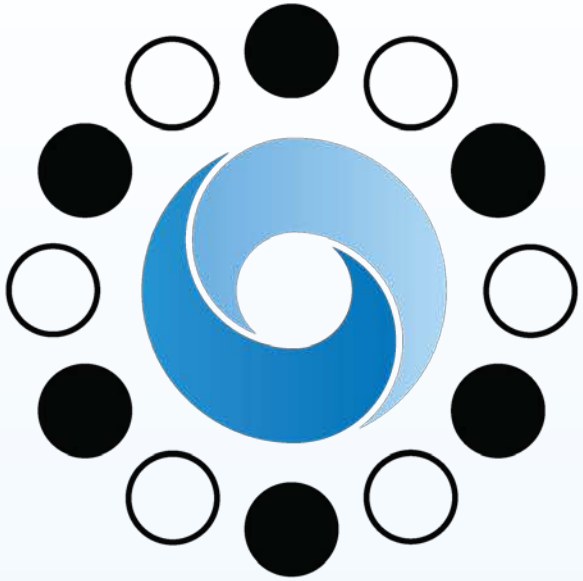
深度學習概論

Introduction to Deep Learning

蔡瑞煌 老師

國立政治大學金融科技研究中心
智能理財與深度學習暑期訓練營



AlphaGo

為什麼要機器學習?

Why we want Machine can be Learning?

人工智慧

機器學習

深度學習

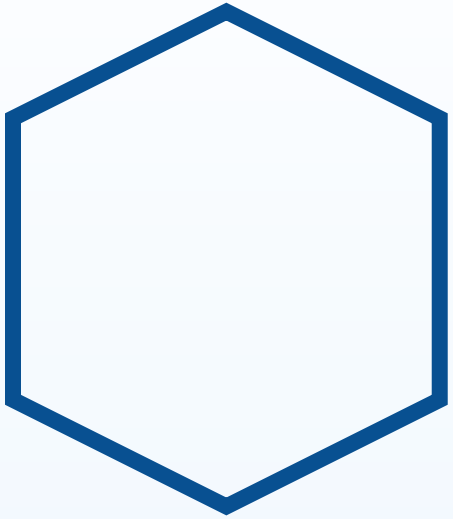
一連串條件判斷式也是一種人工智慧

IBM深藍

IBM DeepBlue

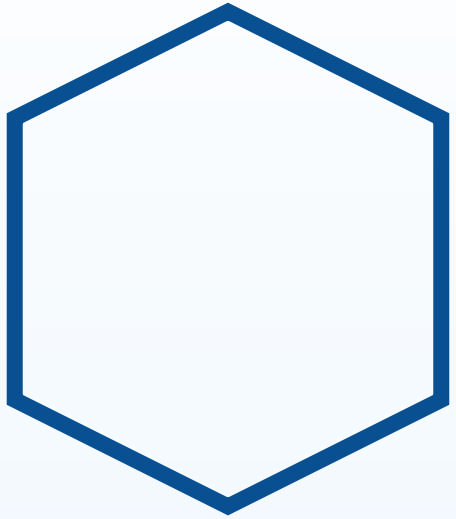
條件式人工智慧

Rule-Based AI’ s Problem



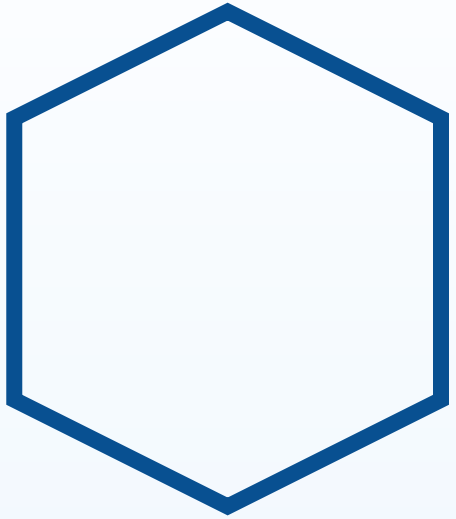
條件建立困難

Rule Build Difficult



笨重

Cumbersome



昂貴

Expensive

條件判斷人工智慧更像是工人智慧

為什麼我們不讓機器自己找到規則關係呢？

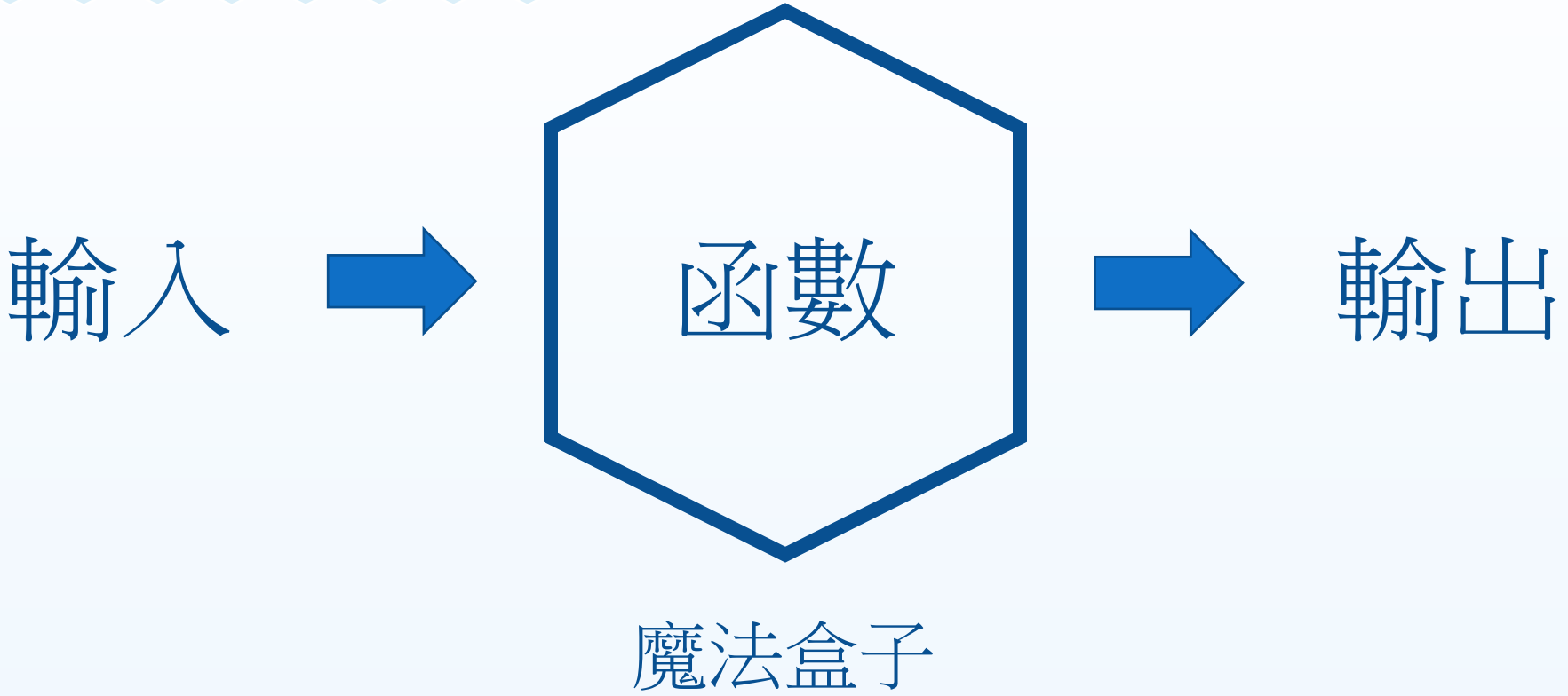
函數是一種關係

Function is a relation

Function, Wikipedia

函數是個魔法盒子

Funtion is an Magic Box

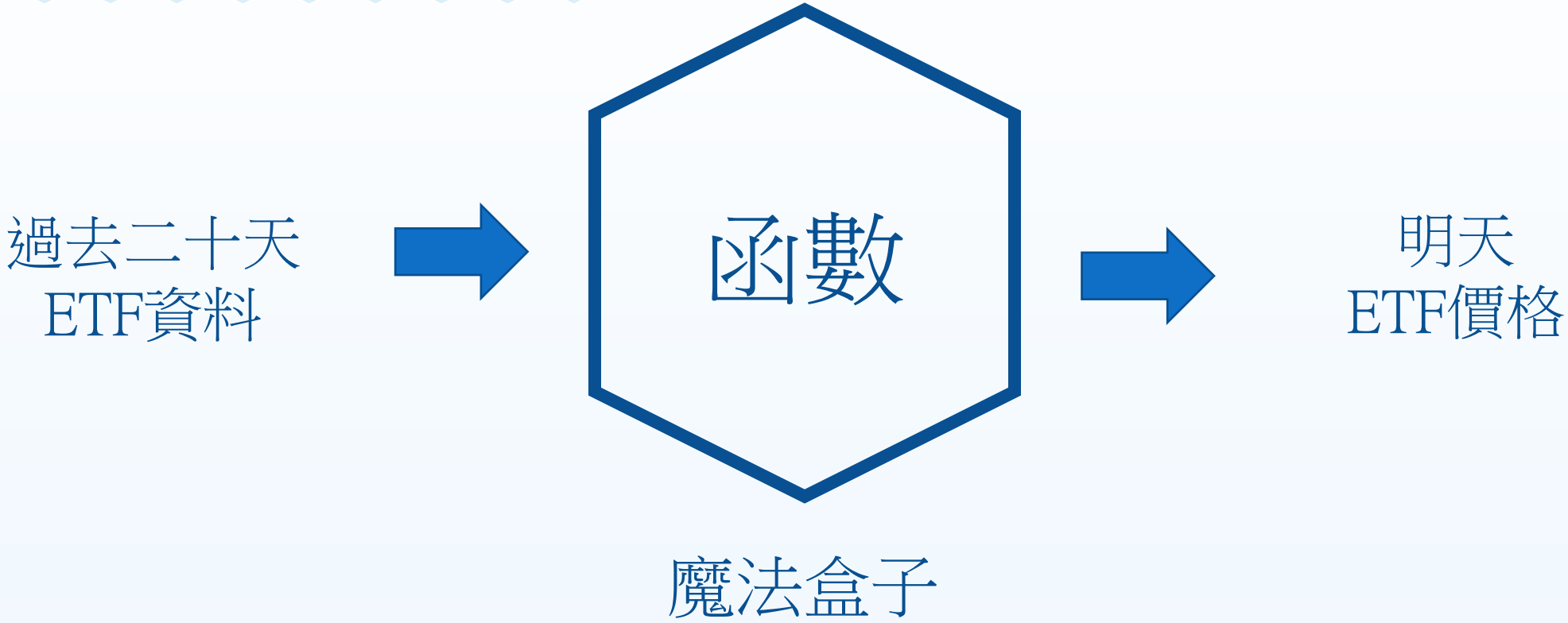


讓你想到國中數學了嗎？

沒關係我們舉個例子

函數是個魔法盒子

Funtion is an Magic Box



恭喜你！
你完成了一次數學建模了！

而且如果模型預測的很好

槓桿開最大催下去，你就發了

問題是

Question is

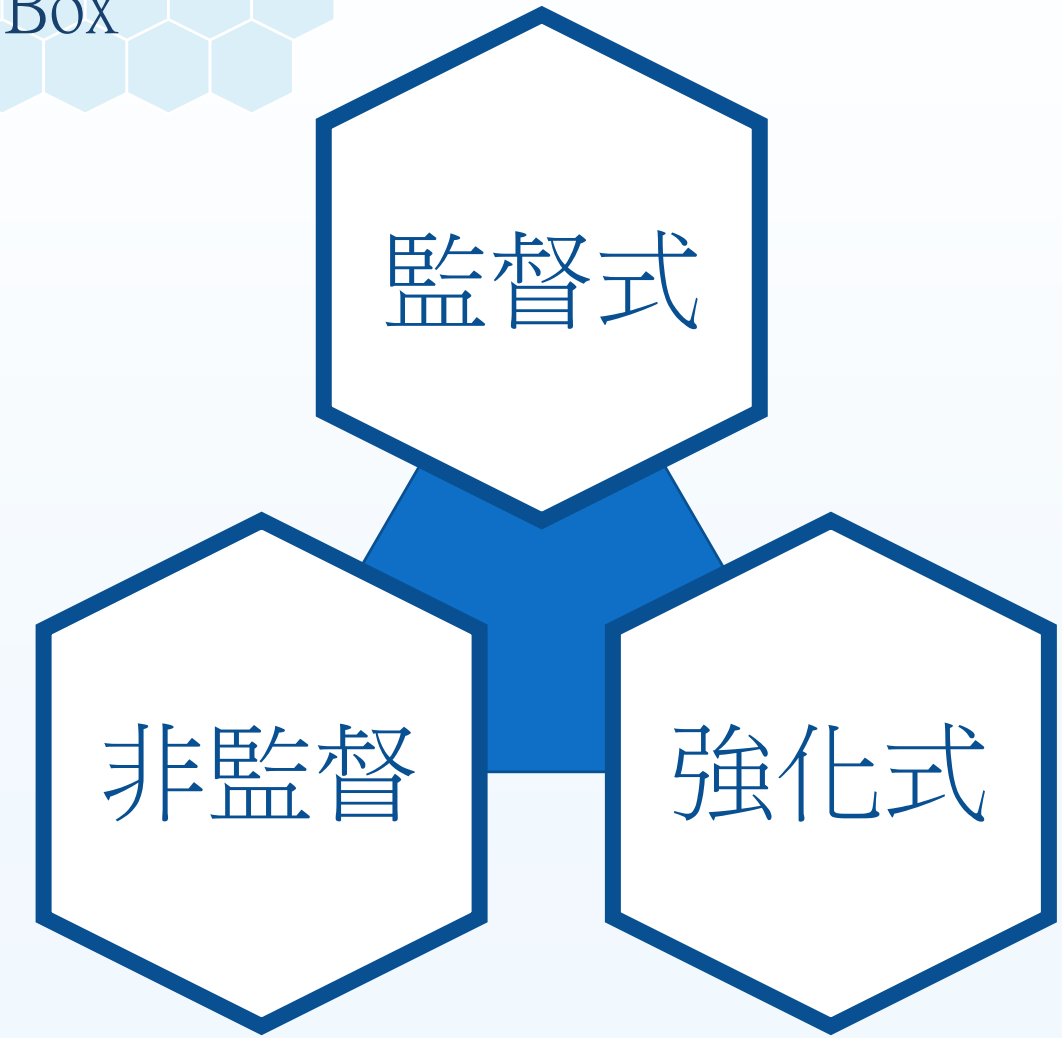
要怎麼讓魔法盒子預測？

要怎麼改善魔法盒子的預測？

我們要如何評估魔法盒子準不準？

怎麼預測

Funtion is an Magic Box



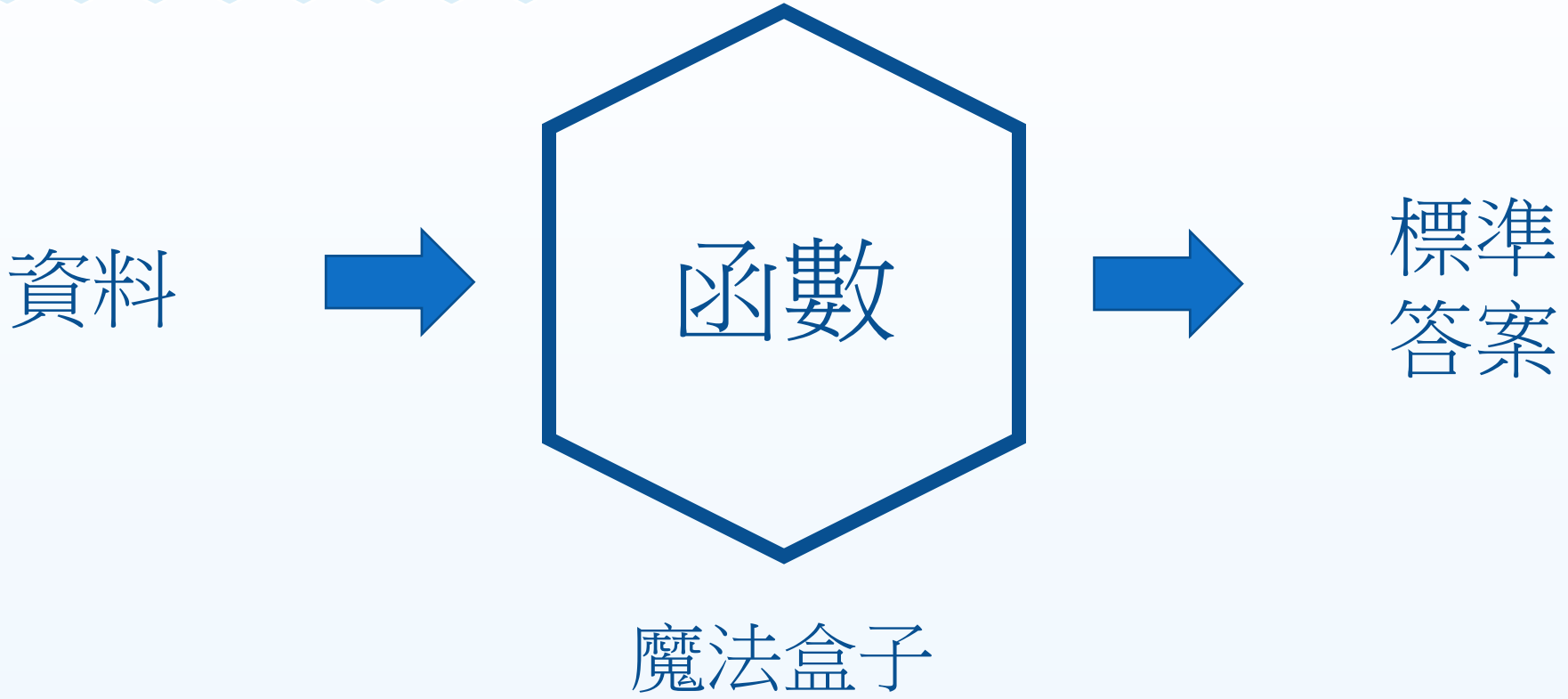
監督式學習

Supervised Learning

有正確答案的學習

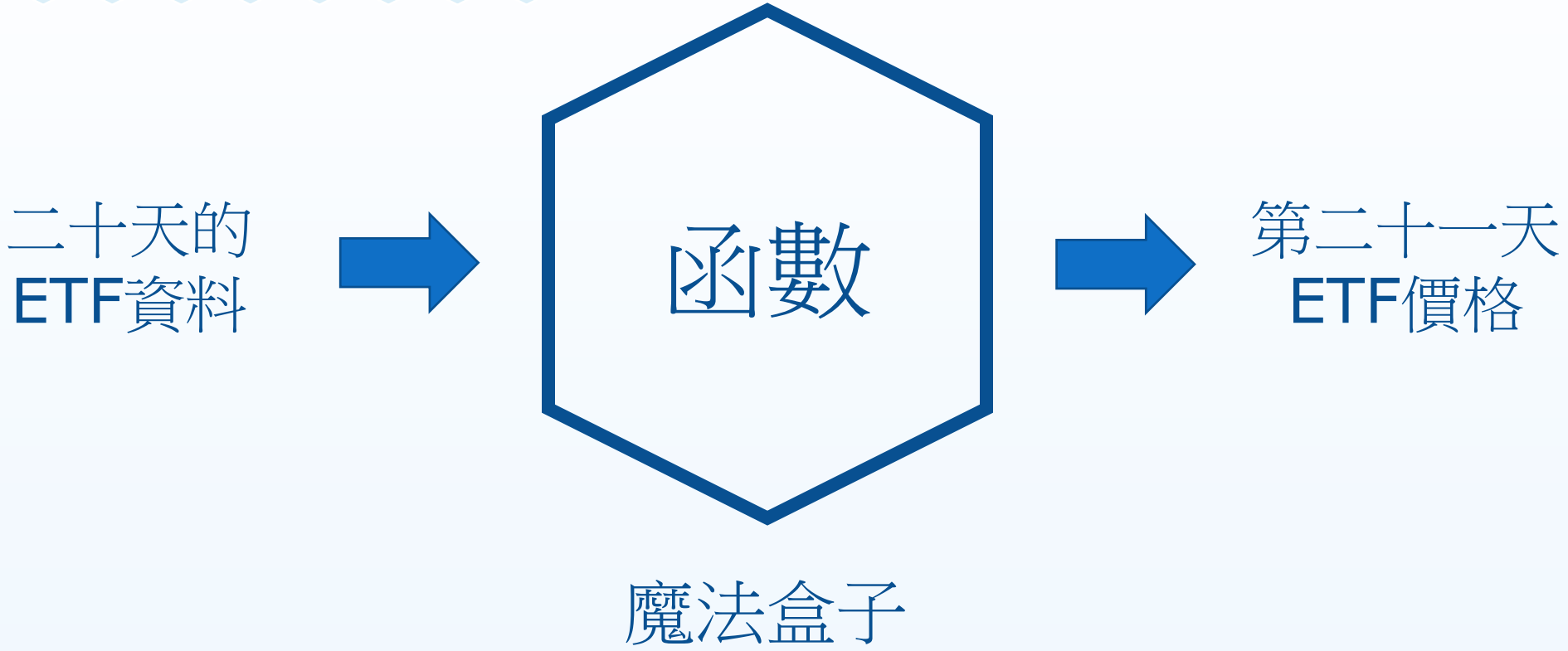
監督式學習

Supervised Learning



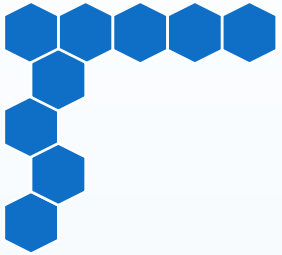
監督式學習

Supervised Learning



非監督式學習

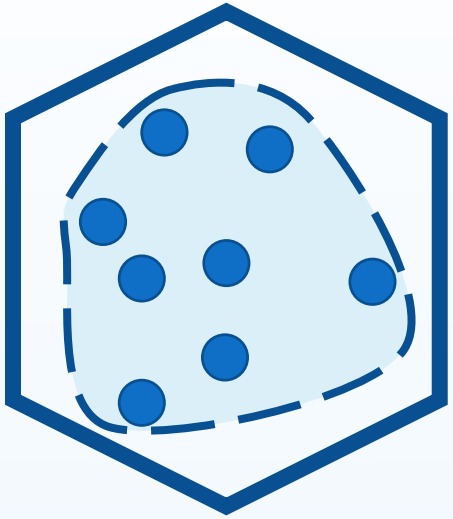
Unsupervised Learning



沒有正確答案的學習

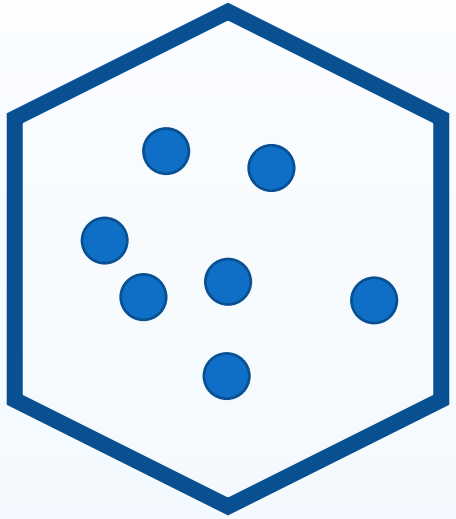
非監督式學習

Unsupervised Learning



生成

Generative



特徵抽取

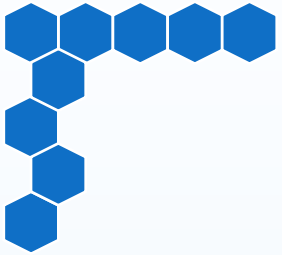
Feature Extraction

分群

Clustering

強化式學習

Reinforcement Learning



靠棒子跟蘿蔔的學習

楊立昆

Yann LeCun

深度學習三巨頭之一
卷積神經網路的發明者
Facebook AI 研究院院長

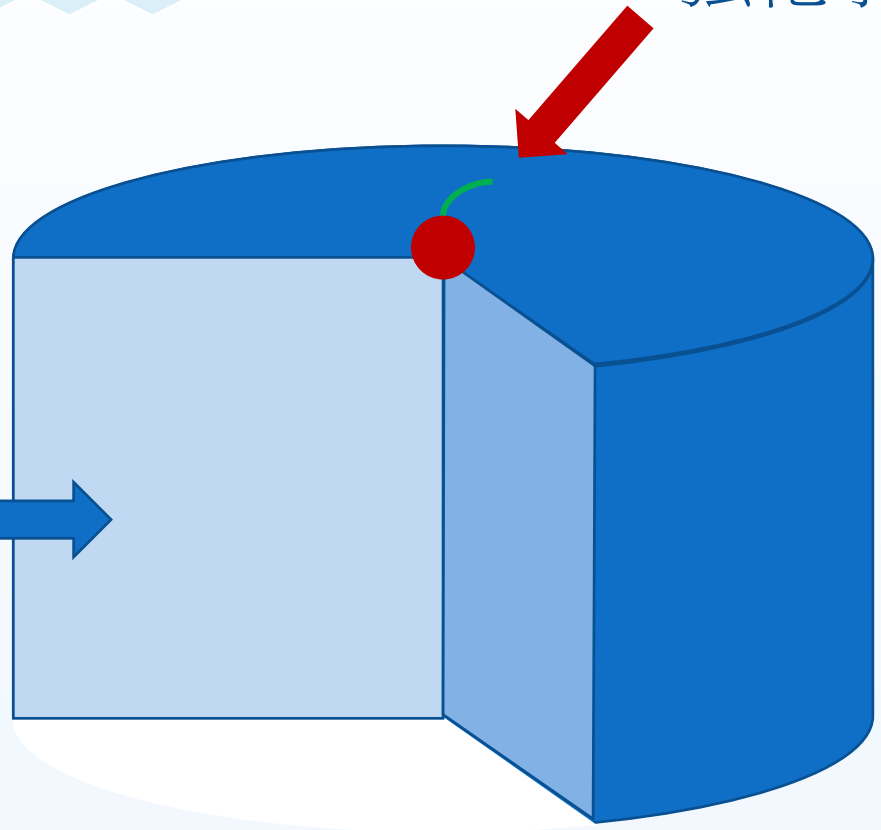
楊立昆對人工智慧提出一個比喻

楊立昆的蛋糕比喻

Yann LeCun’s Cake

強化學習

重要卻不可或缺的
最後一里路



非監督式學習

真正的核心
讓電腦擁有「常識」
也就是用少部分的資料
去猜想整個世界

監督式學習

表層的糖衣
訓練電腦符合
人類設定的答案

怎麼預測

How to Predict

丟資料進模型想辦法訓練他

怎麼預測

How to Predict

要怎的對資料下手呢？

資料


訓練資料
測試資料

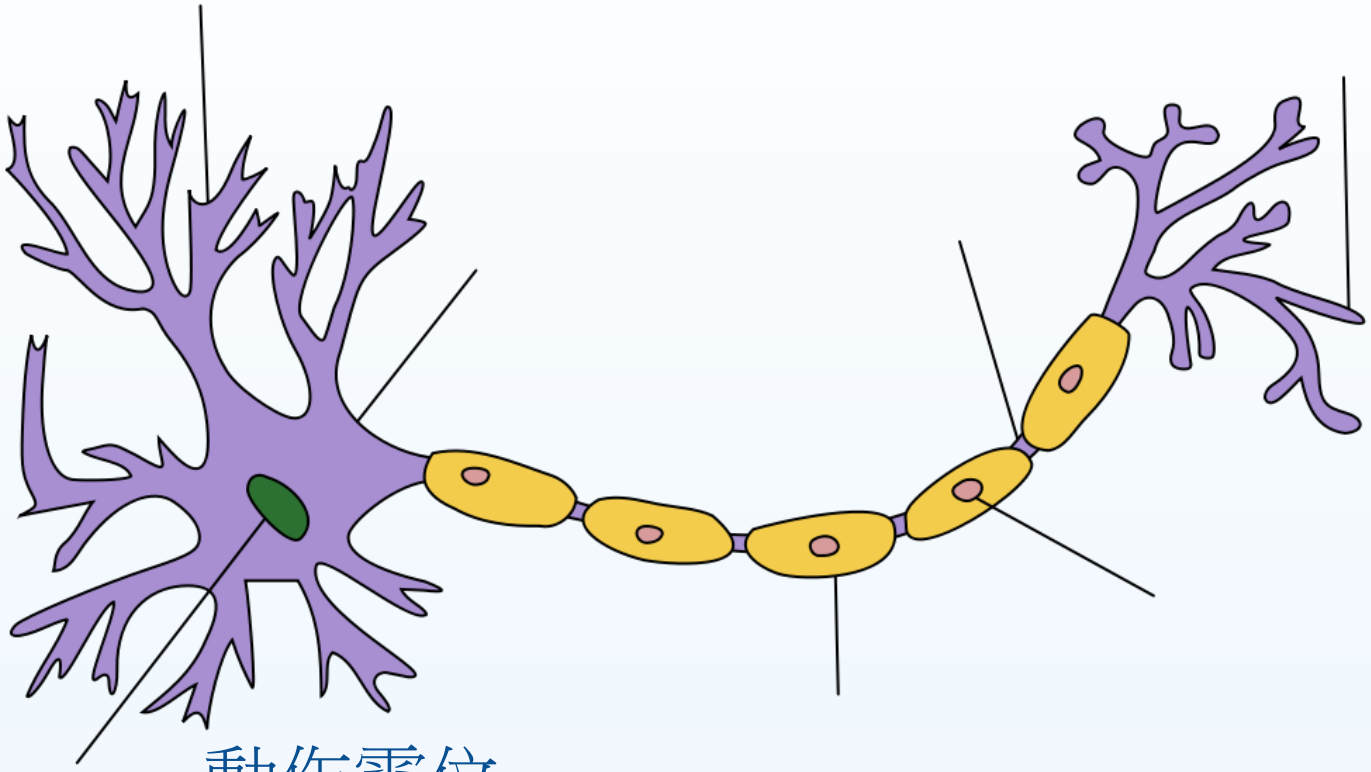

訓練資料
檢驗資料
測試資料

什麼是深度學習?

What is Deep Learning?

向大自然學習

Learn from Nature



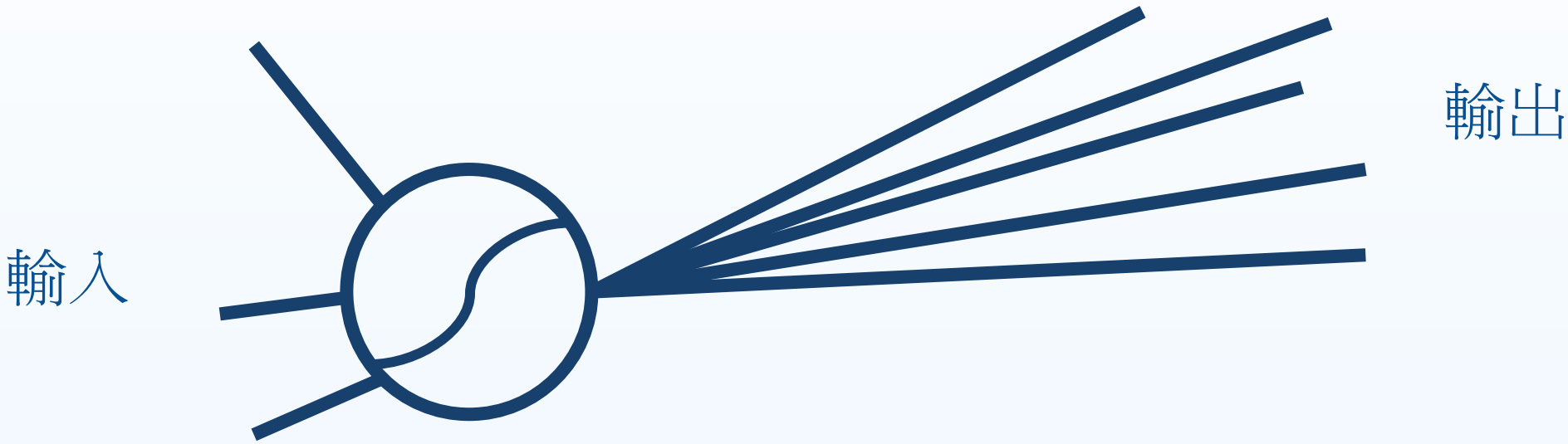
樹突
(輸入)

軸突
(輸出)

動作電位
(修改訊號)

抽像化神經元

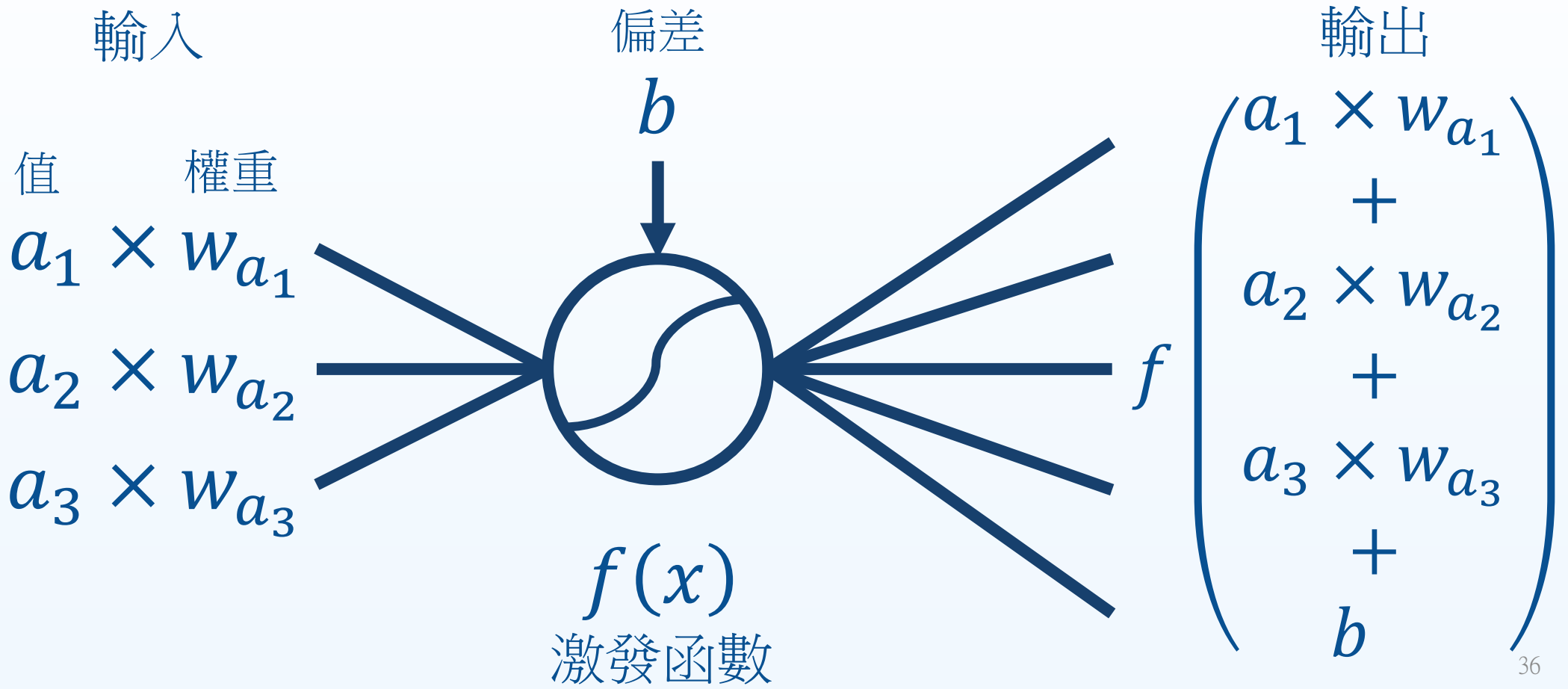
Abstract Neuron



偏差與激發函數
(修改訊號)

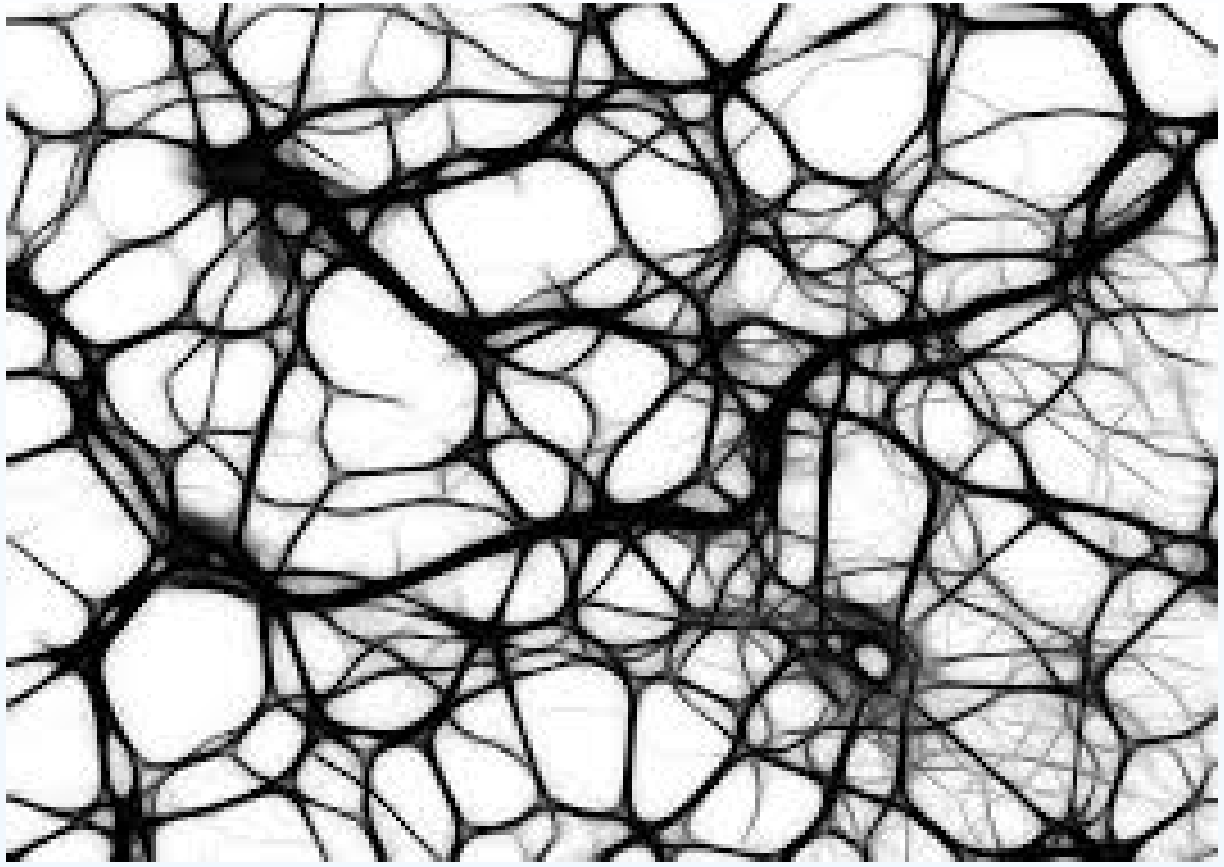
抽像化神經元

Abstract Neuron



全連接神經網路

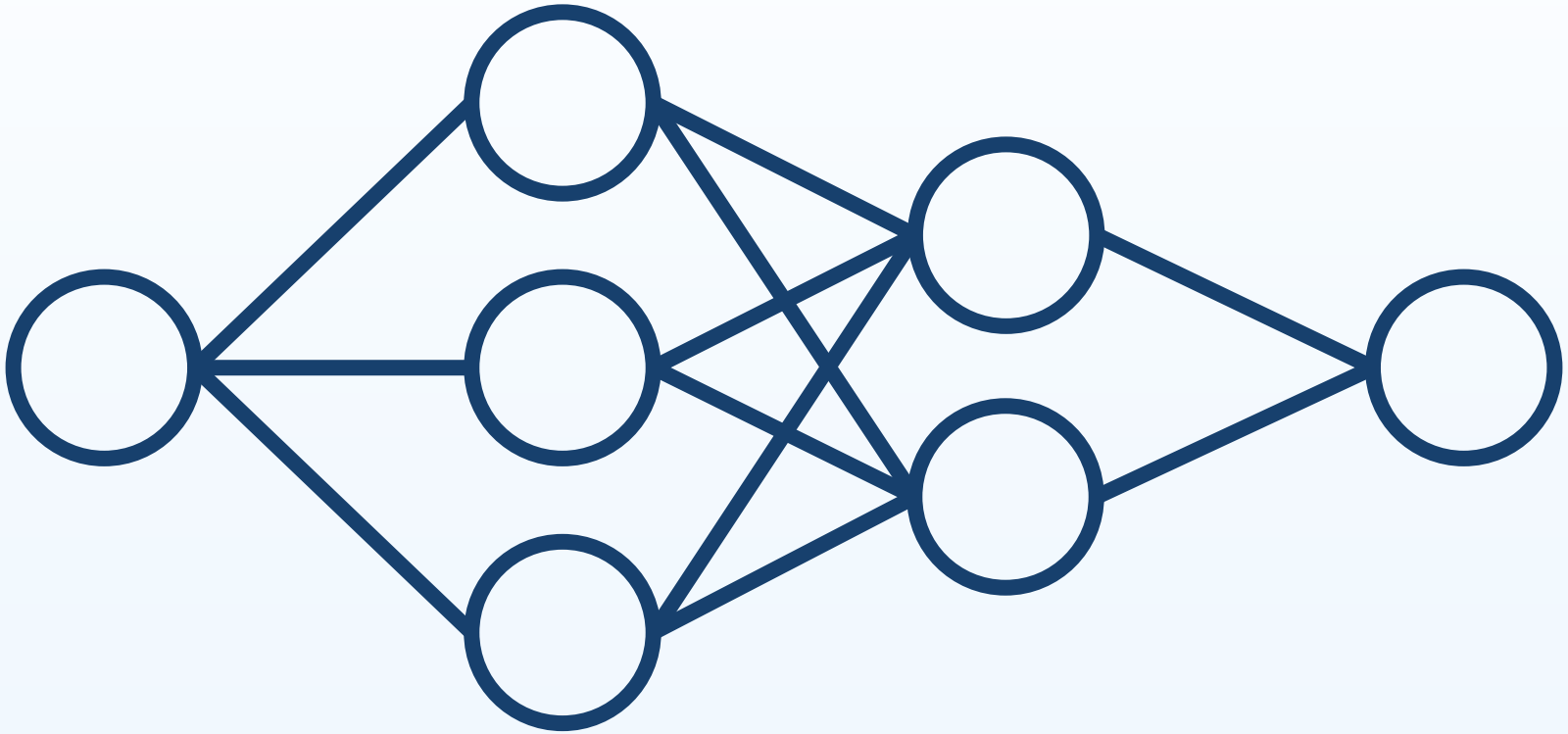
Fully-Connected Neural Network



全連接神經網路

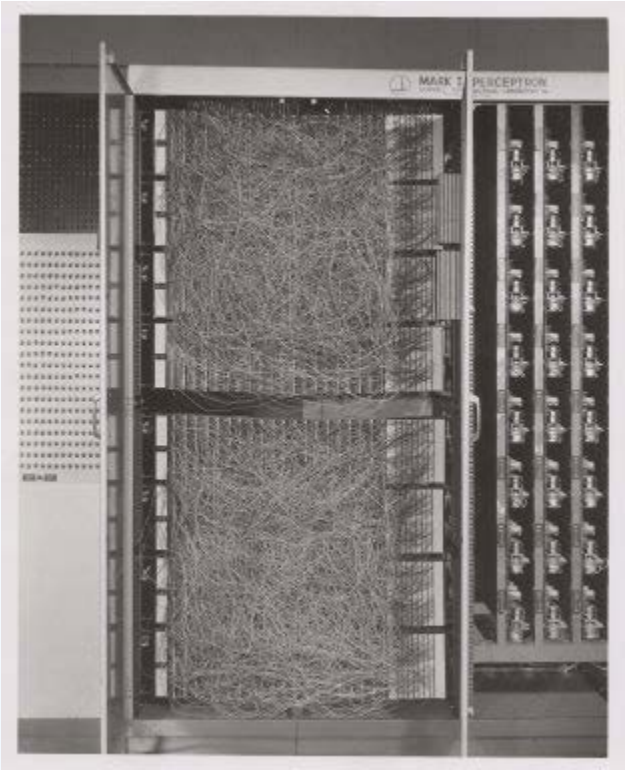
Fully-Connected Neural Network

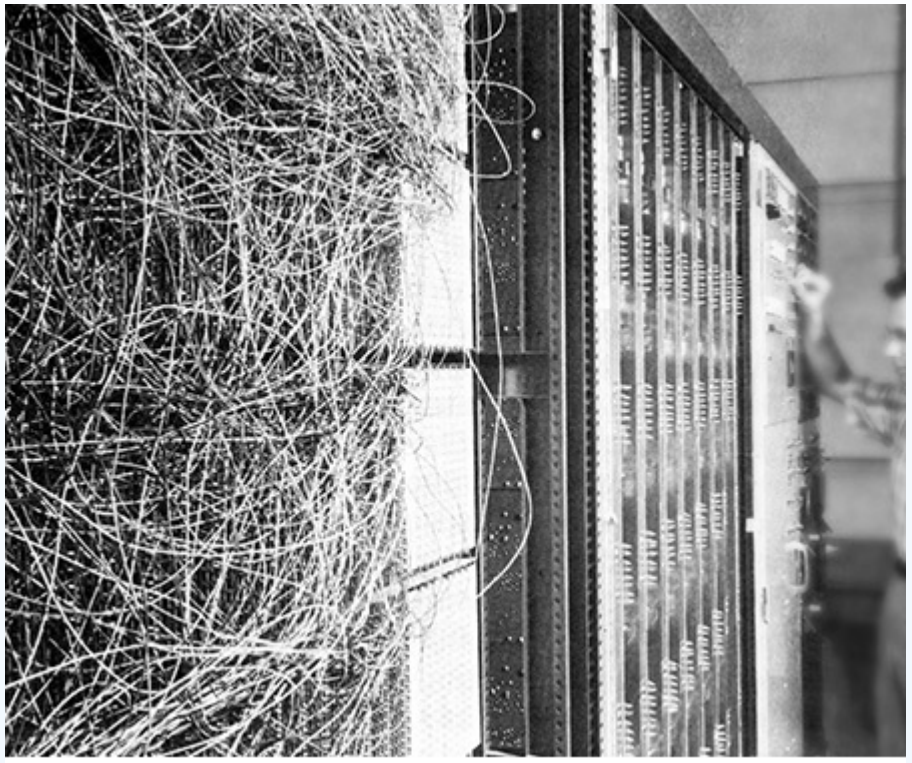
輸入層　隱藏層　隱藏層　輸出層



1957就有人做了

Cornell University, 1957





感知器

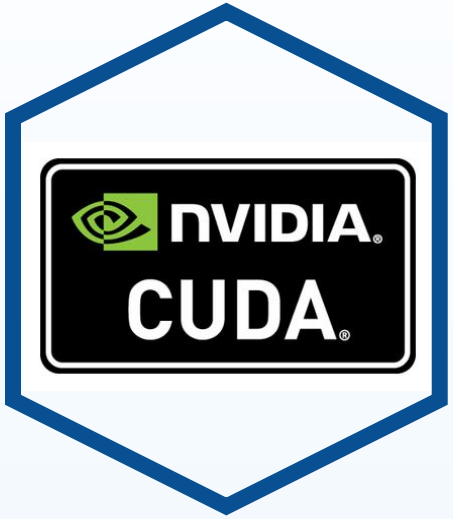
Perceptron

Frank Rosenblatt

等等1959就有人做了
那現在為什麼那麼紅？

為什麼紅？

Why Hot?

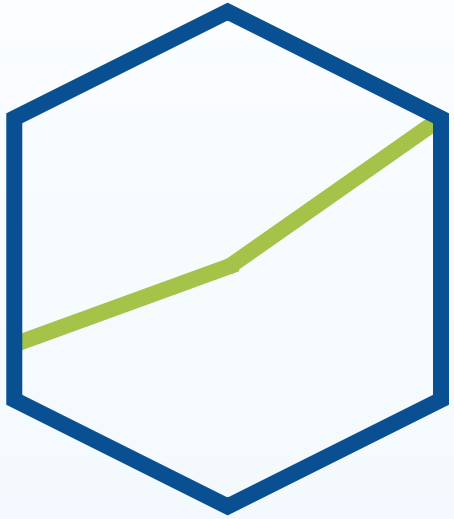


電腦變快

Faster Computer

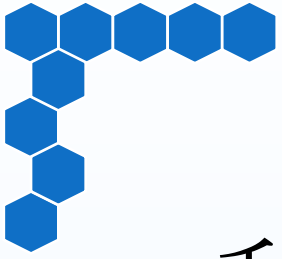
大數據

Big Data

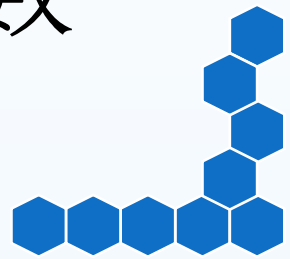


演算法改善

Improved Algorithm



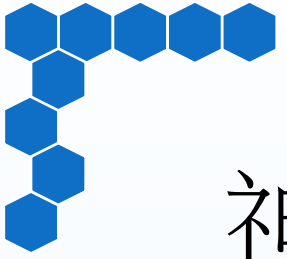
我們其實可以近似任意的函數



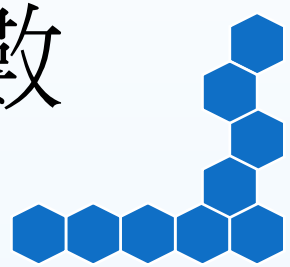
通用近似定理

Universal approximation theorem

只要一層足夠多神經元的隱藏層就能逼近
任何在歐氏空間上緊緻子集的連續函數



神經網路是用很多很多小函數來
逼近一個我們希望得到的大函數



激發函數

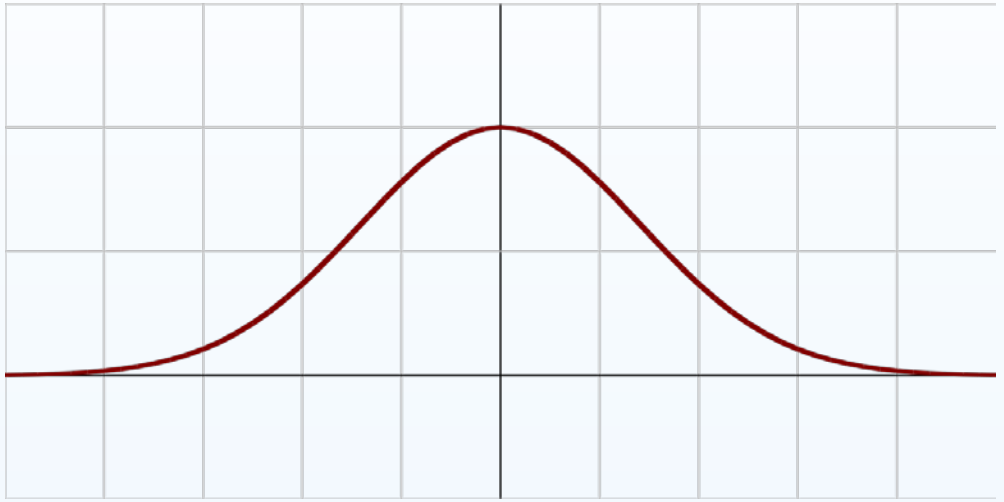
Activate Function

那我們要用哪種小函數呢？

高斯分佈函數

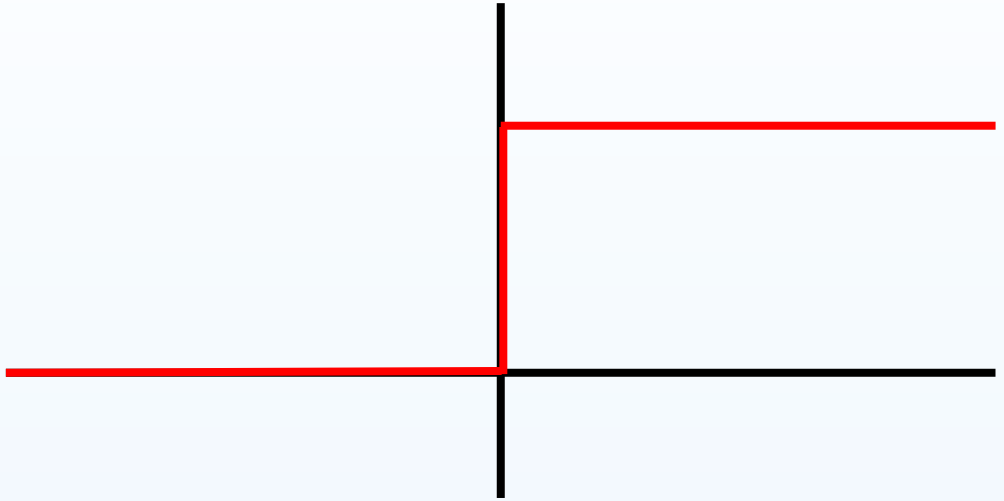
Activate Function

$$f(x) = e^{-x^2}$$



Binary step

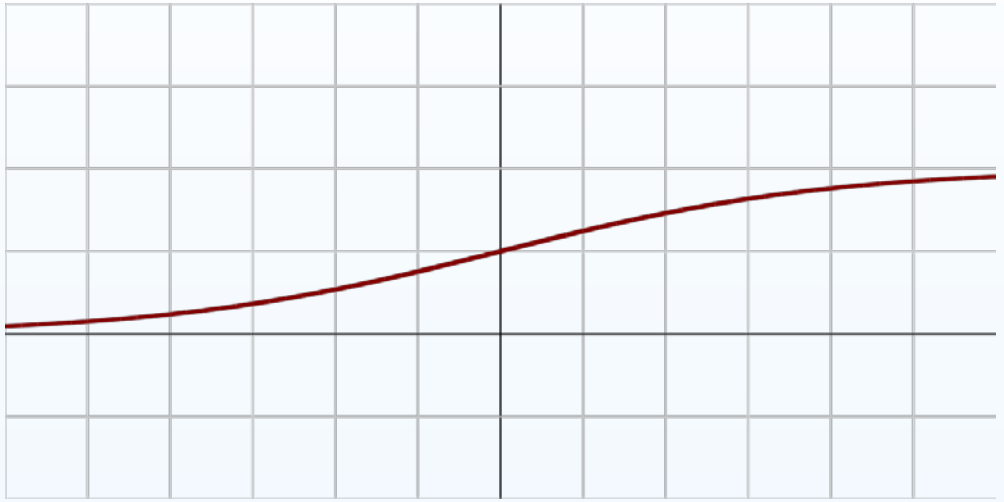
$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$



S型函數

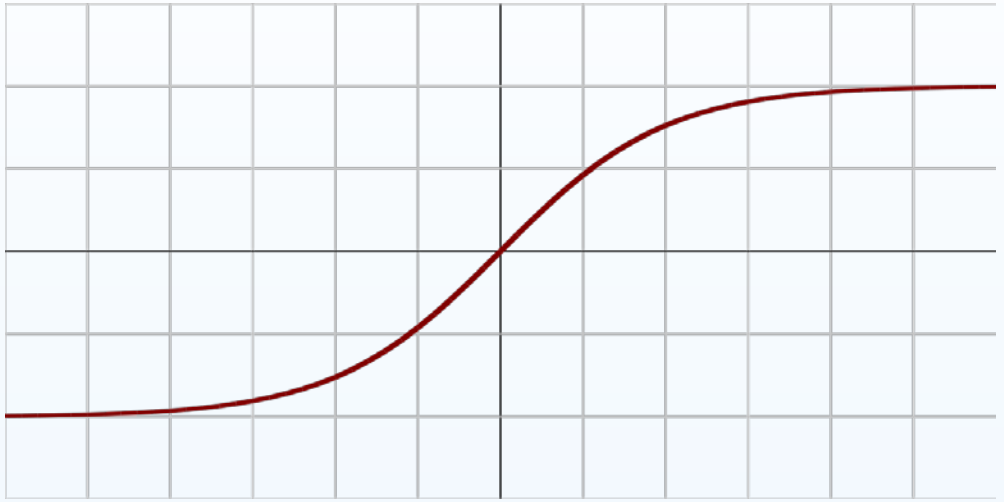
Sigmoid

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



Hyperbolic tangent

$$f(x) = \tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

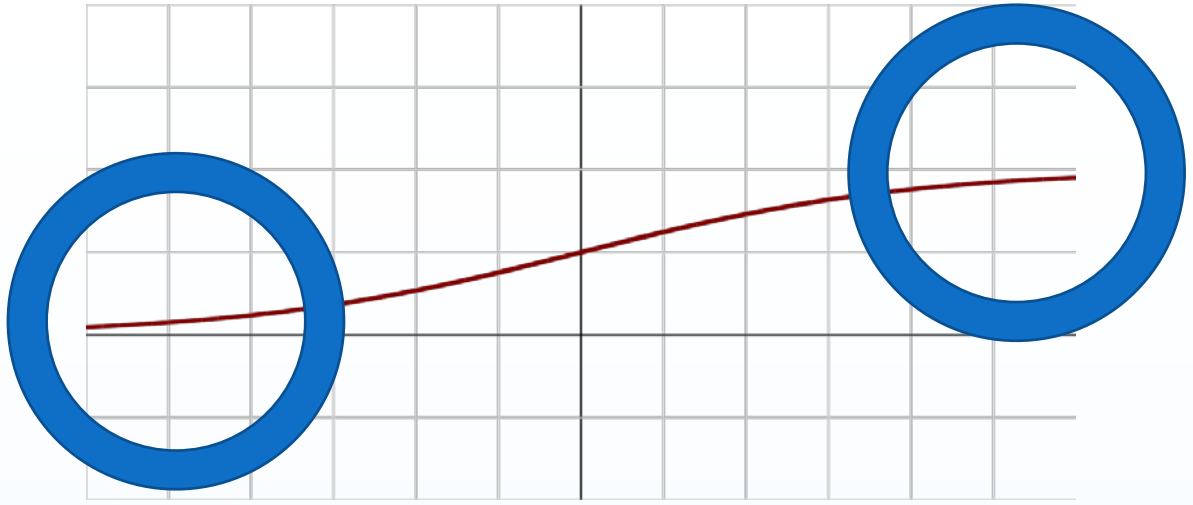


但這些都有個問題

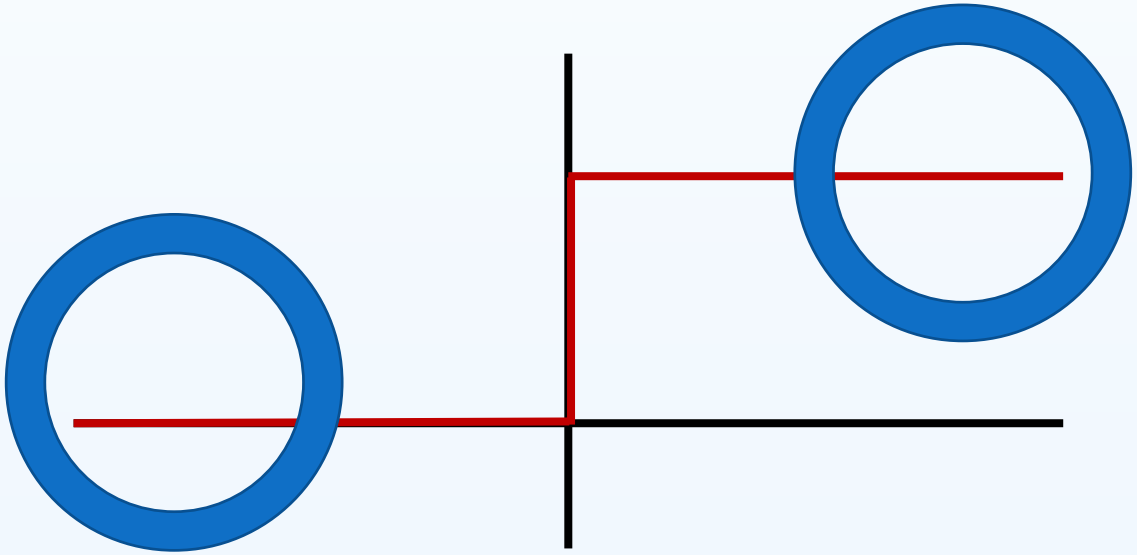
But, here is a problem

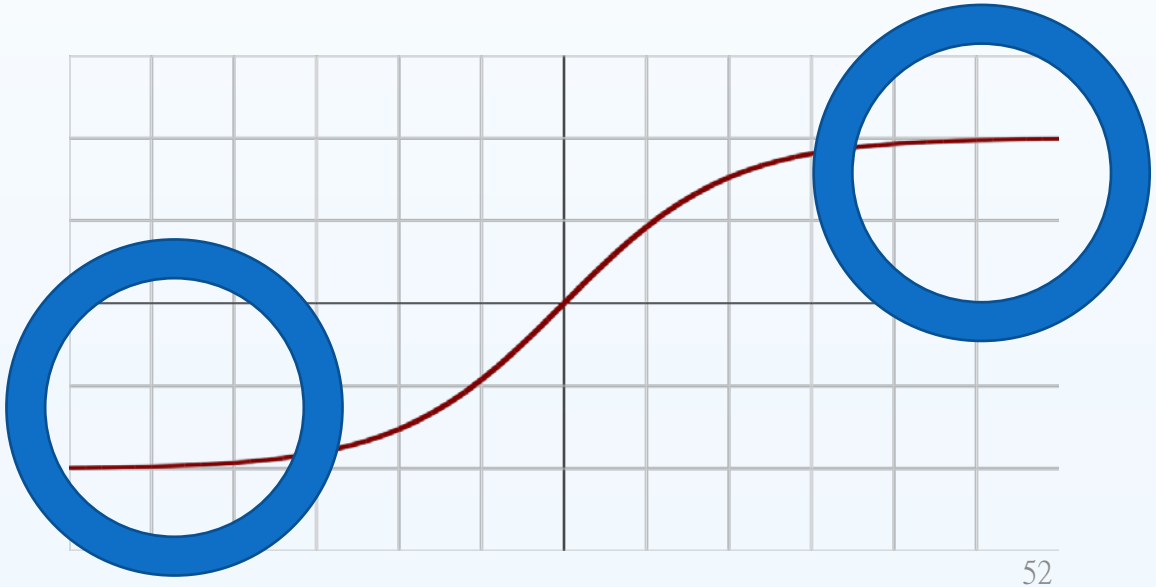
我們在訓練模型的時候是利用微分取梯度的

(後面會詳述)



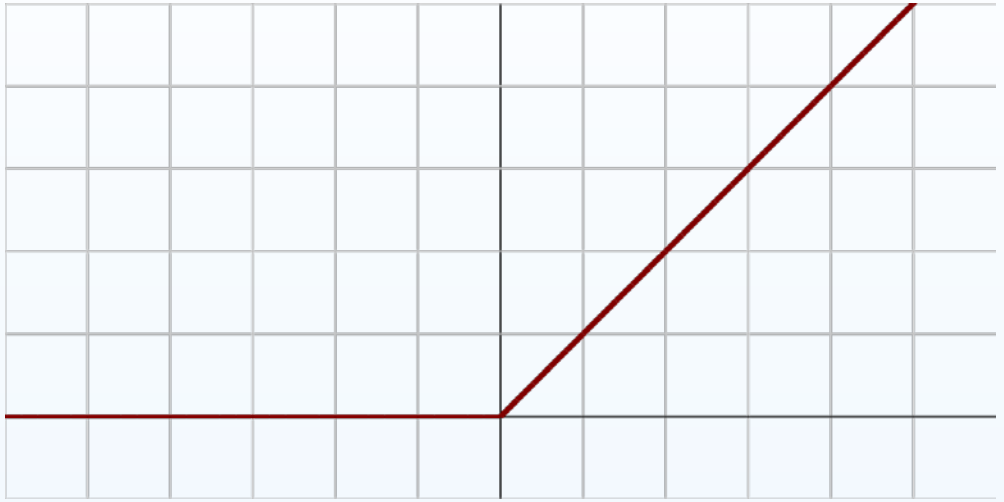
這些函數在兩端都太平了(斜率低)，梯度會消失





ReLu

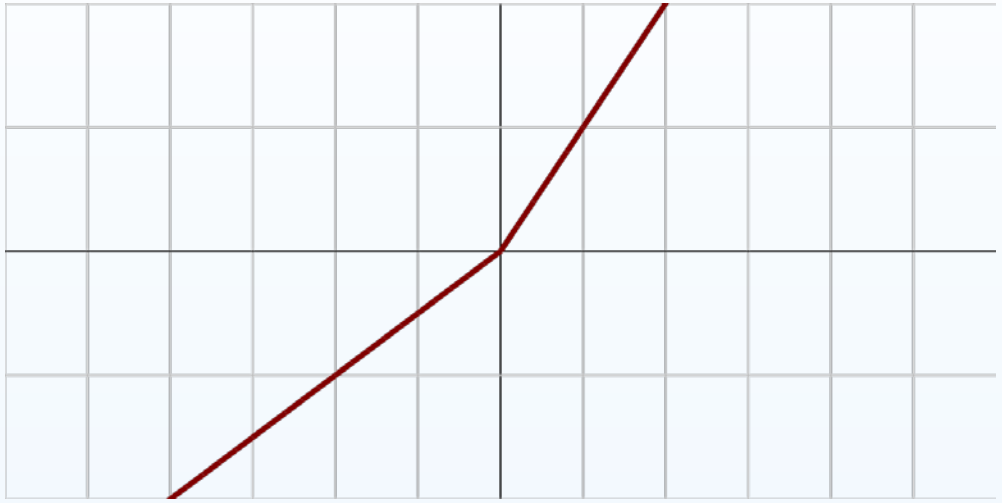
$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$



但是這樣會$x < 0$導致神經元無法重啓
解決方法是加上一個$\alpha > 0$的常數

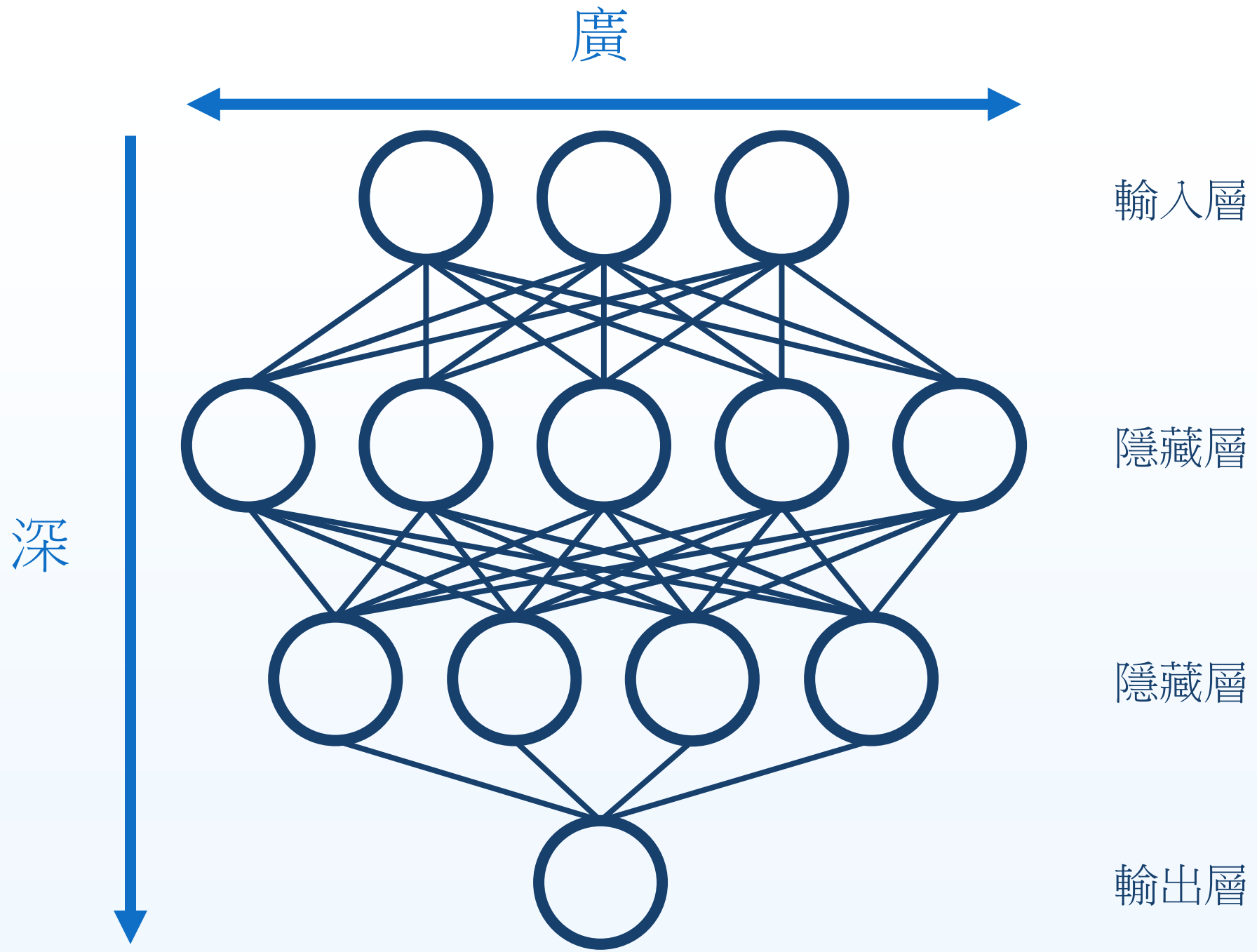
Leaky ReLu

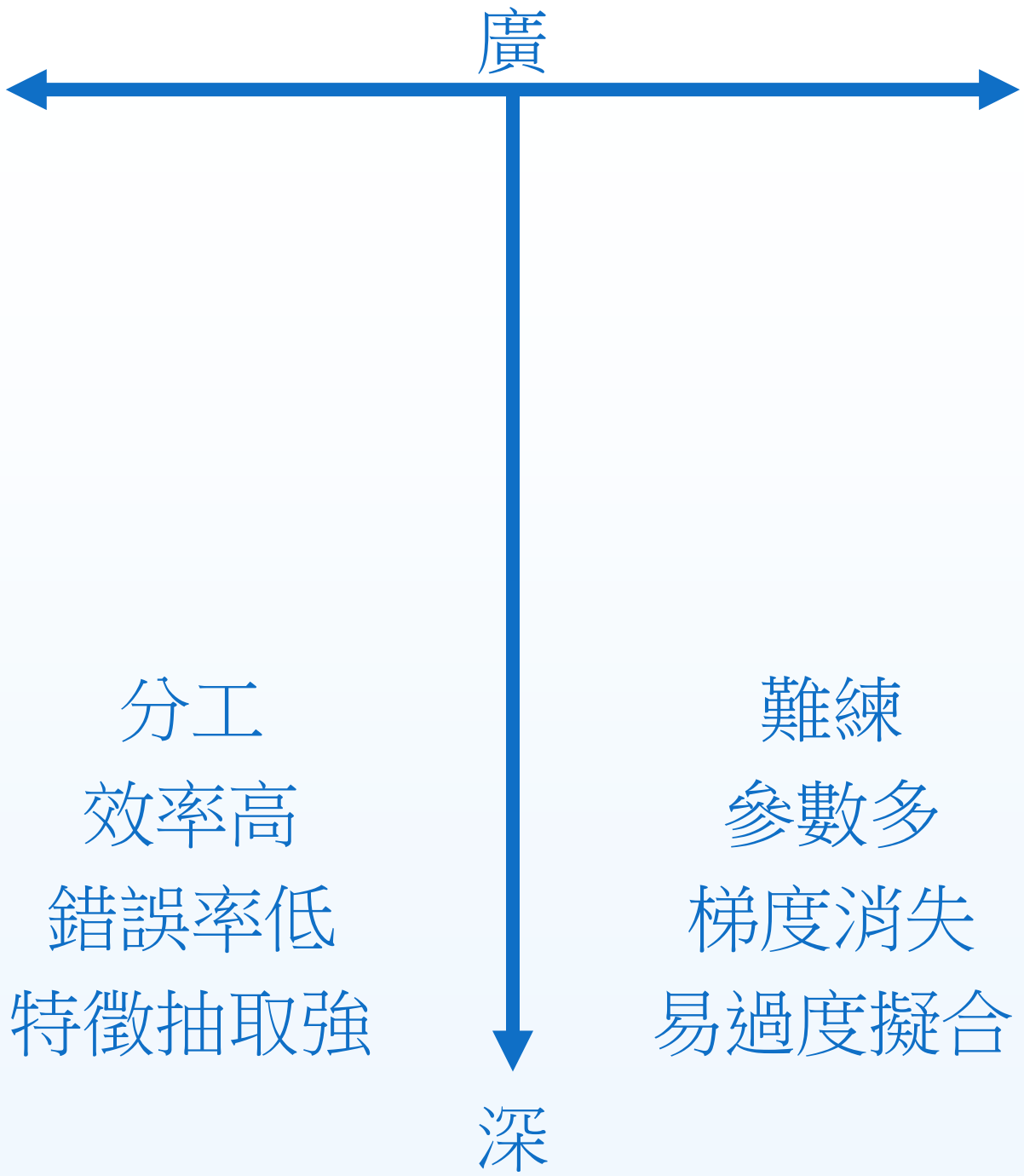
$$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$



深還是廣？

Be Deep or Be Wide


廣
深
輸入層
隱藏層
隱藏層
輸出層


廣
優
缺
分工
效率高
錯誤率低
特徵抽取強
難練
參數多
梯度消失
易過度擬合
深

怎麼評估准不準?

How to Score it

我們其實只有下面兩樣東西

正確答案　　模型答案

損失函數

Loss Function

能夠表達
正確答案和模型答案
差異的函數

平均絕對差

Mean Absolute Error

$$\sum \frac{|y_{data} - y_{predit}|}{n}$$

y_{data} = 正確答案　　y_{predit} = 模型答案

平均平方差

Mean Square Error

$$\sum \frac{(y_{data} - y_{predit})^2}{n}$$

y_{data} = 正確答案　　y_{predit} = 模型答案

更多的損失函數

More Loss Function

MAPE

MALE

Hinge

LogCosh

Categorical Crossentropy

Binary Crossentropy

Total Variation

KL Divergence

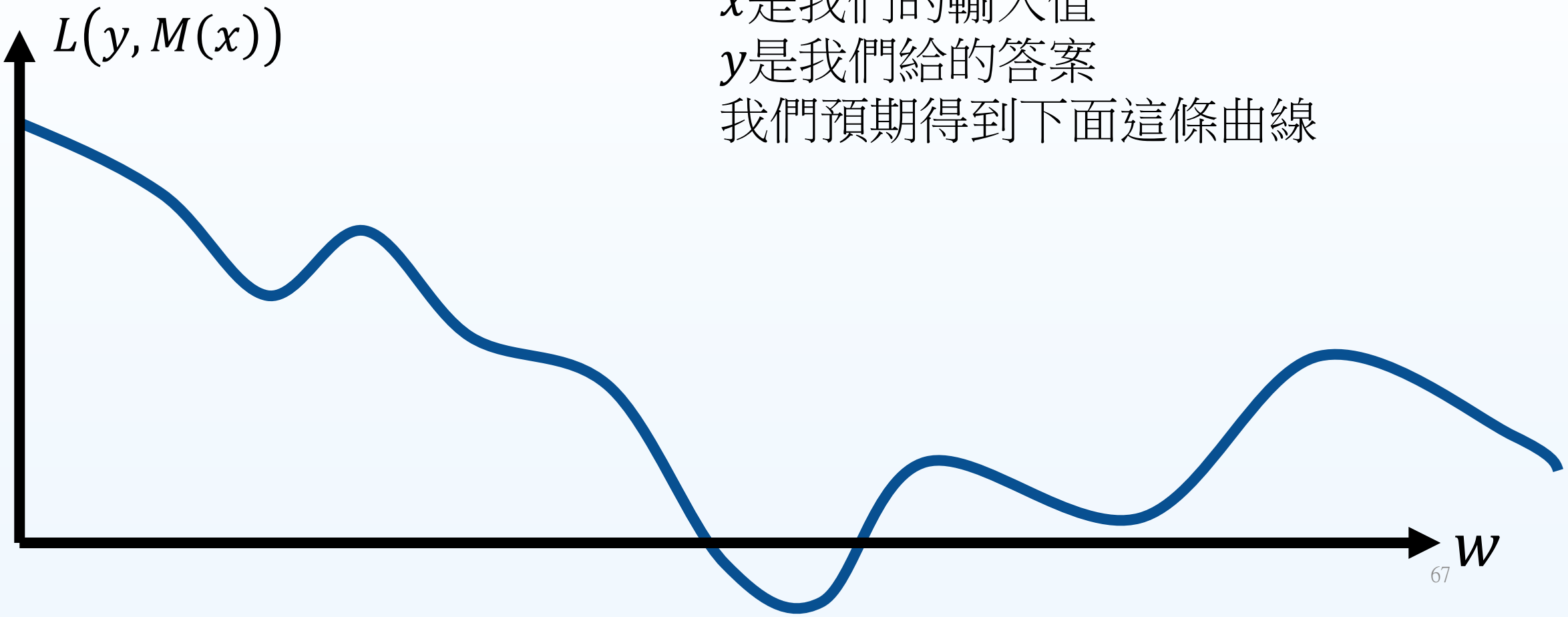
怎麼訓練我的神經網路?

How to Train our Neural Networks

梯度下降法

Gradient Descent

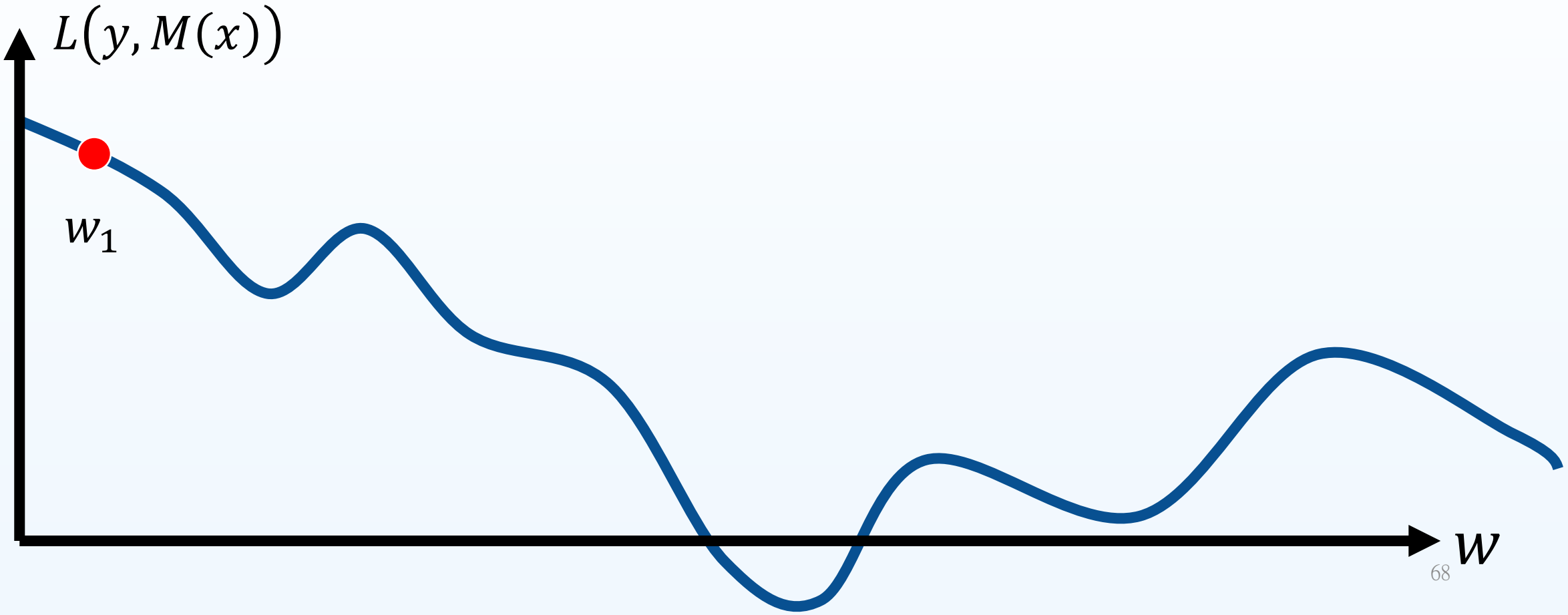
假設我們有個
模型：$M(x)$
模型權重： w
損失函數：$L(y, M(x))$
x是我們的輸入值
y是我們給的答案
我們預期得到下面這條曲線



梯度下降法

Gradient Descent

在w_1這點對$L(y, M(x))$微分



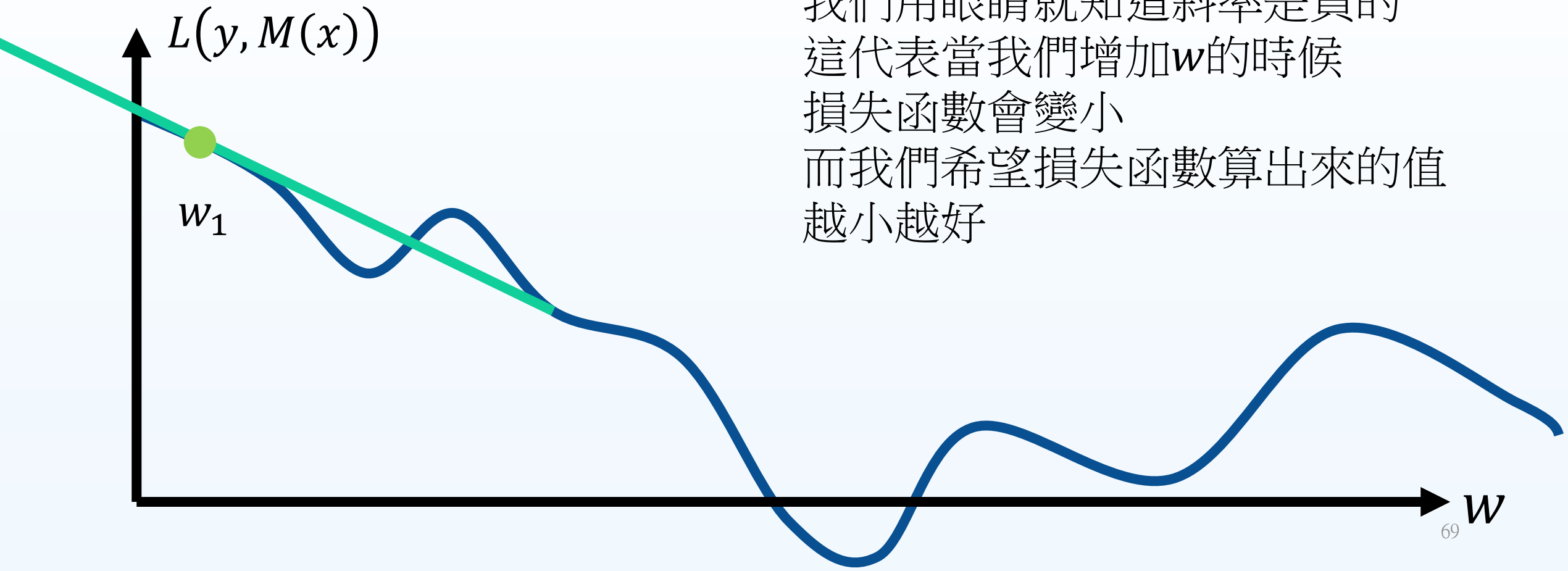
梯度下降法

Gradient Descent

我們可以得到一個斜率

$$\frac{dL}{dw_1}$$

根據國中數學
我們用眼睛就知道斜率是負的
這代表當我們增加w的時候
損失函數會變小
而我們希望損失函數算出來的值
越小越好



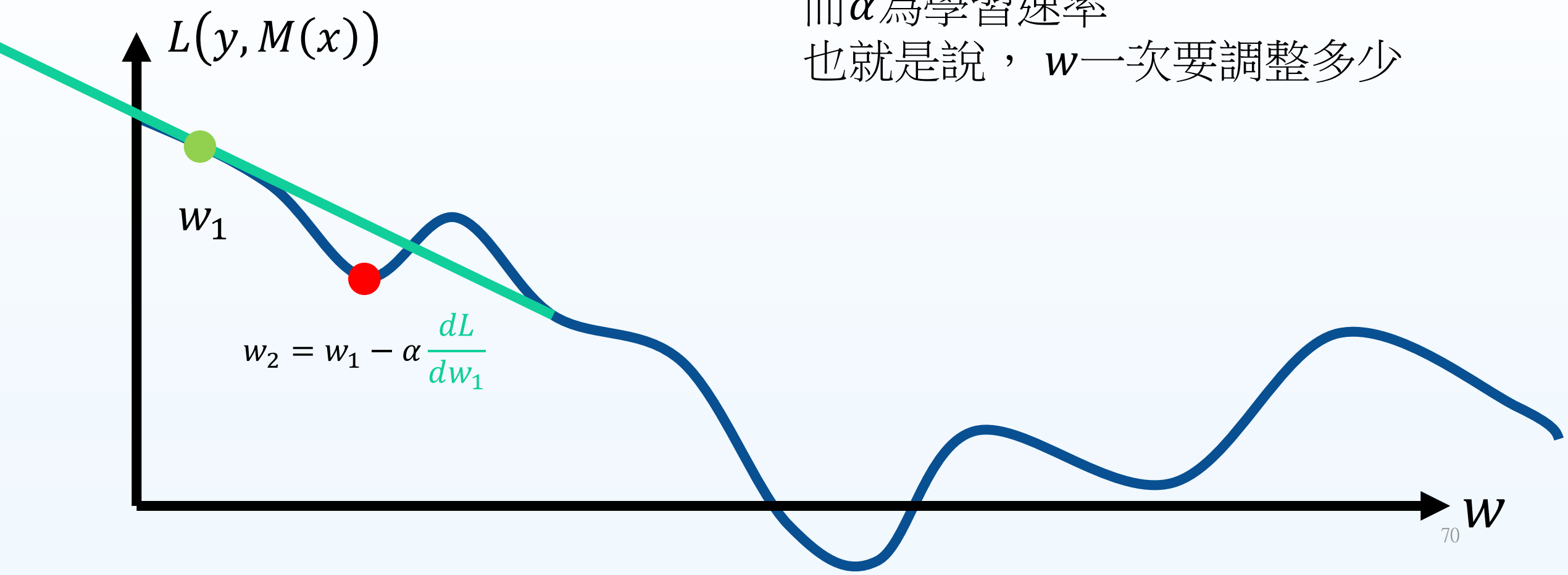
梯度下降法

Gradient Descent

所以我們新的w ，
也就是w_2就會等於

$$w_2 = w_1 - \alpha \frac{\boldsymbol{dL}}{\boldsymbol{dw_1}}$$

而α為學習速率
也就是說， w一次要調整多少



調整每個權重都是這樣
只是把其他權重當作常數微分 (偏微分) 後
得到梯度 (可以視為多維度下的斜率)
實務上會隨機抽一小單位資料改善模型的權重來加速
也就是
隨機梯度下降
Stochastic Gradient Descent
SGD

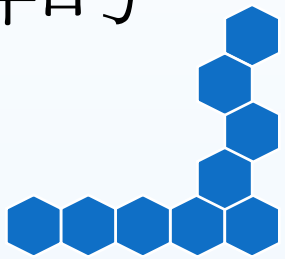
Batch

訓練完整個訓練資料中的一小單位資料

Epoch

訓練完全部的訓練資料

就這樣一步一步總有一天會到最佳解的



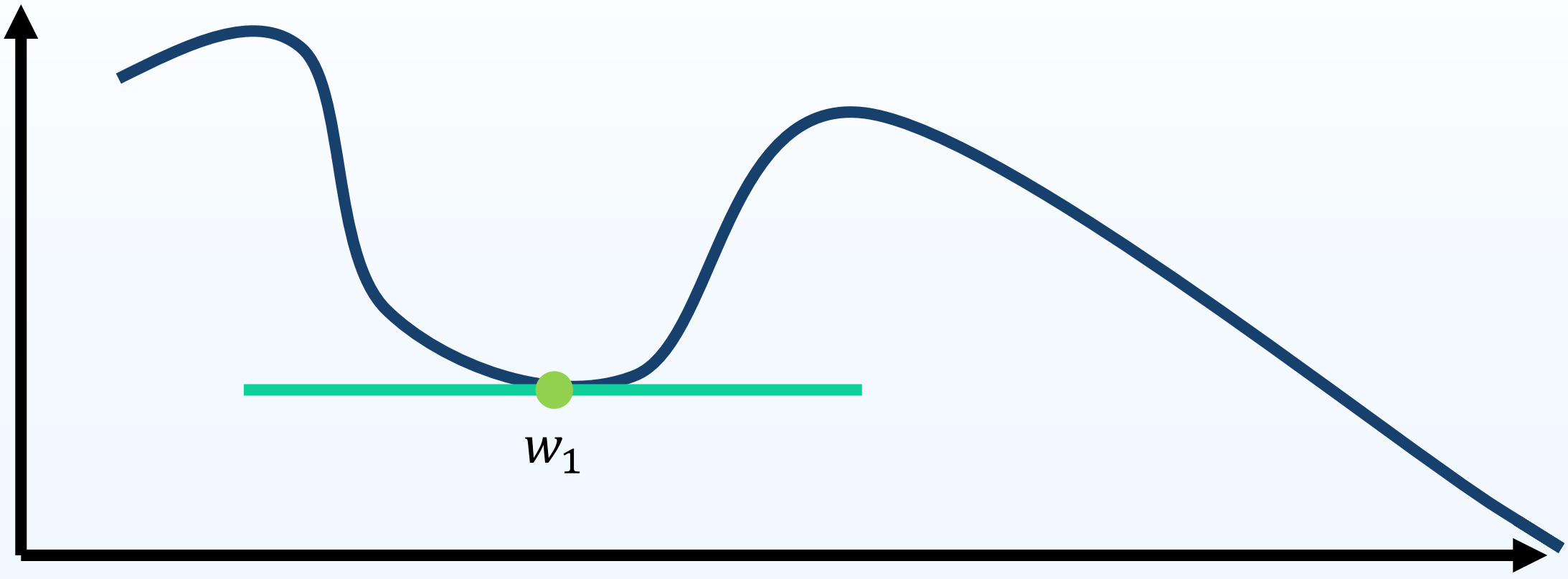
嗎？

動量
Momentum

如果我們今天很不幸的落在低谷
微分後斜率又剛好等於零

$$w_2 = w_1 - \alpha \frac{\boldsymbol{dL}}{\boldsymbol{dw_1}} = w_1 - \alpha \times 0 = w_1$$

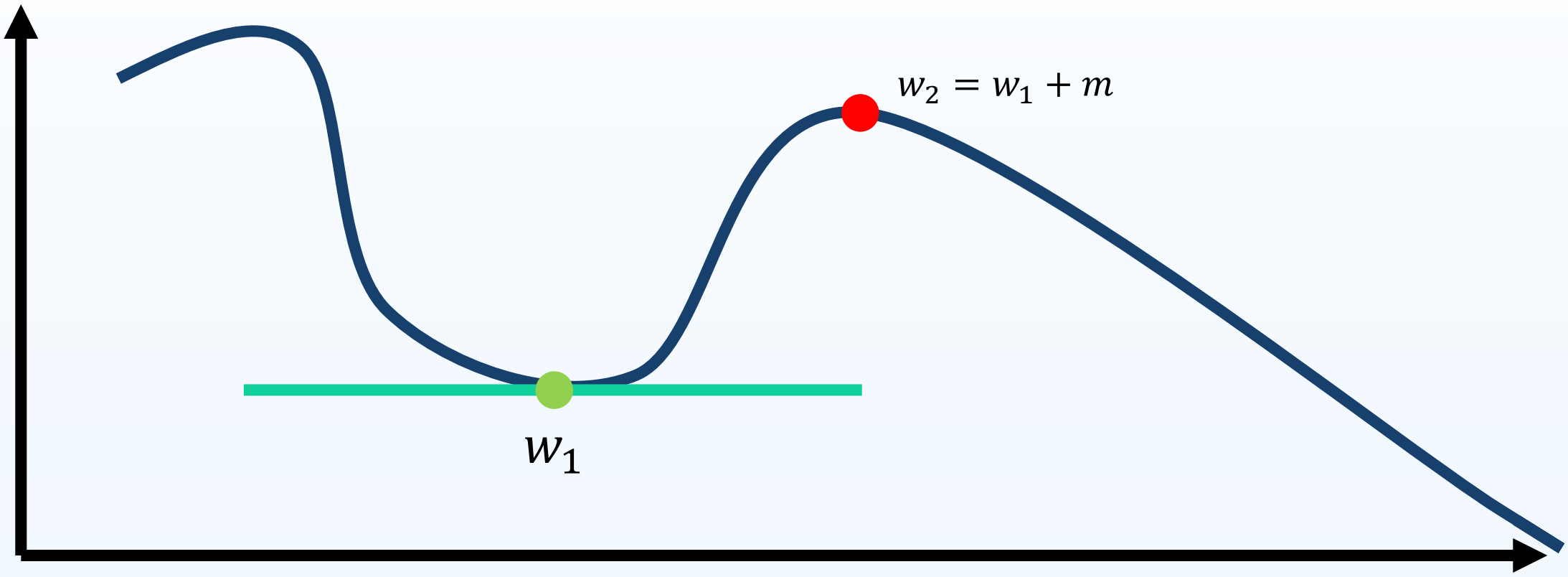
不會改進了，怎麼辦？



動量
Momentum

那就加入一個動量常數m
讓w_2變成

$$w_2 = w_1 + m - \alpha \frac{\boldsymbol{dL}}{\boldsymbol{dw_1}} = w_1 + m$$



更多的優化法

More Optimizers

Adam

Adagrad

Adadelta

Nadam

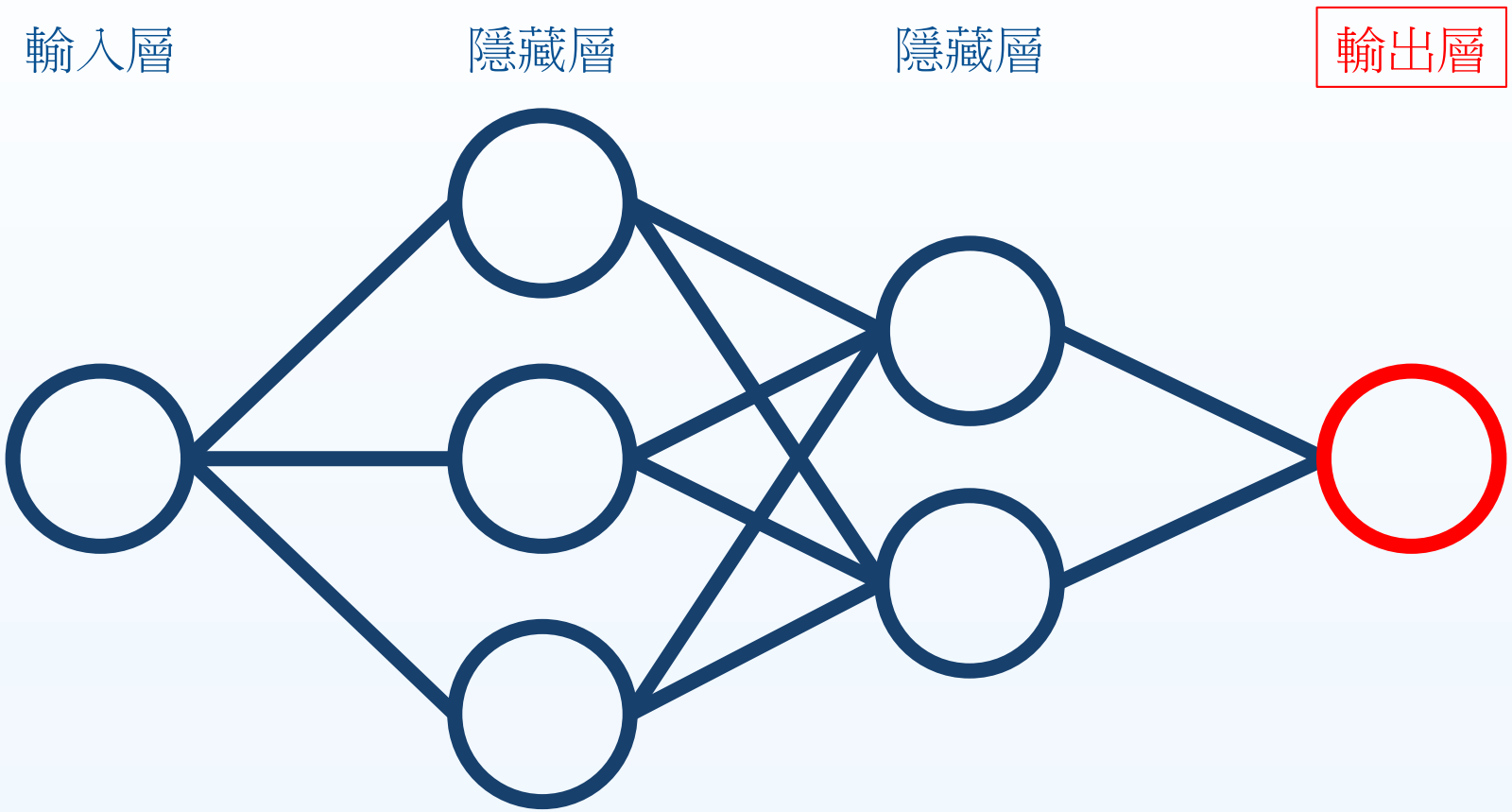
Adamax

RMSprop

反向傳播算法

Back Propagation

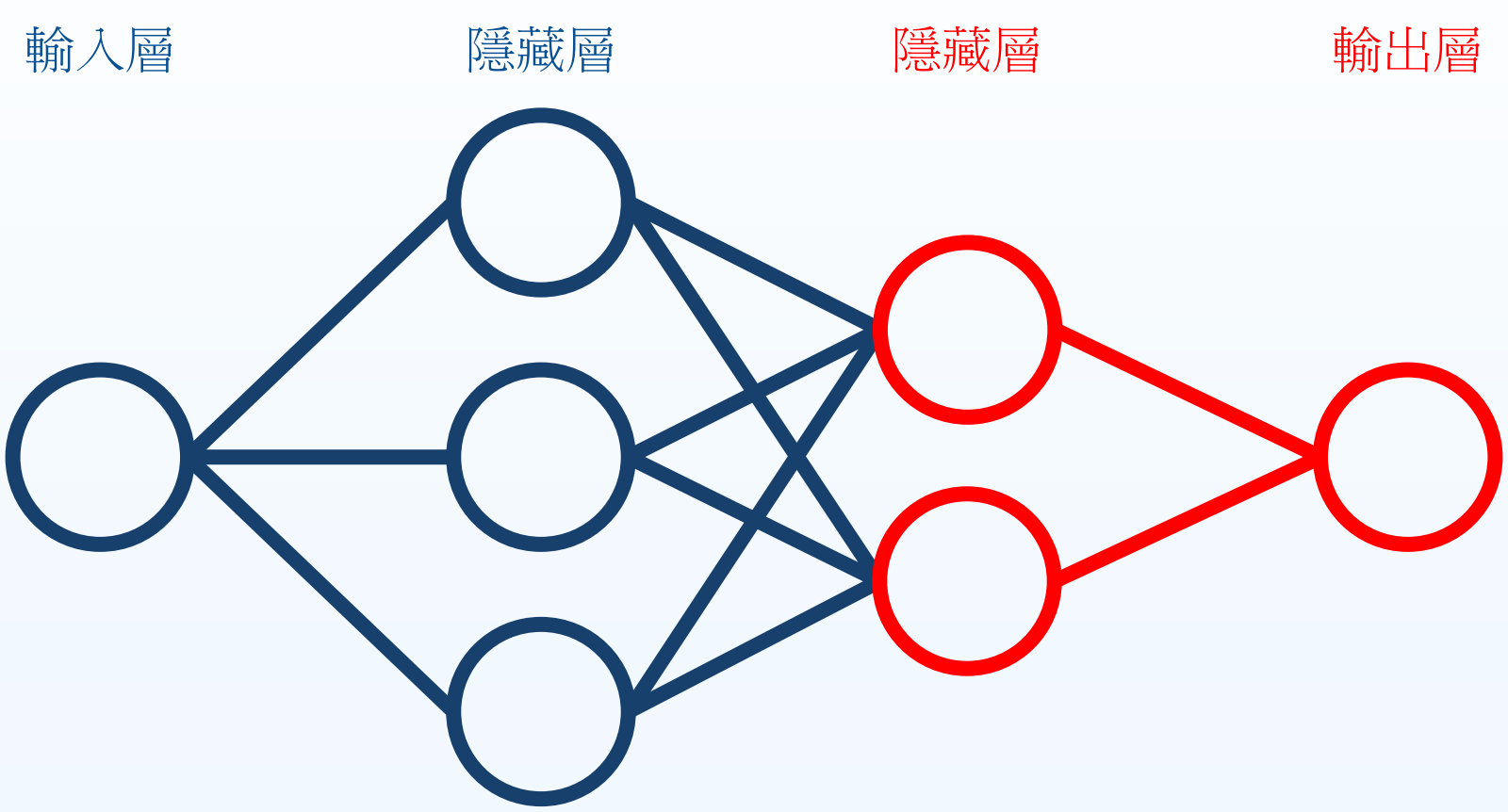
從輸出層向輸入層微分



反向傳播算法

Back Propagation

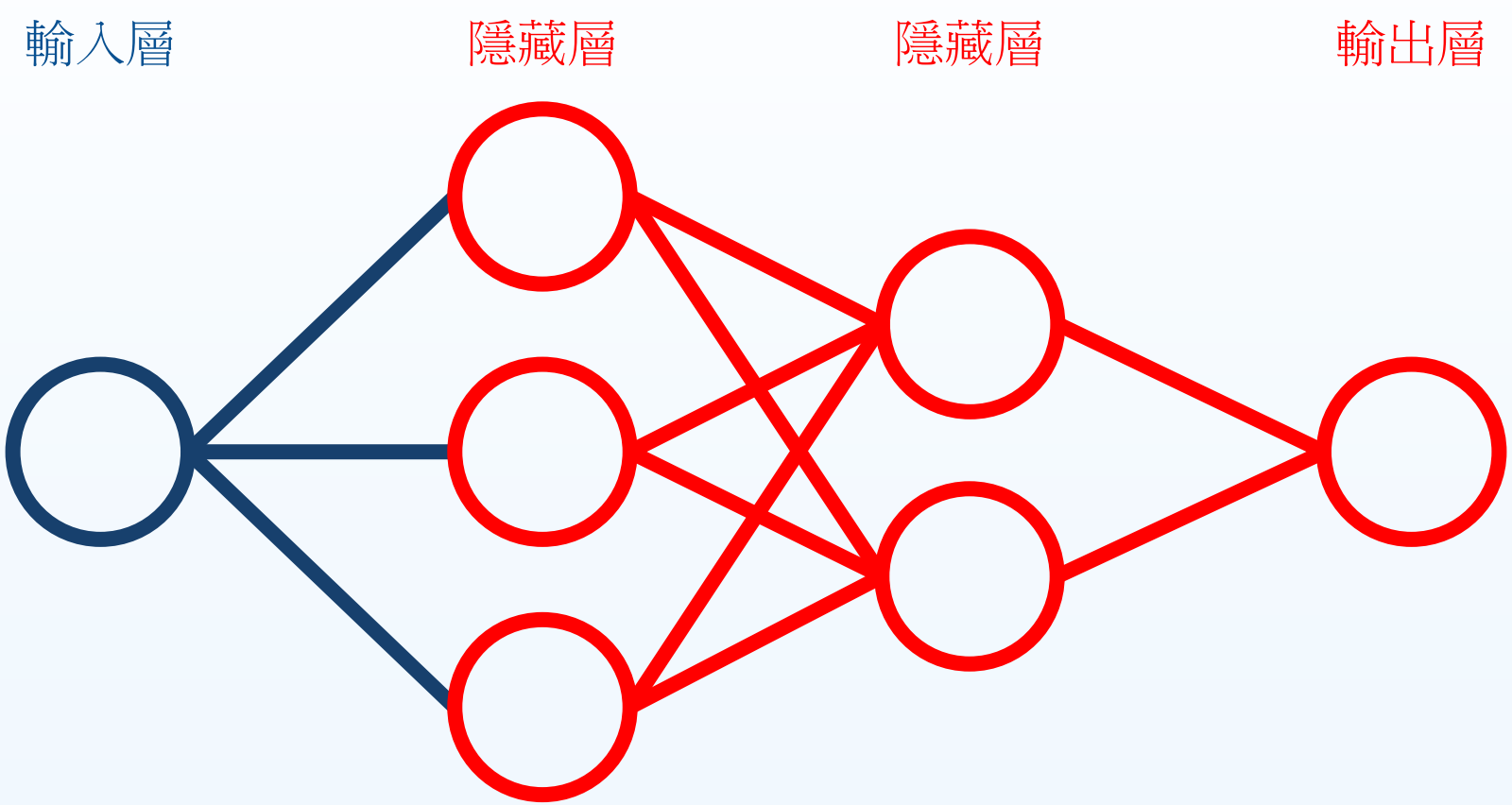
從輸出層向輸入層微分



反向傳播算法

Back Propagation

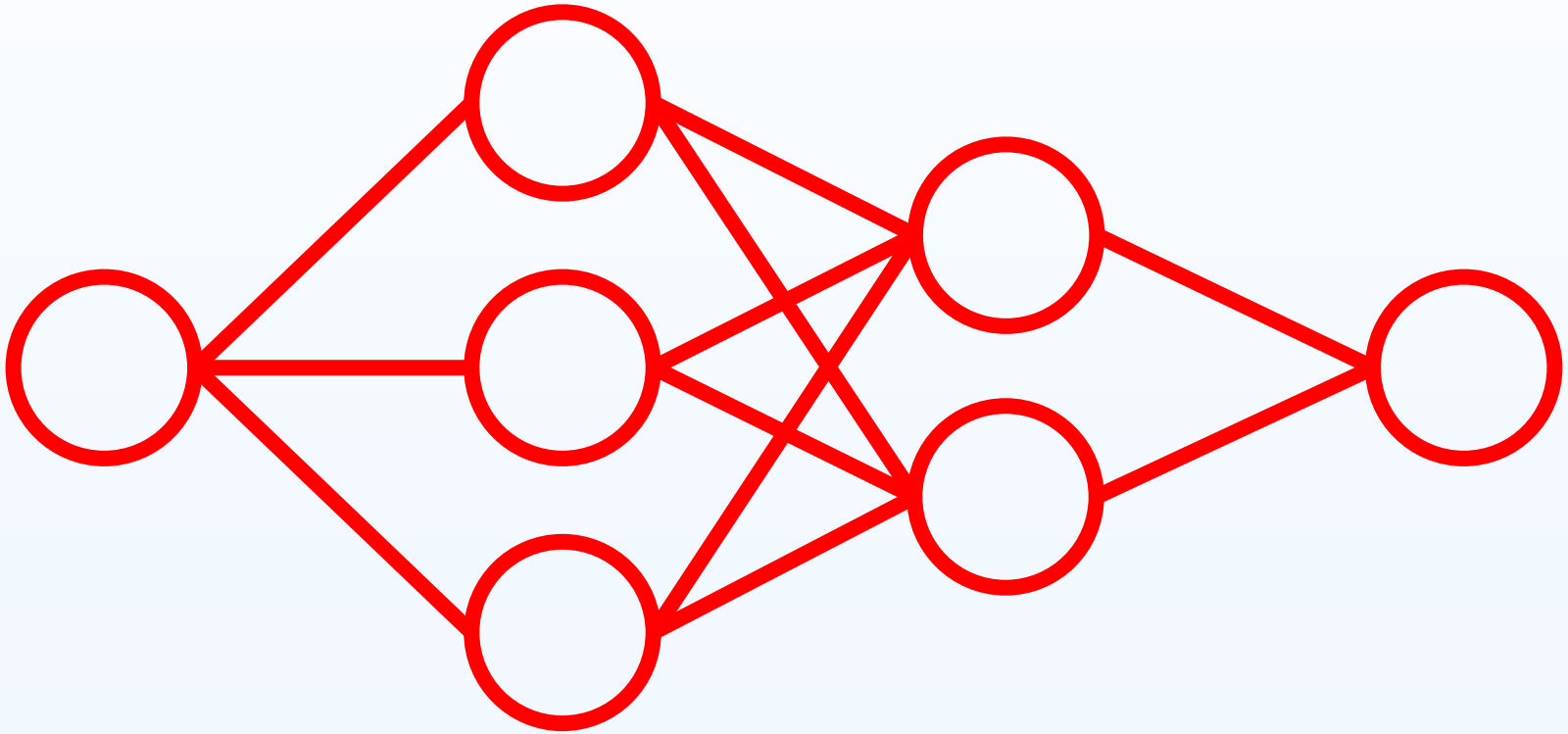
從輸出層向輸入層微分



反向傳播算法

Back Propagation

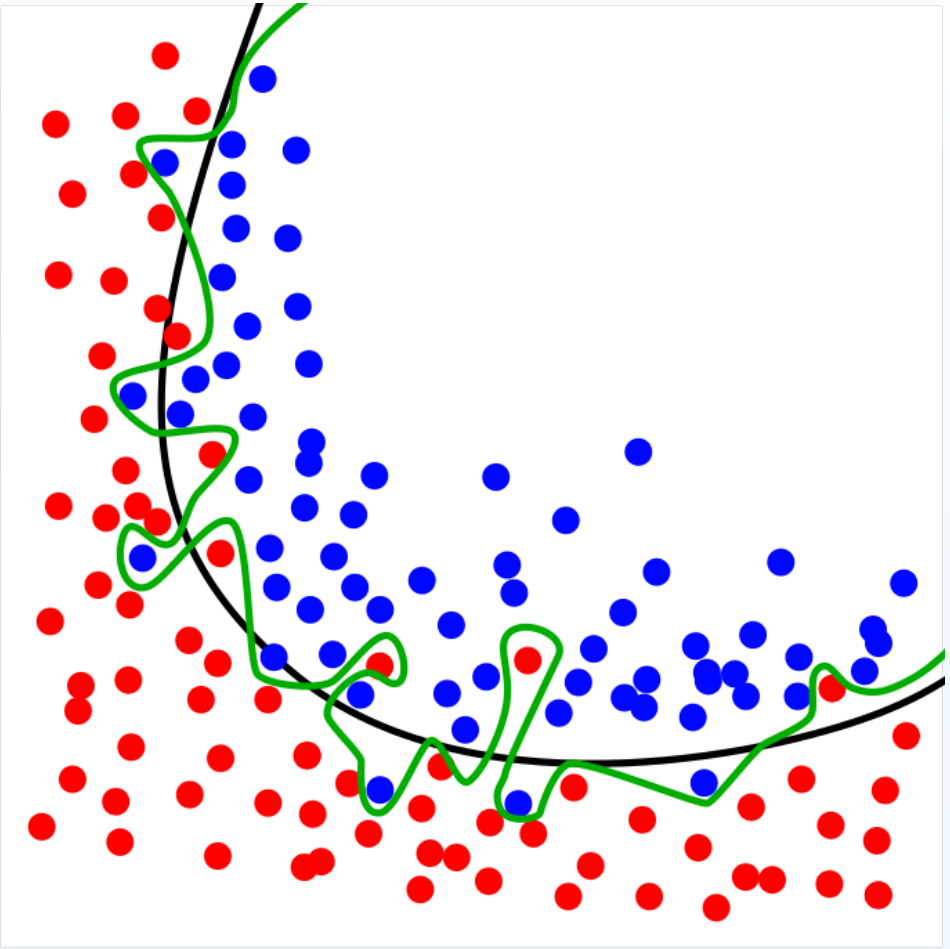
從輸出層向輸入層微分



但其實我們很害怕一件事

過度擬合

Over Fitting

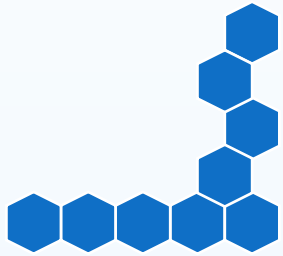


練過頭了
神經網路開始學習不夠廣泛特徵

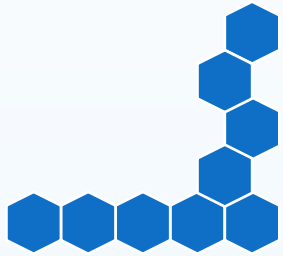
操作上會發現
訓練集的成績很好
但檢驗和測試集的成績很差

只會寫參考書的考生
考大考不會寫也沒有用

怎麼辦？



既然練過頭，那就不要練啊

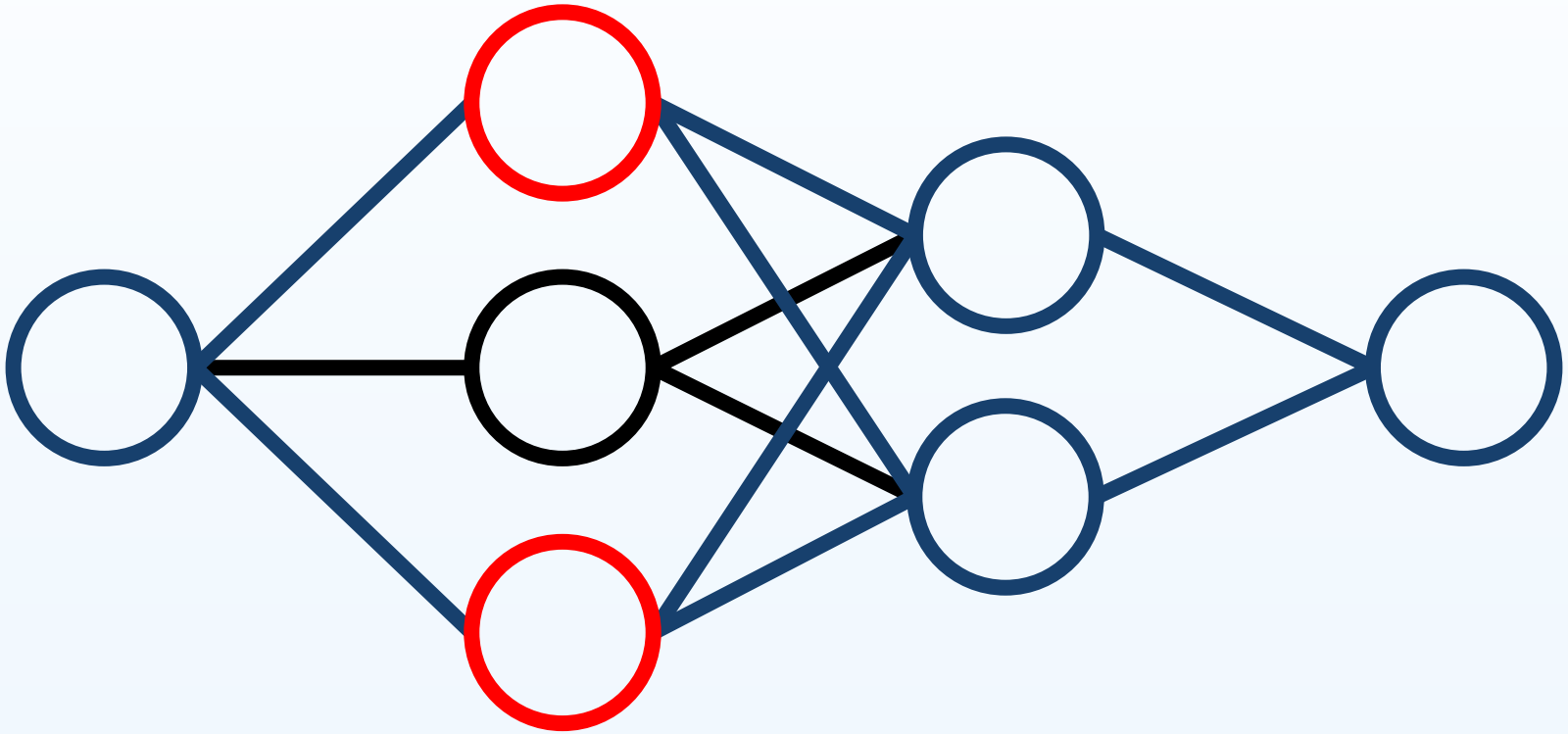


不參與訓練

Drop Out

把部份神經元關掉不做訓練

輸入層　隱藏層　隱藏層　輸出層



Keras 介紹與實做

Introduce Keras

手算

如何實現深度學習模型

How to Achieve Deep Learning Model

手算

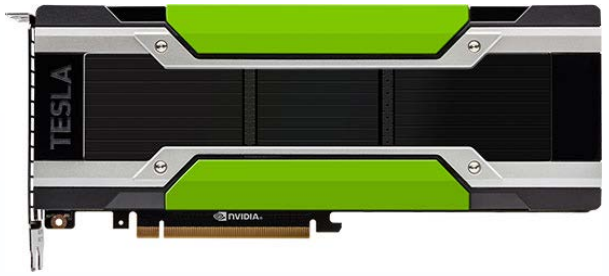
對理論會更有感覺
但是很慢
還會算錯很不方便

Numpy

要刻每一層、優化器、損失函數
調用GPU加速不方便

為什麼要用GPU？

Why GPU?



GPU

核心多
(上千個)

單核心速度慢

擅長平行運算

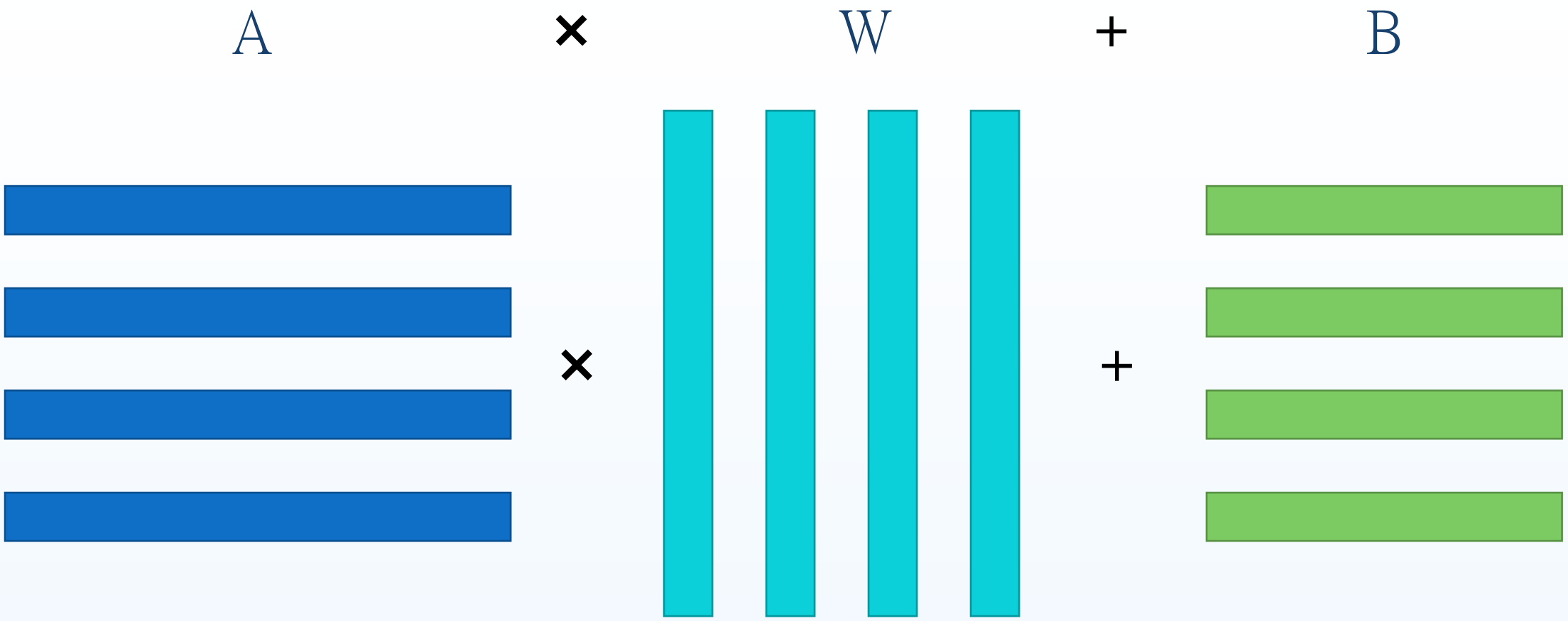
CPU

核心少
(小餘三十個)

單核心速度快

擅長序列運算

建議購買NVidia GPU，CUDA支援的框架比較多



平行運算在矩陣運算速度上具有優勢

深度學習運算框架

Deep Learning Framework

為了方便調用GPU
並且不要重複造輪子
我們通常會使用一些框架
現在還在框架大戰

Keras

支援多種後端

Multiple Backend

模組化

Modular

```
import
keras.backend
as
K
```

適度的擴展性

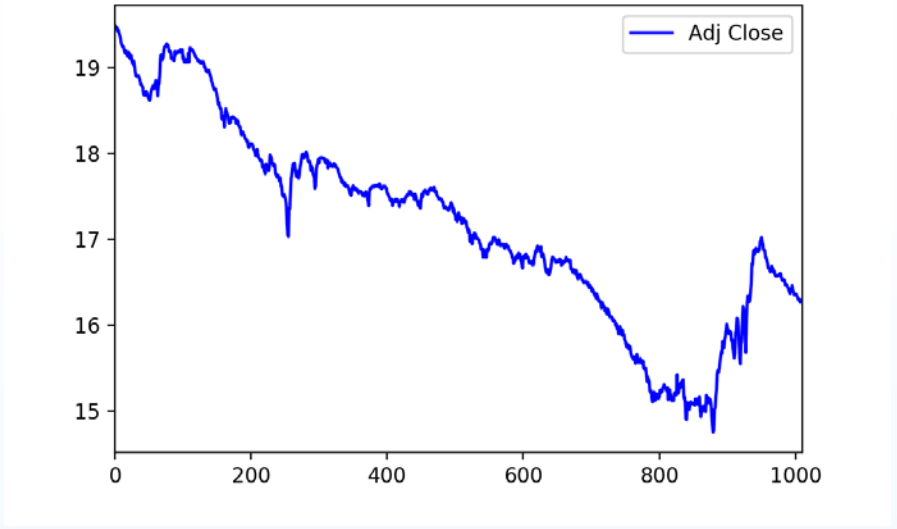
Enough Elasticity

我們今天有兩個資料集

MNIST

手寫數字辨識

深度學習的HelloWorld



ETF

全球智能提供

介紹實務上的應用

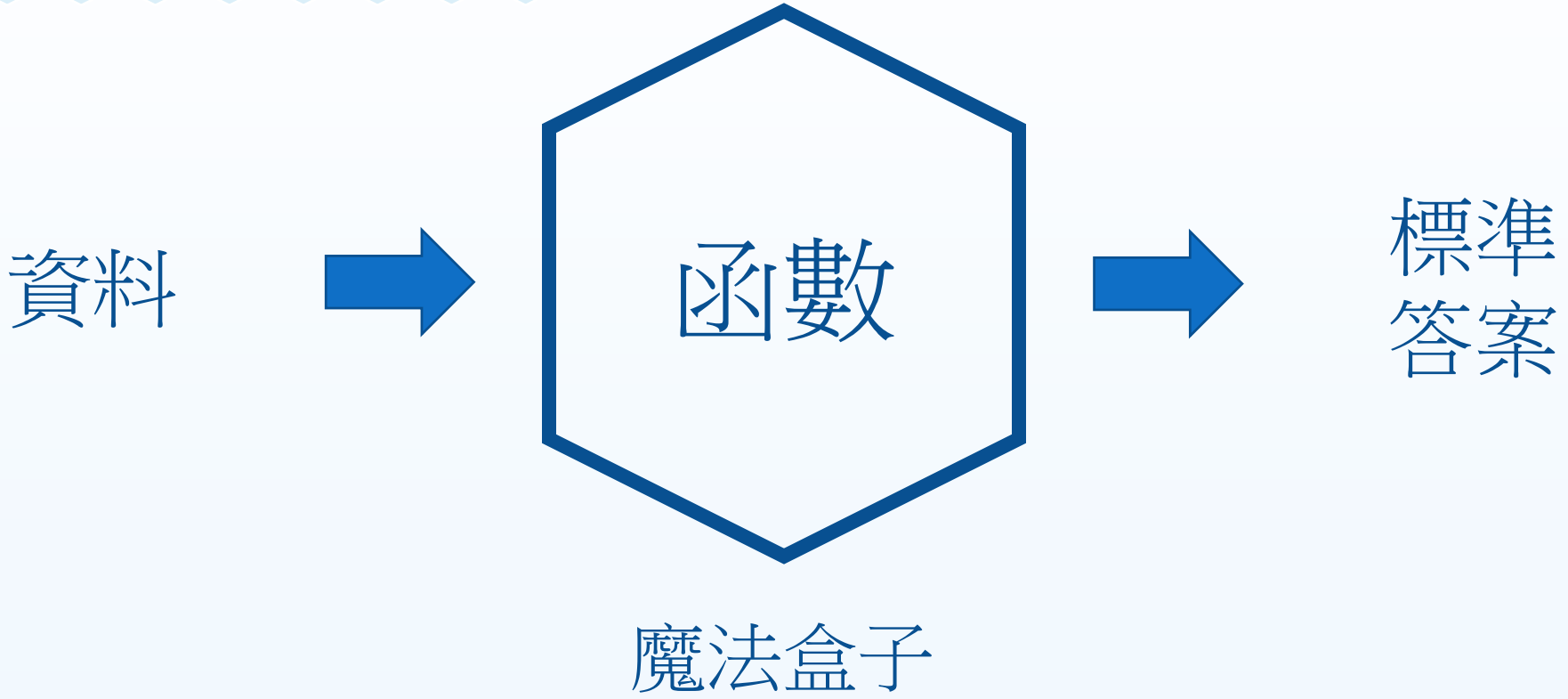
MNIST手寫數字辨識

訓練資料：　　六萬
測試資料：　　一萬

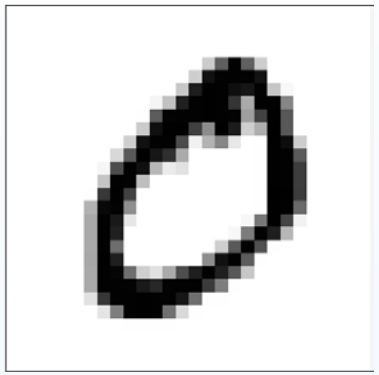
資料分類比夠大
機器學習界的HelloWorld

LeCun, Yann. "The MNIST database of handwritten digits." *http://yann. lecun. com/exdb/mnist/* (1998).

來建模型吧



試試全連接監督學習



28×28

全連接
神經網路

0

1

One-hot Embedding

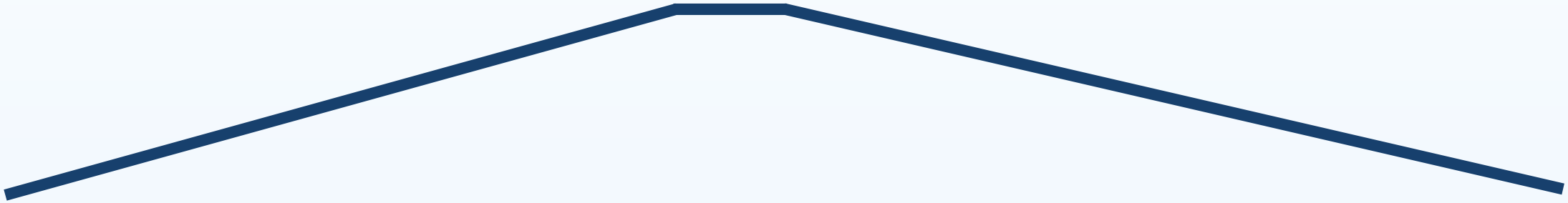
0 1 2 3 4 5 6 7 8 9

這些數字彼此是沒有相關性的
1跟7比較像，但是1卻不一定在7的旁邊

One-hot Embedding

最好的方法是讓每一種分類
自己獨立成一個向量

0 1 2 3 4 5 6 7 8 9



[0 0 0 0 1 0 0 0 0 0]

首先先讀入套件跟資料

```
%matplotlib inline #讓jupyter notebook可以出現你畫的圖片
import numpy as np
import matplotlib.pyplot as plt
from keras.datasets import mnist #資料集
from keras.utils import np_utils # 做one-hot embedding
from keras.models import Sequential #讀入Sequential模式以建立模型
from keras.layers import Dense, Activation #我們需要的keras層
from keras.optimizers import SGD #這次用的優化器

(x0_train, y0_train), (x0_test, y0_test) = mnist.load_data() #讀入MNIST資料集
```

資料整理

```
#將輸入資料打平
x_train = x0_train.reshape(60000, 28*28)
x_test = x0_test.reshape(10000, 28*28)

#將輸出資料做One-hot Embedding
y_train = np_utils.to_categorical(y0_train,10)
y_test = np_utils.to_categorical(y0_test,10)
```

構建模型

序列式模型

```
model = Sequential()
```

構建模型

序列式模型

500個神經元的全連接層

激發函數用S型函數

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

構建模型

序列式模型

500個神經元的全連接層

激發函數用S型函數

500個神經元的全連接層

激發函數用S型函數

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

構建模型

序列式模型

500個神經元的全連接層

500個神經元的全連接層

10個神經元的全連接層

激發函數用Softmax

```
model = Sequential()
model.add(Dense(units=500, input_dim=784))
model.add(Activation('sigmoid'))
model.add(Dense(units=500))
model.add(Activation('sigmoid’))
model.add(Dense(units=10))
model.add(Activation('softmax’))
model.summary() #檢查模型形狀
```

編譯模型

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

決定損失函數和優化器
並且新增準確度作為測量記錄

訓練模型

```
model.fit(x_train, y_train, batch_size=100, epochs=20)
```

決定批次大小以及訓練次數

試用結果

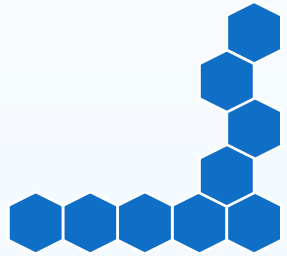
```
result = model.predict_classes(x_test)
```

儲存模型

```
model_json = model.to_json()
open('handwriting_model_architecture.json', 'w').write(model_json)
model.save_weights('handwriting_model_weights.h5')
```

儲存模型形狀及權重

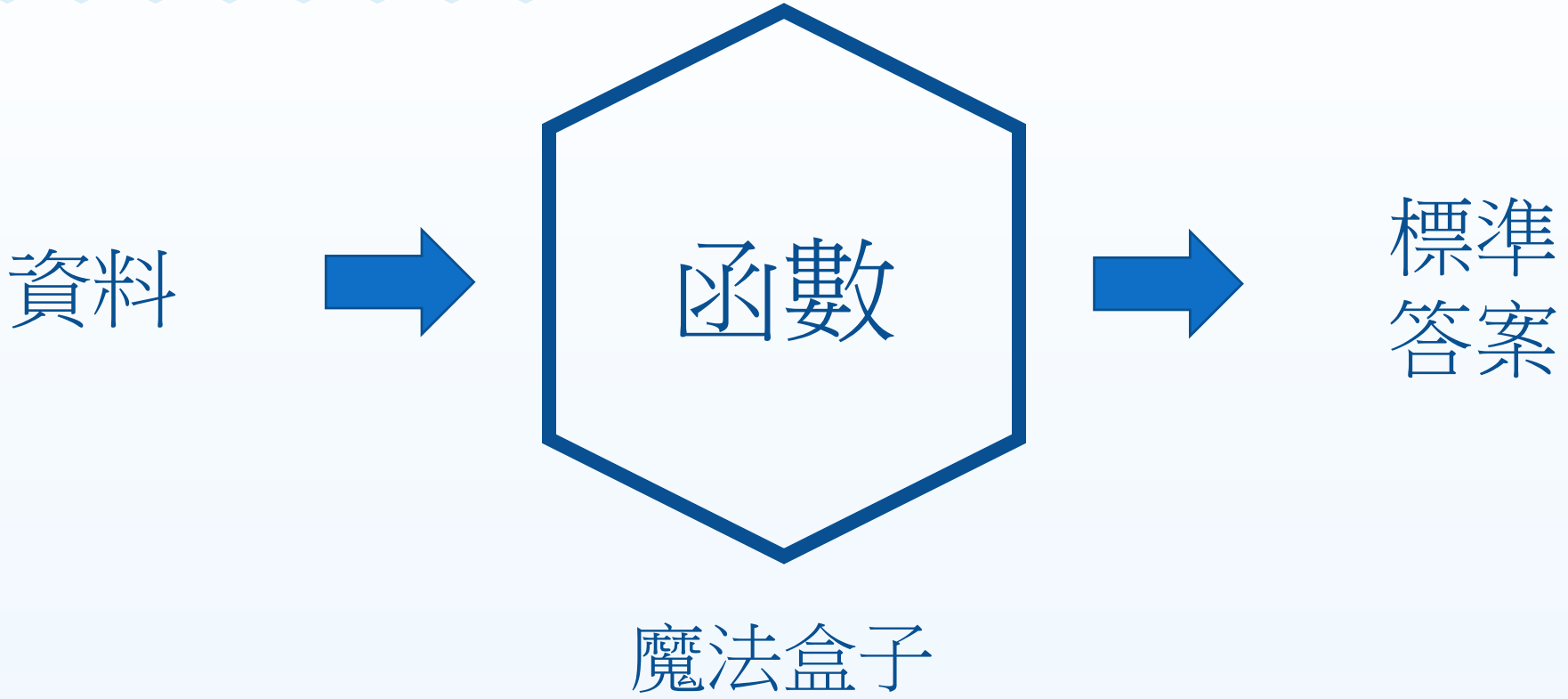
對Keras有基本概念了吧
我們開始試試看預測ETF吧



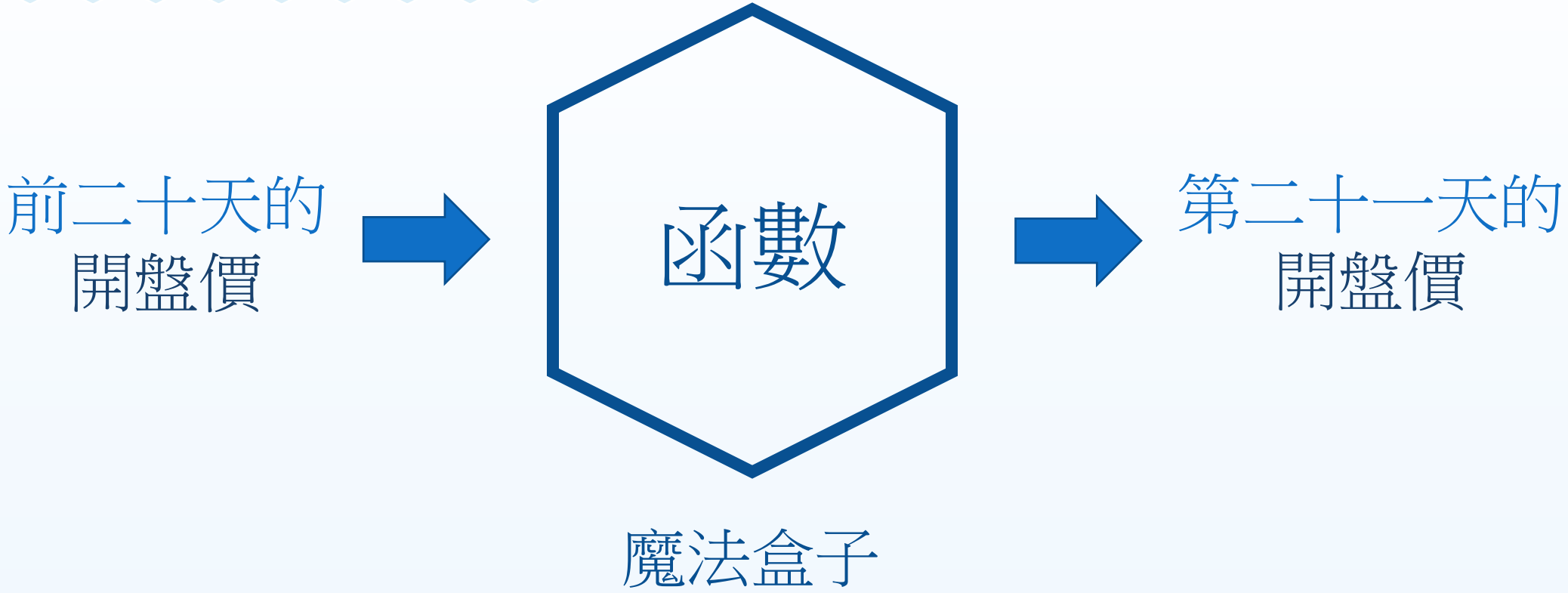
ETF每天都有下面這些資料

開盤價　收盤價　最高價　最低價　交易量　調整收盤價

來建模型吧



來建模型吧



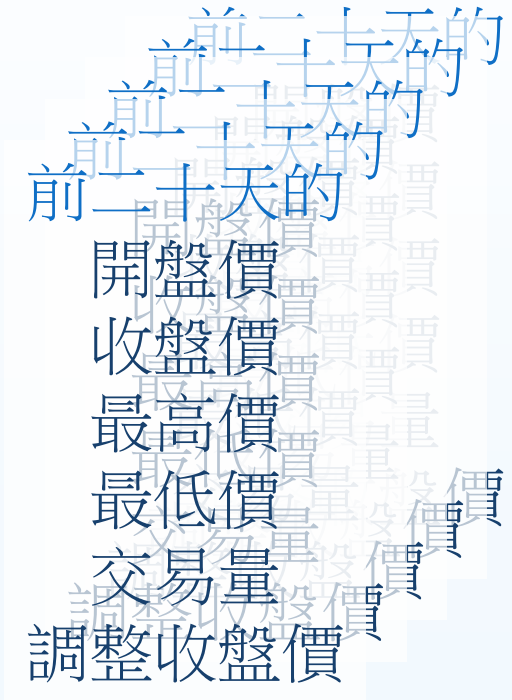
來建模型吧

可能更好的模型

前二十天的
開盤價
收盤價
最高價
最低價
交易量
調整收盤價

第二十一天的
開盤價
漲幅

可能更好的模型



很多份ETF

全連接
神經網路

第二十一天的
開盤價
漲幅