0.660
x_1
x_2



a_1=0.73
a_1=0.27
a_1=0.27
a_1=0.05
x_1
x_2
a_1
a_2
w_1
w_2
+
z
b
a
a_2=0.05
a_2=0.27
a_2=0.27
a_2=0.73
(0.73, 0.05)
(0.27, 0.27)
(0.05,0.73)

Neural Network



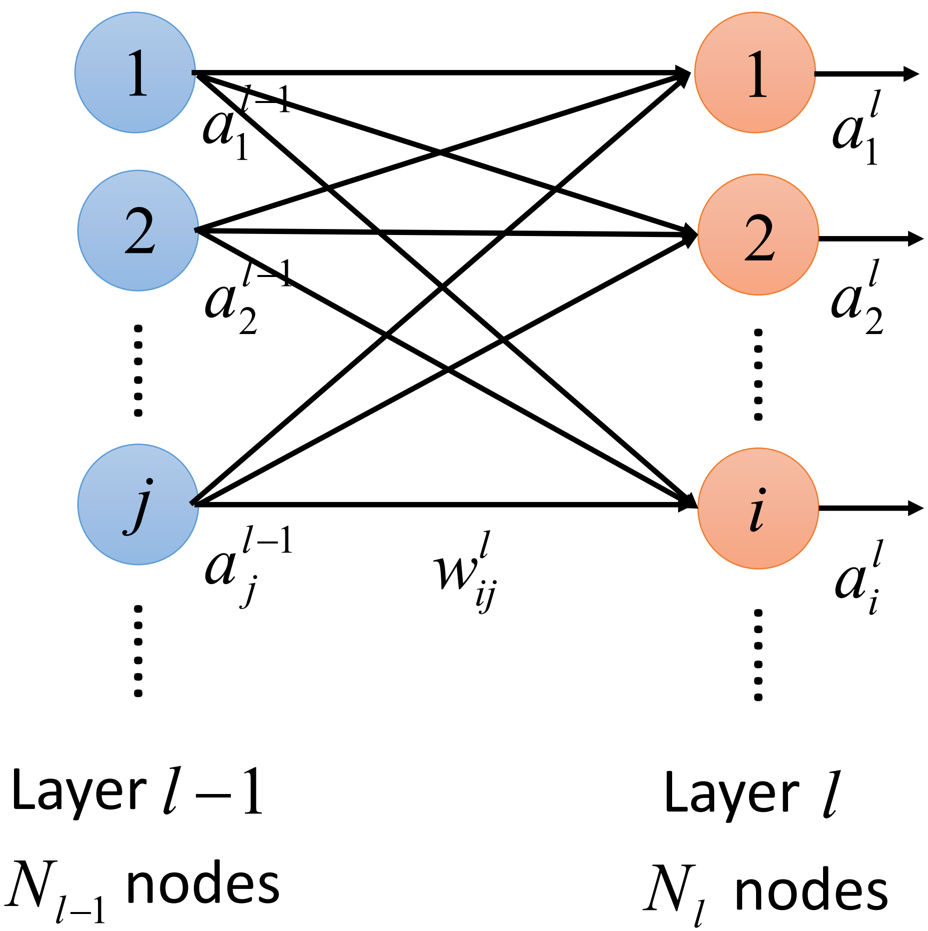


<u>Neural Network</u>

Different connection leads to different network structures

Network parameter θ: all the weights and biases in the "neurons"

Notation



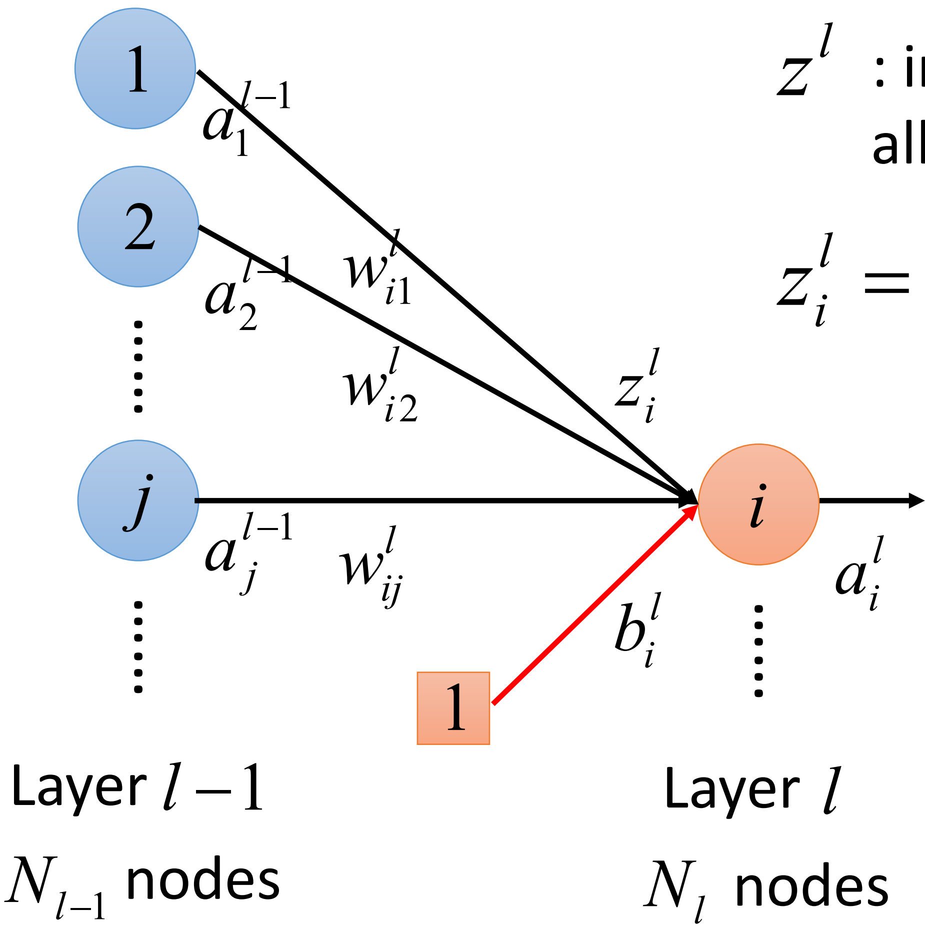
w_{ij}^l

Layer $l-1$ to Layer l

from neuron j (Layer $l-1$)
to neuron i (Layer l)

$$
W^l = \begin{array}{cc} \overbrace{\phantom{w_{11}^l \quad w_{12}^l \quad \cdots \quad}}^{\mathrm{N}_{l-1}} & \\ \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \\ & & \end{bmatrix} & \left.\vphantom{\begin{bmatrix} w \\ w \\ \vdots \\ \, \end{bmatrix}}\right\} \mathrm{N}_l \end{array}
$$

Notation



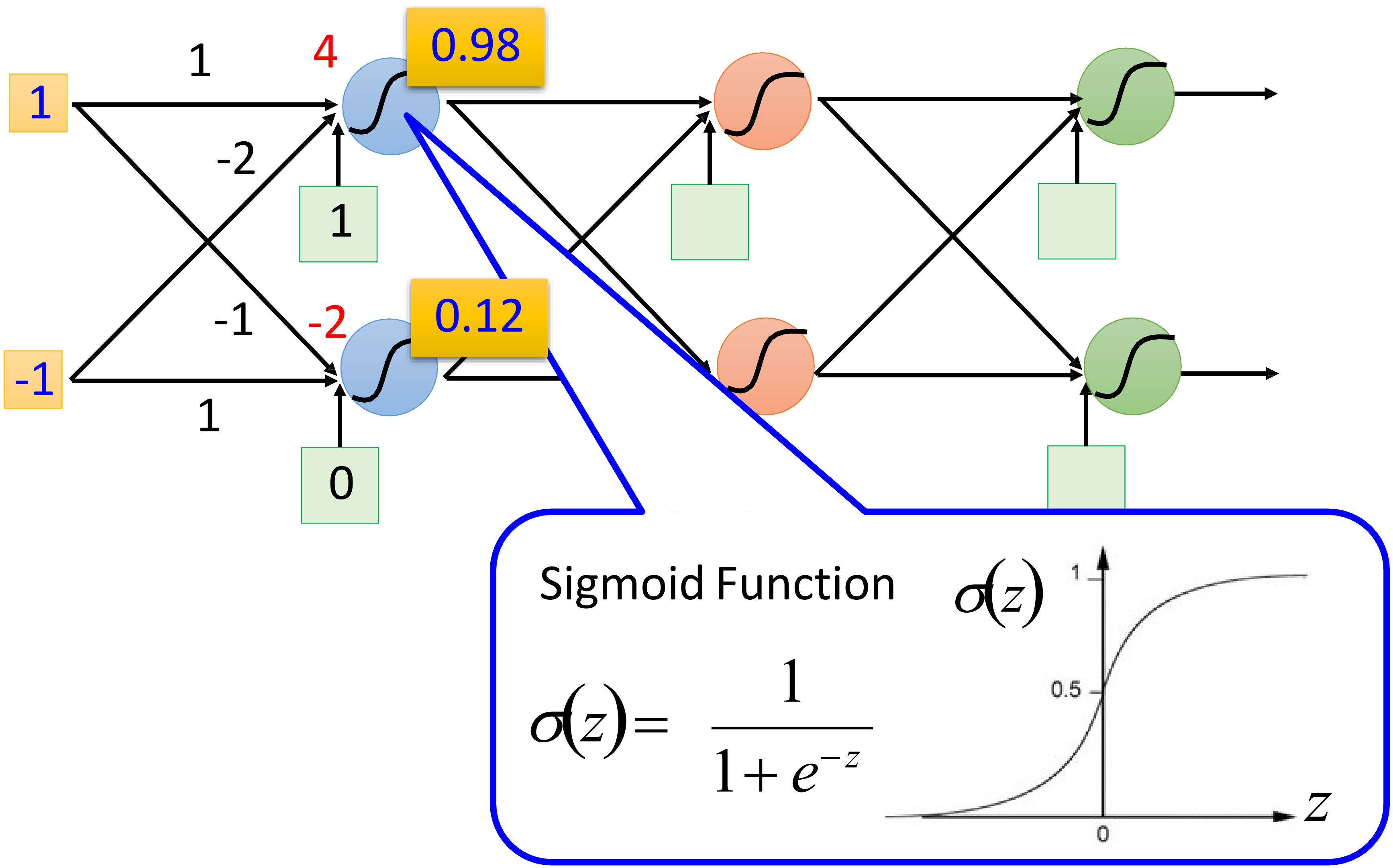
z_i^l : input of the activation function for neuron i at layer l

z^l : input of the activation function all the neurons in layer l

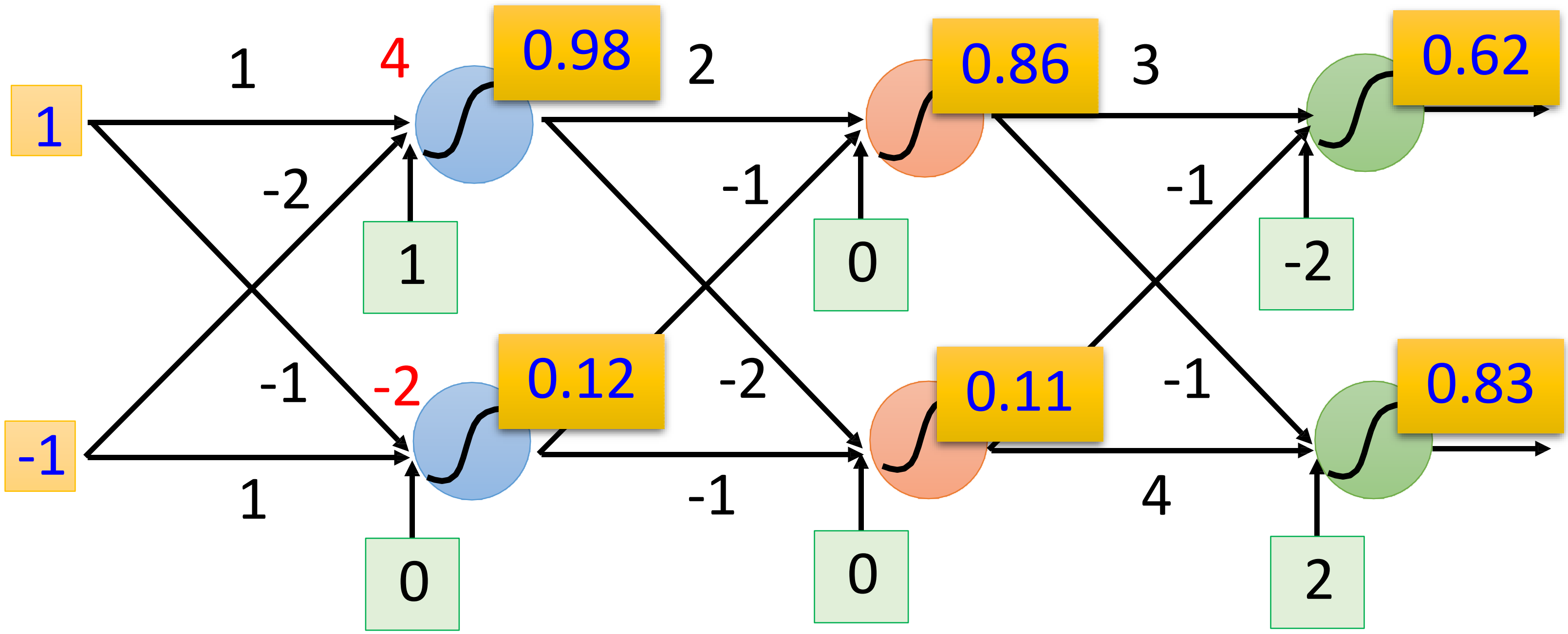
$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} \ldots + b_i^l$$

$$z_i^l = \sum_{j=1}^{N_{l-1}} w_{ij}^l a_j^{l-1} + b_i^l$$

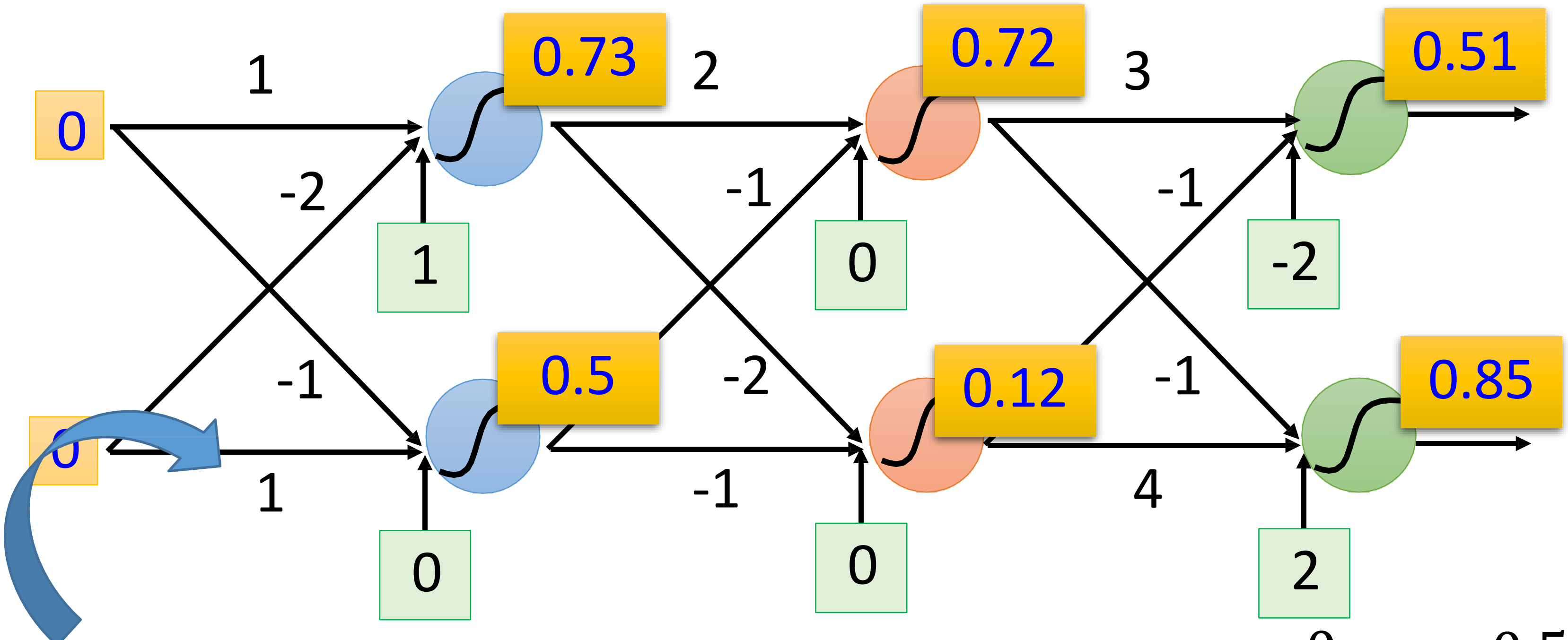
Fully Connect Feedforward Network



Fully Connect Feedforward Network



Fully Connect Feedforward NN



This is a function.

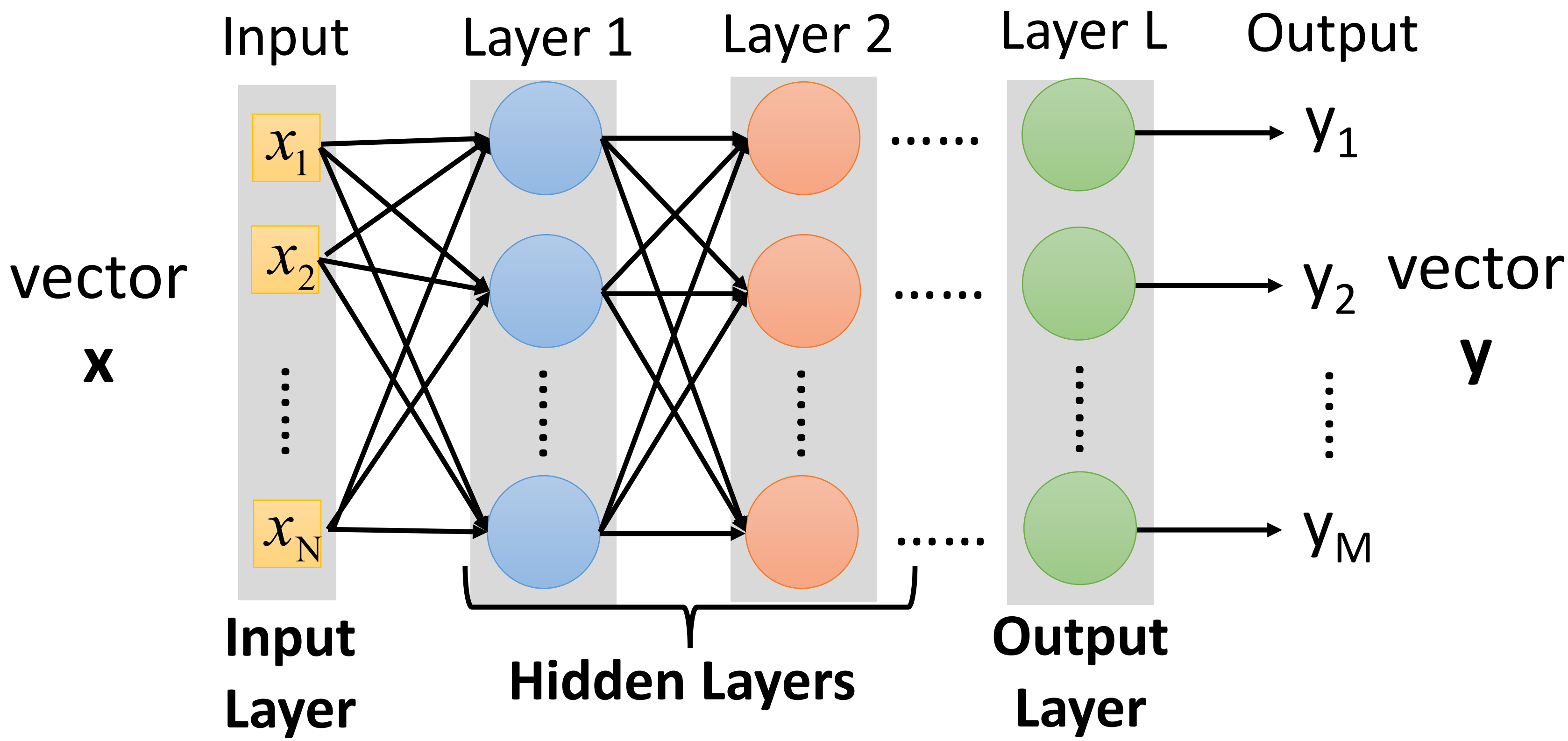
Input vector, output vector

$$f\left(\begin{bmatrix}1\\-1\end{bmatrix}\right)=\begin{bmatrix}0.62\\0.83\end{bmatrix} \quad f\left(\begin{bmatrix}0\\0\end{bmatrix}\right)=\begin{bmatrix}0.51\\0.85\end{bmatrix}$$

Given network structure, define ***a function set***

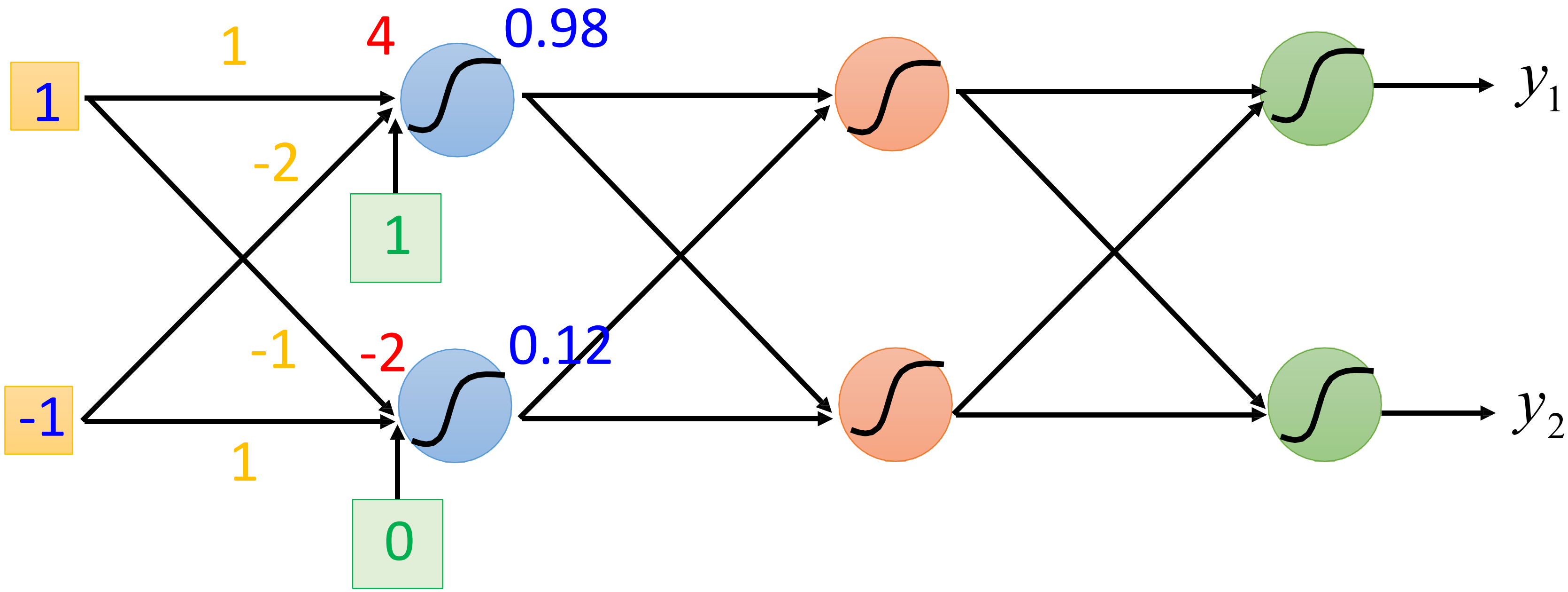
Neural Network as Model

$$f: R^N \to R^M$$



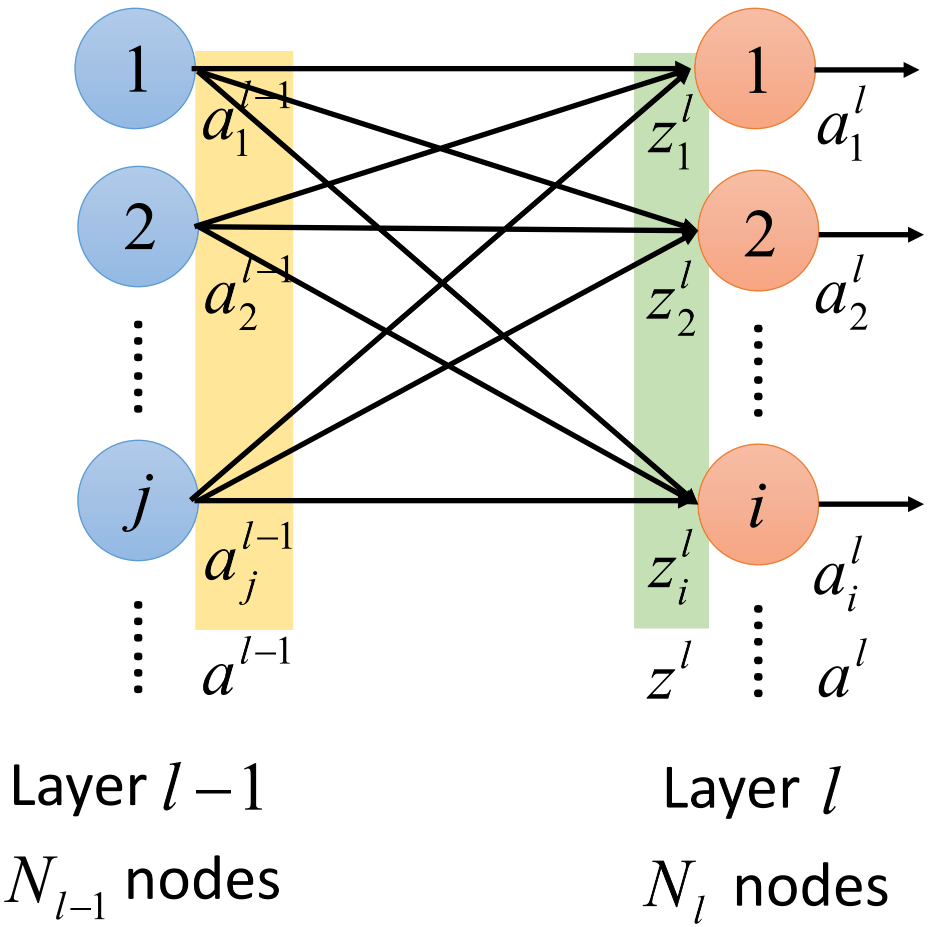
- Fully connected feedforward network
- Deep Neural Network: many hidden layers

Matrix Operation



$$\sigma\left(\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}}\right) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Relations between Layer Outputs



$$z_1^l = w_{11}^l a_1^{l-1} + w_{12}^l a_2^{l-1} + \cdots + b_1^l$$

$$z_2^l = w_{21}^l a_1^{l-1} + w_{22}^l a_2^{l-1} + \cdots + b_2^l$$

$$\vdots$$

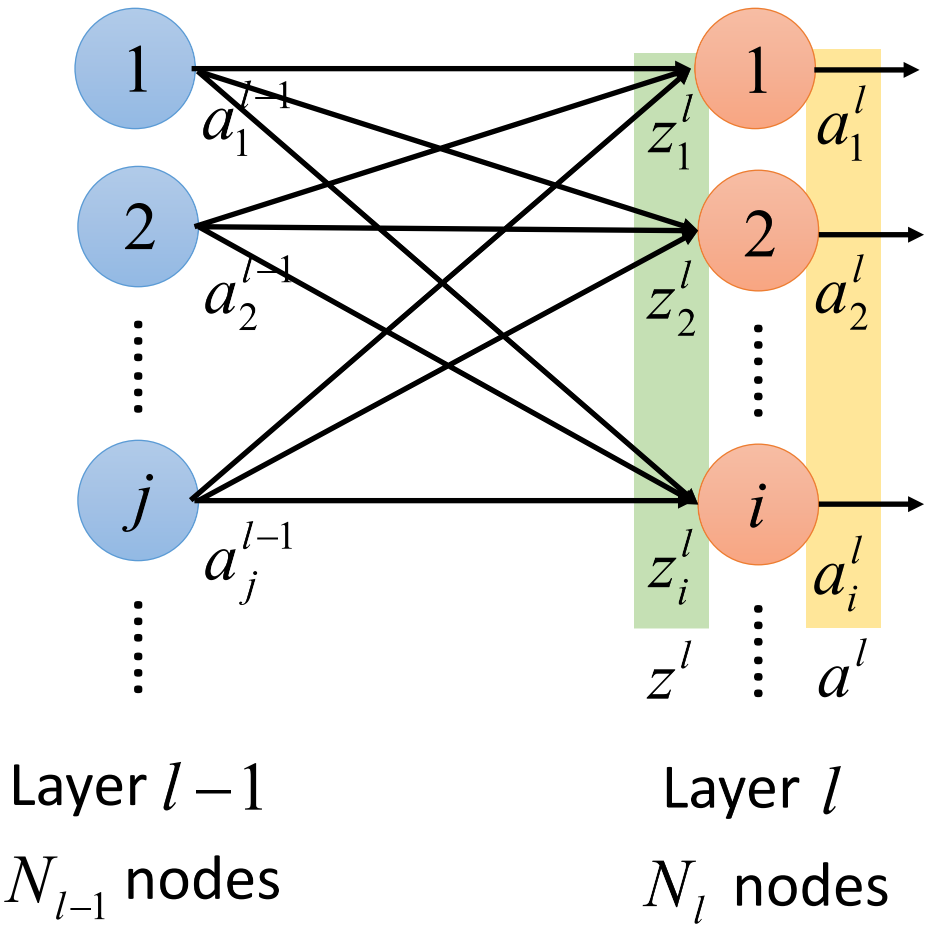
$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} + \cdots + b_i^l$$

$$\vdots$$

$$\begin{bmatrix} z_1^l \\ z_2^l \\ \vdots \\ z_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \end{bmatrix} \begin{bmatrix} a_1^{l-1} \\ a_2^{l-1} \\ \vdots \\ a_i^{l-1} \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

$$z^l = W^l a^{l-1} + b^l$$

Relations between Layer Outputs

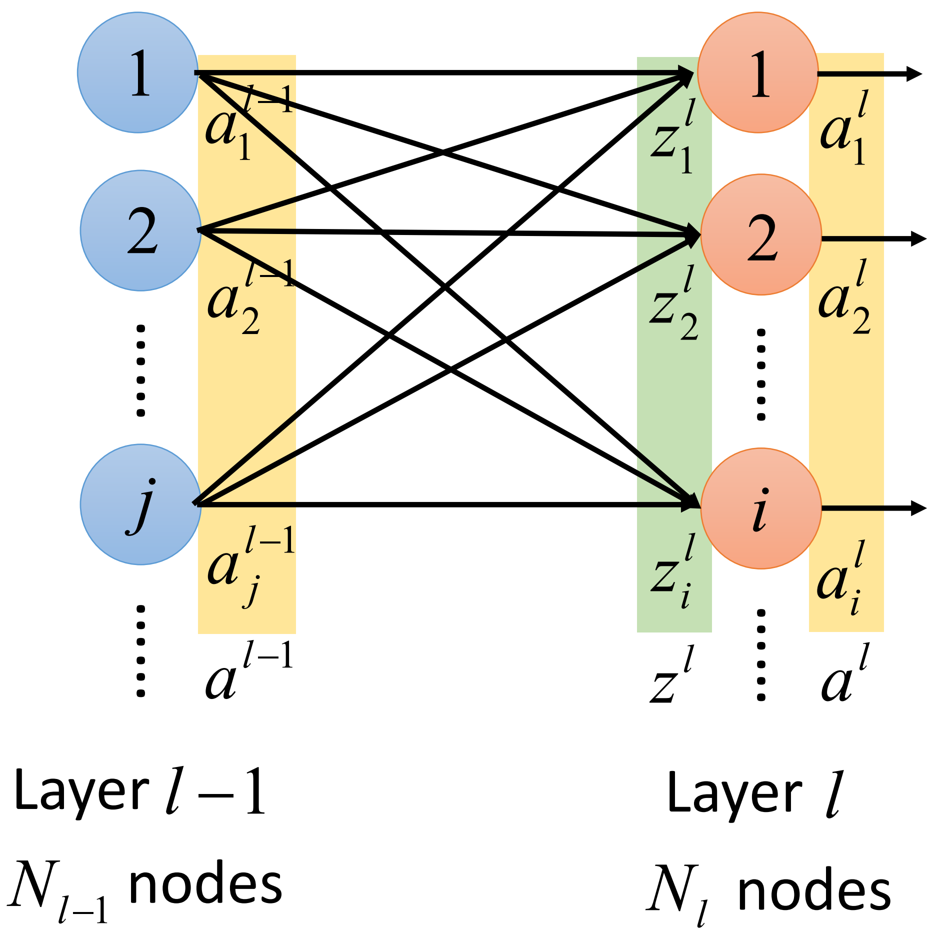


$$a_i^l = \sigma\left(z_i^l\right)$$

$$\begin{bmatrix} a_1^l \\ a_2^l \\ \vdots \\ a_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} \sigma\left(z_1^l\right) \\ \sigma\left(z_2^l\right) \\ \vdots \\ \sigma\left(z_i^l\right) \\ \vdots \end{bmatrix}$$

$$a^l = \sigma\left(z^l\right)$$

Relations between Layer Outputs

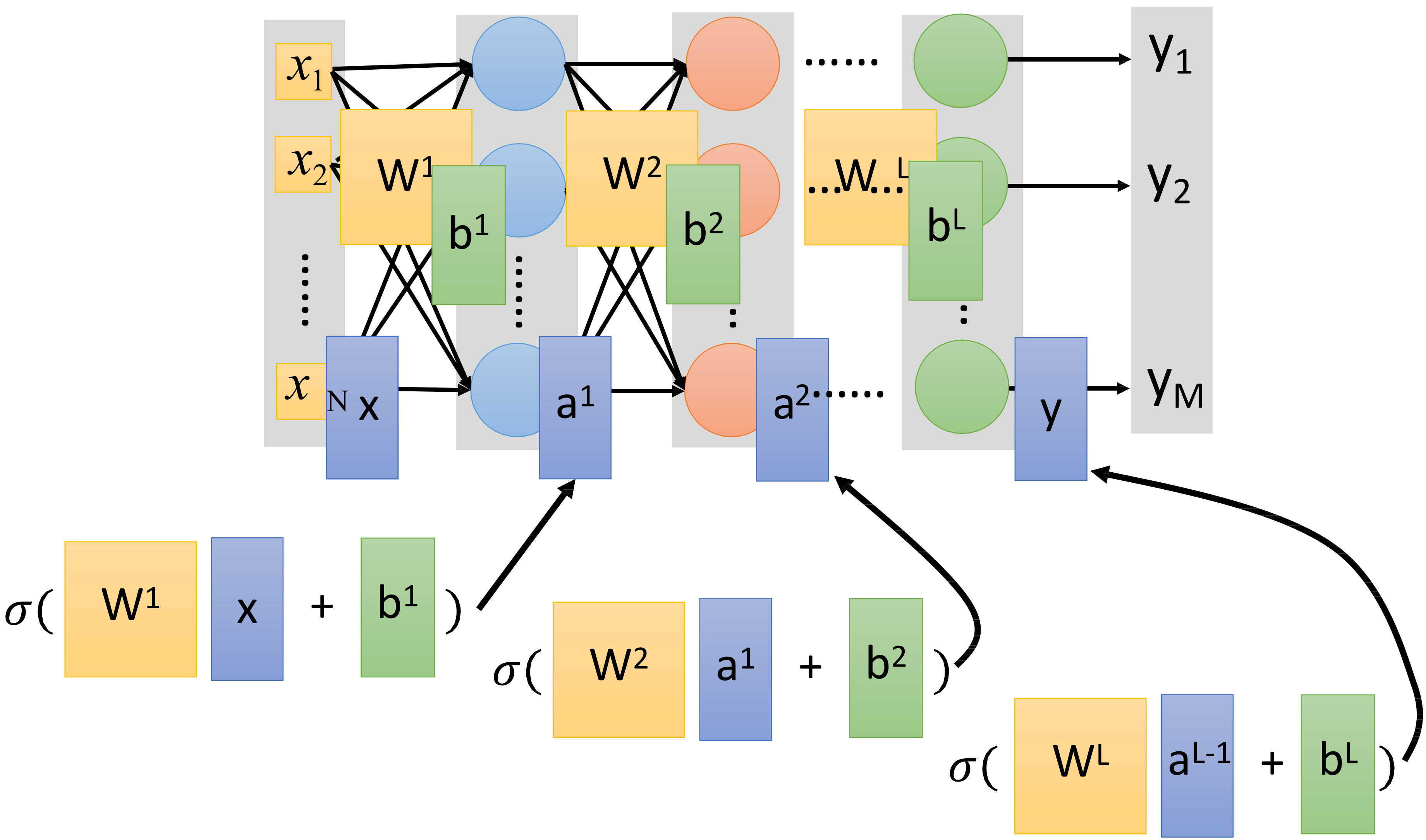


$$z^l = W^l a^{l-1} + b^l$$

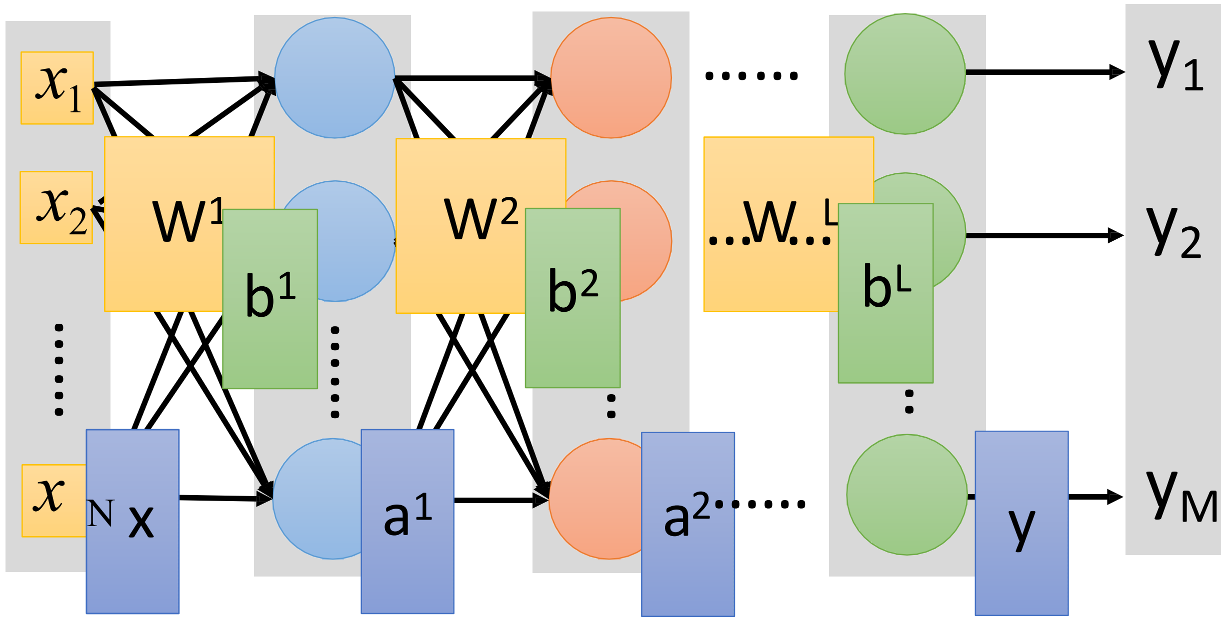
$$a^l = \sigma\left(z^l\right)$$

$$a^l = \sigma\left(W^l a^{l-1} + b^l\right)$$

Neural Network



Neural Network

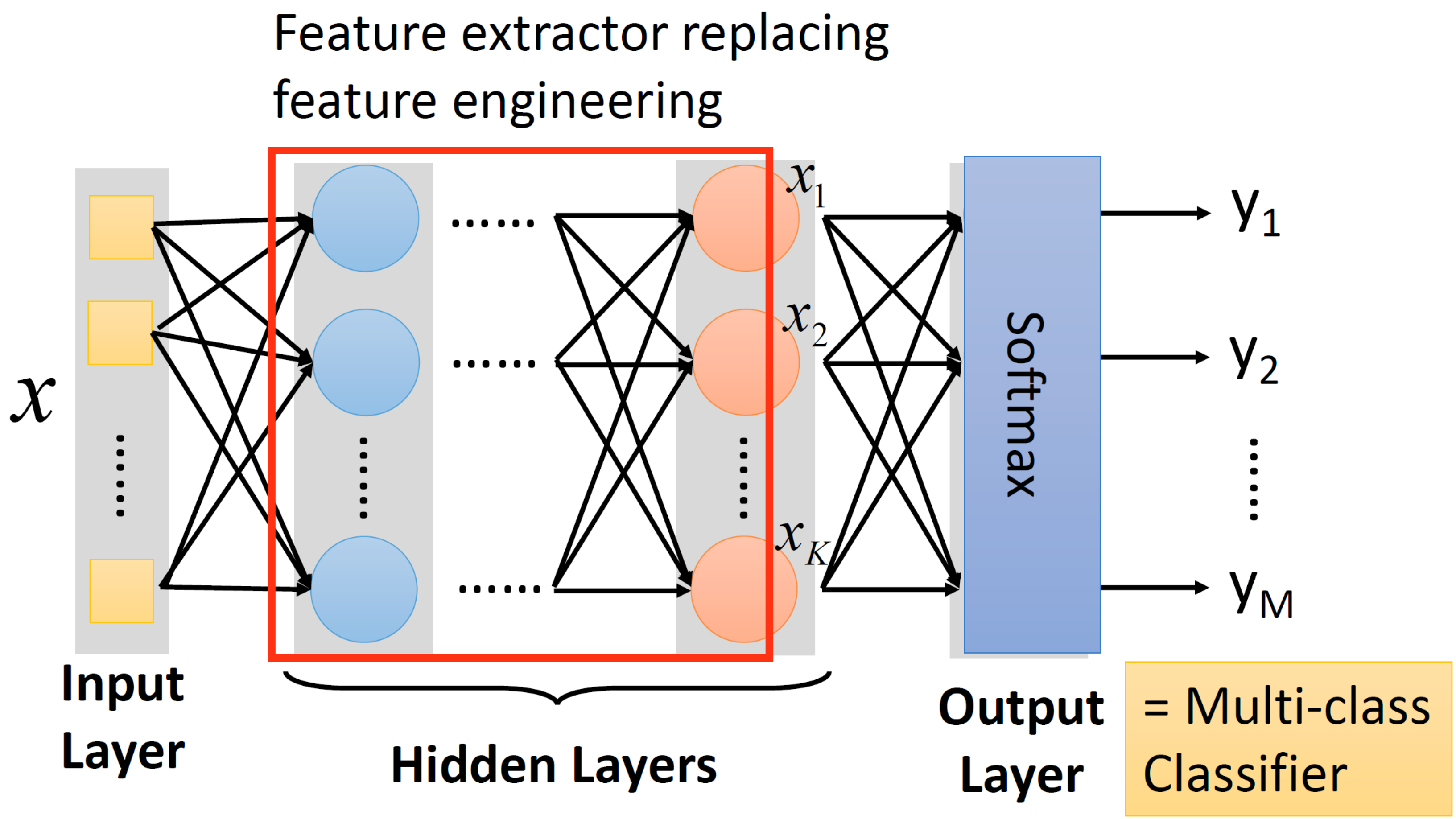


$$y = f(x)$$

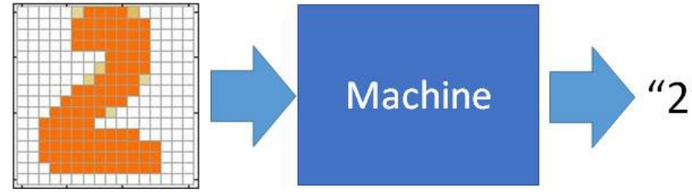
Using parallel computing techniques to speed up matrix operation

$$= \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

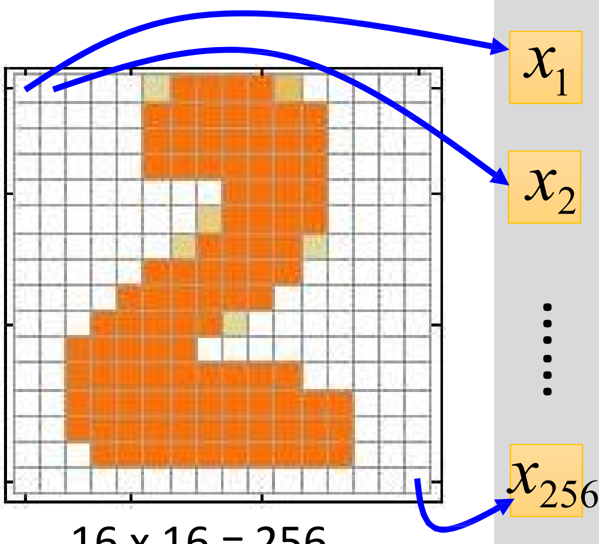
Output Layer



Example Application



Input

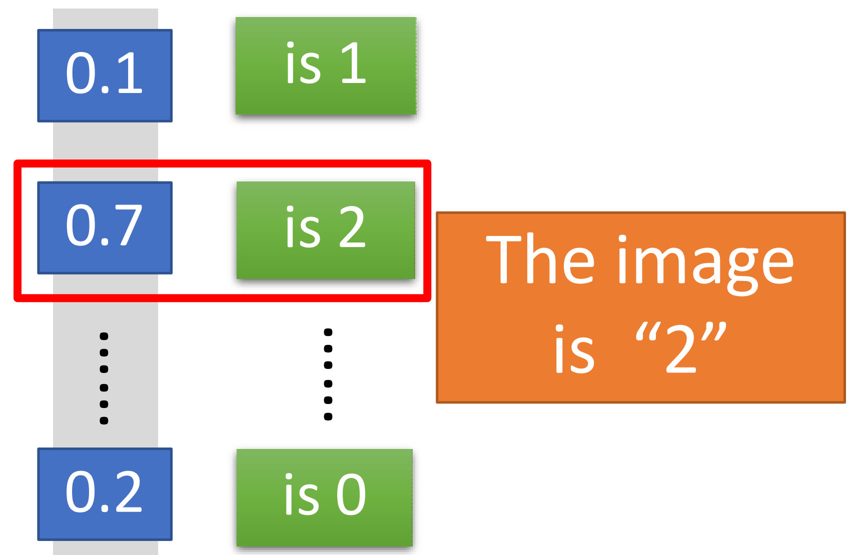


16 x 16 = 256

Ink → 1

No ink → 0

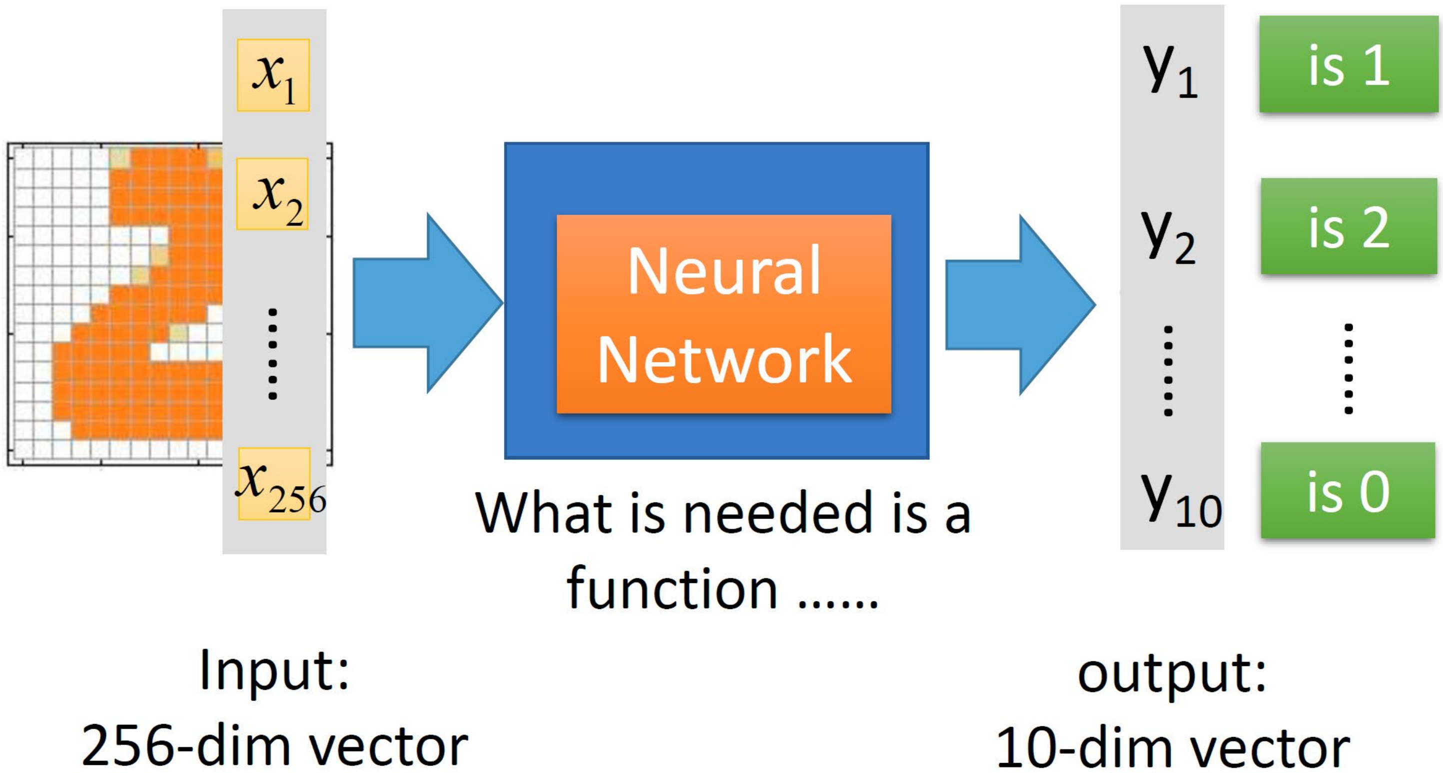
Output



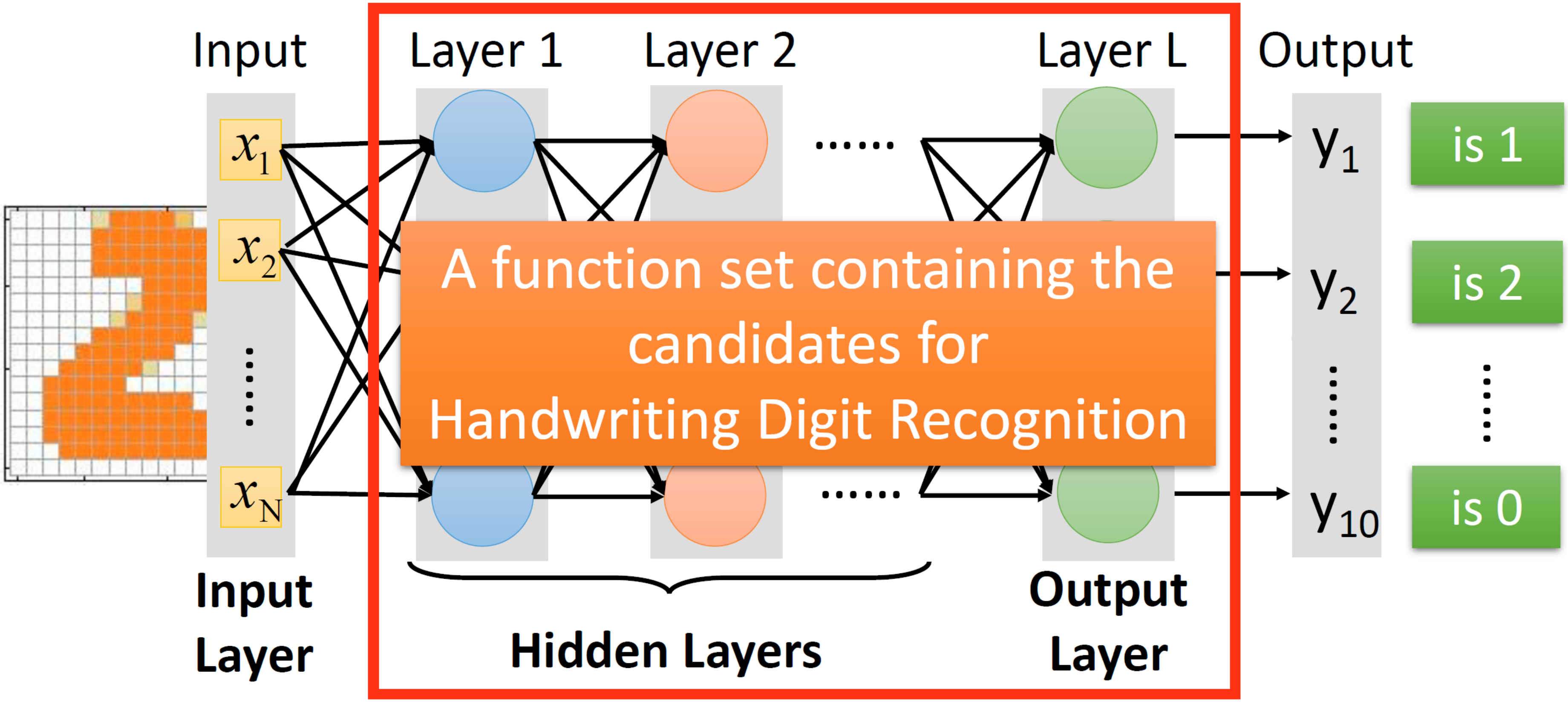
Each dimension represents the confidence of a digit.

Example Application

- Handwriting Digit Recognition

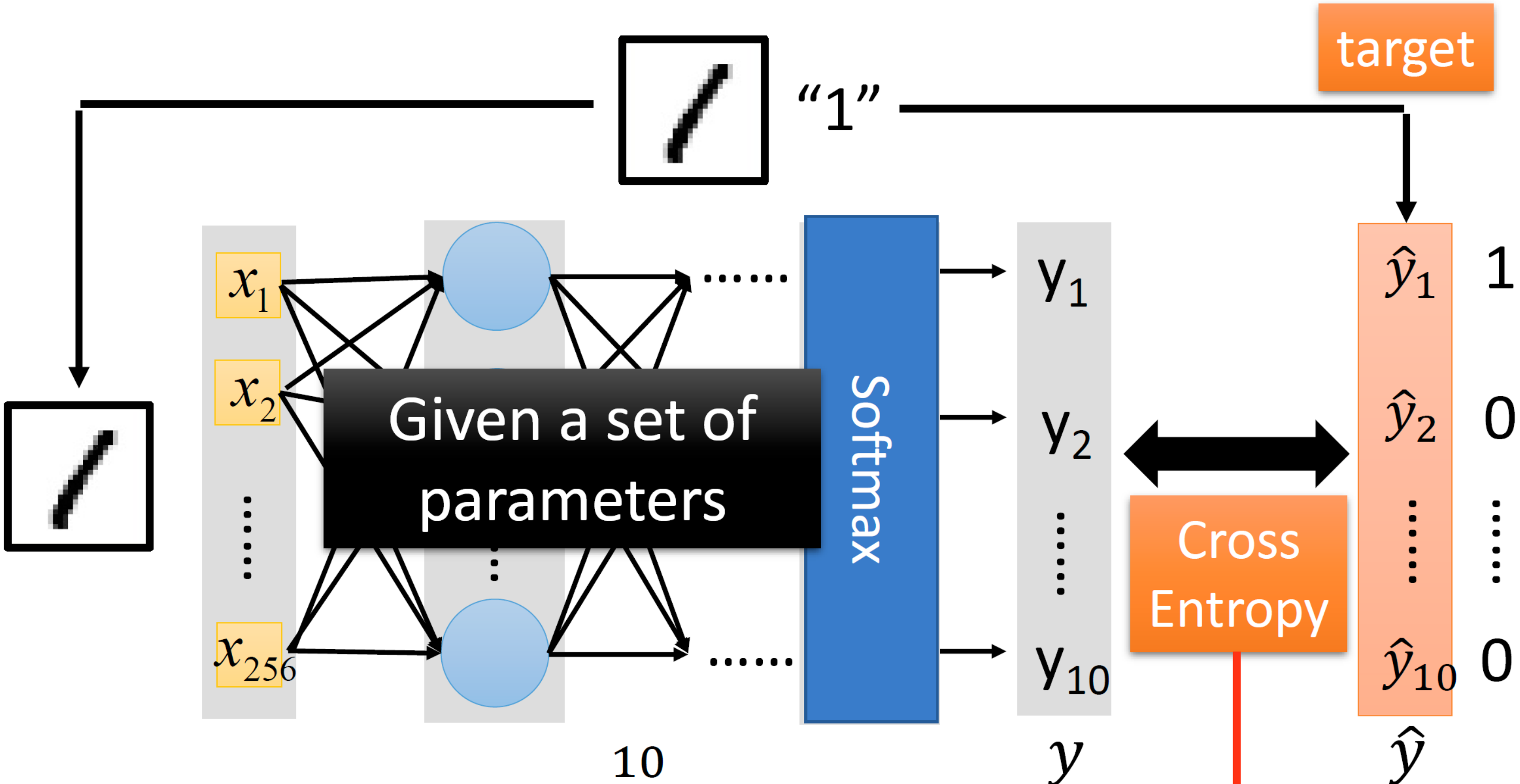


Example Application



You need to decide the network structure to let a good function in your function set.

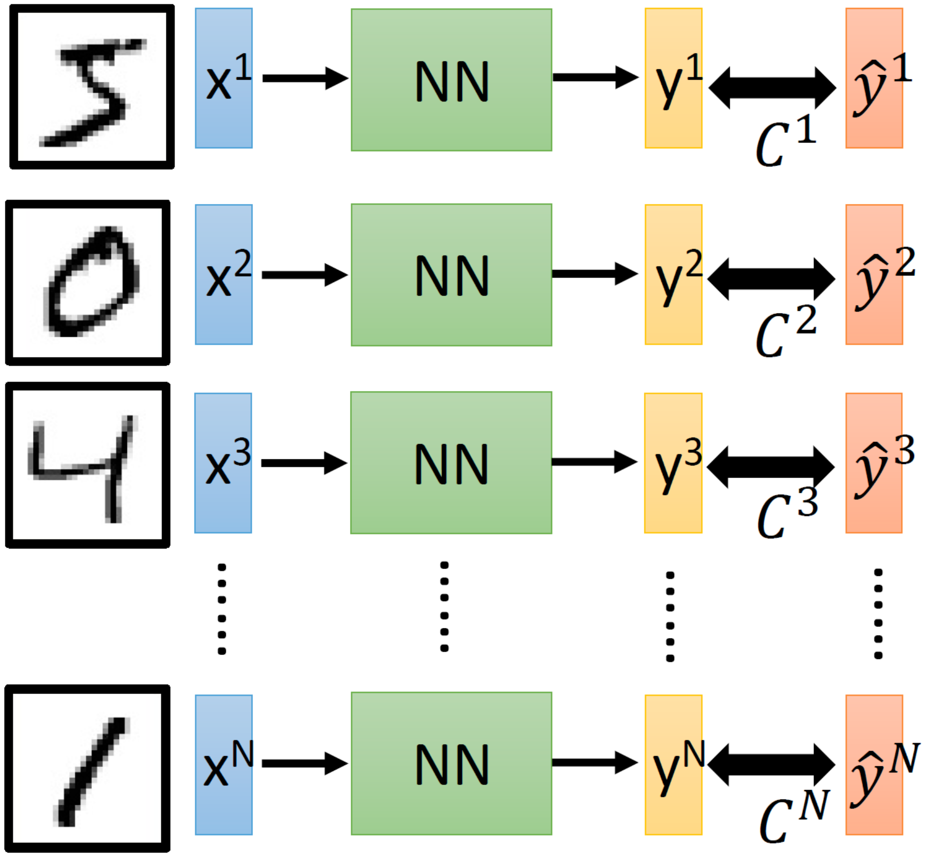
Step 2: Goodness of a function
Loss for an Example



$$C(y, \hat{y}) = -\sum_{i=1}^{10} \hat{y}_i \ln y_i$$

Total Loss

For all training data …



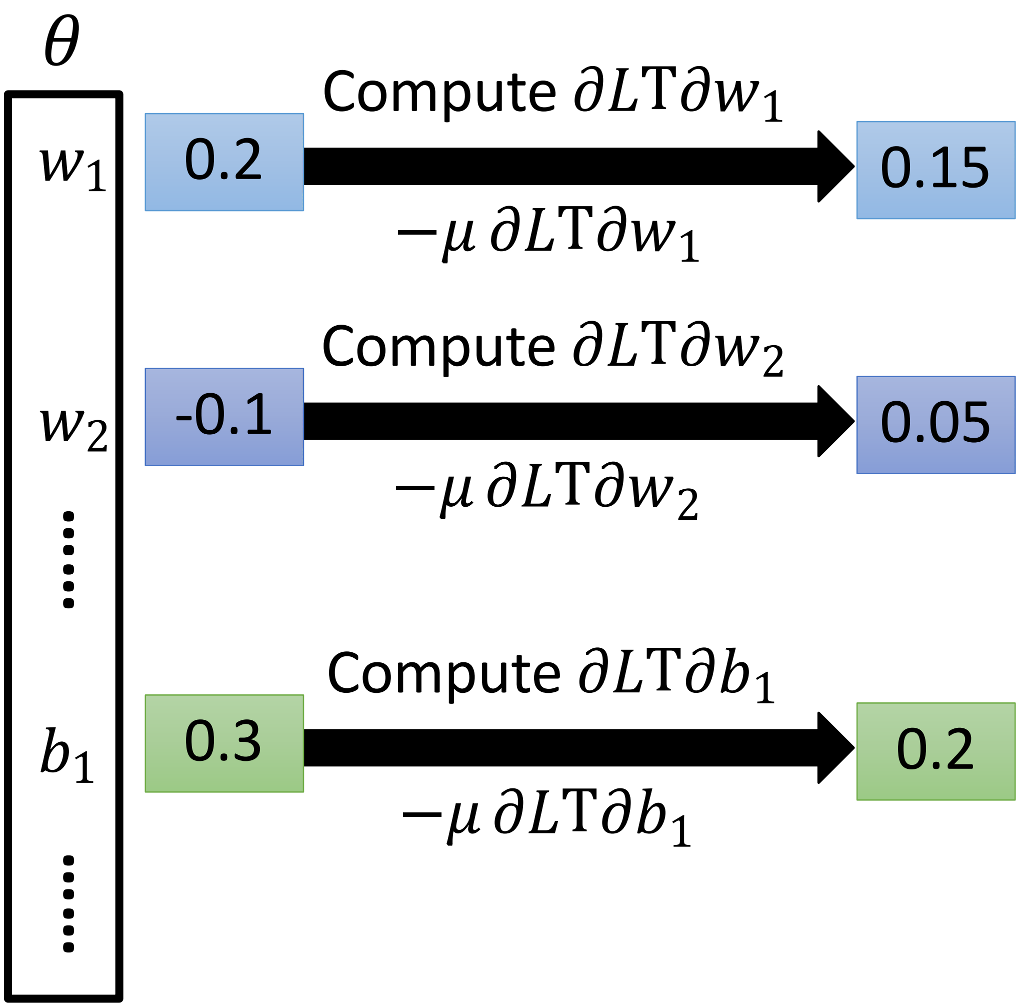
Total Loss:

$$\boxed{L = \sum_{n=1}^{N} C^n}$$

Find ***a function in function set*** that minimizes total loss L

Find ***the network parameters θ^**** that minimize total loss L

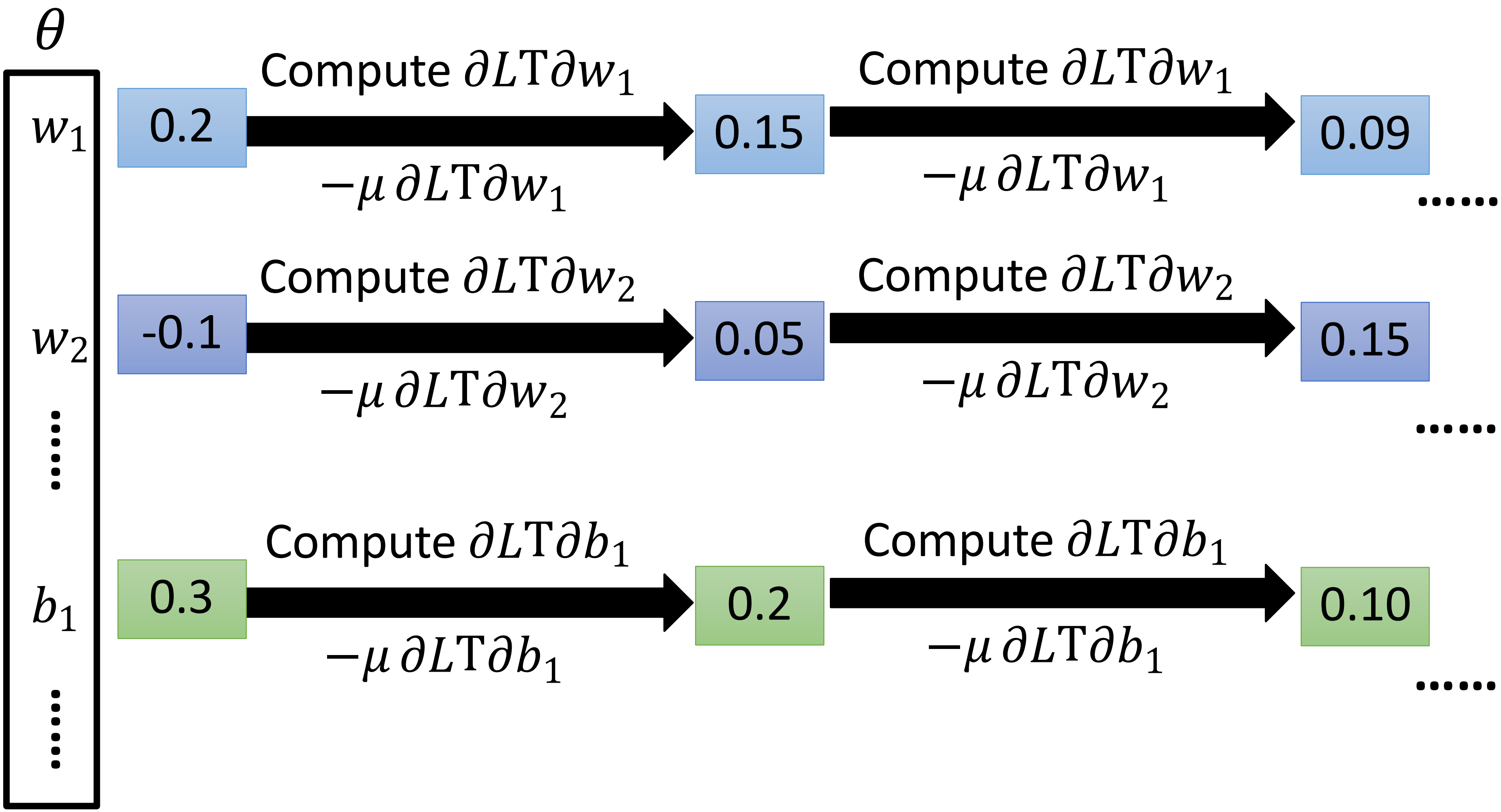
Step 3: Pick the Best Function
Gradient Descent



$$\nabla L = \begin{bmatrix} \dfrac{\partial L}{\partial w_1} \\ \dfrac{\partial L}{\partial w_2} \\ \vdots \\ \dfrac{\partial L}{\partial b_1} \\ \vdots \end{bmatrix}$$

gradient

Gradient Descent

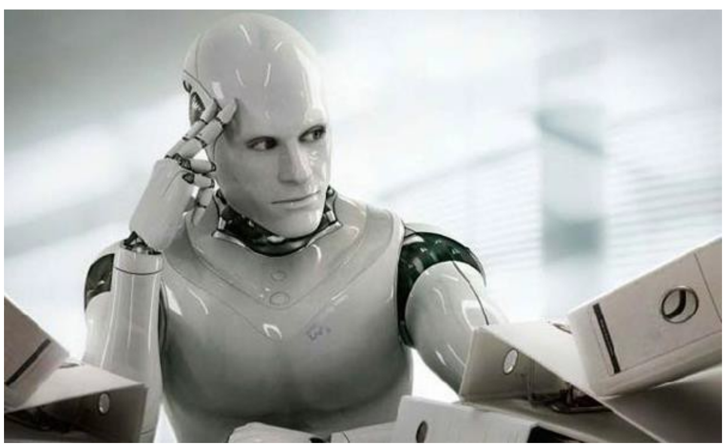


Gradient Descent

This is the “learning” of machines in deep learning ……

➡ Even GPT using this approach.

People image ……



Actually …..

Back Propagation

Gradient Descent

Network parameters $\theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$

Starting Parameters $\theta^0 \longrightarrow \theta^1 \longrightarrow \theta^2 \longrightarrow$ ......

$$\nabla L(\theta) = \begin{bmatrix} \partial L(\theta)/\partial w_1 \\ \partial L(\theta)/\partial w_2 \\ \vdots \\ \partial L(\theta)/\partial b_1 \\ \partial L(\theta)/\partial b_2 \\ \vdots \end{bmatrix}$$

$Compute\ \nabla L(\theta^0)$ $\quad$ $\theta^1 = \theta^0 - \eta \nabla L(\theta^0)$

$Compute\ \nabla L(\theta^1)$ $\quad$ $\theta^2 = \theta^1 - \eta \nabla L(\theta^1)$

Millions of parameters ......

To compute the gradients efficiently, we use ***backpropagation***.

Chain Rule

Case 1

$$y = g(x) \quad z = h(y)$$

$$\Delta x \to \Delta y \to \Delta z \qquad \frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

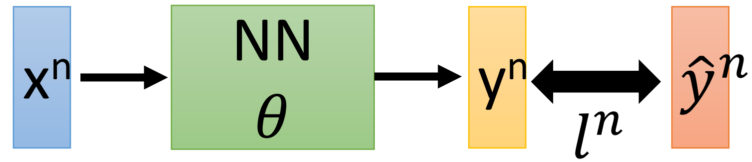
Case 2

$$x = g(s) \qquad y = h(s) \qquad z = k(x, y)$$

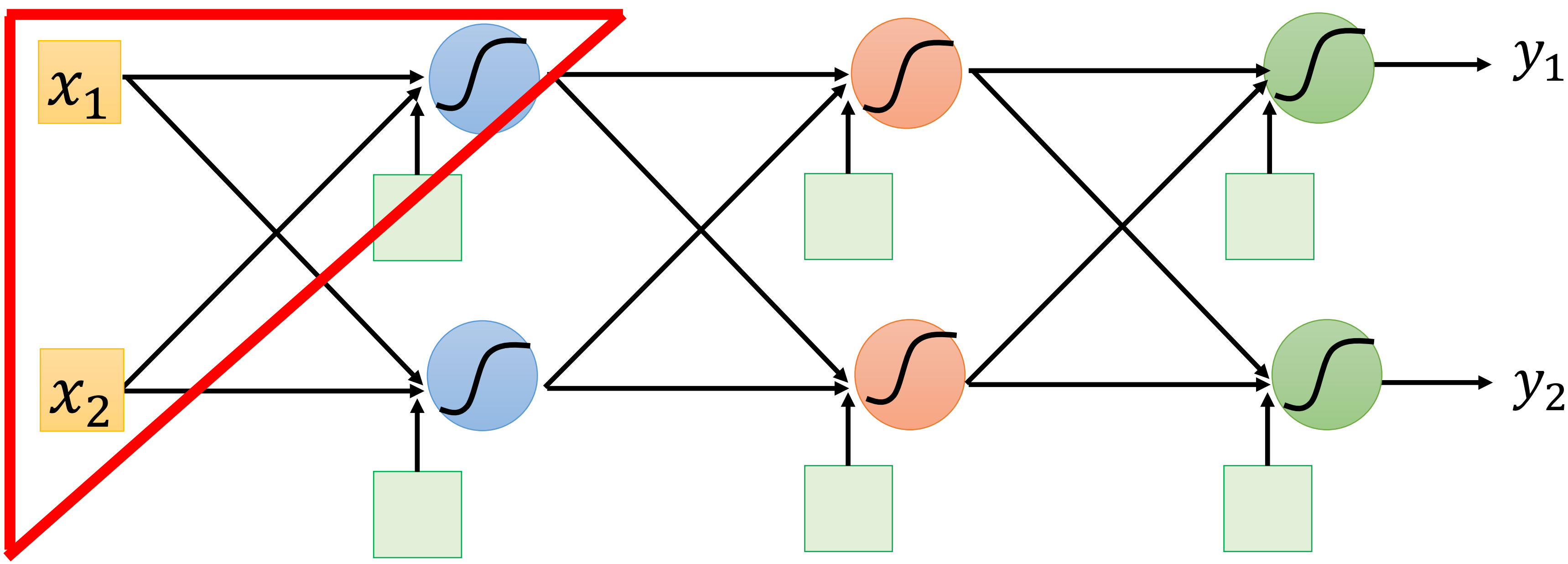
$$\begin{tikzcd} & \Delta x \arrow[dr] & \\ \Delta s \arrow[ur] \arrow[dr] & & \Delta z \\ & \Delta y \arrow[ur] & \end{tikzcd}$$

$$\frac{dz}{ds} = \frac{\partial z}{\partial x}\frac{dx}{ds} + \frac{\partial z}{\partial y}\frac{dy}{ds}$$

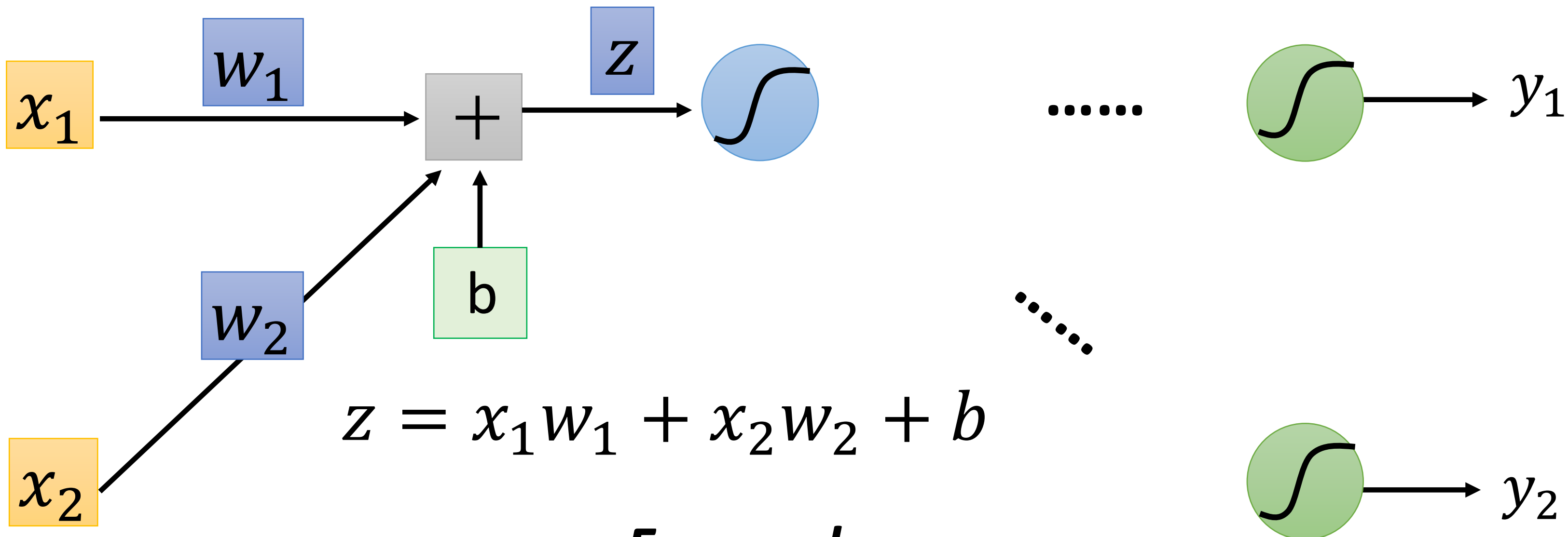
Backpropagation



$$L(\theta) = \sum_{n=1}^{N} l^n(\theta) \quad \Rightarrow \quad \frac{\partial L(\theta)}{\partial w} = \sum_{n=1}^{N} \frac{\partial l^n(\theta)}{\partial w}$$



Backpropagation



$$z = x_1 w_1 + x_2 w_2 + b$$

Forward pass:

Compute $\partial z / \partial w$ for all parameters

$$\frac{\partial l}{\partial w} = ? \qquad \frac{\partial z}{\partial w} \frac{\partial l}{\partial z}$$

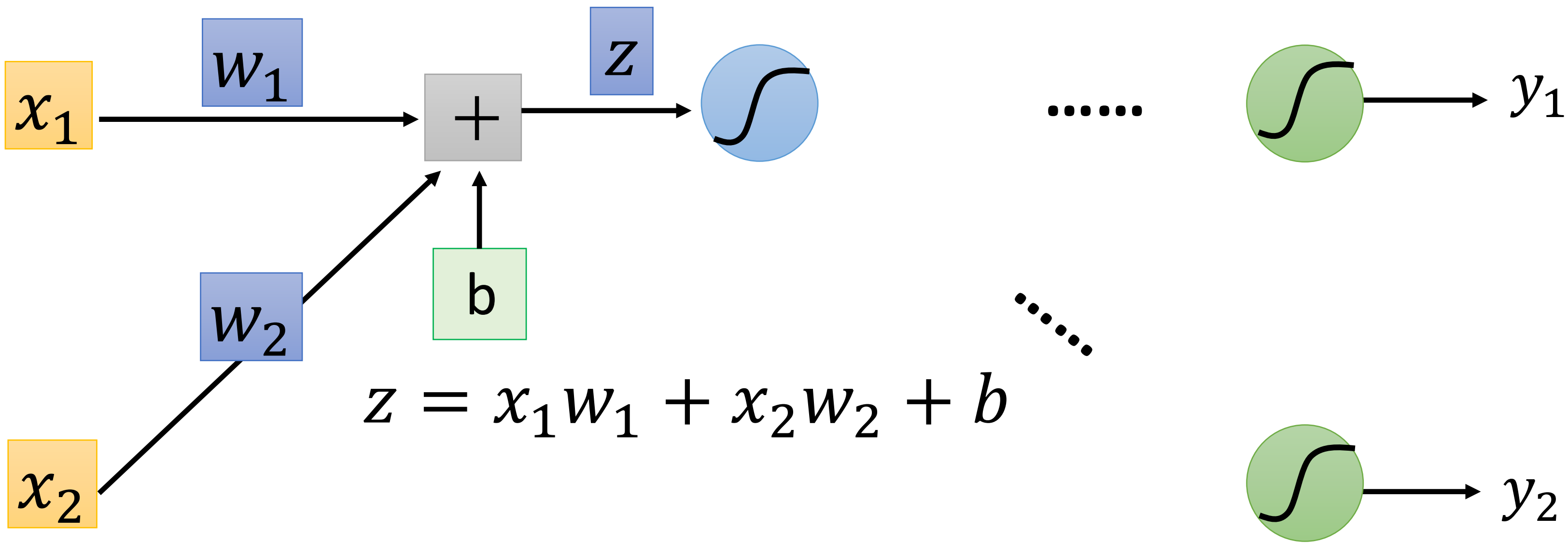
(Chain rule)

Backward pass:

Compute $\partial l / \partial z$ for all activation function inputs z

Backpropagation – Forward pass

Compute $\partial z/\partial w$ for all parameters



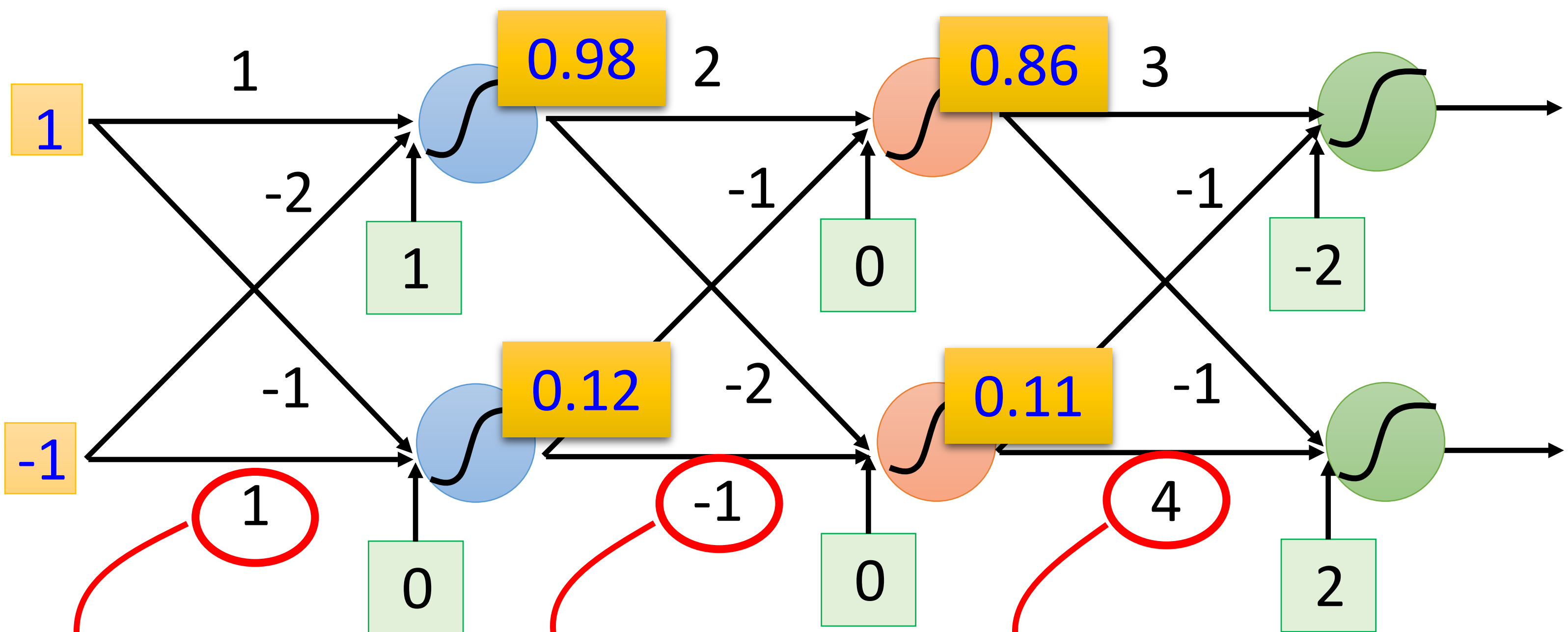
$$z = x_1 w_1 + x_2 w_2 + b$$

$$\left.\begin{array}{l} \partial z/\partial w_1 = ?\ \ x_1 \\ \partial z/\partial w_2 = ?\ \ x_2 \end{array}\right\}$$

The value of the input connected by the weight

Backpropagation – Forward pass

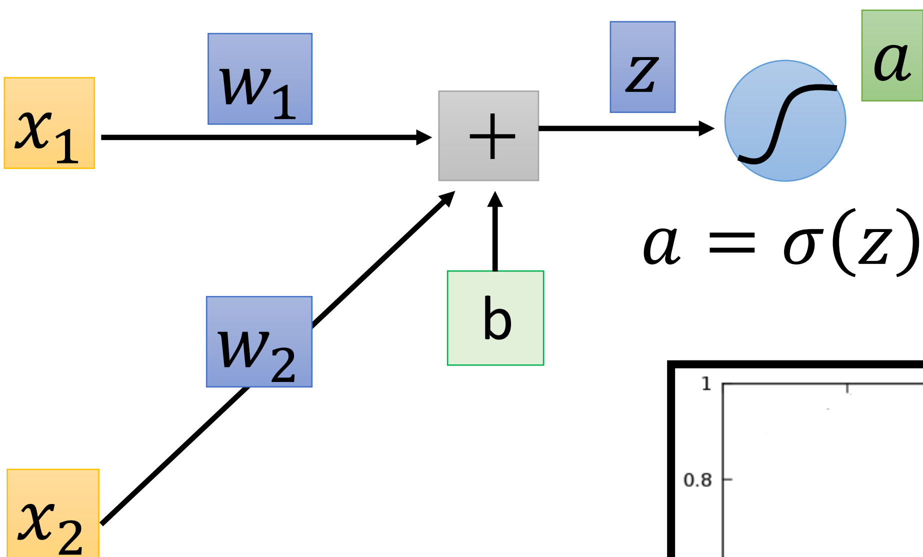
Compute $\partial z/\partial w$ for all parameters



$$\frac{\partial z}{\partial w} = -1 \qquad \frac{\partial z}{\partial w} = 0.12 \qquad \frac{\partial z}{\partial w} = 0.11$$

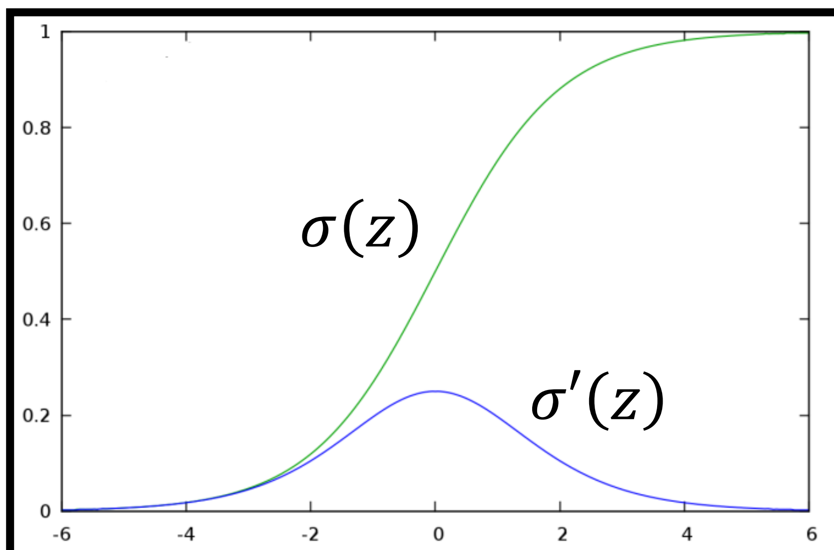
Backpropagation – Backward pass

Compute $\partial l / \partial z$ for all activation function inputs z



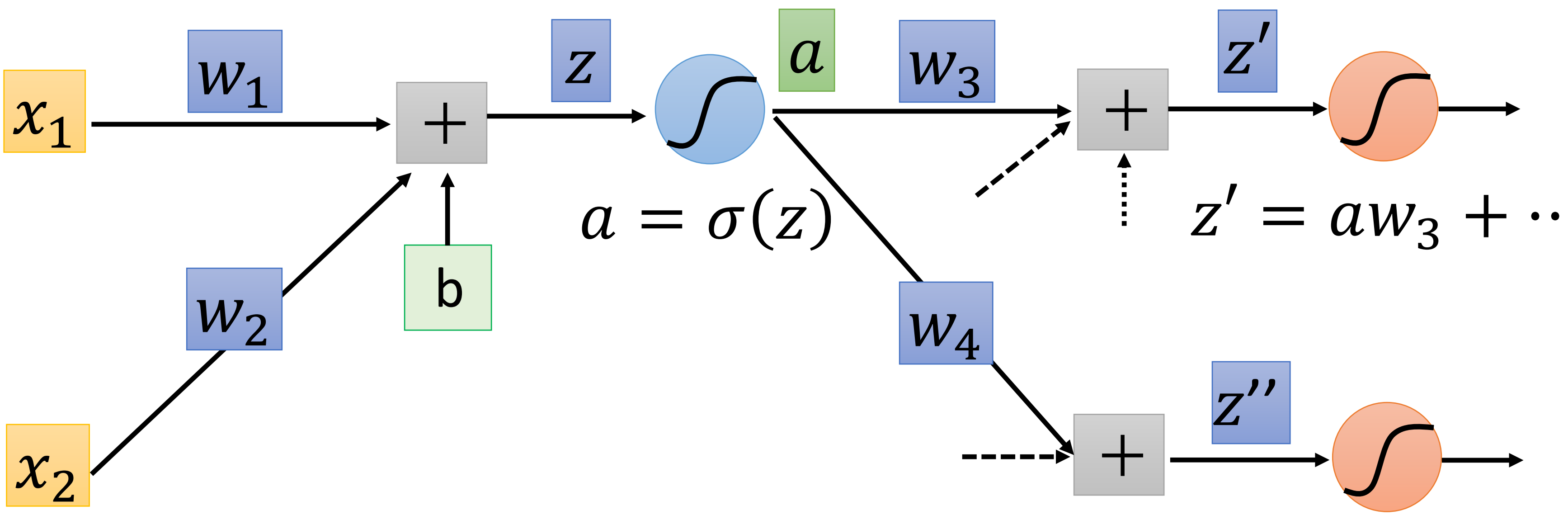
$$\frac{\partial l}{\partial z} = \frac{\partial a}{\underline{\partial z}} \frac{\partial l}{\partial a}$$

$\Rightarrow \sigma'(z)$



Backpropagation – Backward pass

Compute $\partial l / \partial z$ for all activation function inputs z

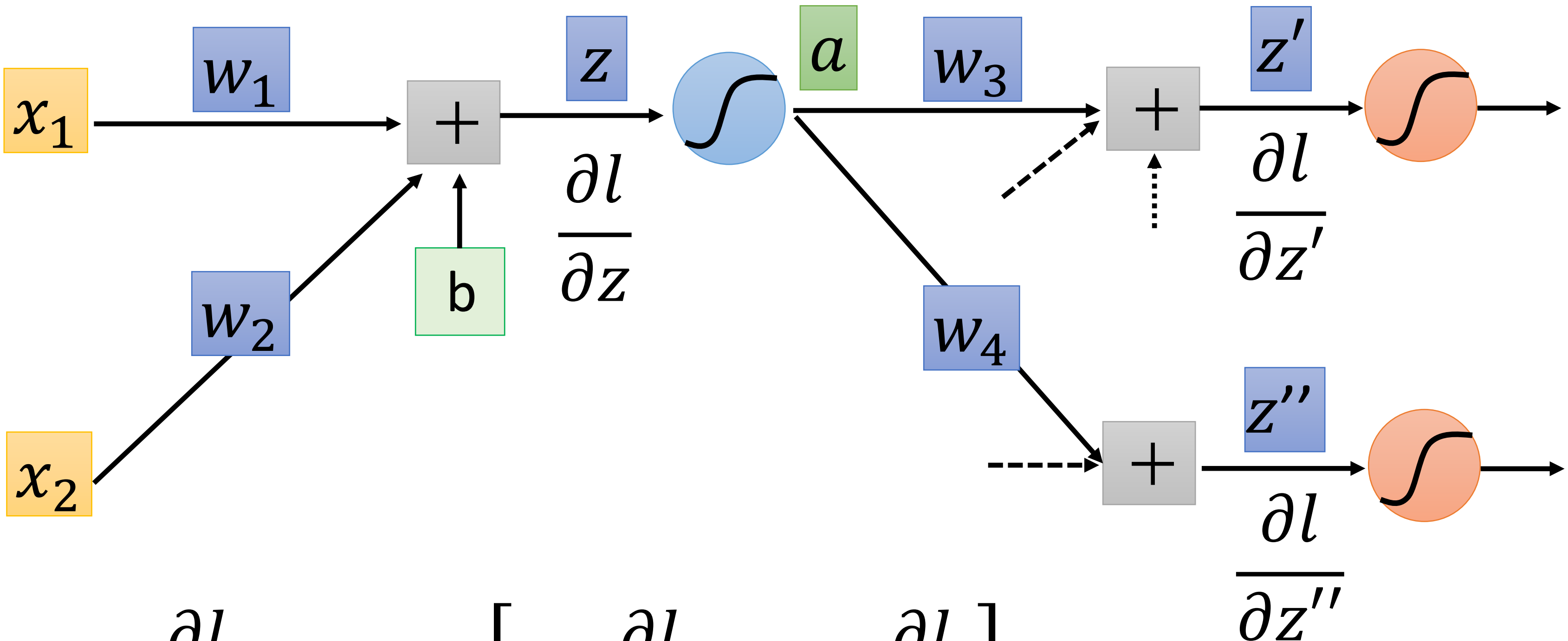


$$\frac{\partial l}{\partial z} = \frac{\partial a}{\partial z} \frac{\partial l}{\partial a} \qquad \frac{\partial l}{\partial a} = \underbrace{\frac{\partial z'}{\partial a}}_{w_3} \underbrace{\frac{\partial l}{\partial z'}}_{?} + \underbrace{\frac{\partial z''}{\partial a}}_{w_4} \underbrace{\frac{\partial l}{\partial z''}}_{?}$$ (Chain rule)

Assumed it's known

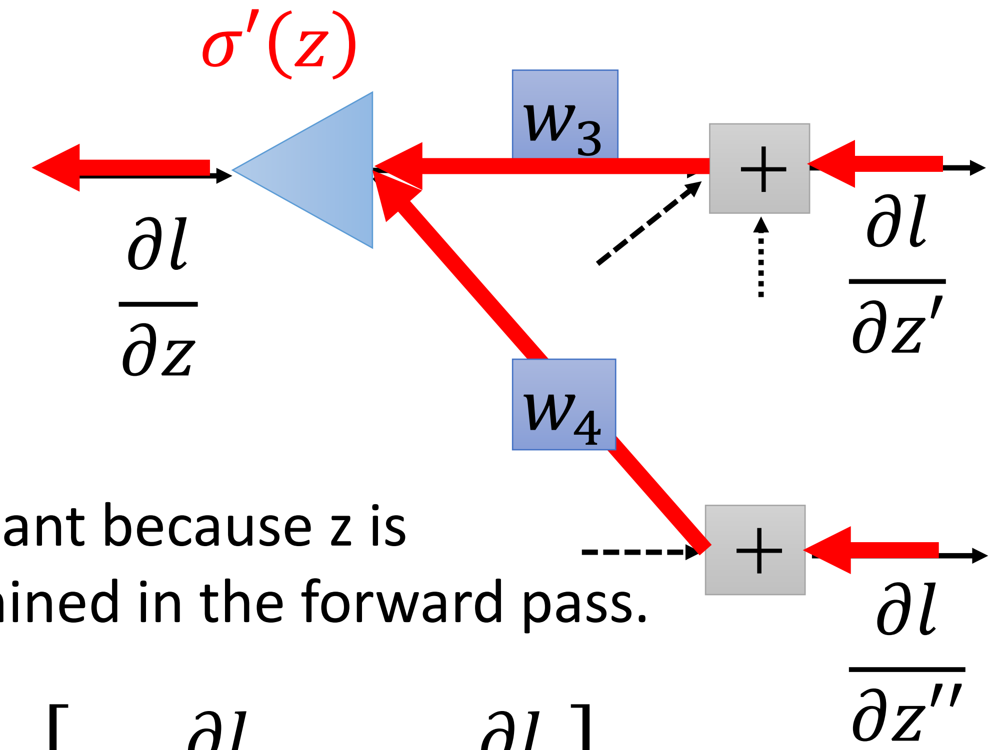
Backpropagation – Backward pass

Compute $\partial l / \partial z$ for all activation function inputs z



$$\frac{\partial l}{\partial z} = \sigma'(z)\left[w_3 \frac{\partial l}{\partial z'} + w_4 \frac{\partial l}{\partial z''}\right]$$

Backpropagation – Backward pass

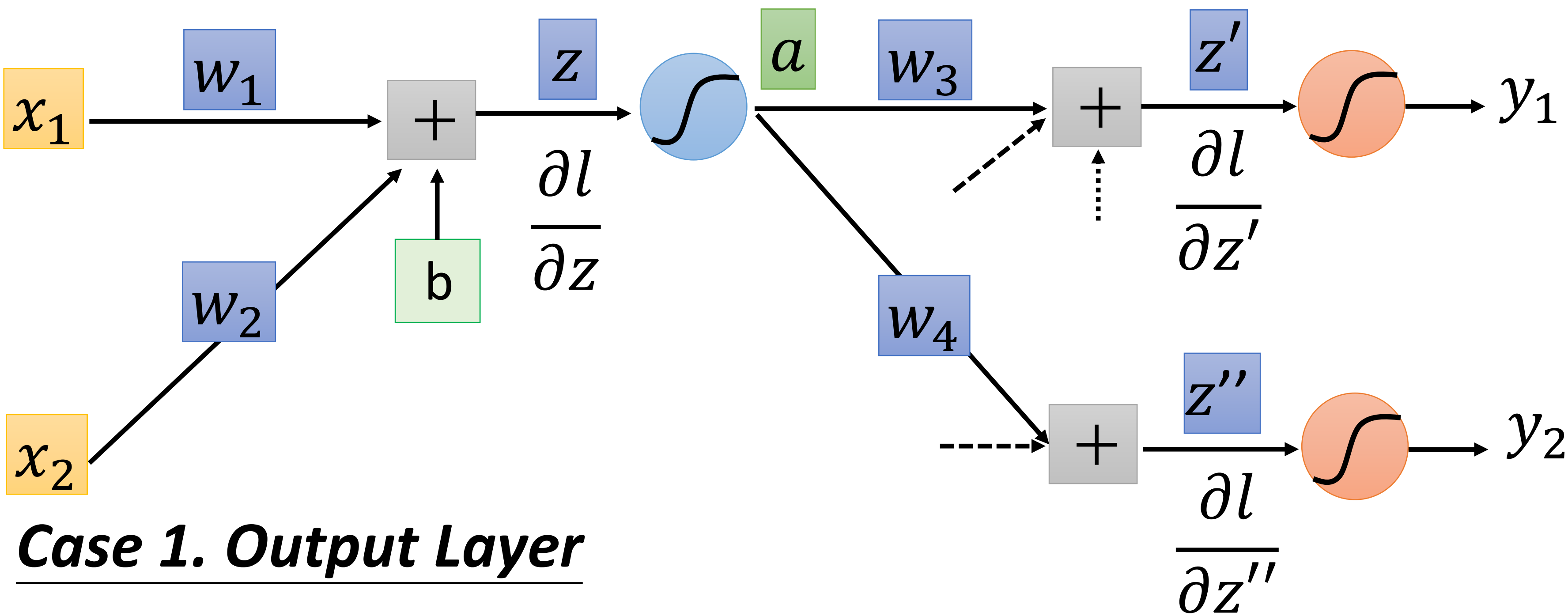


$\sigma'(z)$ is a constant because z is already determined in the forward pass.

$$\frac{\partial l}{\partial z} = \sigma'(z)\left[w_3 \frac{\partial l}{\partial z'} + w_4 \frac{\partial l}{\partial z''}\right]$$

Backpropagation – Backward pass

Compute $\partial l / \partial z$ for all activation function inputs z



Case 1. Output Layer

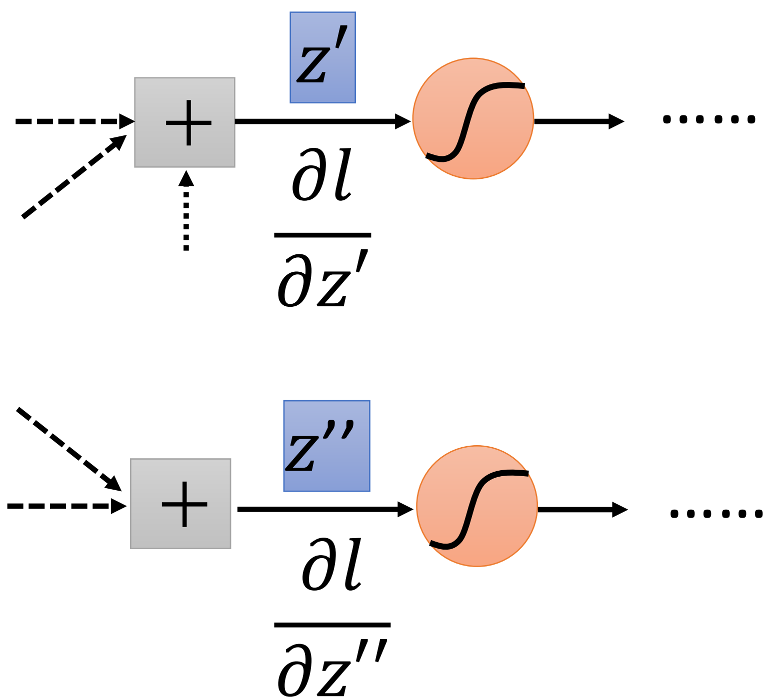
$$\frac{\partial l}{\partial z'} = \frac{\partial y_1}{\partial z'} \frac{\partial l}{\partial y_1} \qquad \frac{\partial l}{\partial z''} = \frac{\partial y_2}{\partial z''} \frac{\partial l}{\partial y_2}$$

Done!

Backpropagation – Backward pass

Compute $\partial l / \partial z$ for all activation function inputs z

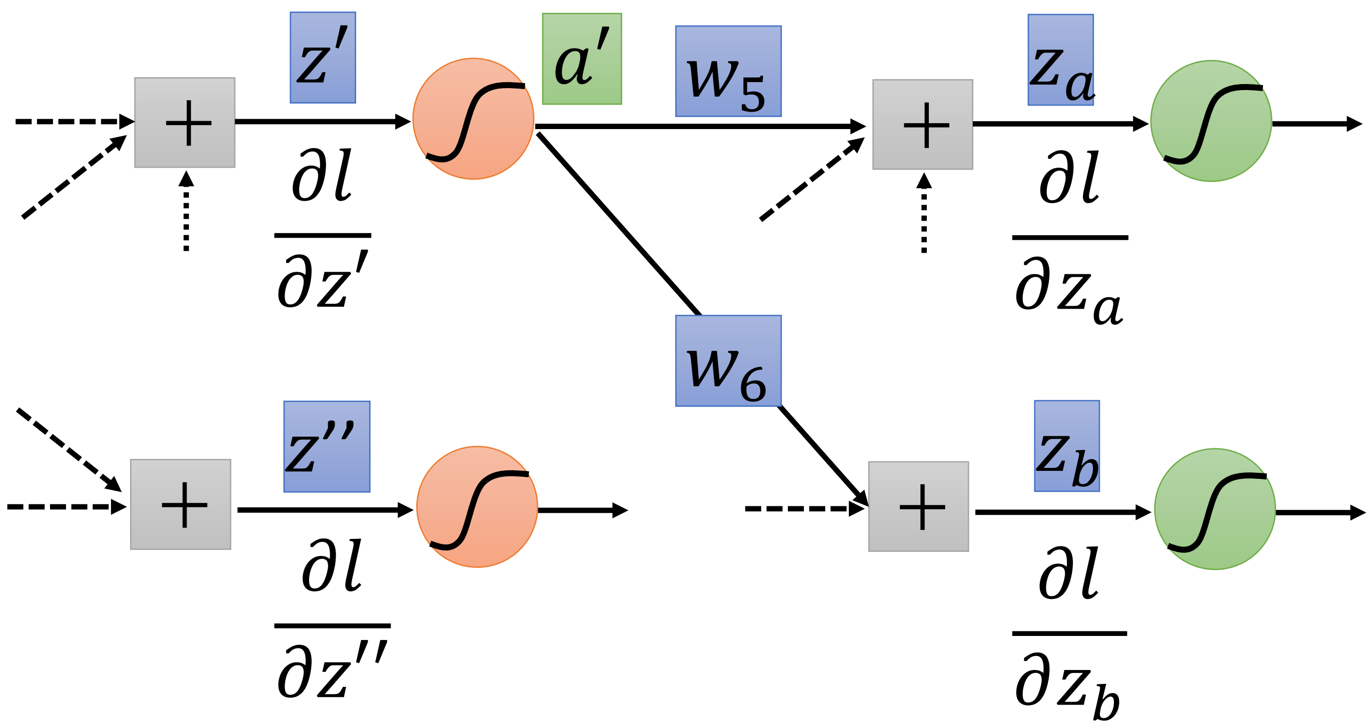
Case 2. Not Output Layer



Backpropagation – Backward pass

Compute $\partial l / \partial z$ for all activation function inputs z

Case 2. Not Output Layer

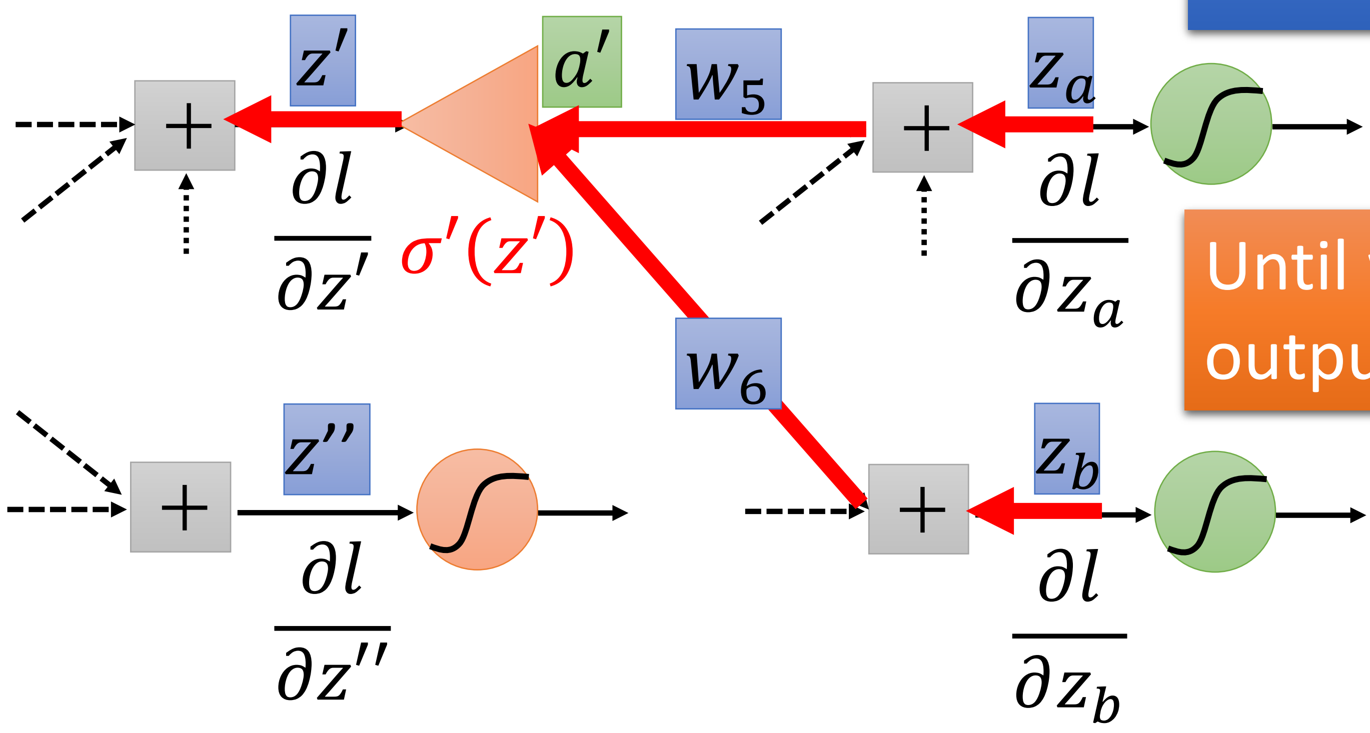


Backpropagation – Backward pass

Compute $\partial l/\partial z$ for all activation function inputs z

Case 2. Not Output Layer

Compute $\partial l/\partial z$ recursively

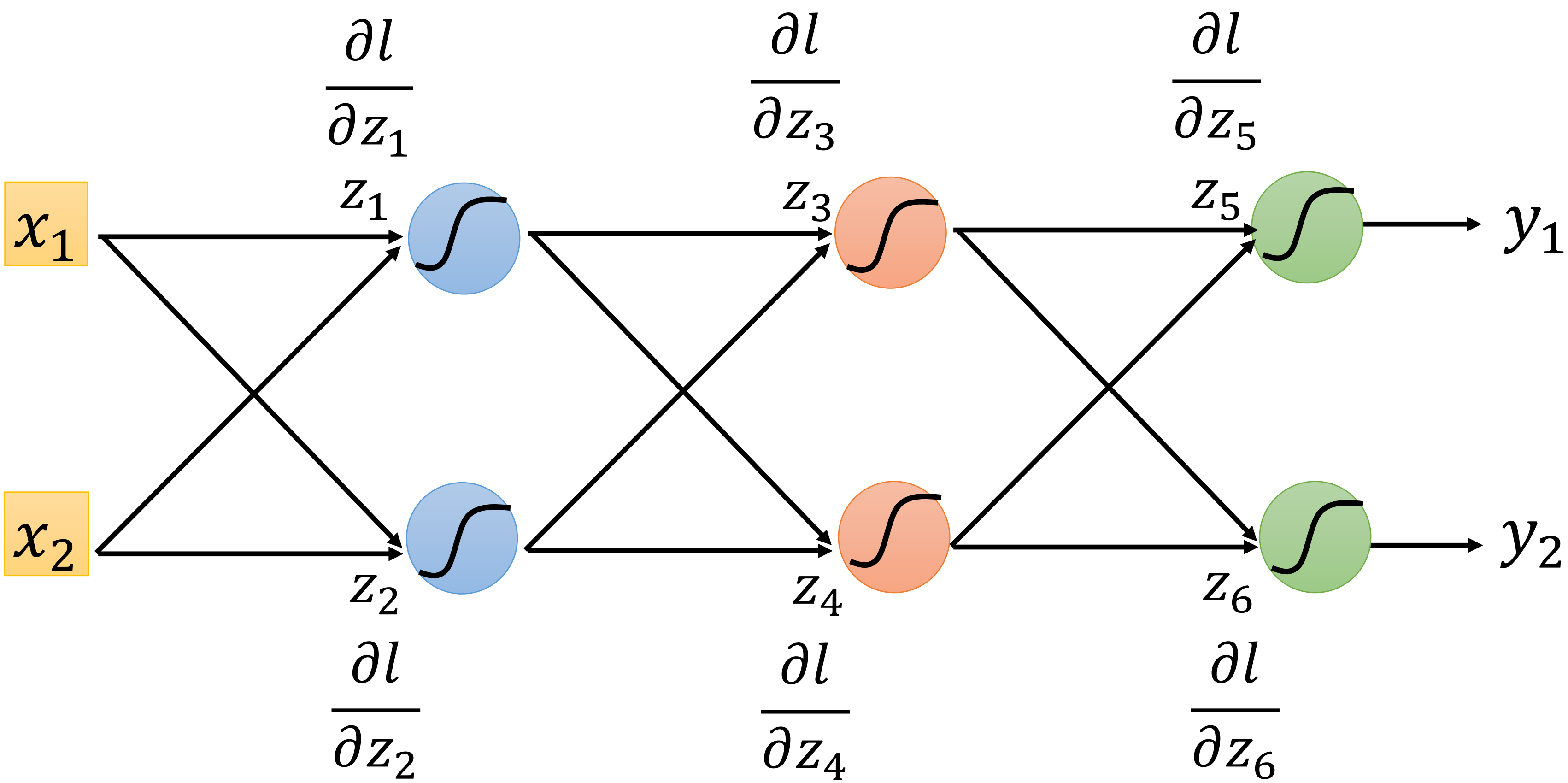


Until we reach the output layer ……

Backpropagation – Backward Pass

Compute $\partial l / \partial z$ for all activation function inputs z

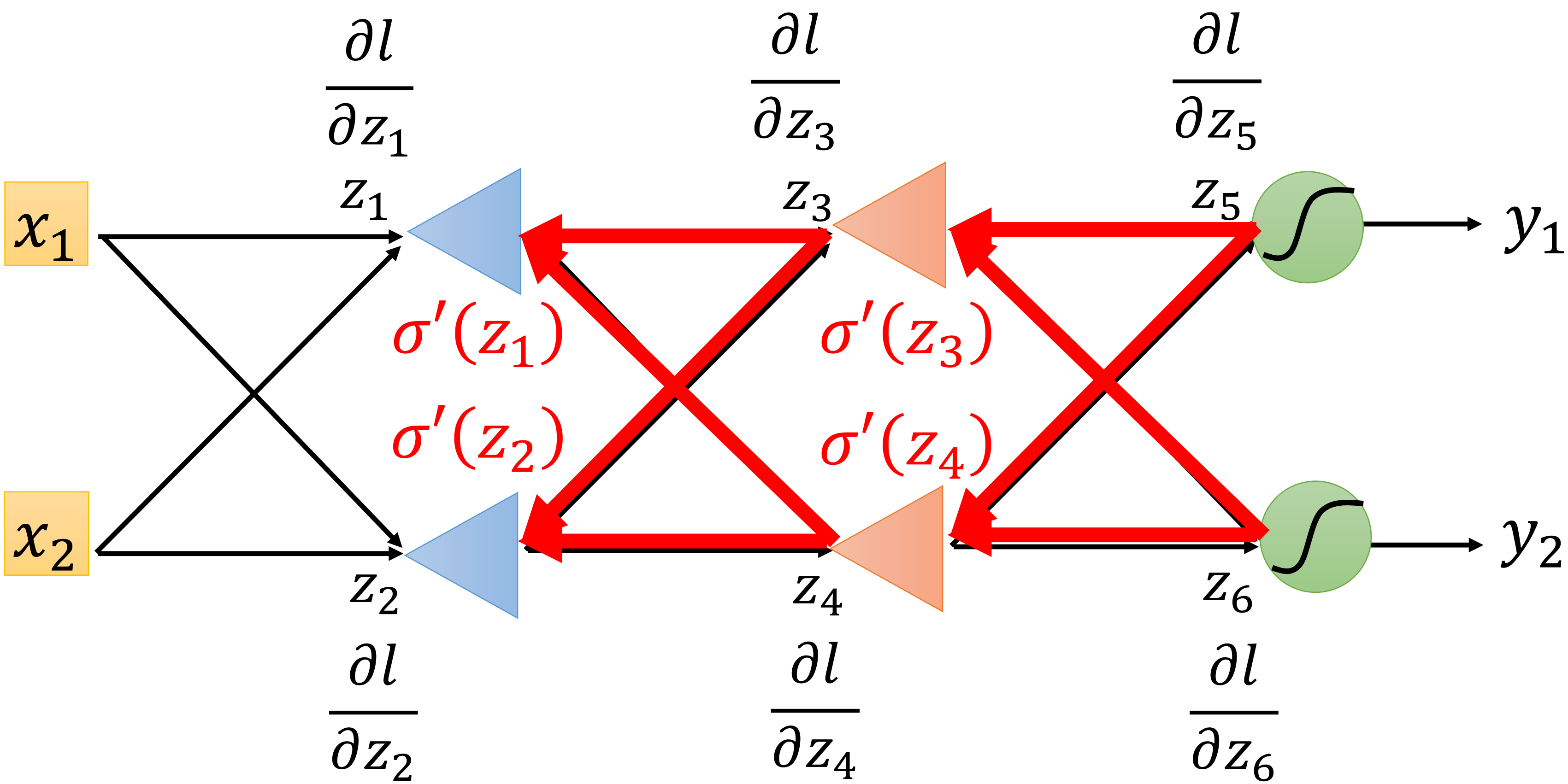
Compute $\partial l / \partial z$ from the output layer



Backpropagation – Backward Pass

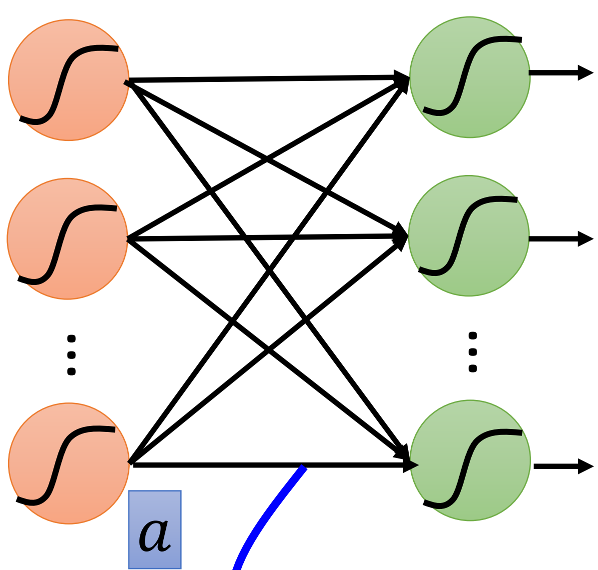
Compute $\partial l/\partial z$ for all activation function inputs z

Compute $\partial l/\partial z$ from the output layer

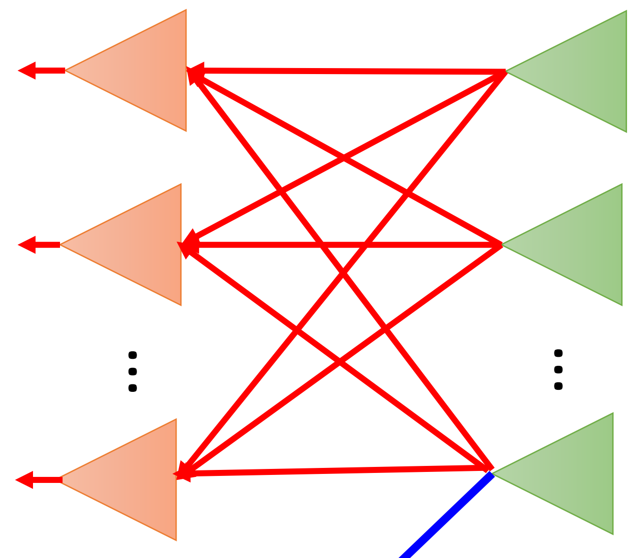


Backpropagation – Summary

Forward Pass



Backward Pass



$$\frac{\partial z}{\partial w} = a \quad \text{X} \quad \frac{\partial l}{\partial z} \quad = \frac{\partial l}{\partial w}$$

for all w