

TWSIAM / NCCU PyDay Deep Learning

手寫工作坊

蔡炎龍 政治大學應用數學系

政大應數 「數學軟體應用」課程

<http://bit.ly/nccujupyter>

政大應數 「數學軟體應用」課程

Facebook 公開社團

[https://www.facebook.com/groups/
159902691120219/](https://www.facebook.com/groups/159902691120219/)

1

復習

“Artificial intelligence
is all about math.”

<https://code.facebook.com/pages/1902086376686983>

— 揚 · 勒丘恩

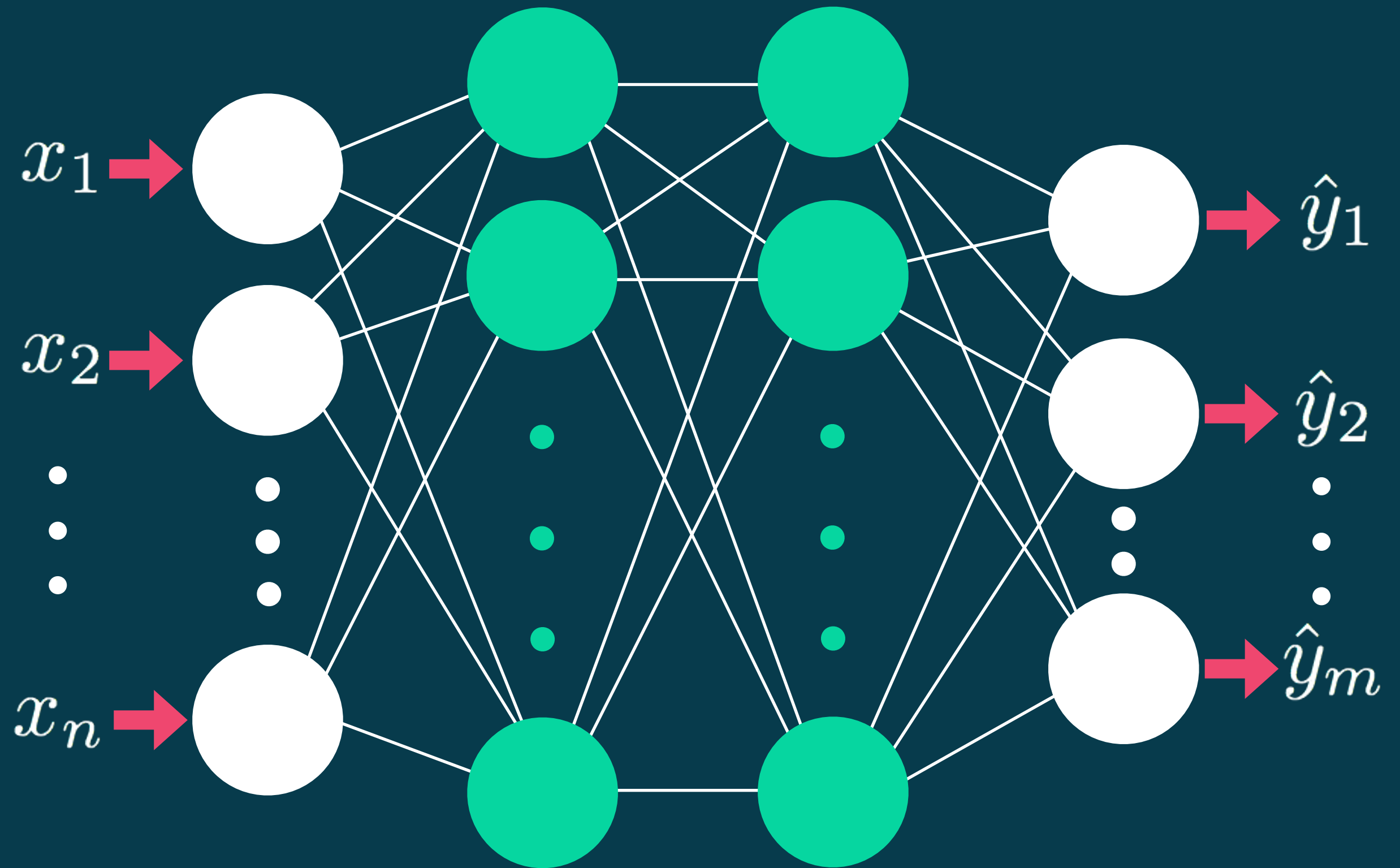
NYU 教授, Facebook AI Director

Deep Learning 不過就是學個函數。

$$F: X \rightarrow Y$$

m維

n維



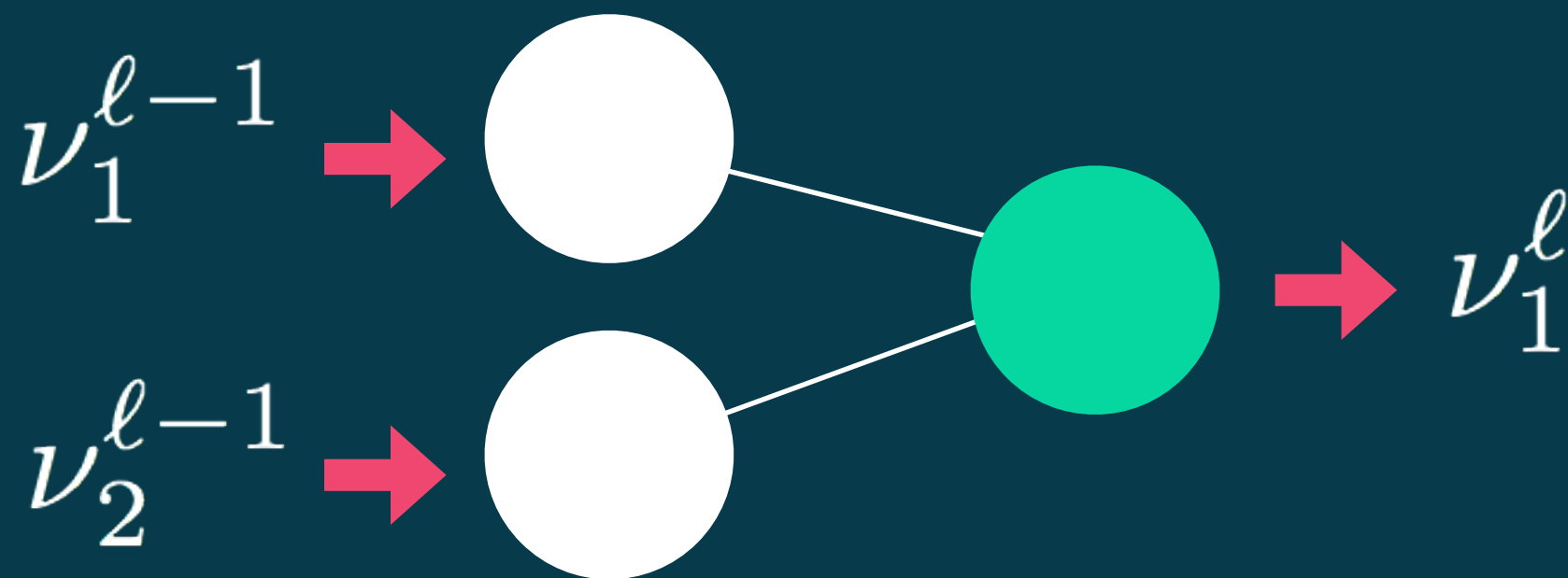
**Input
Layer**

**Hidden
Layer**

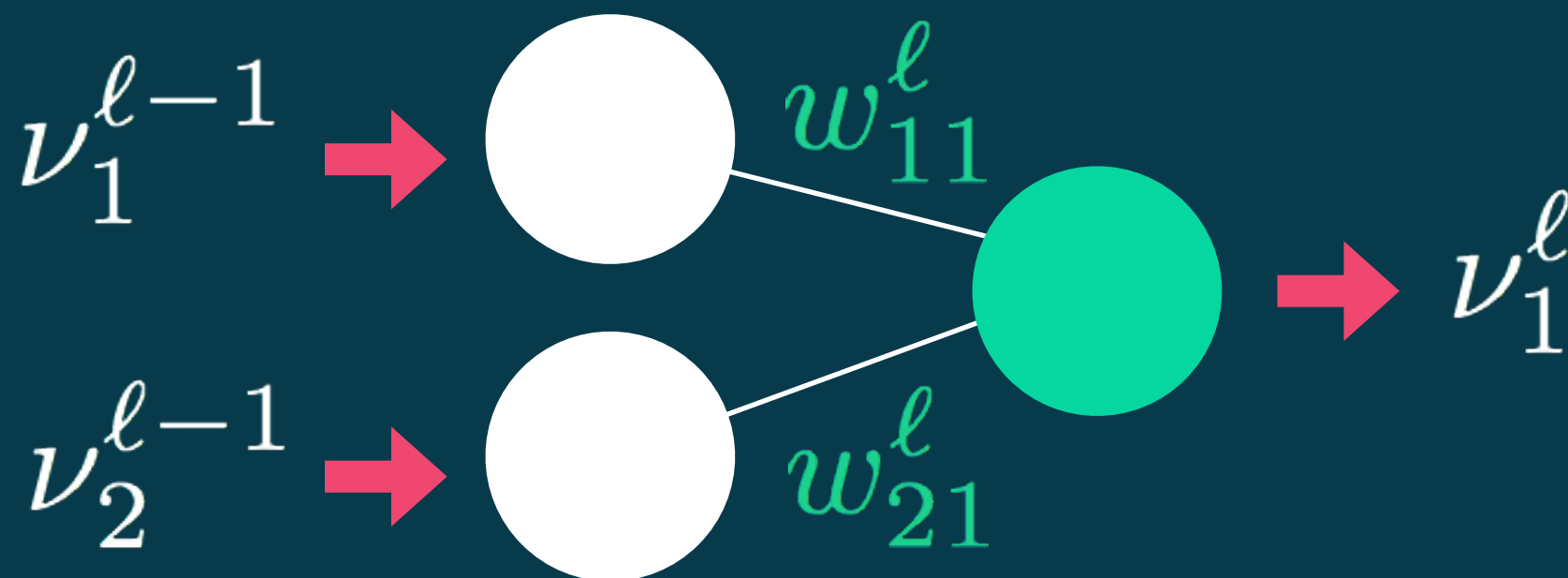
**Output
Layer**

每個神經元動作基本上
是一樣的!

反正就是接受輸入經某種運算輸出

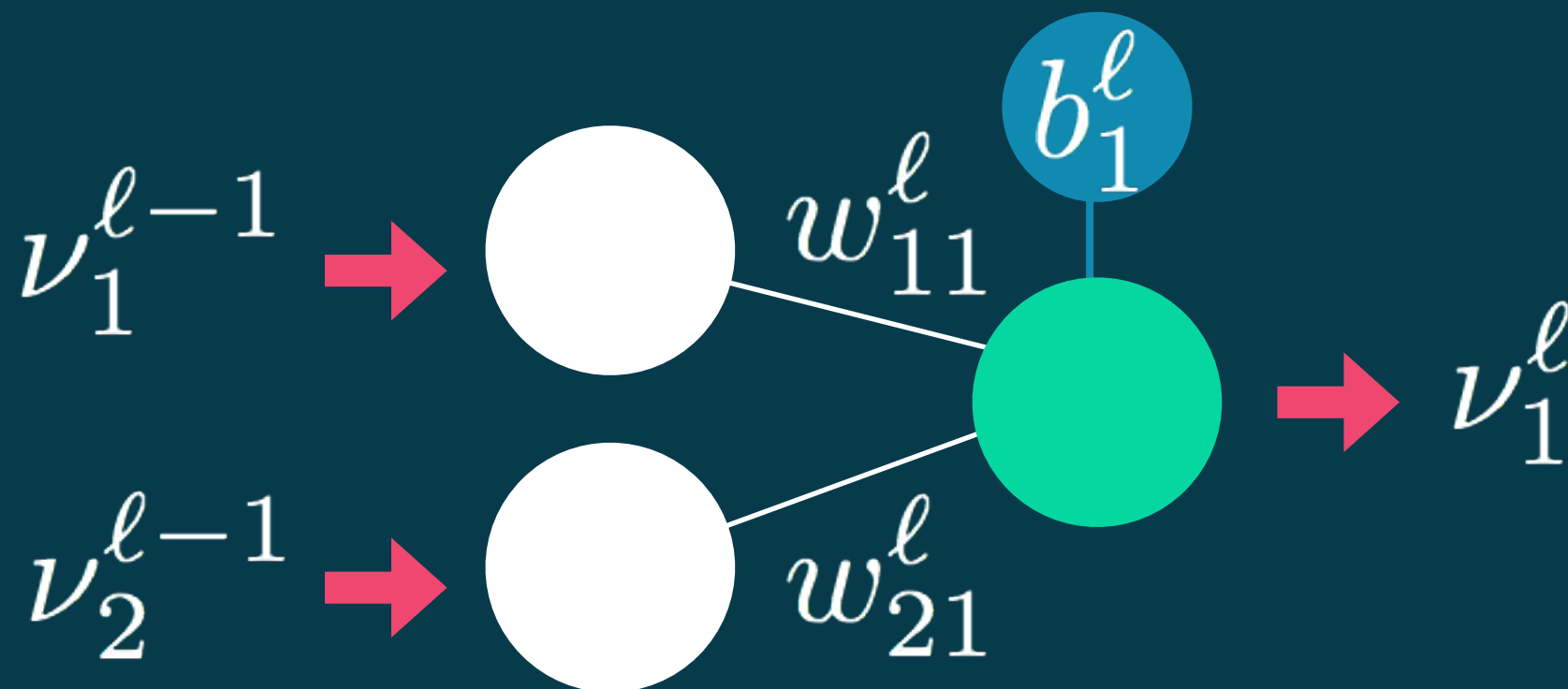


1 經過 weight 加權



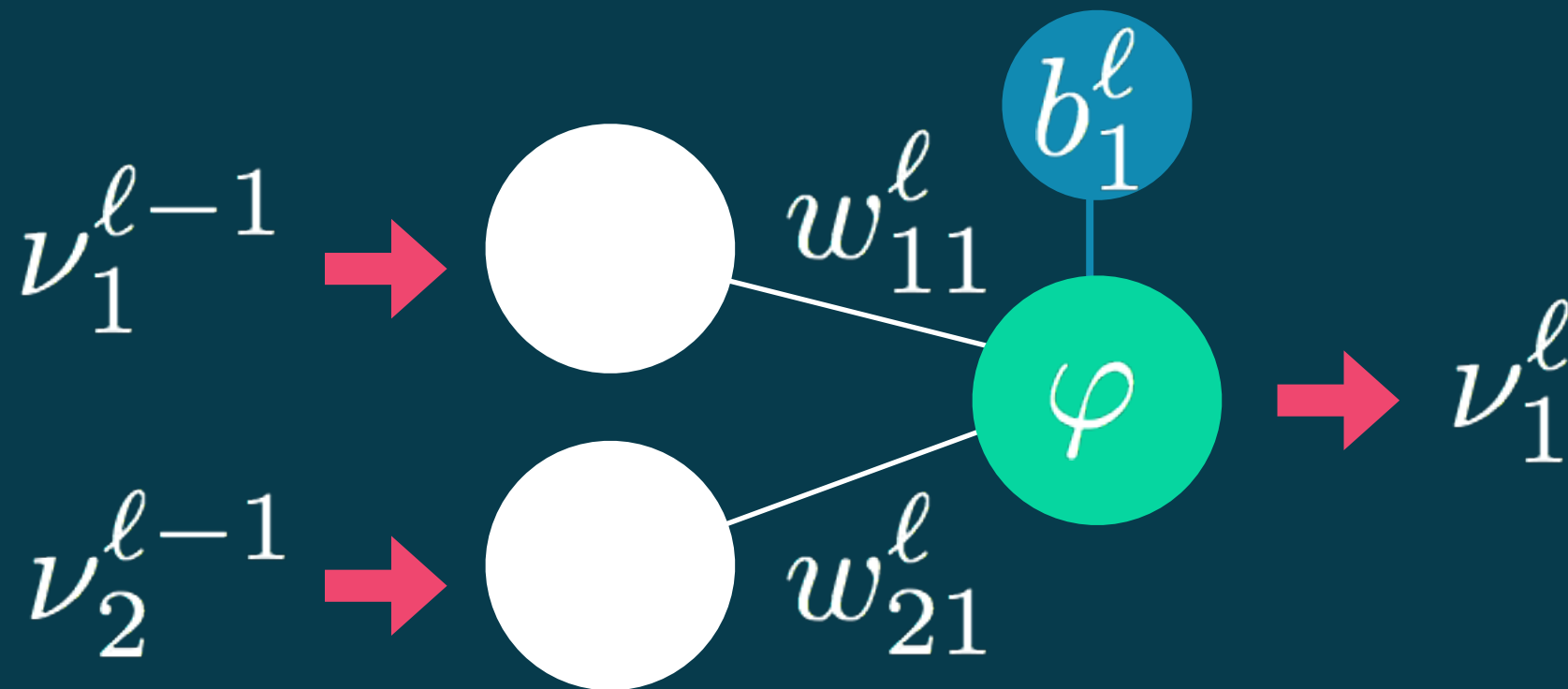
$$w_{11}^{\ell} \nu_1^{\ell-1} + w_{21}^{\ell} \nu_2^{\ell-1}$$

2 每個神經元有個 bias



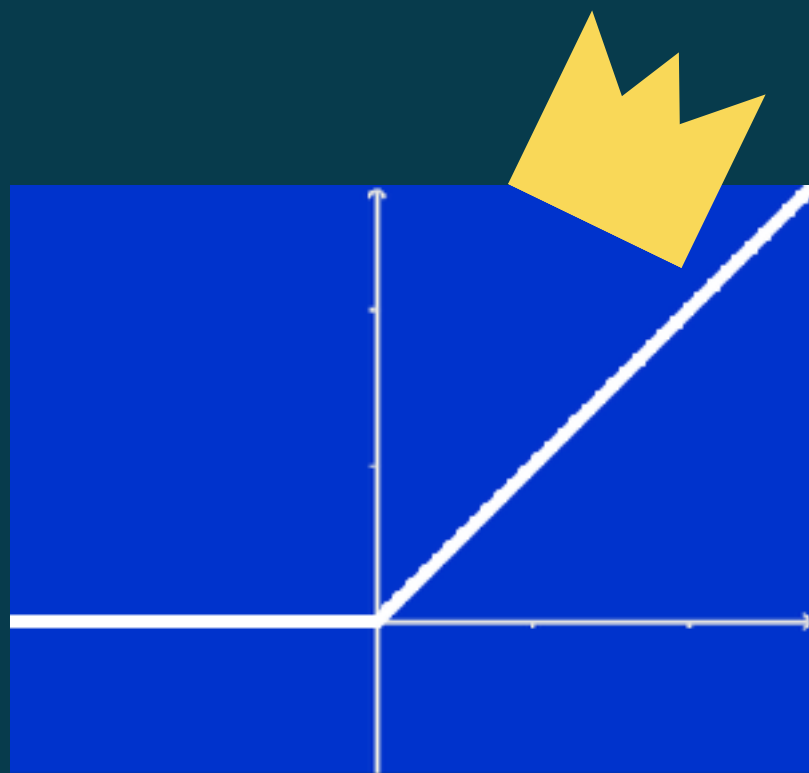
$$w_{11}^{\ell} \nu_1^{\ell-1} + w_{21}^{\ell} \nu_2^{\ell-1} + b_1^{\ell}$$

3 最後代入「激發函數」 (activation function)

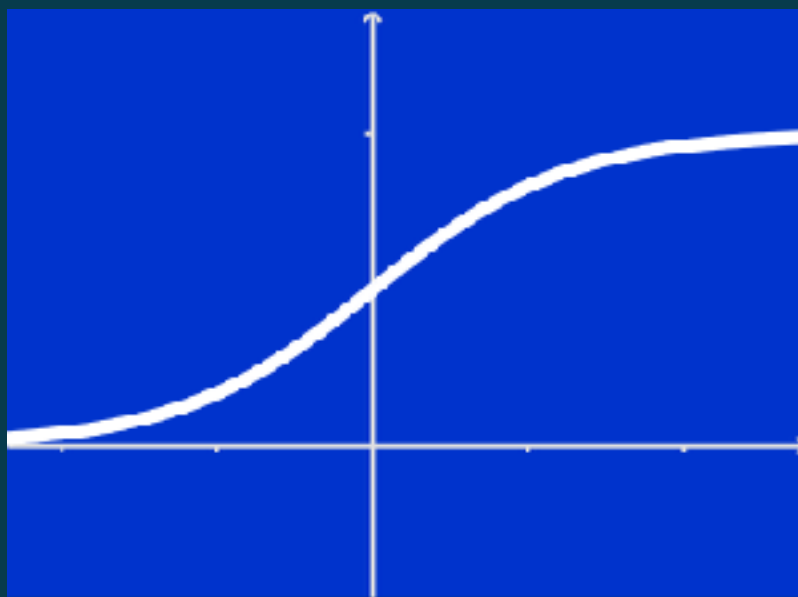


$$\varphi(w_{11}^{\ell} \nu_1^{\ell-1} + w_{21}^{\ell} \nu_2^{\ell-1} + b_1^{\ell}) = \nu_1^{\ell}$$

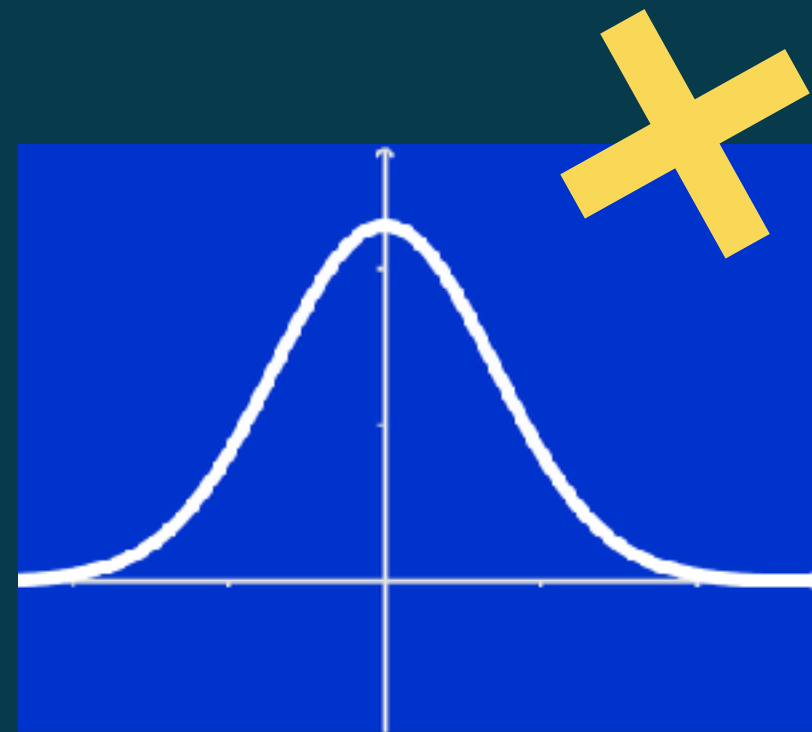
幾個 activation functions



ReLU

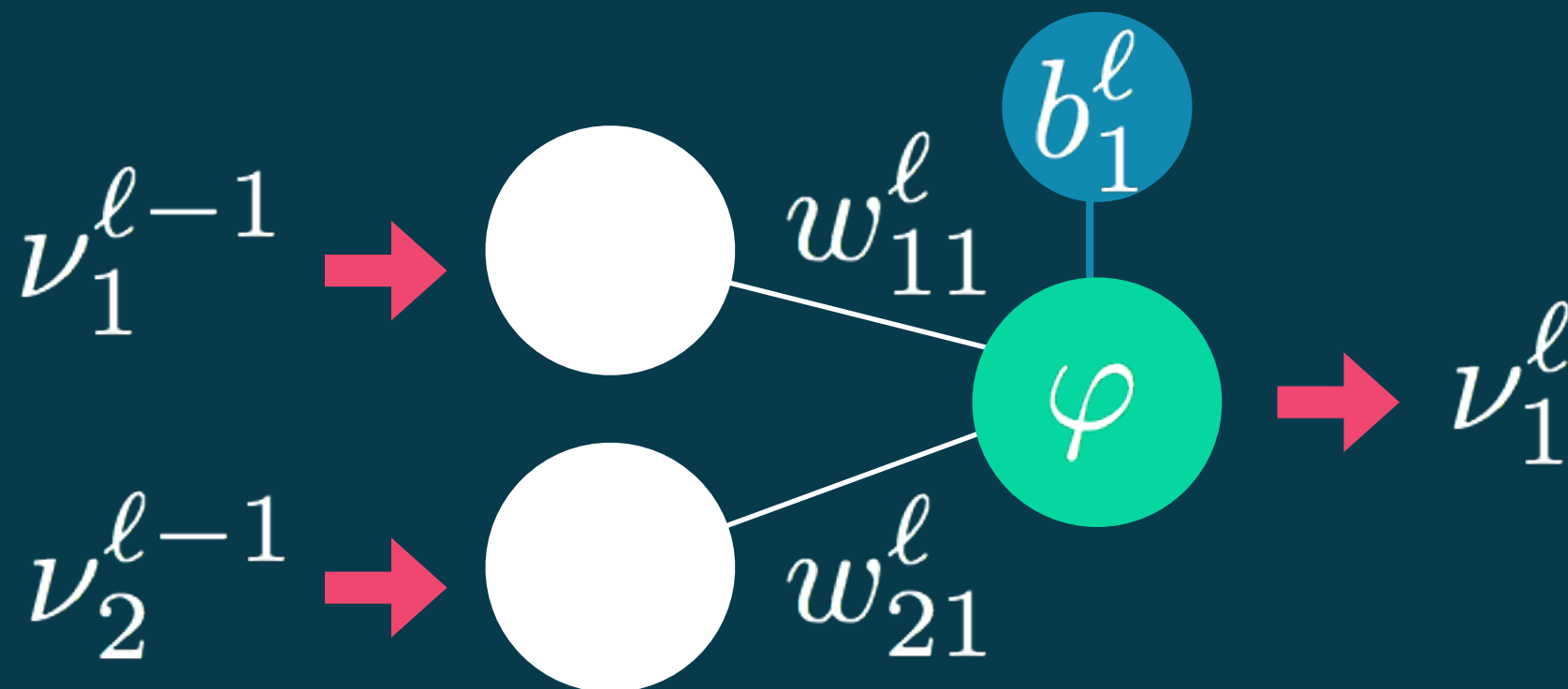


Sigmoid



Gaussian

變數就是 weights, biases



$$\varphi(w_{11}^{\ell} \nu_1^{\ell-1} + w_{21}^{\ell} \nu_2^{\ell-1} + b_1^{\ell}) = \nu_1^{\ell}$$

固定結構神經網路的函數空間

當一個神經網路結構決定、activation functions 也決定，那可以調的就是 **weights, biases**。我們把這些參數的集合叫 θ ，每一個 θ 就定義一個函數，我們把它看成一個集合。

$$\{F_{\theta}\}$$

我們就是要找 θ^*

使得 F_{θ^*} 和目標函數最接近

「最近」是什麼意思

就是 “loss function” 最小

假設我們有訓練資料

$$\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_k, \mathbf{y}_k)\}$$

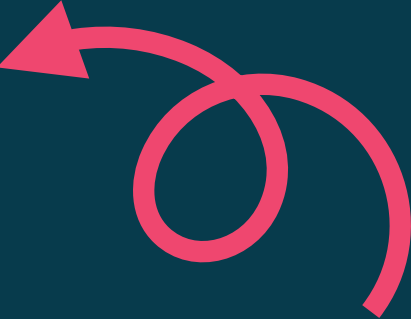
最常用 loss function

$$L(\theta) = \frac{1}{2} \sum_{i=1}^k \| \mathbf{y}_i - F_{\theta}(\mathbf{x}_i) \|^2$$

我們希望它越小越好

基本上這樣調

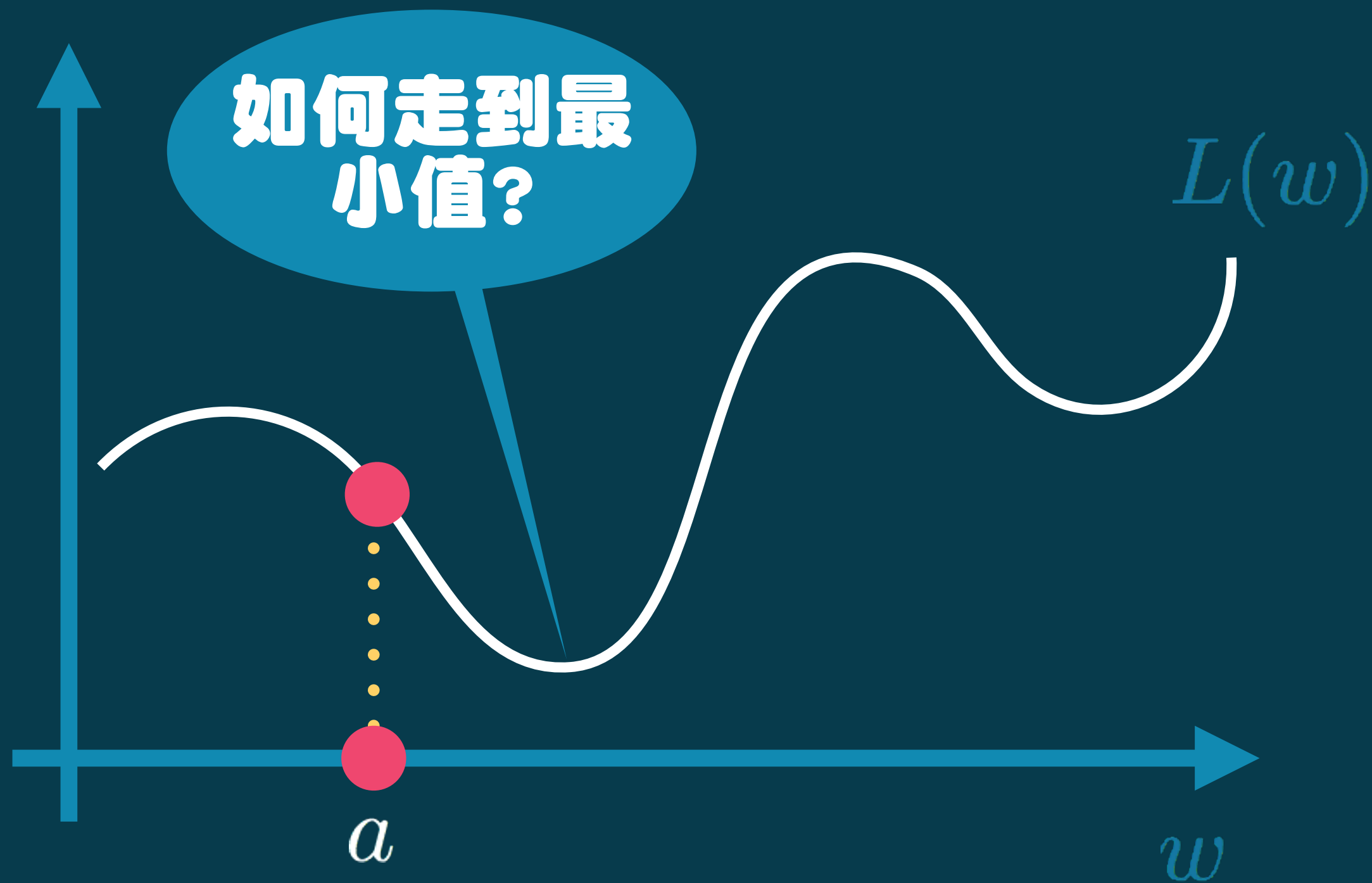
learning rate


$$-\eta \frac{\partial L}{\partial w_{ij}}$$

記得 **L** 是 w_1, w_2, b_1, \dots
的函數

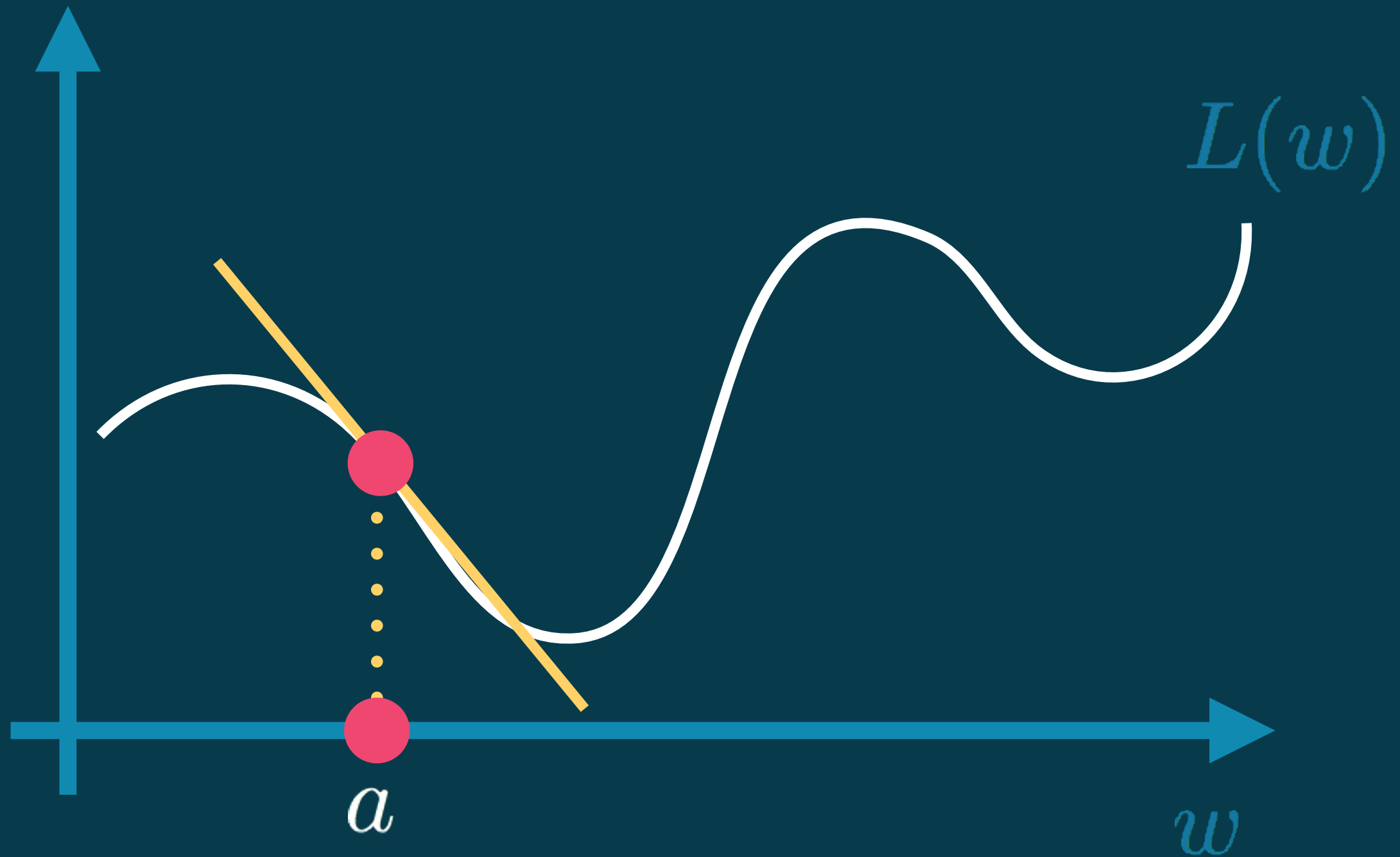
希望越小越好

為了簡化, 我們先把 L
想成只有一個變數 w

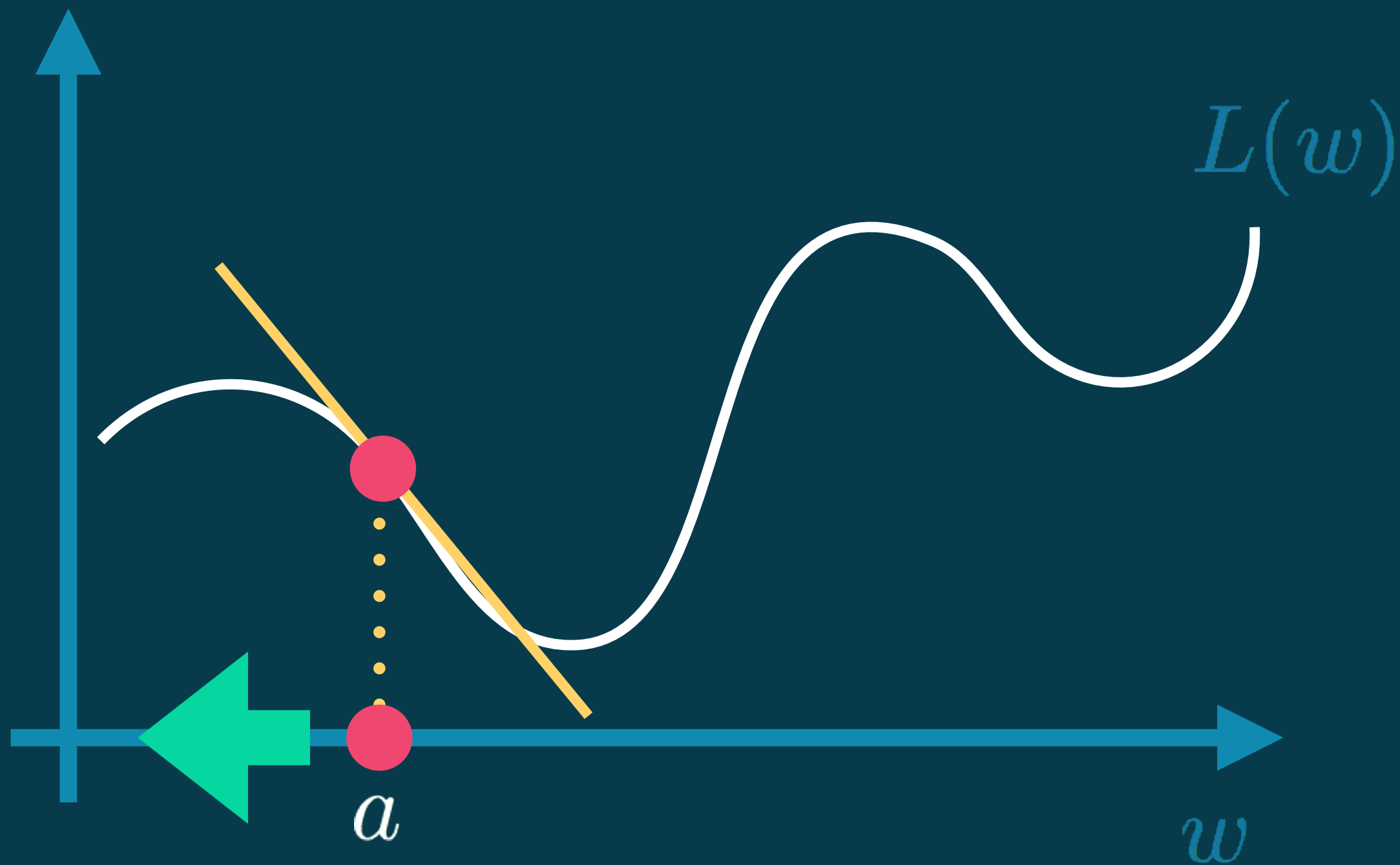


w 目前的值

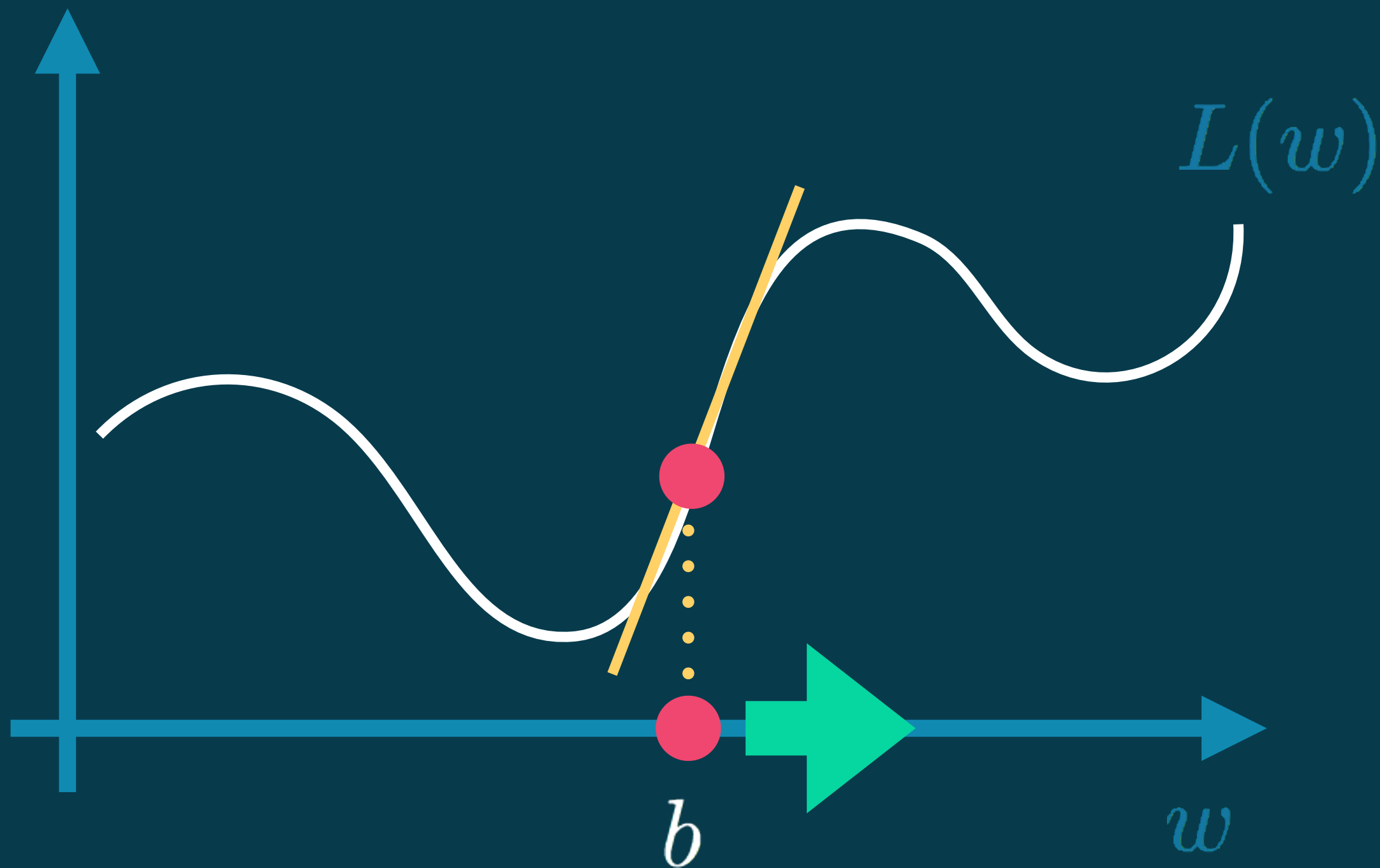
切線是關鍵!



切線斜率 < 0

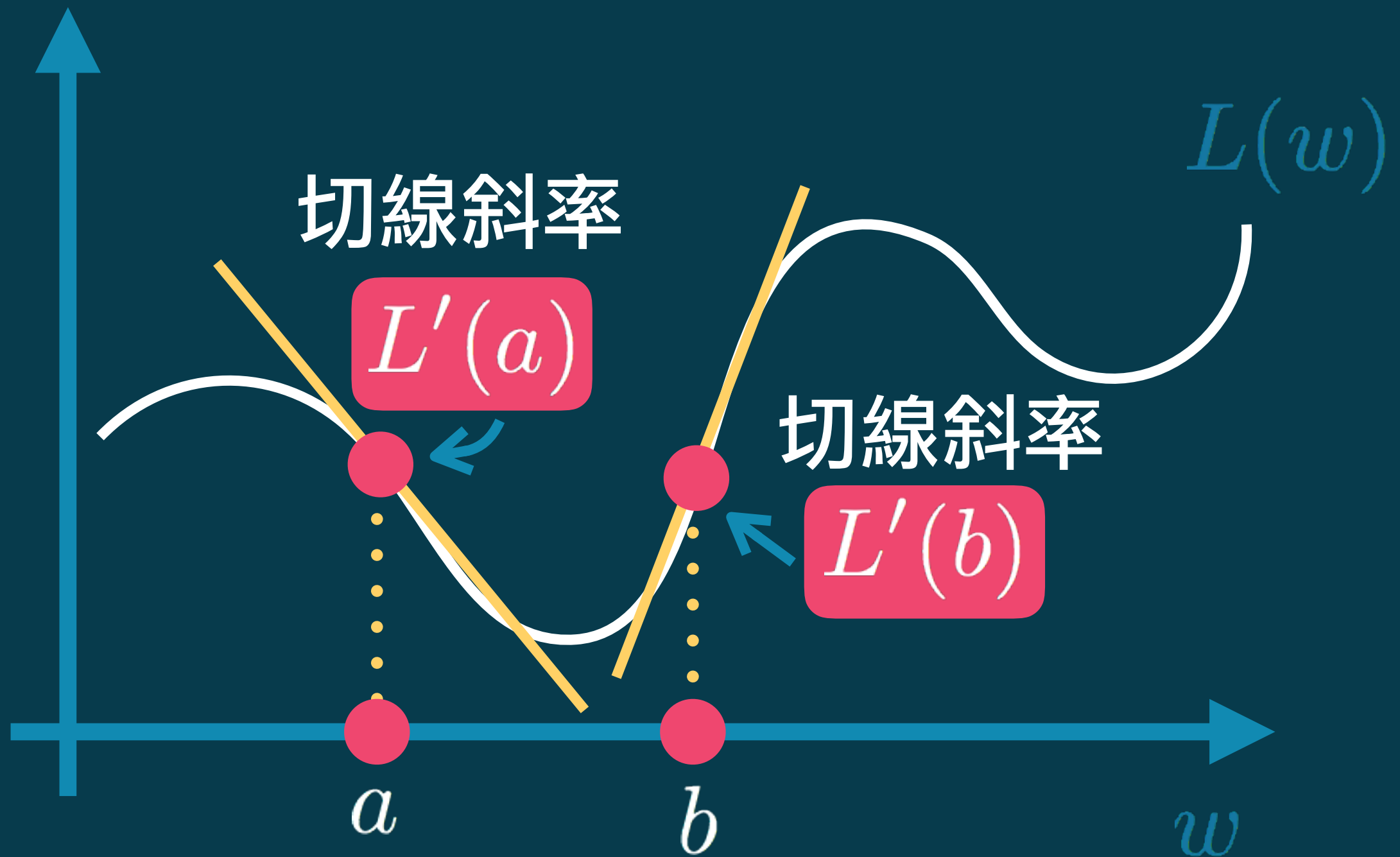


切線斜率 > 0

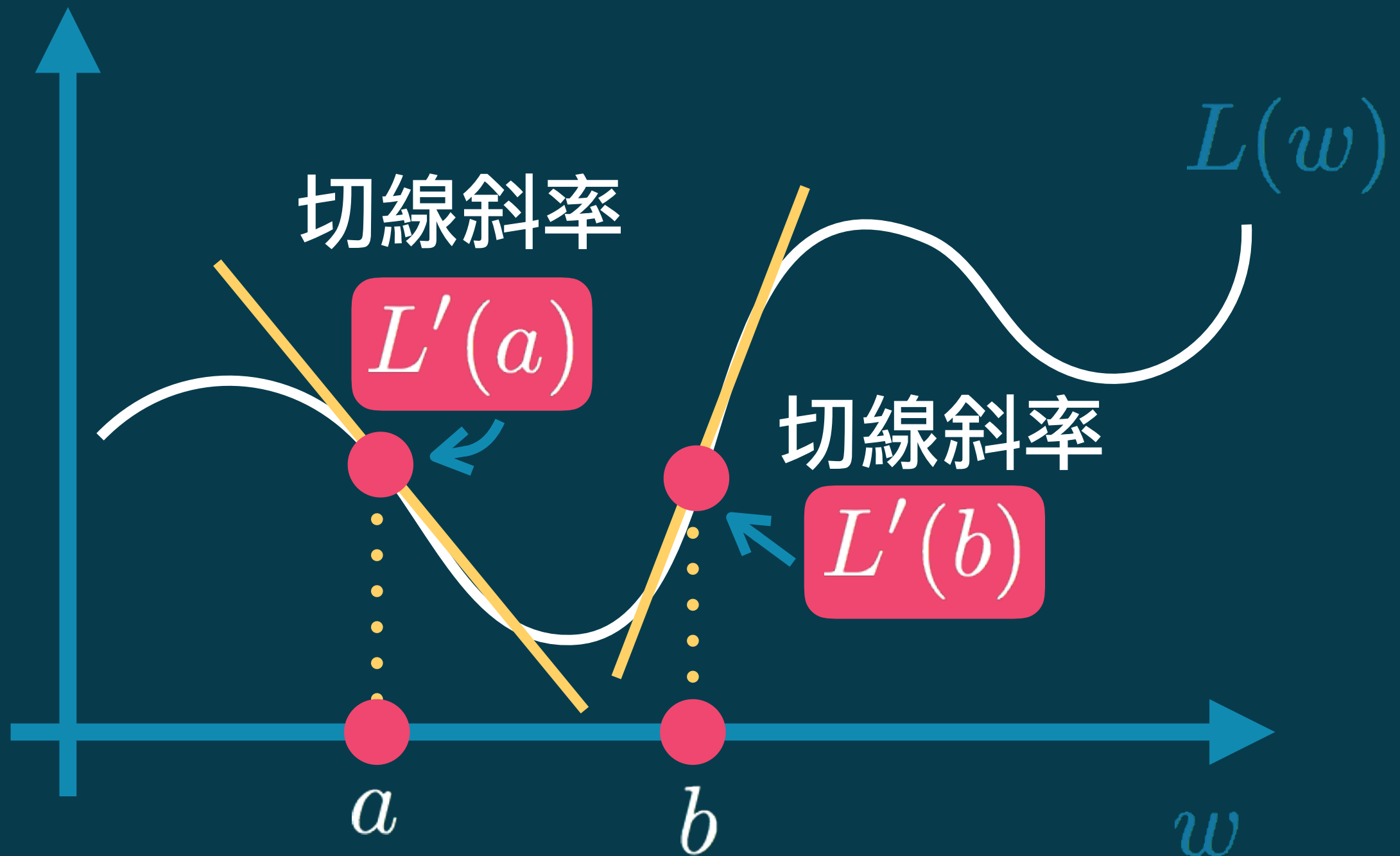


切線斜率指向 (局部)
最大值的方向!!

符號



切線斜率是變化率



符號

梯度 (gradient)

Gradient 當然要有很酷的符號:

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

符號

梯度 (gradient)

我們調整 w_1, w_2, b_1 的方法就變成:

$$\begin{bmatrix} w_1 \\ w_2 \\ b_1 \end{bmatrix} - \eta \nabla L$$

這「學習法」有個很潮的名字

Gradient Descent

梯度下降



Jupyter Notebook

Python 程式語言

數據分析主流程式語言

Python 程式語言

YouTube, Google 第一版都是用 Python

現在很好裝



<https://www.continuum.io/downloads>

請選用 Python 3 版本

為了深度學習, Windows 可能要 Python 3.5

Python 基本套件



Anaconda 包全部

+



TensorFlow

或

Theano

Keras

安裝

重點

Jupyter Notebook

Jupyter Notebook 是很多數據分析師選用的工作環境。

是一個網頁形式 Python 前端開發環境。
之前叫 IPython Notebook, 但後來有更多語言可以使用這個界面, 所以改名叫 Jupyter Notebook。

重點

Jupyter Notebook

Python
Jupyter
Julia R

唸起來又像木星, 紀念伽利略對木星衛星觀測、
紀錄開放分享的精神。

例子

圖形辨識

$$f(\text{28x28 image of 6}) = 6$$

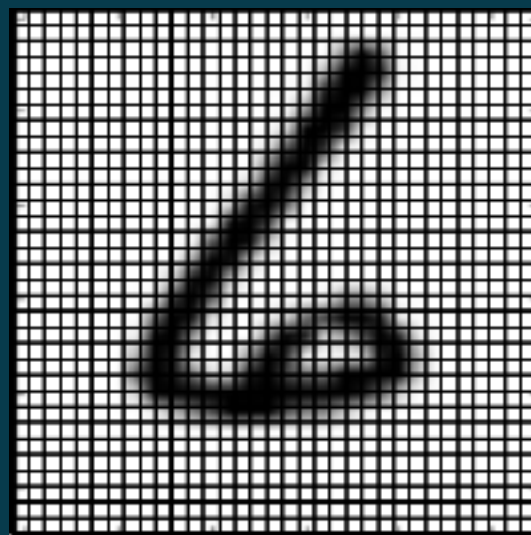
$$f: \mathbb{R}^{784} \rightarrow \mathbb{R}$$

例子

圖形辨識

這種方式會有個缺點, 比如說我們訓練出的函數

$F($



$) = 6.35$

那意思是這個數字有點像 6, 有點像 7?

例子

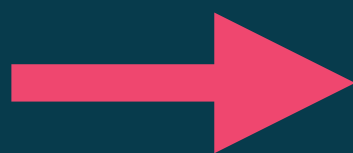
圖形辨識

像這基本上是分類的問題，值域的數字沒有連續概念。我們可以用接下來的方式處理「答案」。

例子

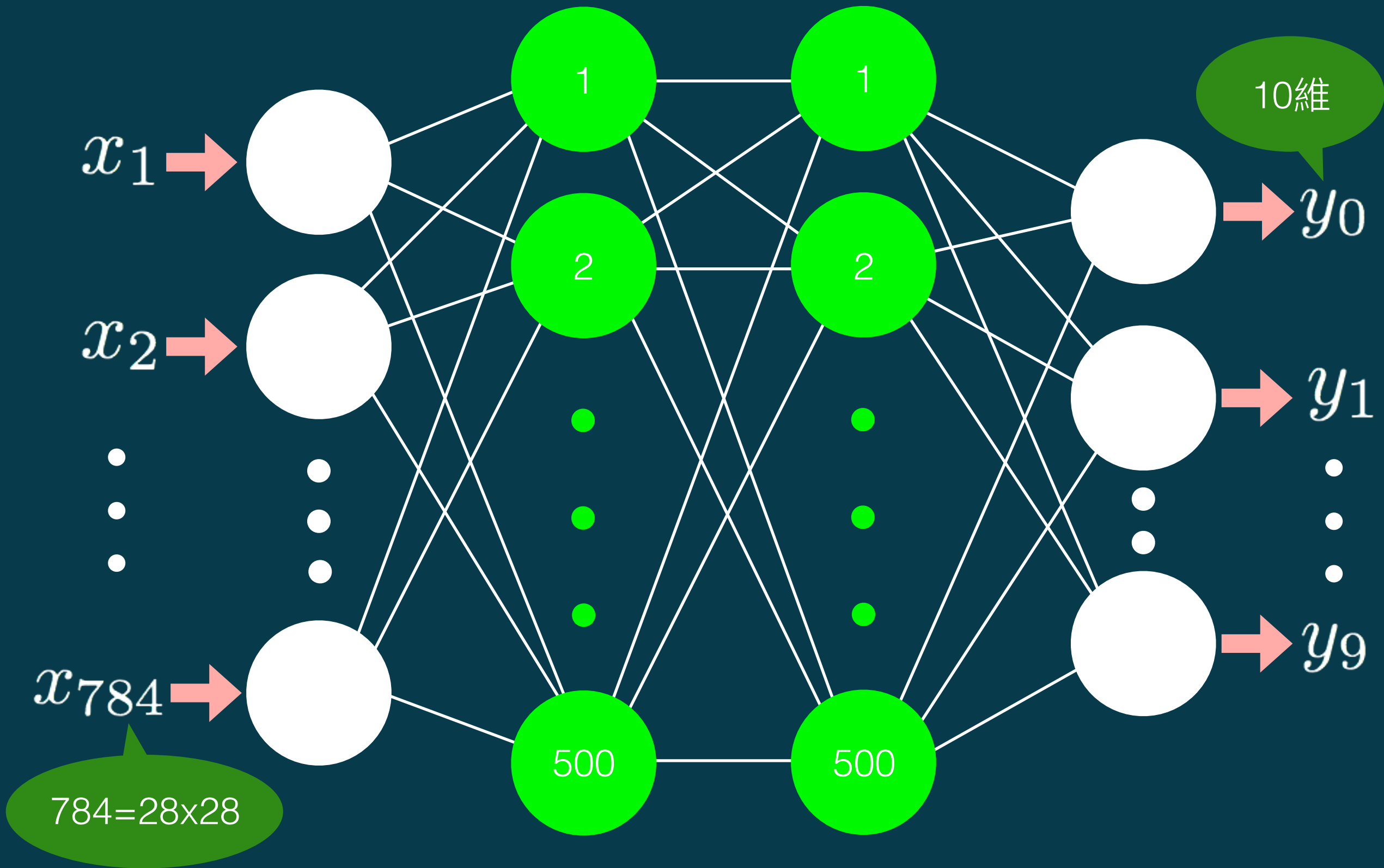
圖形辨識

6



“One Hot”

0	0
0	1
0	2
0	3
0	4
0	5
1	6
0	7
0	8
0	9



Input Layer

Hidden Layer

Output Layer

開始

和很多機器學習一樣, 弄個機器出來, 「正常」的 feedforward 神經網路就用 Sequential。

```
model = Sequential()
```

輸入層+隱藏層1

是不是很简单, 基本上只要指出有幾個節點, 選哪個 activation function 就好。

```
model.add(Dense(500, input_dim=784))  
model.add(Activation('sigmoid'))
```

隱藏層2

輸入就是 500 個不用說, 直接說要輸出多少個, 也就是多少個節點。

```
model.add(Dense(output_dim=500))  
model.add(Activation('sigmoid'))
```


輸出層

10 個輸出, 分別代表辨識為 0, 1, 2, ..., 9 的機率。

```
model.add(Dense(output_dim=10))  
model.add(Activation('softmax'))
```

然後就組裝

說一下要用什麼當 loss function, 還有學習法。

```
model.compile(loss='mse', optimizer=SGD(lr=0.1),  
metrics=[ 'accuracy' ])
```

基本上就好了

Jupyter 示範



CNN

Convolutional Neural Networks



eder @edersantana · 1天



When you gotta hustle selling neural nets in China. Already profitable AI startup!



1990 年代, 雖然大家對
神經網路寄以厚望...

但結果像圖形辨識就被
SVM 打趴了!

差不多到 2000 年,「神經
網路」變成沒有人要談的
禁忌話題...

ImageNet 2012 圖形辨識大賽冠軍

由 AlexNet 拿下, 之後大家幾乎都用某種形式的 CNN

於是神經網路又把 SVM
打趴...

Deep Learning

2015 年, 「CNN 之父」 LeCun 和另外兩位
Deep Learning 巨擘* 在 Nature 發表以 “Deep
Learning” 為題的論文

*Yann LeCun, Yoshua Bengio & Geoffrey Hinton

1 convolutional layer

2 max-pooling layer



**CNN 通常有這兩種
hidden layers**



Convolutional Layer

我們要做 filters

例如 3x3 的大小

內積

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

filter

這學來的

2	5	5	2	5	2	0	1
2	3	4	0	4	2	1	5
4	3	1	3	5	5	4	3
5	3	4	5	0	2	1	5
2	3	1	1	1	0	1	3
4	4	1	1	5	1	1	4
2	3	2	2	0	4	2	4
0	5	4	5	3	4	1	4

35					

想成這是一張圖所成的矩陣

filter

還是一樣的矩陣

右移一格

35 | 27

filter

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

2	5	5	2	5	2	0	1
2	3	4	0	4	2	1	5
4	3	1	3	5	5	4	3
5	3	4	5	0	2	1	5
2	3	1	1	1	0	1	3
4	4	1	1	5	1	1	4
2	3	2	2	0	4	2	4
0	5	4	5	3	4	1	4

35	27	44	32	36	38
36	36	37	36	36	43
37	37	23	26	17	35
29	25	22	18	14	27
27	25	24	21	24	32
31	38	27	34	25	40

一路到最後

神經元是這樣看的

圖片上的點是一個
個輸入層神經元

2	5	5	2	5	2	0	1
2	3	4	0	4	2	1	5
4	3	1	3	5	5	4	3
5	3	4	5	0	2	1	5
2	3	1	1	1	0	1	3
4	4	1	1	5	1	1	4
2	3	2	2	0	4	2	4
0	5	4	5	3	4	1	4

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

filter

35	27	44	32	36	38
36	36	37	36	36	43
37	37	23	26	17	35
29	25	22	18	14	27
27	25	24	21	24	32
31	38	27	34	25	40

filter

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

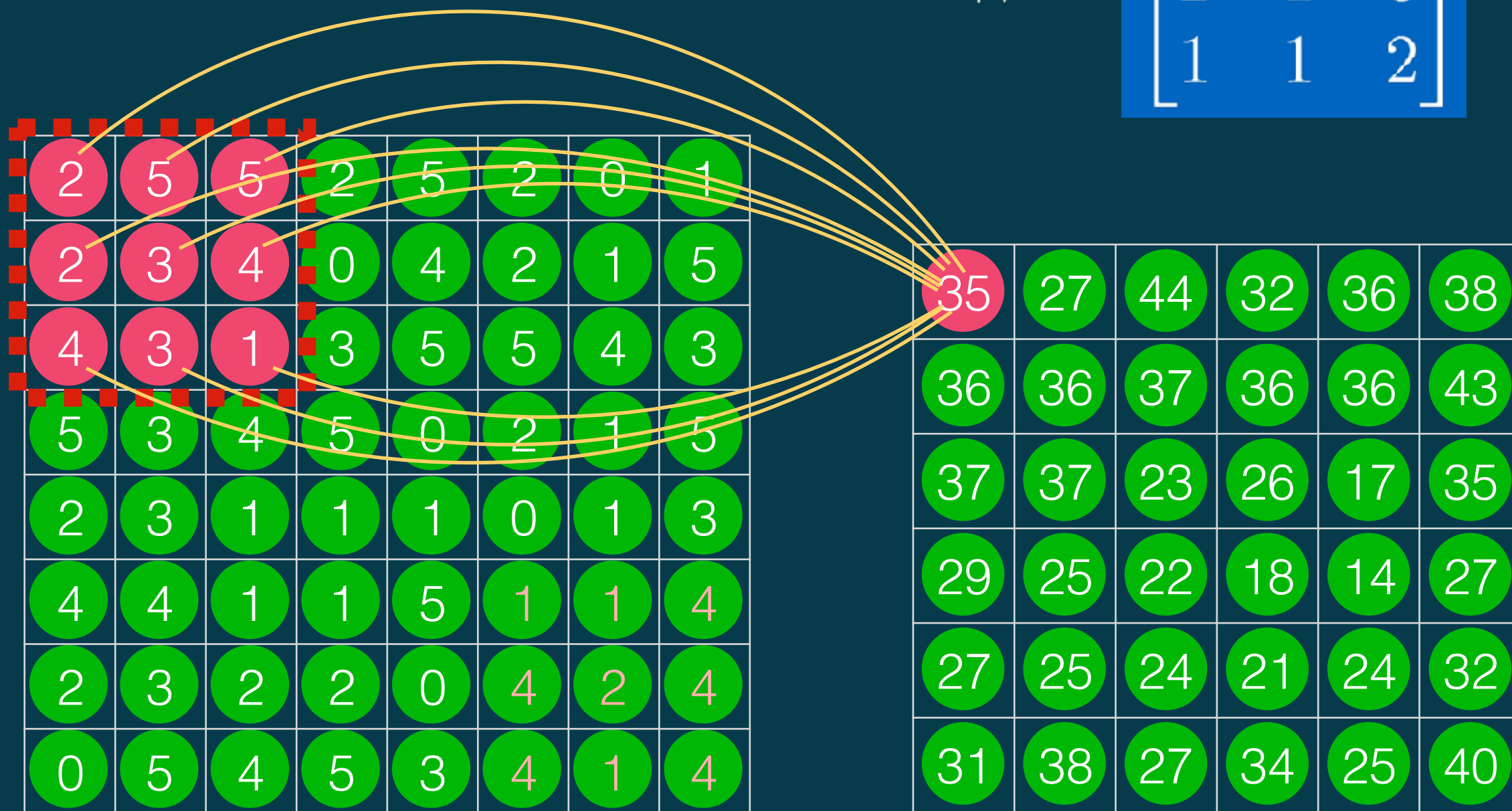
2	5	5	2	5	2	0	1
2	3	4	0	4	2	1	5
4	3	1	3	5	5	4	3
5	3	4	5	0	2	1	5
2	3	1	1	1	0	1	3
4	4	1	1	5	1	1	4
2	3	2	2	0	4	2	4
0	5	4	5	3	4	1	4

35	27	44	32	36	38
36	36	37	36	36	43
37	37	23	26	17	35
29	25	22	18	14	27
27	25	24	21	24	32
31	38	27	34	25	40

Conv 層也是一個個
神經元

filter

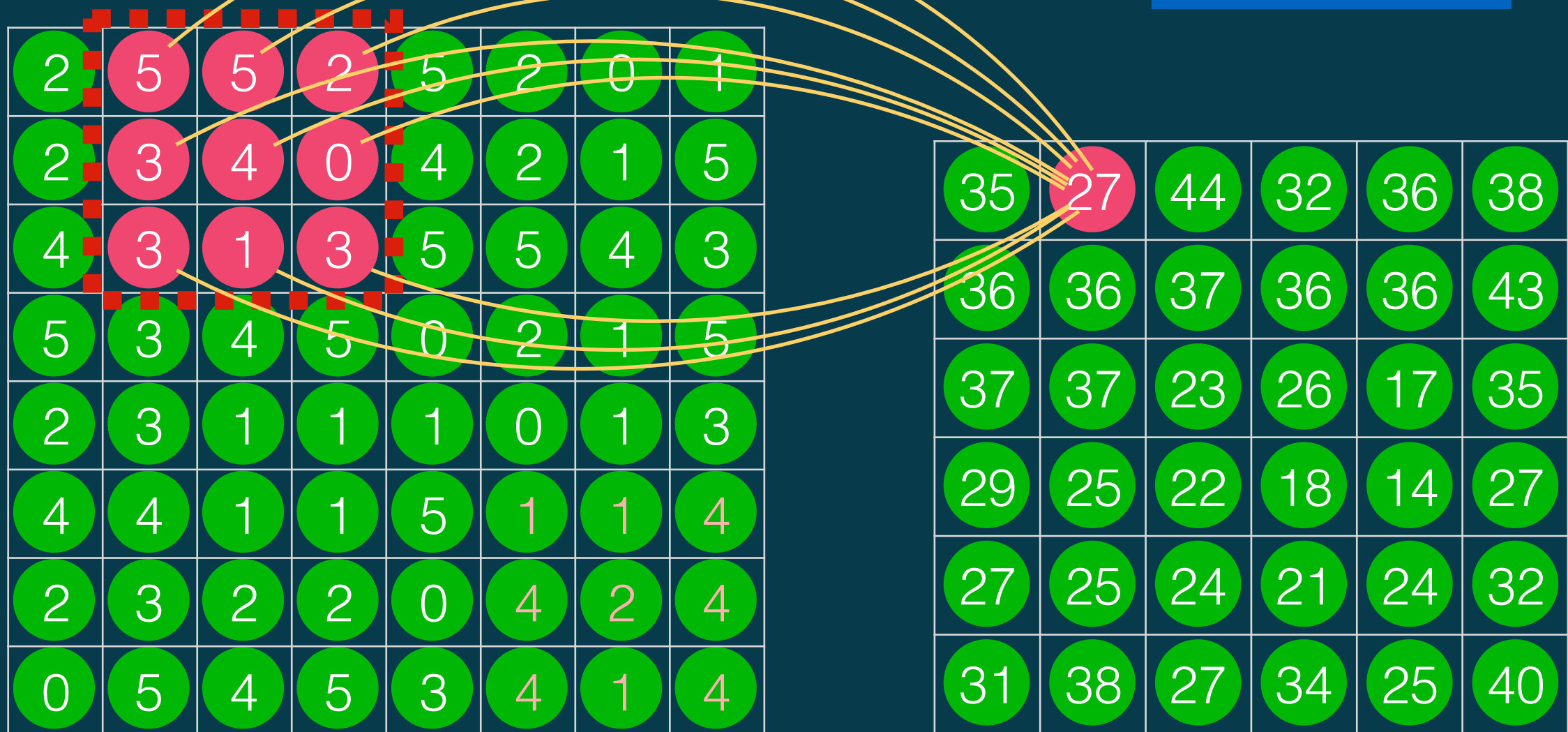
$$W = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$



兩層中沒有完全相連

filter

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$



再來 share 同樣的 weights

- 最後就是一個 6x6 的矩陣
- 有時我們會把它弄成還是 8x8
- 基本上和原矩陣一樣大
- 而且我們通常 filter 會很多!

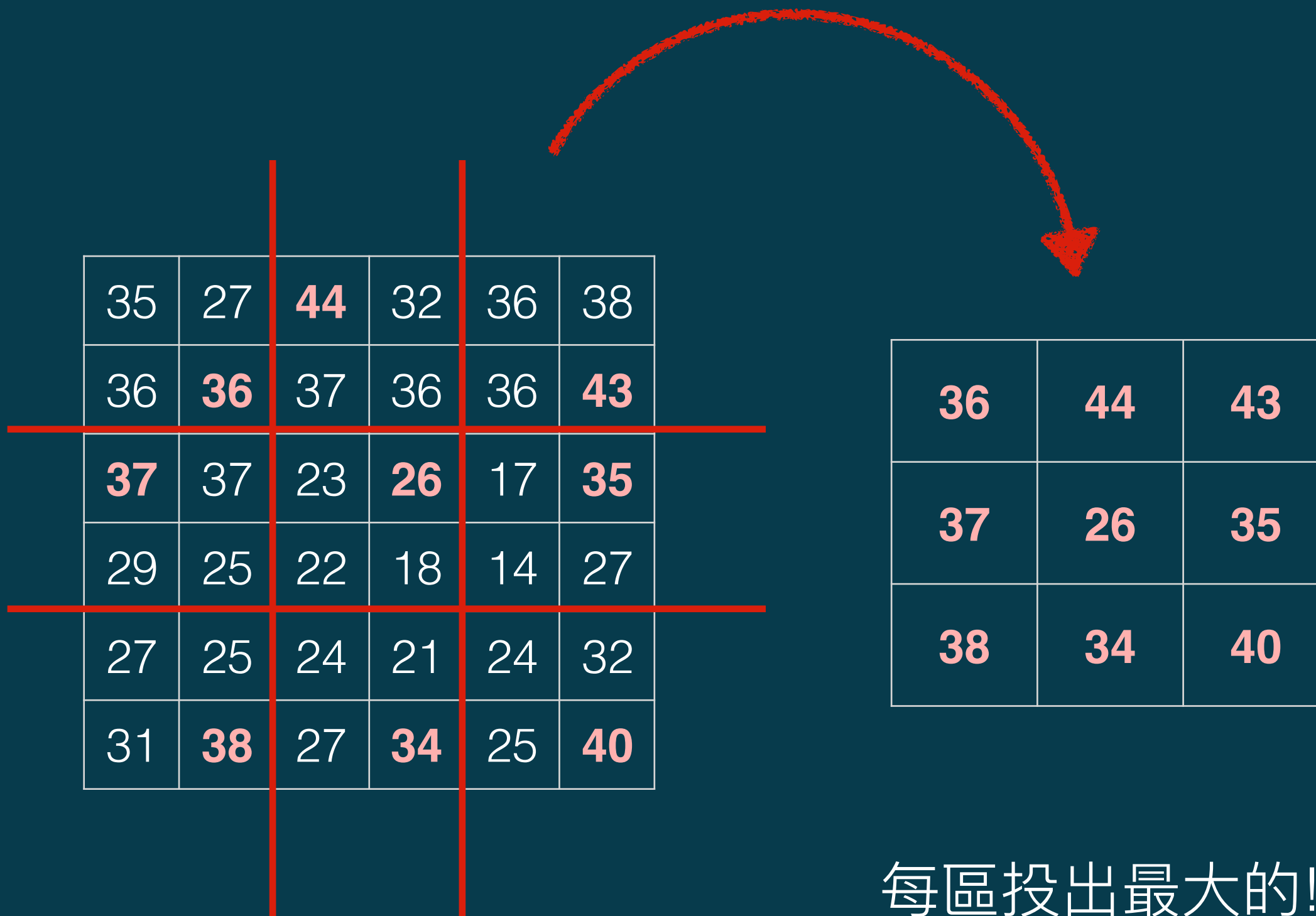
35	27	44	32	36	38
36	36	37	36	36	43
37	37	23	26	17	35
29	25	22	18	14	27
27	25	24	21	24	32
31	38	27	34	25	40



Max-Pooling Layer

基本上就是「投票」

看我們要多大區選一個代表, 例如 2x2



每區投出最大的!!

可以不斷重覆

convolution, max-pooling, convolution,
max-pooling...

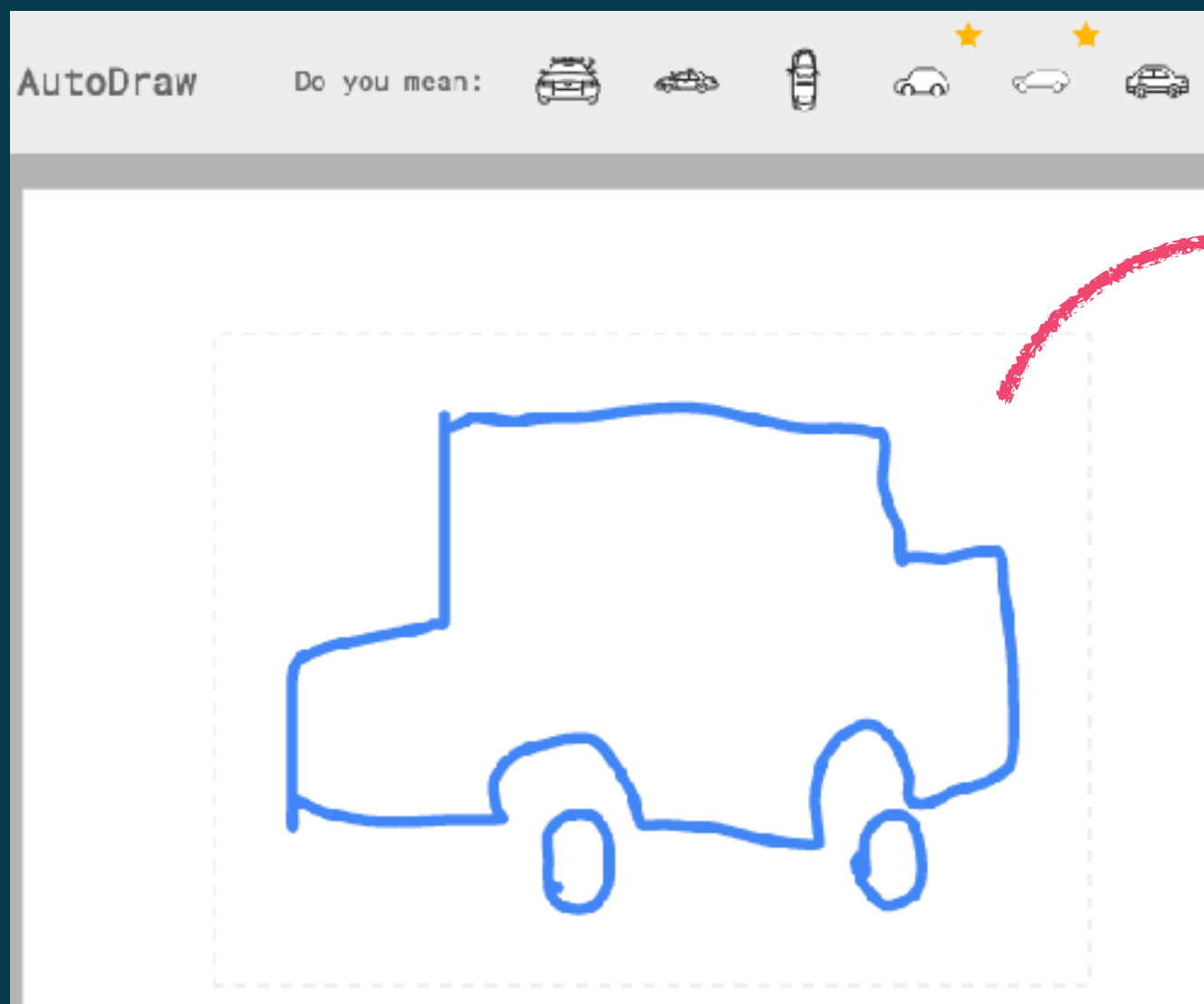
做完再送到「正常的」
神經網路

應用

圖形辨識

Google AutoDraw

<https://www.autodraw.com/>



應用

畫風移轉

A Neural Algorithm of Artistic Style

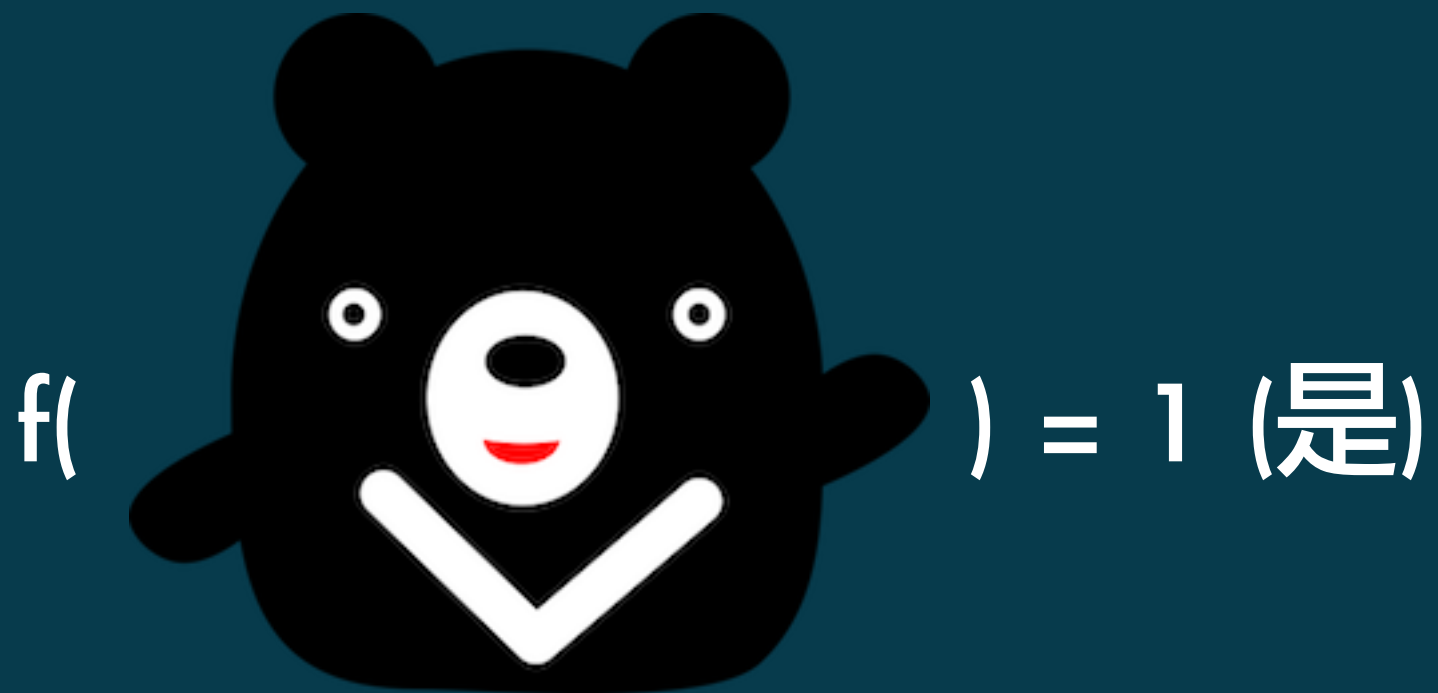
arXiv:1508.06576



手寫 CNN

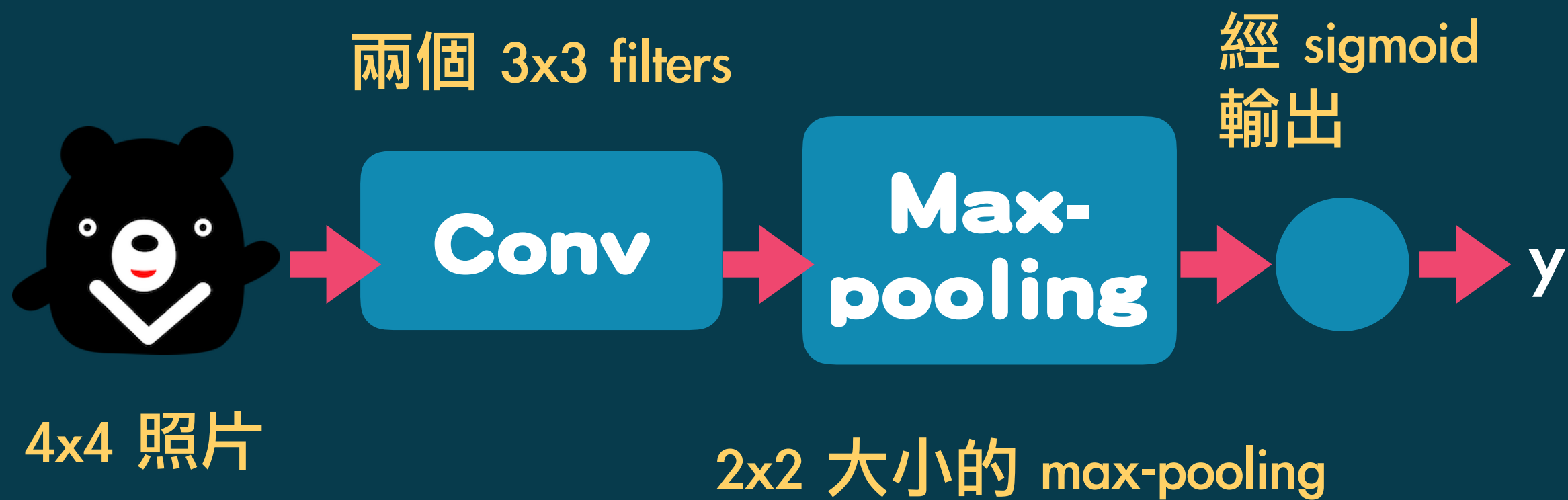
問題

一張 4x4 的灰階照片，判斷是不是台灣黑熊。



當然是假的, 4x4 比較像這樣。

架構



輸入

$$\begin{bmatrix} 2 & 1 & 3 & 4 \\ 0 & 2 & 5 & 1 \\ 6 & 1 & 3 & 7 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

輸入

$$\begin{bmatrix} 2 & 1 & 3 & 4 \\ 0 & 2 & 5 & 1 \\ 6 & 1 & 3 & 7 \\ 1 & 1 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 3 & 4 & 0 \\ 0 & 0 & 2 & 5 & 1 & 0 \\ 0 & 6 & 1 & 3 & 7 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

外面用 0 包一層, 變成 6x6 矩陣

Conv

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 3 & 4 & 0 \\ 0 & 0 & 2 & 5 & 1 & 0 \\ 0 & 6 & 1 & 3 & 7 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

filter 1

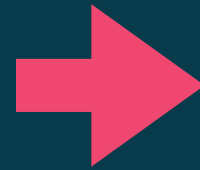
$$\begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 2 \\ 0 & -1 & 1 \\ 2 & 2 & 1 \end{bmatrix}$$

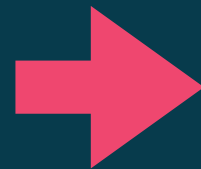
filter 2

經過 convolution 之後...

filter 1


$$\begin{bmatrix} 5 & 11 & 19 & 8 \\ 16 & 23 & 23 & 16 \\ 12 & 7 & 15 & 10 \\ 3 & 10 & 8 & 2 \end{bmatrix}$$

filter 2


$$\begin{bmatrix} 1 & 11 & 16 & 8 \\ 19 & 29 & 23 & 26 \\ 2 & 19 & 17 & 1 \\ 8 & 13 & 17 & 10 \end{bmatrix}$$

filter 1

$$\begin{bmatrix} 5 & 11 & 19 & 8 \\ 16 & 23 & 23 & 16 \\ 12 & 7 & 15 & 10 \\ 3 & 10 & 8 & 2 \end{bmatrix}$$

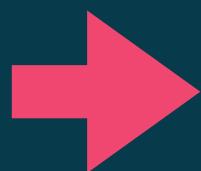

2x2 的 max-
pooling

filter 2

$$\begin{bmatrix} 1 & 11 & 16 & 8 \\ 19 & 29 & 23 & 26 \\ 2 & 19 & 17 & 1 \\ 8 & 13 & 17 & 10 \end{bmatrix}$$


經過 max-pooling 之後...

filter 1

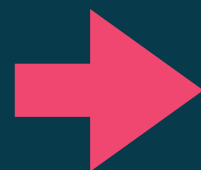


$$\begin{bmatrix} 23 & 23 \\ 12 & 15 \end{bmatrix}$$

拉平



filter 2



$$\begin{bmatrix} 29 & 26 \\ 19 & 17 \end{bmatrix}$$

23

23

12

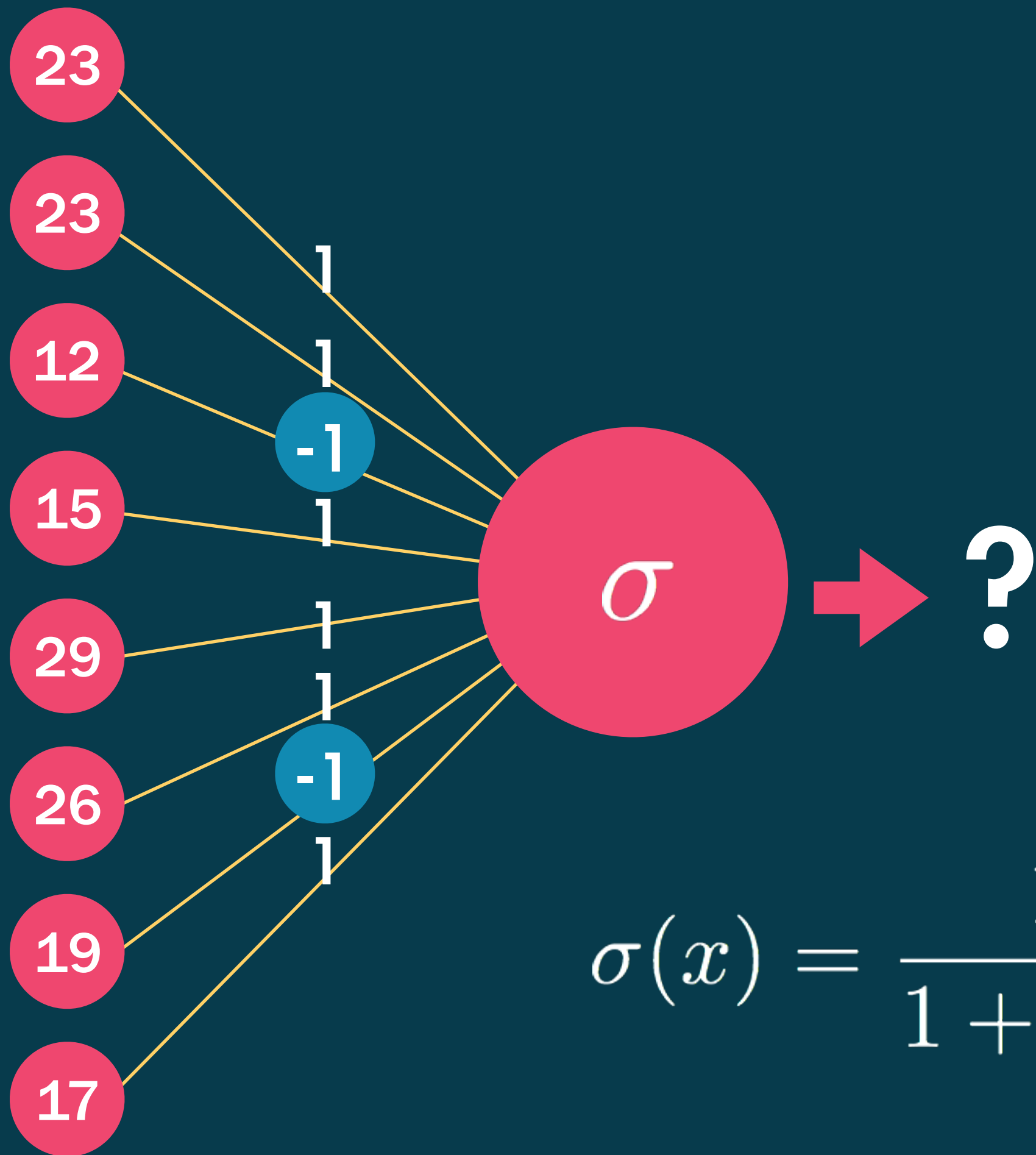
15

29

26

19

17



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

最後的輸出...

$$\sigma(102)=1.0$$

問題討論

問題 1.

Cross entropy 為一個常用的 loss function:

$$L(\theta) = -\frac{1}{n} \sum_{i=1}^n [y_i \log f_{\theta}(x_i) + (1 - y_i) \log(1 - f_{\theta}(x_i))]$$

注意正確答案 $y=1$, 我們只有

$$-y \log f_{\theta}(x) = -\log f_{\theta}(x)$$

但萬一我們遜遜以為是 0...

$$-\log f_{\theta}(x) = -\log 0$$



問題2.

我們 activation function 選用 sigmoid function,

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

微一微會得到很可愛的式子:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

問題3.

你在做 gradient descent 時會發現, 你還要對 \max 微分!! 然後數學魔人又要再度現身, 會告訴你 \max 不可以微分!!

某種意義下, ReLU 不可以微分那個狀況還糟... 這該如何是好?



RNN

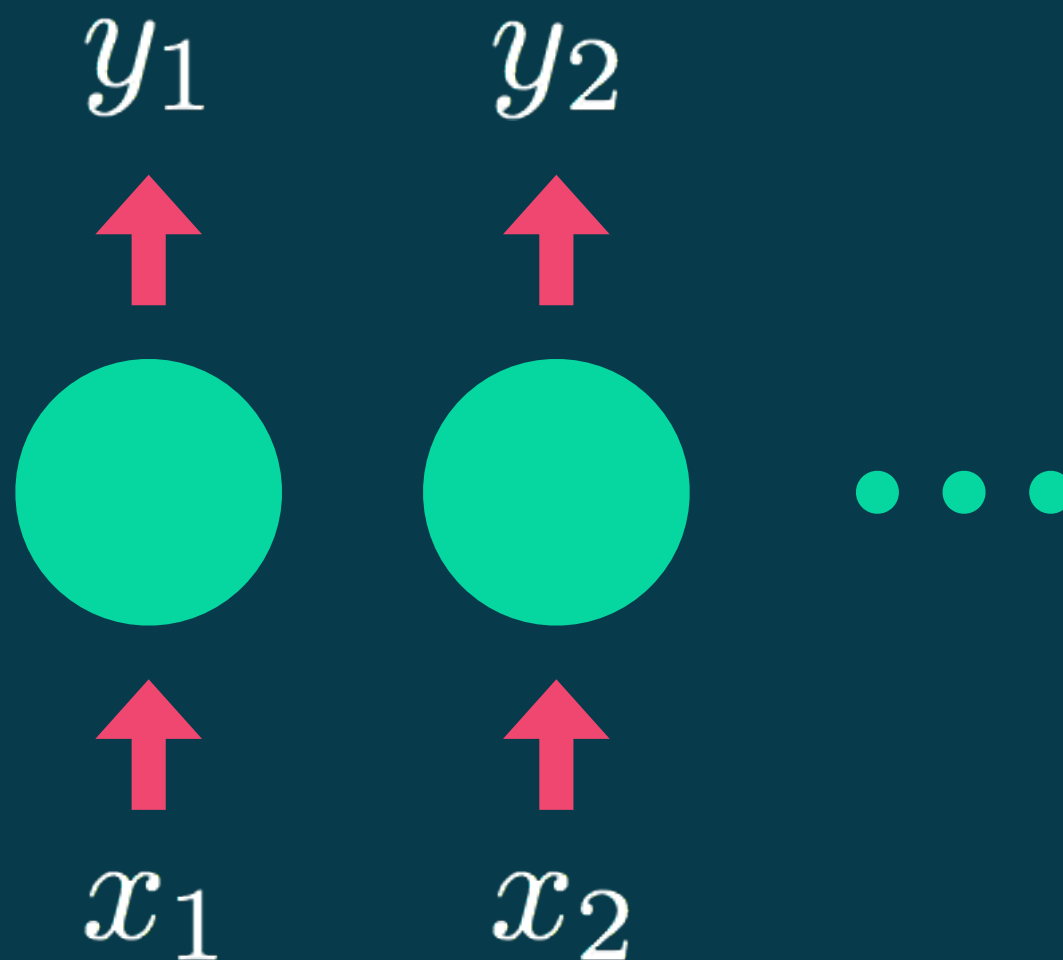
Recurrent Neural Networks

有記憶的神經網路

重點

RNN 的特性

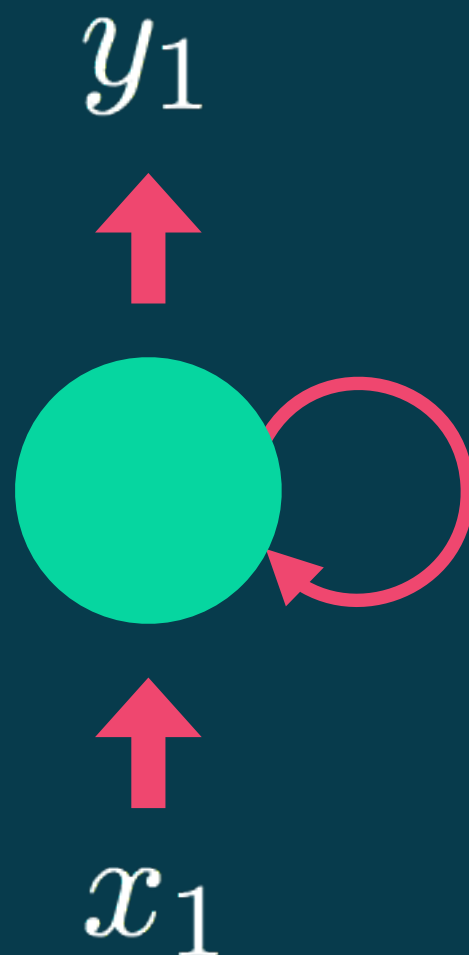
一般的神經網路一筆輸入和下一筆是沒有關係的...



重點

RNN 的特性

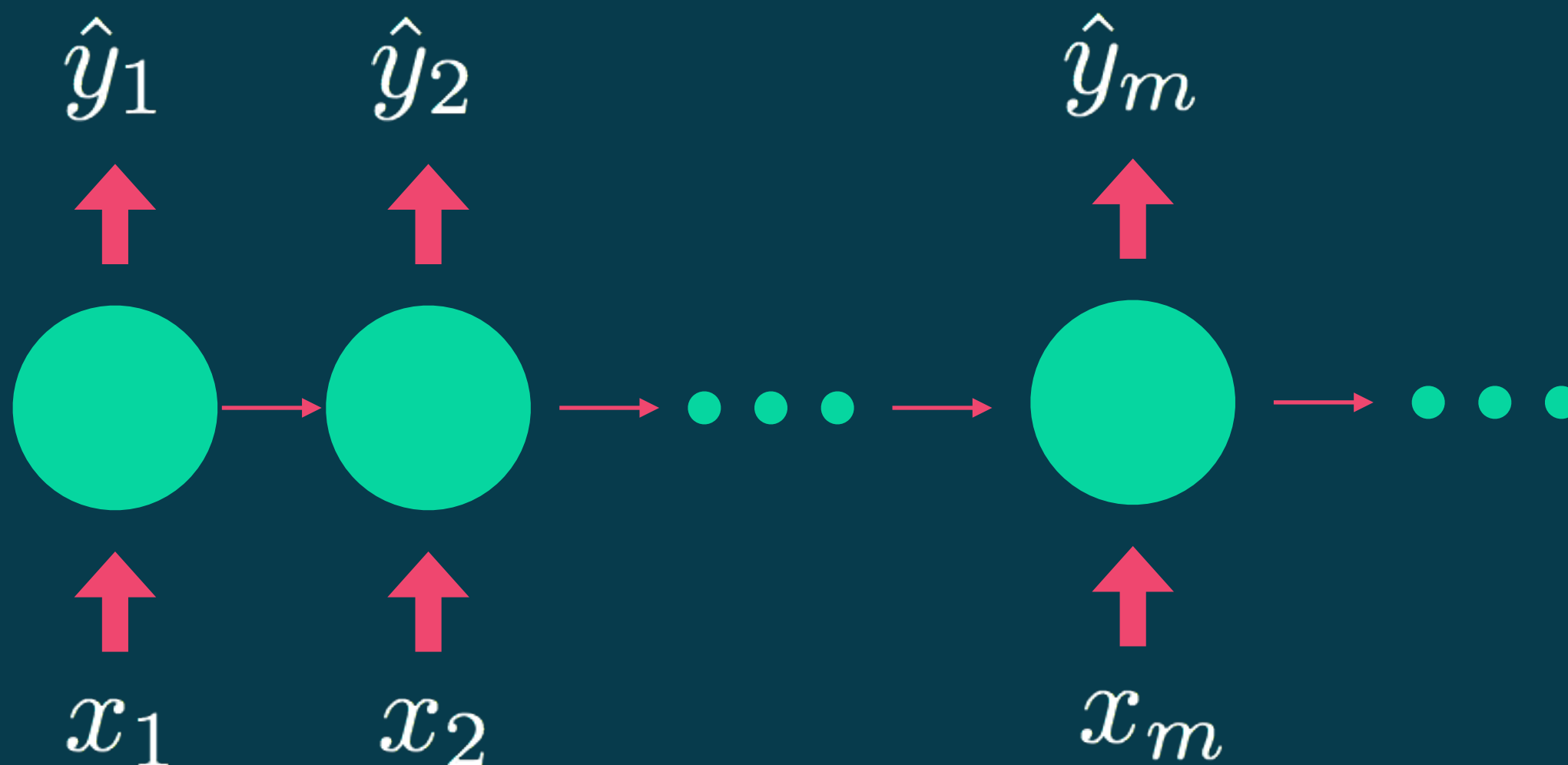
RNN 是「有記憶」的神經網路。



重點

RNN 的特性

很多人畫成這樣。

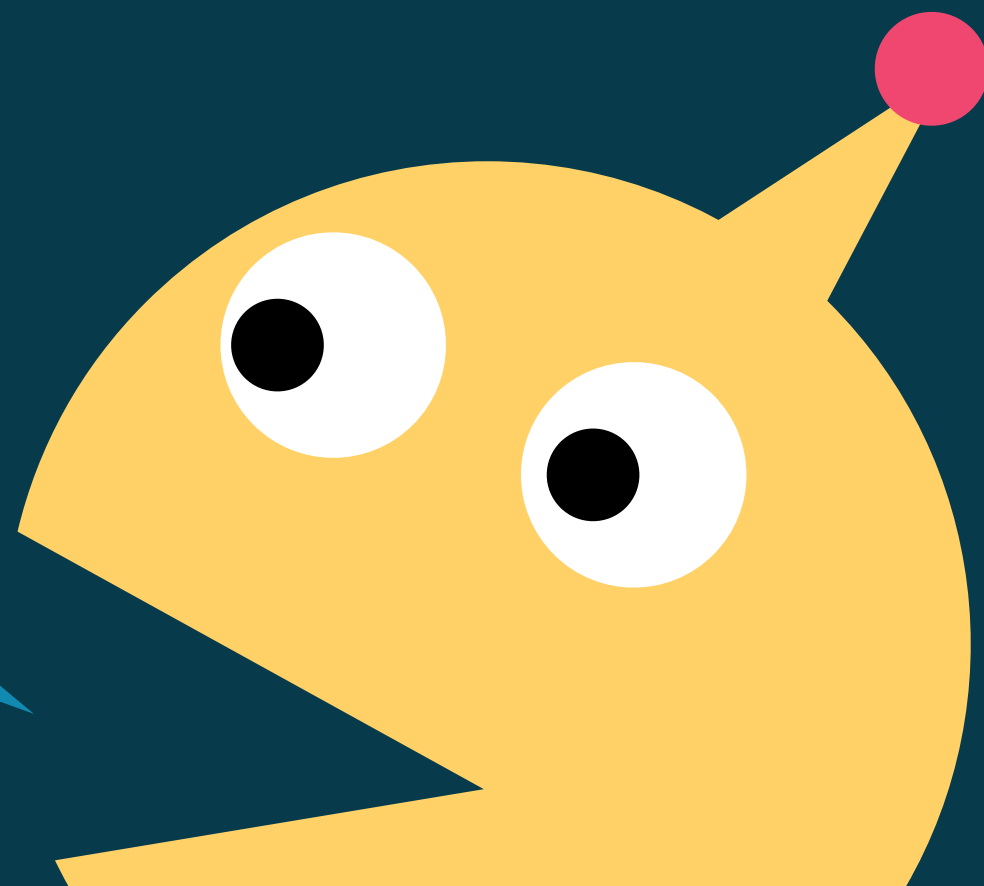


這能有什麼用？

應用

對話機器人

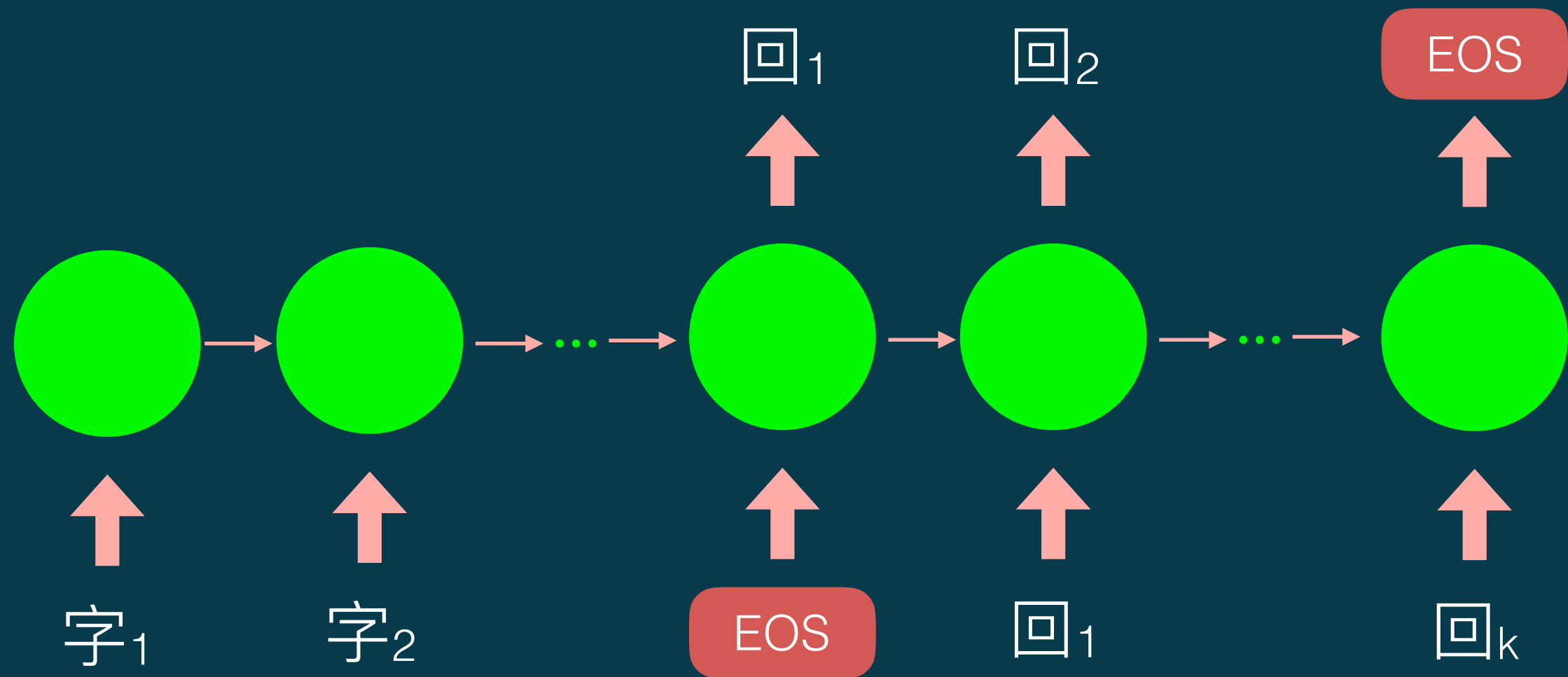
對話機器人



Sequence-to-Sequence
Neural Networks

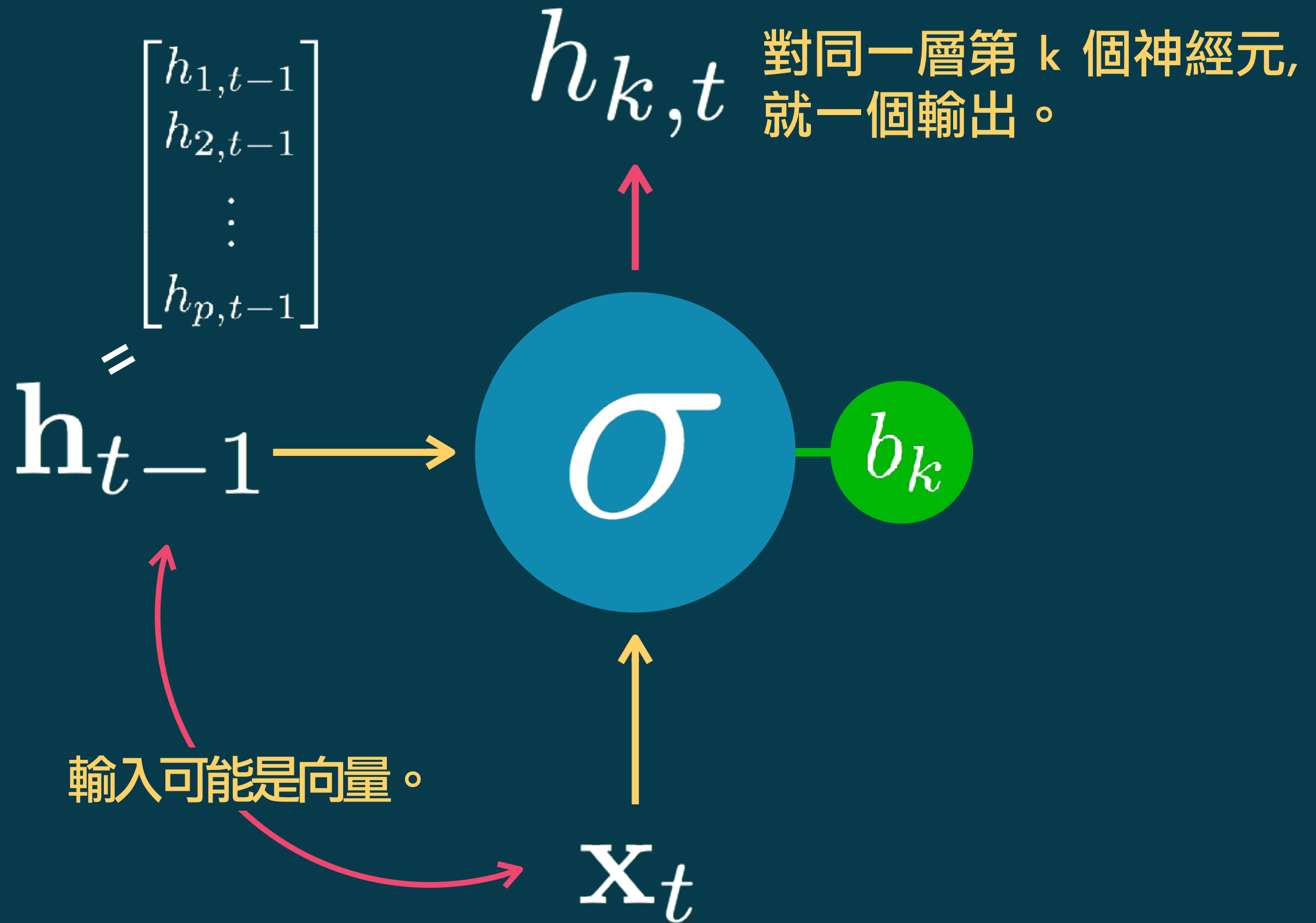
應用

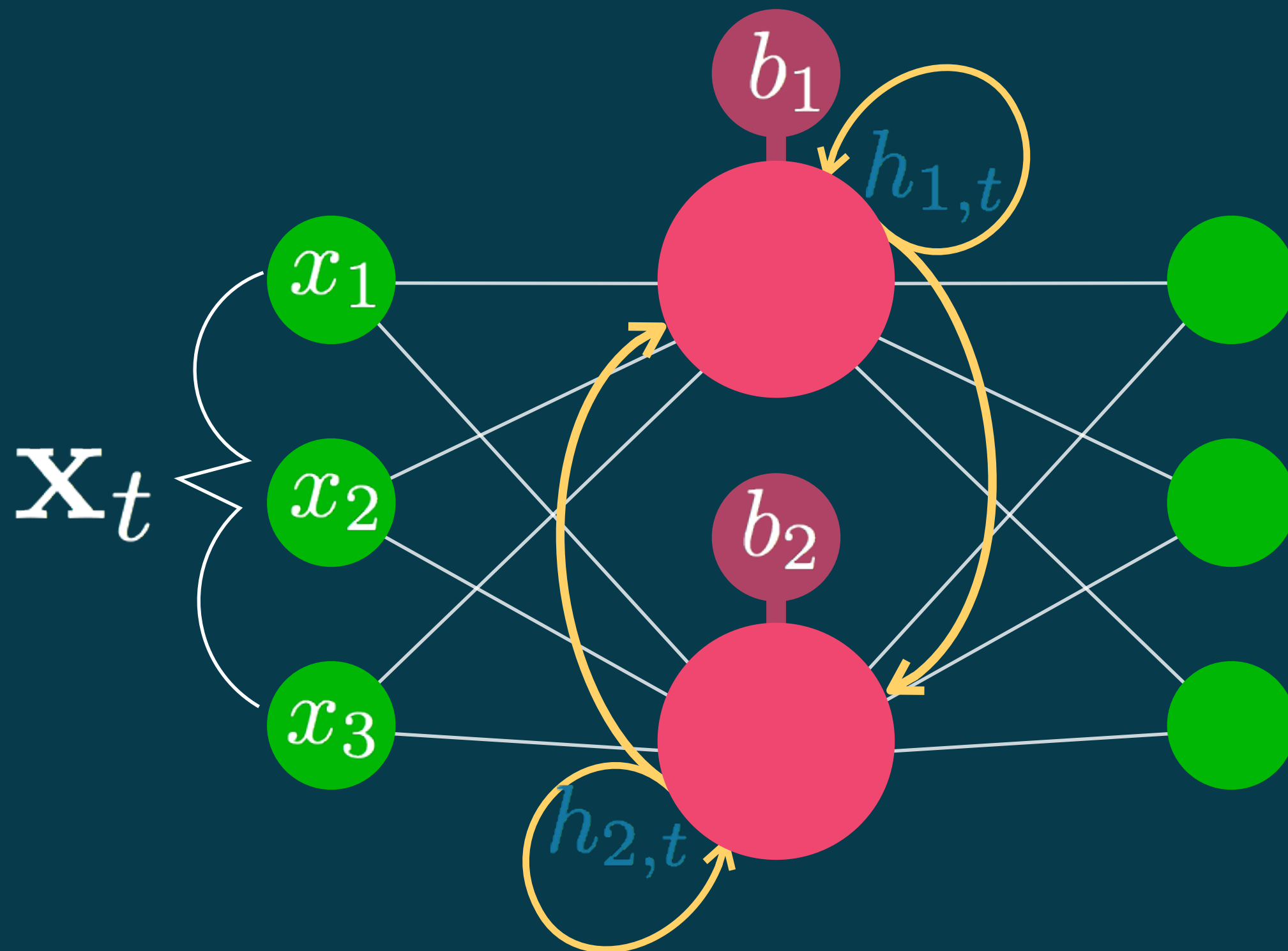
對話機器人



基本 RNN 長這樣

一個 RNN 的 neuron





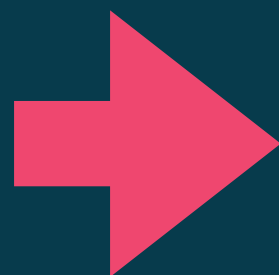
手寫 RNN

問題

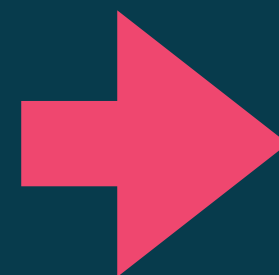
RNN 預測下一個字

我們來做標準 RNN 應用, 預測一句話
下一個字。

字



**用到 RNN 的
Neural
Networks**



字

(下一個)

字集

我們考慮的字集只有三個字。

“好”，“一”，“點”

可以產生如“好一點”，“好好”，“一點一點點”，等等的「句子」。

字集

用高級的 one hot 編碼。

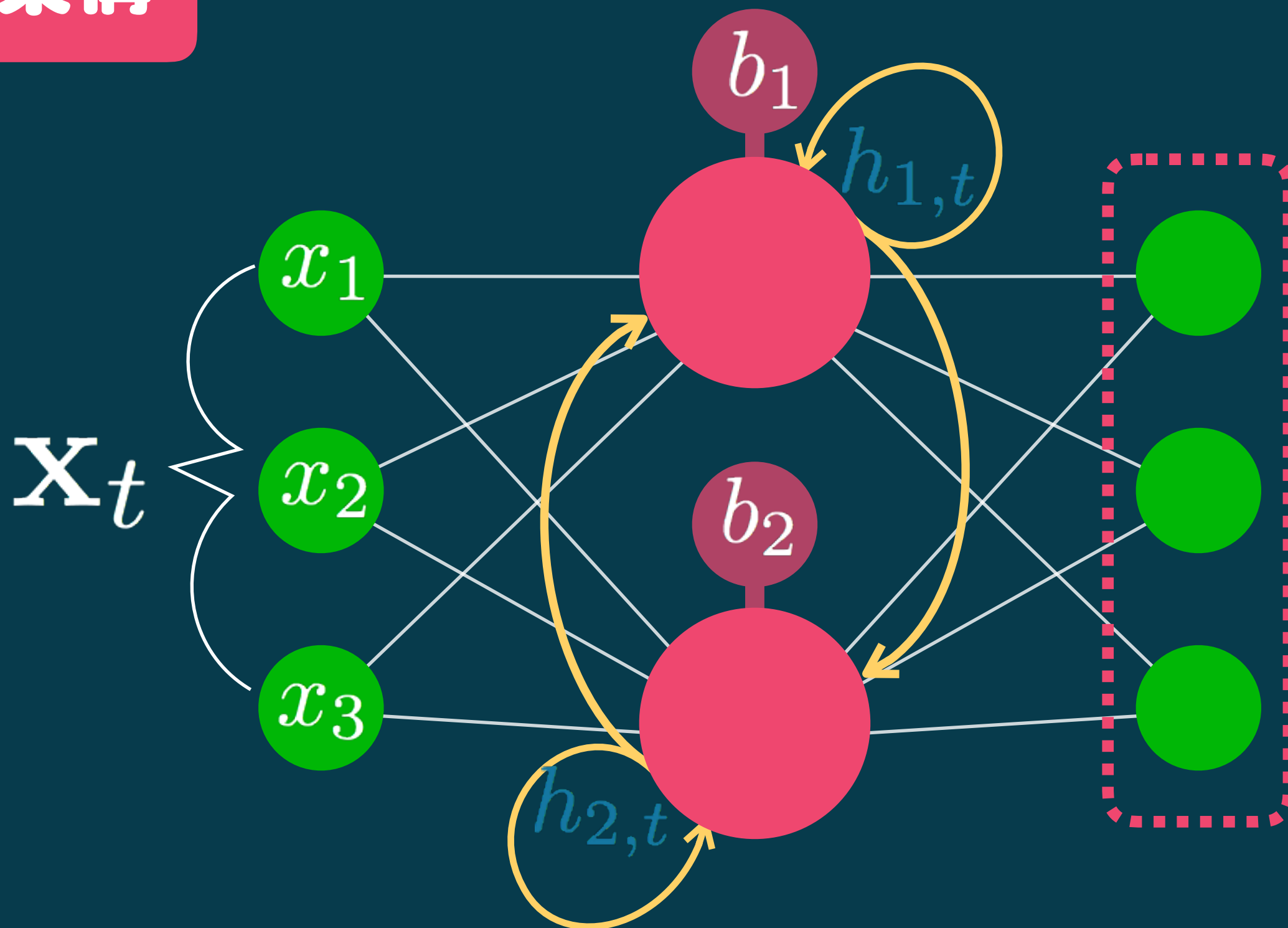
“好”，“一”，“點”


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

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

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

架構



輸入

“好一點”

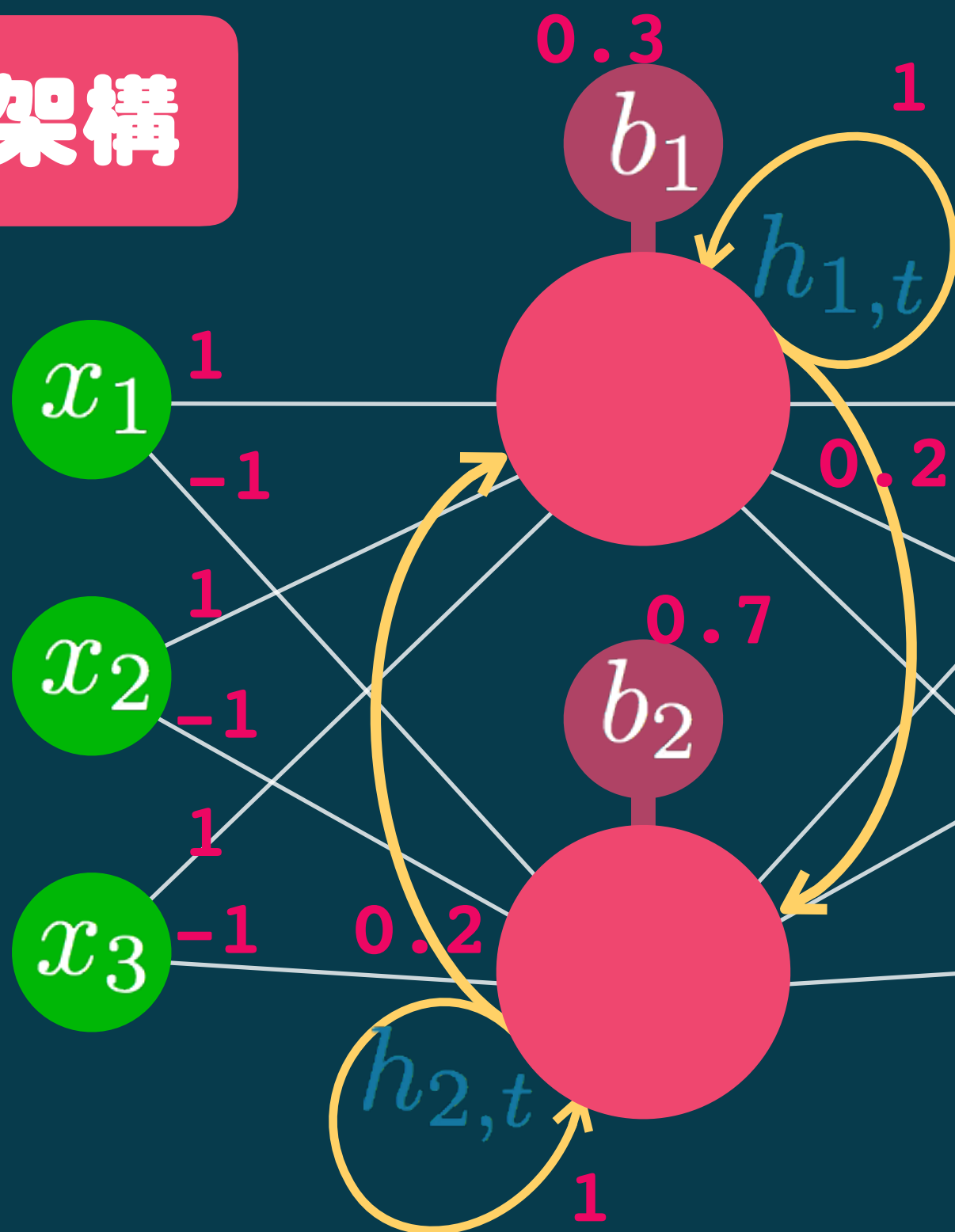
→ $\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$

x_1

x_2

x_3

架構



$$W_h = \begin{bmatrix} 1 & 0.2 \\ 1 & 0.2 \end{bmatrix}$$

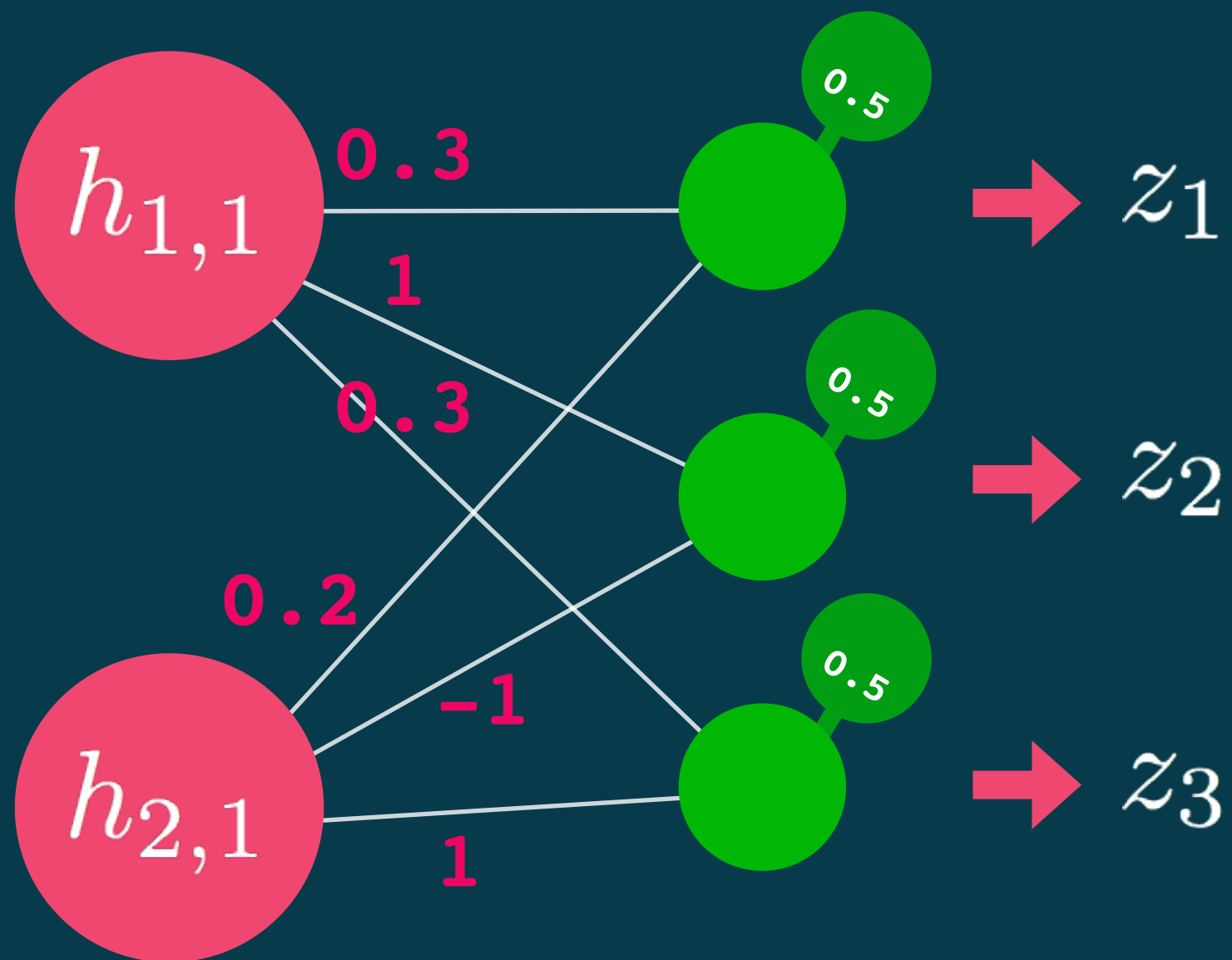
$$W_x = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix}$$

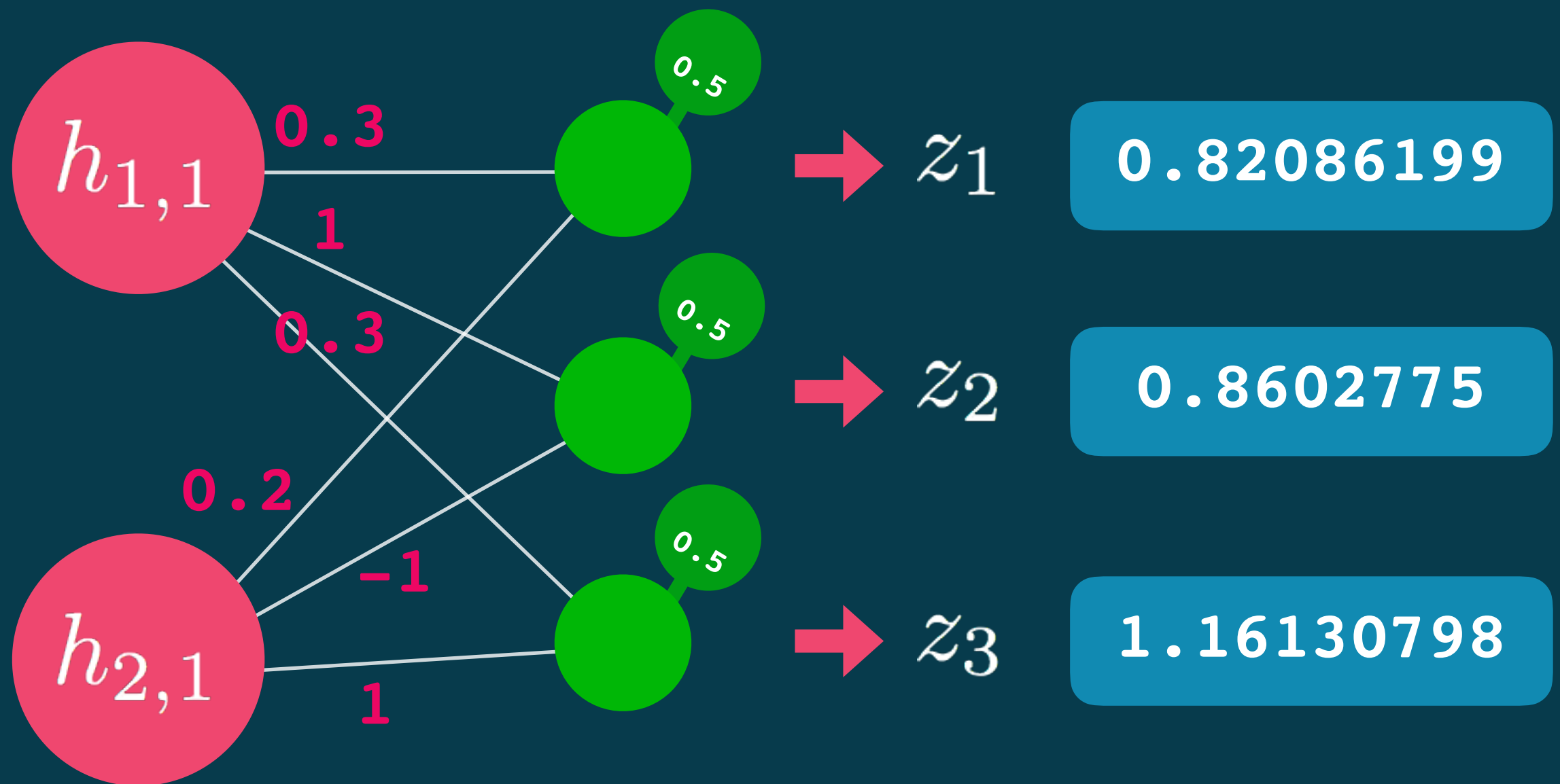
$$\mathbf{h}_t = \sigma(W_h \mathbf{h}_{t-1} + W_x \mathbf{x}_t + \mathbf{b})$$

$$\mathbf{h}_1 = \begin{bmatrix} h_{1,1} \\ h_{2,1} \end{bmatrix} = \begin{bmatrix} 0.78583498 \\ 0.42555748 \end{bmatrix}$$

$$\mathbf{h}_1 = \begin{bmatrix} h_{1,1} \\ h_{2,1} \end{bmatrix} = \begin{bmatrix} 0.78583498 \\ 0.42555748 \end{bmatrix}$$



$$\mathbf{h}_1 = \begin{bmatrix} h_{1,1} \\ h_{2,1} \end{bmatrix} = \begin{bmatrix} 0.78583498 \\ 0.42555748 \end{bmatrix}$$



概念

softmax

z_1

0.82086199

z_2

0.8602775

z_3

1.16130798

希望加起來是 1

概念

softmax

z_1



$$\frac{e^{z_1}}{\sum_{i=1}^3 e^{z_i}}$$

z_2

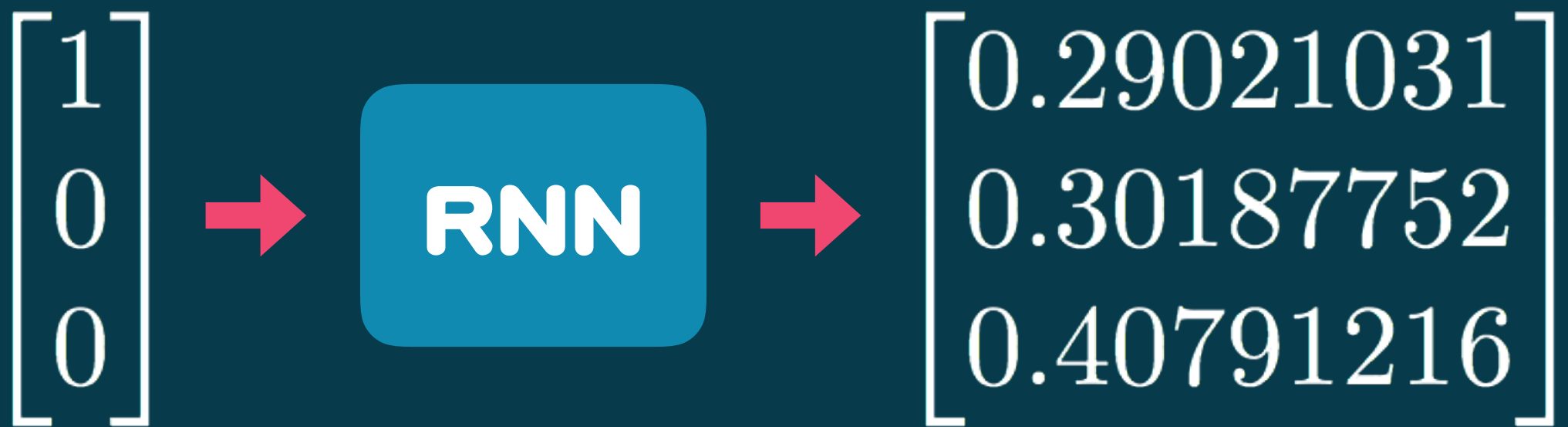


$$\frac{e^{z_2}}{\sum_{i=1}^3 e^{z_i}}$$

z_3



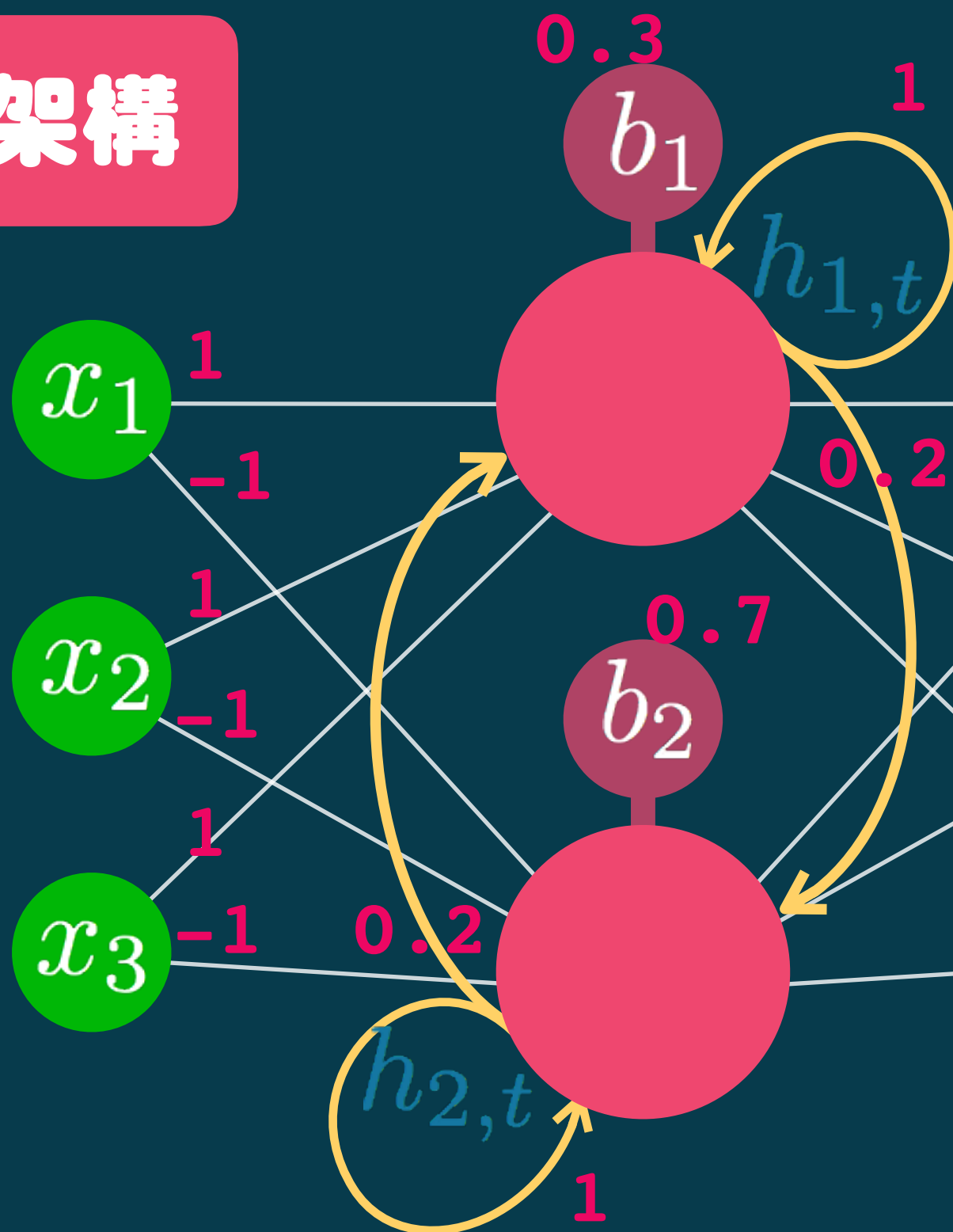
$$\frac{e^{z_3}}{\sum_{i=1}^3 e^{z_i}}$$



再來輸入

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

架構



$$W_h = \begin{bmatrix} 1 & 0.2 \\ 1 & 0.2 \end{bmatrix}$$

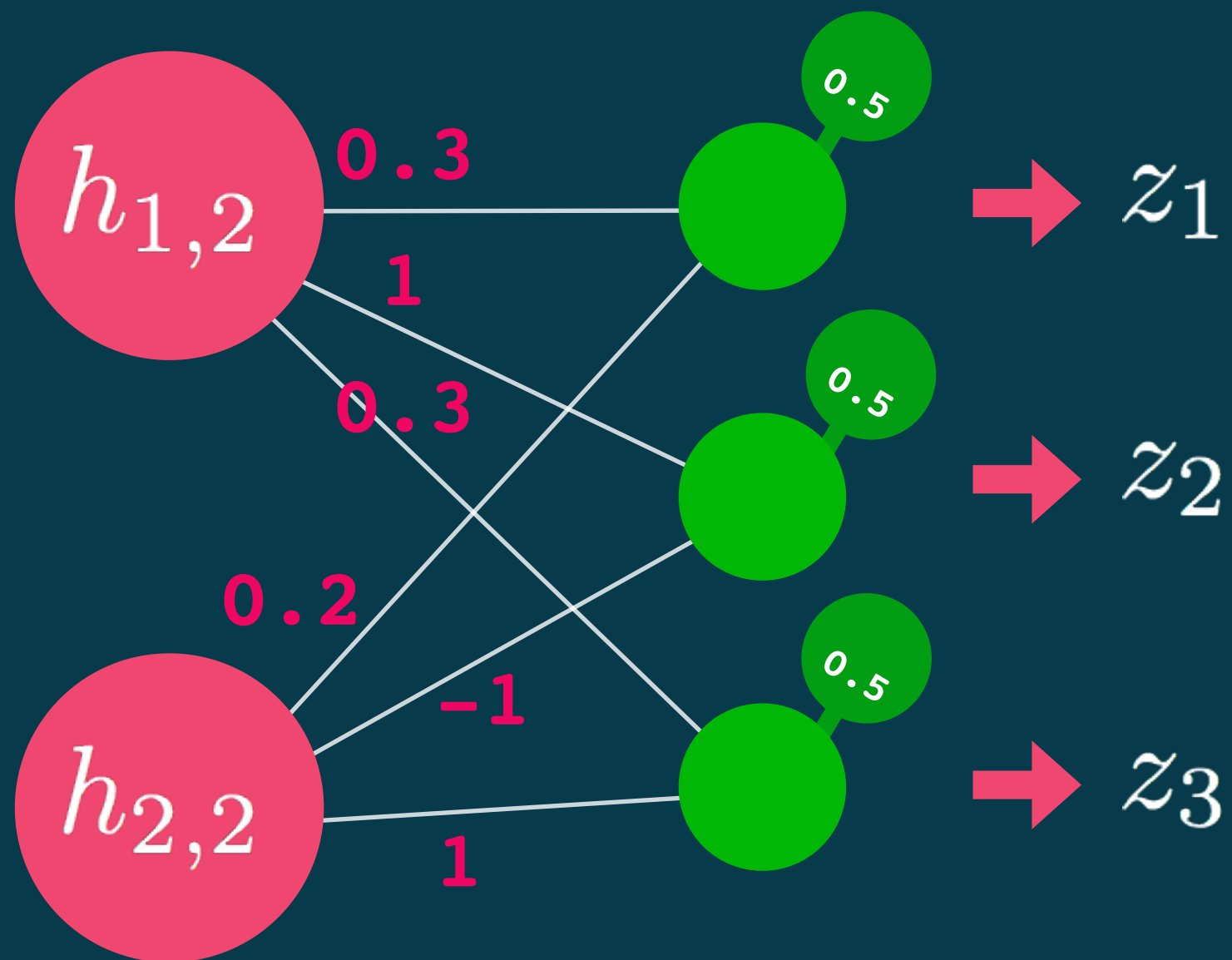
$$W_x = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix}$$

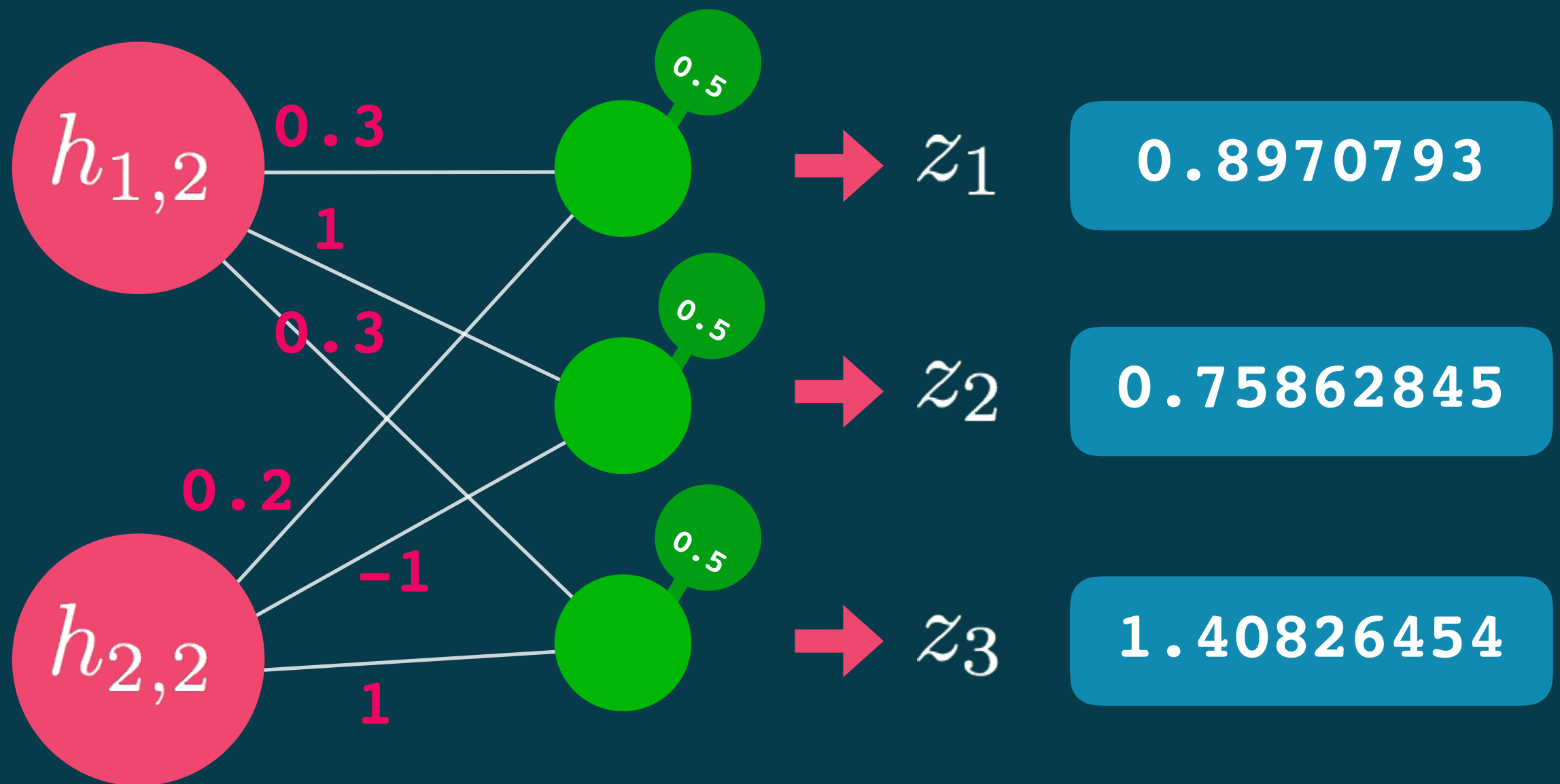
$$\mathbf{h}_t = \sigma(W_h \mathbf{h}_{t-1} + W_x \mathbf{x}_t + \mathbf{b})$$

$$\mathbf{h}_2 = \begin{bmatrix} h_{1,2} \\ h_{2,2} \end{bmatrix} = \begin{bmatrix} 0.89760999 \\ 0.63898154 \end{bmatrix}$$

$$\mathbf{h}_2 = \begin{bmatrix} h_{1,2} \\ h_{2,2} \end{bmatrix} = \begin{bmatrix} 0.89760999 \\ 0.63898154 \end{bmatrix}$$



$$\mathbf{h}_2 = \begin{bmatrix} h_{1,2} \\ h_{2,2} \end{bmatrix} = \begin{bmatrix} 0.89760999 \\ 0.63898154 \end{bmatrix}$$



概念

softmax

z_1



$$\frac{e^{z_1}}{\sum_{i=1}^3 e^{z_i}}$$

z_2



$$\frac{e^{z_2}}{\sum_{i=1}^3 e^{z_i}}$$

z_3



$$\frac{e^{z_3}}{\sum_{i=1}^3 e^{z_i}}$$

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



$$\begin{bmatrix} 0.28264779 \\ 0.24610313 \\ 0.47124908 \end{bmatrix}$$



h_1

其實我有點騙你

RNN 很不容易訓練

Vanishing Gradient



LSTM

Long Short Term
Memory



GRU

Gated Recurrent Unit



LSTM

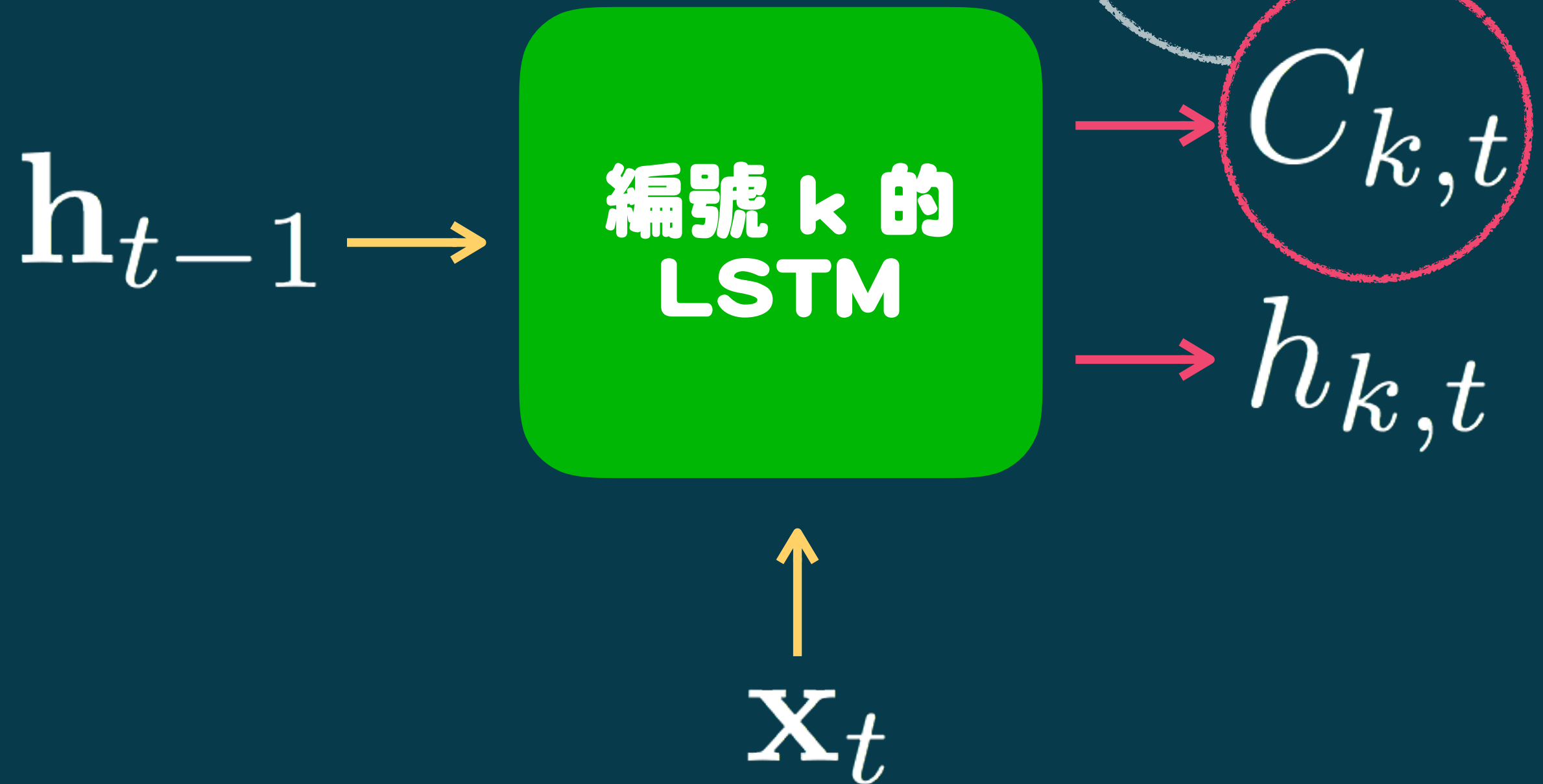
Long Short Term
Memory



GRU

Gated Recurrent Unit

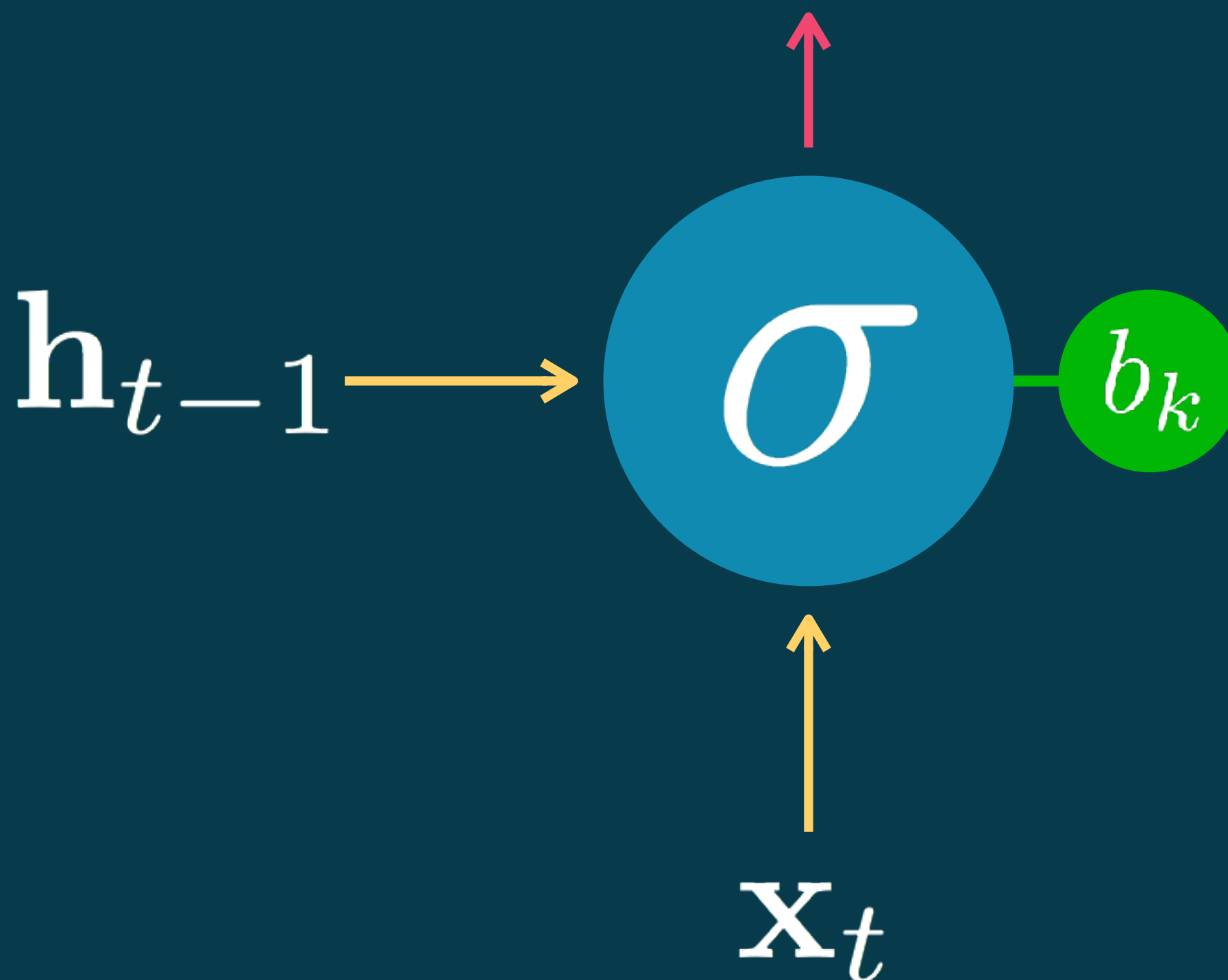
多一個「cell 狀態」



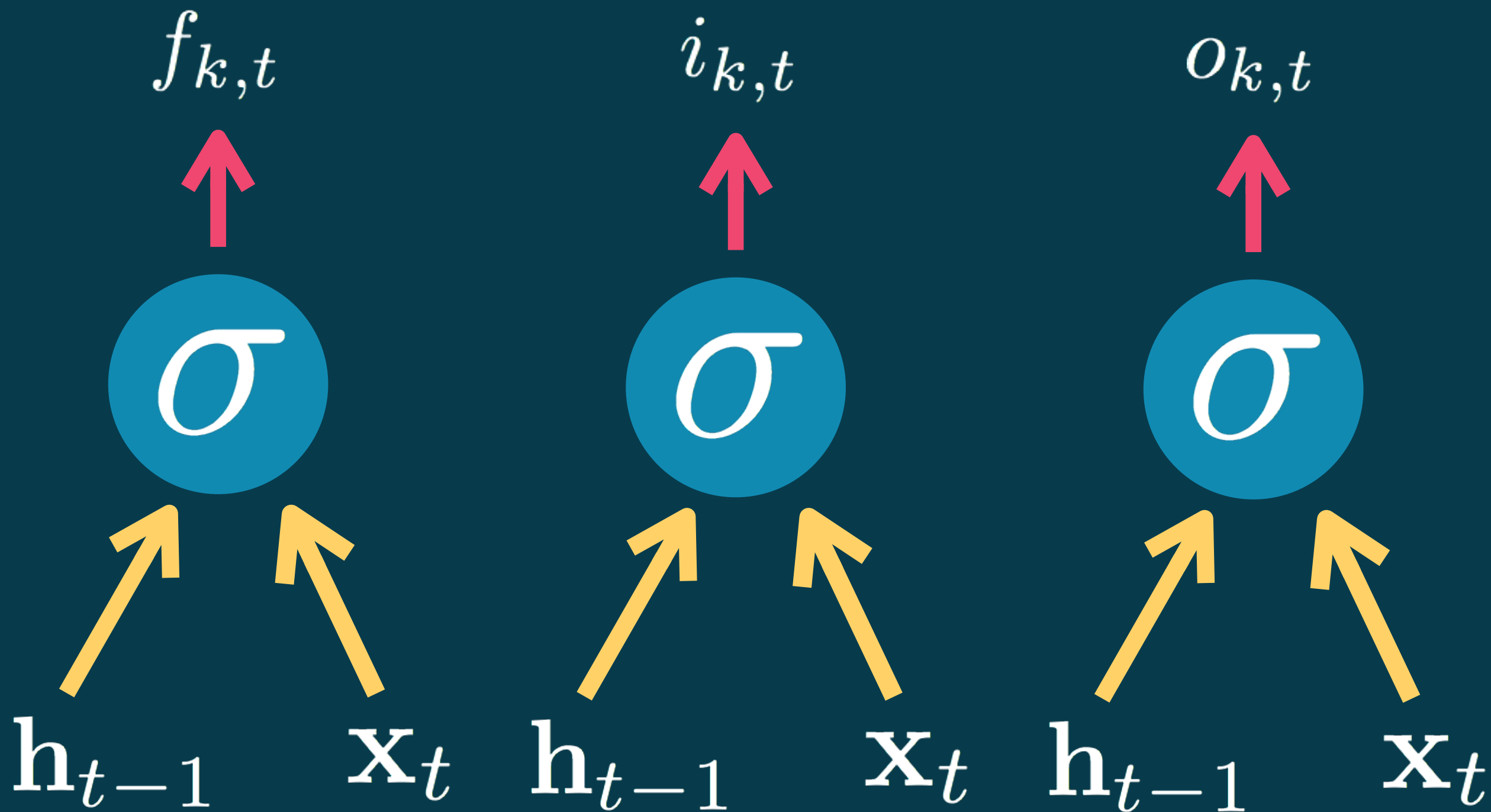
Gate

控制閥

輸出 0 到 1 間的一個數



LSTM 有三個 Gates



忘記門

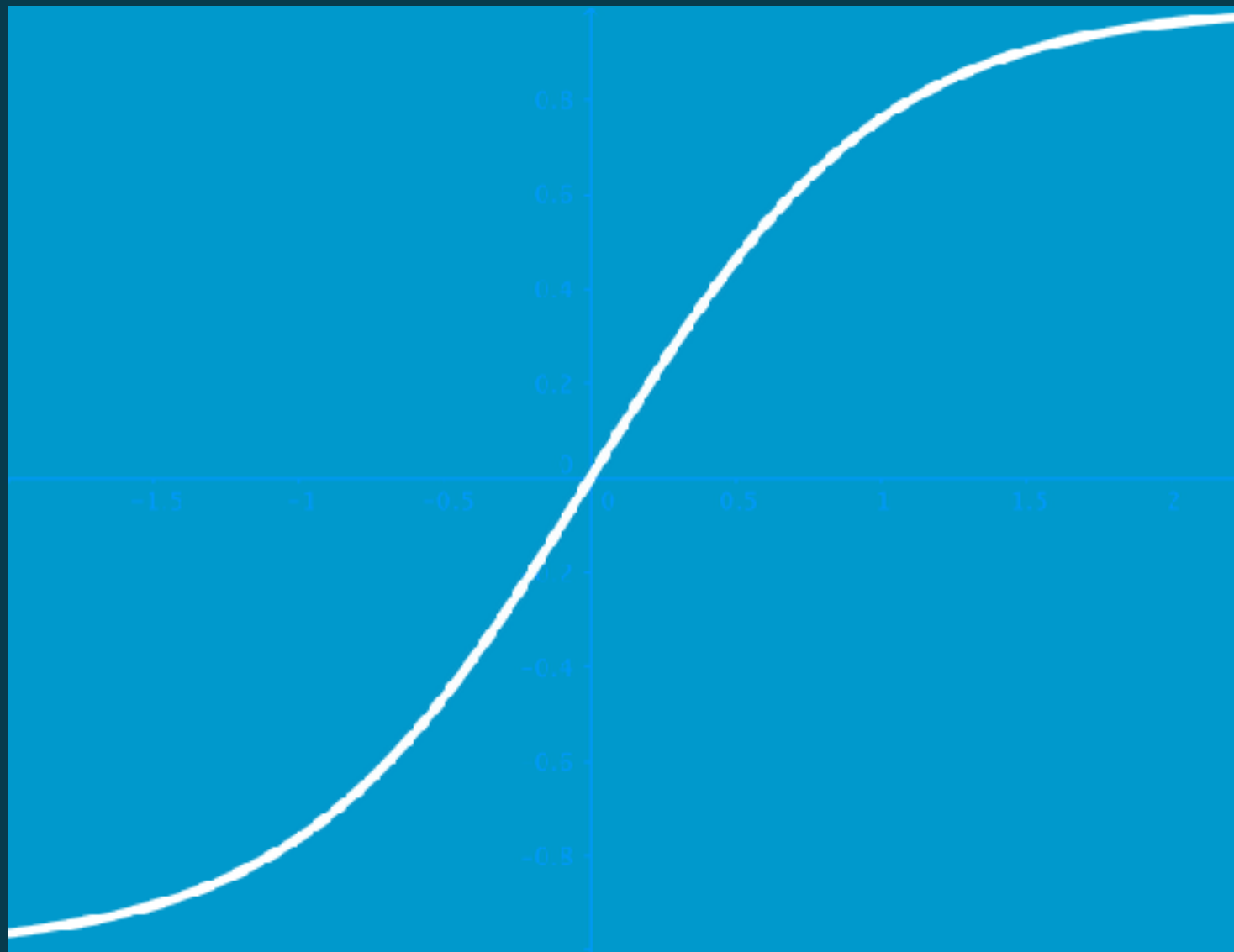
輸入門

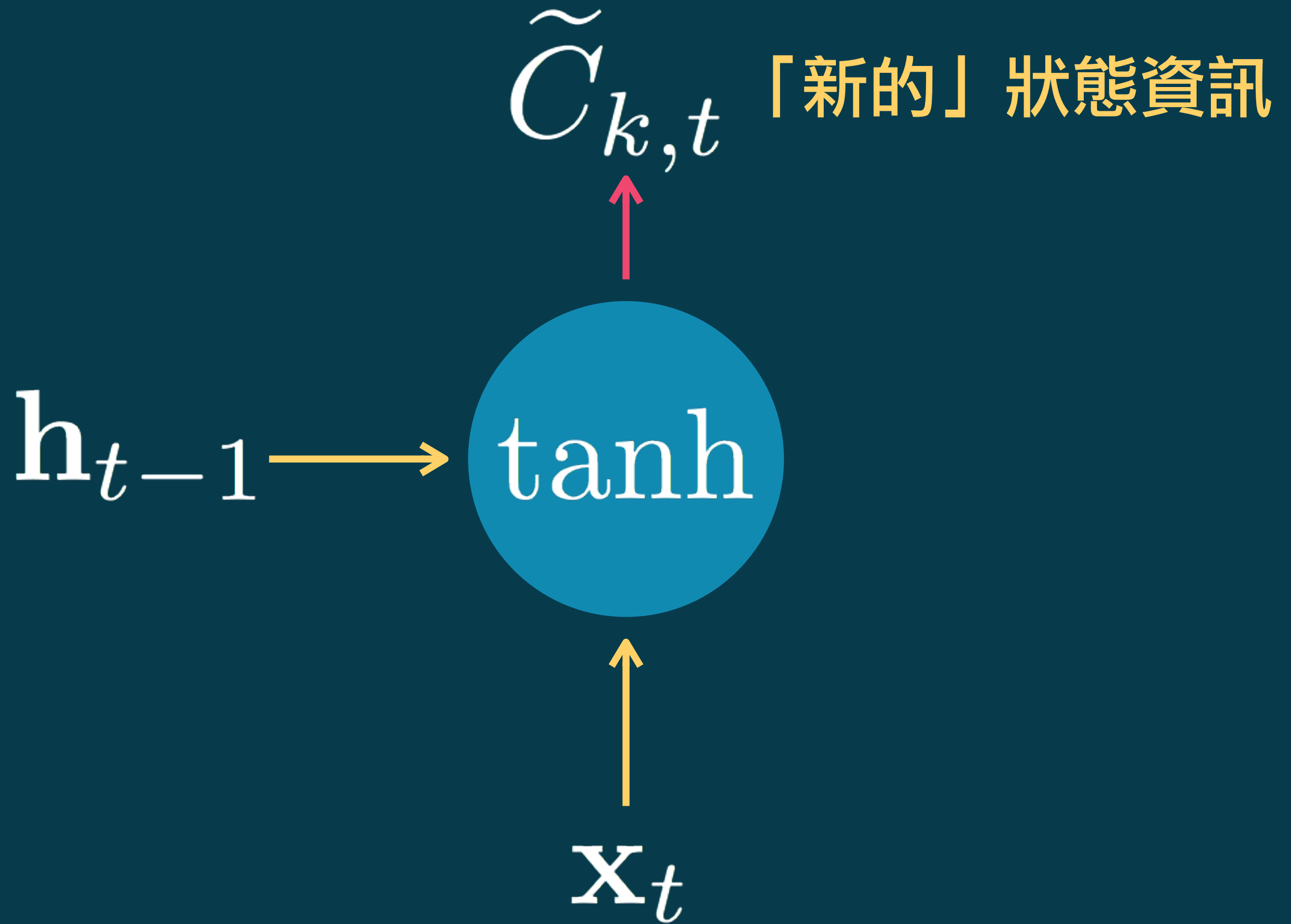
輸出門

插播

可愛的 \tanh

$$\tanh(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$





$$C_{k,t} = f_{k,t}C_{k,t-1} + i_{k,t}\tilde{C}_{k,t}$$

$$h_{k,t} = o_{k,t} \tanh(C_{k,t})$$

真要弄得那麼複雜?

要忘多少和要記多少難道不能一起...



LSTM

Long Short Term
Memory

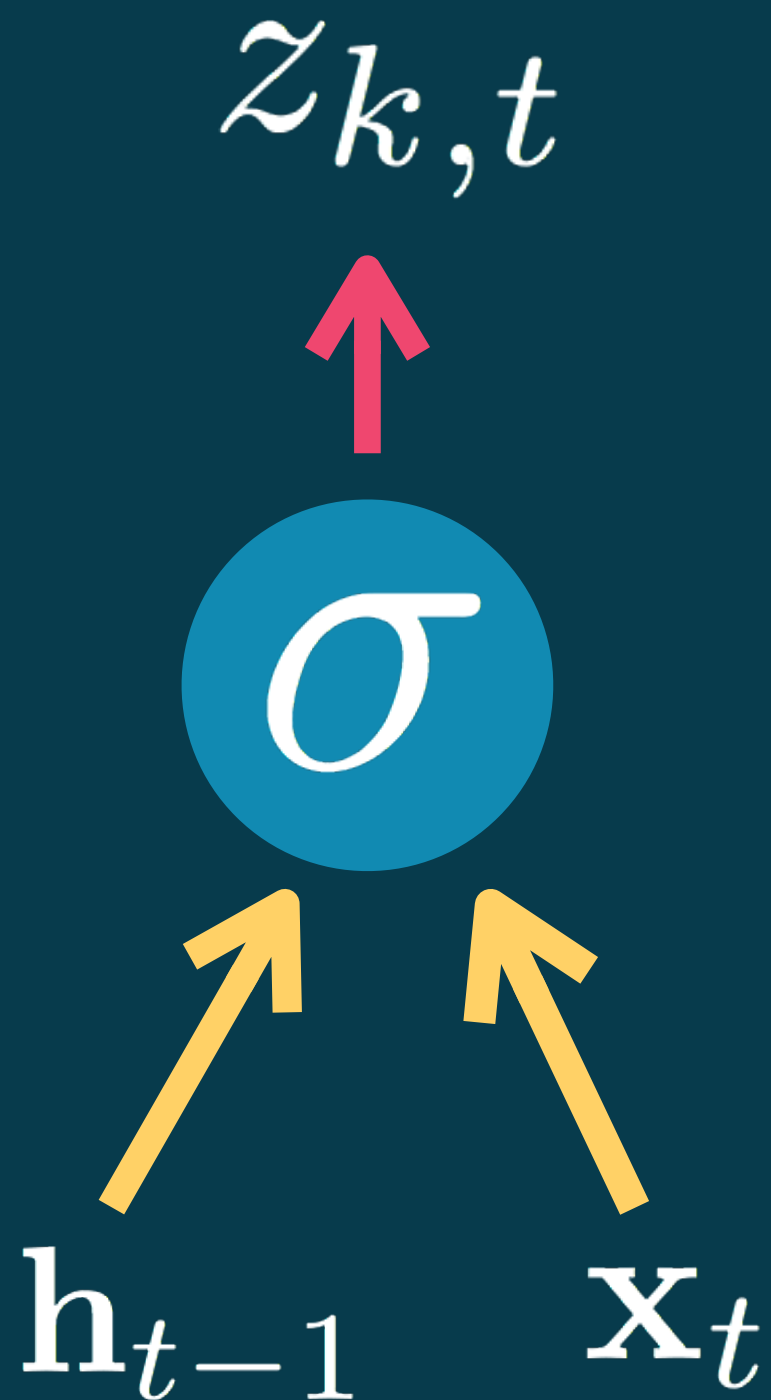


GRU

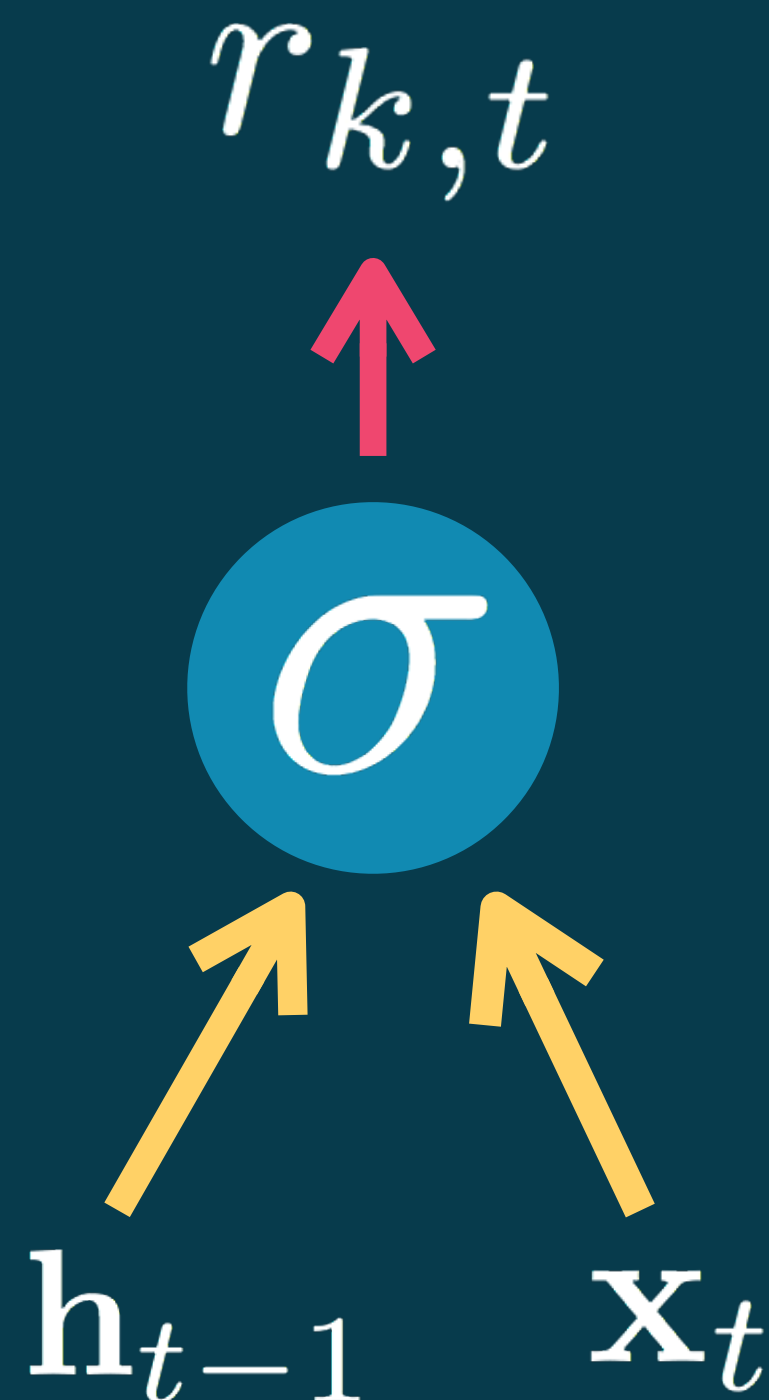
Gated Recurrent Unit

只留兩個 Gates

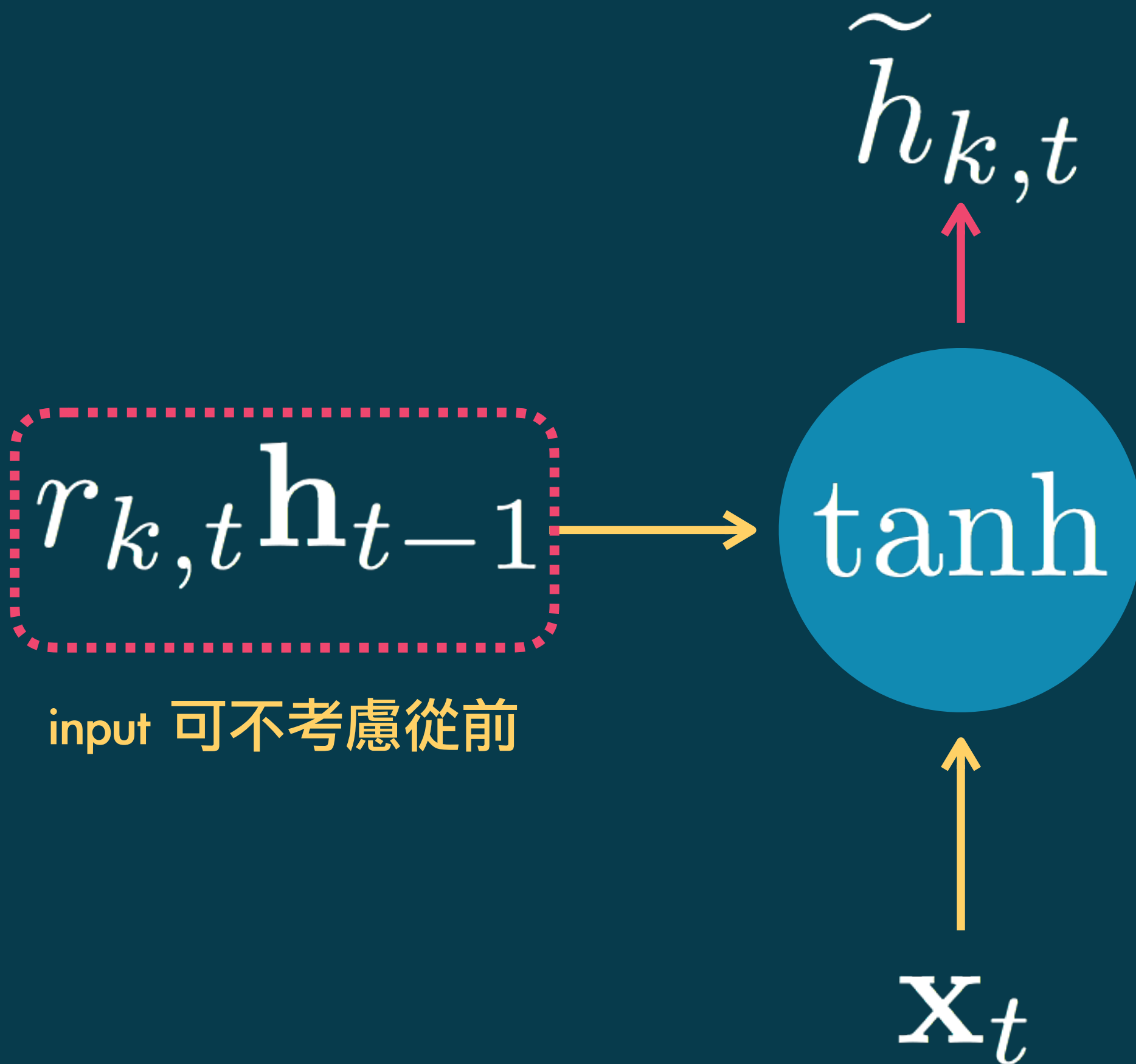
雖然名稱有 gated



記憶門



重設門



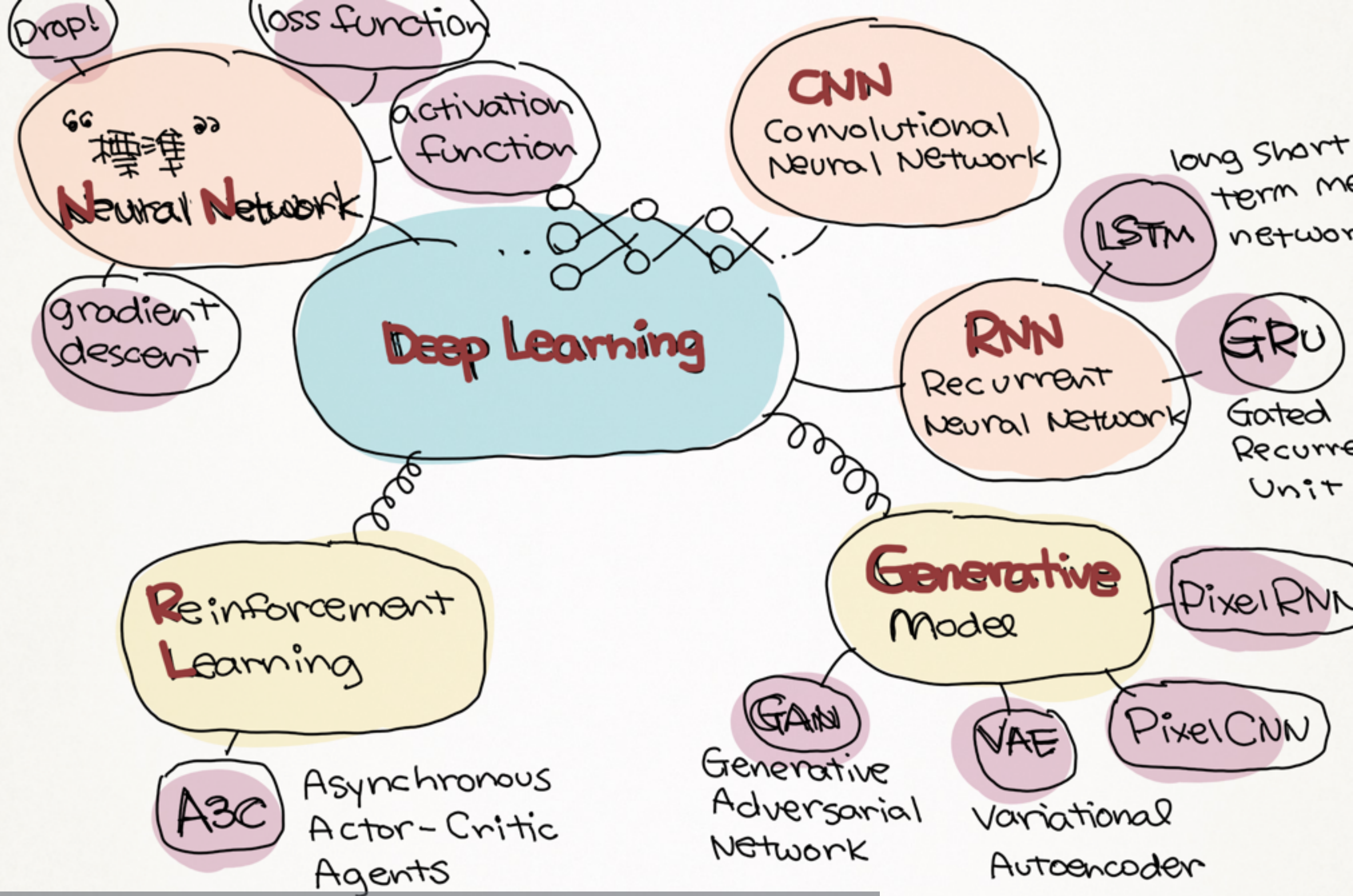
$$h_{k,t} = z_{k,t}h_{k,t-1} + (1 - z_{k,t})\tilde{h}_{k,t}$$

4

深度學習

深度學習「深度學習」

- 1 要相信電腦學得會
- 2 要有夠多的訓練資料
- 3 Dropout 避免 overfitting



亂畫的 deep learning 學習地圖

• yerlung •



很棒的 Keras 範例集 (出自原作)

[https://github.com/fchollet/
keras/tree/master/examples](https://github.com/fchollet/keras/tree/master/examples)

Learning Deep Learning with Keras

[http://p.migdal.pl/2017/04/30/
teaching-deep-learning.html](http://p.migdal.pl/2017/04/30/teaching-deep-learning.html)

很棒的 deep learning 學習指引, 介紹很多學習資源。

李宏毅老師的課程

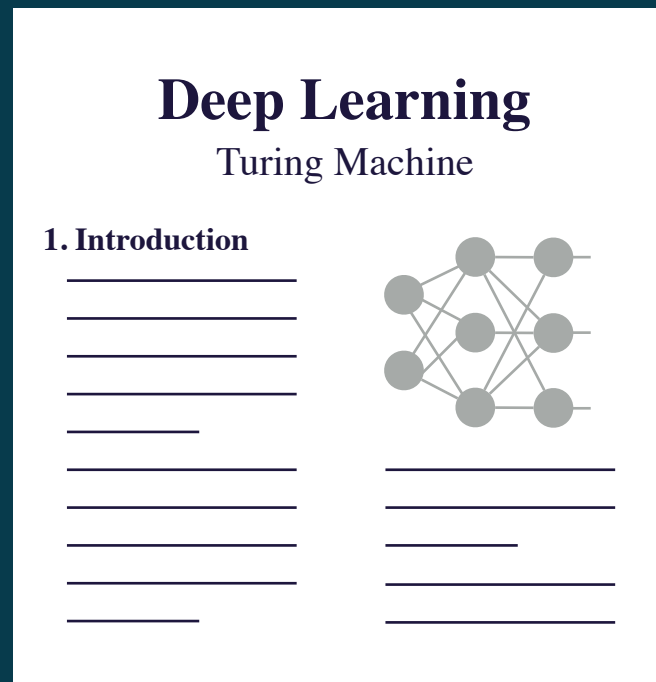
[http://speech.ee.ntu.edu.tw/~tlkagk/
courses_ML16.html](http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML16.html)

非常棒的深度學習影片教學。

Ian Goodfellow, Yoshua Bengio and
Aaron Courville

Deep Learning

<http://www.deeplearningbook.org/>



經典文獻

[https://github.com/terryum/
awesome-deep-learning-papers](https://github.com/terryum/awesome-deep-learning-papers)



FaceBook/**yenlung**
yenlung@nccu.edu.tw