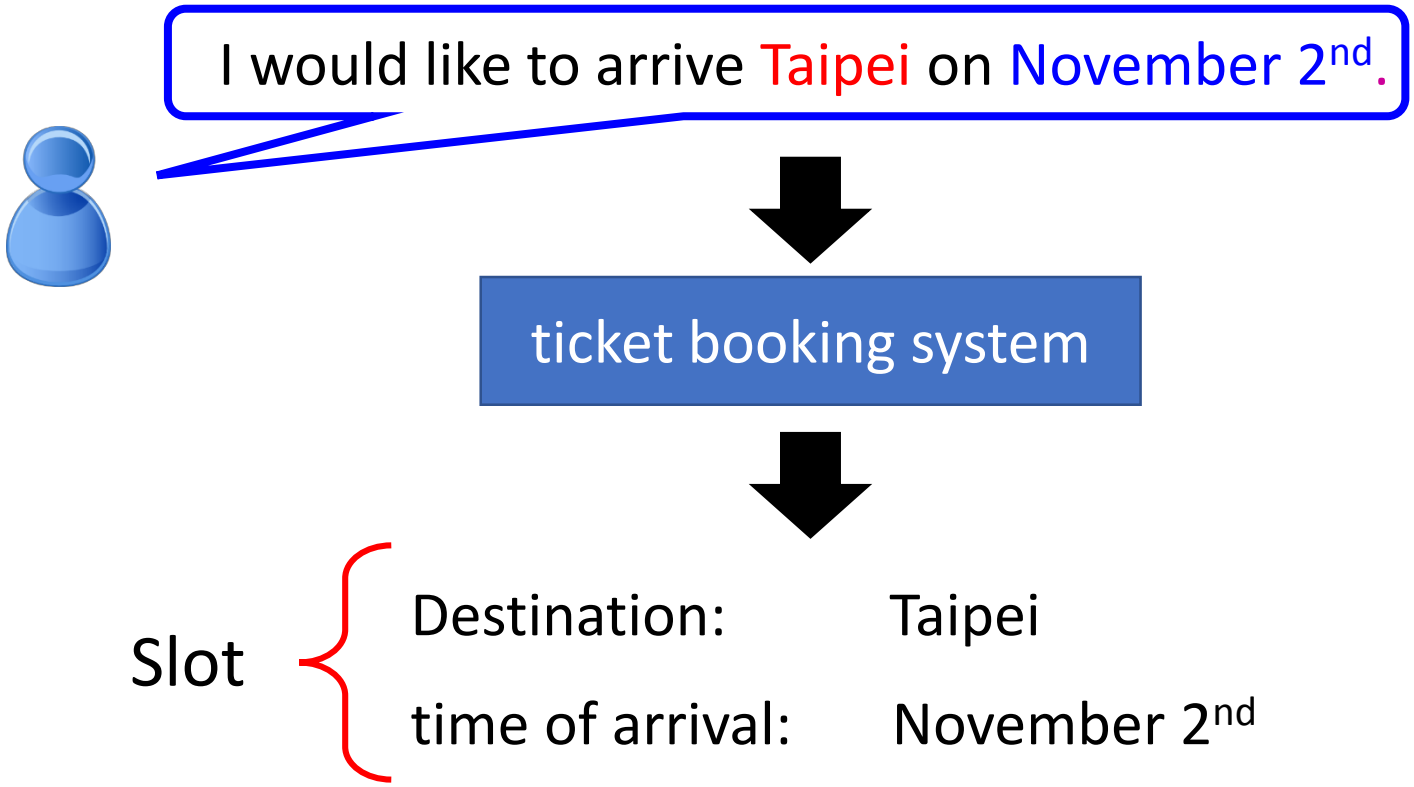
Recurrent Neural Network (RNN)

Example Application

- Slot Filling



Example Application

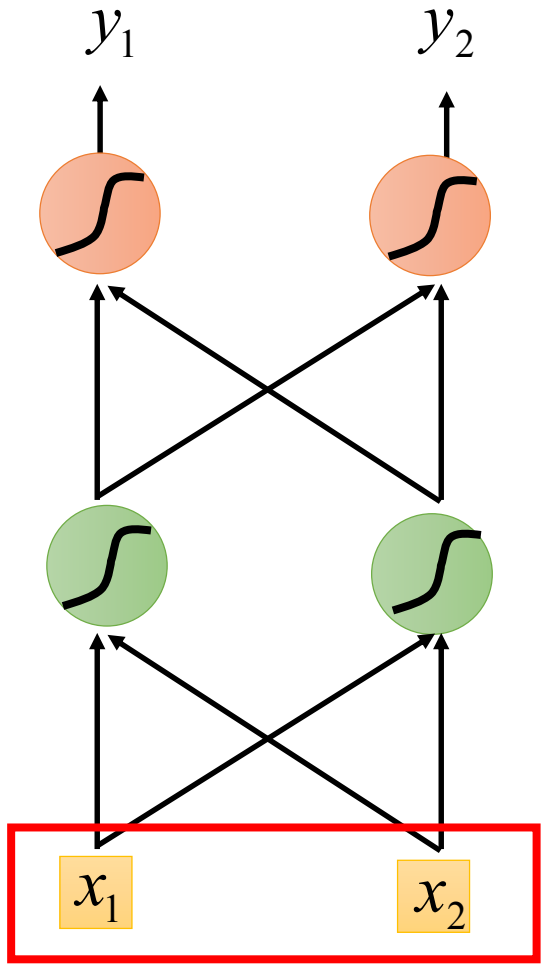
Solving slot filling by Feedforward network?

Input: a word

(Each word is represented as a vector)

若使用一般的 Feedforward Network 來解決 Slot Filling 問題：輸入一個詞彙，輸出即是詞彙屬於每一個 Slot 的機率

前提：必須將輸入的詞彙用「Vector」表示，才能丟進 Neural Network 中！



Taipei

用 Vector 表示詞彙最簡單的方法！

1-of-N encoding

How to represent each word as a vector?

1-of-N Encoding lexicon = {apple, bag, cat, dog, elephant}

The vector is lexicon size.

Each dimension corresponds to a word in the lexicon

The dimension for the word is 1, and others are 0

apple = [1 0 0 0 0]

bag = [0 1 0 0 0]

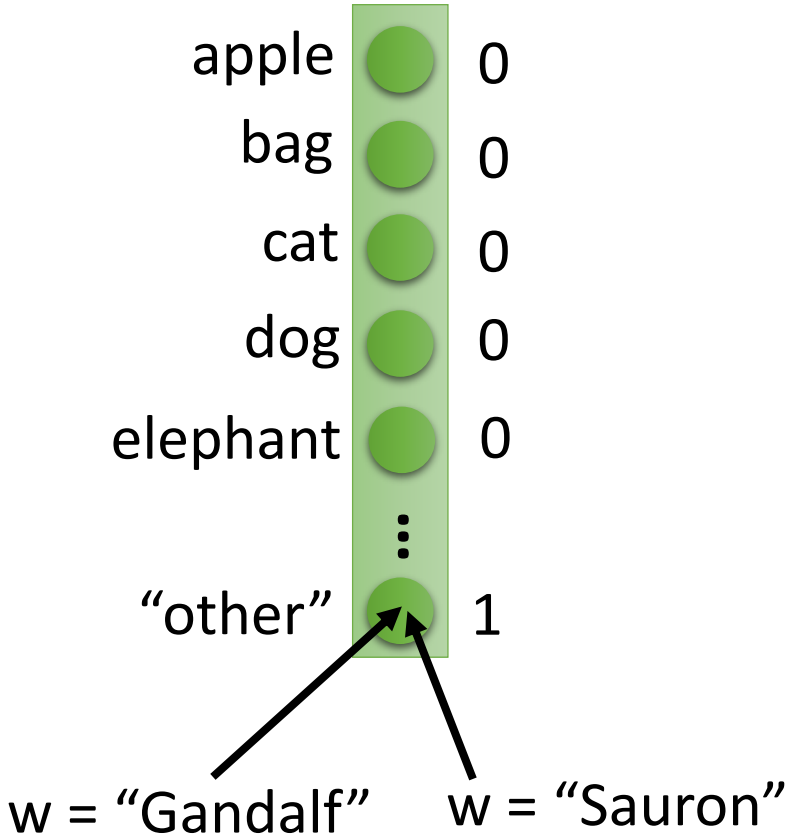
cat = [0 0 1 0 0]

dog = [0 0 0 1 0]

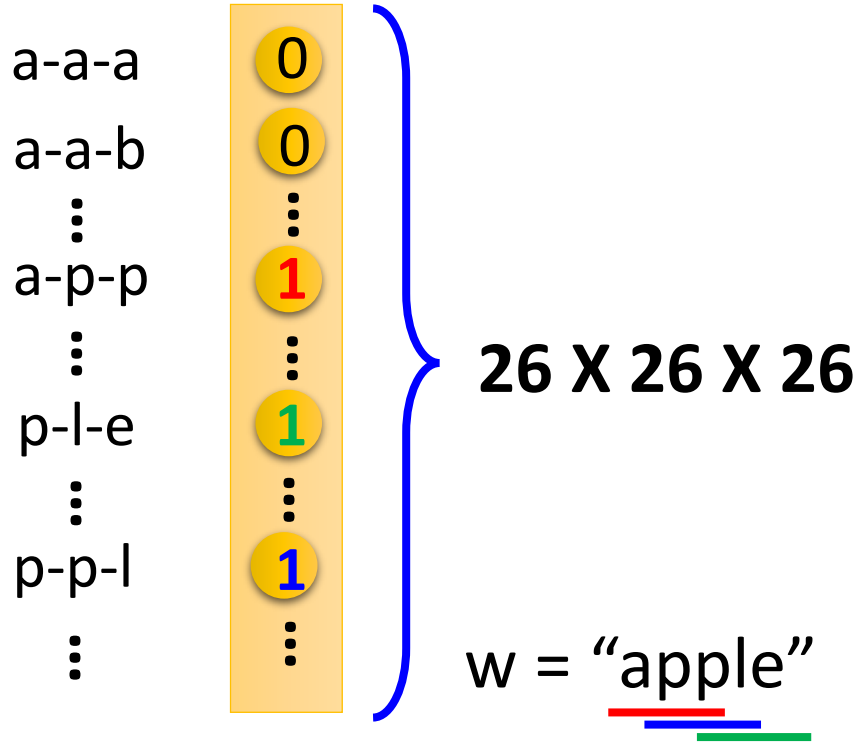
elephant = [0 0 0 0 1]

Beyond 1-of-N encoding

Dimension for "Other"



Word hashing



Example Application

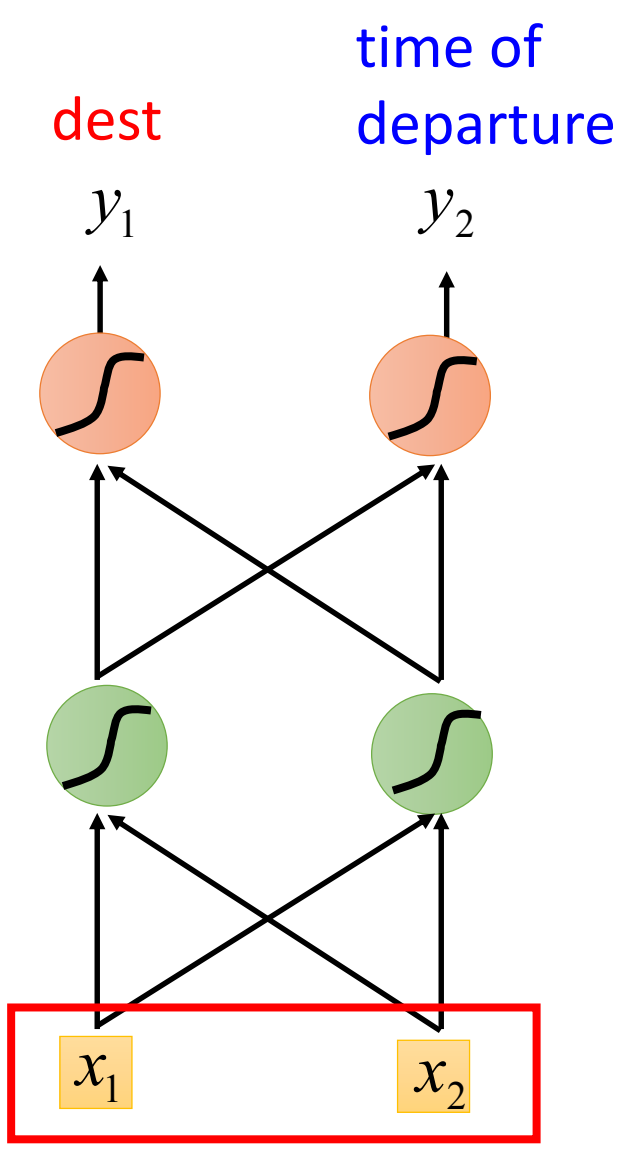
Solving slot filling by Feedforward network?

Input: a word

(Each word is represented as a vector)

Output:

Probability distribution that the input word belonging to the slots



Example Application

arrive Taipei on November 2nd

other dest other time time

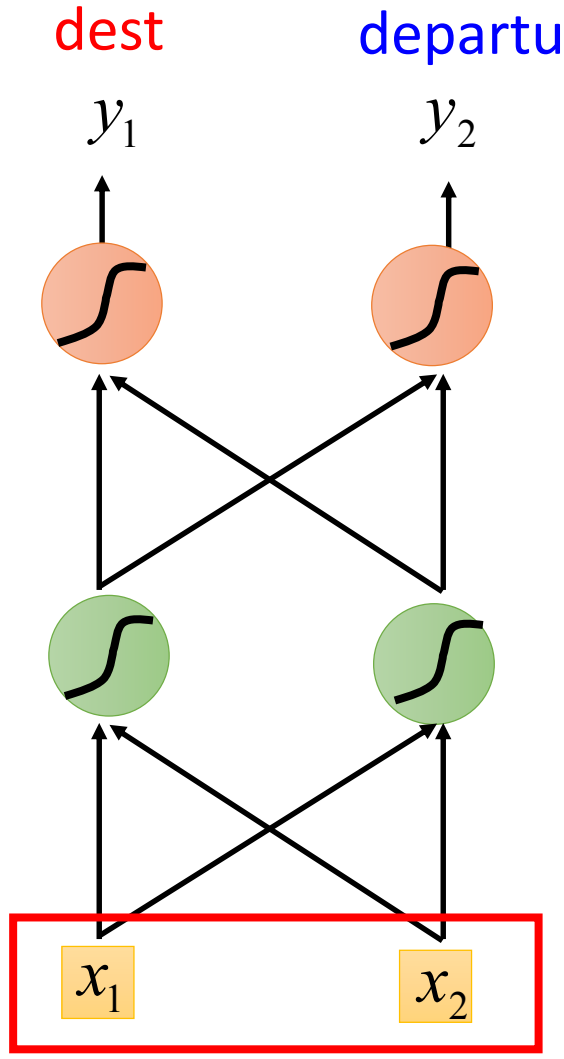
Problem?

leave Taipei on November 2nd

place of departure

Neural network needs memory!

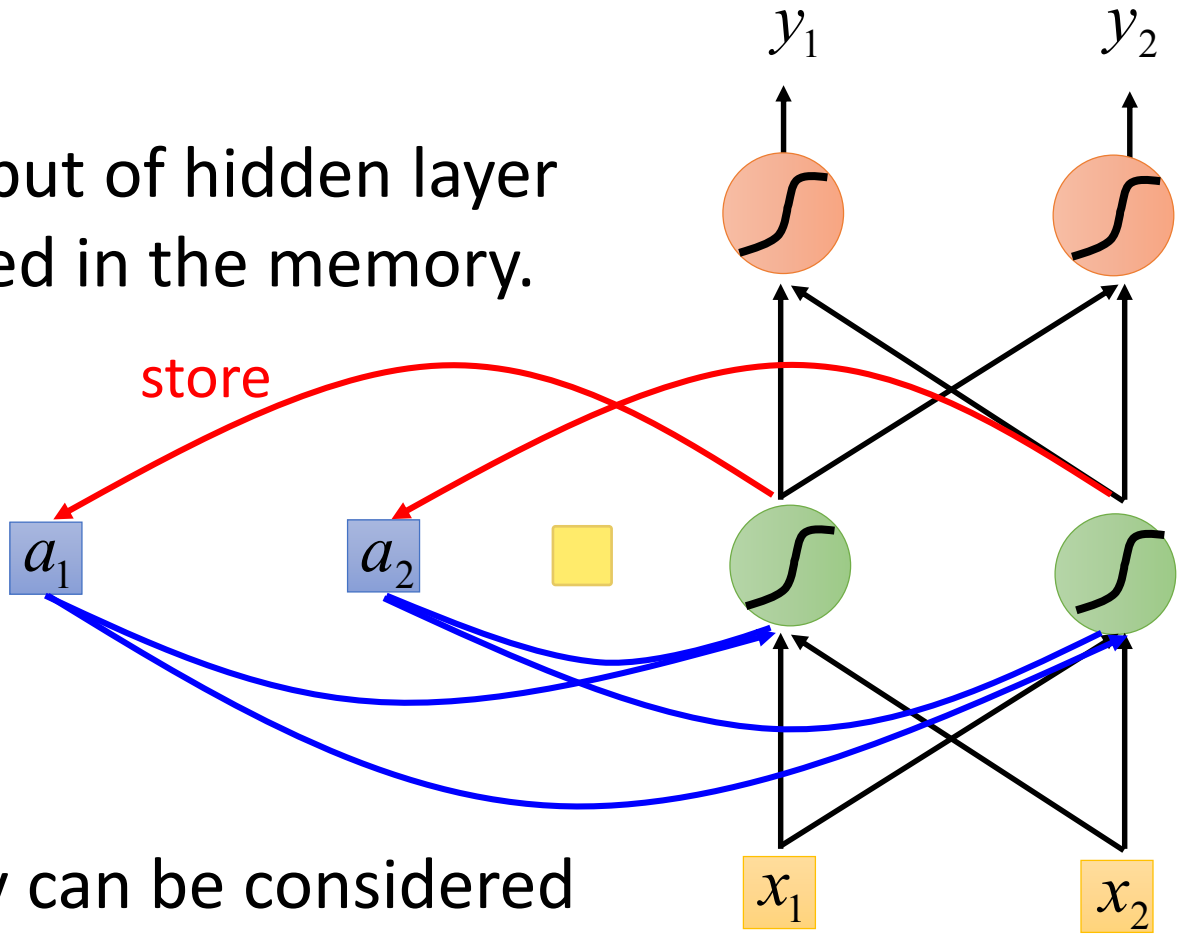
Taipei



具有記憶力的 Neural Network！

Recurrent Neural Network (RNN)

The output of hidden layer are stored in the memory.

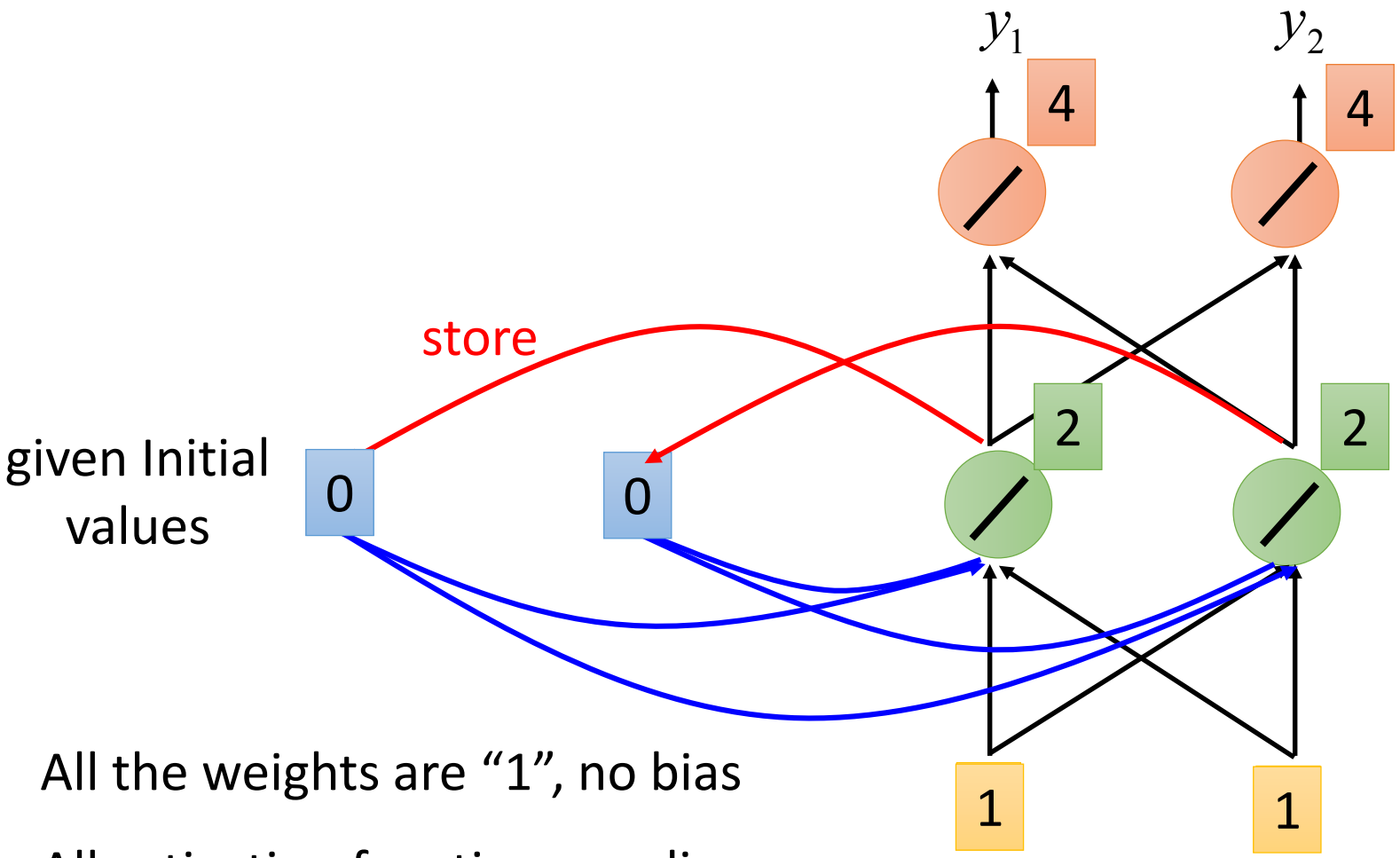


Memory can be considered as another input.

Example

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}2\\2\end{bmatrix}$ ……

output sequence: $\begin{bmatrix}4\\4\end{bmatrix}$



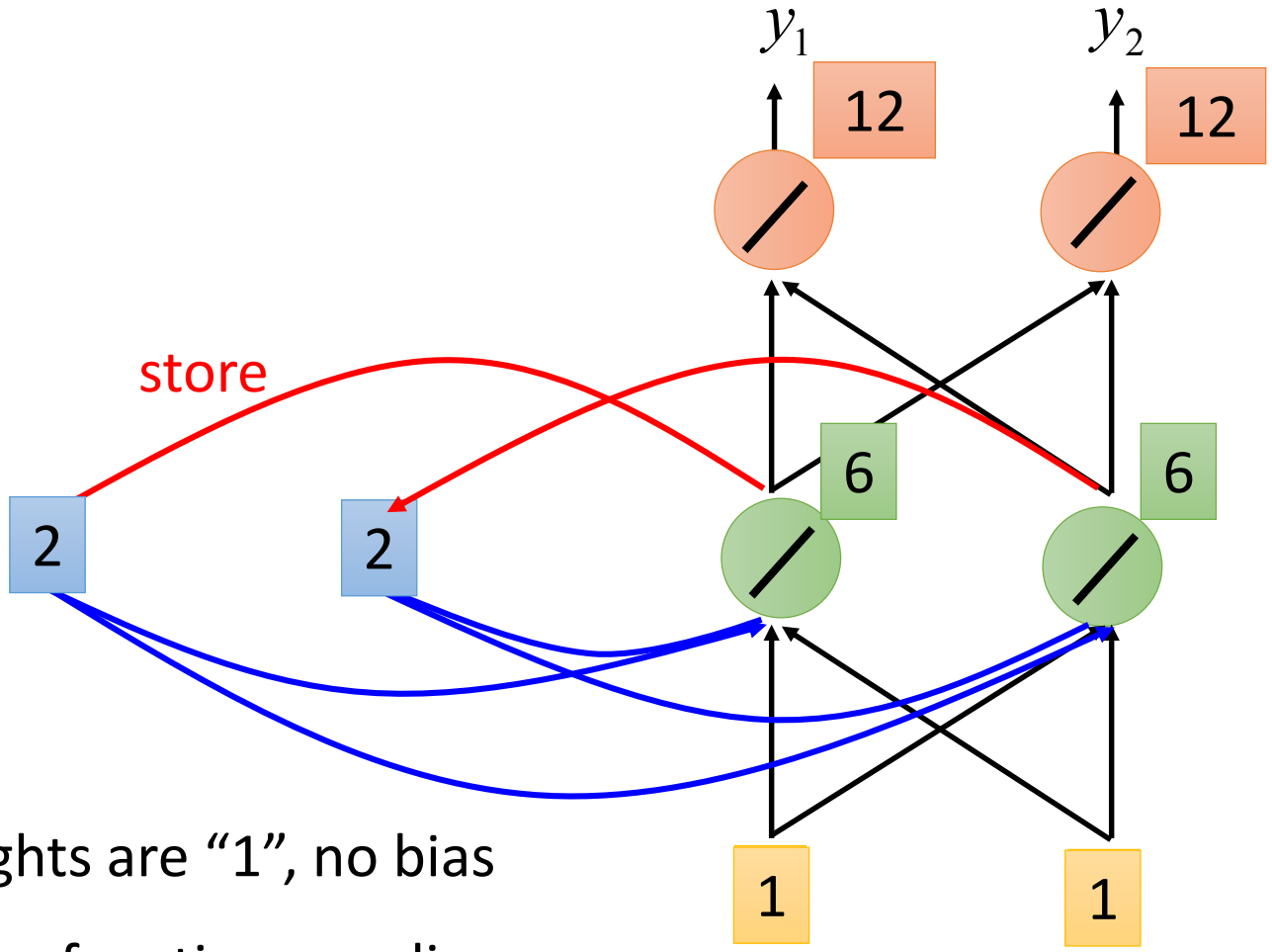
All the weights are “1”, no bias

All activation functions are linear

Example

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}2\\2\end{bmatrix}$ ……

output sequence: $\begin{bmatrix}4\\4\end{bmatrix}$ $\begin{bmatrix}12\\12\end{bmatrix}$



All the weights are “1”, no bias

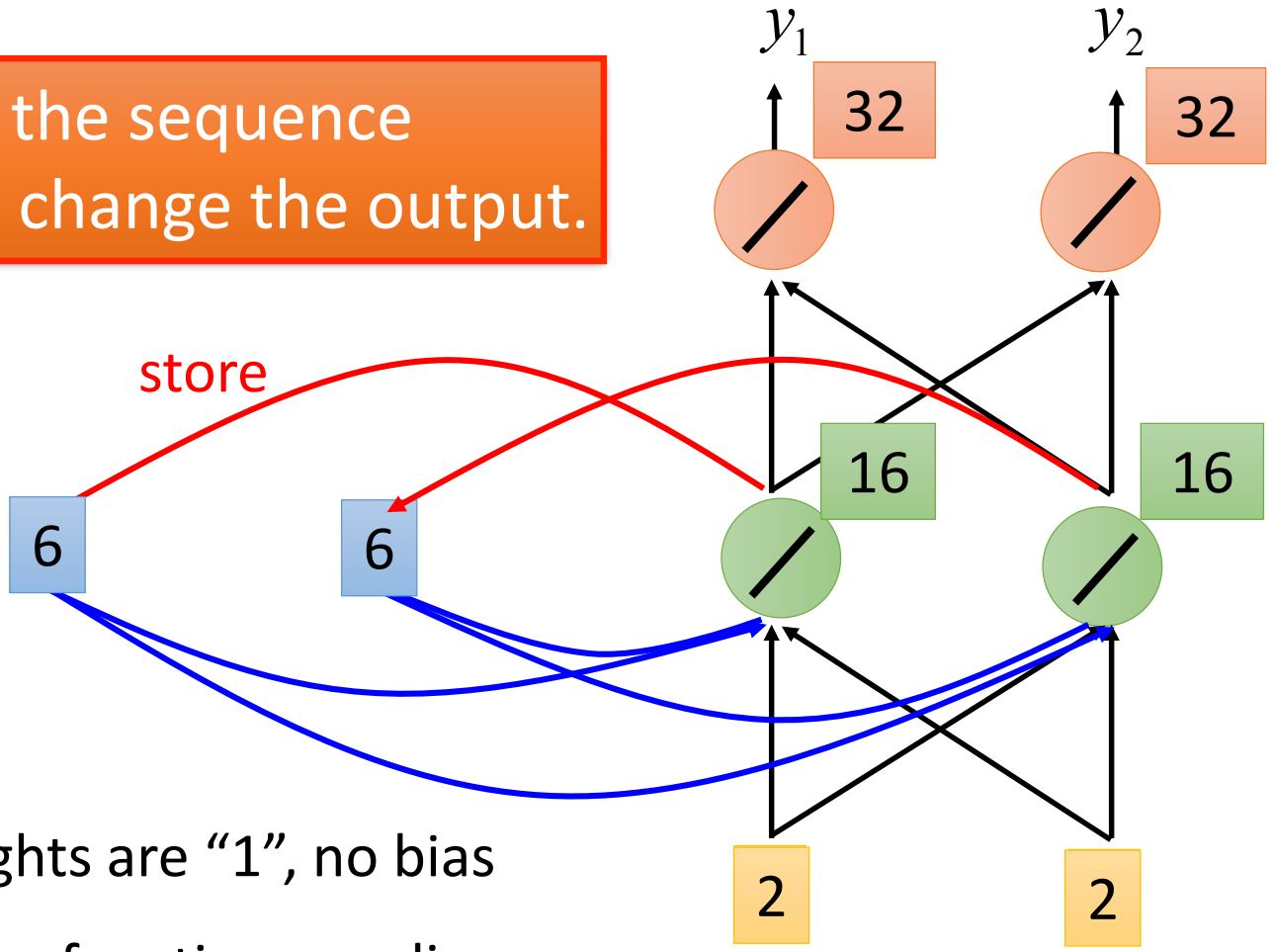
All activation functions are linear

Example

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}\ \begin{bmatrix}1\\1\end{bmatrix}\ \begin{bmatrix}2\\2\end{bmatrix}\ \cdots\cdots$

output sequence: $\begin{bmatrix}4\\4\end{bmatrix}\begin{bmatrix}12\\12\end{bmatrix}\begin{bmatrix}32\\32\end{bmatrix}$

Changing the sequence order will change the output.



All the weights are “1”, no bias

All activation functions are linear

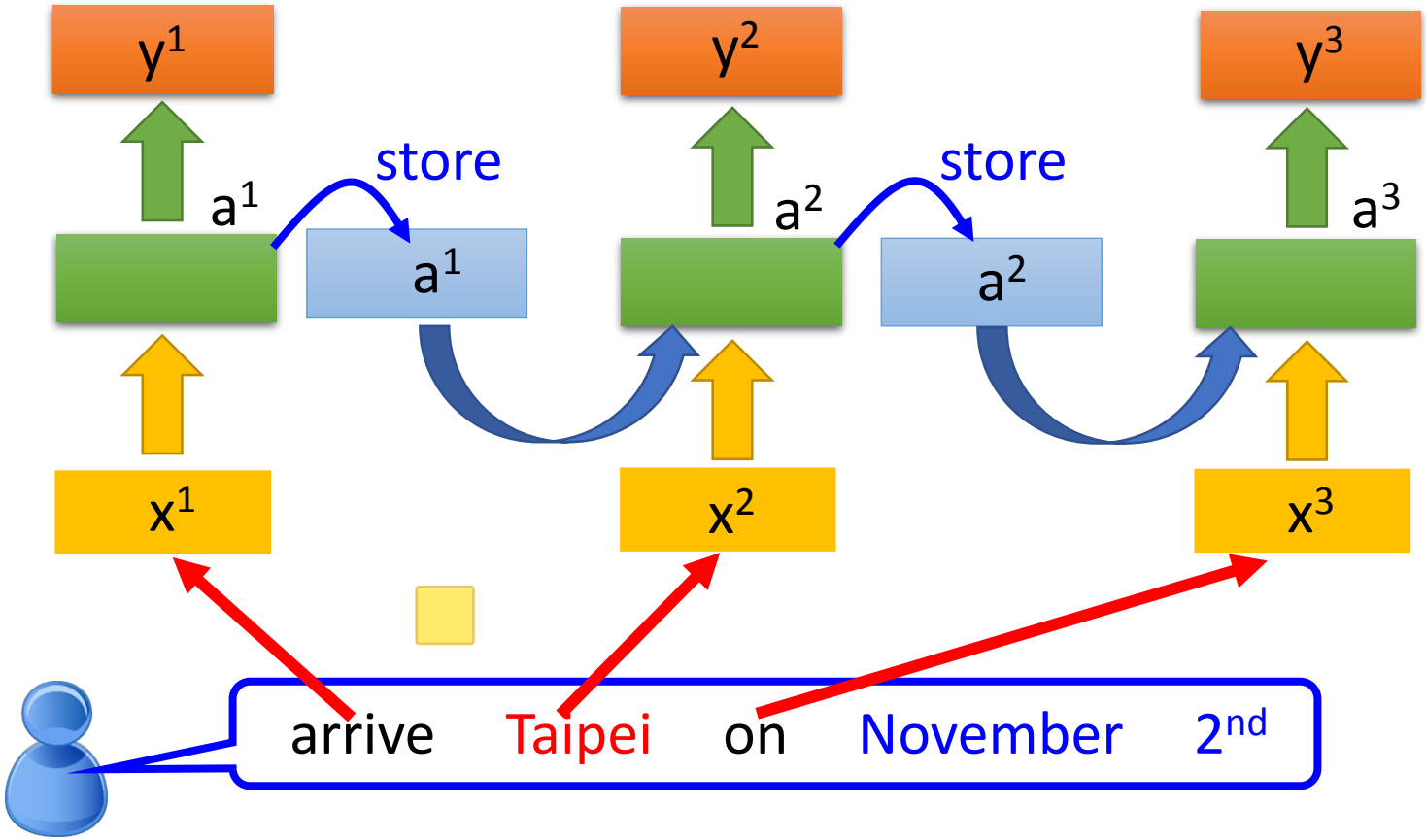
RNN

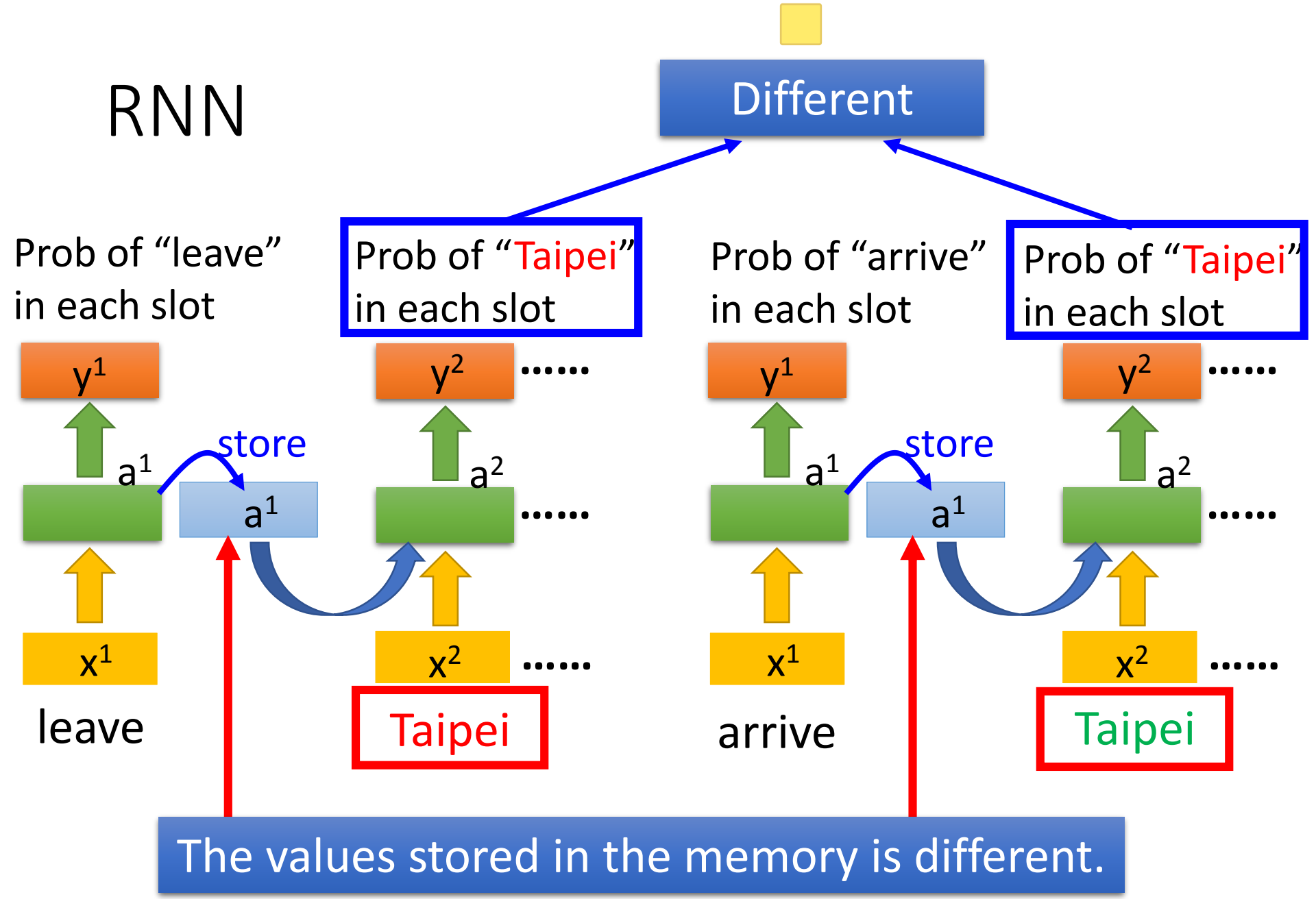
The same network is used again and again.

Probability of "arrive" in each slot

Probability of "Taipei" in each slot

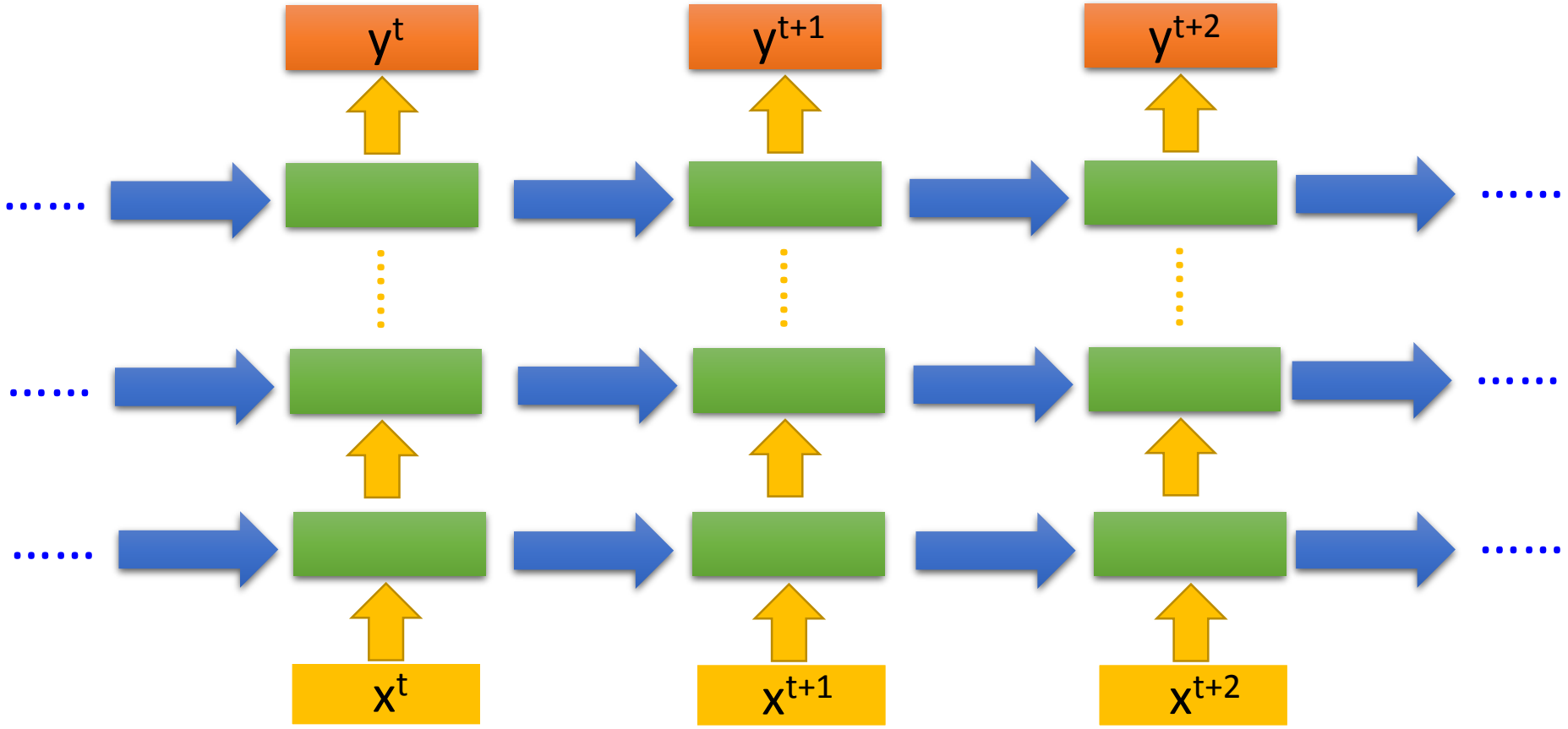
Probability of "on" in each slot



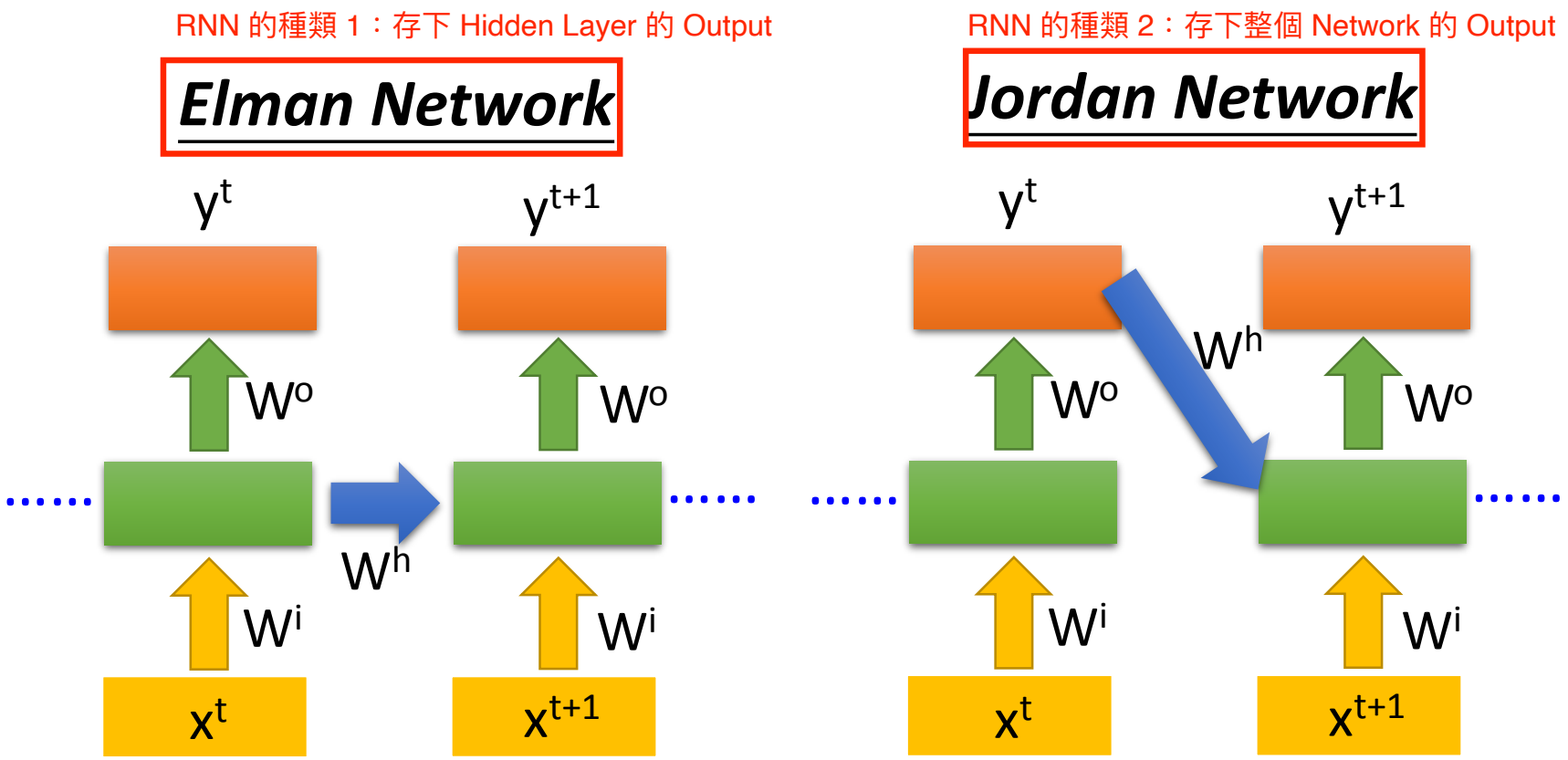

RNN
Different
Prob of "leave" in each slot
Prob of "Taipei" in each slot
Prob of "arrive" in each slot
Prob of "Taipei" in each slot
store
store
leave
Taipei
arrive
Taipei
The values stored in the memory is different.

意思：在 RNN 中，也可以有很多層 Hidden Layer！

Of course it can be deep …

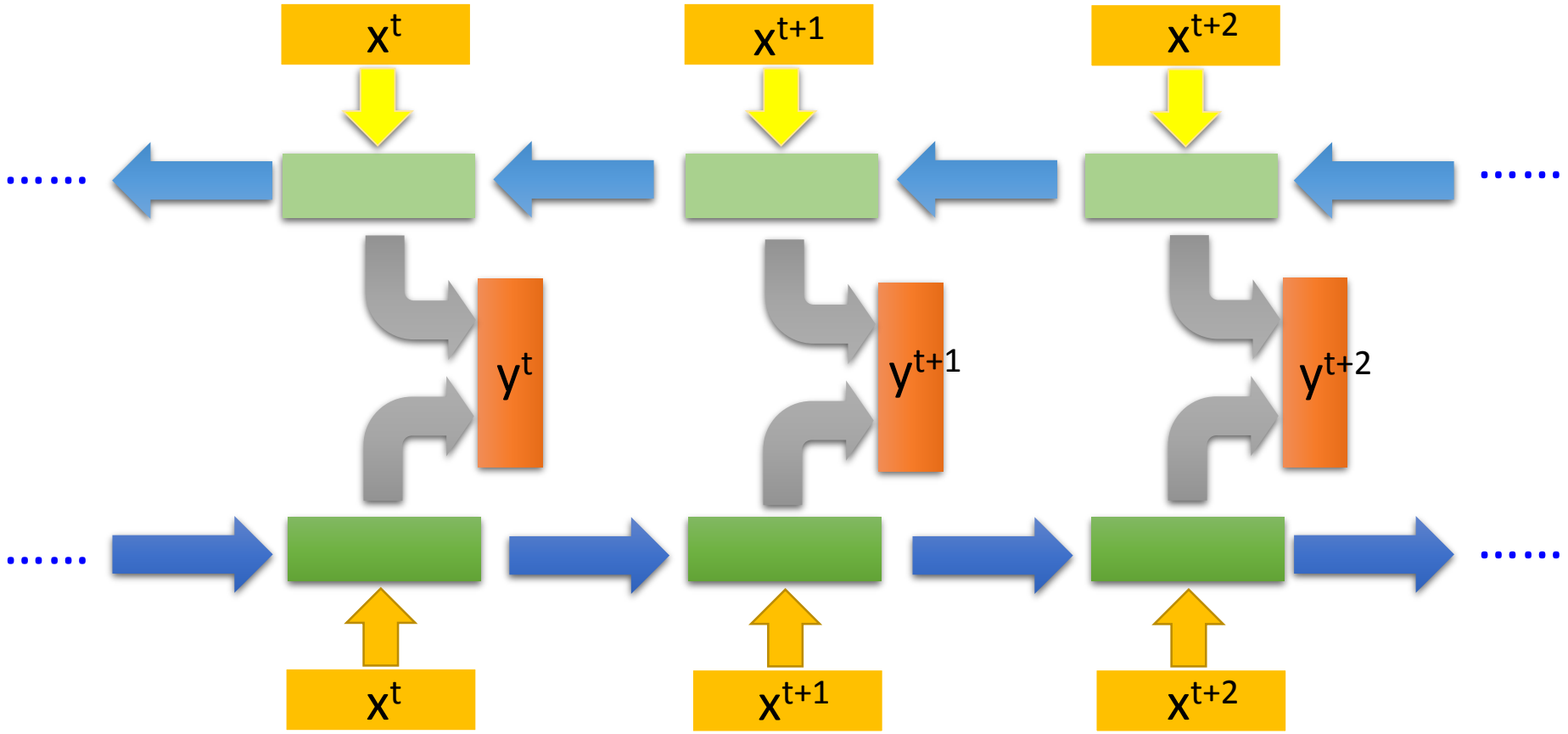


Elman Network & Jordan Network

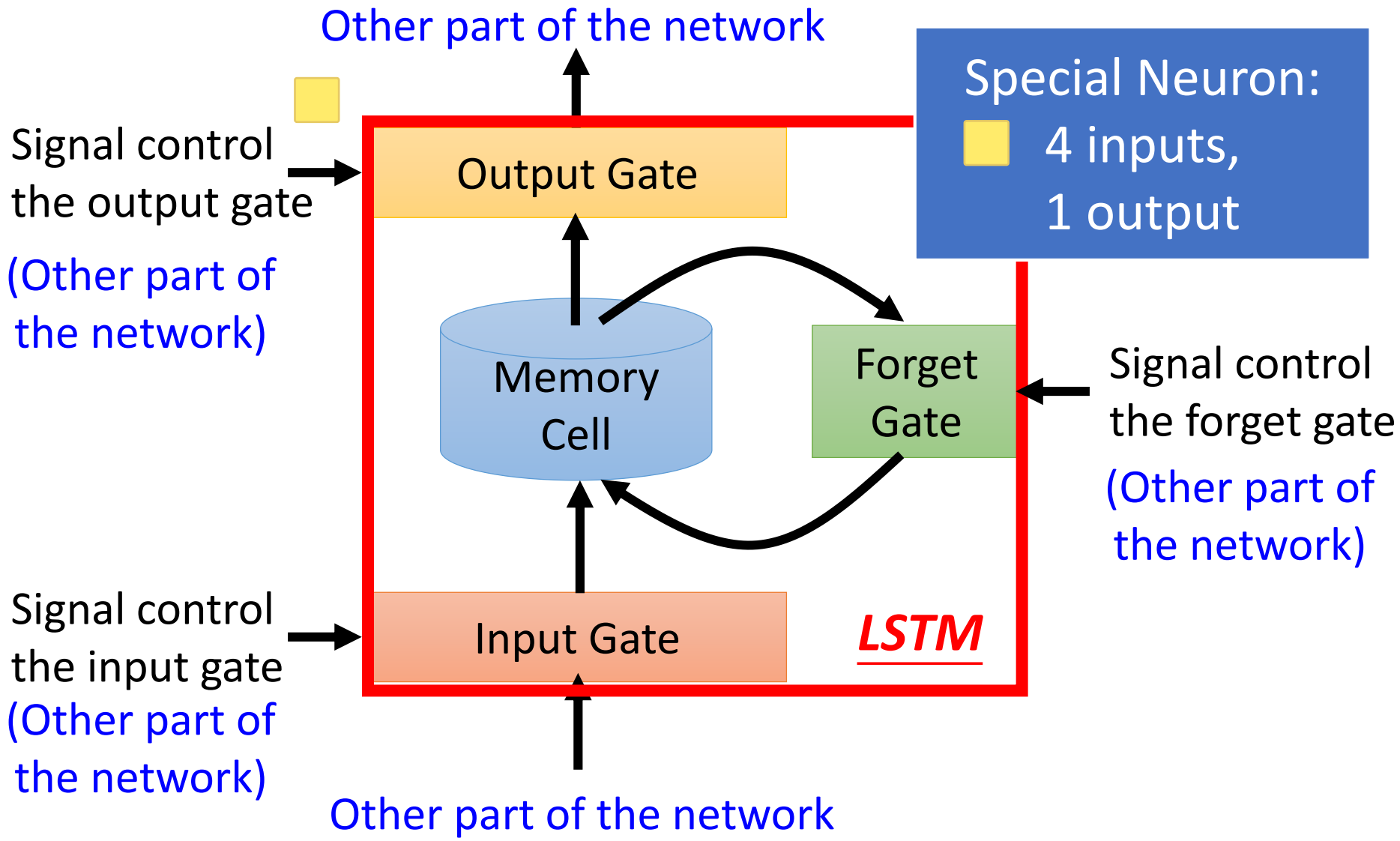


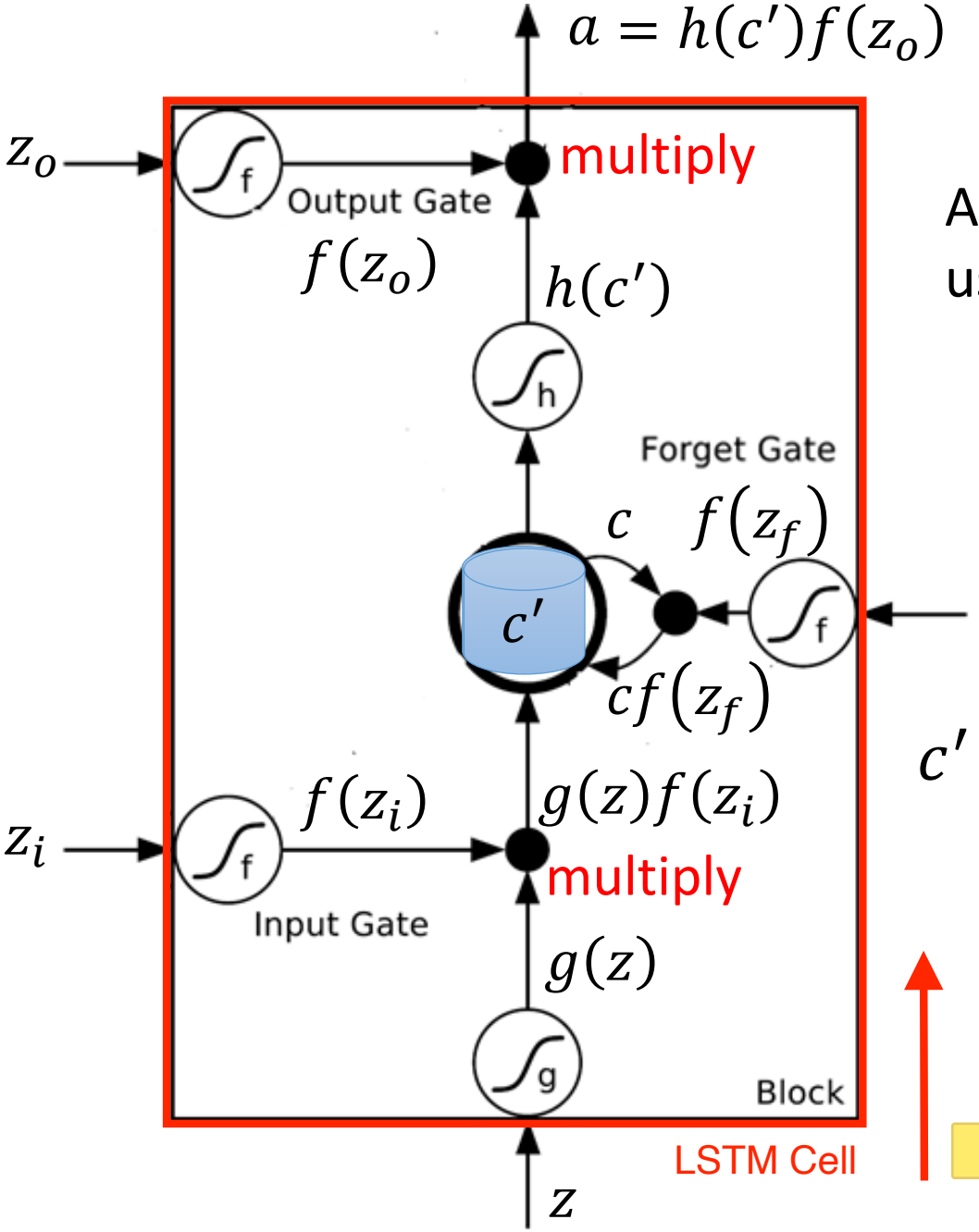
RNN 的種類 3

Bidirectional RNN



Long Short-term Memory (LSTM)



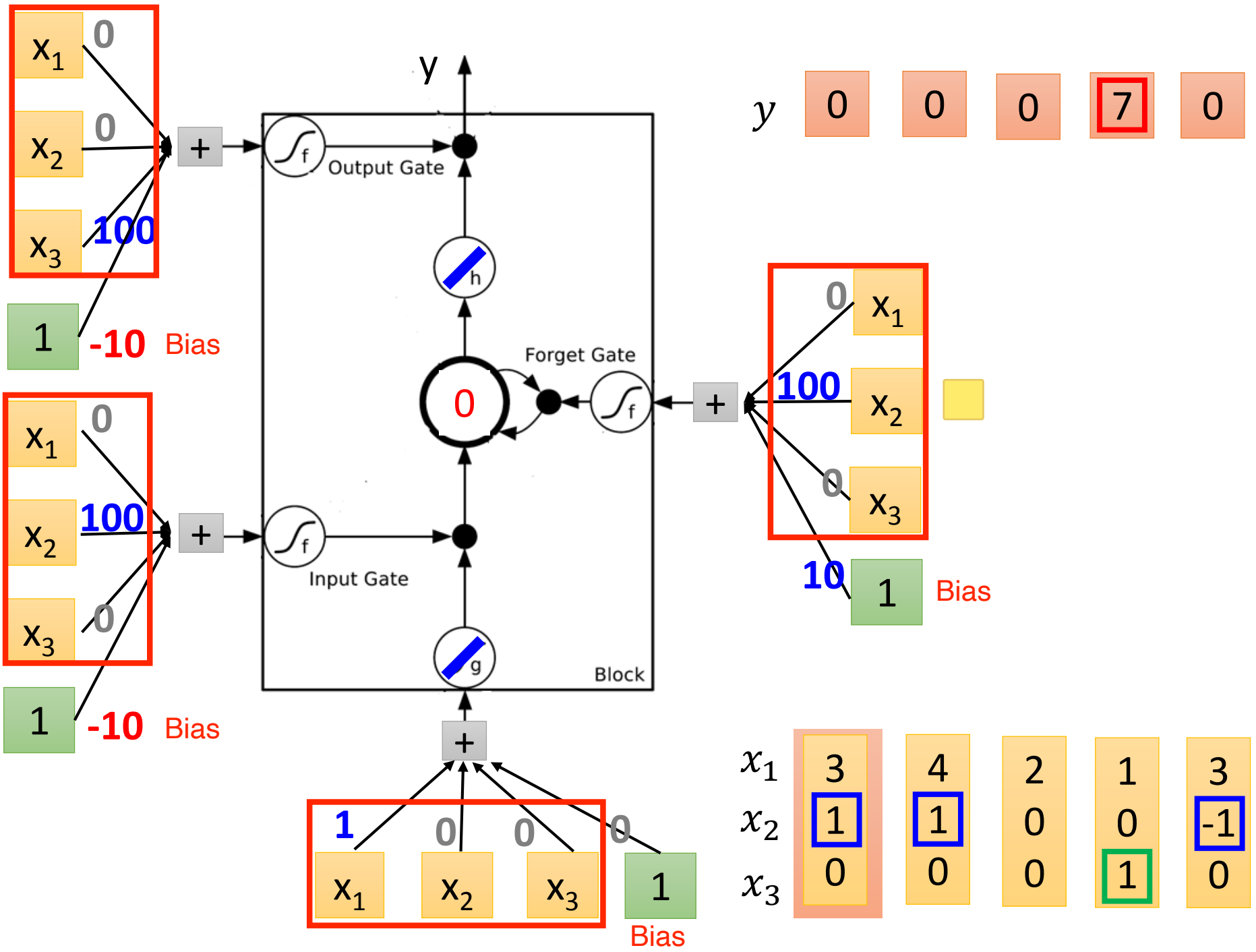


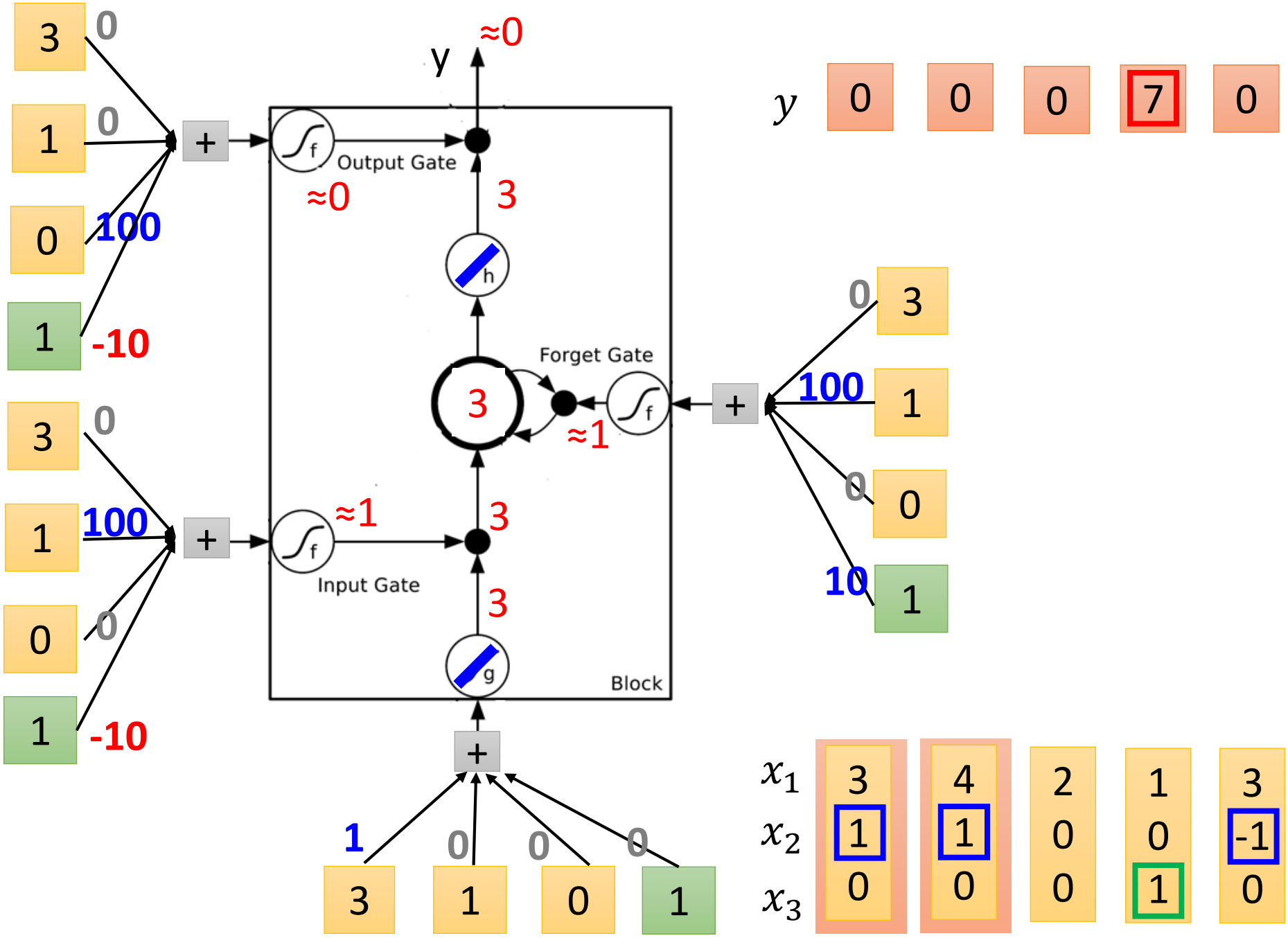
Activation function f is usually a sigmoid function

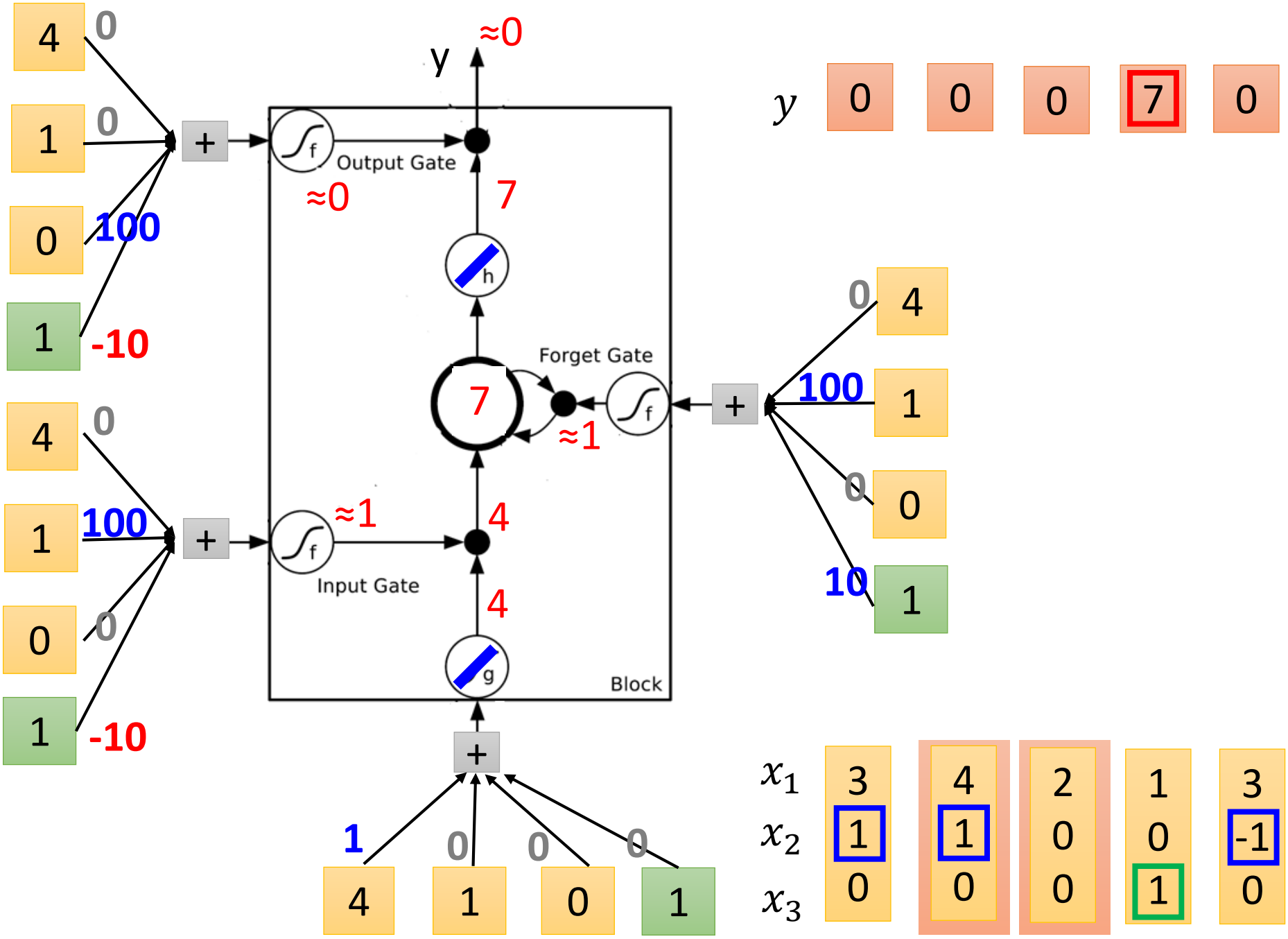
Between 0 and 1

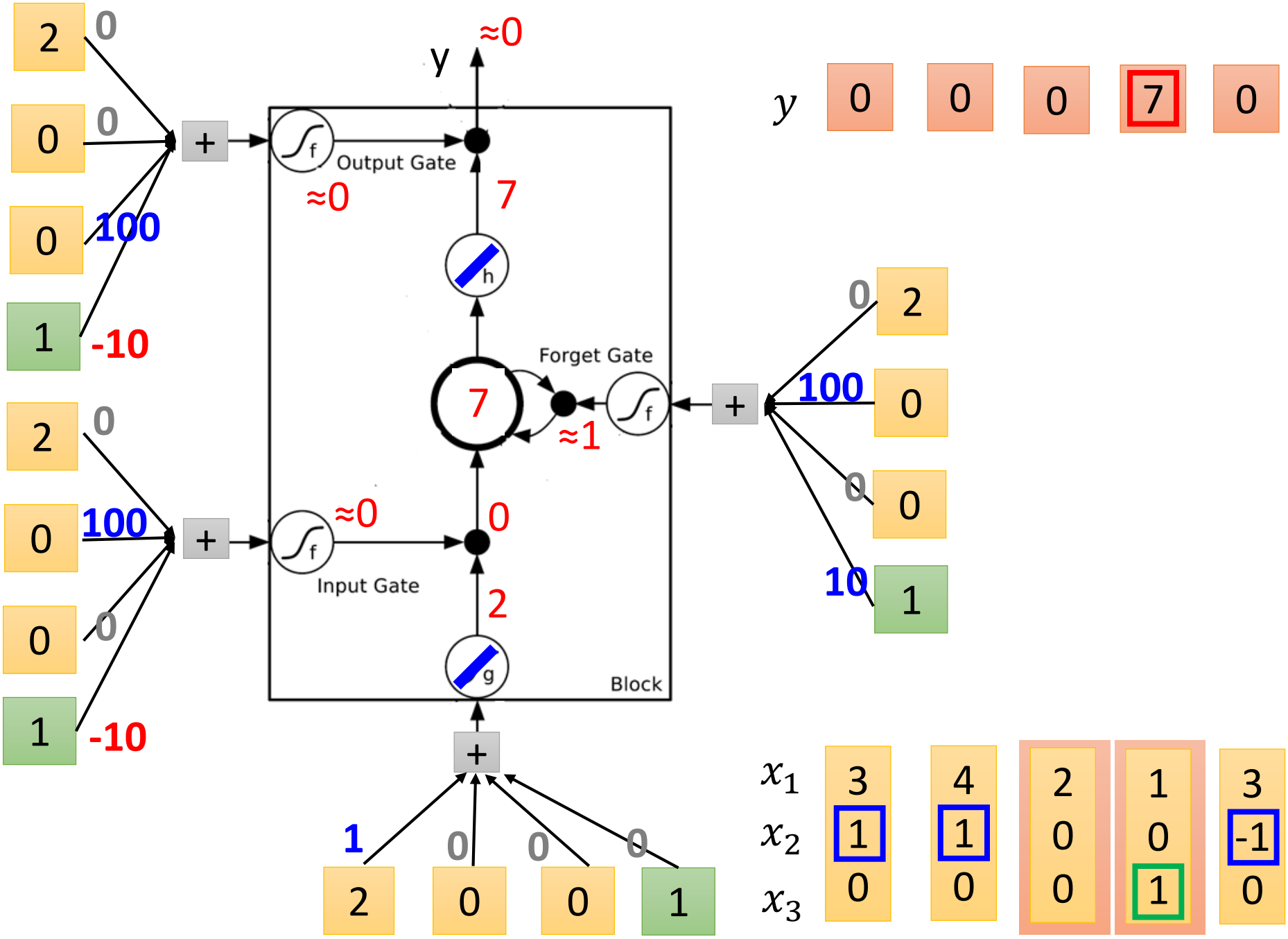
Mimic open and close gate

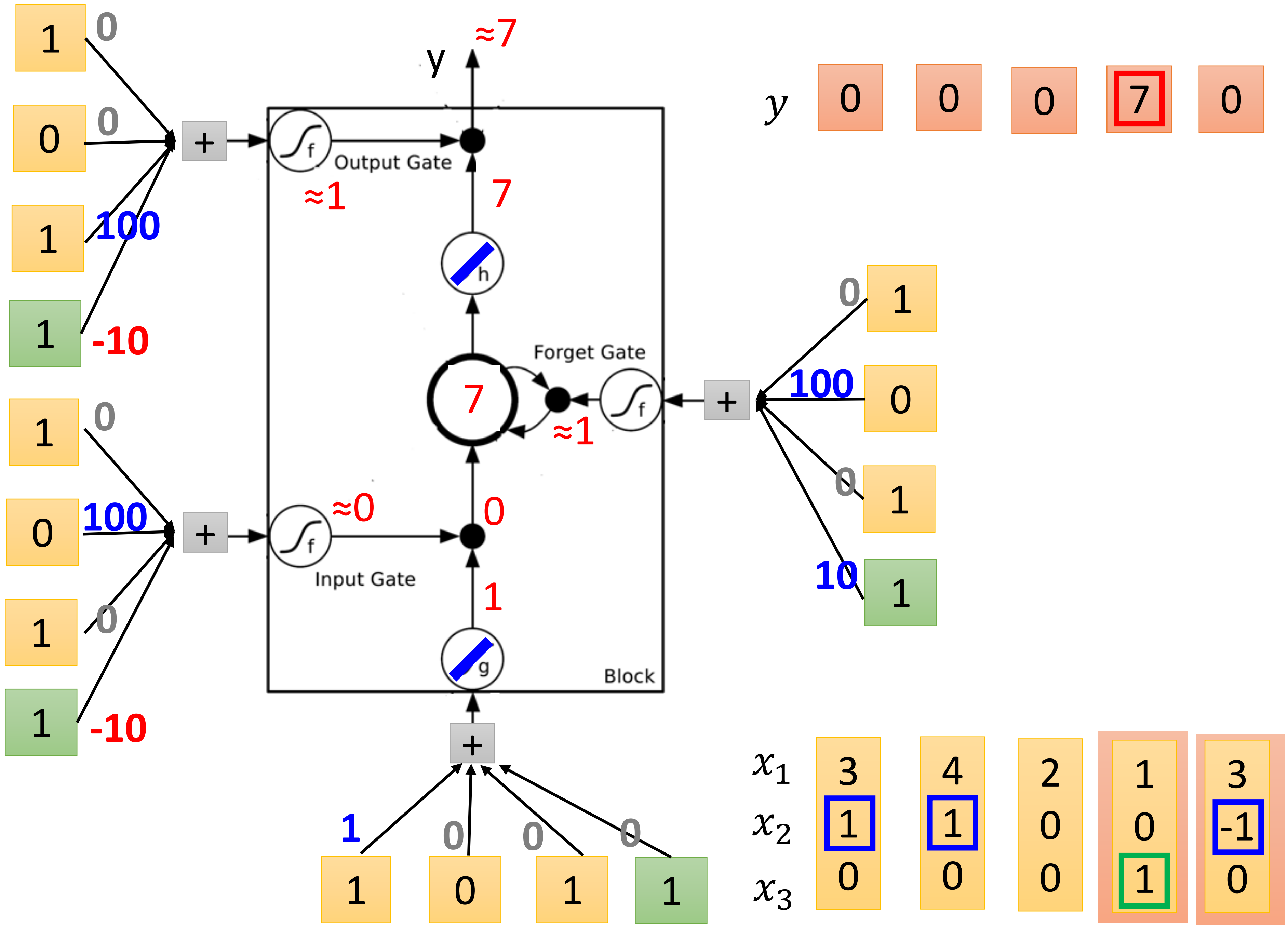
$$c' = g(z)f(z_i) + cf(z_f)$$

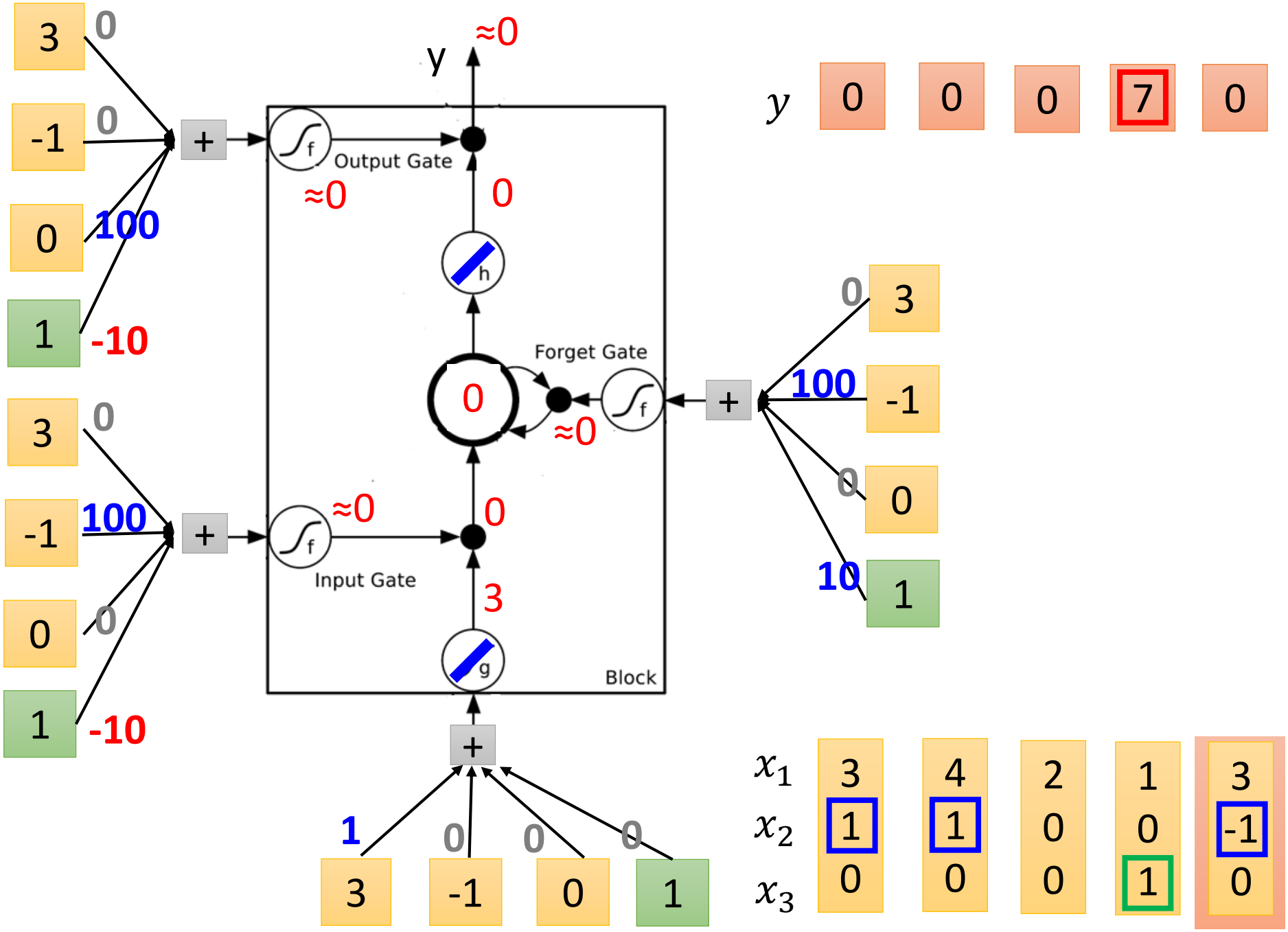

y
Output Gate
Forget Gate
Input Gate
Block
0
100
-10
Bias
1
10
x_1
x_2
x_3
y 0 0 0 7 0
x_1 3 4 2 1 3
x_2 1 1 0 0 -1
x_3 0 0 0 1 0


y
≈0
Output Gate
≈0
3
Forget Gate
3
≈1
Input Gate
≈1
3
3
Block
+
+
+
+
3
0
1
0
100
1
-10
3
0
1
100
0
0
1
-10
0
3
100
1
0
0
10
1
1
0
0
0
3
1
0
1
y
0
0
0
7
0
x_1
x_2
x_3
3
1
0
4
1
0
2
0
0
1
0
1
3
-1
0


y
≈0
Output Gate
≈0
7
Forget Gate
7
≈1
Input Gate
≈1
4
4
Block
Block
4
0
1
100
0
0
1
-10
4
0
1
100
0
0
1
-10
0
4
1
100
0
0
10
1
1
0
0
0
4
1
0
1
y
0
0
0
7
0
x_1
x_2
x_3
3
1
0
4
1
0
2
0
0
1
0
1
3
-1
0


y
≈0
Output Gate
≈0
7
Forget Gate
7
≈1
Input Gate
≈0
0
2
Block
+
f
h
g
2
0
0
1
0
0
0
100
-10
0
2
0
100
0
0
1
-10
0
2
0
0
100
0
0
10
1
1
0
0
0
2
0
0
1
y 0 0 0 7 0
x_1 3 4 2 1 3
x_2 1 1 0 0 -1
x_3 0 0 0 1 0

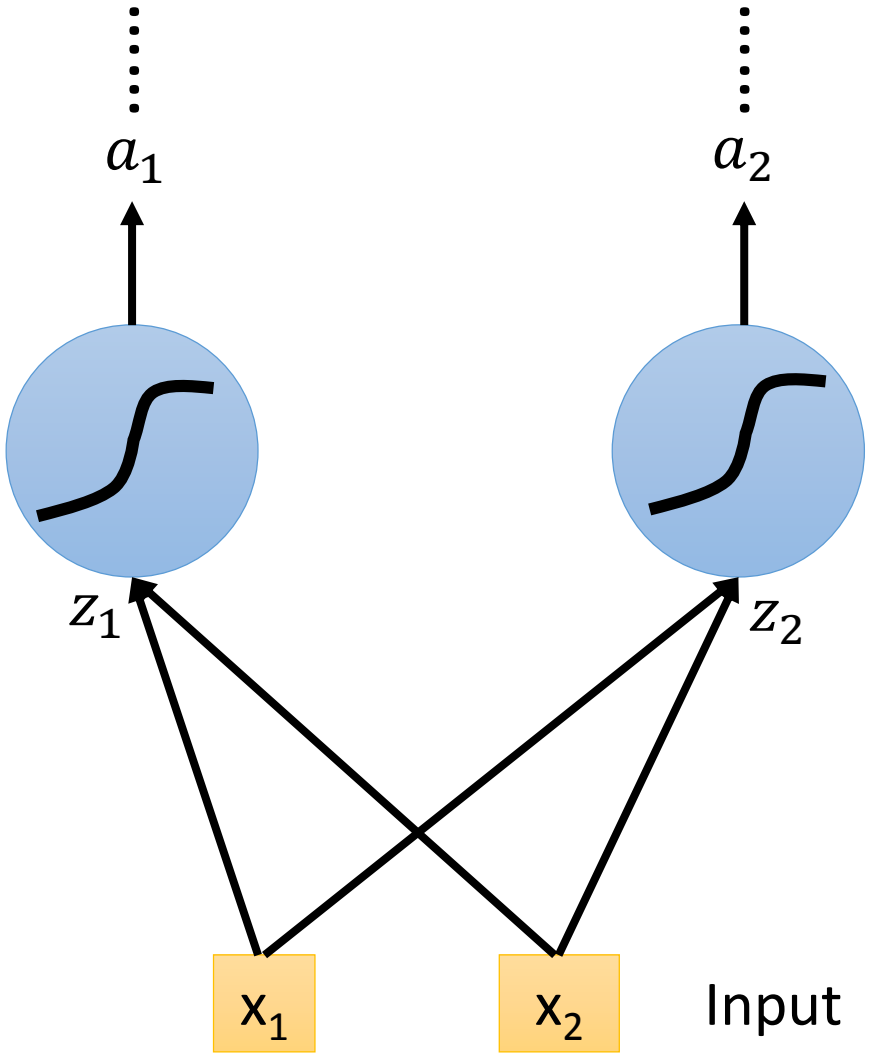

y
≈7
Output Gate
≈1
7
Forget Gate
≈1
Input Gate
≈0
0
1
Block
+
y 0 0 0 7 0
x1 3 4 2 1 3
x2 1 1 0 0 -1
x3 0 0 0 1 0
100
-10
10

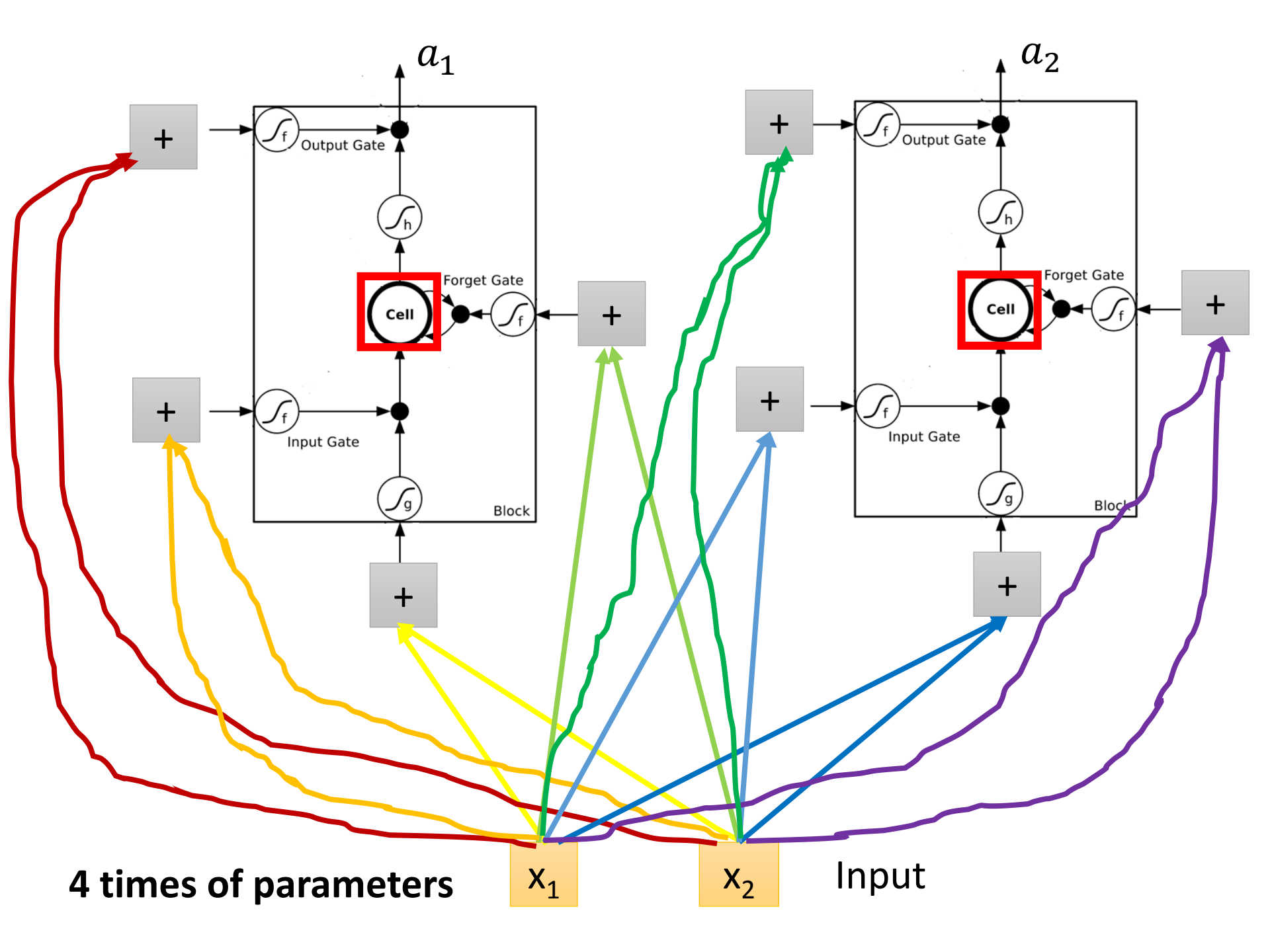

y
≈0
Output Gate
≈0
0
h
Forget Gate
0
≈0
Input Gate
≈0
0
3
Block
+
3
-1
0
1
0
0
0
100
-10
0
3
-1
0
1
100
-10
0
3
-1
0
1
100
10
1
0
0
0
y 0 0 0 7 0
x_1 3 4 2 1 3
x_2 1 1 0 0 -1
x_3 0 0 0 1 0

Original Network:

➢Simply replace the neurons with LSTM

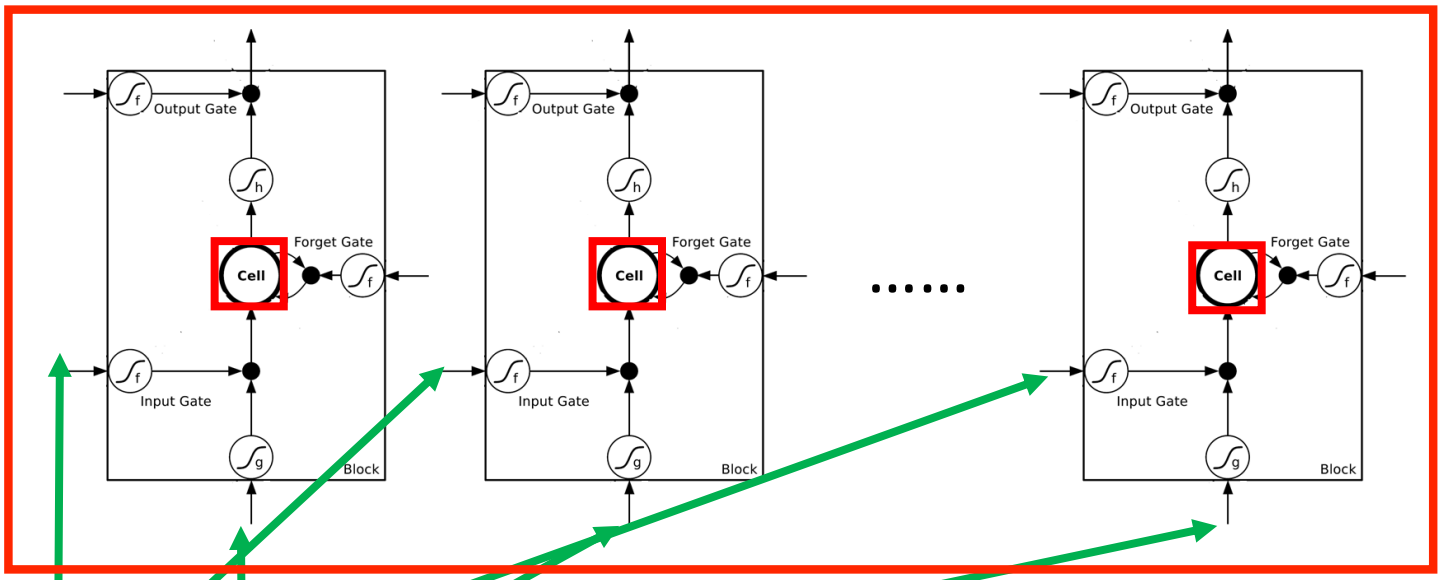
實際上，一個 LSTM Cell
就是原本的 Neural
Network 中的一個 Neuron




a_1
a_2
Output Gate
Forget Gate
Input Gate
Cell
Block
x_1
x_2
Input
4 times of parameters

LSTM

這是一個 Hidden Layer，裡面有很多 Neuron，每一個 Neuron 都是一個 LSTM Cell



c^{t-1}

vector

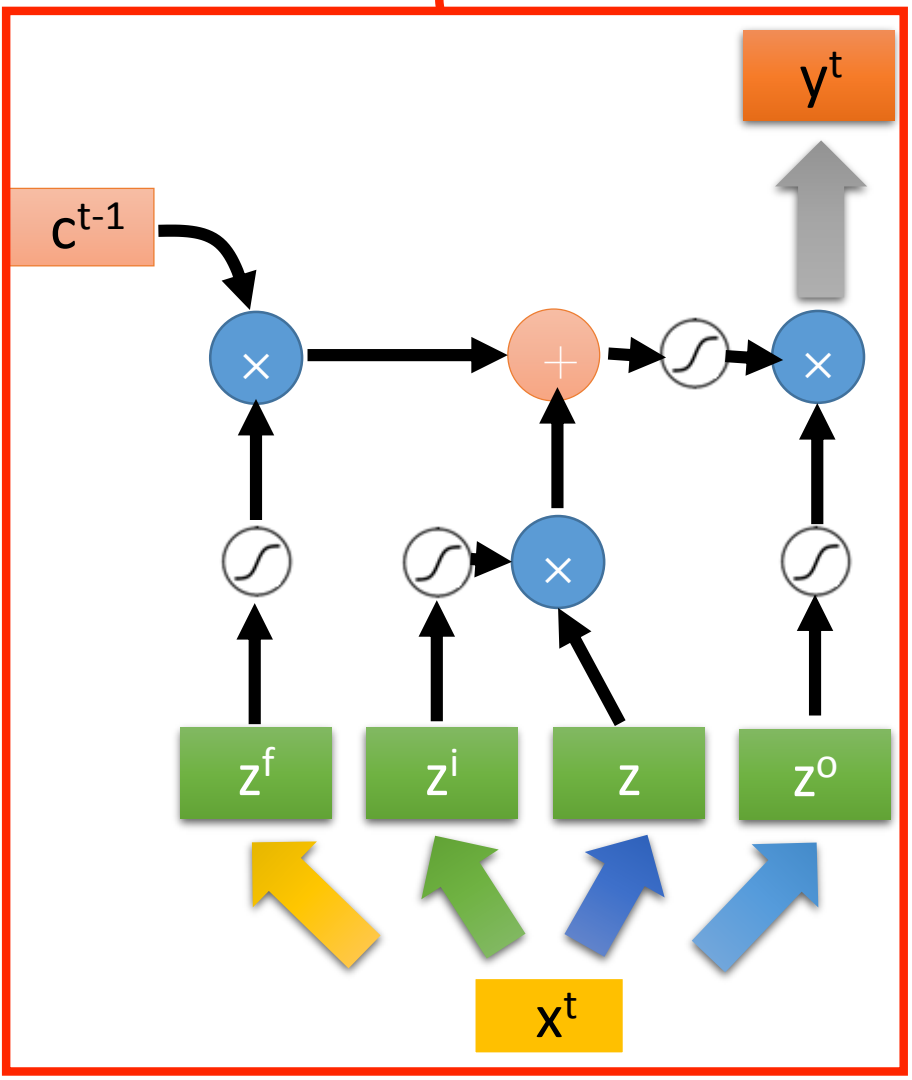
z^f z^i z z^o

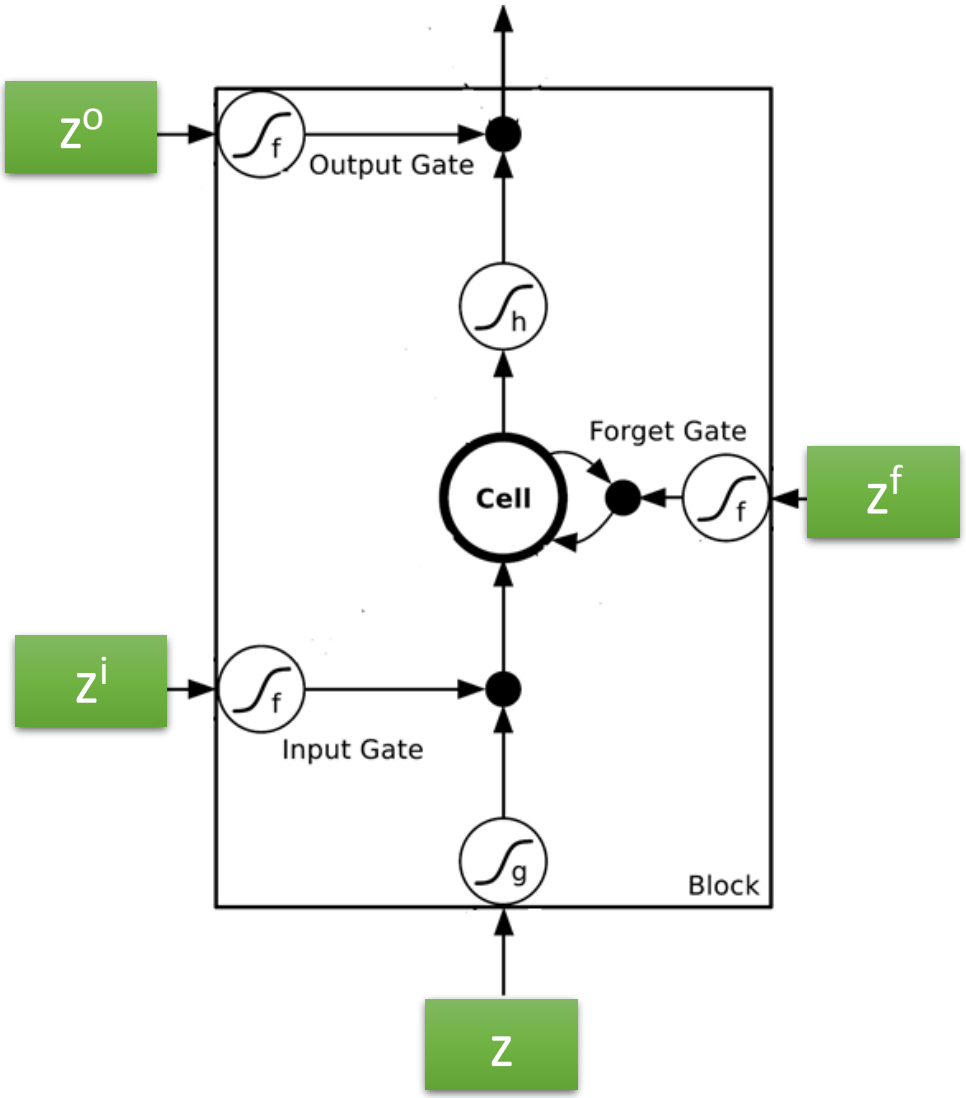
4 vectors

x^t

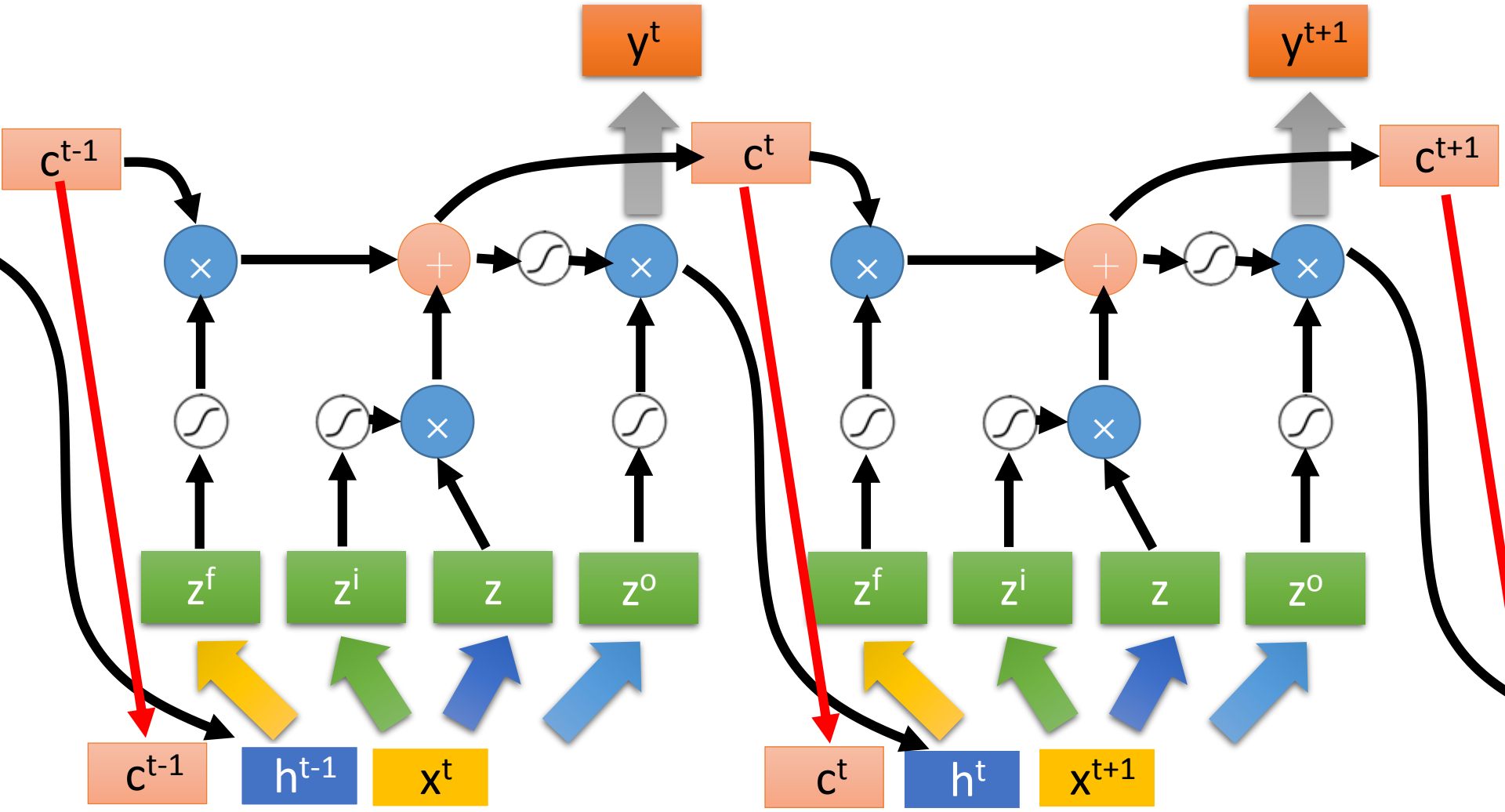
LSTM

實際上，在同一層的所有 LSTM Cell 是可以一起運算的





LSTM


Extension: “peephole”
y^t
y^{t+1}
c^{t-1}
c^t
c^{t+1}
z^f
z^i
z
z^o
c^{t-1}
h^{t-1}
x^t
c^t
h^t
x^{t+1}