Neural Networks

Today's aim

- To understand
  - When we want to use NNs.
  - Hypothesis function represented by a NN.
  - The relation of the NN to other models.
  - The functions represented by basic layers and their hyperparameters.
  - How to implement NNs.

Outline

- Notation and definitions.
- Where NNs are used.
- Definition of the NN.
- Layers of NN.
- Implementation of NN.

Notation and definitions

Notation: vectors

- $(LHS) \coloneqq (RHS)$: The (LHS) is defined as (RHS).
- $\mathbb{R}^n$: the set of n-dimensional real vectors
- A lower bold letter (e.g., $\boldsymbol{v}$): a column vector.
  - $\boldsymbol{v} = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{bmatrix}$, v_i: the i-th element of vector $\boldsymbol{v}$.
- $\boldsymbol{v}^\top$ the transpose of $\boldsymbol{v}$.
  - E.g., $\boldsymbol{v}^\top = [v_0 \quad v_1 \quad \cdots \quad v_{n-1}]$.

Notation: matrices

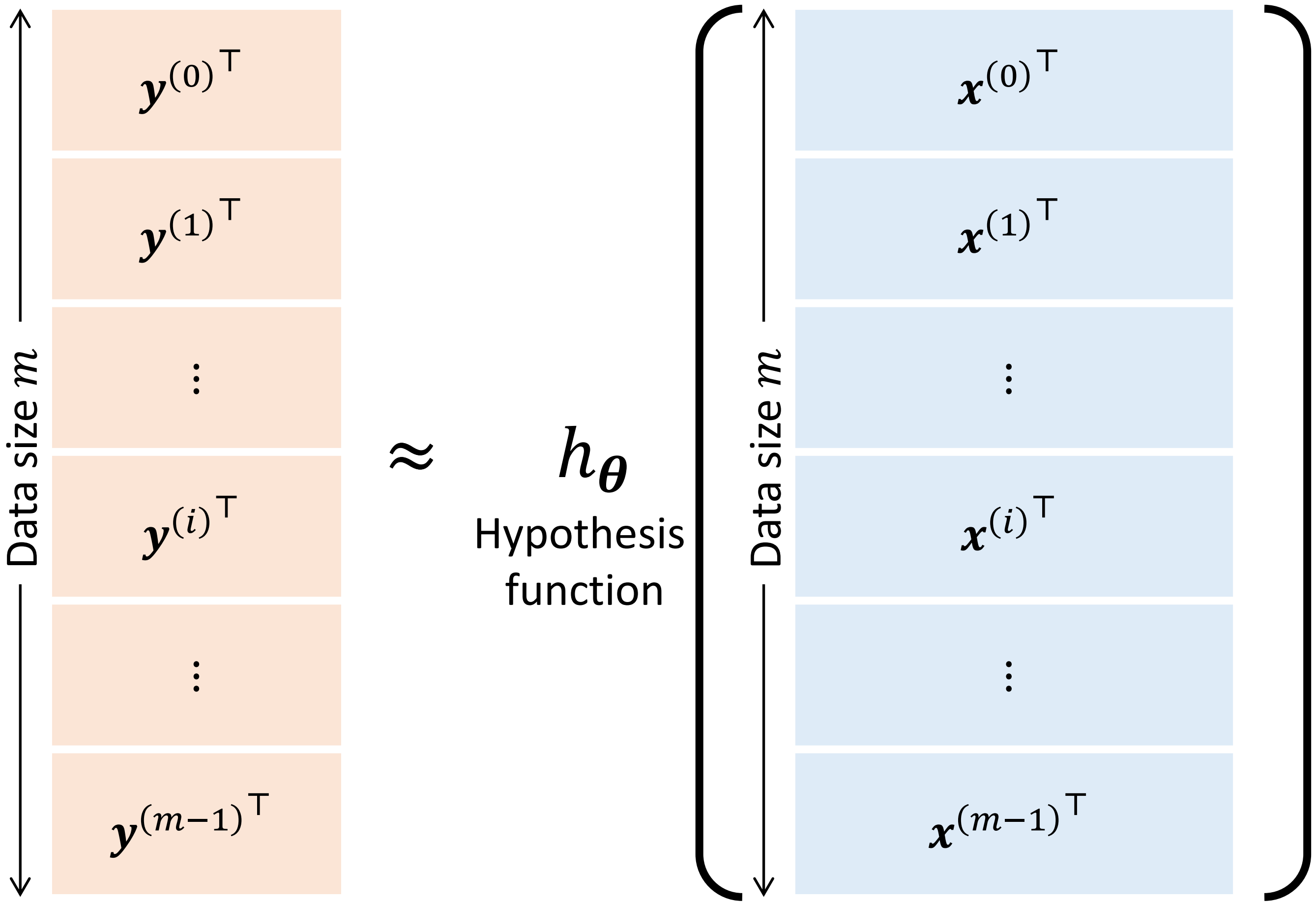
- $\mathbb{R}^{m,n}$: the set of real matrices with the size of $m \times n$.
- A upper bold letter (e.g., $\boldsymbol{A}$): a matrix.
  - $\boldsymbol{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1} \end{bmatrix}$.
  - $a_{i,j}$: the element in the i-th row and the j-th column of matrix $\boldsymbol{A}$.

Notation: matrices

- $\boldsymbol{A}^\top$: the transpose of $\boldsymbol{A}$.
  - E.g., if $\boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$, then $\boldsymbol{A}^\top = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$.

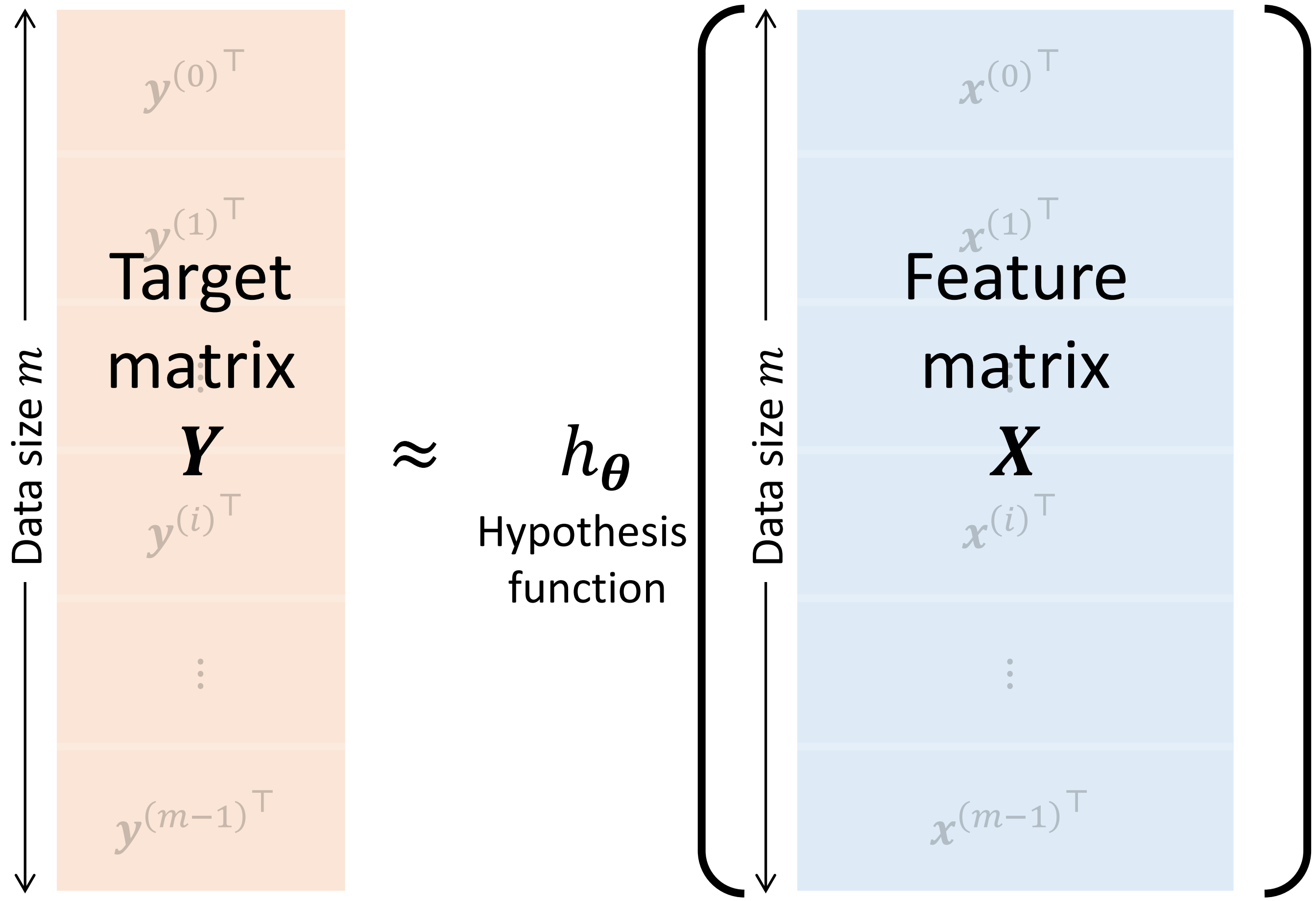
Notation: feature/target vector/matrix

- $\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}$: the feature and target vector of the i-th datapoint.



Notation: feature/target vector/matrix

- $\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}$: the feature and target vector of the i-th datapoint.



Why Neural Networks?

Elements of supervised learning model

- Hypothesis function $h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)$, discriminant function $g_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)$
- (Elementwise) Loss function
- Evaluation function
- Optimisation

Elements of supervised learning model

- Example: linear regression:
  - Hypothesis function $\hat{y}^{(i)} \coloneqq h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) \coloneqq \boldsymbol{x}^{(i)^\top}\boldsymbol{\theta} = 1 \cdot \theta_0 + x_1^{(i)}\theta_1 + \cdots x_n^{(i)}\theta_n$.
  - (Elementwise) loss function
$$\ell\left(y^{(i)}, h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right) \coloneqq \frac{1}{2}\left(y^{(i)} - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)^2$$
  - Evaluation function: mean squared error, RMSE, R2-score
  - Optimisation: gradient descent, the direct method.

Elements of supervised learning model

- Example: logistic regression:
  - Discriminant function $g_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) := \boldsymbol{x}^{(i)\top}\boldsymbol{\theta}$.
  - Hypothesis function $\hat{y}^{(i)} := h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) := \begin{cases} -1 & \text{if } g_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) < 0, \\ +1 & \text{if } g_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) > 0. \end{cases}$
  - (Elementwise) loss function
    $$\ell\left(y^{(i)}, g_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right) := \log\left(1 + \exp\left(-y^{(i)} g_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)\right)\right)$$
  - Evaluation function: accuracy, precision, recall, F1-score, weighted accuracy
  - Optimisation: gradient descent, quasi-Newton methods.

Why do we need a new framework… neural networks (NN)?

- Linear models may be too simple.
- Too complex models may suffer from overfitting.
- Designing a model **specially designed** is often necessary for advanced tasks e.g., natural language processing, digital image processing.
  - Must not be too simple, and must not include hypothesis functions of no use for the task.
- Advantages of NNs
  - **Flexibility** in designing a model: we can design various models to meet the demand of the task
  - No matter what NN models we design, we can optimise it with an integrated algorithm framework, **stochastic gradient descent with backpropagation**.

Supervised learning using NNs

If you use NNs,

- Hypothesis function $h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)$, discriminant function $g_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right)$
- (Elementwise) Loss function
- Evaluation function
- Optimisation

Defined by NNs

Examples

- Digital image processing
  - Convolutional neural networks
  - Residual networks
- Natural language processing
  - Recurrent neural networks
  - Transformers
- In these areas, data are complex but their property is well known.
  We can design NNs from the area knowledge.

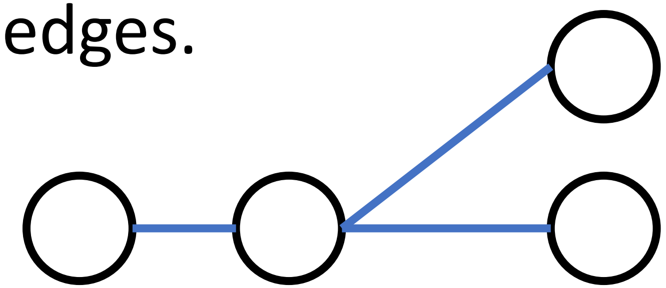
Formal definition and the role of neural networks

The hypothesis function defined by a NN.

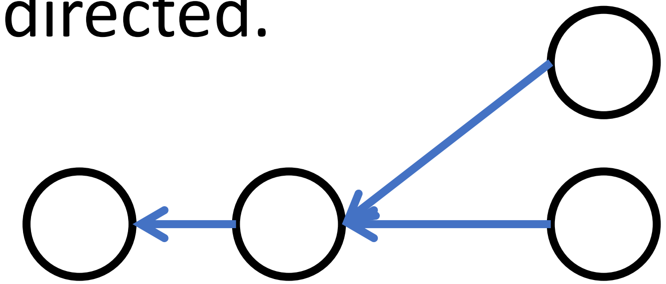
- A function defined by the composition of the functions represented by the nodes and edges in a weighted directed acyclic graph (weighted DAG), where
  - each node represents a fixed function
  - each edge represents a the multiplication by a learnable parameter represented by the weight of the edge

Weighted directed acyclic graph?

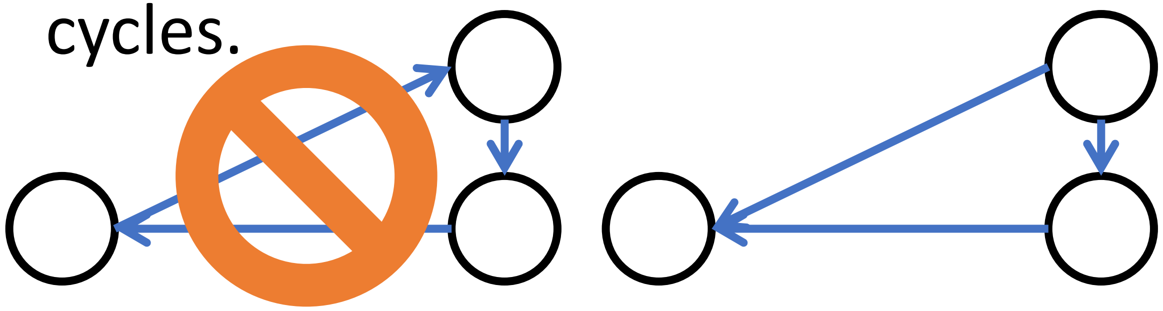
- Graph:
  a set of nodes connected by edges.



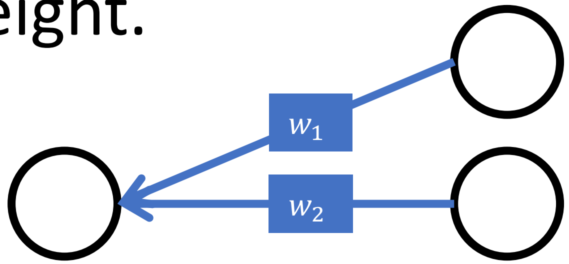
- Directed graph:
  a graph whose edges are all directed.



- Directed acyclic graph (DAG):
  a directed graph w/o directed cycles.

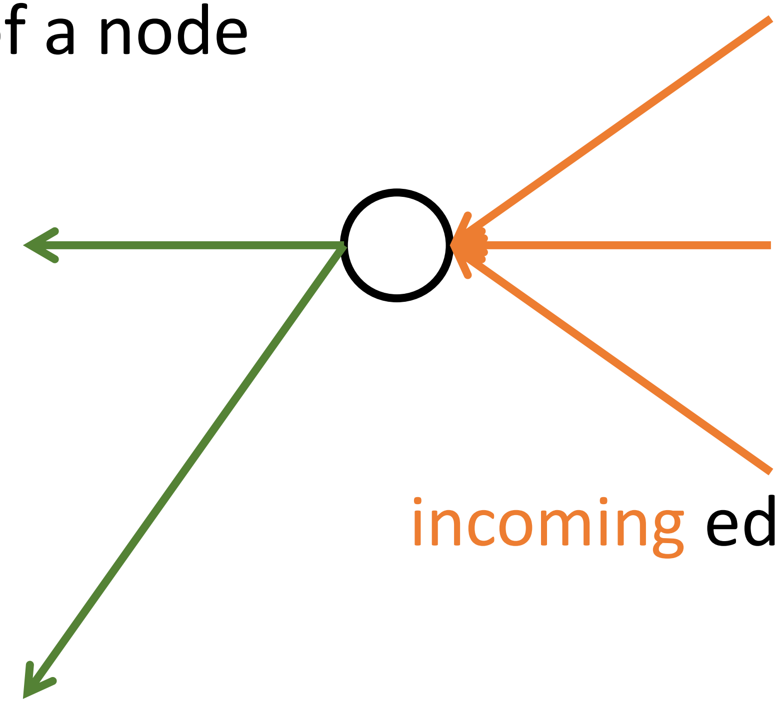


- Weighted DAG:
  a directed graph such that each edge has its own parameter called the weight.



Terminology

- The outgoing/incoming edges of a node



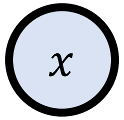
incoming edges

outgoing edges

- The head/tail node of an edge

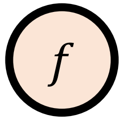
head node   tale node

Hypothesis function of a neural network

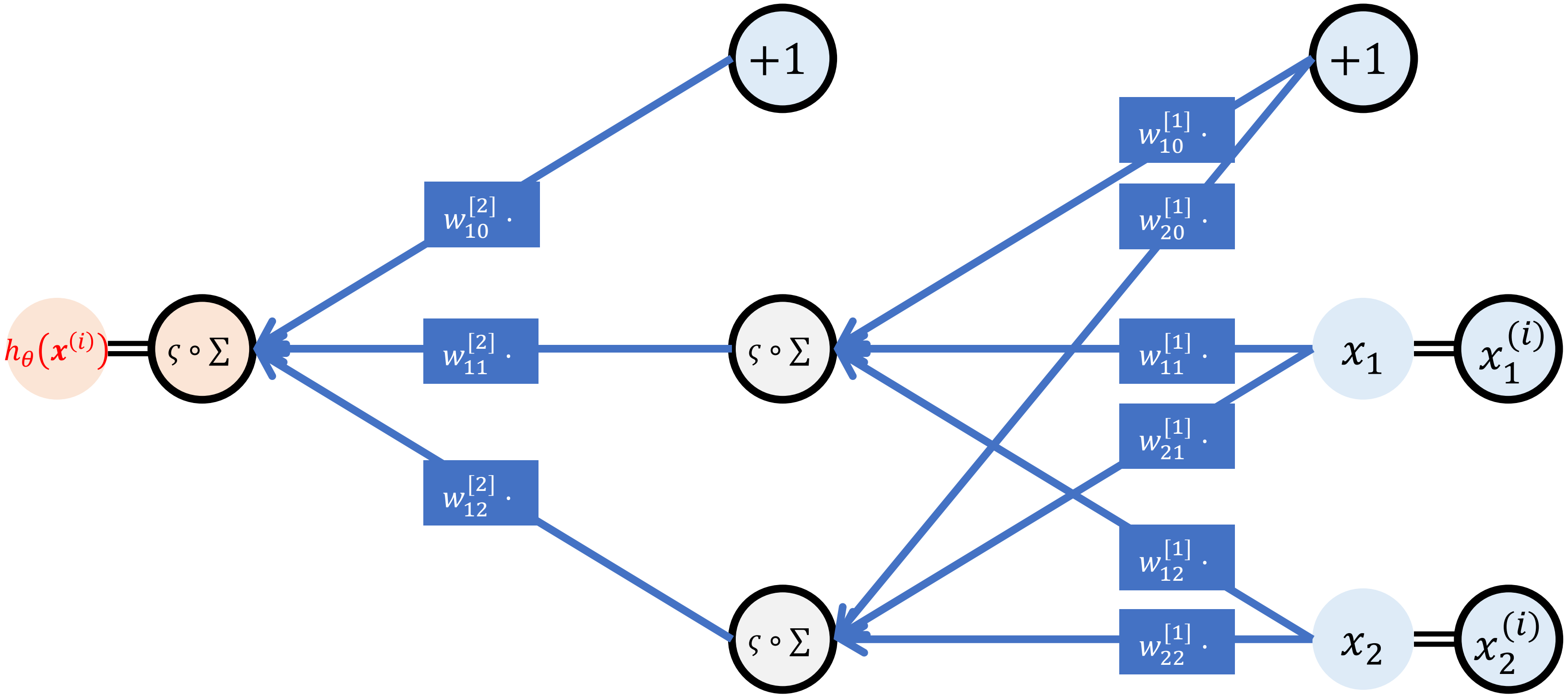


Input node
- Has no incoming edges
- Sends the input

Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)



Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)



Edge
- Multiplication by a learnable parameter w

The weight of the edge
outgoing from the j-th node of the $(k-1)$-th layer
incoming to i-th node of the k-th layer

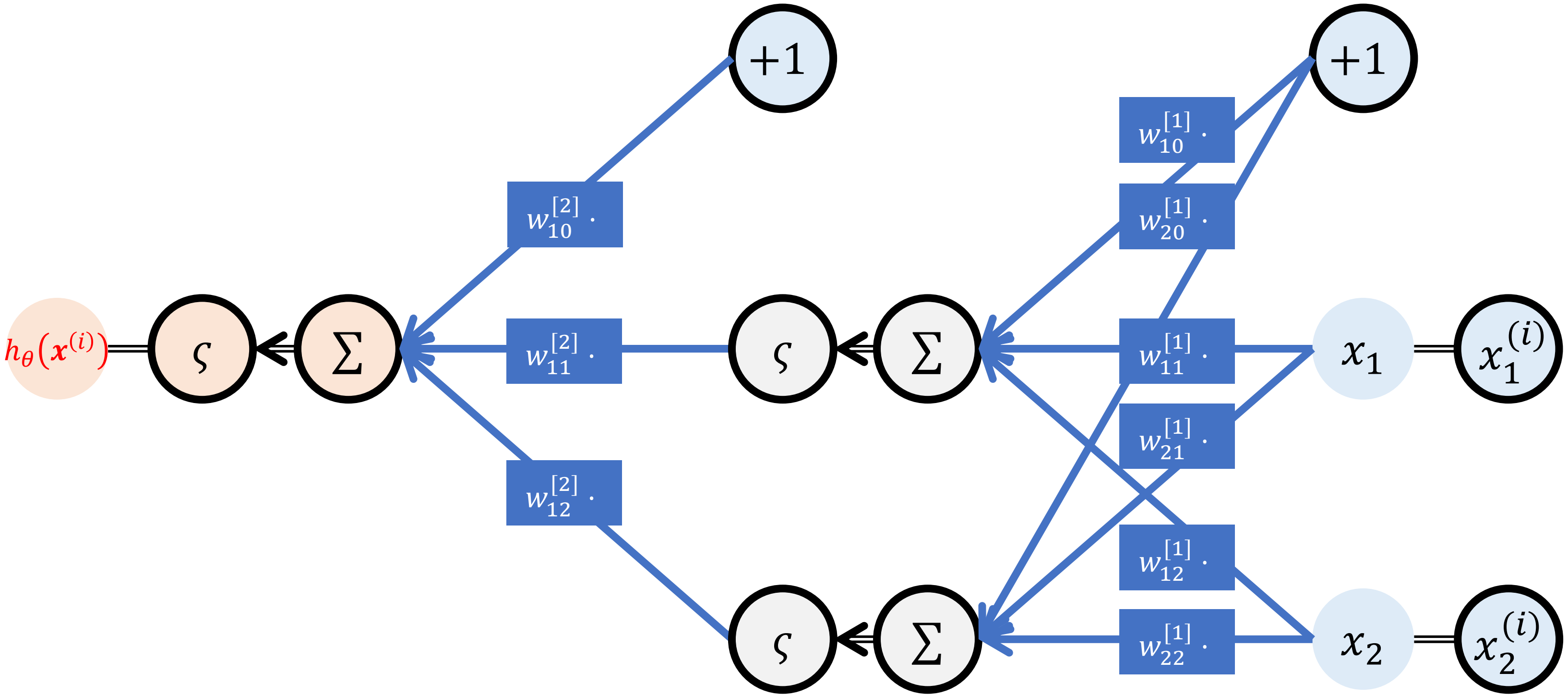
Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$

Hypothesis function of a neural network

x Input node
- Has no incoming edges
- Sends the input

f Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

f Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)



$w\cdot$ Weighted Edge
- Multiplication by a learnable parameter w

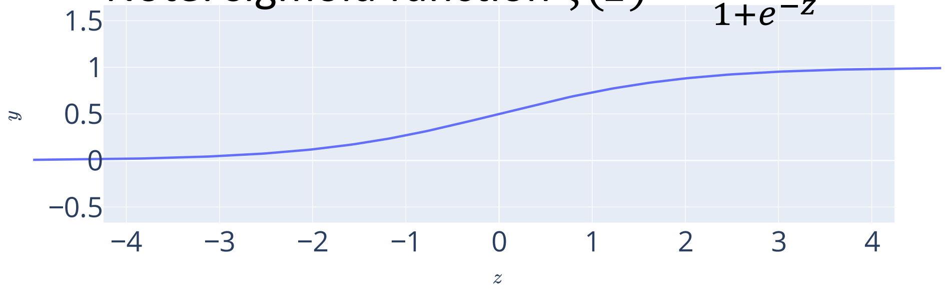
The weight of the edge
outgoing from the j-th node of the $(k-1)$-th layer
incoming to i-th node of the k-th layer

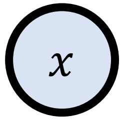
Unweighted Edge
- Direct input

c f f's output equals c.
c is defined
as f's output.

Example: AND function

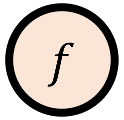
Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$



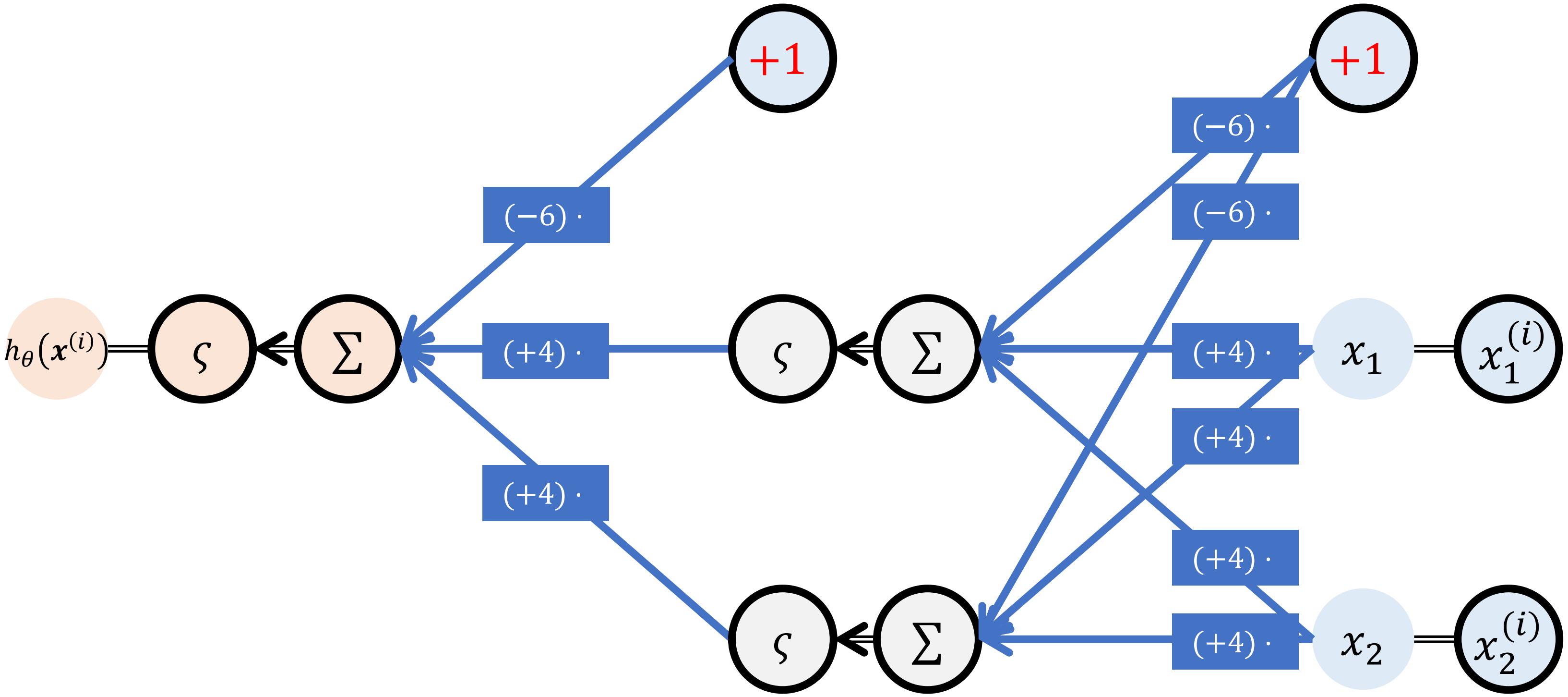


Input node
- Has no incoming edges
- Sends the input

Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

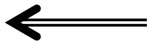


Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)

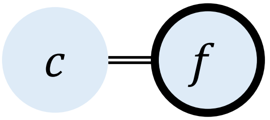


$x_1^{(i)}$	$x_2^{(i)}$	AND
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

Weighted Edge
- Multiplication by a learnable parameter w



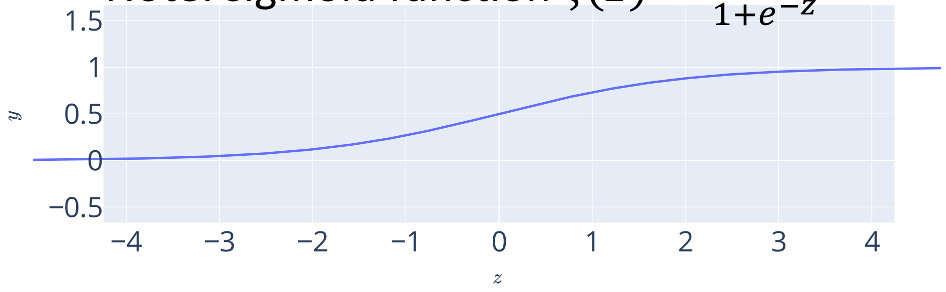
Unweighted Edge
- Direct input



f's output equals c.
c is defined as f's output.

Example: AND function

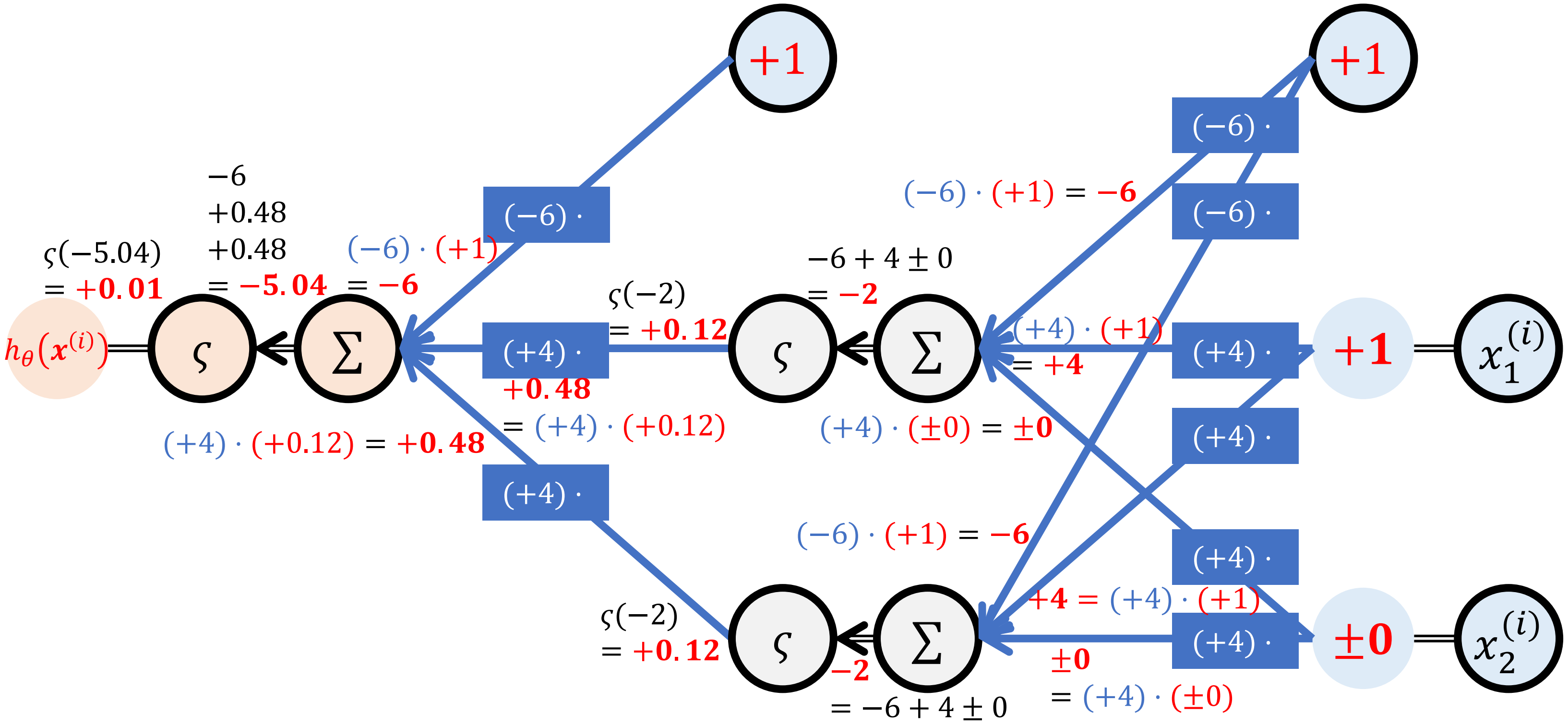
Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$



Input node
- Has no incoming edges
- Sends the input

Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)



Weighted Edge
- Multiplication by a learnable parameter w

Unweighted Edge
- Direct input

f's output equals c.
c is defined as f's output.

Example: AND function

x Input node
- Has no incoming edges
- Sends the input

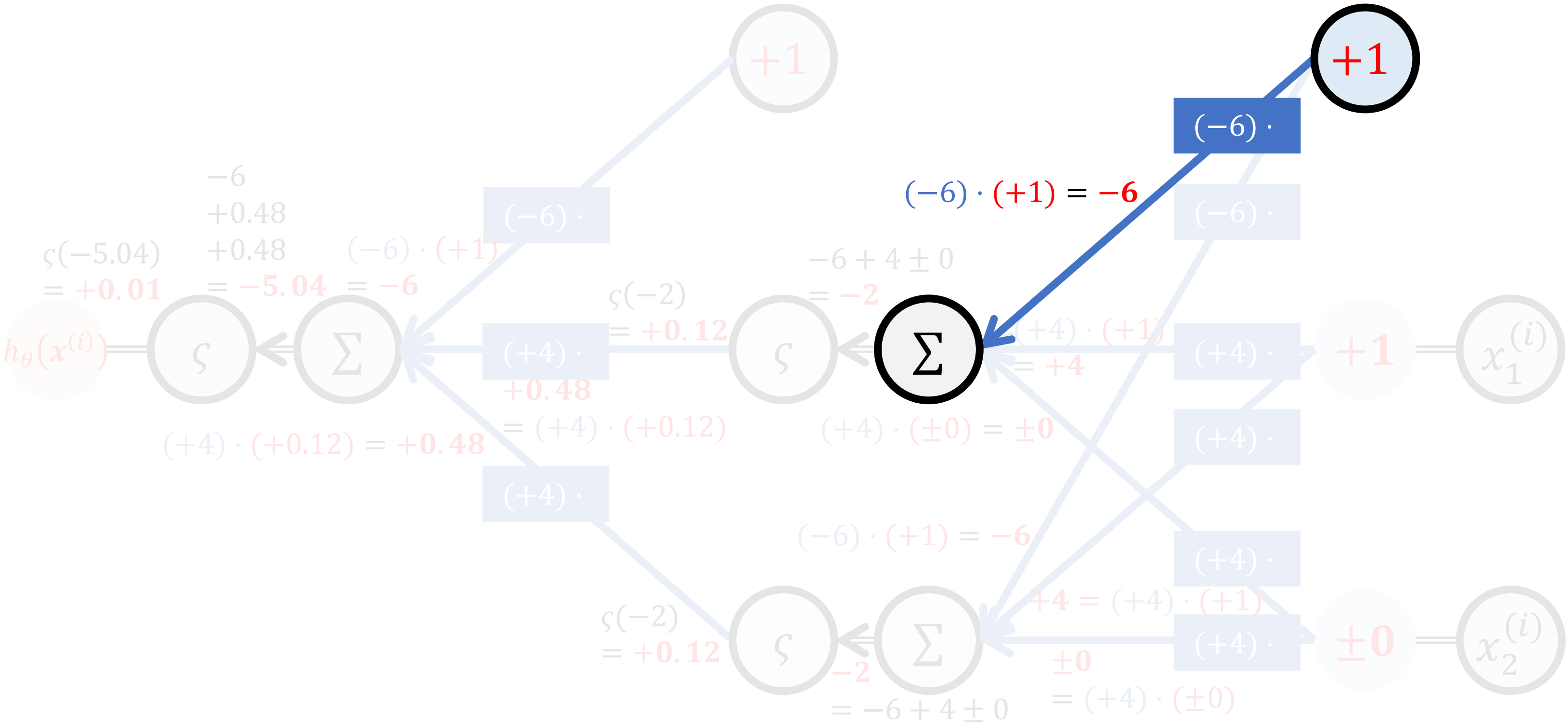
f Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

f Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)

$w\,\cdot$ Weighted Edge
- Multiplication by a learnable parameter w

Unweighted Edge
- Direct input

c = f : f's output equals c.
c is defined as f's output.



Example: AND function

Input node
- Has no incoming edges
- Sends the input

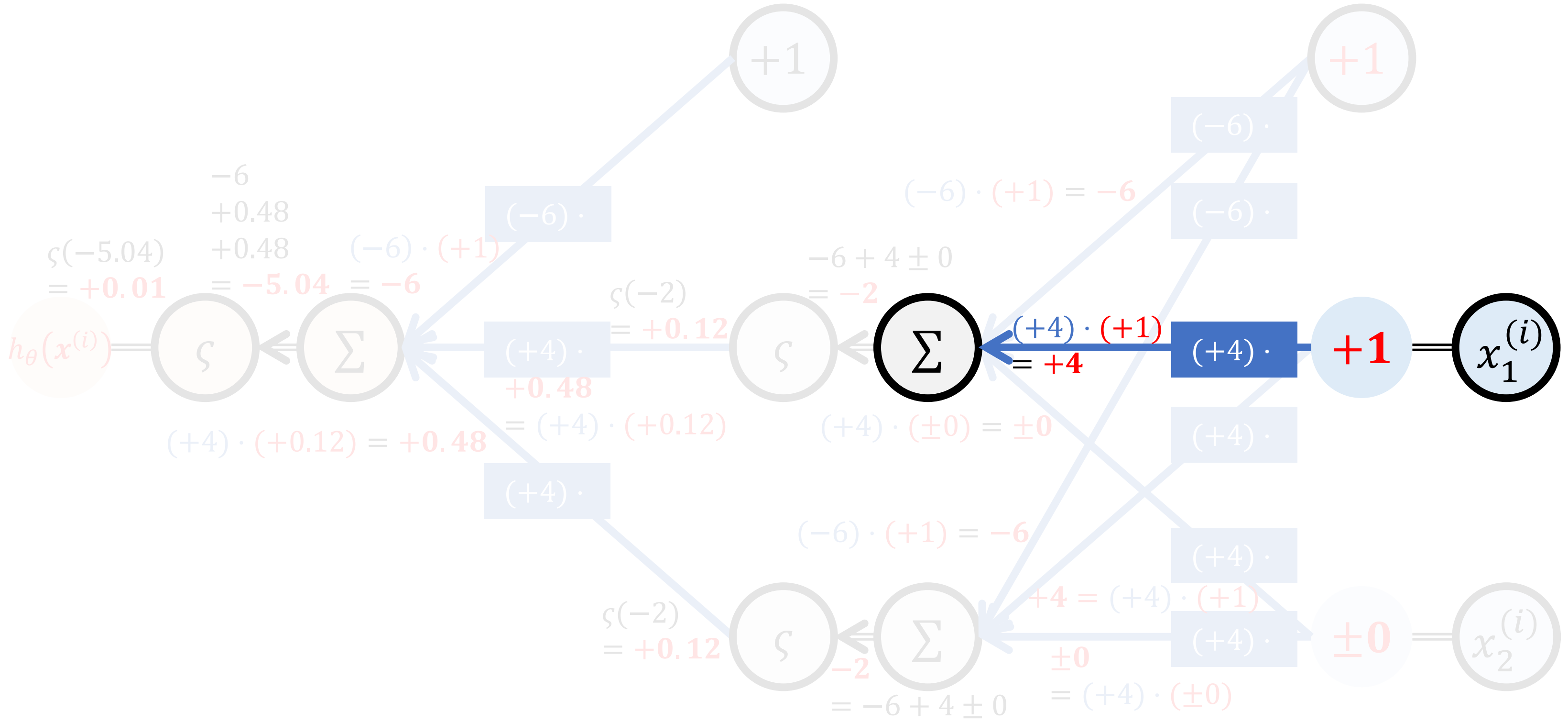
Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters
(activation function)

Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters
(activation function)

Weighted Edge
- Multiplication by a learnable parameter w

Unweighted Edge
- Direct input

f's output equals c.
c is defined as f's output.



Example: AND function

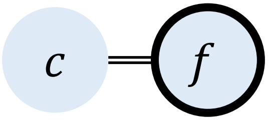
Input node
- Has no incoming edges
- Sends the input

Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

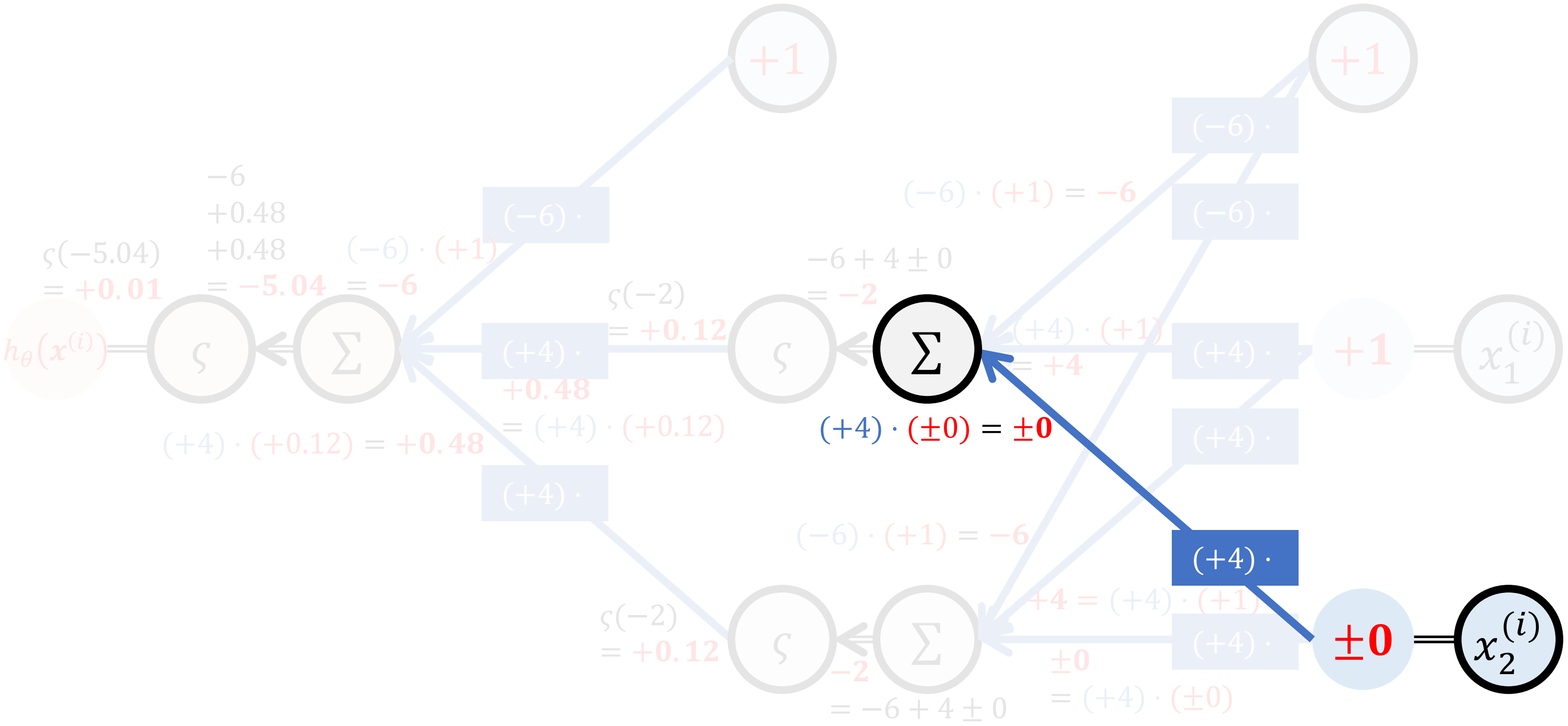
Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)

Weighted Edge
- Multiplication by a learnable parameter w

Unweighted Edge
- Direct input



f's output equals c.
c is defined as f's output.

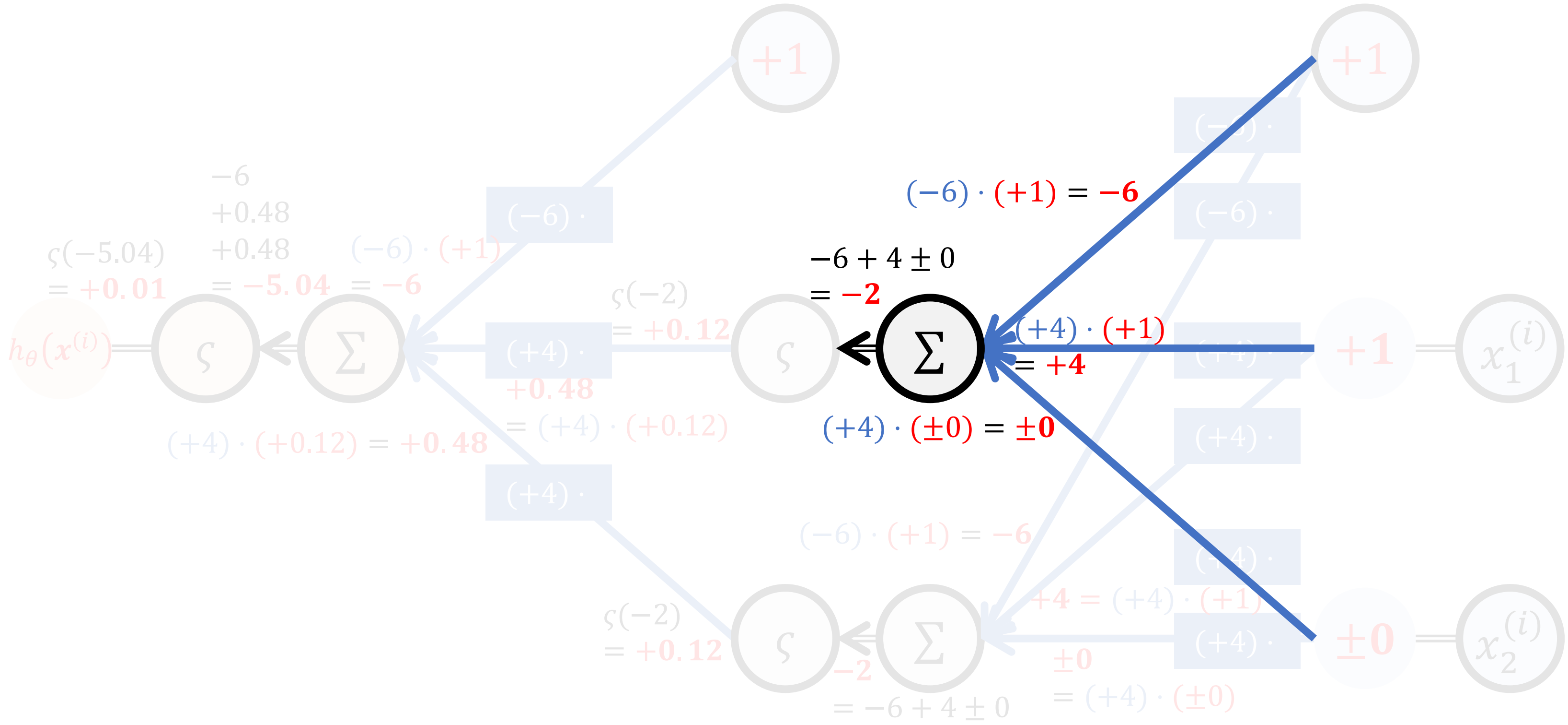


Example: AND function

Input node
- Has no incoming edges
- Sends the input

Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)



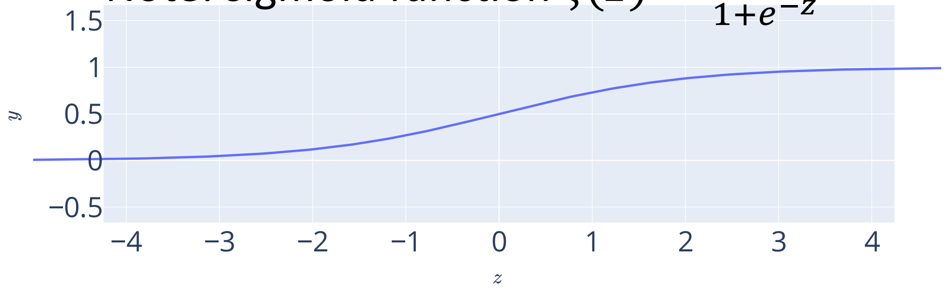
Weighted Edge
- Multiplication by a learnable parameter w

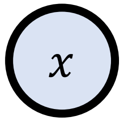
Unweighted Edge
- Direct input

f's output equals c.
c is defined as f's output.

Example: AND function

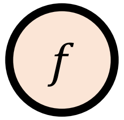
Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$





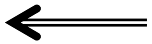
Input node
- Has no incoming edges
- Sends the input

Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

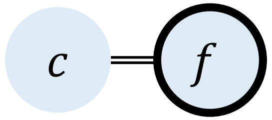


Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)

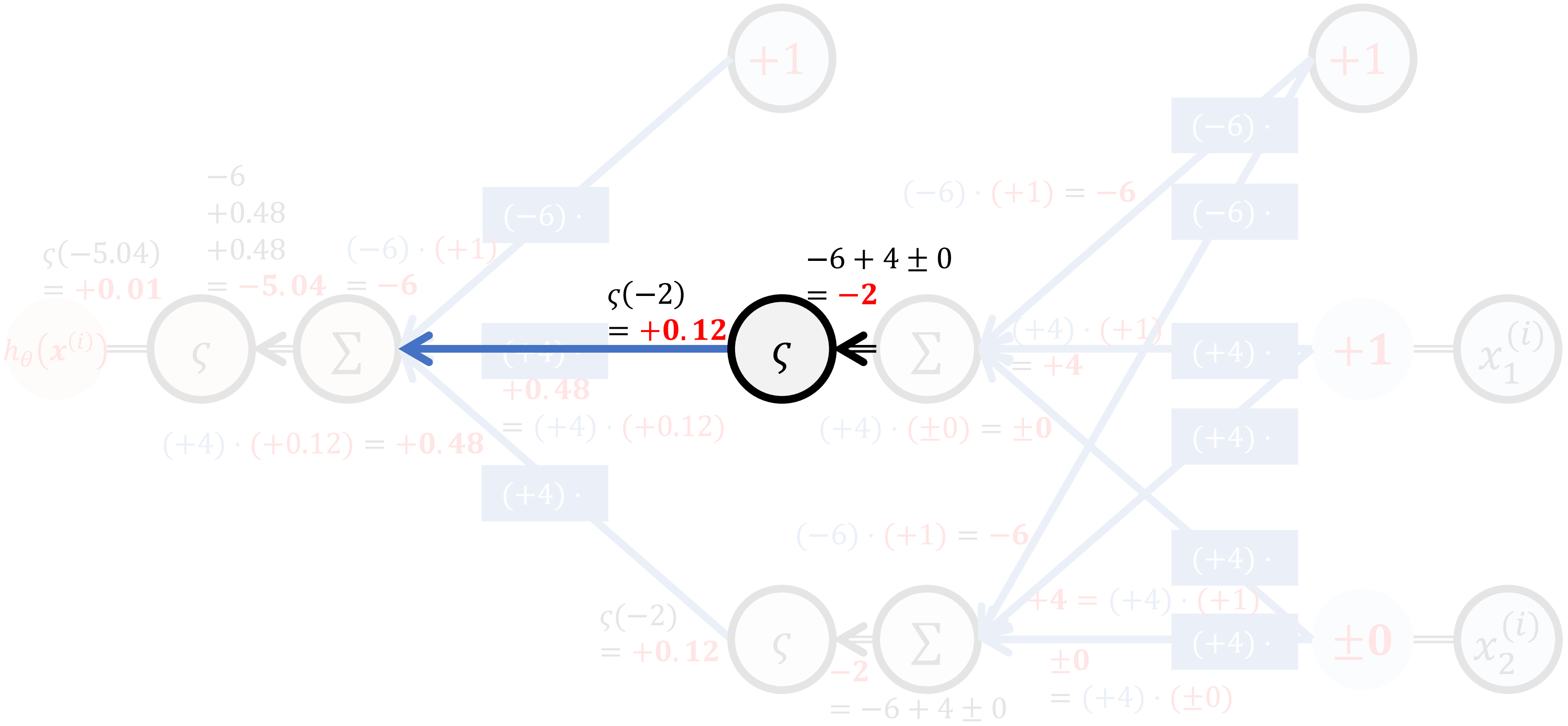
Weighted Edge
- Multiplication by a learnable parameter w



Unweighted Edge
- Direct input



f's output equals c.
c is defined as f's output.



Example: AND function

x Input node
- Has no incoming edges
- Sends the input

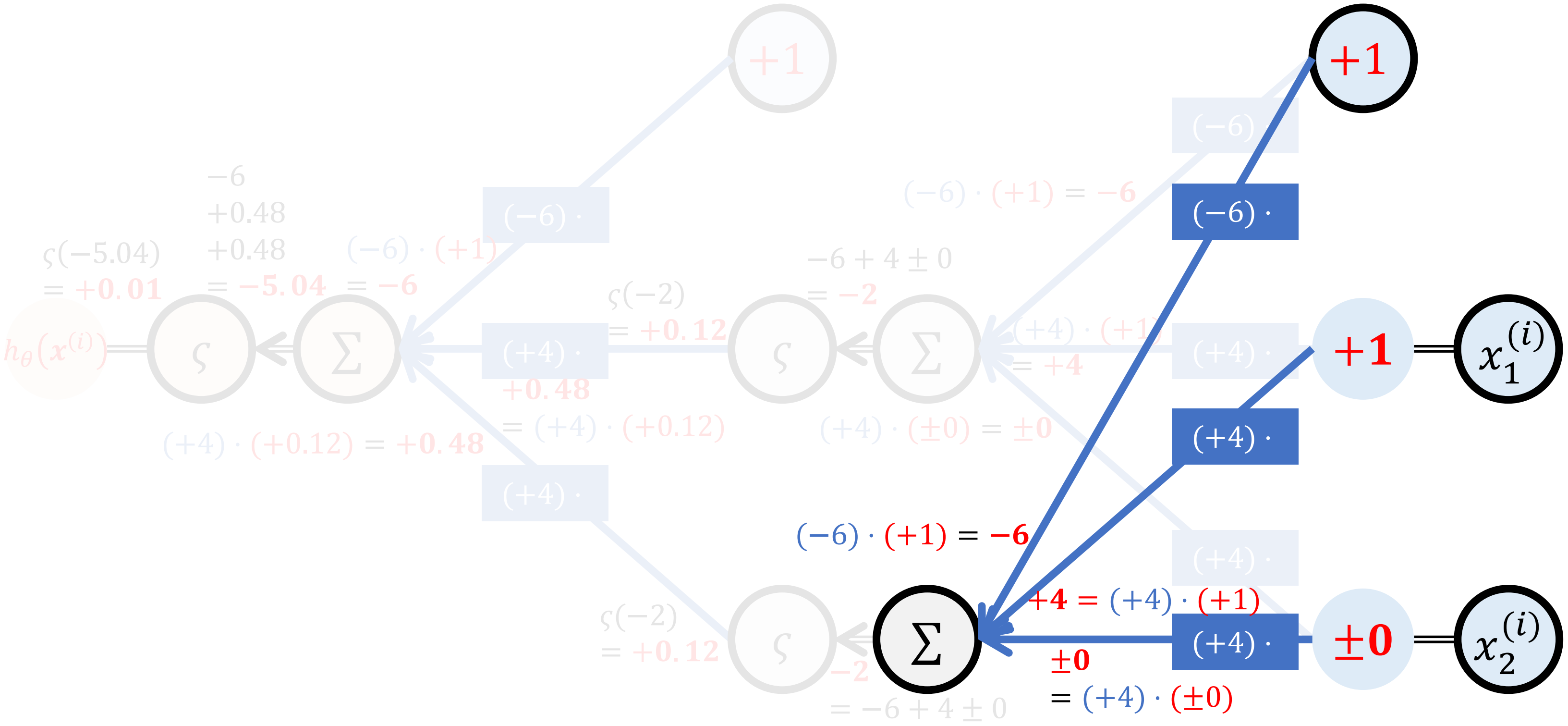
f Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

f Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)

$w\cdot$ Weighted Edge
- Multiplication by a learnable parameter w

Unweighted Edge
- Direct input

c — f: f's output equals c. c is defined as f's output.

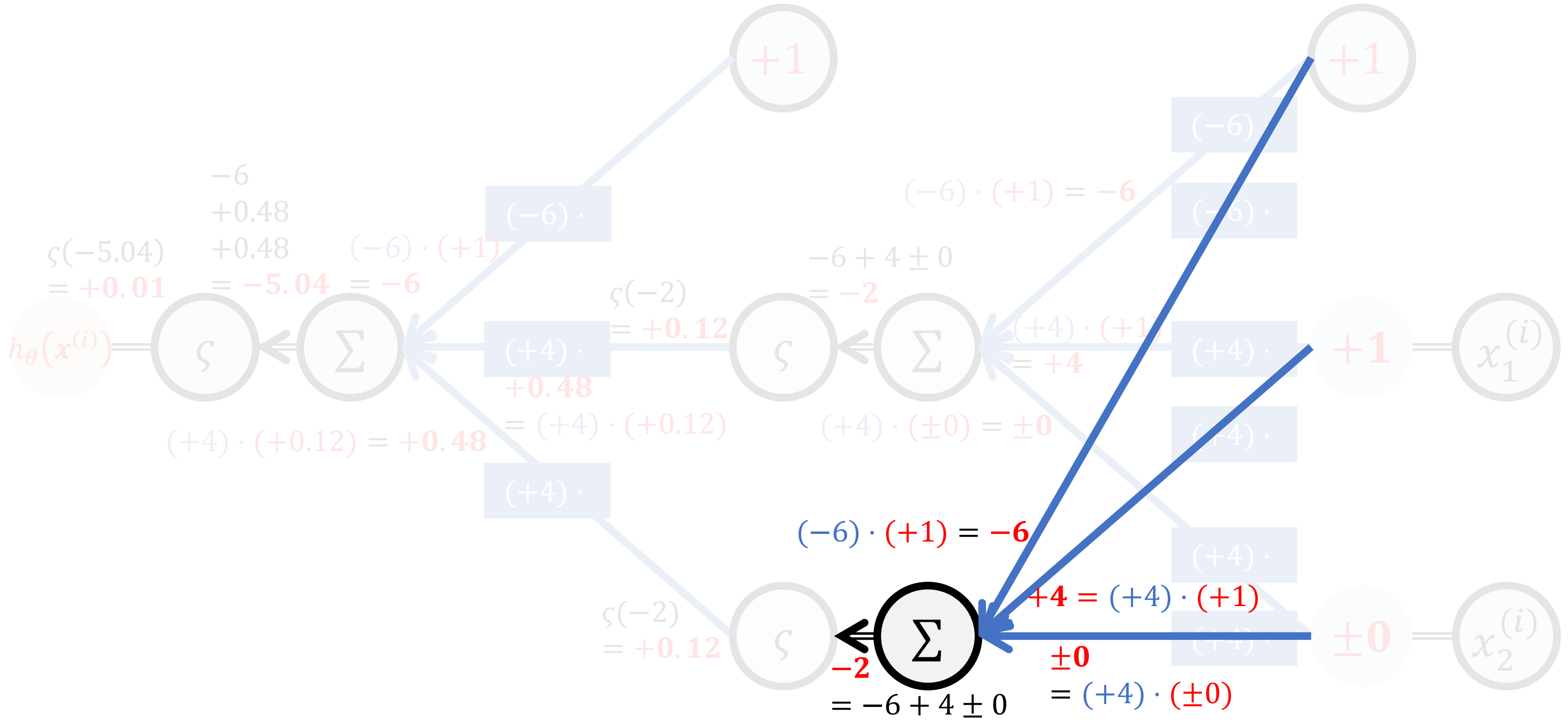


Example: AND function

Input node
- Has no incoming edges
- Sends the input

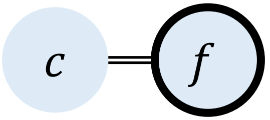
Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)



Weighted Edge
- Multiplication by a learnable parameter w

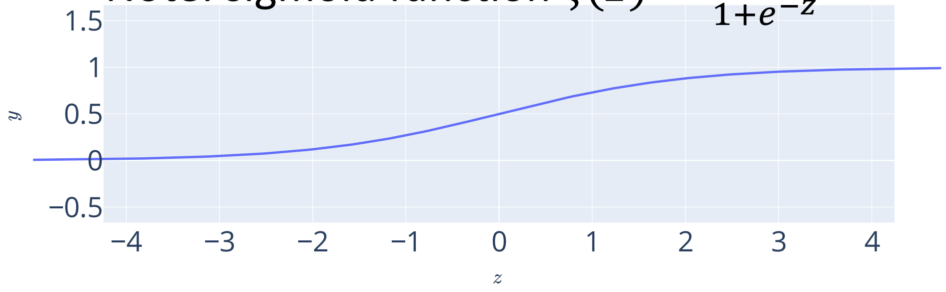
Unweighted Edge
- Direct input

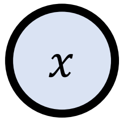


f's output equals c.
c is defined as f's output.

Example: AND function

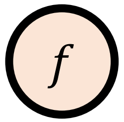
Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$





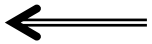
Input node
- Has no incoming edges
- Sends the input

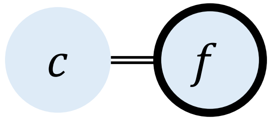
Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)



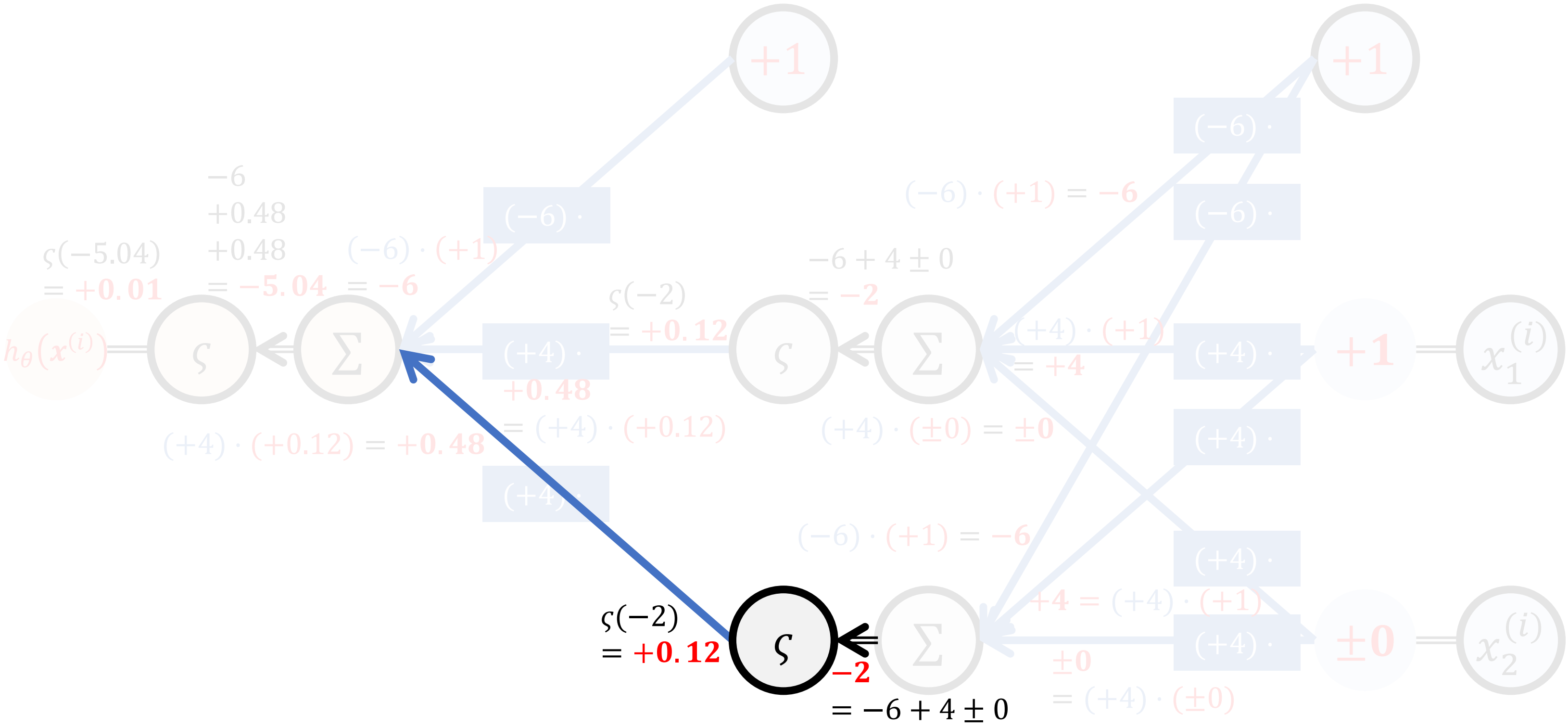
Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)

Weighted Edge
- Multiplication by a learnable parameter w


Unweighted Edge
- Direct input



f's output equals c.
c is defined as f's output.



Example: AND function

Input node
- Has no incoming edges
- Sends the input

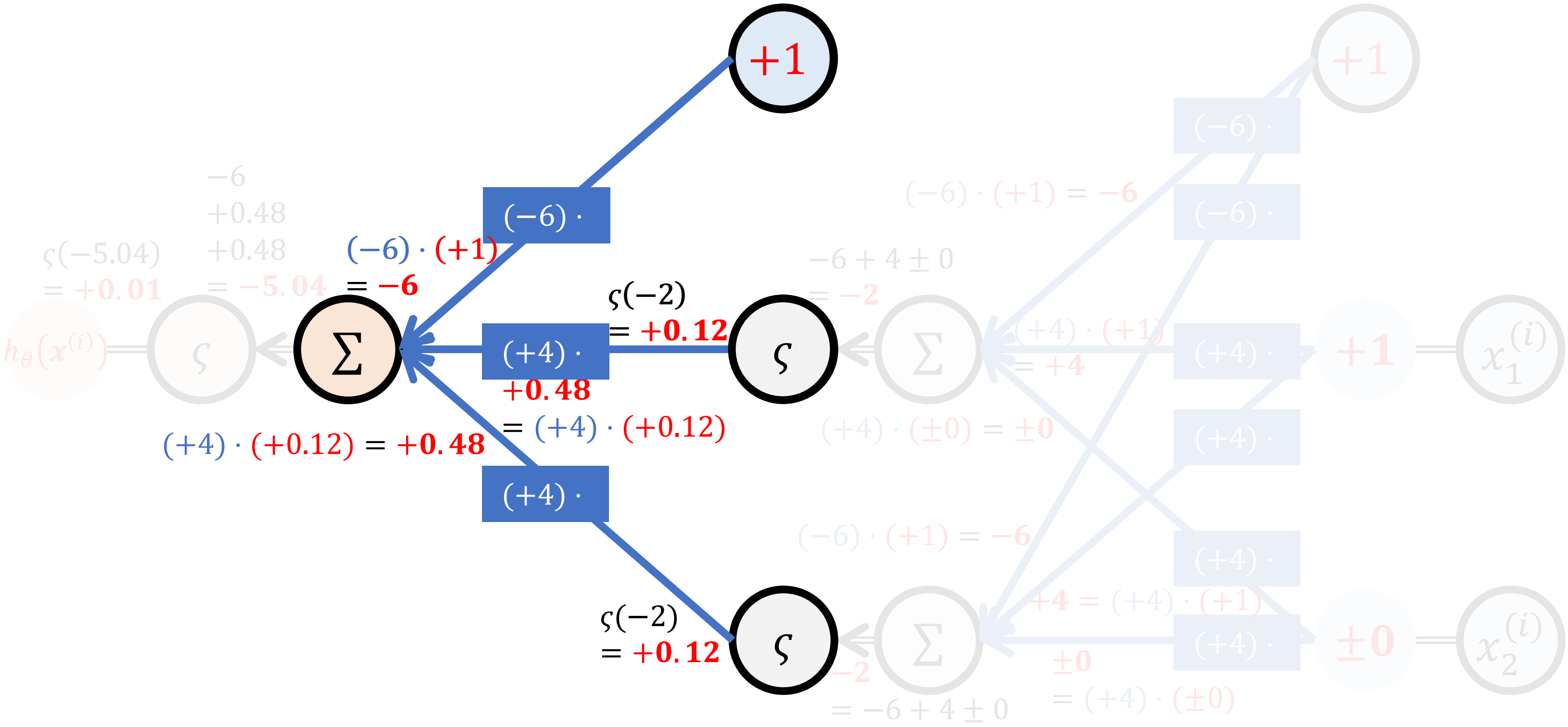
Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)

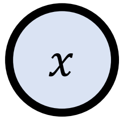
Weighted Edge
- Multiplication by a learnable parameter w

Unweighted Edge
- Direct input

f's output equals c.
c is defined as f's output.

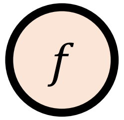


Example: AND function

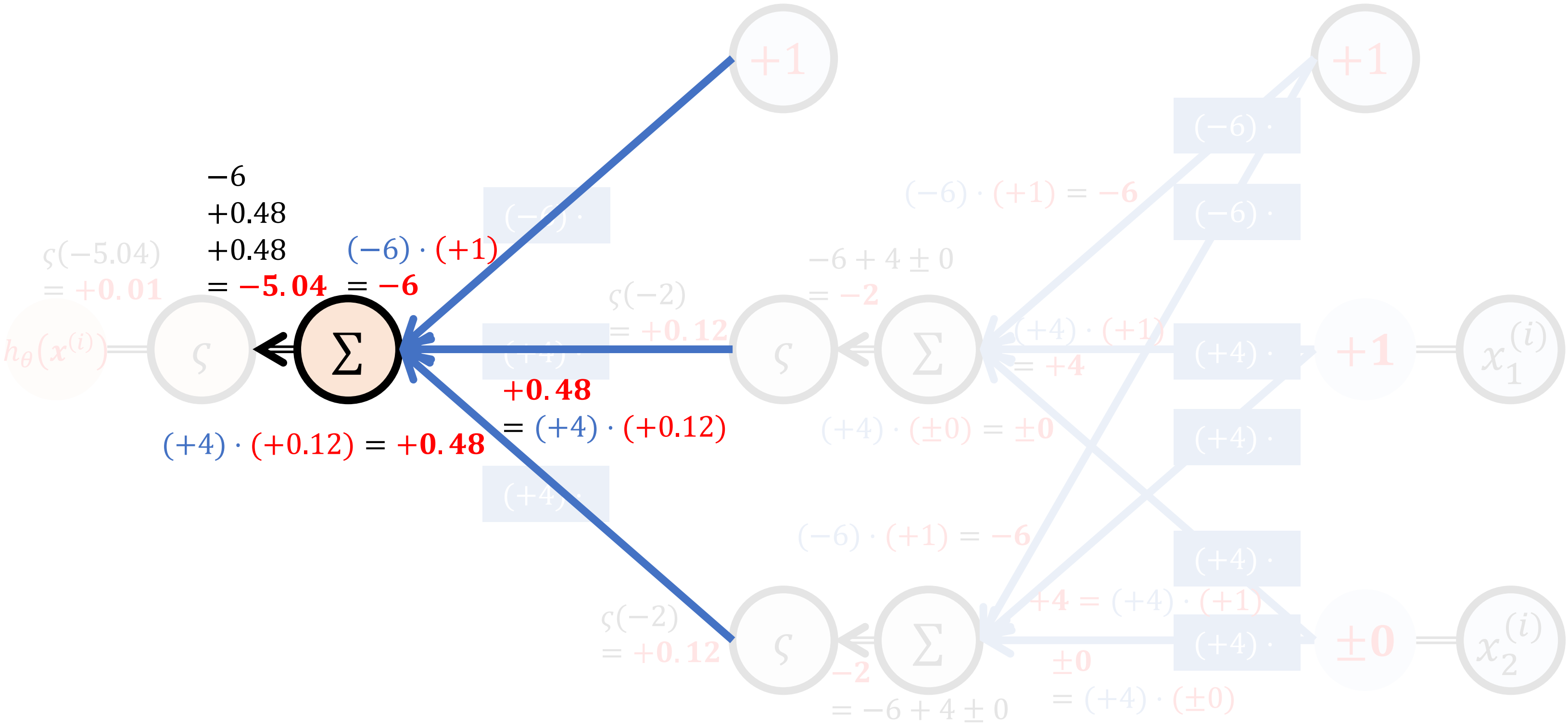


Input node
- Has no incoming edges
- Sends the input

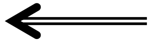
Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)



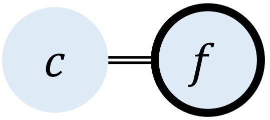
Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)



Weighted Edge
- Multiplication by a learnable parameter w

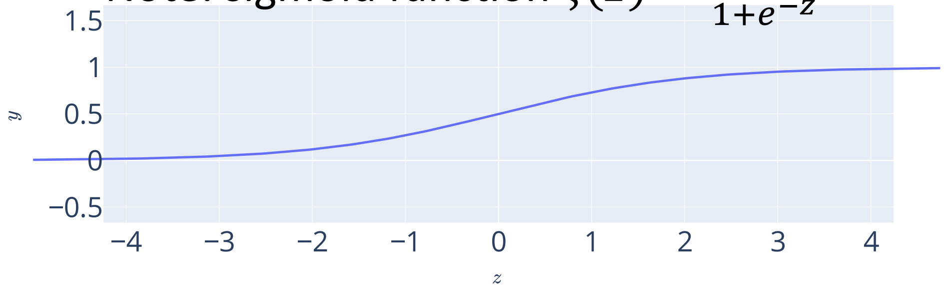


Unweighted Edge
- Direct input



f's output equals c.
c is defined as f's output.

Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$

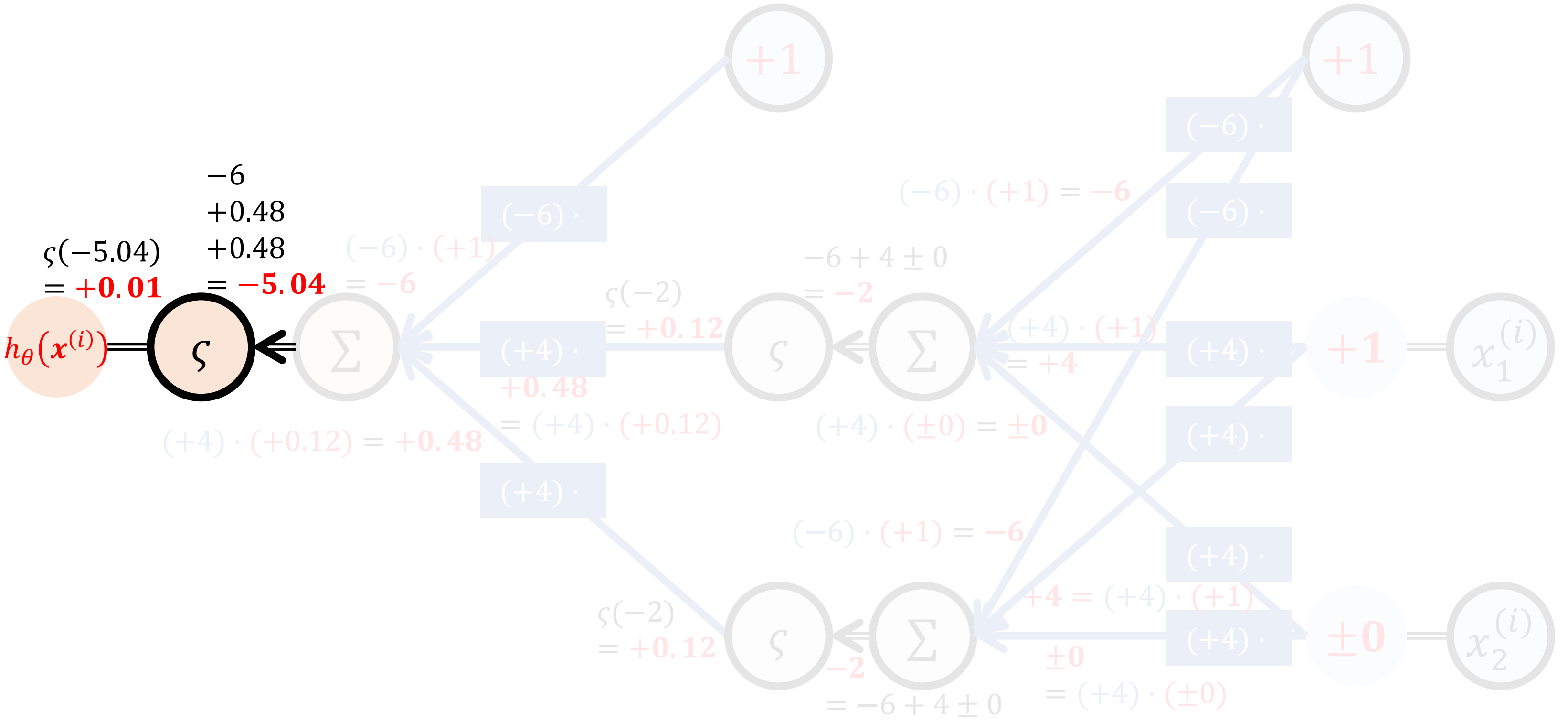


Example: AND function

Input node
- Has no incoming edges
- Sends the input

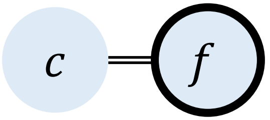
Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)



Weighted Edge
- Multiplication by a learnable parameter w

Unweighted Edge
- Direct input



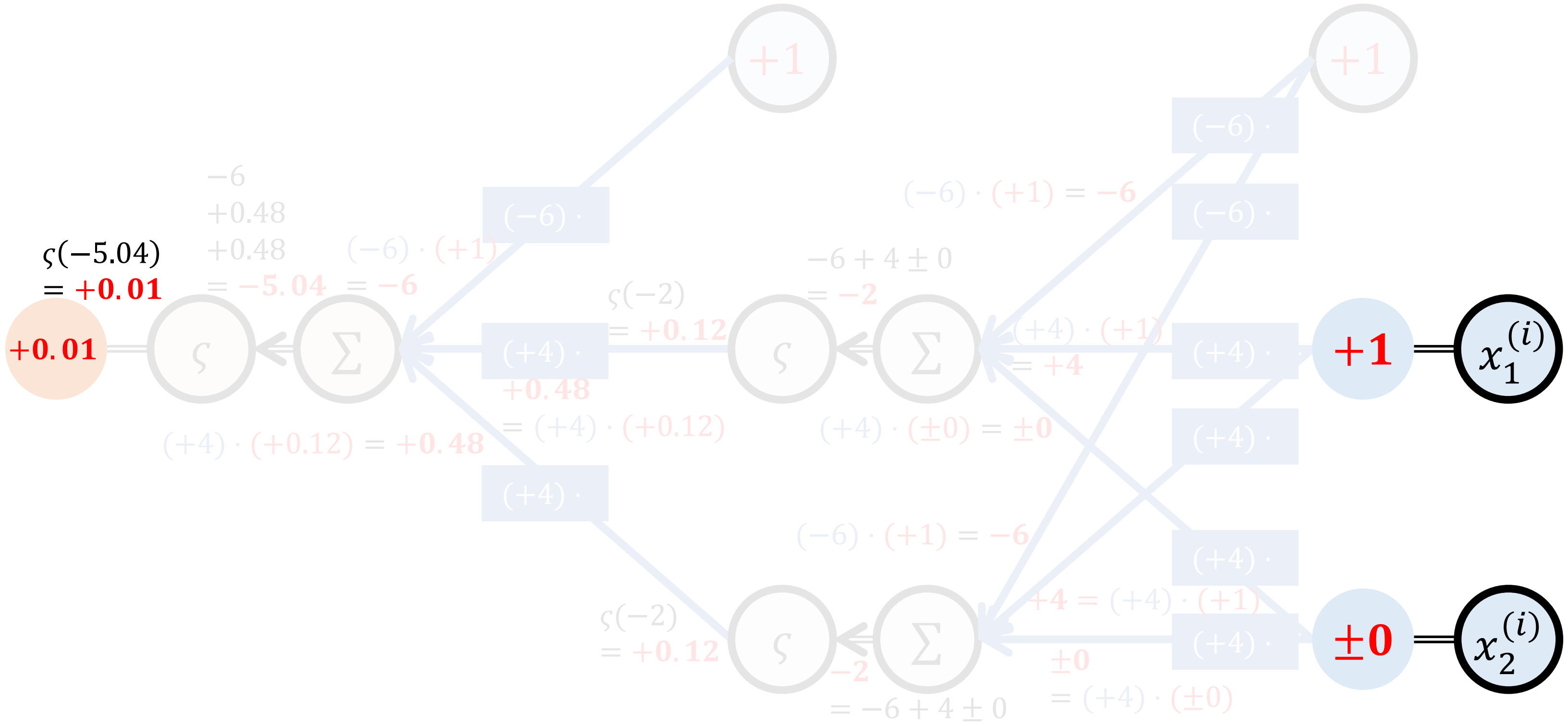
f's output equals c.
c is defined as f's output.

Example: AND function

Input node
- Has no incoming edges
- Sends the input

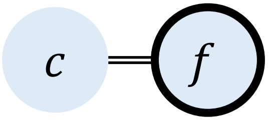
Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters
(activation function)

Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters
(activation function)



Weighted Edge
- Multiplication by a learnable parameter w

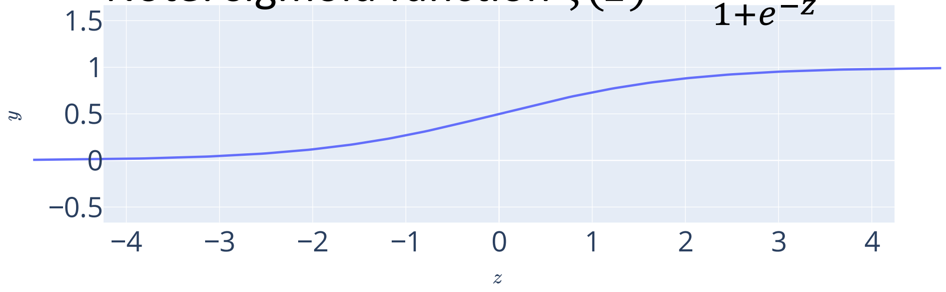
Unweighted Edge
- Direct input



f's output equals c.
c is defined as f's output.

Example: AND function

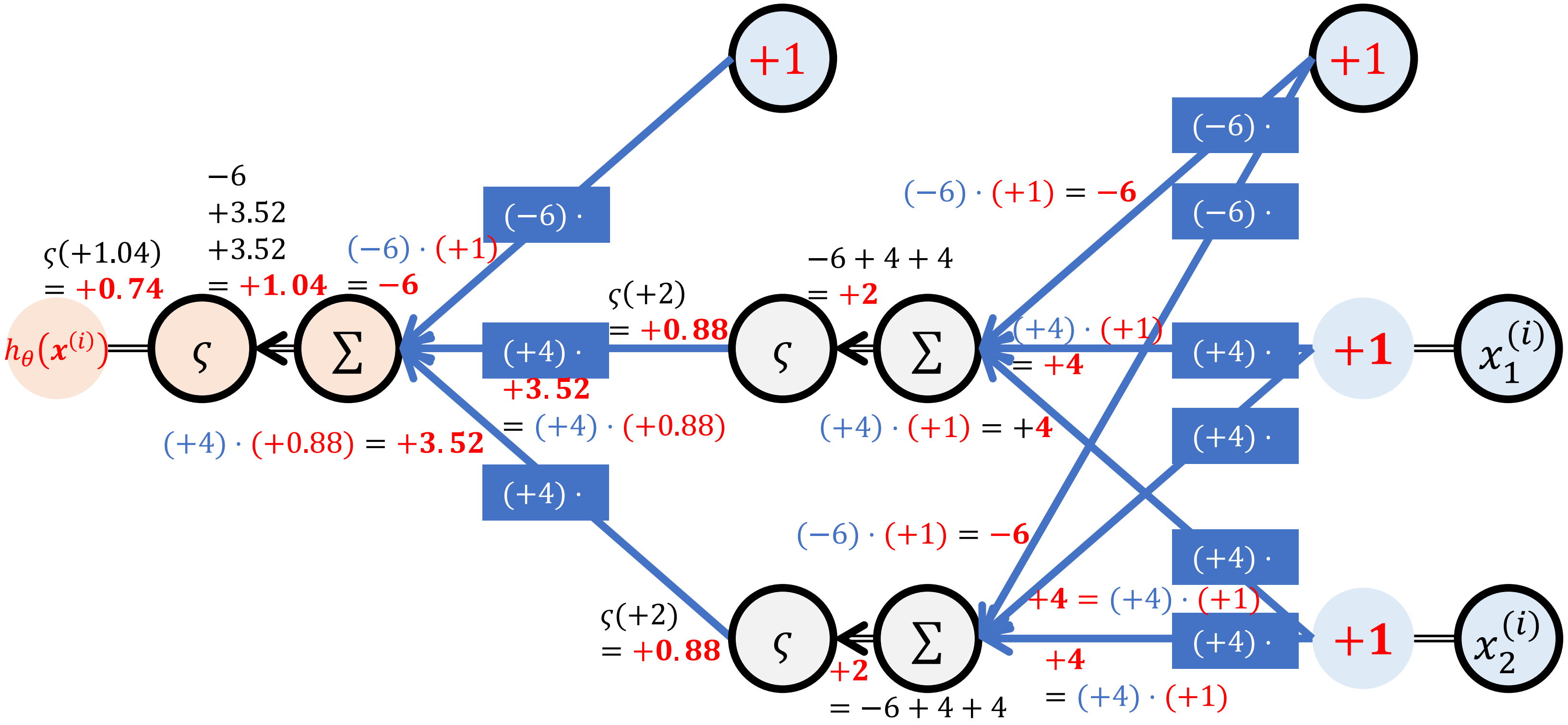
Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$



x Input node
- Has no incoming edges
- Sends the input

f Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

f Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)

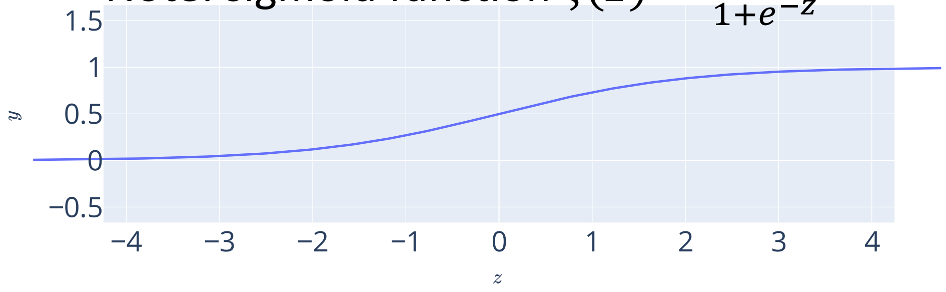


$w\,\cdot$ Weighted Edge
- Multiplication by a learnable parameter w

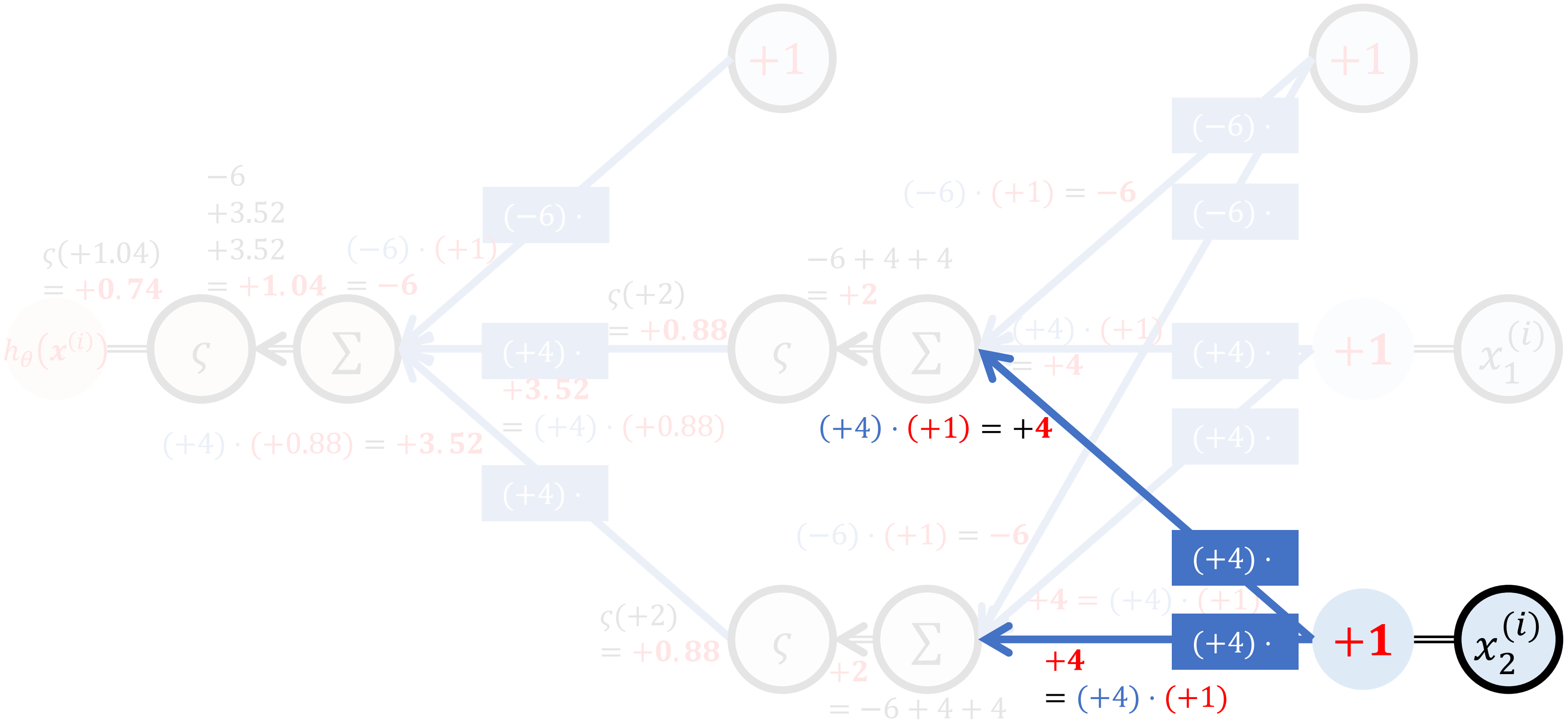
Unweighted Edge
- Direct input

c = f : f's output equals c. c is defined as f's output.

Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$



Example: AND function



x Input node
- Has no incoming edges
- Sends the input

f Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

f Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)

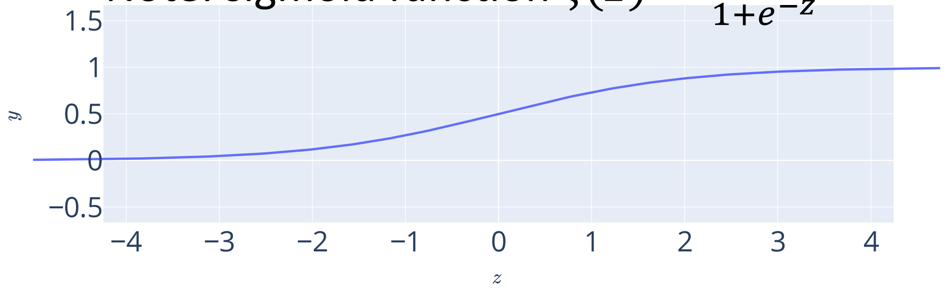
$w \cdot$ Weighted Edge
- Multiplication by a learnable parameter w

Unweighted Edge
- Direct input

c f — f's output equals c. c is defined as f's output.

Example: AND function

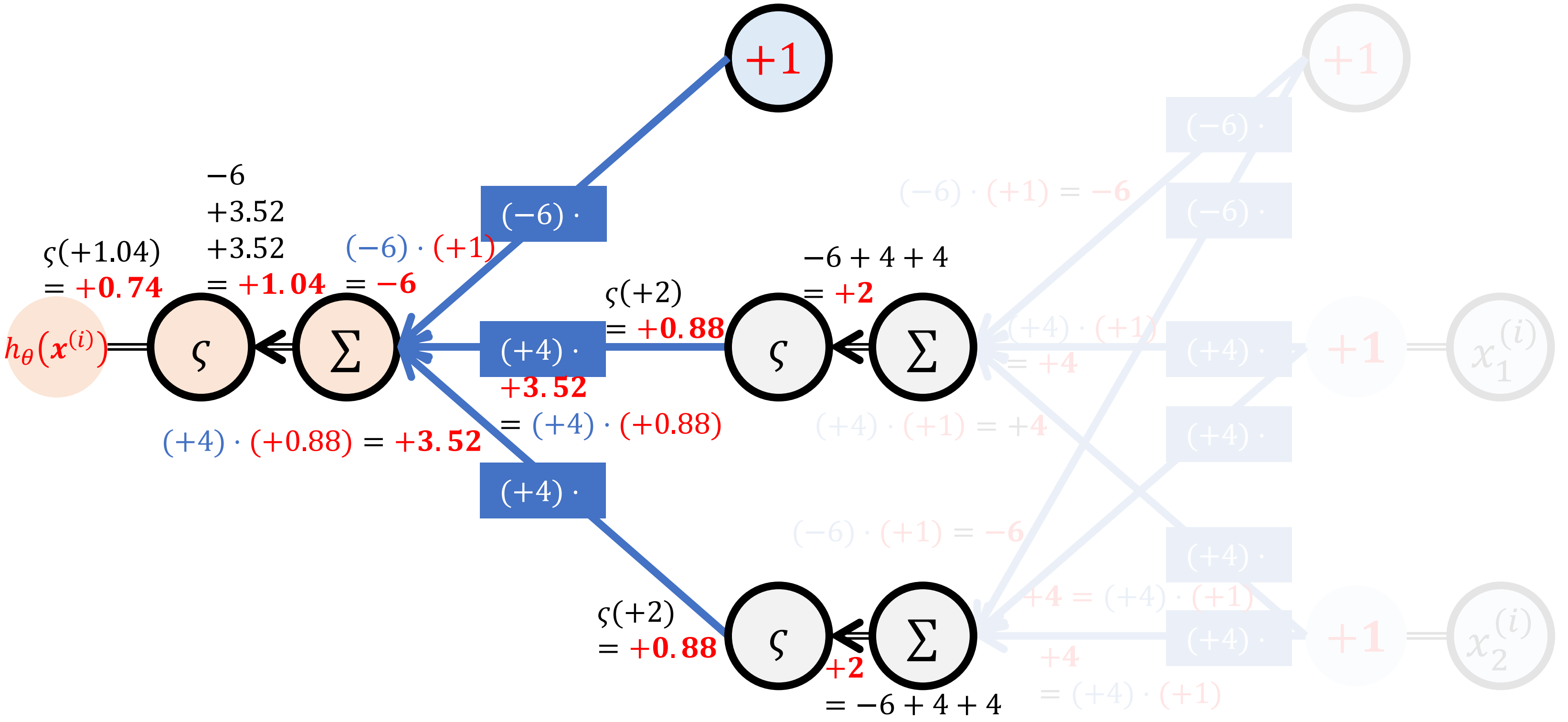
Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$



Input node
- Has no incoming edges
- Sends the input

Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)



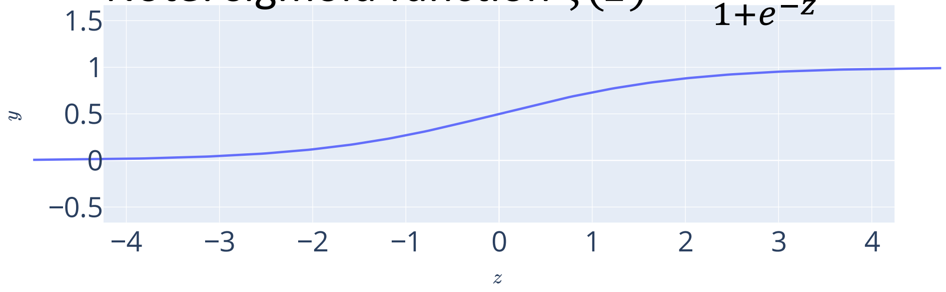
Weighted Edge
- Multiplication by a learnable parameter w

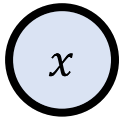
Unweighted Edge
- Direct input

f's output equals c.
c is defined as f's output.

Example: AND function

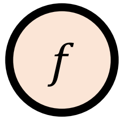
Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$



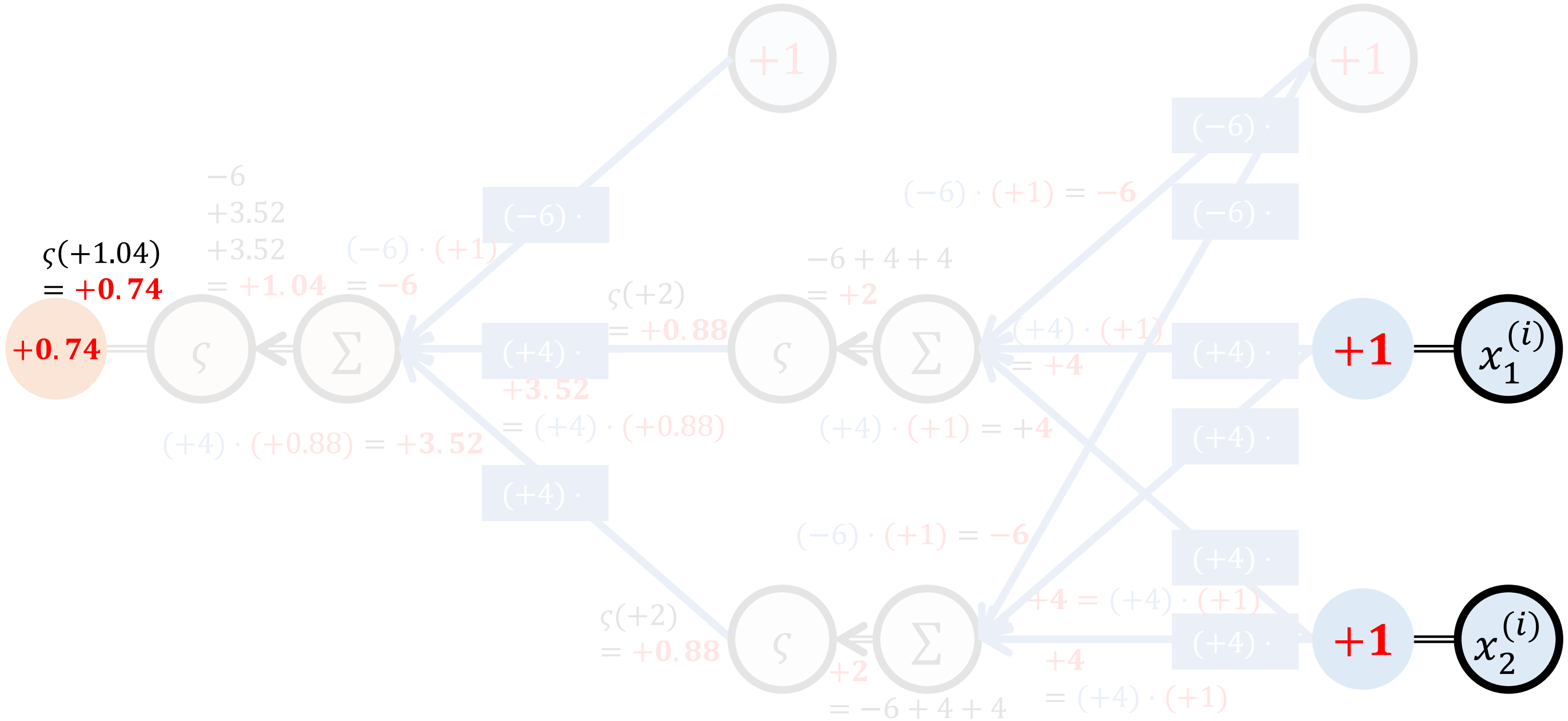


Input node
- Has no incoming edges
- Sends the input

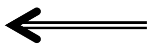
Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)



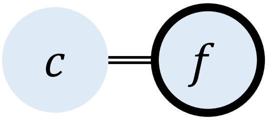
Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)



Weighted Edge
- Multiplication by a learnable parameter w



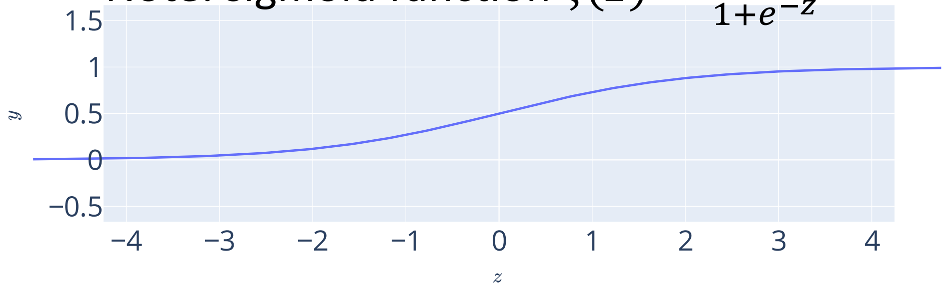
Unweighted Edge
- Direct input



f's output equals c.
c is defined as f's output.

Example: AND function

Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$



Input node
- Has no incoming edges
- Sends the input

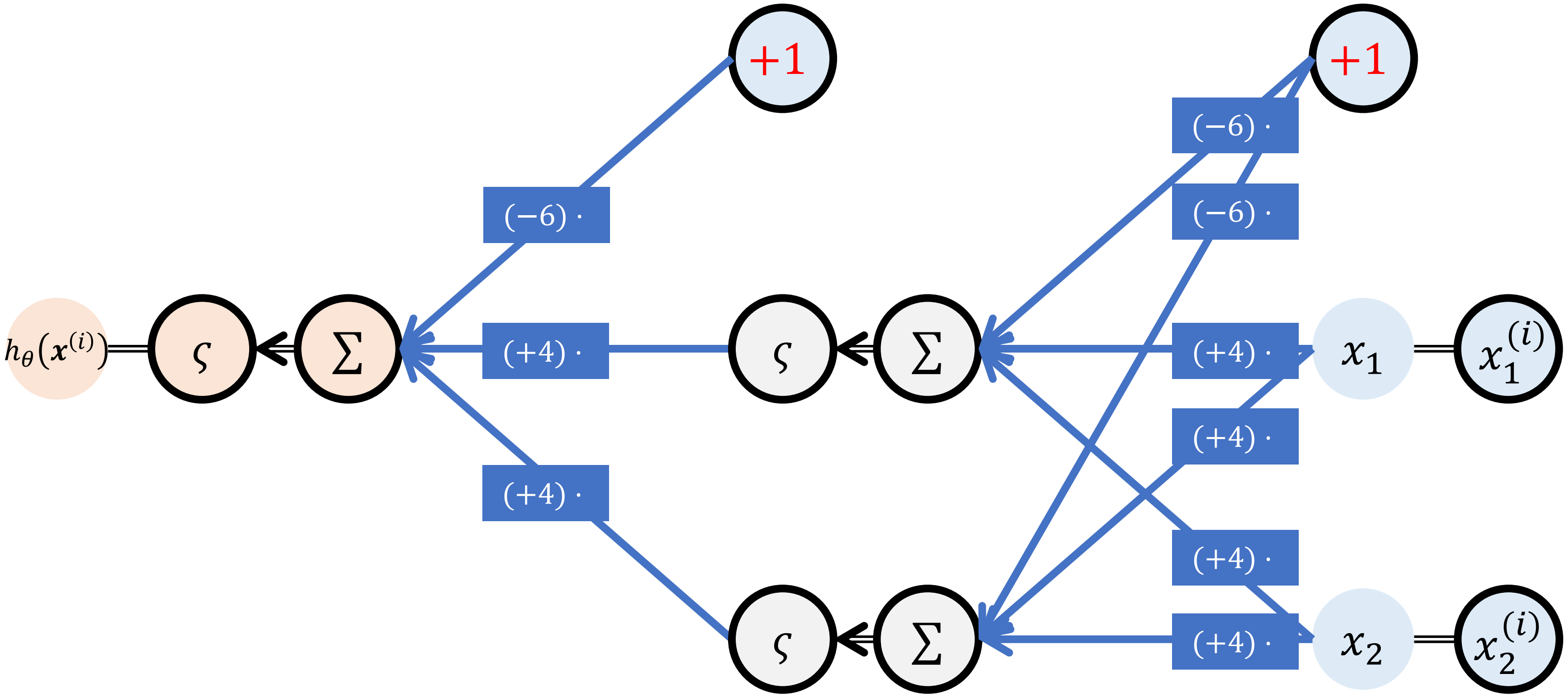
Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)

Weighted Edge
- Multiplication by a learnable parameter w

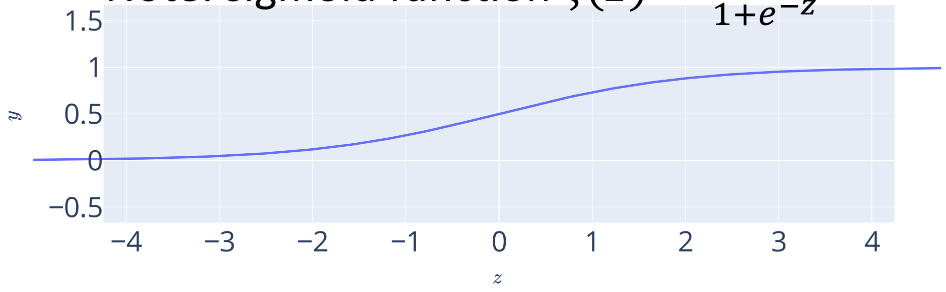
Unweighted Edge
- Direct input

f's output equals c.
c is defined as f's output.

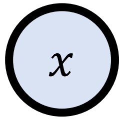


$x_1^{(i)}$	$x_2^{(i)}$	AND	$h_\theta(\boldsymbol{x}^{(i)})$
0	0	**0**	**+0.00**
0	1	**0**	**+0.01**
1	0	**0**	**+0.01**
1	1	**1**	**+0.74**

Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$

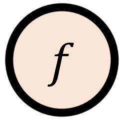


Example: XOR function



Input node
- Has no incoming edges
- Sends the input

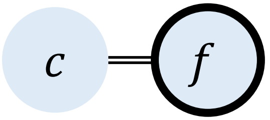
Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)



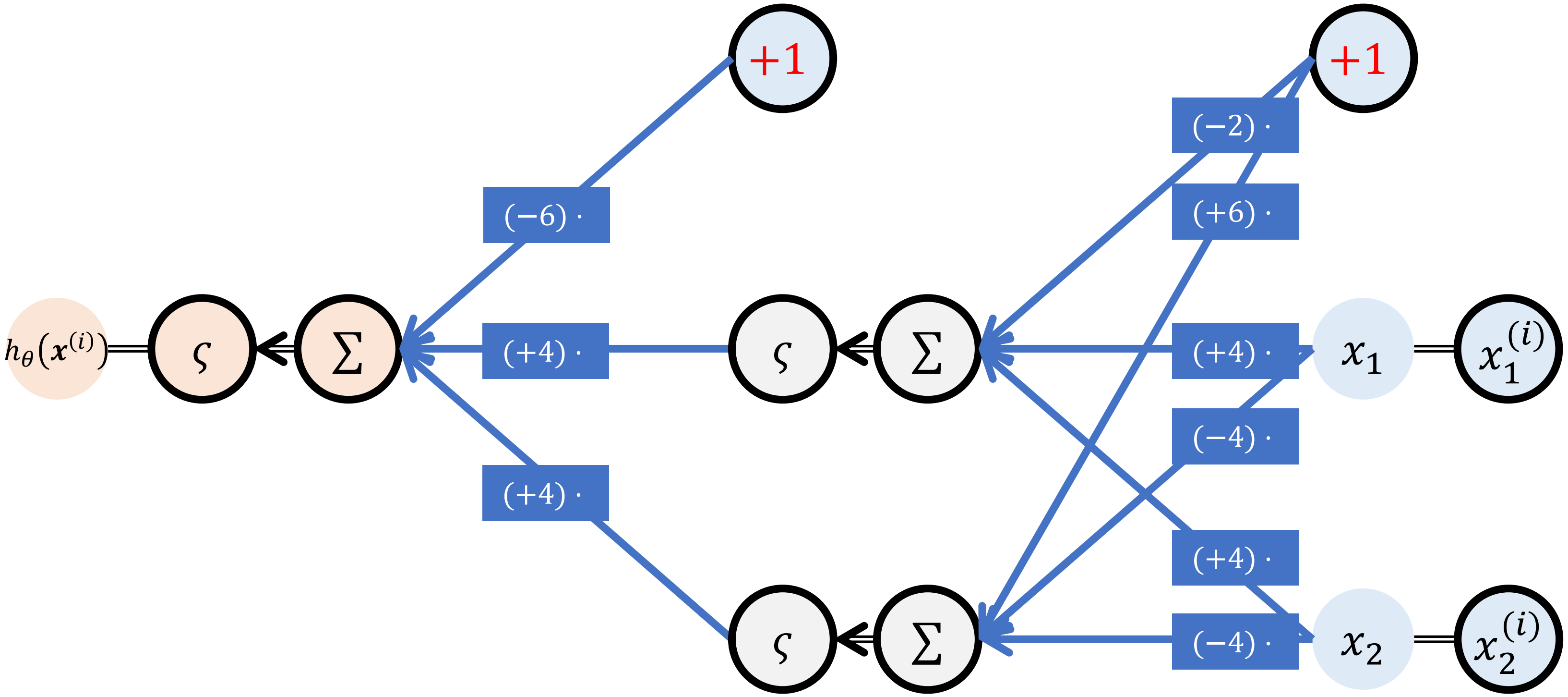
Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)

Weighted Edge
- Multiplication by a learnable parameter w

Unweighted Edge
- Direct input



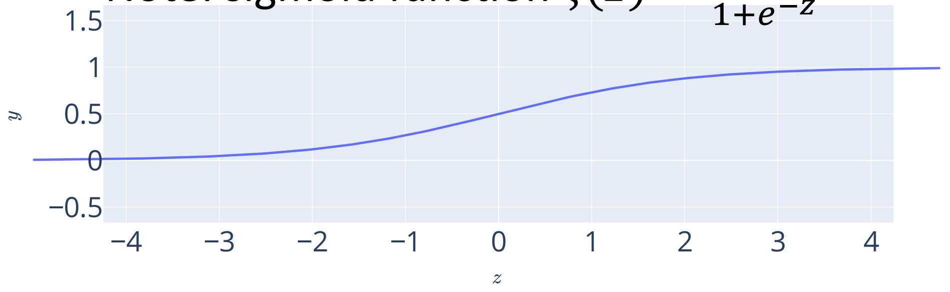
f's output equals c.
c is defined as f's output.



$x_1^{(i)}$	$x_2^{(i)}$	XOR
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Example: XOR function

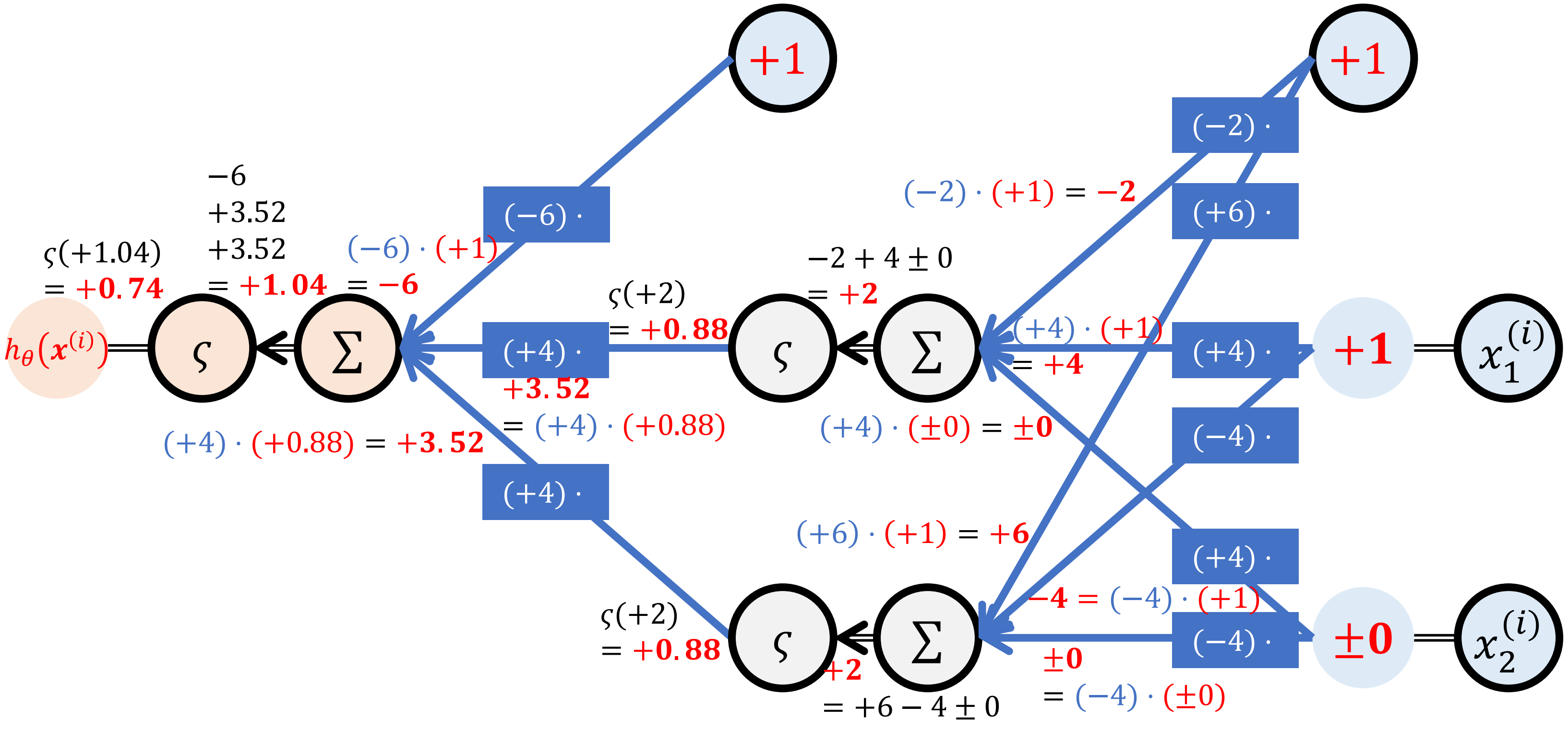
Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$



Input node
- Has no incoming edges
- Sends the input

Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)



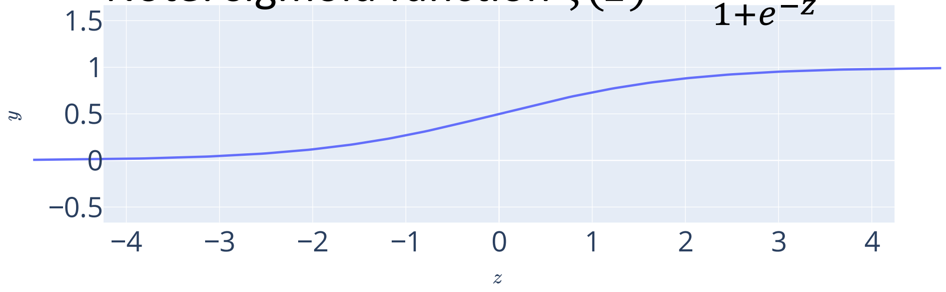
Weighted Edge
- Multiplication by a learnable parameter w

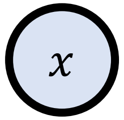
Unweighted Edge
- Direct input

f's output equals c.
c is defined as f's output.

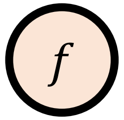
Example: XOR function

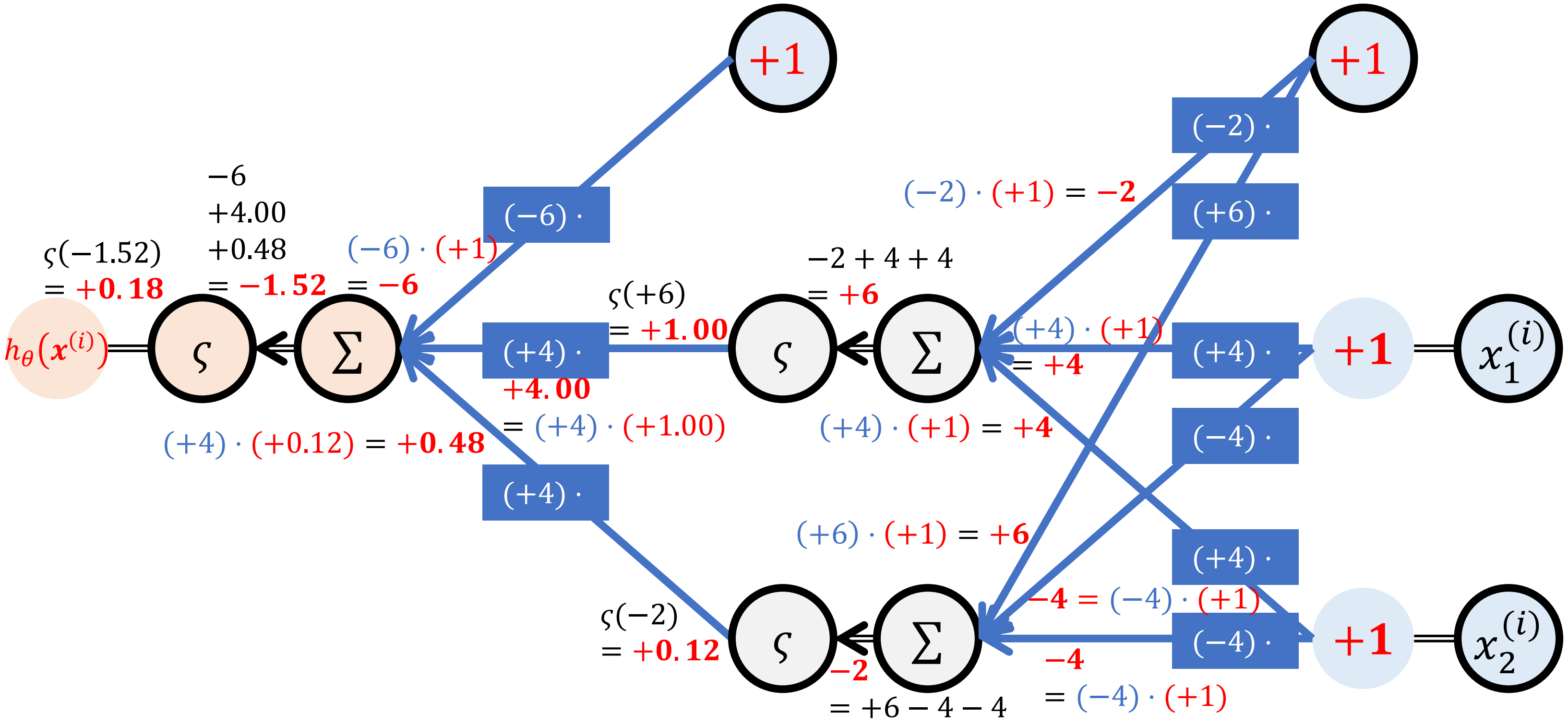
Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$



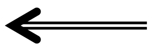

Input node
- Has no incoming edges
- Sends the input

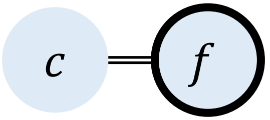

Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)


Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)



Weighted Edge
- Multiplication by a learnable parameter w

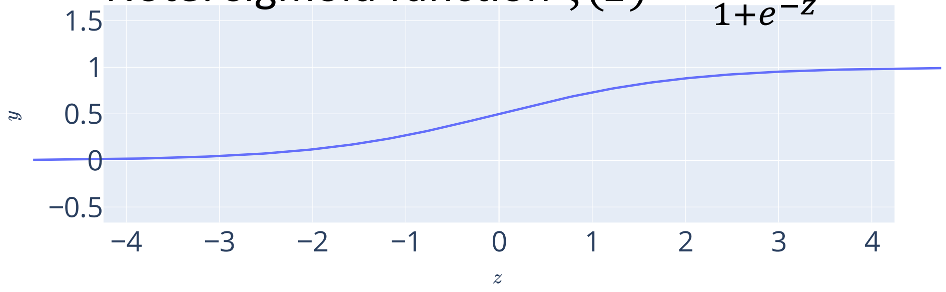

Unweighted Edge
- Direct input



f's output equals c.
c is defined as f's output.

Example: XOR function

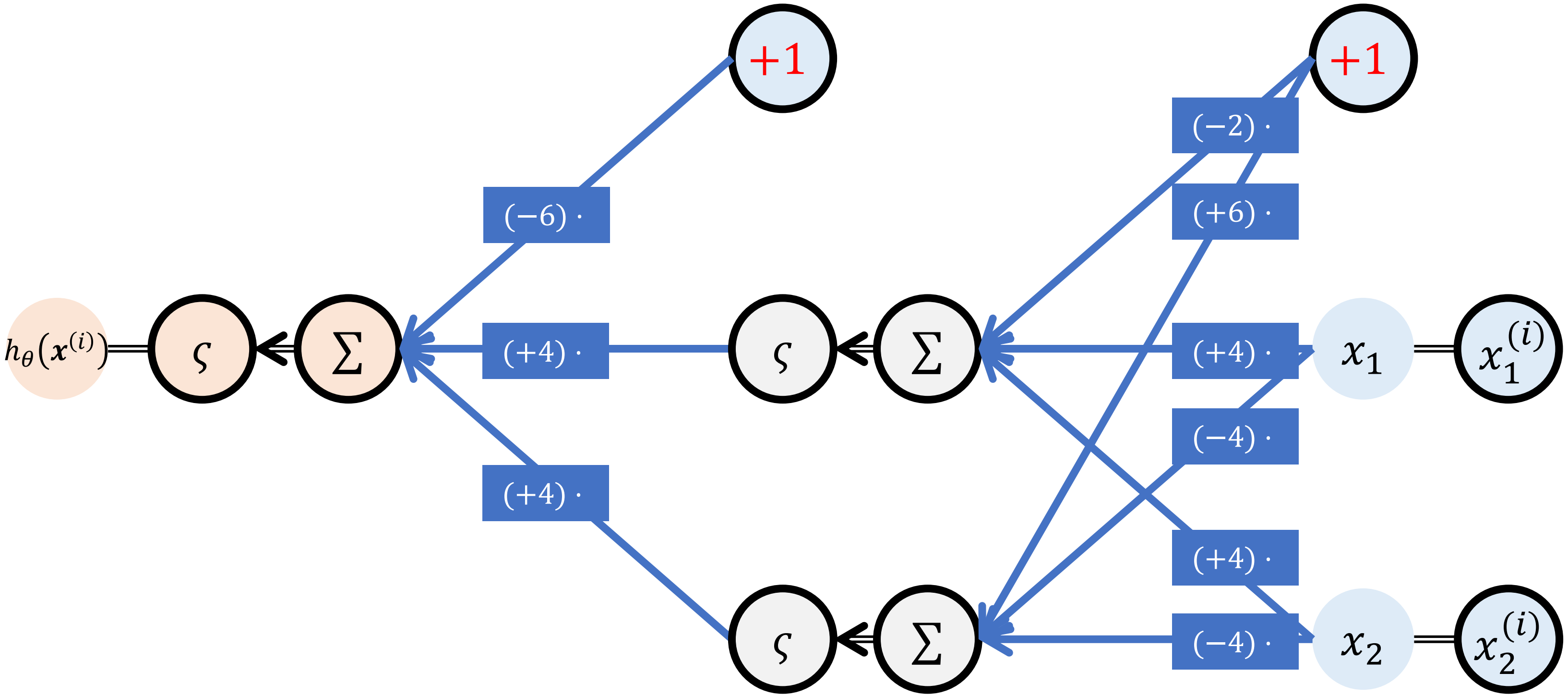
Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$



Input node
- Has no incoming edges
- Sends the input

Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)



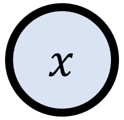
$x_1^{(i)}$	$x_2^{(i)}$	XOR	$h_\theta(\boldsymbol{x}^{(i)})$
0	0	**0**	**+0.18**
0	1	**1**	**+0.74**
1	0	**1**	**+0.74**
1	1	**0**	**+0.18**

Weighted Edge
- Multiplication by a learnable parameter w

Unweighted Edge
- Direct input

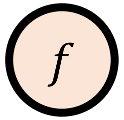
f's output equals c.
c is defined as f's output.

Linear regression as a NN



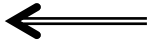
Input node
- Has no incoming edges
- Sends the input

Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

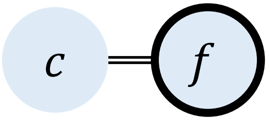


Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)

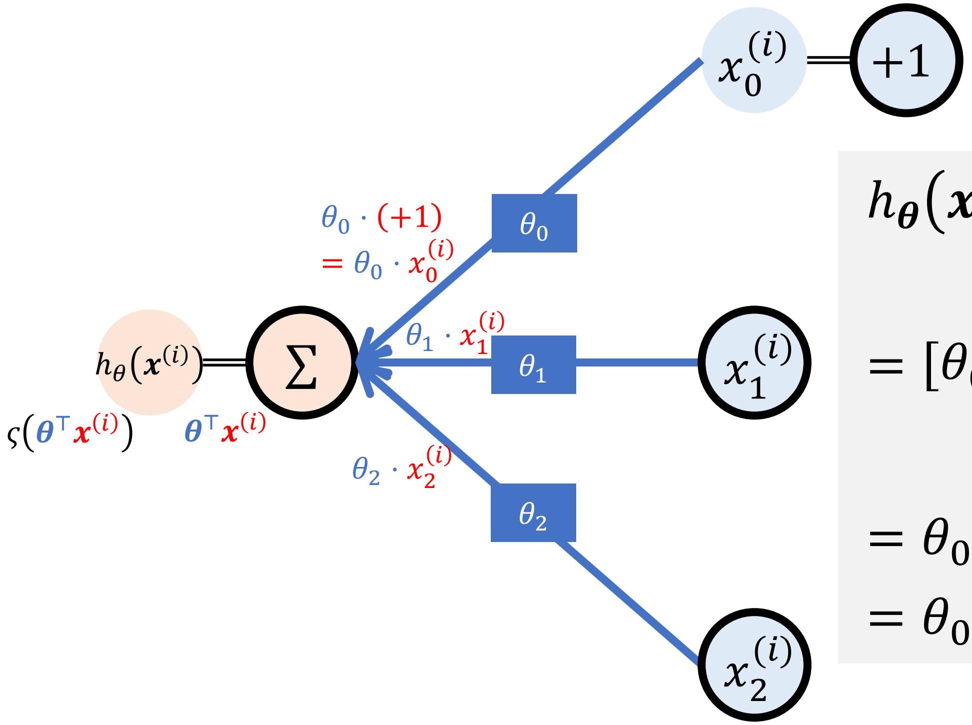
Weighted Edge
- Multiplication by a learnable parameter w



Unweighted Edge
- Direct input



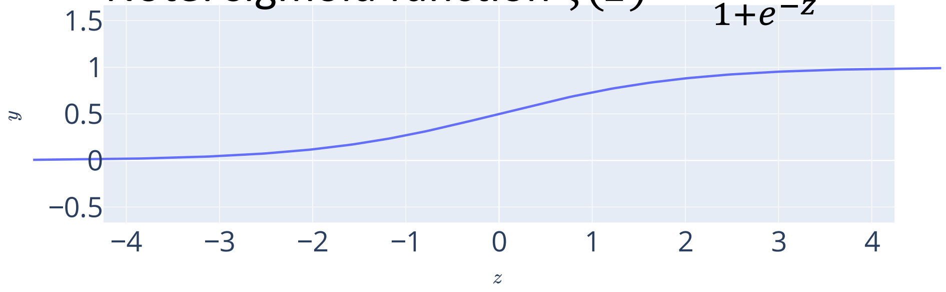
f's output equals c.
c is defined as f's output.



$$
\begin{aligned}
h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) &= \boldsymbol{\theta}^\top \boldsymbol{x}^{(i)} \\
&= \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 \end{bmatrix} \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \end{bmatrix} \\
&= \theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} \\
&= \theta_0 \cdot 1 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)}
\end{aligned}
$$

Logistic regression as a NN

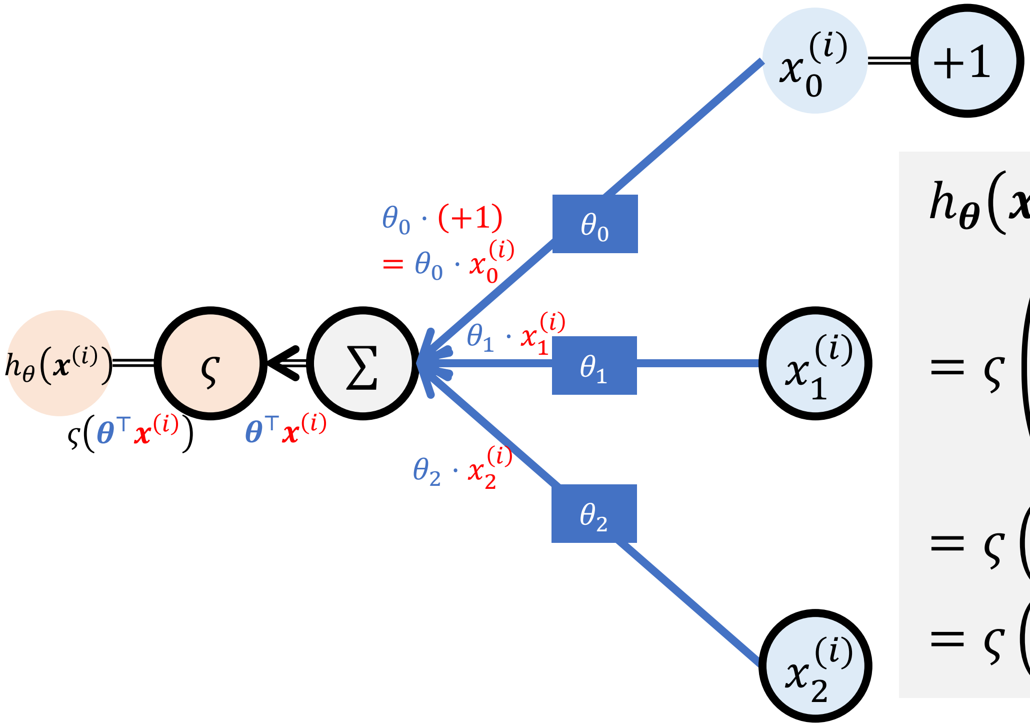
Note: sigmoid function $\varsigma(z) = \frac{1}{1+e^{-z}}$



Input node
- Has no incoming edges
- Sends the input

Hidden node
- Has incoming and outgoing edges
- Fixed function w/o parameters (activation function)

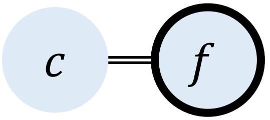
Output node
- No outgoing edges
- Receives the output
- Fixed function w/o parameters (activation function)



$$
\begin{aligned}
h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) &= \varsigma\left(\boldsymbol{\theta}^\top \boldsymbol{x}^{(i)}\right) \\
&= \varsigma\left(\begin{bmatrix}\theta_0 & \theta_1 & \theta_2\end{bmatrix}\begin{bmatrix}x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)}\end{bmatrix}\right) \\
&= \varsigma\left(\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)}\right) \\
&= \varsigma\left(\theta_0 \cdot 1 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)}\right)
\end{aligned}
$$

Weighted Edge
- Multiplication by a learnable parameter w

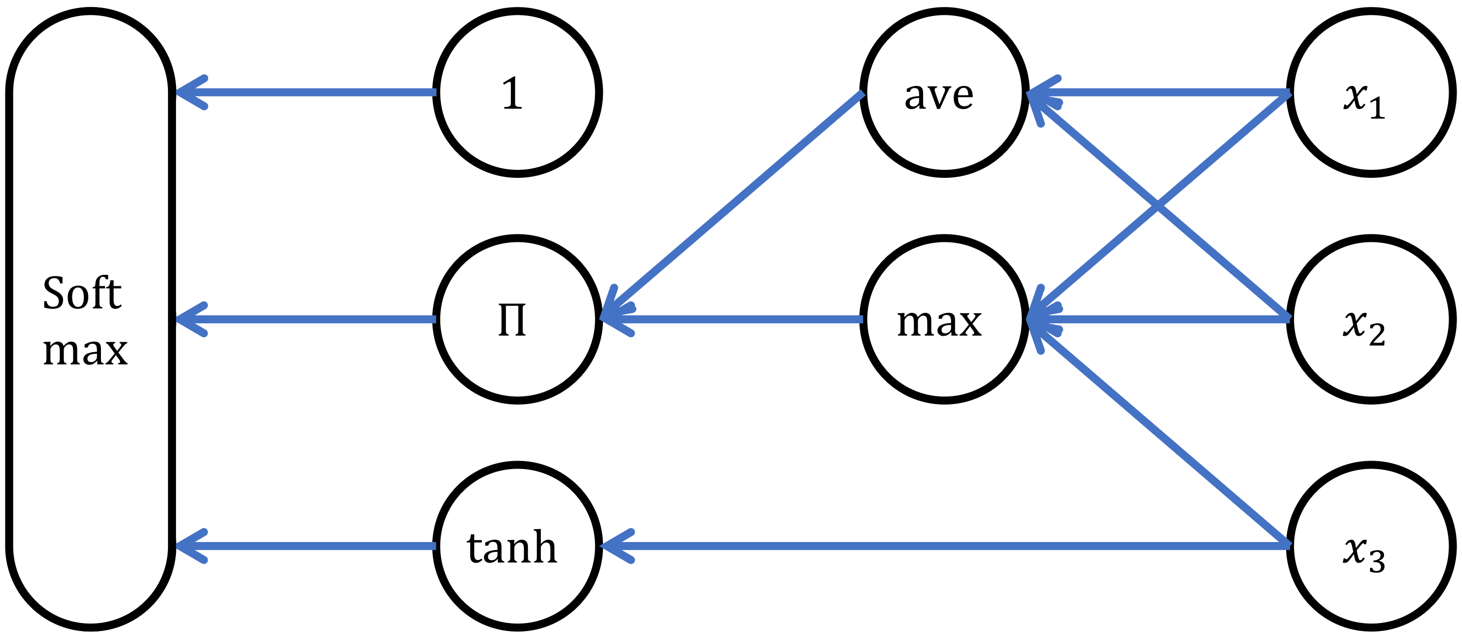
Unweighted Edge
- Direct input



f's output equals c.
c is defined as f's output.

Flexibility NN

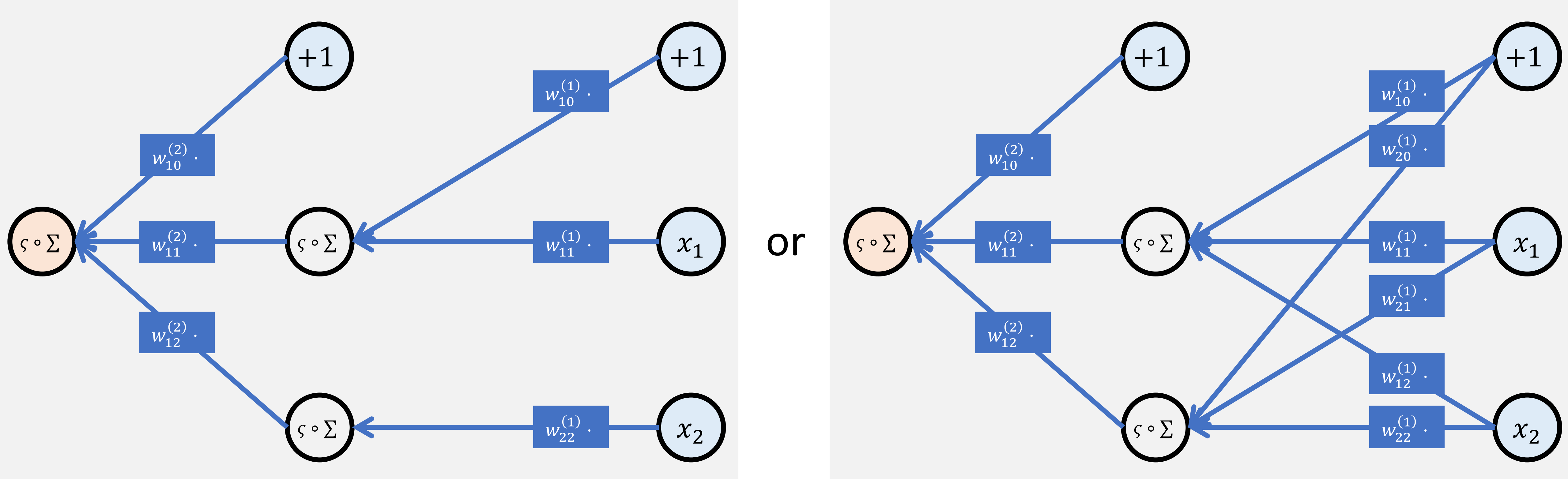
- You can design any directed acyclic graph



NN pipeline 1:
Model selection (this week)

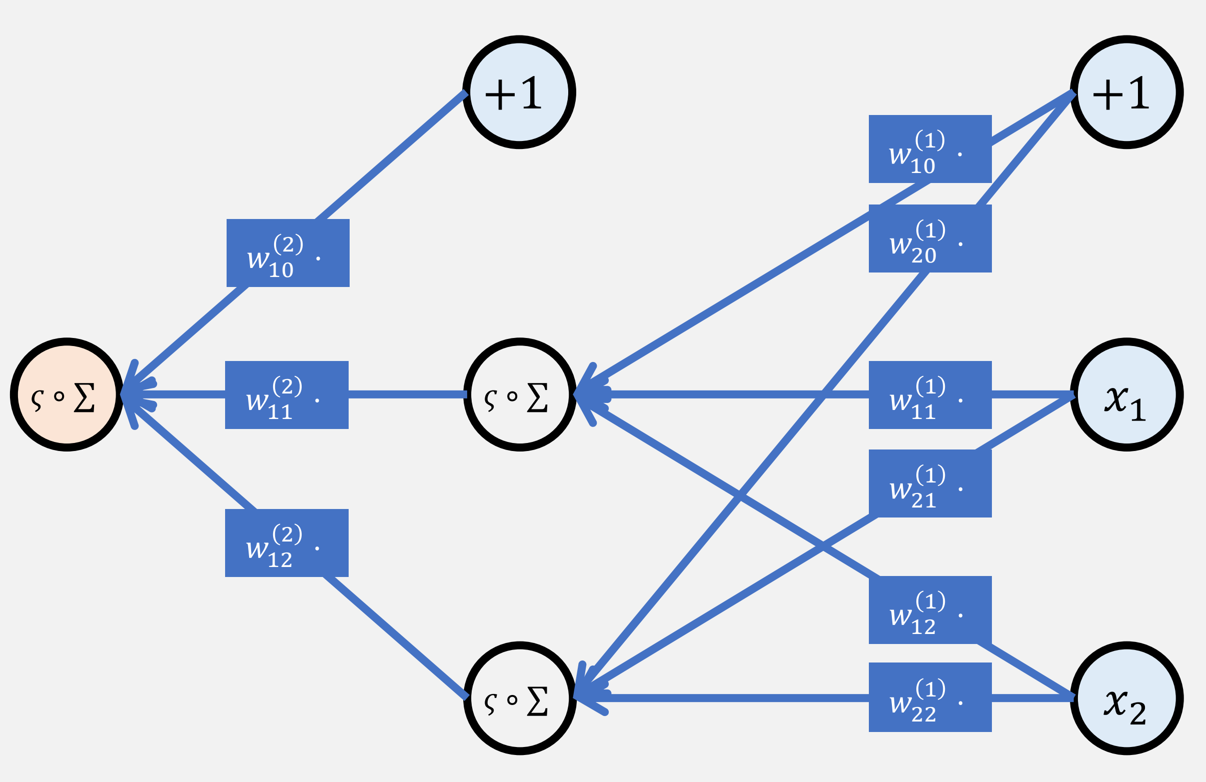
Too many parameters may cause overfitting.
Reducing parameters may cause underfitting.
Appropriate model selection is essential.

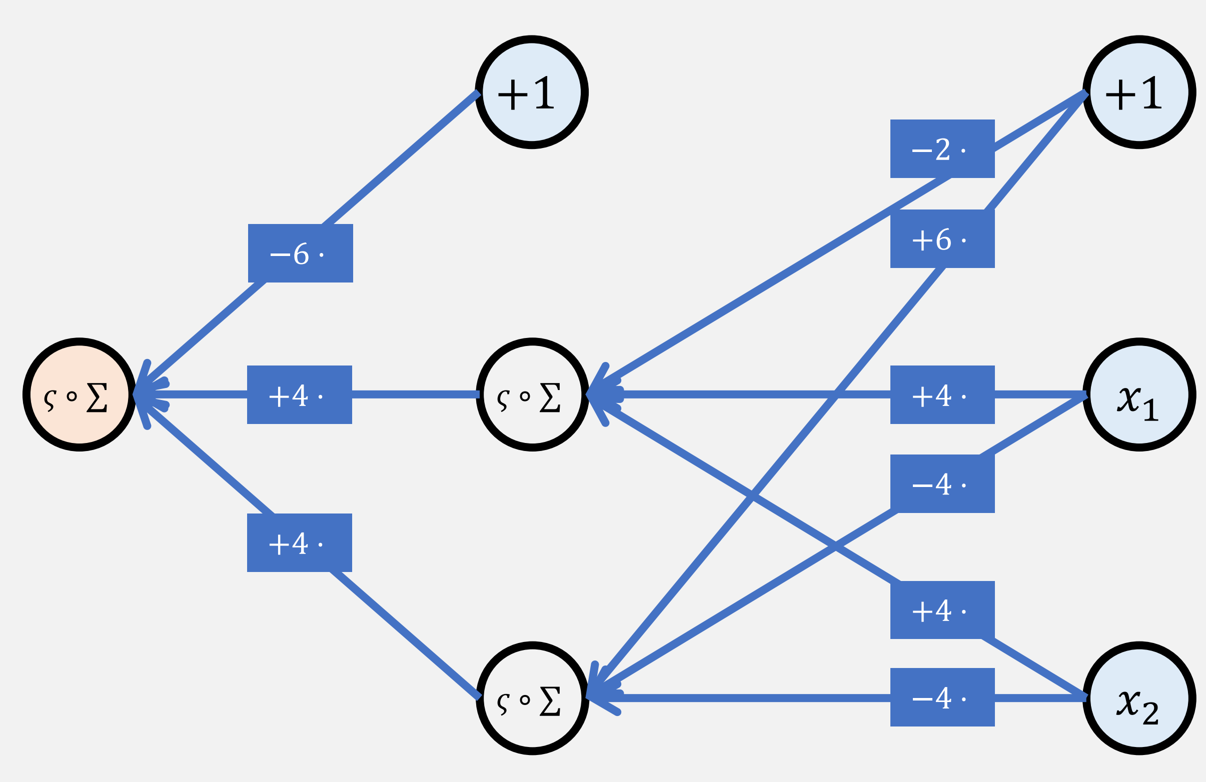
- From domain knowledge



NN pipeline 2:
Parameter learning (not covered by this module)

- From training data





Method: SGDs with backpropagation (for any feedforward NN!)

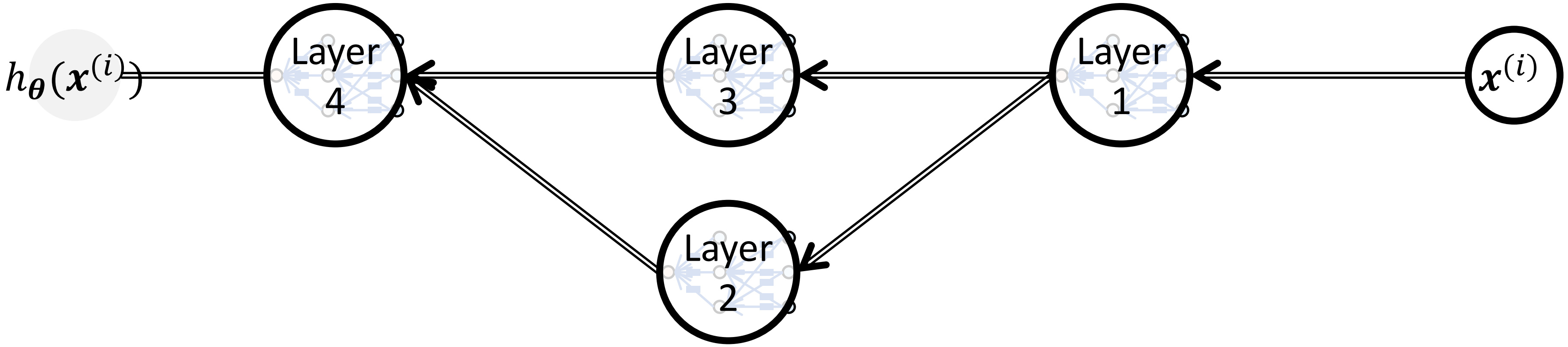
Tips

- In the coursework or MSc project, stating like “a neural network is used” is of no use, since it does not specify the graph structure of the network.
- Stating like “a convolutional neural network is used” or “a long short term memory is used” is not sufficient. These are categories of neural networks. Make sure to explicitly explain the network structure.

Layers

Layer: a subset of a neural network

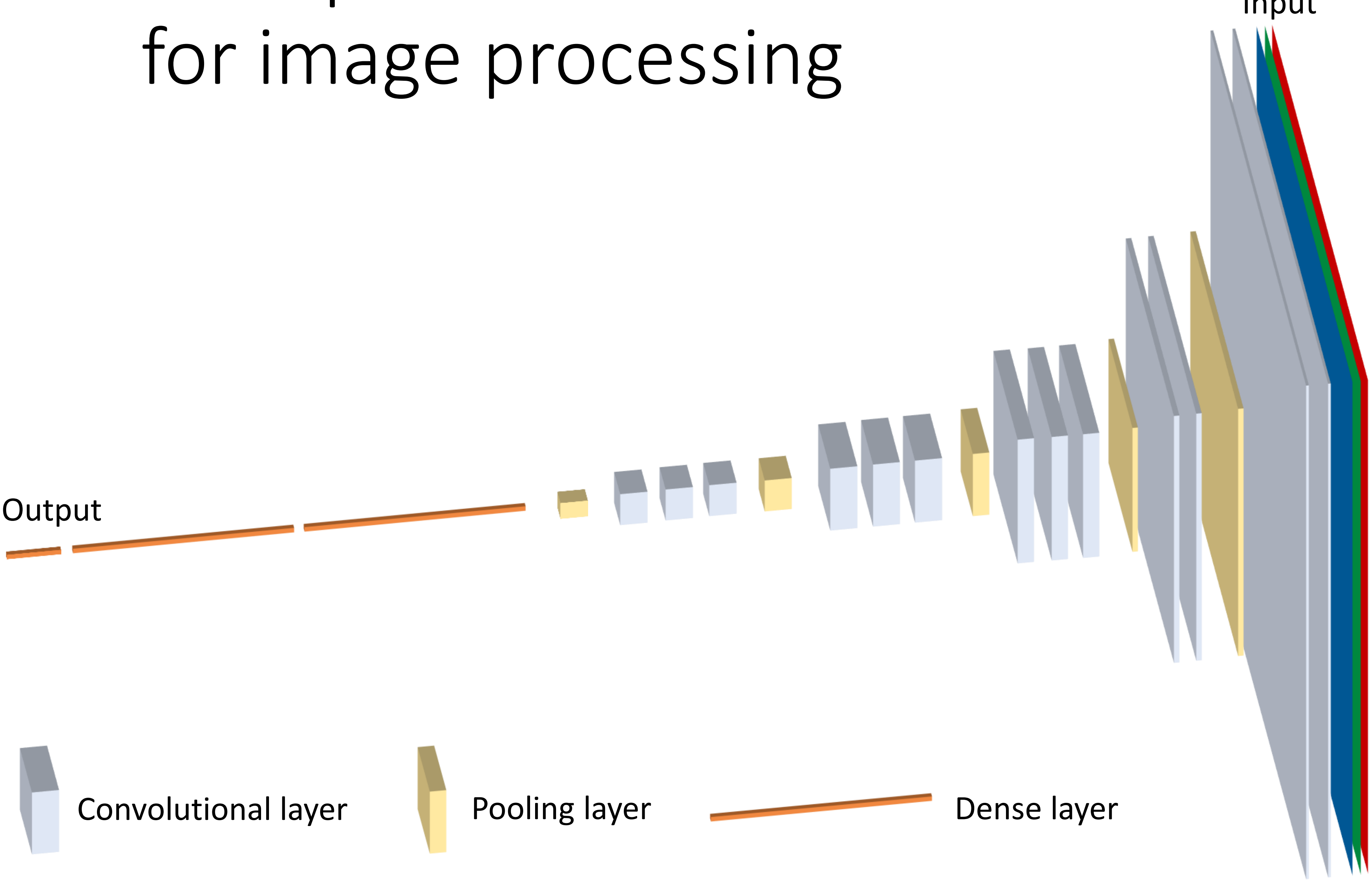
- Designing a NN from scratch has too much freedom.
- We split a NN into useful submodules, called "layers."
- The hypothesis function of the NN is given by the composition of the functions represented by each layer.



$$h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) = f_4\left(f_3(f_1(\boldsymbol{x}^{(i)})), f_2(f_1(\boldsymbol{x}^{(i)}))\right),$$

where f_k is the function represented by layer k.

Example: VGG-16 [Simonyan & Zisserman, 2015] (Visual geometry group) for image processing



Layer	Channel size	# channels
Input	224x224	3 (RGB image)
Conv 3x3	224x224	64
Conv 3x3	224x224	64
Maxpool 2x2	112x112	64
Conv 3x3	112x112	128
Conv 3x3	112x112	128
Maxpool 2x2	56x56	128
Conv 3x3	56x56	256
Conv 3x3	56x56	256
Conv 3x3	56x56	256
Maxpool 2x2	28x28	256
Conv 3x3	28x28	512
Conv 3x3	28x28	512
Conv 3x3	28x28	512
Maxpool 2x2	14x14	512
Conv 3x3	14x14	512
Conv 3x3	14x14	512
Conv 3x3	14x14	512
Maxpool 2x2	7x7	512
Dense	-	4096
Dense	-	4096
Dense	-	1000
Softmax		

Layer: a subset of a neural network

- Designing a NN from scratch has too much freedom.
- We split a NN into useful submodules, called “layers.”
- The hypothesis function of the NN is given by the composition of the functions represented by each layer.
- Examples:
  - General use: dense layer
  - Image processing: convolutional layer, pooling layer, batch normalization layer,
  - Time-series data: long short term memory layer, gated recurrent unit layer
  - Natural language processing: attention layer, multi-head attention layer.

Layer: a subset of a neural network

- Designing a NN from scratch has too much freedom.
- We split a NN into useful submodules, called “layers.”
- The hypothesis function of the NN is given by the composition of the functions represented by each layer.
- Examples:

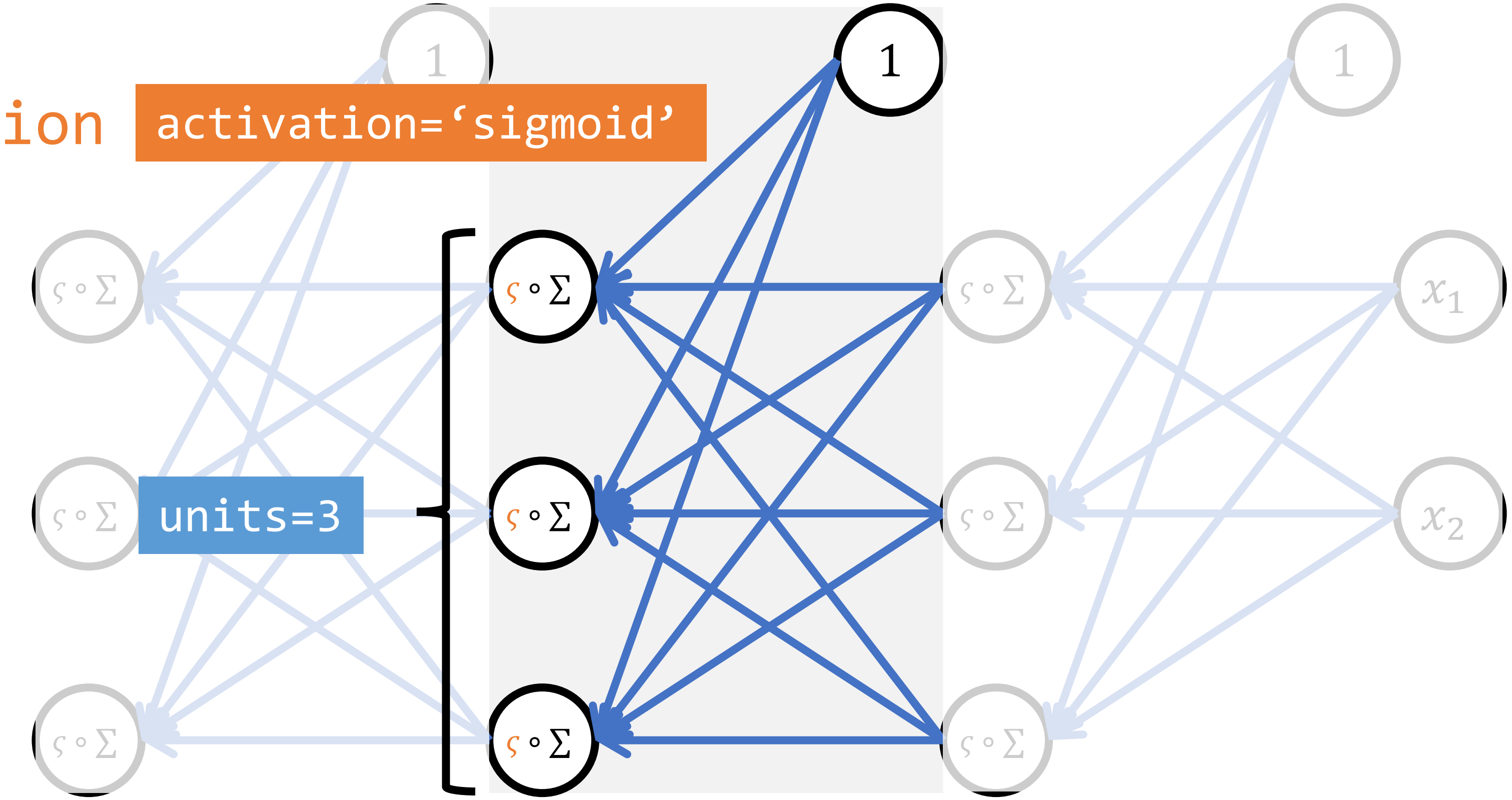
  Today’s focus

  - General use: **dense layer**
  - Image processing: convolutional layer, pooling layer, batch normalization layer,
  - Time-series data: long short term memory layer, gated recurrent unit layer
  - Natural language processing: attention layer, multi-head attention layer.

Dense layer

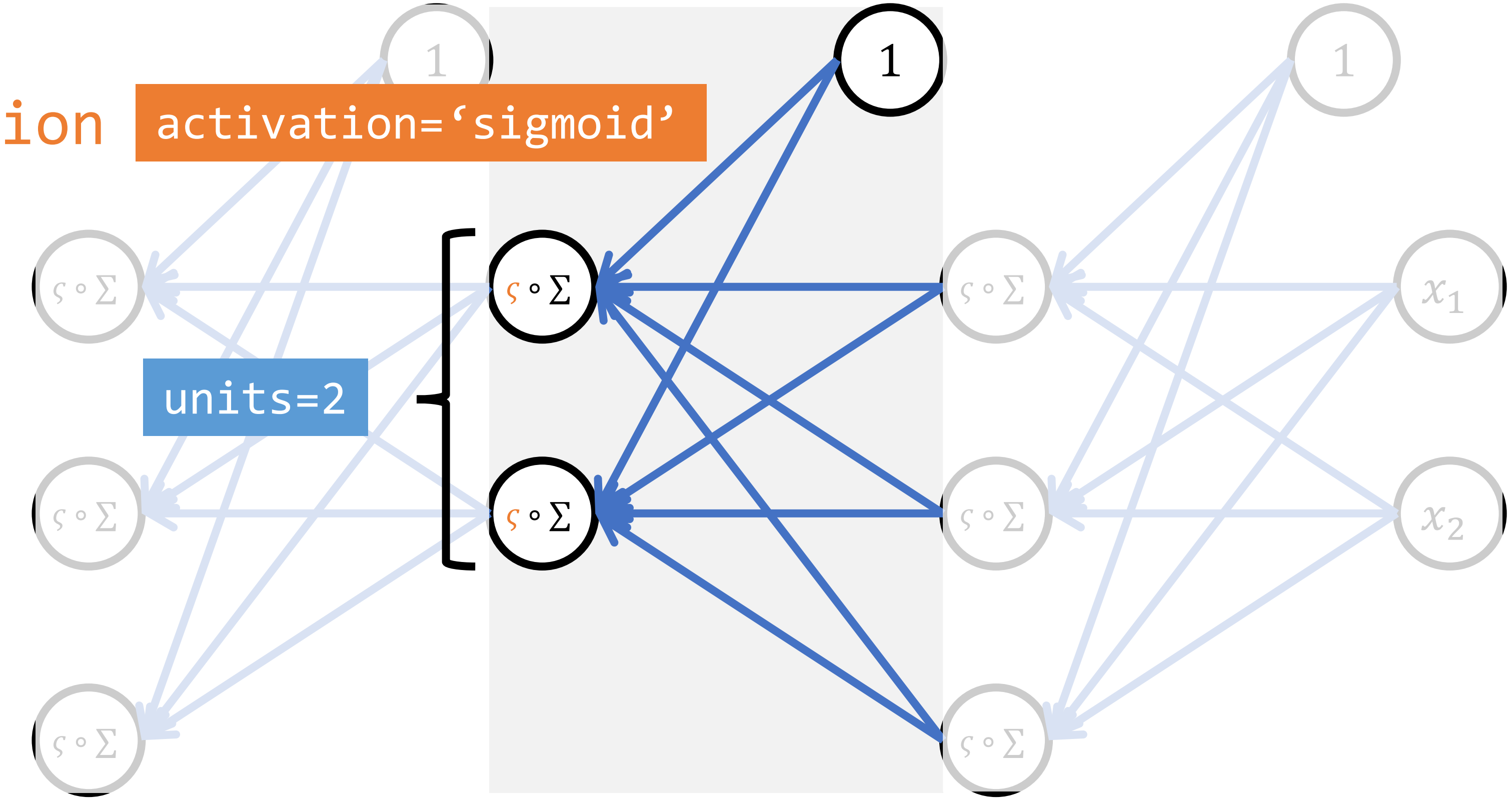
Dense layer (**Dense**)

- Fully connected layer
- Key hyperparameters:
  - `units`
  - `activation`



Dense layer (**Dense**)

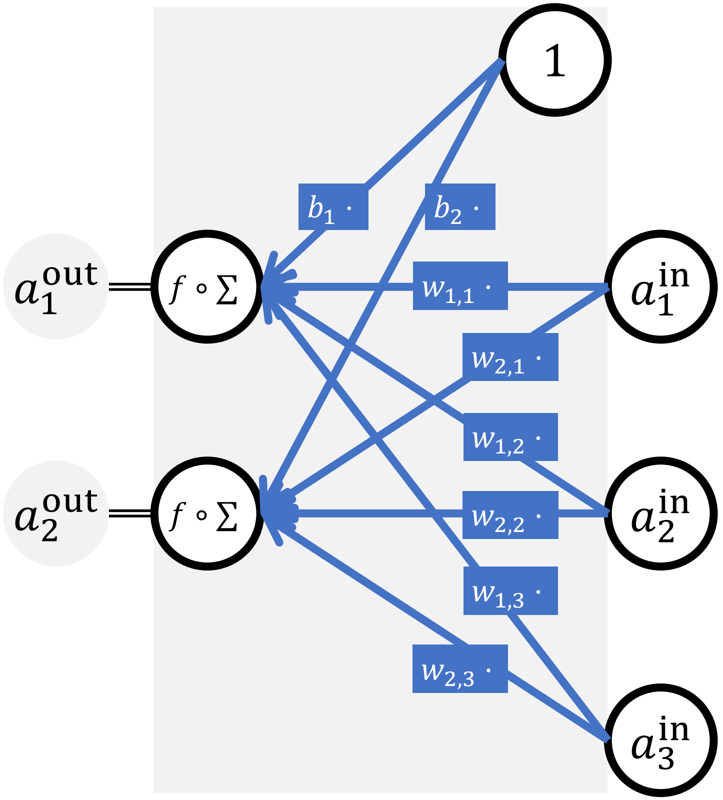
- Fully connected layer
- Key hyperparameters:
  - `units`
  - `activation`



Dense layer: representation by a matrix

- Input: $\boldsymbol{a}^{\text{in}} = \begin{bmatrix} a_1^{\text{in}} \\ a_2^{\text{in}} \\ a_3^{\text{in}} \end{bmatrix}$,

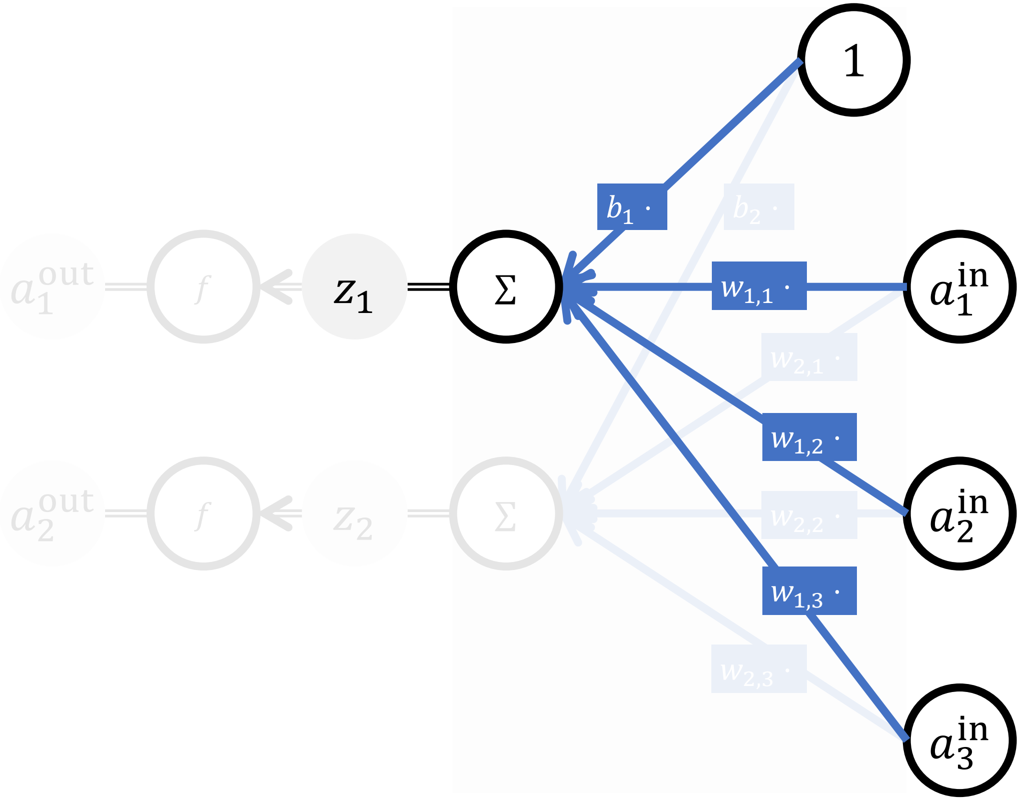
  output: $\boldsymbol{a}^{\text{out}} = \begin{bmatrix} a_1^{\text{out}} \\ a_2^{\text{out}} \end{bmatrix}$.
- The relation between $\boldsymbol{a}^{\text{in}}$ and $\boldsymbol{a}^{\text{out}}$?



Dense layer: representation by a matrix

- $z_1 = b_1 + w_{1,1}a_1^{\text{in}} + w_{1,2}a_2^{\text{in}} + w_{1,3}a_3^{\text{in}}$

$$= [b_1] + [w_{1,1} \quad w_{1,2} \quad w_{1,3}] \begin{bmatrix} a_1^{\text{in}} \\ a_2^{\text{in}} \\ a_3^{\text{in}} \end{bmatrix},$$



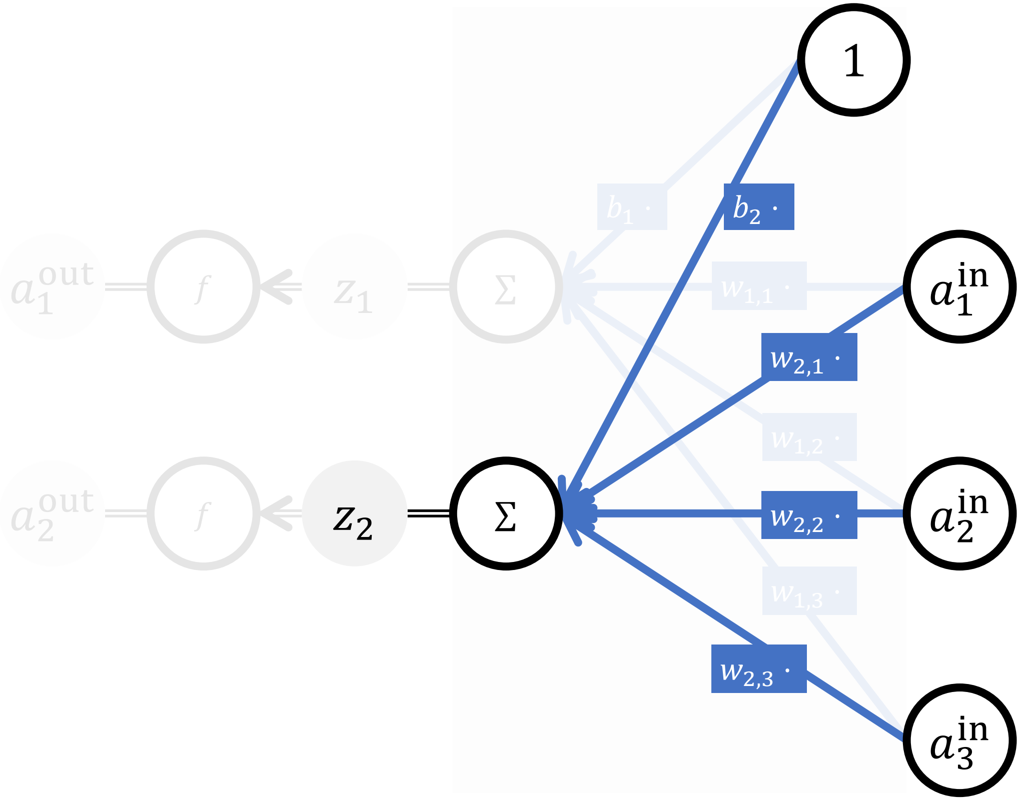
Dense layer: representation by a matrix

- $z_1 = b_1 + w_{1,1}a_1^{\text{in}} + w_{1,2}a_2^{\text{in}} + w_{1,3}a_3^{\text{in}}$

$$= [b_1] + [w_{1,1} \quad w_{1,2} \quad w_{1,3}]\begin{bmatrix} a_1^{\text{in}} \\ a_2^{\text{in}} \\ a_3^{\text{in}} \end{bmatrix},$$

$$z_2 = b_2 + w_{2,1}a_1^{\text{in}} + w_{2,2}a_2^{\text{in}} + w_{2,3}a_3^{\text{in}}$$

$$= [b_2] + [w_{2,1} \quad w_{2,2} \quad w_{2,3}]\begin{bmatrix} a_1^{\text{in}} \\ a_2^{\text{in}} \\ a_3^{\text{in}} \end{bmatrix}.$$

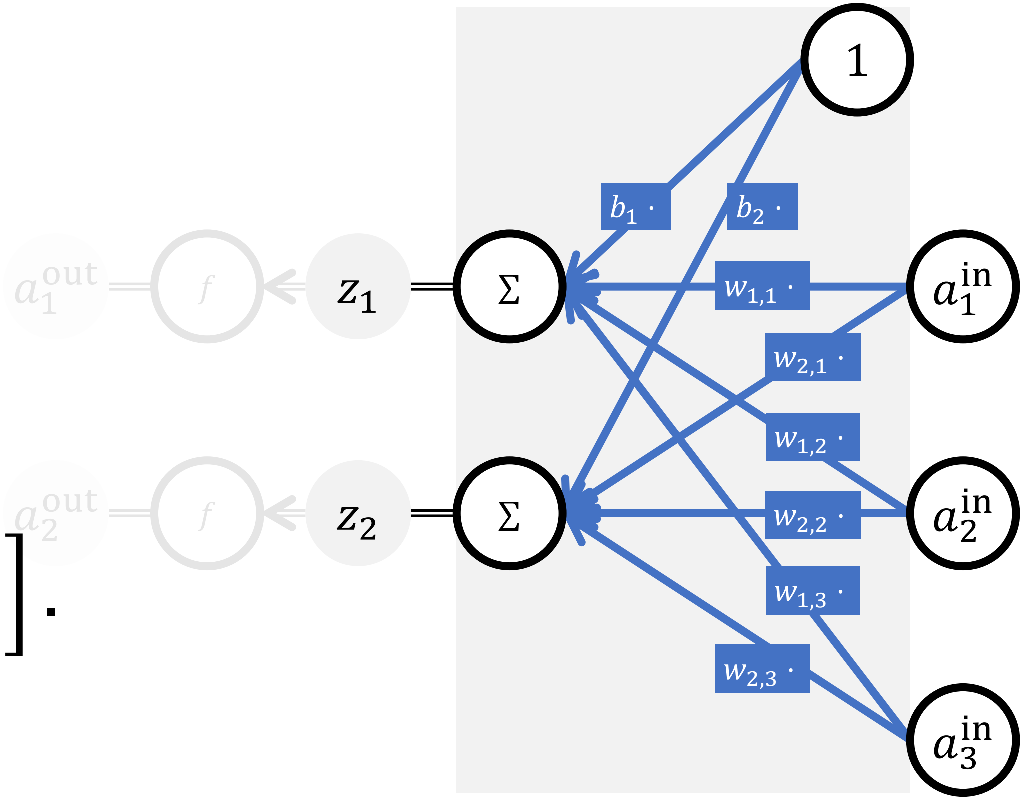


Dense layer: representation by a matrix

- $\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} b_1 + w_{1,1}a_1^{\text{in}} + w_{1,2}a_2^{\text{in}} + w_{1,3}a_3^{\text{in}} \\ b_2 + w_{2,1}a_1^{\text{in}} + w_{2,2}a_2^{\text{in}} + w_{2,3}a_3^{\text{in}} \end{bmatrix}$

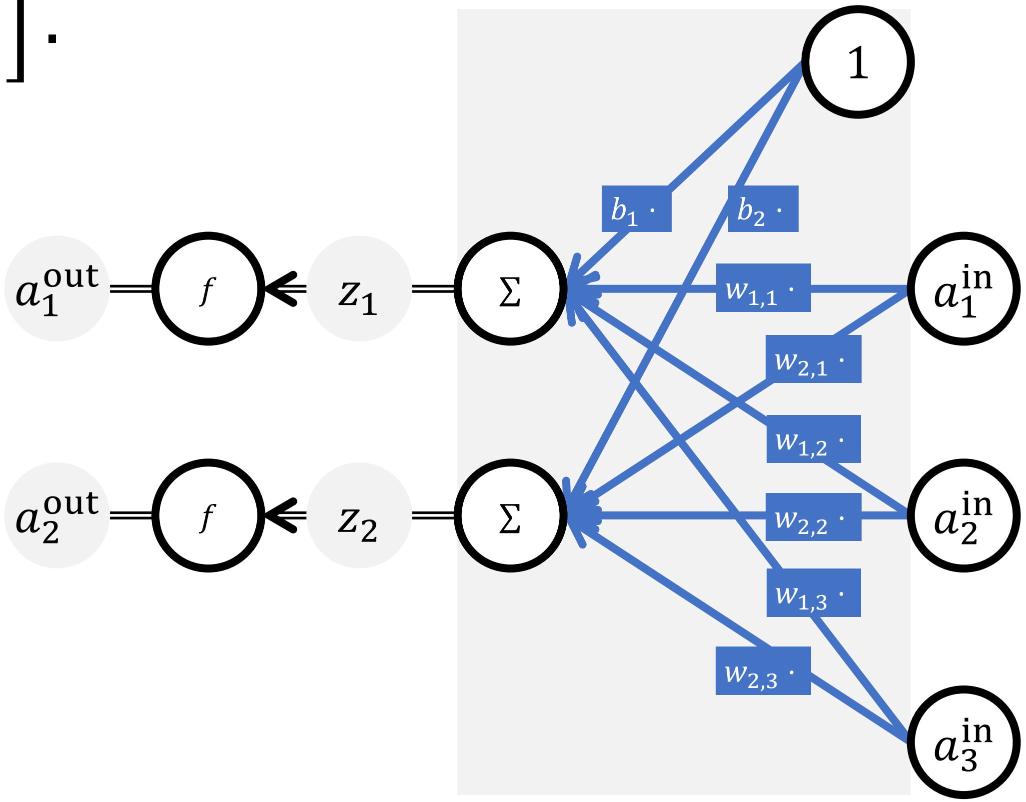
  $= \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{bmatrix} \begin{bmatrix} a_1^{\text{in}} \\ a_2^{\text{in}} \\ a_3^{\text{in}} \end{bmatrix},$

- $\boldsymbol{z} \coloneqq \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ is given by $\boldsymbol{z} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{a}^{\text{in}}$,
  where $\boldsymbol{b} \coloneqq \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \boldsymbol{W} \coloneqq \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{bmatrix}$.



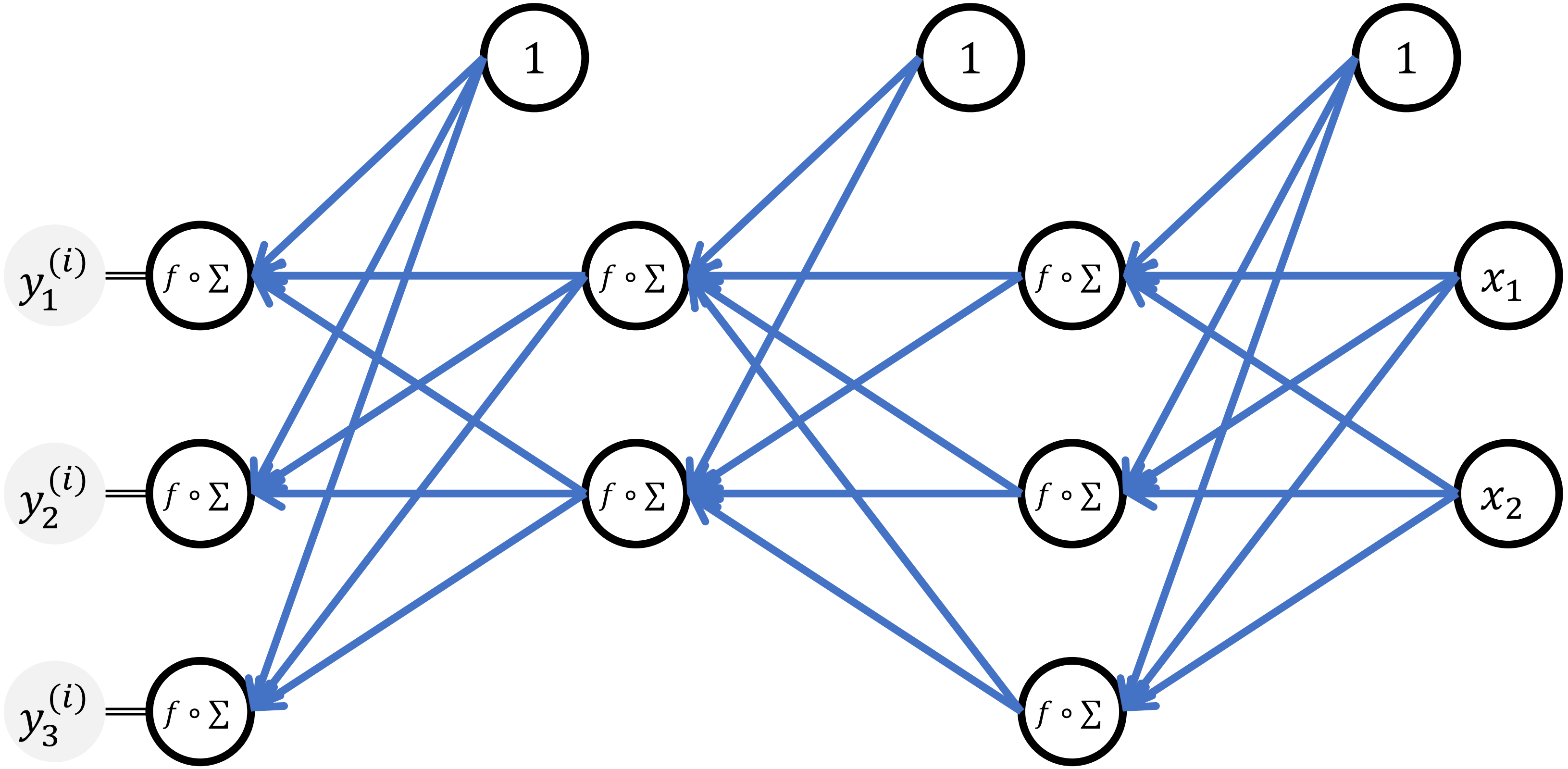
Dense layer: representation by a matrix

- $\boldsymbol{z} := \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ is given by $\boldsymbol{z} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{a}^{\text{in}}$, where $\boldsymbol{b} := \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \boldsymbol{W} := \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{bmatrix}$.
- $\begin{bmatrix} a_1^{\text{out}} \\ a_2^{\text{out}} \end{bmatrix} = \begin{bmatrix} f(z_1) \\ f(z_2) \end{bmatrix} =: f(\boldsymbol{z})$.
- Hence, $\boldsymbol{a}^{\text{out}} = \begin{bmatrix} a_1^{\text{out}} \\ a_2^{\text{out}} \end{bmatrix}$ is given by $\boldsymbol{a}^{\text{out}} = f(\boldsymbol{b} + \boldsymbol{W}\boldsymbol{a}^{\text{in}})$.



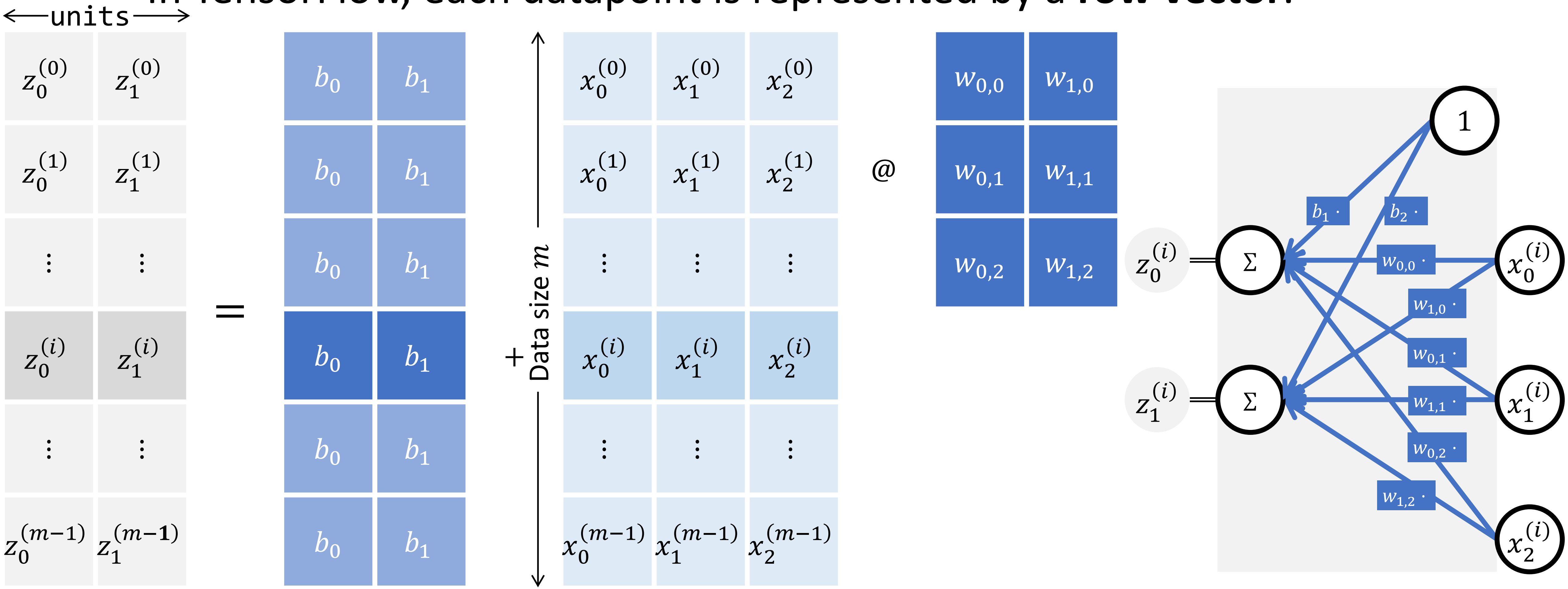
The fully-connected neural network: a NN that consists of dense layers

- $$\boldsymbol{y}^{(i)} = f\left(\boldsymbol{b}^{[3]} + \boldsymbol{W}^{[3]} f\left(\boldsymbol{b}^{[2]} + \boldsymbol{W}^{[2]} f\left(\boldsymbol{b}^{[1]} + \boldsymbol{W}^{[1]} \boldsymbol{x}^{(i)}\right)\right)\right)$$



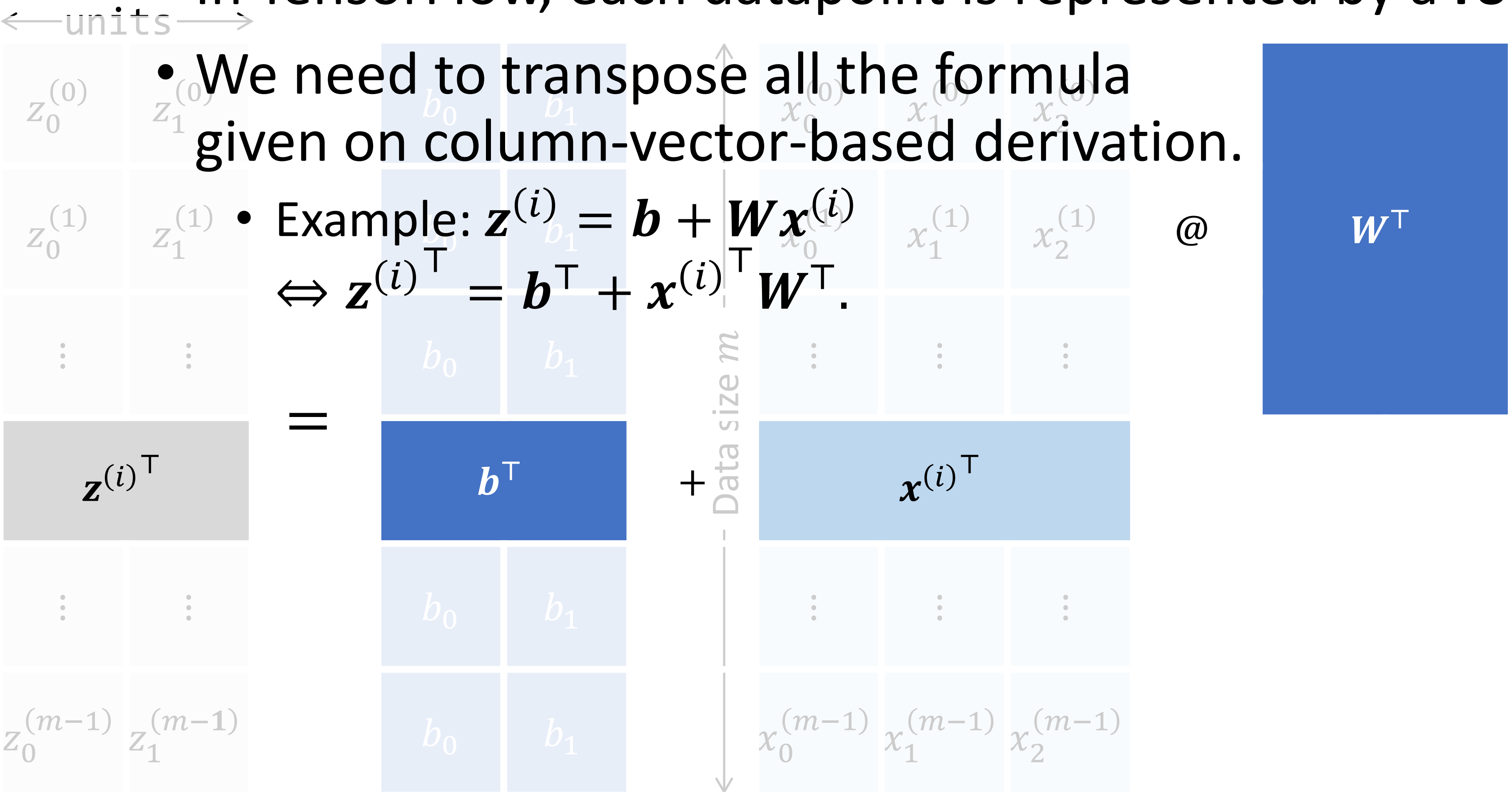
Notes for in TensorFlow implementation

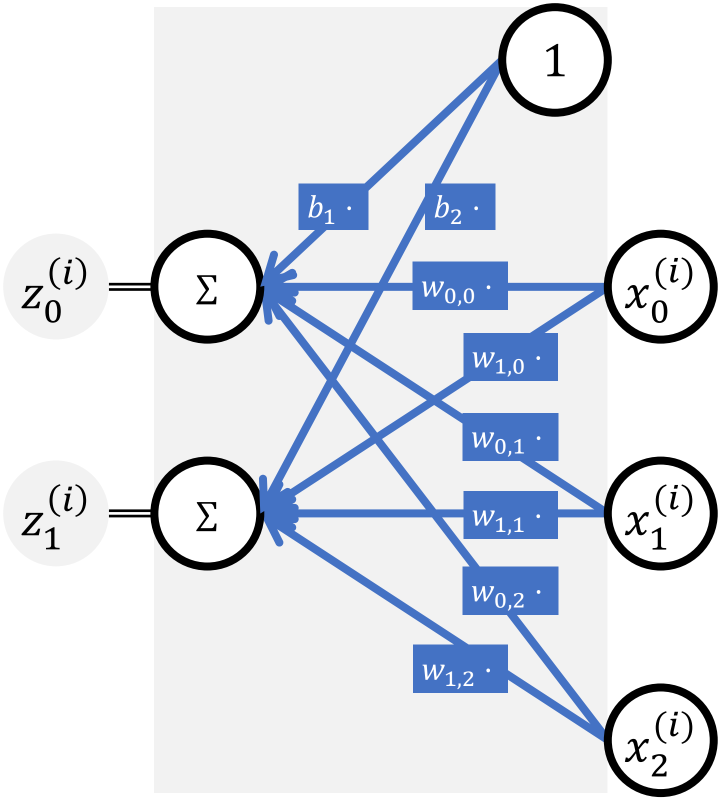
- In TensorFlow, each datapoint is represented by a **row vector**.



Notes for TensorFlow implementation

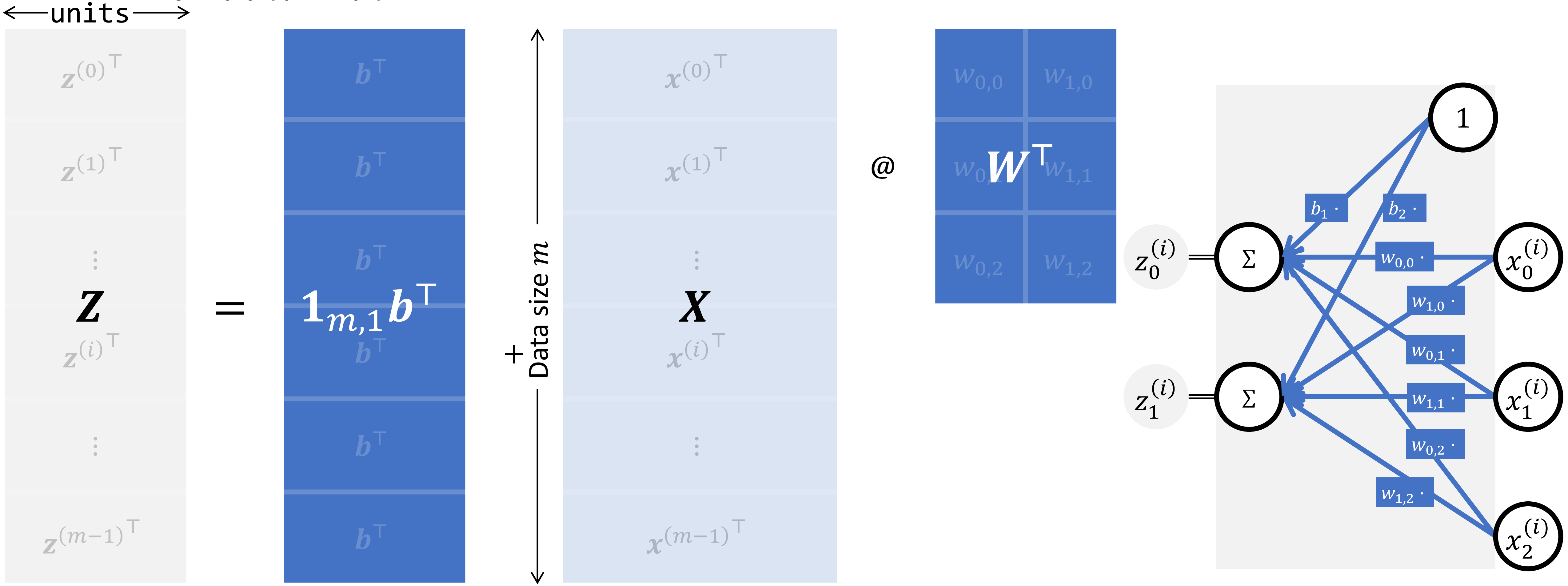
- In TensorFlow, each datapoint is represented by a **row vector**.
- We need to transpose all the formula given on column-vector-based derivation.
  - Example: $\boldsymbol{z}^{(i)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{x}^{(i)}$
    $\Leftrightarrow \boldsymbol{z}^{(i)\top} = \boldsymbol{b}^\top + \boldsymbol{x}^{(i)\top}\boldsymbol{W}^\top$.





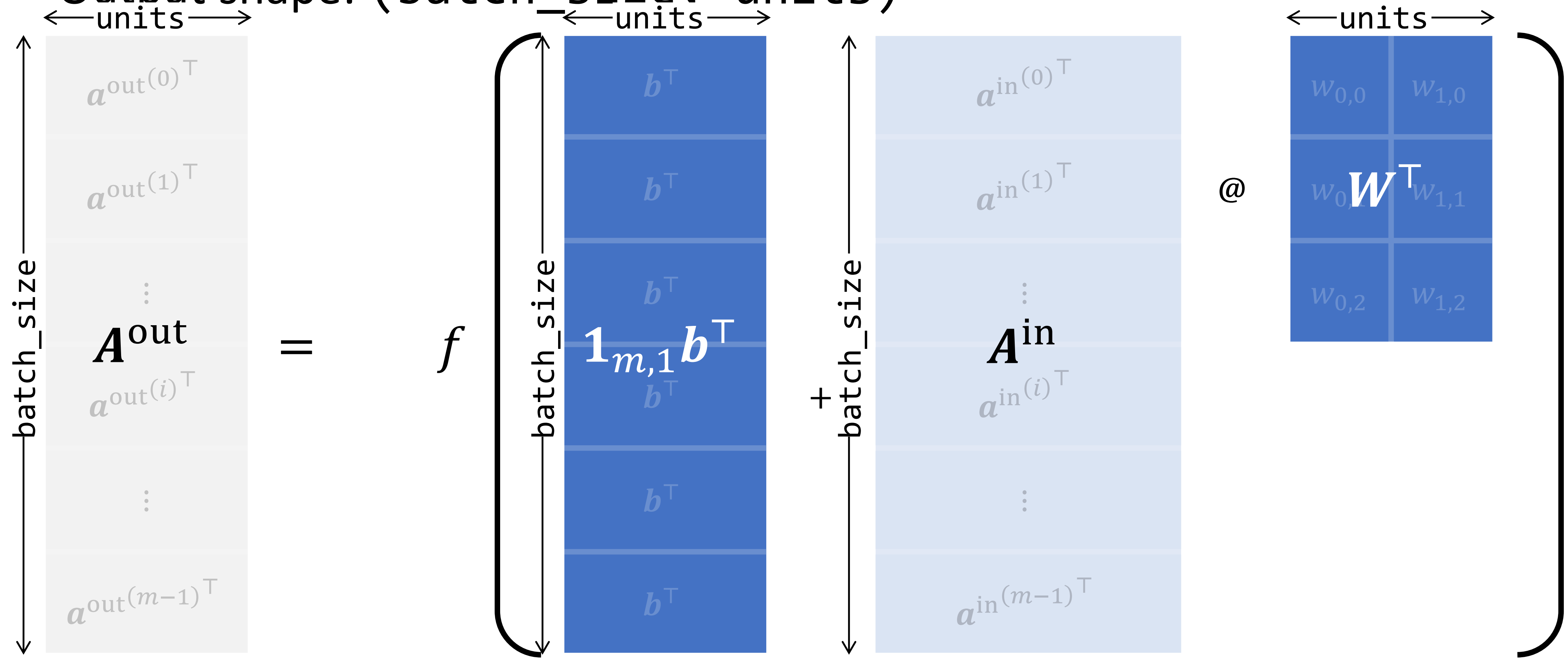
Notes for TensorFlow implementation

- For data matrix $\boldsymbol{X}$:



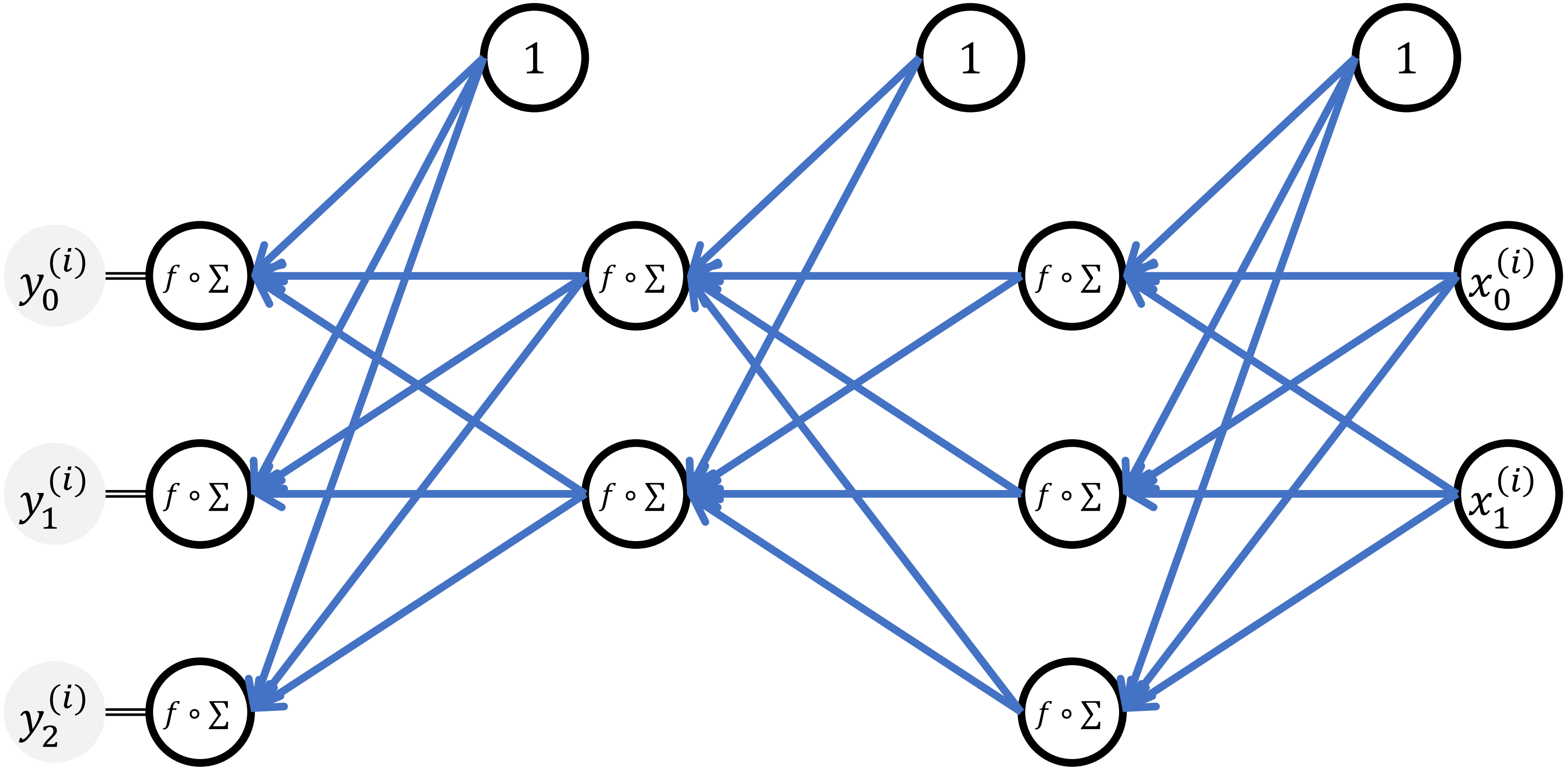
Notes for TensorFlow implementation

- Output shape: `(batch_size, units)`



The fully-connected neural network: matrix-form

- $\boldsymbol{Y} = f\left(\mathbf{1}_{m,1}\boldsymbol{b}^{[3]^\top} + f\left(\mathbf{1}_{m,1}\boldsymbol{b}^{[2]^\top} + f\left(\mathbf{1}_{m,1}\boldsymbol{b}^{[1]^\top} + \boldsymbol{X}\boldsymbol{W}^{[1]^\top}\right)\boldsymbol{W}^{[2]^\top}\right)\boldsymbol{W}^{[3]^\top}\right).$



Summary

- Where the property of data is well known, NNs specially designed for the data may work well.
- A NN defines a set of hypothesis functions determined by a weighted DAG, where the weights are learnable parameters.
- No matter what network you created, you can optimise it by stochastic gradient descent with backpropagation
- Layers are a set of nodes and edges. We can create a complex network by stacking and composing layers.

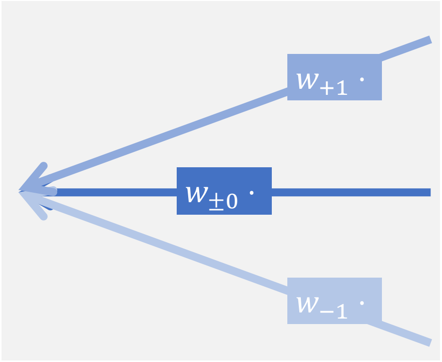
Appendices:
Other layers

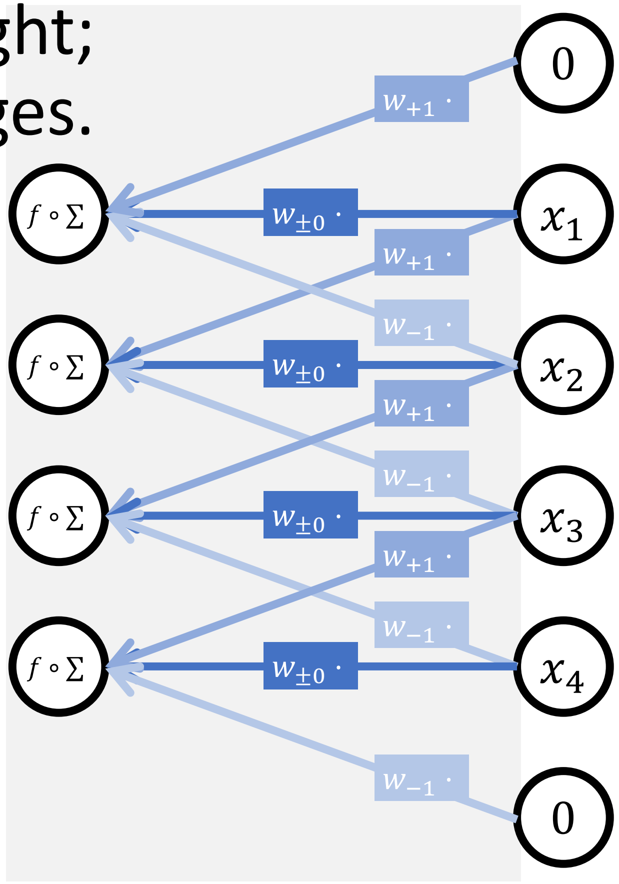
Convolutional layer

Convolutional layer
`(Conv1D, Conv2D, Conv3D)`

- The layer defined by parallel shifting of a filter.
- The same colour indicates the same weight; i.e., weights are shared among those edges.

A filter





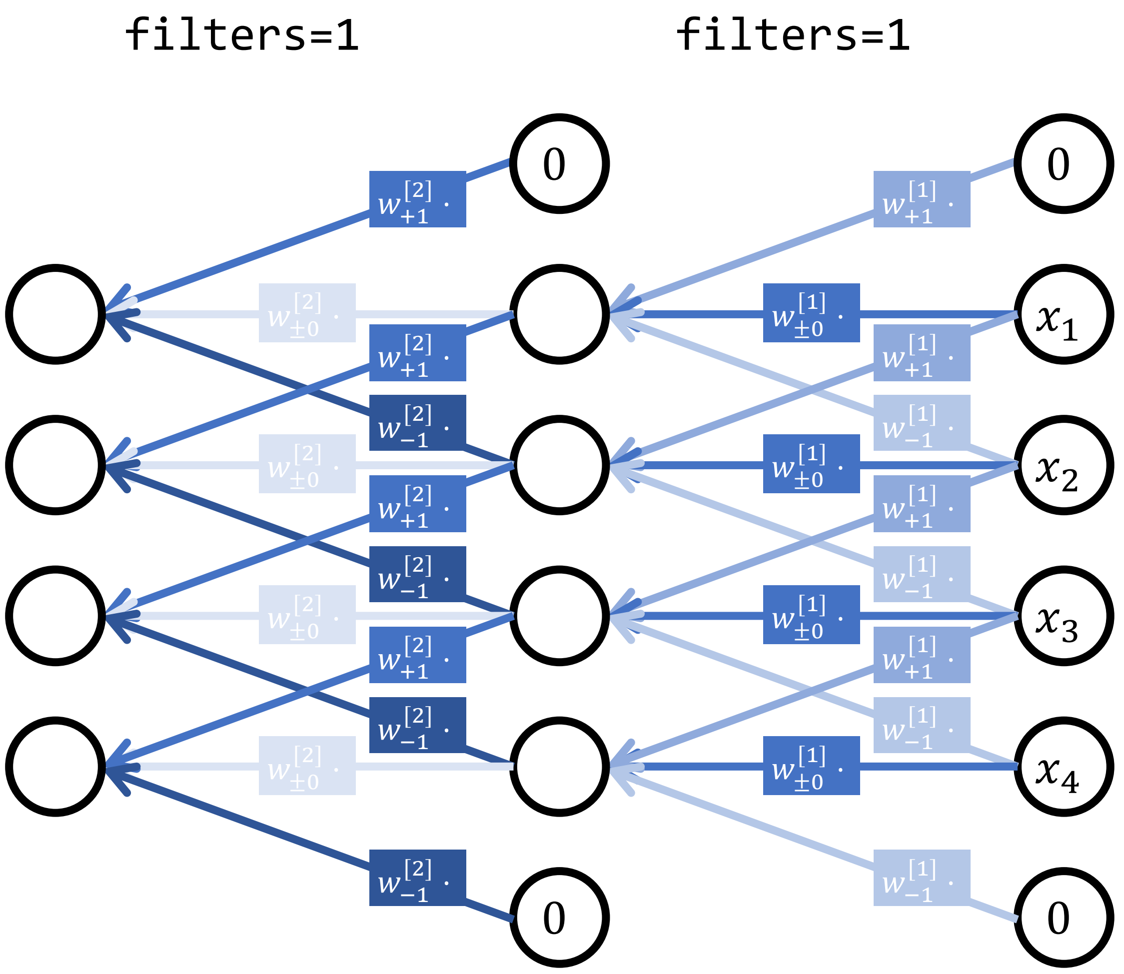
Hyperparameters of the convolutional layer:

Today's focus

- **`filters`**
- `kernel_size`
- `strides`
- `padding`
- `activation`

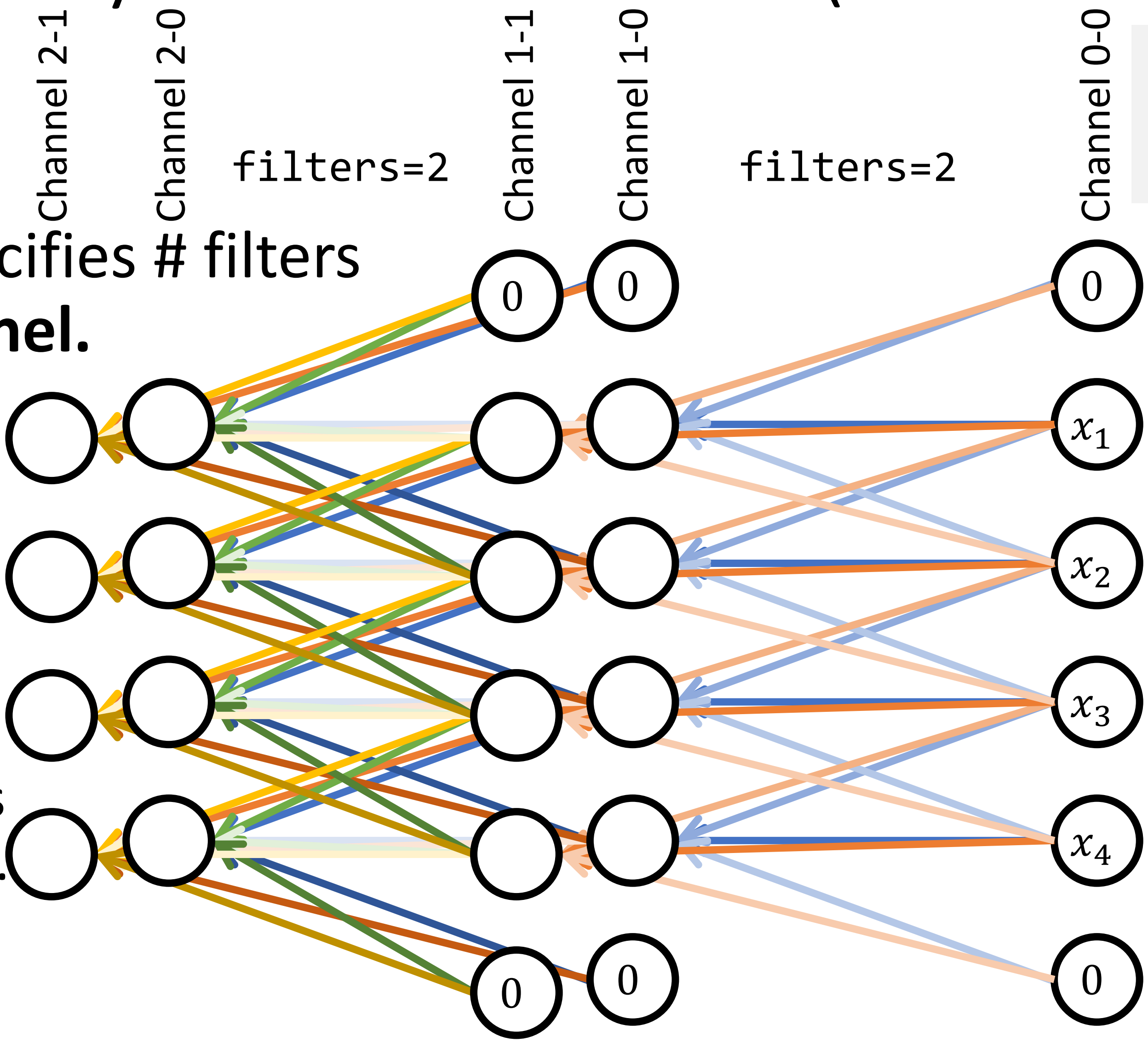
Convolutional layers: # channels (`filters`)

- Deeper case



Convolutional layers: # channels (`filters`)

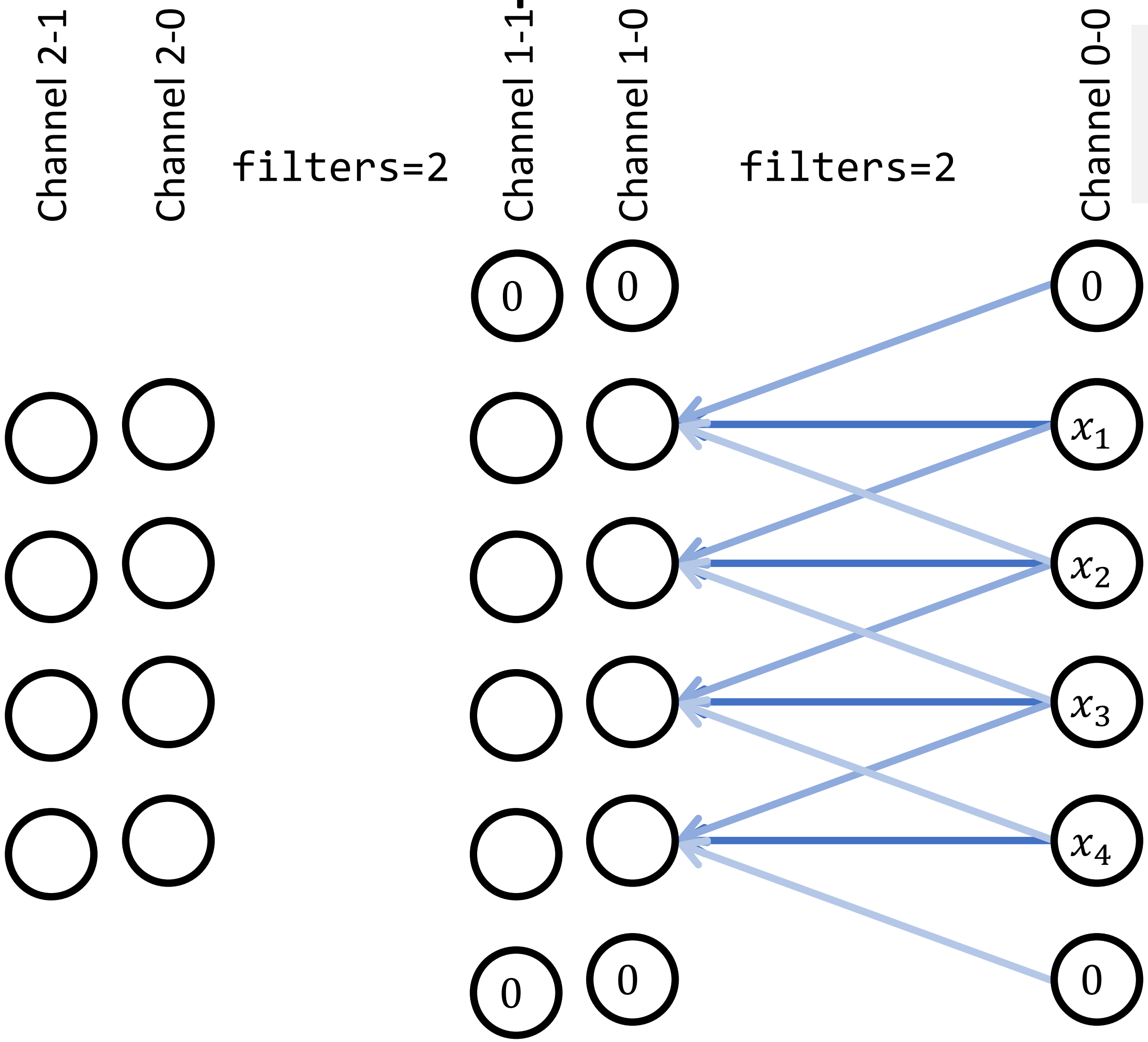
- Multi-channel case
- Note: `filters` specifies # filters **per one input channel.**
  - **$\neq$ the total # filters in the layer**
- In PyTorch, # channels is set by `out_channels`.
  - This name indicates the actual meaning.



"Channel i-j" indicates the j-th channel in the i-th layer

Multi-channel case: examples

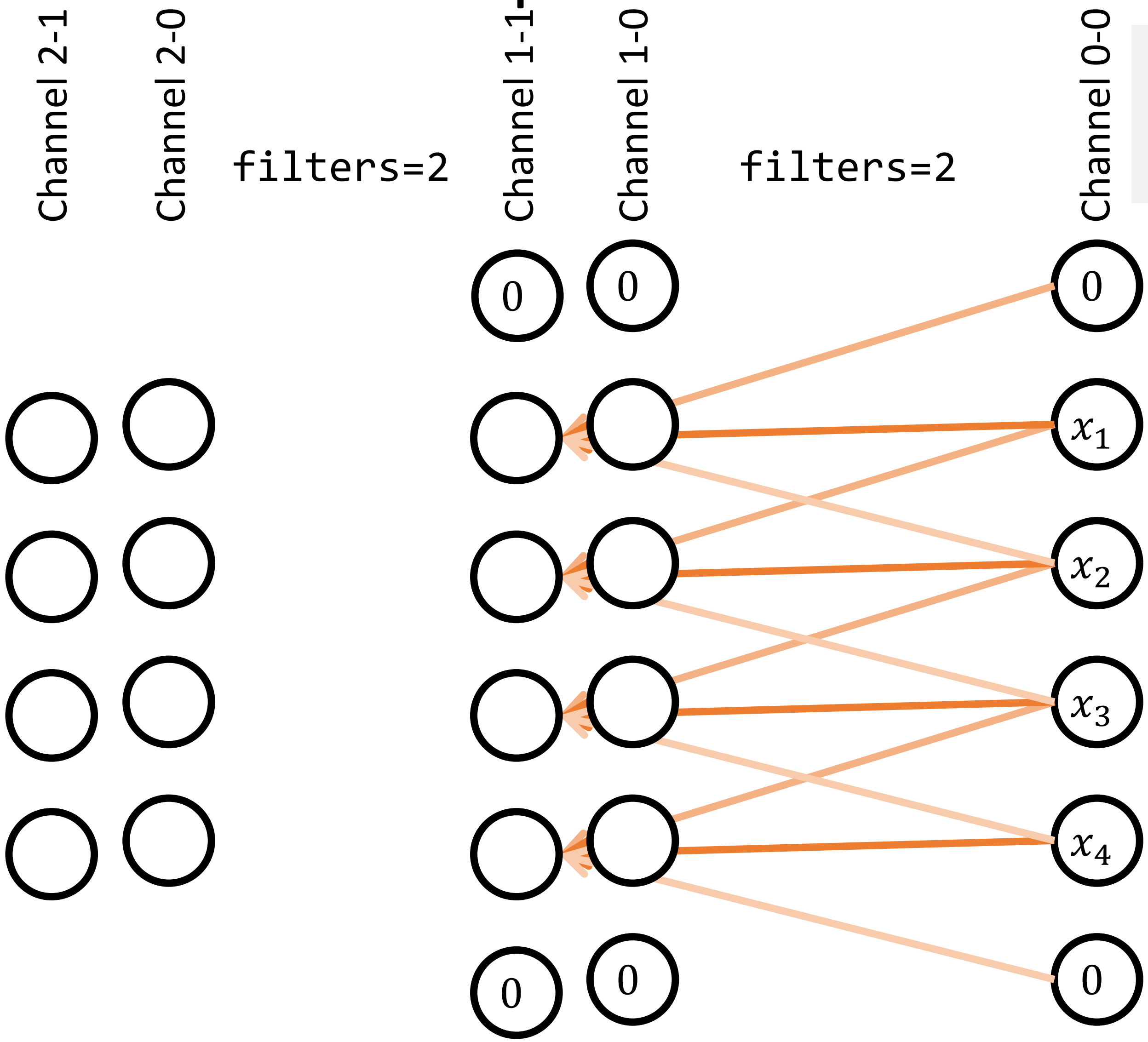
- The filter from Channel 0-0 to Channel 1-0.



"Channel i-j" indicates the j-th channel in the i-th layer

Multi-channel case: examples

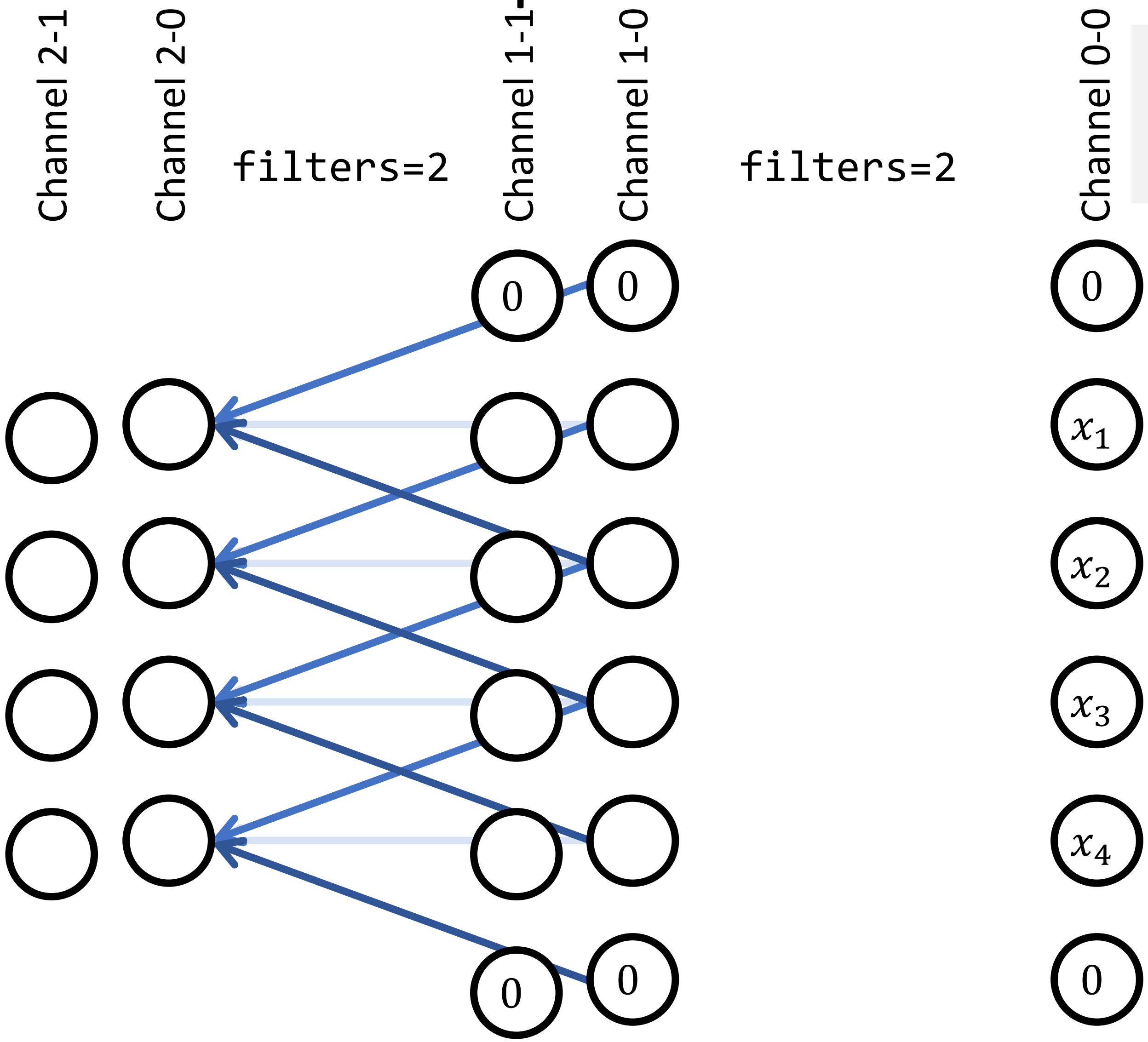
- The filter from Channel 0-0 to Channel 1-1.



"Channel i-j" indicates the j-th channel in the i-th layer

Multi-channel case: examples

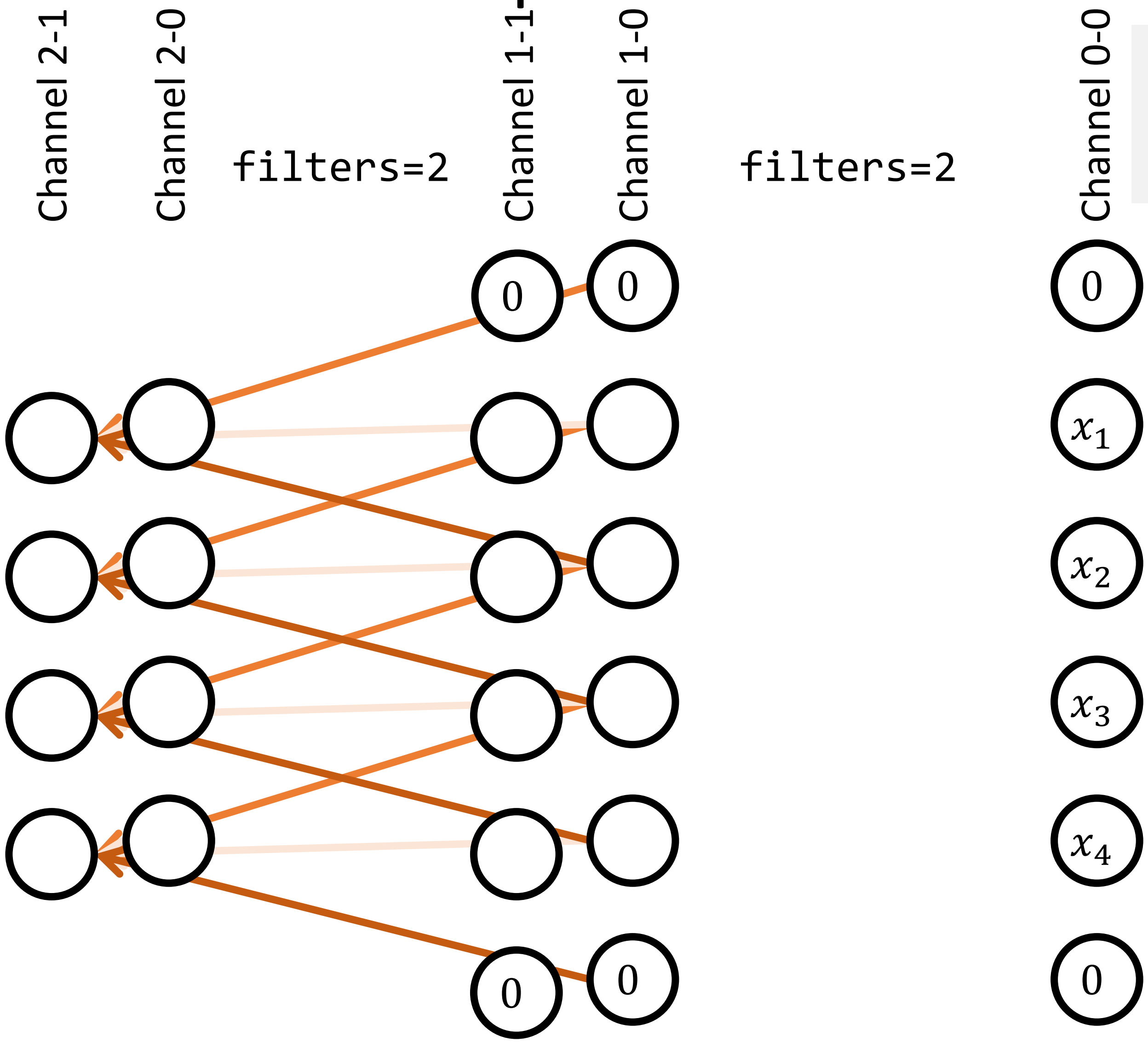
- The filter from Channel 1-0 to Channel 2-0.



"Channel i-j" indicates the j-th channel in the i-th layer

Multi-channel case: examples

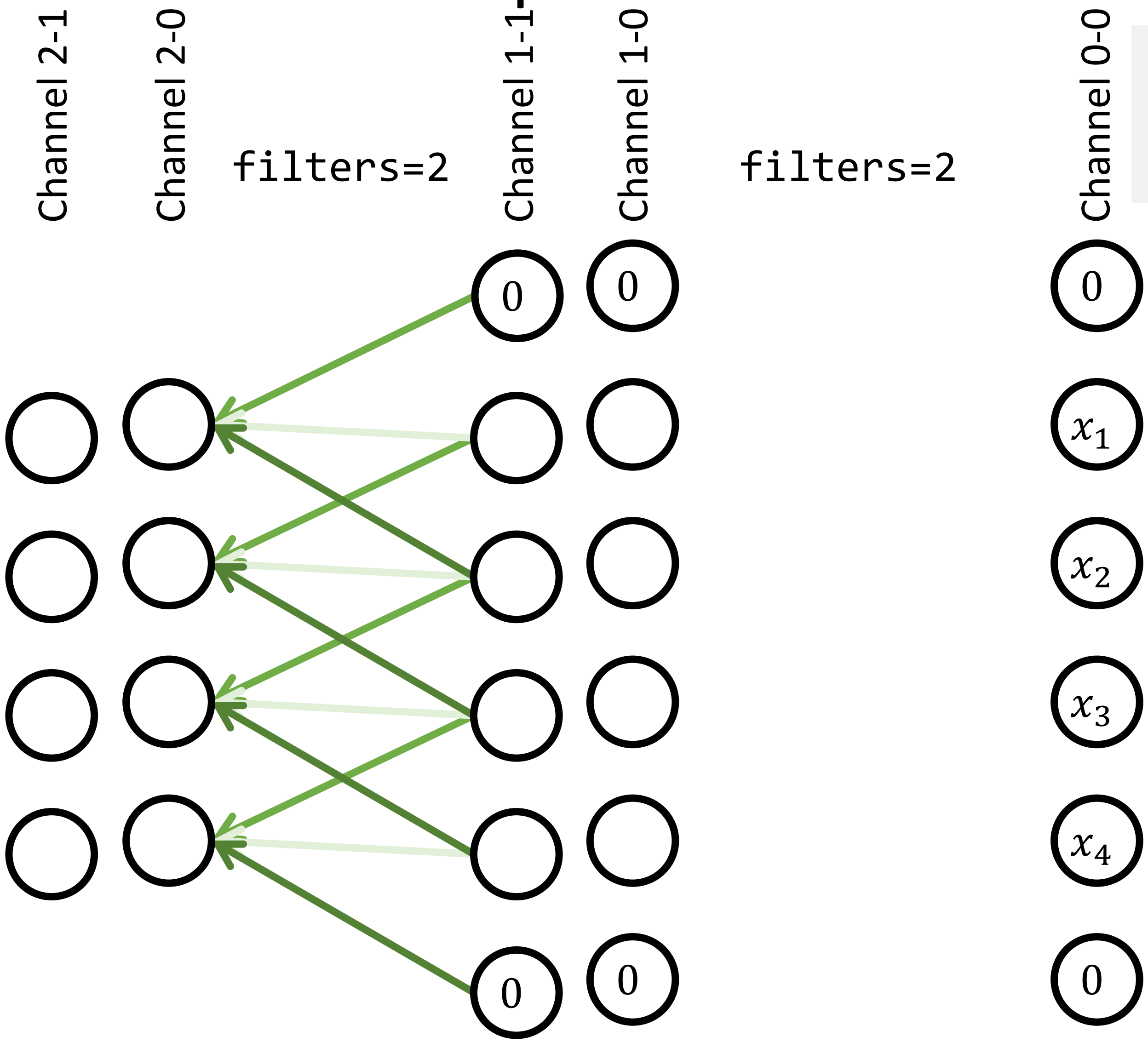
- The filter from Channel 1-0 to Channel 2-1.



"Channel i-j" indicates the j-th channel in the i-th layer

Multi-channel case: examples

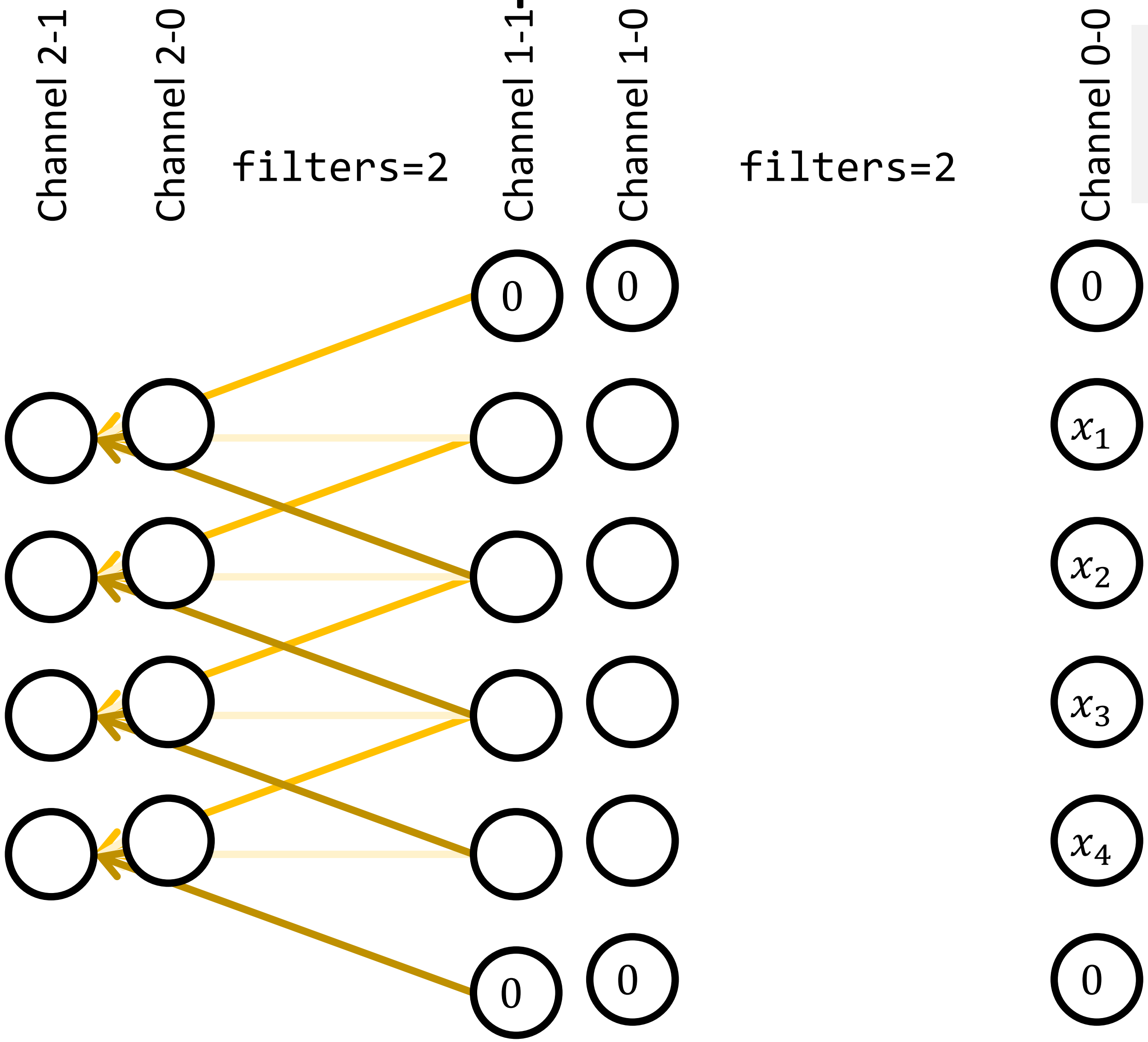
- The filter from Channel 1-1 to Channel 2-0.



"Channel i-j" indicates the j-th channel in the i-th layer

Multi-channel case: examples

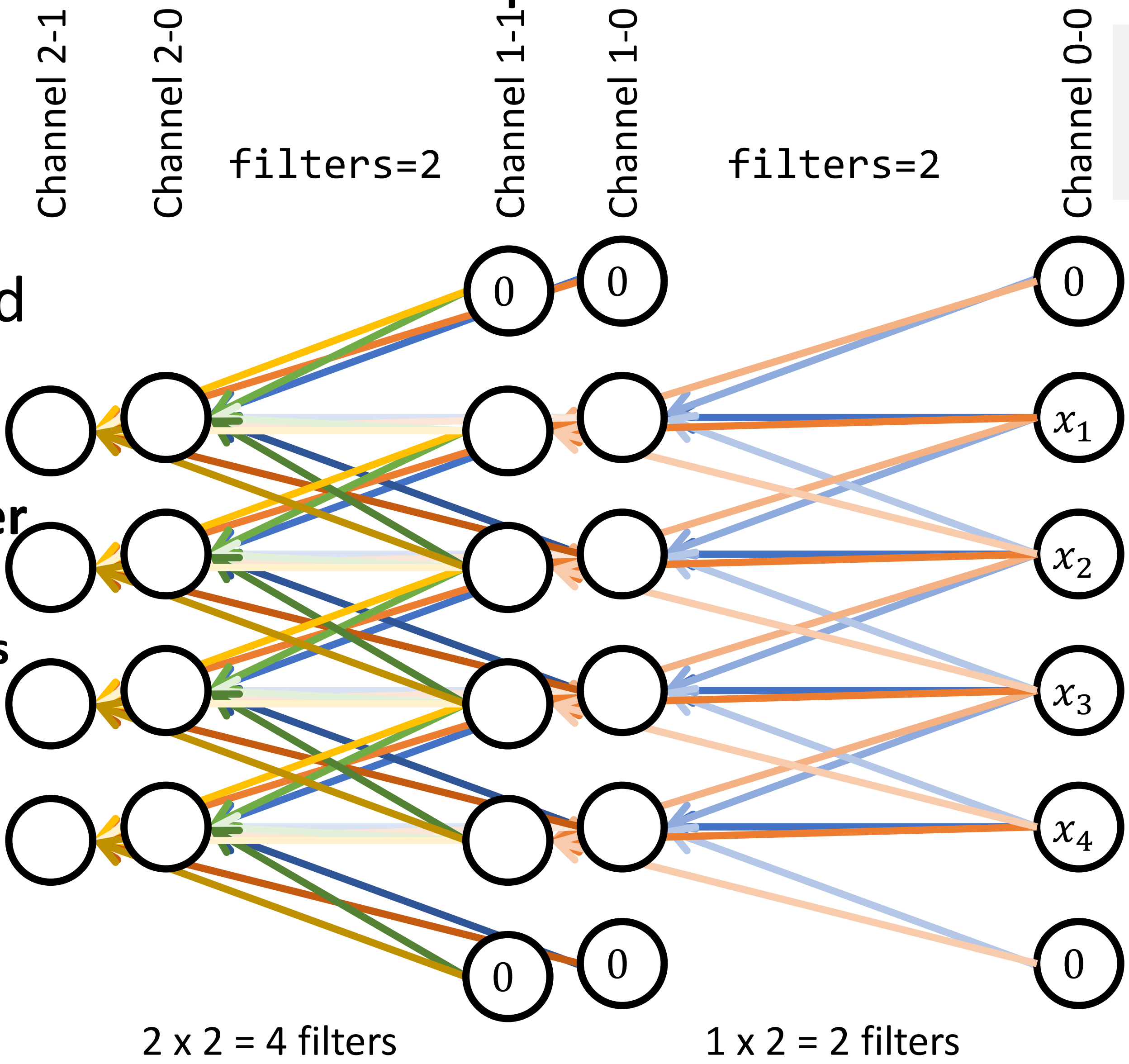
- The filter from Channel 1-1 to Channel 2-0.



"Channel i-j" indicates the j-th channel in the i-th layer

Multi-channel case: examples

- # filters is given by the product of # input channels and # output channels
  - Note: `filters` specifies # filters **per one input channel.**
    - **≠ the total # filters in the layer**
-

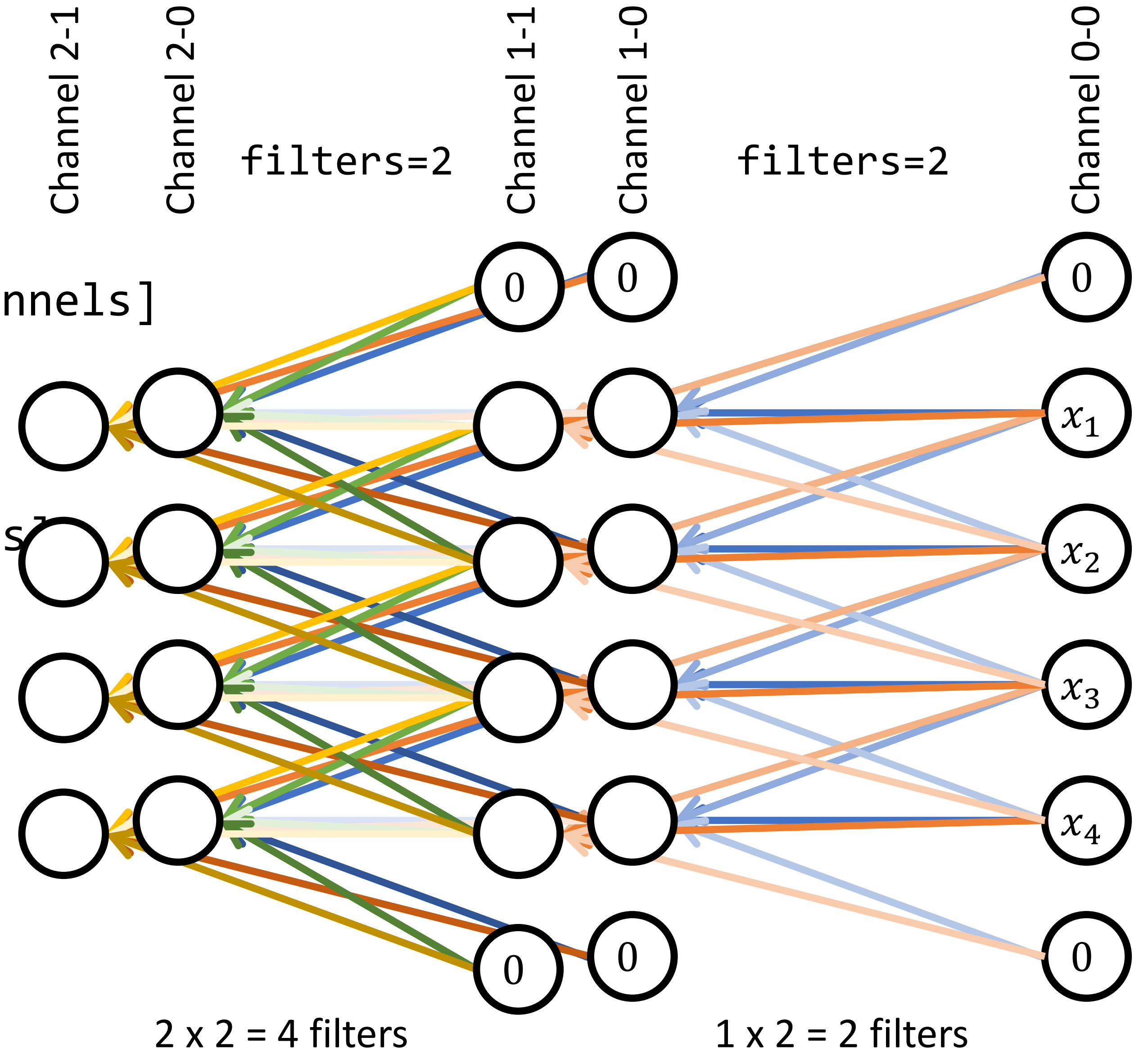


"Channel i-j" indicates the j-th channel in the i-th layer

“Channel i-j” indicates
the j-th channel
in the i-th layer

Conv1D: output shape

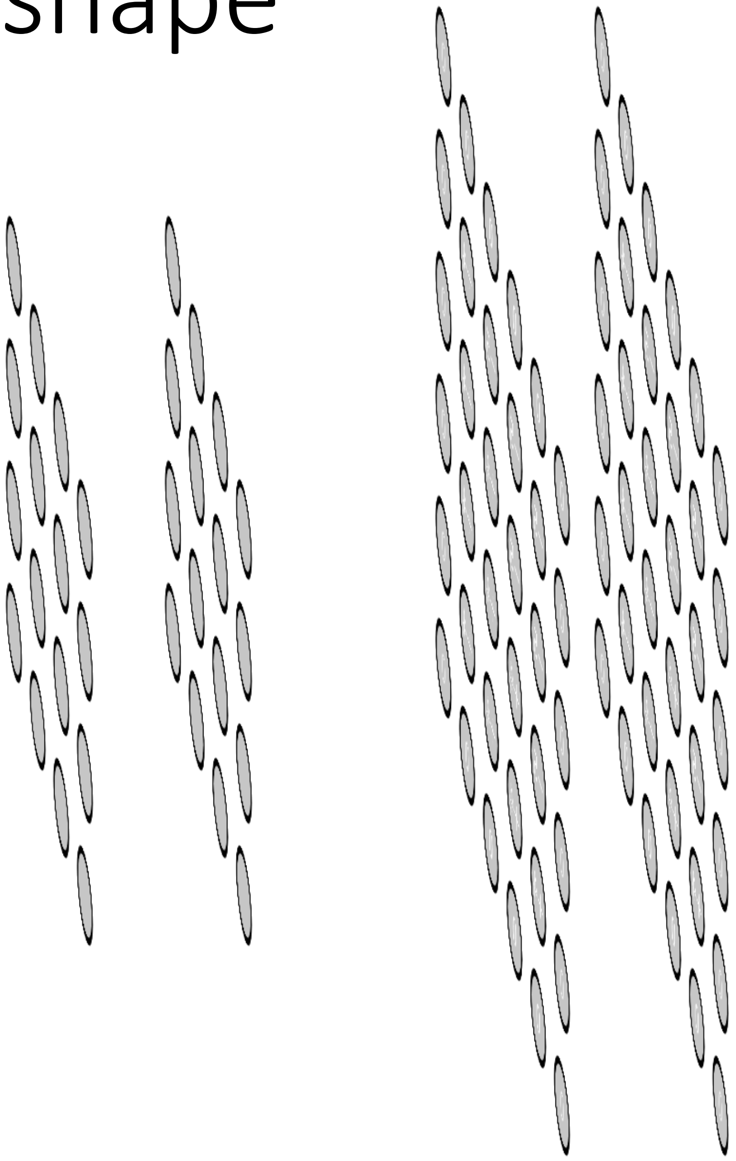
- Suppose that
  - `Input shape:`
    - `[batch_size, in_units, in_channels]`
  - `strides=1, padding=‘same’.`
- Output shape is
    - `[batch_size, in_units, filters]`



filter=2, kernel_size=(3,3), strides=1, padding='same'

2D convolution (**Conv2D**): output shape

- Suppose that
  - `Input shape:`
    - `[batch_size, width, height, in_channels]`
  - `strides=1, padding='same'.`
- Output shape is
    - `[batch_size, width, height, filters]`



Long short term memory

Long short term memory (LSTM) [Hochreiter & Schmidhuber 1997]
Input and output: time-series to time-series

Output shape:
`(batch_size, timesteps, units)`

hyperparameter

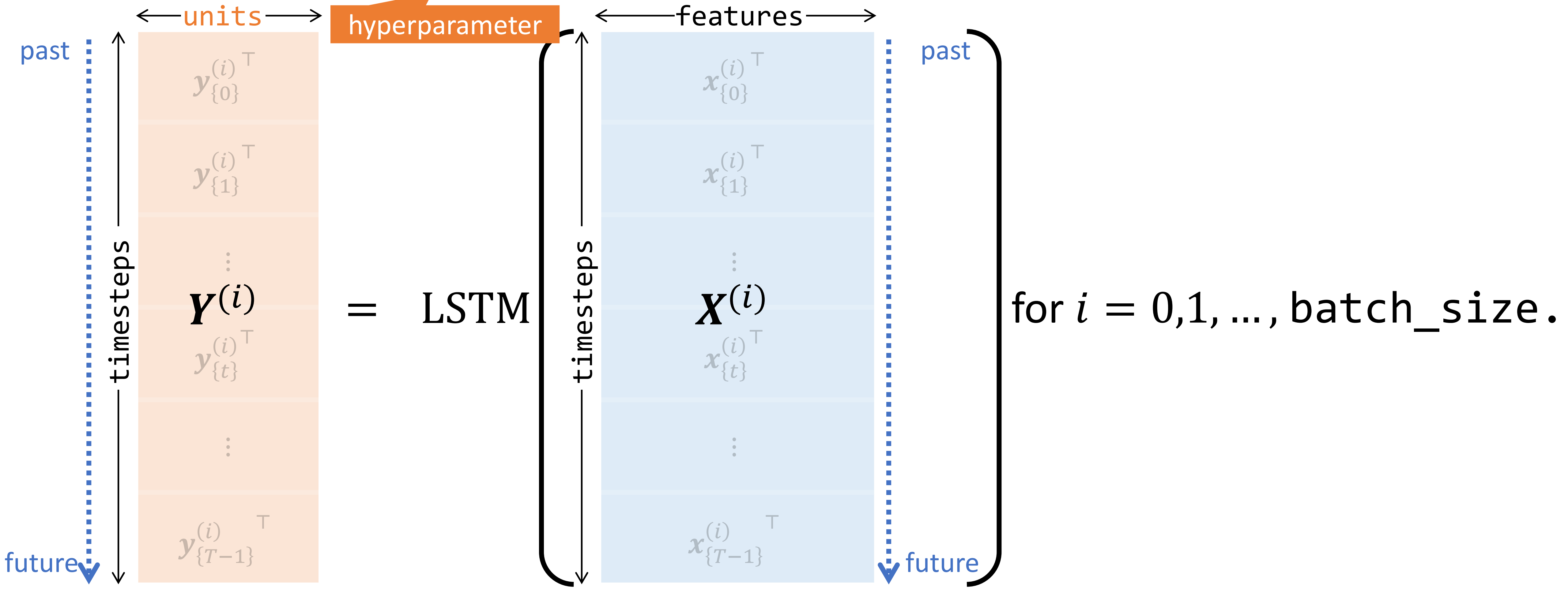
Input shape:
`(batch_size, timesteps, features)`

$$\begin{array}{c} \text{past} \\ \downarrow \\ \texttt{timesteps} \\ \downarrow \\ \text{future} \end{array} \overbrace{\begin{bmatrix} {\boldsymbol{y}^{(i)}_{\{0\}}}^{\top} \\ {\boldsymbol{y}^{(i)}_{\{1\}}}^{\top} \\ \vdots \\ {\boldsymbol{y}^{(i)}_{\{t\}}}^{\top} \\ \vdots \\ {\boldsymbol{y}^{(i)}_{\{T-1\}}}^{\top} \end{bmatrix}}^{\texttt{units}} = \left(\texttt{timesteps} \overbrace{\begin{bmatrix} {\boldsymbol{x}^{(i)}_{\{0\}}}^{\top} \\ {\boldsymbol{x}^{(i)}_{\{1\}}}^{\top} \\ \vdots \\ {\boldsymbol{x}^{(i)}_{\{t\}}}^{\top} \\ \vdots \\ {\boldsymbol{x}^{(i)}_{\{T-1\}}}^{\top} \end{bmatrix}}^{\texttt{features}} \begin{array}{c} \text{past} \\ \downarrow \\ \\ \downarrow \\ \text{future} \end{array} \right) \quad \text{for } i = 0,1,\dots,\texttt{batch\_size}.$$

Long short term memory (LSTM) [Hochreiter & Schmidhuber 1997]
Input and output: time-series to time-series

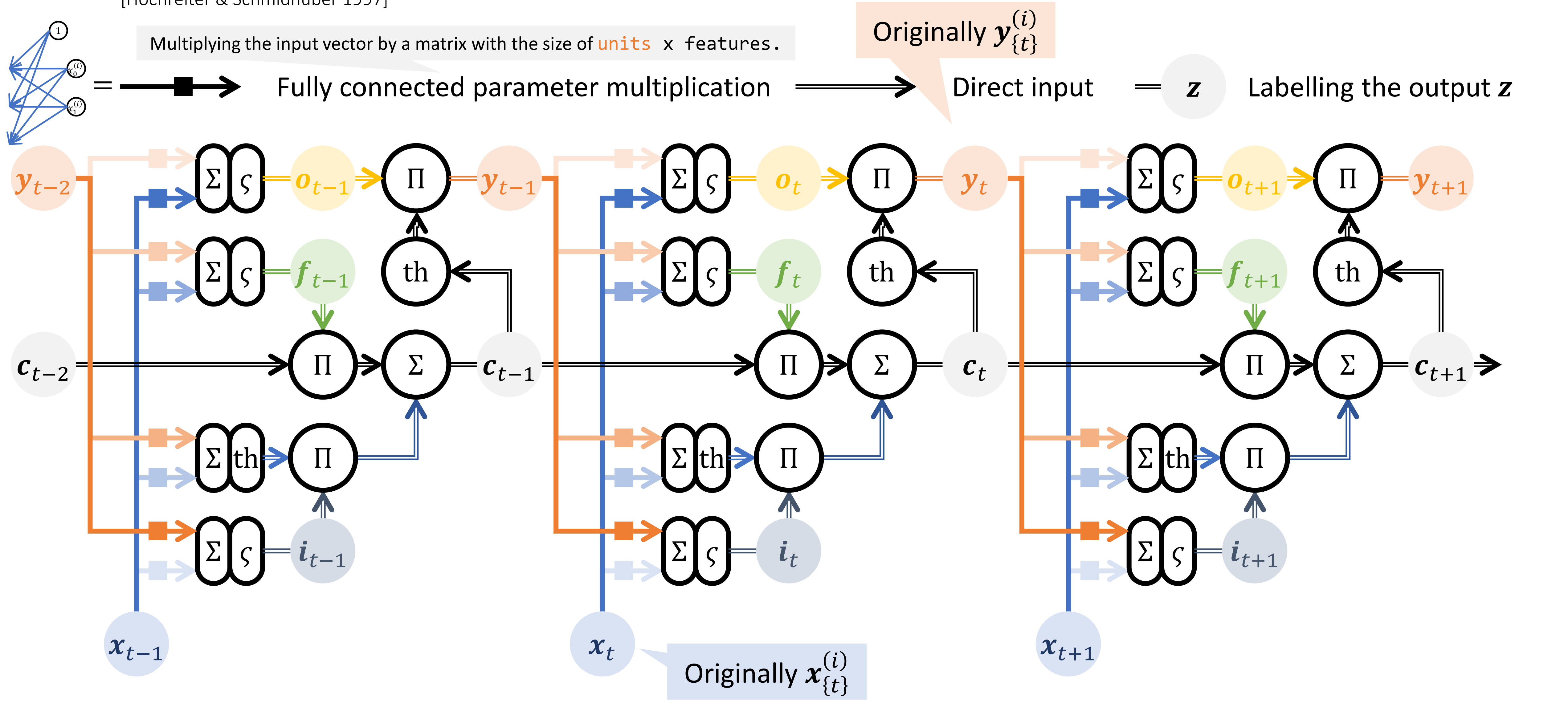
Output shape:
`(batch_size, timesteps, units)`

Input shape:
`(batch_size, timesteps, features)`



Long short term memory (LSTM)

[Hochreiter & Schmidhuber 1997]



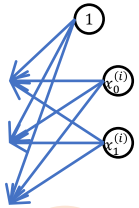
Long short term memory (LSTM)

[Hochreiter & Schmidhuber 1997]

Hyperparameter that determines the dimension of $\boldsymbol{y}_t, \boldsymbol{c}_t, \boldsymbol{i}_t, \boldsymbol{f}_t, \boldsymbol{o}_t$.

Multiplying the input vector by a matrix with the size of `units x features`.

Originally $\boldsymbol{y}^{(i)}_{\{t\}}$



= Fully connected parameter multiplication

Direct input

$\boldsymbol{z}$ Labelling the output $\boldsymbol{z}$