

NTNU CSIE CAMP

# 機器學習概論與 手寫數字辨識實作

# 機器學習概論與手寫數字辨識實作

Machine Learning Theory and MNIST

2024 NTNU CSIE CAMP - 豹肝、地鼠

# Outline

- 機器學習、人工智慧、深度學習
- 機器怎麼學習
- 手寫數字辨識實作

# Machine Learning

# 一起寫程式

The screenshot shows a GitHub Copilot session in a dark-themed code editor. On the left, the GitHub Copilot Chat interface displays a message from user 'monalisa' asking for a unit test for a function. On the right, the code editor shows 'module.py' with a function definition for parsing expenses.

**GITHUB COPILOT: CHAT**

monalisa Write a unit test for this function

**GitHub Copilot**

```
import unittest
import datetime

class TestParseExpenses(unittest.TestCase):
    def test_parse_expenses_with_valid_input(self):
        expenses_string = '2023-01-02 -34.01 USD\n2023-01-03 2.59 DKK
        expected_output = [(datetime.datetime(2023, 1, 2),
                            float(-34.01), 'USD'),
                            (datetime.datetime(2023, 1, 3),
                            float(2.59), 'DKK')]
        self.assertEqual(parse_expenses(expenses_string),
                        expected_output)

    def test_parse_expenses_with_empty_input(self):
        expenses_string = ''
        expected_output = []
        self.assertEqual(parse_expenses(expenses_string),
                        expected_output)

if __name__ == '__main__':
    unittest.main()
```

The code assumes that the datetime module is imported. The test cases cover different scenarios such as valid input, empty input, input with comments, invalid date format, and invalid value format.

**module.py**

```
import datetime

def parse_expenses(expenses_string):
    """Parse the list of expenses and return the list of triples (date, amount, currency).
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
        2023-01-02 -34.01 USD
        2023-01-03 2.59 DKK
        2023-01-03 -2.72 EUR
    """
    expenses = []

    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split(" ")
        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
                        float(value),
                        currency))
    return expenses

expenses_data = '''2023-01-02 -34.01 USD
2023-01-03 2.59 DKK
2023-01-03 -2.72 EUR'''
```

Ask a question or type '/' for commands ➤

GitHub Copilot

# 幫你玩 Minecraft

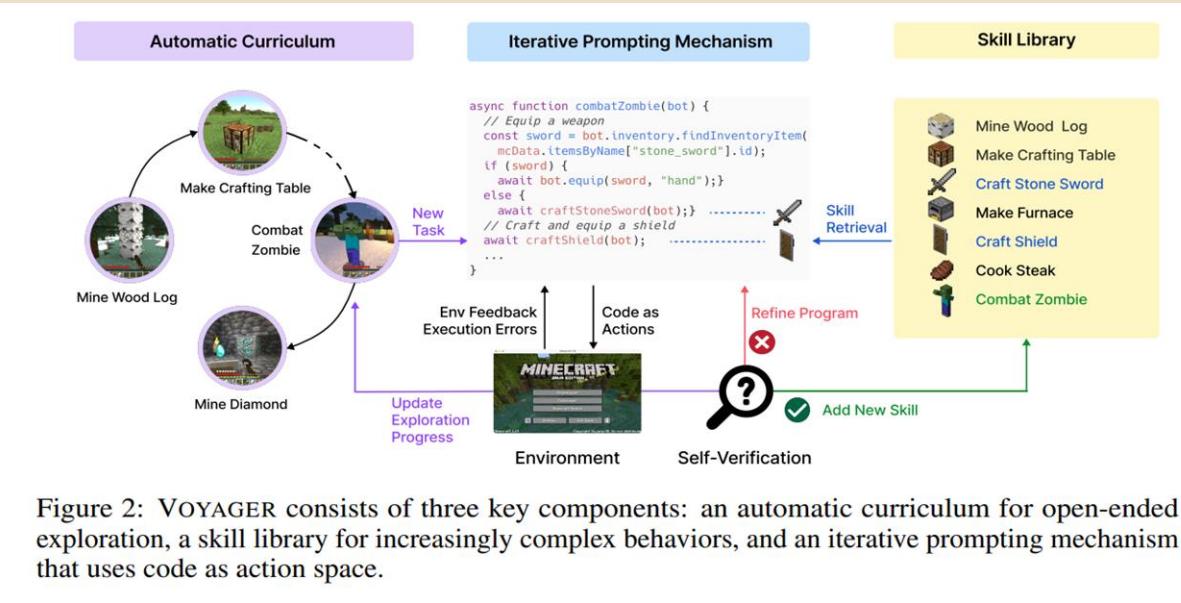


Figure 2: VOYAGER consists of three key components: an automatic curriculum for open-ended exploration, a skill library for increasingly complex behaviors, and an iterative prompting mechanism that uses code as action space.

<https://arxiv.org/pdf/2305.16291.pdf>

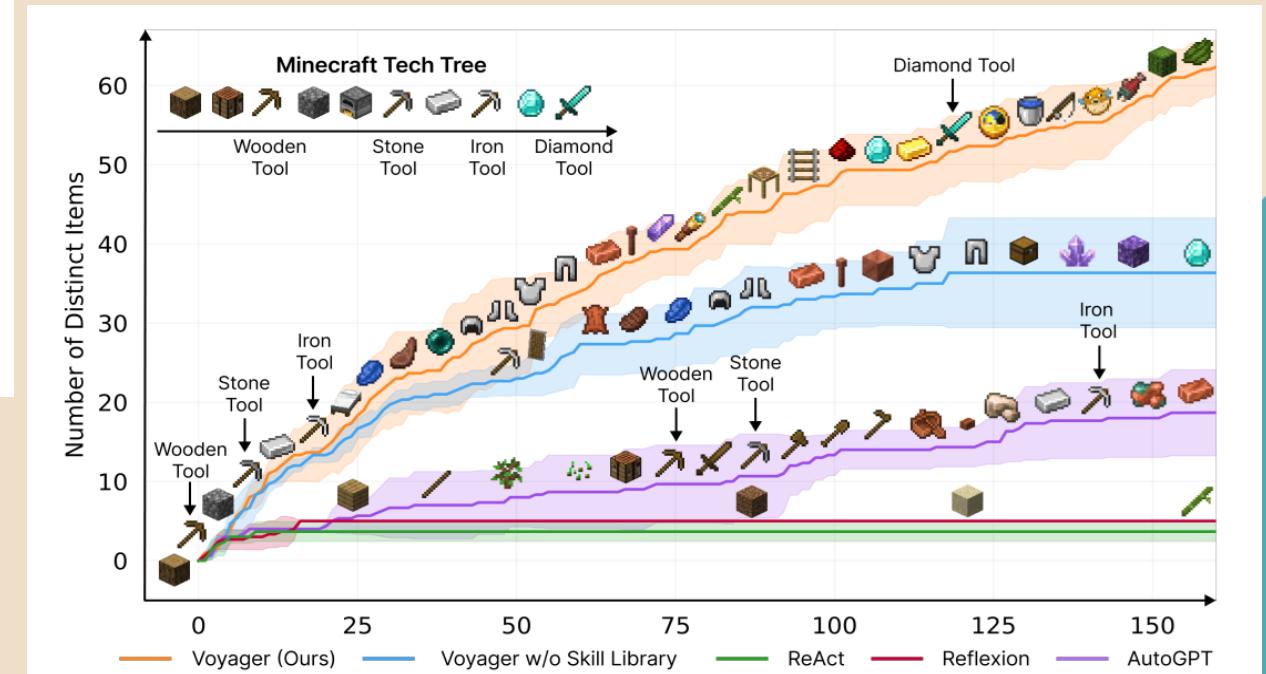


Figure 1: VOYAGER discovers new Minecraft items and skills continually by self-driven exploration, significantly outperforming the baselines. X-axis denotes the number of prompting iterations.

# 亞運圍棋金牌



# 亞運圍棋金牌



Aja Huang



這些都來自於



# Machine Learning



# 為什麼他們都是橡皮筋？

綠色橡皮筋



紅色橡皮筋



黃色橡皮筋  
筋筋有味橡皮筋

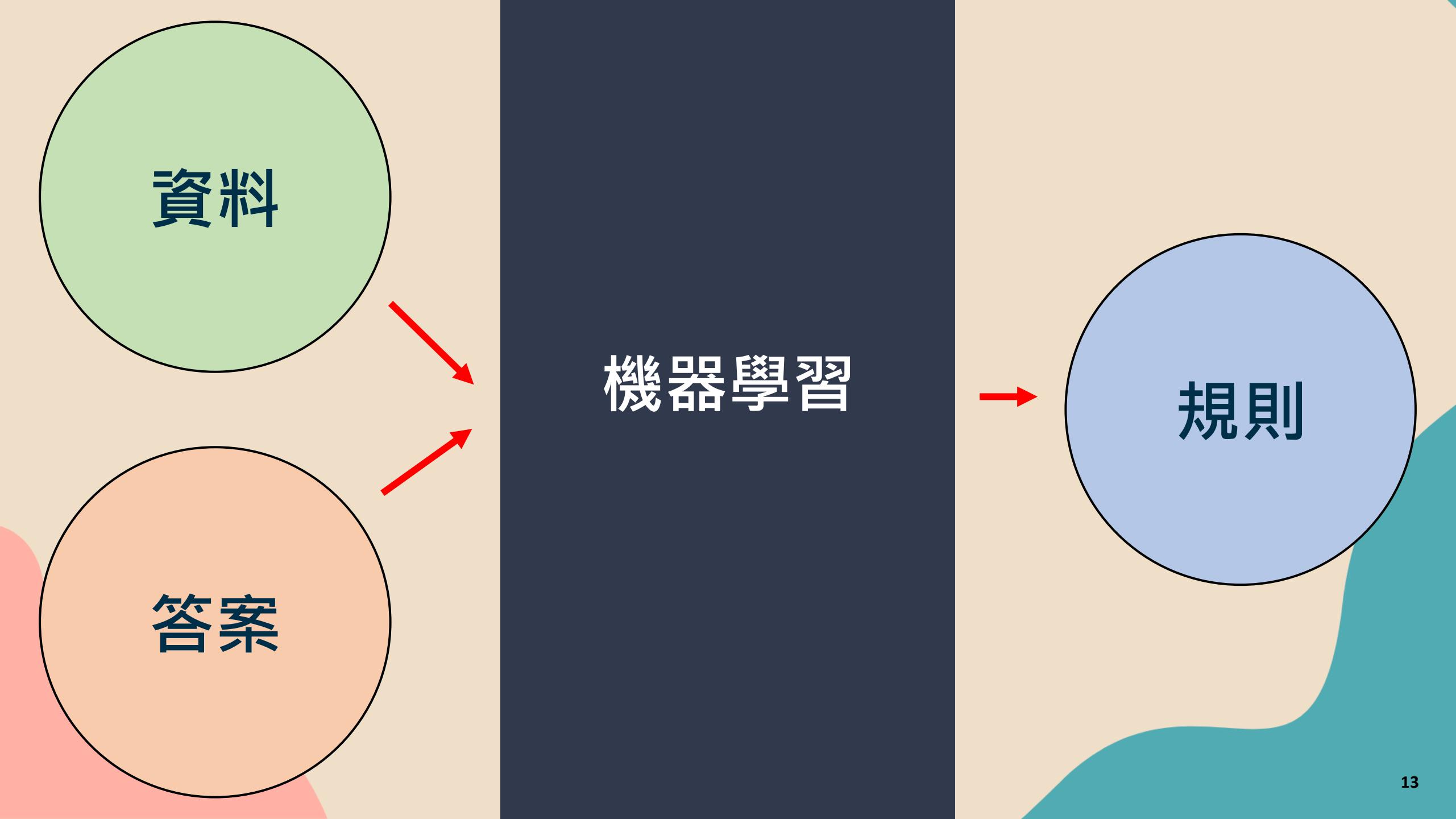


資料

規則

傳統程式

答案



資料

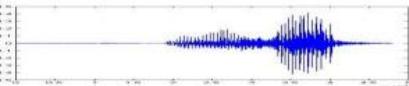
答案

機器學習

規則

機器學習  
↓  
尋找資料的規則

- Speech Recognition

$f($   ) = “How are you”

- Image Recognition

$f($   ) = “Cat”

- Playing Go

$f($   ) = “5-5” (next move)

- Weather Forecast

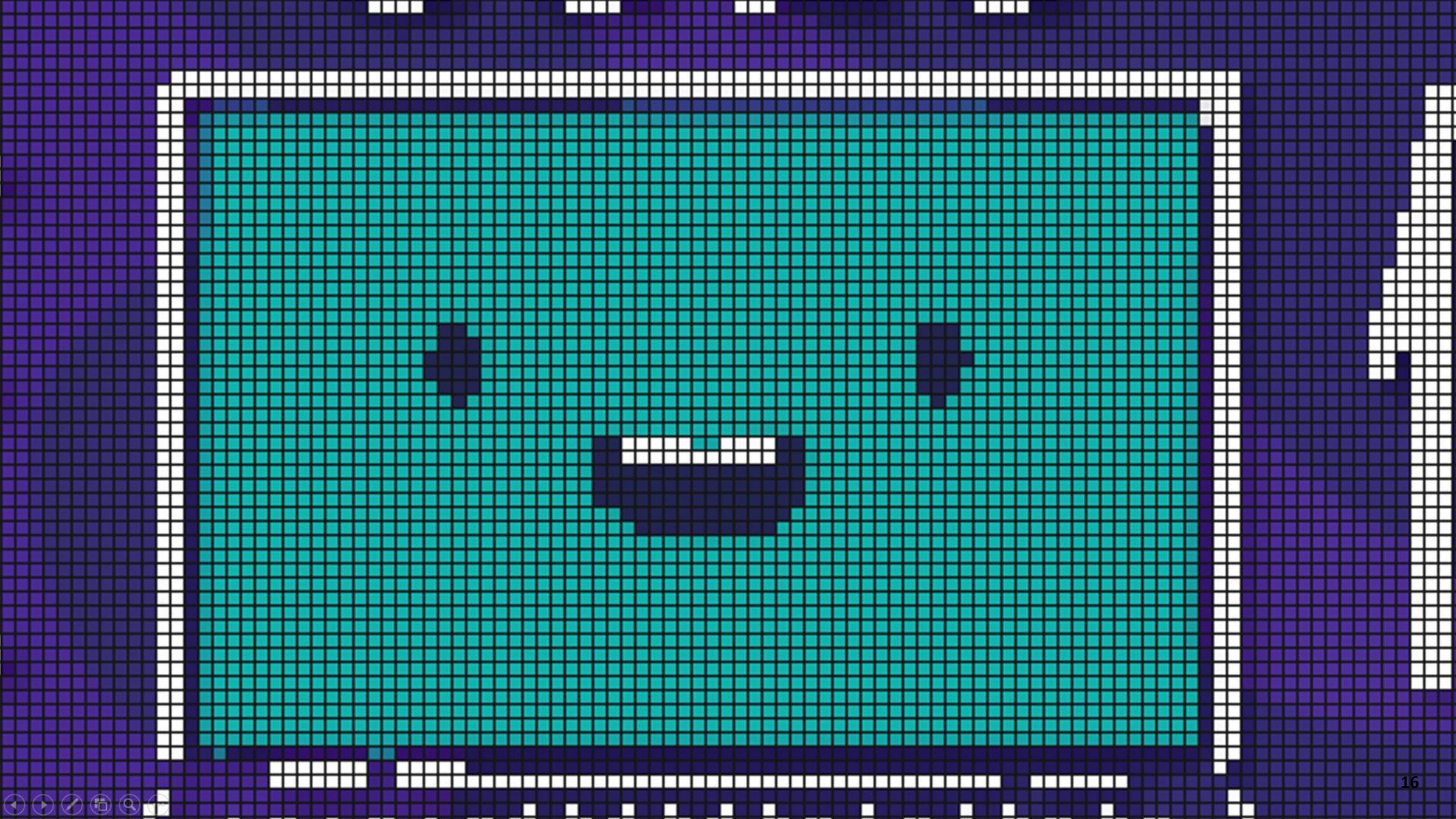
$f(\text{氣壓, 風向, 氣溫...})$  = “陰晴雨”

# 舉個具體的例子來說

2024 師大資工營

NTNU CSIE CAMP





# 舉個具體的例子來說



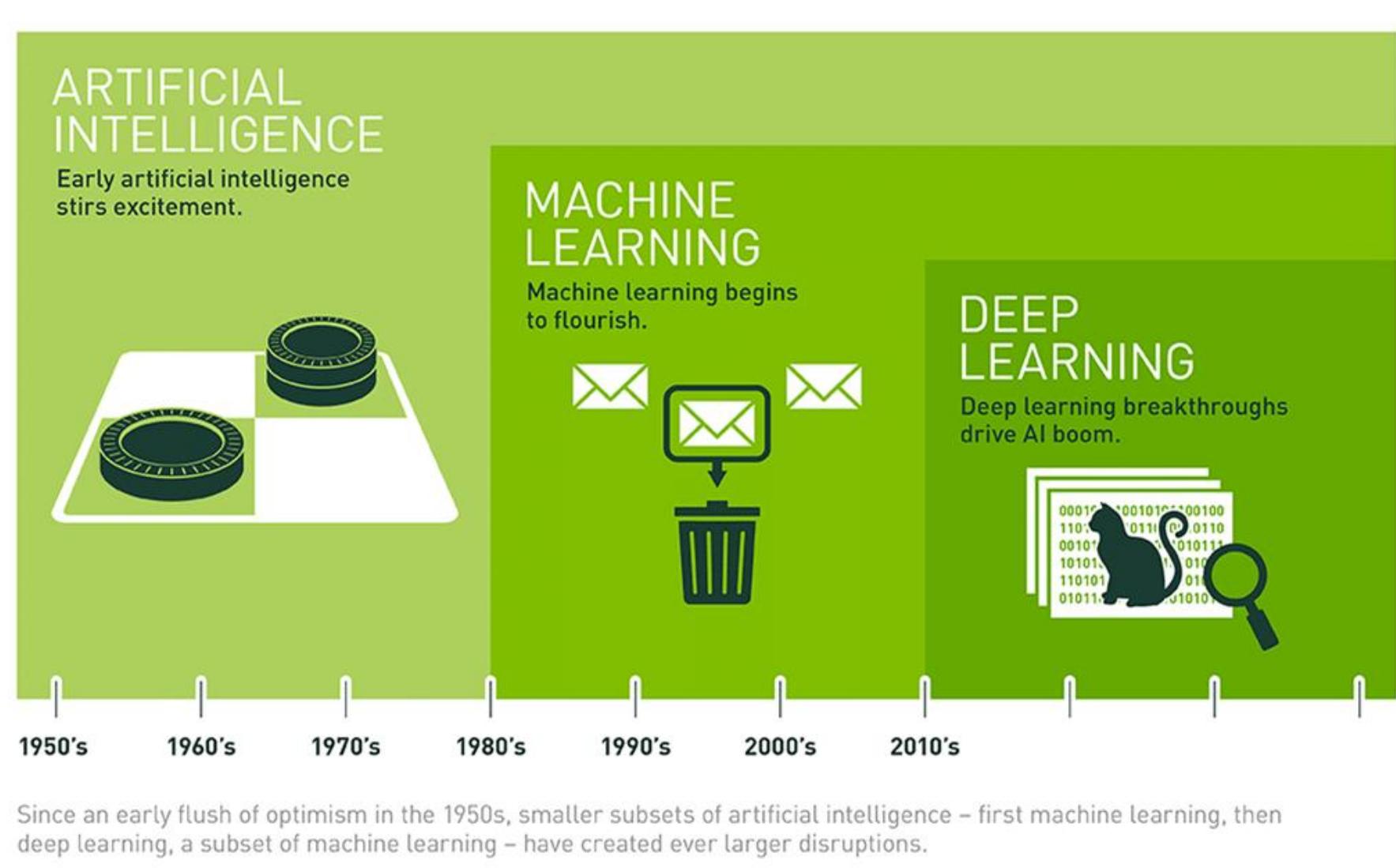
- 白框
- 中間的顏色不一樣
- ~~有笑臉~~

那 人工智慧 又是什麼？



Artificial Intelligence

# AI, ML, DL, Generative AI



# AI, ML, DL, Generative AI

人工智慧（目標）

生成式人工智慧（目標之一）

機器學習（手段）

深度學習（更厲害的手段）

# AI, ML, DL, Generative AI

人工智慧（目標）

生成式人工智慧（目標之一）

機器學習（手段）

深度學習（更厲害的手段）

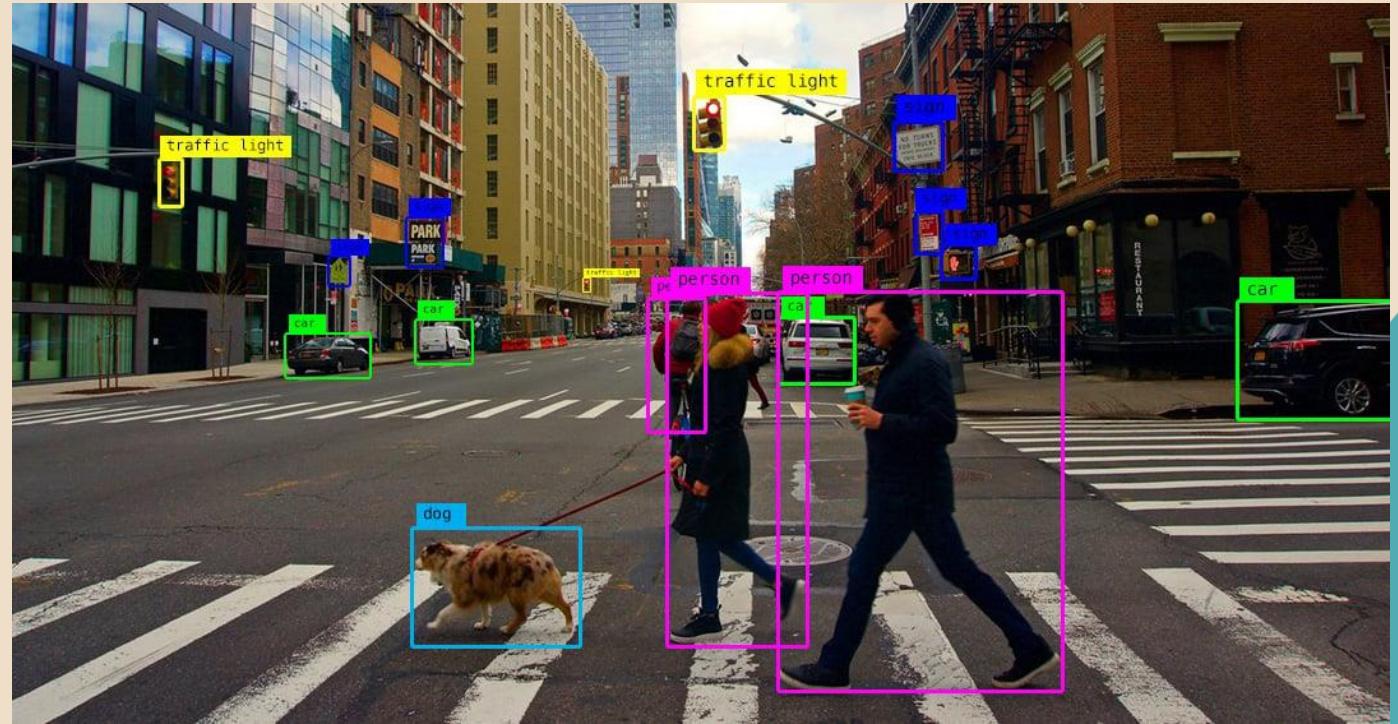
生成式人工智慧（目標之一）

# 介紹幾個有名的領域

- Computer Vision
- NLP
- Generative AI

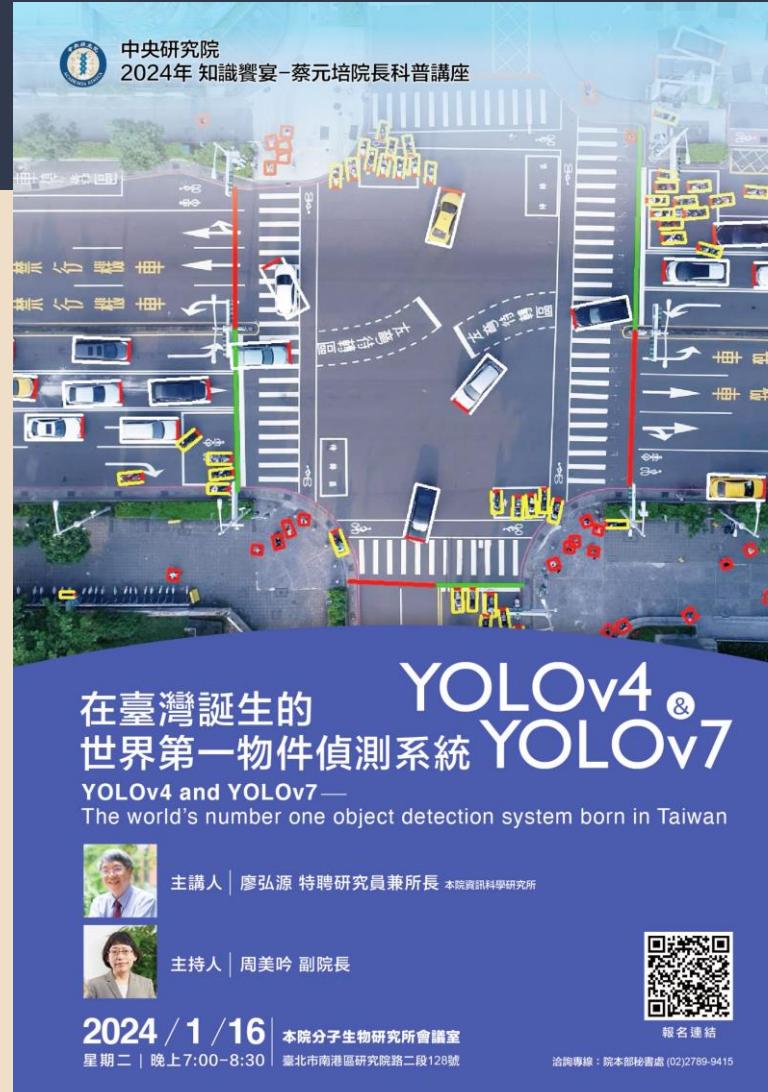
# 介紹幾個有名的領域

- Computer Vision
- NLP
- Generative AI



# 介紹幾個有名的領域

- Computer Vision
- NLP
- Generative AI



You  
Only  
Look  
Once

[在臺灣誕生的世界第一物件偵測系統—YOLOv4 及 YOLOv7 | 廖弘源特聘研究員 - YouTube](#)

[\[2402.13616\] YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information \(arxiv.org\)](#)

# 介紹幾個有名的領域

- Computer Vision
- NLP
- Generative AI

Natural Language Processing

# 介紹幾個有名的領域

- Computer Vision
- NLP
- Generative AI

Natural Language Processing

Natural  
Language  
Understanding

Natural  
Language  
Generation

# 介紹幾個有名的領域

- Computer Vision
- NLP
- Generative AI



# 介紹幾個有名的領域

- Computer Vision
- NLP
- Generative AI

[Sora \(openai.com\)](https://openai.com)



# 依據 **訓練** (演算法) 類型可以分成

- 監督式學習 Supervised Learning
- 非監督式學習 Unsupervised Learning
- 強化式學習 Reinforcement Learning

依據 **目的** 類型可以分成

Classification, Clustering, Regression

# 依據 訓練 (演算法) 類型可以分成

- 監督式學習 Supervised Learning
- 非監督式學習 Unsupervised Learning
- 強化式學習 Reinforcement Learning

資料 ( Data )

身高體重等這種餵給機器學習的資料集

特徵 ( Feature )

資料中足以觀察分類的目標

標記 ( Label )

特徵的答案

# 依據 訓練 (演算法) 類型可以分成

- 監督式學習 Supervised Learning
- 非監督式學習 Unsupervised Learning
- 強化式學習 Reinforcement Learning

# 依據 **訓練** (演算法) 類型可以分成

資料

特徵

標記

- 監督式學習 Supervised Learning
- 非監督式學習 Unsupervised Learning
- 強化式學習 Reinforcement Learning

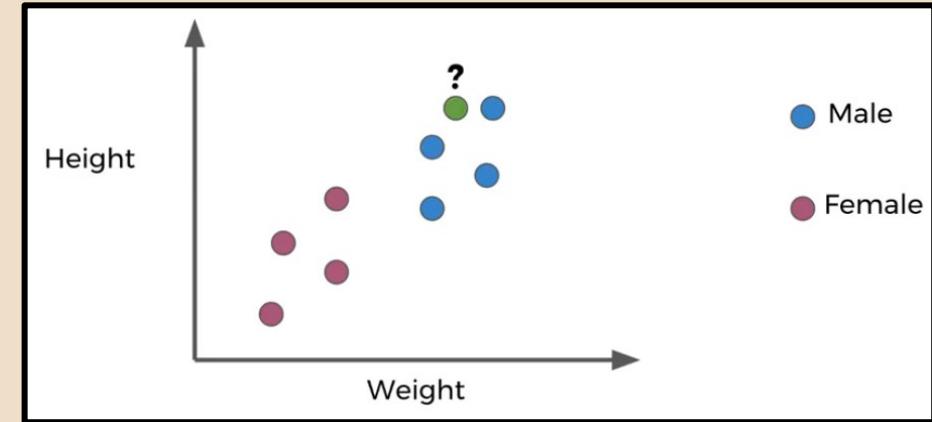
# 依據 訓練 (演算法) 類型可以分成

資料

特徵

標記

- 監督式學習 Supervised Learning
- 非監督式學習 Unsupervised Learning
- 強化式學習 Reinforcement Learning



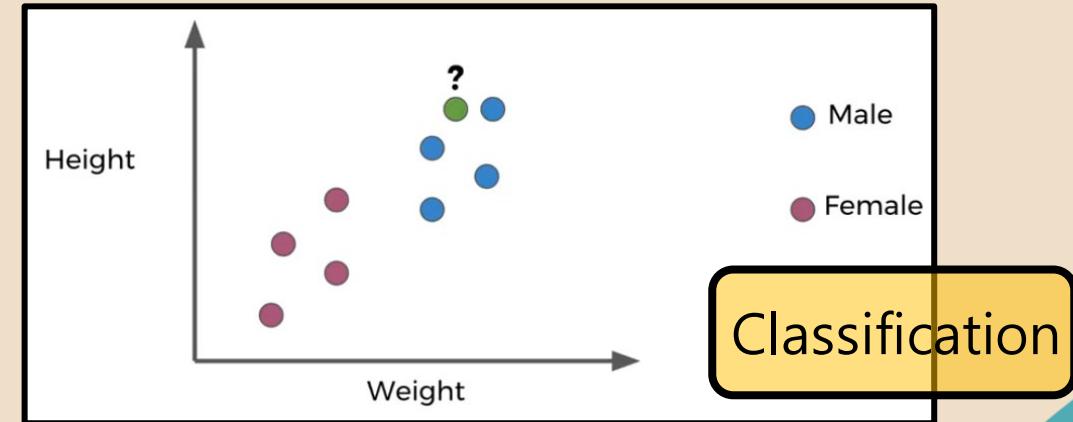
# 依據 訓練 (演算法) 類型可以分成

資料

特徵

標記

- 監督式學習 Supervised Learning
- 非監督式學習 Unsupervised Learning
- 強化式學習 Reinforcement Learning



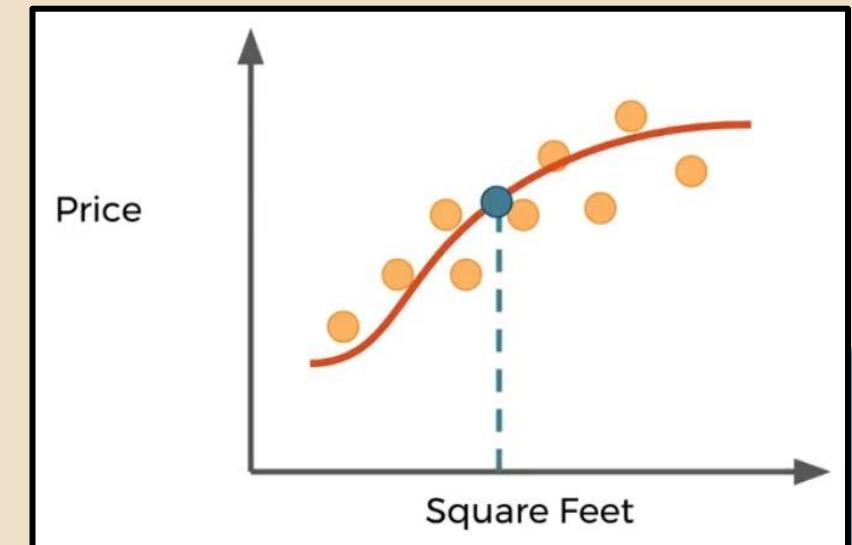
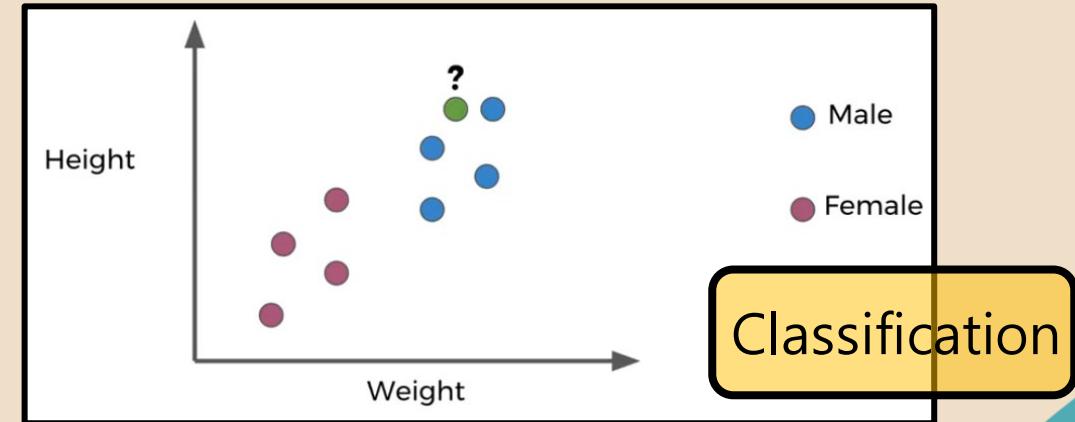
# 依據 訓練 (演算法) 類型可以分成

資料

特徵

標記

- 監督式學習 Supervised Learning
- 非監督式學習 Unsupervised Learning
- 強化式學習 Reinforcement Learning



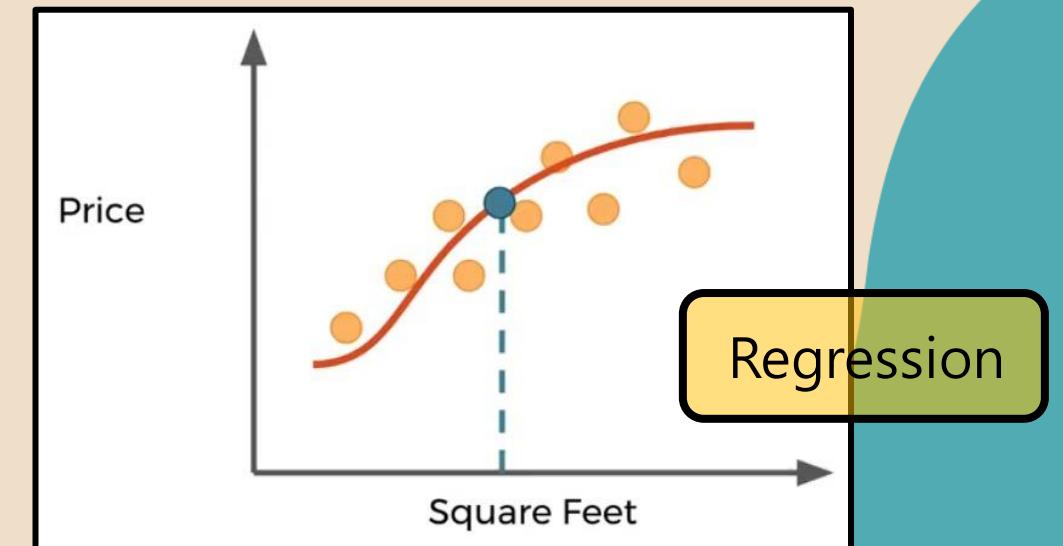
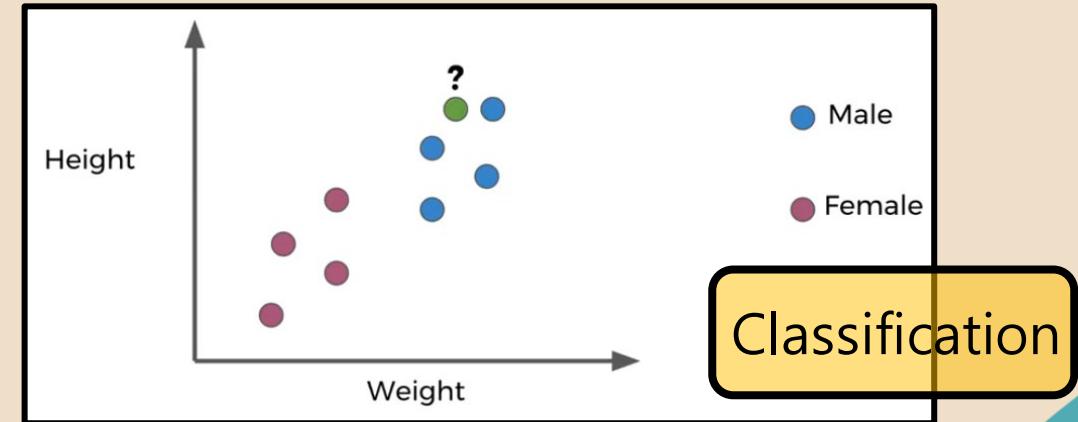
# 依據 訓練 (演算法) 類型可以分成

資料

特徵

標記

- 監督式學習 Supervised Learning
- 非監督式學習 Unsupervised Learning
- 強化式學習 Reinforcement Learning



# 依據 訓練 (演算法) 類型可以分成

資料

特徵

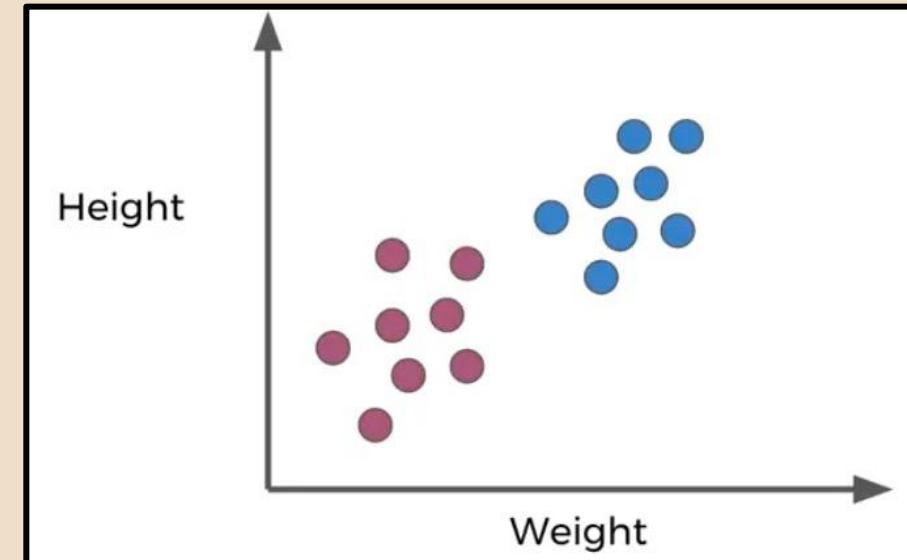
- 監督式學習 Supervised Learning
- 非監督式學習 Unsupervised Learning
- 強化式學習 Reinforcement Learning

# 依據 訓練 (演算法) 類型可以分成

資料

特徵

- 監督式學習 Supervised Learning
- 非監督式學習 Unsupervised Learning
- 強化式學習 Reinforcement Learning

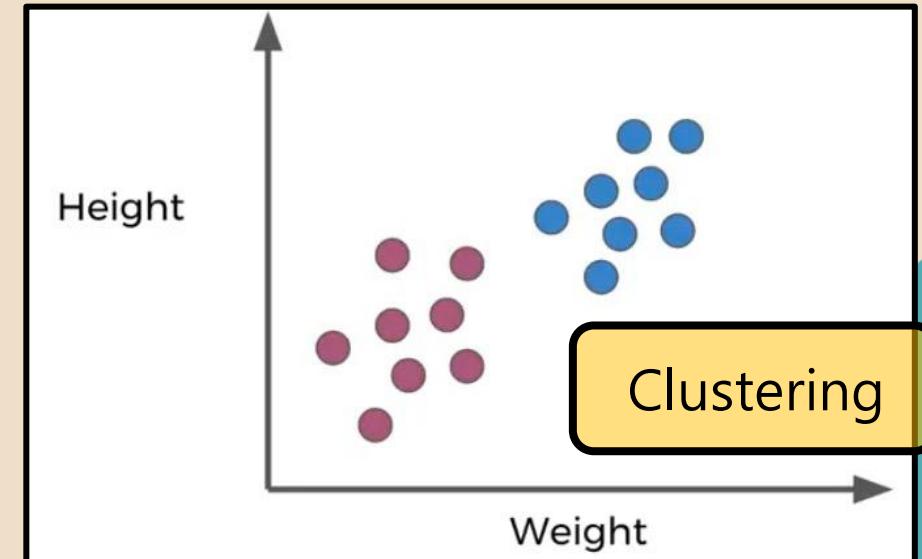


# 依據 訓練 (演算法) 類型可以分成

資料

特徵

- 監督式學習 Supervised Learning
- 非監督式學習 Unsupervised Learning
- 強化式學習 Reinforcement Learning



# 依據 訓練 (演算法) 類型可以分成

資料

- 監督式學習 Supervised Learning
- 非監督式學習 Unsupervised Learning
- 強化式學習 Reinforcement Learning

# 依據 訓練 (演算法) 類型可以分成

資料

不同狀況 (state)

- 監督式學習 Supervised Learning
- 非監督式學習 Unsupervised Learning
- 強化式學習 Reinforcement Learning

嘗試各種決定 (action)

給定時間內的最高獎勵 (reward)



# 依據 **目的** 類型可以分成

- 分類 Classification：就是分類
  - 分類藍色、綠色等
- 迴歸分析 Regression：多為預測一個數值
  - 87%不能在高了
- 分群 Clustering：通常用來處理「沒有正確答案」的問題
  - 貓派、狗派

# 人工智慧三要素

數據

演算法

算力

# 如何 教 模型



# 先來一題數學題

<https://www.desmos.com/calculator?lang=zh-TW>

# 尋找預測線的 **五** 步驟

1. 蒐○資○

# 尋找預測線的 **五** 步驟

1. 蒐集資料

2. O入資料

# 尋找預測線的 **五** 步驟

1. 蒐集資料
2. 讀入資料
3. 畫出分〇圖

# 尋找預測線的 **五** 步驟

1. 蒐集資料
2. 讀入資料
3. 畫出分布圖
4. 畫出〇〇線

# 尋找預測線的 **五** 步驟

1. 蒐集資料
2. 讀入資料
3. 畫出分布圖
4. 畫出預測線
5. 合〇〇〇〇和〇〇〇

# 尋找預測線的 **五** 步驟

1. 蒐集資料
2. 讀入資料
3. 畫出分布圖
4. 畫出預測線
5. 合併分布圖和預測線

# 1. 蒴集資料

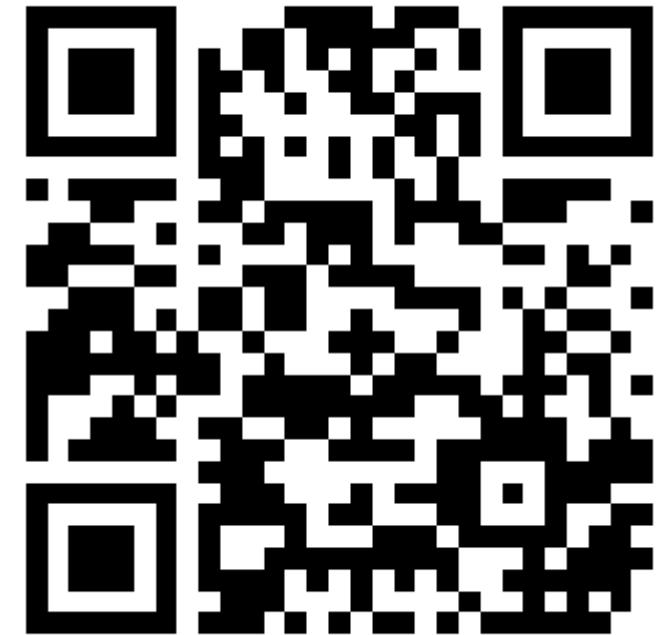


文科理科成績大調查 🎩

歡迎各位特務們來到師大資工營！

為了瞭解各位特務們的特長，我們需要調查你的文科和理科的綜合分數！

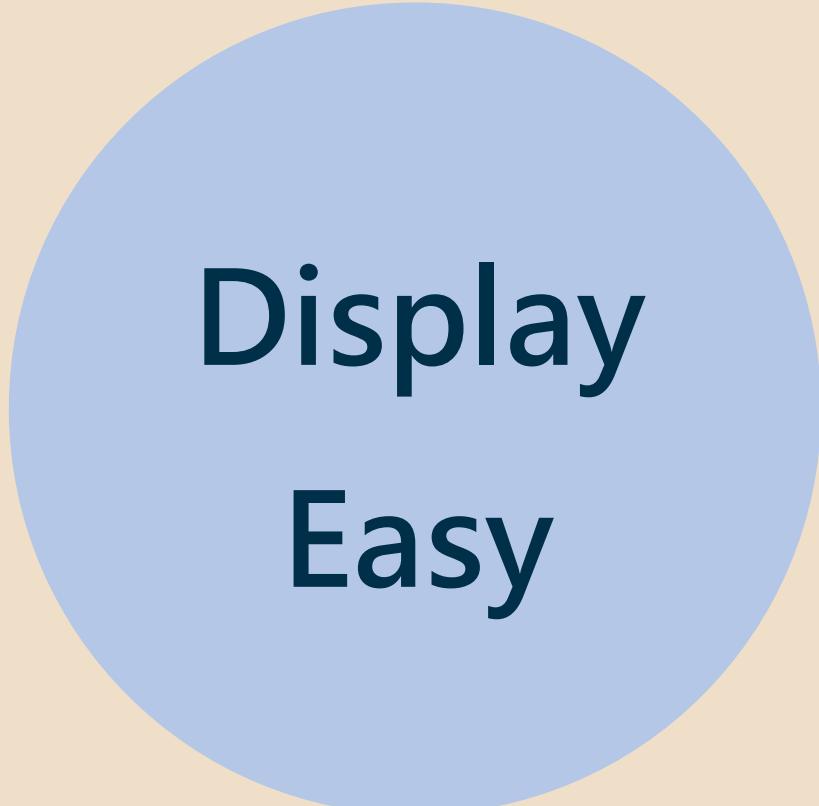
<https://www.surveycake.com/s/xX1d0>



# 開發工具：Jupyter



Debug  
Fast



Display  
Easy

# Debug Fast

Shift + Enter 執行

```
today = "NTNU CSIE Camp"  
year = 2024
```

```
print(today + str(year))  
year += 1
```

```
print(today + str(year))
```

Display  
Easy

# Debug Fast

```
today = "NTNU CSIE Camp"  
year = 2024
```

[1]

Shift + Enter 執行

```
print(today + str(year))  
year += 1
```

```
print(today + str(year))
```

Display  
Easy

# Debug Fast

```
today = "NTNU CSIE Camp"  
year = 2024
```

[1]

```
print(today + str(year))  
year += 1
```

[2]

NTNU CSIE CAMP2024

Shift + Enter 執行

```
print(today + str(year))
```

Display  
Easy

# Debug Fast

```
today = "NTNU CSIE Camp"  
year = 2024
```

[1]

```
print(today + str(year))  
year += 1
```

[2]

NTNU CSIE CAMP2024

```
print(today + str(year))
```

[3]

NTNU CSIE CAMP2025

Display  
Easy

# Debug Fast

```
today = "NTNU CSIE Camp"  
year = 2024
```

[1]

```
print(today + str(year))  
year += 1
```

[2]

NTNU CSIE CAMP2024

Shift + Enter 執行

```
print(today + str(year))
```

Display  
Easy

# Debug Fast

```
today = "NTNU CSIE Camp"  
year = 2024
```

[1]

Shift + Enter 執行

```
print(today + str(year))  
year += 1
```

```
print(today + str(year))
```

Display  
Easy

# Debug Fast

```
today = "NTNU CSIE Camp"  
year = 2024
```

[1]

Shift + Enter 執行

```
print(today + str(year))  
year += 2
```

```
print(today + str(year))
```

Display  
Easy

# Debug Fast

```
today = "NTNU CSIE Camp"  
year = 2024
```

[1]

```
print(today + str(year))  
year += 2
```

[4]

NTNU CSIE CAMP2024

Shift + Enter 執行

```
print(today + str(year))
```

Display  
Easy

# Debug Fast

```
today = "NTNU CSIE Camp"  
year = 2024
```

[1]

```
print(today + str(year))  
year += 2
```

[4]

NTNU CSIE CAMP2024

```
print(today + str(year))
```

[5]

NTNU CSIE CAMP2026

Display  
Easy

# 開發工具：Jupyter

Debug  
Fast

Shift + Enter 執行

## Display Easy

```
course_list = ["py_basic", "py_adven"]  
course_list.append("ML")
```

```
len(course_list)
```

```
course_list[1]
```

```
course_list.pop()  
course_list
```

# 開發工具：Jupyter

Debug  
Fast

## Display Easy

```
course_list = ["py_basic", "py_adven"]  
course_list.append("ML")
```

[1]

Shift + Enter 執行

```
len(course_list)
```

```
course_list[1]
```

```
course_list.pop()  
course_list
```

# 開發工具：Jupyter

Debug  
Fast

## Display Easy

```
course_list = ["py_basic", "py_adven"]  
course_list.append("ML")
```

[1]

```
len(course_list)
```

[2]

3

Shift + Enter 執行

```
course_list[1]
```

```
course_list.pop()  
course_list
```

# 開發工具：Jupyter

Debug  
Fast

## Display Easy

```
course_list = ["py_basic", "py_adven"]  
course_list.append("ML")
```

[1]

```
len(course_list)
```

[2]

3

```
course_list[1]
```

[3]

‘py\_adven’

Shift + Enter 執行

```
course_list.pop()  
course_list
```

# 開發工具：Jupyter

Debug  
Fast

## Display Easy

```
course_list = [ "py_basic", "py_adven"]  
course_list.append("ML")
```

[1]

```
len(course_list)
```

[2]

3

```
course_list[1]
```

[3]

‘py\_adven’

```
course_list.pop()  
course_list
```

[4]

[ ‘py\_basic’, ‘py\_adven’ ]

## 2. 讀入資料

```
1 import pandas as pd  
2  
3 # TODO: 貼上網址抓資料  
4 url = ".../chinese_science_score.csv"  
5 data = pd.read_csv(url)
```

### 3. 畫出分布圖

```
1 # TODO : 設定 x, y 數值
2 x = data['chinese']
3 y = data['science']
4
5 "*****"
6
7 import matplotlib.pyplot as plt
8
9 # TODO : 更改點的樣式與顏色
10 plt.scatter(x, y, marker = "x", color = "red") # 點出所有位置
11
12 plt.title("文科 - 理科")
13 plt.xlabel("文科")
14 plt.ylabel("理科")
15 plt.show() # 顯示圖片
16
17 "*****"
18
19 from matplotlib.font_manager import fontManager
20 import matplotlib as mlp
21 fontManager.addfont("ChineseFont.ttf")
22 mlp.rcParams['font', family = "ChineseFont"]
```

## 4. 畫出預測線

```
● ● ●  
1 a = 0  
2 b = 0  
3  
4 y_pred = a * x + b      # 已知 x 值，預測 y 值  
5  
6 # TODO 將所有點連起來，顏色改成藍色  
7 plt.plot(x, y_pred, color = "blue")  
8  
9 plt.show()
```

# 5. 合併分布圖和預測線

```
1 def plot_pred(a, b):
2
3     # >畫圖
4     y_pred = a*x + b
5     plt.scatter(x, y, marker = "x", color = "red", label = "真實數據")
6     # >預測線
7     plt.plot(x, y_pred, color = "blue", label = "預測線")
8
9     plt.title("文科 - 理科")
10    plt.xlabel("文科")
11    plt.ylabel("理科")
12
13    # 固定 x 軸跟 y 軸
14    plt.xlim([-10, 110])
15    plt.ylim([-10, 110])
16
17    plt.legend() # 合併後，加入圖例
18    plt.show()
19
20    """
21
22 plot_pred(0, 0)
```

# 競賽挑戰

# 6. 增加數值滑桿

```
1 from ipywidgets import interact  
2  
3 # TODO: 增加數值滑桿  
4 interact(plot_pred, a = (-100, 100, 1), b = (-100, 100, 1))
```

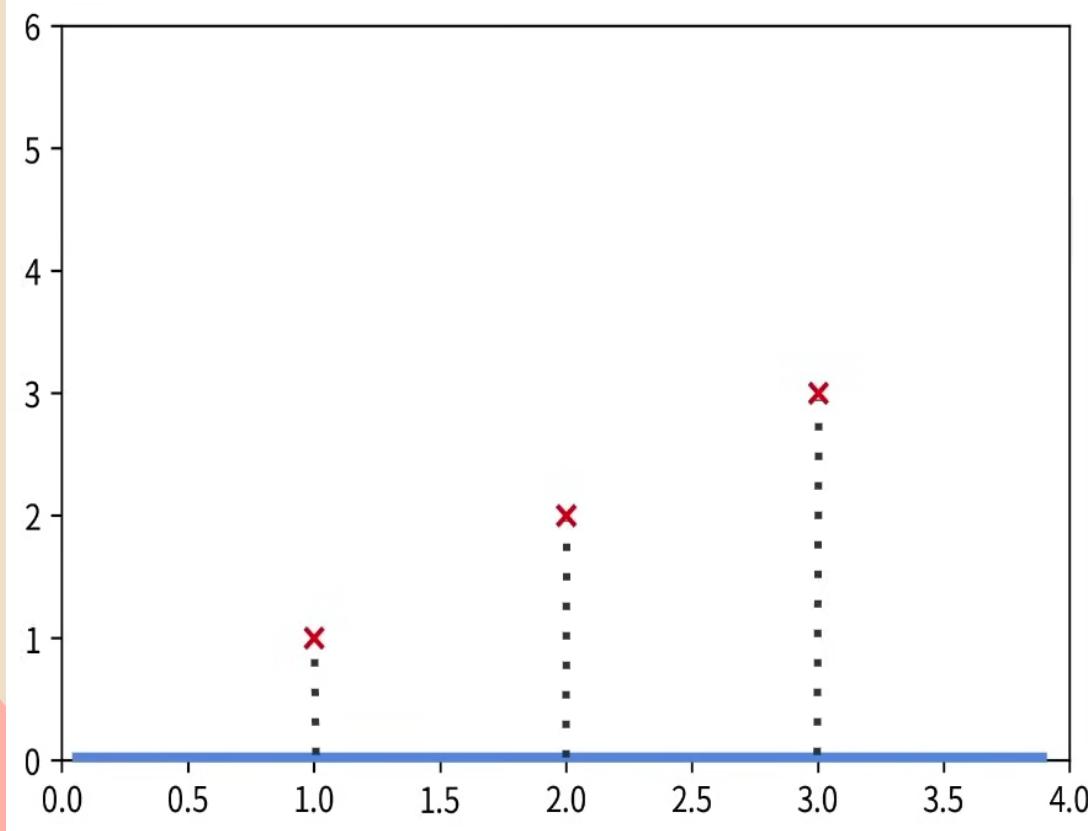
休息一下，馬上回來



如何找出 **最適合資料的直線**

# 如何找出 最適合資料的直線

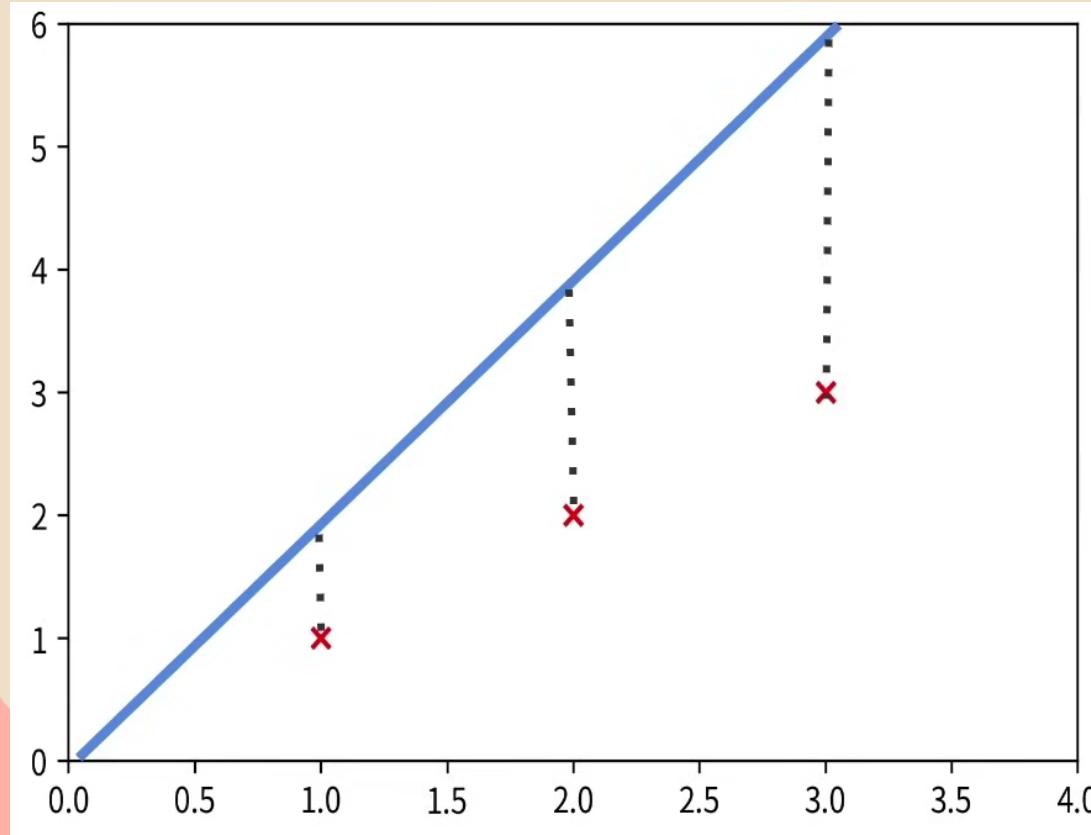
$$a = 0 \quad b = 0$$



$$(1 - 0)^2 + (2 - 0)^2 + (3 - 0)^2 = 14$$

# 如何找出 最適合資料的直線

$$a = 2 \quad b = 0$$

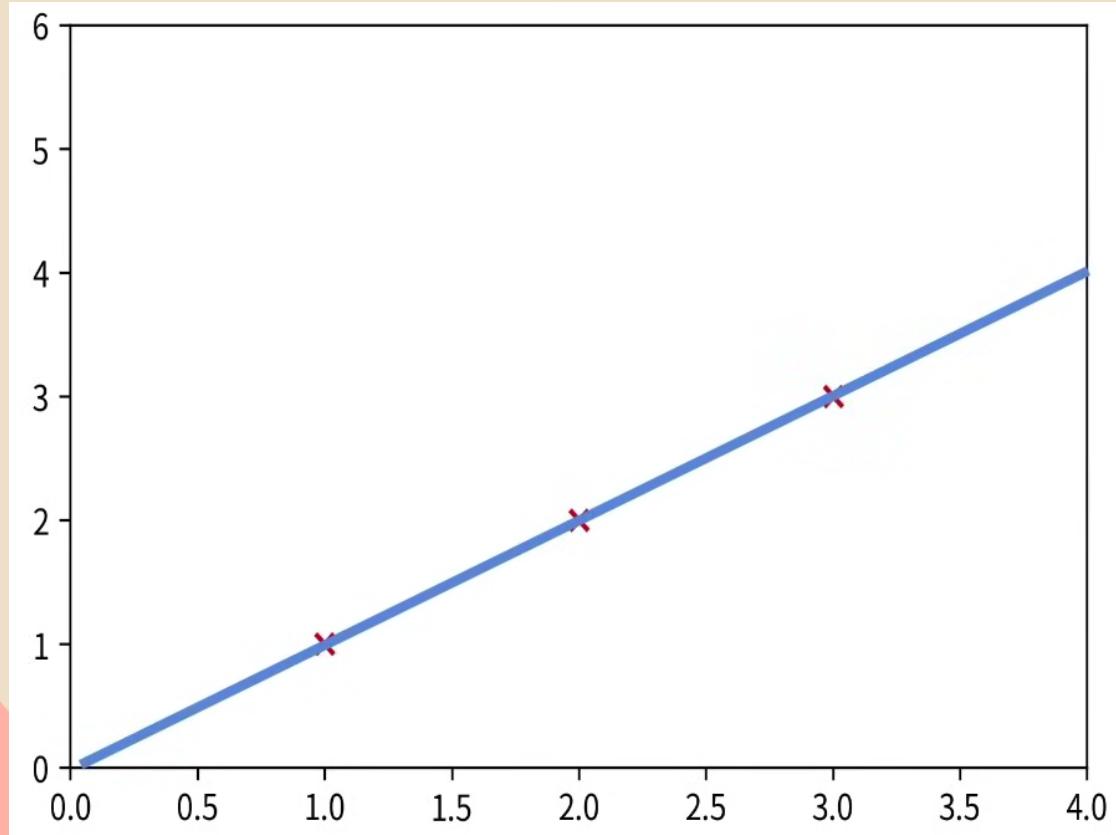


$$(1 - 0)^2 + (2 - 0)^2 + (3 - 0)^2 = 14$$

$$(1 - 2)^2 + (2 - 4)^2 + (3 - 6)^2 = 14$$

# 如何找出 最適合資料的直線

$$a = 1 \quad b = 0$$



$$(1 - 0)^2 + (2 - 0)^2 + (3 - 0)^2 = 14$$

$$(1 - 2)^2 + (2 - 4)^2 + (3 - 6)^2 = 14$$

$$(1 - 1)^2 + (2 - 2)^2 + (3 - 3)^2 = 0$$

# 如何找出 **最適合資料的直線**

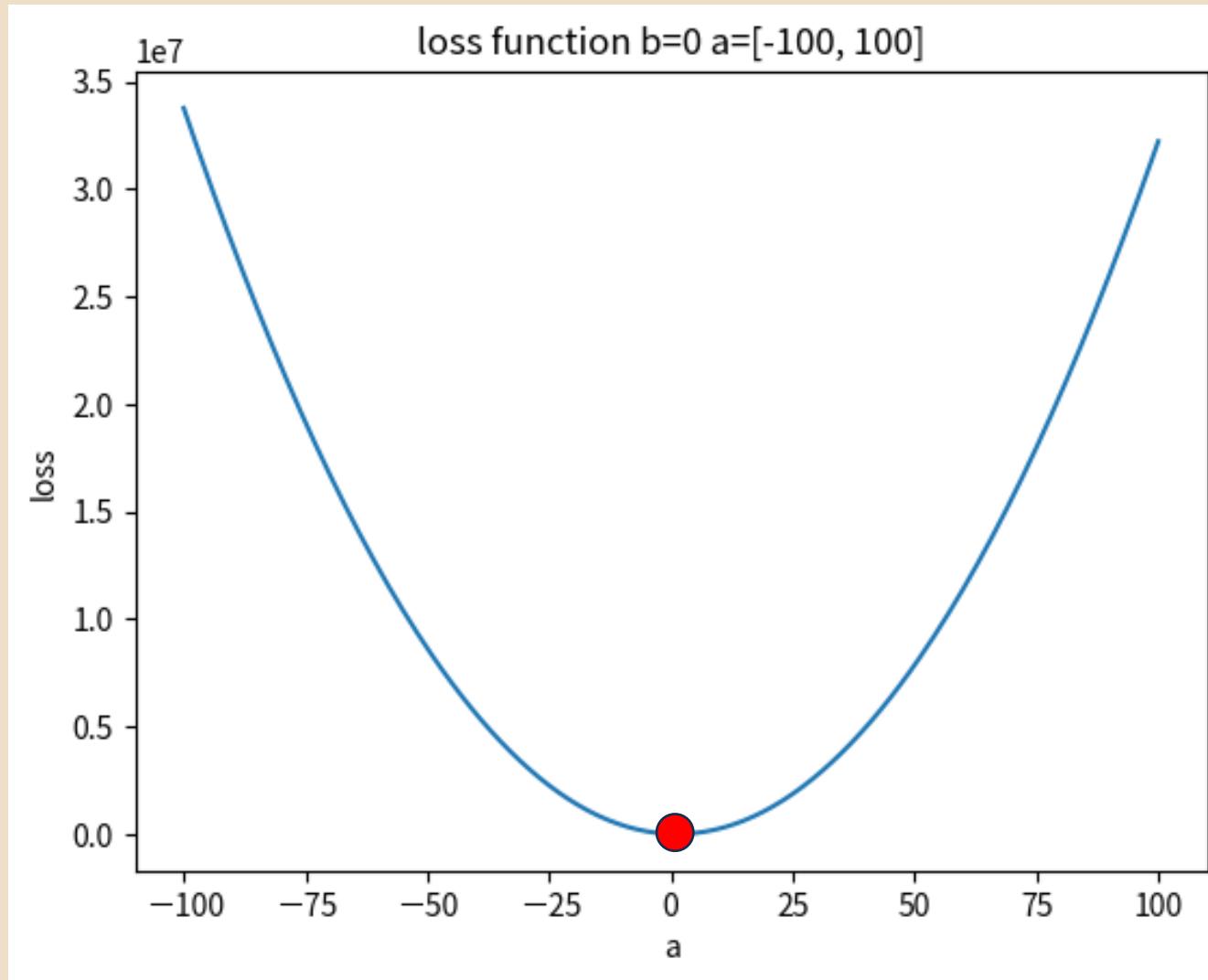
Loss function = (真實數據 – 預測值)<sup>2</sup>

$$(1 - 0)^2 + (2 - 0)^2 + (3 - 0)^2 = 14$$

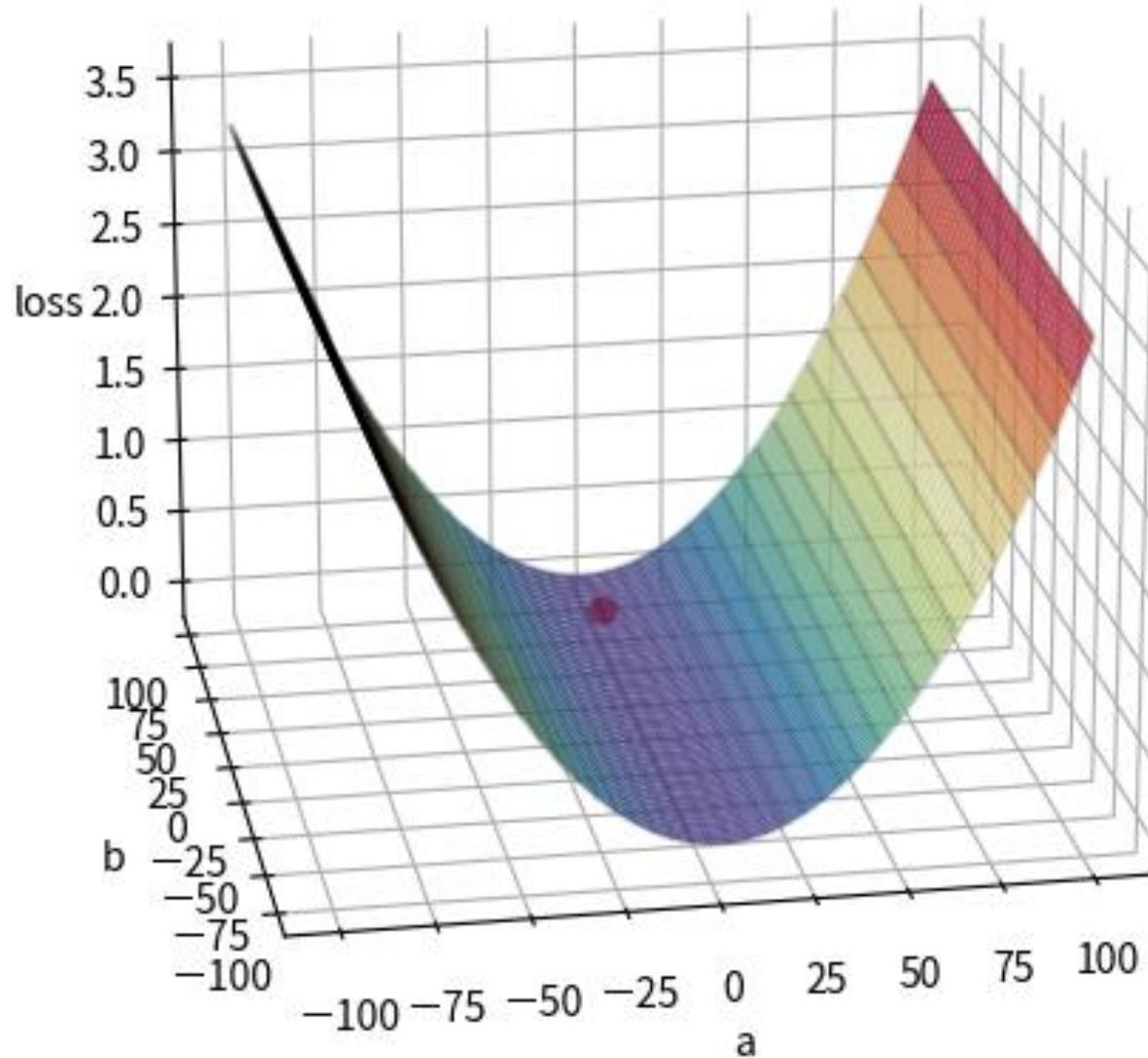
$$(1 - 2)^2 + (2 - 4)^2 + (3 - 6)^2 = 14$$

$$(1 - 1)^2 + (2 - 2)^2 + (3 - 3)^2 = 0$$

# 如何找出 最適合資料的直線



# 如何找出 **最適合資料的直線**



# 1. 讀入資料

```
1 import pandas as pd  
2  
3 # TODO : 貼上網址抓資料  
4 url = ".../chinese_science_score.csv"  
5 data = pd.read_csv(url)
```

## 2. 計算 loss function

```
● ● ●  
1 x = data['chinese']  
2 y = data['science']  
3  
4 #####  
5  
6 a = 0  
7 b = 0  
8  
9 y_pred = a*x + b  
10 # TODO: 設定 loss function  
11 loss = (y - y_pred)**2  
12  
13 # TODO: 計算平均  
14 loss.sum() / len(x)
```

## 2. 計算 loss function



```
1 def compute_loss(x, y, a, b):  
2     y_pred = a*x + b  
3     loss = (y - y_pred)**2  
4     loss = loss.sum( ) / len(x)  
5     return loss  
6  
7 compute_loss(x, y, 0, 0)
```

### 3. 固定 b 值，觀察 loss

```
1 loss_all = []
2
3 for a in range(-100, 101):
4     loss = compute_loss(x, y, a, 0)
5     loss_all.append(loss)
6
7 loss_all
```

### 3. 固定 b 值，觀察 loss

```
1 import matplotlib.pyplot as plt  
2  
3 # TODO：將所有點連起來  
4 plt.plot(range(-100, 101), loss_all)  
5  
6 plt.title("loss function a=[-100, 100], b=0")  
7 plt.xlabel("a")  
8 plt.ylabel("loss")  
9  
10 plt.show()
```

# 4. 窺舉所有 a, b 值的 loss

```
1 import numpy as np
2
3 a_all = np.arange(-100, 101)
4 b_all = np.arange(-100, 101)
5 loss_all = np.zeros((201, 201))
6
7 i = 0
8 for a in a_all:
9     j = 0
10    for b in b_all:
11        loss = compute_loss(x, y, a, b)
12        loss_all[ i, j ] = loss
13
14        j = j+1
15    i = i+1
```

# 4. 窺舉所有 a, b 值的 loss

```
1 plt.figure(figsize=(6, 6)) # 圖片大小
2
3 ax = plt.axes(projection = "3d")
4 ax.view_init(20, -100) # 查看視角
5 ax.xaxis.set_pane_color((1, 1, 1)) # 背景顏色
6 ax.yaxis.set_pane_color((1, 1, 1))
7 ax.zaxis.set_pane_color((1, 1, 1))
8
9 b_grid, a_grid = np.meshgrid(b_all, a_all) # 網格化
10 ##### https://wangyeming.github.io/2018/11/12/numpy-meshgrid/
11 ax.plot_surface(a_grid, b_grid, loss_all, cmap="Spectral_r", alpha=0.8)
12 ax.plot_wireframe(a_grid, b_grid, loss_all, color="black", alpha=0.15)
13
14 # 設定資訊
15 ax.set_title("a b 對應的 loss")
16 ax.set_xlabel("a")
17 ax.set_ylabel("b")
18 ax.set_zlabel("loss")
19
20 plt.show()
```

# 4. 窺舉所有 a, b 值的 loss



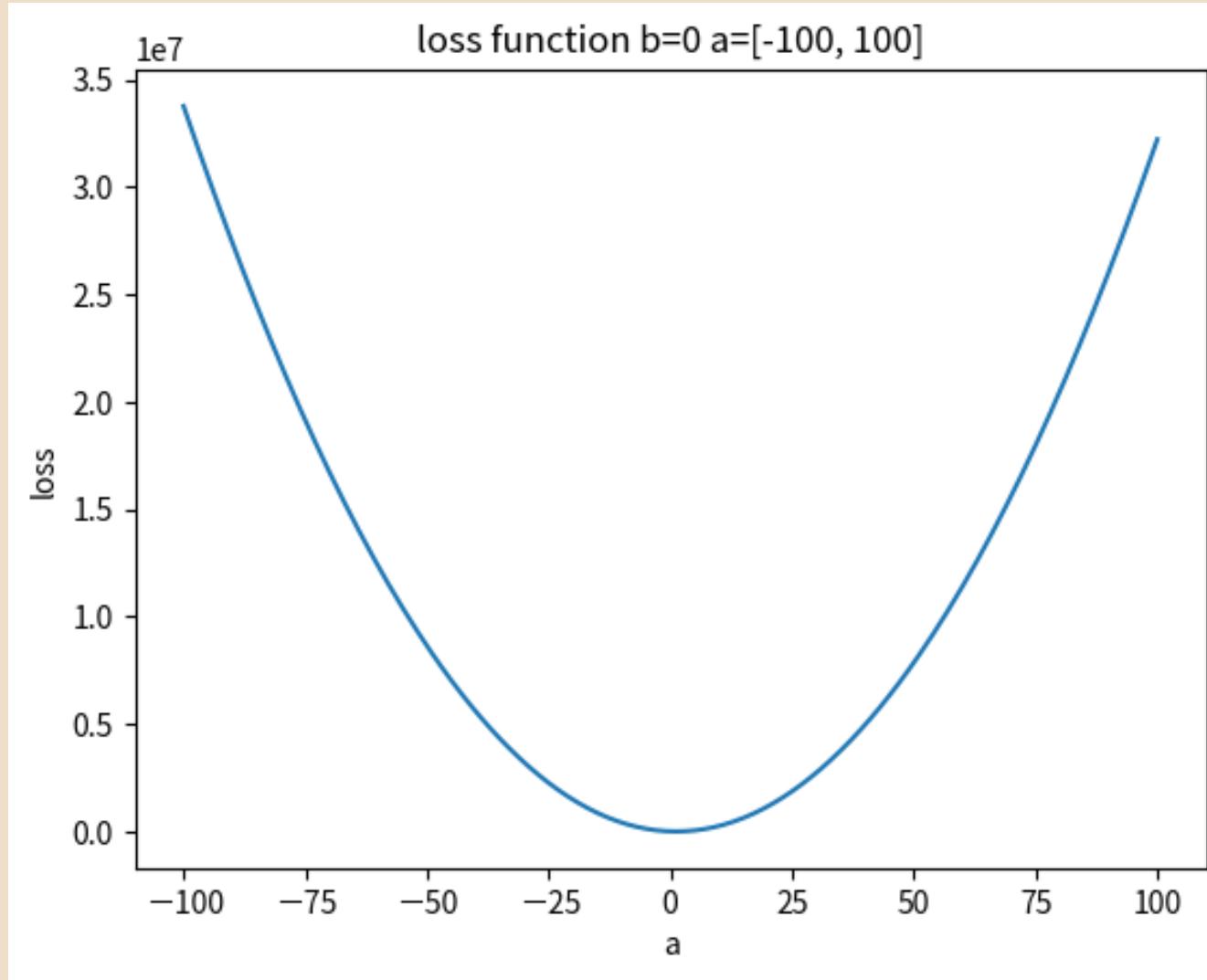
```
1 # TODO : 尋找 loss 最少的點
2 a_index, b_index = np.where(loss_all == np.min(loss_all))
3 print(a_index, b_index, a_all[a_index], b_all[b_index])
4 ax.scatter(a_all[a_index], b_all[b_index], loss_all[a_index, b_index], color="red", s=40)
```

如何 有效率 找出最佳的  $a$   $b$

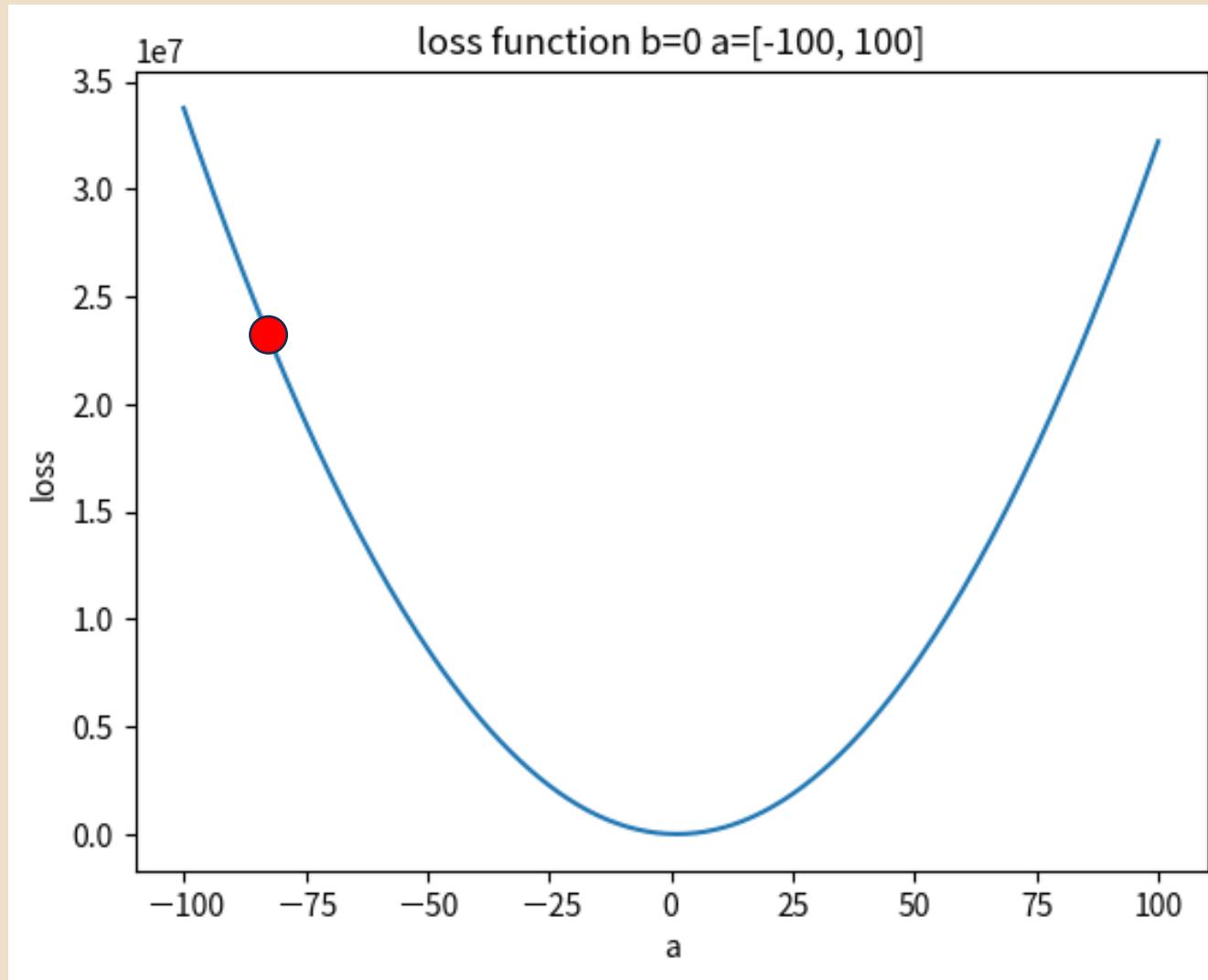
# 如何 **有效率** 找出最佳的 $a$ $b$

讓我們實體試試看

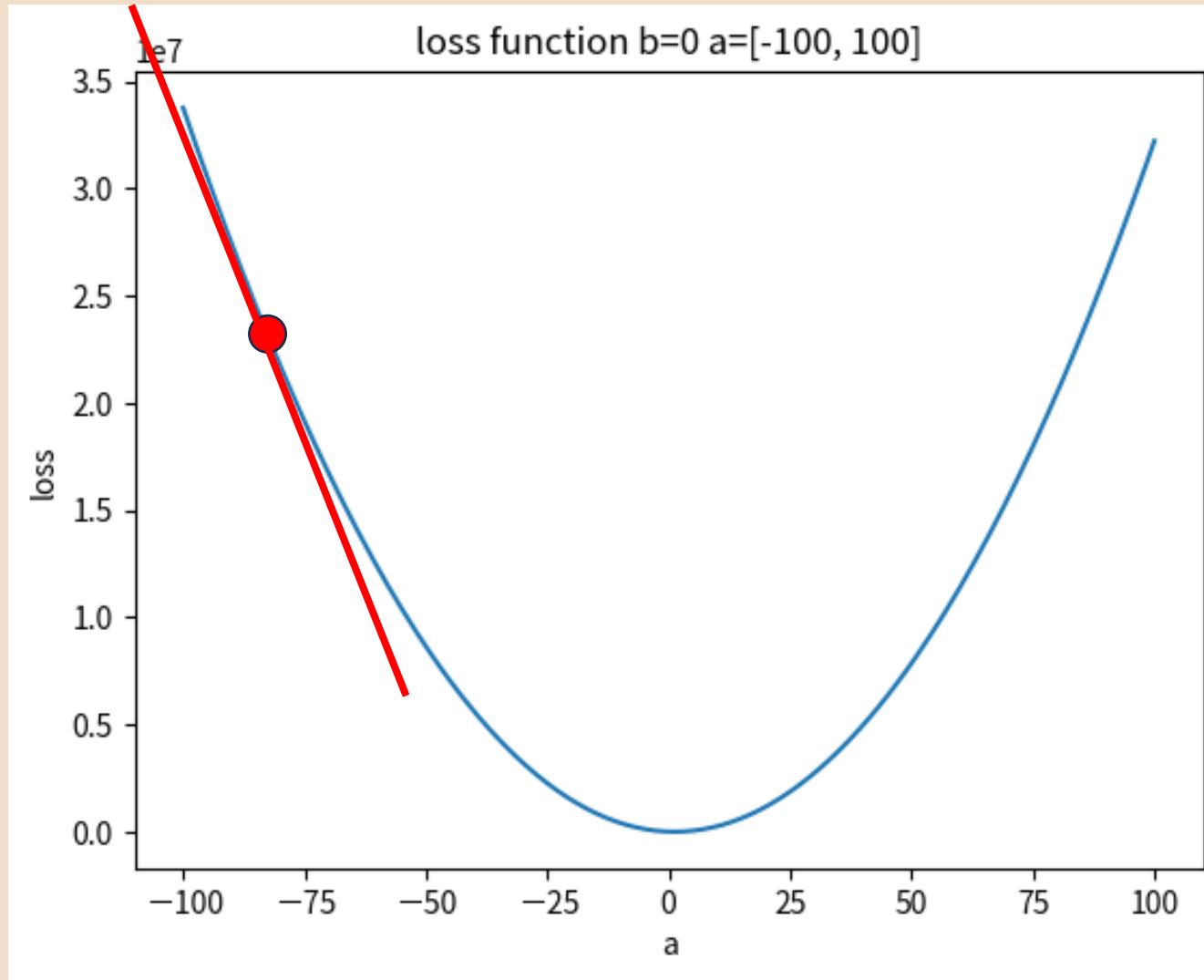
# 如何 有效率 找出最佳的 $a$ $b$



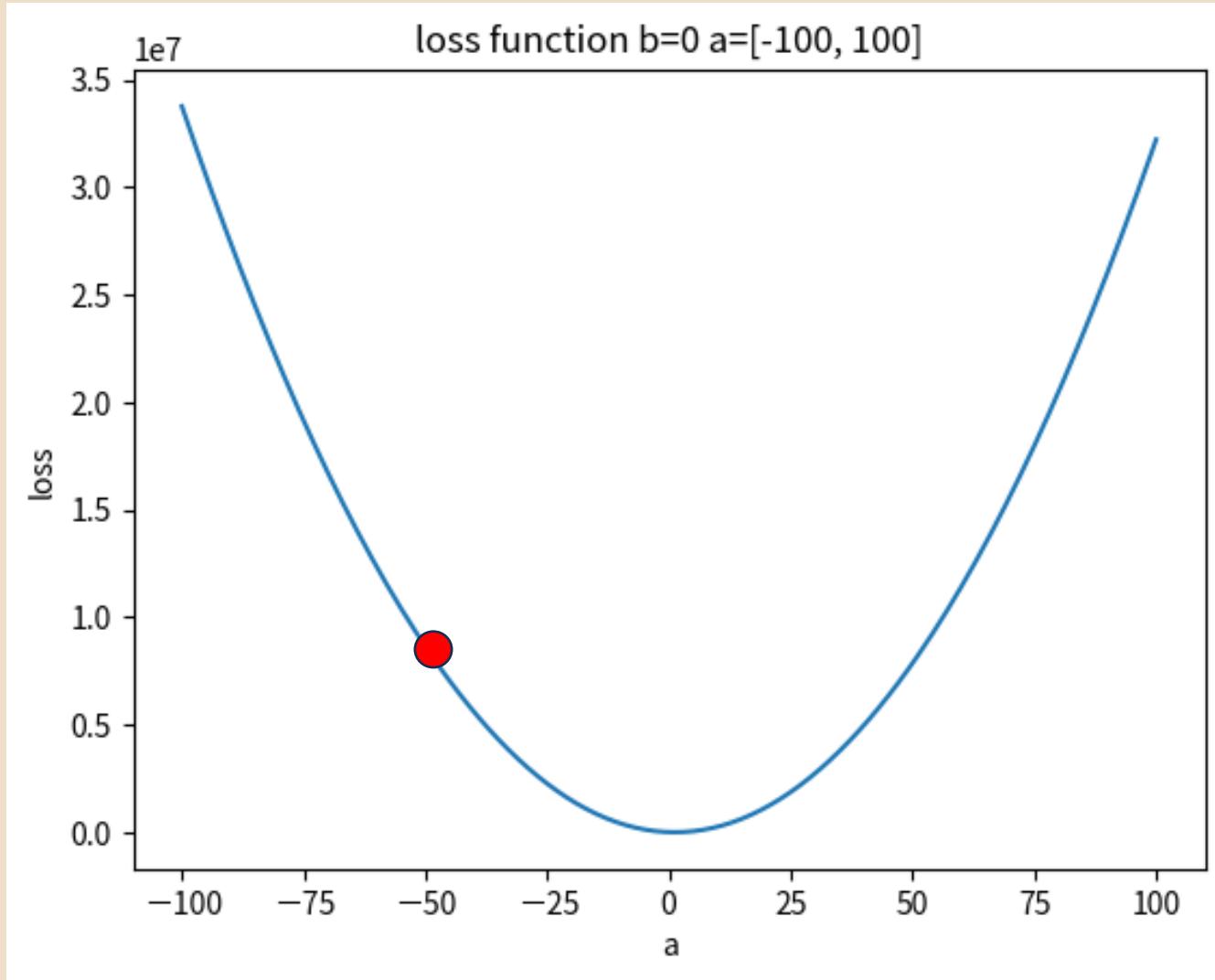
# 如何 有效率 找出最佳的 $a$ $b$



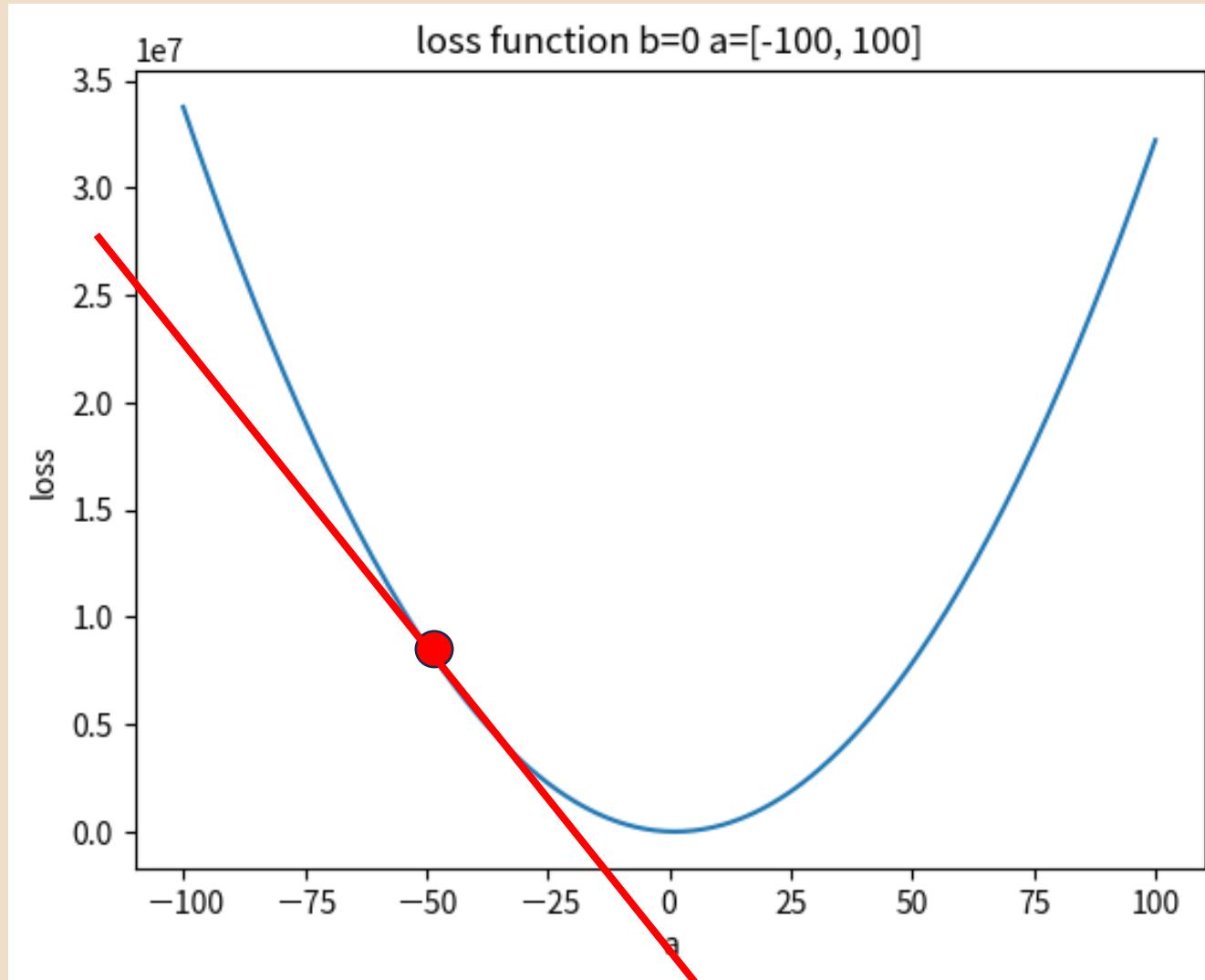
# 如何 有效率 找出最佳的 $a$ $b$



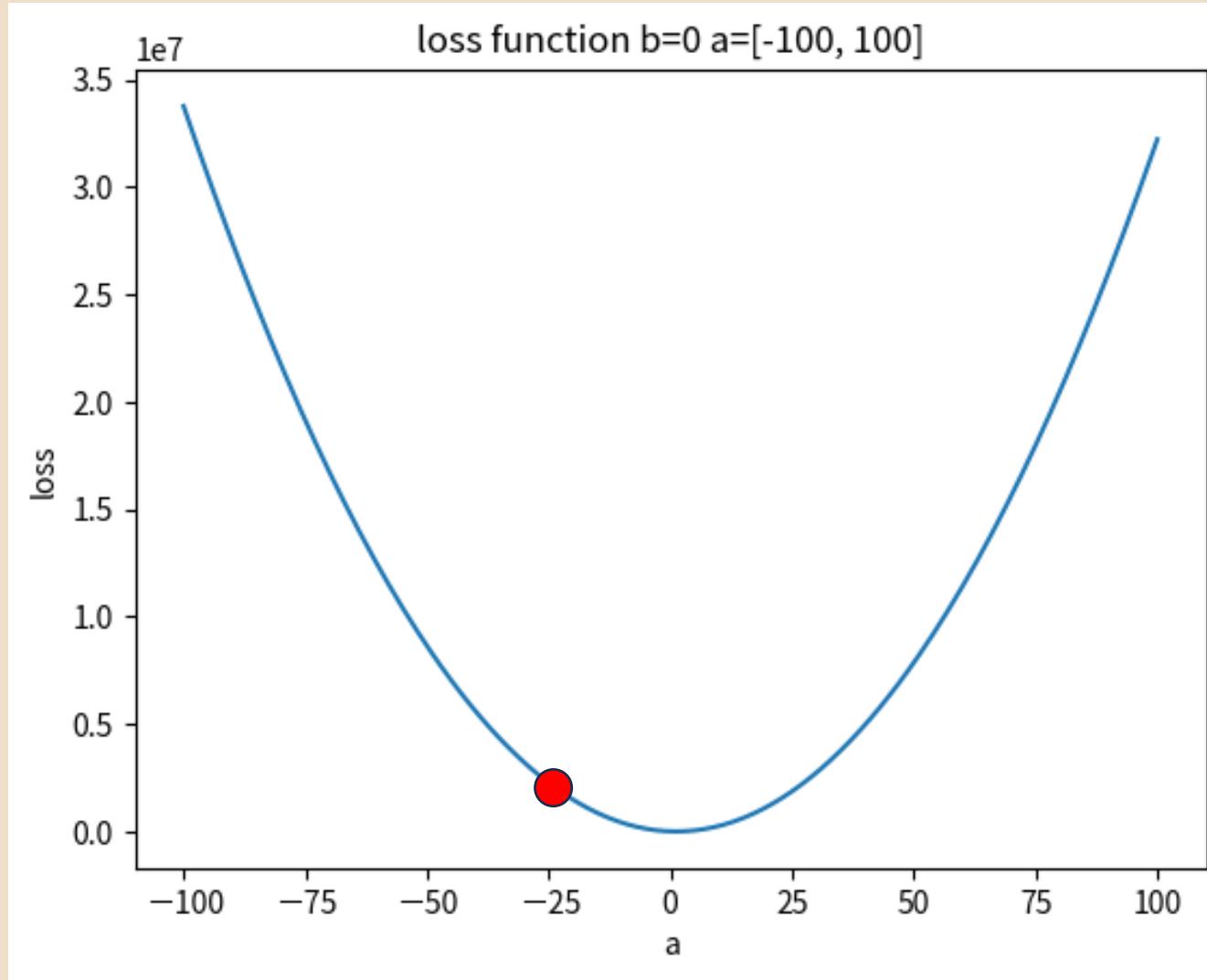
# 如何 有效率 找出最佳的 $a$ $b$



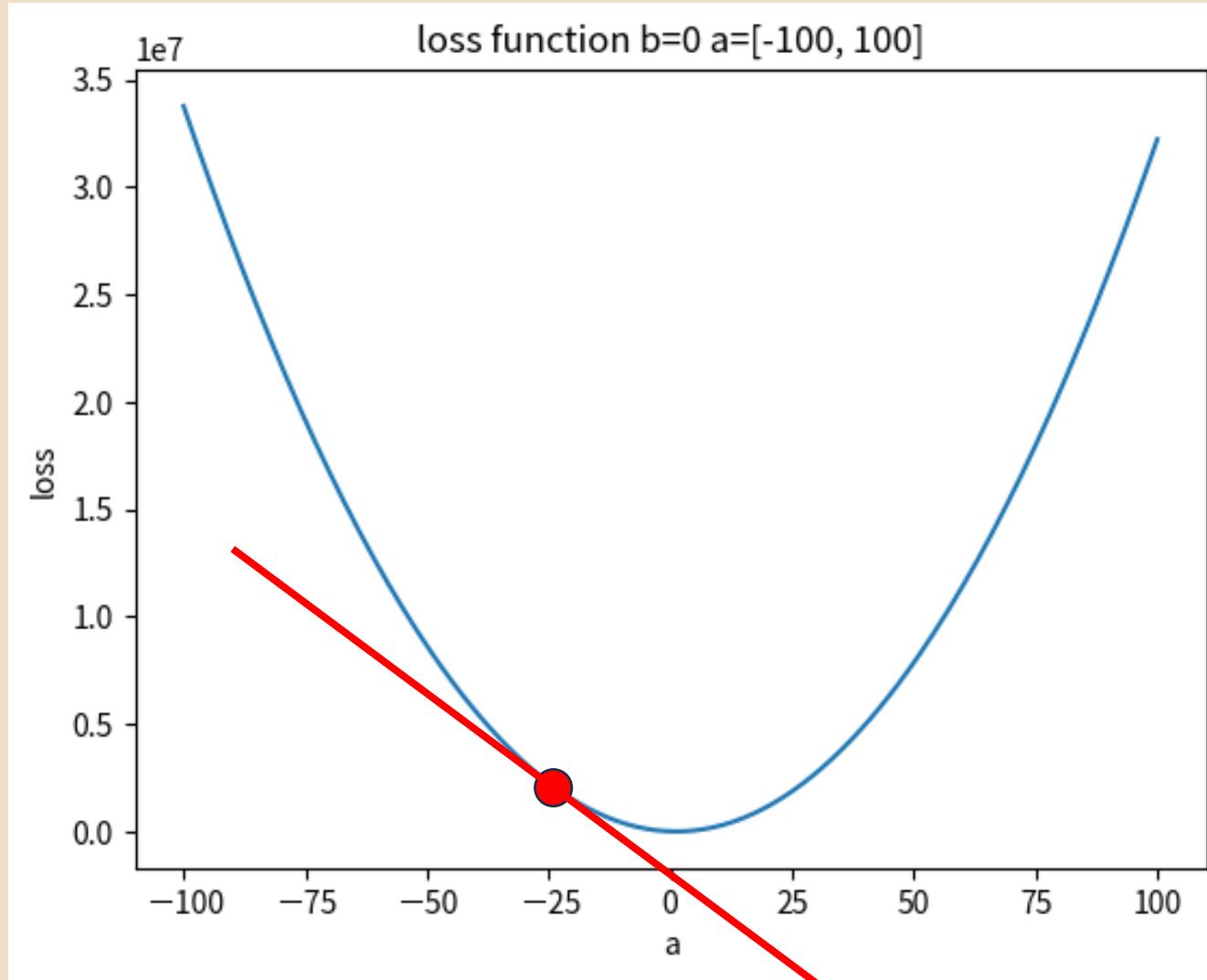
# 如何 有效率 找出最佳的 $a$ $b$



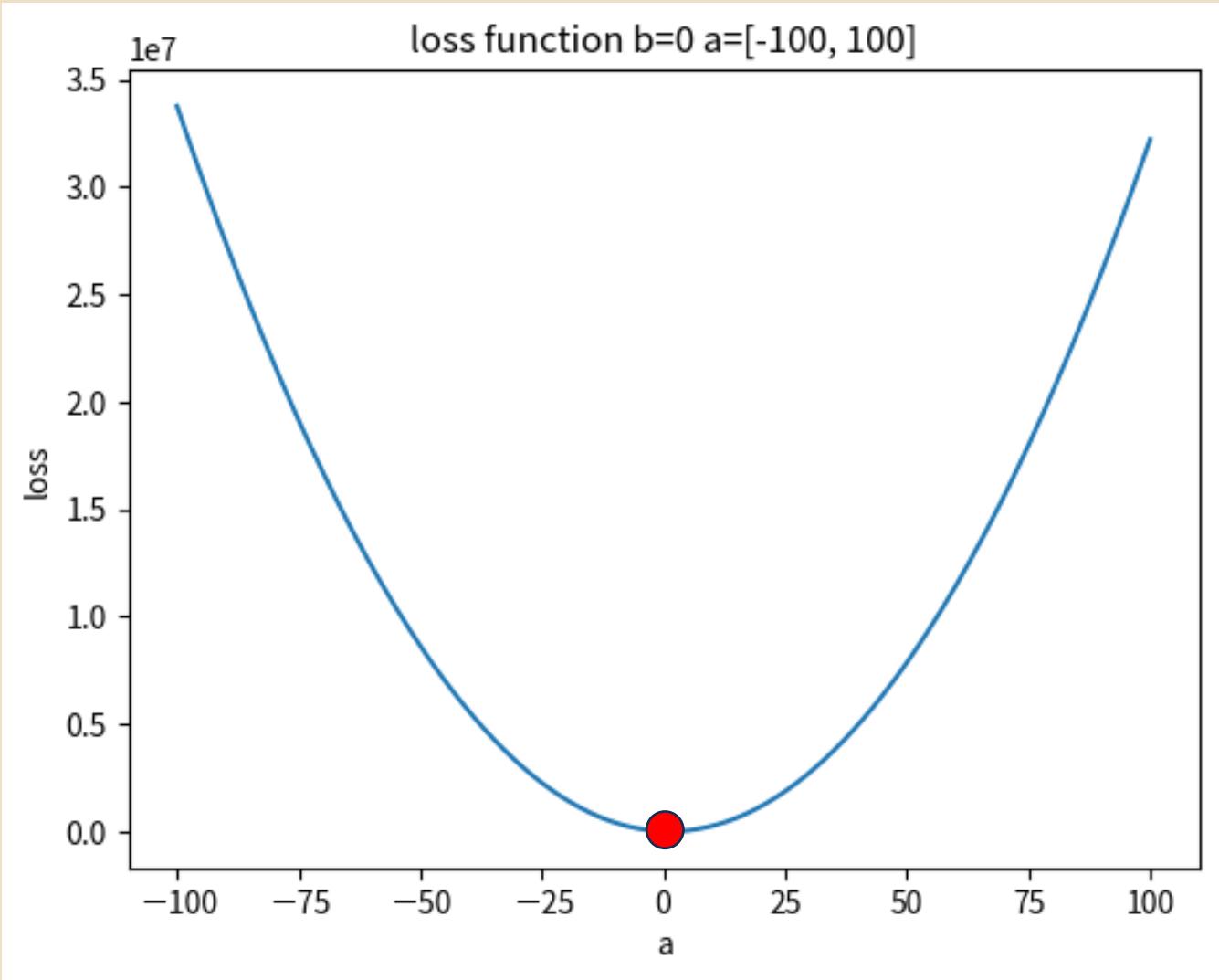
# 如何 有效率 找出最佳的 $a$ $b$



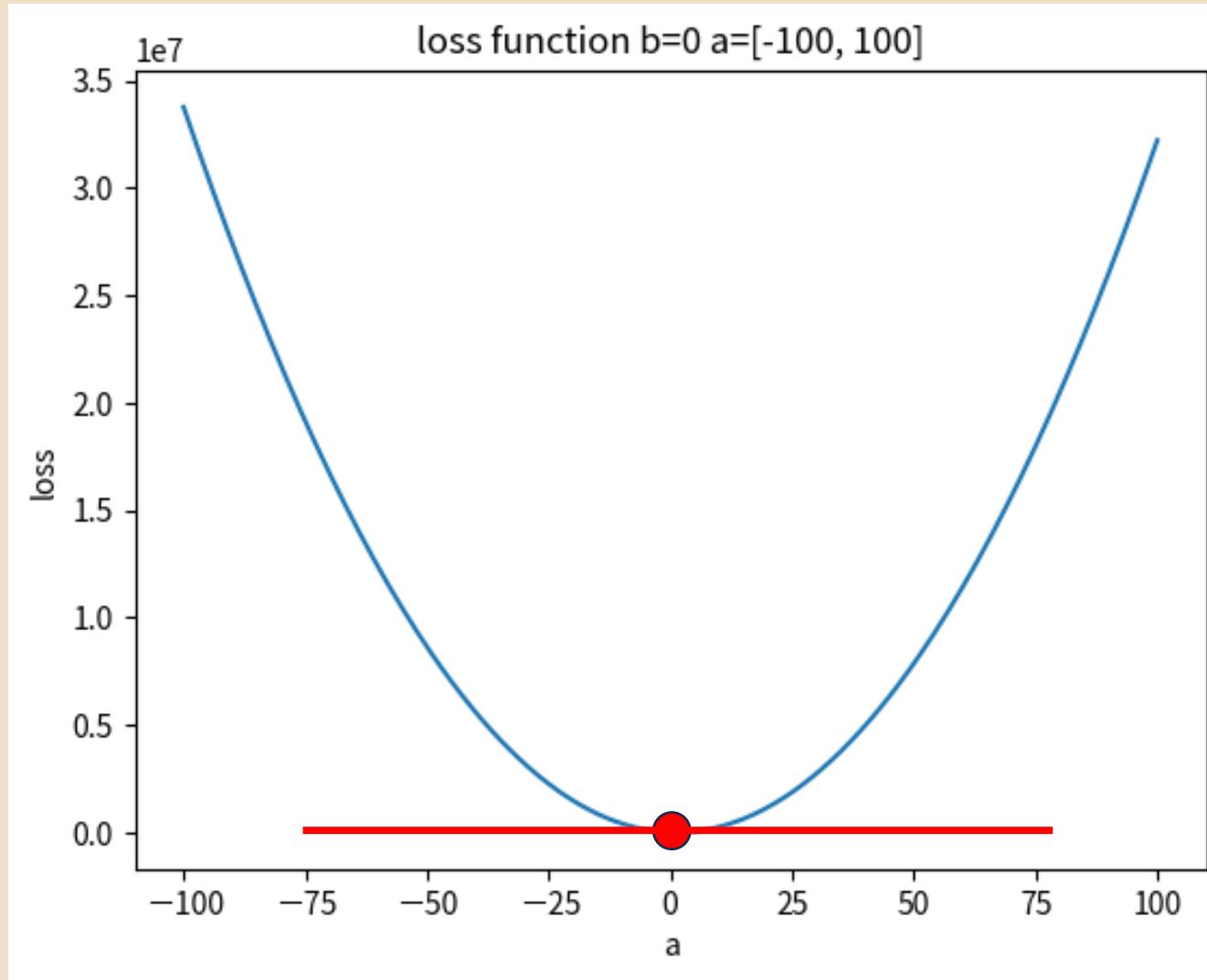
# 如何 有效率 找出最佳的 $a$ $b$



# 如何 有效率 找出最佳的 $a$ $b$



# 如何 有效率 找出最佳的 $a$ $b$

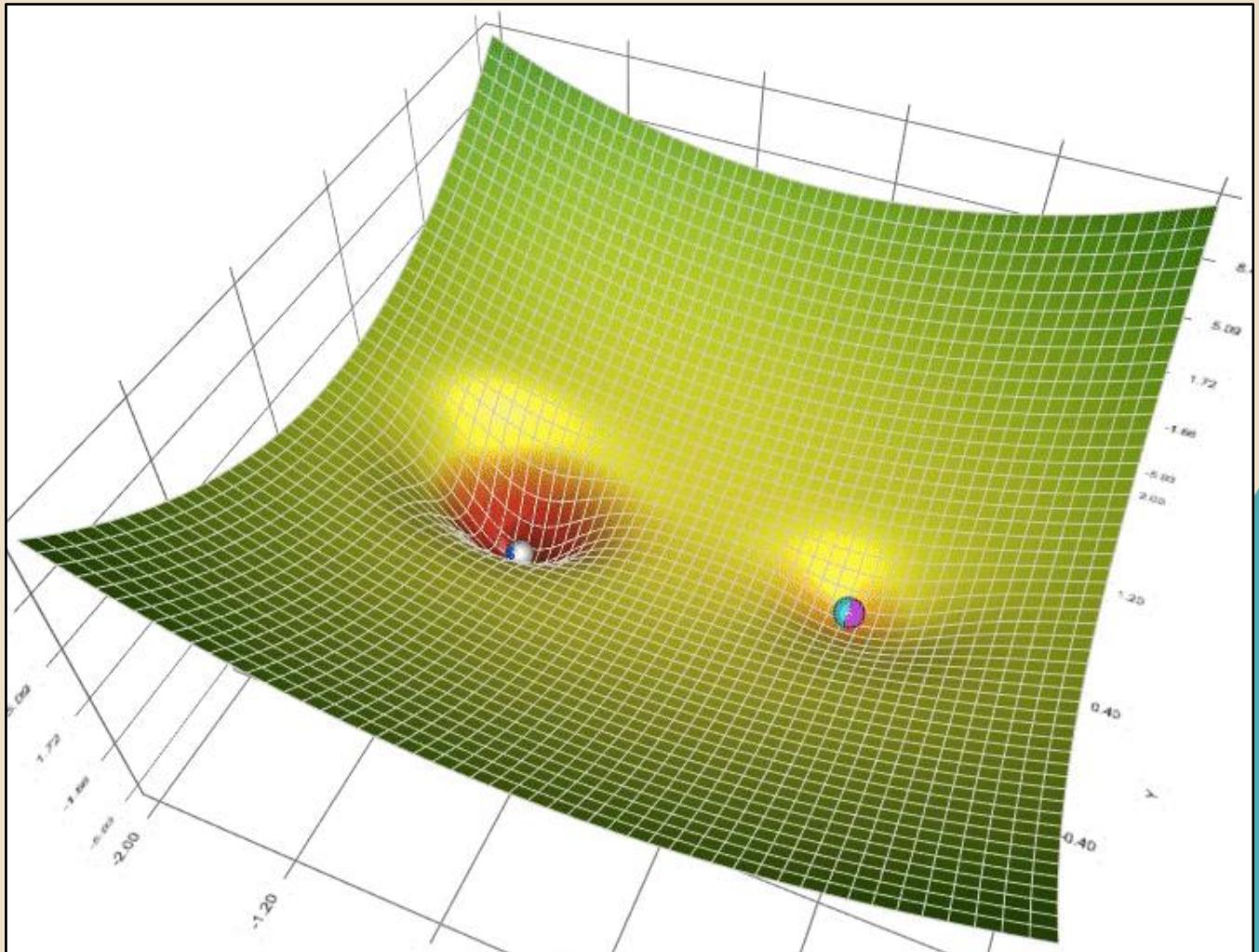


# 如何 **有效率** 找出最佳的 $a$ $b$

梯度下降演算法

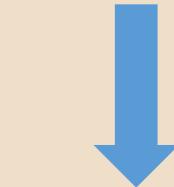
Gradient Descent Algorithm

根據斜率改變參數

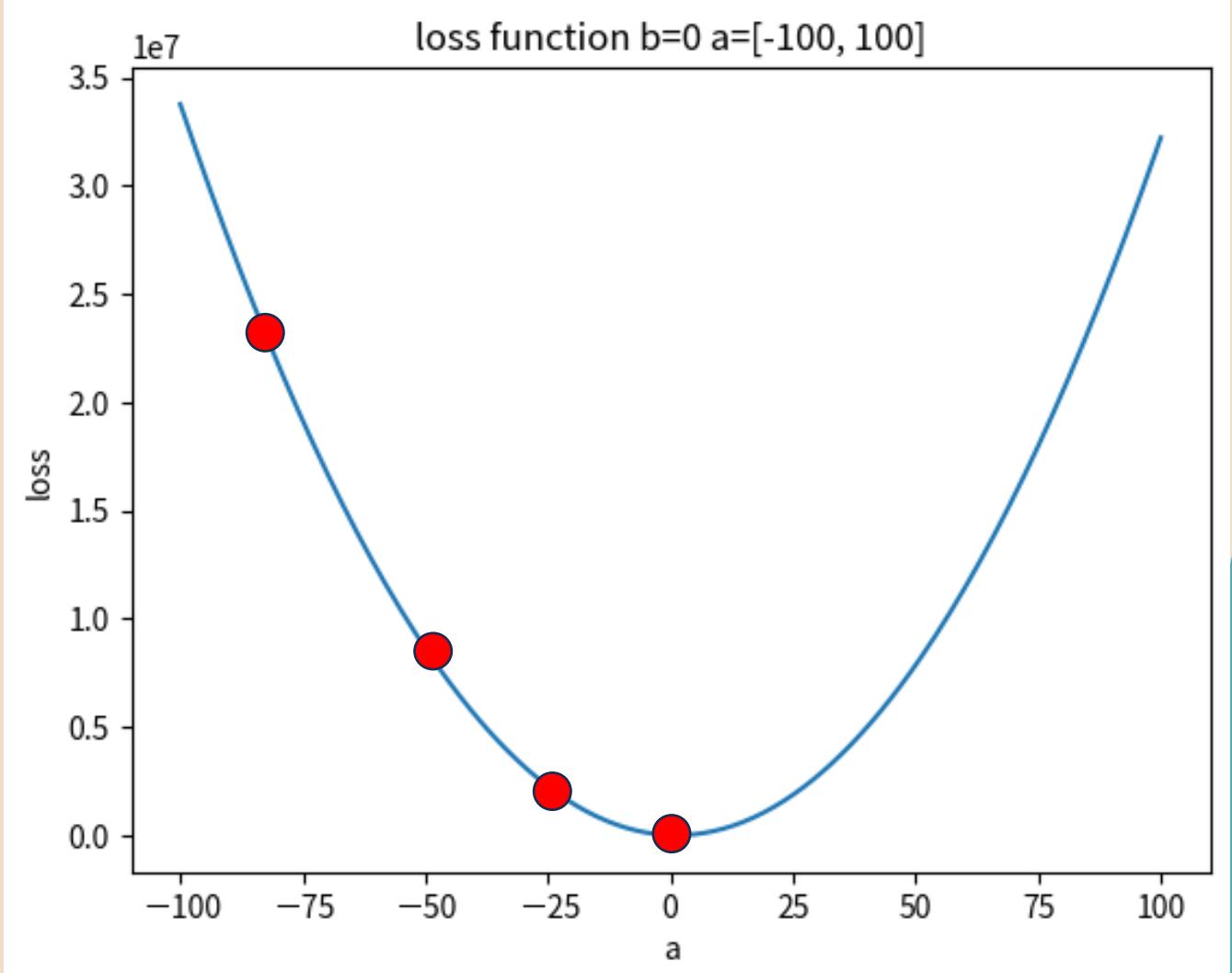


# 如何 **有效率** 找出最佳的 $a$ $b$

$a - \text{斜率} \times \text{學習率}$



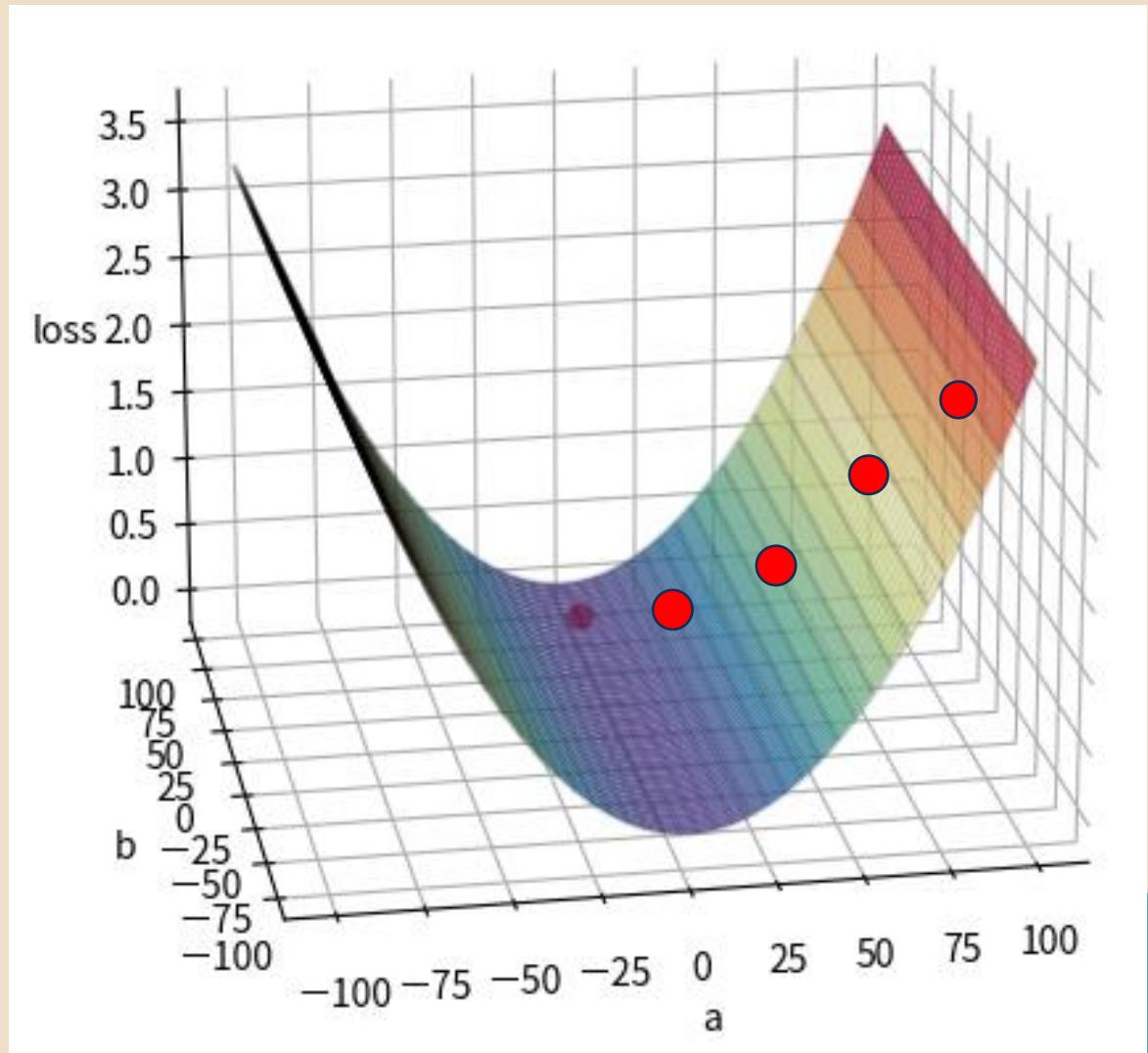
步伐大小



# 如何 **有效率** 找出最佳的 $a$ $b$

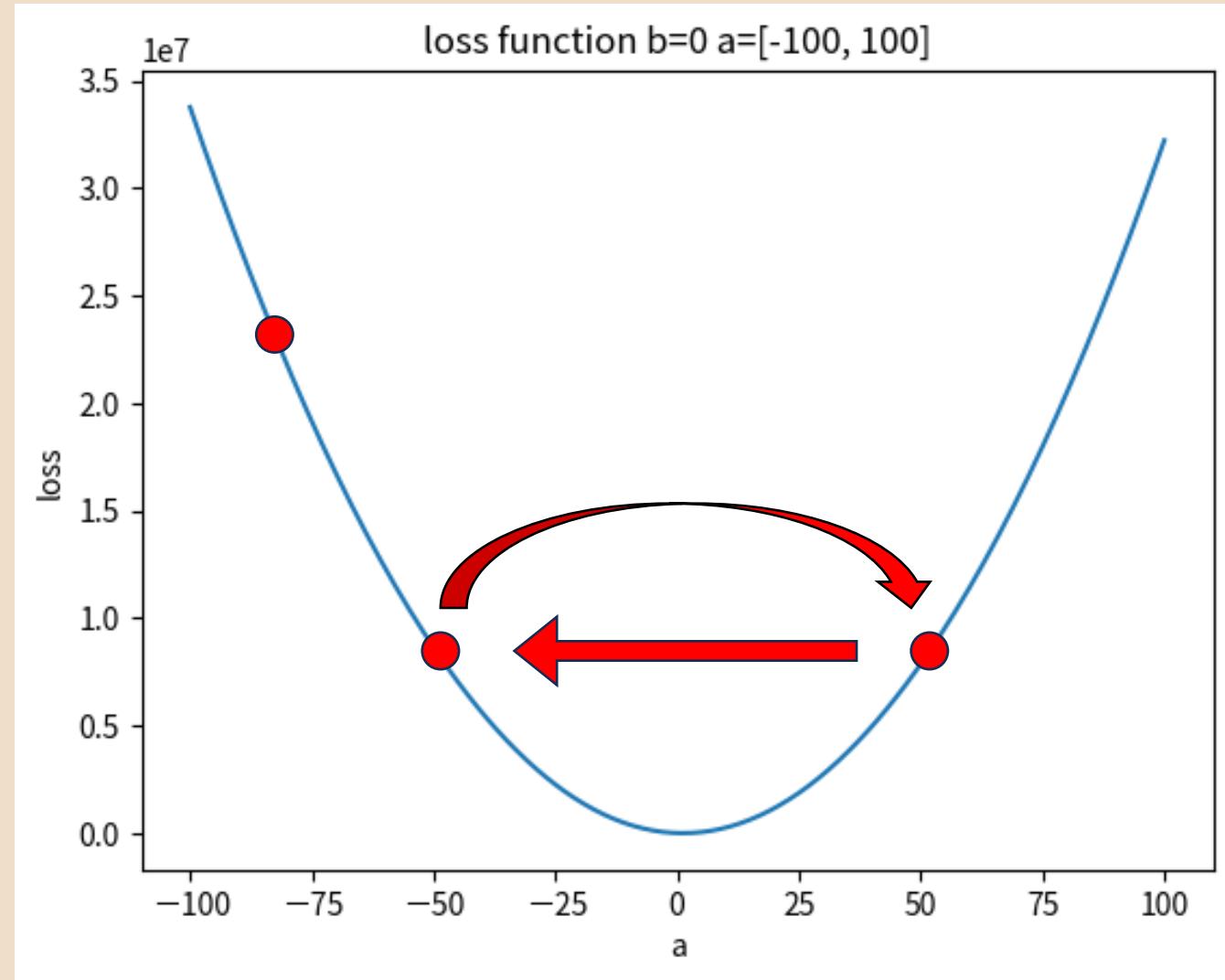
$a$  - 斜率  $\times$  學習率

$b$  - 斜率  $\times$  學習率



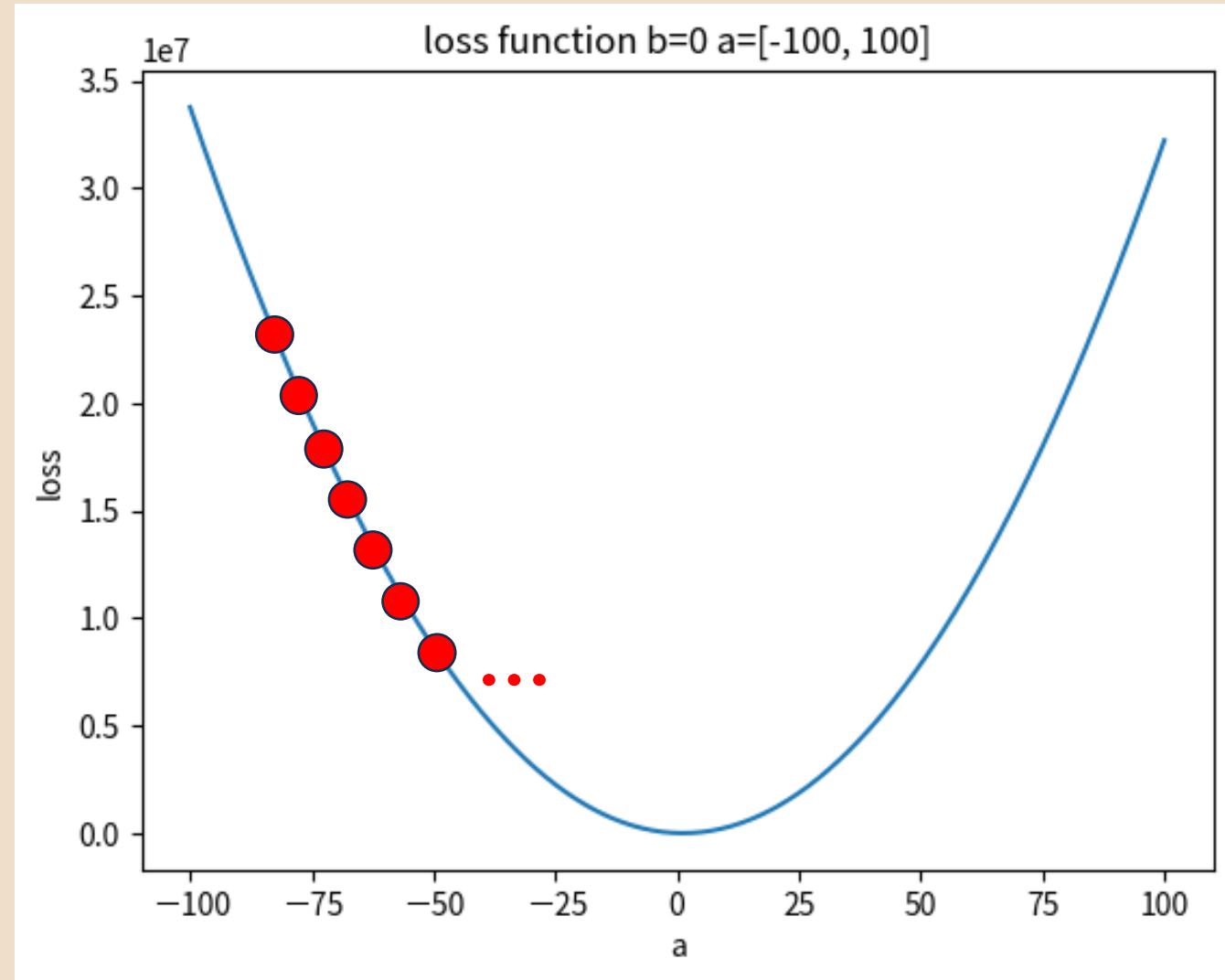
# 學習率的 步伐大小 有差嗎？

學習率過 大



# 學習率的 步伐大小 有差嗎？

學習率過 小



# 1. 讀入資料

```
1 import pandas as pd  
2  
3 # TODO : 貼上網址抓資料  
4 url = ".../chinese_science_score.csv"  
5 data = pd.read_csv(url)
```

## 2. 計算 gradient descent

```
1 a = 100
2 b = 100
3
4 a_gradient = 2 * x * (a * x + b - y)
5 b_gradient = 2 * (a * x + b - y)
6
7 # TODO : 計算平均結果
8 a_gradient = (2 * x * (a * x + b - y)).mean()
9 b_gradient = (2 * (a * x + b - y)).mean()
```

## 2. 計算 gradient descent



```
1 a = 100
2 b = 100
3 a_gradient, b_gradient = compute_gradient(x, y, a, b)
4
5 learning_rate = 0.0000001
6
7 # TODO: 移動 a, b 的數值 (learning rate)
8 a = a - a_gradient * learning_rate
9 b = b - b_gradient * learning_rate
```

### 3. 觀察 loss 值是否有下降

```
1 a = 0
2 b = 0
3
4 # TODO: 印出 loss 值
5 print( compute_loss(x, y, a, b) )
6
7 a_gradient, b_gradient = compute_gradient(x, y, a, b)
8
9 learning_rate = 0.0000001
10 a = a - a_gradient * learning_rate
11 b = b - b_gradient * learning_rate
12
13 # TODO: 印出 loss 值
14 print( compute_loss(x, y, a, b) )
```

# 4. 畫圖觀察 loss 趨勢

```
1 import matplotlib.pyplot as plt  
2 import numpy as np  
3  
4 # TODO : 畫出所有 loss_hist 值  
5 plt.plot(np.arange(0, run_iter), loss_hist)  
6  
7 plt.title("次數 vs loss")  
8 plt.xlabel("次數")  
9 plt.ylabel("loss")  
10  
11 plt.show()
```

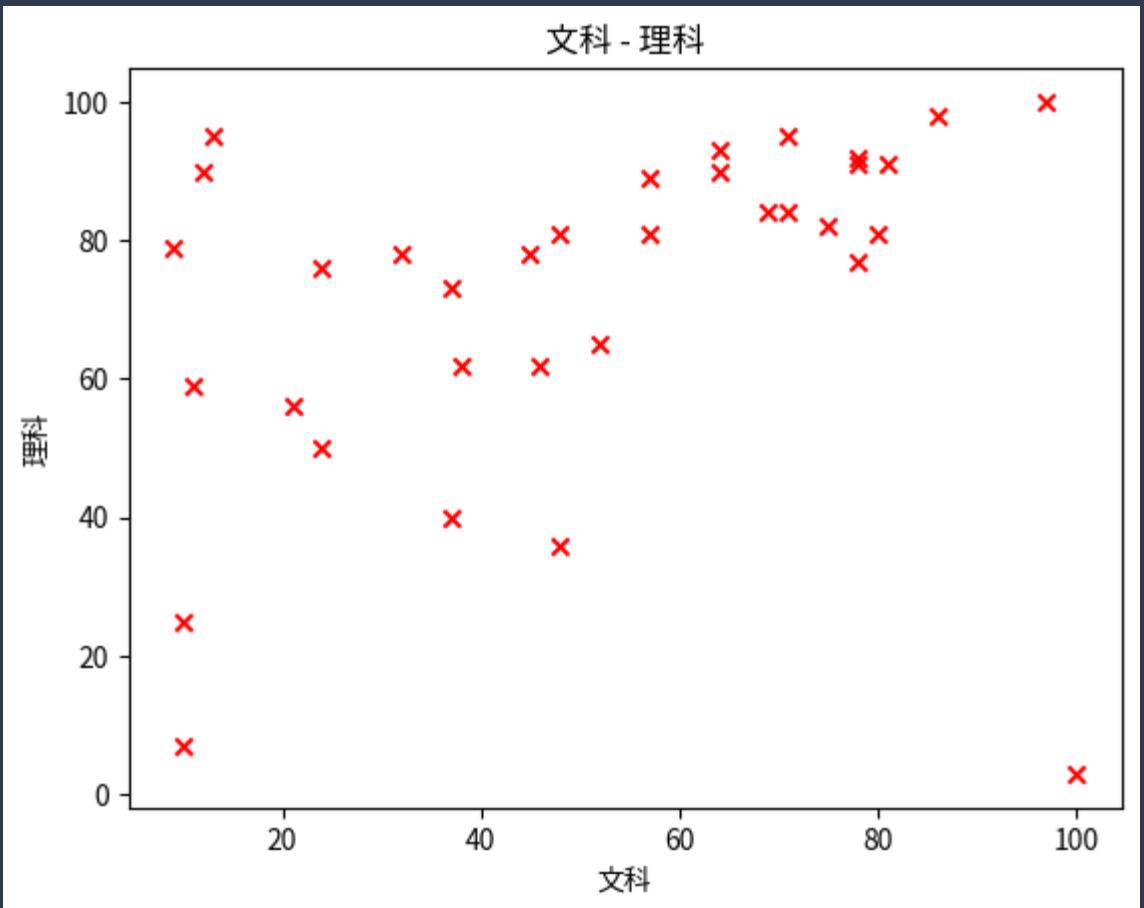
# 5. 畫圖觀察 3D 移動軌跡



```
1 # TODO : 劃出軌跡  
2 ax.scatter(a_hist[0], b_hist[0], loss_hist[0], color="green", s=40)  
3 ax.plot(a_hist, b_hist, loss_hist)
```

# 總結機器學習過程

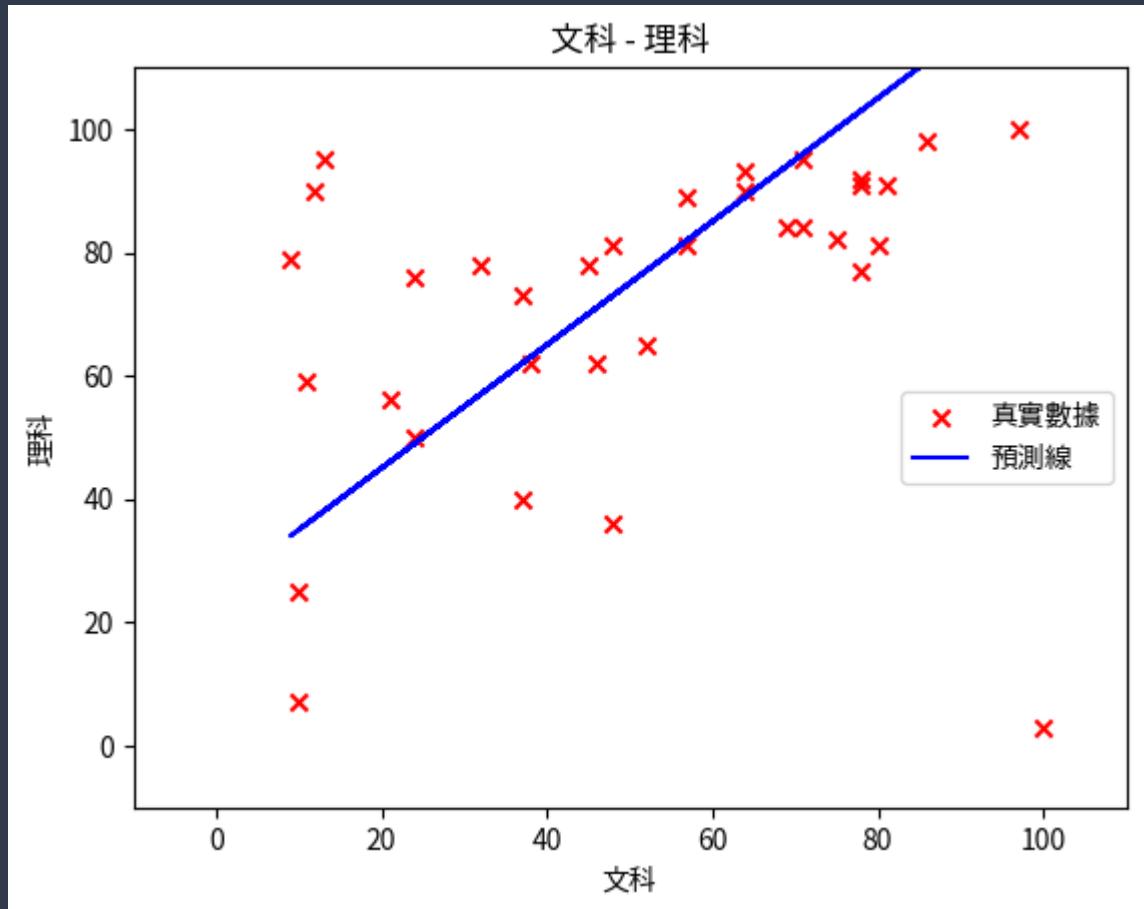
# 總結機器學習過程



準〇〇料

# 總結機器學習過程

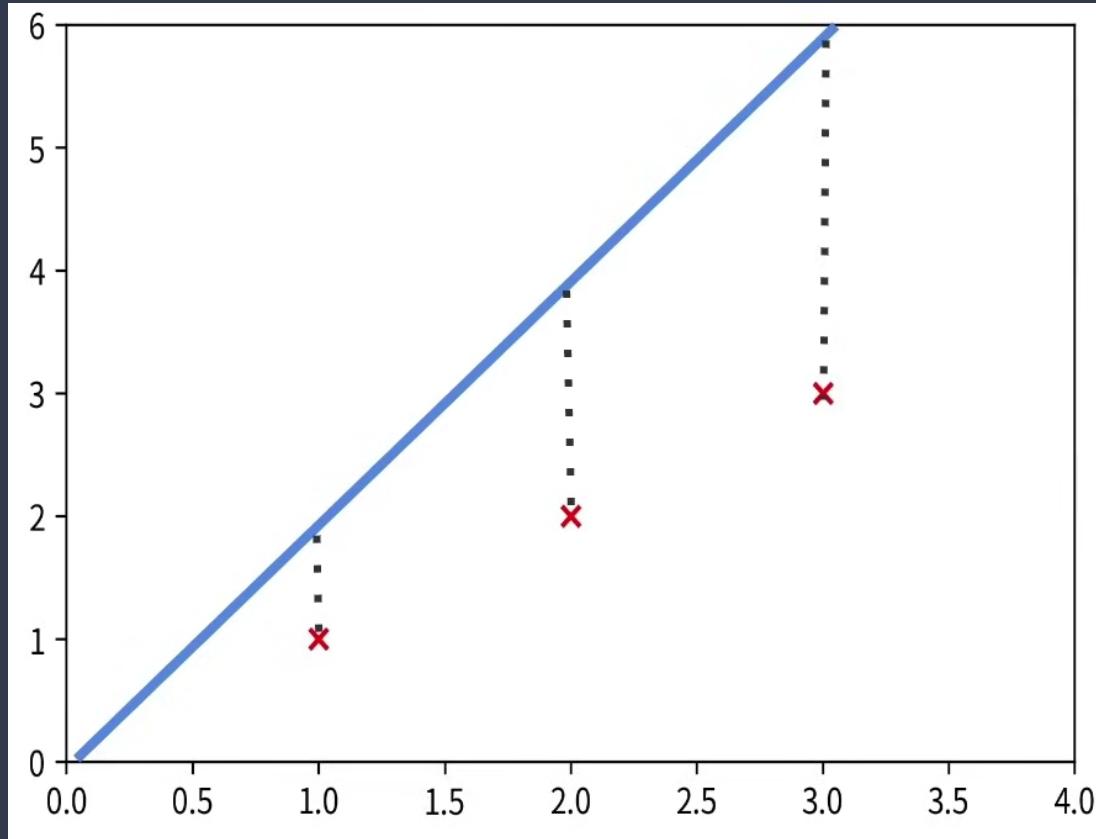
## 1. 準備資料



設定模○

# 總結機器學習過程

(真實數據 – 預測值)<sup>2</sup>

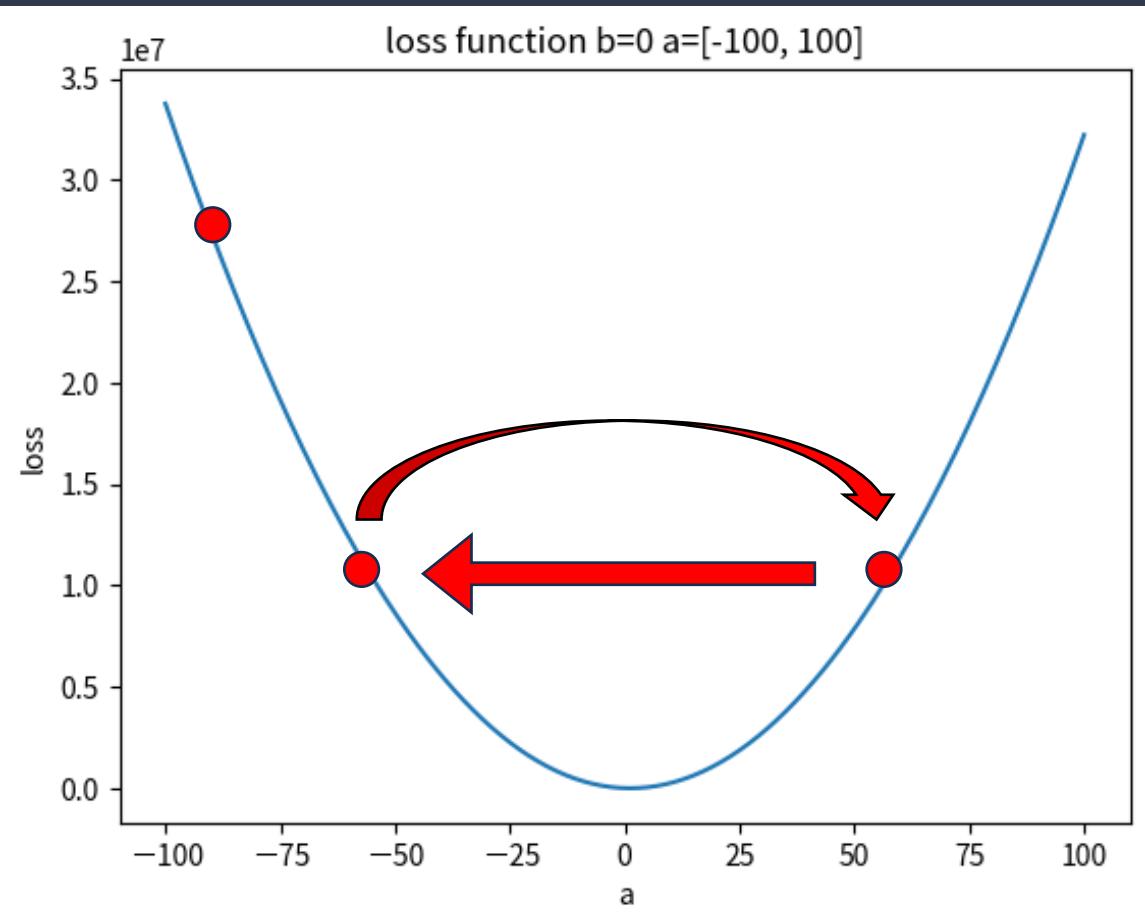


1. 準備資料

2. 設定模型

設定 ? Function

# 總結機器學習過程



1. 準備資料

2. 設定模型

設定 ? Rate

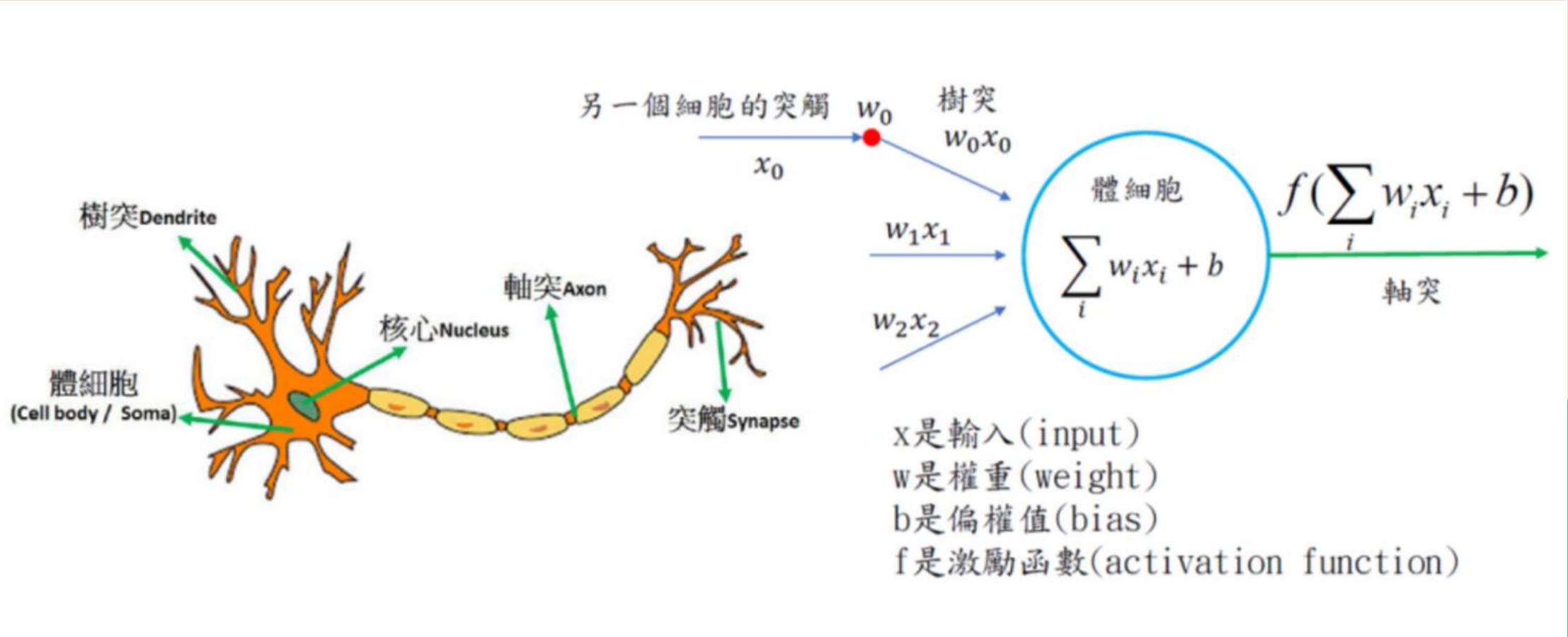
3. 設定 Loss function

# 總結機器學習過程

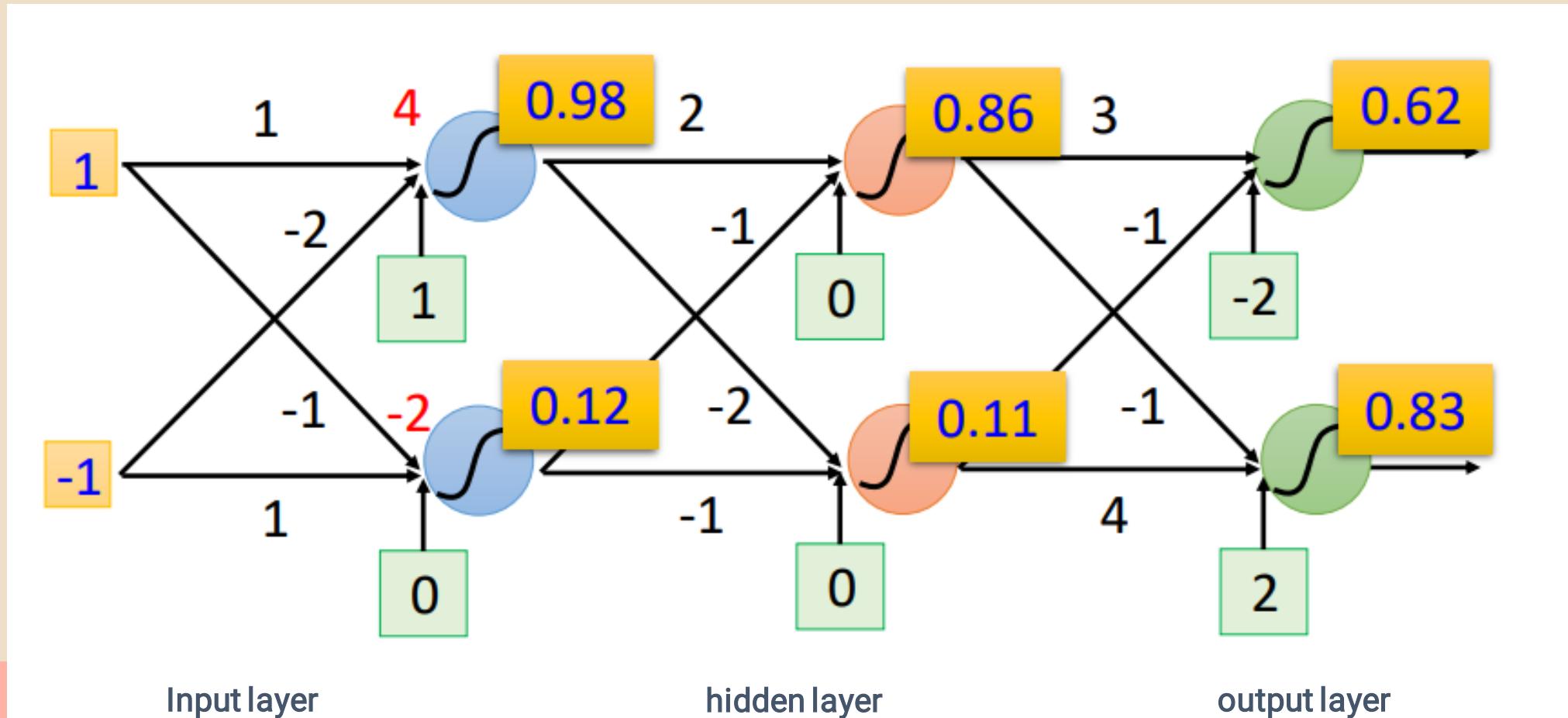
1. 準備資料
2. 設定模型
3. 設定 Loss Function
4. 設定 Learning Rate

# Deep Learning

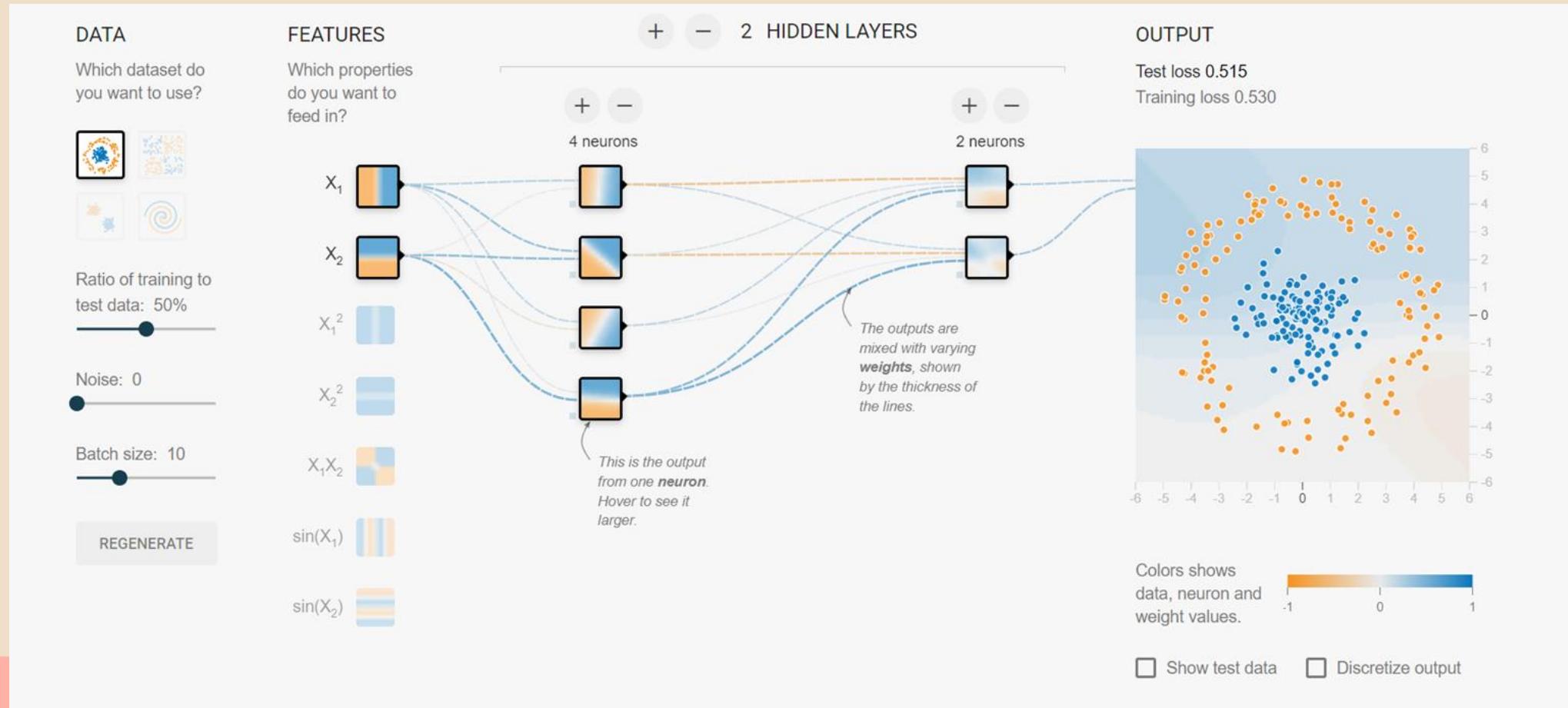
# 這個圖是甚麼？



# NN (Neural Network)

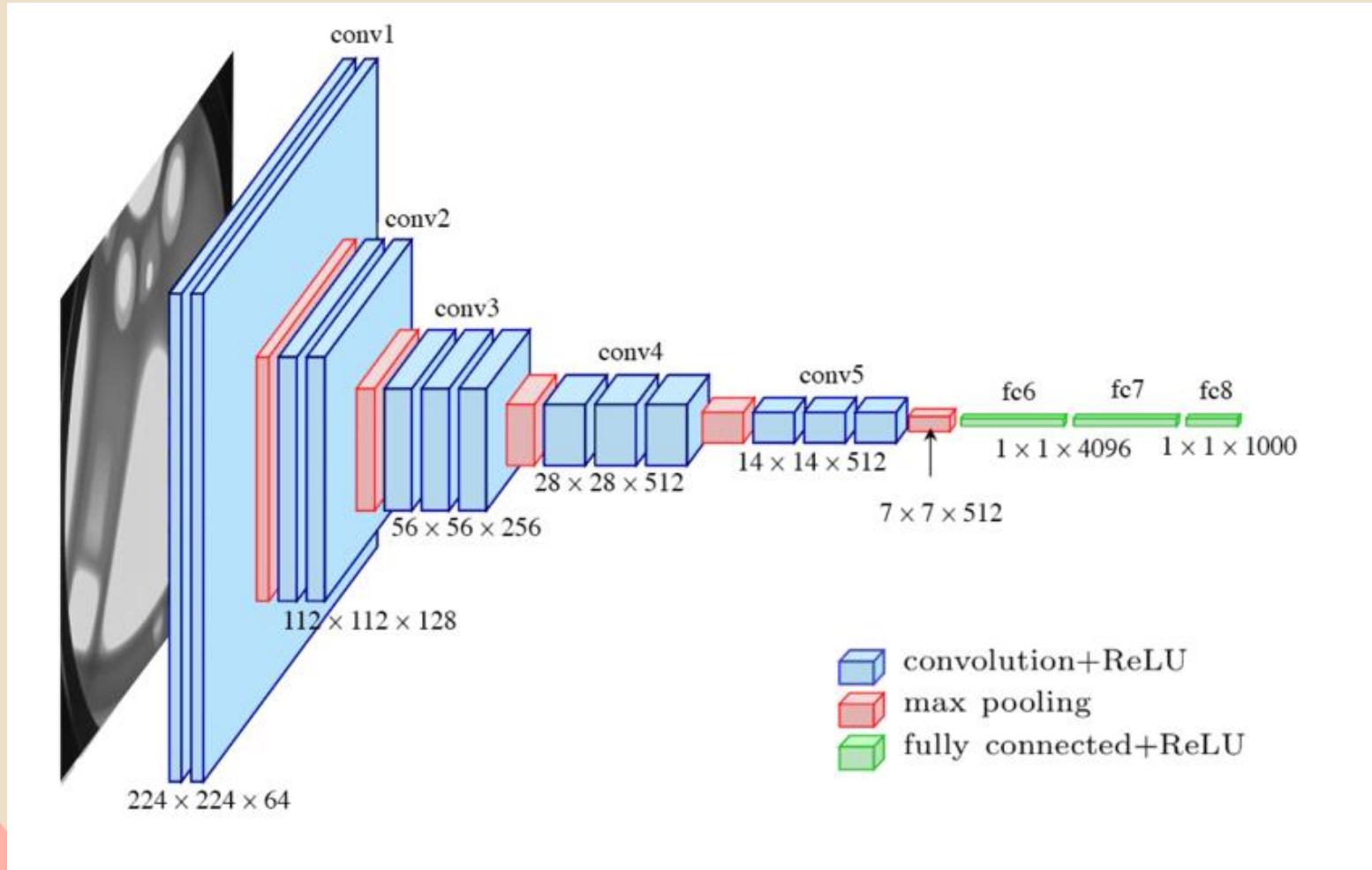


# Tensorflow playground

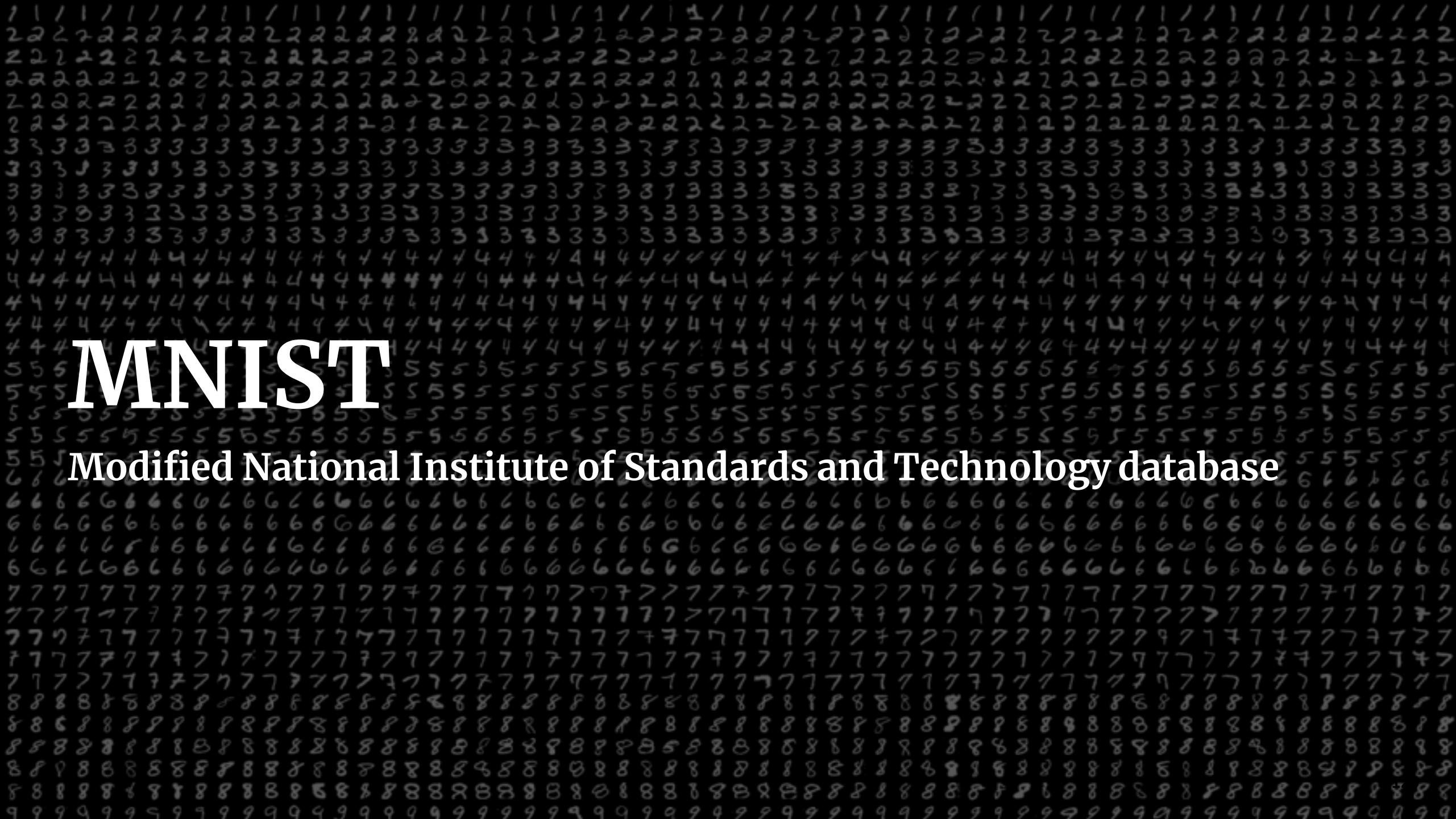


<https://playground.tensorflow.org>

# Deep = many hidden layers



**我們的課程英文名稱是？**



# MNIST

Modified National Institute of Standards and Technology database

# 載入資料

```
# 載入 MNIST 資料集
from tensorflow.keras.datasets import mnist

# 載入 MNIST 資料，資料已分割成影像和標籤，可用於訓練與驗證
(x_train, y_train), (x_valid, y_valid) = mnist.load_data()
```

# 探索 MNIST 資料

```
● ● ●  
x_train.shape # (筆數, 資料 x 軸像素, 資料 y 軸像素)  
x_valid.shape  
  
x_train.dtype # 查看資料類型  
  
x_train.min() # 查看資料當中的最小值  
x_train.max() # 查看資料當中的最大值  
  
x_train[0] # 查看其中一筆資料內容
```

# 資料視覺化

```
● ● ●  
# 查看資料圖像化後的樣貌  
import matplotlib.pyplot as plt  
  
image = x_train[0]  
plt.imshow(image, cmap='gray')  
  
# 查看這張圖代表甚麼數字  
y_train[0]
```

# 資料前處理 – 扁平化 Flatten

```
# Keras API Helper  
x_valid_flatten = x_valid.reshape(10000, 784) # (資料筆數, 陣列長度)  
x_train_flatten = x_train.reshape(60000, 784)  
  
x_train_flatten.shape # 查看 flatten 後的資料大小  
  
x_train_flatten[0] # 查看 flatten 後的資料內容樣貌
```

# 資料前處理 – 影像資料正規化

```
x_train_normalization = x_train_flatten / 255  
x_valid_normalization = x_valid_flatten / 255  
  
x_train_normalization.dtype # 查看正規化後的資料類型（型態）  
  
x_train_normalization.min() # 查看正規化後資料當中的最小值  
x_train_normalization.max() # 查看正規化後資料當中的最大值
```

# 資料前處理 – 分類編碼

```
● ● ●  
import tensorflow.keras as keras  
num_categories = 10 # 有幾個類別  
  
y_train_encoding = keras.utils.to_categorical(y_train, num_categories)  
y_valid_encoding = keras.utils.to_categorical(y_valid, num_categories)  
  
y_train_encoding[0:9] # 查看進行分類編碼後的資料標籤樣貌
```

# 開始建立模型 – 與現化

```
from tensorflow.keras.models import Sequential  
model = Sequential()
```

# 建立模型

```
● ● ●

from tensorflow.keras.layers import Dense

# 建立 input layer
model.add(Dense(units=512, activation='relu', input_shape=(784,)))

# 建立 hidden layers
model.add(Dense(units = 512, activation='relu'))

# 建立 output layer
model.add(Dense(units=10, activation='softmax'))
```

# 總結、編譯模型

```
model.summary() # 總結  
  
model.compile(loss='categorical_crossentropy', metrics=['accuracy']) # 編譯
```

# 訓練模型



訓練模型

```
history = model.fit(  
    x_train_normalization,  
    y_train_encoding,  
    validation_split = 1,  
    epochs=5,  
    verbose=1,  
    #batch_size=10,  
    validation_data=(x_valid_normalization, y_valid_encoding)  
)
```

# 繪製訓練過程的 Accuracy Plot



loss value plot

```
from matplotlib import pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

# 繪製訓練過程的 Loss Value Plot

```
● ● ● accuracy plot  
from matplotlib import pyplot as plt  
  
plt.plot(history.history['loss'])  
plt.plot(history.history['val_loss'])  
plt.title('model loss')  
plt.ylabel('loss')  
plt.xlabel('epoch')  
plt.legend(['train', 'val'], loc='upper left')  
plt.show()
```

# 繪製 Confusion Matrix

```
● ● ● Confusion Matrix Plot

import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import seaborn as sns

y_pred = model.predict(x_valid_normalization)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true_classes = np.argmax(y_valid_encoding, axis=1)

conf_matrix = confusion_matrix(y_true_classes, y_pred_classes)

class_labels = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
            xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

# 儲存與載入模型



## Save Your Model

```
# 儲存模型  
model.save("model/mnist.h5")  
  
# 載入模型  
model_load = keras.models.load_model("model/mnist.h5")  
model_load.summary()
```

# 預測圖像

```
Using Your Model

# 查看圖像
image = x_valid[2024]
plt.imshow(image, cmap='gray')

# 查看圖像的資料型態
predict_img_origin = x_valid[2024]
print(f"這張圖片的資料尺寸 : {predict_img_origin.shape}")
print(f"這張圖片的資料型態 : {predict_img_origin.dtype}")

# 圖像資料前處理
predict_img = (predict_img_origin/255).reshape(1,784).astype("float32")

# 預測圖像
import numpy as np

predictions = model_load.predict(predict_img)
predicted_classes = np.argmax(predictions, axis=1)

print(f"預測結果數值 : {predictions}")
print(f"預測的結果類別 : {predicted_classes}")
```

# 預測自己的拍攝的圖像 – 載入



## Using Your Model

```
from PIL import Image, ImageOps
from matplotlib import pyplot as plt

image_dir = 'images/two_yellow_bg.jpg' # six_blue_bg.jpg
image = Image.open(image_dir)
# 如果圖片讀取後方向不對，可以用下面這行解決
# image = image.transpose(Image.ROTATE_270)

plt.imshow(image)
plt.axis('off') # 不顯示座標軸
plt.show()
```

# 預測自己的拍攝的圖像 – 圖像前處理



## Using Your Model

```
import cv2
import numpy as np

image = cv2.imread(image_dir)
image_resized = cv2.resize(image, (28, 28)) # 調整長寬 px
image_gray = cv2.cvtColor(image_resized, cv2.COLOR_BGR2GRAY) # 轉換成灰階

plt.imshow(image_gray, cmap='gray')
plt.axis('off')
plt.show()
```

# 預測自己的拍攝的圖像 – 資料前處理



## Using Your Model

```
def check_backgroud_black(lst):
    count = sum(1 for color_code in lst if color_code < 100)
    return count >= 3

#以三個角落 ( 背景 ) 來認定圖片是否需要反轉色彩
four_corner = [image_gray[2, 4], image_gray[2, 25], image_gray[25, 2],
               image_gray[25, 25]]
if check_backgroud_black(four_corner) == True:
    predict_img = (image_gray/255).reshape(1,784).astype('float32')
else:
    inverted_image = cv2.bitwise_not(image_gray)
    plt.imshow(inverted_image, cmap='gray')
    plt.axis('off')
    plt.show()
    predict_img = (inverted_image/255).reshape(1,784).astype('float32')
```

# 預測自己的拍攝的圖像 – 預測

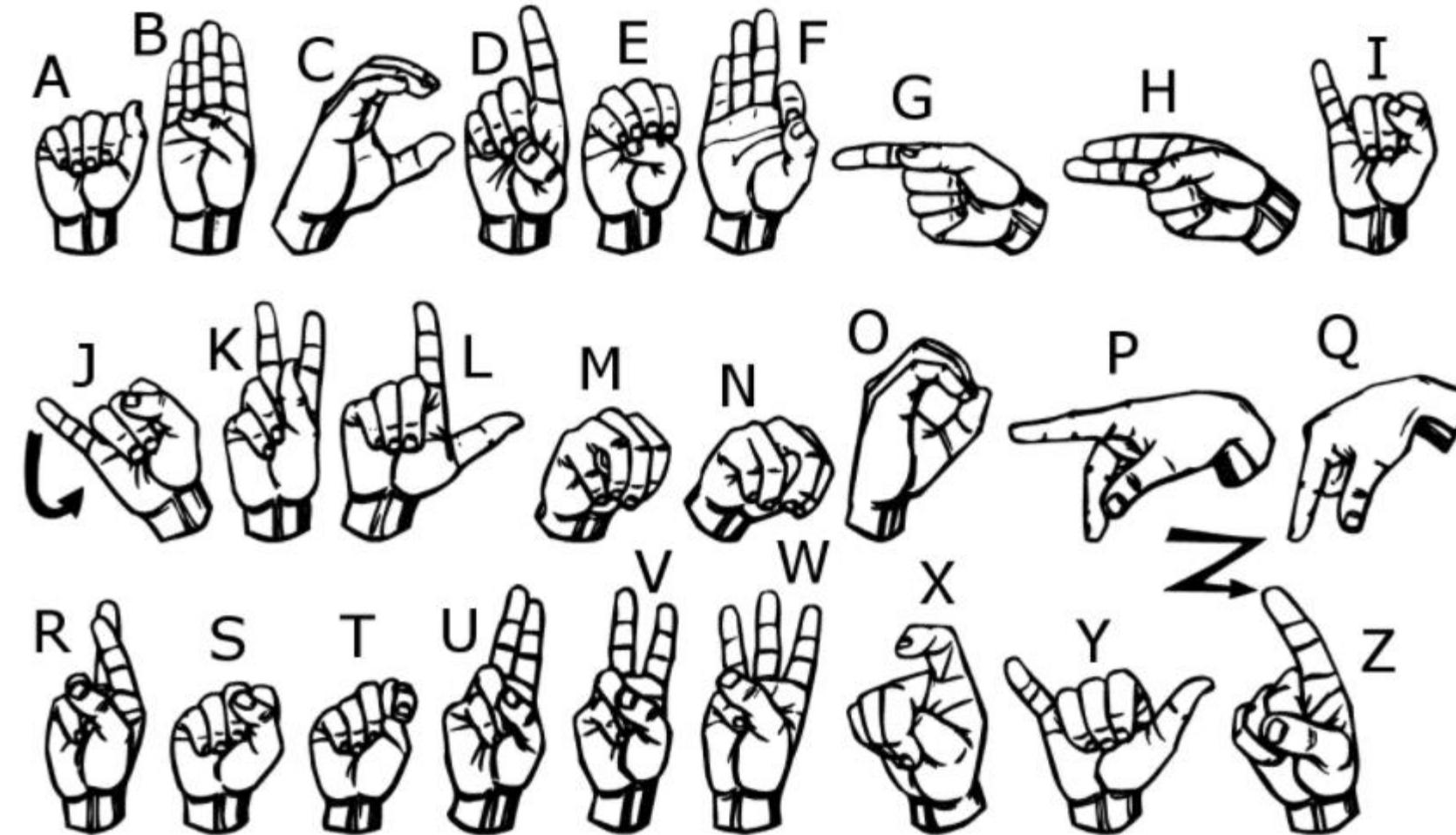


## Using Your Model

```
predictions = model_load.predict(predict_img)
predicted_classes = np.argmax(predictions, axis=1)

print(f"預測結果數值 : {predictions}")
print(f"預測的結果類別 : {predicted_classes}")
```

# It's your turn!





Thank You