機器學習

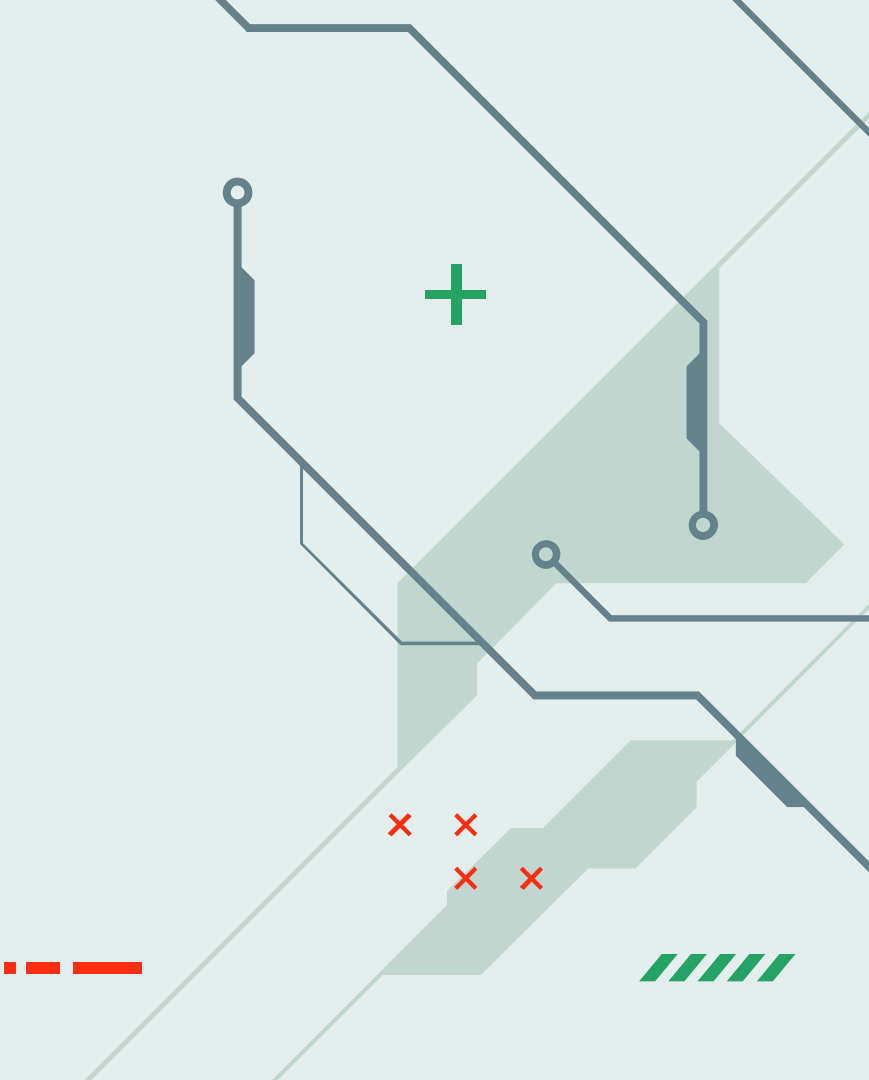
講師 : 陳聖文 (Wayne)

01

前期準備

運行環境

Python 運行環境、套件管理工具



運行環境

- Anaconda
- Nvidia Docker (Docker + Nvidia Container Toolkit)
- **Colab (推薦)**

Anaconda

- Anaconda
- Miniconda
- **Mini-Forge (推薦)**

MiniForge

Nvidia Docker

Windows

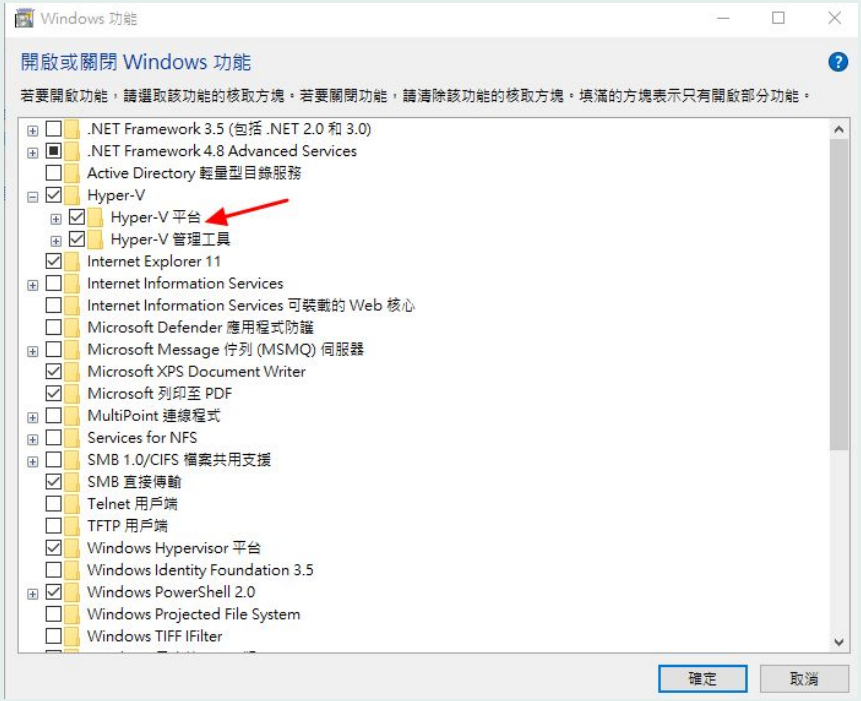
- Nvidia Container Toolkit
- Docker Engine
- WSL2

Linux/Unix

- Nvidia Container Toolkit
- Docker Engine

https://learn.microsoft.com/zh-tw/windows/wsl/install

Window Subsystem for Linux 2



Window Subsystem for Linux 2

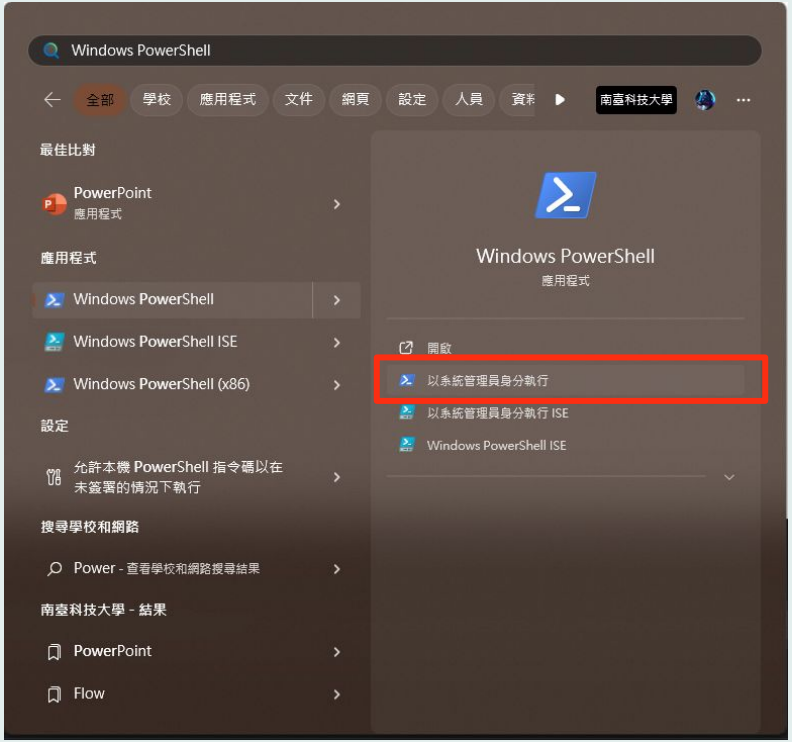
在 PowerShell 以系統管理員執行:

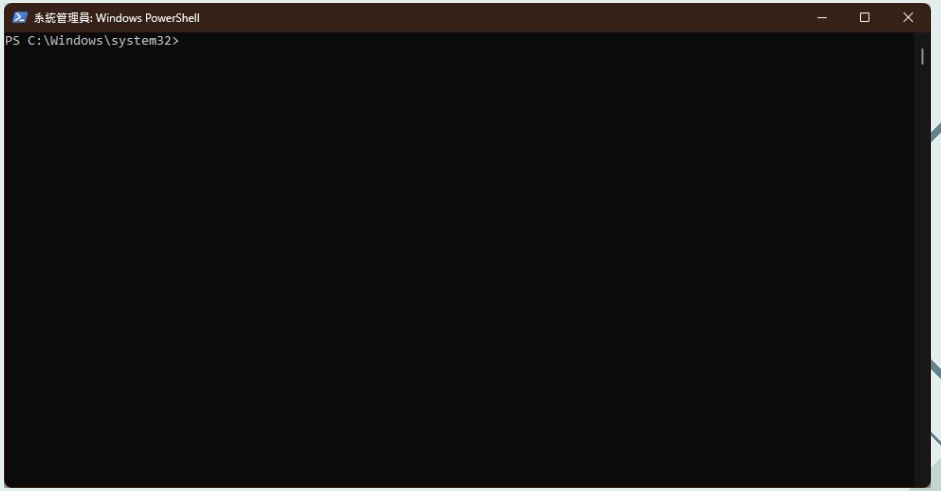
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart

dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

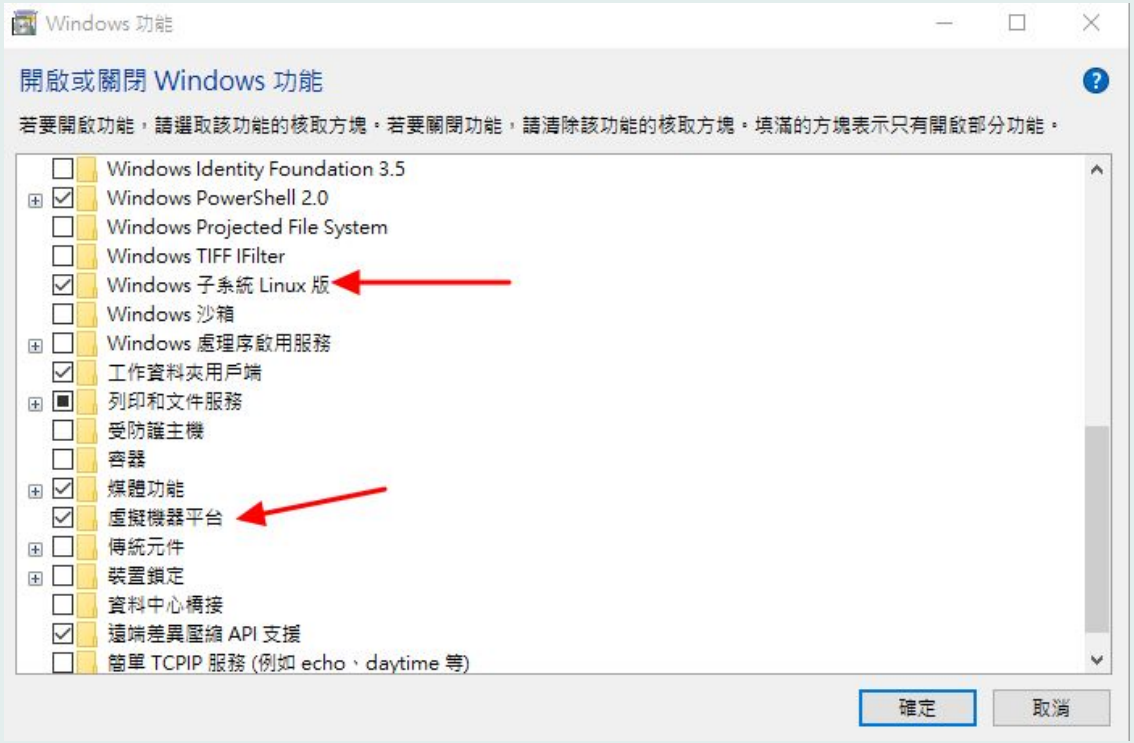
執行後重新開機

Window Subsystem for Linux 2





Window Subsystem for Linux 2

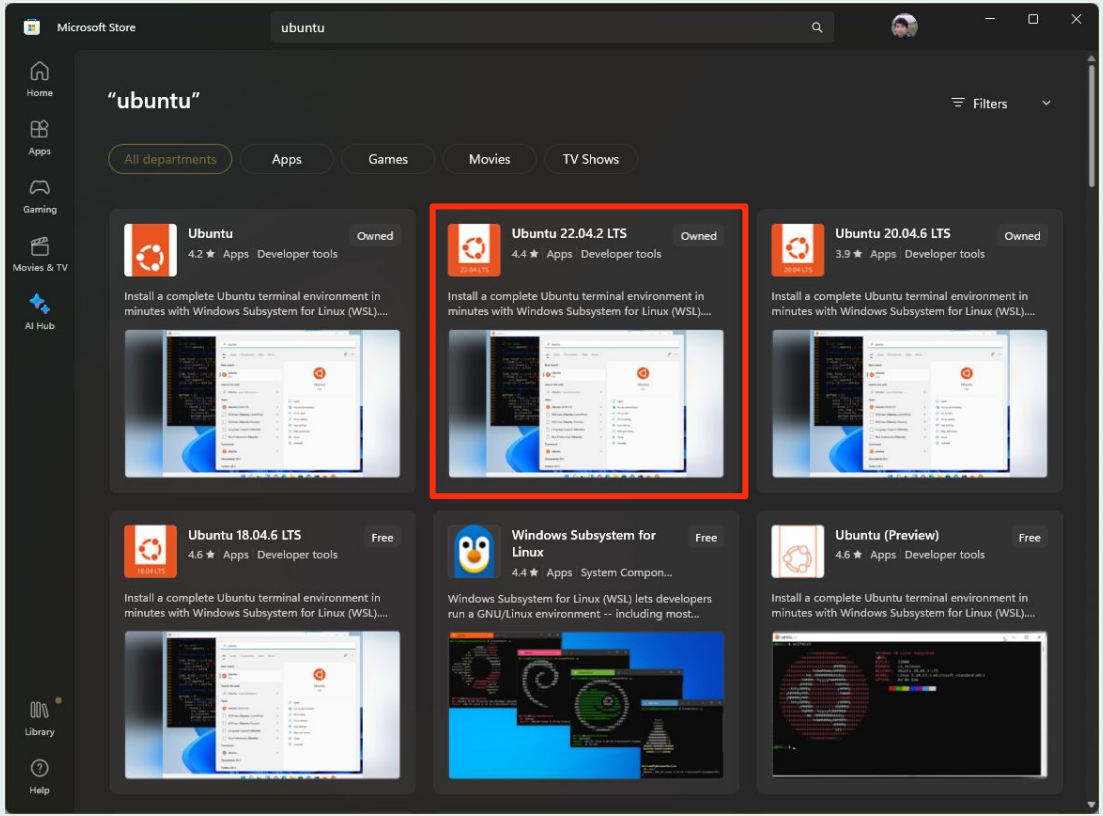


Window Subsystem for Linux 2

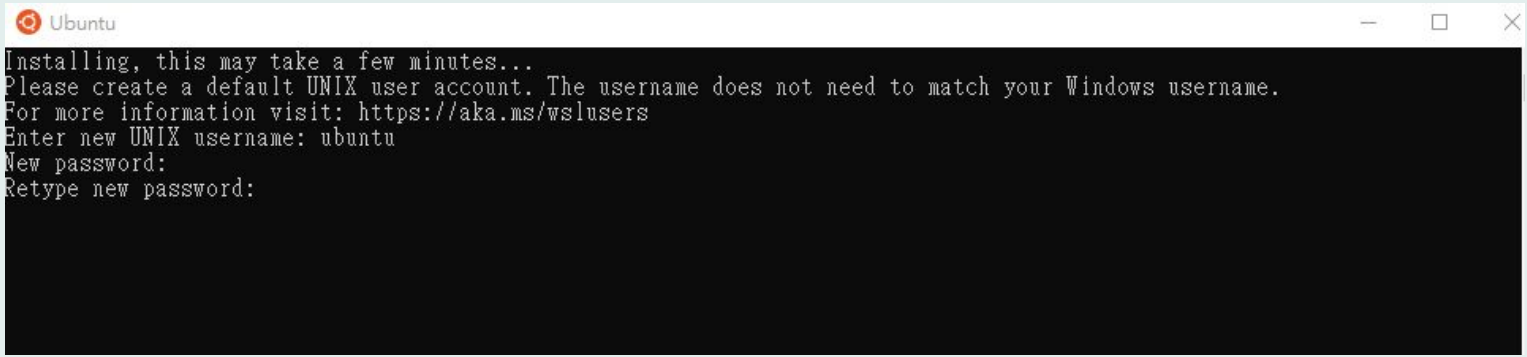
將 WSL 預設版本調成 WSL2:

wsl --set-default-version 2

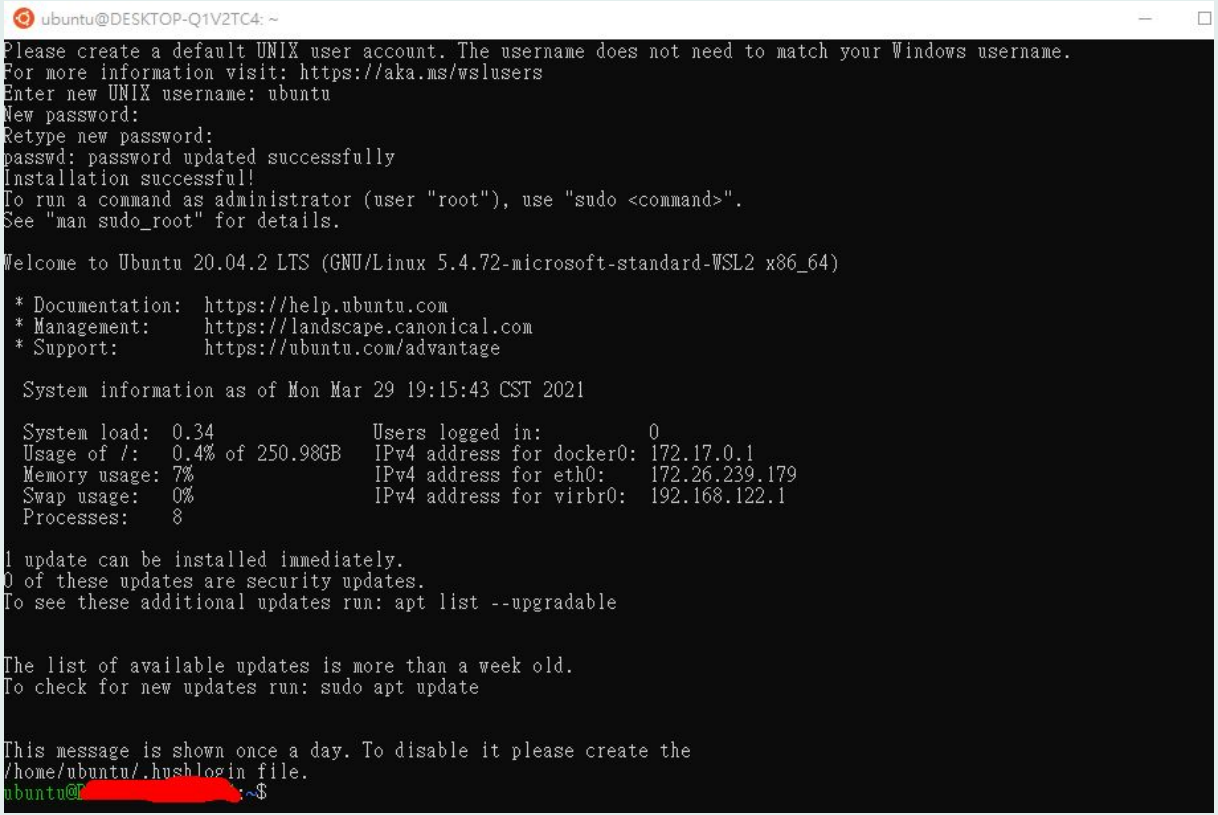
Window Subsystem for Linux 2



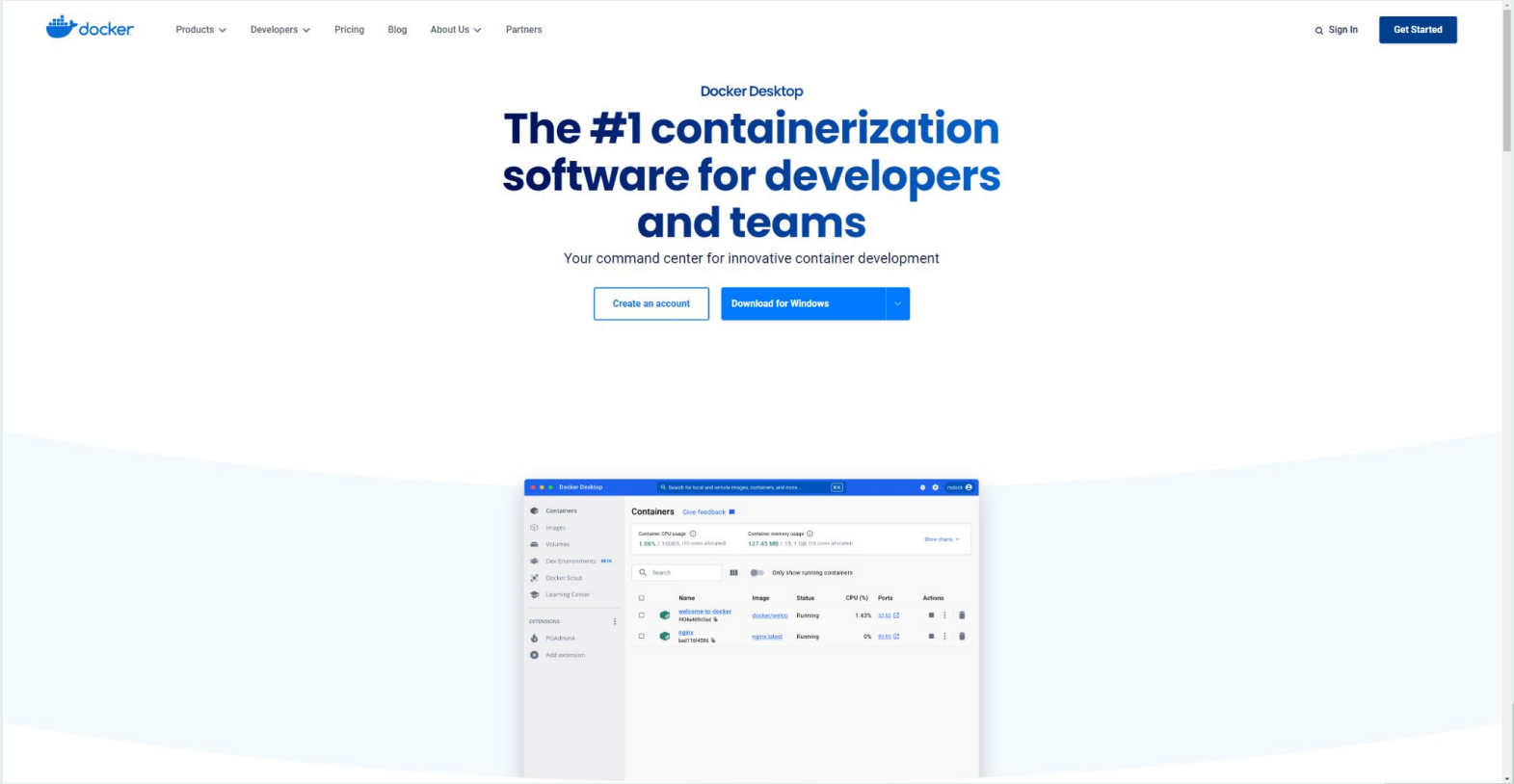
Window Subsystem for Linux 2



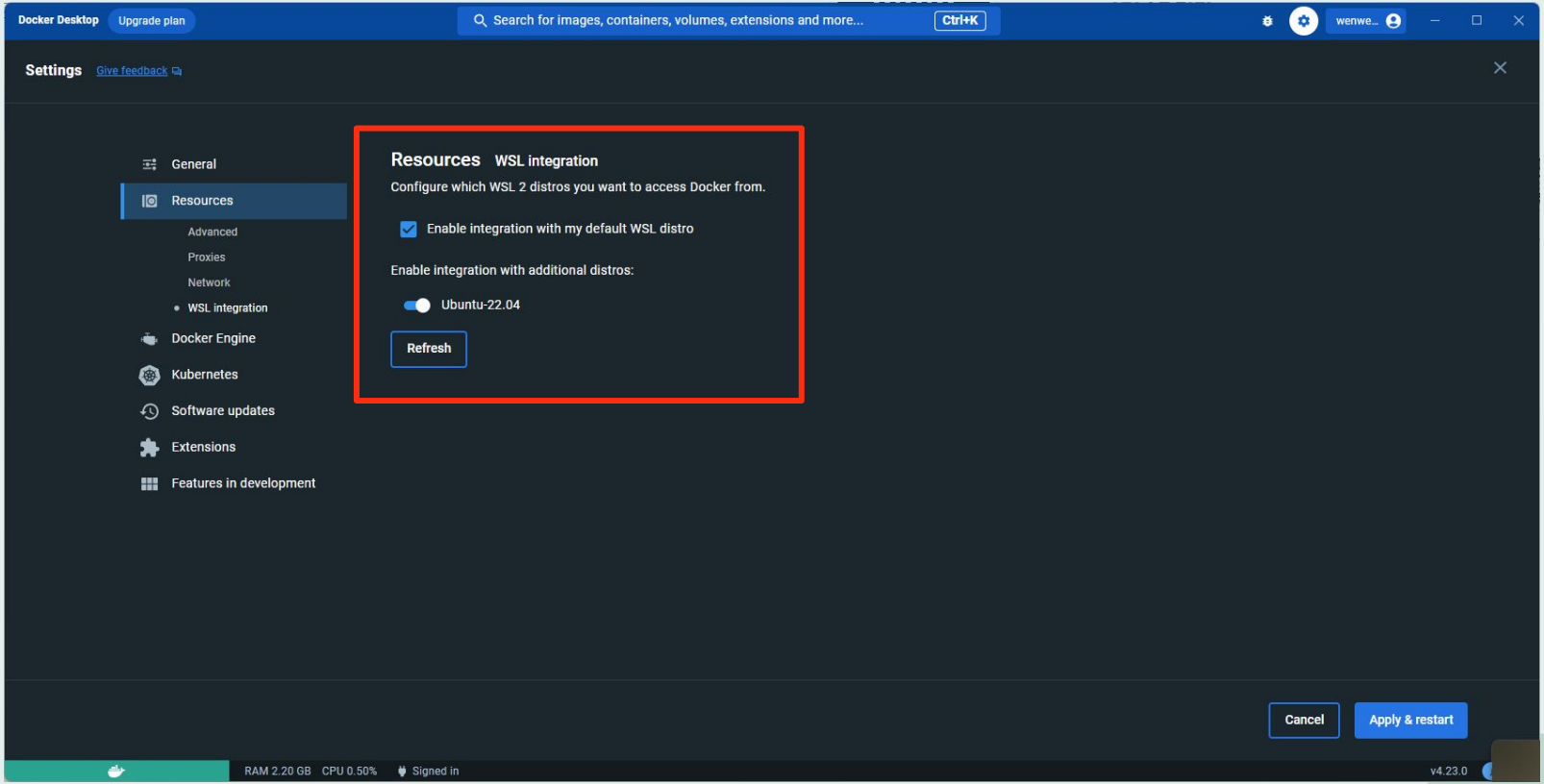
Window Subsystem for Linux 2



Docker

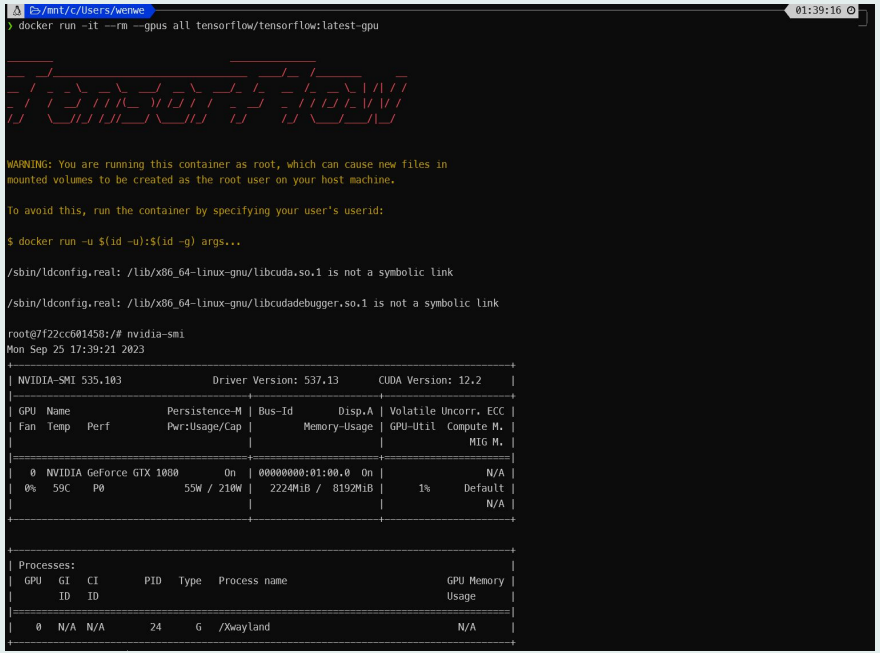

docker
Products
Developers
Pricing
Blog
About Us
Partners
Sign In
Get Started
Docker Desktop
The #1 containerization software for developers and teams
Your command center for innovative container development
Create an account
Download for Windows
Containers
Give feedback
Search
Only show running containers
Name
Image
Status
CPU (%)
Ports
Actions
Running
Running

Docker


Docker Desktop
Upgrade plan
Search for images, containers, volumes, extensions and more...
Ctrl+K
wenwe...
Settings
Give feedback
General
Resources
Advanced
Proxies
Network
WSL integration
Docker Engine
Kubernetes
Software updates
Extensions
Features in development
Resources
WSL integration
Configure which WSL 2 distros you want to access Docker from.
Enable integration with my default WSL distro
Enable integration with additional distros:
Ubuntu-22.04
Refresh
Cancel
Apply & restart
RAM 2.20 GB
CPU 0.50%
Signed in
v4.23.0

Nvidia Docker

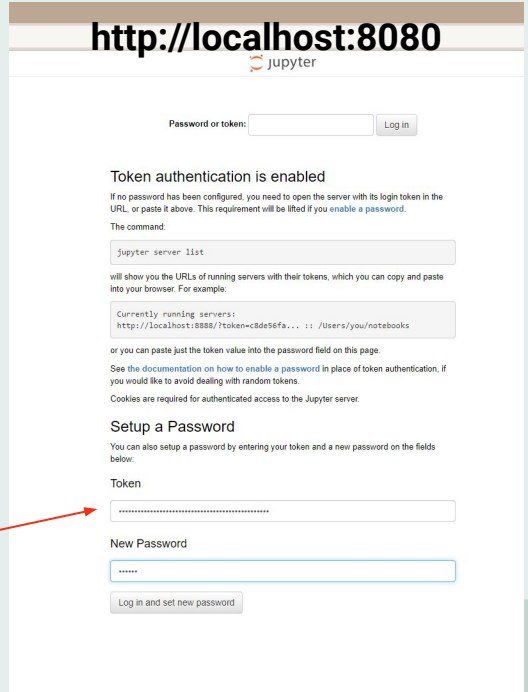
docker run -it --rm --gpus all tensorflow/tensorflow:latest-gpu



Nvidia Docker

docker run -it --rm -p 8080:8888 --gpus all tensorflow/tensorflow:latest-gpu-jupyter

```
/mnt/c/Users/wenwe
❯ docker run -it --rm -p 8080:8888 --gpus all tensorflow/tensorflow:latest-jupyter
[I 17:51:54.601 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie_secret


  _   _          _     _
 | | | |_ __   __| |__ _| |_ ___
 | |_| | '_ \ / _` / _` |  _/ -_)
  \___/| .__/\__,_\__,_|\__\___|
       |_|

Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions.

https://jupyter-notebook.readthedocs.io/en/latest/migrate_to_notebook7.html

Please note that updating to Notebook 7 might break some of your extensions.

jupyter_http_over_ws extension initialized. Listening on /http_over_websocket
[I 17:51:54.777 NotebookApp] Serving notebooks from local directory: /tf
[I 17:51:54.777 NotebookApp] Jupyter Notebook 6.5.4 is running at:
[I 17:51:54.777 NotebookApp] http://3d5823a45e28:8888/?token=2d4b9c886feab4cb4a5eb713d1b1dfd139314bb8fd33c331
[I 17:51:54.777 NotebookApp]  or http://127.0.0.1:8888/?token=2d4b9c886feab4cb4a5eb713d1b1dfd139314bb8fd33c331
[I 17:51:54.777 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 17:51:54.780 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///root/.local/share/jupyter/runtime/nbserver-1-open.html
    Or copy and paste one of these URLs:
        http://3d5823a45e28:8888/?token=2d4b9c886feab4cb4a5eb713d1b1dfd139314bb8fd33c331
     or http://127.0.0.1:8888/?token=2d4b9c886feab4cb4a5eb713d1b1dfd139314bb8fd33c331
```



Nvidia Docker

docker compose up

C:\Users\wenwe\PycharmProjects\ai_course_2023\docker-compose.yaml - Notepad++

檔案(F) 編輯(E) 搜尋(S) 檢視(V) 編碼(N) 語言(L) 設定(T) 工具(O) 巨集(M) 執行(R) 外掛(P) 視窗(W) ?

httpd-vhosts.conf | docker-compose.yaml

```yaml
version: '3.9'

services:
  jupyter:
    image: tensorflow/tensorflow:latest-gpu-jupyter
    container_name: jupyter
    ports:
      - '8080:8080'
    volumes:
      - .:/tf
    environment:
      JUPYTER_PORT: 8080
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: 1
              capabilities:
                - gpu

```

C:\Windows\system32\cmd.e

```
C:\Users\wenwe\PycharmProjects\ai_course_2023>ls -al
total 22
drwxr-xr-x 1 wenwe 197609   0 Sep 26 02:06 .
drwxr-xr-x 1 wenwe 197609   0 Sep 25 20:49 ..
drwxr-xr-x 1 wenwe 197609   0 Sep 26 01:58 .git
drwxr-xr-x 1 wenwe 197609   0 Sep 26 02:05 .idea
-rw-r--r-- 1 wenwe 197609 143 Sep 25 20:50 README.md
-rw-r--r-- 1 wenwe 197609 409 Sep 26 02:06 docker-compose.yaml
drwxr-xr-x 1 wenwe 197609   0 Sep 25 20:50 docs
drwxr-xr-x 1 wenwe 197609   0 Sep 26 01:58 notebooks
drwxr-xr-x 1 wenwe 197609   0 Sep 25 20:50 slides

C:\Users\wenwe\PycharmProjects\ai_course_2023>docker compose up
[+] Running 1/0
 ✔ Container jupyter  Created
Attaching to jupyter
jupyter  | [I 18:09:14.051 NotebookApp] Serving notebooks from local directory: /tf
jupyter  | [I 18:09:14.051 NotebookApp] Jupyter Notebook 6.5.4 is running at:
jupyter  | [I 18:09:14.051 NotebookApp] http://9f48657ac4db:8080/?token=58db8ea39229d6c27bc7890eebe60805774037538a54741a
jupyter  | [I 18:09:14.051 NotebookApp]  or http://127.0.0.1:8080/?token=58db8ea39229d6c27bc7890eebe60805774037538a54741a
jupyter  | [I 18:09:14.051 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
jupyter  | [C 18:09:14.054 NotebookApp]
jupyter  |
jupyter  |     To access the notebook, open this file in a browser:
jupyter  |         file:///root/.local/share/jupyter/runtime/nbserver-1-open.html
jupyter  |     Or copy and paste one of these URLs:
jupyter  |         http://9f48657ac4db:8080/?token=58db8ea39229d6c27bc7890eebe60805774037538a54741a
jupyter  |      or http://127.0.0.1:8080/?token=58db8ea39229d6c27bc7890eebe60805774037538a54741a
```

Colab

檢查安裝

請執行並檢查下方的欄位是否能順利運行。

```
[1]  1  import sys
     2  sys.version # 必須工作
```

```
'3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0]'
```

```
[2]  1  import tensorflow as tf
     2  tf.__version__ # 必須工作
```

```
'2.13.0'
```

```
[3]  1  import tensorflow_probability as tfp
     2  tfp.__version__ # 必須工作且版本 > 0.16
```

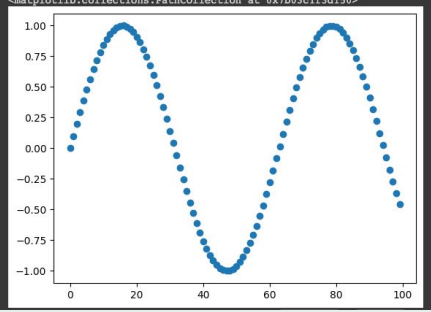
```
'0.20.1'
```

```
[4]  1  import numpy as np
     2  np.__version__ # 必須工作
```

```
'1.23.5'
```

```
1  import matplotlib.pyplot as plt
2  %matplotlib inline
3  plt.scatter(range(100), np.sin(0.1 * np.array(range(100))))
4  # 必須工作
```

```
<matplotlib.collections.PathCollection at 0x7b03c1f3d150>
```



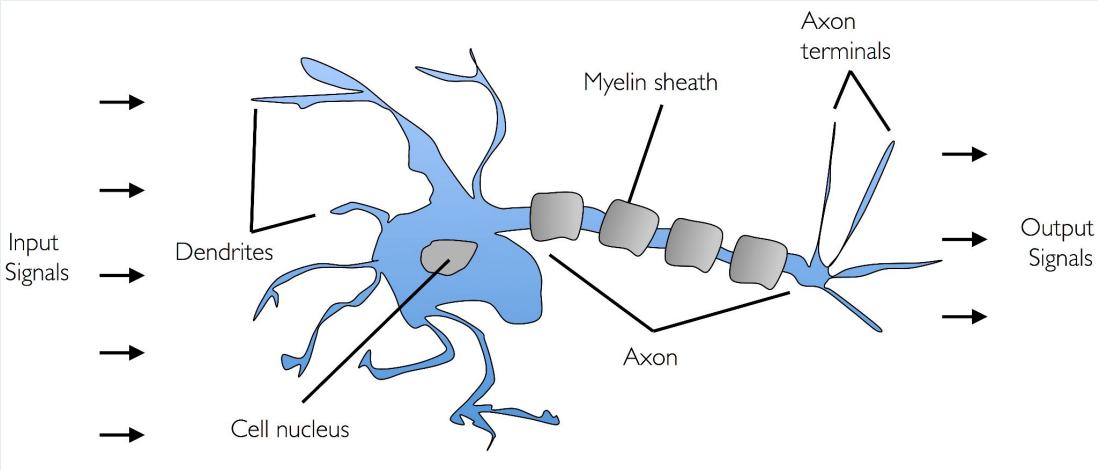
Colab

02

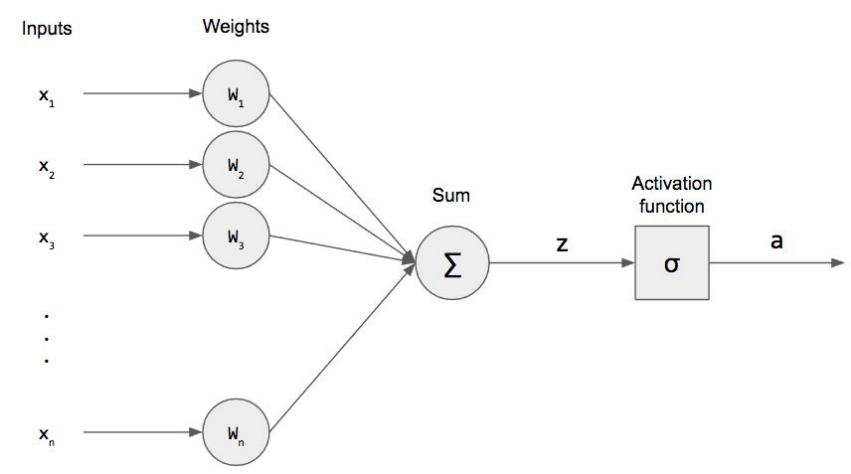
線性分類器

Linear Classification

人體的神經元


Axon
terminals
Myelin sheath
Input
Signals
Dendrites
Output
Signals
Axon
Cell nucleus

人工的神經元



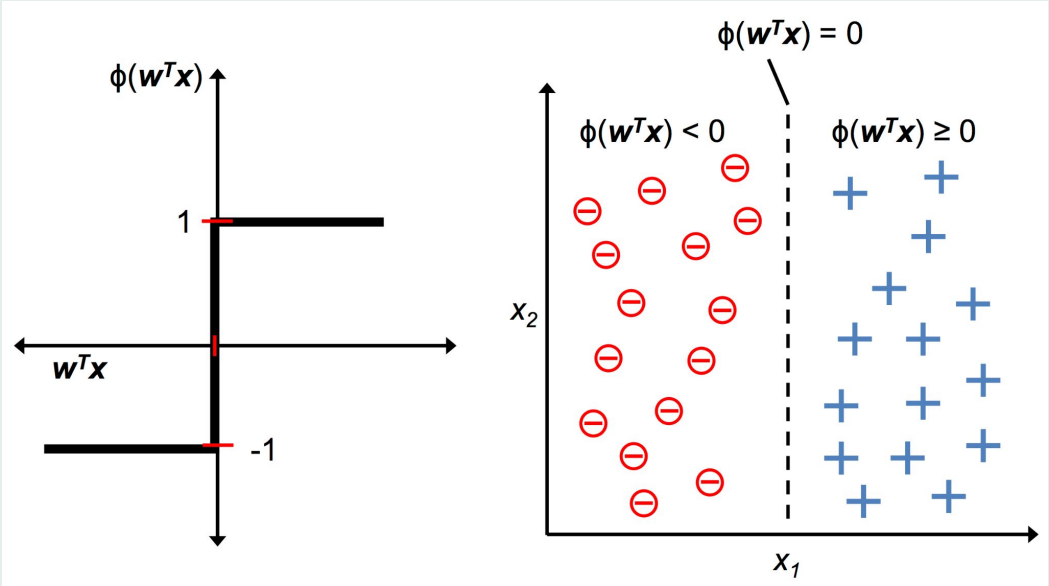
$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \qquad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

$$z = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

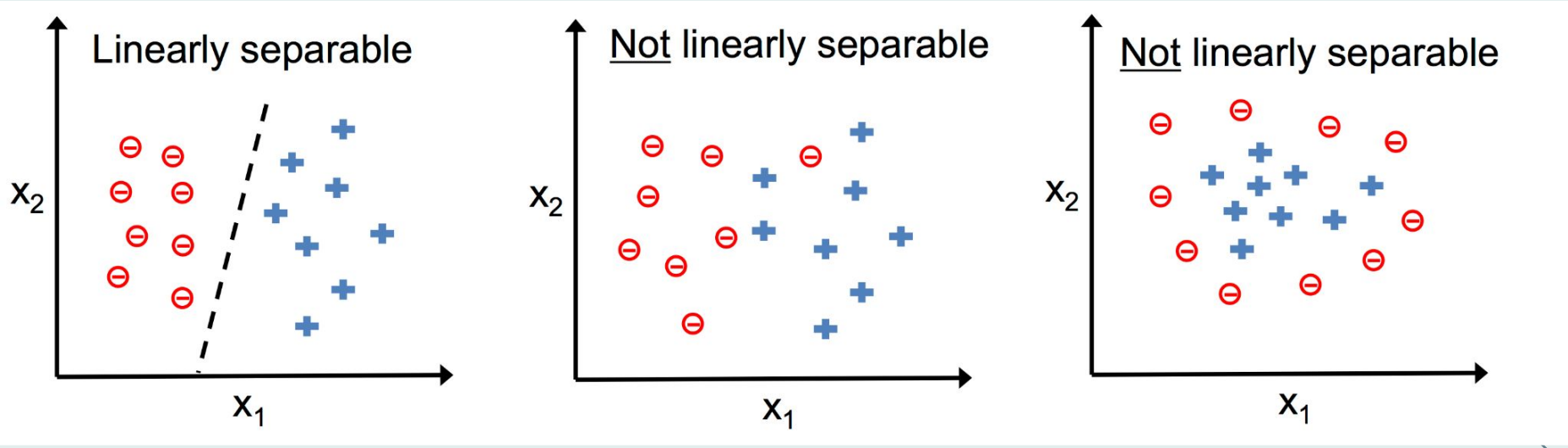
$$\phi(z) = \begin{cases} 1 & \text{if } z \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

人工的神經元

$$\phi(z) = \begin{cases} 1 \text{ if } z \geq \theta \\ -1 \text{ otherwise} \end{cases}$$

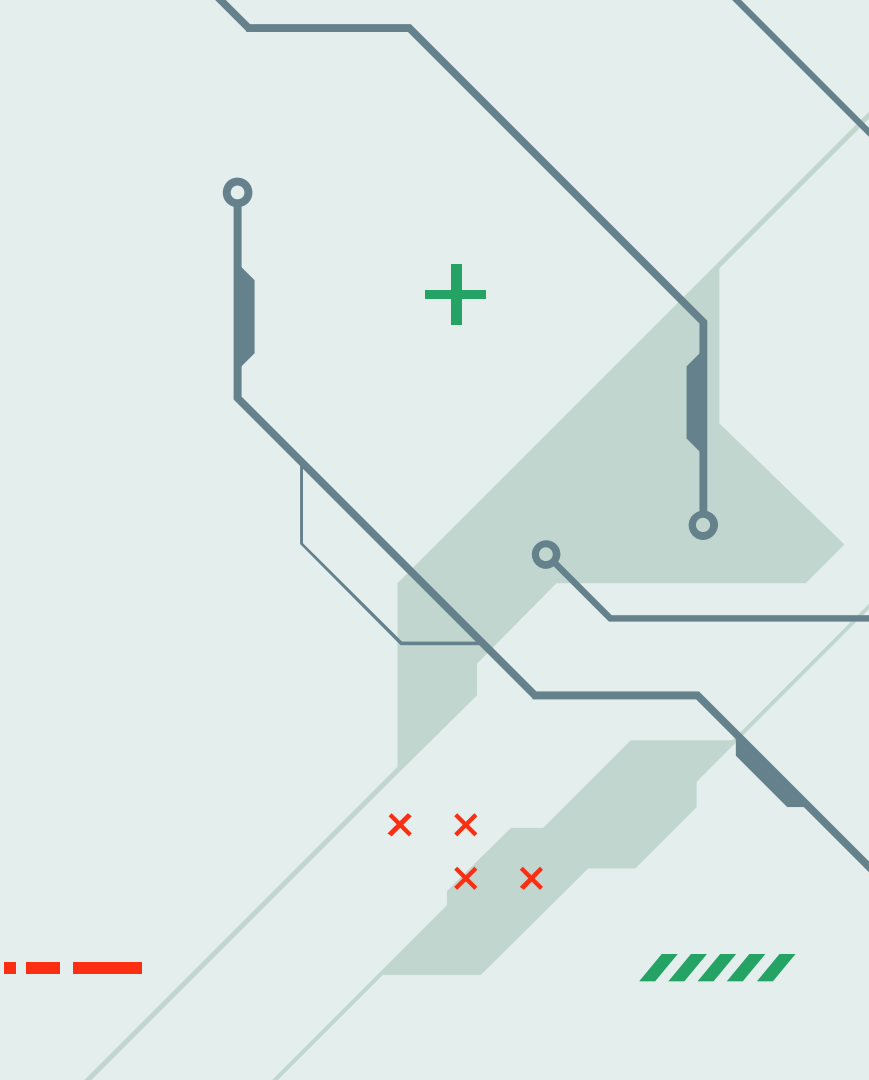


線性可分／線性不可分

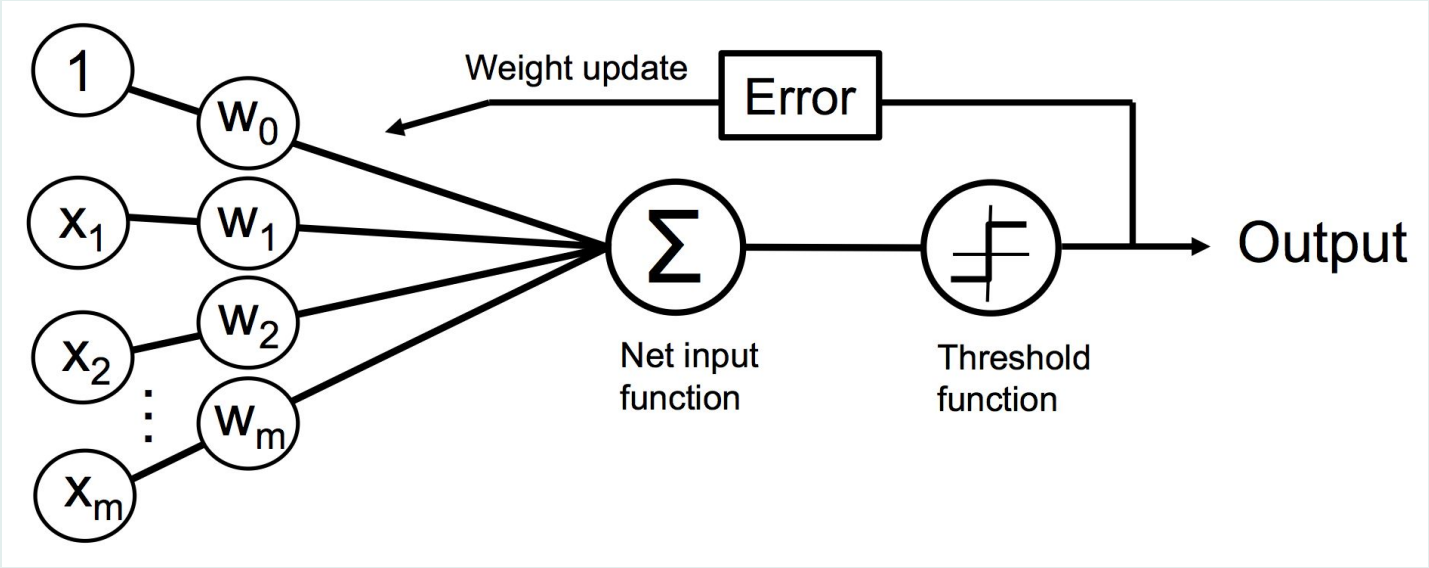

Linearly separable
x_2
x_1
Not linearly separable
x_2
x_1
Not linearly separable
x_2
x_1

Perceptron

感知器



感知器


1
w_0
x_1
w_1
x_2
w_2
w_m
x_m
Weight update
Error
Σ
Net input function
Threshold function
Output

權重更新

$$w_{j+1} = w_j + \Delta w_j$$

$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)}$$

權重更新

預測正確例子：

(1) $y^{(i)} = -1, \quad \hat{y}^{(i)} = -1, \quad \Delta w_j = \eta(-1-(-1))x_j^{(i)} = 0$

(2) $y^{(i)} = 1, \quad \hat{y}^{(i)} = 1, \quad \Delta w_j = \eta(1-1)x_j^{(i)} = 0$

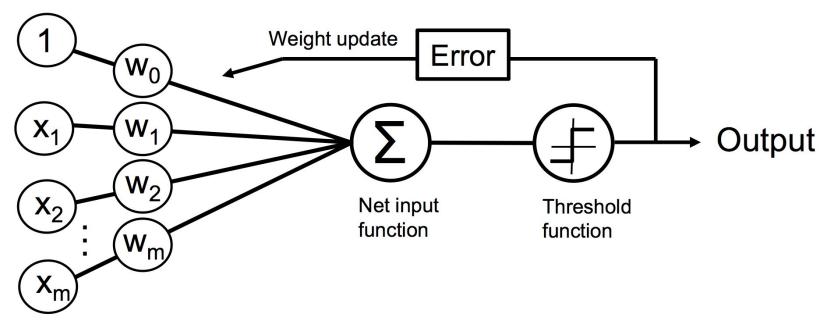
預測錯誤例子:

(3) $y^{(i)} = 1, \quad \hat{y}^{(i)} = -1, \quad \Delta w_j = \eta(1-(-1))x_j^{(i)} = \eta(2)x_j^{(i)}$

(4) $y^{(i)} = -1, \quad \hat{y}^{(i)} = 1, \quad \Delta w_j = \eta(-1-1)x_j^{(i)} = \eta(-2)x_j^{(i)}$

感知器訓練規則

1. 初始化權重為 0 或隨機的數字
2. Foreach 訓練樣本, x^i:
   a. 計算輸出值 ŷ
   b. 更新權重



感知器的更新公式

$$
\begin{aligned}
z &= W^T X = \sum_{i=0}^{m} w_i x_i \\
\hat{y} &= \phi(z) \\
\Delta w_j &= \eta(y^{(i)} - \hat{y}^{(i)}) x_j^{(i)} \\
w_{j+1} &= w_j + \Delta w_j
\end{aligned}
$$

感知器訓練規則

如下圖手算更新權重。

已知 $W_{\square}^{1} = [0.5 \ 0 \ -1 \ 1]^T$ 為目前神經元的連結權重向量，假設學習速率 $\eta = 1$，且神經元為 bipolar binary neuron，即 $\phi(z) = \text{sign}(z)$，利用下面兩組輸入調整目前的權重 $W_{\square}^{1}$，求出 $W_{\square}^{2}$ 及 $W_{\square}^{3}$。

$X_1 = [0 \ 1.5 \ -2 \ 1]^T$

$y_1 = -1$

$X_2 = [-1.5 \ -2 \ -0.5 \ 1]^T$

$y_2 = 1$

實作感知器

訓練機器學習進行分類

目錄

- 人工神經網路
- 利用 Iris 資料集訓練感知器

在 Python 中實作感知器學習演算法

實作感知器

In [1]:

https://colab.research.google.com/github/wenwen357951/ai_course_2023/blob/main/notebooks/01_Machine_Learning_Classification.ipynb

Iris 資料集

- 鳶尾花是一個古典的花朵資料集
- 由英國統計學家 Ronald Fisher 爵士在1936年時，對加斯帕半島上的鳶尾屬花朵所提取的花瓣花萼的長寬數據資料
- 山鳶尾，變色鳶尾，維吉尼亞鳶尾三類進行標示，共150筆資料。

https://archive.ics.uci.edu/dataset/53/iris

Iris 資料集

屬性:

- Sepal length: 花萼長度 (cm)
- Sepal width: 花萼寬度 (cm)
- Petal length: 花瓣長度 (cm)
- Petal width: 花瓣寬度 (cm)

標籤:

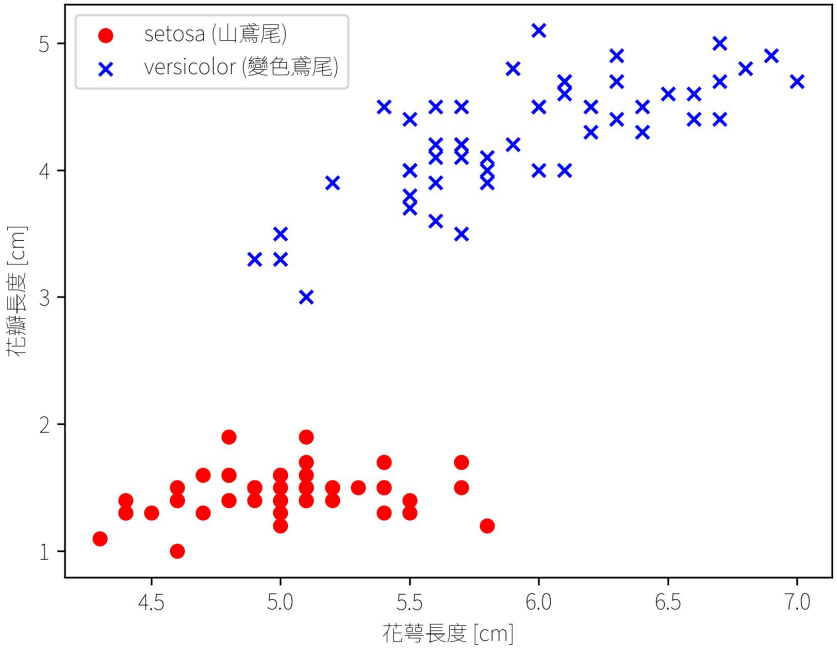
- setosa: 山鳶尾
- versicolor: 變色鳶尾
- virginica: 維吉尼亞鳶尾

Variables Table

Variable Name	Role	Type	Demographic	Description	Units	Missing Values
sepal length	Feature	Continuous			cm	no
sepal width	Feature	Continuous			cm	no
petal length	Feature	Continuous			cm	no
petal width	Feature	Continuous			cm	no
class	Target	Categorical		class of iris plant: Iris Setosa, Iris Versicolour, or Iris Virginica		no

Rows per page 10 0 to 5 of 5

資料分布


setosa (山鳶尾)
versicolor (變色鳶尾)
花瓣長度 [cm]
花萼長度 [cm]
1
2
3
4
5
4.5
5.0
5.5
6.0
6.5
7.0

使用 Iris 資料集訓練感知器

在 Python 中實作感知器學習演算法

實作感知器

In [1]:

利用 Iris 資料集來訓練感知器(Perceptron)

使用網路中的開放資料集，來訓練感知器。

讀取 Iris 資料集 (網路)

In [2]:

Out[2]:

	花萼長度	花萼寬度	花瓣長度	花瓣寬度	標籤
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

讀取 Iris 的資料集 (本地)

In [3]:

Out[3]:

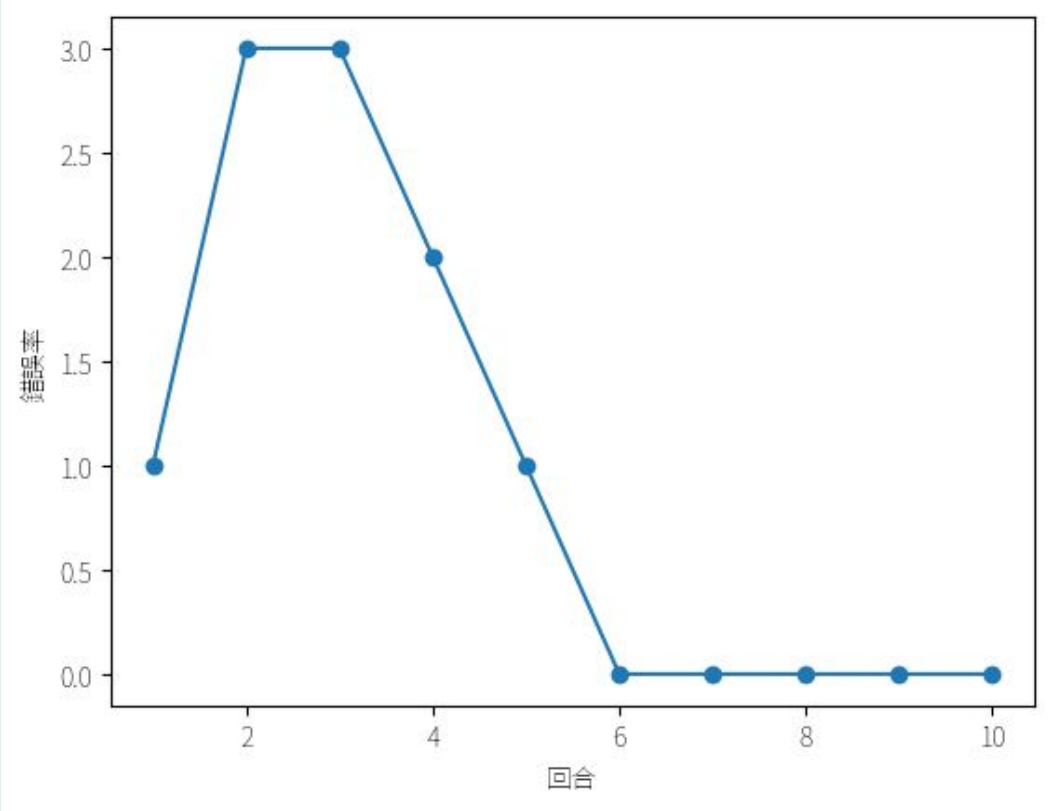
	花萼長度	花萼寬度	花瓣長度	花瓣寬度	標籤
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

讓 MatPlotLib 可以使用中文字型

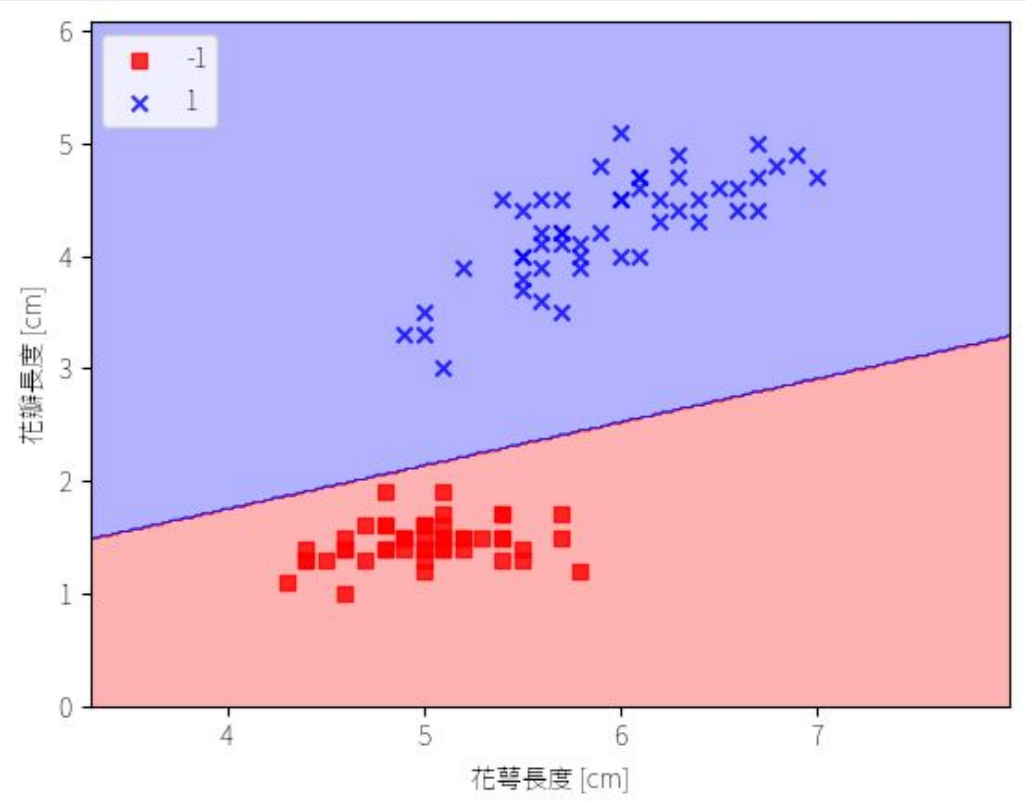
In [4]:

https://colab.research.google.com/github/wenwen357951/ai_course_2023/blob/main/notebooks/01_Machine_Learning_Classification.ipynb

訓練過程

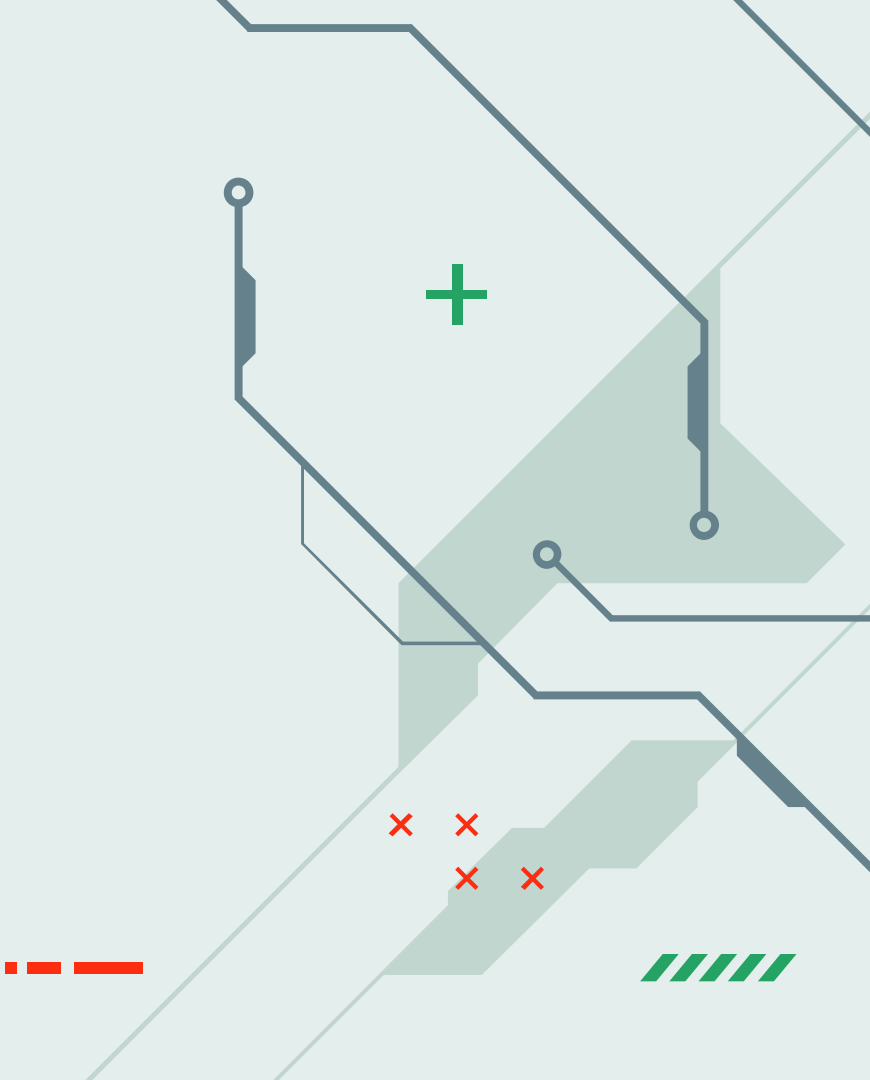


決策邊界

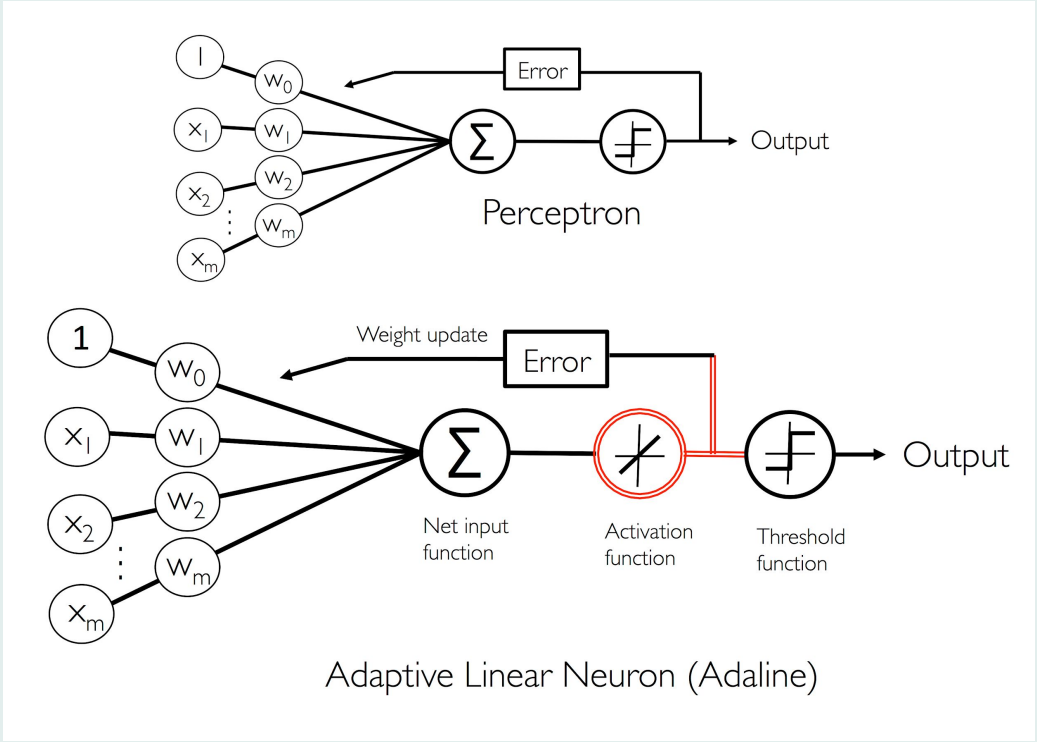


AdalineGD

自適應梯度下降線性神經網路



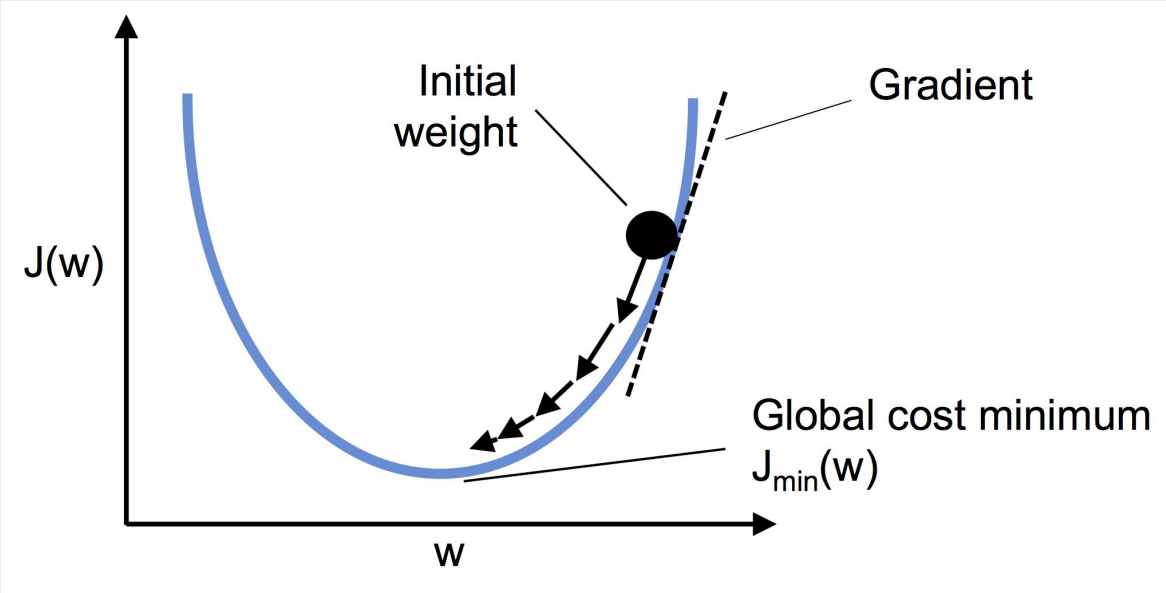
自適應線性神經元(Adaline)



平方和誤差 (Sum of Squared Error)

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i}\left(y^{(i)} - \phi\left(z^{(i)}\right)\right)^2$$

梯度下降



$$w := w + \Delta w$$

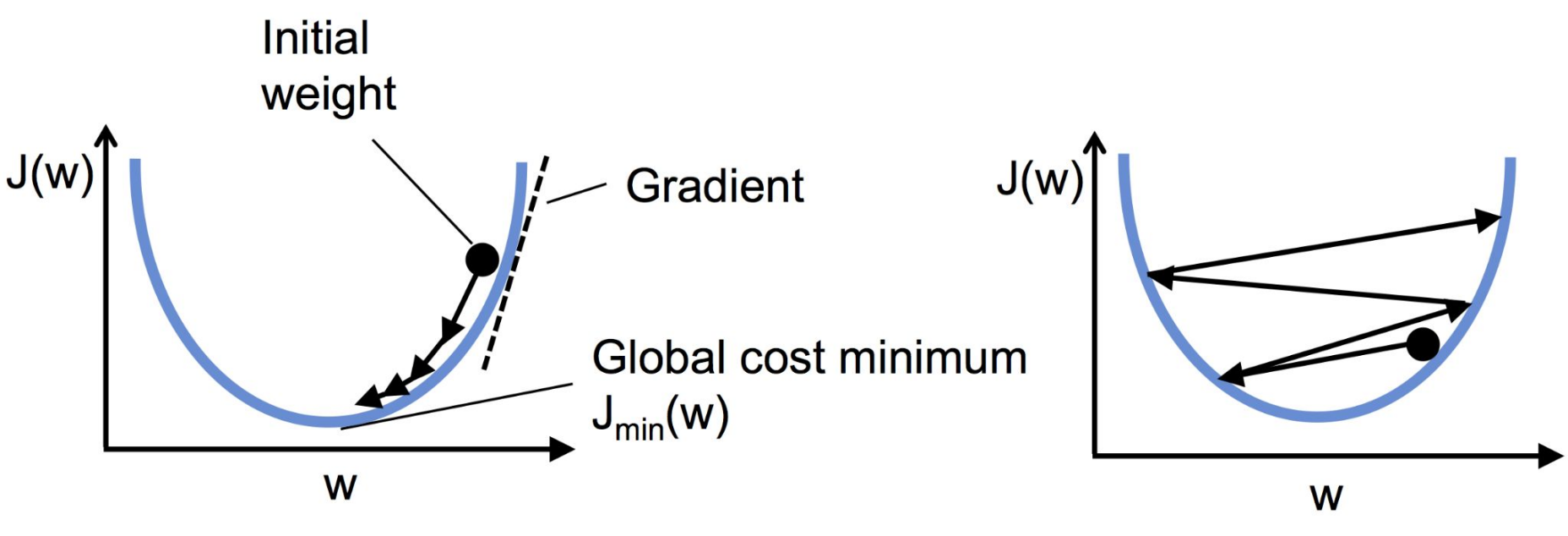
梯度下降公式

利用Adaline更新單一神經元網路之權重。

已知 $W_{\square}^{1} = [1 \ -1 \ 0 \ 0.5]^T$ 為目前神經元的連結權重向量，兩組輸入向量X_1、X_2的預測輸出值為$y_1 = -1$、$y_2 = -1$，假設活化函數為 $\phi(z) = z$ ，其中 $\phi'(z) = 1$ ，其中$y = \phi(z)$，學習速率 $\eta = 1$，試調整目前的權重 W^1，來推求下兩個迭代的權重W^2及W^3。

$X_1 = [1 \ -2 \ 0 \ -1]^T$ $X_2 = [0 \ 1.5 \ -0.5 \ -1]^T$

梯度下降與學習率



實作 AdalineGD

實作自適應梯度下降線性神經網路 (AdalineGD)

In [9]:
```
# 實作 AdalineGD
```

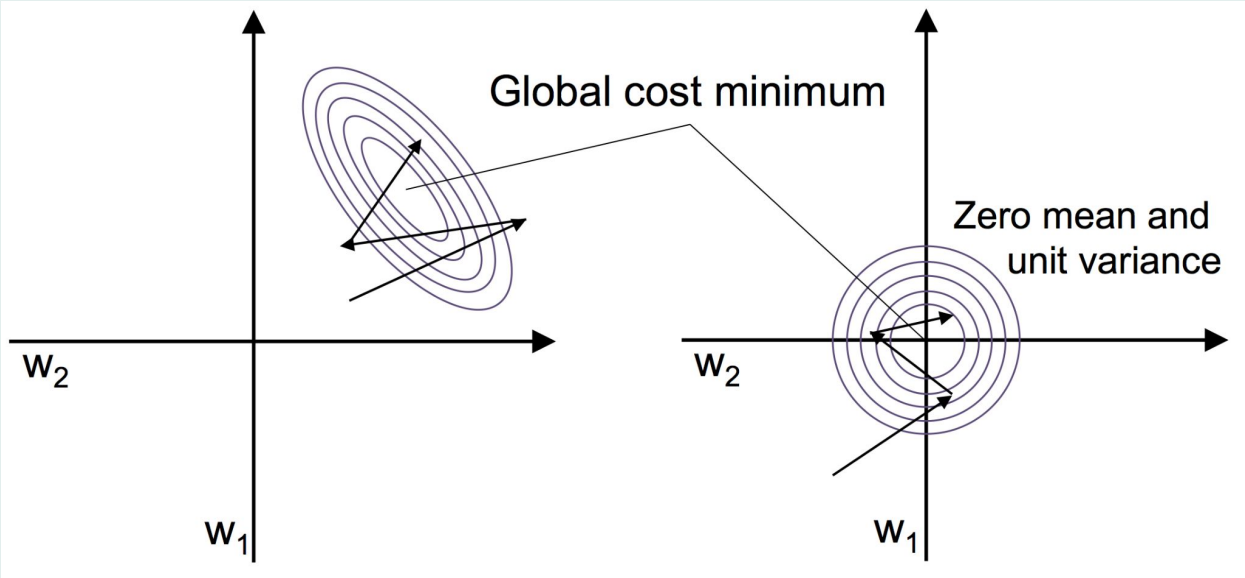
資料標準化，特徵縮放

In [10]:
```
# 原始資料
```

Out[10]:
```
array([[5.1, 1.4],
       [4.9, 1.4],
       [4.7, 1.3],
       [4.6, 1.5],
       [5. , 1.4],
       [5.4, 1.7],
       [4.6, 1.4],
       [5. , 1.5],
       [4.4, 1.4],
       [4.9, 1.5],
       [5.4, 1.5],
       [4.8, 1.6],
       [4.8, 1.4],
       [4.3, 1.1],
       [5.8, 1.2],
       [5.7, 1.5],
       [5.4, 1.3],
       [5.1, 1.4],
       [5.7, 1.7],
       [5.1, 1.5],
       [5.4, 1.7],
       [5.1, 1.5],
       [4.6, 1. ],
       [5.1, 1.7],
       [4.8, 1.9],
       [5. , 1.6],
```

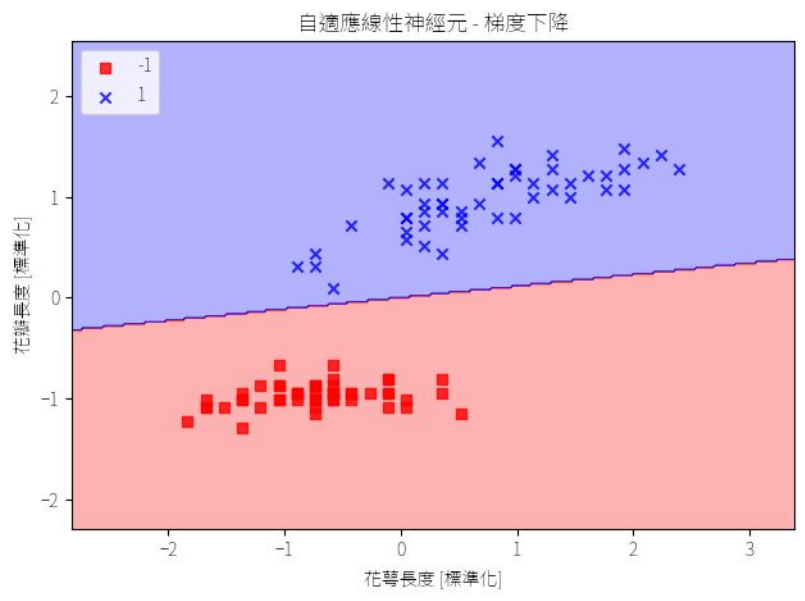
https://colab.research.google.com/github/wenwen357951/ai_course_2023/blob/main/notebooks/01_Machine_Learning_Classification.ipynb

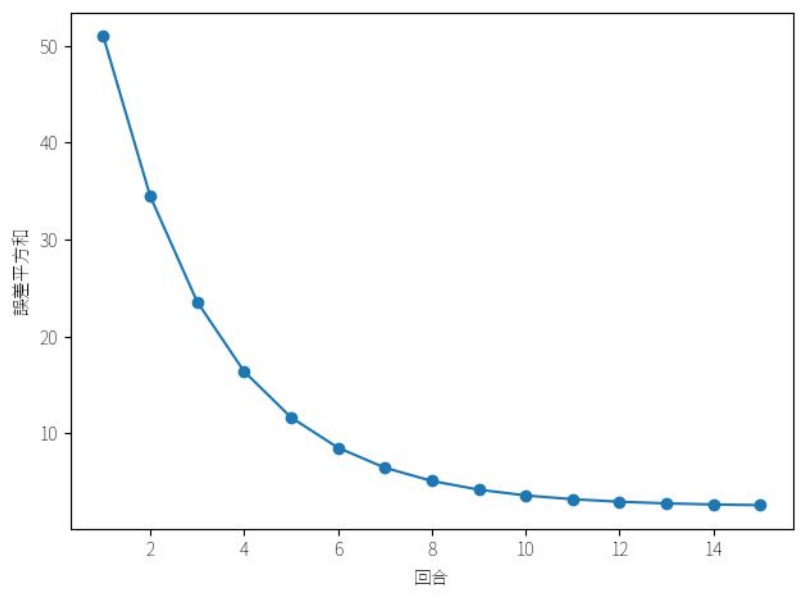
標準化



$$x'_j = \frac{x_j - \mu_j}{\sigma_j}$$

分類結果





03

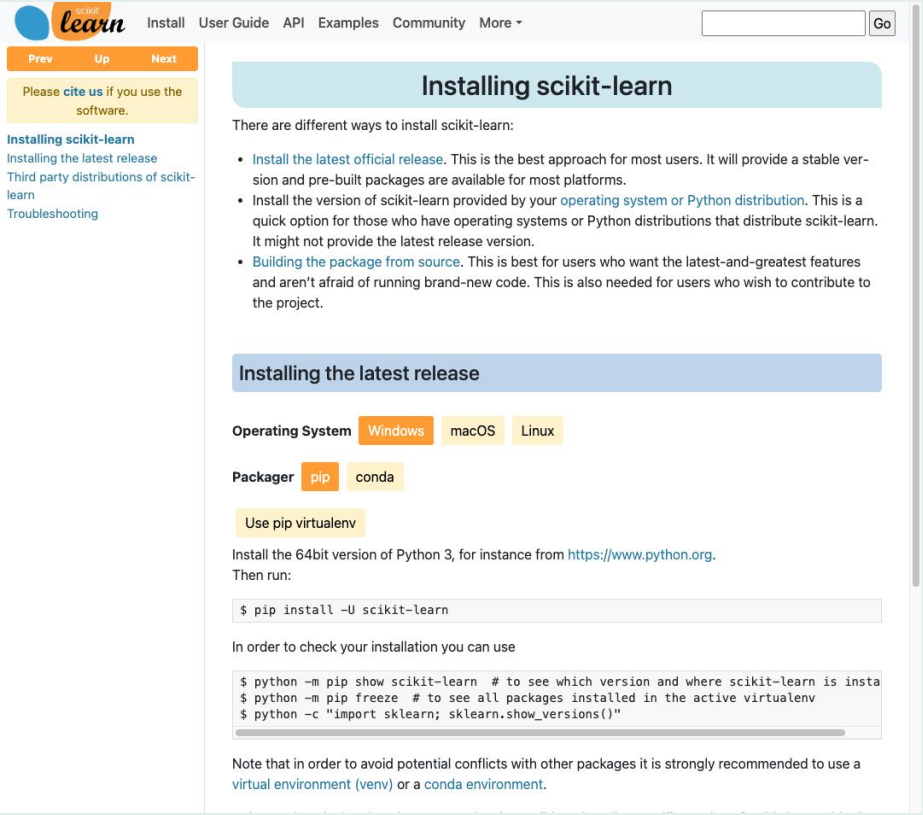
Scikit-Learn

機器學習套件

Scikit Learn

https://scikit-learn.org/stable/index.html

安装方式



Install User Guide API Examples Community More Go

Prev Up Next

Please cite us if you use the software.

Installing scikit-learn
Installing the latest release
Third party distributions of scikit-learn
Troubleshooting

Installing scikit-learn

There are different ways to install scikit-learn:

- Install the latest official release. This is the best approach for most users. It will provide a stable version and pre-built packages are available for most platforms.
- Install the version of scikit-learn provided by your operating system or Python distribution. This is a quick option for those who have operating systems or Python distributions that distribute scikit-learn. It might not provide the latest release version.
- Building the package from source. This is best for users who want the latest-and-greatest features and aren't afraid of running brand-new code. This is also needed for users who wish to contribute to the project.

Installing the latest release

Operating System Windows macOS Linux

Packager pip conda

Use pip virtualenv

Install the 64bit version of Python 3, for instance from https://www.python.org.
Then run:

```
$ pip install -U scikit-learn
```

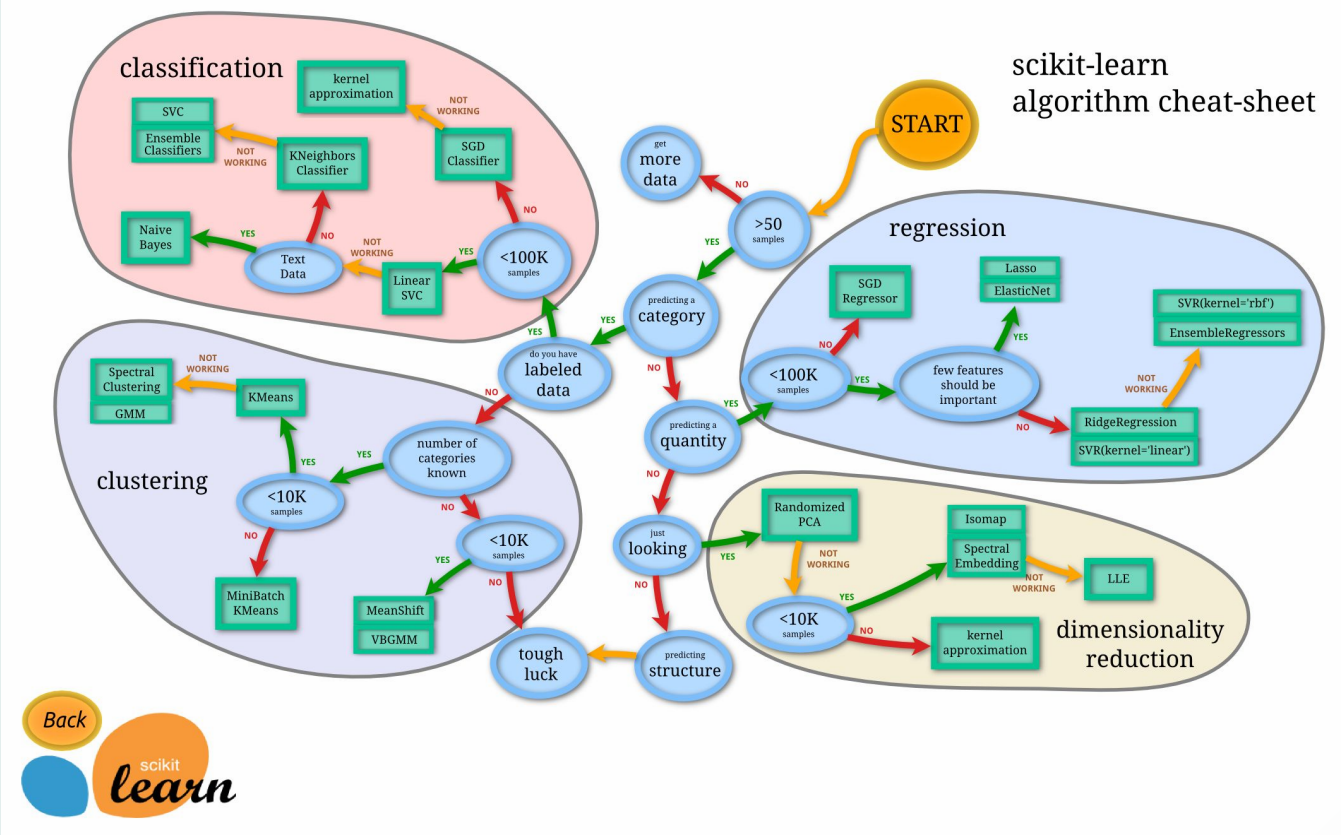
In order to check your installation you can use

```
$ python -m pip show scikit-learn  # to see which version and where scikit-learn is insta
$ python -m pip freeze  # to see all packages installed in the active virtualenv
$ python -c "import sklearn; sklearn.show_versions()"
```

Note that in order to avoid potential conflicts with other packages it is strongly recommended to use a virtual environment (venv) or a conda environment.

https://scikit-learn.org/stable/index.html

演算法選擇地圖


scikit-learn
algorithm cheat-sheet
START
get more data
>50 samples
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
classification
kernel approximation
SVC
Ensemble Classifiers
KNeighbors Classifier
SGD Classifier
Naive Bayes
Text Data
Linear SVC
<100K samples
NOT WORKING
YES
NO
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
<100K samples
few features should be important
RidgeRegression
SVR(kernel='linear')
clustering
Spectral Clustering
GMM
KMeans
number of categories known
<10K samples
MiniBatch KMeans
MeanShift
VBGMM
dimensionality reduction
Randomized PCA
Isomap
Spectral Embedding
LLE
<10K samples
kernel approximation
Back
scikit learn

使用 Scikit-Learn 機器學習套件進行分類

02. 使用 Scikit-Learn 機器學習套件進行分類

目錄

- 使用 Scikit Learn
- 在 Scikit-Learn 中訓練感知器
- 繪製決策邊界
- 透過邏輯迴歸模型來取的類別機率
- 支持向量機 (Support Vector Machines)

使用 Scikit Learn

匯入套件庫

In [1]:
```
# 匯入 Sklearn 套件的資料集
# 匯入 numpy 套件
```

載入 Sklearn 提供的 Iris 鳶尾花資料集

In [2]:
```
# 使用 Sklearn 載入資料集
# 顯示資料數量與類別標籤
```

```
資料數量： 150
類別標籤： [0 1 2]
```

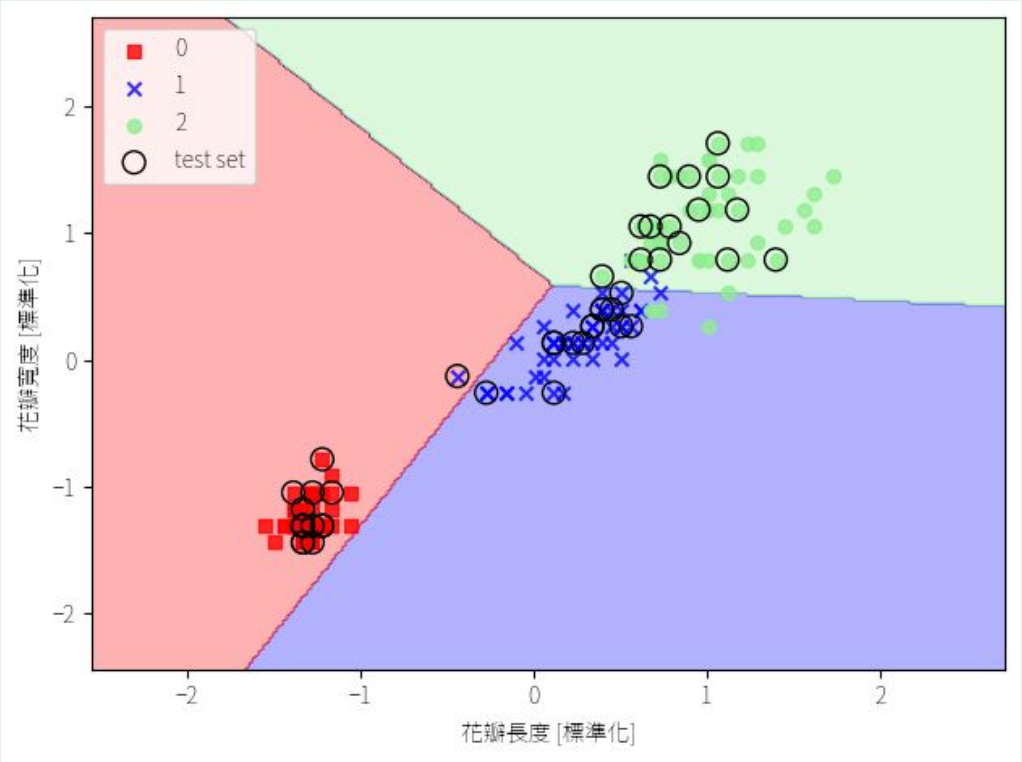
拆分資料集

依照 70% training 與 30% test 比例進行拆分

In [3]:
```
# 從 sklearn.model_selection 匯入 train_test_split 進行資料拆分
# 拆分資料為 7:3
```

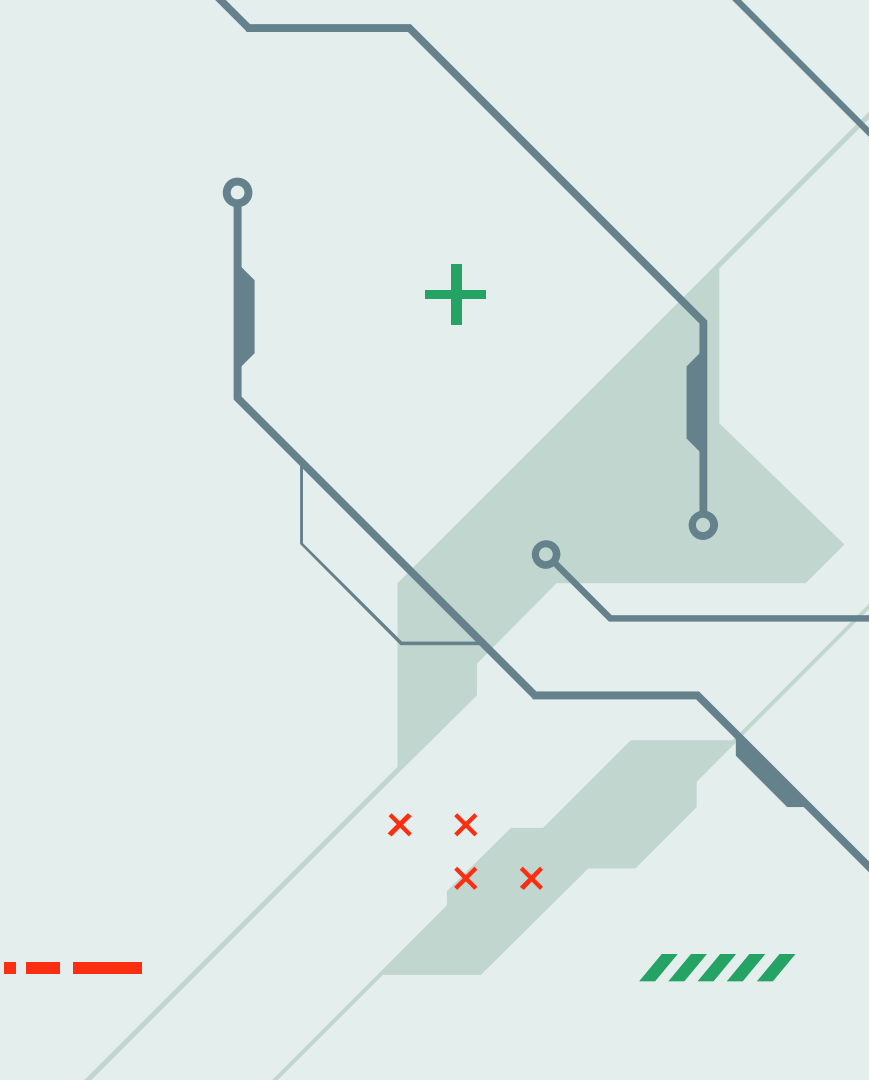
https://colab.research.google.com/github/wenwen357951/ai_course_2023/blob/main/notebooks/02_Scikit_Learn.ipynb

利用 Sklearn 的感知器進行分類

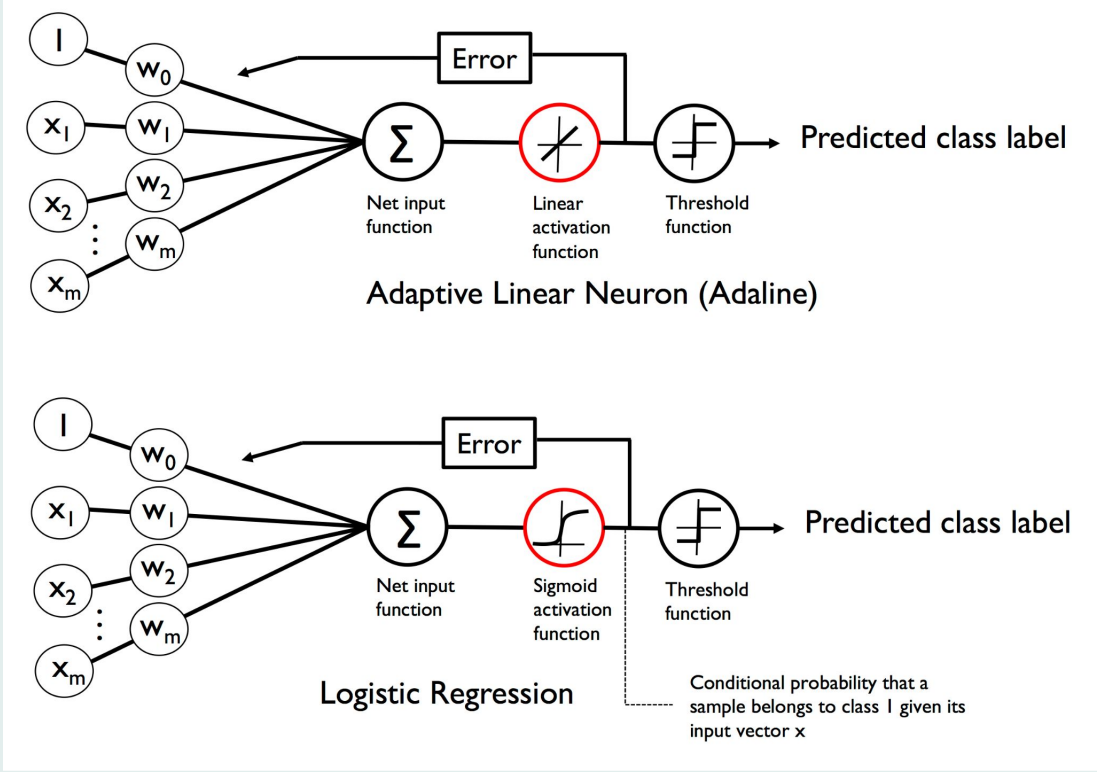


Logistic Regression

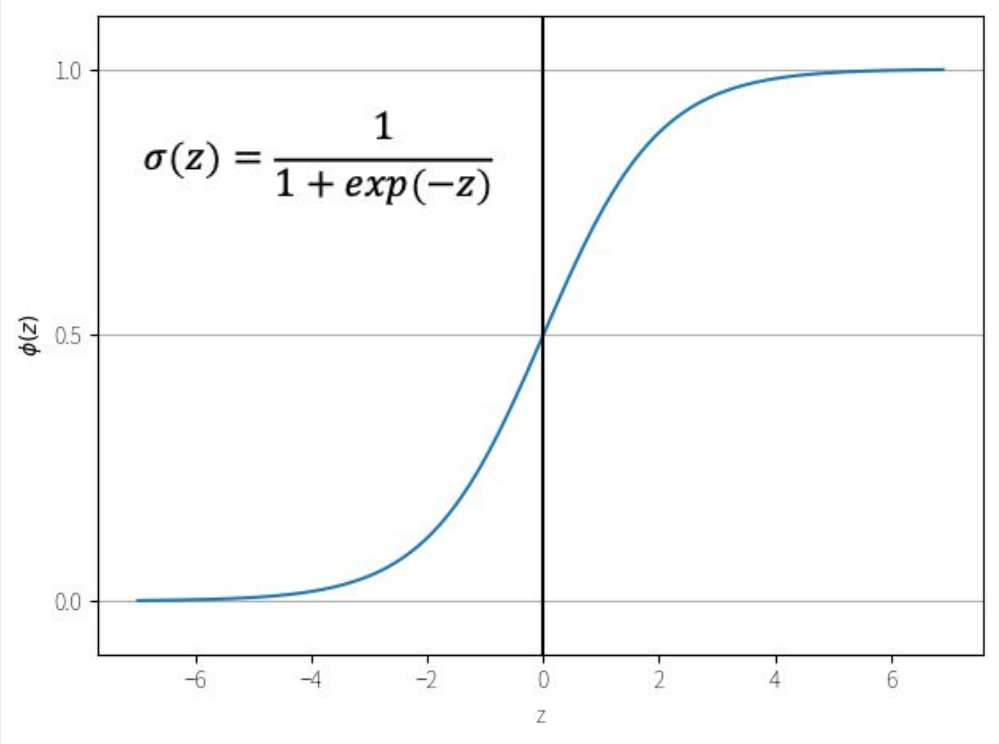
邏輯迴歸



邏輯回歸 v.s. Adaline



Sigmoid 函數

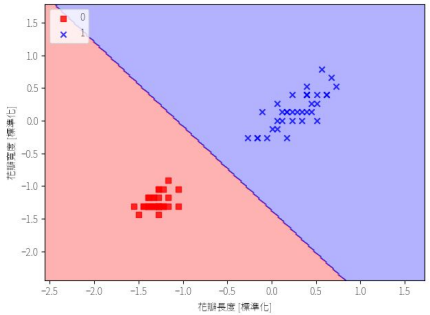


實作邏輯回歸演算法

實作邏輯回歸梯度下降模型

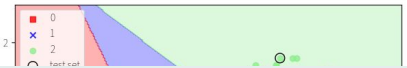
In [15]:
```
# 實作邏輯回歸梯度下降模型
    # 擬合模型
    # 計算網路輸入
    # 活化函數
    # 預測
```

In [16]:
```
# 只取出資料集中的兩個類別
# 使用邏輯回歸模型進行擬合
# 繪製決策邊界
```



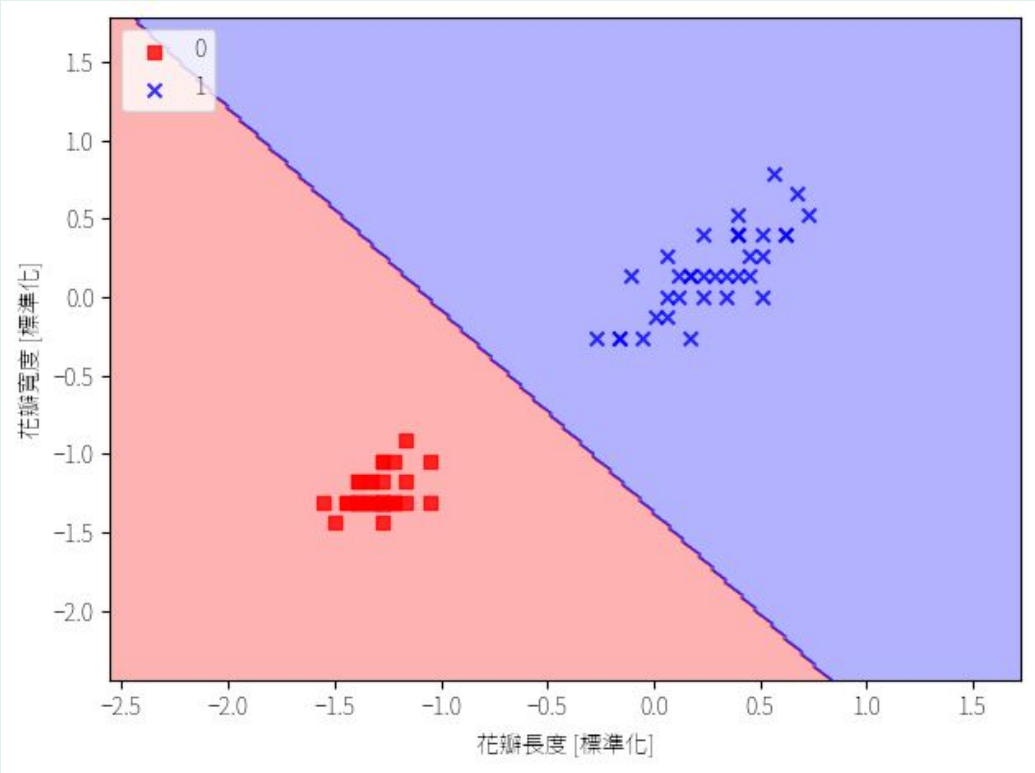
使用 Scikit Learn 訓練邏輯回歸模型

In [17]:
```
# 從 sklearn 匯入邏輯回歸模型
# 建立邏輯回歸模型
# 繪製決策邊界
```

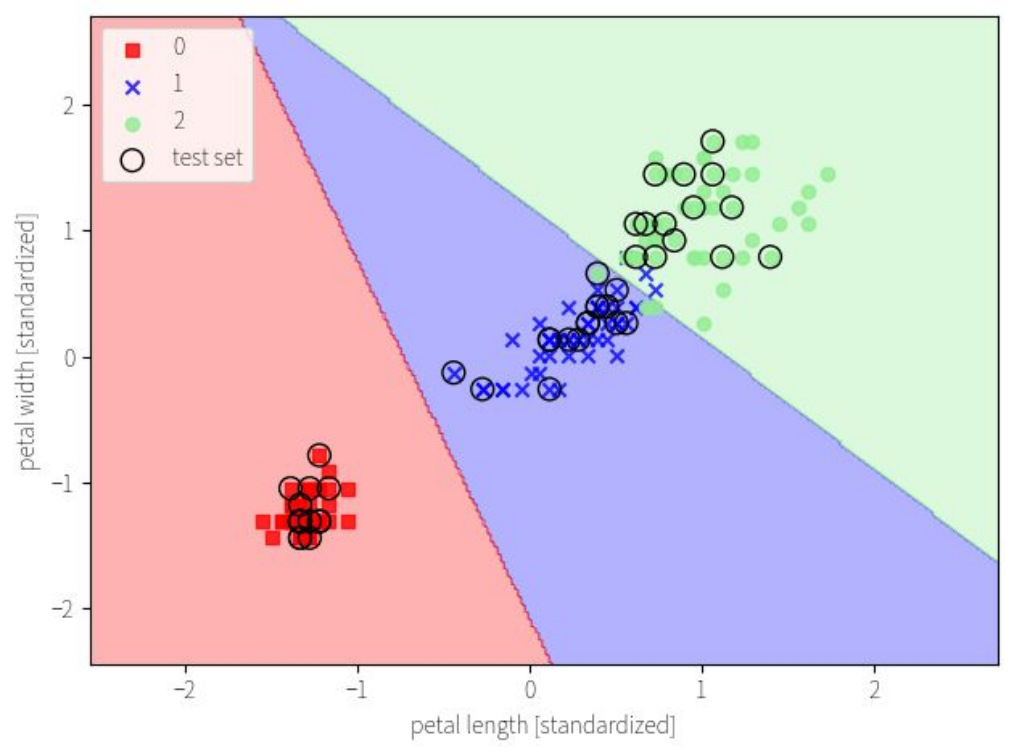


https://colab.research.google.com/github/wenwen357951/ai_course_2023/blob/main/notebooks/02_Scikit_Learn.ipynb

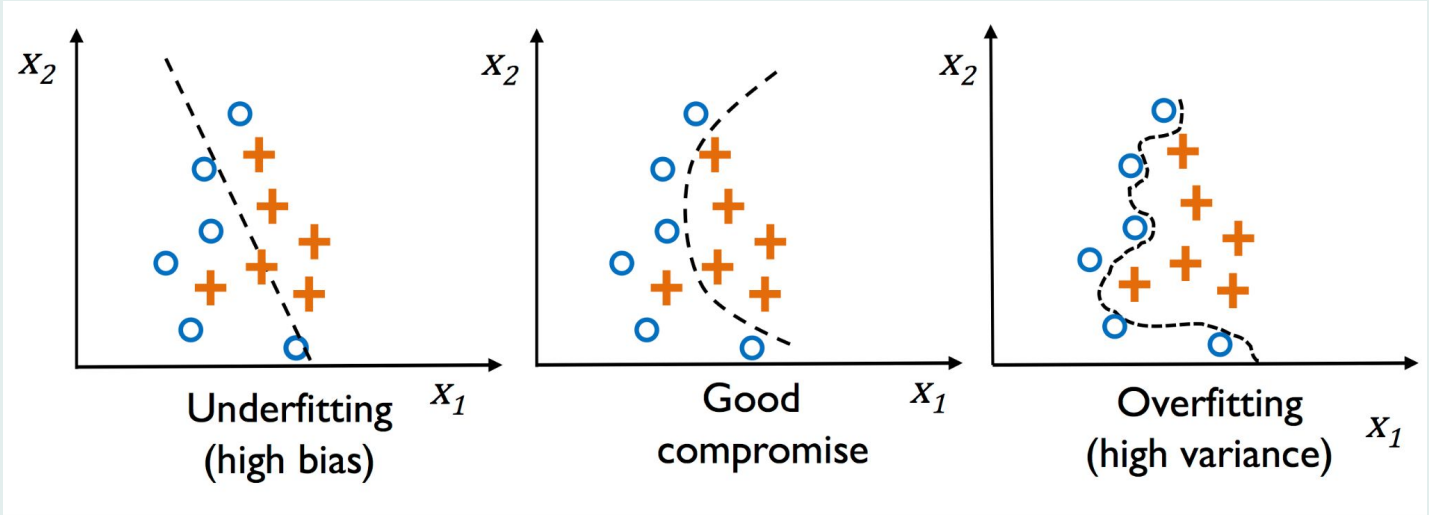
利用邏輯回歸分類


0
1
花瓣寬度 [標準化]
花瓣長度 [標準化]
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

利用 Sklearn 邏輯回歸分類

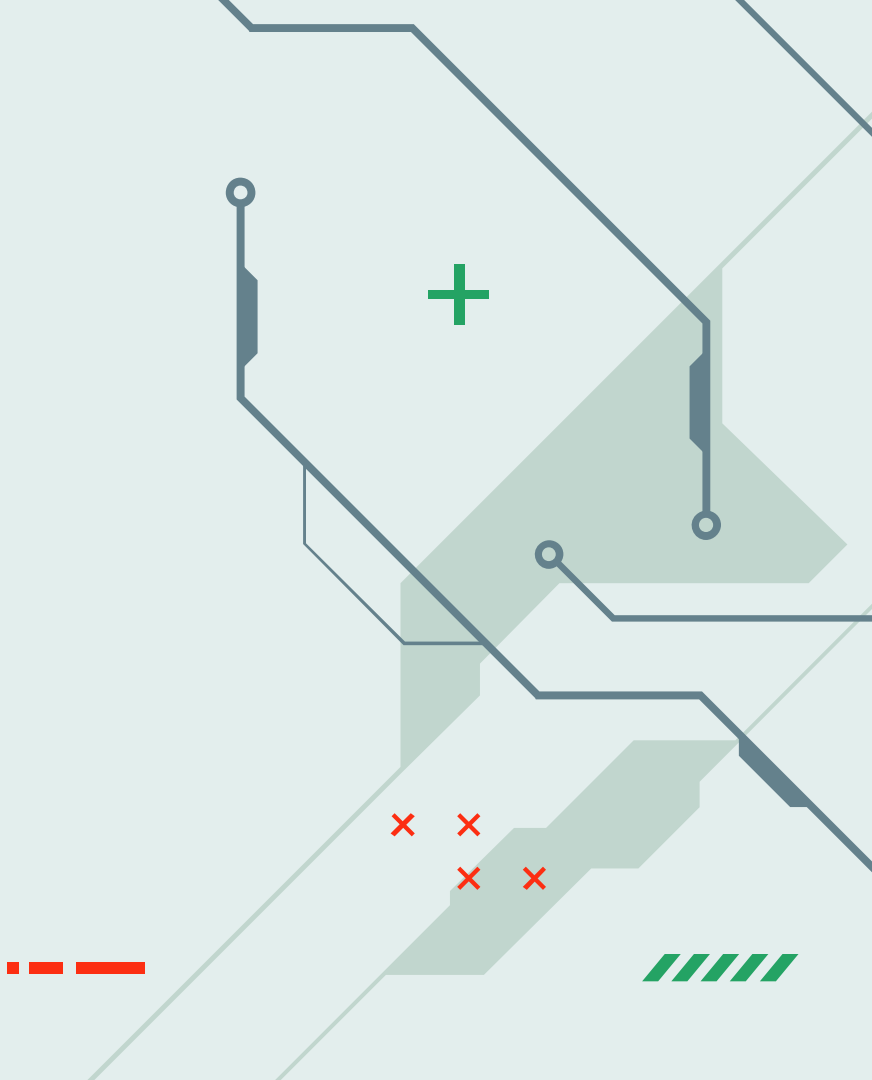


良好的分割線

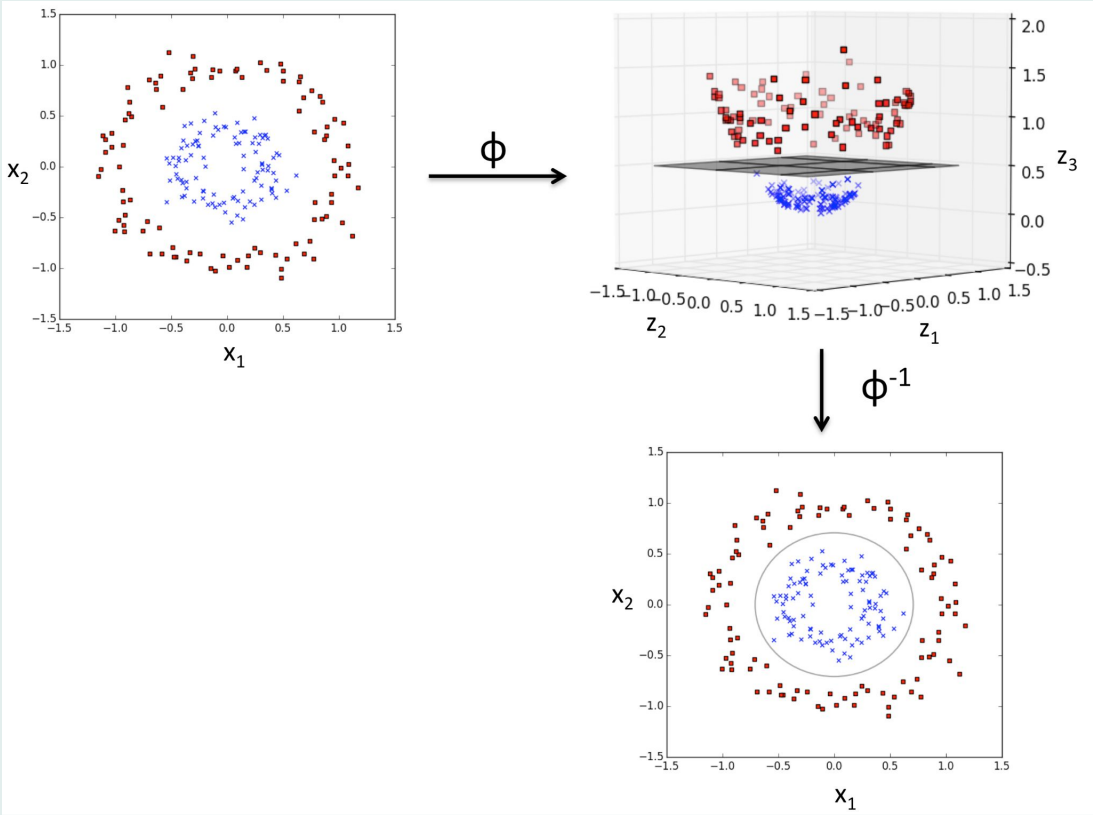

x_2
x_1
Underfitting
(high bias)
x_2
x_1
Good
compromise
x_2
x_1
Overfitting
(high variance)

Support Vector Machines

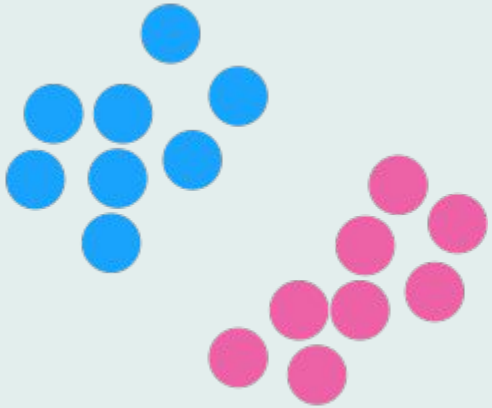
支持向量機



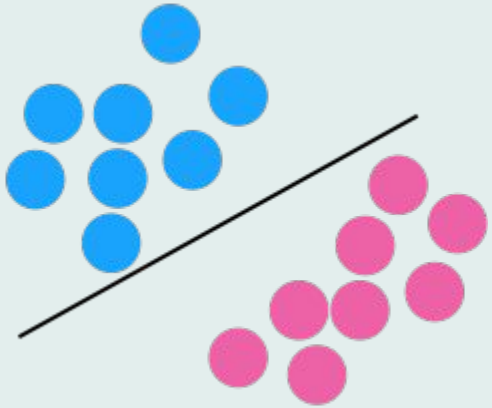
SVM 如何工作的



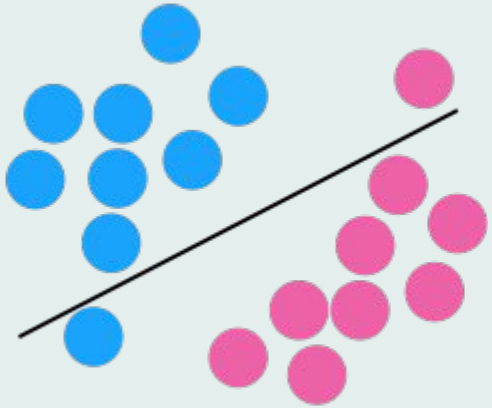
SVM 如何工作的



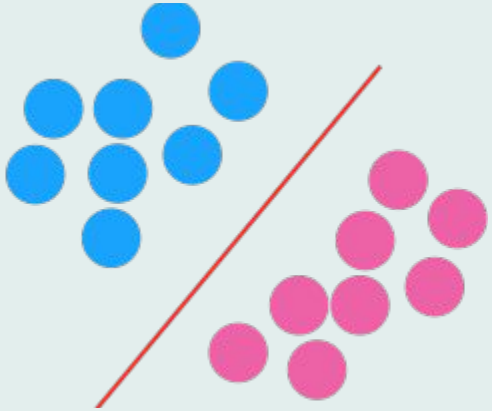
SVM 如何工作的



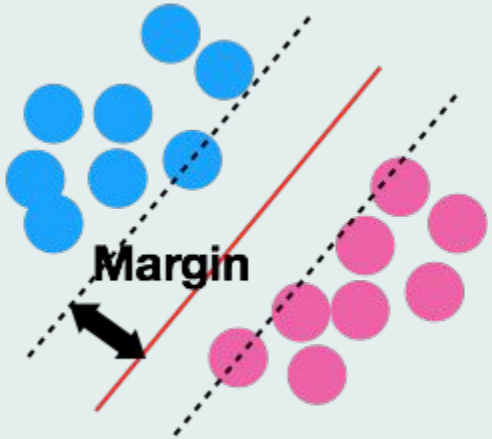
SVM 如何工作的



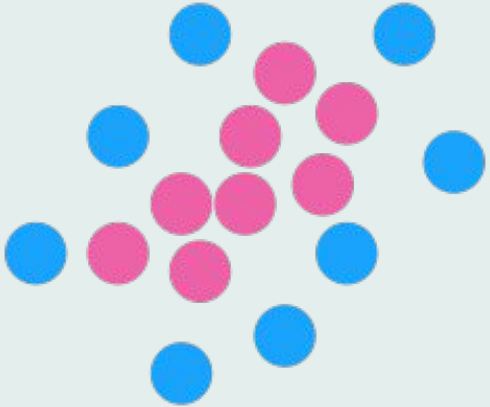
SVM 如何工作的



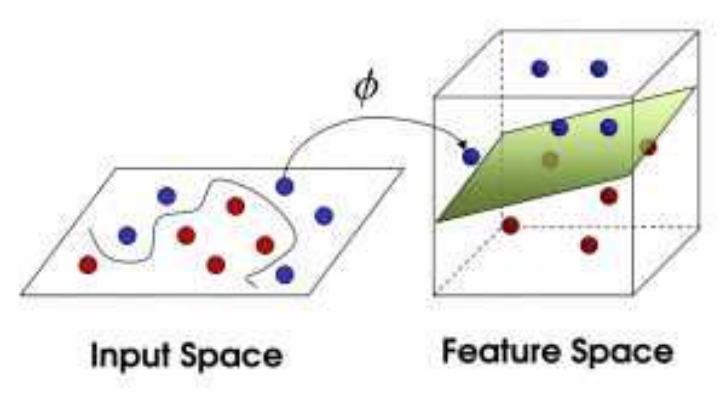
SVM 如何工作的



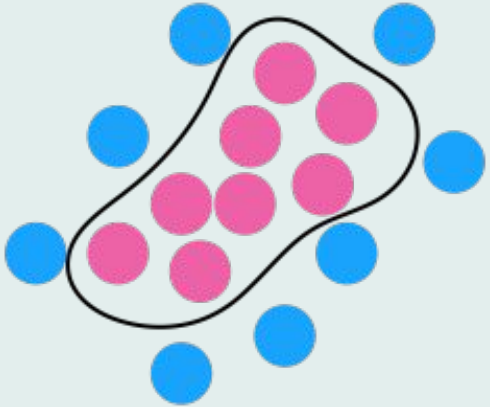
SVM 如何工作的



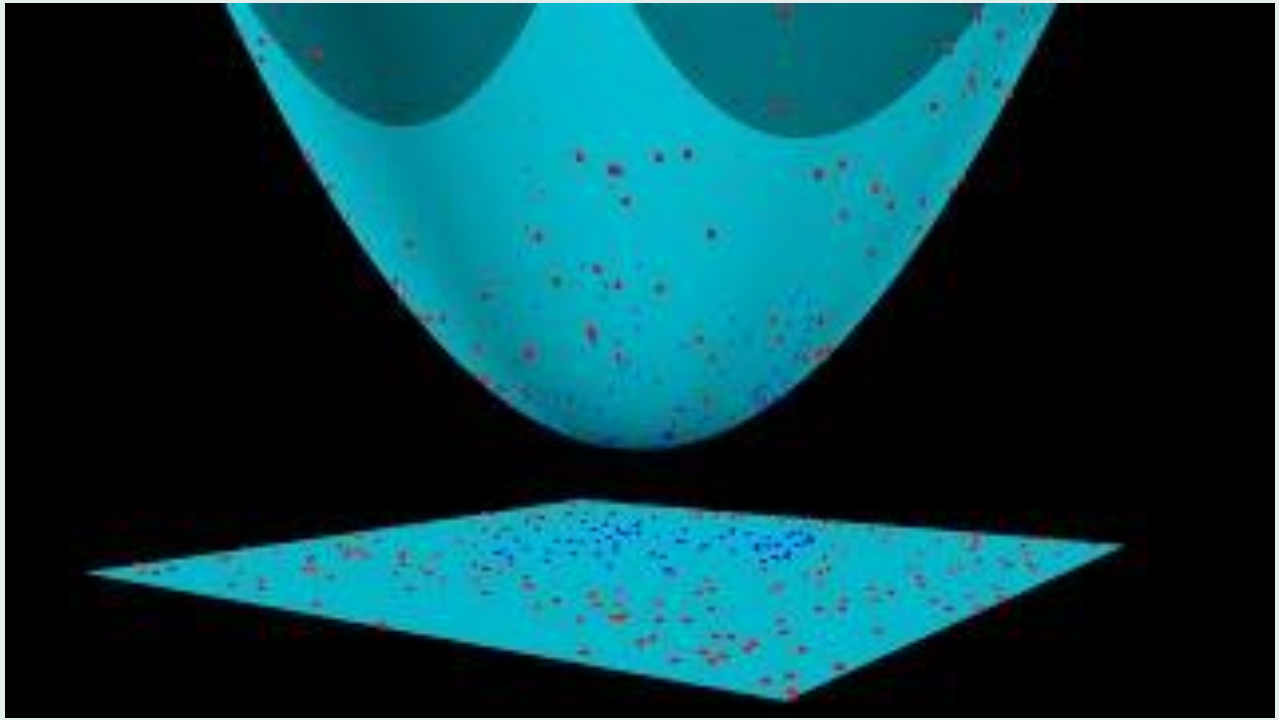
SVM 如何工作的



SVM 如何工作的



SVM 如何工作的

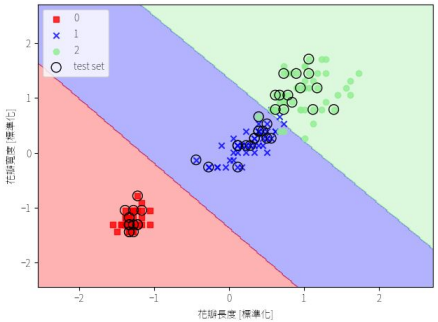


https://www.youtube.com/watch?v=OdlNM96sHio

使用 Sklearn 的 SVM

支持向量機 (Support Vector Machines)

In [22]:



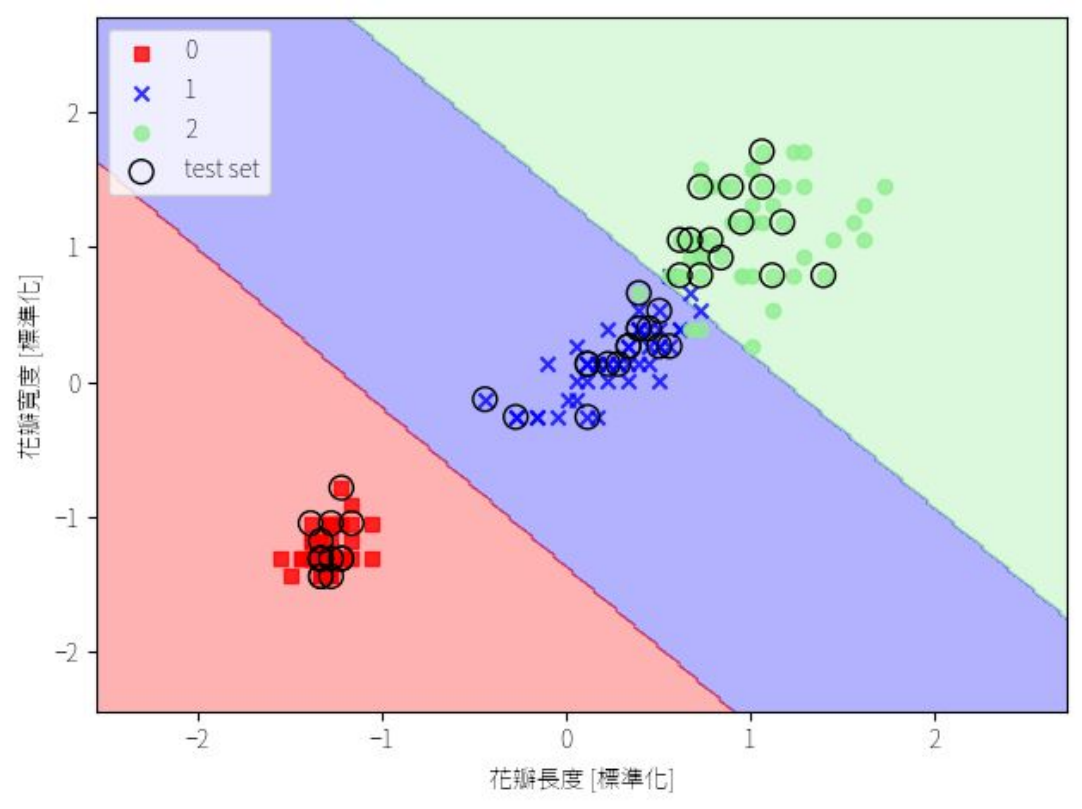
使用 SVM 來解決非線性問題

In [23]:

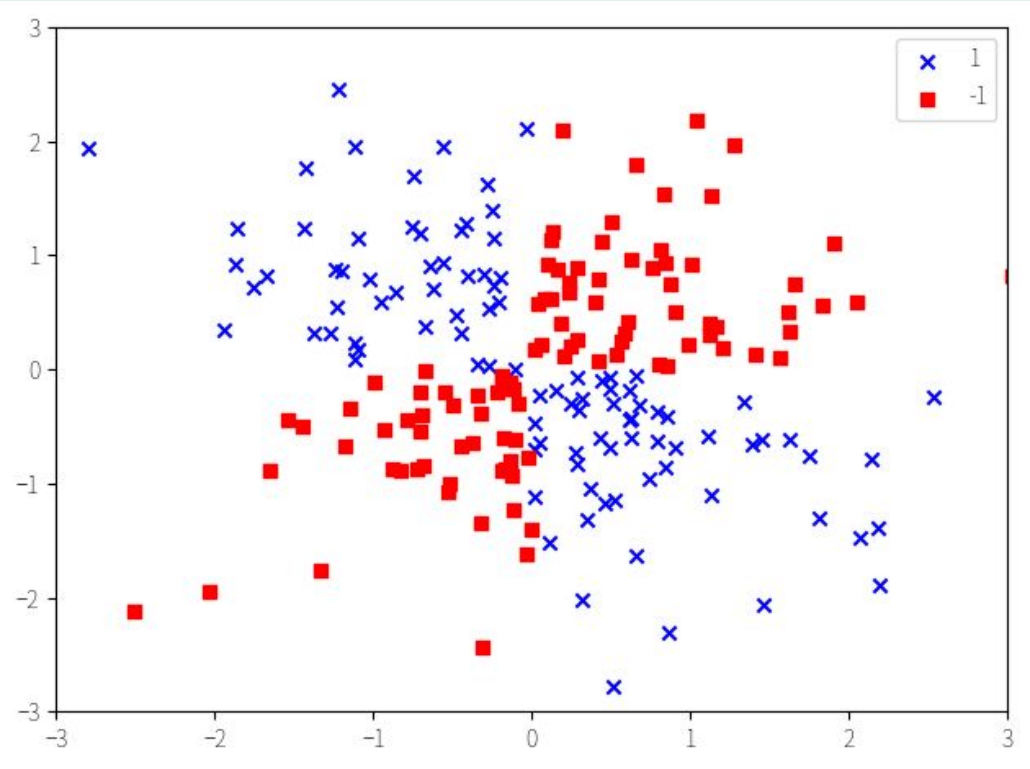
```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(1)
X_xor = np.random.randn(200, 2)
y_xor = np.logical_xor(X_xor[:, 0] > 0,
                       X_xor[:, 1] > 0)
y_xor = np.where(y_xor, 1, -1)

plt.scatter(X_xor[y_xor == 1, 0],
            X_xor[y_xor == 1, 1],
            c='b', marker='x',
            label='1')
plt.scatter(X_xor[y_xor == -1, 0],
            X_xor[y_xor == -1, 1],
            c='r',
            marker='s',
            label='-1')

plt.xlim([-3, 3])
plt.ylim([-3, 3])
```

https://colab.research.google.com/github/wenwen357951/ai_course_2023/blob/main/notebooks/02_Scikit_Learn.ipynb

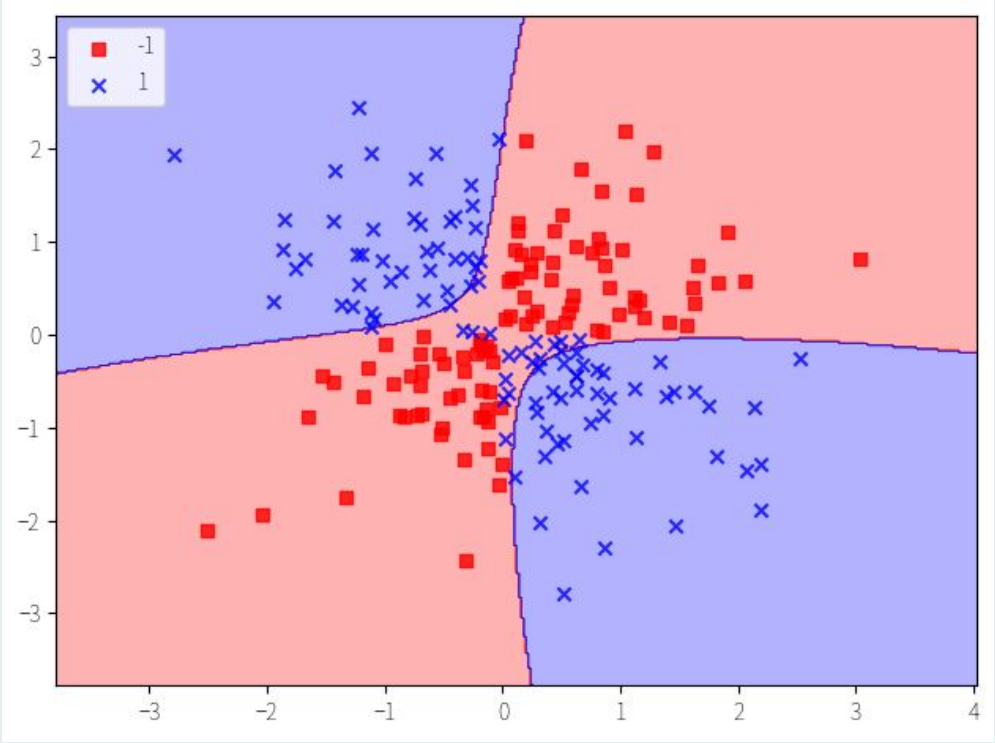
利用 Sklearn 的 SVM



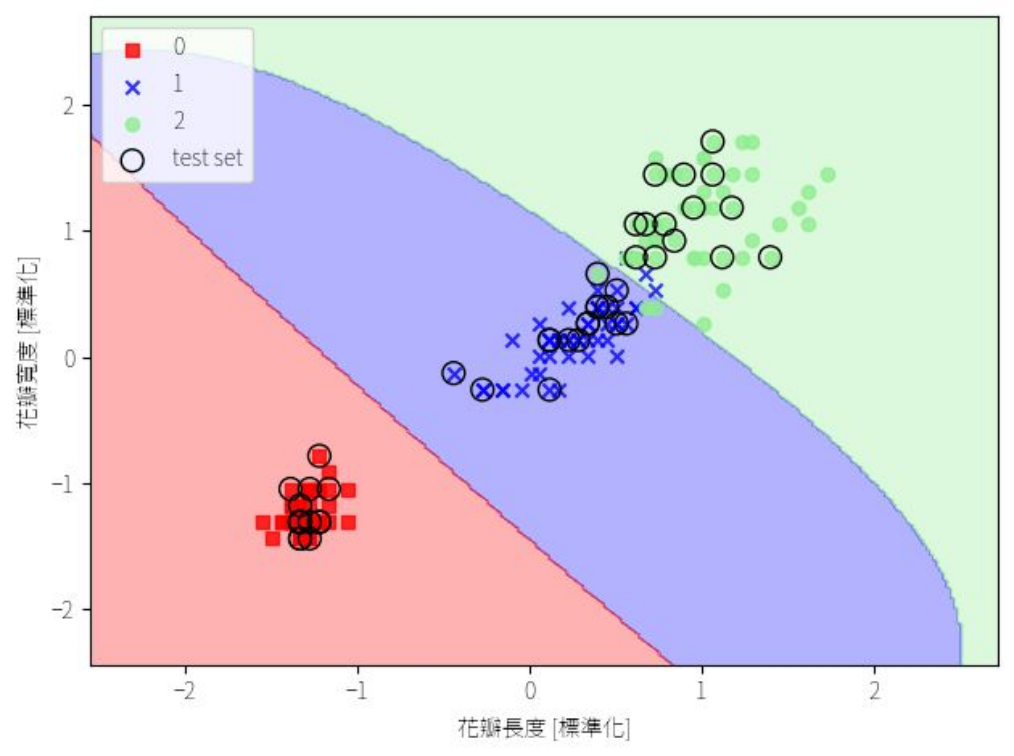
使用 SVM 解決非線性問題



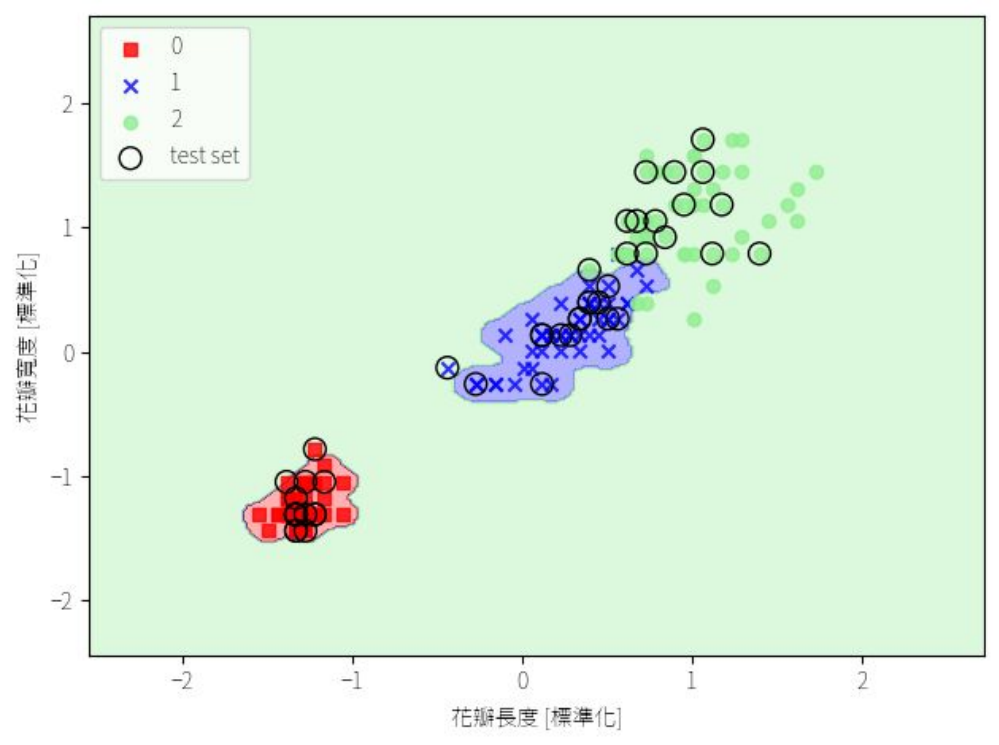
使用 SVM 解決非線性問題



使用 SVM 進行分類



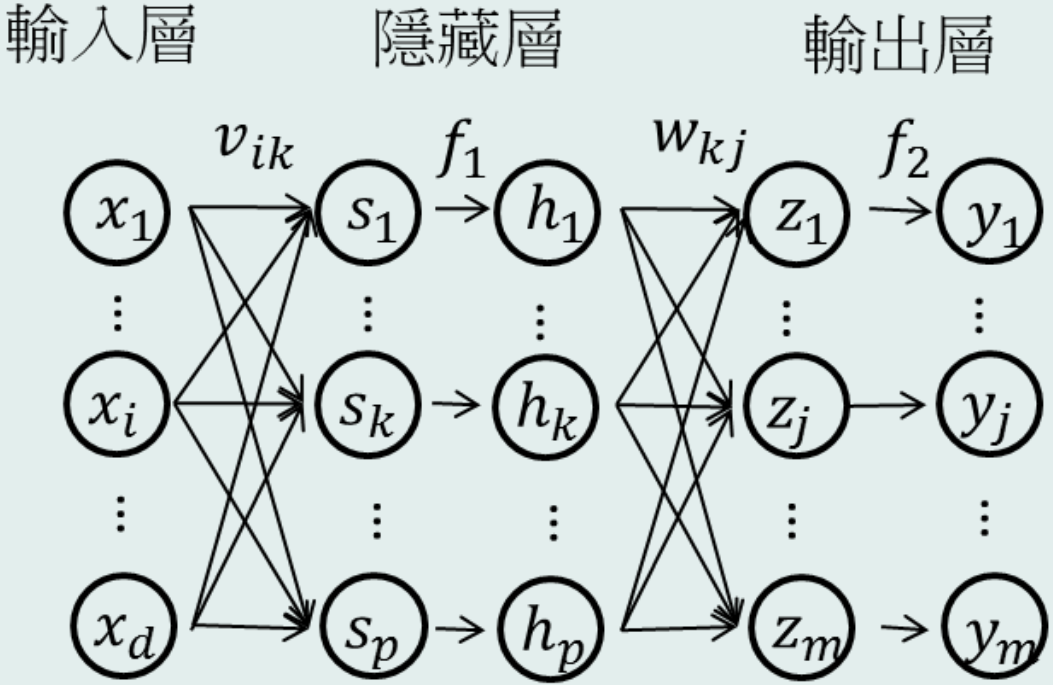
使用 SVM 解決非線性問題



04 Multilayer Perceptron

多層感知器

多層感知器



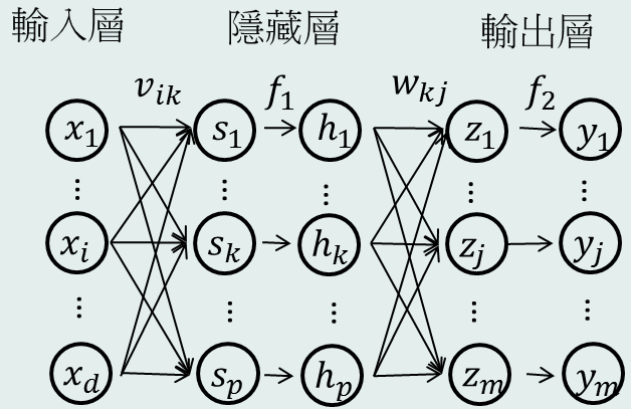
從線性到非線性

$$\mathbf{H} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)},$$
$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}.$$

$$\mathbf{O} = (\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)} = \mathbf{X}\mathbf{W}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} = \mathbf{X}\mathbf{W} + \mathbf{b}.$$

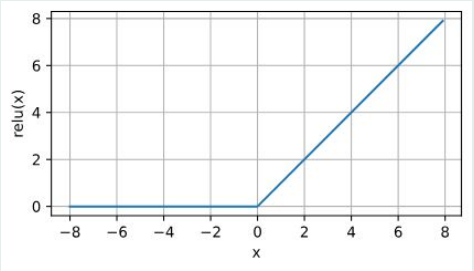
$$\mathbf{H} = \sigma(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}),$$
$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}.$$

H : 隱藏層輸出
W: 權重
b : 偏權值
O : 多層感知器輸出
σ : 活化涵數

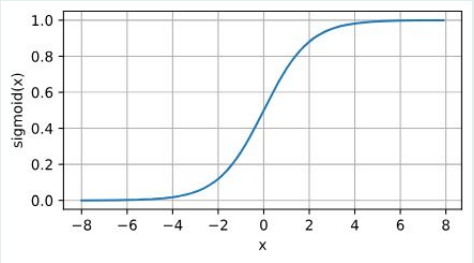


活化函數

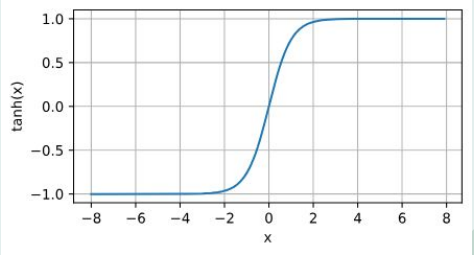
- ReLU 函數(修正線性單元, Rectified Linear Unit)



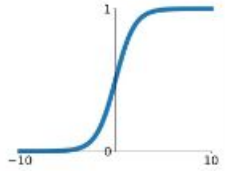
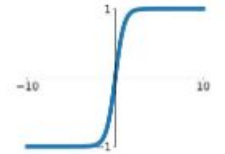
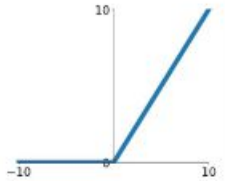
- Sigmoid 函數(乙狀函數, sigmoid function)



- Tanh 函數(雙曲正切函數, hyperbolic tangent function)



活化函數

-	簡單示意圖	梯度消失(飽和)	非0中心輸出	負區域死亡	exp()計算	參數數量2倍
Sigmoid		YES	YES	-	YES	-
tanh		YES	-	-	-	-
ReLU		-	YES	YES	-	-

https://ithelp.ithome.com.tw/articles/10217168